PENGEMBANGAN APLIKASI BADSTICS UNTUK MENENTUKAN KARAKTERISTIK PEMAIN BULU TANGKIS MENGGUNAKAN METODE NAIVE BAYES (STUDI KASUS : PBSI JAWA TIMUR)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Robihamanto
NIM: 145150207111067

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENGEMBANGAN APLIKASI BADSTICS UNTUK MENENTUKAN KARAKTERISTIK PEMAIN BULU TANGKIS MENGGUNAKAN METODE NAIVE BAYES (STUDI KASUS : PBSI JAWA TIMUR)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Robihamanto
NIM: 145150207111067

Skripsi ini telah diuji dan dinyatakan lulus pada
15 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Adam Hendra Brata, S.Kom., M.T., M.Sc.	Komang Candra Brata, S.Kom., M.T., M.Sc.
NIK: 201607 900105 1 001	NIK: 201607 891204 1 001

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 pustaka.

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, 15 Januari 2018

Robihamanto

NIM: 145150207111067

KATA PENGANTAR

Segala puji bagi Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "PENGEMBANGAN APLIKASI BADSTICS UNTUK MENENTUKAN KARAKTERISTIK PEMAIN BULU TANGKIS MENGGUNAKAN METODE NAIVE BAYES (STUDI KASUS : PBSI JAWA TIMUR)" .

Untuk kesempatan ini penulis juga menyampaikan rasa terima kasih kepada pihak – pihak yang telah membantu penulis selama penyusunan skripsi, diantaranya:

1. Allah Subhanahu wa Ta'ala yang telah memberi kemudahan dalam semua proses penulisan skripsi ini.
2. Kepada kedua orang tua penulis, yaitu Bapak Sukamto dan Ibu Fatmudikah beserta keluarga besar yang selalu memberikan segala masukan, doʾa, motivasi dan semangat yang tidak terputus.
3. Bapak Adam Hendra Brata, S.Kom., M.T., M.Sc selaku dosen pembimbing I yang telah meluangkan waktu untuk memberikan masukan, ilmu, serta saran yang bermanfaat dalam proses penyelesaian skripsi ini.
4. Bapak Komang Candra Brata, S.Kom., M.T., M.Sc selaku dosen pembimbing II yang juga telah meluangkan waktu untuk memberikan masukan, ilmu, serta saran yang bermanfaat dalam proses penyelesaian skripsi ini.
5. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Drs. Mardji, M.T, dan Bapak Edy Santoso, S.Si, M.Kom selaku Dekan, Wakil Dekan 1, Wakil Dekan 2, dan Wakil Dekan 3 Fakultas Ilmu Komputer, Universitas Brawijaya.
6. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D dan Bapak Agus Wahyu Widowo, S.T, M.Cs selaku Ketua Jurusan Teknik Informatika dan Kepala Program Studi Teknik Informatika Fakultas Ilmu Komputer, Universitas Brawijaya.
7. Seluruh Dosen Fakultas Ilmu Komputer, Universitas Brawijaya atas

kesediaannya dalam mengajarkan dan membagikan ilmu yang bermanfaat bagi penulis.

8. Rekan-rekan Basic Computing Community (BCC) yang membantu memberikan ilmu dalam implementasi dari penelitian ini.
9. Bapak Purnomo selaku *expert* dalam bidang bulutangkis yang memberikan ilmu mengenai Bulutangkis.
10. Teman-teman satu rumah di Kota Malang yaitu Bossarito Putro, Eko Setiyono, Moch Wahtu Imam Santosa, Mukmin dan Riski Pradana yang telah membantu dalam beberapa aspek tentang arti penelitian dan kekeluargaan.
11. Rekan-rekan PPK-V Study Tour yaitu Andri Suranta Ginting, Asa Kartika Indrajati, Atikah Nur Rahimah dan Rizkia Desi Yudiari yang telah membantu dalam beberapa aspek penelitian serta beberapa pengalaman dalam bepergian.
12. Semua pihak yang tidak semuanya bisa dituliskan disini yang terlibat secara langsung maupun tidak langsung dalam proses pengerjaan skripsi maupun sebagai pemberi semangat dan motivasi.

Penulis menyadari masih banyak kekurangan dalam penyusunan skripsi ini baik dalam teknik pemyajian materi maupun pembahasan. Demi kesempurnaan penelitian skripsi ini, saran dan kritik yang sifatnya membangun sangat penulis harapkan. Semoga karya tulis ini bermanfaat dan dapat memberikan sumbangan yang berarti bagi pihak yang membutuhkan.

Malang, 15 Januari 2018

Penulis
robihamanto04@gmail.com

ABSTRAK

Dalam pertandingan bulutangkis, pelatih memegang peran penting untuk memberi saran kepada atlet, dan pengawasan saat pertandingan bulutangkis. Pelatih perlu tahu persis karakter dari saingan mereka. Memiliki karakter berarti memiliki rasa ketekunan, kebaikan, ketergantungan dan juga memiliki prinsip moral. Karakteristik pemain sangatlah penting untuk diketahui, karena ketika di dalam permainan ketika karakteristik pemain sudah diketahui pelatih akan tahu arahan apa yang harus diberikan kepada atlet sehingga atlet dapat memenangkan pertandingan, dalam pelaksanaannya karakteristik pemain dapat dilihat dari aksi yang dilakukan di lapangan yang selama ini dilakukan dengan *manual* yaitu dengan merekam permainan, lalu dengan rekaman *video* pelatihn menulisnya dalam dokumen yang membutuhkan waktu hingga 3 sampai 4 jam untuk satu pertandingan. Dengan begitu dibutuhkan sebuah aplikasi untuk memenuhi kebutuhan pelatih yang dapat memberikan informasi karakteristik pemain di lapangan ketika permainan sedang berlangsung. Aplikasi ini dibuat dengan harapan agar pelatih bisa mengetahui karakteristik lawan dan bisa memberikan rujukan berdasarkan statistik para pemain saat berada di lapangan. Aplikasi ini dibangun menggunakan iOS. Dalam proses perhitungan, klasifikasi karakteristik menggunakan *naive bayes* yang merupakan metode klasifikasi yang menggunakan konsep peluang, dimana diasumsikan bahwa setiap atribut contoh (sample data) saling berhadapan berdasarkan atribut kelas. Metode ini dipilih karena hanya memerlukan sejumlah kecil data latih untuk mengestimasi parameter (rata – rata dan variansi dari variabel) yang dibutuhkan untuk klasifikasi. Pada penelitian kali ini hasil pengujian diukur menggunakan pengujian post study dengan metode *System Usability Scale* adalah 86.5 yang berarti masuk kedalam kategori *acceptable* sehingga aplikasi sudah sesuai dengan kebutuhan pengguna.

Kata kunci: *naïve bayes, iOS, bulutangkis, karakteristik*

ABSTRACT

In a game of badminton, the coach plays an important role to give the athlete advises, and supervision during the game of badminton. A coach needs to know exactly the characters of their rival. To have a character means to have sense of adamancy, goodness, praiseworthiness, and dependability, and also have moral principles. Characteristics of players is important to know, because when in the current game the player already knows what to give to the athlete so that the athlete can win the game, in the implementation of the player feature can be seen from the action done in the field that has been done with the manual that is with the recording game , then a video recording and then write it a document that takes up to 3 hours 4 hours for one game. So it takes an application to fulfill the needs of the coaches and provide information and characteristics of the players on the field when the game is in progress. This application is made with the expectation that the coach can know the characteristics of the opponent and can provide referrals based on the statistics of the players when in field. This application was built using the iOS. In the calculations process, the characteristic classification using naive bayes which is a classification method that used the concept of opportunity, where it is assumed that each attribute example (data sample) are off to each other based on the class attributes. This method is chosen because it requires only a small amount of training data to estimate the parameters (average and variance of variables) required for classification. In this study, the results of the test using post-test study with the method of System Usability Scale is 86.5 which means entering into the category of certified-density on demand.

Keywords: *naïve bayes, iOS, badminton, characteristic*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar belakang

Menurut Asfiyani(2016), untuk mencapai prestasi dalam kompetisi dibutuhkan dukungan ilmu pengetahuan dan teknologi keolahragaan. Kini teknologi sudah menjadi bagian dari olahraga, seorang atlet tampil perkasa di ajang dunia, tak hanya karena dia memiliki kemampuan. Juga karena ditunjang dengan faktor yang lain, seperti teknologi, padahal harusnya teknologi yang ada saat ini bisa semakin memudahkan atlet-atlet yang ingin mengetahui kekuatan lawan.

Dalam permainan bulutangkis pelatih harus berperan memberikan arahan dan pengawasan dalam permainan bulutangkis ketika permainan sedang berlangsung, sehingga arahan dapat disampaikan dengan tepat. Pelatih harus cermat dalam memberikan arahan sesuai dengan tujuan latihan serta karakteristik pemain. Karakter merupakan perpaduan segala tabiat manusia yang bersifat
permanen yang merupakan ciri khusus untuk membedakan orang yang satu dengan lainnya, tak terkecuali pemain bulutangkis. Berkarakter artinya menunjukkan sifat memiliki pendirian yang teguh, baik, terpuji dan dapat dipercaya serta memiliki prinsip dalam arti moral (Sugiharto, 2008).

Karakteristik pemain sangatlah penting untuk diketahui, karena ketika didalam permainan ketika karakteristik pemain sudah diketahui pelatih akan tahu arahan apa yang harus diberikan kepada atlet sehingga atlet dapat memenangkan pertandingan. Dengan begitu dibutuhkan sebuah aplikasi untuk memenuhi kebutuhan pelatih yang dapat memberikan informasi karakteristik pemain di lapangan ketika permainan sedang berlangsung (Purnomo, 2017).

Aplikasi ini dibangun untuk dapat mengetahui statistik pemain di lapangan dengan cara memasukkan setiap aksi pemain bulutangkis di lapangan oleh pelatih, sehingga pelatih dapat mengetahui aksi apa yang paling sering dilakukan oleh pemain bulu tangkis dan juga pelatih akan tahu apa kelemahan dari pemain.

Aplikasi ini juga dapat memberikan saran kepada pelatih dari statistik pemain di lapangan dengan menggunakan metode naïve bayes yang menghitung statistik pemain di lapangan lalu membandingkan dengan data training yang sudah diberikan sebelumnya pada aplikasi ini. Sebelum aplikasi ini dibangun informasi karakteristik pemain bulutangkis selama ini dilakukan secara manual dengan melihat rekaman video dari pertandingan yang telah berlangsung dan membutuhkan waktu 3 – 4 jam untuk mendapatkan hasilnya sehingga pelatih dapat memberikan saran yang sesuai dengan statistik di lapangan. Dengan dibuatnya aplikasi ini harapannya dapat memudahkan pelatih dapat mengetahui karakteristik lawan dan dapat memberikan arahan berdasarkan statistik pemain ketika di lapangan ketika permainan berlangsung, sehingga pemain yang diberi

arahan dapat memenangkan pertandingan. Kemudahan dalam menggunakan aplikasi ini dibuktikan dengan menggunakan pengujian *usability testing* pengujian ini dilakukan untuk memastkan bahwa sistem yang dibangun sudah sesuai dengan dapat digunakan dengan mudah, mudah dipahami oleh user dan mudah untuk dioperasikan oleh user nantinya.

Aplikasi ini dibangun menggunakan pendekatan OO (pendekatan berorientasi objek) yaitu pemodelan yang digunakan untuk menspesifikasikan atau mendeskripsikan sebuah sistem perangkat lunak yang terkait dengan objek, terdapat beberapa tipe diagram dalam pendekatan OO antara lain *Use-Case* Diagram, *Sequence* Diagram dan *Class* Diagram. Teknologi yang digunakan dalam pembangunan aplikasi ini adalah swift, swift merupakan bahasa pemrograman yang hebat dan intuitif untuk macOS, iOS, watchOS and tvOS. Syntax pada Swift singkat namun ekspresif, swift juga memiliki fitur modern yang disukai oleh pengembang. Swift merupakan bahasa yang aman dengan desain yang baik untuk membangun sebuah perangkat lunak yang dapat berjalan secepat kilat(Apple, 2017). Dalam perhitungannya klasifikasi karakteristik menggunakan naïve bayes yaitu metode pengklasifikasian dengan menggunakan konsep peluang, dimana diasumsikan bahwa setiap atribut contoh (data sampel) bersifat saling lepas satu sama lain berdasarkan atribut kelas. Metode naïve bayes diharapkan mampu memberikan informasi karakteristik pemain bulutangkis sehingga pelatih dapat memberikan saran yang sesuai dengan statistik di lapangan ketika pertandingan berlangsung.

1.2 Rumusan masalah

1. Apa saja kebutuhan yang digunakan dalam pengembangan aplikasi Badstics untuk menentukan karakteristik pemain bulutangkis?
2. Bagaimana hasil rancangan dan implementasi aplikasi Badstics berdasarkan analisis kebutuhan yang telah dilakukan sebelumnya?
3. Bagaimana penerapan algoritme naïve bayes dalam menentukan karakteristik pemain bulu tangkis?
4. Bagaimana aplikasi dapat memudahkan pelatih dengan melakukan pengujian *usability testing* dapat menjawab dan memenuhi kebutuhan dan perancangan sistem yang dibangun?

1.3 Tujuan

1. Mengetahui kebutuhan yang digunakan dalam pembangunan aplikasi Badstics untuk menentukan karakteristik pemain bulutangkis.
2. Mengetahui perancangan dan implementasi berdasarkan analisis kebutuhan yang telah dilakukan sebelumnya.
3. Mengetahui penerapan algoritme naïve bayes dalam menentukan karakteristik pemain bulutangkis.
4. Mengetahui kemudahan dalam penggunaan aplikasi untuk memenuhi kebutuhan pelatih.

1.4 Manfaat

1. Untuk penulis, memberikan wawasan berupa pengetahuan menggunakan metode Naïve Bayes untuk menentukan karakteristik pemain bulu tangkis.
2. Pelatih bulutangkis dapat mengetahui karakteristik pemain sehingga dapat memberikan arahan yang tepat kepada pemain.

1.5 Batasan masalah

Batasan masalah dalam pengembangan aplikasi Badstics ini adalah sebagai berikut:

1. Dalam penelitian ini adalah aplikasi Badstics hanya dapat digunakan untuk menentukan karakteristik pemain bulu tangkis dalam pertandingan Tunggal Putra atau Tunggal Putri.
2. Observasi dilakukan di PBSI Jawa Timur sehingga masalah yang terjadi hanya pada lingkup PBSI Jawa Timur.
3. Data, parameter gerakan dan perhitungan penentuan karakteristik yang diperoleh hanya dari PBSI Jawa Timur sehingga fitur yang dibuat hanya pada kebutuhan PBSI Jawa Timur
4. Pengujian usabilitas hanya dilakukan dari beberapa aspek saja yaitu efektivitas.
5. Aplikasi hanya dapat berjalan pada iPad dengan versi minumum iOS 8.

1.6 Sistematika pembahasan

Untuk mencapai tujuan yang diharapkan maka sistematika penulisan yang disusun dalam tugas akhir ini adalah :

1. Bab I PENDAHULUAN

Pada bab ini akan membahas mengenai latar belakang, identifikasi masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, ruang lingkup, sistemika pembahsan dan jadwal penelitian.

2. Bab II LANDASAN KEPUSTAKAAN

Pada bab ini akan menguraiakan kajian pustaka, dasar-dasar teori yang melandasi penulisan dan penelitian.

3. Bab III METODE PENELITIAN

Pada bab ini menjelaskan mengenai kajian pustaka, analisis kebutuhan, wawancara, pengumpulan data dan pengelolaan data.

4. Bab IV ANALISIS KEBUTUHAN

Pada bab ini akan dijelaskan mengenai analisis kebutuhan untuk membangun aplikasi Badstics ini.

5. Bab V PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai perancangan sistem yang digunakan dalam pembangunan aplikasi Badstics, bab ini juga menjelaskan gambaran sistem dan deskripsi sistem hasil analisa kebutuhan dan perancangan yang diimplementasikan ke dalam program komputer.

6. Bab VI PENGUJIAN DAN ANALISIS

Pada bab ini memuat tentang analisa dan hasil pengujian dari penelitian yang telah dilakukan.

7. Bab VII PENUTUP

Pada bab ini berisi tentang kesimpulan dan saran yang diperoleh dari peneltian serta pemecahan masalah yang dilakukan untuk pengembangan lebih lanjut.

BAB 2
LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi uraian dan pembahasan tentang teori, konsep, model, metode, yang digunakan sebegai penunjangn peneliti untuk penulisan dokumen ini. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian. Dasar teori yang dibahas diantaranya adalah metode naïve bayes. Kemudian pembahasan tentang pengembangan perangkat lunak, seperti pengembangan model *prototyping,* dan pendekatan berorientasi objek. Setelah itu teknologi yang digunakan dalam mengembangkan sistem ini yaitu Swift 4.

2.1 Bulutangkis

Bulutangkis merupakan permainan yang banyak menggunakan kemampuan fisik dengan gerakan yang cepat dan pukulan keras yang dilakukan dalam waktu beberapa detik di antara reli-reli panjang (Ballou, 1998).

Keterampilan dasar yang diperlukan dalam bulutangkis di antaranya adalah cara memegang raket, sikap berdiri, gerakan kaki, dan memukul *shuttlecoc*k (Grice, 1994; Davis, 1998; Djide, 2000). Dalam kaitannya dengan keterampilan dasar memukul satelkok, seseorang sudah dapat bermain bulutangkis apabila dapat melakukan beberapa keterampilan dasar teknik memukul *shuttlecoc*k, yang terdiri atas *servis, lob, drive, netting, dropshot, dan smash* (Wattanasin, 2000; Han Jian, 2000; Grice, 1994).

Gambar 2.1 Pemain Bulutangkis

(Sumber : Republika.co.id)

2.1.1 Peraturan Permainan

Secara sederhana, permainan bulutangkis adalah upaya untuk memasukkan *shuttlecoc*k ke bidang permainan lawan, tanpa *shuttlecoc*k itu tidak bisa dikembalikan. Ada berbagai cara melakukannya, seperti memasukkan shuttlecock ke bidang yang tidak terjaga lawan, atau memasukkan *shuttlecock*

dengan cepat, sehingga tidak sempat dikuasai atau dikejar lawan. Sebelum pertandingan kedua pemain menjalani undian yang dilakukan wasit, biasanya dengan tos menggunakan mata uang logam. Pemenang boleh memilih lapangan dan melakukan servis pertama kali. Untuk ganda, setelah undian hanya satu orang yang melakukan servis dan begitu gagal mendapat angka, maka servis pun berpindah ke lawan.

Angka diperoleh si pelaku servis, sehingga bila dia gagal, servis berpindah, tidak menggunakan *rally point* seperti di tenis meja atau bola voli. Bila shuttlecock tidak bisa dikembalikan lawan, dia akan mendapat angka. Dalam melakukan servis, prinsip yang harus dipegang adalah kepala raket tidak boleh lebih tinggi dari pinggang, *shuttlecock* dalam keadaan dipegang, dan kaki tidak bergerak mendahului gerakan memukul *shuttlecock* . Sedang penerima servis mengalami *foul* bila bergerak sebelum lawan melakukan *servis*. Bola *shuttlecock* juga menjadi mati bila terpukul dua kali, gagal melewati net, mendarat di luar garis, raket melewati atas net atau menyentuh net, kaki melewati batas garis bidang.

Peraturan Terbaru Badminton untuk sistem 21 poin, Setiap *service over* selalu ada pertambahan angka, pemain selalu menghasilkan angka meski tidak melakukan *service*, jika nilai ganjil (1,3,5,dst) dimulai dari kiri, dan nilai genap (2,4,6,dst) dimulai dari kanan pemain pemegang *service*. Sama seperti tunggal, untuk ganda hanya ada 1 *service* (tidak ada memiliki bola ke dua). Jika skor 20-20, pemenang harus unggul 2 poin berurutan (*consecutive*). Misal: 22-20, 23-21,dst, jika skor 29-29, pemenang adalah yang mencapai angka 30. Untuk aturan teknis lainnya, sama dengan aturan terdahulu.

2.2 Karakter Pemain

Karakter merupakan perpaduan segala tabiat manusia yang bersifat permanen yang merupakan ciri khusus untuk membedakan orang yang satu dengan lainnya, tak terkecuali pemain bulutangkis. Berkarakter artinya menunjukkan sifat memiliki pendirian yang teguh, baik, terpuji dan dapat dipercaya serta memiliki prinsip dalam arti moral. Karakter dalam bahasa Inggris disebut character berarti watak, perangai, tabiat atau sifat. Apabila memaknai karakter adalah watak, maka Freud mengatakan bahwa karakter adalah ”a system of striving with underlie behaviour” (Bastaman,1999). Sistem dan upaya yang melandasi perilaku seseorang bersumber dari faktor genetik dan faktor pengalaman hidup dan pendidikan (Sumarno, 2002). Pengakuan bahwa karakter dilandasi oleh genetik telah dipahami oleh masyarakat.

Karakter pemain bulutangkis dapat berlaku bagi semua pemain bulutangkis tunggal maupun ganda baik putera mapun puteri karakter pemain bulutangkis dapat diketahui dari aksi yang dilakukan di lapangan apa saja pukulan yang dilakukan oleh pemain sehingga pelatih dapat menentukan pemain tersebut memiliki karakteristik seperti apa serta teknik pukulan, dalam pengembangan aplikasi ini karakteristik yang digunakan yaitu karakter berdasarkan aksi pemain di lapangan sehingga dapat terlihat ketika sedang dalam permainan karakteristik

yang dapat ditentukan pada penelitian kali ini yaitu karakteristik penyerang dan bertahan.

2.3 Pengembangan Perangkat Lunak

Ada banyak pembahasan tentang pengembangan perangkat lunak, diantaranya adalah model pengembangan perangkat lunak itu sendiri, model prototyping, waterfall, OO (pendekatan berorientasi objek), dan PBO (pemodelan berorientasi objek).

2.3.1 Karakteristik *Object Oriented Programming* (OOP)

Menurut Dan Clark (2011), beberapa karakteristik yang terdapat dalam OOP adalah:

a. *Object*

   Objek adalah sebuah struktur data untuk menggabungkan data dan prosedur untuk bekerja dengan data. Misalnya, apabila ingin memiliki kemampuan berkendara, maka harus bekerja dengan objek mobil yang bertanggung jawab atas data dan metode yang digunakan dengan mobil tersebut.

b. *Inheritance*

   Dalam OOP *inheritance* digunakan untuk mengklasifikasikan objek dalam program sesuai dengan karakteristik dan fungsi yang umum. Hal itu membuat bekerja dengan objek menjadi lebih mudah dan intuitif. Hal itu juga membuat program menjadi lebih mudah, karena memungkinkan pembuat untuk menggabungkan karakteristik umum kedalam objek *parent* dan mewarisi karakteristik tersebut di dalam objek *child*.

c. *Abstaction*

   Abstraksi yaitu menyaring sifat benda asing sehingga hanya memproses sejumlah informasi dan konsentrasi penting pada tugas yang sedang dilakukan.

d. *Encapsulation*

   Enkapsulasi adalah proses dimana tidak ada akses langsung yang diberikan untuk data, melainkan tersembunyi. Untuk mendapatkan akses ke data, diperlukan interaksi dengan objek yang bertanggung jawab untuk data tersebut.

e. *Aggregation*

   Agregasi adalah ketika sebuah objek terdiri dari gabungan dari benda-benda lain yang bekerja sama. Misalnya, mesin pemotong rumput adalah gabungan dari objek roda, mesin, pisai, dan sebagainya. Bahkan, objek mesin adalah gabungan dari berbagai objek. Kemampuan untuk menggunakan agregasi di dalam OOP merupakan

fitur canggih yang memungkinkan implementasi proses bisnis dan model secara akurat di dalam suatu program.

2.3.2 Model Pengembangan Perangkat Lunak

Ada bebagai macam jenis model proses yang digunakan dalam pengembangan perangkat lunak yang mencakup metodologi konvensional dan inovatif seperti *prototyping, waterfall, incremental and iterative development, rapid application development, spiral development,* dan *extreme programming*.

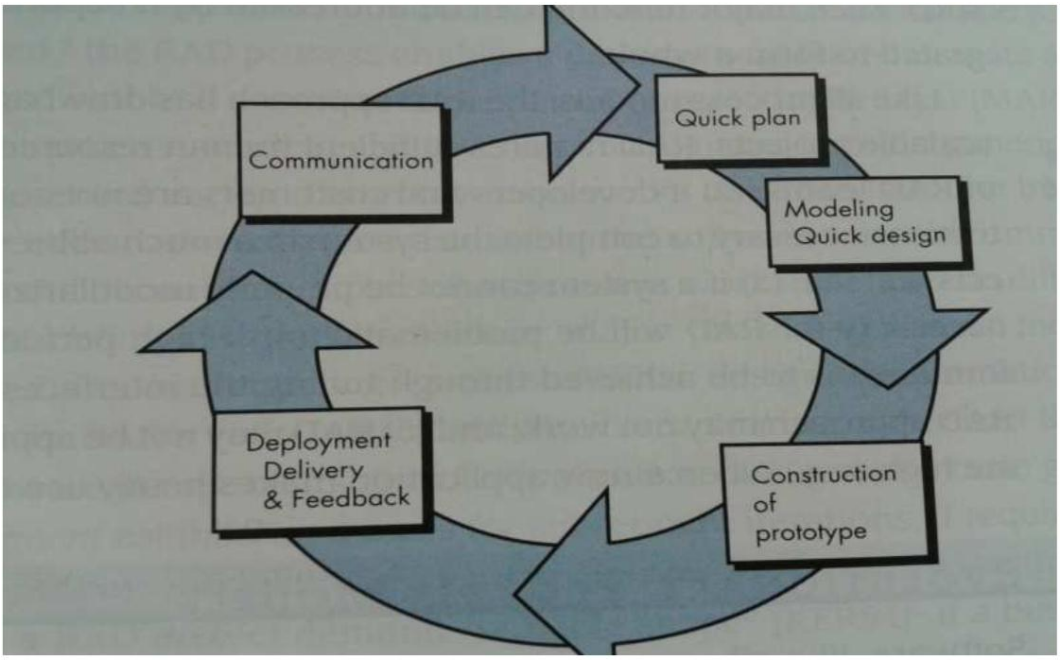
Adapun jenis metodologi memiliki beberapa tahapan sebagai berikut(Caytiles,2014):

1. Tahap Analisis Kebutuhan
2. Tahap Mendesain Sistem
3. Tahap Implementasi (*coding*)
4. Tahap Pengujian
5. Tahap *Deployment*
6. Tahap *Maintenance*

2.3.3 Model *Protoyping*

Dalam proses pengembangan perangkat lunak pemilihan model pengembangan disesuaikan dengan hasi analisis yang didapatkan dari masalah yang dihadapi. Model *prototyping* digunakan karena kebutuhan pengguna yang dapat berubah – ubah, pengguna juga dapat terlibat dalam proses pengembangan yang akhirnya kesalahan dapat terlihat lebih awal seiring berkembangnya kebutuhan untuk menghitung karakteristik pemain bulutangkis.

Pada pelaksanaannya model *prototyping* dapat menerima hal – hal penting yang akan datang seperti tambahan kebutuhan dari pemangku kepentingan sehingga perangkat lunak dapat dimodifikasi kembali, dikembangkan, lalu ditambahkan atau digabungkan dengan sistem yang sudah dibangun sebelumnya. Model *prototying* terdiri dari proses *requirement, System Design, Coding, Testing, Review,* Implementasi dan *Maintenance*(Pressman, 2010). Pada pembangunan aplikasi ini digunakan model *prototyping* dengan iterasi sebanyak minimal dua kali dan hanya sampai proses pengujian *system* tidak sampai *deployment* karna dalam pengembangan ini tidak digunakan semua tahap pada model *prototyping*



Gambar 2.2 Model Pengembangan *Protoyping*

(Sumber: Pressman, 2010)

Pada model pengembangan prototyping sendiri memiliki tahapan-tahapan sebagai berikut:

1. Pengumpulan Kebutuhan
   Pengguna dan pengembang bersama-sama mengindentifikasi dan mendefinisikan seluruh garis besar dan kebutuhan sistem.
2. Pembangunan *Prototyping*
   Membangun *prototyping* dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat input dan format output)
3. Evaluasi *Prototyping*
   Evaluasi ini dilakukan oleh pengguna apakah *prototyping* yang sudah dibangun sesuai dengan keinginan pengguna. Jika sudah sesuai maka langkah 4 akan diambil. Jika tidak prototyping direvisi dengan mengulangi langkah 1, 2, dan 3
4. Pengkodean *System*
   Dalam tahap ini *prototyping* yang sudah disepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai
5. Menguji *System*
   Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus dites dahulu sebelum digunakan. Pengujian ini dilakukan dengan White Box, Black Box, Basis Path, pengujian arsitektur dan lain-lain.
6. Evaluasi sistem
   Pengujian ini dilakukan dengan White Box, Black Box, Basis Path, pengujian arsitektur dan lain-lain. Pelanggan mengevaluasi apakah sistem yang sudah

jadi sudah sesuai dengan yang diharapkan. Jika ya, langkah 7 dilakukan, jika tidak, ulangi langkah 4 dan 5.

7. Menggunakan sistem
   Perangkat lunak yang telah diuji dan diterima pelanggan siap untuk digunakan.

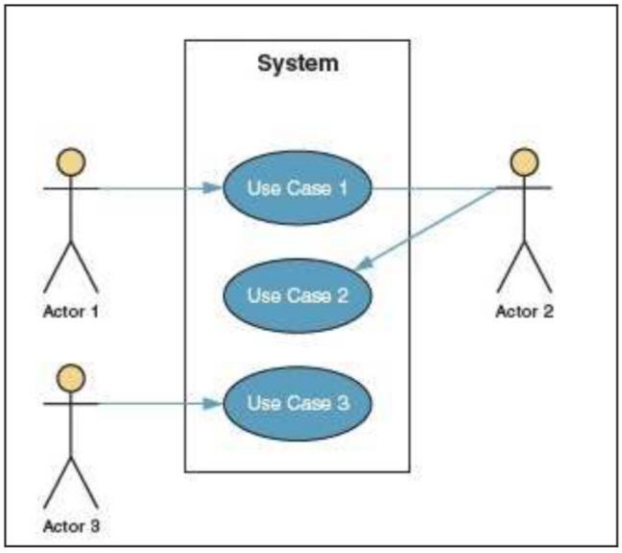
2.4 UML (Unified Modifying Language)

UML adalah suatu konvensi pemodelan yang digunakan untuik menspesifikasikan atau mendeskripsikan sebuah sistem piranti lunak yang terkait dengan objek(Whitten & Bentley, 2007). UML sendiri terdiri dari beberapa tipe diagram, yaitu:

a. *Use Case* Diagram

Use-case diagram adalah diagram yang menggambarkan interaksi antara sistem dan pengguna (Whitten & Bentley, 2007). Dengan kata lain, diagram ini mendeskripsikan siapa yang akan menggunakan sistem itu dengan cara apa pengguna berinteraksi dengan sistem.

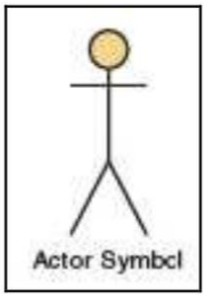
1. *Use Case* adalah urutan langkah-langkah yang secara tindakan saling terkait, baik terotomatisasi maupun secara manual untuk melengkapi suatu tugas bisnis tunggal.



Gambar 2.3 Contoh Diagram *Use Case*

(Sumber: Whitten & Bentley, 2007)

2. Aktor adalah adalah segala sesuatu yang perlu berinteraksi dengan sistem untuk pertukaran informasi.

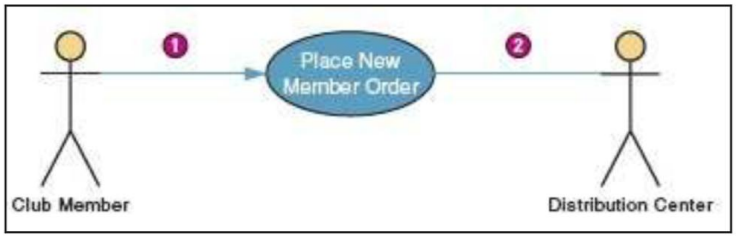


Gambar 2.4 Simbol Aktor

(Sumber: Whitten & Bentley, 2007)

3. Hubungan (*Relationship)* pada diagram *use case* digambarkan sebagai sebuah garis antara dua simbol. Pemaknaan hubungan berbeda-beda tergantung bagaimana garis tersebut digambar dan tipe simbol apa yang digunakan untuk menghubungkan garis tersebut. Beberapa tipe hubungan yang terdapat pada diagram *use case*, yaitu sebagai berikut:

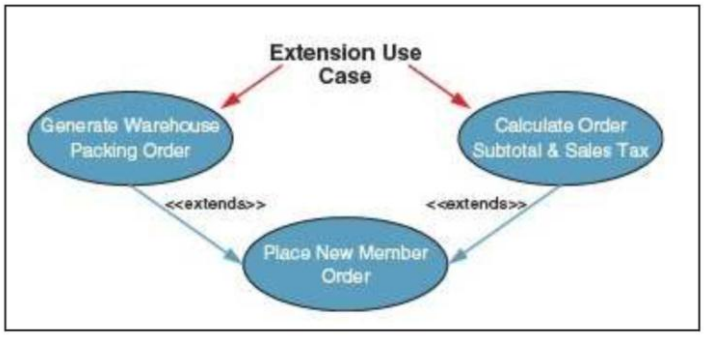
   1) Gabungan (*Association*): hubungan antara pelaku dengan use case, dimana terjadi interaksi di antara mereka.



Gambar 2.5 Contoh Hubungan *Association*

(Sumber: Whitten & Bentley, 2007)

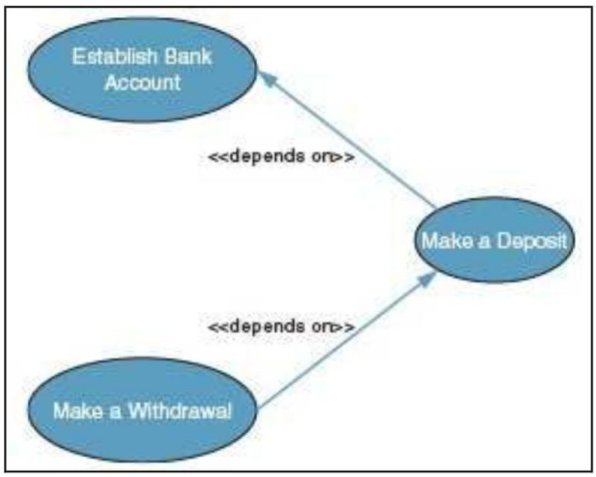
   2) *Extends: use case* yang terdiri dari langkah yang diekstraksi dari *use case* yang lebih kompleks untuk menyederhanakan masalah orisinil dan karena itu memperluas fungsinya.



Gambar 2.6 Contoh Hubungan *Extends*

(Sumber: Whitten & Bentley, 2007)

3) *Depends On: use case* yang memiliki ketergantungan pada *use case* lain untuk menetapkan rangkaian use case yang perlu dikembangkan.



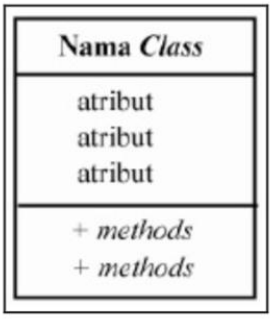
Gambar 2.7 Contoh Hubungan *Depends On*

(Sumber: Whitten & Bentley, 2007)

b. *Class* Diagram

Class diagram menggambarkan struktur objek yang terdapat pada sistem (Whitten & Bentley, 2007). Diagram ini menunjukkan objek yang terdapat pada suatu sistem serta relasi antara objek- objek tersebut.

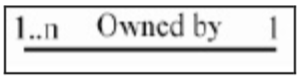
1) *Class* digambarkan sebagai sebuah kotak yang terbagi atas tiga bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan atribut *class*. Bagian akhir mendefinisikan method dari sebuah *class*.



Gambar 2.8 *Class*

(Sumber: Whitten & Bentley, 2007)

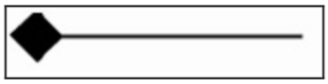
2) Association adalah merupakan sebuah hubungan yang paling umum antara dua *class* dan dilambangkan oleh sebuah garis yang menghubungkan antara dua *class*. Garis ini bisa melambangkan tipe-tipe hubungan. Contohnya *one-to-one, one-to-many, many-to-many.*



Gambar 2.9 *Association*

(Sumber: Whitten & Bentley, 2007)

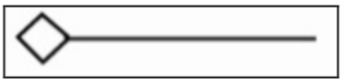
3) *Composition* adalah jika sebuah class tidak bisa berdiri sendiri dan harus merupakan bagian dari class yang lain, maka class tersebut memiliki relasi composition terhadap class tempat dia bergantung tersebut. Sebuah hubungan composition digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.



Gambar 2.10 *Composition*

(Sumber: Whitten & Bentley, 2007)

4) *Agregation* adalah jika sebuah class bisa berdiri sendiri walaupun merupakan bagian dari *class* yang lain.

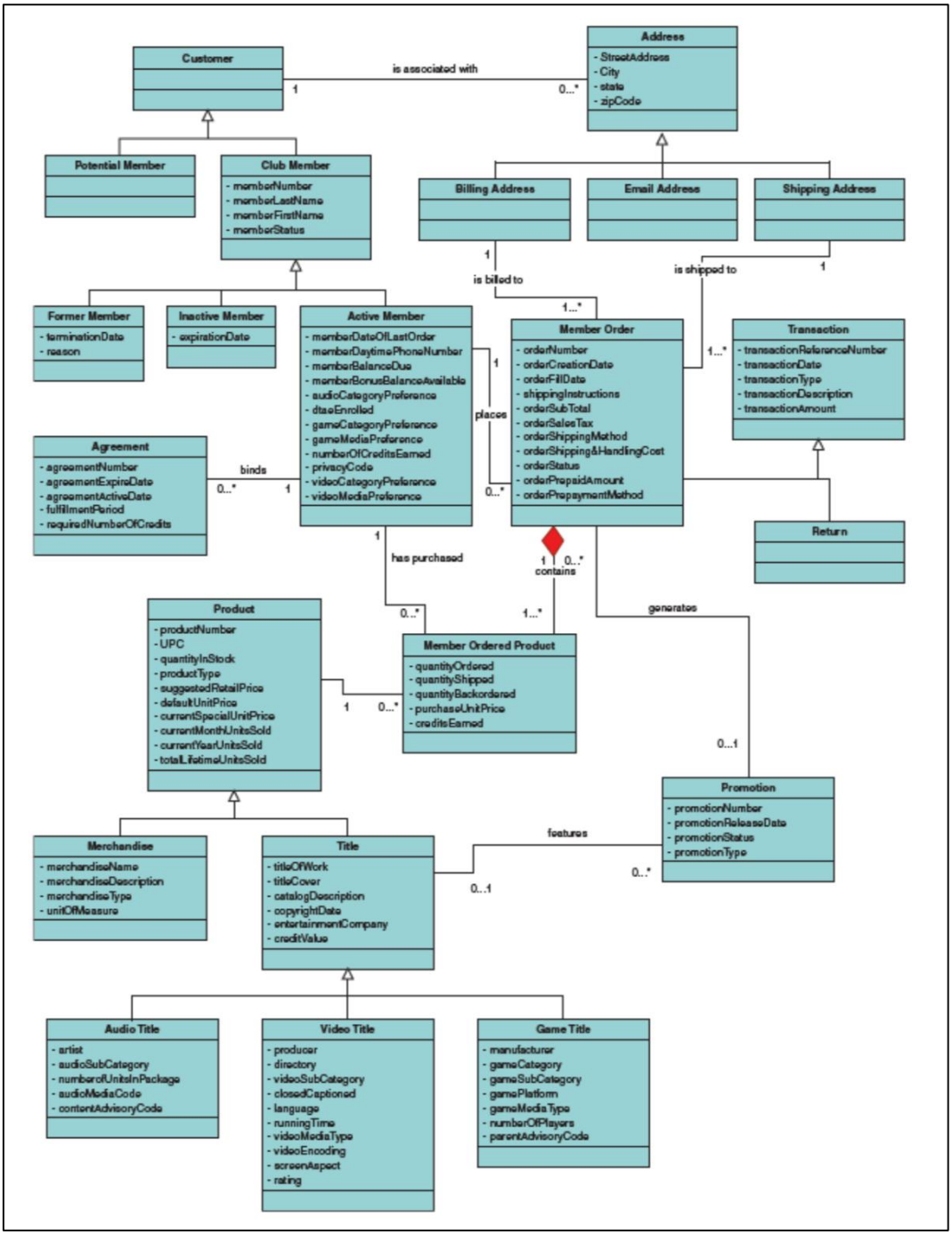


Gambar 2.11 *Agregation*

(Sumber: Whitten & Bentley, 2007)

5) Bagaimana cara mengakses *veasibility* UML menyediakan tiga tingkat dari visibility, yaitu:

   1. Public : ditandai dengan simbol “**+**”
   2. Protected : ditandai dengan simbol “**#**”
   3. Private : ditandai dengan simbol “**-**‘

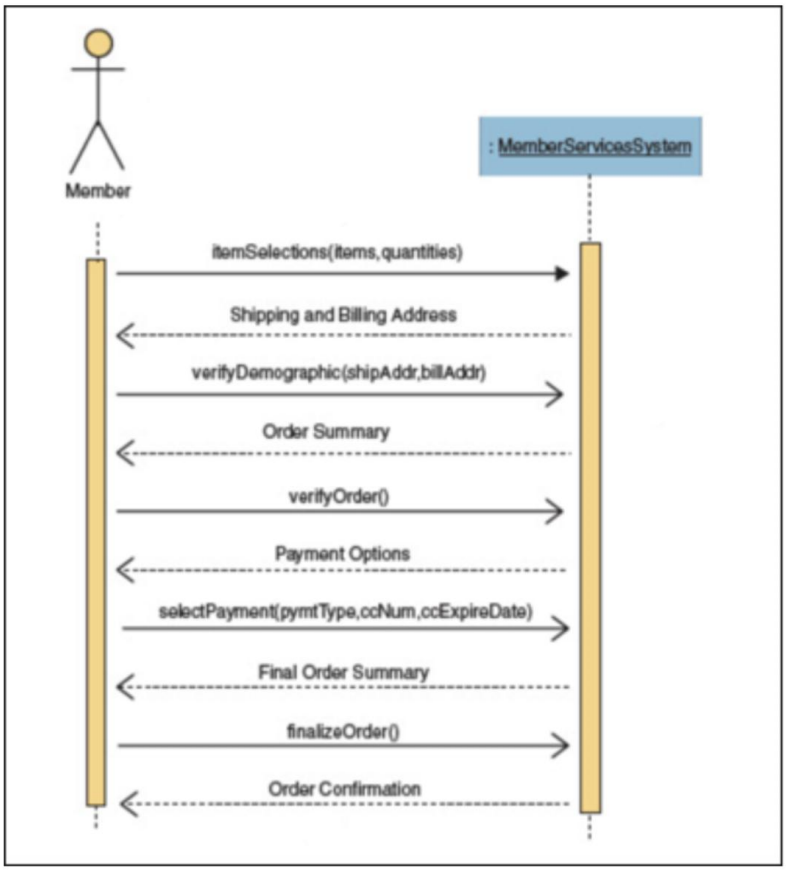


Gambar 2.12 Contoh *Class* Diagram

(Sumber: Whitten & Bentley, 2007)

c. Sequence Diagram

Sequence diagram menggambarkan bagaimana objek-objek berinteraksi satu sama lain melalui pesan dalam eksekusi dari sebuah use case atau operasi (Whitten & Bentley, 2007). Sebuah system sequence diagram membantu untuk mengidentifikasi pesan tingkat tinggi yang masuk dan keluar dari sistem.



Gambar 2.13 Contoh *Sequence* Diagram

(Sumber: Whitten & Bentley, 2007)

Beberapa notasi yang terdapat di dalam system sequence diagram, yaitu:

1. *Actor*, berupa aktor yang memulai pada use case ditunjukkan dengan simbol use case aktor.
2. *System*, berupa kotak yang menunjukkan sistem yang sedang berjalan, ditandai dengan notasi titik dua (:) pada awal nama sistem.
3. Lifelines, berupa garis putus-putus menurun dari simbol aktor dan sistem.
4. Activation bars, berupa balok yang terletak di atas lifelines, menunjukkan periode waktu ketika terdapat interaksi secara aktif.
5. *Input messages*, berupa anak panah mendatar dari aktor ke sistem yang menunjukkan pesan masuk, dimana kata pertama diawali dengan huruf kecil, sedangkan kata berikutnya diawali dengan huruf besar dan tidak ada spasi, diikuti dengan tanda kurung yang berisi parameter yang dibutuhkan.
6. *Output messages*, berupa anak panah mendatar dengan garis putus-putus dari sistem ke aktor yang menunjukkan pesan keluar.
7. Receiver *Actor*, berupa aktor lain yang menerima pesan dari sistem.
8. *Frame*, berupa kotak yang menambahkan pesan terpisah untuk menunjukkan perulangan, alternative, atau pilihan.

2.5 Teknologi Pengembangan Sistem

2.5.1 Naïve Bayes Untuk Klasifikasi

Teorema Naïve Bayes adalah metode pengklasifikasian dengan menggunakan konsep peluang, dimana diasumsikan bahwa setiap atribut contoh (data sampel) bersifat saling lepas satu sama lain berdasarkan atribut kelas.

Metode klasifikasi ini diturunkan dari penerapan teorema Bayes dengan asumsi independence (saling bebas). Asumsi ini disebut class conditional independence yang dibuat untuk memudahkan perhitungan-perhitungan pengertian ini dianggap "naive", dalam bahasa lebih sederhana naïve itu mengasumsikan bahwa kemunculan suatu term kata dalam suatu kalimat tidak dipengaruhi kemungkinan kata-kata yang lain dalam kalimat padahal dalam kenyataanya bahwa kemungkinan kata dalam kalimat sangat dipengaruhi kemungkinan keberadaan kata-kata yang dalam kalimat (Surya, 2009).

Untuk mengilustrasikan pendekatan ini, pertimbangkan bahwa peminjam pinjaman tidak dapat membayar tagihannya. Data pada table 2.1 menunjukkan data *training* dengan atribut: Pemilik Rumah, Status Pernikahan dan Pendapatan Tahunan. Peminjam pinjaman yang gagal membayar pembayaran mereka diklasifikasikan sebagai Ya, sementara mereka yang melunasi pinjaman mereka diklasifikasikan sebagai No.

Tabel 2.1 Data *Training* untuk memprediksi masalah peminjaman

No	Pemilik Rumah	Status Pernikahan	Pendapatan Tahunan	Gagal Membayar
1	Iya	Belum Menikah	$125.000	Tidak
2	Tidak	Menikah	$100.000	Tidak
3	Tidak	Belum Menikah	$70.000	Tidak
4	Iya	Menikah	$120.000	Tidak
5	Tidak	Cerai	$95.000	Iya
6	Tidak	Menikah	$60.000	Tidak
7	Iya	Cerai	$220.000	Tidak
8	Tidak	Belum Menikah	$85.000	Iya
9	Tidak	Menikah	$75.000	Tidak
10	Tidak	Belum Menikah	$90.000	Iya

Misalkan kita diberi catatan uji dengan atribut berikut: X: (Pemilik Rumah: Tidak, Status Perkawinan: Menikah, Penghasilan Tahunan: $ 120.000). Untuk mengklasifikasikan catatan, kita perlu menghitung probabilitas posterior

$P(Iya|X)$ dan $P(Tidak|X)$ berdasarkan informasi yang tersedia dalam data pelatihan. Jika $P(Iya|X) > P(Tidak|X)$, maka catatannya adalah klasemen Iya; Jika tidak, ini adalah klasifikasi $Tidak$ (Tan, 2006).

Memperkirakan probabilitas posterior akurat untuk setiap kemungkinan kombinasi untuk label kelas dan nilai atribut adalah masalah yang sulit. Teorema Bayes berguna karena memungkinkan kita untuk mengekspresikan probabilitas posterior dalam hal probabilitas sebelumnya $P(Y)$, probabilitas bersyarat kelas $P(X|Y)$, dan buktinya $P(X)$:

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)}$$

Bila membandingkan probabilitas posterior untuk nilai Y yang berbeda, penyebut $P(X)$ akan selalu konstan, dan dengan demikian dapat diabaikan. Kemungkinan sebelumnya $P(Y)$ dapat dengan mudah diperkirakan dari pelatihan yang ditetapkan dengan menghitung fraksi catatan pelatihan yang termasuk dalam setiap kelas. Untuk memperkirakan probabilitas bersyarat kelas $P(X|Y)$ (Tan, 2006).

Untuk memprediksi label kelas dari catatan uji X : (Pemilik Rumah : Tidak, Status Perkawinan: Menikah, Penghasilan: $ 120.000), kita perlu menghitung probabilitas posterior $P(Tidak|X)$ dan $P(Iya|X)$. Probabilitas posterior ini dapat diestimasi dengan menghitung antara probabilitas $P(Y)$ sebelumnya dan probabilitas bersyarat kelas $\Pi_i P(X_i|Y)$ yang sesuai dengan pembilang dalam Tabel 2.2.

Tabel 2.2 Perhitungan untuk klasifikasi naïve bayes masalah peminjaman

P (Pemilik Rumah = Iya | Tidak) = 3/7
P (Pemilik Rumah = Tidak | Tidak) = 4/7
P (Pemilik Rumah = Iya | Iya) = 0
P (Pemilik Rumah = Tidak | Iya) = 1
P (Status Pernikahan = Belum Menikah | Tidak) = 2/7
P (Status Pernikahan = Cerai | Tidak) = 1/7
P (Status Pernikahan = Menikah | Tidak) = 4/7
P (Status Pernikahan = Belum Menikah | Tidak) = 2/3
P (Status Pernikahan = Cerai | Tidak) = 1/3
P (Status Pernikahan = Menikah | Tidak) = 0
Kelas Tidak :
Sample mean = 110
Sample variance = 2975

Kelas Iya : *Sample* mean = 90 *Sample* variance = 25

Probabilitas sebelumnya dari masing-masing kelas dapat diestimasi dengan menghitung fraksi catatan pelatihan yang dimiliki oleh masing-masing kelas. Karena ada tiga catatan yang termasuk dalam kelas Iya dan tujuh catatan yang menjadi milik kelas Tidak (Tan, 2006).

P(Iya) = 0.3 dan P(Tidak) = 0.7 dengan menggunakan informasi yang ada di kelas kondisi pada tabel 2.2 kita dapat menghitung seperti dibawah ini:

P (X|Tidak) = P (Pemilik Rumah = Tidak | Tidakl) × P (Status Pernikahan = Menikah

| Tidak) × P(Pendapatan Tahunan = \$120.000 | Tidak)

$= 4/7 \times 4/7 \times 0.0072 = 0.0024$

P(X|Iya) = P(Pemilik Rumah = Tidak | Iya) × P(Status Pernikahan = Menikah|

Iya) × P(Pendapatan Tahunan = \$120.000 | Iya)

$= 1 \times 0 \times 10^{-9} = 0$

Probabilitas posterior dari kelas Tidak adalah P(X|Iya) = $\alpha \times \frac{7}{10} \times 0.0024 = 0.0016\alpha$ dimana $\alpha = 1 / P(X)$ adalah bilangan konstan. Dengan pendekatan yang sama kita dapat menunjukkan bahwa probabilitas posterior kelas Iya adalah 0 karna probabilitas bersyarat kelasnya adalah 0. Karna $P(Tidak|X) > P(Iya|X)$ data tersebut termasuk kedalam kelas Tidak (Tan, 2006).

2.5.2 Swift 4

Swift merupakan bahasa pemrograman yang baik dan intuitif untuk macOS, iOS, watchOS and tvOS. Syntax pada Swift singkat namun ekspresif, swift juga memiliki fitur modern yang disukai oleh pengembang. Swift merupakan bahasa yang aman dengan desain yang baik untuk membangun sebuah perangkat lunak yang dapat berjalan secepat kilat(Apple 2017).

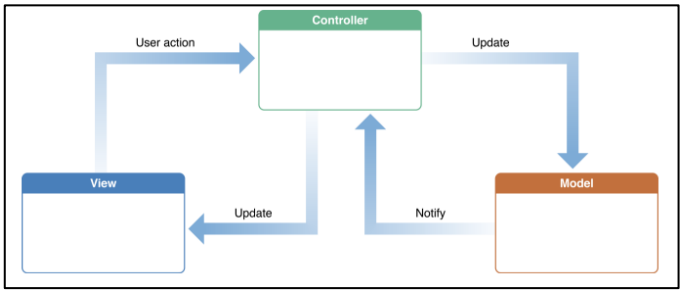
Swift mencakup fitur yang membuat kode lebih mudah dibaca dan ditulis, sekaligus memberi pengembang kontrol yang dibutuhkan dalam bahasa pemrograman sistem yang sebenarnya. Memori dikelola secara otomatis, dan tidak perlu menuliskan *semi-colons*. Beberapa kode Swift juga menggunakan dari bahasa lain yaitu Objective-C diekspresikan dalam sintaks bersih yang membuat

API di Swift mudah dibaca dan dipelihara(Swift Org, 2017). Beberapa fitur dari Swift 4 :

- Tuple dan nilai kembalian lebih dari satu
- Generik
- Mendukung metode, ekstensi, dan protokol
- Penanganan kesalahan yang lebih baik dan *bult-in*
- Kontrol *advance* dengan *do, guard, defer* dan *repeat keywords.*

2.5.2.1 Arsitektur Aplikasi iOS

Arsitektur aplikasi pada iOS secara *default* menggunakanan arsitektur MVC (*Model-View-Controller).* Model seperti ini Arsitektur ini memisahkan *model* yang bertanggungjawab sebagai *layer* untuk menangani data dan manipulasi data, *view* yaitu *layer* yang bertanggungjawab untuk mengatur tampilan aplikasi atau antar muka pengguna (UI), dan *controller* yaitu *layer yang bertanggungjawab* untuk menghubungkan antar layer *view* dan layer *model,* layer ini berguna untuk mengubah model dan menginterpretasikan input atau aksi dari pengguna dari *view layer*.



Gambar 2.14 Diagram Arsitektur MVC

Sumber: Apple, 2017

User Action merupakan sebuah masukan dari kelas view yang memberikan masukan kepada kelas *controller* lalu klas *controller* akan memberikan *update* kepada klas *model* yang akan mengubah data, setelah itu klas *model* akan memberikan *notify* atau akan memberikan pemberitahuan bahwa data telah diubah, setelah itu klas *controller* akan memberikan *update* kepada klas view yang akan memberikan tampilan kepada pengguna bahwa aplikasi sudah merespon dengan meng-*update* tampilan.

2.6 Teori Pengujian

2.6.1 *Blackbox Testing*

Black-box testing merupakan pengujian yang berfokus terhadap persyaratan dari perangkat lunak yang memungkinkan penguji untuk mendapatkan set kondisi input yang secara keseluruhan melakukan persyaratan fungsional untuk sebuah program (Pressman, 2010). Dalam kategori yang diuji *black-box testing*

antara lain fungsi yang tidak sesuai dengan fungsional ataupun fungsi yang hilang dan antara lain antarmuka, struktur data, perilaku (behavior) dan inisialisasi dan pemutusan. *Black-box testing* bertujuan untuk menemukan kesalahan-kesalahan fungsi yang tidak benar atau belum ada, kesalahan antarmuka, kesalahan struktur data atau akses database, kesalahan kinerja, kesalahan inisialisasi dan kesalahan terminasi.

2.6.2 *Whitebox Testing*

White-box testing adalah pendekatan dalam pengujian program dimana tes didasarkan pada pengetahuan tentang struktur program dan komponennya (Pressman, 2010). Pada *white-box* testing pengujian didasarkan pada kasus uji yang diperoleh melalui pencarian jalur independen kode program. Jalur independen kode program dapat diketaui melalui perancangan *flow graph*. *Flow graph* adalah diagram yang menggambarkan alur kode program yang terdiri dari node (simpul) yang biasanya direpresentasikan dalam bulatan berisi data (biasanya angka yang merepresentasikan urutan alur program dan edge yaitu penghubung antar node yang memiliki arah pada satu simpul tertentu.

2.6.3 *Usability Testing*

Usability Testing dilakukan untuk melakukan pengujian sistem dengan cara menguji sistem secara langsung kepada sejumlah responden untuk mengetahui tingkat kemudahan penggunaan sebuah sistem. Pengujian ini dilakukan setelah sistem sudah siap digunakan sehingga diperoleh penilaian berdasarkan hasil dan analisis responden yang dihitung berdasarkan metode yang digunakan. Dalam pengujian usabilitas kali ini digunakan metode *System Usability Scale* (SUS) (Brooke, 1986).

Pengujian usabilitas dilakukan untuk melakukan pengujian sistem dengan cara menguji sistem secara langsung kepada sejumlah responden untuk mengetahui tingkat kemudahan penggunaan sebuah sistem. Responden akan diminta untuk mengisi sebuah kuisioner berisi sejumlah pernyataan seputar kenyamanan dan kemudahan dalam menggunakan sistem yang telah dikembangkan.Dalam pengujian disabilitas tedapat tiga aspek pengujian yaitu dalam aspek efektivitas, efisiensi dan kepuasan pengguna(ISO 9241 – 11, 1998), akan tetapi dalam pengujian kali ini pengujian usabilitas yang dilakukan hanya pada aspek efektivitas yaitu adalah tingkat akurasi dan kelengkapan pengguna dilakukan untuk mencapai tujuan yang ditetapkan. Adapun cara penghitungan metrik efektivitas dapat dilihat pada persamaan 6.1 dibawah ini(Usabilitygeek, 2015)

$$Efektivitas = \frac{jumlah\ task\ yang\ berhasil\ dilakukan}{total\ jumlah\ task\ yang\ diberikan} \times 100\% \qquad \textbf{(Persamaan 6.1)}$$

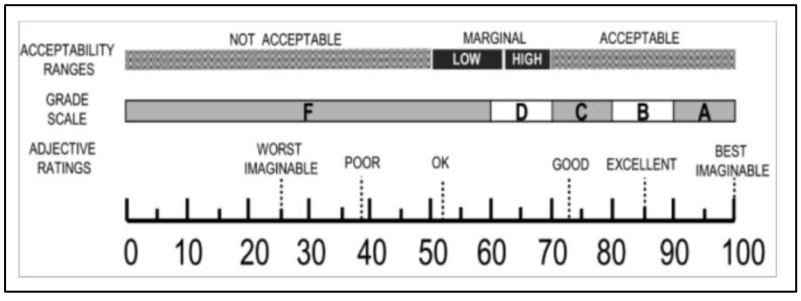
2.6.3.1 System Usability Scale (SUS)

System Usability Scale (SUS) merupakan sebuah pengujian pada sisi pengguna yang menyediakan alat ukur yang *"quick and dirty"* dan dapat diandalkan(Brooke, 1986). Pada *SUS* metode ini menggunakan 10 pernyataan berbentuk kuisoner yang dijawab dengan 5 opsi jawaban untuk setiap pernyataannya, mulai dari Sangat Setuju hingga Sangat Tidak Setuju dengan nilai liker 0 sampai 4. Metode ini pertamakali diperkenalkan oleh John Brooke pada tahun 1986 yang digunakan untuk mengevaluasi berbagai jenis produk ataupun servis, termasuk *hardware, software*, perangakat *mobile, website* bahkan aplikasi.

SUS terdiri dari 10 pertanyaan dengan menggunakan skala *likert* 1 sampai dengan 5 sesuai pada tabel 6.263. Pertanyaan nomor ganjil (1, 3, 5, 7, 9) merupakan pertanyaan dengan bernada positif. Sedangkan pertanyaan nomor genap (2, 4, 6, 8, 10) merupakan pertanyaan dengan bernada negatif seperti yang ditunjukkan pada tabel 2.263 Setiap pertanyaan diberi bobot antara 0-4. Penilaian dilakukan dengan pengurangan 1 nilai dari respon yang diberikan user, untuk pertanyaan ganjil dan 5 dikurangi dengan respon yang diberikan user untuk pertanyaan genap. Setelah itu skor SUS didapat dengan cara mengkalikan total skor dengan 2.5. Skor akhir SUS akan berada pada kisaran 0-100. Sesuai sengan gambar 6.16 berdasarkan skor akhir SUS tersebut akan bisa diketahui seberapa tinggi tingkat usability dan akseptabilitas (*acceptable*) desain sistem aplikasi yang dikembangkan. Penilaiannya berdasarkan tiga kategori yaitu *Not Acceptable* dengan rentang skor SUS 0 - 50.9, Marginal 51 - 70.9, dan Acceptable 71 - 100. Responden diminta menjawab semua butir pertanyaan yang diisi setelah pengguna selesai menggunakan sistem secara keseluruhan.

Tabel 2.3 Keterangan Skor Skala Likert Tiap Pernyataan SUS

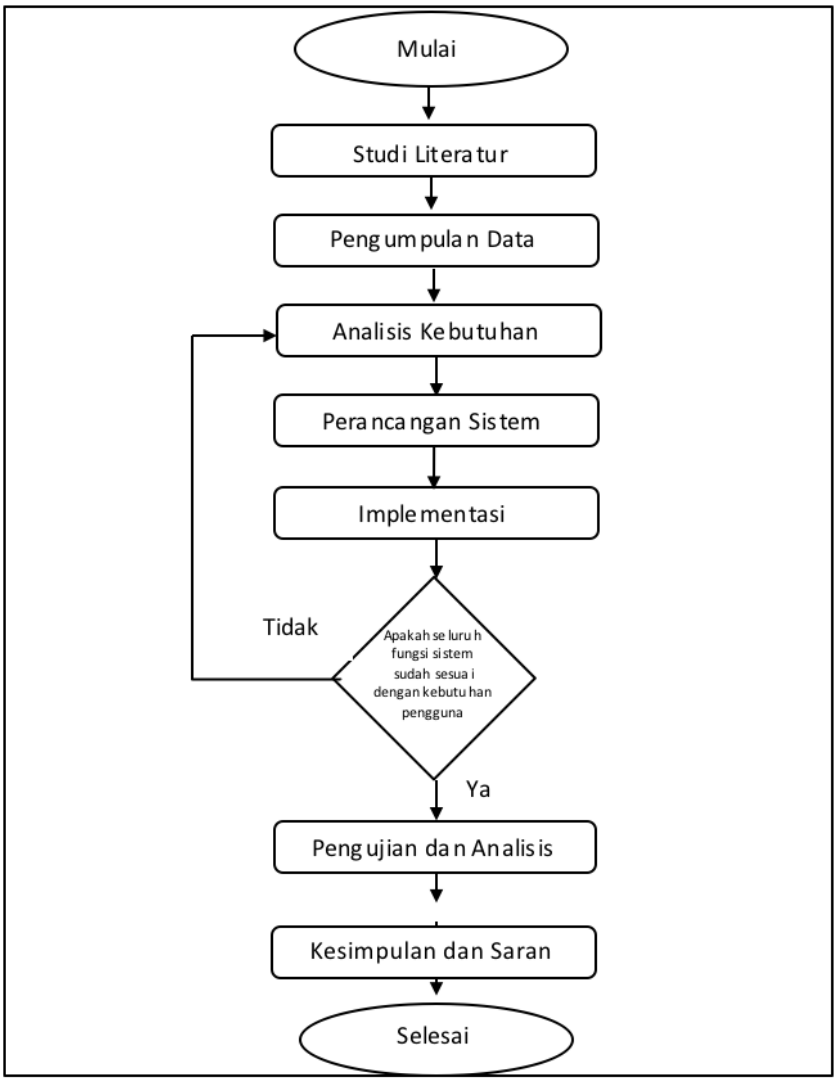
No	Keterangan
1	Sangat Tidak Setuju
2	Tidak Setuju
3	Netral
4	Setuju
5	Sangat Setuju



Gambar 2.15 Rating dan skala konversi skor rata-rata SUS

BAB 3
METODOLOGI

Bab ini menjelaskan mengenai metode penelitian yang dilakukan dalam pengembangan aplikasi Badstics untuk menentukan karakteristik pemain bulutangkis. Metodologi yang digunakan dalam penelitian ini mengadopsi SDLC *prototyping* dengan dua kali iterasi. Beberapa hal dalam metodologi penelitian yaitu tahapan penelitian, studi literature, pengumpulan dan analisis data, analisis dan kebutuhan sistem, perancangan sistem, implementasi dan pembahasan, pengujian dan analisis, pengambilan kesimpulan dan saran.



Gambar 3.1 Diagram Metodologi Penelitian

3.1 Studi Pustaka

Studi literatur menjelaskan dasar teori yang digunakan didapat dari penelitian yang pernah dilakukan sebelumnya, buku, *ebook*, *journal,* dan beberapa literatur

dari internet untuk menunjang penulisantugas akhir. Berikut teori-teori pendukung tersebut meliputi:

1. Metode Naïve Bayes
2. Rekayasa Perangkat Lunak
3. Pengenalan Swift 4
4. Pengenalan Bulut Tangkis
5. Pengenakan Karakteristik Pemain Bulu Tnagkis
6. Pengengembangan Perangkat Lunak
7. Model *Prototyping*
8. UML(*Unified Modeling Language*)

3.2 Pengumpulan Data

Pada penelitian ini data diperoleh dari PBSI Jawa Timur yang berada di Surabaya, Jawa timur. Data yang digunakan berupa video rekaman pertandingan bulutangkis yang sudah pernah berlangsung, hasil perhitungan karakteristik pemain, dan perhitungan untuk menentukan karakter pemain bulu tangkis. Pertandingan bulutangkis yang digunakan adalah pertandingan Tunggal Putra dan Tunggal putri dengan permainan yang diselesaikan dalam dua set, nantinya data ini digunakan sebagai data uji. Observasi juga dilakukan untuk mendapatkan kebutuhan dalam pembangunan aplikasi Badstics ini yaitu dilakukan dengan dengan metode survey. Survey merupakan penelitian yang mengambil sampel dari satu populasi dan menggunakan kuesioner sebagai alat pengumpul data yang pokok (Singarimbun,1991). Pada survey kali ini dilakukan pada 2 staff PBSI, 5 pelatih bulutangkis, dan 10 atlet bulutangkis di PBSI Jawa Timur. Pada survey kali ini digunakan wawancara terstruktur, wawancara terstruktur yaitu dengan memberikan satu set pertanyaan kepada responden, alasan digunakannya metode survey terstruktur adalah bisa direplikasi kepada narasumber lain, mudah untuk dianalisis dan potensi bias rendah.

3.3 Analisis Data

Analisis data betujuan untuk mendapatkan struktur data penyimpanan yang akan diimplemntasikan pada aplikasi Badstics, data yang akan digunakan dalam aplikasi Badstics iniberdasarkan analisis sebagai berikut:

a. Data player yang merupakan representasi data dari pemain yang memiliki beberapa *attribute age, characteristic, height, name, weight.*

b. Data match yaitu data yang menyimpan semua data pertandingan yaitu *match_time, match_date dan Game log.*

c. Log merupakan data dari semua *history* dari permainan yaitu berupa Model Log.

d. Action merupakan data yang menyimpan semua aksi pemain yang ada yaitu sebanyak 28 aksi pemain di lapangan.

3.4 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperkukan dari *system* yang akan dibangun. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirement*) *system* dan siapa saja yang terlibat dialamnya. Analisis juga dilakukan untuk mengetahui kondisi yang ada dilapangan sehingga dapat diketahui implementasi perangkat lunak yang akan digunakan(Andi Nugroho, 2009).

Dari proses ini akan menghasilkan suatu gambaran sistem yang dapat memberikan kemudahan dalam memahami sistem yang dibuat serta proses-proses selanjutnya. Analisis dilakukan dengan dua kali iterasi, proses analisis kebutuhan aplikasi Badstics untuk menentukan karakteristik pemain ini meliputi:

1. Elisitasi Kebutuhan
   Pada tahap ini elisitasi kebutuhan dilakukan proses wawancara untuk mengetahui apa saja kebutuhan sistem yang akan dibangun. Wawancara akan dilakukan kepada Staff PBSI Jawa Timur Bapak Purnomo, pelatih bulutangkis, dan atlet bulutangkis. Hasil wawancara dapat dilihat pada Lampiran A.
2. Spesifikasi kebutuhan
   Pada proses ini kebutuhan yang sudah didapatkan lebih di spesifikkan. Dalam proses ini juga melakukan identifikasi aktor yang akan menggunakan aplikasi ini dan juga kebutuhan yang sudah didapatkan kedalam dua jenis kebutuhan yaitu fungsional dan non fungsional untuk mengetahui secara rinci aktifitas apa saja yang bisa dilakukan oleh pengguna maupun sistem dan apa saja batasan-batasan sistem yang akan dibangun. Serta dilakukan pemodelan kebutuhan yang berupa *use case* diagram dan use case scenario.
3. Manajemen Kebutuhan
   Pada proses ini dilakukan kontrol terhadap kebutuhan yang sudah didefinisikan dan memberikan kode pada kebutuhan dalam menulis kebutuhan sistem dengan kode SRS_BDS_F_1 untuk kebutuhan fungsional dan SRS_BDS_NF_1 untuk kebutuhan non fungsional.

3.5 Perancangan Sistem

3.5.1 Perancangan

Perancangan dibuat sebagai acuan dalam implementasi dan pengujian yang nantinya akan dilakukan setelah pembangunan sistem. Analisis dilakukan dengan dua kali iterasi, berikut tahapan-tahapan perancangan dalam membangun sistem ini:

1. Perancangan Arsitektur
   Dalam perancangan arsitektur ini akan dilakukan pemodelan dengan menggunakan diagram UML seperti sequence diagram dan class diagram.

2. Perancangan Komponen
   Dalam perancangan komponen ini akan dituliskan beberapa sampel algoritme-algoritme utama yang diambil dari setiap klas pada *controller*. Algoritma ini akan ditulis dalam bentuk *pseudocode*.
3. Perancangan Data
   Dalam perancangan data memuat rancangan tabel *database* yang menjelaskan proses normalisasi dari ERD (Entity Reational Diagram). Perancangan data ini akan menjadi dasar dalam implementasi *database* pada sistem.
4. Perancangan Antarmuka
   Perancangan antarmuka dari sistem yang akan dibangun ini terdiri dari tata letak komponen yang harus disediakan oleh sistem berdasarkan kebutuhan sistem. Dalam perancangan antarmuka ini akan dituliskan beberapa sampel antarmuka utama berdasarkan level pengguna. Perancangan antarmuka ini akan menjadi dasar dari implementasi antarmuka sistem.

3.6 Implementasi

Implementasi sistem adalah fase pembangunan sistem yang mengacu pada perancangan sistem. Implementasi dilakukan dengan dua kali iterasi, dalam implementasi sistem ini dibangun menggunakan Swift 4 Proses yang ada dalam implementasi sistem ini antara lain:

1. Implementasi sistem dilakukan dengan mengacu pada spesifikasi sistem yang digunakan untuk membangun apliksi ini.
2. Implementasi basis data dilakukan berdasarakan kebutuhan yang dilakukan pada analisis kebutuhan, data yang diimplementasikan menggunakan Core Data yang dimiliki oleh iOS sehingga bisa langsung digunakan.
3. Implementasi kode program mengacu pada perancangan komponen yang memuat algoritma-algoritma pada perancangan sistem. Logika-logika program tersebut iimplementasikan menggunakan bahasa pemrograman Swift 4. Logika program ini akan diterapkan pada layer controller dan juga bisa diterapkan pada layer model.
4. Implementasi antarmuka, Implementasi ini dibuat berdasarkan perancangan antarmuka pada perancangan sistem. Implementasi ini menggunakan bahasa pemrograman Swift 4.

3.7 Pengujian dan Analisis

Tujuan dari pengujian sistem adalah untuk mengetahui apakah sistem telah berjalan sesuai dengan yang diharapkan. Dengan meminimalisir bahkan menghilangkan masalah-masalah yang muncul pada sistem yang sudah dibangun. Strategi pengujian sistem yang akan dibangun ini mempunyai beberapa tahapan yang harus dilakukan seperti yang diungkapkan oleh Pressman (2010) sebagai berikut:

1. *Unit Testing*
   Pengujian yang dilakukan dengan menguji setiap unit seperti *class*, komponen atau objek dari perangkat lunak yang telah didefinisikan pada perancangan. Teknik basis *path* digunakan untuk menguji kode program berdasarkan algoritma yang terdapat pada setiap fungsi pada setiap kelas yang pengujiannya menggunakan tiga sample uji, teknik basis path merupakan pengujian white box pada pengujian ini cara pengujian dilakukan dengan melihat kedalam unit atau modul untuk melihat kode – kode program yang ada dan menganalisis apakah ada kesalahan atau tidak. Jika terdapat unit atau modul yang ketika dieksekusi dan hasilnya tidak sesuai dengan proses bisnis yang diharapkan maka baris – baris program, variable, parameter dan fungsi yang terlibat dalam unit tersebut dicek satu – persatu lalu diperbaiki dan dijalankan kembali.
2. *Integration Testing*
   Pengujian ini dilakukan untuk melakukan pembuatan uji fungsi sebelum pembuatan fungsi aslinya setelah itu fungsi aslinya dibuat sesuai dengan kebutuhan yang diinginkan, lalu hasil dari fungsi yang dibuat sebelum fungsi implementasi dilakukan pemeriksaan apakah fungsi sudah sesuai dengan fungsi yang ingin dibuat sebelumnya.
3. *Usability Testing*
   Pengujian ini dilakukan untuk memastkan bahwa sistem yang dibangun sudah sesuai dengan dapat digunakan dengan mudah, mudah dipahami oleh *user* dan mudah untuk dioperasikan oleh *user* nantinya. Pengujian ini dilakukan ketika sistem sudah siap digunakan dan digunakan oleh pengguna sehingga pengguna dapat merasakan akan kemudahan menggunakan sistem dan juga sistem mudah dipelajari dan dipahami.
4. *Validation Testing*
   Pada pengujian ini dilakukan pemeriksaan bahwa semua fungsi pada perangkat lunak sudah dibuat dengan benar. *Validation Testing* merupakan serangkaian seri uji coba *black* box *testing* yang menunjukkan sesuai dengan kebutuhan. *Black Box Testing* adalah uji coba perangkat lunak dengan cara menjalankan atau dengan mengeksekusi model atau unit kemudian dilakukan pengamatan pada hasil dari eksekusi unit atau modul apakah sudah sesuai dengan proses bisnis yang diinginkan. Dalam pengujiannya *black box testing* ini berusaha untuk menemukan kesalahan – kesalahan dalam kategori seperti fungsi – fungsi yang tidak benar atau hilang, ketidaksesuaian pada pembuatan *interface*, struktur data, akses *database*, dan kinerja.

3.8 Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan peracangan perangkat lunak. Kesimpulan diambil dari hasil pengujian dan anaisis terhadap system yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksud untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan perancangan perangkat lunak lebih lanjut.

BAB 4
ANALISIS KEBUTUHAN

Dalam analisis kebutuhan yang dilakukan pada bab ini menjelaskaan gambaran umum sistem yang akan dibangun, aktor yang akan menggunakan aplikasi dan kebutuhan dalam membangun sistem.

4.1 Gambaran Umum Aplikasi

Aplikasi Badstics ini merupakan aplikasi yang dibuat bertujuan untuk memberikan informasi mengenai karakteristik pemain bulutangkis, pelatih dapat memasukkan action sesuai apa yang dilakukan pemain ketika sedang bertanding di lapangan, terdapat 30 jenis aksi yang dapat dimasukkan kedalam sistem setelah itu action yang telah diinputkan oleh pelatih akan diproses oleh sistem dengan mengimplementasikan algoritma naïve bayes untuk menentukan karakteristik pemain bulutangkis dengan menggunakan data training yang telah ada didalam aplikasi dan membandingkan dengan masukan yang diberikan pelatih sehingga aplikasi Badstics dapat memberikan informasi karakteristik kepada pelatih yang selanjutnya dapat digunakan pelatih sebagai informasi yang dapat menentukan langkah apa yang harus diambil oleh pelatih untuk diberikan kepada pemain bulutangkis.

4.2 Identifikasi Aktor

Aktor merupakan seseorang ataupun sistem yang dapat berinteraksi dengan sistem. Adapun aktor dalam aplikasi ini ditujukkan pada tabel 4.1.

Tabel 4.1 Identifikasi Aktor

Aktor	**Deskripsi**
Pelatih	Merupakan seseorang yang faham dengan 30 aksi pemain bulutangkis di lapangan sehingga dapat memberikan masukan kedalam sistem sesuai apa yang dilihat pada pertandingan bulutangkis yang sedang berlangsung

4.3 Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem merupakan layanan dari sebuah sistem yang harus tersedia, bagaimana sistem seharusnya merespon pada masukan tertentu dan bagaimana sistem harus melakukan apa pada situasi tertentu (Kurniawan, 2016). Kebutuhan fungsional harus mencakup apa saja yang harus diselesaikan oleh sistem untuk dapat menyelesaikan masalah yang telah ditentukan

sebelumnya. Kebutuhan fungsional juga harus diberikan suatu identitas untuk mempermudah proses identifikasi kebutuhan juga untuk menjaga konsistensi yang terkait dengan kebutuhan suatu sistem, mulai dari proses perancangan hingga pengujian sistem selesai dilakukan. Adapun kebutuhan fungsional sistem serta spesifikasinya ditunjukkan pada Tabel 4.2. Setiap Kebutuhan akan diberikan kode SRS-BDS-F-X untuk kebutuhan fungsional dan SRS-BDS-NF-X untuk kebutuhan non fungsional. SRS merupakan *System Requirement Spesification*, BDS merupakan singkatan dari *Badstics*, F untuk kebutuhan fungsional, NF untuk kebutuhan non fungsional dan X menunjukkan nomor dari definisi kebutuhan utama.

Tabel 4.2 Kebutuhan Fungsional Sistem Iterasi pertama

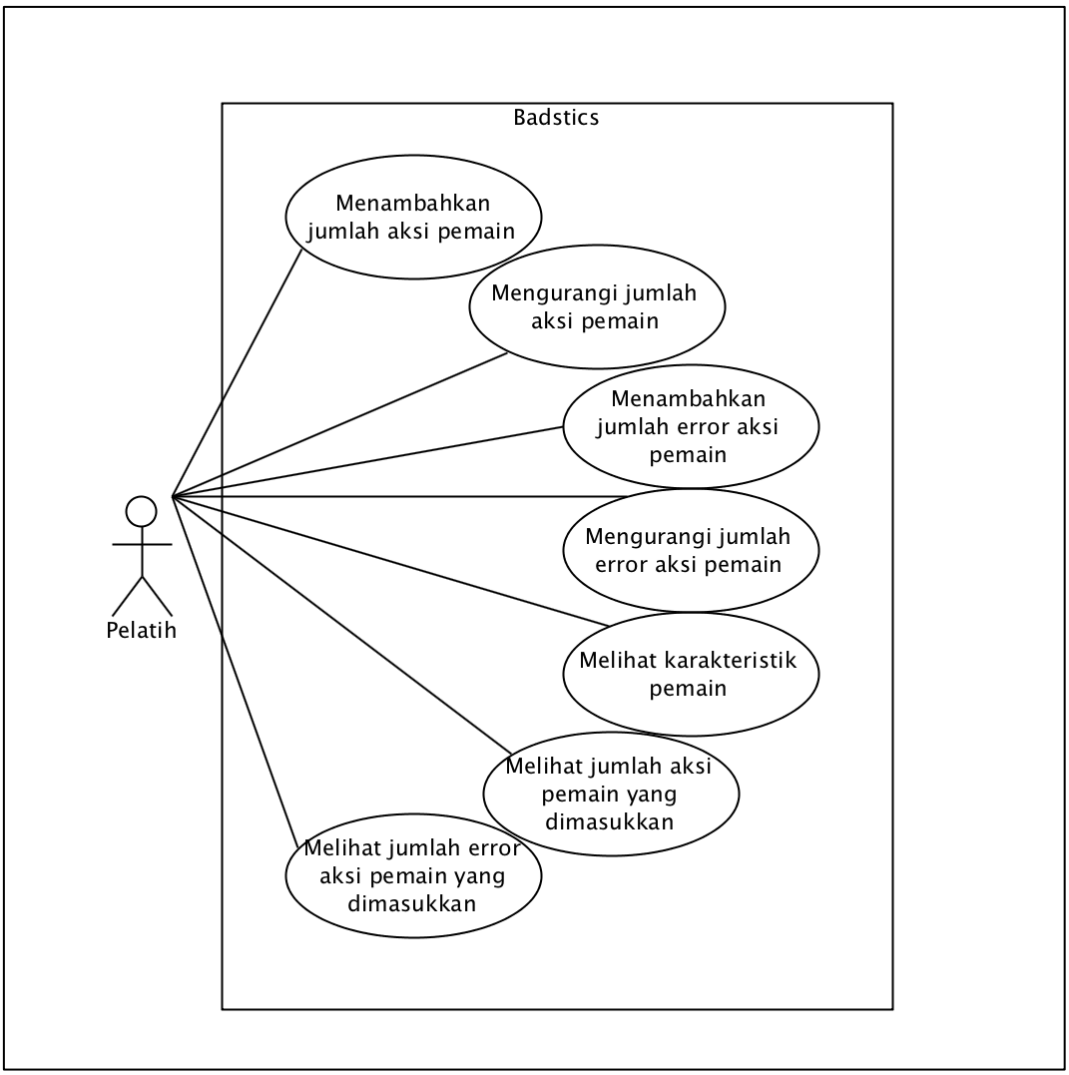
No.	Kode Kebutuhan Sistem	Deskripsi Kabutuhan
1	SRS-BDS-F-1	Sistem dapat menerima masukan untuk menambahkan aksi pemain bulutangkis dari pelatih
2	SRS-BDS-F-2	Sistem dapat menerima masukan untuk mengurangi aksi pemain bulutangkis dari pelatih
3	SRS-BDS-F-3	Sistem dapat menerima masukan untuk menambahkan error aksi pemain bulutangkis dari pelatih
4	SRS-BDS-F-4	Sistem dapat menerima masukan untuk mengurangi error aksi pemain bulutangkis dari pelatih
5	SRS-BDS-F-5	Sistem dapat memberikan keluaran informasi karakteristik pemain berdasarkan masukan dari pelatih
6	SRS-BDS-F-6	Sistem dapat memberikan keluaran berupa presentase aksi pemain di lapangan berdasarkan masukan dari pelatih
7	SRS-BDS-F-7	Sistem dapat memberikan keluaran berupa presentasi kesalahan pemain yang dilakukan berdasarkan masukan dari pelatih

Tabel 4.3 Kebutuhan Non Fungsional Sistem

No.	Kode Kebutuhan Sistem	Deskripsi Kabutuhan
1	SRS-BDS-NF-1	*Usability* yaitu kemudahan dalam menggunakan aplikasi yang digunakan oleh pelatih

4.3.1 *Diagram* Use Case

Diagram *use case* berisi sejumlah aksi yang dilakukan oleh aktor kepada sistem untuk mencapai tujuan tertentu. Aktor dalam diagram *use case* bisa berupa manusia atau sistem eksternal yang berinteraksi dengan sistem yang dimaksud. Diagram *use case* ditunjukkan dalam Gambar 4.1.



Gambar 4.1 Diagram *Use Case* Sistem

4.4 Analisis Data

Analisis data betujuan untuk mendapatkan struktur data penyimpanan yang akan diimplemntasikan pada aplikasi Badstics, data yang akan digunakan dalam aplikasi Badstics iniberdasarkan analisis sebagai berikut:

a. Data player yang merupakan representasi data dari pemain yang memiliki beberapa *attribute age, characteristic, height, name, weight*.

b. Data match yaitu data yang menyimpan semua data pertandingan yaitu *match_time, match_date dana Game log.*

c. Log merupakan data dari semua *history* dari permainan yaitu berupa Model Log.

d. Action merupakan data yang menyimpan semua aksi pemain yang ada yaitu sebanyak 28 aksi pemain di lapangan.

4.5 Skenario *Use Case*

Skenario *use case* merupakan penjabaran lebih detail dari tiap *use case* yang digambarkan pada diagram *use case.* Pada skenario *use case* terdapat beberapa bagian, diantaranya nama *use case,* kode kebutuhan, aktor yang berinteraksi, tujuan *use case, pre condition, main flow, alternative flow* & *post condition.* Pada bagian *pre condition,* berisi berbagai kondisi yang harus dipenuhi oleh sistem sebelum menjalankan *main flow* pada *use case.* Sedangkan *main flow* sendiri berisi alur-alur pengerjaan *use case* hingga selesai mendapatkan hasil yang diharapkan pada *post condition*, *alternative flow* sendiri berisi alur pengerjaan ketika dalam pengerjaan *main flow* terjadi skenario yang tidak dideklarasikan pada *main flow,* sedangkan *post condition* berisi kondisi akhir setelah aktor menjalankan *main flow.* Diperoleh 5 *use case scenario* yaitu Menambahkan jumlah aksi pemain, Mengurangi aksi pemain, Melihat Karakteristik pemain, Melihat presentase aksi pemain yang dimasukkan, melihat presentase kesalahan pemain yang dimasukkan. Skenario *use case* akan ditunjukkan oleh Tabel 4.3 hingga Tabel 4.7.

Tabel 4.4 Skenario *Use Case* Menambahkan Jumlah Aksi Pemain Iterasi Pertama

Nama *Use Case*	Menambahkan jumlah aksi pemain
Kode Kebutuhan Terkait	SRS-BDS-F-1
Aktor	Pelatih
Tujuan	Menambahkan jumlah aksi pemain
Pre Condition	Aktor telah berada dalam tab *input statistics*
Main flow	1. Pelatih menekan salah satu tombol aksi 2. Sistem menerima masukan lalu sistem menambahkan 1 nilai pada jumlah aksi berdasarkan tombol yang ditekan oleh pelatih
Post Condition	Sistem akan menambahkan 1 nilai pada data aksi sesuai tombol yang ditekan oleh pelatih
Alternative Flow	-

Tabel 4.5 Skenario *Use Case* Mengurangi Jumlah Aksi Pemain Iterasi Pertama

Nama *Use Case*	Mengurangi jumlah aksi pemain
Kode Kebutuhan Terkait	SRS-BDS-F-2
Aktor	Pelatih
Tujuan	Mengurangi jumlah aksi pemain
Pre Condition	Pelatih telah berada dalam *tab input statistics*
Main flow	1. Pelatih menekan dan menahan selama 2 detik salah satu tombol aksi 2. Sistem mengurangi 1 nilai pada jumlah aksi yang ditekan oleh pelatih
Post Condition	Sistem akan mengurangi 1 nilai pada data aksi sesuai tombol yang ditekan oleh pelatih
Alternative Flow	-

Dalam iterasi yang kedua diperoleh bahwa terdapat perubahan *main flow* dalam pengurangan jumlah aksi pemain.

Tabel 4.6 Skenario *Use Case* Mengurangi Jumlah Aksi Pemain Iterasi Kedua

Nama *Use Case*	Mengurangi jumlah aksi pemain
Kode Kebutuhan Terkait	SRS-BDS-F-2
Aktor	Pelatih
Tujuan	Mengurangi jumlah aksi pemain
Pre Condition	Pelatih telah berada dalam *tab action charts*
Main flow	1. Pelatih menekan salah satu aksi pemain dalam tab statistik dengan menggunakan satu jari 2. Pelatih memilih button *decrease* untuk mengurangi jumlah aksi pemain 3. Sistem mengurangi 1 nilai pada jumlah aksi yang ditekan oleh pelatih
Post Condition	Sistem akan mengurangi 1 nilai pada data aksi sesuai tombol yang ditekan oleh pelatih
Alternative Flow	-

Tabel 4.7 Skenario *Use Case* Menambahkan Jumlah Error Aksi Pemain

Nama *Use Case*	Mengurangi jumlah aksi pemain
Kode Kebutuhan Terkait	SRS-BDS-F-3
Aktor	Pelatih
Tujuan	Menambahkan jumlah error aksi pemain
Pre Condition	Pelatih telah berada dalam *tab action charts*
Main flow	1. Pelatih menekan salah satu error aksi pemain dalam tab statistik dengan menggunakan satu jari 2. Pelatih memilih button *increase* untuk mengurangi jumlah aksi pemain 3. Sistem menambahkan 1 nilai pada jumlah aksi yang ditekan oleh pelatih
Post Condition	Sistem akan menambahkan 1 nilai pada data aksi sesuai tombol yang ditekan oleh pelatih
Alternative Flow	-

Tabel 4.8 Skenario *Use Case* Mengurangi Jumlah Error Aksi Pemain

Nama *Use Case*	Mengurangi jumlah aksi pemain
Kode Kebutuhan Terkait	SRS-BDS-F-3
Aktor	Pelatih
Tujuan	Mengurangi jumlah error aksi pemain
Pre Condition	Pelatih telah berada dalam *tab action charts*
Main flow	1. Pelatih menekan salah satu error aksi pemain dalam tab statistik dengan menggunakan satu jari 2. Pelatih memilih button *decrease* untuk mengurangi jumlah aksi pemain 3. Sistem mengurangi 1 nilai pada jumlah aksi yang ditekan oleh pelatih
Post Condition	Sistem akan mengurangi 1 nilai pada data aksi sesuai tombol yang ditekan oleh pelatih
Alternative Flow	-

Tabel 4.9 Skenario *Use Case* Melihat Karakteristik Pemain Iterasi Pertama

Nama *Use Case*	Melihat karakteristik pemain
Kode Kebutuhan Terkait	SRS-BDS-F-5
Aktor	Pelatih
Tujuan	Melihat karakteristik pemain
Pre Condition	Aktor telah berada dalam aplikasi
Main flow	1. Pelatih menekan tombol profil pada *tab bar* 2. Sistem akan menentukan karakteristik pemain dengan menghitung total aksi yang dimasukkan oleh pelatih dengan menggunakan metode naive bayes 3. Sistem menampilkan karakteristik pemain
Post Condition	Sistem akan menampilkan informasi karakteristik pemain pada layar
Alternative Flow	-

Dalam iterasi yang kedua diperoleh bahwa terdapat perubahan *main flow* dalam melihat karakteristik pemain.

Tabel 4.10 Skenario *Use Case* Melihat Karakteristik Pemain Iterasi Kedua

Nama *Use Case*	Melihat karakteristik pemain
Kode Kebutuhan Terkait	SRS-BDS-F-5
Aktor	Pelatih
Tujuan	Melihat karakteristik pemain
Pre Condition	Aktor telah berada dalam aplikasi
Main flow	1. Pelatih menekan tombol *input Statistic* pada *tab bar* 2. Sistem akan menentukan karakteristik pemain dengan menghitung total aksi yang dimasukkan oleh pelatih dengan menggunakan metode naive bayes 3. Sistem menampilkan karakteristik pemain

Post Condition	Sistem akan menampilkan informasi karakteristik pemain pada layar
Alternative Flow	-

Tabel 4.11 Skenario *Use Case* Melihat Presentase Aksi Pemain

Nama *Use Case*	Melihat Presentase Aksi Pemain
Kode Kebutuhan Terkait	SRS-BDS-F-6
Aktor	Pelatih
Tujuan	Melihat presentase dari aksi pemain
Pre Condition	Aktor telah berada dalam aplikasi
Main flow	1. Pelatih menekan tombol *action chart* pada *tab bar* 2. Sistem menampilkan statistik dari aksi pemain
Post Condition	Sistem akan menampilkan informasi karakteristik pemain berupa presentasi pada layar
Alternative Flow	-

Tabel 4.12 Skenario *Use Case* Melihat Presentase Kesalahan Pemain

Nama *Use Case*	Melihat Presentase Kesalahan Pemain
Kode Kebutuhan Terkait	SRS-BDS-F-7
Aktor	Pelatih
Tujuan	Melihat presentase dari aksi pemain
Pre Condition	Aktor telah berada dalam *tab statistic charts*
Main flow	1. Pelatih menekan tombol *action chart* pada *tab bar* 2. Sistem menampilkan statistik dari kesalahan pemain
Post Condition	Sistem akan menampilkan informasi karakteristik pemain berupa presentasi pada layar
Alternative Flow	-

BAB 5
PERANCANGAN DAN IMPLEMENTASI

Sebelum mengimplementasikan sistem, dilakukan perancangan sistem yang digunakan sebagai dasar untuk mengimplementasikan sistem, dalam tahap ini perancangan yang akan dilakukan adalah perancangan arsitektur sistem, perancangan komponen, perancangan data dan perancangan antar muka.

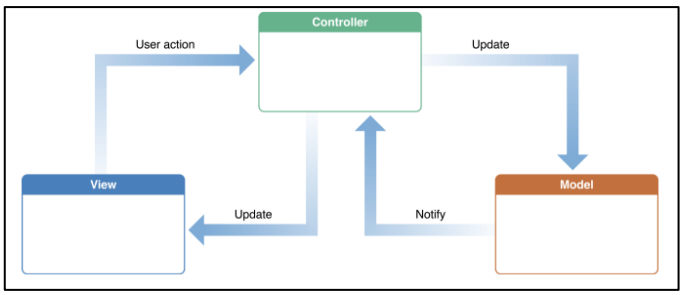
5.1 Perancangan

Setelah proses analisis kebutuhan selesai dilakukan, tahap berikutnya adalah perancangan. Perancangan dilakukan berdasarkan hasil dari analisis kebutuhan yang telah dilakukan. Proses perancangan Badstics ini dibagi menjadi empat tahap yaitu perancangan arsitektur, perancangan komponen, perancangan Data, dan perancangan antarmuka.

5.1.1 Perancangan Arsitektur Sistem

Aplikasi Badstics ini dibangun dengan menggunakan arsitektur MVC atau *Model View Controller* dimana *Model* merupakan klas yang berisi data dari aplikasi, *View* merupakan klas yang dibangun untuk sebagai antarmuka dari aplikasi dan *Controller* sebagai klas yang mengatur proses dari klas *View*. Dalam aplikasi ini digunakan *Design Pattern Singleton* dimana semua data terpusat pada satu *instance class model* sehingga dapat diakses dari semua klas dengan nilai yang sama.

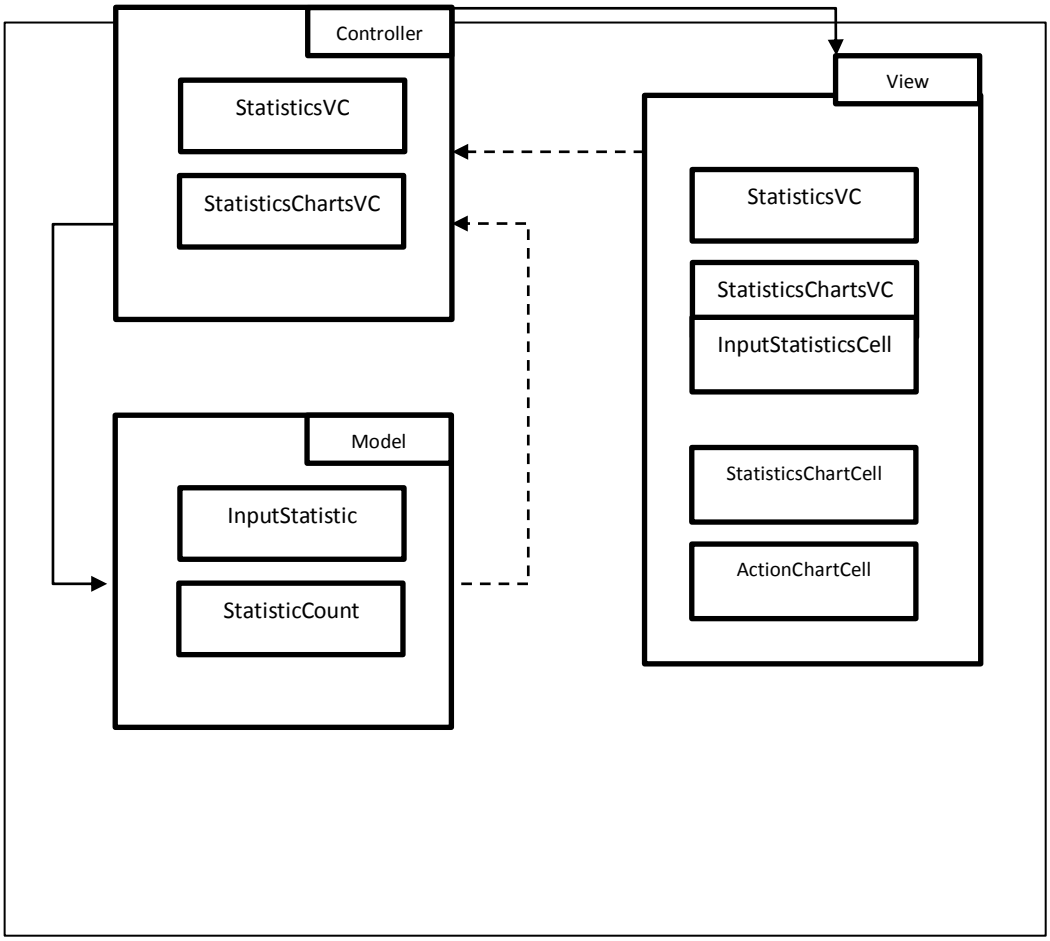
Pada dasarnya arsitektur aplikasi pada iOS secara *default* menggunakanan arsitektur MVC (*Model-View-Controller).* Model seperti ini Arsitektur ini memisahkan *model* yang bertanggungjawab sebagai *layer* untuk menangani data dan manipulasi data, *view* yaitu *layer* yang bertanggungjawab untuk mengatur tampilan aplikasi atau antar muka pengguna (UI), dan *controller* yaitu *layer yang bertanggungjawab* untuk menghubungkan antar layer *view* dan layer *model,* layer ini berguna untuk mengubah model dan menginterpretasikan input atau aksi dari pengguna dari *view layer*.



Gambar 5.1 Diagram Arsitektur MVC

Sumber: Apple, 2017

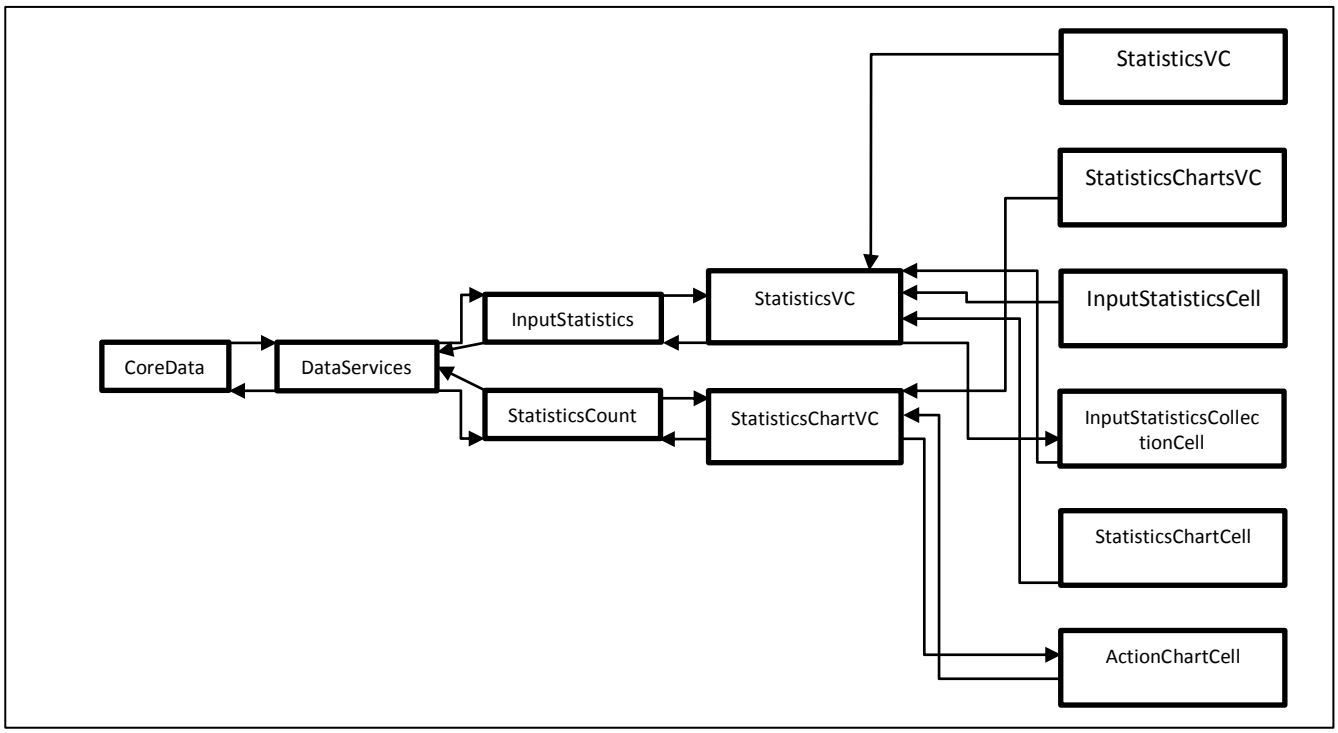
Setelah proses analisis kebutuhan selesai dilakukan, tahap berikutnya adalah perancangan. Perancangan dilakukan berdasarkan hasil dari analisis kebutuhan yang telah dilakukan, dalam perancangannya badstics dirancang sesuai dengan arsitektur iOS seperti pada gambar 5.1. Proses perancangan Badstics ini dibagi menjadi tiga tahap yaitu perancangan arsitektur, perancangan komponen,



dan perancangan antarmuka.

Gambar 5.2 Diagram Arsitektur MVC

Pada Gambar 5.2 Klas InputStatistic dan klas `StatisticCount` bekerja sebagai klas *model* yang dibantu dengan beberapa *entity class* yang mewakili beberapa table dalam database, klas `StatisticsVC` dan klas `SatatisticsChartVC` bekerja sebagai *controller* dimana *controller* berguna untuk menghubungkan antar model dan klas-klas antar muka atau klas *View* yang ada.

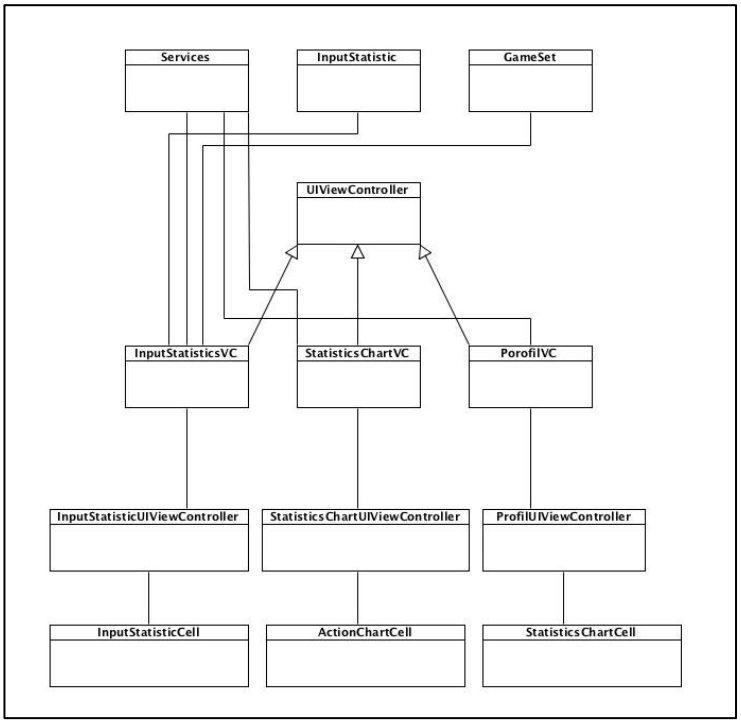


Gambar 5.3 Diagram blok sistem perangkat lunak Badstics

Pada Gambar 5.3 dijelaskan alur komunikasi tiap bagian perangkat lunak. Komunikasi antar *view* yaitu `StatisticsVC`, `StatisticsChartsVC`, `InputStatisticsCell`, `InputStatisticsCollectionCell`, `StatisticsChartsCell` dan `ActionChartsCell` berhubungan dengan klas *controller* yaitu klas `StatisticsVC` dan `StatisticsChartsVC`, klas *controller* menghubungi kelas *model* yang akan menghubungi kelas Dataservices setellah itu semua fungsi yang ada di DataServices yang akan menghubungi CoreData pada aplikasi, Core Data merupakan *database system* yang ada pada aplikasi iOS.

5.1.2 Class Diagram

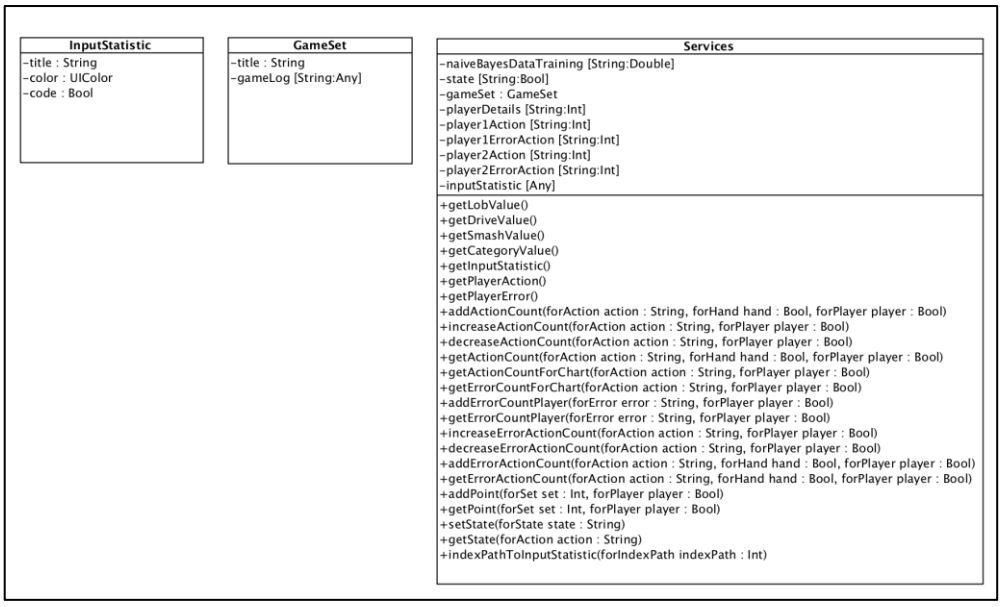
Pada perancangan arsitektur *class* diagram dibagi menjadi dua bagian, dimana yang pertama adalah *class* diagram pada *class* controller dan *class* diagram pada *class model. Class diagram Controller* dijelaskan pada gambar 5.1 dan *class diagram Model* dijelaskan pada gambar 5.2.



Gambar 5.4 ***Class Diagram Aplikasi Badstics***

Sesuai dengan Gambar 5.4 class diagram diatas memiliki kelas induk yaitu UIViewController, yang mempunyai tiga kelas turunan yaitu `InputStatisticVC` dan `StatisticsChartVC`. Ketiga kelas turunan dari kelas UIViewController memiliki hubungan

5.1.2.1 Informasi klas Model

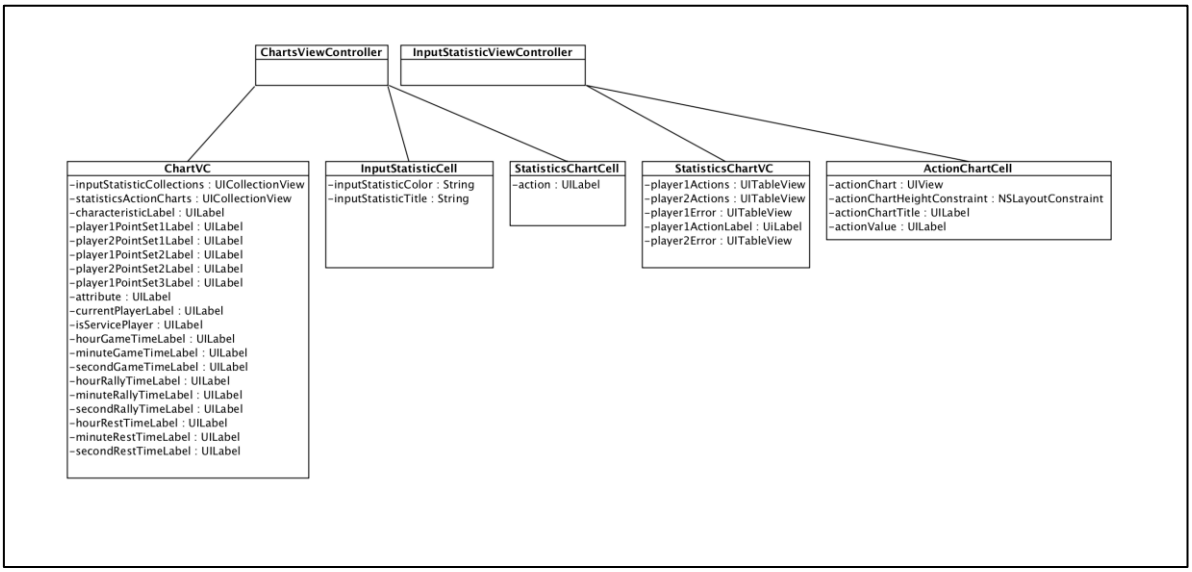


Gambar 5.5 ***Class Diagram Model***

Pada Gambar 5.2 terdapat tiga klas model yang digunakan dalam aplikasi Badstics ini, klas yang pertama yaitu klas Services yaitu klas yang

digunakan untuk menyimpan semua data yang diperlukan dalam aplikasi, selanjutnya yaitu klas InputStatistic yaitu klas model yang menyimpan jenis aksi dari pemain berupa nama aksi, kode aksi dan status aksi dan yang terakhir yaitu GameSet yang menyimpan semua log data dari pertandingan yang dilakukan oleh pemain.

5.1.2.2 Informasi klas View

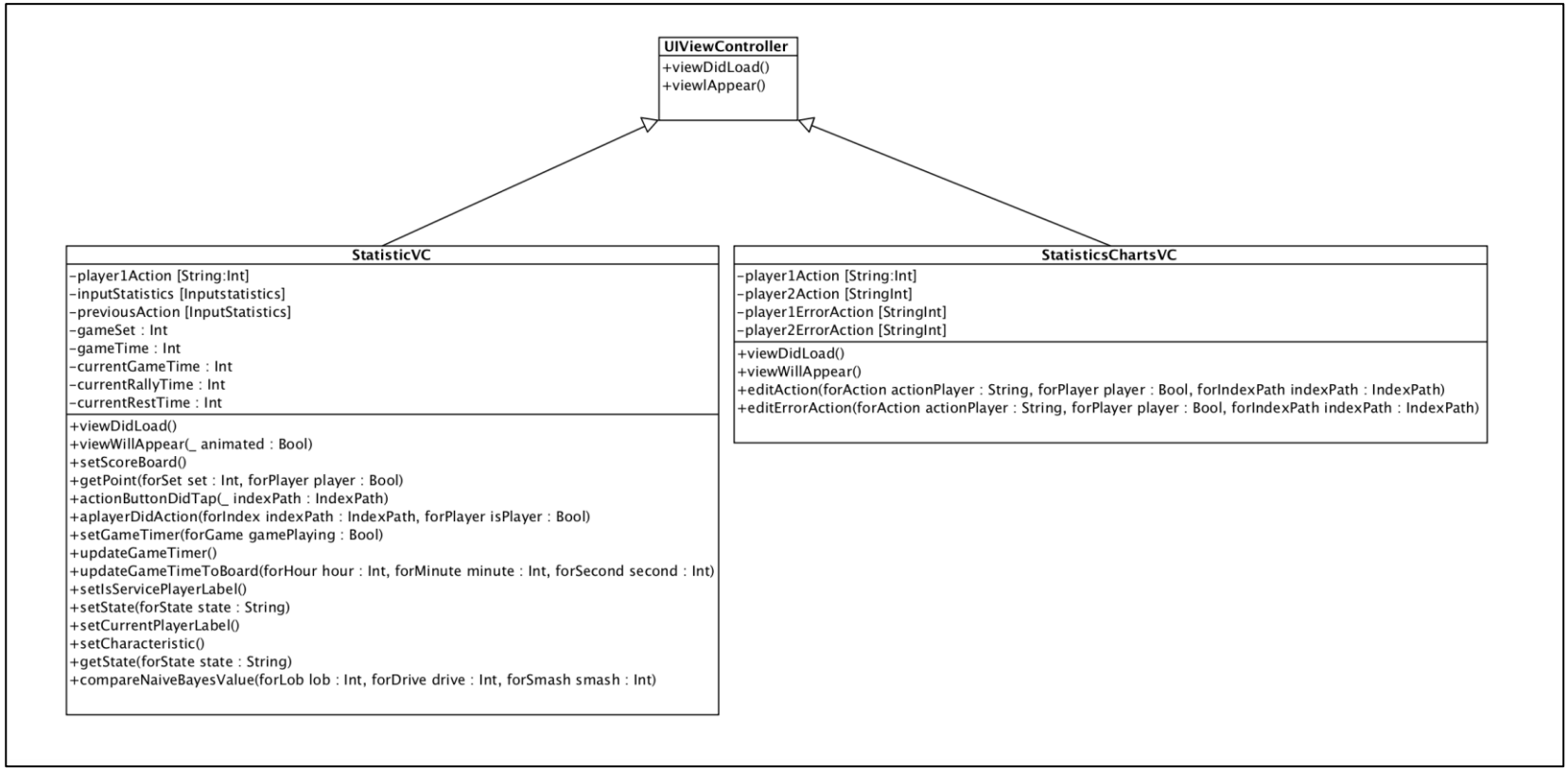


Gambar 5.6 *Class Diagram View*

Pada gambar 5.6 terdapat enam klas view yang digunakan dalam aplikasi Badstics dimana terdapat tigas klas utama yaitu InputStatisticUIViewController, StatisticChartsUIViewController dan ProfilUIViewController dimana ketiga kelas tersebut memiliki klas atribut klas lain yaitu InputStatisticCell yang terdapat sebagai attribute klas InputStatisticUIViewController, ActionChartCell yang terdapat sebagai attribute klas StatisticChartsUIViewController dan StatisticsChartCell yang terdapat sebagai attribute klas ProfilUIViewController.

5.1.2.3 Informasi klas Controller

Gambar class controller



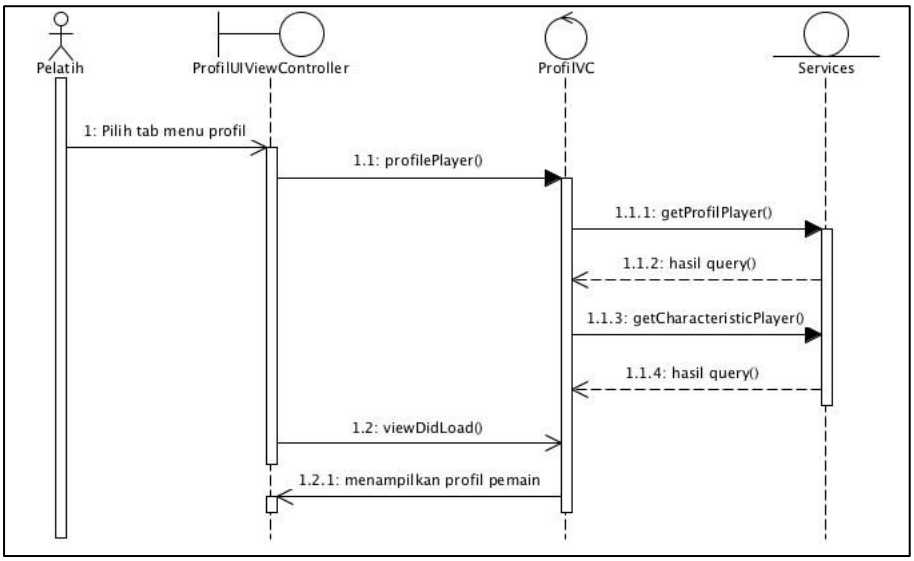
Gambar 5.7 ***Class Diagram Controller***

Sesuai dengan Gambar 5.7 pada klas Controller terdapat tiga klas utama yaitu `InputStatisticVC` dan `StatisticsChartVC` dimana tigas klas tersebut merupakan turunan dari klas UIViewController yang digunakan untuk Controller dari klas view yaitu `InputStatisticVC` untuk klas view `InputStatisticUIViewController`, `StatisticsChartVC` untuk klas view `StatisticChartsUIViewController`.

5.1.3 *Sequence* Diagram

Pada pemodelan *sequence diagram* dijelaskan bahwa urutan proses yang terjadi untuk mencapai suatu kebutuhan sistem. Semua objek pada *sequence diagram* merupakan hasil identifikasi dari spesifikasi kebutuhan dan *use case scenario* yang ada pada tahap analisis kebutuhan sebelumnya. Setiap *sequence diagram* menggambarkan setiap *use case* yang ada.

5.1.3.1 Sequence Diagram Melihat Profil Iterasi Pertama



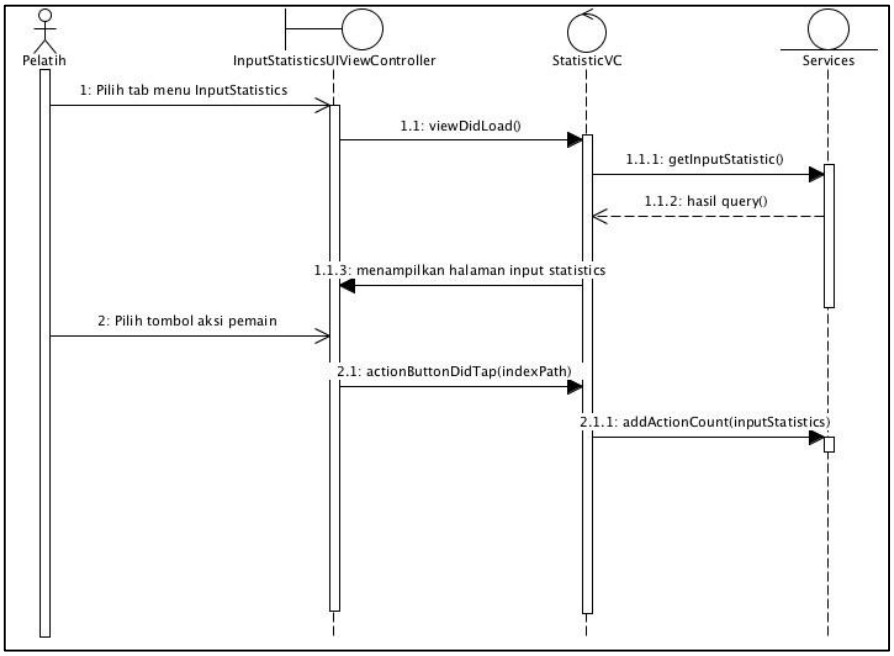
Gambar 5.8 *Sequence Diagram* Melihat Profil Iterasi Pertama

Dalam Gambar 5.8 dalam *Sequence diagram* melihat profil , pelatih cukup memilih tab bar menu profil. Setelah pelatih memilih tab bar menu profil maka sistem akan memanggil fungsi viewDidLoad() yang akan memanggil ProfilPlayer() pada *controller* ProfilVC. Kemudian meminta data pribadinya menggunakan fungsi getProfilPlayer() pada *Entity DataServices* yang hasilnya dikembalikan langsung ke *controller*. Kemudian *controller* memanggil *boundary* ProfilUIViewController untuk menampilkan data-data yang telah diminta *controller* di halaman profil.

5.1.3.2 Sequence Diagram Melihat Profil Iterasi Kedua

Dalam iterasi kedua Melihat profil telah dihapus dan digabungkan dengan *tab menu* Input Statistics sehingga sequence diagram melihat profil pun dihapus

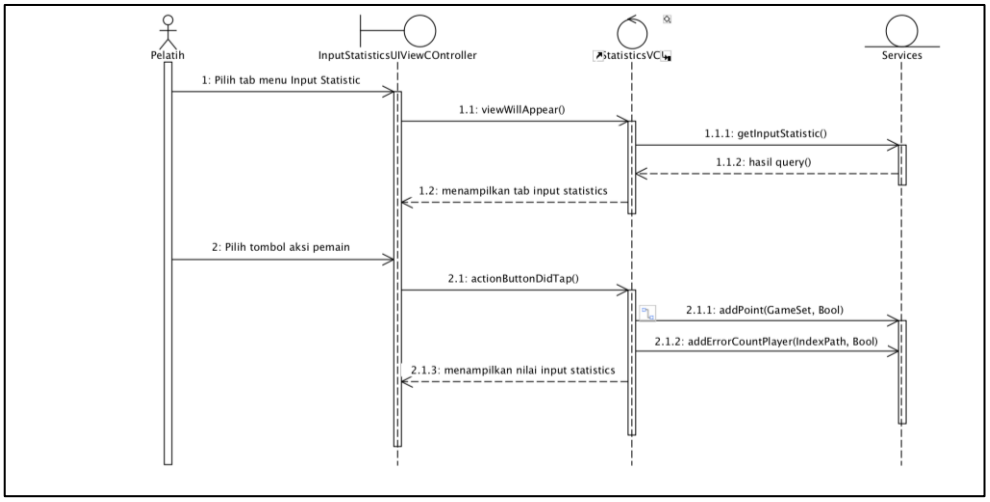
5.1.3.3 Sequence Diagram Input Statistics Iterasi Pertama



Gambar 5.9 *Sequence Diagram Input Statistics* Iterasi Pertama

Dalam Gambar 5.9 *Sequence diagram* melihat input statistics iterasi pertama, pelatih cukup memilih tab bar menu input statistics. Setelah pelatih memilih tab bar menu profil maka sistem akan memanggil fungsi viewDidLoad() yang akan memanggil getInputStatistics() dan add pada *controller* StatisticVC. Kemudian meminta data *input statistics* dengan memanggil fungsi getInputStatistics() pada *Entity Services* yang hasilnya dikembalikan langsung ke *controller*. Kemudian *controller* memanggil *boundary* InputStatisticsUIViewController untuk menampilkan data-data yang telah diminta *controller* di halaman InputStatistics.

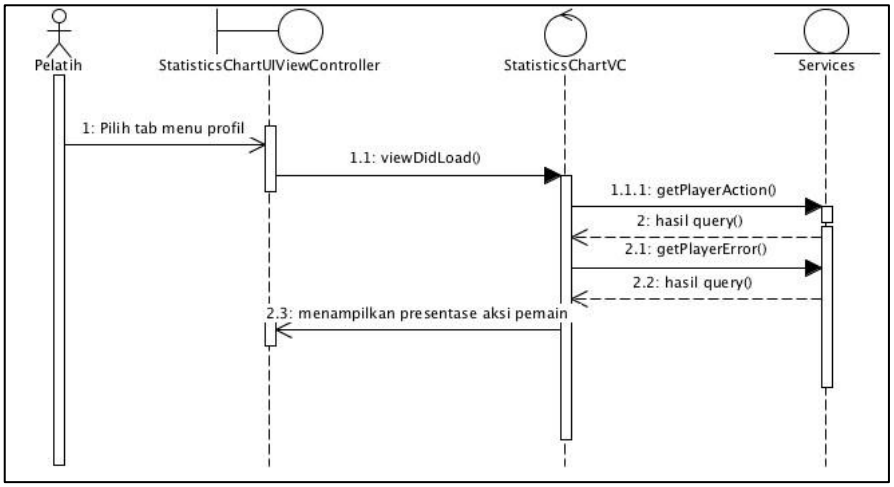
5.1.3.4 Sequence Diagram Input Statistics Iterasi Kedua



Gambar 5.10 *Sequence Diagram Input Statistics* Iterasi Kedua

Dalam Gambar 5.10 *Sequence diagram* input statistics itersi kedua, pelatih cukup memilih tab bar menu input statistics. Setelah pelatih memilih tab bar menu profil maka sistem akan memanggil fungsi viewDidLoad() yang akan memanggil getInputStatistics() pada *Entity Services* yang hasilnya dikembalikan langsung ke *controller*, Kemudian *controller* memanggil *boundary* InputStatisticsUIViewController untuk menampilkan data-data yang telah diminta *controller* di halaman InputStatistics yaitu input statistics dan profil pemain. Pada saat pelatih menekan tombol aksi pemain acctionbuttonDidTap() pada *controller* StatisticVC akan dipanggil kemudian meminta data *input statistics* dengan memanggil fungsi getInputStatistics() pada *Entity Services* kemudian *controller* memanggil *boundary* InputStatisticsUIViewController untuk menampilkan data-data yang telah ter-*update* pada tab input statistics.

5.1.3.5 *Sequence Diagram* Chart



Gambar 5.11 *Sequence Diagram Statistic Charts*

Dalam Gambar 5.11 *Sequence Diagram Statistics charts* , pelatih cukup memilih tab bar menu Statistics Charts. Setelah pelatih memilih tab bar menu Statistics Charts maka sistem akan memanggil fungsi viewDidLoad() getPlayerAction() yang akan memanggil getPlayerAction() dan pada *controller* StatisticsChartsVC. Kemudian meminta data jumlag aksi pemain dengan memanggil fungsi getPlayerAction() dan getPlayerError pada *Entity DataServices* yang hasilnya dikembalikan langsung ke *controller*. Kemudian *controller* memanggil *boundary* StatisticsUIViewController untuk menampilkan data-data yang telah diminta *controller* di halaman InputStatistics.

5.1.4 Perancangan Komponen

Perancangan komponen menggambarkan rincian sub-sistem dari setiap komponen perangkat lunak. Untuk mencapai hal ini, perancangan komponen harus mendefinisikan struktur data untuk semua objek data lokal dan rincian algoritma untuk proses yang terjadi di dalam komponen. Dalam perancangan komponen ini dipilih 1 Algoritma *method* utama dari setiap *class* pada *controller*.

5.1.4.1 Perancangan Komponen Klas StatisticVC

Nama Operasi : actionButtonDidTap(IndexPath)

Algoritma :

```
if state == true then
   memanggil fungsi playerDidAction(IndexPath, Bool)
else
   memanggil fungsi playerDidAction(IndexPath, Bool)
reload statisticActionCharts
set current player label
set is service player
set scoreboard
set characteristic player
```

Nama Operasi : playerDidAction (IndexPath, Bool)

Algoritma :

```
if title == "SERVICE PENDEK" or title == "SERVICE PANJANG" then
   if isServiceState != true then
      return
   endif
   memulai waktu permainan
   merubah state untuk "isService"
   if title == "FORCED ERROR" or title == "UNFORCED ERROR" then
   if isServicestate != true{
```

```
      merubah state untuk "isService"
   endif
   memulai waktu permainan
   menambahkan point untuk pemain lawan
   menambahkan error cause untuk pemain
   if title == "FORCED ERROR" then
      menambahkan error cause untuk pemain
      menambahkan jumlah error untuk pemain
   endif
else
   if isServiceState == true then
      return
   endif
   menambahkan jumlah aksi ke pemain
endif
merubah state isPlayerOne
previousAction = inputStatistics
```

Nama Operasi : compareNaiveBayesValue (Int, Int, Int)

Algoritma :

```
Inisialisasi priorOffensive = prior Offensive
Inisialisasi priorDefensive = prior Defensive
Inisialisasi    priorOffensiveValue    =    likelihoodLob    *
likelihoodDrive * likelihoodSmash
Inisialisasi    priorDefensiveValue    =    likelihoodLob    *
likelihoodDrive * likelihoodSmash
if offensiveValue > defensiveValue then
   return true
else
return false
endif
```

Penjelasan pseudocode dengan menggunakan baris

Nama Operasi : getCategoryValue (Int)

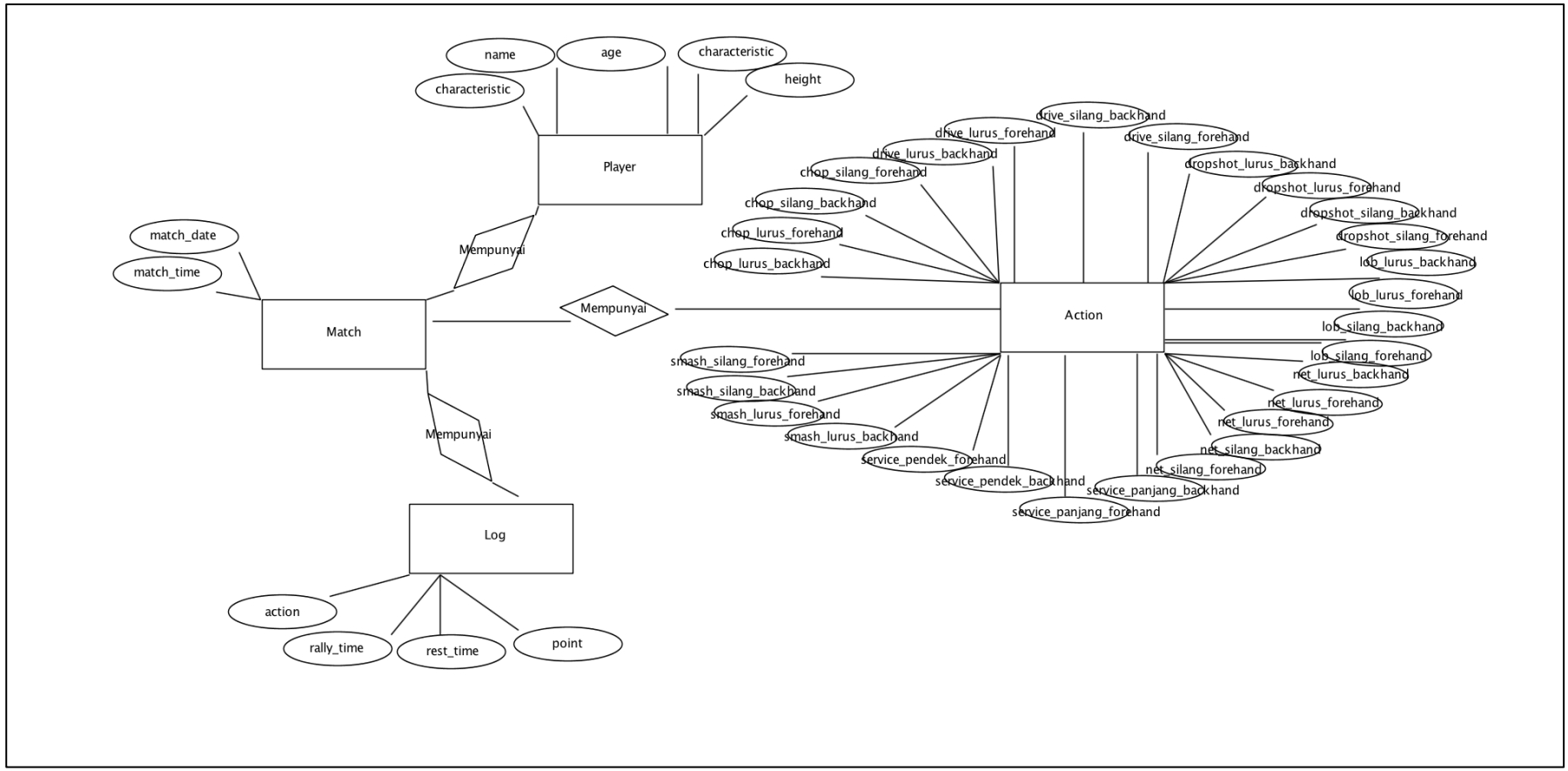
Algoritma :

```
inisialisasi category = 0
```

```
category = value < 10 ? 0 : value < 22 ? 1 : value < 34 ? 2 : 3
return category
```

5.1.5 Perancangan Data

Pada perancangan data dari aplikasi Badstics ini akan dijelaskan secara rinci mengenai pembuatan *Entity* Relational Diagram (ERD) dan Physical Data Model(PDM).



Gambar 5.12 Perancangan ERD

Setelah dilakukan pembuatan ERD, langkah berikutnya adalah membuat rancangan physical data model(PDM). Jumlah tabel yang dibuat adalah sama dengan jumlah entitas yang telah didefinisikan. Perancangan PDM akan dijelaskan pada Gambar 5.12

1. Tabel Player

Nama tabel : player

Jumlah *field* : 4

Fungsi : Untuk menyimpan biodata atlete

Tabel 5.1 Struktur tabel player

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Age	Int32	-	Umur atlete
2	Characteristics	String	-	Karakteristik atlete
3	Name	String	-	Nama atlete
4	Height	Int32	-	Tinggi atlete
5	Weight	Int32	-	Berat atlete

2. Tabel Match

Nama tabel : match

Jumlah *field* : 2

Fungsi : menyimpan detil permainan

Tabel 5.2 Struktur tabel match

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Match_time	Int32	-	Waktu pertandingan
2	Match_date	Int32	-	Tanggal pertandingan

3. Tabel Log

Nama tabel : log

Jumlah *field* : 4

Fungsi : menyimpah *history* permainan tiap poin

Tabel 5.3 Struktur tabel log

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Action	String	-	Aksi pemain
2	Rally_time	Int32	-	Waktu aktif pertandingan
3	Rest_time	Int32	-	Waktu istirahat pertandingan
4	Poin	Int32	-	Poin pemain

4. Tabel Action

Nama tabel : action

Jumlah *field* : 28

Fungsi : menyimpan aksi pemain

Tabel 5.4 Struktur tabel action

No.	Nama *field*	Tipe	Lebar	Keterangan
1	Service_Pendek_Forehand	Int32	-	Jumlah aksi
2	Service_Panjang_Forehand	Int32	-	Jumlah aksi
3	Smash_Lurus_Forehand	Int32	-	Jumlah aksi
4	Smash_Silang_Forehand	Int32	-	Jumlah aksi
5	Drive_Lurus_Forehand	Int32	-	Jumlah aksi

6	Drive_Silang_Forehand	Int32	-	Jumlah aksi
7	Dropshot_Lurus_Forehand	Int32	-	Jumlah aksi
8	Dropshot_Silang_Forehand	Int32	-	Jumlah aksi
9	Net_Lurus_Forehand	Int32	-	Jumlah aksi
10	Net_Silang_Forehand	Int32	-	Jumlah aksi
11	Lob_Lurus_Forehand	Int32	-	Jumlah aksi
12	Lob_Silang_Forehand	Int32	-	Jumlah aksi
13	Chop_Lurus_Forehand	Int32	-	Jumlah aksi
14	Chop_Silang_Forehand	Int32	-	Jumlah aksi
15	Service_Pendek_Backhand	Int32	-	Jumlah aksi
16	Service_Panjang_Backhand	Int32	-	Jumlah aksi
17	Smash_Lurus_Backhand	Int32	-	Jumlah aksi
18	Smash_Silang_Backhand	Int32	-	Jumlah aksi
19	Drive_Lurus_Backhand	Int32	-	Jumlah aksi
20	Drive_Silang_Backhand	Int32	-	Jumlah aksi
21	Dropshot_Lurus_Backhand	Int32	-	Jumlah aksi
22	Dropshot_Silang_Backhand	Int32	-	Jumlah aksi
23	Net_Lurus_Backhand	Int32	-	Jumlah aksi
24	Net_Silang_Backhand	Int32	-	Jumlah aksi
25	Lob_Lurus_Backhand	Int32	-	Jumlah aksi
26	Lob_Silang_Backhand	Int32	-	Jumlah aksi
27	Chop_Lurus_Backhand	Int32	-	Jumlah aksi
28	Chop_Silang_Backhand	Int32	-	Jumlah aksi

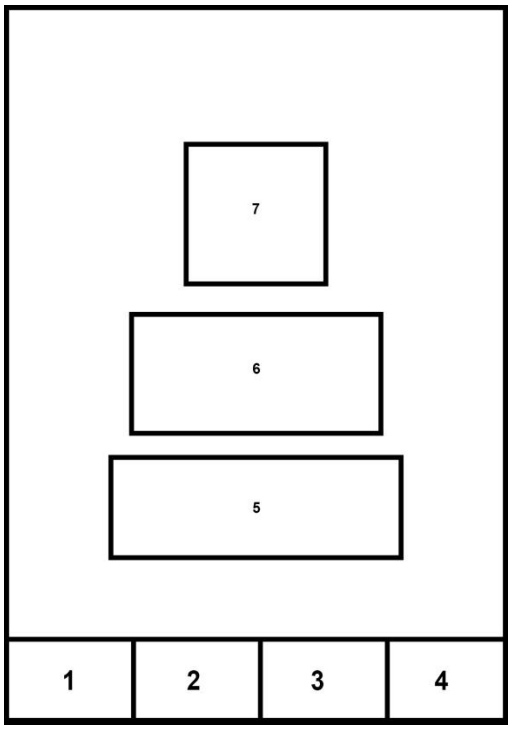
29	Foced_Error	Int32	-	Jumlah aksi
30	Unforced_Error	Int32	-	Jumlah aksi

5.1.6 Perancangan Antarmuka

Dalam perancangan antarmuka ini akan ditunjukkan beberapa sampel antarmuka seperti antarmuka *input statistics*, antarmuka *action charts* yang didaptkan dari dua kali iterasi seusai dengan kebutuhan dari pengguna yaitu pelatih yang akan menggunakan aplikasi ini nantinya.

5.1.6.1 Perancangan Antarmuka Presentase Profil Iterasi Pertama

Perancangan antarmuka profil pemain pada Gambar 5.11, dan Penjelasan Detail dari perancangan antarmuka tersebut dijelaskan pada Tabel 5.1.



Gambar 5.13 Perancangan antarmuka profil Iterasi Kedua

Tabel 5.5 Penjelasan Antarmuka Profil Iterasi Pertama

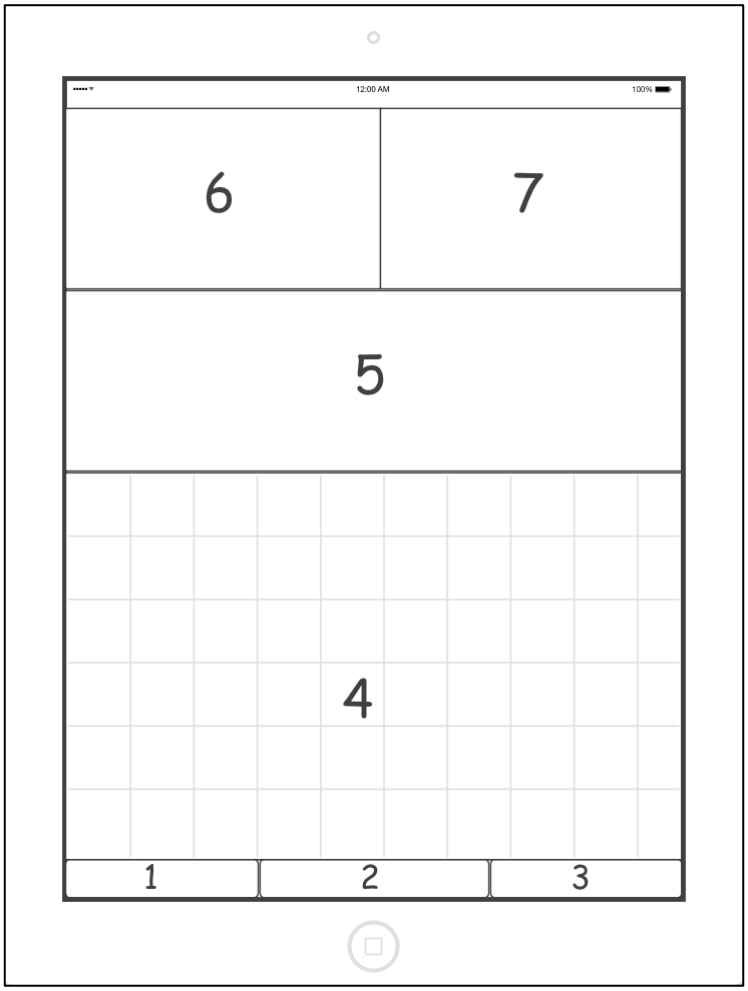
No.	Nama Objek	Tipe	Keterangan
1	Tab Bar Profil	Logo *Button*	Tab baru untuk menampilkan menu profil
2	Tab Bar *Input Statistics*	Logo *Button*	Tab baru untuk menampilkan menu *Input statistics*

3	Tab Bar Presentase Aksi Pemain	Logo *Button*	Tab baru untuk menampilkan menu Presentase aksi pemain
4	Tab Bar menu lainnya	Logo *Button*	Tab baru untuk menampilkan menu lainnya
5	Karakteristik Pemain	*Text*	Menampilkan karakteristik pemain berdasarkan masukan yang dilakukan oleh pelatih
6	Detail Pemain	*Text*	Menampilkan detail dari profil pemain
7	Foto Pemain	Gambar	Menampilkan foto pemain

Pada iterasi kedua tab menu profil tidak lagi diperlukan karena telah digabungkan dalam tab menu input statistic sehingga dapat langsung terlihat profil ketika menginputkan aksi pemain

5.1.6.2 Perancangan Antarmuka Presentase Profil Iterasi Kedua

Perancangan antarmuka profil pemain pada Gambar 5.12, dan Penjelasan Detail dari perancangan antarmuka tersebut dijelaskan pada Tabel 5.2



Gambar 5.14 Perancangan antarmuka profil Iterasi Kedua

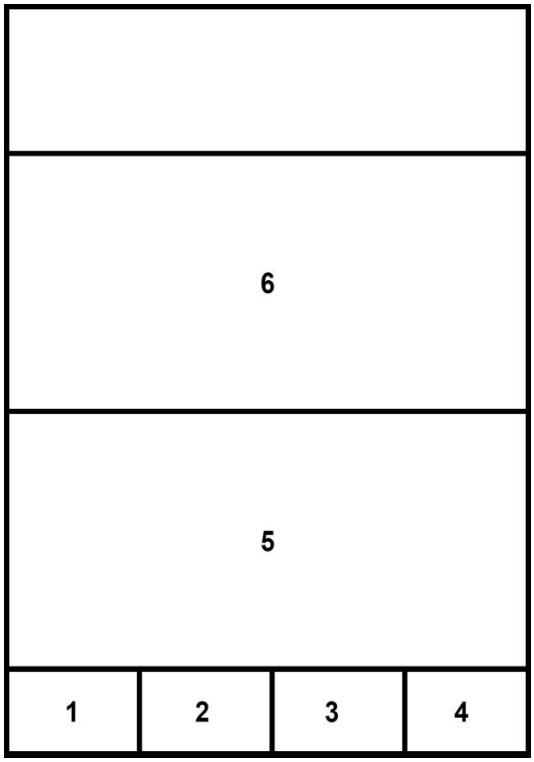
Tabel 5.6 Penjelasan Antarmuka Profil Iterasi Kedua

No.	Nama Objek	Tipe	Keterangan
1	Tab Bar *Input Statistics*	Logo *Button*	Tab baru untuk menampilkan menu *Input statistics*
2	Tab Bar Presentase Aksi Pemain	Logo *Button*	Tab baru untuk menampilkan menu Presentase aksi pemain
3	Tab Bar menu lainnya	Logo *Button*	Tab baru untuk menampilkan menu lainnya
4	Input statistics	Logo *Button*	*Button* untuk menambahkan aksi pemain ketika bermain di lapangan

5	Diagram Aksi Pemain	Gambar	Menampilkan detail dari jumlah aksi dari pemain di lapangan
6	Profil Pemain	Text	Menampilkan detail dari pemain
7	Detail Pertandingan	Text	Menampilkan detail dari pertandingan yang sedang berlangsung

5.1.6.3 Perancangan Antarmuka *Input statistics* Iterasi Pertama

Perancangan antarmuka *input statistics* pemain pada Gambar 5.13, dan Penjelasan Detail dari perancangan antarmuka tersebut dijelaskan pada Tabel 5.3.



Gambar 5.15 Perancangan antarmuka *input statistics* Iterasi Pertama

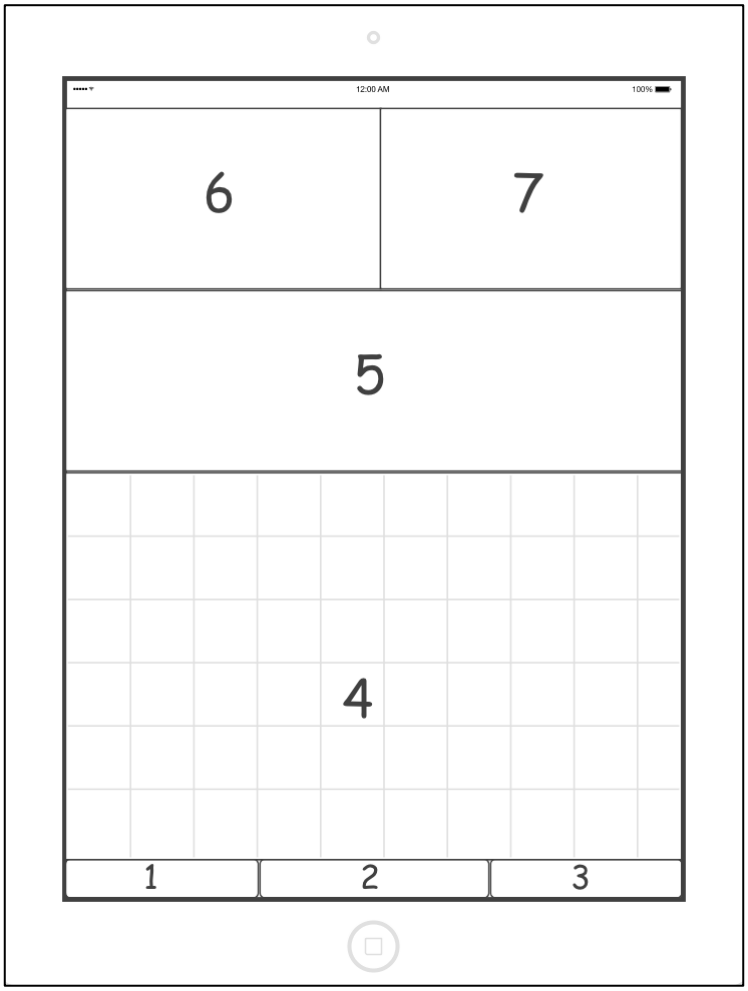
Tabel 5.7 Penjelasan Antarmuka *input statistics* Iterasi Pertama

No.	Nama Objek	Tipe	Keterangan
1	Tab Bar Profil	Logo *Button*	Tab baru untuk menampilkan menu profil
2	Tab Bar *Input Statistics*	Logo *Button*	Tab baru untuk menampilkan menu *Input statistics*

3	Tab Bar Presentase Aksi Pemain	Logo *Button*	Tab baru untuk menampilkan menu Presentase aksi pemain
4	Tab Bar menu lainnya	Logo *Button*	Tab baru untuk menampilkan menu lainnya
5	Tombol masukan aksi pemain	*Logo Button*	Menampilkan hasil perhitungan presentase dari jumlah aksi yang dilakukan pemain 2
6	Jumlah aksi pemain	*Widget*	Menampilkan hasil perhitungan presentase dari jumlah kesalahan yang dilakukan pemain 2

5.1.6.4 Perancangan Antarmuka *Input statistics* Iterasi Kedua

Perancangan antarmuka *input statistics* pemain pada Gambar 5.14, dan Penjelasan Detail dari perancangan antarmuka tersebut dijelaskan pada Tabel 5.4.



Gambar 5.16 Perancangan antarmuka *input statistics* Iterasi Kedua

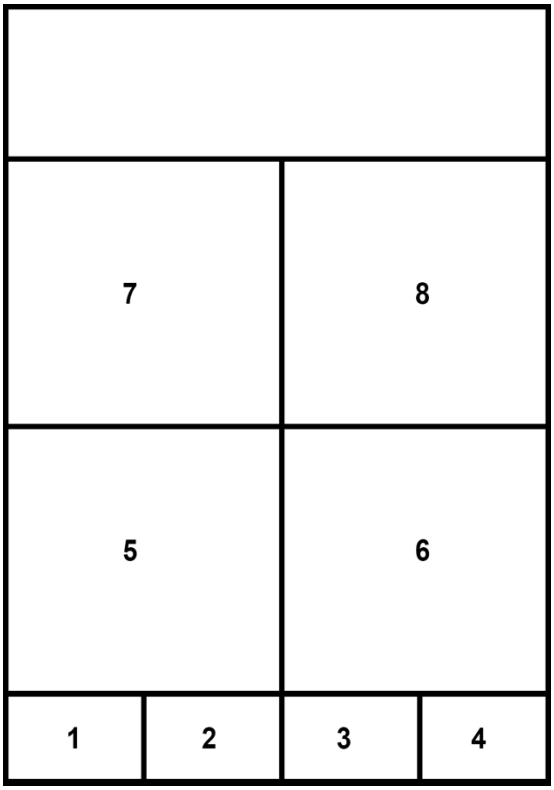
Tabel 5.8 Penjelasan Antarmuka *input statistics* Iterasi Kedua

No.	Nama Objek	Tipe	Keterangan
1	Tab Bar *Input Statistics*	Logo *Button*	Tab baru untuk menampilkan menu *Input statistics*
2	Tab Bar Presentase Aksi Pemain	Logo *Button*	Tab baru untuk menampilkan menu Presentase aksi pemain
3	Tab Bar menu lainnya	Logo *Button*	Tab baru untuk menampilkan menu lainnya
4	Input statistics	Logo *Button*	*Button* untuk menambahkan aksi pemain ketika bermain di lapangan

5	Diagram Aksi Pemain	Gambar	Menampilkan detail dari jumlah aksi dari pemain di lapangan
6	Profil Pemain	Text	Menampilkan detail dari pemain
7	Detail Pertandingan	Text	Menampilkan detail dari pertandingan yang sedang berlangsung

5.1.6.5 Perancangan Antarmuka Presentase Aksi Pemain Iterasi Pertama

Perancangan antarmuka profil pemain pada Gambar 5.15, dan Penjelasan Detail dari perancangan antarmuka tersebut dijelaskan pada Tabel 5.3.



Gambar 5.17 Perancangan antarmuka Presentase Pemain Iterasi Pertama

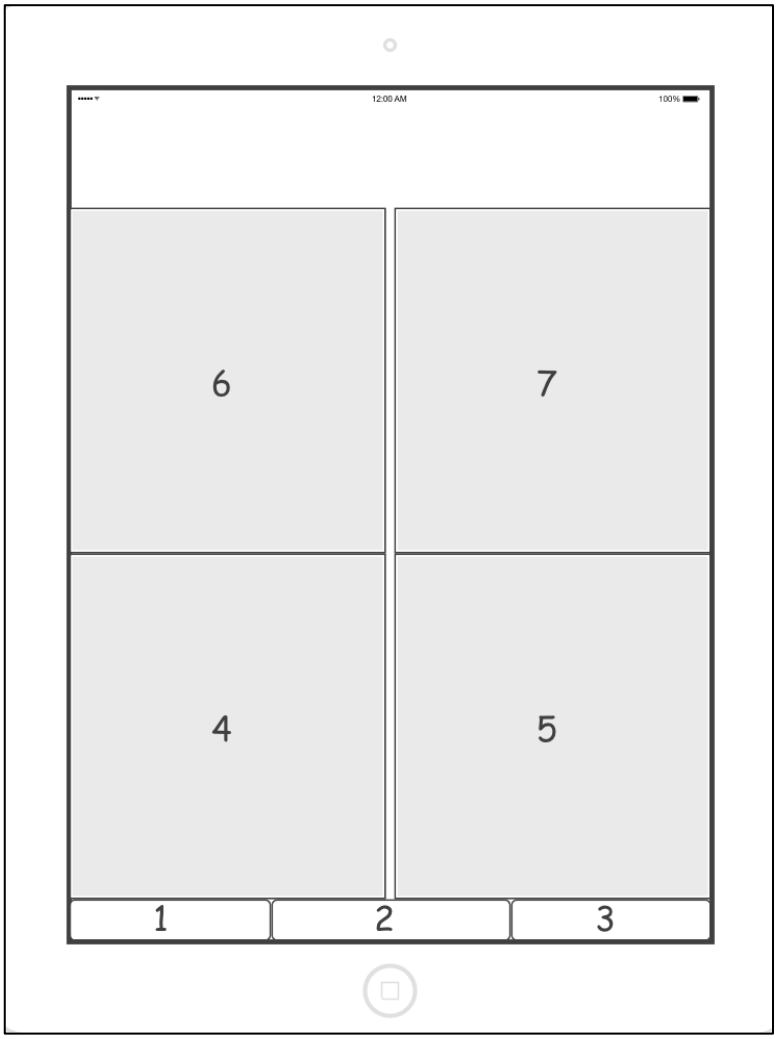
Tabel 5.9 Penjelasan Antarmuka Presentase Pemain Iterasi Pertama

No.	Nama Objek	Tipe	Keterangan
1	Tab Bar Profil	Logo *Button*	Tab baru untuk menampilkan menu profil
2	Tab Bar *Input Statistics*	Logo *Button*	Tab baru untuk menampilkan menu *Input statistics*

3	Tab Bar Presentase Aksi Pemain	Logo *Button*	Tab baru untuk menampilkan menu Presentase aksi pemain
4	Tab Bar menu lainnya	Logo *Button*	Tab baru untuk menampilkan menu lainnya
5	Presentase jumlah aksi pemain 2	*Text*	Menampilkan hasil perhitungan presentase dari jumlah aksi yang dilakukan pemain 2
6	Presentase jumlah kesalahan pemain 2	*Text*	Menampilkan hasil perhitungan presentase dari jumlah kesalahan yang dilakukan pemain 2
7	Presentase jumlah aksi pemain 1	*Text*	Menampilkan hasil perhitungan presentase dari jumlah aksi yang dilakukan pemain 1
8	Presentase jumlah kesalahan pemain 1	*Text*	Menampilkan hasil perhitungan presentase dari jumlah kesalahan yang dilakukan pemain 1

5.1.6.6 Perancangan Antarmuka Presentase Aksi Pemain Iterasi Pertama

Perancangan antarmuka profil pemain pada Gambar 5.3, dan Penjelasan Detail dari perancangan antarmuka tersebut dijelaskan pada Tabel 5.3.



Gambar 5.18 Perancangan antarmuka Presentase Pemain Iterasi Pertama

Tabel 5.10 Penjelasan Antarmuka Presentase Pemain Iterasi Pertama

No.	Nama Objek	Tipe	Keterangan
1	Tab Bar *Input Statistics*	Logo *Button*	Tab baru untuk menampilkan menu *Input statistics*
2	Tab Bar Presentase Aksi Pemain	Logo *Button*	Tab baru untuk menampilkan menu Presentase aksi pemain
3	Tab Bar menu lainnya	Logo *Button*	Tab baru untuk menampilkan menu lainnya
4	Presentase jumlah aksi pemain 2	*Text*	Menampilkan hasil perhitungan presentase dari jumlah aksi yang

			dilakukan pemain 2
5	Presentase jumlah kesalahan pemain 2	*Text*	Menampilkan hasil perhitungan presentase dari jumlah kesalahan yang dilakukan pemain 2
6	Presentase jumlah aksi pemain 1	*Text*	Menampilkan hasil perhitungan presentase dari jumlah aksi yang dilakukan pemain 1
7	Presentase jumlah kesalahan pemain 1	*Text*	Menampilkan hasil perhitungan presentase dari jumlah kesalahan yang dilakukan pemain 1

5.2 Implementasi Sistem

Setelah proses analisis kebutuhan selesai dilakukan, tahap berikutnya adalah perancangan. Perancangan dilakukan berdasarkan hasil dari analisis kebutuhan yang telah dilakukan. Proses perancangan Badstics ini dibagi menjadi empat tahap yaitu perancangan arsitektur, perancangan komponen, perancangan Data, dan perancangan antarmuka.

5.2.1 Spesifikasi Sistem

5.2.1.1 Spesifikasi Perangkat Keras

Tabel 5.11 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Laptop	1. MacBook Pro (13-inch, 2016, Two Thunderbolt 3 ports) 2. Processor 2 GHz Intel Core i5 3. Memory 8 GB 1867 MHz LPDDR3 4. Graphic Intel Iris Graphics 540 1536 MB

5.2.1.2 Spesifikasi Perangkat Lunak

Tabel 5.12 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
*Edit*or Dokumentasi	Microsoft Office Word 2015
*Edit*or Perancangan	*Visual Paradigm*
*Edit*or Pemrograman	Xcode 9.0
Bahasa Pemrograman	Swift 4
DBMS	*CoreData*
Emulator	*Apple iPad 10.5”*

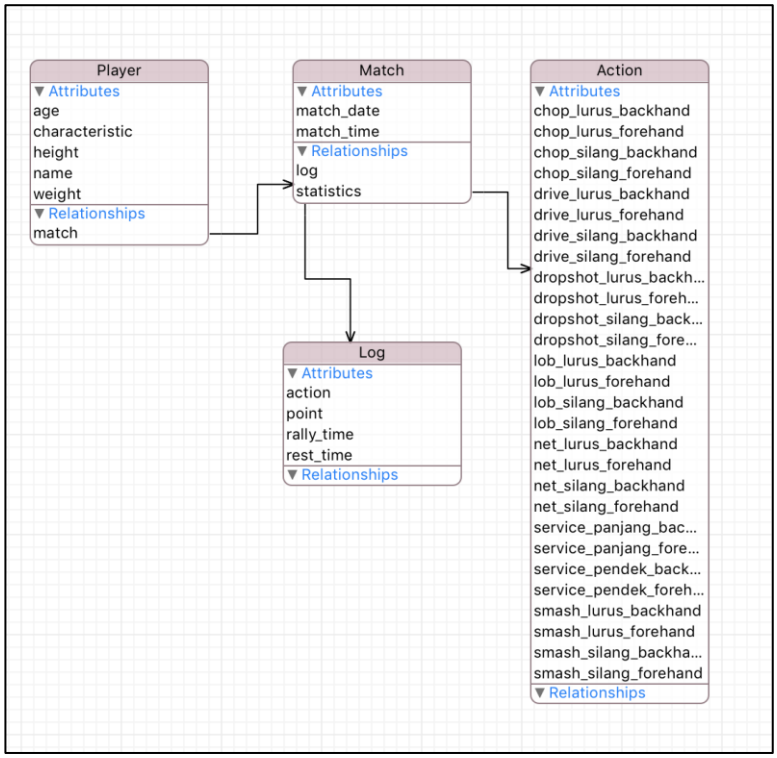
5.2.1.3 Spesifikasi Sistem Operasi

Tabel 5.13 Penjelasan Spesifikasi Sistem Operasi

Nama Komponen	Spesifikasi
Sistem Oprasi	macOS High Sierra 10.13.1

5.2.2 Implementasi Basis Data

Implementasi data dilakukan berdasarkan analisis data yang dilakukan pada analisis data sebelumnya yang telah dilakukan sesuai dengan kebutuhan sistem.



Gambar 5.19 Perancangan *physical data model (PDM)*

Entitas *Player* pada PDM diatas merupakan representasi data dari atlet yang akan disimpan datanya lalu entitas tersebut berhubungan dengan Match yang mempunyai hubungan dengan entitas Action dan Log yang menyimpan semua *history* dari permainan yang sedang berlangsung.

5.2.3 Implementasi Kode Program

Setelah proses analisis kebutuhan selesai dilakukan, tahap berikutnya adalah implementasi kode program seperti dibawah ini:

5.2.3.1 Implementasi Kode Program Klas StatisticsVC

Setelah proses analisis kebutuhan selesai dilakukan, tahap berikutnya adal

Nama Operasi : `actionButtonDidTap` ()

Source Code :

```
1  func actionButtonDidTap(_ indexPath: IndexPath) {
2
3          if         DataServices.instance.getState(forAction:
"isPlayerOne") {
4
5              //Set for player 1
6
7              //Changing state for player1 to player 2
8
9              DataServices.instance.setState(forState:
"isPlayerOne")
10
11
12             //IF SERVICE PRESSED
13
```

```
            if    inputStatistics[indexPath.row].title    ==
"SERVICE PENDEK" || inputStatistics[indexPath.row].title ==
"SERVICE PANJANG"{

                //Changin state isService

                DataServices.instance.setState(forState:
"isService")

            }


            //IF ERROR PRESSED

            if    inputStatistics[indexPath.row].title    ==
"FORCED ERROR" || inputStatistics[indexPath.row].title ==
"UNFORCED ERROR"{

                //Changin state isService

                DataServices.instance.setState(forState:
"isService")


                //Adding point to player 2

                DataServices.instance.addPoint(forSet:
self.gameSet, forPlayer: false)

                print("Set \(self.gameSet) -> Point of
player     2:     \(DataServices.instance.getPoint(forSet:
self.gameSet, forPlayer: false))")


                //Adding error count to player 1


DataServices.instance.addErrorCountPlayer(forError:
inputStatistics[indexPath.row].title, forPlayer: true)

                print("Player                                1
\(inputStatistics[indexPath.row].title)                    :
\(DataServices.instance.getErrorCountPlayer(forError:
inputStatistics[indexPath.row].title, forPlayer: true))")


                //If FORCED ERROR pressed

                if inputStatistics[indexPath.row].title ==
"FORCED ERROR" {

                    //Adding error cause if forced error to
player 1


DataServices.instance.addErrorActionCount(forAction:
previousAction.title,     forHand:     previousAction.code,
forPlayer: true)

                    print("Error          player          1
\(previousAction.title)                                    :
\(DataServices.instance.getErrorActionCount(forAction:
self.previousAction.title,                          forHand:
self.previousAction.code, forPlayer: true))")

```

```
                }


            } else {
                //Adding action for player 1

DataServices.instance.addActionCount(forAction:
inputStatistics[indexPath.row].title,              forHand:
inputStatistics[indexPath.row].code, forPlayer: true)


                print("Player                                1
\(inputStatistics[indexPath.row].title)                    :
\(DataServices.instance.getActionCount(forAction:
inputStatistics[indexPath.row].title,              forHand:
inputStatistics[indexPath.row].code, forPlayer: true))")
            }
        } else {
            //Set for player 2
            //Changing state for player 2 to player 1
            DataServices.instance.setState(forState:
"isPlayerOne")


            //IF SERVICES PRESSED
            if    inputStatistics[indexPath.row].title    ==
"SERVICE PENDEK" || inputStatistics[indexPath.row].title ==
"SERVICE PANJANG"{
                //Changin state isService
                DataServices.instance.setState(forState:
"isService")
            }


            //IF ERROR PRESSED
            if    inputStatistics[indexPath.row].title    ==
"FORCED ERROR" || inputStatistics[indexPath.row].title ==
"UNFORCED ERROR"{
                //Changin state isService
                DataServices.instance.setState(forState:
"isService")


                //Adding point to player 1
                DataServices.instance.addPoint(forSet:
self.gameSet, forPlayer: true)
                print("Set  \(self.gameSet)  ->  Point  of
player     1:     \(DataServices.instance.getPoint(forSet:
self.gameSet, forPlayer: true))")

```

```
132
133                 //Adding error count to player 1
134
135
136 DataServices.instance.addErrorCountPlayer(forError:
137 inputStatistics[indexPath.row].title, forPlayer: false)
138
139                 print("Player                                  2
140 \(inputStatistics[indexPath.row].title)                       :
151 \(DataServices.instance.getErrorCountPlayer(forError:
152 inputStatistics[indexPath.row].title, forPlayer: false))")
153
154
155                 //If FORCED ERROR pressed
156                 if  inputStatistics[indexPath.row].title  ==
157 "FORCED ERROR" {
158
159                     //Adding error cause if forced error
160
161 DataServices.instance.addErrorActionCount(forAction:
162 previousAction.title,     forHand:     previousAction.code,
163 forPlayer: false)
164
165                     print("Error          player          2
166 \(previousAction.title)                                     :
167 \(DataServices.instance.getErrorActionCount(forAction:
168 self.previousAction.title,                              forHand:
169 self.previousAction.code, forPlayer: false))")
170                 }
171
172
173             } else {
174
175                 //Adding action for player 2
176
177 DataServices.instance.addActionCount(forAction:
178 inputStatistics[indexPath.row].title,              forHand:
179 inputStatistics[indexPath.row].code, forPlayer: false)
180
181
182                 print("Player                                  2
183 \(inputStatistics[indexPath.row].title)                       :
184 \(DataServices.instance.getActionCount(forAction:
185 inputStatistics[indexPath.row].title,              forHand:
186 inputStatistics[indexPath.row].code, forPlayer: false))")
187
188             }
189
190         }
191
192
193         //Set PreviousAction
194         self.previousAction                                    =
195 inputStatistics[indexPath.row]
196
197     }
```

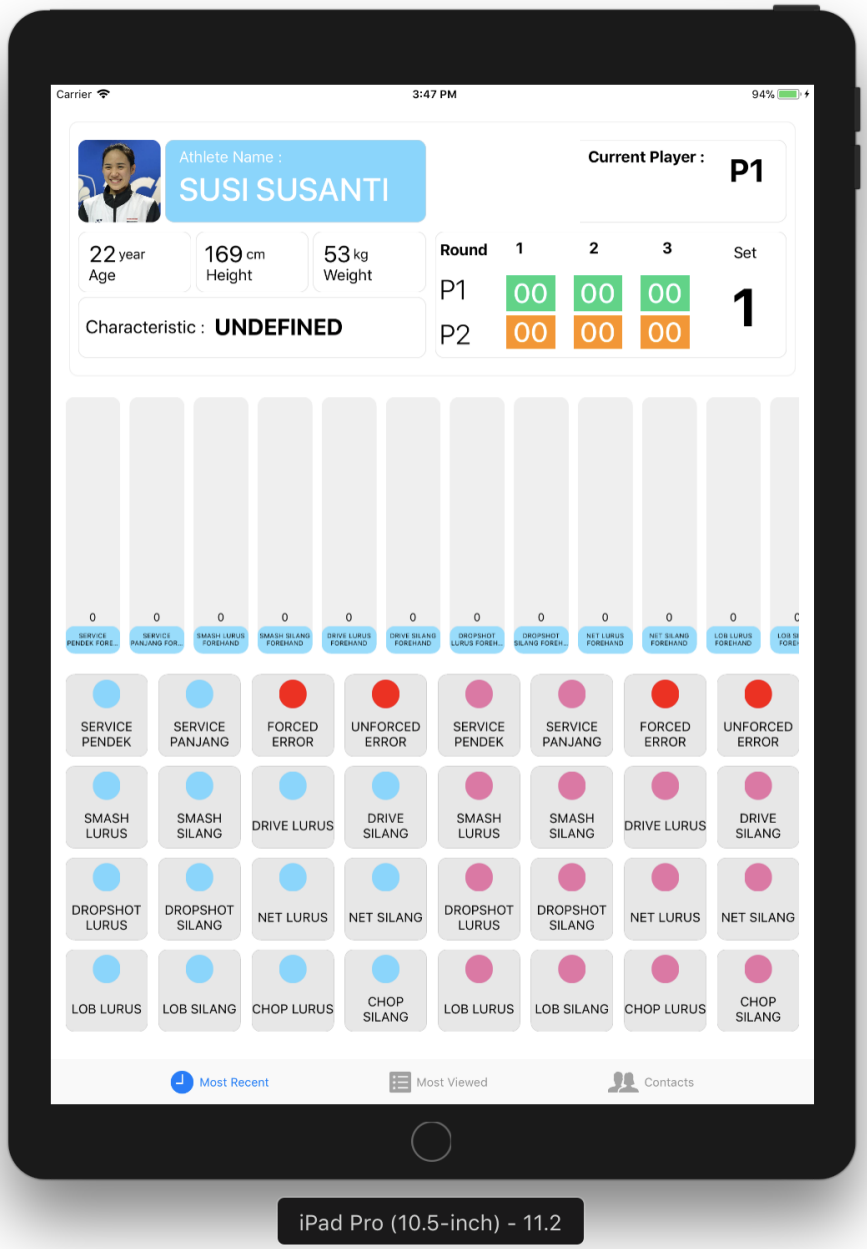
Tabel 5.14 Tabel Penjelasan Method

Nomor Baris	Penjelasan
8	merupakan pemanggilan method untuk merubah *state* dari *isPlayerOne*
14 – 22	merupakan seleksi kondisi yang memeriksa apakah aksi merupakan Service Pendek atau Service Panjang setelah itu jika pemeriksaan terpenuhi akan merubah *state* dari isService
26 – 73	merupakan seleksi kondisi yang memeriksa apakah aksi merupana *Forced Error* atau *Unforced Error*
31	jika pemeriksaan terpenuhi maka untuk *state isService* akan dirubah dan selanjutnya
37 – 42	akan menambahkan poin ke pemain lawan
47 – 53	menambahkan error pemain
57 – 73	merupakan seleksi kondisi yang memeriksa apakah aksi merupakan *Forced Error* atau bukan setelah itu jika pemeriksaan terpenuhi *error cause* pemain lawan akan ditambakan
201 - 189	merupakan seleksi kondisi yang memeriksa apakah aksi merupakan Service Pendek atau Service Panjang setelah itu jika pemeriksaan terpenuhi akan merubah *state* dari isService
104 - 131	merupakan seleksi kondisi yang memeriksa apakah aksi merupana *Forced Error* atau *Unforced Error*
125 – 130	jika pemeriksaan terpenuhi maka untuk *state isService* akan dirubah dan selanjutnya
135 - 151	akan menambahkan poin ke pemain lawan
155 - 170	merupakan seleksi kondisi yang memeriksa apakah aksi merupakan *Forced Error* atau bukan setelah itu jika pemeriksaan terpenuhi *error cause* pemain lawan akan ditambakan
194	*previous action* akan ditambahkan

5.2.4 Implementasi Antarmuka

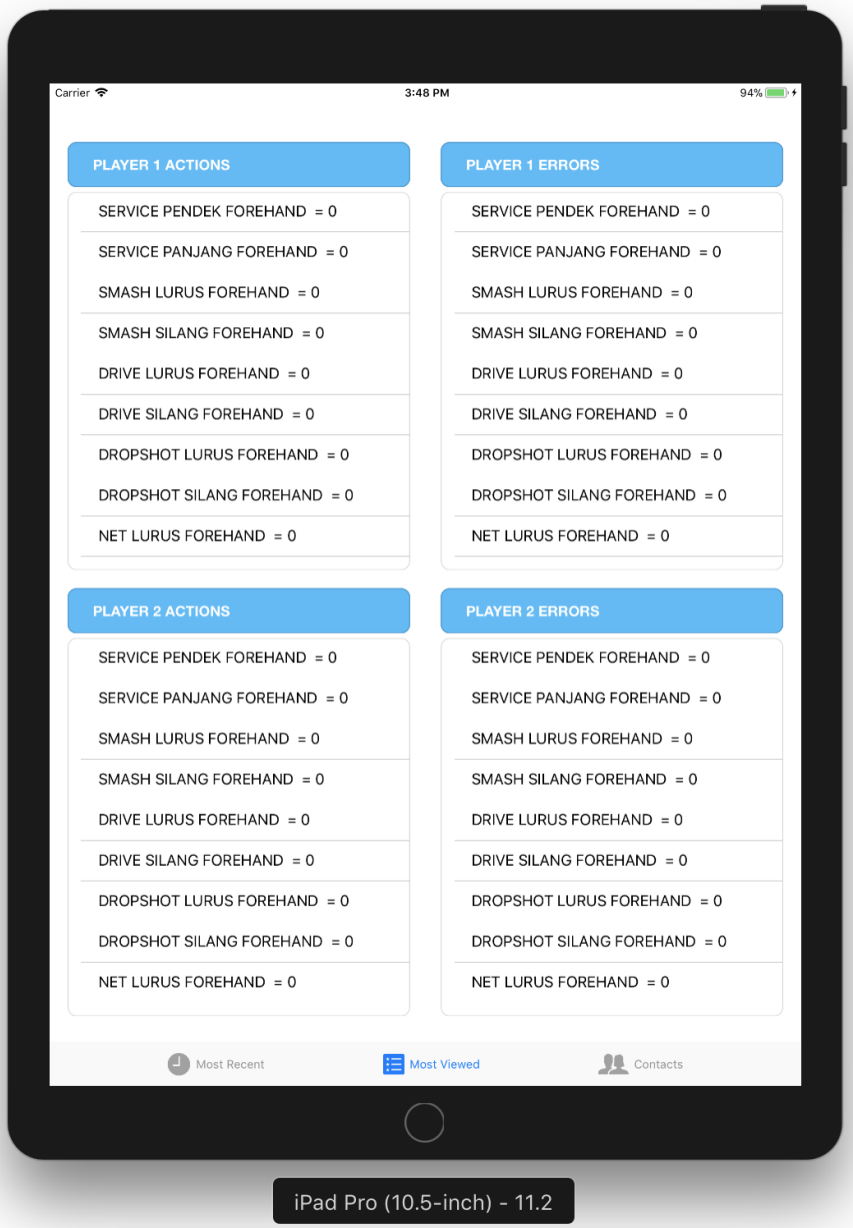
Pada Implementasi antarmuka ini menggambarkan antarmuka yang sudah diimplementasikan ke sistem.

5.2.4.1 Implementasi Antarmuka Profil



Gambar 5.20 Implementasi Antarmuka Input Statistic

5.2.4.2 Implementasi Antarmuka Statistics



Gambar 5.21 Implementasi Statistics Chart

BAB 6
PENGUJIAN

Tahap pengujian dilakukan setelah sudah melakukan implementasi sistem. Pengujian bertujuan untuk memeriksa apakah implementasi sudah sesuai dengan analisis kebutuhan dan perancangan sistem atau tidak. Tahap pengujian yang akan dilakukan adalah pengujian unit, pengujian integrasi, dan pengujian validasi.

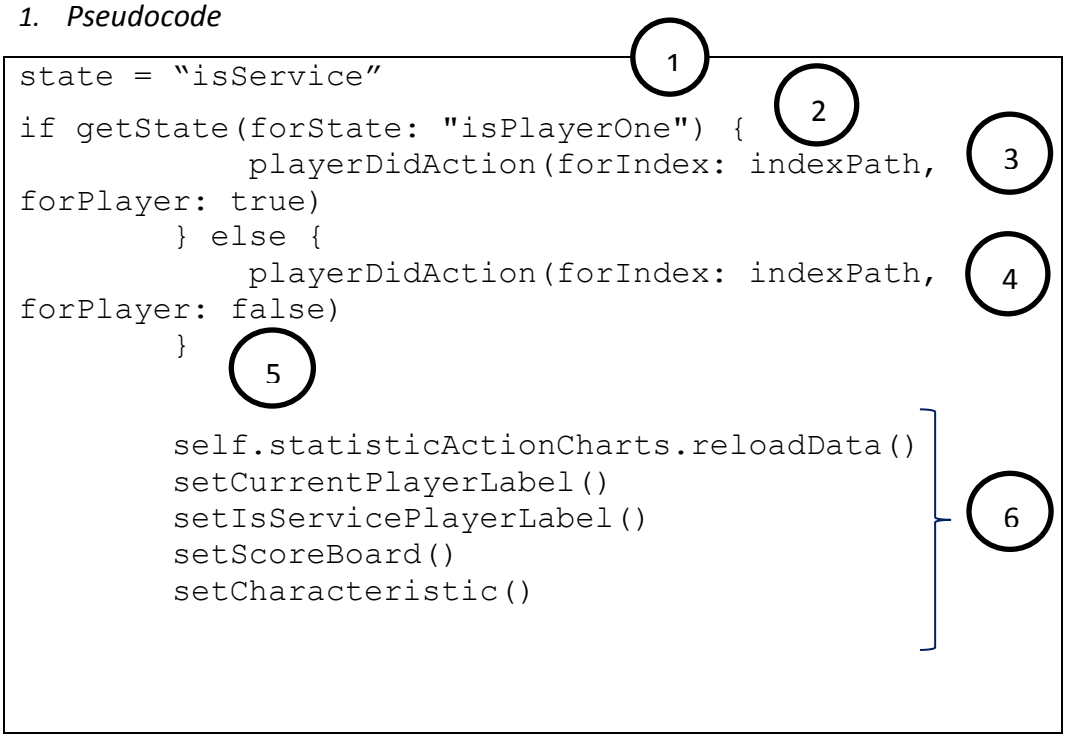
6.1 Pengujian Unit

Pengujian unit ini berkonsentrasi pada setiap unit (misalnya: komponen, klas, atau objek) dari perangkat lunak yang diterapkan. Pada pengujian unit ini menggunakan metode *whitebox testing* dengan teknik pengujian *basis path*. Metode ini memungkinkan perancang *test case* mendapatkan ukuran kompleksitas logika dari perancangan prosedural dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. *Test case* didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba. Langkah-langkah dalam pembuatan *test case* pada *basis path testing* adalah sebagai berikut:

1. Pembuatan diagram alir dari perancangan prosedural atau *source code*
2. Menentukan *cyclomatic complexcity* dari diagram alir
3. Menentukan *independent path* dari diagram alir

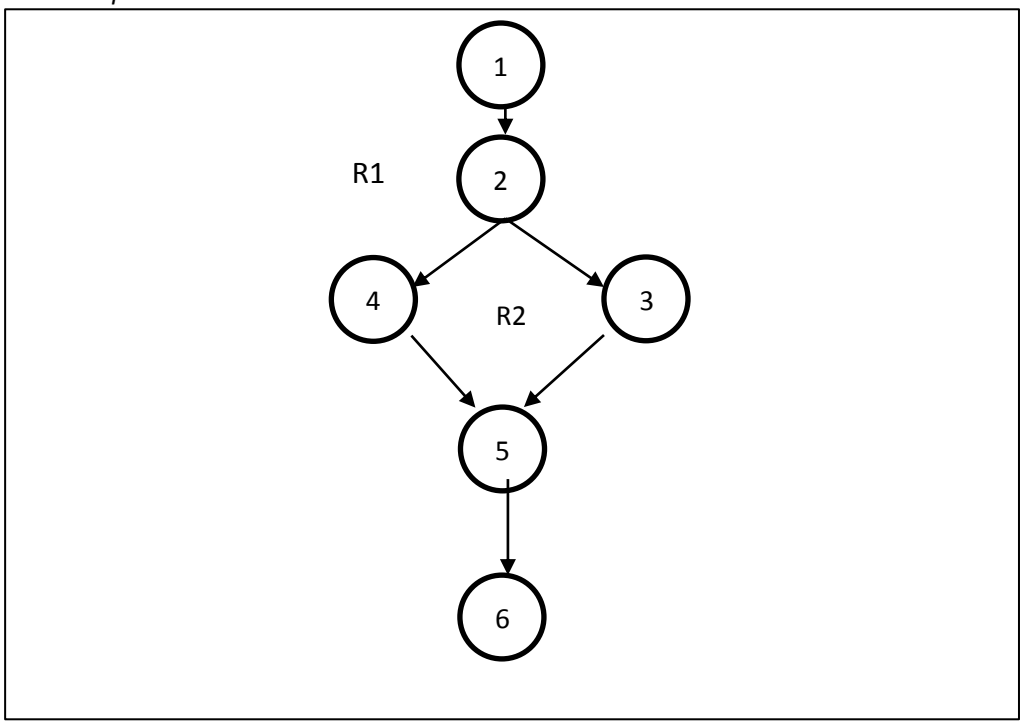
6.1.1 Pengujian Unit Klas StatisticVC Operasi actionButtonDidTap(IndexPath)

1. Pseudocode



2. *Basis Path Testing*

a. *Flow Graph*



b. *Cyclomatic Complexity*

- V(G) = 2, ada 2 *region* R1, R2
- V(G) = 6 *edges* – 6 *nodes* + 2 = 2
- V(G) = 1 *predicate nodes* + 1 = 2

c. *Independent Path*

- Jalur 1 = 1 – 2 – 3 – 4 – 5 – 6
- Jalur 2 = 1 – 2 – 4 – 5 – 6

Test case dan hasil akan dijelaskan pada tabel 6.1 dibawah ini.

Tabel 6.1 Hasil pengujian unit kelas admin operasi actionButtonDidTap()

No	No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1	Memanggil operasi playerDidAction dengan variabel state= true	Berhasil menambah aksi ke pemain 1	Berhasil menambah aksi ke pemain 1	Valid
2.	2	Memanggil operasi	Berhasil menambah aksi	Berhasil menambah aksi	Valid

		playerDidAction dengan variabel state= false	ke pemain 2	ke pemain 2	

6.1.2 Pengujian Unit Klas StatisticVC Operasi compareNaiveBayes(Int, Int, Int)

1. Pseudocode

```
let priorOffensive =
DataServices.instance.getPrior(forCharacteristic:
true)
        let priorDefensive =
DataServices.instance.getPrior(forCharacteristic:
false)

        let offensiveValue = priorOffensive *

DataServices.instance.getLikelihood(forAction: "Lob",
forCategory: lob, forCharacter: "Offensive") *

DataServices.instance.getLikelihood(forAction:
"Drive", forCategory: drive, forCharacter:
"Offensive") *

DataServices.instance.getLikelihood(forAction:
"Smash", forCategory: smash, forCharacter:
"Offensive")

        let defensiveValue = priorDefensive *

DataServices.instance.getLikelihood(forAction: "Lob",
forCategory: lob, forCharacter: "Defensive") *

DataServices.instance.getLikelihood(forAction:
"Drive", forCategory: drive, forCharacter:
"Defensive") *

DataServices.instance.getLikelihood(forAction:
"Smash", forCategory: smash, forCharacter:
"Defensive")

        print("Offensive : \(offensiveValue)
\nDefensive : \(defensiveValue)")

        if offensiveValue > defensiveValue {
```

1

2

```
        return true
    } else {
        return false
    }
```

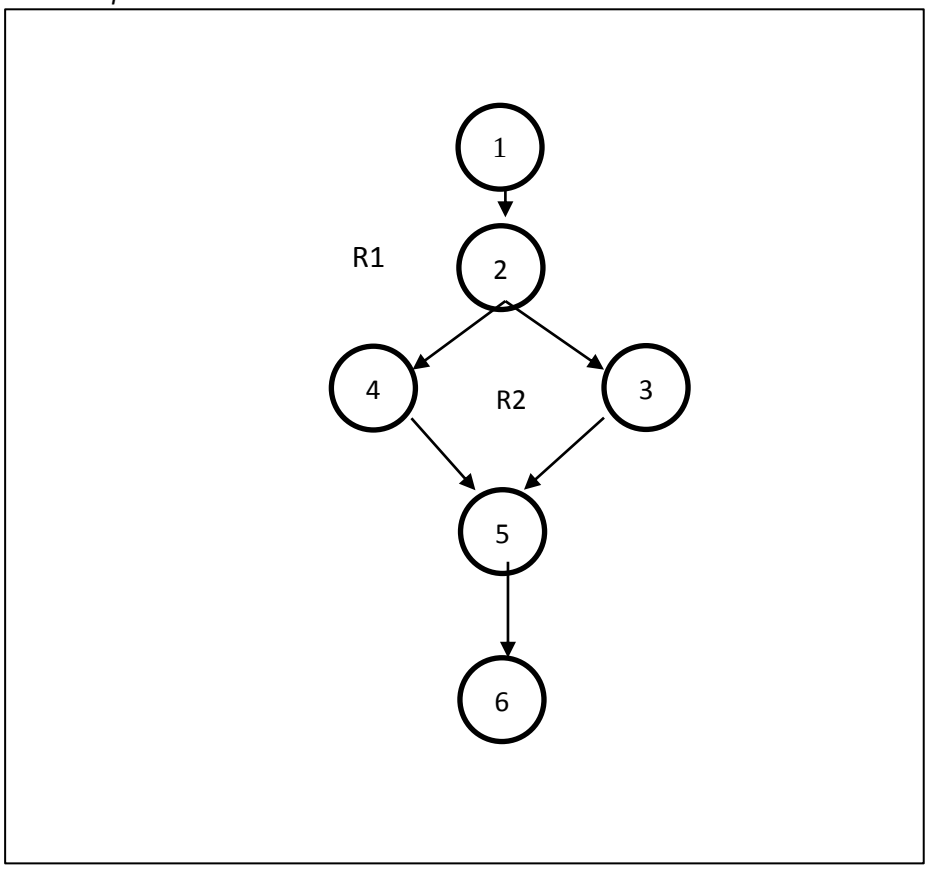
3

4

5

2. *Basis Path Testing*

a. *Flow Graph*



b. *Cyclomatic Complexity*

- V(G) = 2, ada 2 *region* R1, R2
- V(G) = 6 *edges* – 6 *nodes* + 2 = 2
- V(G) = 1 *predicate nodes* + 1 = 2

c. *Independent Path*

- Jalur 1 = 1 – 2 – 3 – 4 – 5 – 6
- Jalur 2 = 1 – 2 – 4 – 5 – 6

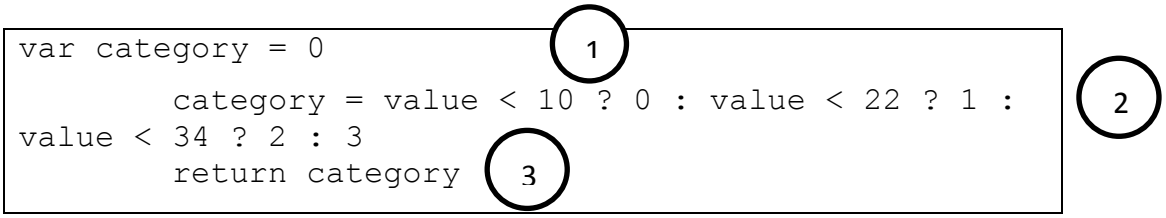
Test case dan hasil akan dijelaskan pada tabel 6.1 dibawah ini.

Tabel 6.2 Hasil pengujian unit kelas admin operasi compareNaiveBayes()

No	No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1.	1	Memanggil operasi compareNaiveBayes dengan lob = 34, drive = 18, smash 20	Operasi akan mengembalikan nilai false	Berhasil mengembalikan nilai false	Valid
2.	2	Memanggil operasi compareNaiveBayes dengan lob = 13, drive = 25, smash 34	Operasi akan mengembalikan nilai false	Berhasil mengembalikan nilai true	Valid

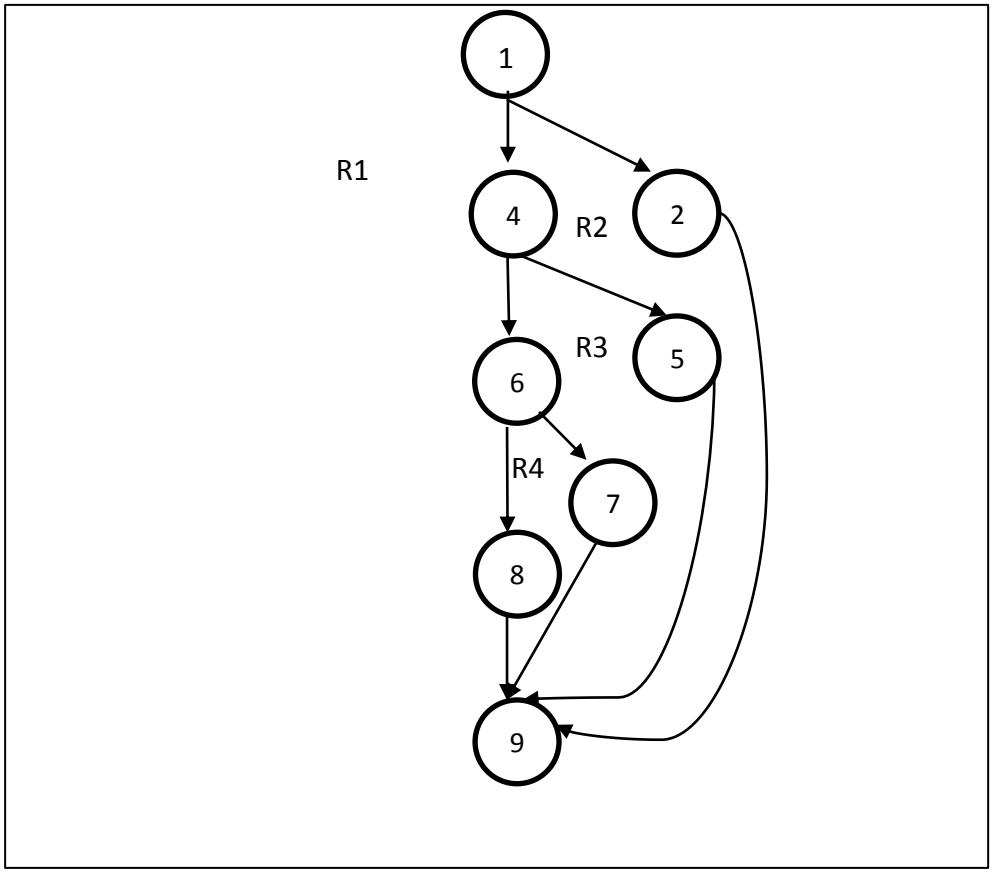
6.1.3 Pengujian Unit Klas StatisticVC Operasi getCategoryValue(Int)

1. Pseudocode



2. Basis Path Testing

a. Flow Graph



b. Cyclomatic Complexity

- V(G) = 4, ada 1 *region* R4
- V(G) = 11 *edges* – 9 *nodes* + 2 = 4
- V(G) = 3 *predicate nodes* + 1 = 4

c. Independent Path

- Jalur 1 = 1 – 2 – 9
- Jalur 2 = 1 – 4 – 5 – 9
- Jalur 3 = 1 – 4 – 6 – 7 – 9
- Jalur 4 = 1 – 4 – 6 – 7 – 8 – 9

Test case dan hasil akan dijelaskan pada tabel 6.1 dibawah ini.

Tabel 6.3 Hasil pengujian unit kelas admin operasi getCategoryValue(Int)

No	No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1.	1	Memanggil operasi getCategoryValue dengan	Operasi berhasil mengembalikan nilai 0	Berhasil mengembalikan nilai 0	Valid

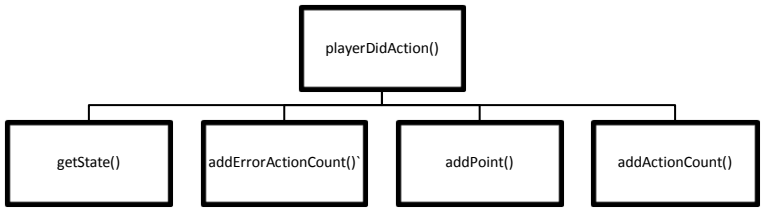
		parameter value = 7			
2.	2	Memanggil operasi getCategoryValue dengan parameter value = 15	Operasi berhasil mengembalikan nilai 1	Berhasil mengembalikan nilai 1	Valid
3.	3	Memanggil operasi getCategoryValue dengan parameter value = 28	Operasi berhasil mengembalikan nilai 2	Berhasil mengembalikan nilai 2	Valid
4.	4	Memanggil operasi getCategoryValue dengan parameter value = 38	Operasi berhasil mengembalikan nilai 3	Berhasil mengembalikan nilai 3	Valid

6.2 Pengujian Integrasi

Pengujian intergasi adalah fase dari semua proses pengujian dimana setiap modul dalam aplikasi digabungkan dan diuji lalu dievaluasi antar pengujian tersebut. Pengujian integrasi menguji setiap modul yang terintegrasi dan diuji berkelompok, karena biasanya program dibuat berdasarkan beberapa modul dan dikodekan oleh beberapa *programmer*. Ada dua strategi dalam pengujian integrasi pada *Non Incremental software integration* yaitu ada *Big Bang Strategy* dan *Incremental software integration* yaitu *Top-Down software integration, Bottom-up software integration* dan *Sandwich integration.* Dalam pengujian kali ini digunakan *Bottom-up,* dalam pengujian integrasi menggunakan *bottom-up* terdapat *Driver* yaitu sebuah *mock* atau gambaran dari sebuah fungsi yang digunakan sebagai parameter dalam suatu fungsi.

6.2.1 Pengujian Integrasi Menambah Aksi Pemain

a. Pengujian integrasi method playerDidAction()

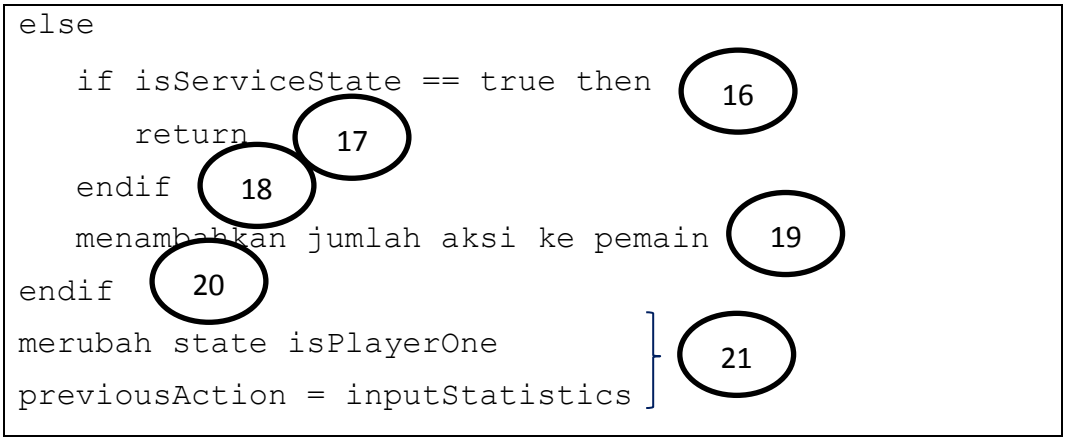


Gambar 6.1 Hirarki Fungsi playerDidAction

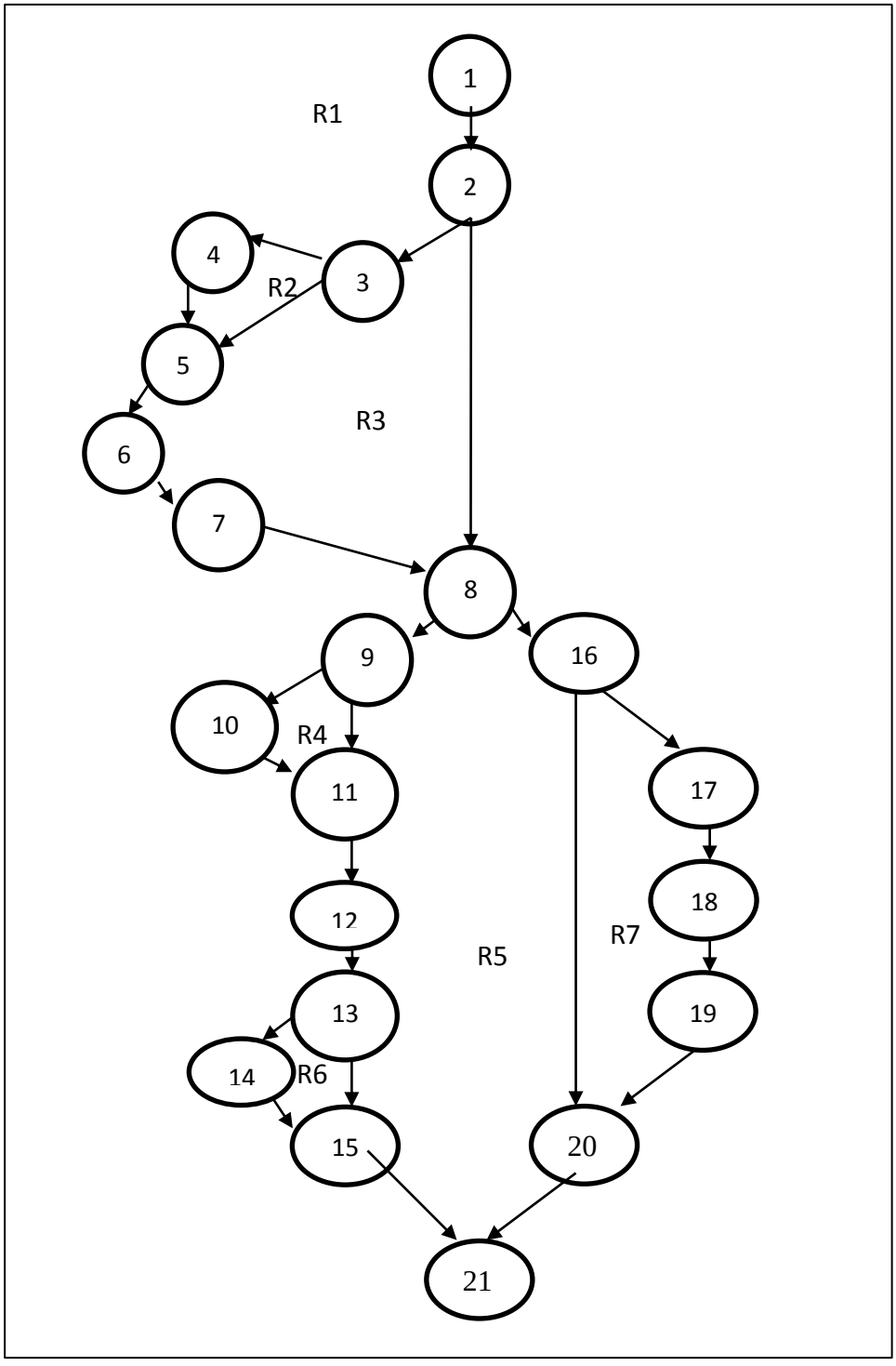
Pengujian unit playerDidAction(IndexPath, Bool)

1. Pseudocode

```
Inisialisasi title = title                                          1
if title == "SERVICE PENDEK" or title == "SERVICE PANJANG" then     2
   if isServiceState != true then                                   3
      return                                                        4
   endif                                                            5
   memulai waktu permainan                                          6
   merubah state untuk "isService"
endif                                                               7
if title == "FORCED ERROR" or title == "UNFORCED ERROR" then        8
   if isServicestate != true then                                   9
      merubah state untuk "isService"                               10
   endif                                                            11
   memulai waktu permainan                                          12
   menambahkan point untuk pemain lawan
   menambahkan error cause untuk pemain
   if title == "FORCED ERROR" then                                  13
      menambahkan error cause untuk pemain                          14
      menambahkan jumlah error untuk pemain
   endif                                                            15
```



2. *Basis Path Testing*
   a. *Flow Graph*



b. Cyclomatic Complexity

- V(G) = 7, ada 7 *region* R1, R2, R3, R4, R5, R6, R7
- V(G) = 26 *edges* – 21 *nodes* + 2 = 7
- V(G) = 6 *predicate nodes* + 1 = 7

c. Independent Path

- Jalur 1 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 16 – 17 – 18 – 19 – 20 –21
- Jalur 2 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 16 – 20 –21
- Jalur 3 = 1 – 2 – 8 – 9 – 10 – 11 – 12 – 13 –14 – 15 –21
- Jalur 4 = 1 – 2 – 8 – 9 – 11 – 12 – 13 –14 – 15 –21
- Jalur 5 = 1 – 2 – 8 – 9 – 11 – 12 – 13 – 15 –21
- Jalur 6 = 1 – 2 – 8 – 16 – 17 – 18 – 19 – 20 –21
- Jalur 7 = 1 – 2 – 8 – 16 – 20 –21

Test case dan hasil akan dijelaskan pada tabel 6.4 dibawah ini.

Tabel 6.4 Hasil pengujian unit kelas admin operasi playerDidAction()

No	No. Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1	Memanggil operasi playerDidAction dengan inputan variabel title "SERVICE PENDEK"; Player = true dan isService state = true	Berhasil menambahkan 1 nilai pada aksi "Service Pendek" pada player 1 dan memulai waktu permainan	Berhasil menambahkan value "Service Pendek" pada player 1 dan memulai waktu permainan	Valid
2.	2	Memanggil operasi playerDidAction dengan inputan variabel title "SERVICE PENDEK"; Player = true dan isService state = false	Tidak berhasil menambahkan 1 nilai pada aksi "Service Pendek" pada player 1 dan tanpa memulai waktu permainan	Tidak berhasil menambahkan value "Service Pendek" pada player 1 dan tanpa memulai waktu permainan	Valid
3	3	Memanggil operasi playerDidAction dengan inputan variabel title "FORCED ERROR"; Player = true; isService	Berhasil menambahkan 1 nilai pada aksi "FORCED ERROR" pada player 1 dan tanpa memulai waktu permainan,	Berhasil menambahkan 1 nilai pada aksi "FORCED ERROR" pada player 1 dan tanpa memulai waktu	Valid

		state = false	menambahkan point untuk pemain lawan, dan menyimpan error cause dari pemain	permainan, menambahkan point untuk pemain lawan, dan menyimpan error cause dari pemain	
4	4	Memanggil operasi playerDidAction dengan inputan variabel title "FORCED ERROR"; Player = true; isService state = false	Berhasil menambahkan 1 nilai pada aksi "FORCED ERROR" pada player 1 dan memulai waktu permainan, menambahkan point untuk pemain lawan lalu menyimpan error cause dari pemain	Berhasil menambahkan 1 nilai pada aksi "FORCED ERROR" pada player 1 dan memulai waktu permainan, menambahkan point untuk pemain lawan lalu menyimpan error cause dari pemain	Valid
5	5	Memanggil operasi playerDidAction dengan inputan variabel title "UNFORCED ERROR"; Player = true; isService state = true	Berhasil menambahkan 1 nilai pada aksi "FORCED ERROR" pada player 1 dan memulai waktu permainan, menambahkan point untuk pemain lawan tanpa menyimpan error cause dari pemain	Berhasil menambahkan 1 nilai pada aksi "FORCED ERROR" pada player 1 dan memulai waktu permainan, menambahkan point untuk pemain lawan tanpa menyimpan error cause dari pemain	Valid
6	6	Memanggil operasi playerDidAction dengan inputan variabel title "SMASH LURUS FOREHAND"; Player = true	Tidak Berhasil menambahkan 1 nilai pada aksi "SMASH LURUS FOREHAND" pada player 1 dan menyimpan error cause dari	Menampilkan halaman login	Valid

		dan isService state = true	pemain		
7	7	Memanggil operasi playerDidAction dengan inputan variabel title "SMASH LURUS FOREHAND"; Player = true dan isService state = false	Berhasil menambahkan 1 nilai pada aksi "SMASH LURUS FOREHAND" pada player 1 dan menyimpan previous action dari pemain	Menampilkan halaman login	Valid

6.1.4 Pengujian Fungsi Menggunakan *Bottom-Up* dengan Driver

a. Source Code playerDidActionTest()

```
func playerDidActionTest(isPlayer: Bool) {
    let title = "FORCED ERROR" //Driver
    if title == "SERVICE PENDEK" || title == "SERVICE PANJANG"{
        if !getStateTest() {
            return
        }
        if !getStateTest() {
            setStateTest()
        }
        if getStateTest() {
            setStateTest()
        }
        setStateTest()
    }

    if title == "FORCED ERROR" || title == "UNFORCED ERROR"{
        if !getStateTest() {
            setStateTest()
        }
        if getStateTest() {
            setStateTest()
        }
        if !getStateTest() {
            setStateTest()
        }
        addPointTest()
        addErrorCountPlayerTest()
        if title == "FORCED ERROR"{
            addErrorCountPlayerTest()
        }
    } else {
        if getStateTest(){
            return
        }
        addActionCountTest()
    }
```

```
    setStateTest()
}
```

b. *Source Code addActionCountTest()*

```
func addActionCountTest() {
    var action = 1
    action += 1
    print(action)
}
```

c. *Source Code addErrorCountPlayerTest()*

```
func addErrorCountPlayerTest() {
    var errorAction = 1
    errorAction += 1
    print(errorAction)
}
```

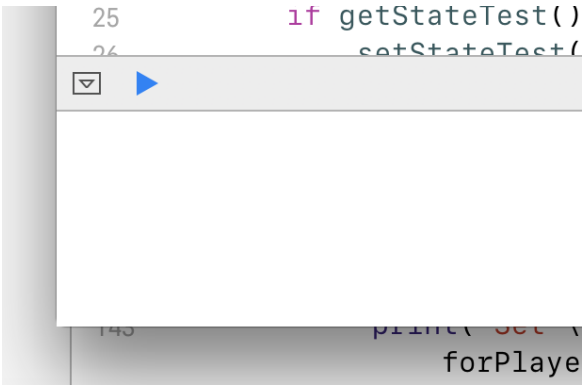
d. *Source Code addPointTest()*

```
func addPointTest() {
    var point = 1
    point += 1
    print(point)
}
```

e. *Source Code getStateTest()*

```
func getStateTest() -> Bool {
    return true
}
```

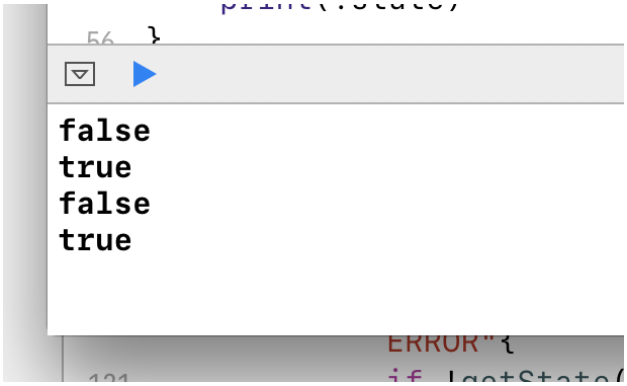
1. Uji integrasi dengan Driver title = "SERVIS PENDEK" dan isPlayer = true isService = true
   Output :



Gambar 6.2 Keluaran Pengujian Integrasi dengan Driver "SERVIS PENDEK", isPlayer = true dan isService = true

Program tidak dapat menambahkan aksi "SERVIS PENDEK" yang kedua karena setelah menekan aksi "SERVIS PENDEK" seharusnya pelatih menekan aksi lain.

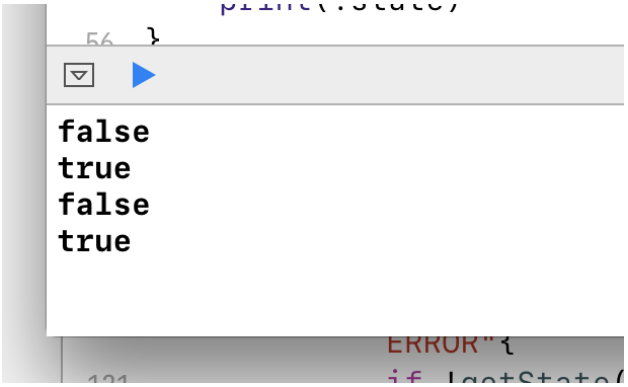
2. Uji intergasi denga Driver title = “SERVIS PENDEK” dan isPlayer = true isService = false
   Output :



Gambar 6.3 Keluaran Pengujian Integrasi dengan Driver “SERVIS PENDEK”, isPlayer = true dan isService = false

Program mengubah state isService menjadi false dan mengubah state isPlayer menjadi false melanjutkan menekan aksi lainnya.

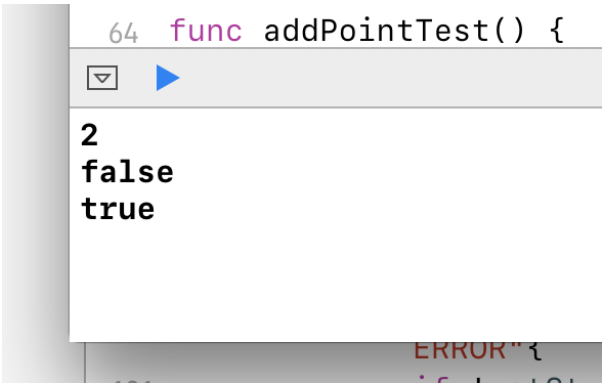
3. Uji intergasi denga Driver title = “SERVIS PENDEK” dan isPlayer = false isService = false
   Output:



Gambar 6.4 Keluaran Pengujian Integrasi dengan Driver “SERVIS PENDEK”, isPlayer = false dan isService = false

Program mengubah state isService menjadi false dan mengubah state isPlayer menjadi true melanjutkan menekan aksi lainnya.

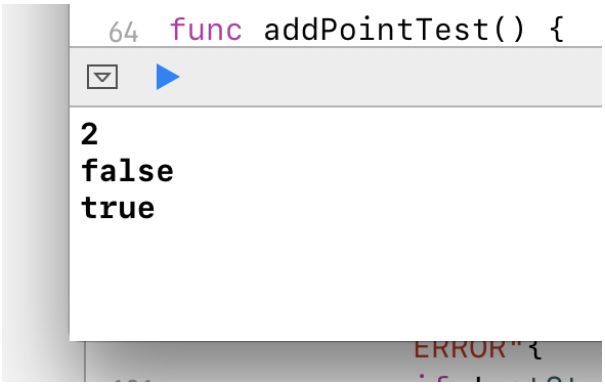
4. Uji intergasi denga Driver title = “SMASH LURUS FOREHAND” dan isPayer = true isService = false
   Output:



Gambar 6.5 Keluaran Pengujian Integrasi dengan Driver "SMASH LURUS FOREHAND", isPlayer = true dan isService = false

Program menambahkan 1 dari total aksi pemain sesuai drive yang dibuat yaitu 1 menjadi 2 dan mengubah state isPlayer menjadi false menjadi true.

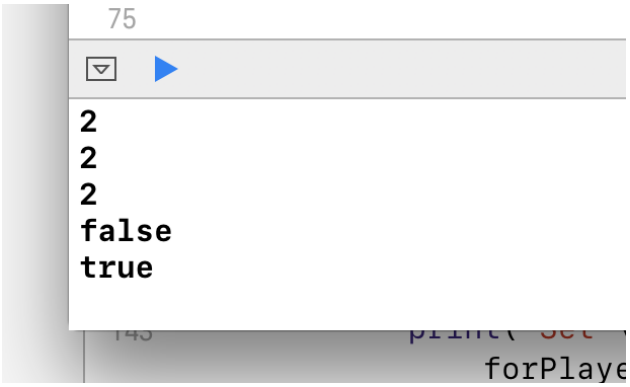
5. Uji intergasi denga Driver title = "SMASH LURUS FOREHAND" dan isPlayer = true isService = false
   Output:



Gambar 6.6 Keluaran Pengujian Integrasi dengan Driver "SMASH LURUS FOREHAND", isPlayer = true dan isService = false

Program menambahkan 1 dari total aksi pemain sesuai drive yang dibuat yaitu 1 menjadi 2 dan mengubah state isPlayer menjadi false menjadi true

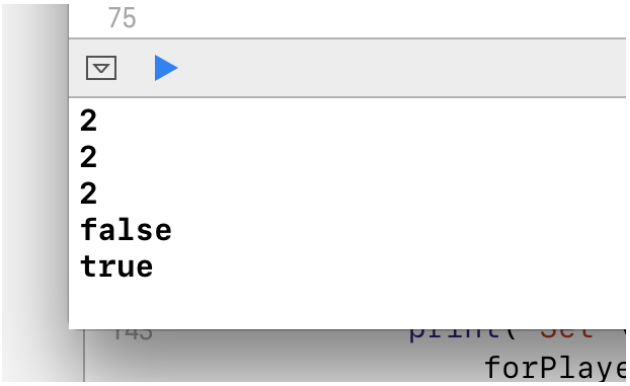
6. Uji intergasi denga Driver title = "FORCED ERROR" dan isPlayer = true isService = false
   Output:



Gambar 6.7 Keluaran Pengujian Integrasi dengan Driver "FORCED ERROR", isPlayer = true dan isService = false

Program menambahkan 1 dari total aksi pemain sesuai drive yang dibuat yaitu 1 menjadi 2, total aksi error pemain sesuai drive yang dibuat yaitu 1 menjadi 2, total poin pemain sesuai drive yang dibuat yaitu 1 menjadi 2 dan mengubah state isPlayer menjadi false menjadi true

7. Uji intergasi denga Driver title = "FORCED ERROR" dan isPlayer = false isService = false
   Output:



Gambar 6.8 Keluaran Pengujian Integrasi dengan Driver "FORCED ERROR", isPlayer = false dan isService = false

Program menambahkan 1 dari total aksi pemain sesuai drive yang dibuat yaitu 1 menjadi 2, total aksi error pemain sesuai drive yang dibuat yaitu 1 menjadi 2, total poin pemain sesuai drive yang dibuat yaitu 1 menjadi 2 dan mengubah state isPlayer menjadi false menjadi true

6.1.5 Pengujian Fungsi playerDidAction()

a. Source Code playerDidAction()

```
func playerDidAction(forIndex indexPath: IndexPath, forPlayer
isPlayer: Bool) {
        if inputStatistics[indexPath.row].title == "SERVICE
PENDEK" || inputStatistics[indexPath.row].title == "SERVICE
PANJANG"{
```

```
            if !getState(forState: "isService") {
                return
            }
            if !getState(forState: "isRallyTime") {
                setState(forState: "isRallyTime")
            }
            if getState(forState: "isRestTime") {
                setState(forState: "isRestTime")
            }
            setState(forState: "isService")
            setGameTimer(forGame: getState(forState:
"isGameStarted"))
        }

        if inputStatistics[indexPath.row].title == "FORCED
ERROR" || inputStatistics[indexPath.row].title == "UNFORCED
ERROR"{
            if !getState(forState: "isService") {
                setState(forState: "isService")
            }
            if getState(forState: "isRallyTime") {
                setState(forState: "isRallyTime")
            }
            if !getState(forState: "isRestTime") {
                setState(forState: "isRestTime")
            }
            setGameTimer(forGame: getState(forState:
"isGameStarted"))
            DataServices.instance.addPoint(forSet: self.gameSet,
forPlayer: !isPlayer)

            print("Set \(self.gameSet) -> Point of player 2:
\(DataServices.instance.getPoint(forSet: self.gameSet,
forPlayer: !isPlayer))")

            DataServices.instance.addErrorCountPlayer(forError:
inputStatistics[indexPath.row].title, forPlayer: isPlayer)

            print("Player
\(inputStatistics[indexPath.row].title) :
\(DataServices.instance.getErrorCountPlayer(forError:
inputStatistics[indexPath.row].title, forPlayer: isPlayer))")

            if inputStatistics[indexPath.row].title == "FORCED
ERROR" {

DataServices.instance.addErrorActionCount(forAction:
previousAction.title, forHand: previousAction.code, forPlayer:
isPlayer)
                print("Error player 1 \(previousAction.title) :
\(DataServices.instance.getErrorActionCount(forAction:
self.previousAction.title, forHand: self.previousAction.code,
forPlayer: isPlayer))")
            }
        } else {
            if getState(forState: "isService"){
                return
            }
            DataServices.instance.addActionCount(forAction:
```

```
inputStatistics[indexPath.row].title, forHand:
inputStatistics[indexPath.row].code, forPlayer: isPlayer)

            print("Player
\(inputStatistics[indexPath.row].title) :
\(DataServices.instance.getActionCount(forAction:
inputStatistics[indexPath.row].title, forHand:
inputStatistics[indexPath.row].code, forPlayer: isPlayer))")
        }
        setState(forState: "isPlayerOne")
        self.previousAction = inputStatistics[indexPath.row]
    }
```

b. Source Code addActionCount()

```
public func addActionCount(forAction action: String, forHand
hand: Bool, forPlayer player: Bool){
        let handAction = hand ? " FOREHAND" : " BACKHAND"
        if player {
            guard let value = player1Action[action + handAction]
else { return }
            player1Action[action + handAction] = value + 1
        }else{
            guard let value = player2Action[action + handAction]
else { return }
            player2Action[action + handAction] = value + 1
        }
    }
```

c. Source Code addErrorCount()

```
public func addErrorActionCount(forAction action: String,
forHand hand: Bool, forPlayer player: Bool){
        let handAction = hand ? " FOREHAND" : " BACKHAND"
        if player {
            guard let value = player1Error[action + handAction]
else { return }
            player1Error[action + handAction] = value + 1
        }else{
            guard let value = player2Error[action + handAction]
else { return }
            player2Error[action + handAction] = value + 1
        }
    }
```

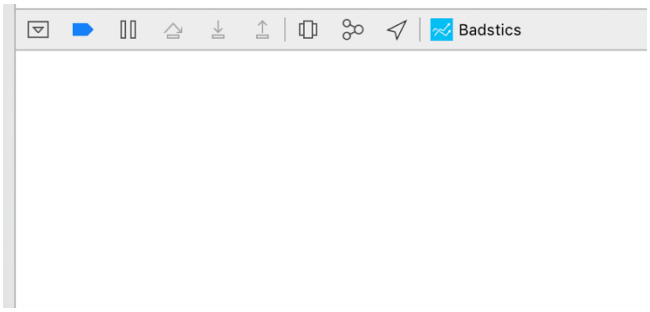
d. Source Code addPoint()

```
public func addPoint(forSet set: Int, forPlayer player: Bool){
        if player {
            guard let value = player1Point["POINT SET \(set)"]
else { return }
            player1Point["POINT SET \(set)"] = value + 1
        }else{
            guard let value = player2Point["POINT SET \(set)"]
else { return }
            player2Point["POINT SET \(set)"] = value + 1
        }
    }
```

e. Source Code getState()

```
public func setState(forState state: String) {
        self.state[state] = self.state[state]! ? false : true
    }
```

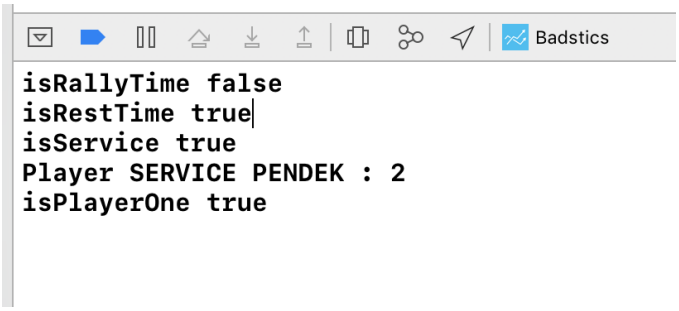
1. Uji fungsi dengan input title = "SERVIS PENDEK" dan isPlayer = true isService = true
   Output :



Gambar 6.9 Keluaran Pengujian Integrasi dengan Driver "SERVIS PENDEK", isPlayer = true dan isService = true

Program tidak dapat menambahkan aksi "SERVIS PENDEK" yang kedua karena setelah menekan aksi "SERVIS PENDEK" seharusnya pelatih menekan aksi lain

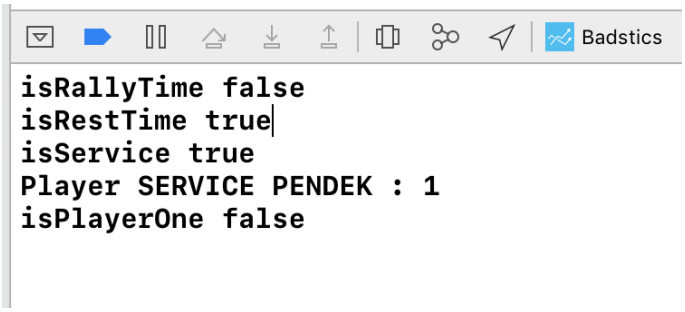
2. Uji fungsi dengan input title = "SERVIS PENDEK" dan isPlayer = true isService = false
   Output :



Gambar 6.10 Keluaran Pengujian Integrasi dengan Driver "SERVIS PENDEK", isPlayer = true dan isService = true

Program mengubah state isService menjadi true dan mengubah state isPlayer menjadi true dan menambahkan 1 pada total aksi pemain yang ditekan.

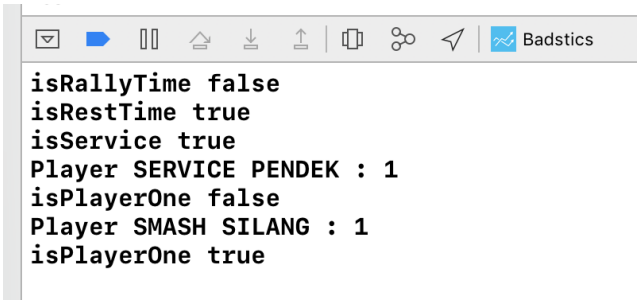
3. Uji fungsi dengan input title = "SERVIS PENDEK" dan isPlayer = false isService = false
   Output:



Gambar 6.11 Keluaran Pengujian Integrasi dengan Driver "SERVIS PENDEK", isPlayer = false dan isService = false

Program mengubah state isService menjadi true dan mengubah state isPlayer menjadi false dan menambahkan 1 pada total aksi pemain yang ditekan.

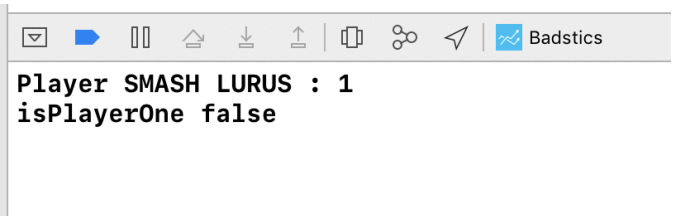
4. Uji fungsi dengan input title = "SMASH LURUS FOREHAND" dan isPlayer = true isService = false
   Output:



Gambar 6.12 Keluaran Pengujian Integrasi dengan Driver "SMASH LURUS FOREHAND", isPlayer = true dan isService = false

Program menambahkan 1 dari total aksi pemain sesuai dengan total sebelumnya dan mengubah state isPlayer menjadi false menjadi true.

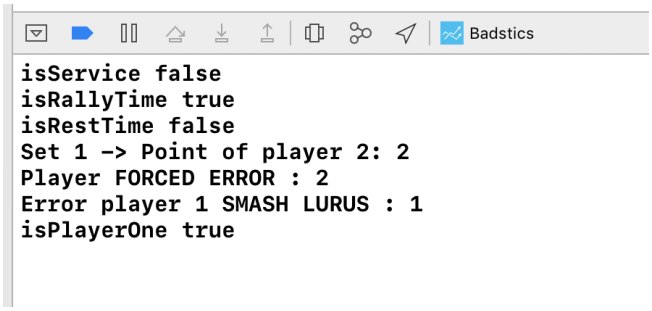
5. Uji fungsi dengan input title = "SMASH LURUS FOREHAND" dan isPlayer = false isService = false
   Output:



Gambar 6.13 Keluaran Pengujian Integrasi dengan Driver "SMASH LURUS FOREHAND", isPlayer = false dan isService = false

Program menambahkan 1 dari total aksi pemain sesuai dengan total sebelumnya dan mengubah state isPlayer menjadi false menjadi true

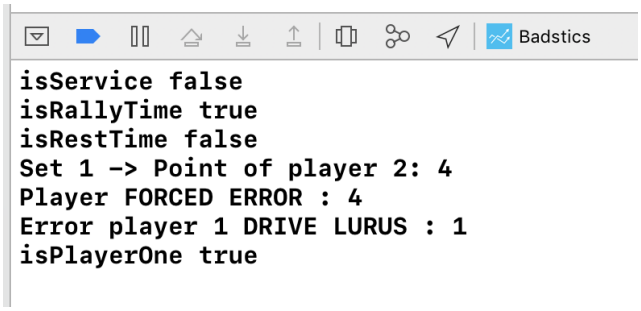
6. Uji fungsi dengan input title = “FORCED ERROR” dan isPlayer = true isService = false
   Output:



Gambar 6.14 Keluaran Pengujian Integrasi dengan Driver “FORCED ERROR”, isPlayer = true dan isService = false

Program menambahkan 1 dari total aksi pemain sesuai total aksi sebelumnya ditambah 1, total error aksi pemain sesuai total aksi sebelumnya ditambah 1, total poin pemain 2 ditambahkan 1 menjadi 4 dan mengubah state isPlayer menjadi false menjadi true

7. Uji fungsi dengan input title = “FORCED ERROR” dan isPlayer = false isService = false
   Output:



Gambar 6.15 Keluaran Pengujian Integrasi dengan Driver “FORCED ERROR”, isPlayer = false dan isService = false

Program menambahkan 1 dari total aksi pemain sesuai total aksi sebelumnya ditambah 1, total error aksi pemain sesuai total aksi sebelumnya ditambah 1, total poin pemain 1 ditambahkan 1 menjadi 4 dan mengubah state isPlayer menjadi false menjadi true

Test Case Number	*Test Case Description*	*Input Data*	*Expected Result*	*Actual Result*	*Pass/Fail*	*Remarks*
1	Pengujian dengan awal ketika pemain sudah mekalukan servis pendek dan akan menekan servis pendek yang kedua	title = “SERVIS PENDEK”, isPlayer = true dan isService True	Sistem berhasil menambahkan aksi pemain karena menekan servis ke player 1	Program menambahkan aksi servis ke pemain 1	*Pass*	-
2	Pengujian dengan awal ketika pemain belum mekalukan servis dan akan menekan servis pendek	title = “SERVIS PENDEK”, isPlayer = true dan isService false	Sistem tidak dapat menambahkan aksi pemain karena menekan servis pendek dua kali	Program tidak menambahkan aksi servis pendek dua kali	*Pass*	-
3	Pengujian dengan awal ketika pemain sudah mekalukan servis pendek dan akan menekan servis pendek yang kedua	title = “SERVIS PENDEK”, isPlayer = true dan isService True	Sistem berhasil menambahkan aksi pemain karena menekan servis ke player 2	Program menambahkan aksi servis ke pemain 2	*Pass*	-
4	Pengujian dengan menekan aksi pemain yang sudah melakukan servis	title = “SMASH LURUS FOREHAND”, isPlayer = true dan isService false	Sistem berhasil menambahkan aksi pemain kepada pemain 1	Program menambahkan aksi ke pemain 1	*Pass*	-

5	Pengujian dengan menekan aksi pemain yang sudah melakukan servis	title = “SMASH LURUS FOREHAND”, isPlayer = false dan isService false	Sistem berhasil menambahkan aksi pemain kepada pemain 1	Program menambahkan aksi ke pemain 2	*Pass*	-
6	Pengujian dengan menekan aksi Forced Error kepada pemain	title = “FORCED ERROR”, isPlayer = true dan isService false	Sistem berhasil menambahkan aksi ke pemain 1 dan menambahkan poin ke pemain 2	Program menambahkan aksi ke pemain 1 dan menambahkan poin ke pemain 2	*Pass*	-
7	Pengujian dengan menekan aksi Forced Error kepada pemain	title = “FORCED ERROR”, isPlayer = false dan isService false	Sistem berhasil menambahkan aksi ke pemain 2 dan menambahkan poin ke pemain 1	Program menambahkan aksi ke pemain 2 dan menambahkan poin ke pemain 1	*Pass*	-

6.3 Pengujian Validasi

Pengujian validasi dilakukan ketika kebutuhan yang sudah ditetapkan sebagai bagian dari pemodelan kebutuhan divalidasi terhadap perangkat lunak yang telah dibangun. Pengujian validasi dikatakan berhasil apabila fungsi yang ada pada perangkat lunak sesuai dengan yangg diharapkan. Pada pengujian validasi ini digunakan metode *blackbox testing*. Dengan *blackbox testing* memungkinkan pembuat perangkat lunak untuk memperoleh kondisi yang terjadi untuk suatu masukan yang akan menjalankan semua kebutuhan.

6.3.1 Pengujian Validasi Menambahkan Nilai Aksi Pemain dengan Tab Menu Input Statistics

a. Kasus Uji Berhasil Menambahkan Nilai Aksi SERVIS PENDEK FOREHAND

Tabel 6.5 Kasus Uji Berhasil Menambah Nilai Aksi SERVIS PENDEK FOREHAND

Nama Kasus Uji	Kasus Uji berhasil menambahkan nilai aksi SERVIS PENDEK FOREHAND
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Input Statistics 3. Menekan tombol SERVICE PENDEK FOREHAND
Hasil yang diharapkan	Sistem menambahkan 1 nilai pada aksi SERVIS PENDEK FOREHAND
Hasil	Sistem menambahkan 1 nilai pada aksi SERVIS PENDEK FOREHAND
Status	Valid

b. Kasus Uji Berhasil Menambahkan Nilai Aksi SERVIS PANJANG FOREHAND

Tabel 6.6 Kasus Uji Berhasil Menambah Nilai Aksi SERVIS PANJANG FOREHAND

Nama Kasus Uji	Kasus Uji berhasil menambahkan nilai aksi SERVIS PANJANG FOREHAND
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Input Statistics 3. Menekan tombol SERVICE PANJANG FOREHAND
Hasil yang diharapkan	Sistem menambahkan 1 nilai pada aksi SERVIS PANJANG FOREHAND
Hasil	Sistem menambahkan 1 nilai pada aksi SERVIS PANJANG FOREHAND
Status	Valid

6.3.2 Pengujian Validasi Menambahkan Nilai Aksi Player 1 dengan Tab Menu Presentase Aksi

a. Kasus Uji Berhasil Menambahkan Nilai Aksi SERVIS PENDEK FOREHAND Player 1

Tabel 6.7 Kasus Uji Berhasil Menambah Nilai Aksi SERVIS PENDEK FOREHAND Player 1

Nama Kasus Uji	Kasus Uji berhasil menambahkan nilai aksi SERVIS PENDEK FOREHAND Player 1
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Aksi 3. Menekan tombol SERVICE PENDEK FOREHAND PLAYER 1 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem menambahkan 1 nilai pada aksi SERVIS PENDEK FOREHAND Player 1
Hasil	Sistem menambahkan 1 nilai pada aksi SERVIS PENDEK FOREHAND Player 1
Status	valid

b. Kasus Uji Berhasil Menambahkan Nilai Aksi SERVIS PENDEK FOREHAND PLAYER 1

Tabel 6.8 Kasus Uji Berhasil Menambah Nilai Aksi SERVIS PANJANG FOREHAND PLAYER 1

Nama Kasus Uji	Kasus Uji berhasil menambahkan nilai aksi SERVIS PANJANG FOREHAND PLAYER 1
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Aksi 3. Menekan tombol SERVICE PANJANG FOREHAND PLAYER 1 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem menambahkan 1 nilai pada aksi SERVIS PANJANG FOREHAND PLAYER 1
Hasil	Sistem menambahkan 1 nilai pada aksi SERVIS PANJANG FOREHAND PLAYER 1

Status	valid

6.3.3 Pengujian Validasi Menambahkan Nilai Aksi Player 2 dengan Tab Menu Presentase Aksi

a. Kasus Uji Berhasil Menambahkan Nilai Aksi SERVIS PENDEK FOREHAND Player 2

Tabel 6.9 Kasus Uji Berhasil Menambah Nilai Aksi SERVIS PENDEK FOREHAND Player 2

Nama Kasus Uji	Kasus Uji berhasil menambahkan nilai aksi SERVIS PENDEK FOREHAND Player 2
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Aksi 3. Menekan tombol SERVICE PENDEK FOREHAND PLAYER 2 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem menambahkan 1 nilai pada aksi SERVIS PENDEK FOREHAND Player 2
Hasil	Sistem menambahkan 1 nilai pada aksi SERVIS PENDEK FOREHAND Player 2
Status	valid

b. Kasus Uji Berhasil Menambahkan Nilai Aksi SERVIS PENDEK FOREHAND PLAYER 2

6.3.4 Pengujian Validasi Menambahkan Nilai Error Aksi Player 1 dengan Tab Menu Presentase Error Aksi

a. Kasus Uji Berhasil Menambahkan Nilai Error Aksi SERVIS PENDEK FOREHAND Player 1

Tabel 6.10 Kasus Uji Berhasil Menambah Nilai Error Aksi SERVIS PENDEK FOREHAND Player 1

Nama Kasus Uji	Kasus Uji berhasil menambahkan nilai Error Aksi SERVIS PENDEK FOREHAND Player 1
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Error Aksi 3. Menekan tombol SERVICE PENDEK FOREHAND

	PLAYER 1 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem menambahkan 1 nilai pada Error Aksi SERVIS PENDEK FOREHAND Player 1
Hasil	Sistem menambahkan 1 nilai pada Error Aksi SERVIS PENDEK FOREHAND Player 1
Status	valid

b. Kasus Uji Berhasil Menambahkan Nilai Error Aksi SERVIS PENDEK FOREHAND PLAYER 1

Tabel 6.11 Kasus Uji Berhasil Menambah Nilai Error Aksi SERVIS PANJANG FOREHAND PLAYER 1

Nama Kasus Uji	Kasus Uji berhasil menambahkan nilai Error Aksi SERVIS PANJANG FOREHAND PLAYER 1
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Error Aksi 3. Menekan tombol SERVICE PANJANG FOREHAND PLAYER 1 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem menambahkan 1 nilai pada Error Aksi SERVIS PANJANG FOREHAND PLAYER 1
Hasil	Sistem menambahkan 1 nilai pada Error Aksi SERVIS PANJANG FOREHAND PLAYER 1
Status	valid

6.3.5 Pengujian Validasi Menambahkan Nilai Error Aksi Player 1 dengan Tab Menu Presentase Error Aksi

a. Kasus Uji Berhasil Menambahkan Nilai Error Aksi SERVIS PENDEK FOREHAND Player 1

Tabel 6.12 Kasus Uji Berhasil Menambah Nilai Error Aksi SERVIS PENDEK FOREHAND Player 2

Nama Kasus Uji	Kasus Uji berhasil menambahkan nilai Error Aksi SERVIS PENDEK FOREHAND Player 2
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Error Aksi

	3. Menekan tombol SERVICE PENDEK FOREHAND PLAYER 2 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem menambahkan 1 nilai pada Error Aksi SERVIS PENDEK FOREHAND Player 2
Hasil	Sistem menambahkan 1 nilai pada Error Aksi SERVIS PENDEK FOREHAND Player 2
Status	valid

b. Kasus Uji Berhasil Menambahkan Nilai Error Aksi SERVIS PENDEK FOREHAND PLAYER 2

Tabel 6.13 Kasus Uji Berhasil Menambah Nilai Error Aksi SERVIS PANJANG FOREHAND PLAYER 2

Nama Kasus Uji	Kasus Uji berhasil menambahkan nilai Error Aksi SERVIS PANJANG FOREHAND PLAYER 2
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Error Aksi 3. Menekan tombol SERVICE PANJANG FOREHAND PLAYER 2 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem menambahkan 1 nilai pada Error Aksi SERVIS PANJANG FOREHAND PLAYER 2
Hasil	Sistem menambahkan 1 nilai pada Error Aksi SERVIS PANJANG FOREHAND PLAYER 2
Status	valid

6.3.6 Pengujian Validasi Mengurangi Nilai Aksi Player 1 dengan Tab Menu Presentase Aksi

a. Kasus Uji Berhasil Mengurangi Nilai Aksi SERVIS PENDEK FOREHAND Player 1

Tabel 6.14 Kasus Uji Berhasil Menambah Nilai Aksi SERVIS PENDEK FOREHAND Player 1

Nama Kasus Uji	Kasus Uji berhasil Mengurangi nilai aksi SERVIS PENDEK FOREHAND Player 1
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Aksi 3. Menekan tombol SERVICE PENDEK FOREHAND

	PLAYER 1 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem Mengurangi 1 nilai pada aksi SERVIS PENDEK FOREHAND Player 1
Hasil	Sistem Mengurangi 1 nilai pada aksi SERVIS PENDEK FOREHAND Player 1
Status	valid

b. Kasus Uji Berhasil Mengurangi Nilai Aksi SERVIS PENDEK FOREHAND PLAYER 1

Tabel 6.15 Kasus Uji Berhasil Menambah Nilai Aksi SERVIS PANJANG FOREHAND PLAYER 1

Nama Kasus Uji	Kasus Uji berhasil Mengurangi nilai aksi SERVIS PANJANG FOREHAND PLAYER 1
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Aksi 3. Menekan tombol SERVICE PANJANG FOREHAND PLAYER 1 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem Mengurangi 1 nilai pada aksi SERVIS PANJANG FOREHAND PLAYER 1
Hasil	Sistem Mengurangi 1 nilai pada aksi SERVIS PANJANG FOREHAND PLAYER 1
Status	valid

6.3.7 Pengujian Validasi Mengurangi Nilai Aksi Player 2 dengan Tab Menu Presentase Aksi

a. Kasus Uji Berhasil Mengurangi Nilai Aksi SERVIS PENDEK FOREHAND Player 2

Tabel 6.16 Kasus Uji Berhasil Menambah Nilai Aksi SERVIS PENDEK FOREHAND Player 2

Nama Kasus Uji	Kasus Uji berhasil Mengurangi nilai aksi SERVIS PENDEK FOREHAND Player 2
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Aksi 3. Menekan tombol SERVICE PENDEK FOREHAND PLAYER 2 pada Table Player1Actions

	4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem Mengurangi 1 nilai pada aksi SERVIS PENDEK FOREHAND Player 2
Hasil	Sistem Mengurangi 1 nilai pada aksi SERVIS PENDEK FOREHAND Player 2
Status	valid

b. Kasus Uji Berhasil Mengurangi Nilai Aksi SERVIS PENDEK FOREHAND PLAYER 2

Tabel 6.17 Kasus Uji Berhasil Menambah Nilai Aksi SERVIS PANJANG FOREHAND PLAYER 2

Nama Kasus Uji	Kasus Uji berhasil Mengurangi nilai aksi SERVIS PANJANG FOREHAND PLAYER 2
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Aksi 3. Menekan tombol SERVICE PANJANG FOREHAND PLAYER 2 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem Mengurangi 1 nilai pada aksi SERVIS PANJANG FOREHAND PLAYER 2
Hasil	Sistem Mengurangi 1 nilai pada aksi SERVIS PANJANG FOREHAND PLAYER 2
Status	valid

6.3.8 Pengujian Validasi Mengurangi Nilai Error Aksi Player 1 dengan Tab Menu Presentase Error Aksi

a. Kasus Uji Berhasil Mengurangi Nilai Error Aksi SERVIS PENDEK FOREHAND Player 1

Tabel 6.18 Kasus Uji Berhasil Menambah Nilai Error Aksi SERVIS PENDEK FOREHAND Player 1

Nama Kasus Uji	Kasus Uji berhasil Mengurangi nilai Error Aksi SERVIS PENDEK FOREHAND Player 1
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Error Aksi 3. Menekan tombol SERVICE PENDEK FOREHAND PLAYER 1 pada Table Player1Actions

	4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem Mengurangi 1 nilai pada Error Aksi SERVIS PENDEK FOREHAND Player 1
Hasil	Sistem Mengurangi 1 nilai pada Error Aksi SERVIS PENDEK FOREHAND Player 1
Status	valid

b. Kasus Uji Berhasil Mengurangi Nilai Error Aksi SERVIS PENDEK FOREHAND PLAYER 1

Tabel 6.19 Kasus Uji Berhasil Menambah Nilai Error Aksi SERVIS PANJANG FOREHAND PLAYER 1

Nama Kasus Uji	Kasus Uji berhasil Mengurangi nilai Error Aksi SERVIS PANJANG FOREHAND PLAYER 1
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Error Aksi 3. Menekan tombol SERVICE PANJANG FOREHAND PLAYER 1 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem Mengurangi 1 nilai pada Error Aksi SERVIS PANJANG FOREHAND PLAYER 1
Hasil	Sistem Mengurangi 1 nilai pada Error Aksi SERVIS PANJANG FOREHAND PLAYER 1
Status	valid

6.3.9 Pengujian Validasi Mengurangi Nilai Error Aksi Player 1 dengan Tab Menu Presentase Error Aksi

a. Kasus Uji Berhasil Mengurangi Nilai Error Aksi SERVIS PENDEK FOREHAND Player 1

Tabel 6.20 Kasus Uji Berhasil Menambah Nilai Error Aksi SERVIS PENDEK FOREHAND Player 2

Nama Kasus Uji	Kasus Uji berhasil Mengurangi nilai Error Aksi SERVIS PENDEK FOREHAND Player 2
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Error Aksi 3. Menekan tombol SERVICE PENDEK FOREHAND

	PLAYER 2 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem Mengurangi 1 nilai pada Error Aksi SERVIS PENDEK FOREHAND Player 2
Hasil	Sistem Mengurangi 1 nilai pada Error Aksi SERVIS PENDEK FOREHAND Player 2
Status	valid

b. Kasus Uji Berhasil Mengurangi Nilai Error Aksi SERVIS PENDEK FOREHAND PLAYER 2

Tabel 6.21 Kasus Uji Berhasil Menambah Nilai Error Aksi SERVIS PANJANG FOREHAND PLAYER 2

Nama Kasus Uji	Kasus Uji berhasil Mengurangi nilai Error Aksi SERVIS PANJANG FOREHAND PLAYER 2
Prosedur	1. Membuka Aplikasi Badstics 2. Menekan tab Bar Presentasi Error Aksi 3. Menekan tombol SERVICE PANJANG FOREHAND PLAYER 2 pada Table Player1Actions 4. Menekan Decrease Pada *Alert*
Hasil yang diharapkan	Sistem Mengurangi 1 nilai pada Error Aksi SERVIS PANJANG FOREHAND PLAYER 2
Hasil	Sistem Mengurangi 1 nilai pada Error Aksi SERVIS PANJANG FOREHAND PLAYER 2
Status	valid

6.4 Pengujian Usabilitas

Pengujian usabilitas dilakukan untuk melakukan pengujian sistem dengan cara menguji sistem secara langsung kepada sejumlah responden untuk mengetahui tingkat kemudahan penggunaan sebuah sistem. Responden akan diminta untuk mengisi sebuah kuisioner berisi sejumlah pernyataan seputar kenyamanan dan kemudahan dalam menggunakan sistem yang telah dikembangkan.Dalam pengujian disabilitas tedapat tiga aspek pengujian yaitu dalam aspek efektivitas, efisiensi dan kepuasan pengguna(ISO 9241 – 11, 1998), akan tetapi dalam pengujian kali ini pengujian usabilitas yang dilakukan hanya pada aspek efektivitas yaitu adalah tingkat akurasi dan kelengkapan pengguna dilakukan untuk mencapai tujuan yang ditetapkan. Adapun cara penghitungan metrik efektivitas dapat dilihat pada persamaan 6.1 dibawah ini(Usabilitygeek, 2015)

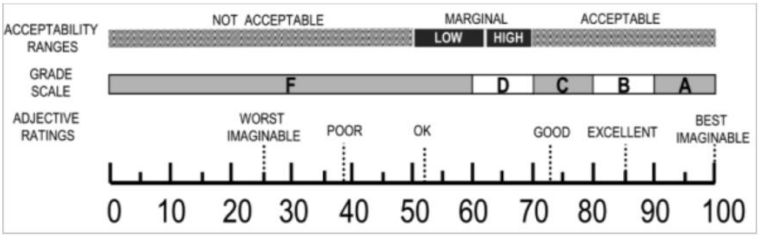
$$Efektivitas = \frac{jumlah\ task\ yang\ berhasil\ dilakukan}{total\ jumlah\ task\ yang\ diberikan} \times 100\% \quad \textbf{(Persamaan 6.1)}$$

Sedangkan kinerja *usability* diukur dengan menggunakan pengujian *post study* dengan metode *System Usability Scale (SUS)* (Brooke, 2013). *SUS* merupakan sebuah pengujian usability yang efektif dan andal untuk digunakan pada berbagai produk dan aplikasi (Bangor, Kortum, & Miller, 2009).

SUS terdiri dari 10 pertanyaan dengan menggunakan skala *likert* 1 sampai dengan 5 sesuai pada tabel 6.263. Pertanyaan nomor ganjil (1, 3, 5, 7, 9) merupakan pertanyaan dengan bernada positif. Sedangkan pertanyaan nomor genap (2, 4, 6, 8, 10) merupakan pertanyaan dengan bernada negatif seperti yang ditunjukkan pada tabel 2.263 Setiap pertanyaan diberi bobot antara 0-4. Penilaian dilakukan dengan pengurangan 1 nilai dari respon yang diberikan user, untuk pertanyaan ganjil dan 5 dikurangi dengan respon yang diberikan user untuk poertanyaan genap. Setelah itu skor SUS didapat dengan cara mengkalikan total skor dengan 2.5. Skor akhir SUS akan berada pada kisaran 0-100. Sesuai sengan gambar 6.16 berdasarkan skor akhir SUS tersebut akan bisa diketahui seberapa tinggi tingkat usability dan akseptabilitas (acceptable) desain sistem aplikasi yang dikembangkan. Penilaiannya berdasarkan tiga kategori yaitu Not Acceptable dengan rentang skor SUS 0 - 50.9, Marginal 51 - 70.9, dan Acceptable 71 - 100. Responden diminta menjawab semua butir pertanyaan yang diisi setelah pengguna selesai menggunakan sistem secara keseluruhan.

Tabel 6.22 Keterangan Skor Skala Likert Tiap Pernyataan SUS

No	Keterangan
1	Sangat Tidak Setuju
2	Tidak Setuju
3	Netral
4	Setuju
5	Sangat Setuju



Gambar 6.16 Rating dan skala konversi skor rata-rata SUS

6.4.1 Prosedur Pengujian Usabilitas

Pelaksanaan pengujian usabilitas kali ini dilakukan aplikasi yang coba digunakan oleh pelatih untuk menggunakan aplikai Badstics untuk menentukan karakteristik pemain bulutangkis ini. Responden pada pengujian kali ini diminta untuk menggunakan aplikasi, mengerjakan *task* dan diberikan lembar angket *SEQ* dan *SUS*. Responded pertama-tama diberikan penjelasan mengenai cara menggunakan aplikasi Badstics untuk menentukan karakteristik pemain bulutangkis. Selanjutnya responded mencoba aplikasi dengan mengikuti *task* yang ada seperti pada tabel 6.24, setelah melakukan langkah-langkah sesuai skenario responden diminta untuk memilih pilihan pada angket yang tersedia dari pengujian *SUS.* Responden kali ini merupakan pelatih bulutangkis berjumlah 5 orang.

Tabel 6.23 Daftar Task SEQ

No.	Nama Task	Task
1	B01	Masukkan aksi pemain berdasarkan pemain bulutangkis di lapangan
2	B02	Kurangi jumlah aksi pemain yaitu pada aksi “SERIVICE PENDEK”
3	B03	Tambahkan jumlah aksi pemain yaitu pada aksi “SERIVICE PENDEK”
4	B04	Tambahkan jumlah error aksi pemain yaitu pada aksi apapun
5	B05	Kurangi jumlah error aksi pemain yaitu pada aksi apapun

Tabel 6.24 Daftar Pernyataan SUS

No.	Pertanyaan
1	Saya dapat mengoperasikan aplikasi ini dengan mudah
2	Cara memasukkan aksi pemain ke aplikasi cukup sulit
3	Informasi yang diberikan sangat saya butuhkan sebagai pelatih

4	Saya merasa perlu pendampingan dari orang yg paham untuk dapat menggunakan aplikasi ini
5	Bahasa dan instruksi yang diberikan mudah dipahami
6	Saya merasa huruf pada aplikasi ini terlalu kecil
7	Saya suka dengan warna yang digunakan
8	Saya merasa tombol terlalu berdekatan sehingga akan sering salah menekan tombol
9	Jika saya akan mengevaluasi pemain bulutangkis saya, saya akan menggunakan aplikasi ini
10	Saya tidak akan menggunakan aplikasi ini

6.4.2 Analisis Data dan Hasil Pengujian Usabilitas

Setelah prosedur pengujian dilakukan dilakukan perhitungan data dan analisis diperoleh hasil dari responden sebagai berikut yaitu yang ditampilkan pada tabel 6.31

Tabel 6.25 Hasil Evaluasi Antarmuka Pengguna Aspek Efektivitas

No	Nama *Task*	Tugas yang harus dikerjalan	Responden1	Responden2	Responden3	Responden4	Responden5
			Berhasil/Gagal	Berhasil/Gagal	Berhasil/Gagal	Berhasil/Gagal	Berhasil/Gagal
1	B01	Masukkan aksi pemain berdasarkan pemain bulutangkis di lapangan	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
2	B02	Kurangi jumlah aksi pemain yaitu pada aksi “SERIVICE PENDEK”	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
3	B03	Tambahkan jumlah aksi pemain yaitu pada aksi “SERIVICE PENDEK”	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
4	B04	Tambahkan jumlah error aksi pemain yaitu pada aksi apapun	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
5	B05	Kurangi jumlah error aksi pemain yaitu pada aksi apapun	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil

Tabel 6.26 Hasil Evaluasi Antarmuka Pengguna Aspek Efektivitas

	Task1	*Task2*	*Task3*	*Task4*	*Task5*	*Rata-Rata Presentase*
Jumlah *Task* Berhasil	5	5	5	5	5	100%
Presentase Keberhasilan	100%	100%	100%	100%	100%	
Jumlah *Task* Gagal	0	0	0	0	0	0%
Presentase Kegagalan	0%	0%	0%	0%	0%	

Sesuai dengan hasil yang terdapat pada Tabel 6.27 seluruh responden dapat menyelesaikan semua *task* sehingga rata-rata presentase yang dihasilkan yaitu 100% dan presentase kegagalan yaitu 0% dengan begitu dapat diketahui bahwa responden sudah dapat menggunakan aplikasi dengan baik sesuai dengan *task* yang ditentukan.

Tabel 6.27 Hasil Kuisioner Skor tiap pertanyaan SUS

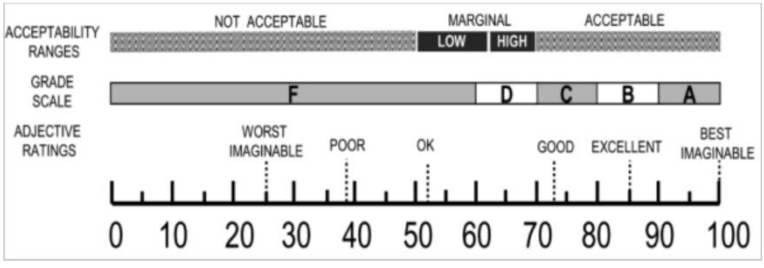
Nama / **No**	**Responden1**	**Responden2**	**Responden3**	**Responden4**	**Responden5**
1	4	3	4	4	4
2	0	1	1	0	1
3	4	4	3	3	3
4	1	1	0	1	0
5	3	4	4	3	4
6	1	0	1	0	1
7	4	3	3	4	4
8	0	1	2	1	0
9	3	4	4	2	4
10	0	1	1	1	0

Dalam proses analisis median akan digunakan mengetahui tendensi sentral pada variabel pengujian yang bertipe *categorical* (Tarquinio, 2015). Pada skor SUS dianalisis menggunakan kisaran rating atau penetimaan

Tabel 6.28 Hasil Konversi Skor tiap pertanyaan SUS

Nama No	Responden1	Responden2	Responden3	Responden4	Responden5	
1	3	2	3	3	3	
2	5	4	4	5	4	
3	3	3	2	2	2	
4	4	4	5	4	5	
5	2	3	3	2	3	
6	4	5	4	5	4	
7	3	2	2	3	3	
8	5	4	3	4	5	
9	2	3	3	1	3	
10	5	4	4	4	5	
Jumlah	50	46	38	60	67	Rata-Rata
Dikali 2.5	90	85	82.5	82.5	92.5	86.5

Berdasarkan hasil yang didapatkan dari lima orang responden, seperti yang terlamipir pada Lampiran A, skor likert yang diperoleh akan dilakukan analisis Hasil dari pengujian kali ini usability diukur menggunakan pengujian post study dengan metode System Usability Scale adalah 86.5 yang berarti masuk kedalam *acceptability rating* yaitu diperoleh *acceptable* dan *adjective ratings* yaitu excelent sehingga aplikasi sudah sesuai dengan kebutuhan pengguna.



6.5 Analisis Hasil Pengujian

Setelah semua pengujian telah selesai dilakukan, hasil dari semua pengujian dianalisis untuk dapat menarik kesimpulan bahwa hasil dari semua

pengujian yang telah selesai dilakukan telah memenuhi kebutuhan yang telah dijelaskan pada analsis kebutuhan.

6.5.1 Pengujian Unit

Dari semua pengujian unit yang dilakukan yaitu diantaranya `actionButtonDidTap()`, `compareNaiveBayes(Int, Int, Int)`, `getCategory(Int)`, dihasilkan beberapa *basis path* atau jalur dasar yaitu semua kemungkinan yang dapat terjadi tersebut sudah diuji dan mendapatkan hasil yang valid pada setiap pengujiannya sehingga semua pengujian sudah sesuai dengan kebutuhan yang telah ditentukan pada analsisis kebutuhan.

6.5.2 Pengujian Integrasi

Dalam pengujian integrasi fungsi `playerDidAction()` terdapat 4 fungsi yang terintegrasi dengan fungsi utama yaitu fungsi `playerDidAction()` yaitu fungsi `addActionCountTest()`, `addErrorCountPlayerTest()`, `addPointTest()` dan `getStatetest()` fungsi tersebut merupakan fungsi yang dibuat untuk menguji *expected result* sudah sesuai dengan kebutuhan. Selanjutnya fungsi setelah diimplementasikan pengujian dilakukan perbandingan untuk mengetahui apakah *expected result* sudah sesuai dengan fungsi yang diimplementasikan.

6.5.3 Pengujian Validasi

Dalam pengujian validasi semua fungsi telah diuji berdasarkan pengujian yang dilakukan dengan mencoba aplikasi yang dijalankan dan menjalankan aplikasi sesuai dengan prosedur pengujian yang sudah ditentukan, dari semua pengujian validasi tidak terdapat fungsi yang tidak valid yang berarti bahwa semua fungsi telah sesuai dengan kebutuhan yang telah ditentukan pada analisis kebutuhan.

6.5.4 Pengujian Usabilitas

Pengujian usabilitas dilakukan dengan menguji aplikasi kepada responden, yaitu pelatih yang mencoba aplikasi Badstics setelah itu responded diberikan angket untuk diisi yaitu berupa kuisioner untuk menentukan seberapa baikkah aplikasi sudah sesuai dengan kebutuhan yang telah ditentukan pada analisis kebutuhan. Hasil dari pengujian kali ini usability diukur menggunakan pengujian post study dengan metode System Usability Scale adalah 86.5 yang berarti masuk kedalam kategori *acceptable* sehingga aplikasi sudah sesuai dengan kebutuhan pengguna.

BAB 7
PENUTUP

7.1 Kesimpulan

Berdasarkan hasil analisis kebutuhan, perancangan, implementasi, dan pengujian yang telah dilakukan, maka dapat disimpulkan sebagai berikut:

1. Berdasarkan hasil analisis kebutuhan yang dilakukan, aplikasi Badstics untuk menentukan karakteristik pemain bulutangkis ini mempunyai 5 kebutuhan fungsional dan 1 kebutuhan non fungsional yang akan membantu proses penentuan karakteristik pemain bulutangkis. Penentuan kebutuhan fungsional diperoleh berdasarkan permasalahan yang ada yaitu dibutuhkannya waktu yang lama untuk menentukan karakteristik pemain bulu tangkis. Seleain itu juga diperoleh beberapa pemodelan kebutuhan yang digunakan supaya memudahkan pengembang dalam memahami sistem, seperti use case diagram, class diagram dan sequence diagram.

2. Setelah perancangan yang dilakukan berdasarkan analisis kebutuhan diperoleh perancangan data, perancangan komponen, perancangan antarmuka. Dalam perancangan arsitektur sistem terdapat rancangan class diagram yang dijelaskan dengan rinci pada setiap operasi dan atributnya. Pada perancangan data diperoleh rancangan database yang berupa *Entity Relational Diagram (ERD)*. Pada perancangan komponen terdapat algoritma-algoritma yang akan digunakan dan diimplementasikan pada sistem. Pada perancangan antarmuka berisi *mock-up* atau gambaran dari sistem yang akan dibangun. Pada tahap Implementasi sistem diperoleh *Physical Data Model (PDM)*, implementasi kode program yang menjelaskan bagaimana naïve bayes tersebut dapat digunakan dengan method `compareNaiveBayes()` dan juga method seperti `addActionAcount()` implementasi antarmuka pun menghasilkan dua *screen*, yaitu input statistics dan player actions.

3. Penerapan algoritme Naive Bayes digunakan untuk menentukan karakteristik pemain, penerapan algoritme tersebut digunakan pada saat implementasi program yang dilakukan setelah analisis kebutuhan. Algoritme Naive Bayes yang digunakan telah menggunakan data training yang diperoleh dari PBSI: Jawa Timur dengan jumlah 43 data, data training tersebut yang digunakan dalam menentukan karakteristik pemain bulutangkis. Data training tersebut lalu digunakan sebagai atribut dari perhitungan dari algoritme yang di gabungkan dengan aksi pemain di lapangan yang telah diinputkan oleh pelatih.

4. Dengan hasil pengujian usability mengenai kemudahan penggunaan aplikasi Badstics untuk menentukan karakteristik pemain bulutangkis yaitu sebesar 86.5 dapat disimpulkan bahwa kemudahan dalam menggunakan aplikasi ini sudah terpenuhi.

7.2 Saran

Saran yang diberikan untuk pengembangan Badstics sebagai aplikasi untuk menentukan karakteristik pemain bulutangkis ini selanjutnya antara lain:

1. Dapat menyimpan *history* statistik pemain setiap kali bermain.
2. Data pemain dapat diakses dari berbagai tempat sehingga dapat digunakan untuk penelitian lain.
3. Sistem dapat memberikan grafik mengenai performa dari pemain sehingga dapat diprediksi bagaimana performa dari pemain kedepannya.

DAFTAR PUSTAKA

Apple 2017. “Swift”. Diakses pada 20 Oktober 2017. https://developer.apple.com/swift/.

Apple. 2017. “Event Handling iPhone OS”. Diakses pada 20 Oktober 2017. https://developer.apple.com/library/content/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/index.html#//apple_ref/doc/uid/TP40009541.

Apple. 2017. “The App Life Cycle”. Diakses pada 20 Oktober 2017. https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

Apple. 2015. “Model View Controller”. Diakses pada 20 Oktober 2017. https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

Arikunto, S. (2009). *Prosedur Penelitian Suatu Pendekatan Praktik. Edisi Revisi 6. Jakarta : Rineka Cipta*.

Asfiyani. 2016. Analisis Pembinaan Cabang Olahraga Bulutangkis Pada Klub Persatuan Bulutangkis Tri Darma Di Kabupaten Tuban. Fakultas Ilmu Keolahragaan, Universitas Negeri Surabaya

Bindal, Nancy and Mehta, Aanchal., 2015. *Survey On Software Development Processing Models.* International Journal of Exploring Emerging Trends in Engineering (IJEETE)

Brooke John. 2011. Measuring Usability With The System Usability Scale (SUS). http://www.measuringu.com/sus.php. Diakses pada tanggal 22 Desember 2017.

Caytiles, Ronnie D. and Lee, Sunguk., 2014. *A Review of an MVC Framework based Software Development*

Eisenman, Bonni. 2016. *Learning React Native,* United States of America: O’Reilly Media, Inc

Bangor, A., Kortum, P., & Miller, J. 2009. Determining What Individual SUS Scores
Mean : Adding an Adjective Rating Scale. Journal of Usability Studies

Ballou, Ralph B. 1998. *Badminton for Beginners,* 2nd ed. Colorado: Morton Publishing Co.

Grice, Tony. 1994. *Badminton for the College Student,* 4th ed. Boston, Massachusetts: American Press.

PB Djarum 2017. “Living Data” Diakses Pada 18 Agustus 2017.

https://www.pbdjarum.org/livingdata.

Pressman, R.S. 2010, Software Engineering : a practitioner's approach, McGraw-Hill, New York

Rumbaugh, James., 2005. *The Unified Modeling Language Reference Manual.* 2nd ed. Boston:Addison-Wesley

Sugiharto. 2008. Tipe dan Karakter Pemain Bulutangkis, Fakultas Ilmu Keolahragaan Universitas Negri Semarang.

Surya, Sumpeno. 2009 Klasifikasi Emosi Untuk Teks Bahasa Indonesia Menggunakan Metode Naïve Bayes, Seminar Nasional Pascasarjana, Institut Teknologi Sepuluh Nopember.

Tan. 2006. Introduction to Data Mining. United State of America, PearsonEducation, Inc.

Wattanasin, Charuen. 2000. *Badminton a Simple Way,* Badminton Booklet. London: The IBF.

Whitten Bentley. 2007. System analysis & Design Methods, McGraw-Hill/Irwin, New York

LAMPIRAN A HASIL WAWANCARA

A.1 Pelatih Bulutangkis

DAFTAR PERTANYAAN WAWANCARA

Daftar pertanyaan dalam wawancara ini untuk menjawab rumusan masalah yang dibuat oleh penulis dalam **"Pengembangan Aplikasi Badstics untuk Menentukan Karateristik Pemain Bulutangkis"** berikut ini adalah pertanyaan yang diberikan kepada pelatih bulutangkis yaitu bapak Purnomo :

1. Apa yang selama ini dilakukan oleh pelatih bulutangkis dalam mengevaluasi pemainnya?
2. Bagaimana proses mengevaluasinya?
3. Berapa lama waktu yang dibutuhakan untuk mengevaluasi pemain di lapangan?
4. Apakah menurut bapak metode untuk sudah sesuai dengan kebutuhan pelatih selama ini?
5. Berapa banyak aksi pemain di lapangan yang harus disimpan dalam mengevaluasi pemain?
6. Apakah kebutuhan yang selama ini belum ada dalam metode untuk mengevaluasi pemain bulutangkis?
7. Apakah karakteristik pemain penting untuk diketahui ketika permainan sudah selesai dilakukan?
8. Apa saja aksi pemain yang harus disimpan dalam aplikasi?

HASIL INTERVIEW

Tanggal : 28 Oktober 2017

Waktu : 14:00 – 16:00

Narasumber : Bapak Purnomo

Jabatan : Pelatih Bulutangkis PBSI Jawa Timur

JAWABAN

1. Selama ini pelatih mengevaluasi pemain dengan cara menggunaakan rekaman video yang dilakukan dengan merekam video pertandingan bulutangkis yang pemainnya akan dilakukan evaluasi, setelah itu video ajan di *replay* atau diputar ulang untuk mengetahui apa saja aksi yang dilakukan oleh pemain, jadi yang ingin mengevaluasi haru mem-*pause* dan *play* video hingga pertandingan selesai.

2. Setelah mendapatkan data dari pemain yang diambil dari *video* tadi data akan disimpan dalam bentuk excel setelah itu data akan dianalisis oleh pelatih apa yang harus dilakukan oleh pemain di pertandingan selanjutnya.
3. Waktu yang dibutuhkan dalam mengevaluasi pemain dengan metode ini membutuhkan waktu 3 – 4 jam.
4. Sudah sesuai akan tetapi butuh yang lebih mudah dari metode yang ada.
5. Aksi yang disimpan dalam metode ini ada sebanyak 30 aksi.
6. Dalam metode ini belum ada data aksi *error* pemain, yaitu kesalahan yang dilakukan oleh pemain kita ketika dikalahkan oleh lawan.
7. Iya perlu karena untuk pertandingan kedepannya perlu diketahui sehingga dapat dievaluasi dengan lebih baik lagi.
8. Aksi pemain sesuai dengan metode yang digunakan adalah Service pendek forehand, Service panjang forehand, Smash lurus forehand, Smash silang forehand, Drive lurus forehand, Drive silang forehand, Dropshot lurus forehand, Dropshot silang forehand, Net lurus forehand, Net silang forehand, Lob lurus forehand, Lob silang forehand, Chop lurus forehand, Chop silang forehand, Service pendek backhand, Service panjang backhand, Smash lurus backhand, Smash silang backhand, Drive lurus backhand, Drive silang backhand, Dropshot lurus backhand, Dropshot silang backhand, Net lurus backhand, Net silang backhand, Lob lurus backhand, Lob silang backhand , Chop lurus backhand, Chop silang backhand.

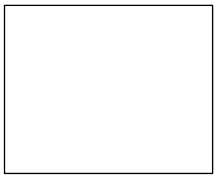
Penulis	Narasumber
Robihamanto	Purnomo

LAMPIRAN B KUISIONER PENGUJIAN USABILITAS

Bagian 1. Background Questionnaire

Berikan uraian atau pilihlah jawaban dari pertanyaan-pertanyaan berikut.

1. Nama :
2. Gender : ○ Laki-laki ○ Perempuan
3. Usia : _____ tahun
4. No. Hp :
5. Email : _____



6. Apakah Anda memiliki dan atau menggunakan *smartphone/tablet* dalam kegiatan sehari-hari?
   ○ Ya ○ Tidak

7. Sistem operasi apakah yang digunakan oleh *smartphone/tablet* Anda?
   ○ iOS
   ○ Android
   ○ Windows
   ○ Lainnya: _____

8. Aktivitas apa saja yang Anda lakukan dengan *smartphone/tablet* Anda?
   Urutkan layanan/fungsi berikut berdasarkan frekuensi pemakaian dengan menggunakan angka 1-4 (angka 1 untuk layanan/fungsi yang paling sering digunakan, angka 4 untuk layanan/fungsi yang paling jarang digunakan)
   _____ komunikasi/social media
   _____ hiburan (games, video, dll)
   _____ information/news (web browsing, dll)
   _____ lainnya (kamera, GPS)

9. Seberapa sering Anda berpergian dari rumah atau kantor untuk tujuan melatih/mengevaluasi pemain?
   ○ Kurang dari satu kali dalam seminggu
   ○ Sekali atau lebih dari sekali dalam seminggu
   ○ Setiap hari
   ○ Lainnya: _____

Bagian 2. Pengoperasian Aplikasi (1)

Pada bagian ini Anda diminta mengoperasikan aplikasi berdasarkan panduan yang telah disiapkan.

Anda juga diminta untuk mengungkapkan secara langsung (bukan bertanya) segala hal
yang Anda alami dan rasakan terkait pengoperasian aplikasi tersebut kepada moderator.

Task 1: Memulai aplikasi “Badstics”

Task 2: Mulai permainan bulutangkis

Task 3: Masukkan aksi pemain berdasarkan pemain bulutangkis di lapangan

Task 4: Kurangi jumlah aksi pemain yaitu pada aksi “SERIVICE PENDEK”

Task 5: Tambahkan jumlah aksi pemain yaitu pada aksi “SERIVICE PENDEK”

Task 6: Tambahkan jumlah error aksi pemain yaitu pada aksi apapun

Task 7: Kurangi jumlah error aksi pemain yaitu pada aksi apapun

Task 8: Tambahkan poin pemain 1

SELESAI

Bagian 3. Kuesioner (1) *SEQ

		Sangat tidak setuju	Tidak Setuju	Netral	Setuju	Sangat setuju
1	Menurut saya tampilan aplikasi “Badstics” menarik (bagus) dilihat?					
		1	2	3	4	5
2	Menurut saya tampilan awal pada aplikasi ini menarik (bagus) dilihat?					
		1	2	3	4	5
3	Menurut saya menu-menu pada aplikasi ini mudah untuk dipahami?					
		1	2	3	4	5
4	Menurut saya tulisan teks yang digunakan mudah dan jelas?					
		1	2	3	4	5
5	Menurut saya menu pada aplikasi yang ditampilkan dapat dipahami dengan mudah?					
		1	2	3	4	5
6	Menurut saya menu yang di klik dapat menampilkan dengan cepat?					
		1	2	3	4	5
7	Menurut saya tata letak desain interface dapat mudah diingat?					
		1	2	3	4	5
8	Menurut saya fitur memasukkan aksi pemain mudah untuk digunakan ?					
		1	2	3	4	5
9	Menurut saya fitur menambahkan dan mengurangi aksi pemain					

	mudah untuk digunakan ?					
		1	2	3	4	5
10	Menurut saya selama menggunakan aplikasi ini merasa sangat nyaman?					
		1	2	3	4	5

Bagian 4. Pengoperasian Aplikasi (2)

Pada bagian ini Anda diminta mengoperasikan aplikasi berdasarkan panduan yang telah disiapkan.

Anda juga diminta untuk mengungkapkan secara langsung (bukan bertanya) segala hal
yang Anda alami dan rasakan terkait pengoperasian aplikasi tersebut kepada moderator.

Task 1: Memulai aplikasi “Badstics”

Task 2: Mulai permainan bulutangkis

Task 3: Masukkan aksi pemain berdasarkan pemain bulutangkis di lapangan

Task 4: Kurangi jumlah aksi pemain yaitu pada aksi “SERIVICE PENDEK”

Task 5: Tambahkan jumlah aksi pemain yaitu pada aksi “SERIVICE PENDEK”

Task 6: Tambahkan jumlah error aksi pemain yaitu pada aksi apapun

Task 7: Kurangi jumlah error aksi pemain yaitu pada aksi apapun

Task 8: Tambahkan poin pemain 1

SELESAI

Bagian5. Kuesioner (2) *SUS-modified

		Sangat tidak setuju	Tidak Setuju	Netral	Setuju	Sangat setuju
1	Saya dapat mengoperasikan aplikasi ini dengan mudah					
		1	2	3	4	5
2	Cara memasukkan aksi pemain ke aplikasi cukup sulit					
		1	2	3	4	5
3	Informasi yang diberikan sangat saya butuhkan sebagai pelatih					
		1	2	3	4	5
4	Saya merasa perlu pendampingan dari orang yg paham untuk dapat menggunakan aplikasi ini					
		1	2	3	4	5
5	Bahasa dan instruksi yang diberikan mudah dipahami					
		1	2	3	4	5
6	Saya merasa huruf pada aplikasi ini terlalu kecil					
		1	2	3	4	5
7	Saya suka dengan warna yang digunakan					
		1	2	3	4	5
8	Saya merasa tombol terlalu berdekatan sehingga akan sering salah menekan tombol					
		1	2	3	4	5
9	Jika saya akan mengevaluasi pemain bulutangkis saya, saya akan menggunakan aplikasi ini					
		1	2	3	4	5
10	Saya tidak akan menggunakan aplikasi ini					
		1	2	3	4	5