Aplikasi Manajemen *Multi* Proyek menggunakan Metode *Scrum*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Andri Wijaya Kusuma
NIM: 155150200111075

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

APLIKASI MANAJEMEN *MULTI* PROYEK MENGGUNAKAN METODE *SCRUM*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Andri Wijaya Kusuma
NIM: 155150200111075

Skripsi ini telah diuji dan dinyatakan lulus pada
20 September 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Nurudin Santoso, S.T., M.T.	Arief Andy Soebroto, S.T, M.Kom.
NIP: 197409162000121001	NIP: 19720425 199903 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 20 September 2019

Andri Wijaya Kusuma

NIM: 155150200111075

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik serta hidayah-Nya sehingga laporan skripsi dengan judul "Aplikasi Manajemen *Multi* Proyek menggunakan Metode *Scrum*" ini dapat terselesaikan. Penulis menyadari bahwa skripsi ini tidak akan berhasil terselesaikan tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Nurudin Santoso, S.T., M.T. dan Bapak Arief Andy Soebroto, S.T, M.Kom. selaku Pembimbing skripsi yang telah memberi, membimbing dan mengarahkan, sehingga penulis dapat menyelesaikan skripsi ini,
2. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Teknik Informatika,
3. Bapak Subandi dan Ibu Rumini selaku orang tua penulis, dan seluruh keluarga besar yang senantiasa memberikan doa dan semangat demi terselesaikannya skripsi ini,
4. Seluruh dosen Teknik Informatika Universitas Brawijaya yang telah memberikan ilmu yang bermanfaat selama penulis menempuh studi dan selama menyelesaikan skripsi ini,
5. Narasumber narasumber dari beberapa *software house* yang telah membantu memberikan informasi yang berkaitan dengan skripsi yang penulis kerjakan,
6. Seluruh teman teman penulis, khususnya BSC dan teman teman kelas F Informatika serta seluruh pihak yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih terdapat banyak kekurangan, sehingga penulis mengharapkan saran dan kritik yang membangun. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 20 September 2019

Penulis

wijayakusuma1972@gmail.com

ABSTRAK

Andri Wijaya Kusuma, Aplikasi Manajemen *Multi* Proyek menggunakan Metode *Scrum*

Pembimbing: Nurudin Santoso, S.T., M.T. dan Arief Andy Soebroto, S.T, M.Kom.

Dalam suatu software house, manajer proyek memegang peranan penting dalam proses pengelolan dan pembuatan perangkat lunak. Manajer proyek untuk setiap proyeknya diberikan kepada orang yang berbeda. Namun dimungkinkan juga untuk seseorang berperan sebagai manajer proyek pada lebih dari 1 proyek dalam waktu yang bersamaan. Untuk cara pelaporan dari tiap pekerja kepada manajer proyek-pun bervariasi. Ada yang melakukan rapat setiap seminggu sekali, atau pada hari kerja sebelum pulang, atau manajer proyek harus menanyakan sendiri kepada tiap anggotanya. Sehingga manajer proyek tidak dapat memantau perkembangan pengerjaan terkini dari anggota tim setiap waktunya dan juga tidak dapat secara langsung memberikan tugas untuk anggota tim begitu pula anggota tim tidak dapat mengetahui tugas yang harus dikerjakan jika tugas sebelumnya telah selesai dikerjakan. Untuk menangani permasalahan tersebut, penulis mengembangkan aplikasi manajemen multi proyek yang dapat mengelola aktivitas pekerjaan yang dilakukan, dan memantau perkembangan proyek. Aplikasi tersebut menerapkan metode scrum, yang didalamnya terdapat iterasi untuk proses pengerjaan proyek. Dalam pengembangan aplikasi penulis menggunakan waterfall model. Berdasarkan pengujian yang dilakukan, pengujian unit, pengujian integrasi, dan pengujian validasi menghasilkan nilai valid pada kasus uji yang telah dilakukan. Sedangkan pengujian compatibility menunjukkan bahwa aplikasi dapat berjalan baik pada 9 dari 11 versi browser yang diujikan.

Kata kunci: manajemen multi proyek, scrum, aplikasi, software house

ABSTRACT

Andri Wijaya Kusuma, *Application of Multi Project Management Using Scrum Method*

Supervisors: Nurudin Santoso, S.T., M.T. and Arief Andy Soebroto, S.T, M.Kom.

In a software house, project managers take an important role in the process of managing and making software. The project manager for each project is given to different people. But it is also possible for someone to act as a project manager on more than 1 project at the same time. For how to report from each worker to the project manager also varies. There are those who hold meetings once a week, or on weekdays before going home, or the project manager must ask to each member. So that the project manager cannot monitor the progress of the latest workmanship of the team members at all times and also cannot directly assign assignments to the team members as well as team members cannot know the tasks that must be done if the previous task has been completed. To solve the problem, the author developed a multi-project management application that can manage work activities carried out, and monitor the progress of the project. The application applies the Scrum method, in which there is an iteration for the process of working on the project. In developing the application the author uses the waterfall model. Based on the testing performed, unit testing, integration testing, and validation testing produced a valid value in the test case that has been done. While compatibility testing shows that the application can run well on 9 of the 11 browser versions tested.

Keywords: multi project management, scrum, application, software house.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pada era teknologi sekarang, perkembangan teknologi informasi semakin cepat, baik dari hal pengadaan proyek, ataupun biaya yang digunakan untuk proyek tersebut. Pertumbuhan negara Indonesia sebagai negara berkembang ditandai dengan banyaknya proyek berskala besar yang dibangun oleh pemerintah maupun swasta khususnya teknologi. Pada perusahaan yang bergerak dibidang perangkat lunak atau *software house*, memiliki berbagai macam proyek yang mana hasil dari proyek tadi dapat dijadikan sebagai produk dari perusahaan tersebut. Produk dari *software house* dapat berupa perangkat lunak untuk *client*. Menurut Larson (2006:3), Proyek merupakan usaha kompleks, tidak rutin, yang dibatasi oleh waktu, anggaran, sumber daya, dan spesifikasi kinerja yang dirancang untuk memenuhi kebutuhan pelanggan. Suatu proyek terdiri dari beberapa orang anggota yang dipimpin oleh manager proyek.

Manajer Proyek memegang peranan yang penting pada proses pembuatan produk. Dari hasil wawancara yang dilakukan oleh penulis, diperoleh bahwa proses pelaporan progress pekerjaan biasanya dilakukan seminggu sekali, atau pada waktu hari kerja sebelum pulang, atau manajer proyek harus menanyakan sendiri kepada tiap anggotanya. Sehingga manajer proyek tidak dapat memantau perkembangan pengerjaan terkini dari anggota tim setiap waktunya. Selain tidak dapat memantau perkembangan terkini, manajer proyek juga tidak dapat secara langsung memberikan tugas untuk anggota tim begitu pula anggota tim tidak dapat mengetahui tugas yang harus dikerjakan jika tugas sebelumnya telah selesai dikerjakan. Masalah lain yang timbul dari software house adalah adanya *miscommunication* yang terjadi antar pekerja. Untuk penanggung jawab setiap proyek diberikan kepada orang yang berbeda. Namun dalam waktu bersamaan, seorang pekerja dimungkinkan berperan sebagai manajer proyek pada dua atau lebih pekerjaan.

Seiring berkembangnya teknologi informasi, semakin ketat pula kompetisi dalam penyelenggaraan proyek pada *software house* untuk memenuhi kebutuhan *client*, sehingga dibutuhkan cara pengelolaan, metode dan teknik yang paling baik. Hal tersebut diharapkan supaya pengunaan sumber daya yang dimiliki benar-benar efektif dan efisien sehingga dibutuhkan suatu manajemen proyek yang berguna dalam mengatur keseluruhan proyek perangkat lunak yang dikerjakan. Manajemen Proyek berisi proses perencanaan, mengarahkan dan menggunakan sumber daya perusahaan untuk mencapai tujuan dalam waktu yang telah ditentukan. Manajemen proyek menempatkan anggotanya pada tugas tertentu. Sedangkan perangkat lunak adalah sebuah perintah program dalam sebuah komputer, yang apabila dieksekusi oleh usernya akan memberikan fungsi dan unjuk kerja seperti yang diharapkan oleh usernya (Roger S. Pressman, 2002).

Pesatnya teknologi informasi, menuntut manusia dalam menggunakan komputer sebagai perangkat untuk proses perhitungan dan manajemen proyek.

Perkembangan perangkat lunak juga mendorong perubahan manajemen proyek dari tradisional menjadi sebuah perangkat lunak manajemen proyek. Dalam satu waktu, suatu perusahaan dapat menjalankan lebih dari satu proyek yang dikerjakan oleh tiap tiap tim. Perangkat lunak manajemen proyek ini membantu manajer proyek dalam perencanaan, pengorganisasian, dan manajemen sumber daya dalam proses penyelesaian suatu proyek sehingga dapat tercapainya tujuan proyek. Untuk mengatasi masalah yang kompleks dari suatu proyek, maka diperlukan metode dalam penyelesaiannya. Contoh metode yang mana dapat diterapkan pada manajemen proyek adalah metode scrum.

Scrum adalah proses pengembangan perangkat lunak *Agile* yang berfokus pada praktik manajemen proyek. Scrum merupakan kerangka kerja yang ringan dan memuat langkah-langkah untuk mengelola dan mengontrol proses pengembangan suatu produk. Metode Scrum mulai popular dalam beberapa tahun terakhir dan terbukti sangat berguna walaupun bukan metode yang selalu digunakan (Schwaber. K, 2002). Akhir-akhir ini Scrum telah mendapatkan popularitas besar dan diadopsi oleh perusahaan besar seperti Yahoo (G. Cloke, 2007), Microsoft, Intel, dan Nokia.

1.2 Rumusan Masalah

1. Bagaimanakah hasil dari analisis dan spesifikasi kebutuhan Aplikasi Manajemen *Multi* Proyek?
2. Bagaimanakah hasil perancangan Aplikasi Manajemen *Multi* Proyek yang sesuai dengan hasil analisis kebutuhan?
3. Bagaimanakah hasil implementasi metode *Scrum* terhadap Aplikasi Manajemen *Multi* Proyek yang sesuai dengan rancangan sistem tersebut?
4. Bagaimanakah hasil pengujian sistem perangkat lunak untuk Aplikasi Manajemen *Multi* Proyek tersebut?

1.3 Tujuan

Berikut merupakan tujuan utama dari penelitian yang dilakukan :

1. Menganalisis dan menyusun kebutuhan Aplikasi Manajemen *Multi* Proyek.
2. Merancang sistem sesuai dengan hasil dari analisis kebutuhan.
3. Mengimplementasikan metode *Scrum* pada Aplikasi Manajemen *Multi* Proyek yang sesuai dengan hasil perancangan sistem tersebut.
4. Melakukan pengujian terhadap Aplikasi Manajemen *Multi* Proyek yang dibangun.

1.4 Manfaat

Manfaat dari hasil penelitian dapat dijabarkan sebagai dampak positif dari penelitian

1. Manager proyek dapat memberikan tugas kepada pekerja secara *online*.
2. Manager proyek dapat memantau perkembangan proyek proyek yang dikerjakan.

3. Pekerja dapat melaporkan perkembangan pekerjaan yang sedang dikerjakan secara langsung.
4. Mengatasi masalah kemungkinan terjadinya *miscommunication*.

1.5 Batasan Masalah

Berikut beberapa batasan masalah mengenai penelitian ini sehingga mencegah terjadinya penyimpangan dari tujuan yang sidah direncanakan :

1. Aplikasi ini berfokus pada manajemen *multi* proyek pada pengembangan perangkat lunak.
2. Aplikasi ini akan dibangun menggunakan *platform* web.
3. Aplikasi dikembangkan menggunakan bahasa PHP, HTML5, CSS, dan Javascript.
4. Aplikasi dibangun menggunakan *framework* Laravel.
5. Aplikasi meggunakan *database* MySQL.
6. Aplikasi tidak membutuhkan koneksi internet, namun dapat juga membutuhkan koneksi internet jika Aplikasi dionlinekan.

1.6 Sistematika Pembahasan

Sistematika pembahasan yang digunakan dalam penulisan penelitian Aplikasi Manajemen *Multi* Proyek Menggunakan Metode *Scrum* adalah:

BAB I PENDAHULUAN

Bab ini memuat Latar Belakang dari Masalah, Rumusan Masalah, Tujuan dan Manfaat Penelitian, serta Sistematika Pembahasan dari skripsi berjudul Aplikasi Manajemen *Multi* Proyek Menggunakan Metode Scrum.

BAB II TINJAUAN PUSTAKA

Bab ini membahas tentang metode metode dan teori yang diperlukan sebagai dasar penelitian dan pengembangan Aplikasi Manajemen *Multi* Proyek Menggunakan Metode *Scrum*.

BAB III METODOLOGI PENELITIAN

Bab berikut membahas metode penelitian yang menjelaskan metode dan langkah proses yang digunakan dalam Perancangan Aplikasi Manajemen *Multi* Proyek Menggunakan Metode *Scrum*.

BAB IV ANALISIS KEBUTUHAN

Bab ini membahas analisis kebutuhan yang dibutuhkan pada Aplikasi dan proses perancangan solusi dalam Perancangan Aplikasi Manajemen *Multi* Proyek Menggunakan Metode *Scrum*.

BAB V PERANCANGAN DAN IMPLEMENTASI

Bab berikut membahas perihal perancangan perangkat lunak serta implementasi dari hasil perancangan yang akan dilakukan berdasarkan analisis kebutuhan dari Aplikasi Manajemen *Multi* Proyek Menggunakan Metode *Scrum*.

BAB VI PENGUJIAN

Bab ini membahas teknik pengujian yang dilakukan dan analisis hasil pegujian terhadap sistem Aplikasi Manajemen *Multi* Proyek Menggunakan Metode *Scrum*.

BAB VII KESIMPULAN DAN SARAN

Bab terakhir berikut membahas mengenai kesimpulan yang didapatkan dari hasil perancangan Aplikasi Manajemen *Multi* Proyek Menggunakan Metode *Scrum* beserta saran untuk pengembangan yang lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab berikut membahas kajian pustaka dan sekumpulan dasar teori atau pegetahuan yang digunakan penulis dalam rangka menyusun penelitian ini.

2.1 Kajian Pustaka

Kajian pustaka dilakukan guna sebagai referensi dalam penyusunan penelitian ini. Referensi yang digunakan adalah dari beberapa penelitian penelitian sebelumnya yang terkait tentang scrum dan manajemen muti proyek. Dari penelitian yang ditemukan, ada tiga penelitian yang akan digunakan sebagai referensi pada penelitian kali ini.

Penelitian pertama yang dilakukan oleh Marchenko yang berjudul "*Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges*" (Marchenko, 2008) diperoleh bahwa saat itu, departemen kasus yang merupakan bagian dari tanggung jawab Nokia pada *speech recognition*, *speech synthesis* dan area yang terkait, memiliki kesan positif dari proses *Scrum*. Masalah prioritas terhadap kebutuhan dianggap menjadi salah satu elemen kesuksesan dari Scrum. Terdapat sepuluh tantangan khusus yamg diidentifikasi selama penelitian. Selanjutnya, aplikasi kasus Scrum dalam penelitian meningkatkan pemahaman tentang hambatan dalam situasi praktis yang berbeda. Dalam penelitian tersebut terdapat beberapa keterbatasan dalam penelitian, seperti periode waktu yang relatif singkat. Jika tujuannya adalah untuk berhasil dalam lingkungan pada penelitian, maka hal terbaik yang harus dilakukan adalah membuat persiapan terlebih dahulu.

Menurut romano didalam penelitiannya yang berjudul "*Project management using the Scrum agile method:A case study within a small enterprise*" (Romano, 2015), yang mana terdapat perbandingan sebelum dan sesudah penyebaran metode Scrum. Dalam semua kasus ada perubahan positif, bahkan partisipasi tim dalam item dinilai memuaskan. Hasil juga menunjukkan peningkatan kualitas dan waktu yang signifikan. Pada penelitian yang berjudul "*SCRUM Model for Agile Methodology*" menjelaskan tentang scrum yang memiliki siklus yang telah terbukti dapat mangakomodasi perubahan dalam tiap iterasi (Srivastava, et al., 2017).

Untuk penelitian ini akan berfokus pada pengembangan Aplikasi Manajemen Multi Proyek menggunakan Scrum, yang meliputi penyusunan backlog, membuat sprint, hingga melakukan sprint review, dll. Berikut akan disajikan table daftar kajian pustaka antara metode metode yang dapat digunakan dalam melakukan manajemen proyek. Selain Scrum terdapat beberapa metode dalam manajemen proyek, berikut beberapa penelitian tentang metode metode yang dapat digunakan untuk melakukan manajemen proyek :

Tabel 2.1 Daftar Kajian Pustaka

No	Judul	Objek	Metode	Hasil
		Input	Proses	
1.	Scrum Agile Project Management Methodology Application for Workflow Management: A Case Study (Carneiro1 & Silva1 & Alencar1, 2018)	Untuk mendapatkan data guna penelitian berikut, dilakukan dengan mengirimkan kuesioner untuk manajer PT daerah, analisis dokumen yang diproduksi oleh sektor objek studi kasus ini, dan pengamatan.	Metode Scrum	Dalam konteks ini, dimungkinkan untuk memverifikasi adaptasi konsep Scrum untuk mengakui penggunaannya dalam struktur organisasi fungsional, terutama yang harus diperhatikan tuntutan prioritas yang timbul dalam organisasi setiap hari. Dapat dilihat juga Scrum adalah alternatif menarik untuk aplikasi yang berbeda konteks, mengingat tiga pilar fundamentalnya adalah transparansi, inspeksi dan adaptasi.
2.	Using Scrum for Software Engineering Class Projects (Ramrao, 2012)	Pelatihan dan pembelajaran pada siswa tentang Scrum	Terdapat kelas teori sebagai pembelajaran bagi siswa. Dalam pelatihan diberikan tugas proyek untuk kelompok dengan masalah yang	Sekitar 85% dari total 29 siswa merasakan ada peningkatan dalam cara pengelolaan proyek menggunakan Scrum.

Tabel 2.1 Daftar Kajian Pustaka

No	Judul	Objek	Metode	Hasil
		Input	Proses	
			berbeda tiap kelompok.	
3.	How Does Kanban Impact Communication and Collaboration in Software Engineering Teams? (Oza & Fagerholm & Münch, 2013)	Menganalisa dampak dari proses kolaborasi tim dan komunikasi secara luas. Studi kasus pada penelitian ini adalah mengembangkan produk perangkat lunak pembayaran mobile dalam enam iterasi lebih dari tujuh minggu.	Menggunakan proses pengembangan Kanban. Data dikumpulkan dari kuesioner dan, diulangi pada akhir setiap iterasi	Hasil menunjukkan bahwa Kanban memiliki efek positif pada awalnya tim bekerja bersama untuk mengidentifikasi dan mengoordinasikan pekerjaan. Ketika anggota tim telah membentuk hubungan baik di antara mereka, pentingnya untuk memfasilitasi kolaborasi tim tidak dapat ditampilkan. Selain itu, Kanban membantu tim dalam mengidentifikasi tugas-tugas yang hilang untuk menjaga laju perkembangan seluruh tim.
4.	Using Trello to Support Agile and Lean Learning with Scrum and Kanban in Teacher	Berfokus pada penerapan layanan untuk guru. Penerapan metode ini untuk	Para peserta bekerja melalui proses Scrum selama periode dua jam, dibagi menjadi tiga sprint, dan tiap	Trello tidak mendukung sepenuhnya kolaboratif dalam proses pengeditan. Jika murid terbiasa

Tabel 2.1 Daftar Kajian Pustaka

No	Judul	Objek	Metode	Hasil
		Input	Proses	
	Professional Development (Parsons & Thorn & Inkila & MacCallum, 2018)	pendidikan di bidang pengetahuan di luar pengembangan perangkat lunak	sprint diikuti oleh pertemuan. Satu siswa dari setiap tim diundang ke dewan Trello sebagai dan dapat mengalami peran Scrum Master, yang memimpin Scrum. Papan Kanban memvisualisasikan alur kerja dengan memiliki beberapa kolom bernama, misalnya, 'Backlog', 'Ready', 'In Process' dan 'Done'. Papan Kanban memiliki batas dalam pekerjaan, yang berarti ada batas jumlah kartu yang dapat diproses pada satu waktu.	bekerja sepenuhnya dengan alat kolaboratif ini, dapat menyebabkan kesalahpahaman, sangat jelas komunikasi tentang peran dan tanggung jawab diperlukan. Dengan menggunakan Kanban, siswa lebih mampu secara efisien gunakan waktu mereka daripada menghabiskan usaha pada tugas yang konstan.

2.2 Manajemen Proyek

Manajemen adalah suatu ilmu pengetahuan tentang seni memimpin organisasi yang terdiri atas kegiatan perencanaan, pengorganisasian, pelaksanaan, dan pengendalian terhadap sumber-sumber daya terbatas dalam usaha mencapai tujuan dan sasaran yang efektif dan efisien (Husen, 2009). Sedangkan Proyek adalah gabungan dari sumber daya seperti manusia material, peralatan, dan modal atau biaya yang dihimpun dalam suatu wadah organisasi sementara untuk mencapai sasaran dan tujuan (Husen, 2009).

Berdasarkan dua pengertian diatas, dapat ditarik kesimpulan mengenai pengertian manajemen proyek, yaitu sebagai penerapan dari ilmu pengetahuan,

keahlian dan keterampilan, cara teknis yang terbaik dan dengan sumber daya yang terbatas, untuk mencapai sasaran dan tujuan yang telah ditentukan agar mendapatkan hasil yang optimal dalam hal kinerja biaya, mutu dan waktu serta keselamatan kerja (Husen, 2009). Sebagai suatu proses, manajemen dapat bermakna bahwa proyek dilasanakan berdasarkan langkah-langkah yang sesuai dengan tujuan, target, dan keterbatasan yang ada.

Peran dari manajemen proyek sangatlah penting dalam mendukung kegiatan pengembangan perusahaan. Dalam skala yang lebih luas pastinya manajemen multi proyek sangat diperlukan guna hasil yang diperoleh dari tiap-tiap proyek mulai dari perintisan proyek hingga tahap akhir dapat berjalan dengan lancar dan sesuai target menurut batas waktu pengerjaan serta hasilnya sesuai dengan tujuan awal yang ingin dicapai.

2.3 Scrum

2.3.1 Definisi

Scrum adalah suatu *framework* dimana penggunanya akan dapat mengatasi masalah kompleks adaptif, dan pada waktu bersamaan mereka juga menghantarkan produk dengan nilai tinggi secara produktif dan kreatif (Ken & Jeff, 2017). Scrum merupakan kerangka kerja proses yang sudah digunakan sejak awal tahun 1990-an untuk mengatur proses pengembangan produk yang kompleks. Scrum mengekspos ketidak-efektifan dari manajemen produk dan teknik kerja, sehingga pengguna dapat secara terus-menerus meningkatkan kinerja produk, tim, dan lingkungan kerja (Ken & Jeff, 2017).

2.3.2 Tim Scrum

Menurut (Ken & Jeff, 2017), Tim Scrum meliputi Product Owner, Development Team dan Scrum Master. Tim Scrum memiliki kecakapan yang diperlukan dalam menyelesaikan pekerjaan mereka tanpa bergantung pada orang lain. Tim Scrum dirancang sedemikian rupa supaya mengoptimalkan fleksibilitas, kreativitas dan produktivitas. Tim Scrum menghantarkan produk secara iteratif dan inkremental untuk memaksimalkan peluang guna mendapatkan *feedback*. Berikut penjelasan dari Tim Scrum (Ken & Jeff, 2017) :

a. Product Owner, merupakan bagian yang memegang tanggung jawab dalam memaksimalkan nilai bisnis dari produk yang dihasilkan oleh Development Team. Cara melakukannya sangat bervariasi antar organisasi, Scrum Team dan individu.
b. Development Team berisikan beberapa para ahli profesi yang bekerja guna menghantarkan Increment “Selesai” yang berpotensi untuk dirilis di setiap akhir Sprint. Increment “Selesai” wajib tersedia pada saat Sprint Review. Development Team dibentuk dan diberikan wewenang oleh organisasi untuk menyusun dan mengelola pekerjaan mereka sendiri.
c. Scrum Master bertanggung jawab dalam mengenalkan penggunaan Scrum dan membantu orang-orang di luar Tim Scrum supaya dapat memahami

interaksi yang menguntungkan dan tidak bermanfaat guna memaksimalkan nilai bisnis yang dihasilkan.

2.3.3 Scrum Events

Acara yang harus ada pada Scrum berguna untuk menciptakan sifat rutin dan mengurangi pertemuan acara lain yang bukan bagian *Scrum* (Ken & Jeff, 2017). Terdapat batas waktu maksimal untuk tiap acara pda *scrum*. Menurut *Scrum* (Ken & Jeff, 2017), terdapat beberapa kegiatan dalam acara *Scrum*, seperti :

a. Sprint, merupakan acara dengan durasi kurang dari satu bulan dan didalamnya terjadi proses pembuatan Increment yang "Selesai", dapat dipakai serta memiliki potensi untuk dirilis. Durasi pada sprint cenderung konsisten sepanjang siklus pengembangan sistem. Untuk sprint selanjutnya akan dijalankan ketika sprint sebelumnya telah selesai.
b. Sprint Planning, berisi pekerjaan yang akan dikerjakan. Perencanaan ini dilakukan secara kolaboratif oleh seluruh anggota tim Scrum. Batas waktu pada sprint planning hanya 8 jam untuk Sprint yang berdurasi satu bulan.
c. Daily Scrum merupakan kegiatan pada Development Team dengan batas waktu selama 15 menit. Kegiatan ini berjalan tiap hari selama Sprint berjalan. Development Team akan merencanakan program kerja untuk 24 jam mendatang. Daily Scrum terjadi pada waktu dan lokasi yang sama tiap harinya guna mengurangi tingkat kompleksitas. Development Team menggunakan Daily Scrum untuk menginspeksi perkembangan pekerjaan.

2.3.4 Scrum Artifacts

Artefak Scrum merepresentasikan suatu pekerjaan dan nilai bisnis yang berfungsi untuk menciptakan transparansi serta peluang untuk menginspeksi dan mengadaptasi. Artefak tersebut digunakan guna memaksimalkan transparansi informasi utama supaya setiap orang memiliki pemahaman yang sama mengenai artefak tersebut. Berikut beberapa artefak pada Scrum Menurut (Ken & Jeff, 2017):

a. Product Backlog adalah daftar terurut semua hal yang telah diketahui hingga saat ini harus ada di dalam produk. Product Backlog berisi daftar kebutuhan yang diketahui dan dipahami saat ini. Product Backlog berisi daftar dari seluruh fitur, fungsi, kebutuhan, peningkatan, dan perbaikan yang perlu diberlakukan terhadap produk pada rilis mendatang.
b. Sprint Backlog merupakan daftar Product Backlog item yang terpilih untuk Sprint ditambah perencanaan untuk menghantarkan Increment dan mencapai Sprint Goal. Sprint Backlog adalah perencanaan yang cukup rinci sehingga perubahan yang sedang dikerjakan dapat dipahami pada saat Daily Scrum.
c. Increment adalah manifestasi dari Product Backlog item yang diselesaikan dalam Sprint dan total nilai bisnis Increment dari seluruh Sprint yang lalu. Di akhir Sprint, Increment yang baru harus "Selesai", yang artinya Increment

tersebut harus berada pada kondisi yang dapat digunakan dan sesuai dengan definisi “Selesai” milik Tim Scrum.

2.4 Bahasa Skrip

2.4.1 PHP

PHP adalah bahasa pemrograman yang dapat dimplementasikan untuk membuat website dan berbentuk scripting. PHP tidak sebagai compiler, melainkan bertindak sebagai interpreter. PHP merupakan kode-kode yang berguna dalam mengembangkan sebuah halaman web menjadi lebih dinamis.

Terdapat aturan penulisan pada PHP. Dengan mengikuti kaidah penulisan PHP, kita dapat mengembangkan suatu program yang dikerjakan. Untuk memulai program PHP, kita harus menuliskan tag <?php. Kemudian kita dapat menulis kode program dari algoritma yang telah dibuat didalamnya. Untuk mengakhiri kode program PHP, kita dapat menutupnya dengan ?>.

PHP mempunyai sejumlah built-in function untuk tujuan pengiriman data ini, yang paling umum menggunakan echo() dan print(). Contohnya sebagai berikut :

```
Echo ‘Halo Dunia’;
Echo “Halo Dunia”;
```

Kode juga dapat ditulis menggunakan print(), seperti :

```
Print ‘Halo Dunia’;
Print “Halo Dunia”;
```

PHP merupakan bahasa server-side, kode yang dibuat dalam PHP diakses oleh user melalui server, kemudian server mengirim permintaan halaman web yang dikunjungi. Ketika user mengunjungi halaman web yang ditulis dalam PHP, server membaca skrip PHP, lalu memrosesnya menurut arah skrip. Pada Script PHP memanggil server untuk kemudian mengirim data dalam bentuk script HTML agar dapat dijalankan dalam web browser.

2.4.2 Javascript

JavaScript adalah bahas scripting yang digunakan oleh client atau bekerja pada browser dan berfungsi untuk membuat situs web menjadi lebih interaktif (Sigit, 2011). Salah satu fungsi Javascript adalah supaya dapat terintegrasi dalam HTML. Kebanyakan web browser mendukung penggunaan JavaScript, walaupun ada beberapa browser yang memberikan pilihan pada pengguna untuk menonaktifkannya.

Javascript dapat dihubungkan dengan objek-objek dalam program sehingga dapat melakukan kontrol atas objek-objek tersebut. JavaScript berisi library standar objek, seperti Array, date dan math, dan juga seperangkat elemen bahasa operator, struktur kontrol dan statements.JavaScript dapat digunakan ekstensi untuk berbagai keperluan dengan memberikan objek tambahan.

2.5 HTML (Hyper Text Markup Language)

Menurut Nugroho (2004), HTML adalah bahasa Scripting yang berguna untuk menuliskan halaman Web. Pada suatu halaman web, HTML berperan sebagai bahasa script utama yang berjalan dengan bahasa pemrograman lainnya. HTML memiliki 3 struktur dasar, yaitu tag, element, dan atribut. Untuk penjelasannya sebagai berikut :

1. Tag merupakan teks berupa dua karakter < dan >. Jika dituliskan misalnya `<html>`, `<head>`, `<body>`, dan `<p>`.
2. Untuk element html memiliki 3 komponen, antara lain tag sebagai pembuka, isi yang ingin ditampilkan, serta tag sebagai penutup untuk tag pembuka. Jika dituliskan misalnya:

   ```
   <p>Contoh element dari html</p>.
   ```

3. Atribut merupakan script didalam element yang berfungsi untuk memberikan informasi tambahan untuk element tersebut. Nilai suatu atribut harus ditutup dengan tanda kutip.

2.6 MySQL

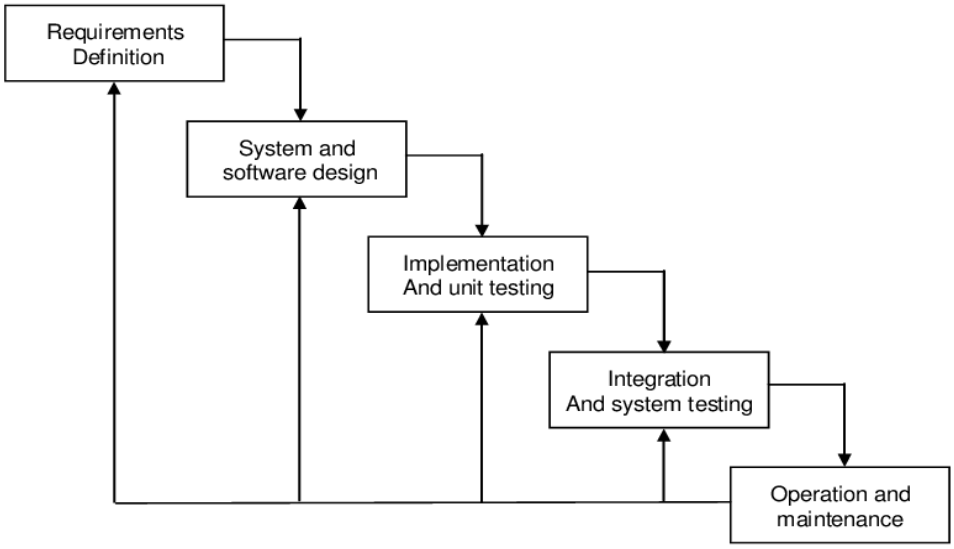
MySQL merupakan sistem dalam mengelola database SQL yang bersifat *open source* dan didukung serta dikembangkan oleh MySQL AB (Tjahyadi, et al., 2007). Dengan MySQL kita dapat membuat database dan melakukan query ke server MySQL. MySQL memiliki antarmuka yang dapat diakses melalu web browser dan juga dapat dijalankan melalui *command-line*. MySQL dapat berjalan dengan cepat dan mudah dioperasikan. MySQL merupakan *relational database management system.* Selain itu, dapat berjalan pada client atau server pada embedded system (Tjahyadi, et al., 2007). MySQL merupakan database yang cukup terkenal di dunia pada saat ini, karena bahasa yang digunakan untuk mengakses database terebut adalah SQL atau *Structured Query Language.* Didalam MySQL kita dapat membuat database yang menyimpan tabel tabel. Setiap tabel bisa memiliki banyak column sesuai dengan kebutuhan program. Setiap kolom memiliki tipe data, panjang isi, serta atribut atribut lain yang dapat diatur ketika ataupun setelah membuat kolom tersebut.

2.7 SDLC

SDLC menggambarkan tentang siklus dari pembuatan atau pengembangan sistem. Berperan sebagai metode dalam mengembangkan sistem tradisional yang telah membantu banyak perusahaan besar. SDLC merupakan suatu kerangka kerja yang didalamnya terdapat proses-proses berurutan untuk mengembangkan suatu sistem. Ada banyak metode yang untuk mengembangkan sistem, seperti : *waterfall*, *V model*, *prototyping*, metode formal, dan *Extreme Programming.* Untuk penelitian ini, penulis menggunakan SDLC *Waterfall*.

2.7.1 Metode *Waterfall*

Metode *waterfall* bersifat berurutan dan sistematis dalam pegembangan system. Tahapannya dimulai dari analisis, desain, pengkodean, pengujian dan pemeliharaan. Berikut merupakan gambar metode *waterfall*.



Gambar 2.1 Metode Waterfall

Sumber : (Sommerville, 2011)

Menurut (Sommerville, 2011) terdapat beberapa fase dalam metode Waterfall, antara lain:

1. Analisis Kebutuhan

   Layanan dari sistem, kendala, dan tujuan sistem ditetapkan melalui konsultasi dengan pengguna sistem. Kemudian dilakukan pendefinisian secara rinci dan berfungsi sebagai spesifikasi sistem.

2. Desain sistem dan perangkat lunak

   Proses desain sistem mengalokasikan persyaratan untuk sistem perangkat keras atau perangkat lunak dengan membangun arsitektur sistem secara keseluruhan. Desain perangkat lunak melibatkan mengidentifikasi dan menggambarkan tentang abstraksi dari sistem perangkat lunak mendasar dan hubungannya.

3. Implementasi dan pengujian unit

   Desain perangkat lunak diwujudkan sebagai satu set program atau unit program. Pengujian unit melibatkan verifikasi bahwa setiap unit memenuhi spesifikasinya.

4. Integrasi dan pengujian sistem

   Masing-masing unit diintegrasikan dan dilakukan pengujian sebagai sistem yang komplit guna memastikan bahwa persyaratan perangkat lunak terpenuhi.

5. Operasi dan pemeliharaan

   Biasanya ini merupakan fase siklus hidup terpanjang. Sistem ini diinstal dan digunakan secara praktis. Pemeliharaan melibatkan perbaikan kesalahan yang tidak ditemukan pada tahap awal siklus hidup, meningkatkan implementasi unit sistem dan meningkatkan layanan sistem ketika persyaratan baru ditemukan.

2.8 Laravel Framework

Laravel adalah salah satu *framework* untuk pengembangan aplikasi web dengan sintaksis yang ekspresif dan elegan dan memberikan solusi untuk pengembangan dengan memfasilitasi tugas umum di sebagian besar proyek web besar (Parkar, et al., 2016). Laravel dirilis dibawah lisensi MIT dan menerapkan metode pengembangan *MVC (Model, View, Controller). Framework* ini bertujuan supaya memudahkan developer dalam membangun sistem berbasis web. MVC merupakan pendekatan perangkat lunak yang mana memisahkan komponen logika, manipulasi data dan presentasi atau user interface.

1. *Model*, bertugas mewakili struktur data dan pengelolaan basis data mulai dari memasukkan data hingga perbaruhan pada data termasuk penghapusan data.
2. *View*, merupakan bagian yang berisi tamilan antar muka untuk pengguna.
3. *Controller*, berisi kode kode logika pada program yang dibuat dan bertugas untuk menjembatani antara model dan view.

Alur kerjanya, ketika terdapat *request* dari pengguna yaitu dari view, maka akan diterima oleh controller untuk diproses. Kemudian jika *request* memerlukan data untuk diambil atau diolah maka proses menggunakan *model*, lalu dikembalikan ke *controller* dan ditampilkan di *view* sebagai tampilan antarmuka. Laravel sendiri memiliki beberapa fitur yang dapat digunakan sepeerti *bundles, Eloquent ORM, Routing, Restful controller, Migration, Unit testing,* dan lain lain. Fitur tersebut dapat digunakan sesuai dari fungsi masing masing fitur, yang mana akan mempermudah programmer dalam melakukan pengembangan program.

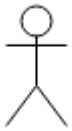
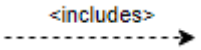
2.9 Bahasa Pemodelan

2.9.1 Use Case Diagram

Use Case Diagram merupakan penggambaran dari aktor, *use case*, dan interaksi serta menunjukkan gambaran luar dari perilaku sistem. *Use case diagram* memberikan gambaran singkat tentang hubungan antara aktor, use case, dan sistem serta tidak menjelaskan secara detail. Berikut adalah elemen pada use case diagram.

Tabel 2.2 Elemen elemen pada use case diagram

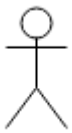
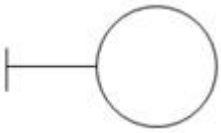
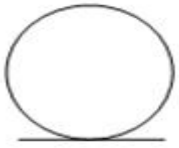
Gambar	Nama	Fungsi

	Aktor	Mempresentasikan orang atau komponen lain yang melakukan interaksi dengan sistem
	Use case	Gambaran fungsionalitas dari suatu sistem, sehingga pengguna sistem mengerti mengenai kegunaan sistem yang akan dibangun
	Asosiasi	Menghubungkan link antar element
	<<Include>>	Kelakuan yang harus terpenuhi supaya event dapat terjadi. Pada kondisi ini sebuah use case merupakan bagian dari use case lainnya.

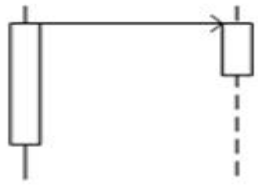
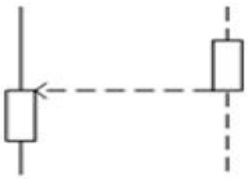
2.9.2 Sequence Diagram

Sequence diagram adalah sebuah diagram interaksi yang menekankan pada urutan waktu pesan dimana memperlihatkan objek objek dan pesan yang dikirim dan diterima.

Tabel 2.3 Komponen pada sequence diagram

Gambar	Nama	Keterangan
	Aktor	Mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem
	Boundary	Mempresentasikan Batasan dan juga interaksi yang dilakukan aktor dengan sistem.
	Control	Menggambarkan “perilaku mengatur” dan mengatur urutan kerja suatu sistem.
	Entity	Sebagai media untuk menyimpan informasi.

Tabel 2.3 Komponen pada sequence diagram

Gambar	Nama	Keterangan
	Object Message	Mempresentasikan pesan antar obyek yang menunjukkan urutan kejadian yang terjadi
	Return Message	Menggambarkan pesan antar obyek, yang menunjukan urutan kejadian yang terjadi.

2.9.3 Class Diagram

Menurut (Sommerville, 2011) *Class Diagram* digunakan ketika mengembangkan model sistem berorientasi objek untuk ditampilkan kelas-kelas dalam suatu sistem dan asosiasi antara kelas-kelas. *Class Diagram* dalam *UML* dapat diekspresikan pada level detail yang berbeda. Ketika mengembangkan model, tahap pertama biasanya adalah melihat dunia, mengidentifikasi objek penting, dan mewakili ini sebagai kelas. Cara paling sederhana untuk menulis ini adalah dengan tulis nama kelas dalam kotak. Selain itu dapat juga mencatat keberadaan asosiasi dengan menggambar garis antar kelas.

2.10 Pengujian Perangkat Lunak

Pengujian adalah metode untuk melakukan verifikasi dengan tujuan mencari kesalahan pada sebuah aplikasi. Dari hasil verifikasi tersebut, hasilnya didokumentasikan dan ditinjau ulang lebih lanjut untuk menentukan kapabilitas dari kebutuhan perangkat lunak (SoftRel.org). Pengujian perangkat lunak diasumsikan sebagai siklus hidup dari awal hingga akhir proses dari rekayasa perangkat lunak untuk mengukur kualitas, kehandalan perangkat lunak yang diuji.

2.10.1 White Box Testing

Whitebox Testing adalah jenis pengujian yang lebih fokus pada isi atau *source code* dari perangkat lunak. *Whitebox Testing* menggunakan pengetahuan tentang logika internal perangkat lunak untuk membuat kasus pengujian mencapai kriteria cakupan yang diinginkan (Farrel-Vinay, 2008). Secara waktu pengujian white box lebih lama karena memerlukan ketelitian dalam mengujinya. Jenis pengujian ini hanya bisa berjalan ketika sistem telah selesai dikerjakan. Prinsip keluaran dari tipe pengujian ini adalah :

1. Menjamin semua alur program independent sudah dilakukan pengujian sebanyak paling sedikit sekali.

2. Telah dilakukan pengujian pada semua percabangan dengan nilai berhasil dan gagal.

3. Telah dilakukan pengujian pada perulangan dengan kondisi normal dan melebihi nilai batas perulangan.

4. Telah dilakukan pengujian pada struktur data internal supaya terjaga validitasnya.

2.10.2 Black Box Testing

Blackbox Testing berisi proses menguji sistem tanpa menggunakan pengetahuan desain atau konstruksinya, hanya menggunakan kebutuhan dari awal perancangan. Fungsi perangkat lunak diuji terhadap spesifikasinya (Farrel-Vinay, 2008). Para penguji memandang perangkat lunak seperti sebuah "kotak hitam" yang tidak dapat dilihat isinya namun cukup dilakukan proses pengujian di bagian luar.

Beberapa teknik pengujian yang tergolong dalam *Blackbox Testing* adalah (Farrel-Vinay, 2008) :

1. *Equivalence Partitioning*

   Setiap masukan dikelompokkan dalam kelompok tertentu untuk selanjutnya dilakukan perbandingan hasil keluarannya.

2. *Boundary Value Analysis*

   Pada teknik ini, dilakukan masukan yang melebihi batasan dari sebuah data. Jika perangkat lunak berhasil mengatasi masukan yang salah maka teknik ini selesai dilakukan.

3. *Cause Effect Graph*

   Pada teknik ini, dilakukan proses pengujian yang menghubungkan sebab dari sebuah masukan dan akibatnya ada keluaran yang dihasilkan.

4. *Random Data Selection*

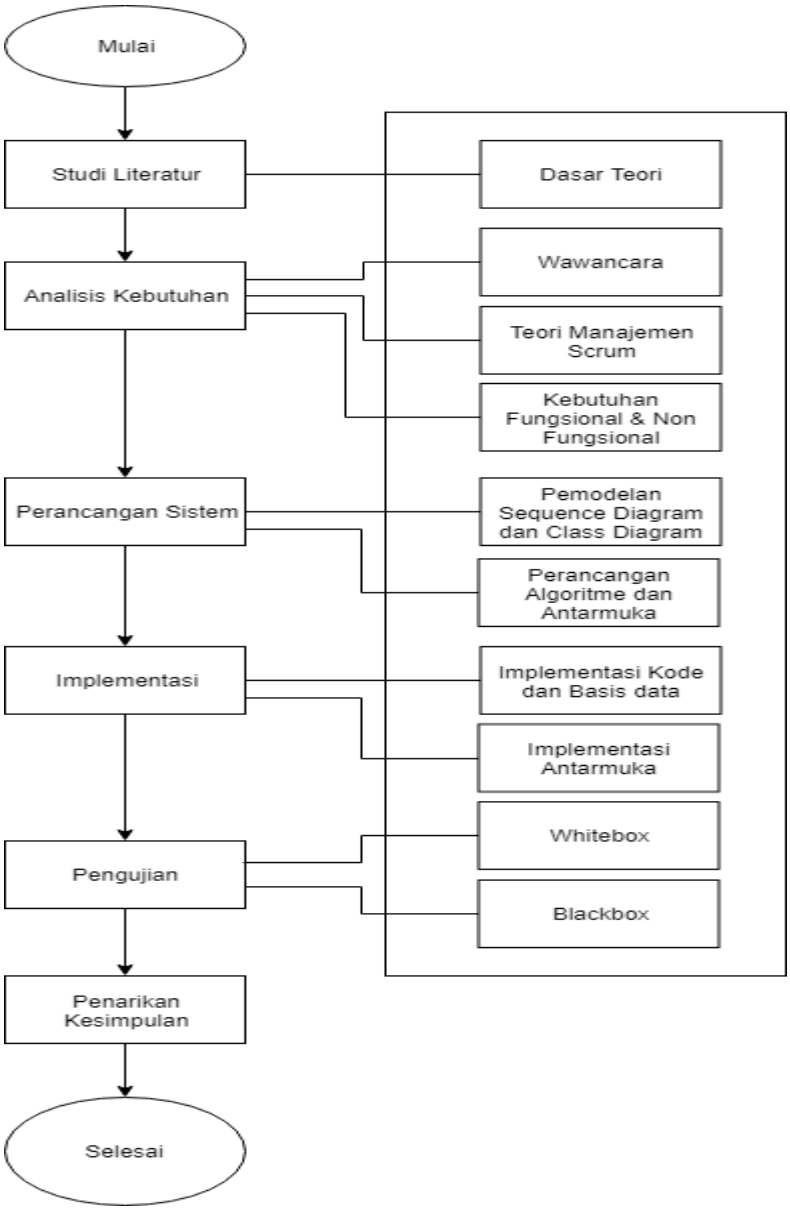
   Teknik ini berusaha untuk melakukan proses memasukkan data dengan menggunakan nilai acak. Dari hasil masukan tersebut kemudian dibuat tabel yang menyatakan validitas dari keluaran yang dihasilkan.

5. *Feature Test*

   Pada teknik ini, dilakukan proses pengujian terhadap spesifikasi dari perangkat lunak yang telah selesai dikerjakan.

BAB 3 METODOLOGI

Pada bab ini penulis akan menguraikan langkah-langkah dalam perencanaan penelitian yang akan dilakukan dalam menyelesaikan penelitian ini. Metode penelitian digunakan sebagai landasan atau pedoman dalam mengerjakan penelitian. Langkah langkahnya dimulai dari studi literatur, analisis kebutuhan, perancangan sistem, implementasi, pengujian, penarikan kesimpulan. Diagram alir ini dapat dilihat pada gambar 3.1.



Gambar 3.1 Diagram Alir Penelitian

3.1 Studi Literatur

Studi Literatur berisi studi kepustakaan yang dapat menunjang penulisan penelitian. Sumber yang digunakan berasal dari buku, jurnal ilmiah, situs, dan

tulisan yang berhubungan dengan analisis dan perancangan sistem. Selain itu, penulis juga melakukan analisis tentang *framework* serta teknologi yang akan digunakan dalam pengembangan Aplikasi Manajemen Multi Proyek. Sehingga pokok bahasan dalam penelitian ini adalah manajemen proyek, metode Scrum, sistem, bahasa PHP, HTML, javascript, css, MySQL, *software development life cycle* (SDLC) pendekatan *waterfall*, pemodelan *Unified Modeling Language* (UML), metode *Model View Controller* (MVC) dengan kerangka kerja Laravel, dan pegujian perangkat lunak.

3.2 Analisis Kebutuhan

Pada tahap ini, analisis kebutuhan dilakukan dengan dengan mengidentifikasi seluruh kebutuhan oleh pengguna. Selain itu, dapat diperoleh suatu kebutuhan dari metode yang digunakan yaitu *Scrum*. Untuk pemodelan yang akan diterapkan merupakan pemodelan kebutuhan berorientasi objek yang mana akan memanfaatkan diagram UML. Selai itu, digunakan pula *Usecase Diagram* dan *Usecase Scenario. Usecase Diagram* mengambarkan gambaran umum aplikasi dari tampak luar, yang berisi aktor pengguna sistem disertai fungsi yang dapat dilakukan. Selain itu, dilakukan pula pemodelan terhadap analisis data. Pemodelan tersebut menggunakan *Entity Relationship Diagram*. Pemodelan ERD berguna untuk menjelaskan hubungan antar data di dalam suatu basis data dan relasi antar objek.

3.3 Perancangan Sistem

Perancangan sistem dilakukan sebagai landasan dari implementasi. Perancangan ini bersumber pada analisis kebutuhan. Dari kebutuhan yang diperoleh kemudian akan dilakukan pemodelan dalam bentuk *Unfied Modeling Language* (UML). Pemodelan UML yang akan dilakukan adalah perancangan *Sequence Diagram*, perancangan *Class Diagram*, perancangan algoritme, dan perancangan antarmuka.

3.4 Implementasi

Pada tahap ini, penulis akan mengubah hasil perancangan yang telah dilakukan kedalam bentuk kode kode pemrograman, sehingga akan dapat dimengerti oleh komputer. Bahasa pemrograman yang akan digunakan adalah bahasa PHP dibantu dengan *framework* Laravel. Untuk tampilan antarmuka menggunakan HTML dan CSS dibantu dengan *famework* BootStrap. Untuk basis data menggunakan MySQL.

3.5 Pengujian

Pengujian dilakukan untuk menemukan kesalahan pada Aplikasi seperti menguji apakah semua kebutuhan telah dapat dan sesuai dengan analisis kebutuhan. Pengujian yang akan dilakukan menggunakan metode *Black box Testing* dan *White box Testing*. Pada *White box Testing* akan dilakukan pengujian jalur dasar dari algoritma yang telah dirancang. Tujuannya untuk mengukur kompleksitas algoritma. Untuk *Black Box Testing* dilakukan *Requirement Testing*

yang bertujuan supaya mengetahui mengenai apakah aplikasi telah sesuai dengan kebutuhan. Untuk pengujian integrasi dapat dilakukan dengan menguji unit yang berinteraksi dalam suatu sistem untuk dapat menghasilkan suatu fungsional. Selain itu terdapat pengujian *Compatibility* untuk memeriksa apakah aplikasi dapat berjalan atau diakses pada beberapa *browser* dan versi browsernya, serta pada beberapa *platform.*

3.6 Penarikan Kesimpulan

Penarikan kesimpulan dilakukan setelah semua tahapan dalam metode penelitian ini telah dilakukan. Kesimpulan diperoleh dari hasil perancangan dan pengujian. Kesimpulan yang diperoleh harus dapat menjawab rumusan masalah yang telah dipaparkan sebelumnya. Sehingga diharapkan dapat memberikan inti dari keseluruhan hasil penelitian. Dengan begitu, diharapkan dari hasil kesimpulan diperoleh saran untuk perancangan pengembangan aplikasi perangkat lunak selanjutnya.

BAB 4 ANALISIS KEBUTUHAN

4.1 Deskripsi Aplikasi

Aplikasi ini berguna untuk memudahkan tim dalam pengembangan perangkat lunak pada suatu *software house*. Aplikasi ini dapat mengelola proyek yang sedang berjalan. Setiap anggota dapat berkontribusi dalam proses pengelolaan suatu proyek. Semua aktivitas dari anggota tim juga akan terekam dan dapat diketahui perubahan apa saja yang telah dilakukan. Karena menggunakan *Scrum*, maka beberapa kebutuhan juga berasal dari teori tentang *Scrum*. Kebutuhan tersebut meliputi Tugas Backlog yang berisi tugas tugas yang harus diselesaikan tim, Sprint yang berjalan pada rentang waktu yang telah ditetapkan, *Scrum Board* untuk mengubah status tugas.

Pada hasil wawancara diperoleh bahwa terdapat beberapa permasalahan pada software house. Permasalahan tersebut dapat berguna untuk membentuk suatu kebutuhan pada aplikasi. Permasalahan pertama adalah manajer proyek tidak dapat memantau perkembangan pengerjaan terkini dari anggota tim setiap waktunya, karena pembahasan tersebut biasanya dilakukan seminggu sekali, atau pada waktu hari kerja sebelum pulang, atau manajer proyek harus menanyakan sendiri kepada tiap anggotanya. Berdasarkan masalah tersebut, dapat diatasi menggunakan scrum, yaitu *scrum board*. Melalui *scrum board*, manajer proyek dapat melihat status pengerjaan dari tugas tugas yang diberikan. Terdapat tiga status yaitu, "*To Do, "Doing", "Done".* Selain tidak dapat memantau perkembangan terkini, manajer proyek juga tidak dapat secara langsung memberikan tugas untuk anggota tim. Maka dari itu, pada scrum terdapat *sprint backlog* yang merupakan tugas yang harus diselesaikan pada sprint yang sedang berjalan dan ada *backlog* yang berisi keseluruhan tugas yang telah diketahui. Dengan begitu manajer proyek dapat langsung memberikan tugas baru dan pekerja bisa melihat tugas baru tersebut untuk kemudian dikerjakan.

Pada salah satu *software house* terdapat permasalahan *miscommunication* antar pekerja. Sedangkan pada scrum terdapat *daily scrum* yang merupakan acara harian dengan durasi waktu 15 menit. Karena struktur dari pertemuan *daily scrum* dapat diadakan melalui berbagai cara, maka peneliti menerapkannya dengan memberi fitur komentar untuk setiap tugas. Dengan berkomentar pekerja dapat menanyakan atau saling memberi informasi dari suatu tugas. Selain itu, fungsi dari daily scrum adalah untuk meningkatkan kualitas komunikasi, dan mengidentifikasi hambatan untuk dihilangkan. Beberapa *software house* melakukan rapat mingguan pada hari tertentu, yang mana hal tersebut sama dengan salah satu acara pada scrum, yaitu *Sprint*. Sehingga, pada acara rapat dapat juga dibahas mengenai tugas yang telah selesai maupun belum selesai, serta merencanakan tugas tugas yang akan dikerjakan pada *sprint* selanjutnya.

Tabel 4.1 Permasalahan dan Solusi

No.	Software House	Permasalahan	Solusi	Target
1.	Profile Image Studio, Komal Developmet, Sarana Utama Solusindo.	Manajer proyek tidak dapat memantau perkembangan pengerjaan terkini dari anggota tim setiap waktunya, karena pembahasan tersebut biasanya dilakukan seminggu sekali, atau pada waktu hari kerja sebelum pulang, atau manajer proyek harus menanyakan sendiri kepada tiap anggotanya.	Dapat menggunakan *scrum*. Melalui *scrum board*, manajer proyek dapat melihat status pengerjaan dari tugas tugas yang diberikan. Terdapat tiga status yaitu, "*To Do, "Doing", "Done"*	Selesai
2.	Profile Image Studio, Komal Developmet, Sarana Utama Solusindo.	Manajer proyek tidak dapat secara langsung memberikan tugas untuk anggota tim.	Pada *scrum* terdapat *sprint backlog* yang merupakan tugas yang harus diselesaikan pada sprint yang sedang berjalan dan ada *backlog* yang berisi keseluruhan tugas yang telah diketahui	Selesai
3.	Komal Developmet	Adanya *Miscommunication* antar pekerja	Pada *scrum* terdapat *daily scrum* yang merupakan acara harian dengan durasi waktu 15 menit. Karena struktur dari pertemuan *daily scrum* dapat diadakan melalui berbagai cara, maka peneliti menerapkannya dengan memberi fitur komentar untuk setiap tugas	Selesai
4.	Profile Image Studio,	Terdapat rapat mingguan pada hari tertentu.	Hal tersebut sama dengan salah satu acara pada *scrum*, yaitu Sprint.	Selesai

Tabel 4.1 Permasalahan dan Solusi

No.	Software House	Permasalahan	Solusi	Target
	Sarana Utama Solusindo.		Dengan begitu pada rapat tersebut dapat juga dibahas mengenai tugas yang telah selesai maupun belum selesai, serta merencanakan tugas tugas yang akan dikerjakan pada sprint selanjutnya	

4.2 Kebutuhan Pengguna Aplikasi

4.2.1 Analisis Pengguna Aplikasi

Pada aplikasi Manajemen Multi Proyek ini terdapat beberapa pengguna antara lain, pengguna dan pengunjung. Untuk penjelasan dari pengguna tersebut akan dipaparkan pada tabel 4.2.

Tabel 4.2 Analisis Pengguna

No.	Jenis Pengguna	Penjelasan
1.	Pengunjung	Merupakan pengguna aplikasi yang belum masuk ke dalam sistem. Pengunjung hanya dapat menggunakan fitur login dan register
2.	Pengguna	Merupakan pengunjung aplikasi yang telah terdaftar dan masuk ke dalam aplikasi Manajemen Multi Proyek. Pengguna dapat menjalakan atau memiliki hak akses terhadap semua fungsi yang ada pada aplikasi Manajemen Multi Proyek.

4.2.2 Kebutuhan Fungsional dan Spesifikasi Kebutuhan

A. Pengunjung

Berikut daftar kebutuhan untuk Pengunjung setelah dilakukan wawancara kepada dua software house, akan dipaparkan pada tabel 4.3.

Tabel 4.3 Kebutuhan Fungsional Pengunjung

No.	**Kode**	**Nama Fungsi**	**Deskripsi**
1.	MMPS_F_01	Register	Aplikasi dapat menyediakan fitur register untuk pengunjung aplikasi.

Tabel 4.3 Kebutuhan Fungsional Pengunjung

No.	Kode	Nama Fungsi	Deskripsi
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form register untuk memasukkan data nama, e-mail, password, dan konfirmasi password untuk mendaftar. (MMPS_F_01_01) 2. Aplikasi harus menyediakan tombol register. (MMPS_F_01_02)
2.	MMPS_F_02	Login	Aplikasi dapat menyediakan fitur login untuk pengguna yang sudah terdaftar.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form login untuk memasukkan data e-mail dan password yang telah digunakan pada saat registrasi. (MMPS_F_02_01) 2. Aplikasi harus menyediakan tombol login. (MMPS_F_02_02)

B. Pengguna

Berikut daftar kebutuhan untuk Pengguna setelah dilakukan pendalaman materi tentang metode Scrum dan wawancara kepada dua software house, akan dipaparkan pada tabel 4.4.

Tabel 4.4 Kebutuhan Fungsional Pengguna

No.	Kode	Nama Fungsi	Deskripsi
1.	MMPS_F_03	Melihat daftar Proyek	Aplikasi harus dapat menampilkan semua Proyek yang dikerjakan oleh pengguna tesebut.
			Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan nama proyek, dan nama proyek manager. (MMPS_F_03_01) 2. Aplikasi harus menyediakan hypelink untuk masuk kedalam manajemen proyek yang dipilih. (MMPS_F_03_02)
2.	MMPS_F_04	Membuat Proyek	Aplikasi harus menyediakan fitur tambah proyek untuk Pengguna.

Tabel 4.4 Kebutuhan Fungsional Pengguna

No.	Kode	Nama Fungsi	Deskripsi
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form tambah proyek dengan masukan berupa nama proyek dan deskripsi untuk proyek. (MMPS_F_04_01) 2. Aplikasi harus menyediakan tombol untuk membuat proyek tersebut.
3.	MMPS_F_05	Melihat daftar Modul	Aplikasi harus dapat menampilkan semua modul dari proyek yang dibuka.
			Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan nama modul, tanggal mulai, dan tanggal selesai. (MMPS_F_05_01) 2. Aplikasi harus menyediakan hypelink untuk masuk kedalam manajemen proyek yang dipilih. (MMPS_F_05)
4.	MMPS_F_06	Merubah Timeline Modul	Aplikasi harus menyediakan fitur untuk mengubah tanggal mulai dan tanggal selesai dari setiap modul.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form untuk merubah timeline modul dengan masukan berupa tanggal mulai dan tanggal selesai. (MMPS_F_06_01) 2. Kolom tanggal mulai dan tanggal selesai ditampilkan dalam bentuk datepicker. (MMPS_F_06_02) 3. Form harus menyediakan tombol untuk merubah timeline modul. (MMPS_F_06_03)
5.	MMPS_F_07	Membuat Modul	Aplikasi harus menyediakan fitur tambah modul pada Proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form tambah modul dengan masukan berupa nama modul. (MMPS_F_07_01)

Tabel 4.4 Kebutuhan Fungsional Pengguna

No.	Kode	Nama Fungsi	Deskripsi
			2. Form harus menyediakan tombol untuk menambah modul tersebut. (MMPS_F_07_02)
6.	MMPS_F_08	Mengubah data Modul	Aplikasi harus menyediakan fitur untuk mengubah data modul pada Proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form ubah data modul dengan masukan berupa nama modul. (MMPS_F_08_01) 2. Form harus menyediakan tombol untuk mengubah data modul. (MMPS_F_08_02)
7.	MMPS_F_09	Menghapus Modul	Aplikasi harus menyediakan fitur untuk menghapus modul pada Proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan tombol untuk menghapus modul. (MMPS_F_09_01)
8.	MMPS_F_10	Mengubah data Tugas	Aplikasi harus menyediakan fitur untuk mengubah data tugas pada Proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form ubah data tugas dengan masukan berupa nama modul dan nama modul. (MMPS_F_10_01) 2. Form harus menyediakan tombol untuk mengubah data modul. (MMPS_F_10_02)
9.	MMPS_F_11	Menghapus Tugas	Aplikasi harus menyediakan fitur untuk menghapus tugas pada Proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan tombol untuk menghapus tugas. (MMPS_F_11_02)
10.	MMPS_F_12		Aplikasi harus dapat menampilkan semua tugas backlog pada proyek yang dibuka.

Tabel 4.4 Kebutuhan Fungsional Pengguna

No.	Kode	Nama Fungsi	Deskripsi
		Melihat daftar tugas Backlog	Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan nama tugas backlog. (MMPS_F_12_01) 2. Aplikasi harus menyediakan hypelink untuk masuk kedalam rincian tugas backlog yang dipilih. (MMPS_F_12_02
11.	MMPS_F_13	Membuat tugas Backlog	Aplikasi harus menyediakan fitur tambah tugas Backlog.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form tambah tugas backlog dengan masukan berupa nama tugas backlog. (MMPS_F_13_01) 2. Aplikasi harus menyediakan tombol untuk membuat tugas backlog tersebut. (MMPS_F_13_02)
12.	MMPS_F_14	Mengubah posisi tugas Backlog	Aplikasi harus menyediakan fitur untuk mengubah posisi tugas backlog.
			Spesifikasi Kebutuhan : 1. Posisi tugas backlog hanya dapat dirubah ke dalam posisi tugas sprint. (MMPS_F_14_01) 2. Aplikasi harus menyediakan fitur drag and drop untuk melakukan perubahan posisi pada tugas backlog. (MMPS_F_14_02)
13.	MMPS_F_15	Melihat daftar tugas Sprint	Aplikasi harus dapat menampilkan tugas sprint pada proyek yang dibuka.
			Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan nama tugas sprint. (MMPS_F_15_01) 2. Aplikasi harus menyediakan hypelink untuk masuk kedalam rincian tugas sprint yang dipilih. (MMPS_F_15_02) 3. Aplikasi hanya menampilkan tugas sprint dari sprint yang berjalan ketika

Tabel 4.4 Kebutuhan Fungsional Pengguna

No.	Kode	Nama Fungsi	Deskripsi
			aplikasi diakses dan tugas sprint yang belum masuk kedalam suatu sprint sebelumnya. (MMPS_F_15_03)
14.	MMPS_F_16	Membuat tugas Sprint	Aplikasi harus menyediakan fitur tambah tugas Backlog.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form tambah tugas sprint dengan masukan berupa nama tugas sprint. (MMPS_F_16_01) 2. Aplikasi harus menyediakan tombol untuk membuat tugas backlog tersebut. (MMPS_F_16_02)
15.	MMPS_F_17	Mengubah posisi tugas Sprint	Aplikasi harus menyediakan fitur untuk mengubah posisi tugas sprint.
			Spesifikasi Kebutuhan : 1. Posisi tugas sprint hanya dapat dirubah ke dalam posisi tugas backlog. (MMPS_F_17_01) 2. Aplikasi harus menyediakan fitur drag and drop untuk melakukan perubahan posisi pada tugas sprint. (MMPS_F_17_02)
16.	MMPS_F_18	Memulai Sprint	Aplikasi harus dapat memulai kegiatan sprint pada suatu proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan tombol untuk membuka pilihan berapa lama sprint tersebut akan dijalankan. (MMPS_F_18_01) 2. Aplikasi menyediakan pilihan waktu sprint untuk satu, dua, tiga, empat minggu. (MMPS_F_18_02) 3. Aplikasi harus menyediakan tombol untuk memulai sprint. (MMPS_F_18_03)

Tabel 4.4 Kebutuhan Fungsional Pengguna

No.	Kode	Nama Fungsi	Deskripsi
17.	MMPS_F_19	Mengakhiri Sprint	Aplikasi harus dapat mengakhiri kegiatan sprint yang sedang berjalan pada suatu proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan tombol untuk melihat total tugas sprint yang belum selesai pada sprint tersebut. (MMPS_F_19_01) 2. Aplikasi harus menyediakan tombol untuk mengakhiri sprint. (MMPS_F_19_02)
18.	MMPS_F_20	Melihat daftar tugas Sprint pada Scum Board	Aplikasi harus dapat menampilkan tugas sprint yang sedang berjalan pada proyek yang dibuka.
			Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan nama tugas sprint dan tanggal dibuatnya tugas sprint. (MMPS_F_20_01) 2. Aplikasi harus menyediakan hypelink untuk masuk kedalam rincian tugas sprint yang dipilih. (MMPS_F_20_02) 3. Aplikasi harus dapat mengkategorikan tugas sprint kedalam 3 status, yaitu To Do, Doing, dan Done. (MMPS_F_20_03) 4. Aplikasi hanya menampilkan tugas sprint dari sprint yang berjalan ketika aplikasi diakses. (MMPS_F_20_04)
19.	MMPS_F_21	Mengubah status tugas Sprint	Aplikasi harus menyediakan fitur untuk mengubah status tugas sprint pada sprint yang sedang berjalan melalui srum board.
			Spesifikasi Kebutuhan : 1. Status tugas sprint dapat dirubah ke dalam status lainnya. (MMPS_F_21_01)

Tabel 4.4 Kebutuhan Fungsional Pengguna

No.	Kode	Nama Fungsi	Deskripsi
			2. Aplikasi harus menyediakan fitur drag and drop untuk melakukan perubahan status pada tugas sprint. (MMPS_F_21_02)
20.	MMPS_F_22	Melihat semua Sprint	Aplikasi harus dapat menampilkan semua sprint dan tugas sprint dari proyek yang dibuka.
			Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan Sprint ke berapa beserta tanggal mulai dan berakhirnya Sprint. (MMPS_F_22_01) 2. Aplikasi harus dapat menampilkan nama tugas tugas sprint yang selesai pada setiap Sprint. (MMPS_F_22_02) 3. Aplikasi harus menyediakan hypelink untuk masuk kedalam rincian tugas sprint yang dipilih. (MMPS_F_22_03)
21.	MMPS_F_23	Melihat daftar Anggota Tim	Aplikasi harus dapat menampilkan semua anggota tim yang terlibat pada pengerjaan proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan nama dan email manajer proyek. (MMPS_F_23_01) 2. Aplikasi harus dapat menampilkan nama dan email anggota yang terlibat pada proyek tersebut. (MMPS_F_23_02)
22.	MMPS_F_24	Menambah Anggota Tim	Aplikasi harus menyediakan fitur untuk menambah anggota dalam pengerjaan proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form tambah proyek dengan masukan berupa email pekerja. (MMPS_F_24_01)

Tabel 4.4 Kebutuhan Fungsional Pengguna

No.	Kode	Nama Fungsi	Deskripsi
			2. Aplikasi harus menyediakan tombol untuk membuat proyek tersebut. (MMPS_F_24_02)
23.	MMPS_F_25	Melihat Detail Tugas	Aplikasi harus dapat menampilkan informasi dari tugas yang dipilih.
			Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan nama tugas, status, waktu dibuatnya tugas tesebut, dan komentar komentar. (MMPS_F_25_01)
24.	MMPS_F_26	Melihat Komentar	Aplikasi harus dapat menampilkan komentar dari tugas yang dilihat.
			Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan nama orang yang berkomentar, isi komentar, dan waktu komentar. (MMPS_F_26_01)
25.	MMPS_F_27	Menambah Komentar	Aplikasi harus menyediakan fitur untuk menambah komentar pada tugas.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan form tambah komentar dengan masukan berupa isi komentar. (MMPS_F_27_01) 2. Aplikasi harus menyediakan tombol untuk menambah komentar. (MMPS_F_27_02)
26.	MMPS_F_28	Melihat Riwayat Proyek	Aplikasi harus dapat menampilkan informasi dari perubahan yang dilakukan oleh tim dalam suatu proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan nama orang yang melakukan perubahan atau penambahan atau penghapusan sesuatu dari proyek, aktivitas yang dilakukan, dan waktu

Tabel 4.4 Kebutuhan Fungsional Pengguna

No.	Kode	Nama Fungsi	Deskripsi
			aktivitas tersebut dilakukan. (MMPS_F_28_01)
27.	MMPS_F_29	Filter Tugas	Aplikasi harus dapat melakukan filter terhadap tugas.
			1. Filter dilakukan berdasarkan nama modul pada proyek tersebut. (MMPS_F_29_01) 2. Aplikasi hanya menampilkan tugas yang termasuk dalam modul yang dipilih. (MMPS_F_29_02) 3. Aplikasi dapat menyediakan pilihan untuk tidak melakukan filter. (MMPS_F_29_03) 4. Aplikasi harus menyediakan tombol untuk filter. (MMPS_F_29_04)
28.	MMPS_F_30	Keluar Proyek	Aplikasi harus menyediakan fitur bagi pengguna yang telah masuk kedalam proyek untuk keluar dari proyek tersebut.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan hypelink untuk keluar dari manajemen proyek yang sedan dibuka. (MMPS_F_30_01)
29.	MMPS_F_31	Logout	Aplikasi harus menyediakan fitur bagi pengguna yang telah login untuk keluar dari aplikasi.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan hypelink untuk keluar dari Aplikasi. (MMPS_F_31_01)
30.	MMPS_F_32	Report	Aplikasi harus dapat menampilkan daftar tugas berdasarkan modulnya dalam suatu proyek.
			Spesifikasi Kebutuhan : 1. Aplikasi harus dapat menampilkan nama modul, nama daftar tugas yang

Tabel 4.4 Kebutuhan Fungsional Pengguna

No.	Kode	Nama Fungsi	Deskripsi
			termasuk dari setiap modul dan status dari tugas. (MMPS_F_32_01) 2. Aplikasi harus dapat menampilkan daftar tugas yang belum memiliki modul. (MMPS_F_32_02)
32.	MMPS_F_33	Mengerjakan Ulang Tugas	Aplikasi harus menyediakan fitur bagi pengguna untuk memindahkan tugas yang telah selesai ke dalam sprint yang saat ini berjalan.
			Spesifikasi Kebutuhan : 1. Aplikasi harus menyediakan *option* untuk mengerjakan ulang pada setiap tugas. (MMPS_F_32_01) 2. Option akan muncul jika terdapat sprint yang sedang berjalan pada saat itu. (MMPS_F_32_02) 3. Aplikasi harus dapat mengubah status tugas yang dikerjakan kembali mejadi “Doing”. (MMPS_F_32_03)

4.2.3 Kebutuhan Non Fungsional

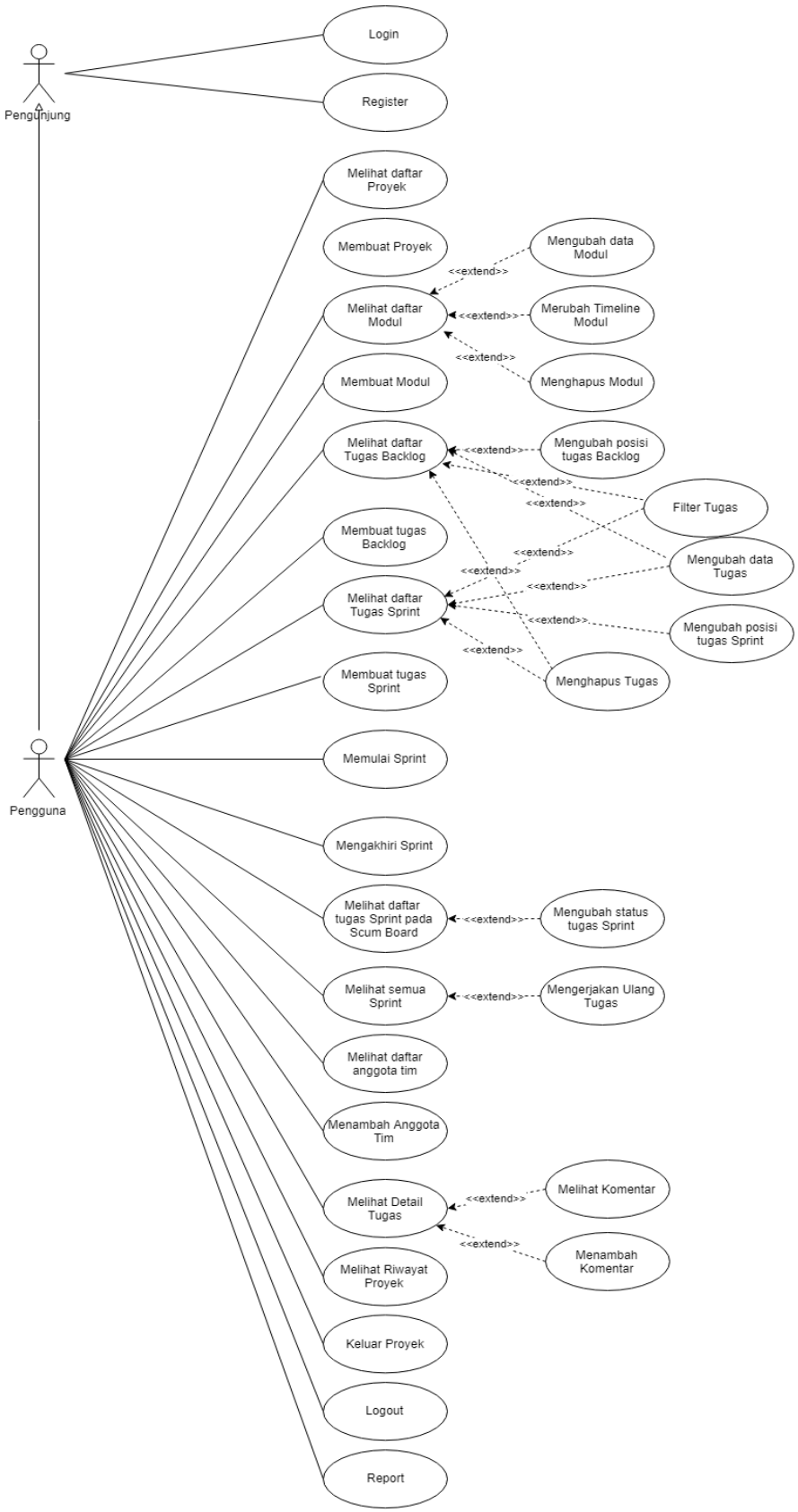
Berikut beberapa kebutuhan non fungsional dari aplikasi manajemen multi proyek menggunakan metode scrum. Untuk detailnya dapat dilihat pada tabel 4.5.

Tabel 4.5 Kebutuhan Non Fungsional

No.	Kode	Nama Fungsi	Deskripsi
1.	MMPS_NF_01	*Compatibility*	Aplikasi harus dapat diakses pada web browser meliputi Google Chrome, Microsoft Edge, Mozilla Firefox.

4.3 Use Case Diagram

Berikut merupakan use case diagram dari Aplikasi Manajemen Multi Proyek menggunakan metode Scrum :



Gambar 4.1 Use Case Diagram

4.4 Use Case Scenario

Use case scenario adalah penjelasan dan detail alur dari use case diagram yang telah didefinisikan. Pada use case scenario terdapat actor, objective, pre-condition, main flow, alternative flow, dan post-condition.

4.4.1 Register

Tabel 4.6 *Use Case Scenario* Register

Actor	Pengunjung
Objective	Pengunjung dapat melakukan registrasi dan menjadi Pengguna
Pre-condition	Pengunjung membuka halaman awal atau halaman login.
Main flow	1. Pengunjung membuka aplikasi dan mengunjungi menu Register. 2. Aplikasi menampilkan form registrasi. 3. Pengunjung mengisi form registrasi yang terdiri dari data nama, e-mail, password, dan konfirmasi password kemudian pengunjung klik tombol register. 4. Aplikasi menyimpan data pada tabel Pengguna. 5. Aplikasi kemudian mengarahkan pengunjung yang telah menjadi pengguna ke halaman beranda.
Alternative flows	• Jika kolom nama tidak berisi nilai, maka aplikasi akan memberikan pesan bahwa nama tidak boleh kosong. • Jika kolom email tidak berisi nilai, maka aplikasi akan memberikan pesan bahwa email tidak boleh kosong. • Jika kolom password tidak berisi nilai, maka aplikasi akan memberikan pesan bahwa password tidak boleh kosong. • Jika kolom konfirmasi password tidak berisi nilai, maka aplikasi akan memberikan pesan bahwa konfirmasi password tidak boleh kosong • Jika email tidak sesuai format, maka aplikasi akan memberikan pesan bahwa email tidak sesuai format. • Jika email yang dimasukkan telah digunakan, maka sistem akan memberikan pesan bahwa email tersebut tidak dapat digunakan. • Jika field password dan konfirmasi password tidak cocok, maka sistem akan memberikan pesan bahwa password tidak cocok.

	• Jika password kurang dari 6 karakter maka system akan memberikan pesan error.
Post-condition	Pengunjung berhasil mendaftar dan dapat login dengan akun yang didaftarkan.

4.4.2 Login

Tabel 4.7 *Use Case Scenario* Login

Actor	Pengunjung
Objective	Pengunjung dapat masuk kedalam Aplikasi agar dapat menggunakan fungsi - fungsi yang ada dalam aplikasi dan statusnya berubah menjadi Pengguna
Pre-condition	Pengunjung sudah melakukan registrasi.
Main flow	1. Pengunjung memasukan email dan password pada form login. 2. Pengunjung menekan tombol Login. 3. Aplikasi melakukan pengecekan apakah email dan password yang dimasukkan terdaftar dalam database. 4. Aplikasi mengarahkan pengguna ke halaman semua proyek.
Alternative flows	• Jika email dan password tidak terdaftar dalam database pengguna, maka aplikasi menampikan pesan bahwa email atau password salah.
Post-condition	Pengunjung mendapatkan hak akses sebagai pengguna.

4.4.3 Melihat daftar Proyek.

Tabel 4.8 *Use Case Scenario* Melihat daftar Proyek

Actor	Pengguna
Objective	Pengguna dapat melihat daftar proyek yang pengguna tersebut kerjakan.
Pre-condition	Aktor telah login ke aplikasi.
Main flow	1. Aplikasi akan secara otomatis mengarahkan pengguna ke halaman semua proyek dan menampilkan daftar Proyek.
Alternative flows	• -
Post-condition	Daftar semua proyek yang dikerjakan pengguna tersebut ditampilkan.

4.4.4 Membuat Proyek.

Tabel 4.9 *Use Case Scenario* Membuat Proyek

Actor	Pengguna
Objective	Pengguna dapat membuat proyek baru.
Pre-condition	Aktor telah login ke aplikasi.
Main flow	1. Aplikasi akan secara otomatis mengarahkan pengguna ke halaman semua proyek 2. Pengguna menekan tombol buat proyek. 3. Aplikasi menampilkan pop up untuk membuat proyek. 4. Pengguna memasukkan nama proyek dan deskripsi. 5. Pengguna menekan tombol Buat. 6. Aplikasi menyimpan data pada tabel Proyek. 7. Aplikasi mengarahkan pengguna ke halaman saat ini.
Alternative flows	• Jika pengguna tidak memasukkan nama proyek maka aplikasi akan memuat ulang halaman tersebut serta membawa pesan bahwa kolom nama proyek harus berisi nilai.
Post-condition	Proyek berhasil dibuat dan ditampilkan pada halaman daftar semua proyek.

4.4.5 Melihat daftar Modul.

Tabel 4.10 *Use Case Scenario* Melihat daftar Modul

Actor	Pengguna
Objective	Pengguna dapat melihat daftar modul dari proyek yang dibuka.
Pre-condition	Aktor mengunjungi halaman semua proyek.
Main flow	1. Aktor memilih salah satu proyek yang akan dibuka. 2. Aplikasi akan mengarahkan pengguna ke halaman road map yang memuat daftar semua modul.
Alternative flows	• -
Post-condition	Daftar semua modul yang dibuat berhasil ditampilkan.

4.4.6 Merubah Timeline Modul.

Tabel 4.11 *Use Case Scenario* Merubah Timeline Modul

Actor	Pengguna
Objective	Pengguna dapat mengatur timeline modul pada suatu proyek.
Pre-condition	Aktor telah melakukan fungsi melihat daftar modul.
Main flow	1. Pengguna mengatur tanggal mulai dan tanggal selesai pada suatu modul. 2. Pengguna menekan tombol Set. 3. Aplikasi menyimpan data pada tabel Riwayat dan melakukan *update* pada tabel Modul. 4. Aplikasi mengarahkan pengguna ke halaman Semua Modul.
Alternative flows	• Jika pengguna tidak memasukkan nilai pada kolom tanggal mulai dan tanggal selesai, maka aplikasi akan memuat ulang halaman tersebut serta membawa pesan bahwa tanggal mulai dan tanggal selesai harus berisi nilai. • Jika nilai pada kolom tanggal mulai lebih besar dari nilai pada kolom tanggal selesai, maka aplikasi akan memuat ulang halaman tersebut serta membawa pesan bahwa tanggal selesai harus bernilai setelah tanggal mulai.
Post-condition	Timeline modul yang dirubah akan berganti.

4.4.7 Membuat Modul.

Tabel 4.12 *Use Case Scenario* Membuat Modul

Actor	Pengguna
Objective	Pengguna dapat membuat modul pada suatu proyek.
Pre-condition	Aktor telah memilih salah satu proyek.
Main flow	1. Aplikasi akan secara otomatis mengarahkan pengguna ke halaman road map setelah pengguna memilih salah satu proyek. 2. Pengguna memasukkan nama modul. 3. Pengguna menekan tombol Add. 4. Aplikasi menyimpan data pada tabel Modul dan Riwayat. 5. Aplikasi mengarahkan pengguna ke halaman Semua Modul.

Alternative flows	• Jika pengguna tidak memasukkan nama modul maka aplikasi akan memuat ulang halaman tersebut serta membawa pesan bahwa kolom nama modul harus berisi nilai.
Post-condition	Modul berhasil dibuat dan ditampilkan pada halaman daftar semua modul.

4.4.8 Mengubah Data Modul.

Tabel 4.13 *Use Case Scenario* Mengubah Data Modul

Actor	Pengguna
Objective	Pengguna dapat merubah data modul.
Pre-condition	Aktor telah melakukan fungsi melihat daftar modul.
Main flow	1. Pengguna mengklik nama salah satu modul. 2. Aplikasi menampilkan modal pop up untuk merubah data modul. 3. Pengguna memasukkan nama baru untuk modul. 4. Pengguna menekan tombol edit. 5. Aplikasi menyimpan data pada tabel Riwayat dan melakukan *update* pada tabel Modul. 6. Aplikasi mengarahkan pengguna ke halaman Semua Modul.
Alternative flows	• Jika pengguna tidak memasukkan nilai pada kolom nama modul maka aplikasi akan memberi peringatan bahwa nama modul harus berisi nilai.
Post-condition	Data nama modul berhasil diganti.

4.4.9 Menghapus Modul.

Tabel 4.14 *Use Case Scenario* Menghapus Modul

Actor	Pengguna
Objective	Pengguna dapat menghapus suatu modul.
Pre-condition	Aktor telah melakukan fungsi melihat daftar modul.
Main flow	1. Pengguna mengklik nama salah satu modul 2. Aplikasi menampilkan modal pop up untuk merubah data modul. 3. Pengguna menekan tombol Hapus. 4. Aplikasi menghapus data Modul yang dipilih

	5. Aplikasi mengarahkan pengguna ke halaman Semua Modul.
Alternative flows	• -.
Post-condition	Modul berhasil dihapus.

4.4.10 Mengubah Data Tugas.

Tabel 4.15 *Use Case Scenario* Mengubah Data Tugas

Actor	Pengguna
Objective	Pengguna dapat merubah data tugas backlog atau sprint.
Pre-condition	Aktor telah memilih salah satu proyek.
Main flow	1. Pengguna mengunjungi halaman Back Log. 2. Aplikasi menampilkan daftar tugas sprint dan tugas back log. 3. Pengguna mengklik piihan edit pada tugas. 4. Aplikasi menampilkan modal pop up untuk merubah data tugas. 5. Pengguna memasukkan nama baru atau memilih modul baru untuk tugas tersebut. 6. Pengguna menekan tombol edit. 7. Aplikasi menyimpan data pada tabel Riwayat dan melakukan *update* pada tabel Tugas. 8. Aplikasi mengarahkan pengguna ke halaman Back Log
Alternative flows	• Jika pengguna tidak memasukkan nilai pada kolom nama tugas maka aplikasi akan memberi peringatan bahwa nama modul harus berisi nilai.
Post-condition	Data nama tugas dan modul dari tugas tersebut berhasil diganti.

4.4.11 Menghapus Tugas.

Tabel 4.16 *Use Case Scenario* Menghapus Tugas

Actor	Pengguna
Objective	Pengguna dapat menghapus suatu tugas.
Pre-condition	Aktor telah memilih salah satu proyek.
Main flow	1. Pengguna mengunjungi halaman Back Log.

	2. Aplikasi menampilkan daftar tugas sprint dan tugas back log. 3. Pengguna mengklik piihan edit pada tugas. 4. Aplikasi menampilkan modal pop up untuk merubah data tugas. 5. Pengguna menekan tombol Hapus. 6. Aplikasi menghapus data yang dipilih dari tabel Tugas. 7. Aplikasi mengarahkan pengguna ke halaman saat ini
Alternative flows	• -.
Post-condition	Tugas berhasil dihapus.

4.4.12 Melihat daftar Tugas Backlog.

Tabel 4.17 *Use Case Scenario* Melihat daftar Tugas Backlog

Actor	Pengguna
Objective	Pengguna dapat melihat semua tugas backlog pada suatu proyek.
Pre-condition	Aktor sudah login dan telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Back Log. 2. Aplikasi mengambil data dari tabel Tugas, Modul, dan Sprint. 3. Aplikasi menampilkan daftar tugas backlog
Alternative flows	• -.
Post-condition	Daftar tugas backlog dapat dilihat oleh pengguna.

4.4.13 Membuat tugas Backlog

Tabel 4.18 *Use Case Scenario* Membuat tugas Backlog

Actor	Pengguna
Objective	Pengguna dapat membuat tugas backlog.
Pre-condition	Aktor sudah login dan telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Back Log. 2. Aplikasi menampilkan halaman Back Log

	3. Pengguna memasukkan nama tugas backlog. 4. Pengguna menekan tombol Add. 5. Aplikasi menyimpan data pada tabel Tugas dan Riwayat. 6. Aplikasi mengarahkan pengguna ke halaman saat ini.
Alternative flows	• Jika pengguna tidak memasukkan nama tugas maka aplikasi akan memuat ulang halaman tersebut serta membawa pesan bahwa kolom tugas harus berisi nilai.
Post-condition	Daftar tugas backlog dapat dilihat oleh pengguna.

4.4.14 Mengubah posisi tugas Backlog.

Tabel 4.19 *Use Case Scenario* Mengubah posisi tugas Backlog

Actor	Pengguna
Objective	Pengguna dapat mengubah status tugas backlog menjadi tugas sprint.
Pre-condition	Aktor sudah login dan telah memilih salah satu proyek.
Main flow	Cara Pertama : 1. Aktor mengunjungi halaman Back Log. 2. Aplikasi menampilkan daftar tugas backlog. 3. Aktor menyeret nama tugas backlog dan meletakkannya pada bagian tugas sprint. 4. Aplikasi melakukan *update* pada tabel Tugas. 5. Aplikasi akan menempatkan tugas backlog ke bagian tugas sprint. Cara Kedua : 1. Aktor mengunjungi halaman Back Log. 2. Aplikasi menampilkan daftar tugas backlog. 3. Aktor menekan pilihan Move to Sprint. 4. Aplikasi melakukan *update* pada tabel Tugas 5. Aplikasi akan memuat ulang halaman Back Log.
Alternative flows	• Jika gagal mengubah posisi tugas backlog, maka akan muncul alert.
Post-condition	Tugas backlog berubah menjadi tugas Sprint.

4.4.15 Melihat daftar Tugas Sprint.

Tabel 4.20 *Use Case Scenario* Melihat daftar Tugas Sprint

Actor	Pengguna
Objective	Pengguna dapat melihat semua tugas sprint pada suatu proyek.
Pre-condition	Aktor sudah login dan telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Back Log. 2. Aplikasi mengambil data pada tabel Tugas, Sprint, Modul. 3. Aplikasi menampilkan daftar tugas sprint.
Alternative flows	• -.
Post-condition	Daftar tugas sprint dapat dilihat oleh pengguna.

4.4.16 Membuat tugas Sprint

Tabel 4.21 *Use Case Scenario* Membuat tugas Sprint

Actor	Pengguna
Objective	Pengguna dapat membuat tugas Sprint.
Pre-condition	Aktor sudah login dan telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Back Log. 2. Aplikasi menampilkan halaman Back Log 3. Pengguna memasukkan nama tugas sprint. 4. Pengguna menekan tombol Add. 5. Aplikasi menyimpan data pada tabel Tugas dan Riwayat. 6. Aplikasi mengarahkan pengguna ke halaman saat ini
Alternative flows	• Jika pengguna tidak memasukkan nama tugas maka aplikasi akan memuat ulang halaman tersebut serta membawa pesan bahwa kolom tugas harus berisi nilai.
Post-condition	Daftar tugas sprint dapat dilihat oleh pengguna.

4.4.17 Mengubah posisi tugas Sprint.

Tabel 4.22 *Use Case Scenario* Mengubah posisi tugas Sprint

Actor	Pengguna

Objective	Pengguna dapat mengubah status tugas sprint menjadi tugas backlog.
Pre-condition	Aktor sudah login dan telah memilih salah satu proyek.
Main flow	Cara Pertama : 1. Aktor mengunjungi halaman Back Log. 2. Aplikasi menampilkan daftar tugas sprint. 3. Aktor menyeret nama tugas sprint dan meletakkannya pada bagian tugas backlog. 4. Aplikasi melakukan update pada tabel Tugas. 5. Aplikasi akan menempatkan tugas sprint ke bagian tugas backlog Cara Kedua : 1. Aktor mengunjungi halaman Back Log. 2. Aplikasi menampilkan daftar tugas backlog. 3. Aktor menekan pilihan Move to Sprint. 4. Aplikasi melakukan update pada tabel Tugas. 5. Aplikasi akan memuat ulang halaman Back Log.
Alternative flows	• Jika gagal mengubah posisi tugas sprint, maka akan muncul alert.
Post-condition	Tugas sprint berubah menjadi tugas backlog.

4.4.18 Memulai Sprint.

Tabel 4.23 *Use Case Scenario* Memulai Sprint

Actor	Pengguna
Objective	Pengguna dapat memulai sprint dengan rentang waktu yang telah ditentukan.
Pre-condition	Aktor sudah login dan telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Back Log. 2. Aktor menekan tombol Mulai Sprint. 3. Aplikasi menampilkan modal *pop up* untuk memilih lama waktu untuk Sprint tersebut. 4. Aktor memilih lama waktu pengerjaan Sprint.

	5. Aktor menekan tombol OK 6. Aplikasi menyimpan data pada tabel Sprint dan Riwayat, serta melakukan perubahan data pada tabel Tugas. 7. Aplikasi memuat kembali halaman Back Log
Alternative flows	• -.
Post-condition	Sprint aktif berjalan selama waktu yang telah ditentukan dan tombol mulai sprint berganti dengan informasi bahwa Sprint sedang berjalan.

4.4.19 Mengakhiri Sprint.

Tabel 4.24 *Use Case Scenario* Mengakhiri Sprint

Actor	Pengguna
Objective	Pengguna dapat menghentikan sprint yang sedang berjalan.
Pre-condition	Aktor telah memilih salah satu proyek dan terdapat sprint yang sedang berjalan.
Main flow	1. Aktor mengunjungi halaman Board. 2. Aktor menekan tombol Selesai. 3. Aplikasi menampilkan modal pop up konfirmasi untuk mengakhiri sprint. 4. Aktor menekan tombol OK. 5. Aplikasi memuat kembali halaman Board.
Alternative flows	• -.
Post-condition	Sprint telah selesai berjalan dan aktor dapat memulai sprint baru.

4.4.20 Melihat daftar tugas Sprint pada Scum Board.

Tabel 4.25 *Use Case Scenario* Melihat daftar tugas Sprint pada Scrum Board

Actor	Pengguna
Objective	Pengguna dapat melihat semua tugas sprint yang sedang berjalan dan dikategorikan tiap status pada suatu proyek.
Pre-condition	Aktor telah memilih salah satu proyek serta terdapat Sprint yang sedang berjalan.
Main flow	1. Aktor mengunjungi halaman Board. 2. Aplikasi mengambil data pada Tabel Tugas dan Sprint.

	3. Aplikasi menampilkan daftar tugas sprint berdasarkan statusnya.
Alternative flows	• -.
Post-condition	Daftar tugas sprint dapat dilihat oleh pengguna.

4.4.21 Mengubah status tugas Sprint.

Tabel 4.26 *Use Case Scenario* Mengubah status tugas Sprint

Actor	Pengguna
Objective	Pengguna dapat mengubah status tugas sprint yang sedang berjalan.
Pre-condition	Aktor sudah login dan telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Board. 2. Aplikasi menampilkan daftar tugas sprint berdasarkan statusnya. 3. Aktor menyeret nama tugas sprint dan meletakkannya pada bagian status lainnya. 4. Aplikasi menyimpan data pada tabel Riwayat dan melakukan *update* pada tabel Tugas 5. Aplikasi akan menempatkan tugas sprint ke bagian status yang ditentukan.
Alternative flows	• Jika gagal mengubah posisi tugas, maka akan muncul alert.
Post-condition	Tugas sprint berubah statusnya.

4.4.22 Melihat semua Sprint.

Tabel 4.27 *Use Case Scenario* Melihat semua Sprint

Actor	Pengguna
Objective	Pengguna dapat melihat semua sprint beserta tugas sprint yang telah atau sedang berjalan pada suatu proyek.
Pre-condition	Aktor telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman All Sprint.

	2. Aplikasi menampilkan daftar Spint serta tugas tugas dari tiap Sprint.
Alternative flows	• -.
Post-condition	Daftar semua sprint dapat dilihat oleh pengguna.

4.4.23 Melihat daftar anggota tim.

Tabel 4.28 *Use Case Scenario* Melihat daftar anggota tim

Actor	Pengguna
Objective	Pengguna dapat mengetahui nama manajer proyek dan para anggota yang terlibat dalam pengerjaan proyek tersebut.
Pre-condition	Aktor telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Project Setting. 2. Aplikasi mengambil data pada tabel Proyek dan Pengguna. 3. Aplikasi menampilkan daftar anggota Tim dan manajer proyek.
Alternative flows	• -.
Post-condition	Informasi tentang manajer proyek dan anggota tim dapat dilihat oleh pengguna.

4.4.24 Menambah Anggota Tim

Tabel 4.29 *Use Case Scenario* Menambah Anggota Tim

Actor	Pengguna
Objective	Pengguna dapat menambahkan pengguna lain untuk berkontribusi terhadap suatu proyek.
Pre-condition	Aktor telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Project Setting. 2. Aplikasi menampilkan halaman Project Setting. 3. Pengguna memasukkan email pengguna yang akan didaftarkan sebagai anggota tim baru. 4. Pengguna menekan tombol Add. 5. Aplikasi menyimpan data pada tabel Team.

	6. Aplikasi mengarahkan pengguna ke halaman Project Setting
Alternative flows	• Jika email yang dimasukkan tidak terdaftar dalam database maka aplikasi akan memberi informasi bahwa email tidak terdaftar. • Jika pengguna tidak memasukkan email maka aplikasi akan memberi informasi bahwa kolom email harus berisi nilai • Jika email yang dimasukkan telah terdaftar pada proyek, maka aplikasi akan memberi informasi bahwa email telah terdaftar.
Post-condition	Pengguna lain behasil ditambahkan dan pengguna tersebut dapat mengakses proyek ini.

4.4.25 Melihat Detail Tugas.

Tabel 4.30 *Use Case Scenario* Melihat Detail Tugas

Actor	Pengguna
Objective	Pengguna dapat mengetahui informasi dari tugas yang dipilih.
Pre-condition	Aktor telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Backlog. 2. Aplikasi menampilkan daftar tugas baik di bagian sprint maupun backlog. 3. Pengguna menekan nama salah satu tugas. 4. Aplikasi mengambil data pada tabel Tugas dan Komentar 5. Aplikasi berpindah ke halaman Detail tugas.
Alternative flows	• -.
Post-condition	Informasi tentang suatu tugas dapat dilihat oleh pengguna.

4.4.26 Melihat Komentar.

Tabel 4.31 *Use Case Scenario* Melihat Komentar

Actor	Pengguna
Objective	Pengguna dapat melihat semua komentar dari tugas yang dipilih.
Pre-condition	Aktor telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Backlog.

	2. Aplikasi menampilkan daftar tugas baik di bagian sprint maupun backlog. 3. Pengguna menekan nama salah satu tugas. 4. Aplikasi mengambil data pada tabel Tugas dan Komentar. 5. Aplikasi berpindah ke halaman Detail tugas yang juga menampilkan komentar dari suatu tugas.
Alternative flows	• -.
Post-condition	Semua komentar dari suatu tugas dapat dilihat oleh pengguna.

4.4.27 Menambah Komentar

Tabel 4.32 *Use Case Scenario* Menambah Komentar

Actor	Pengguna
Objective	Pengguna dapat memberikan komentar pada suatu tugas.
Pre-condition	Aktor telah melakukan use case Melihat Detail Tugas.
Main flow	1. Pengguna memasukkan komentar pada form komentar. 2. Pengguna menekan tombol Komentar. 3. Aplikasi menyimpan data pada tabel Komentar dan Riwayat. 4. Aplikasi memuat ulang halaman Detail Tugas
Alternative flows	• Jika komentar tidak berisi teks apapun, maka aplikasi akan memberi informasi bahwa kolom komentar tidak boleh kosong.
Post-condition	Komentar behasil ditambahkan dan dapat dilihat oleh pengguna lain juga.

4.4.28 Melihat Riwayat Proyek.

Tabel 4.33 *Use Case Scenario* Melihat Riwayat Proyek

Actor	Pengguna
Objective	Pengguna dapat melihat semua aktivitas manajemen pada proyek yang dipilih.
Pre-condition	Aktor telah memilih salah satu proyek.
Main flow	1. Aktor mengunjungi halaman Riwayat Manajemen.

	2. Aplikasi menampilkan daftar riwayat aktivitas yang telah dilakukan oleh tim.
Alternative flows	• -.
Post-condition	Semua riwayat aktivitas dari suatu proyek dapat dilihat oleh pengguna.

4.4.29 Filter Tugas

Tabel 4. 34 *Use Case Scenario* Filter Tugas

Actor	Pengguna
Objective	Pengguna dapat melakukan filter terhadap tugas berdasarkan nama modul.
Pre-condition	Aktor telah mengunjungi halaman Back Log.
Main flow	1. Pengguna memilih salah satu nama modul pada dropdown pilihan filter. 2. Pengguna menekan tombol filter.
Alternative flows	• -.
Post-condition	Pengguna dapat melihat tugas tugas yang termasuk kedalam salah satu modul yang dipilih.

4.4.30 Keluar Proyek

Tabel 4.35 *Use Case Scenario* Keluar Proyek

Actor	Pengguna
Objective	Pengguna dapat keluar dari proyek yang sedang dibuka untuk memilih proyek lain.
Pre-condition	Aktor telah memilih salah satu proyek.
Main flow	1. Pengguna menekan menu Keluar proyek. 2. Aplikasi akan menghapus session proyek dan mengarahkan pengguna ke halaman daftar semua proyek
Alternative flows	• -.
Post-condition	Pengguna behasil keluar dari manajemen proyek dan dapat memilih proyek lainnya untuk dibuka.

4.4.31 Logout

Tabel 4.36 *Use Case Scenario* Logout

Actor	Pengguna
Objective	Pengguna dapat keluar dari Aplikasi dan menjadi pengunjung.
Pre-condition	Aktor telah login ke dalam aplikasi.
Main flow	1. Pengguna menekan menu Keluar. 2. Aplikasi akan mengarahkan pengguna ke halaman Login.
Alternative flows	• -.
Post-condition	Pengguna behasil keluar dari Aplikasi dan menjadi Pengunjung.

4.4.32 Report

Tabel 4.37 *Use Case Scenario* Report

Actor	Pengguna
Objective	Pengguna dapat melihat nama modul dan tugas tugas didalamya serta status tugas.
Pre-condition	Aktor telah login ke dalam aplikasi.
Main flow	1. Pengguna menekan menu Report. 2. Aplikasi mengambil data pada tabel Tugas dan Modul. 3. Aplikasi akan mengarahkan pengguna ke halaman Report.
Alternative flows	• -.
Post-condition	Daftar report dapat dilihat oleh pengguna.

4.4.33 Mengerjakan Ulang Tugas

Tabel 4.38 *Use Case Scenario* Mengerjakan Ulang Tugas

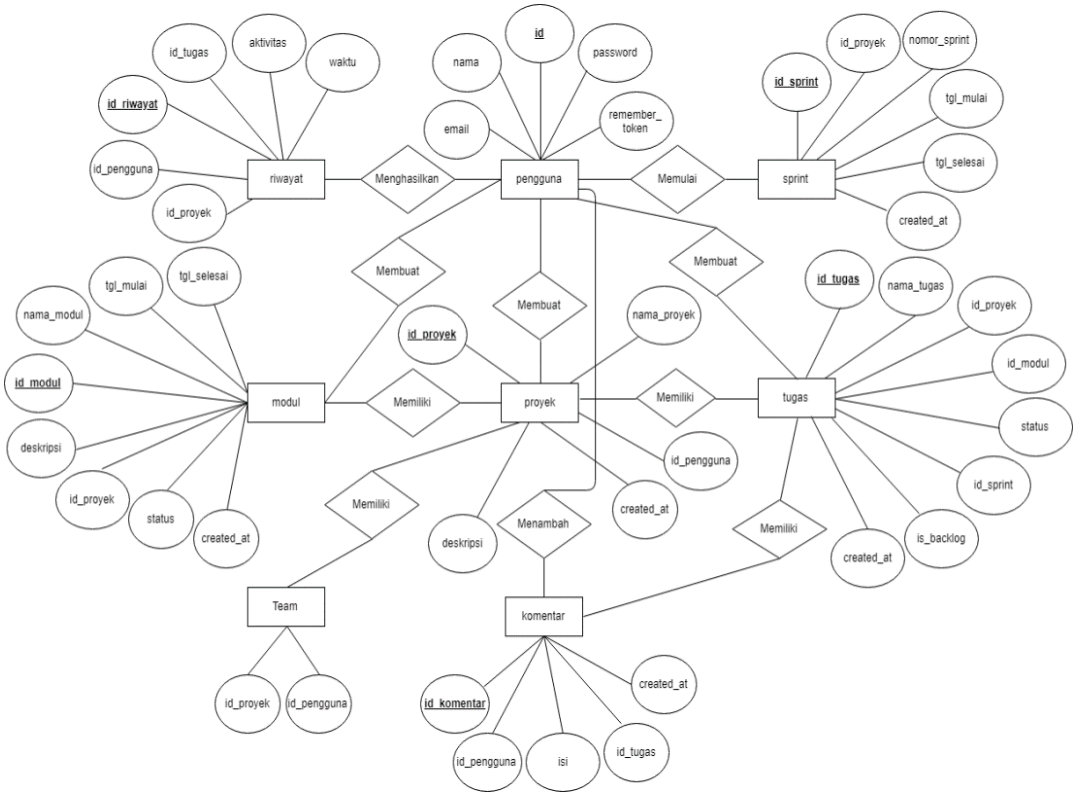
Actor	Pengguna
Objective	Pengguna dapat mengerjakan kembali tugas yang berstatus “Done” pada sprint yang telah selesai.
Pre-condition	Aktor telah login ke dalam aplikasi.

Main flow	1. Aktor mengunjungi halaman All Sprint. 2. Aplikasi menampilkan daftar semua sprint beserta tugas tugas. 3. Aktor menekan pilihan Move to Current Sprint. 4. Aplikasi menyimpan data pada tabel Riwayat dan melakukan update pada tabel Tugas. 5. Aplikasi akan memuat ulang halaman Semua sprint.
Alternative flows	• -.
Post-condition	Tugas yang dipilih pada sprint yang telah selesai akan berubah status dan masuk ke dalam sprint yang sedang berjalan saat itu.

4.5 Analisis Data

Analisis data berguna dalam proses identifikasi entitas yang diperlukan pada aplikasi manajemen *multi* proyek menggunakan metode *scrum*. Analisis data akan dimodelkan pada bentuk *Entity Relationship Diagram*.

Gambar 4.2 merupakan ERD dari aplikasi manajemen *multi* proyek menggunakan metode *scrum* yang didalamnya menunjukkan hubungan hubungan antar entitas dengan entitas lain. Dalam ERD tersebut terdapat delapan entitas meliputi pengguna, proyek, modul, tugas, komentar, riwayat, sprint, dan team. Semua entitas tersebut memiliki relasi antar entitas. Pada gambar berikut dijelaskan tentang relasi yang terdapat pada aplikasi manajemen *multi* proyek menggunakan metode *scrum* dan juga entitasnya.



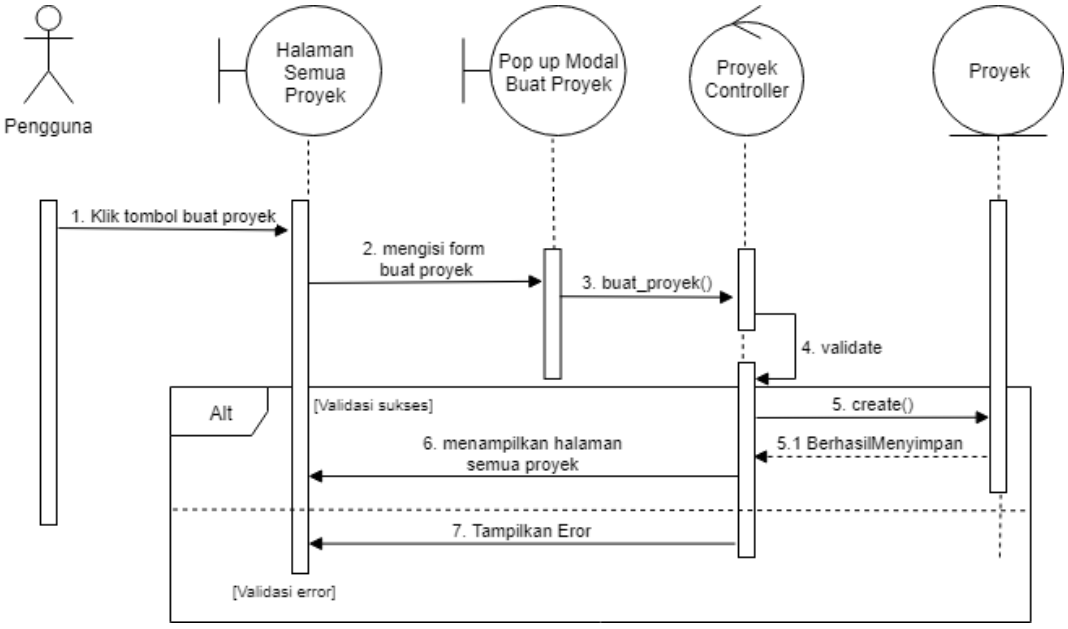
Gambar 4.2 ERD atau Entity Relationship Diagram

BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan

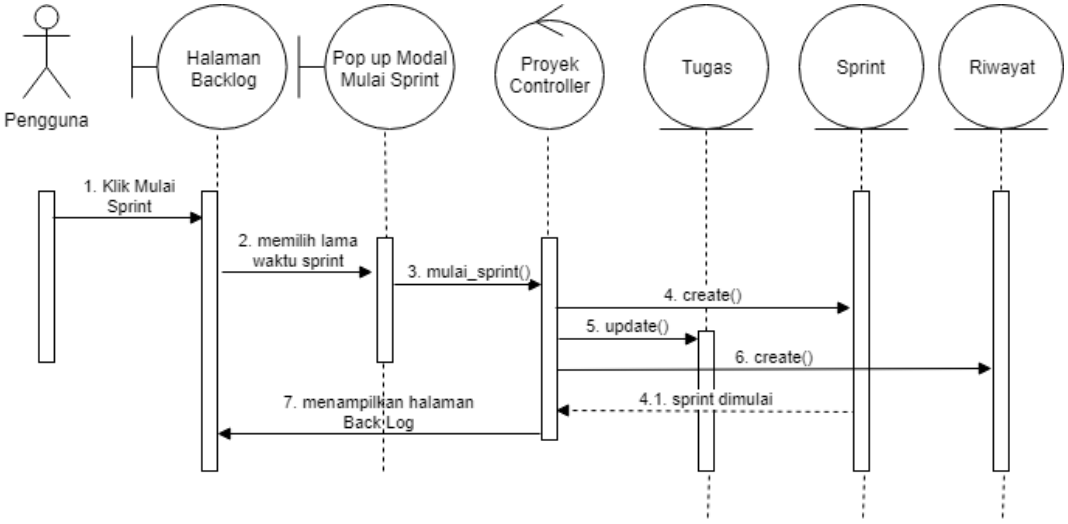
Pada tahap perancangan akan dipaparkan tentang perancangan yang dilakukan dari hasil analisis untuk sebelum dilanjutkan ke proses implementasi. Perancangan terdiri dari perancangan *sequence diagram*, data, *class diagram*, perancangan algoritme, dan perancangan antarmuka.

5.1.1 Pemodelan *Sequence Diagram*



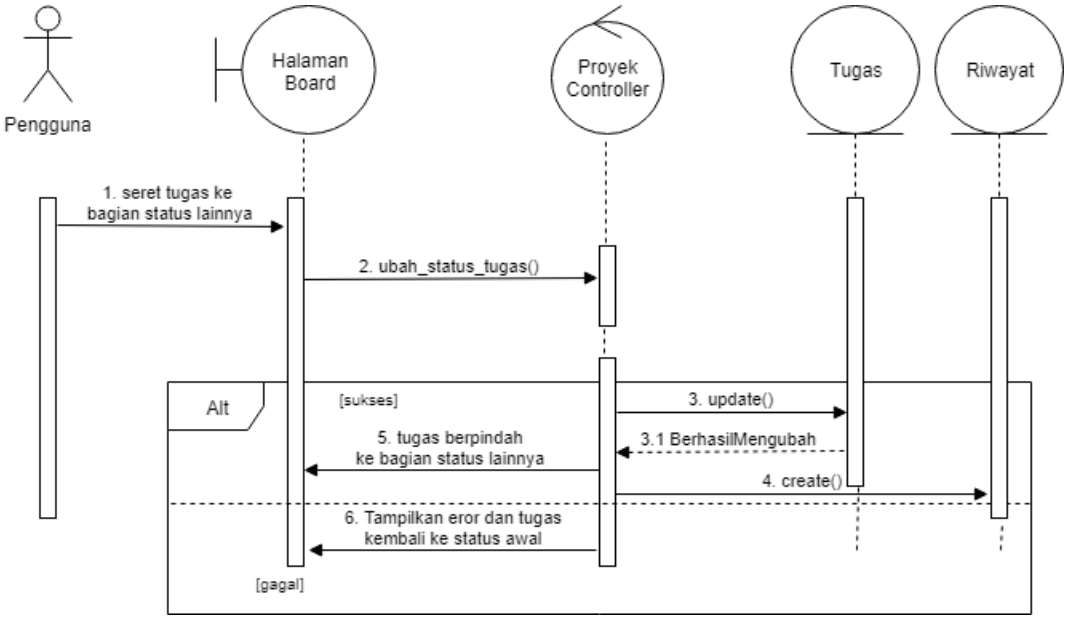
Gambar 5.1 *Sequence Diagram* Membuat Proyek

Gambar 5.1 merupakan *sequence diagram* dari membuat proyek. Untuk membuat proyek, pengguna menekan tombol but proyek dan system akan menampilkan pop up untuk mengisi form buat proyek, lalu memasukkan nama proyek pada *boundary* Halaman Semua Proyek lalu menekan tombol Buat. Aplikasi akan memanggil method buat_proyek() pada controller ProyekController. Pada method buat_proyek(), dilakukan validasi terhadap masukkan dari pengguna. Jika masukkan sesuai kriteria maka akan memanggil method create() untuk membuat data record baru pada database atau entity Proyek. Setelah berhasil menyimpan, maka aplikasi akan kembali ke halaman semua proyek. Namun, jika proses validasi pada *controller* ProyekController gagal, maka Aplikasi akan menampilkan pesan error pada halaman semua proyek.



Gambar 5.2 *Sequence Diagram* Memulai Sprint

Gambar 5.2 merupakan *sequence diagram* dari memulai sprint. Untuk memulai sprint pengguna harus menekan tombol mulai sprint pada boundary Halaman Back Log, kemudian muncul *pop up* modal untuk memasukkan lama waktu berjalannya sprint tersebut. Aplikasi akan memanggil method mulai_sprint() pada controller ProyekController. Pada method buat_proyek(), dilakukan juga pemanggilan untuk mengakses database, yaitu method create untuk membuat data *record* baru pada *database* atau entity Sprint dan Riwayat, serta method update untuk mengubah data pada database atau entity Tugas. Setelah berhasil menyimpan data baru sprint dan melakukan *update*, maka aplikasi akan kembali ke halaman Backlog *boundary*.

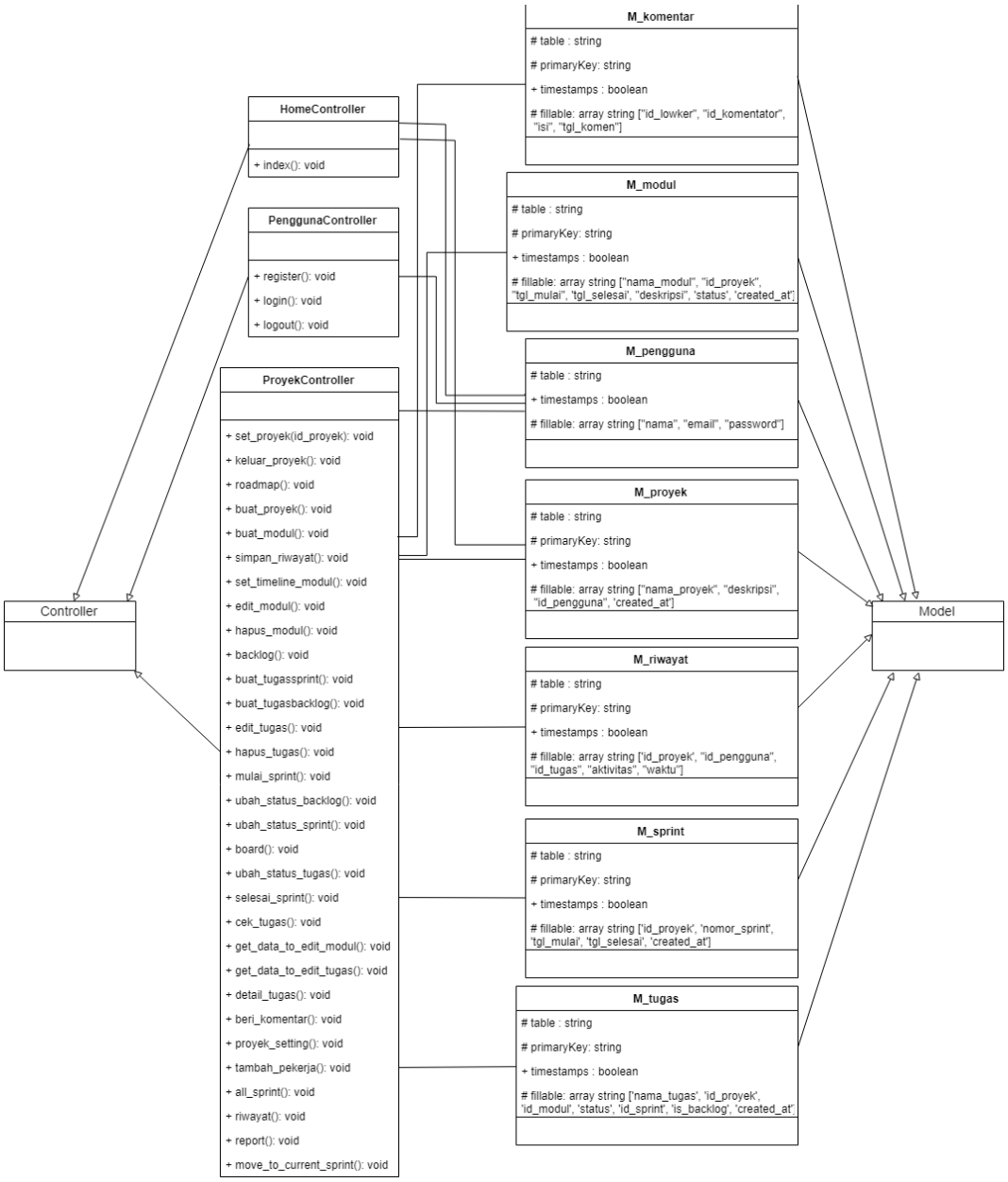


Gambar 5.3 *Sequence Diagram* Mengubah Status Tugas Sprint

Gambar 5.3 merupakan *sequence diagram* dari mengubah status tugas sprint. Untuk mengubah status tugas sprint pengguna harus menyeret nama tugas sprint dari bagian status tertentu kedalam bagian status lainnya pada *boundary* Halaman

Board. Aplikasi akan memanggil method ubah_status_tugas() pada *controller* ProyekController. Pada method ubah_status_tugas (), dilakukan juga pemanggilan untuk mengakses *database*, yaitu method update untuk mengubah data pada database atau entity Tugas dan memanggil *method create* untuk mentimpan data baru pada entity Riwayat. Setelah berhasil menyimpan perubahan, maka tugas tersebut telah berada pada bagain status yang baru di halaman Backlog boundary. Jika terdapat kesalahan pada proses pemindahan, yaitu pada saat menyeret nama tugas, maka tugas tersebut akan kembali ke status semula.

5.1.2 Pemodelan Class Diagram



Gambar 5.4 Class Diagram

Gambar 5.4 diatas merupakan class diagram dari aplikasi manajemen multi proyek menggunakan metode scrum. Aplikasi ini menggunakan framework

Laravel, sehingga untuk class HomeController, PenggunaController dan ProyekController melakukan inheritance terhadap class Controller. Sedangkan untuk class M_komentar, M_modul, M_pengguna, M_proyek, M_riwayat, M_sprint, dan M_tugas melakukan inheritance terhadap Model. Class Controller dan Model tersebut merupakan class bawaan dari framework Laravel untuk memudahkan dalam pengembangan Aplikasi.

5.1.3 Perancangan Algoritme

Tabel 5.1 *Pseudocode* Membuat Proyek

No.	Algotitma
1.	Start method buat_proyek.
2.	If nilai form nama_proyek == null
3.	Kembali kehalaman semua proyek
4.	End if
5.	Simpan proyek baru dalam database
6.	Kembali kehalaman semua proyek
7.	End

Tabel 5.1 merupakan *pseudocode* membuat proyek. *Pseudocode* ini diawali dengan proses seleksi apakah nilai dari form nama_proyek bernilai null. Jika null maka akan kembali kehalaman semua proyek. Namun jika tidak maka menyimpan data proyek baru dalam *database* kemudian kembali ke halaman semua proyek.

Tabel 5.2 *Pseudocode* Memulai Sprint

No.	Algotitma
1.	Start method mulai_sprint.
2.	obj_nomor_sprint = data sprint yang terakhir dari suatu proyek
3.	If total obj_nomor_sprint == 0
4.	nomor_sprint = 1
5.	else
6.	nomor_sprint = obj_nomor_sprint[0]->nomor_sprint
7.	End if
8.	today = tanggal hari ini
9.	tgl_selesai = tanggal hari ini + lama waktu sprint
10.	Simpan sprint baru dalam database.
11.	Simpan riwayat baru dalam database.
12.	new_sprint = data sprint yang terakhir dari suatu proyek

Tabel 5.2 *Pseudocode* Memulai Sprint

No.	Algotitma
13.	update tugas, rubah data id sprint menjadi new_sprint[0]->id_sprint, dimana jika nilai id_sprint = null dan is_backlog = 0.
14.	Kembali kehalaman backlog.
15.	End.

Tabel 5.2 merupakan *pseudocode* memulai sprint. *Pseudocode* ini diawali dengan mengakses data sprint yang terkahir dari suatu proyek. Kemudian dilakukan pengecekan untuk menentukan nomor sprint. Selanjutnya mengambil nilai tanggal hari ini dan tanggal selesai serta menyimpan sprint dan riwayat baru dalam *database*. Setelah itu dilakukan perubahan data terhadap tugas dengan kriteria tertentu dan proses akan kembali ke halaman backlog.

Tabel 5.3 *Pseudocode* Mengubah Status Tugas Sprint

No.	Algotitma
1.	Start method ubah_status_tugas.
2.	id_tugas = nilai id_tugas dari form ketika dilakukan perubahan status
3.	tugas = data tugas dari id_tugas
4.	update tugas, rubah data status menjadi nilai status baru dari proses perubahan, dimana jika nilai id_tugas =. id_tugas.
5.	Simpan riwayat baru dalam database.
6.	Kembali kehalaman board
7.	End

Tabel 5.3 merupakan *pseudocode* mengubah status tugas sprint. *Pseudocode* ini diawali dengan mendapatkan id tugas dari hasil submit form. Kemudian data tugas dengan id tersebut dirubah nilai statusnya, disertai penyimpanan data baru untuk table riwayat dan proses kembali ke halaman board.

Tabel 5.4 *Pseudocode* Menampilkan Halaman Backlog

No.	Algotitma
1.	Start method backlog.
2.	nomor_sprint = data sprint yang terakhir dari suatu proyek
3.	where1 = array untuk kriteria pengambilan data tugas.
4.	where2 = array untuk kriteria pengambilan data tugas.
5.	where3 = array untuk kriteria pengambilan data tugas.
6.	where_backlog = array untuk kriteria pengambilan data tugas.

Tabel 5.4 *Pseudocode* Menampilkan Halaman Backlog

No.	Algotitma
7.	id_modul = 0;
8.	if atribut id_modul dari form berisi suatu nilai
9	if atribut id_modul dari form bernilai != “all”
10.	where1['id_modul'] = nilai atribut id_modul dari form submit
11.	where2['id_modul'] = nilai atribut id_modul dari form submit
12.	where3['id_modul'] = nilai atribut id_modul dari form submit
13.	where_backlog['id_modul'] = nilai atribut id_modul dari form submit
14.	id_modul = nilai atribut id_modul dari form submit
15.	endif
16.	endif
17.	if total data variabel nomor_sprint = 0
18.	sprint = data tugas dengan kriteria variabel where1
19.	else
20.	if nilai tgl_selesai dari data ke 0 variabel nomor_sprint = waktu saat ini
21.	sprint = data tugas dengan kriteria variabel where2
22.	else
23.	sprint = data tugas dengan kriteria variabel where3
24.	endif
25.	endif
26.	modul = mengambil data modul pada proyek yang dibuka
27.	backlog = data tugas dengan kriteria variabel where_backlog
28.	Kembali kehalaman backlog dengan membawa nilai variabel nomor_sprint, sprint, backlog, modul, dan id_modul.
29.	End

Tabel 5.4 merupakan *pseudocode* menampilkan halaman backlog. *Pseudocode* ini berisi pengambilan data tugas maupun modul untuk dibawa dan ditampilkan pada halaman backlog. Didalamnya terdapat beberapa kasus dalam pengambilan data tersebut, khususnya untuk tugas sprint. Selain itu terdapat beberapa pseudocode untuk menangani kasus dimana pengguna mengakses halaman backlog menggunakan fitur filter yang terdapat pada halaman backlog. Dimana filter tersebut akan menampilkan tugas yang termasuk pada salah satu modul saja.

Tabel 5.5 *Pseudocode* Mengubah Posisi tugas Backlog

No.	Algotitma
1.	Start method ubah_status_backlog.
2.	last_sprint = data sprint yang terakhir dari suatu proyek
3.	if total data variabel last_sprint = 0
4.	Update data tugas, ganti is_backlog = 0, pada data dengan id_tugas = Nilai form submit field id_tugas.
5.	else
6.	if nilai tgl_selesai dari data ke 0 variabel last_sprint = waktu saat ini
7.	Update data tugas is_backlog = 0 dan id_sprint = nilai last_sprint, pada data dengan id_tugas = Nilai form submit field id_tugas.
8.	Else
9	Update data tugas, ganti is_backlog = 0, pada data dengan id_tugas = Nilai form submit field id_tugas.
10.	endif
11.	endif
12.	Kembali kehalaman backlog
13.	End

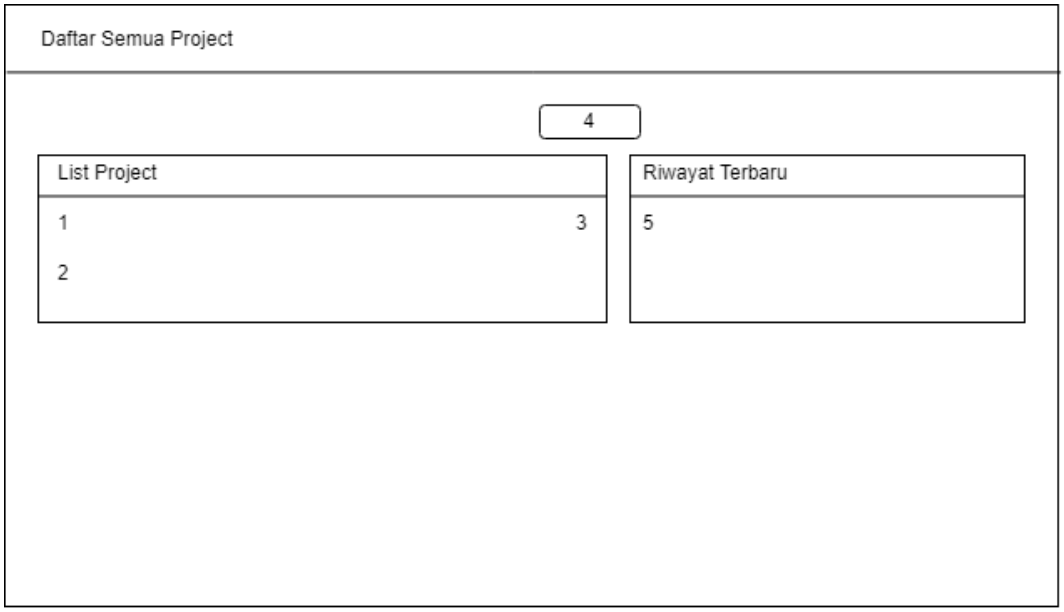
Tabel 5.5 merupakan *pseudocode* mengubah posisi tugas backlog. *Pseudocode* ini berisi pengambilan data sprint terakhir dari suatu proyek, untuk nantinya dilakukan pengecekan terhadap data sprint tersebut. Terdapat beberapa kasus perubahan data tugas yang mana tergantung dari data sprint terakhr dari proyek yang dibuka.

5.1.4 Perancangan Antarmuka

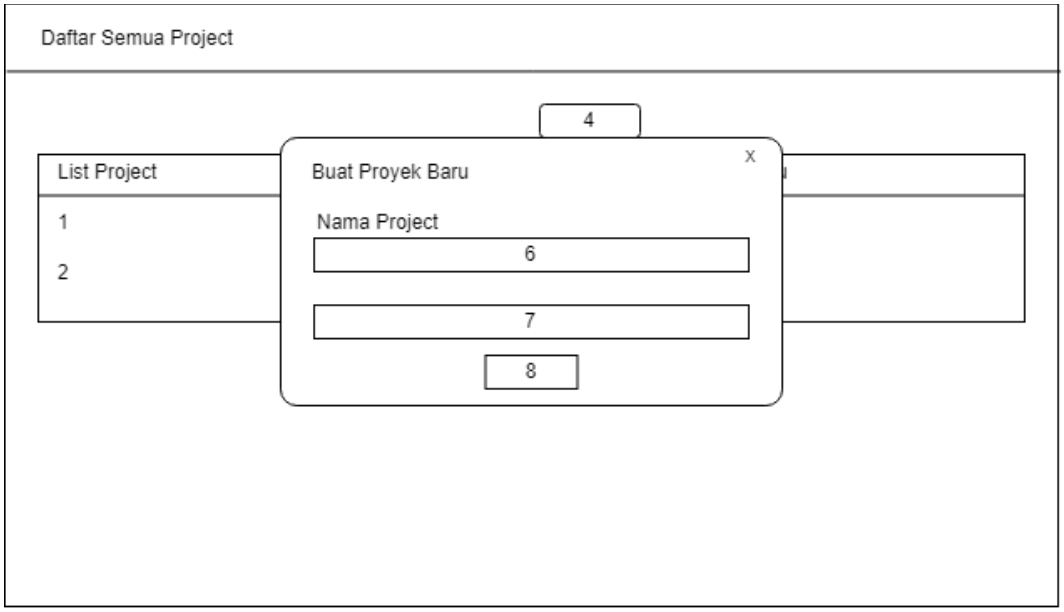
Perancangan antarmuka memuat desain tampilan dari setiap halaman yang ada pada Aplikasi Manajemen Multi Proyek menggunakan Metode *Scrum*.

5.1.4.1 Halaman Semua Proyek

Gambar 5.5 merupakan perancangan dari halaman semua proyek. Terdapat beberapa tambahan gambar untuk menampilkan antarmuka dari suatu pop up. Gambar 5.6 akan muncul jika objek nomor 4 ditekan. Penjelasan setiap objek dari perancangan antarmuka halaman semua proyek dijelaskan pada Tabel 5.6.



Gambar 5.5 Perancangan Antarmuka Halaman Semua Proyek



Gambar 5.6 Perancangan Antarmuka Halaman Untuk Buat Proyek

Tabel 5.6 Penjelasan Perancangan Antarmuka Halaman Semua Proyek

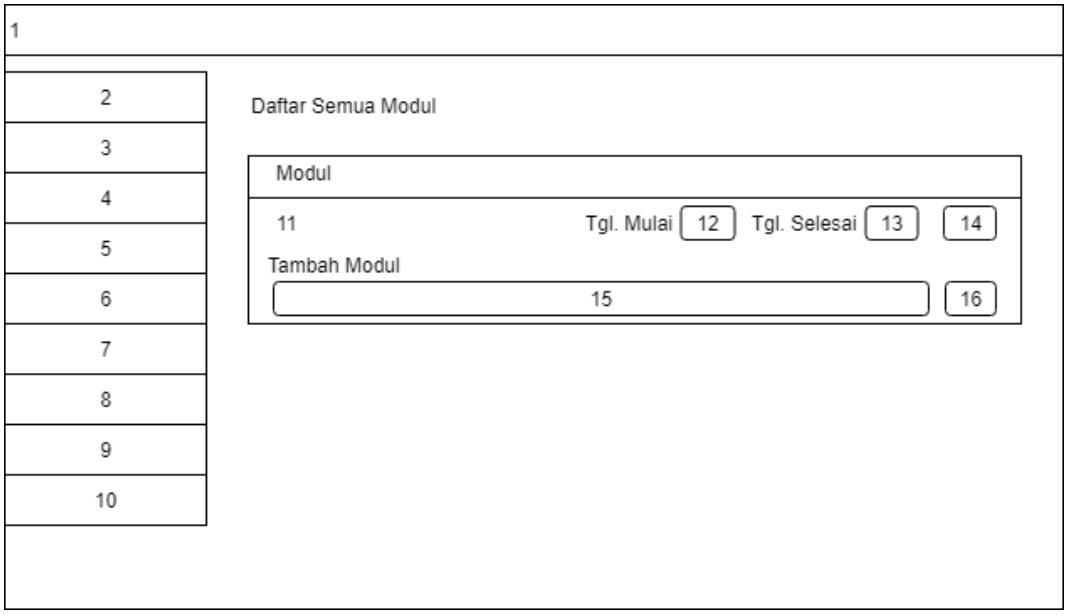
No.	Nama Objek	Tipe	Keterangan
1	Nama Proyek	Text	Menampilkan nama Proyek.

Tabel 5.6 Penjelasan Perancangan Antarmuka Halaman Semua Proyek

No.	Nama Objek	Tipe	Keterangan
2	Informasi Proyek	Text	Menampilkan nama Proyek Manager, deskripsi dan tanggal proyek dibuat.
3	Progress	Text	Menampilkan total progress dari suatu proyek dan dalam bentuk persentase.
4	Buat Proyek	Tombol	Tombol untuk membuka pop up pembuatan proyek.
5	Riwayat	Text	Menampilkan riwayat pengelolan proyek.
6	Nama Proyek	Text field	Field untuk mengisi Nama Proyek.
7	Deskripsi	Text field	Field untuk mengisi Deskripsi dari Proyek.
8	Buat Proyek	Tombol	Tombol untuk membuat proyek baru.

5.1.4.2 Halaman Semua Modul

Gambar 5.7 merupakan perancangan dari halaman semua modul. Penjelasan setiap objek dari perancangan antarmuka halaman semua modul dijelaskan pada Tabel 5.7.



Gambar 5.7 Perancangan Antarmuka Halaman Semua Modul

Tabel 5.7 Penjelasan Perancangan Antarmuka Halaman Semua Modul

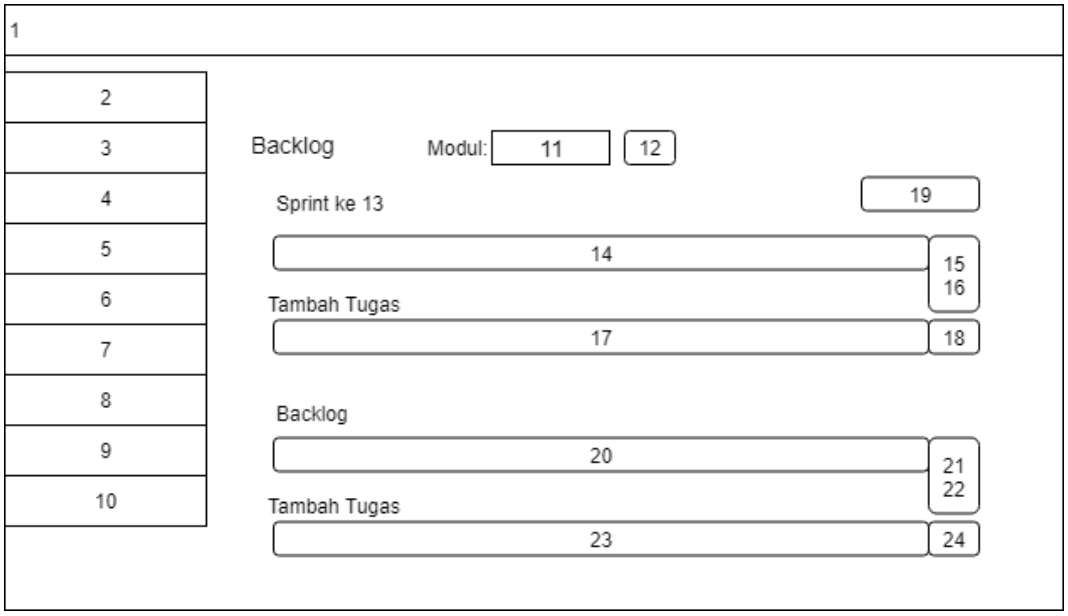
No.	Nama Objek	Tipe	Keterangan
1	Nama Proyek	Text	Menampilkan nama Proyek.
2	Road Map	Navigasi	Navigasi untuk mengakses halaman roadmap

Tabel 5.7 Penjelasan Perancangan Antarmuka Halaman Semua Modul

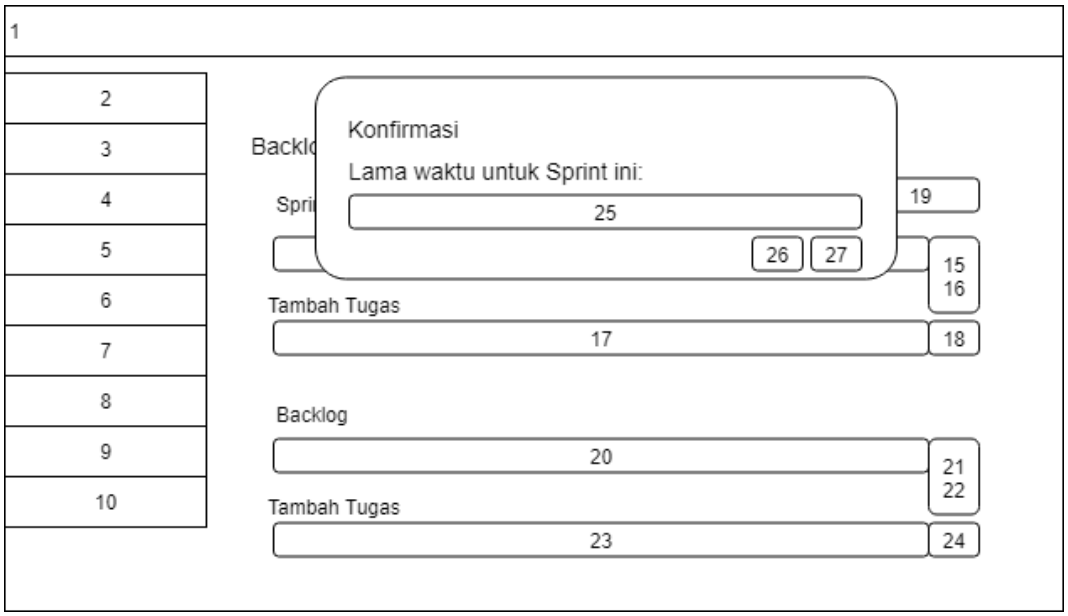
No.	Nama Objek	Tipe	Keterangan
3	Back Log	Navigasi	Navigasi mengakses halaman backlog
4	Board	Navigasi	Navigasi mengakses halaman board
5	All Sprint	Navigasi	Navigasi mengakses halaman semua sprint
6	Project Setting	Navigasi	Navigasi mengakses halaman project setting
7	Riwayat Aktivitas	Navigasi	Navigasi mengakses halaman riwayat manajemen proyek
8	Report	Navigasi	Navigasi mengakses halaman report
9	Keluar Proyek	Navigasi	Navigasi untuk keluar dari halaman manajemen proyek
10	Keluar	Navigasi	Navigasi untuk keluar dari aplikasi
11	Nama Modul	Text	Menampilkan nama modul.
12	Tanggal mulai	Date	Field untuk mengisi tanggal mulai.
13	Tanggal selesai	Date	Field untuk mengisi tanggal selesai.
14	Set	Tombol	Tombol untuk mengatur timeline modul
15	Nama Modul	Text field	Field untuk mengisi nama modul.
16	Add	Tombol	Tombol untuk membuat modul baru

5.1.4.3 Halaman Backlog

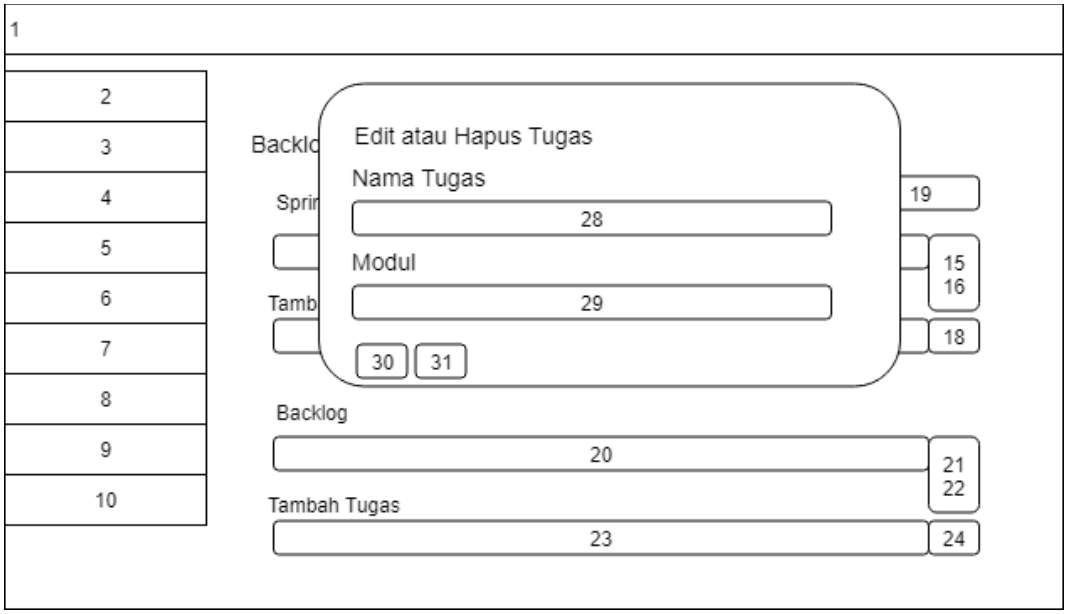
Gambar 5.8 merupakan perancangan dari halaman semua backlog. Terdapat beberapa tambahan gambar untuk menampilkan antarmuka dari suatu pop up. Gambar 5.9 akan muncul jika objek nomor 19 ditekan. Gambar 5.10 akan muncul jika objek dari nomor 16 atau 22 ditekan. Penjelasan setiap objek dari perancangan antarmuka halaman backlog dijelaskan pada Tabel 5.8.



Gambar 5.8 Perancangan Antarmuka Halaman Backlog



Gambar 5. 9 Perancangan Antarmuka Memulai Sprint



Gambar 5.10 Perancangan Antarmuka Edit Tugas

Tabel 5.8 Penjelasan Perancangan Antarmuka Halaman Backlog

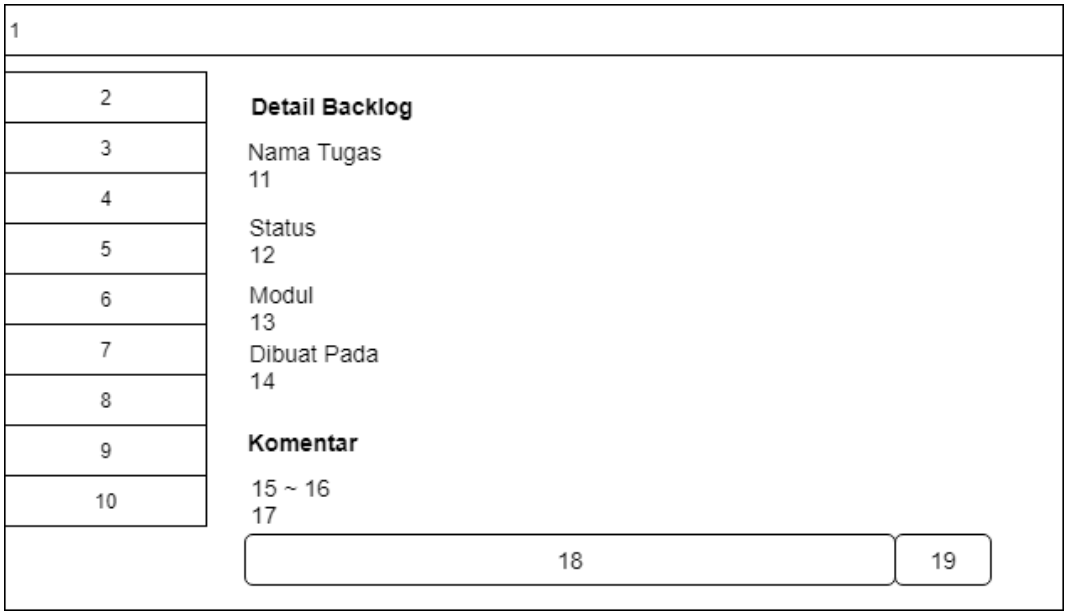
No.	Nama Objek	Tipe	Keterangan
1	Nama Proyek	Text	Menampilkan nama Proyek.
2	Road Map	Navigasi	Navigasi mengakses halaman roadmap
3	Back Log	Navigasi	Navigasi mengakses halaman backlog
4	Board	Navigasi	Navigasi mengakses halaman board
5	All Sprint	Navigasi	Navigasi mengakses halaman semua sprint
6	Project Setting	Navigasi	Navigasi mengakses halaman project setting
7	Riwayat Aktivitas	Navigasi	Navigasi mengakses halaman riwayat manajemen proyek
8	Report	Navigasi	Navigasi mengakses halaman report
9	Keluar Proyek	Navigasi	Navigasi untuk keluar dari halaman manajemen proyek
10	Keluar	Navigasi	Navigasi untuk keluar dari aplikasi
11	Nama Modul	Dropdown	Menampilkan daftar nama modul.
12	Filter	Tombol	Tombol untuk melakukan filter terhadap tugas
13	Nomor sprint	Text	Menampilkan nomor sprint.
14	Nama tugas	Text	Menampilkan nama tugas

Tabel 5.8 Penjelasan Perancangan Antarmuka Halaman Backlog

No.	Nama Objek	Tipe	Keterangan
15	Move to Backlog	Option	Untuk memindahkan tugas ke bagian backlog
16	Edit	Option	Untuk membuka pop up edit atau hapus tugas
17	Nama tugas	Text field	Field untuk mengisi nama tugas
18	Add	Tombol	Tombol untuk menambah tugas baru
19	Mulai Sprint	Tombol	Tombol untuk membuka pop up memulai sprint
20	Nama tugas	Text	Menampilkan nama tugas
21	Move to Sprint	Option	Untuk memindahkan tugas ke bagian sprint
22	Edit	Option	Untuk membuka pop up edit atau hapus tugas
23	Nama tugas	Text field	Field untuk mengisi nama tugas
24	Add	Tombol	Tombol untuk menambah tugas baru
25	Lama Sprint	Dropdown	Menampilkan daftar lama waktu pengerjaan sprint.
26	Cancel	Tombol	Tombol untuk menutup pop up
27	Ok	Tombol	Tombol untuk memulai sprint
28	Nama tugas	Text field	Field untuk edit nama tugas
29	Modul	Dropdown	Menampilkan daftar modul dari suatu proyek.
30	Edit	Tombol	Tombol untuk melakukan perubahan data tugas
31	Hapus	Tombol	Tombol untuk menghapus tugas

5.1.4.4 Halaman Detail Backlog

Gambar 5.11 merupakan perancangan dari halaman detail backlog. Penjelasan setiap objek dari perancangan antarmuka halaman detail backlog dijelaskan pada Tabel 5.9.



Gambar 5.11 Perancangan Antarmuka Halaman Detail Backlog

Tabel 5.9 Penjelasan Perancangan Antarmuka Halaman Detail Backlog

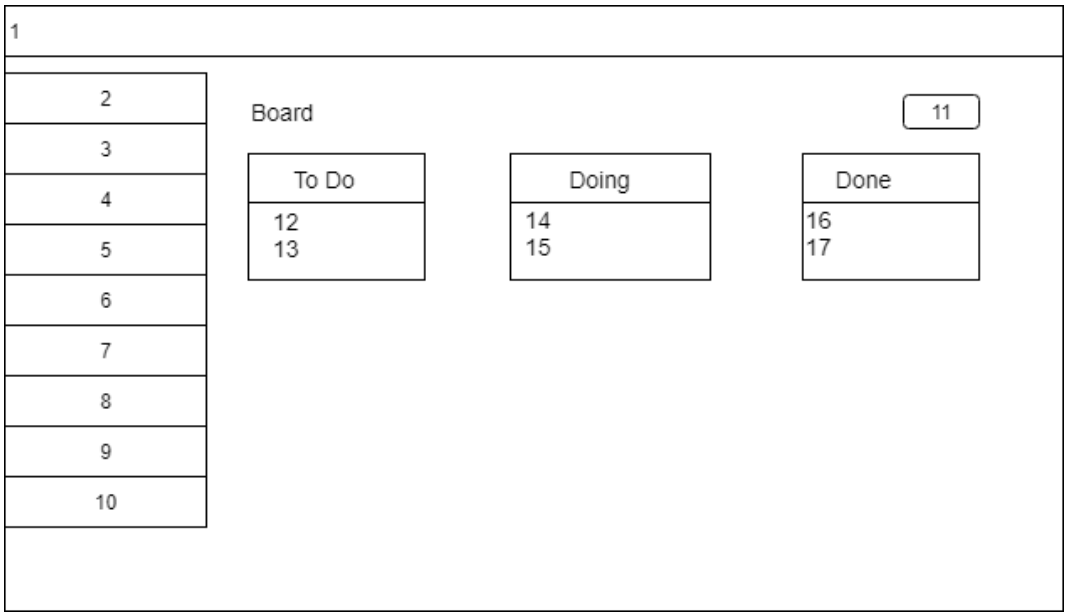
No.	Nama Objek	Tipe	Keterangan
1	Nama Proyek	Text	Menampilkan nama Proyek.
2	Road Map	Navigasi	Navigasi mengakses halaman roadmap
3	Back Log	Navigasi	Navigasi mengakses halaman backlog
4	Board	Navigasi	Navigasi mengakses halaman board
5	All Sprint	Navigasi	Navigasi mengakses halaman semua sprint
6	Project Setting	Navigasi	Navigasi mengakses halaman project setting
7	Riwayat Aktivitas	Navigasi	Navigasi mengakses halaman riwayat manajemen proyek
8	Report	Navigasi	Navigasi mengakses halaman report
9	Keluar Proyek	Navigasi	Navigasi untuk keluar dari halaman manajemen proyek
10	Keluar	Navigasi	Navigasi untuk keluar dari aplikasi
11	Nama Tugas	Text	Menampilkan nama tugas.
12	Status	Text	Menampilkan status pengerjaan dari tugas.
13	Modul	Text	Menampilkan nama modul dari tugas.
14	Dibuat	Text	Menampilkan tanggal tugas tersebut dibuat.

Tabel 5.9 Penjelasan Perancangan Antarmuka Halaman Detail Backlog

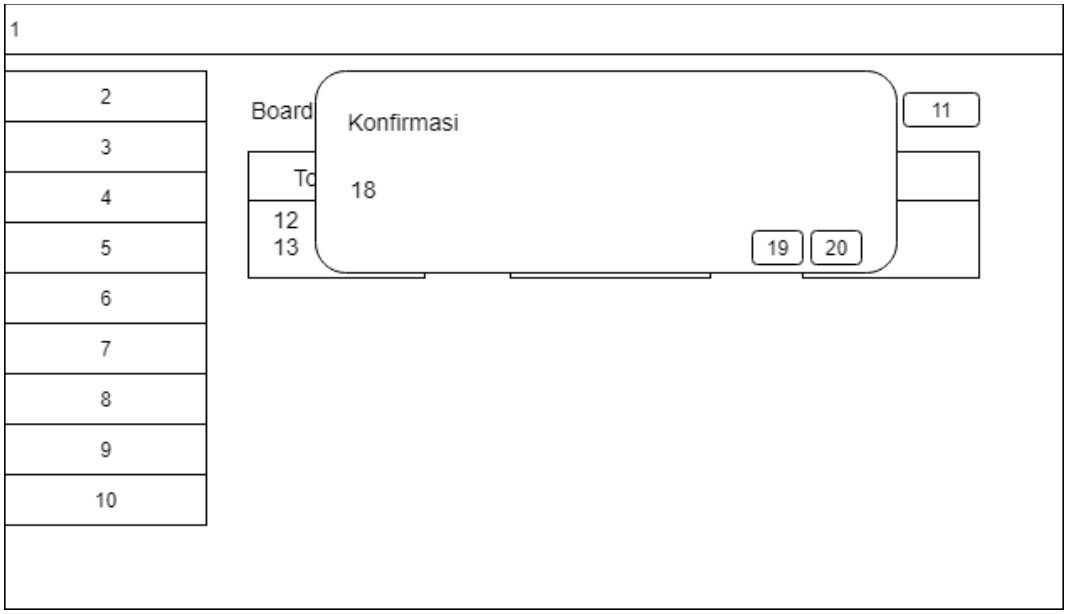
No.	Nama Objek	Tipe	Keterangan
15	Nama pengguna	Text	Menampilkan nama orang yang berkomentar.
16	Tanggal	Text	Menampilkan tanggal komentar tersebut dibuat.
17	Isi komentar	Text	Menampilkan komentar yang disampaikan oleh pengguna.
18	Komentar	Text field	Field untuk mengisi komentar
19	Komentar	Tombol	Untuk menyimpan komentar

5.1.4.5 Halaman Board

Gambar 5.12 merupakan perancangan dari halaman semua proyek. Terdapat beberapa tambahan gambar untuk menampilkan antarmuka dari suatu *pop up*. Gambar 5.13 akan muncul jika objek nomor 11 ditekan. Penjelasan setiap objek dari perancangan antarmuka halaman semua proyek dijelaskan pada Tabel 5.10.



Gambar 5.12 Perancangan Antarmuka Halaman Board



Gambar 5.13 Perancangan Antarmuka Selesai Sprint

Tabel 5.10 Penjelasan Perancangan Antarmuka Halaman Board

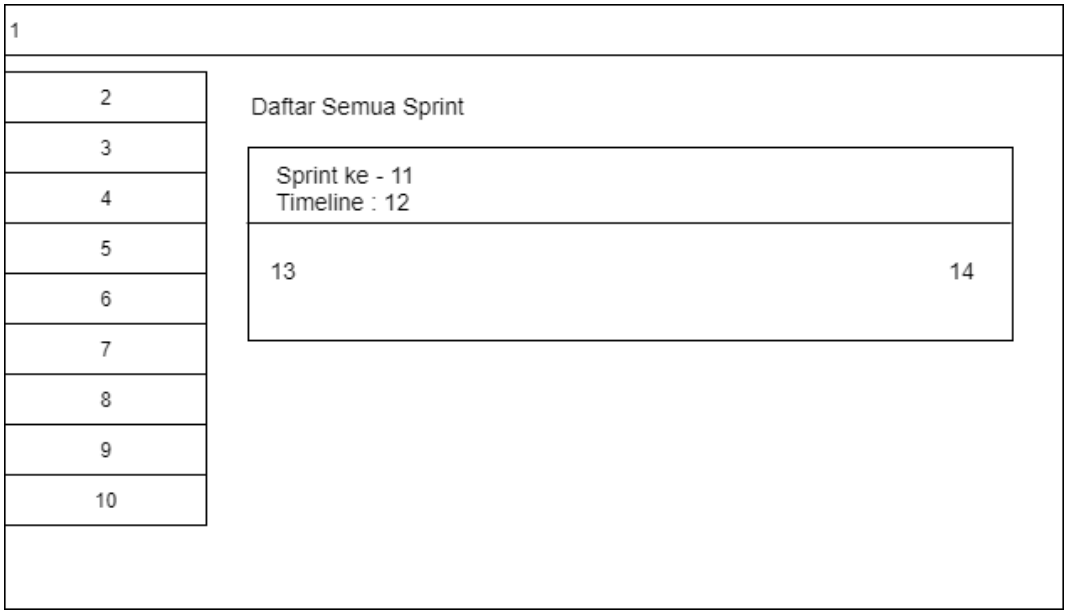
No.	Nama Objek	Tipe	Keterangan
1	Nama Proyek	Text	Menampilkan nama Proyek.
2	Road Map	Navigasi	Navigasi mengakses halaman roadmap
3	Back Log	Navigasi	Navigasi mengakses halaman backlog
4	Board	Navigasi	Navigasi mengakses halaman board
5	All Sprint	Navigasi	Navigasi mengakses halaman semua sprint
6	Project Setting	Navigasi	Navigasi mengakses halaman project setting
7	Riwayat Aktivitas	Navigasi	Navigasi mengakses halaman riwayat manajemen proyek
8	Report	Navigasi	Navigasi mengakses halaman report
9	Keluar Proyek	Navigasi	Navigasi untuk keluar dari halaman manajemen proyek
10	Keluar	Navigasi	Navigasi untuk keluar dari aplikasi
11	Selesai Sprint	Tombol	Untuk memampilkan pop up menghentikan sprint.
12	Nama tugas	Text	Menampilkan nama tugas yang berstatus To Do.
13	Tanggal	Text	Menampilkan tanggal tugas tersebut dibuat.

Tabel 5.10 Penjelasan Perancangan Antarmuka Halaman Board

No.	Nama Objek	Tipe	Keterangan
14	Nama tugas	Text	Menampilkan nama tugas yang berstatus Doing.
15	Tanggal	Text	Menampilkan tanggal tugas tersebut dibuat.
16	Nama tugas	Text	Menampilkan nama tugas yang berstatus Done.
17	Tanggal	Text	Menampilkan tanggal tugas tersebut dibuat.
18	Pesan	Text	Pesan konfirmasi dan dapat berisi jumlah tugas yang belum selesai.
19	Cancel	Tombol	Tombol untuk menutup pop up
20	Ok	Tombol	Tombol untuk menghentikan sprint

5.1.4.6 Halaman Semua Sprint

Gambar 5.14 merupakan perancangan dari halaman semua sprint. Penjelasan setiap objek dari perancangan antarmuka halaman semua sprint dijelaskan pada Tabel 5.11.



Gambar 5.14 Perancangan Antarmuka Halaman Semua Sprint

Tabel 5.11 Penjelasan Perancangan Antarmuka Halaman Semua Sprint

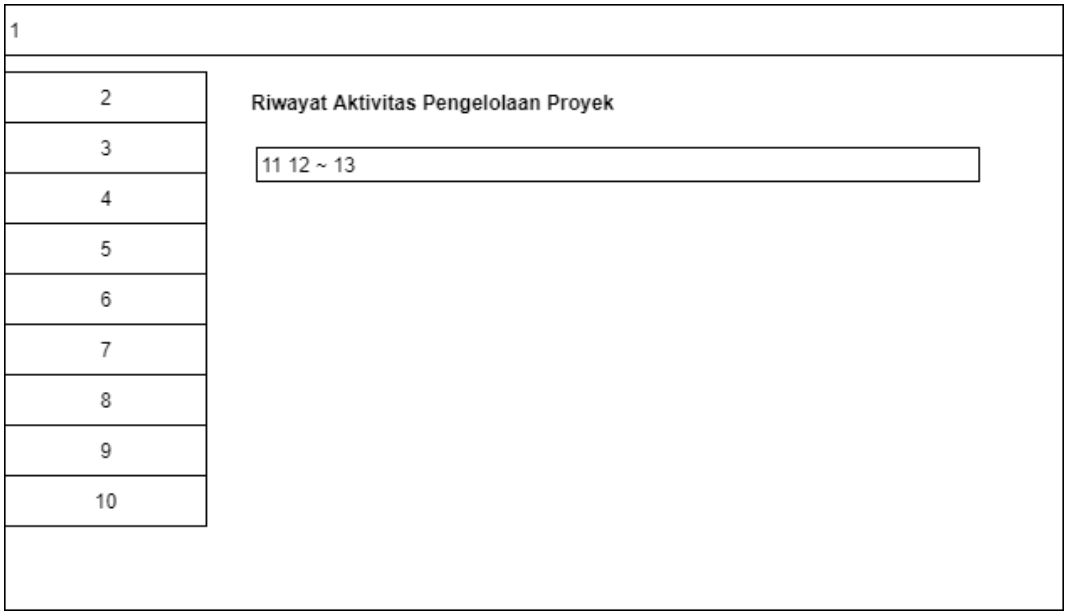
No.	Nama Objek	Tipe	Keterangan
1	Nama Proyek	Text	Menampilkan nama Proyek.
2	Road Map	Navigasi	Navigasi mengakses halaman roadmap

Tabel 5.11 Penjelasan Perancangan Antarmuka Halaman Semua Sprint

No.	Nama Objek	Tipe	Keterangan
3	Back Log	Navigasi	Navigasi mengakses halaman backlog
4	Board	Navigasi	Navigasi mengakses halaman board
5	All Sprint	Navigasi	Navigasi mengakses halaman semua sprint
6	Project Setting	Navigasi	Navigasi mengakses halaman project setting
7	Riwayat Aktivitas	Navigasi	Navigasi mengakses halaman riwayat manajemen proyek
8	Report	Navigasi	Navigasi mengakses halaman report
9	Keluar Proyek	Navigasi	Navigasi untuk keluar dari halaman manajemen proyek
10	Keluar	Navigasi	Navigasi untuk keluar dari aplikasi
11	Nomor Sprint	Text	Menampilkan nomor sprint.
12	Timeline	Text	Menampilkan Tanggal mulai dan tanggal selesai dari sprint.
13	Nama tugas	Text	Menampilkan nama tugas yang selesai pada sprint tersebut.
14	Move to Current Sprint	Option	Untuk mengerjakan ulang tugas ke bagian sprint yang saat ini sedang berjalan

5.1.4.7 Halaman Riwayat Aktivitas

Gambar 5.15 merupakan perancangan dari halaman Riwayat Aktivitas. Penjelasan setiap objek dari perancangan antarmuka halaman Riwayat Aktivitas dijelaskan pada Tabel 5.12.



Gambar 5.15 Perancangan Antarmuka Halaman Riwayat Aktivitas

Tabel 5.12 Penjelasan Perancangan Antarmuka Halaman Riwayat Aktivitas

No.	Nama Objek	Tipe	Keterangan
1	Nama Proyek	Text	Menampilkan nama Proyek.
2	Road Map	Navigasi	Navigasi mengakses halaman roadmap
3	Back Log	Navigasi	Navigasi mengakses halaman backlog
4	Board	Navigasi	Navigasi mengakses halaman board
5	All Sprint	Navigasi	Navigasi mengakses halaman semua sprint
6	Project Setting	Navigasi	Navigasi mengakses halaman project setting
7	Riwayat Aktivitas	Navigasi	Navigasi mengakses halaman riwayat manajemen proyek
8	Report	Navigasi	Navigasi mengakses halaman report
9	Keluar Proyek	Navigasi	Navigasi untuk keluar dari halaman manajemen proyek
10	Keluar	Navigasi	Navigasi untuk keluar dari aplikasi
11	Nama pengguna	Text	Menampilkan nama orang yang melakukan kegiatan manajemen.
12	Isi kegiatan	Text	Menampilkan aktivitas yang dilakukan pengguna tersebut.
13	Waktu	Text	Menampilkan waktu aktivitas tersebut dilakukan.

5.2 Implementasi

Pada tahap implementasi, penulis akan memaparkan tentang implementasi dari aplikasi manajemen multi proyek menggunakan metode scrum, berdasarkan perancangan yang telah dibuat. Implementasi tersebut terdiri dari spesifikasi pengembangan aplikasi, implementasi kode program, dan implementasi antarmuka.

5.2.1 Spesifikasi Pengembangan Aplikasi

Spesifikasi pengembangan aplikasi terdiri dari informasi spesifikasi perangkat keras dan lunak yang digunakan penulis dalam proses pengembangan aplikasi manajemen multi proyek menggunakan metode scrum.

5.2.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras meliputi komponen perangkat seperti processor, hard disk, RAM, dan system model dari perangkat penulis dalam mengembangkan aplikasi. Spesifikasi perangkat keras dapat dilihat pada Tabel 5.13.

Tabel 5.13 Spesifikasi Perangkat Keras

Komponen	Spesifikasi
Processor	Intel Core i3
Hard Disk	500 GB
RAM	4.00 GB
System Model	Acer Apsire 4750

5.2.1.1 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak meliputi komponen seperti sistem operasi, Bahasa pemrograman, *text editor*, dan server yang digunakan oleh penulis dalam mengembangkan aplikasi. Spesifikasi perangkat lunak dapat dilihat pada Tabel 5.14.

Tabel 5.14 Spesifikasi Perangkat Lunak

Komponen	Spesifikasi
Sistem Operasi	Widows 10 Pro
Bahasa Pemrograman	PHP
Text Editor	Sublime Text
Server	XAMPP

5.2.2 Implementasi Kode Program

Implementasi kode program memuat tentang kode program dari aplikasi manajemen multi proyek menggunakan metode scrum berdasarkan *pseudocode* yang telah dibuat pada Bab 5.1.3

5.2.2.1 Implementasi Kode Program *Method* buat_proyek

Nama fungsi : `buat_proyek()`

Nama kelas : `ProyekController`

Source Code :

Tabel 5.15 *Source Code* Membuat Proyek

```
1  public function buat_proyek(Request $req){
2        $this->validate($req, [
               'nama_proyek'     => 'required'
         ]);
3        M_proyek::create([
               'nama_proyek'     => $req->nama_proyek,
               'deskripsi'       => $req->deskripsi,
               'id_pengguna'     => Auth::user()->id
         ]);
4        return back();
5  }
```

Penjelasan:

1	Deklarasi fungsi buat proyek dengan parameter dari hasil submit form.
2	Melakukan validasi terhadap form input, bahwa field nama_proyek harus bernilai atu tidak boleh kosong.
3	Membuat data record baru ke dalam table proyek dengan nilai nama_proyek dan deskripsi yang diperoleh dari form submit.
4	Mengakses halaman semula untuk membuat proyek.
5	Akhir method buat_proyek.

5.2.2.2 Implementasi Kode Program *Method* mulai_sprint

Nama fungsi : `mulai_sprint()`

Nama kelas : `ProyekController`

Source Code :

Tabel 5.16 *Source Code* Memulai Sprint

```
1  public function mulai_sprint(Request $req){
```

```
2       $obj_nomor_sprint = M_sprint::where('id_proyek',
Session::get('id_proyek'))
                  ->orderBy('nomor_sprint', 'desc')
                  ->limit(1)
                ->get();
3
4         if(count($obj_nomor_sprint) == 0){
5             $nomor_sprint = 1;
6         } else {
7             $nomor_sprint = $obj_nomor_sprint[0]-
>nomor_sprint + 1;
8         }
9
10            $today = date('Y-m-d');
11        $tgl_selesai = date('Y-m-d 23:59:59',
strtotime($today. " +" . ($req->jangka_waktu * 7) . "
day"));
12
13      M_sprint::create([
              'id_proyek'       => Session::get('id_proyek'),
              'nomor_sprint'    => $nomor_sprint,
              'tgl_mulai'       => $today,
              'tgl_selesai'     => $tgl_selesai
      ]);
14
15        $this->simpan_riwayat("memulai sprint ke
$nomor_sprint, dari tanggal $today hingga $tgl_selesai
(selama $req->jangka_waktu minggu)");
16
17        $new_sprint = M_sprint::where('id_proyek',
Session::get('id_proyek'))
                ->orderBy('nomor_sprint', 'desc')
                ->limit(1)
                ->get();
18
19        M_tugas::where([
              'id_proyek' => Session::get('id_proyek'),
              'id_sprint' => null,
```

```
            'is_backlog'=> 0
        ])->update([
            'id_sprint' => $new_sprint[0]->id_sprint
        ]);

    return back();
}
```

Penjelasan:

1	Deklarasi fungsi mulai_sprint dengan parameter dari hasil submit form.
2	Inisialisasi variabel obj_nomor_sprint dengan nilai dari pengaksesan tabel sprint pada data sprint terakhir dari setiap proyek.
4	Jika total data pada variabel obj_nomor_sprint bernilai 0, maka
5	Inisialisasi variabel nomor_sprint bernilai 1.
6	Jika tidak, maka
7	Inisialisasi variabel nomor_sprint bernilai data dari obj_nomor_sprint atribut nomor_sprint ditambah 1.
10	Inisialisasi variabel today bernilai hari ini.
11	Inisialisasi variabel tgl_selesai bernilai hari ini ditambah sekian hari sesuai inputan pengguna dari form mulai sprint.
13	Membuat data record baru ke dalam tabel sprint dengan nilai dari form submit yang dilakukan pengguna.
15	Memanggil method simpan_riwayat untuk membuat data baru pada tabel riwayat.
17	Inisialisasi variabel new_sprint dengan nilai dari pengaksesan tabel sprint pada data sprint terakhir dari setiap proyek.
19	Melakukan perubahan data pada tabel tugas untuk mengganti nilai kolom id_sprint.
21	Mengakses halaman semula untuk membuat proyek.
22	Akhir method mulai_sprint.

5.2.2.3 Implementasi Kode Program *Method* ubah_status_tugas

Nama fungsi : `ubah_status_tugas()`

Nama kelas : `ProyekController`

Source Code :

Tabel 5.17 *Source Code* Mengubah Status Tugas Sprint

```
1  public function ubah_status_tugas(Request $req){
2          $tugas = M_tugas::find($req->id_tugas);
3          M_tugas::where('id_tugas', $req->id_tugas)-
   >update(['status' => $req->status]);
4          $this->simpan_riwayat("mengubah status tugas dari
    $tugas->status ke $req->status", $req->id_tugas);
5          return back();
6  }
```

Penjelasan:

1	Deklarasi fungsi ubah_status_tugas dengan parameter dari hasil submit form.
2	Inisialisasi variabel tugas dengan nilai dari pengaksesan tabel tugas pada data yang memiliki id_tugas sesuai dari hasil submit form.
3	Melakukan perubahan data pada tabel tugas untuk mengganti nilai kolom status.
4	Memanggil method simpan_riwayat untuk membuat data baru pada tabel Riwayat
5	Mengakses halaman semula.
6	Akhir method ubah_status_tugas.

5.2.2.4 Implementasi Kode Program *Method* backlog

Nama fungsi : `backlog()`

Nama kelas : `ProyekController`

Source Code :

Tabel 5.18 *Source Code* Menampilkan Halaman Backlog

```
1  public function backlog(Request $req){
2          $nomor_sprint = M_sprint::where('id_proyek',
   Session::get('id_proyek'))
                   ->orderBy('nomor_sprint', 'desc')
                   ->limit(1)->get();
3
4          $where1 = [ 'id_proyek'     =>
   Session::get('id_proyek'),
                       'is_backlog'    => 0];
```

```
5         $where2 = [ 'id_proyek'     =>
Session::get('id_proyek'),
                    'id_sprint'     => $nomor_sprint[0]-
>id_sprint,
                    'is_backlog'    => 0];
6         $where3 = [ 'id_proyek'     =>
Session::get('id_proyek'),
                    'id_sprint'     => null,
                    'is_backlog'    => 0];
7         $where_backlog = [ 'id_proyek'     =>
Session::get('id_proyek'),
                            'is_backlog'    => 1];
8
9         $id_modul = 0;
10        if(isset($req->id_modul)){
11            if($req->id_modul != 'all'){
12                $where1['id_modul'] = $req->id_modul;
13                $where2['id_modul'] = $req->id_modul;
14                $where3['id_modul'] = $req->id_modul;
15                $where_backlog['id_modul'] = $req->id_modul;
16                $id_modul = $req->id_modul;
17            }
18        }
19
20        if (count($nomor_sprint) == 0) {
21            $sprint = M_tugas::where($where1)->get();
22        } else {
23            if ($nomor_sprint[0]->tgl_selesai >= date('Y-m-d
G:i:s')) {
24                $sprint = M_tugas::where($where2)->get();
25            } else {
26                $sprint = M_tugas::where($where3)->get();
27            }
28        }
29
30        $modul = M_modul::where('id_proyek',
Session::get('id_proyek'))->get();
31        $backlog = M_tugas::where($where_backlog)->get();
```

```
32
33          return view('backlog', compact('nomor_sprint',
    'sprint', 'backlog', 'modul', 'id_modul'));
34  }
```

Penjelasan:

1	Deklarasi fungsi backlog dengan parameter dari hasil submit form.
2	Inisialisasi variabel nomor_sprint dengan nilai dari pengaksesan tabel sprint pada data sprint terakhir dari setiap proyek.
4	Inisialisasi variabel where1 bertipe array yang menyimpan nilai id_proyek dan is_backlog.
5	Inisialisasi variabel where2 bertipe array yang menyimpan nilai id_proyek, id_sprint, dan is_backlog.
6	Inisialisasi variabel where3 bertipe array yang menyimpan nilai id_proyek, id_sprint dan is_backlog.
7	Inisialisasi variabel where_backlog bertipe array yang menyimpan nilai id_proyek, dan is_backlog.
9	Inisialisasi variabel id_modul bernilai 0.
10	Jika field id_modul memiliki nilai, maka
11	Jika nilai field id_modul tidak benilai “all”, maka
12	Inisialisasi variabel where1[‘id_modul’] bernilai dari field form id_modul.
13	Inisialisasi variabel where2[‘id_modul’] bernilai dari field form id_modul.
14	Inisialisasi variabel where3[‘id_modul’] bernilai dari field form id_modul.
15	Inisialisasi variabel where_backlog[‘id_modul’] bernilai dari field form id_modul.
16	Inisialisasi variabel id_modul bernilai dari field form id_modul.
17	Penutup if baris 11.
18	Penutup if baris 10.
20	Jika total data pada variabel nomor_sprint bernilai 0, maka
21	Inisialisasi variabel sprint dengan nilai dari pengaksesan tabel tugas menggunakan kriteria dari variabel where1.
22	Jika tidak, maka
23	Jika nomor_sprint index ke 0 atribut tgl_selesai bernilai lebih dari sama dengan waktu saat ini, maka

24	Inisialisasi variabel sprint dengan nilai dari pengaksesan tabel tugas menggunakan kriteria dari variabel where2.
25	Jika tidak, maka
26	Inisialisasi variabel sprint dengan nilai dari pengaksesan tabel tugas menggunakan kriteria dari variabel where3.
27	Penutup else baris 25.
28	Penutup else baris 22.
30	Inisialisasi variabel modul dengan nilai dari pengaksesan tabel modul mengambil data dengan id_proyek sesuai dengan proyek yang dibuka.
31	Inisialisasi variabel backlog dengan nilai dari pengaksesan tabel tugas menggunakan kriteria dari variabel where_backlog.
33	Mengakses halaman backlog dengan membawa data variabel nomor_sprint, sprint, backlog, modul, dan id_modul.
34	Akhir method backlog.

5.2.2.5 Implementasi Kode Program *Method*

Nama fungsi : `ubah_status_backlog`

Nama kelas : `ProyekController`

Source Code :

Tabel 5.19 *Source Code* Mengubah Posisi tugas Backlog

```
1  public function ubah_status_backlog(Request $req){
2        $last_sprint = M_sprint::where('id_proyek',
   Session::get('id_proyek'))
                    ->orderBy('nomor_sprint', 'desc')
                    ->limit(1)->get();
3
4        if (count($last_sprint) == 0) {
5              M_tugas::where('id_tugas', $req->id_tugas)-
   >update(['is_backlog' => 0]);
6        } else {
7              if ($last_sprint[0]->tgl_selesai >= date('Y-m-d
   G:i:s')) {
8                    M_tugas::where('id_tugas', $req-
   >id_tugas)->update([
                           'is_backlog'      => 0,
```

```
                        'id_sprint'       =>
$last_sprint[0]->id_sprint
                    ]);
9           } else {
10                  M_tugas::where('id_tugas', $req-
>id_tugas)->update(['is_backlog' => 0]);
11          }
12      }
13  }
```

Penjelasan:

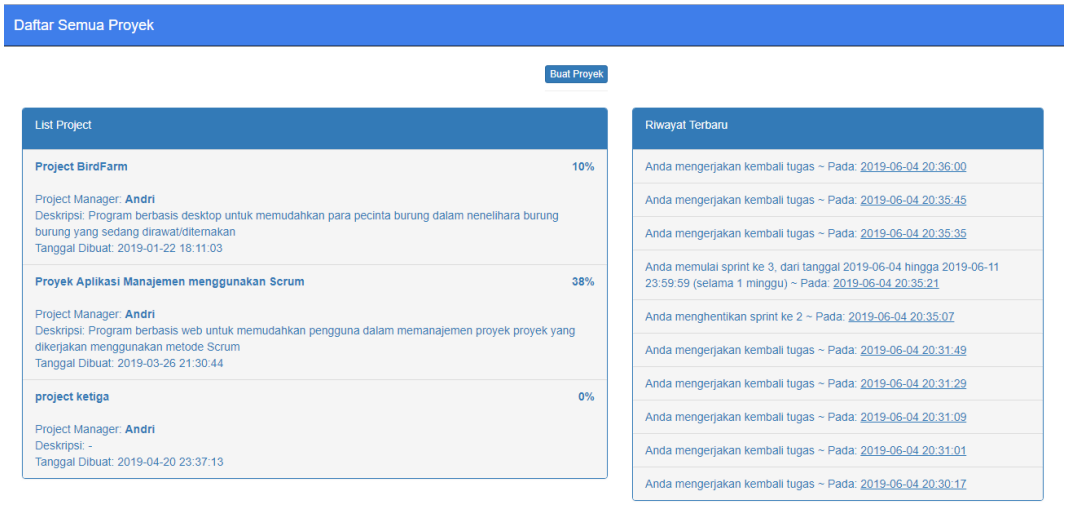
1	Deklarasi fungsi ubah_status_backlog dengan parameter dari hasil submit form.
2	Inisialisasi variabel last_sprint dengan nilai dari pengaksesan tabel sprint pada data sprint terakhir dari setiap proyek.
4	Jika total data pada variabel last_sprint bernilai 0, maka
5	Melakukan perubahan data pada tabel tugas untuk mengganti nilai kolom Is_backlog dengan nilai 0.
6	Jika tidak, maka
7	Jika last_sprint index ke 0 atribut tgl_selesai bernilai lebih dari sama dengan waktu saat ini, maka
8	Melakukan perubahan data pada tabel tugas untuk mengganti nilai kolom Is_backlog dengan nilai 0, dan id_sprint dengan nilai dari last_sprint index ke 0 atribut id_sprint
9	Jika tidak, maka
10	Melakukan perubahan data pada tabel tugas untuk mengganti nilai kolom Is_backlog dengan nilai 0
11	Penutup else baris 9.
12	Penutup else baris 6.
13	Akhir method ubah_status_backlog.

5.2.3 Implementasi Antarmuka

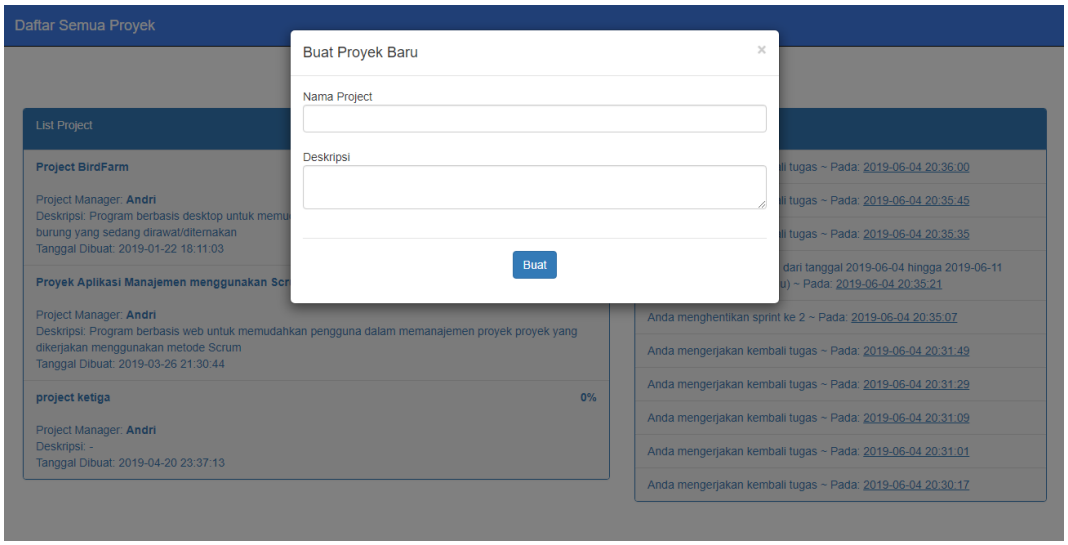
Implementasi antarmuka aplikasi merupakan hasil penerapan dari perancangan antarmuka yang telah dibuat sebelumnya pada Bab 5.1.4.

5.2.3.1 Halaman Semua Proyek

Halaman Semua Proyek pada aplikasi manajemen multi proyek menggunakan metode scrum merupakan halaman yang pertama kali muncul setelah pengunjung berhasil masuk. Aplikasi akan menampilkan semua proyek yang sedang atau telah dikerjakan. Implementasi antarmuka halaman semua proyek dapatdilhat pada gambar 5.16. Sedangkan untuk membuat proyek baru, pengguna dapat menekan tombol buat proyek kemudian akan muncul pop up form pembuatan proyek. Implementasi antarmuka Pop Up untuk membuat proyek dapat dilihat pada gambar 5.17.



Gambar 5.16 Implementasi Antarmuka Halaman Semua Proyek

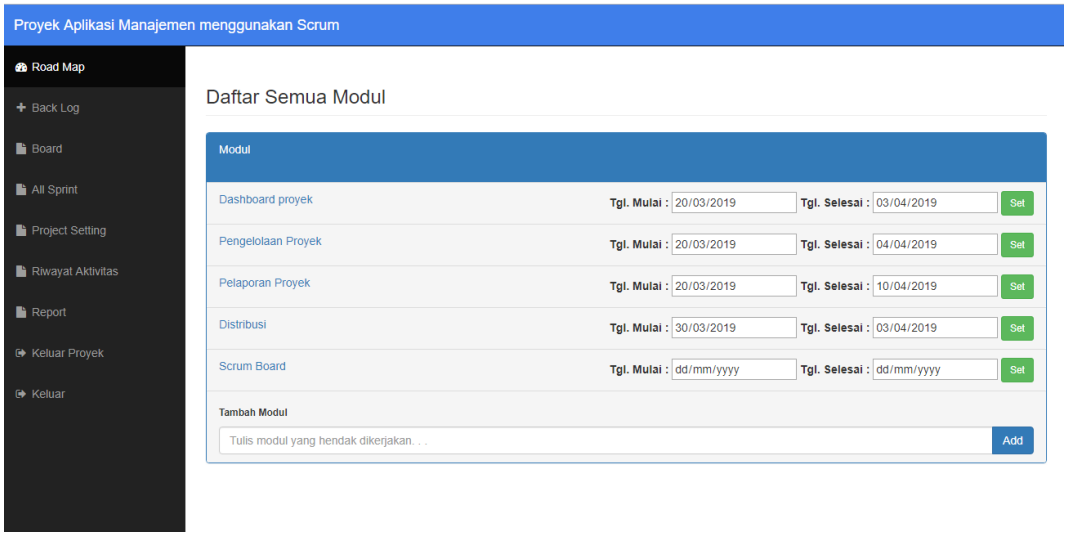


Gambar 5.17 Implementasi Antarmuka Pop Up untuk Buat Proyek

5.2.3.2 Halaman Semua Modul

Halaman semua modul pada aplikasi manajemen multi proyek menggunakan metode scrum merupakan halaman yang berisi tentang modul modul yang terdapat pada suatu proyek. Pengguna juga dapat menambah modul baru dengan

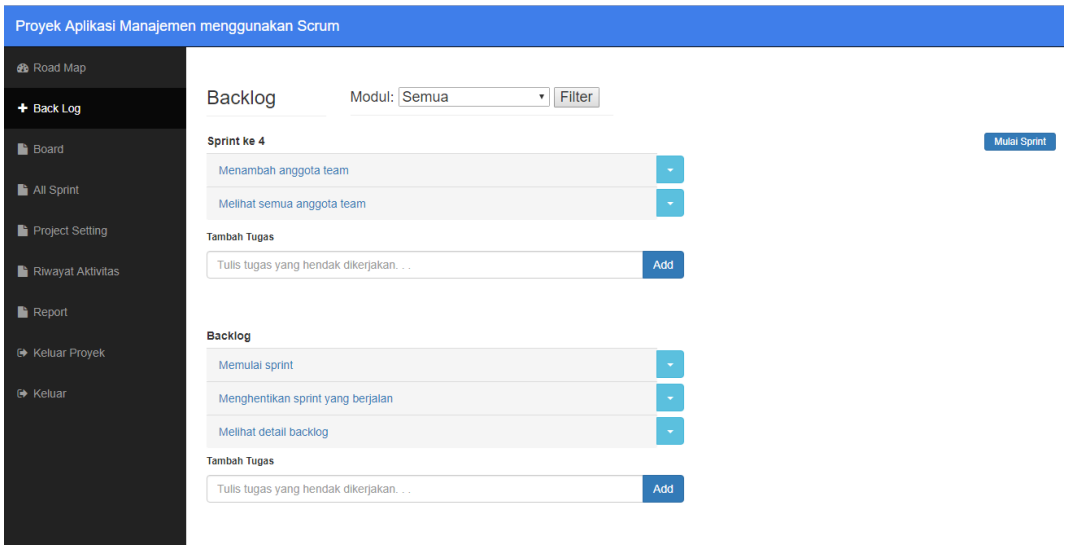
memasukkan nama modul. Implementasi antarmuka halaman semua modul dapat dilihat pada gambar 5.18.



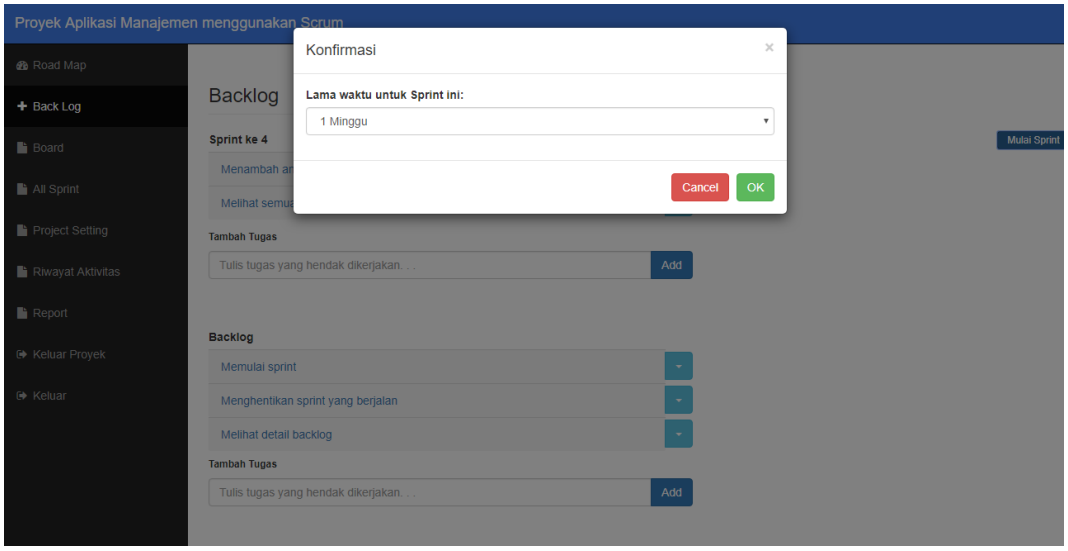
Gambar 5.18 Implementasi Antarmuka Halaman Semua Modul

5.2.3.3 Halaman Backlog

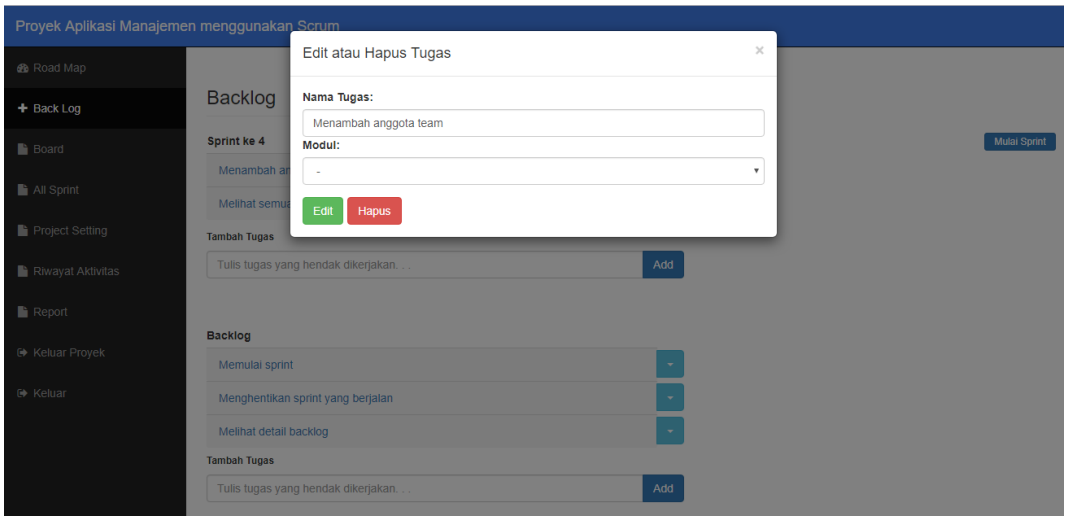
Halaman backlog pada aplikasi manajemen multi proyek menggunakan metode scrum merupakan halaman yang menampilkan tugas tugas yang tersimpan pada sprint dan backlog. Pada halaman ini pengguna dapat memulai sprint dan melakukan edit pada tugas. Implementasi antarmuka halaman backlog dapat dilihat pada gambar 5.19. Untuk implementasi antarmuka pop up untuk memulai sprint dapat dilihat pada gambar 5.20. Implementasi antarmuka pop up untuk melakukan edit atau hapus tugas dapat dilihat pada gambar 5.21.



Gambar 5.19 Implementasi Antarmuka Halaman BackLog



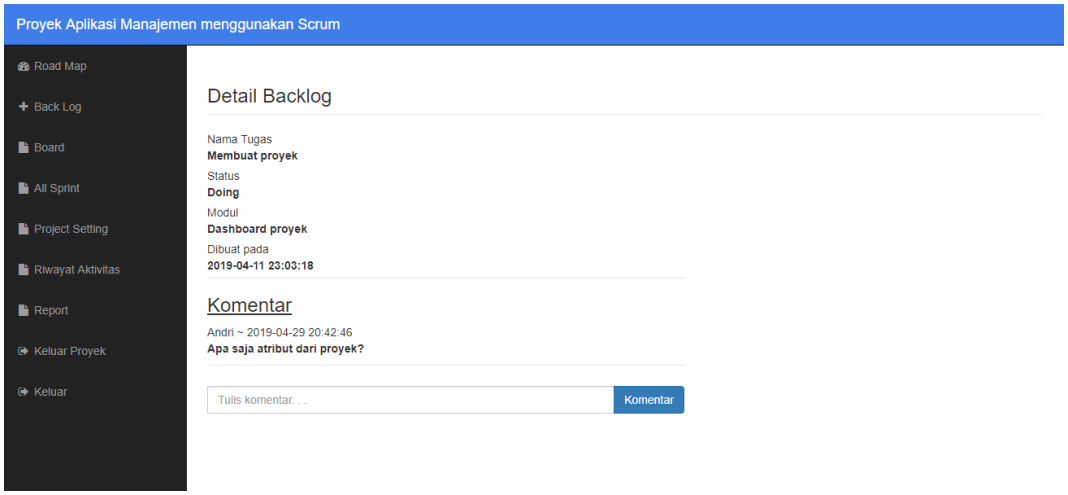
Gambar 5.20 Implementasi Antarmuka Pop Up untuk Memulai Sprint



Gambar 5.21 Implementasi Antarmuka Pop Up untuk Edit Tugas

5.2.3.4 Halaman Detail Backlog

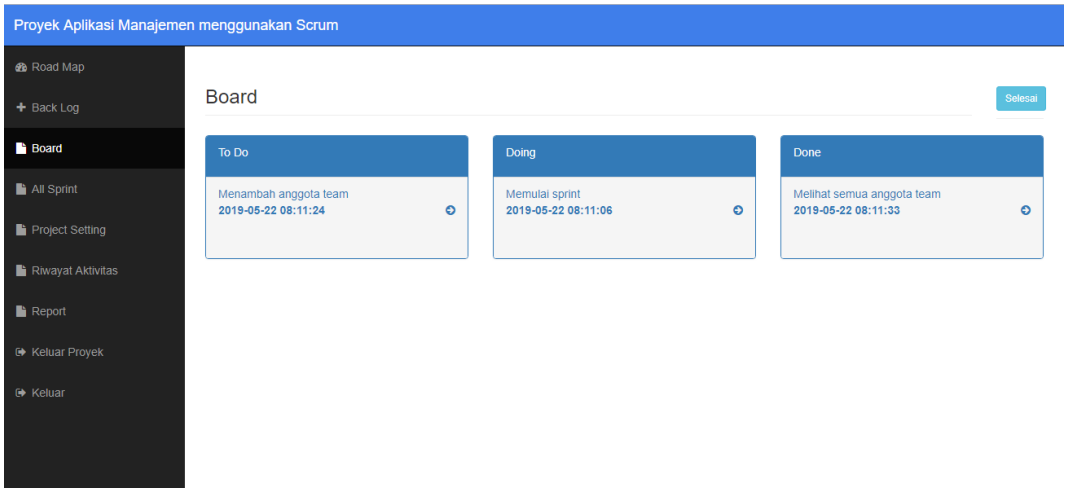
Halaman detail backlog pada aplikasi manajemen multi proyek menggunakan metode scrum merupakan halaman untuk menampilkan infomasi tentang detail tugas. Pada halaman ini pengguna juga dapat memberikan komentar. Implementasi dari antarmuka halaman detail backlog dapat dilihat pada gambar 5.22.



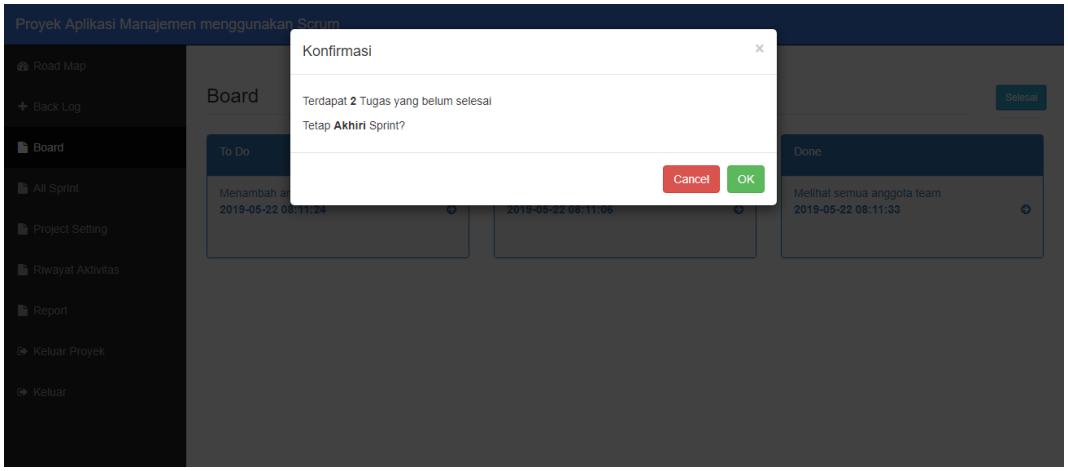
Gambar 5.22 Implementasi Antarmuka Halaman Detail Backlog

5.2.3.5 Halaman Board

Halaman board pada aplikasi manajemen multi proyek menggunakan metode scrum merupakan halaman untuk menampilkan tugas berdasarkan status pengerjaannya. Pengguna dapat mengubah status tugas dengan menyeret tugas tersebut. Implementasi dari antarmuka halaman board dapat dilihat pada gambar 5.23. Untuk implementasi antarmuka pop up untuk menghentikan sprintdapat dilihat pada gambar 5.24.



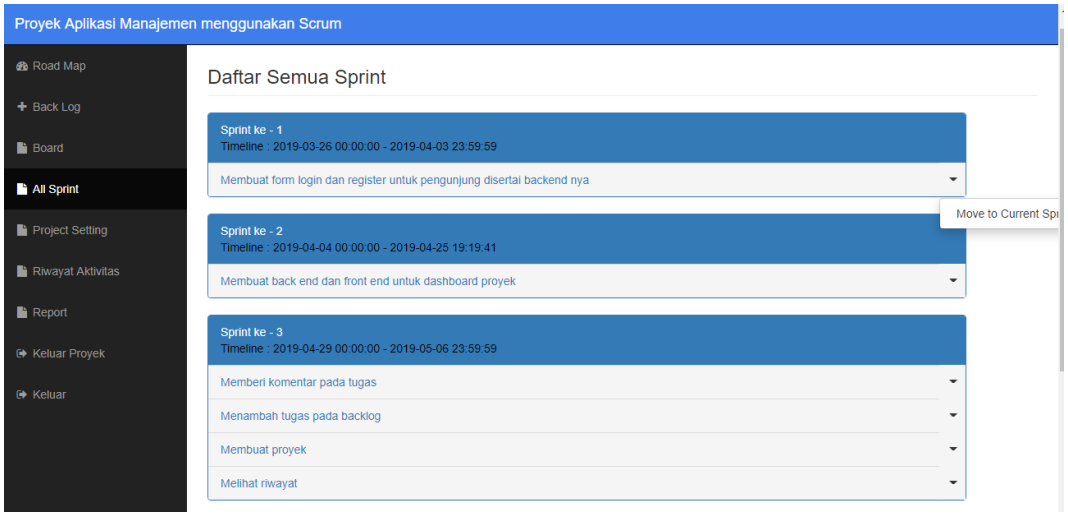
Gambar 5.23 Implementasi Antarmuka Halaman Board



Gambar 5.24 Implementasi Antarmuka Pop Up untuk Menghetikan Sprint

5.2.3.6 Halaman Semua Sprint

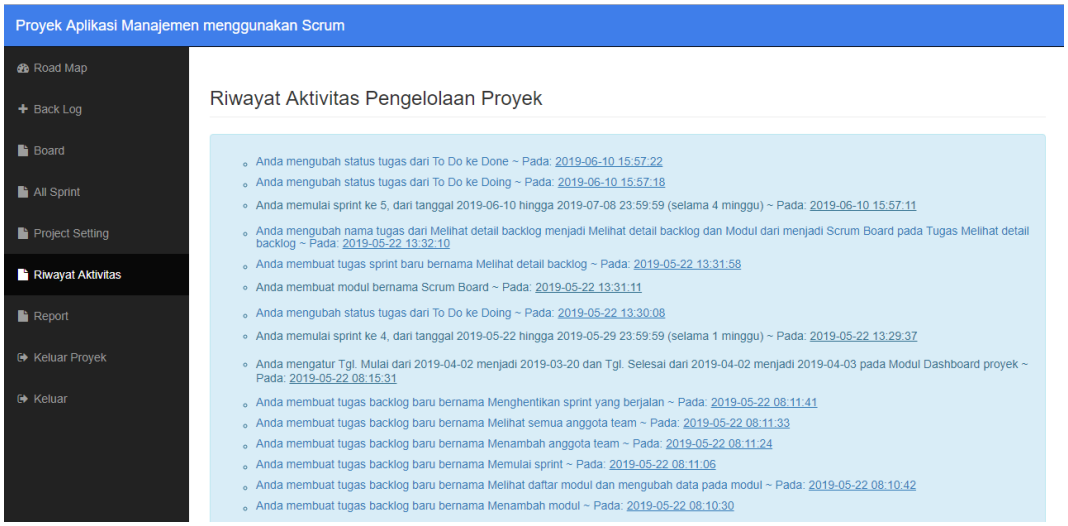
Halaman semua proyek pada aplikasi manajemen multi proyek menggunakan metode scrum merupakan halaman untuk menampilkan semua sprint yang telah atau sedang berjalan. Didalam setiap sprint berisi daftar tugas yang selesai dikerjakan pada sprint tersebut. Implementasi dari antarmuka halaman semua sprint dapat dilihat pada gambar 5.25.



Gambar 5.25 Implementasi Antarmuka Halaman Semua Sprint

5.2.3.7 Halaman Riwayat Aktivitas

Halaman riwayat aktivitas pada aplikasi manajemen multi proyek menggunakan metode scrum merupakan halaman untuk menampilkan semua kegiatan yang dilakukan oleh anggota tim. Kegiatan tersebut dapat berupa pembuatan, perubahan data, penghapusan, dan lainnya yang berkaitan dengan proses manajemen proyek. Implementasi dari antarmuka halaman riwayat aktivitas dapat dilihat pada gambar 5.26



Gambar 5.26 Implementasi Antarmuka Halaman Riwayat Aktivitas

5.2.4 Implementasi Data

Implementasi data memuat tentang DDL atau Data Definition Language berdasarkan basis data yang digunakan dalam mengembangkan aplikasi manajemen multi proyek menggunakan metode scrum. Berikut akan dicantumkan query yang digunakan dalam membuat tabel untuk aplikasi manajemen multi proyek menggunakan metode scrum. DDL tersebut dapat dilihat pada Tabel 5.23.

Tabel 5.20 Implementasi Data

```
CREATE TABLE `komentar` (
  `id_komentar` int(11) NOT NULL,
  `id_pengguna` int(11) NOT NULL,
  `isi` text NOT NULL,
  `id_tugas` int(11) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


CREATE TABLE `modul` (
  `id_modul` int(11) NOT NULL,
  `nama_modul` varchar(100) NOT NULL,
  `id_proyek` int(11) NOT NULL,
  `tgl_mulai` date DEFAULT NULL,
  `tgl_selesai` date DEFAULT NULL,
  `deskripsi` text,
  `status` varchar(10) DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `pengguna` (
  `id` int(11) NOT NULL,
  `nama` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(70) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `remember_token` varchar(70) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `proyek` (
  `id_proyek` int(11) NOT NULL,
  `nama_proyek` varchar(100) NOT NULL,
  `deskripsi` text,
  `id_pengguna` int(11) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `riwayat` (
  `id_riwayat` int(11) NOT NULL,
  `id_proyek` int(11) NOT NULL,
  `id_pengguna` int(11) NOT NULL,
  `id_tugas` int(11) DEFAULT NULL,
  `aktivitas` text NOT NULL,
  `waktu` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `sprint` (
  `id_sprint` int(11) NOT NULL,
  `id_proyek` int(11) NOT NULL,
  `nomor_sprint` int(11) NOT NULL,
  `tgl_mulai` timestamp NULL DEFAULT NULL,
  `tgl_selesai` timestamp NULL DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `team` (
  `id_proyek` int(11) NOT NULL,
```

```
  `id_pengguna` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `tugas` (
  `id_tugas` int(11) NOT NULL,
  `nama_tugas` varchar(100) NOT NULL,
  `id_proyek` int(11) NOT NULL,
  `id_modul` int(11) DEFAULT NULL,
  `status` varchar(50) DEFAULT NULL,
  `id_sprint` int(11) DEFAULT NULL,
  `is_backlog` tinyint(1) DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

ALTER TABLE `komentar`
  ADD PRIMARY KEY (`id_komentar`);
ALTER TABLE `modul`
  ADD PRIMARY KEY (`id_modul`);
ALTER TABLE `pengguna`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `email` (`email`);
ALTER TABLE `proyek`
  ADD PRIMARY KEY (`id_proyek`);
ALTER TABLE `riwayat`
  ADD PRIMARY KEY (`id_riwayat`);
ALTER TABLE `sprint`
  ADD PRIMARY KEY (`id_sprint`);
ALTER TABLE `tugas`
  ADD PRIMARY KEY (`id_tugas`);
```

BAB 6 PENGUJIAN

Pada tahap pengujian, dilakukan pemeriksaan dari hasil proses implementasi aplikasi untuk mengetahui apakah sistem telah sesuai dengan analisis kebutuhan dan perancangan. Pengujian adalah bagian yang dilakukan setelah melakukan implementasi. Pengujian yang dilakukan berupa pengujian unit, pengujian validasi, dan pengujian *compatibility*.

6.1 Pengujian Unit

Pengujian unit adalah pengujian yang dilakukan dengan cara menguji setiap unit dari sistem meliputi komponen, kelas, atau objek. Pengujian unit ini merupakan pengujian dengan metode white-box dengan menggunakan jenis basis *path testing*.

6.1.1 Pengujian Unit *Method* buat_proyek

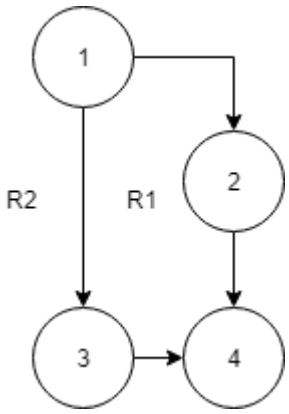
1. Pseudocode

Tabel 6.1 *Pseudocode* Pengujian *Method* buat_proyek

Start method buat_proyek. (1)	
If nilai form nama_proyek == null Kembali kehalaman semua proyek	(2)
End if Simpan proyek baru dalam database Kembali kehalaman semua proyek	(3)
End (4)	

2. Basis Path Testing

a. Flow Graph



Gambar 6.1 *Flow Graph Method* buat_proyek

b. Cyclomatic Complexity

i. V(G) = 2, terdapat 2 region

ii. V(G) = 4 edges – 4 nodes + 2 = 2

iii. V(G) = 1 predicate nodes + 1 = 2

c. Independent Path

a) Jalur 1 = 1 – 2 – 4

b) Jalur 2 = 1 – 3 – 4

Tabel 6.2 Hasil Pengujian Unit *Method* buat_proyek

No.	No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1	nama_proyek = “”.	Aplikasi akan kembali memuat halaman semua proyek.	Aplikasi kembali memuat halaman semua proyek.	Valid
2.	2	nama_proyek = “Proyek Aplikasi Manajemen Multi Proyek.	Aplikasi akan menyimpan proyek dan kembali kehalaman semua proyek.	Aplikasi berhasil menyimpan proyek dan kembali kehalaman semua proyek.	Valid

6.1.2 Pengujian Unit *Method* mulai_sprint

1. Pseudocode

Tabel 6.3 *Pseudocode* Pengujian *Method* mulai_sprint

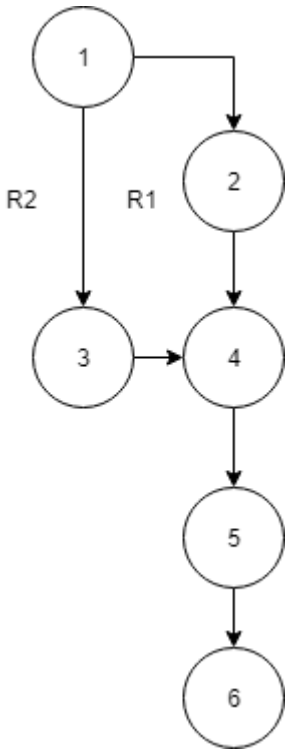
```
Start method mulai_sprint.
obj_nomor_sprint = data sprint yang terakhir dari suatu proyek        (1)
If total obj_nomor_sprint == 0
   nomor_sprint = 1                                                   (2)
else
   nomor_sprint = obj_nomor_sprint[0]->nomor_sprint                   (3)
End if  (4)
today = tanggal hari ini
tgl_selesai = tanggal hari ini + lama waktu sprint                    (5)
Simpan sprint baru dalam database.
```

Simpan riwayat baru dalam database.
new_sprint = data sprint yang terakhir dari suatu proyek
update tugas, rubah data id sprint menjadi new_sprint[0]->id_sprint,
dimana jika nilai id_sprint = null dan is_backlog = 0.
Kembali kehalaman backlog.
End. (6)

(5)

2. *Basis Path Testing*

1. *Flow Graph*



Gambar 6.2 *Flow Graph Method* mulai_sprint

b. *Cyclomatic Complexity*

i. V(G) = 2, terdapat 2 region

ii. V(G) = 6 edges – 6 nodes + 2 = 2

iii. V(G) = 1 predicate nodes + 1 = 2

c. *Independent Path*

i. Jalur 1 = 1 – 2 – 4 – 5 – 6

ii. Jalur 2 = 1 – 3 – 4 – 5 – 6

Tabel 6.4 Hasil Pengujian Unit *Method* mulai_sprint

No.	No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1	total data variabel obj_nomor_sprint = 0	Sprint akan berjalan dan nomor sprint bernilai 1	Sprint berhasil berjalan dan nomor sprint bernilai 1	Valid
2.	2	total data variabel obj_nomor_sprint = 2	Sprint akan berjalan dan nomor sprint adalah nilai atribut nomor_sprint pada variabel obj_nomor_sprint ditambah 1	Sprint berhasil berjalan dan nomor sprint adalah nilai atribut nomor_sprint pada variabel obj_nomor_sprint ditambah 1	Valid

6.1.3 Pengujian Unit Method backlog

a. Pseudocode

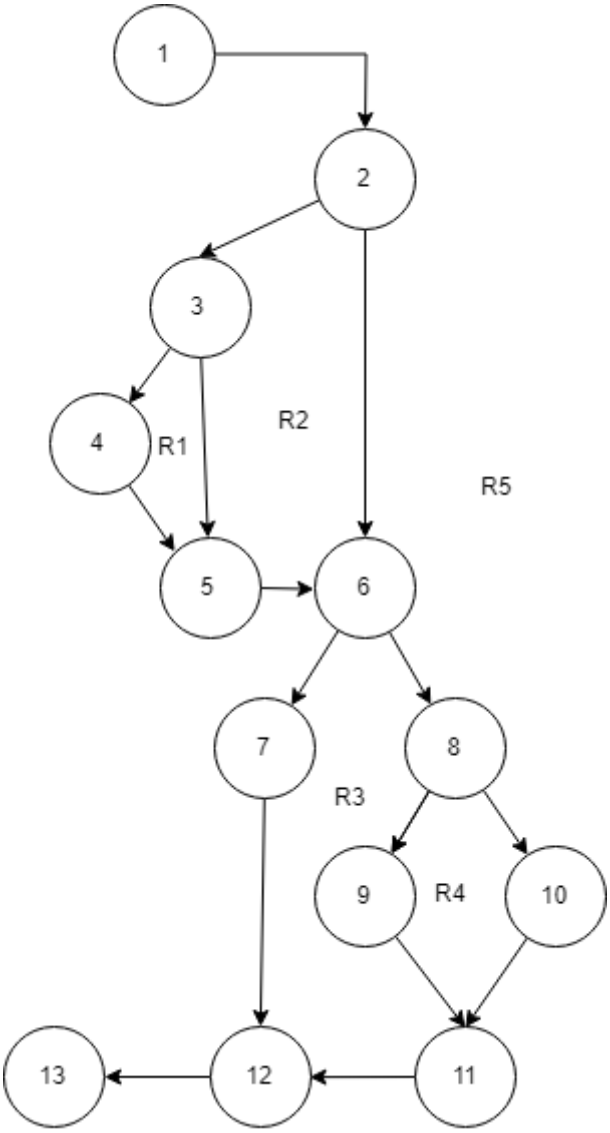
Tabel 6.5 *Pseudocode* Pengujian *Method* backlog

```
Start method backlog.
nomor_sprint = data sprint yang terakhir dari suatu proyek
where1 = array untuk kriteria pengambilan data tugas.
where2 = array untuk kriteria pengambilan data tugas.
where3 = array untuk kriteria pengambilan data tugas.
where_backlog = array untuk kriteria pengambilan data tugas.
id_modul = 0;                                                    (1)
if atribut id_modul dari form berisi suatu nilai                 (2)
  if atribut id_modul dari form bernilai != “all”                (3)
     where1['id_modul'] = nilai atribut id_modul dari form submit
     where2['id_modul'] = nilai atribut id_modul dari form submit
     where3['id_modul'] = nilai atribut id_modul dari form submit
     where_backlog['id_modul'] = nilai atribut id_modul dari form
     submit
     id_modul = nilai atribut id_modul dari form submit          (4)
  endif                                                          (5)
```

```
endif
if total data variabel nomor_sprint = 0                    (6)
  sprint = data tugas dengan kriteria variabel where1      (7)
else   (8)
  if nilai tgl_selesai data ke 0 variabel nomor_sprint = waktu saat ini
    sprint = data tugas dengan kriteria variabel where2    (9)
  else
    sprint = data tugas dengan kriteria variabel where3    (10)
  endif   (11)
endif     (12)
modul = mengambil data modul pada proyek yang dibuka
backlog = data tugas dengan kriteria variabel where_backlog
Kembali kehalaman backlog dengan membawa nilai variabel
nomor_sprint, sprint, backlog, modul, dan id_modul.        (13)
End
```

b. Basis Path Testing

a. Flow Graph



Gambar 6.3 *Flow Graph Method* backlog

b. Cyclomatic Complexity

a) V(G) = 5, terdapat 5 region

b) V(G) = 16 edges – 13 nodes + 2 = 5

a) V(G) = 4 predicate nodes + 1 = 5

c. Independent Path

a) Jalur 1 = 1 – 2 – 6 – 7 – 12 – 13

b) Jalur 2 = 1 – 2 – 6 – 8 – 9 – 11 – 12 – 13

c) Jalur 3 = 1 – 2 – 6 – 8 – 10 – 11 – 12 – 13

d) Jalur 4 = 1 – 2 – 3 – 5 – 6 – 7 – 12 – 13

iii. Jalur 5 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 12 – 13

Tabel 6.6 Hasil Pengujian Unit *Method* backlog

No	No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1	Filter modul = "Semua". Total variabel nomor_sprint = 0.	*Method* akan inisialisasi variabel sprint menggunakan data Tugas dengan kriteria variabel where1	*Method* berhasil inisialisasi variabel sprint menggunakan data Tugas dengan kriteria variabel where1	Valid
2.	2	Filter modul = "Semua". Total variabel nomor_sprint tidak sama dengan 0. Nilai tgl_selesai data ke 0 variabel nomor_sprint >= waktu saat ini	*Method* akan inisialisasi variabel sprint menggunakan data Tugas dengan kriteria variabel where2	*Method* berhasil inisialisasi variabel sprint menggunakan data Tugas dengan kriteria variabel where2	Valid
3.	3	Filter modul = "Semua".. Total variabel nomor_sprint tidak sama dengan 0. Nilai tgl_selesai data ke 0 variabel nomor_sprint kurang dari waktu saat ini	*Method* akan inisialisasi variabel sprint menggunakan data Tugas dengan kriteria variabel where3	*Method* berhasil inisialisasi variabel sprint menggunakan data Tugas dengan kriteria variabel where3	Valid
4.	4	Field id_modul dari form bernilai "all". Total variabel nomor_sprint = 0.	Aplikasi akan menampilkan halaman backlog, namun tidak ada sprint yang berjalan dan terdapat tombol untuk memulai sprint. Serta menampilkan	Aplikasi berhasil menampilkan halaman backlog, namun tidak ada sprint yang berjalan dan terdapat tombol untuk memulai sprint. Serta menampilkan	Valid

Tabel 6.6 Hasil Pengujian Unit *Method* backlog

No	No. Jalur	Prosedur Uji	Expected Result	Result	Status
			tugas dengan modul apapun.	tugas dengan modul apapun.	
5.	5	Field id_modul dari form tidak bernilai "all". Total variabel nomor_sprint = 0.	Aplikasi akan menampilkan halaman backlog, namun tidak ada sprint yang berjalan dan terdapat tombol untuk memulai sprint. Serta menampilkan tugas dengan modul sesuai yang dipilih ketika menekan tombol fiter.	Aplikasi berhasil menampilkan halaman backlog, namun tidak ada sprint yang berjalan dan terdapat tombol untuk memulai sprint. Serta menampilkan tugas dengan modul sesuai yang dipilih ketika menekan tombol fiter.	Valid

6.1.4 Pengujian Unit Method ubah_status_backlog

a. Pseudocode

Tabel 6.7 *Pseudocode* Pengujian *Method* ubah_status_backlog

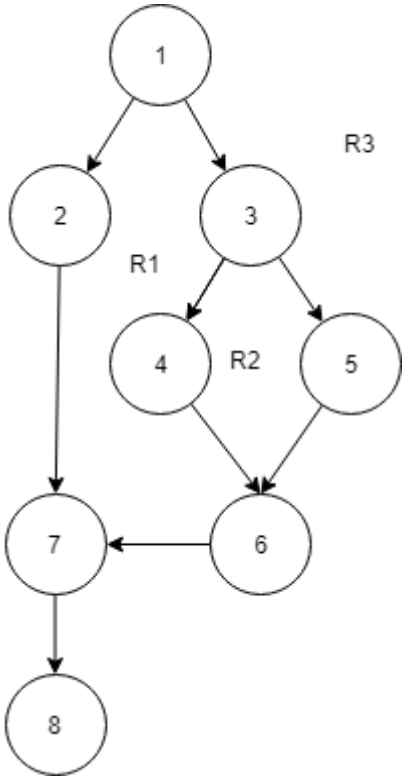
```
Start method ubah_status_backlog.
last_sprint = data sprint yang terakhir dari suatu proyek        ┐
if total data variabel last_sprint = 0                           ┘ (1)
  Update data tugas, ganti is_backlog = 0, pada data dengan id_tugas =
  Nilai form submit field id_tugas.        (2)
else    (3)
  if nilai tgl_selesai dari data ke 0 variabel last_sprint = waktu saat ini
    Update data tugas is_backlog = 0 dan id_sprint = nilai last_sprint,   ┐
    pada data dengan id_tugas = Nilai form submit field id_tugas.         ┘ (4)
  else
    Update data tugas, ganti is_backlog = 0, pada data dengan             ┐
    id_tugas = Nilai form submit field id_tugas.                           ┘ (5)
  endif   (6)
```

endif (7)

Kembali kehalaman backlog
End
(8)

b. Basis Path Testing

a. Flow Graph



Gambar 6.4 *Flow Graph Method* ubah_status_backlog

b. Cyclomatic Complexity

a) V(G) = 3, terdapat 3 region
b) V(G) = 9 edges – 8 nodes + 2 = 3
c) V(G) = 2 predicate nodes + 1 = 3

c. Independent Path

a) Jalur 1 = 1 – 2 – 7 – 8
b) Jalur 2 = 1 – 2 – 3 – 4 – 6 – 7 – 8

iv. Jalur 3 = 1 – 2 – 3 – 5 – 6 – 7 – 8

Tabel 6.8 Hasil Pengujian Unit *Method* ubah_status_backlog

No	No. Jalur	Prosedur Uji	Expected Result	Result	Status

1.	1	Total variabel last_sprint = 0.	Tugas backlog akan berpindah ke bagian tugas sprint yang mana sedang tidak ada sprint yang berjalan	Tugas backlog berhasil berpindah ke bagian tugas sprint yang mana sedang tidak ada sprint yang berjalan	Valid
2.	2	Total variabel last_sprint tidak sama dengan 0. Nilai tgl_selesai data ke 0 variabel last_sprint >= waktu saat ini	Tugas backlog akan berpindah ke bagian tugas sprint yang mana sprint sedang berjalan	Tugas backlog berhasil berpindah ke bagian tugas sprint yang mana sprint sedang berjalan	Valid
3.	3	Total variabel last_sprint tidak sama dengan 0. Nilai tgl_selesai data ke 0 variabel last_sprint kurang dari waktu saat ini	Tugas backlog akan berpindah ke bagian tugas sprint yang mana sedang tidak ada sprint yang berjalan	Tugas backlog berhasil berpindah ke bagian tugas sprint yang mana sedang tidak ada sprint yang berjalan	Valid

6.1.5 Pengujian Unit Method set_timeline_modul

1. Pseudocode

Tabel 6.9 *Pseudocode* Pengujian *Method* set_timeline_modul

```
Start method set_timeline_modul.
if nilai field tanggal mulai kosong                                  (1)
   errors['tgl_mulai'] = pesan kesalahan                             (2)
endif
if nilai field tanggal selesai kosong                                (3)
   errors['tgl_selesai] = pesan kesalahan                            (4)
endif
if total errors > 0                                                  (5)
   Kembali kehalaman semua modul dan manampilkan pesan error         (6)
endif
if nilai field tanggal mulai > nilai field tanggal selesai           (7)
```

Kembali kehalaman semua modul dan menampilkan pesan error (8)

endif

modul = pengambilan data modul dengan id dari nilai field id_modul

Update data modul tgl_mulai = nilai dari field tgl_mulai dan tgl_selesai = nilai dari field tgl_selesai, pada data dengan id_modul = nilai field id_modul (9)

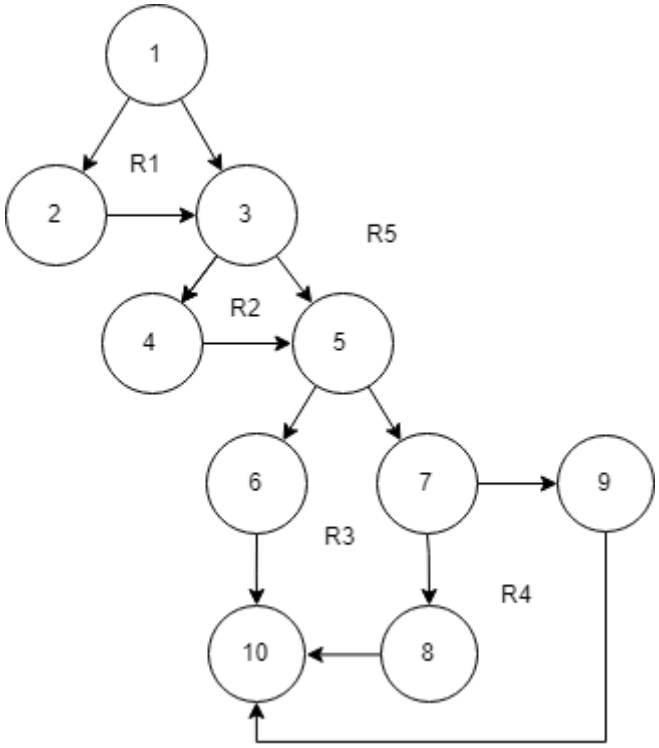
Simpan data riwayat perubahan pada database

Kembali ke halaman semua modul

End (10)

2. Basis Path Testing

a. Flow Graph



Gambar 6.5 *Flow Graph Method* set_timeline_modul

b. Cyclomatic Complexity

a) V(G) = 5, terdapat 5 region

b) V(G) = 13 edges – 10 nodes + 2 = 5

c) V(G) = 4 predicate nodes + 1 = 5

c. Independent Path

a) Jalur 1 = 1 – 3 – 5 – 7 – 9 – 10

b) Jalur 2 = 1 – 3 – 5 – 7 – 8 – 10

c) Jalur 3 = 1 – 2 – 3 – 5 – 6 – 10

d) Jalur 4 = 1 – 3 – 4 – 5 – 6 – 10

e) Jalur 5 = 1 – 2 – 3 – 4 – 5 – 6 – 10

Tabel 6.10 Hasil Pengujian Unit Method set_timeline_modul

No	No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1	tgl_mulai = “08/07/2019”. tgl_selesai = “30/07/2019”.	Aplikasi akan merubah timeline modul tersebut dan memuat halaman semua modul.	Aplikasi berhasil merubah timeline modul tersebut dan memuat halaman semua modul.	Valid
2.	2	tgl_mulai = “30/07/2019”. tgl_selesai “08/07/2019”.	Aplikasi akan gagal melakukan perubahan timeline dan akan menampilkan pesan eror bahwa nilai tgl mulai harus lebih dari tgl selesai.	Aplikasi gagal melakukan perubahan timeline dan menampilkan pesan eror bahwa nilai tgl mulai harus lebih dari tgl selesai.	Valid
3.	3	tgl_mulai kosong. tgl_selesai = “08/07/2019”.	Aplikasi akan gagal melakukan perubahan timeline dan akan menampilkan pesan eror bahwa nilai tgl mulai tidak boleh kosong.	Aplikasi gagal melakukan perubahan timeline dan menampilkan pesan eror bahwa nilai tgl mulai tidak boleh kosong.	Valid
4.	4	tgl_mulai =“08/07/2019”. tgl_selesai = “”.	Aplikasi akan gagal melakukan perubahan timeline dan akan menampilkan pesan eror bahwa nilai tgl selesai tidak boleh kosong.	Aplikasi gagal melakukan perubahan timeline dan menampilkan pesan eror bahwa nilai tgl selesai tidak boleh kosong.	Valid

Tabel 6.10 Hasil Pengujian Unit Method set_timeline_modul

No	No. Jalur	Prosedur Uji	Expected Result	Result	Status
5.	5	tgl_mulai kosong. tgl_selesai kosong.	Aplikasi akan gagal melakukan perubahan timeline dan akan menampilkan pesan eror bahwa nilai tgl mulai dan tgl selesai tidak boleh kosong.	Aplikasi gagal melakukan perubahan timeline dan menampilkan pesan eror bahwa nilai tgl mulai dan tgl selesai tidak boleh kosong.	Valid

6.2 Pengujian Integrasi

Pengujian integrasi dilakukan dengan menguji unit yang berinteraksi dalam sistem untuk dapat menghasilkan suatu fungsional. Pengujian integrasi ini dilakukan dengan menggunakan beberapa sampel dari kebutuhan fungsional menambah komentar. Untuk menjalankan fungsional ini, *method* beri_komentar pada kelas ProyekController terintegrasi dengan *method* create yang terdapat pada kelas M_komentar.

Tabel 6.11 Tabel Pengujian Integrasi

Input Pertama	Method dari kelas ProyekController	Output Pertama / Input Kedua	Method dari kelas M_komentar	Expected Result	Result	Status
Komentar tentang suatu tugas	beri_komentar ()	Komentar tentang suatu tugas	create()	Sistem menyimpan komentar dan memuat ulang halaman detail tugas yang dikomentari oleh pengguna	Sistem berhasil menyimpan komentar dan memuat ulang halaman detail tugas yang dikomentari oleh pengguna	Valid

Tabel 6.12 *Source Code* Method beri_komentar

```
public function beri_komentar(Request $req){
    $this->validate($req, [
        'komentar'  => 'required'
    ]);
    M_komentar::create([
        'id_pengguna'   => Auth::user()->id,
        'isi'           => $req->komentar,
        'id_tugas'      => $req->id_tugas
    ]);
    $tugas = M_tugas::find($req->id_tugas);

    $this->simpan_riwayat("berkomentar pada tugas $tugas->nama_tugas", $req->id_tugas);
    return back();
}
```

Tabel 6.12 merupakan *source code* dari *method* beri_komentar. Di dalamnya terdapat perintah untuk mengakses *method create* yang terdapat pada kelas M_komentar. *Method create* tersebut digunakan untuk menyimpan komentar pada *database*.

6.3 Pengujian Validasi

Pengujian validasi adalah pengujian dalam proses memastikan bahwa perangkat lunak yang dikembangkan sudah benar sesuai dengan kebutuhan yang telah didefinisikan sebelumnya. Pengujian ini dilakukan dengan metode black-box dengan jenis scenario-based testing.

6.3.1 Pengujian Validasi Register

1. Kasus Uji Register

Tabel 6.13 Kasus Uji Register

Nama Kasus Uji	Kasus Uji Mendaftar
Kode Kebutuhan	MMPS_F_01
Prosedur	1. Mengakses halaman register 2. Memasukkan nama = “Andri Wijaya” 3. Memasukkan email = “wijayakusuma@gmail.com” 4. Memasukkan password = “andri123” 5. Memasukkan confirm password= “andri123”

	6. Menekan tombol Register
Hasil yang diharapkan	Aplikasi akan menampilkan pesan berhasil daftar
Hasil	Aplikasi menampilkan pesan berhasil daftar
Status	Valid

2. Kasus Uji Register Alternatif 1

Tabel 6.14 Kasus Uji Register Alternatif 1

Nama Kasus Uji	Kasus Uji Mendaftar Alternatif 1
Kode Kebutuhan	MMPS_F_01
Prosedur	1. Mengakses halaman register 2. Tidak memasukkan nama 3. Memasukkan email = “andri@gmail.com” 4. Memasukkan password = “andri123” 5. Memasukkan confirm password= “andri123” 6. Menekan tombol Register
Hasil yang diharapkan	Pendaftaran gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* nama
Hasil	Pendaftaran gagal dan aplikasi menampilkan pesan untuk mengisi *field* nama

3. Kasus Uji Register Alternatif 2

Tabel 6.15 Kasus Uji Register Alternatif 2

Nama Kasus Uji	Kasus Uji Mendaftar Alternatif 2
Kode Kebutuhan	MMPS_F_01
Prosedur	1. Mengakses halaman register 2. Memasukkan nama = “Andri Wijaya” 3. Tidak memasukkan email. 4. Memasukkan password = “andri123” 5. Memasukkan confirm password= “andri123” 6. Menekan tombol Register
Hasil yang diharapkan	Pendaftaran gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* email.
Hasil	Pendaftaran gagal dan aplikasi menampilkan pesan untuk mengisi *field* email.

4. Kasus Uji Register Alternatif 3

Tabel 6.16 Kasus Uji Register Alternatif 3

Nama Kasus Uji	Kasus Uji Mendaftar Alternatif 3
Kode Kebutuhan	MMPS_F_01
Prosedur	1. Mengakses halaman register 2. Memasukkan nama = “Andri Wijaya” 3. Memasukkan email = “andri@” 4. Memasukkan password = “andri123” 5. Memasukkan confirm password= “andri123” 6. Menekan tombol Register
Hasil yang diharapkan	Pendaftaran gagal dan aplikasi akan menampilkan pesan bahwa *field* email harus sesuai dengan format email.
Hasil	Pendaftaran gagal dan aplikasi menampilkan pesan bahwa *field* email harus sesuai dengan format email.

5. Kasus Uji Register Alternatif 4

Tabel 6.17 Kasus Uji Register Alternatif 4

Nama Kasus Uji	Kasus Uji Mendaftar Alternatif 4
Kode Kebutuhan	MMPS_F_01
Prosedur	1. Mengakses halaman register 2. Memasukkan nama = “Andri Wijaya” 3. Memasukkan email = “andri@gmail.com” 4. Memasukkan password = “andri123” 5. Memasukkan confirm password= “andri123” 6. Menekan tombol Register
Hasil yang diharapkan	Pendaftaran gagal dan aplikasi akan menampilkan pesan bahwa email sudah digunakan.
Hasil	Pendaftaran gagal dan aplikasi menampilkan pesan bahwa email sudah digunakan.

6. Kasus Uji Register Alternatif 5

Tabel 6.18 Kasus Uji Register Alternatif 5

Nama Kasus Uji	Kasus Uji Mendaftar Alternatif 5
Kode Kebutuhan	MMPS_F_01

Prosedur	1. Mengakses halaman register 2. Memasukkan nama = “Andri Wijaya” 3. Memasukkan email = “wijaya@mail.com. 4. Tidak memasukkan password. 5. Memasukkan confirm password= “andri123” 6. Menekan tombol Register
Hasil yang diharapkan	Pendaftaran gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* password.
Hasil	Pendaftaran gagal dan aplikasi menampilkan pesan untuk mengisi *field* password.

7. Kasus Uji Register Alternatif 6

Tabel 6.19 Kasus Uji Register Alternatif 6

Nama Kasus Uji	Kasus Uji Mendaftar Alternatif 6
Kode Kebutuhan	MMPS_F_01
Prosedur	1. Mengakses halaman register 2. Memasukkan nama = “Andri Wijaya” 3. Memasukkan email = “wijaya@mail.com. 4. Memasukkan password = ”12345” 5. Memasukkan confirm password= “12345” 6. Menekan tombol Register
Hasil yang diharapkan	Pendaftaran gagal dan aplikasi akan menampilkan pesan bahwa *field* password minimal 6 karakter.
Hasil	Pendaftaran gagal dan aplikasi menampilkan pesan bahwa *field* password minimal 6 karakter.

8. Kasus Uji Register Alternatif 7

Tabel 6.20 Kasus Uji Register Alternatif 7

Nama Kasus Uji	Kasus Uji Mendaftar Alternatif 7
Kode Kebutuhan	MMPS_F_01
Prosedur	1. Mengakses halaman register 2. Memasukkan nama = “Andri Wijaya” 3. Memasukkan email = “wijaya@mail.com. 4. Memasukkan password = ”123456” 5. Tidak memasukkan confirm password.

	6. Menekan tombol Register
Hasil yang diharapkan	Pendaftaran gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* confirm password.
Hasil	Pendaftaran gagal dan aplikasi menampilkan pesan untuk mengisi *field* confirm password.

9. Kasus Uji Register Alternatif 8

Tabel 6.21 Kasus Uji Register Alternatif 8

Nama Kasus Uji	Kasus Uji Mendaftar Alternatif 8
Kode Kebutuhan	MMPS_F_01
Prosedur	1. Mengakses halaman register 2. Memasukkan nama = “Andri Wijaya” 3. Memasukkan email = “wijaya@mail.com. 4. Memasukkan password = ”123456” 5. Memasukkan confirm password= “12345a” 6. Menekan tombol Register
Hasil yang diharapkan	Pendaftaran gagal dan aplikasi akan menampilkan pesan bahwa *field* confirm password harus sama dengan *field* password.
Hasil	Pendaftaran gagal dan aplikasi menampilkan pesan bahwa *field* confirm password harus sama dengan *field* password.

6.3.2 Pengujian Validasi Login

1. Kasus Uji Login

Tabel 6.22 Kasus Uji Login

Nama Kasus Uji	Kasus Uji Login
Kode Kebutuhan	MMPS_F_02
Prosedur	1. Mengakses halaman login 2. Memasukkan email = “wijayakusuma@gmail.com”. 3. Memasukkan password = ”andri123” 4. Menekan tombol Login
Hasil yang diharapkan	Aplikasi akan menampilkan halaman semua proyek

Hasil	Aplikasi menampilkan halaman semua proyek
Status	Valid

2. Kasus Uji Login Alternatif 1

Tabel 6.23 Kasus Uji Login Alternatif 1

Nama Kasus Uji	Kasus Uji Login Alternatif 1
Kode Kebutuhan	MMPS_F_02
Prosedur	1. Mengakses halaman login 2. Memasukkan email = “wijayakusuma@gmail.com”. 3. Memasukkan password = ”123456” 4. Menekan tombol Login
Hasil yang diharapkan	Aplikasi akan menampilkan pesan bahwa email atau password salah.
Hasil	Aplikasi menampilkan pesan bahwa email atau password salah.
Status	Valid

6.3.3 Pengujian Validasi Melihat Daftar Proyek

1. Kasus Uji Melihat Daftar Proyek

Tabel 6.24 Kasus Uji Melihat Daftar Proyek

Nama Kasus Uji	Kasus Uji Melihat Daftar Proyek
Kode Kebutuhan	MMPS_F_03
Prosedur	1. Melakukan proses login dengan berhasil
Hasil yang diharapkan	Aplikasi akan menampilkan halaman semua proyek
Hasil	Aplikasi menampilkan halaman semua proyek
Status	Valid

6.3.4 Pengujian Validasi Membuat Proyek

1. Kasus Uji Membuat Proyek

Tabel 6.25 Kasus Uji Membuat Proyek

Nama Kasus Uji	Kasus Uji Membuat Proyek
Kode Kebutuhan	MMPS_F_04
Prosedur	1. Mengakses halaman semua proyek

	2. Menekan tombol buat proyek 3. Memasukkan nama proyek = “Proyek Aplikasi Manajemen menggunakan Scrum”. 4. Memasukkan deskripsi = “Program berbasis web untuk memudahkan pengguna dalam memanajemen proyek proyek yang dikerjakan menggunakan metode Scrum”. 5. Menekan tombol Buat.
Hasil yang diharapkan	Proyek berhasil ditambahkan dan aplikasi akan memuat ulang halaman semua proyek.
Hasil	Proyek berhasil ditambahkan dan aplikasi memuat ulang halaman semua proyek.
Status	Valid

2. Kasus Uji Membuat Proyek Alternatif 1

Tabel 6.26 Kasus Uji Membuat Proyek Alternatif 1

Nama Kasus Uji	Kasus Uji Membuat Proyek Alternatif 1
Kode Kebutuhan	MMPS_F_04
Prosedur	1. Mengakses halaman semua proyek 2. Menekan tombol buat proyek 3. Tidak memasukkan nama proyek. 4. Tidak Memasukkan deskripsi. 5. Menekan tombol Buat.
Hasil yang diharapkan	Proyek gagal dibuat dan aplikasi akan memberi informasi bahwa pembuatan proyek gagal.
Hasil	Proyek gagal dibuat dan aplikasi memberi informasi bahwa pembuatan proyek gagal.
Status	Valid

6.3.5 Pengujian Validasi Melihat Daftar Modul

1. Kasus Uji Melihat Daftar Modul

Tabel 6.27 Kasus Uji Melihat Daftar Modul

Nama Kasus Uji	Kasus Uji Melihat Daftar Modul
Kode Kebutuhan	MMPS_F_05
Prosedur	1. Mengakses halaman semua proyek.

	2. Memilih salah satu proyek.
Hasil yang diharapkan	Aplikasi akan menampilkan halaman modul
Hasil	Aplikasi menampilkan halaman modul
Status	Valid

6.3.6 Pengujian Validasi Merubah Timeline Modul

1. Kasus Uji Merubah Timeline Modul

Tabel 6.28 Kasus Uji Merubah Timeline Modul

Nama Kasus Uji	Kasus Uji Merubah Timeline Modul
Kode Kebutuhan	MMPS_F_06
Prosedur	1. Mengakses halaman daftar modul. 2. Memasukkan tanggal mulai = “01/05/2019”. 3. Memasukkan tanggal selesai = “14/05/2019”. 4. Menekan tombol Set.
Hasil yang diharapkan	Timeline modul berhasil dirrubah dan aplikasi akan memuat ulang halaman modul
Hasil	Timeline modul berhasil dirrubah dan aplikasi memuat ulang halaman modul
Status	Valid

2. Kasus Uji Merubah Timeline Modul Alternatif 1

Tabel 6.29 Kasus Uji Merubah Timeline Modul Alternatif 1

Nama Kasus Uji	Kasus Uji Merubah Timeline Modul Alternatif 1
Kode Kebutuhan	MMPS_F_06
Prosedur	1. Mengakses halaman daftar modul. 2. Tidak memasukkan tanggal mulai. 3. Memasukkan tanggal selesai = “14/05/2019”. 4. Menekan tombol Set.
Hasil yang diharapkan	Perubahan gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* tanggal mulai.
Hasil	Perubahan gagal dan aplikasi menampilkan pesan untuk mengisi *field* tanggal mulai.
Status	Valid

3. Kasus Uji Merubah Timeline Modul Alternatif 2

Tabel 6.30 Kasus Uji Merubah Timeline Modul Alternatif 2

Nama Kasus Uji	Kasus Uji Merubah Timeline Modul Alternatif 2
Kode Kebutuhan	MMPS_F_06
Prosedur	1. Mengakses halaman daftar modul. 2. Memasukkan tanggal mulai = “01/05/2019”. 3. Tidak memasukkan tanggal selesai. 4. Menekan tombol Set.
Hasil yang diharapkan	Perubahan gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* tanggal selesai.
Hasil	Perubahan gagal dan aplikasi menampilkan pesan untuk mengisi *field* tanggal selesai.
Status	Valid

4. Kasus Uji Merubah Timeline Modul Alternatif 3

Tabel 6.31 Kasus Uji Merubah Timeline Modul Alternatif 3

Nama Kasus Uji	Kasus Uji Merubah Timeline Modul Alternatif 3
Kode Kebutuhan	MMPS_F_06
Prosedur	1. Mengakses halaman daftar modul. 2. Memasukkan tanggal mulai = “01/05/2019”. 3. Memasukkan tanggal selesai = “20/04/2019”. 4. Menekan tombol Set.
Hasil yang diharapkan	Perubahan gagal dan aplikasi akan menampilkan pesan bahwa tanggal selesai harus setelah tanggal mulai.
Hasil	Perubahan gagal dan aplikasi akan menampilkan pesan bahwa tanggal selesai harus setelah tanggal mulai.
Status	Valid

6.3.7 Pengujian Validasi Membuat Modul

1. Kasus Uji Membuat Modul

Tabel 6.32 Kasus Uji Membuat Modul

Nama Kasus Uji	Kasus Uji Membuat Modul
Kode Kebutuhan	MMPS_F_07
Prosedur	1. Mengakses halaman daftar modul.

	2. Memasukkan nama modul = “Pengelolaan proyek”. 3. Menekan tombol Add.
Hasil yang diharapkan	Modul berhasil ditambahkan dan aplikasi akan memuat ulang halaman daftar modul
Hasil	Modul berhasil ditambahkan dan aplikasi akan memuat ulang halaman daftar modul
Status	Valid

2. Kasus Uji Membuat Modul Alternatif 1

Tabel 6.33 Kasus Uji Membuat Modul Alternatif 1

Nama Kasus Uji	Kasus Uji Membuat Modul Alternatif 1
Kode Kebutuhan	MMPS_F_07
Prosedur	1. Mengakses halaman daftar modul. 2. Tidak memasukkan nama modul. 3. Menekan tombol Add.
Hasil yang diharapkan	Pembuatan gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* nama modul.
Hasil	Pembuatan gagal dan aplikasi menampilkan pesan untuk mengisi *field* nama modul.
Status	Valid

6.3.8 Pengujian Validasi Mengubah Data Modul

1. Kasus Uji Mengubah Data Modul

Tabel 6.34 Kasus Uji Mengubah Data Modul

Nama Kasus Uji	Kasus Uji Mengubah Data Modul
Kode Kebutuhan	MMPS_F_08
Prosedur	1. Mengakses halaman daftar modul. 2. Menekan salah satu nama modul. 3. Memasukkan nama baru modul = “Pengelolaan proyek baru”. 4. Tidak memasukkan deskripsi modul. 5. Menekan tombol Edit.
Hasil yang diharapkan	Modul berhasil diperbaruhi dan aplikasi akan memuat ulang halaman daftar modul

Hasil	Modul berhasil diperbaruhi dan aplikasi akan memuat ulang halaman daftar modul
Status	Valid

2. Kasus Uji Mengubah Data Modul Alternatif 1

Tabel 6.35 Kasus Uji Mengubah Data Modul Alternatif 1

Nama Kasus Uji	Kasus Uji Membuat Modul Alternatif 1
Kode Kebutuhan	MMPS_F_08
Prosedur	1. Mengakses halaman daftar modul. 2. Menekan salah satu nama modul. 3. Tidak memasukkan nama baru modul. 4. Tidak memasukkan deskripsi modul. 5. Menekan tombol Edit.
Hasil yang diharapkan	Aplikasi akan menampilkan peringatan untuk mengisi *field* nama modul.
Hasil	Aplikasi akan menampilkan peringatan untuk mengisi *field* nama modul.
Status	Valid

6.3.9 Pengujian Validasi Menghapus Modul

1. Kasus Uji Menghapus Modul

Tabel 6.36 Kasus Uji Menghapus Modul

Nama Kasus Uji	Kasus Uji Menghapus Modul
Kode Kebutuhan	MMPS_F_09
Prosedur	1. Mengakses halaman daftar modul. 2. Menekan salah satu nama modul. 3. Menekan tombol Hapus.
Hasil yang diharapkan	Modul berhasil dihapus dan aplikasi akan memuat ulang halaman daftar modul
Hasil	Modul berhasil dihapus dan aplikasi akan memuat ulang halaman daftar modul
Status	Valid

6.3.10 Pengujian Validasi Mengubah Data Tugas

1. Kasus Uji Mengubah Data Tugas

Tabel 6.37 Kasus Uji Mengubah Data Tugas

Nama Kasus Uji	Kasus Uji Mengubah Data Tugas
Kode Kebutuhan	MMPS_F_10
Prosedur	1. Mengakses halaman Backlog. 2. Memilih option Edit pada salah satu tugas. 3. Memasukkan nama tugas = "Membuat back end dan front end untuk dashboard proyek". 4. Memilih Modul "Dashboard Proyek". 5. Menekan tombol Edit.
Hasil yang diharapkan	Tugas berhasil diperbaruhi dan aplikasi akan memuat ulang halaman backlog.
Hasil	Tugas berhasil diperbaruhi dan aplikasi memuat ulang halaman backlog.
Status	Valid

2. Kasus Uji Mengubah Data Tugas Alternatif 1

Tabel 6.38 Kasus Uji Mengubah Data Tugas Alternatif 1

Nama Kasus Uji	Kasus Uji Mengubah Data Tugas Alternatif 1
Kode Kebutuhan	MMPS_F_10
Prosedur	1. Mengakses halaman Backlog. 2. Memilih option Edit pada salah satu tugas. 3. Tidak memasukkan nama tugas. 4. Memilih Modul "-". 5. Menekan tombol Edit.
Hasil yang diharapkan	Aplikasi akan menampilkan peringatan untuk mengisi *field* nama tugas.
Hasil	Aplikasi akan menampilkan peringatan untuk mengisi *field* nama tugas.
Status	Valid

6.3.11 Pengujian Validasi Menghapus Tugas

1. Kasus Uji Menghapus Tugas

Tabel 6.39 Kasus Uji Menghapus Tugas

Nama Kasus Uji	Kasus Uji Menghapus Tugas
Kode Kebutuhan	MMPS_F_11

Prosedur	1. Mengakses halaman Backlog. 2. Memilih option Edit pada salah satu tugas. 3. Menekan tombol Hapus.
Hasil yang diharapkan	Tugas berhasil dihapus dan aplikasi akan memuat ulang halaman Backlog.
Hasil	Tugas berhasil dihapus dan aplikasi memuat ulang halaman Backlog.
Status	Valid

6.3.12 Pengujian Validasi Melihat Daftar Tugas Backlog

1. Kasus Uji Melihat Daftar Tugas Backlog

Tabel 6.40 Kasus Uji Melihat Daftar Tugas Backlog

Nama Kasus Uji	Kasus Uji Melihat Daftar Tugas Backlog
Kode Kebutuhan	MMPS_F_12
Prosedur	1. Menekan Backlog pada navigasi di Aplikasi.
Hasil yang diharapkan	Aplikasi akan menampilkan halaman backlog
Hasil	Aplikasi menampilkan halaman backlog
Status	Valid

6.3.13 Pengujian Validasi Membuat Tugas Backlog

1. Kasus Uji Membuat Tugas Backlog

Tabel 6.41 Kasus Uji Membuat Tugas Backlog

Nama Kasus Uji	Kasus Uji Membuat Tugas Backlog
Kode Kebutuhan	MMPS_F_13
Prosedur	1. Mengakses halaman backlog. 2. Memasukkan nama tugas = “Membuat proyek” pada bagian tambah tugas backlog. 3. Menekan tombol Add.
Hasil yang diharapkan	Tugas backlog berhasil ditambahkan dan aplikasi akan memuat ulang halaman backlog.
Hasil	Tugas backlog berhasil ditambahkan dan aplikasi memuat ulang halaman backlog.
Status	Valid

2. Kasus Uji Membuat Tugas Backlog Alternatif 1

Tabel 6.42 Kasus Uji Membuat Tugas Backlog Alternatif 1

Nama Kasus Uji	Kasus Uji Membuat Tugas Backlog Alternatif 1
Kode Kebutuhan	MMPS_F_13
Prosedur	1. Mengakses halaman backlog. 2. Tidak memasukkan nama tugas pada bagian tambah tugas backlog. 3. Menekan tombol Add.
Hasil yang diharapkan	Pembuatan gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* nama tugas.
Hasil	Pembuatan gagal dan aplikasi menampilkan pesan untuk mengisi *field* nama tugas.
Status	Valid

6.3.14 Pengujian Validasi Mengubah Posisi Tugas Backlog

1. Kasus Uji Mengubah Posisi Tugas Backlog

Tabel 6.43 Kasus Uji Mengubah Posisi Tugas Backlog

Nama Kasus Uji	Kasus Uji Mengubah Posisi Tugas Backlog
Kode Kebutuhan	MMPS_F_14
Prosedur	1. Mengakses halaman backlog. 2. Menyeret nama tugas backlog dan meletakkannya pada bagian tugas sprint.
Hasil yang diharapkan	Tugas backlog berhasil berubah posisi dan akan berada pada bagian tugas sprint.
Hasil	Tugas backlog berubah posisi dan berada pada bagian tugas sprint.
Status	Valid

2. Kasus Uji Mengubah Posisi Tugas Backlog Alternatif 1

Tabel 6.44 Kasus Uji Mengubah Posisi Tugas Backlog Alternatif 1

Nama Kasus Uji	Kasus Uji Mengubah Posisi Tugas Backlog Alternatif 1
Kode Kebutuhan	MMPS_F_14
Prosedur	1. Mengakses halaman backlog. 2. Menekan pilihan Move to Sprint pada salah satu tugas backlog.

Hasil yang diharapkan	Aplikasi akan memuat ulang halaman backlog dan tugas yang dirubah telah berganti posisi ke bagian tugas sprint.
Hasil	Aplikasi memuat ulang halaman backlog dan tugas yang dirubah berganti posisi ke bagian tugas sprint.
Status	Valid

6.3.15 Pengujian Validasi Melihat Daftar Tugas Sprint

1. Kasus Uji Melihat Daftar Tugas Sprint

Tabel 6.45 Kasus Uji Melihat Daftar Tugas Sprint

Nama Kasus Uji	Kasus Uji Melihat Daftar Tugas Sprint
Kode Kebutuhan	MMPS_F_15
Prosedur	1. Menekan Backlog pada navigasi di Aplikasi.
Hasil yang diharapkan	Aplikasi akan menampilkan halaman backlog
Hasil	Aplikasi menampilkan halaman backlog
Status	Valid

6.3.16 Pengujian Validasi Membuat Tugas Sprint

1. Kasus Uji Membuat Tugas Sprint

Tabel 6.46 Kasus Uji Membuat Tugas Sprint

Nama Kasus Uji	Kasus Uji Membuat Tugas Sprint
Kode Kebutuhan	MMPS_F_16
Prosedur	1. Mengakses halaman backlog. 2. Memasukkan nama tugas = “Membuat modul” pada bagian tambah tugas sprint. 3. Menekan tombol Add.
Hasil yang diharapkan	Tugas sprint berhasil ditambahkan dan aplikasi akan memuat ulang halaman backlog.
Hasil	Tugas sprint berhasil ditambahkan dan aplikasi memuat ulang halaman backlog.
Status	Valid

2. Kasus Uji Membuat Tugas Sprint Alternatif 1

Tabel 6.47 Kasus Uji Membuat Tugas Sprint Alternatif 1

Nama Kasus Uji	Kasus Uji Membuat Tugas Sprint Alternatif 1

Kode Kebutuhan	MMPS_F_16
Prosedur	1. Mengakses halaman backlog. 2. Tidak memasukkan nama tugas pada bagian tambah tugas sprint. 3. Menekan tombol Add.
Hasil yang diharapkan	Pembuatan gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* nama tugas.
Hasil	Pembuatan gagal dan aplikasi menampilkan pesan untuk mengisi *field* nama tugas.
Status	Valid

6.3.17 Pengujian Validasi Mengubah Posisi Tugas Sprint

1. Kasus Uji Mengubah Posisi Tugas Sprint

Tabel 6.48 Kasus Uji Mengubah Posisi Tugas Sprint

Nama Kasus Uji	Kasus Uji Mengubah Posisi Tugas Sprint
Kode Kebutuhan	MMPS_F_17
Prosedur	1. Mengakses halaman backlog. 2. Menyeret nama tugas sprint dan meletakkannya pada bagian tugas backlog.
Hasil yang diharapkan	Tugas sprint berhasil berubah posisi dan akan berada pada bagian tugas backlog.
Hasil	Tugas sprint berubah posisi dan berada pada bagian tugas backlog.
Status	Valid

2. Kasus Uji Mengubah Posisi Tugas Sprint Alternatif 1

Tabel 6.49 Kasus Uji Mengubah Posisi Tugas Sprint Alternatif 1

Nama Kasus Uji	Kasus Uji Mengubah Posisi Tugas Sprint Alternatif 1
Kode Kebutuhan	MMPS_F_14
Prosedur	1. Mengakses halaman backlog. 2. Menekan pilihan Move to Backlog pada salah satu tugas sprint.
Hasil yang diharapkan	Aplikasi akan memuat ulang halaman backlog dan tugas yang dirubah telah berganti posisi ke bagian tugas backlog.

Hasil	Aplikasi memuat ulang halaman backlog dan tugas yang dirubah berganti posisi ke bagian tugas backlog.
Status	Valid

6.3.18 Pengujian Validasi Memulai Sprint

1. Kasus Uji Memulai Sprint

Tabel 6.50 Kasus Uji Memulai Sprint

Nama Kasus Uji	Kasus Uji Memulai Sprint
Kode Kebutuhan	MMPS_F_18
Prosedur	1. Mengakses halaman backlog. 2. Menekan tombol Mulai Sprint. 3. Memasukkan lama waktu sprint = “2 minggu”. 4. Menekan tombol Ok.
Hasil yang diharapkan	Sprint akan mulai berjalan.
Hasil	Sprint mulai berjalan.
Status	Valid

6.3.19 Pengujian Validasi Mengakhiri Sprint

1. Kasus Uji Mengakhiri Sprint

Tabel 6.51 Kasus Uji Mengakhiri Sprint

Nama Kasus Uji	Kasus Uji Mengakhiri Sprint
Kode Kebutuhan	MMPS_F_19
Prosedur	1. Mengakses halaman Board. 2. Menekan tombol Selesai. 3. Menekan tombol Ok.
Hasil yang diharapkan	Sprint akan berhenti berjalan.
Hasil	Sprint berhenti berjalan.
Status	Valid

6.3.20 Pengujian Validasi Melihat Daftar Tugas Sprint pada Scum Board

1. Kasus Uji Melihat Daftar Tugas Sprint pada Scum Board

Tabel 6.52 Kasus Uji Melihat Daftar Tugas Sprint pada Scum Board

Nama Kasus Uji	Kasus Uji Melihat Daftar Tugas Sprint pada Scum Board
Kode Kebutuhan	MMPS_F_20
Prosedur	1. Menekan Board pada navigasi di Aplikasi.
Hasil yang diharapkan	Aplikasi akan menampilkan halaman board.
Hasil	Aplikasi menampilkan halaman board.
Status	Valid

6.3.21 Pengujian Validasi Mengubah Status Tugas Sprint

1. Kasus Uji Mengubah Status Tugas Sprint

Tabel 6.53 Kasus Uji Mengubah Status Tugas Sprint

Nama Kasus Uji	Kasus Uji Mengubah Status Tugas Sprint
Kode Kebutuhan	MMPS_F_21
Prosedur	1. Mengakses halaman Board. 2. Menyeret nama tugas sprint dan meletakkannya pada bagian status lainnya.
Hasil yang diharapkan	Tugas sprint berhasil berubah status dan akan berada pada bagian status yang dituju.
Hasil	Tugas sprint berubah status dan berada pada bagian status yang dituju.
Status	Valid

2. Kasus Uji Mengubah Status Tugas Sprint Alternatif 1

Tabel 6.54 Kasus Uji Mengubah Status Tugas Sprint Alternatif 1

Nama Kasus Uji	Kasus Uji Mengubah Status Tugas Sprint Alternatif 1
Kode Kebutuhan	MMPS_F_21
Prosedur	1. Mengakses halaman Board. 2. Menyeret nama tugas sprint dan meletakkannya diluar bagian atau area dari status yang ada.
Hasil yang diharapkan	Tugas sprint tidak berubah status dan akan kembali pada bagian status semula.
Hasil	Tugas sprint tidak berubah status dan kembali pada bagian status semula.

Status	Valid

6.3.22 Pengujian Validasi Melihat Semua Sprint

1. Kasus Uji Melihat Semua Sprint

Tabel 6.55 Kasus Uji Melihat Semua Sprint

Nama Kasus Uji	Kasus Uji Melihat Semua Sprint
Kode Kebutuhan	MMPS_F_22
Prosedur	1. Menekan All Sprint pada navigasi di Aplikasi.
Hasil yang diharapkan	Aplikasi akan menampilkan halaman Semua Sprint
Hasil	Aplikasi menampilkan halaman Semua Sprint.
Status	Valid

6.3.23 Pengujian Validasi Melihat Daftar Anggota Tim

1. Kasus Uji Melihat Daftar Anggota Tim

Tabel 6.56 Kasus Uji Melihat Daftar Anggota Tim

Nama Kasus Uji	Kasus Uji Melihat Daftar Anggota Tim
Kode Kebutuhan	MMPS_F_23
Prosedur	1. Menekan Projec Setting pada navigasi di Aplikasi.
Hasil yang diharapkan	Aplikasi akan menampilkan halaman yang berisi daftar anggota tim yang mengerjakan proyek tersebut.
Hasil	Aplikasi menampilkan halaman yang berisi daftar anggota tim yang mengerjakan proyek tersebut.
Status	Valid

6.3.24 Pengujian Validasi Menambah Anggota Tim

1. Kasus Uji Menambah Anggota Tim

Tabel 6.57 Kasus Uji Menambah Anggota Tim

Nama Kasus Uji	Kasus Uji Menambah Anggota Tim
Kode Kebutuhan	MMPS_F_24
Prosedur	1. Mengakses halaman project setting. 2. Memasukkan email = “wijaya@mail.com”. 3. Menekan tombol Add.

Hasil yang diharapkan	Anggota baru berhasil ditambahkan dan aplikasi akan memuat ulang halaman project setting.
Hasil	Anggota baru berhasil ditambahkan dan aplikasi akan memuat ulang halaman project setting.
Status	Valid

2. Kasus Uji Menambah Anggota Tim Alternatif 1

Tabel 6.58 Kasus Uji Menambah Anggota Tim Alternatif 1

Nama Kasus Uji	Kasus Uji Menambah Anggota Tim Alternatif 1
Kode Kebutuhan	MMPS_F_24
Prosedur	1. Mengakses halaman project setting. 2. Tidak memasukkan email. 3. Menekan tombol Add.
Hasil yang diharapkan	Penambahan gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* email.
Hasil	Penambahan gagal dan aplikasi menampilkan pesan untuk mengisi *field* nama email.
Status	Valid

3. Kasus Uji Menambah Anggota Tim Alternatif 2

Tabel 6.59 Kasus Uji Menambah Anggota Tim Alternatif 2

Nama Kasus Uji	Kasus Uji Menambah Anggota Tim Alternatif 2
Kode Kebutuhan	MMPS_F_24
Prosedur	1. Mengakses halaman project setting. 2. Memasukkan email = “kusuma@yahoo.com”. 3. Menekan tombol Add.
Hasil yang diharapkan	Penambahan gagal dan aplikasi akan menampilkan pesan bahwa email tidak terdaftar.
Hasil	Penambahan gagal dan aplikasi menampilkan pesan bahwa email tidak terdaftar.
Status	Valid

4. Kasus Uji Menambah Anggota Tim Alternatif 3

Tabel 6.60 Kasus Uji Menambah Anggota Tim Alternatif 3

Nama Kasus Uji	Kasus Uji Menambah Anggota Tim Alternatif 3
Kode Kebutuhan	MMPS_F_24

Prosedur	1. Mengakses halaman project setting. 2. Memasukkan email = “wijayakusuma@gmail.com”. 3. Menekan tombol Add.
Hasil yang diharapkan	Aplikasi akan menampilkan pesan bahwa email telah ditambahkan pada proyek ini sebelumnya.
Hasil	Aplikasi menampilkan pesan bahwa email telah ditambahkan pada proyek ini sebelumnya.
Status	Valid

6.3.25 Pengujian Validasi Melihat Detail Tugas

1. Kasus Uji Melihat Detail Tugas

Tabel 6.61 Kasus Uji Melihat Detail Tugas

Nama Kasus Uji	Kasus Uji Melihat Detail Tugas
Kode Kebutuhan	MMPS_F_25
Prosedur	1. Mengakses halaman Backlog atau Semua Sprint 2. Menekan nama salah satu tugas.
Hasil yang diharapkan	Aplikasi akan mengarahkan ke halaman detail tugas.
Hasil	Aplikasi mengarahkan ke halaman detail tugas.
Status	Valid

6.3.26 Pengujian Validasi Melihat Komentar

1. Kasus Uji Melihat Komentar

Tabel 6.62 Kasus Uji Melihat Komentar

Nama Kasus Uji	Kasus Uji Melihat Komentar
Kode Kebutuhan	MMPS_F_26
Prosedur	1. Mengakses halaman Backlog atau Semua Sprint 2. Menekan nama salah satu tugas.
Hasil yang diharapkan	Aplikasi akan mengarahkan ke halaman detail tugas dan juga menampilkan komentar untuk tugas tersebut.

Hasil	Aplikasi mengarahkan ke halaman detail tugas dan juga menampilkan komentar untuk tugas tersebut.
Status	Valid

6.3.27 Pengujian Validasi Menambah Komentar

1. Kasus Uji Menambah Komentar

Tabel 6.63 Kasus Uji Menambah Komentar

Nama Kasus Uji	Kasus Uji Menambah Komentar
Kode Kebutuhan	MMPS_F_27
Prosedur	1. Mengakses halaman Detail Tugas. 2. Memasukkan komentar = “Apa saja atribut dari proyek?”. 3. Menekan tombol Komentar.
Hasil yang diharapkan	Komentar berhasil ditambahkan dan aplikasi akan memuat ulang halaman Detail Tugas.
Hasil	Komentar berhasil ditambahkan dan aplikasi memuat ulang halaman Detail Tugas.
Status	Valid

2. Kasus Uji Menambah Komentar Alternatif 1

Tabel 6.64 Kasus Uji Menambah Komentar Alternatif 1

Nama Kasus Uji	Kasus Uji Menambah Komentar Alternatif 1
Kode Kebutuhan	MMPS_F_27
Prosedur	1. Mengakses halaman Detail Tugas. 2. Tidak memasukkan komentar. 3. Menekan tombol Komentar.
Hasil yang diharapkan	Penambahan gagal dan aplikasi akan menampilkan pesan untuk mengisi *field* komentar.
Hasil	Penambahan gagal dan aplikasi menampilkan pesan untuk mengisi *field* nama komentar.
Status	Valid

6.3.28 Pengujian Validasi Melihat Riwayat Proyek

1. Kasus Uji Melihat Riwayat Proyek

Tabel 6.65 Kasus Uji Melihat Riwayat Proyek

Nama Kasus Uji	Kasus Uji Melihat Riwayat Proyek
Kode Kebutuhan	MMPS_F_28
Prosedur	1. Menekan Riwayat Aktivitas pada navigasi di Aplikasi.
Hasil yang diharapkan	Aplikasi akan menampilkan halaman riwayat aktivitas manajemen.
Hasil	Aplikasi menampilkan halaman riwayat aktivitas manajemen.
Status	Valid

6.3.29 Pengujian Validasi FilterTugas

1. Kasus Uji FilterTugas

Tabel 6.66 Kasus Uji FilterTugas

Nama Kasus Uji	Kasus Uji FilterTugas
Kode Kebutuhan	MMPS_F_29
Prosedur	1. Mengakses halaman Backlog. 2. Memilih *option* "Pengelolaan Proyek" pada filter berdasarkan modul. 3. Menekan tombol Filter.
Hasil yang diharapkan	Aplikasi akan memuat ulang halaman Backlog dan hanya akan menampilkan tugas yang termasuk modul Pengelolaan Proyek.
Hasil	Aplikasi memuat ulang halaman Backlog dan hanya menampilkan tugas yang termasuk modul Pengelolaan Proyek.
Status	Valid

2. Kasus Uji FilterTugas Alternatif 1

Tabel 6.67 Kasus Uji FilterTugas Alternatif 1

Nama Kasus Uji	Kasus Uji FilterTugas Alternatif 1
Kode Kebutuhan	MMPS_F_29
Prosedur	1. Mengakses halaman Backlog. 2. Memilih *option* "Semua" pada filter berdasarkan modul. 3. Menekan tombol Filter.

Hasil yang diharapkan	Aplikasi akan memuat ulang halaman Backlog dan akan menampilkan tugas dari semua modul dan yang belum memiliki modul.
Hasil	Aplikasi memuat ulang halaman Backlog dan menampilkan tugas dari semua modul dan yang belum memiliki modul.
Status	Valid

6.3.30 Pengujian Validasi Keluar Proyek

1. Kasus Uji Keluar Proyek

Tabel 6.68 Kasus Uji Keluar Proyek

Nama Kasus Uji	Kasus Uji Keluar Proyek
Kode Kebutuhan	MMPS_F_30
Prosedur	1. Menekan Keluar Proyek pada navigasi di Aplikasi.
Hasil yang diharapkan	Pengguna keluar dari halaman manajemen proyek dan aplikasi akan menampilkan halaman semua proyek.
Hasil	Pengguna keluar dari halaman manajemen proyek dan aplikasi menampilkan halaman semua proyek.
Status	Valid

6.3.31 Pengujian Validasi Logout

1. Kasus Uji Logout

Tabel 6.69 Kasus Uji Logout

Nama Kasus Uji	Kasus Uji Logout
Kode Kebutuhan	MMPS_F_31
Prosedur	1. Menekan Keluar pada navigasi di Aplikasi.
Hasil yang diharapkan	Pengguna keluar dari aplikasi dan akan menuju ke halaman login
Hasil	Pengguna keluar dari aplikasi dan menuju ke halaman login
Status	Valid

6.3.32 Pengujian Validasi Report

1. Kasus Uji Report

Tabel 6.70 Kasus Uji Report

Nama Kasus Uji	Kasus Uji Report
Kode Kebutuhan	MMPS_F_32
Prosedur	1. Menekan Report pada navigasi di Aplikasi.
Hasil yang diharapkan	Aplikasi akan menampilkan halama report.
Hasil	Aplikasi menampilkan halama report.
Status	Valid

6.3.33 Pengujian Validasi Mengerjakan Ulang Tugas

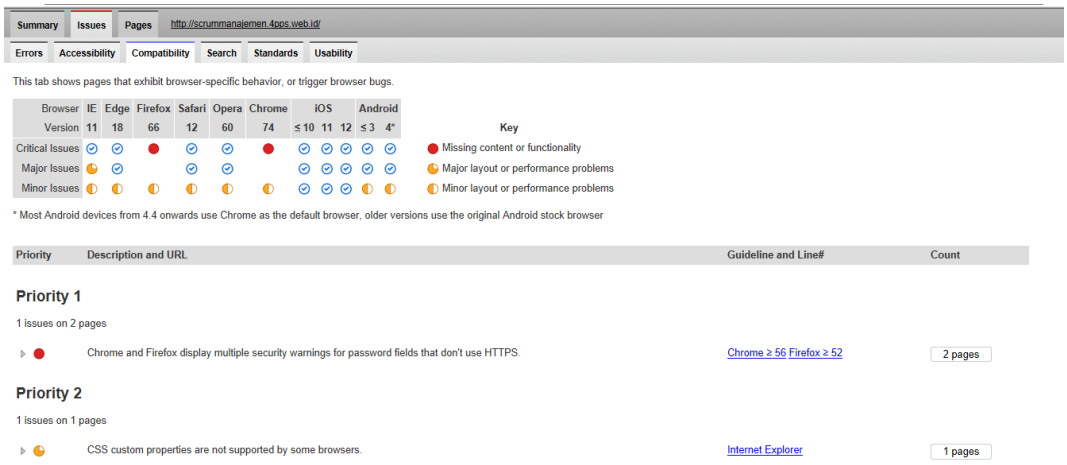
1. Kasus Uji Mengerjakan Ulang Tugas

Tabel 6.71 Kasus Uji Mengerjakan Ulang Tugas

Nama Kasus Uji	Kasus Uji Mengerjakan Ulang Tugas
Kode Kebutuhan	MMPS_F_33
Prosedur	1. Mengakses halaman Semua Sprint. 2. Menekan option Move to Current Sprint pada salah satu tugas
Hasil yang diharapkan	Aplikasi akan memuat ulang halaman Semua Proyek dan tugas yang dipilih akan berada pada sprint yang saat ini sedang berjalan serta mengubah status tugas menjadi Doing.
Hasil	Aplikasi memuat ulang halaman Semua Proyek dan tugas yang dipilih berada pada sprint yang saat ini sedang berjalan serta mengubah status tugas menjadi Doing.
Status	Valid

6.4Pengujian *Compatibility*

Pengujian *Compatibility* merupakan pengujian yang dilakukan untuk mengetahui dan memeriksa apakah aplikasi manajemen *multi* proyek menggunkan metode scrum dapat berjalan atau diakses pada beberapa *browser* dan versi browsernya, serta pada beberapa *platform*. Dalam pengujian *compatibility* ini, dilakukan menggunakan aplikasi pihak ketiga yang bernama SortSite versi 5.



Gambar 6.6 Hasil Pengujian *Compatibility*

Gambar 6.6 adalah hasil dari proses pengujian *compatibility* menggunakan aplikasi *SortSite* versi 5. Terdapat delapan *browser* dengan versi keseluruhan mencapai 11, dan 3 *platform*. Dari hasil pengujian menunjukkan bahwa aplikasi dapat berjalan dengan baik pada 9 dari 11 versi *browser* yang diujikan. Terdapat tiga tingkatan untuk permasalahan yang muncul, yaitu *Critical Issues, Major Issues,* dan *Minor Issues.* Terdapat dua *Critical Issues* pada *browser* firefox dan chrome yang terjadi karena tidak digunakannya HTTPS pada aplikasi. Namun, jika aplikasi diakses secara langsung, akan tetap berjalan baik tanpa ada fungionalitas yang tidak berfungsi walaupun tidak menggunakan HTTPS. Selain itu terdapat satu *Critical Issues* dan delapan *Minor Issues* dari hasil pengujian aplikasi yang berkaitan dengan penulisan sintaks pada kode CSS.

BAB 7 KESIMPULAN DAN SARAN

7.1 Kesimpulan

Kesimpulan berdasarkan hasil yang diperoleh tentang aplikasi manajemen multi proyek meggunakan metode *scrum*, diperoleh kesimpulan sebagai berikut:

1. Berdasarkan hasil dari analisis kebutuhan aplikasi manajemen multi proyek meggunakan metode scrum, diperoleh total 34 kebutuhan fungsional dan 1 kebutuhan non fungsional, yaitu pada aspek *compatibility* dari aplikasi. Pada proses analisis kebutuhan, dimodelkan dalam bentuk *use case diagram* dan dijelaskan pada *use case scenario*.
2. Berdasarkan hasil dari perancangan aplikasi manajemen *multi* proyek meggunakan metode *scrum*, diperoleh beberapa perancangan meliputi, pemodelan *sequence diagram* yang berisi penjelasan tentang urutan proses aplikasi dan interaksi yang terjadi antar objek di dalam setiap fungsionalitas aplikasi. Pemodelan *class diagram* yang di dalamnya memuat kelas kelas yang digunakan dalam pengembangan aplikasi, dalam kasus ini menggunakan arsitektur MVC. Perancangan algoritme yang menghasilkan *pseudocode* untuk nantinya diimplementasikan pada kode program. Perancangan antarmuka yang berisi desain dari antarmuka dan penjelasan pada setiap komponennya. Sedangkan untuk perancangan data dimodelkan dalam bentuk ERD atau *Entity Relationship Diagram*.
3. Berdasarkan hasil dari implementasi aplikasi manajemen multi proyek meggunakan metode scrum, diperoleh spesifikasi pengembangan aplikasi baik dari segi perangkat lunak maupun perangkat keras yang digunakan penulis dalam melakukan proses implementasi dari perancangan dan pemodelan yang telah dilakukan sebelumnya. Untuk implementasi kode berasal dari perancangan algoritme, implementasi antarmuka berasal dari perancangan antarmuka, serta implementasi data.
4. Berdasarkan dari pengujian aplikasi manajemen *multi* proyek meggunakan metode *scrum*, diperoleh hasil yang valid untuk pengujian unit menggunakan metode *white box testing* pada keseluruhan jalur uji. Untuk pengujian integrasi didapatkan hasil yang valid pada proses pengujian. Pada pengujian validasi menggunakan metode *black box testing,* diperoleh nilai valid terhadap keseluruhan kasus uji. Sedangkan pada pengujian *compatibility* menunjukkan bahwa aplikasi dapat berjalan dengan baik pada 9 dari 11 versi browser yang diujikan.

7.2 Saran

Saran untuk pengembangan selanjutnya adalah penambahan fitur notifikasi kepada pengguna untuk aktivitas yang terjadi, dan memberi informasi jika terdapat pekerjaan yang akan mencapai deadline pengerjaan.

DAFTAR REFERENSI

Carneiro, L. B., Silva A. C. C. L. M. dan Alencar, L. H, 2018. *Scrum Agile Project Management Methodology Application for Workflow Management: A Case Study*, [e-journal]. Tersedia di: <https://ieeexplore.ieee.org/document/8607356> [Diakses 28 Februari 2019].

Farrel-Vinay. (2008). *Manage Software Testing.* Auerbach Publications.

G. Cloke, "Get Your Agile Freak On! Agile Adoption at Yahoo! Music", in AGILE 2007, 2007, pp. 240-248.

Husen, Abrar. 2009, Manajemen Proyek. Yogyakarta: Andi Offset.

K. Schwaber and M. Beedle, "Agile software development with Scrum", in Series in agile software development Upper Saddle River, NJ: Prentice Hall, 2002, pp. 31-56.

Ken, S. & Jef, S., 2017. Panduan Scrum. s.I.:ScrumOrg & ScrumInc.

Marchenko, A dan Abrahamsson, P, 2008. *Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges*, [e-journal]. Tersedia di: <https://ieeexplore.ieee.org/document/4599449> [Diakses 13 Januari 2019].

Oza, N., Fagerholm, F. dan Münch, J, 2013. *How Does Kanban Impact Communication and Collaboration in Software Engineering Teams?*, [e-journal]. Tersedia di: <https://ieeexplore.ieee.org/document/6614747> [Diakses 28 Februari 2019].

Parsons, D., Thorn, R., Inkila, M., dan MacCallum, K, 2018. *Using Trello to Support Agile and Lean Learning with Scrum and Kanban in Teacher Professional Development*, [e-journal]. Tersedia di: <https://ieeexplore.ieee.org/document/8615399> [Diakses 28 Februari 2019].

Romano B.L., and Silva A.D, 2015. *Project management using the Scrum agile method: A case study within a small enterprise*, [e-journal]. Tersedia di: <https://ieeexplore.ieee.org/document/7113578> [Diakses 13 Januari 2019].

Sommerville, I., 2011. *Software engineering*. 9th ed. London: Addison-Wesley.

Srivastava A., Saraswat S., dan Bhardwaj S, 2017. *SCRUM model for agile methodology*, [e-journal]. Tersedia di: <https://ieeexplore.ieee.org/document/8229928> [Diakses 13 Januari 2019].

Wagh, R, 2012. *Using Scrum for Software Engineering Class Projects*, [e-journal]. Tersedia di: <https://ieeexplore.ieee.org/document/6170011> [Diakses 28 Februari 2019].

LAMPIRAN A HASIL WAWANCARA

LAMPIRAN A MATERI WAWANCARA

Berikut merupakan lampiran dari materi wawancara untuk mendukung penelitian ini. Adapun materi ini dibagi menjadi 3 bagian yang terdiri dari :

1. **Gambaran Umum *Software House***

   Materi wawancara ini bertujuan untuk mengetahui gambaran umum tentang *Software House* yang di wawancarai.

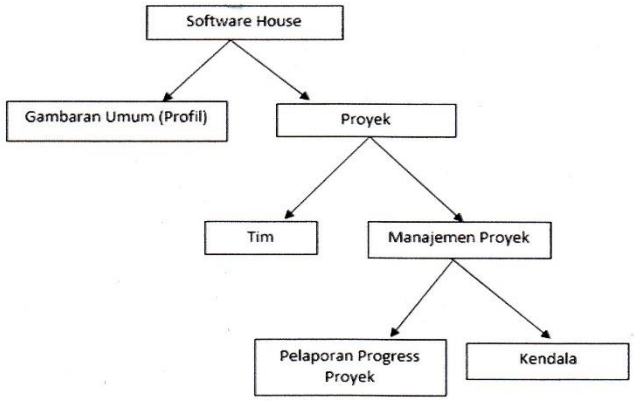
2. **Mekanisme Pengerjaan Proyek pada *Software House***

   Materi wawancara ini bertujuan untuk mengetahui bagaimana cara sebuah *software house* mengerjakan suatu proyek dan memanajemennya.

3. **Kendala Mekanisme Pengerjaan Proyek pada *Software House***

   Materi wawancara ini bertujuan untuk mengetahui permasalahan selama pengerjaan proyek pada *Software house.*

Untuk susunan *tree/work structured* terkait materi wawancara dengan narasumber digambarkan pada Gambar A.



Gambar A Tree/Work Structured Materi Wawancara

Berikut merupakan lampiran dari materi wawancara untuk penelitian ini, yang telah dilaksanakan pada :

Hari/ Tanggal : Jumat, 15 Maret 2019

Waktu : 09.30 – 10.00

Tempat : Software House Profile Image Studio

Narasumber : Anggrean Renozonarca

Jabatan : Direktur Operasional

No *Handphone* : 081357477083

Wawancara ini dilakukan untuk mendukung informasi terkait dengan manajemen proyek yang ada pada *Software House* di Kota Malang.

A.1 Gambaran Umum *Software House*

Materi wawancara ini bertujuan untuk mengetahui gambaran umum mengenai *Software House* yang di wawancarai.

No.	Pertanyaan	Jawaban	Checklist
1.	Informasi *Software House:* - Nama - Alamat - Tahun Berdiri - Nama Pemilik	- Nama : Profile Image Studio - Alamat : Perum. Bumi Tunggulwulung Indah G/8 Lowokwaru Malang 65143 - Tahun berdiri : 2011 - Nama Pemilik : Amar Alpabet	√
2.	Apa saja jenis proyek yang biasa dikerjakan oleh profile image?	Development web, aplikasi android, manajemen bisnis, dan fokus ke penanganan bisnis secara digital. Pernah juga mengembangkan sistem tentang paket tour didaerah termasuk pemasaran, pengaturan keuangan dan manajemen tamu.	√
3.	Ada berapa pekerja dan tim yang bekerja pada profile image?	Total ada 7 orang yang bekerja pada profile image. 3 orang bagian managerial meliputi CEO, creative dan operasional. Sisi teknis ada CTO yang memegang 2 orang bagian backend dan front end. Sama 1 orang bertanggung jawab atas sistem tour. Untuk jumlah tim dinamis, tergantung banyak proyek dan skalanya.	√

A.2 Mekanisme Pengerjaan Proyek pada *Software House*

Materi wawancara ini bertujuan untuk mengetahui bagaimana cara sebuah *software house* mengerjakan suatu proyek dan memanajemennya.

No.	Pertanyaan	Jawaban	Checklist
1.	Ada berapa proyek yang	Antara 1 – 3 proyek bersamaan	

	biasa dikerjakan oleh profile image dalam satu waktu?	per bulan, tergantung skala proyeknya.	√
2.	Ketika ada proyek perangkat lunak, apakah penanggung jawabnya sama?	Penanggung jawab berbeda.	√
3.	Dalam 1 waktu apa pernah memegang lebih dari 1 proyek setiap proyek managernya?	Kalo selama ini belum pernah, pasti hanya 1 proyek dalam 1 waktu. Kalo ada 2 proyek misalnya, maka di proyek ke A seseorang tadi jadi project manager, namun di project B hanya jadi developer biasa. Setiap orang tidak mungkin rangkap menjabat project manager dalam 1 waktu, karena nanti pasti managenya susah.	√
4.	Apakah setiap proyek dikerjakan oleh tim yang sama atau berbeda-beda?	Tergantung tipe proyeknya juga. Kalo proyek simple bisa 1 orang saja. Kalo memang ada lebih dari 1 proyek maka tiap orang akan megang lebih dari 1 proyek. Jika kesulitan di SDM, biasanya pakai semacam recruitment per proyek. Bisa juga kerja sama dengan start up lain.	√
5.	Untuk dokumentasi pengerjaan dalam bentuk apa dan apakah ada kendala?	Kalo internal kita pake beberapa tools tambahan kayak kita menyimpan dalam 1 drive yang sama. Untuk progress dan data kita pakai trello. Kalau secara koding kita pakai gitlab. Kalau untuk klien kita kasih manual book.	√
6.	Bagaimana pembagian pekerjaan/tugas dari satu tim?	Sesuai kebutuhannya saja. Sudah terdapat backend dan front end di sisi teknis	√
7.	Apakah klien selalu tahu progress dari proyeknya?	Biasanya kesepakatan dengan klien, kita bikin grup WA atau telegram. Kalau ada report kita sampaikan. Di awal sudah kita kasih timeline pengerjaan. Ketika timeline selesai klien biasanya baru menanyakan	√

		progress.	
8.	Apakah klien ikut andil dalam pembentukan kebutuhan?	Otomatisi kut andil, karena setelah kita ada kontrak untuk start itu kita sudah jelas akan mengerjakan apa, dan klien sudah setuju dengan apa yang akan kita kerjakan. Biasanya diawal ada 2 sampai 3 pertemuan dengan klien untuk menggali kebutuhan.	√
9.	Bagaimana manajer proyek dalam mengetahui progres dari proyek yang di pimpinnya?	Biasanya kita bahas di meeting dan itukan tertulis di trello. Tiap hari senin ada evaluasi untuk semua proyek yang berjalan dan update apapun dibahas pada waktu itu. Namun jika ada kebutuhan yang harus dikerjakan satu tim, kita bias kapan saja. Hari selasa biasanya kita bahas untuk produk.	√
10.	Bagaimana alur pemesanan suatu perangkat lunak mulai dari awal hingga selesai?	Pertama dilakukan pertemuan untuk menggali kebutuhan, dari sisi developer juga memberi saran atau solusi dari masalahnya. Setelah tadi sudah setuju baru kita mulai pengerjaan. Dari situ kalau klien setuju kita buatkan potation penawaran. Kalau mereka sudah setuju timeline dan harganya. Kita bikinkan invoice untuk DP pembayaran. Jika sudah dibayar baru kita start pengerjaan sampai selesai. Lalukita deliver dan berikan manual book sama pendampingan penggunaan	√

A.3 Kendala Mekanisme Pengerjaan Proyek pada *Software House*

Materi wawancara ini bertujuan untuk mengetahui permasalahan selama pengerjaan proyek pada *Software house.*

No.	Pertanyaan	Jawaban	Checklist
1.	Apakah klien pernah	Pasti sempat ada, karena	√

	melakukan komplain terhadap aplikasi yang telah selesai dikerjakan?	seringkali klien keliru dalam penggunaan atau load lebih besar/tinggi dari perkiraan tapi biaya kita sudah termasuk maintenance selama 3 bulan. Jika proyek dengan pemerintah garansinya 1 tahun.	
2.	Apakah pengerjaan suatu proyek pernah melebihi batas waktu yang telah di tetapkan? Apakah penyebabnya?	Selama ini kalaupun ada yang molor waktunya, biasanya lebih karena ada permintaan tambahan dari klien. Namun jika menginginkan perubahan dari sistem, perubahan tadi tidak termasuk kedalam timeline. Untuk timeline biasanya berisi contoh, minggu pertama desain, kemudian developmentnya, terus integrasi antara front end dan back end, baru terakhir UAT sama deployment. Standarnya kita kasih seperti itu. Sesuatu yang dapat dimengerti oleh klien. Jika mereka mintanya modul modul, maka kita menyertakan ditimeline tadi berupa modul. Misalnya tanggal sekian modul ini sudah jadi.	√
3.	Apakah pernah terjadi *miscommunication* antar pekerja dalam pengembangan perangkat lunak?	Kalau itu engga, karena kita pakai git jadi sudah kelihatan semua. Jadi tidak ada permasalahan disitu. Permasalahan miscommunication kadang terjadi jika melibatkan anak magang atau bukan tim utama.	√

Malang, 19-06-2019

Direktur Operasional

profile image studio

Anggrean Renozonarca

LAMPIRAN A MATERI WAWANCARA

Berikut merupakan lampiran dari materi wawancara untuk mendukung penelitian ini. Adapun materi ini dibagi menjadi 3 bagian yang terdiri dari :

1. **Gambaran Umum *Software House***

   Materi wawancara ini bertujuan untuk mengetahui gambaran umum tentang *Software House* yang di wawancarai.

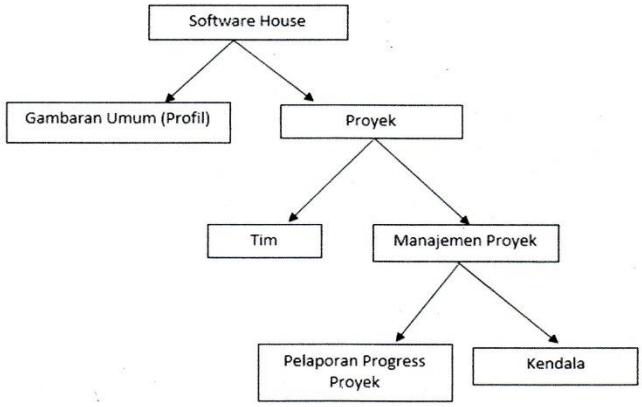
2. **Mekanisme Pengerjaan Proyek pada *Software House***

   Materi wawancara ini bertujuan untuk mengetahui bagaimana cara sebuah *software house* mengerjakan suatu proyek dan memanajemennya.

3. **Kendala Mekanisme Pengerjaan Proyek pada *Software House***

   Materi wawancara ini bertujuan untuk mengetahui permasalahan selama pengerjaan proyek pada *Software house.*

Untuk susunan *tree/work structured* terkait materi wawancara dengan narasumber digambarkan pada Gambar A.



Gambar A Tree/Work Structured Materi Wawancara

Berikut merupakan lampiran dari materi wawancara untuk penelitian ini, yang telah dilaksanakan pada :

Hari/ Tanggal : Jumat, 15 Maret 2019

Waktu : 17.30 – 17.50

Tempat : Software House Komal Development

Narasumber :M Ziaelfikar Albaba

Jabatan :Owner

No *Handphone* : 082337576338

Wawancara ini dilakukan untuk mendukung informasi terkait dengan manajemen proyek yang ada pada *Software House* di Kota Malang.

A.1 Gambaran Umum *Software House*

Materi wawancara ini bertujuan untuk mengetahui gambaran umum mengenai *Software House* yang di wawancarai.

No.	Pertanyaan	Jawaban	Checklist
1.	Informasi*Software House:* - Nama - Alamat - Tahun Berdiri - Nama Pemilik	- Nama : Komal Development - Alamat : DILo Malang, Jl. Basuki Rachmad 7-9 East Java, Indonesia - Tahun berdiri :2016 - Nama Pemilik :M Ziaelfikar Albaba	√
2.	Apa saja jenis proyek yang biasa dikerjakan oleh profile image?	Development web, ui dan ux. Untuk android tidak.	√
3.	Ada berapa pekerja dan tim yang bekerja pada profile image?	Ada 4 pekerja	√

A.2 Mekanisme Pengerjaan Proyek pada *Software House*

Materi wawancara ini bertujuan untuk mengetahui bagaimana cara sebuah *software house* mengerjakan suatu proyek dan memanajemennya.

No.	Pertanyaan	Jawaban	Checklist
1.	Ada berapa proyek yang biasa dikerjakan oleh profile image dalam satu waktu?	2 hingga 3 proyek	√
2.	Ketika ada proyek perangkat lunak, apakah penanggung jawabnya sama?	Penanggung jawab berbeda. Tapi mungkin sama jika jumlah proyek melebihi jumlah tim	√
3.	Dalam 1 waktu apa pernah memegang lebih dari 1 proyek setiap proyek managernya?	Satu orang memegang 1 proyek. Bisa programmer atau yang lain termasuk saya (founder). Jika dalam satu waktu ada lebih dari 4 proyek maka mungkin 1 orang berperan jadi project	√

		managernya. Tapi jika dirasa lebih dari 4 proyek dalam 1 waktu keberatan, maka dari kami menolak proyek tersebut	
4.	Apakah setiap proyek dikerjakan oleh tim yang sama atau berbeda-beda?	Timnya sama, dari 4 orang. Hanya beda di project managernya	✓
5.	Untuk dokumentasi pengerjaan dalam bentuk apa dan apakah ada kendala?	Untuk kode di gitlab. Komunikasi menggunakan discord juga, yang untuk main game itu. Kami juga menggunakan trello. Untuk kendala kurang up to date karena lupa mengisi trello	✓
6.	Bagaimana pembagian pekerjaan/tugas dari satu tim?	Sudah terdapat bagiaanya masing masing. Hanya berbeda di manager proyek	✓
7.	Apakah klien selalu tahu progress dari proyeknya?	Iya. Biasanya ada 3 term in atau sesi, seperti opening, analisis progress, dan final presentasi. Untuk jangka waktu antar sesi tergantung waktu deadline proyek, kalau 1 bulan ya tinggal dibagi jumlah sesi tadi	✓
8.	Apakah klien ikut andil dalam pembentukan kebutuhan? Apakah hanya andil di awal saja?	Iya pasti, karena semua dari klien data datanya, atau mungkin klien sudah memiliki desain. Ya diawal, kalau ditengah tengah jarang terjadi. Mungkin kalau ada perubahan kebutuhan.	✓
9.	Bagaimana manajer proyek dalam mengetahui progres dari proyek yang di pimpinnya?	Ya lewat disord tadi, kami komunikasi.	✓

A.3 Kendala Mekanisme Pengerjaan Proyek pada *Software House*

Materi wawancara ini bertujuan untuk mengetahui permasalahan selama pengerjaan proyek pada *Software house.*

No.	Pertanyaan	Jawaban	Checklist
1.	Apakah klien pernah	Sering, bahkan hampir	

	melakukan komplain terhadap aplikasi yang telah selesai dikerjakan?	disemua proyek. Masalah masalah umum seperti bug fixing, server down. Untuk garansinya 1 tahun.	✓
2.	Apakah pengerjaan suatu proyek pernah melebihi batas waktu yang telah di tetapkan? Apakah penyebabnya?	Pernah, terjadi saat fitur atau fungsinya sangat kompleks dan juga sudah ada komunikasi dan kesepakatan dengan klien tentang itu	✓
3.	Apakah pernah terjadi *miscommunication* antar pekerja dalam pengembangan perangkat lunak?	Sering. Cara mengatasinya kadang kita ngopi bareng.	✓

Malang, 17-06-2019

Owner Komal Development

KOMAL TECHNO .com

M. Ziaelfikar Albaba

LAMPIRAN A MATERI WAWANCARA

Berikut merupakan lampiran dari materi wawancara untuk mendukung penelitian ini. Adapun materi ini dibagi mpenjadi 3 bagian yang terdiri dari :

1. **Gambaran Umum *Software House***

   Materi wawancara ini bertujuan untuk mengetahui gambaran umum tentang *Software House* yang di wawancarai.

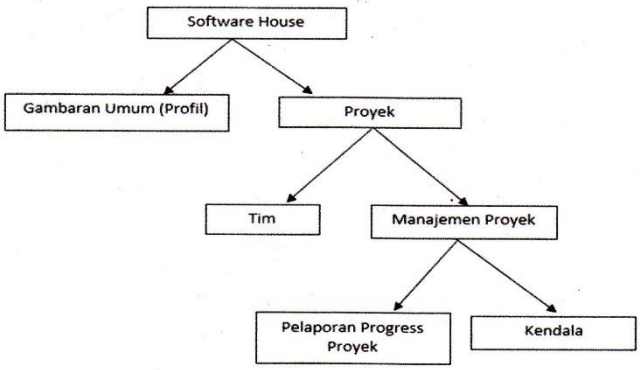
2. **Mekanisme Pengerjaan Proyek pada *Software House***

   Materi wawancara ini bertujuan untuk mengetahui bagaimana cara sebuah *software house* mengerjakan suatu proyek dan memanajemennya.

3. **Kendala Mekanisme Pengerjaan Proyek pada *Software House***

   Materi wawancara ini bertujuan untuk mengetahui permasalahan selama pengerjaan proyek pada *Software house.*

Untuk susunan *tree/work structured* terkait materi wawancara dengan narasumber digambarkan pada Gambar A.



Gambar A Tree/Work Structured Materi Wawancara

Berikut merupakan lampiran dari materi wawancara untuk penelitian ini, yang telah dilaksanakan pada :

Hari/ Tanggal	: Kamis, 25 Juli 2019
Waktu	: 15.10 – 15.30
Tempat	: CV. Sarana Utama Solusindo
Narasumber	: 1. Sony Witarto

2. Deni Apriadi

Jabatan : 1. General Manager

2. Junior Programmer

No *Handphone* : 081805173445

Wawancara ini dilakukan untuk mendukung informasi terkait dengan manajemen proyek yang ada pada *Software House* di Kota Malang.

A.1 Gambaran Umum *Software House*

Materi wawancara ini bertujuan untuk mengetahui gambaran umum mengenai *Software House* yang di wawancarai.

No.	Pertanyaan	Jawaban	Checklist
1.	Informasi *Software House:* - Nama - Alamat - Tahun Berdiri - Nama Pemilik	- Nama : CV. Sarana Utama Solusindo - Alamat : Jl. Anggrek Garuda No.49, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur 65141 - Tahun berdiri : 2016 - Nama Pemilik : Arief Andy	√
2.	Apa saja jenis proyek yang biasa dikerjakan oleh sarana utama solusindo?	Kami menangani semua jenis proyek perangkat lunak termasuk aplikasi desktop, website, mobile, dan IoT.	√
3.	Ada berapa pekerja dan tim yang bekerja pada sarana utama solusindo?	Total ada 5 pekerja. Untuk tim ada bagiannya masing masing. Disini ada aplikasi web dan android. Untuk web sendiri dan android sendiri dari 5 orang itu.	√

A.2 Mekanisme Pengerjaan Proyek pada *Software House*

Materi wawancara ini bertujuan untuk mengetahui bagaimana cara sebuah *software house* mengerjakan suatu proyek dan memanajemennya.

No.	Pertanyaan	Jawaban	Checklist
1.	Ada berapa proyek yang biasa dikerjakan oleh sarana utama solusindo ge dalam satu waktu?	1 – 2 proyek dalam waktu bersamaan. Setiap 1 proyek bias sampai 3 bulan.	√
2.	Ketika ada proyek perangkat lunak, apakah penanggung jawabnya sama?	Penanggung jawabnya berbeda.	√

3.	Dalam 1 waktu apa pernah memegang lebih dari 1 proyek setiap proyek managernya?	Selama ini belum, karena nanti tidak fokus. Kalau memang ada proyek bebarengan diselesaikan dulu baru nanti pindah.	√
4.	Apakah setiap proyek dikerjakan oleh tim yang sama atau berbeda-beda?	Tergantung tipe proyeknya juga. Disini ada aplikasi web dan android. Untuk web sendiri dan android sendiri dari 5 orang itu	√
5.	Untuk dokumentasi pengerjaan dalam bentuk apa dan apakah ada kendala?	Dokumentasi mulai dari foto, pertemuan awal yang membicarakan konsep, ada dokumentasi fotonya. Apa yang dikerajakan juga tertulis. Jadi ada lembaran lembaran kertas ditulis lagi di word atau excel. Jadi itu untuk bahan ketika klien lupa. Kalo kendala, klien sendiri kurang menguasai bisnis proses. Jadi akhirnya kita yang membuatkan bisnis proses berdasarkan pengamatan kita.	√
6.	Bagaimana pembagian pekerjaan/tugas dari satu tim?	Tergantung proyeknya, jika website biasanya kami bagi dengan bagian seperti front-end dan back-end.	√
7.	Apakah klien selalu tahu progress dari proyeknya?	Kebanyakan dari klien tidak pernah menanyakan progress. Pekerjaan yang siap untuk kita simulasikan, baru kita ke kliennya. Untuk klien kita bikin jadwal proyek, misalnya proyek ini selesainya dalam 3 bulan. Untuk deadline dalam proses pengembangan sebenarnya dari kita, yang mana mengacu dari proyek. Misalnya 3 bulan, maka kita bikin timeline untuk 3 bulan.	√
8.	Apakah klien ikut andil dalam pembentukan kebutuhan?	Iya, pertama ada pertemuan awal membicarakan konsep, termasuk yang membahas bisnis proses dari klien untuk pembentukan kebutuhan. Selain itu klien ikut andil ketika telah kita lakukan simulasi sistem.	√

9.	Bagaimana manajer proyek dalam mengetahui progres dari proyek yang di pimpinnya?	Setiap sabtu ada meeting untuk progress pekerjaan. Progress harian melalui whatsapp. Jadi setiap 15 menit sebelum pulang itu kirim laporan untuk progress.	√

A.3 Kendala Mekanisme Pengerjaan Proyek pada *Software House*

Materi wawancara ini bertujuan untuk mengetahui permasalahan selama pengerjaan proyek pada *Software house.*

No.	Pertanyaan	Jawaban	Checklist
1.	Apakah klien pernah melakukan komplain terhadap aplikasi yang telah selesai dikerjakan?	Klien tidak pernah komplain namun biasanya klien kurang memahami apa yang ingin diminta dan bisnis prosesnya sehingga akhirnya kami membuatkan bisnis prosesnya berdasarkan pengamatan. Klien hanya komplain ketika dilakukan simulasi system kepada klien dan klien merasa kurang atau tidak cocok. Disini jika aplikasi telah selesai terdapat garansi 1 tahun untuk maintence.	√
2.	Apakah pengerjaan suatu proyek pernah melebihi batas waktu yang telah di tetapkan? Apakah penyebabnya?	Kalau sesuai dengan kontrak belum pernah terjadi. Kalau ada kurang lebihnya itu sudah diluar kontrak. Maksudnya ketika kita simulasikan dan ada tambahan menurut klien harusnya tidak seperti ini.	√
3.	Apakah pernah terjadi *miscommunication* antar pekerja dalam pengembangan perangkat lunak?	Kalau itu engga, karena sudah ada bagian dan timeline masing masing.	√

Malang, 11-07-2019

General Manager

Sarana Utama Solusindo

Sony Witarto