RANCANG BANGUN APLIKASI PERANGKAT BERGERAK PENANDA DINI BAHAYA BANJIR MENGGUNAKAN *WEB SERVICE* BERBASIS SOAP

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar sarjana komputer

Disusun Oleh :
ADIEN FAISHOL HABIB
NIM. 0910680065

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
PROGRAM STUDI TEKNIK INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2013

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI PERANGKAT BERGERAK PENANDA DINI BAHAYA BANJIR MENGGUNAKAN *WEB SERVICE* BERBASIS SOAP

SKRIPSI

Untuk Memenuhi Sebagian Persyaratan Mencapai Gelar Sarjana Komputer

Disusun Oleh :

ADIEN FAISHOL HABIB

NIM. 0910680065

Telah diperiksa dan disetujui oleh:

Pembimbing I	Pembimbing II,
Arief Andy Soebroto, ST., M.Kom.	**Denny Sagita Rusdianto, S.Kom., M.Kom.**
NIP. 19720425 199903 1 002	**NIK. 851124 06 1 1 0250**

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI PERANGKAT BERGERAK PENANDA DINI BAHAYA BANJIR MENGGUNAKAN *WEB SERVICE* BERBASIS SOAP

SKRIPSI

Untuk Memenuhi Sebagian Persyaratan Mencapai Gelar Sarjana Komputer

Disusun oleh :

ADIEN FAISHOL HABIB
NIM. 0910680065

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 13 Desember 2013

Penguji I	Penguji II
Ismiarta Aknuranda, ST., M.Sc., Ph.D.	**Fajar Pradana, S.ST., M.Eng.**
NIK. 740719 06 1 1 0079	**NIK. 850905 16 1 1 0371**

Penguji III

Yusi Tyroni Mursityo, S.Kom., MS.

NIP. 19800228 200604 1 001

Mengetahui
Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.

NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

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 dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan 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, 13 Desember 2013

Mahasiswa,

Adien Faishol Habib

NIM. 0910680065

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa karena hanya dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi dengan judul“Rancang Bangun Aplikasi Perangkat Bergerak Penanda Dini Bahaya Banjir Menggunakan *Web Service* Berbasis SOAP”.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama penulisan skripsi, diantaranya:

1. Ibunda Hj. Husnawiyah, Almarhum Ayahanda Wagito, Kakak H. Samsul Umam, Kakak H. Haiburrohib Muslim, Kakak H. Nur Hakiki Muslim, Kakak H. M. Nur Kholis Muslim, S.H., Kakak Faridatus Syuhadak, M.H.I., Kakak Habiburrahman, S.Pd.I., Kakak Khoirul Badi’ah, S.E, Kakak Hapsa Hidayati, S.Pt. dan seluruh keluaga besar atas segala nasehat, kasih sayang, perhatian dan kesabarannya dalam mendidik penulis serta memberikan doa dan semangat secara terus-menerus demi terselesaikannya skripsi ini.
2. Bapak Ir. Sutrisno, M.T., Bapak Ir. Heru Nurwasito, M.Kom., Bapak Himawat Aryadita, S.T., M.Sc., dan Bapak Eddy Santoso, S.Kom. selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Drs. Marji, M.T. dan Bapak Issa Arwani, S.Kom., M.Sc. selaku Ketua dan Sekretaris Program Studi Teknik Informatika Universitas Brawijaya.
4. Bapk Arief Andy Soebroto, S.T.,M.Kom. selaku dosen pembimbing I yang telah memberikan ilmu dan saran untuk menyelesaikan skripsi ini.
5. Bapak Denny Sagita Rusdianto, S.Kom., M.Kom. selaku dosen pembimbing II yang juga memberikan ilmu dan saran untuk menyelesaikan skripsi ini.
6. Bapak Suprapto, S.T., M.T. selaku dosen pembimbing akademik yang telah memberikan pengarahan selama penulis menempuh pendidikan di Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
7. Bapak Bayu Priyambadha, S.Kom., Bapak Eriq M. Adams J., S.T., M.Kom, Bapak Widhy Hayuhardhika Nugraha Putra, S.Kom yang telah banyak

membantu dan memberi inspirasi kepada penulis dalam penulisan skripsi ini.

8. Seluruh Dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
9. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
10. Vinishya Amin yang selalu menemani, membantu, memberikan semangat serta mendoakan penulis dalam menyelesaikan skripsi ini.
11. Rina Nurhanifah, S.I.Kom yang telah memberikan bantuan finansial kepada penulis demi terselesaikannya skripsi ini.
12. Sahabat-sahabat penulis TIF C 2009 yang telah memberikan doa, bantuan serta semangat kepada penulis selama menempuh studi di Teknik Informatika Universitas Brawijaya.
13. Sahabat-sahabat penulis Angkatan 2009 Teknik Informatika atas segala bantuannya selama menempuh studi di Teknik Informatika Universitas Brawijaya.
14. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun yang tidak langsung demi terselesaikannya skripsi ini.

Hanya doa yang bisa penulis berikan semoga Allah SWT memberikan pahala serta balasan kebaikan yang berlipat. Penulis menyadari bahwa skripsi ini masih banyak kekurangan. Oleh karena itu, kritik dan saran yang bersifat membangun sangat diharapkan untuk menyempurnakan skripsi ini. Semoga skripsi ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya.

Malang, 13 Desember 2013

Penulis

ABSTRAK

Adien Faishol Habib. 2013. : Rancang Bangun Aplikasi Perangkat Bergerak Penanda Dini Bahaya Banjir Menggunakan *Web Service* Berbasis SOAP. Skripsi Program Studi Informatika/Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.
Dosen Pembimbing : Arief Andy Soebroto, S.T., M.Kom. dan Denny Sagita Rusdianto, S.Kom., M.Kom.

Bencana banjir masih terjadi secara teratur dan terus-menerus di Indonesia. Bencana banjir dapat mengakibatkan dampak yang merusak pada bidang ekonomi, sosial dan lingkungan. Penelitian ini menghasilkan sebuah aplikasi sistem pendukung keputusan deteksi dini bahaya banjir berbasis perangkat bergerak dengan menggunakan teknologi *web service* dari penelitian yang dibuat sebelumnya. Hasil deteksi setiap data diperoleh dari sebuah server yang dihubungkan dengan sebuah *web service* berbasis *Simple Object Access Protocol* (SOAP). Data akan dikirim kepada *client* berupa data *eXtensible Markup Language* (XML) kemudian akan diolah menjadi sebuah informasi pendukung keputusan di dalam perangkat bergerak. Keuntungan dari sistem ini yaitu dirancang untuk penyebaran informasi peringatan dini bahaya banjir yang dilakukan secara cepat karena dilakukan melalui internet. Pengujian perangkat lunak dari penelitian ini menggunakan pengujian validasi dan *user acceptance testing*. Hasil pengujian validasi menunjukkan perangkat lunak sudah memenuhi spesifikasi kebutuhan dengan prosentase 100%. Berdasarkan dari hasil *user acceptance testing*, 66% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori bagus, 6% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori cukup, 27% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori sangat bagus dan 1% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori kurang.

Kata Kunci : SOAP, XML, Android, *Web Service, Flood Early Warning System*

ABSTRACT

Adien Faishol Habib. 2013. : Development of Mobile Application for Flood Early Warning System Using Web Service Based SOAP. Skripsi Program Studi Informatika / Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Supervisors : Arief Andy Soebroto, S.T., M.Kom. and Denny Sagita Rusdianto, S.Kom., M.Kom

Flood disasters still occur regularly and continuously in Indonesia. Flood disasters have destructive impact to economics, social and environment sector. This study resulted in amobile-based decision support system application of early detection of flooding with use of web service technology that were made earlier. Detection results of each data is processed by a server that connected a web service technology based Simple Object Access Protocol (SOAP). Data will be sent to client in eXtensible Markup Language (XML) data then it will be processed to become a decision support information in mobile phone. The advantage of this system is designed for dissemination of early flood warning information that can be done quickly because it is done through internet. The software testing of this research is using validation testing and user acceptance testing. The result of validation testing show that application have conformed requirements specification with a percentage of 100%. Based on the result of user acceptance testing, 66% of respondents gives a good category in the assessment of the application, 6% of respondents gives a enough category in the assessment of the application, 27% of respondents gives a very good category in the assessment of the application, 1% of respondents gives a deficient category in the assessment of the application.

***Keywords* :** SOAP, XML, Android, *Web Service, Flood Early Warning System*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia merupakan salah satu negara yang memiliki intensitas curah hujan yang cukup tinggi di benua Asia. Peningkatan intensitas curah hujan yang berlebihan akan menimbulkan sebuah fenomena alam seperti bencana banjir. Salah satu penyebab terjadinya banjir adalah luapan air sungai yang tidak dapat menampung debit air hujan yang masuk ke dalam aliran air sungai. Bencana banjir dapat mengakibatkan dampak yang merusak pada bidang ekonomi, sosial dan lingkungan. Salah satu dampak yang paling parah dari bencana banjir yaitu kematian. Dalam rangka upaya mengurangi dampak yang berlebihan dari bencana banjir maka perlu adanya sebuah sistem yang memantau peningkatan tinggi air sungai. Sistem pemantauan yang diperlukan adalah sebagai sistem pendukung keputusan yang dapat memberikan peringatan atau mendeteksi terjadinya bahaya banjir. Berdasarkan dari penelitian sebelumnya yaitu *Flood Early Warning System* (FEWS), sistem tersebut telah menyediakan berbagai informasi mengenai bahaya banjir yang berperan sebagai sistem pendukung keputusan terhadap terjadinya bahaya banjir. FEWS juga menyediakan sebuah teknologi *web service* yang dapat digunakan untuk melakukan pertukaran data tanpa terikat pada satu platform. Teknologi *web service* tersebut memungkinkan FEWS dapat dikembangkan ke dalam sistem operasi perangkat bergerak sehingga manfaat yang terdapat di dalam FEWS dapat dimanfaatkan oleh berbagai kalangan masyarakat.

FEWS merupakan sebuah *prototype* sistem pendukung keputusan penanda dini bahaya banjir secara *real time* menggunakan sebuah *datalogger* berbasis *Global System for Mobile Communications* (GSM). FEWS memantau kondisi DAS (Daerah Aliran Sungai) disepanjang hulu dan hilir. Proses pemantauan dilakukan dengan memasang sebuah perangkat keras yang digunakan sebagai stasiun pemantauan kondisi DAS. Di setiap stasiun pemantauan kondisi DAS memiliki beberapa modul sensor yang terhubung dengan *datalogger* pada GSM. Modul sensor akan memantau tiga parameter yaitu tingkat ketinggian air sungai, suhu dan kelembaban yang dilakukan secara terus-menerus. Hasil deteksi setiap

data dari modul sensor diproses oleh *datalogger* kemudian dikirim ke *server* stasiun pusat melalui SMS *Gateway*. Hasil penerimaan dan pengolahan data akan ditampilkan ke dalam sebuah aplikasi *web* dengan beberapa fitur yaitu status pemantauan banjir, pemantauan tingkat air, peta wilayah, tanya jawab, manajemen berita, pemantauan grafik, dan dokumentasi pencetakan *log* kondisi air [SSE-13].

Berdasarkan paparan informasi tersebut, penulis mengambil judul skripsi *"Rancang Bangun Aplikasi Perangkat Bergerak Penanda Dini Bahaya Banjir Menggunakan Web Service Berbasis SOAP"*. Sistem informasi berbasis perangkat bergerak yang akan dibuat ini difokuskan untuk mengimplementasikan *web service* ke dalam sistem operasi android dari aplikasi *web Flood Early Warning System.*

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka dapat dirumuskan permasalahan pada skripsi ini yaitu sebagai berikut:

1. Bagaimana menganalisa kebutuhan pembuatan aplikasi *mobile* di dalam sistem operasi android dengan menggunakan teknologi *web service* berbasis SOAP.
2. Bagaimana merancang dan mengimplementasi sistem informasi penanda dini bahaya banjir di dalam sistem operasi android.
3. Bagaimana skenario pengujian sistem informasi penanda dini bahaya banjir di dalam sistem operasi android.

1.3 Batasan Masalah

Agar permasalahan yang dirumuskan dapat lebih terfokus, maka pada penelitian ini dibatasi dalam hal:

1. Pengujian sistem informasi penanda dini bahaya banjir dilakukan di dalam sistem operasi android versi 2.3.
2. Aplikasi ini akan mengimplementasi sebuah *web service* dari *Flood Early Warning System.*
3. KSOAP2 versi 2.5.8 merupakan *library* yang digunakan untuk mengimplementasikan *web service* di dalam sistem operasi android.

4. Proses validitas pengguna ditentukan oleh administrator *Flood Early Warning System*.
5. AChartEngine versi 1.0.0 merupakan *library* yang digunakan untuk menampilkan informasi dalam bentuk grafik.

1.4 Tujuan

Menganalisa kebutuhan, merancang, mengimplementasi dan melakukan pengujian pada sistem informasi penanda dini bahaya banjir di dalam sistem operasi android untuk memberikan gambaran pengembangan teknologi *web service* berbasis SOAP.

1.5 Manfaat

Penulisan skripsi ini diharapkan mempunyai manfaat yang baik dan berguna bagi pembaca dan penulis. Adapun manfaat yang diharapkan adalah sebagai berikut:

- Bagi Penulis
  1. Mendapatkan pengetahuan dan wawasan terkait analisa kebutuhan untuk pengembangan sistem informasi.
  2. Mendapatkan pengetahuan dan wawasan terkait perancangan sistem informasi berbasis perangkat bergerak di dalam sistem operasi android.
- Bagi pembaca/pengguna
  1. Mendapatkan wawasan tentang proses implementasi sistem informasi berbasis perangkat bergerak pada sistem operasi Android.
  2. Sebagai sarana publikasi informasi mengenai keadaan sungai dan ketinggian air sungai suatu daerah tertentu.

1.6 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat, metodologi pembahasan, dan sistematika penulisan.

BAB II Tinjauan Pustaka

Menguraikan tentang kajian pustaka dan dasar teori yang mendasari pembuatan *mobile application* pada sistem operasi Android.

BAB III Metodologi Penelitian

Membahas metode yang digunakan dalam penelitian yang terdiri dari studi literatur, perancangan sistem perangkat lunak, implementasi sistem perangkat lunak, pengujian dan analisis, serta penulisan laporan.

BAB IV Perancangan

Membahas analisis kebutuhan dan perancangan Sistem informasi penanda dini bahaya banjir berbasis *Mobile Application* pada sistem operasi android.

BAB V Implementasi

Membahas implementasi dari Sistem informasi penanda dini bahaya banjir berbasis *Mobile Application* pada sistem operasi android sesuai dengan perancangan sistem yang telah dibuat.

BAB VI Pengujian dan Analisis

Memuat proses dan hasil pengujian terhadap sistem yang telah direalisasikan.

BAB VII Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangkan dalam skripsi ini serta saran – saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi tinjauan pustaka yang meliputi kajian pustaka dan dasar teori yang diperlukan untuk penelitian. Kajian pustaka adalah membahas penelitian yang telah ada dan yang diusulkan. Dasar teori yang dibutuhkan dalam penelitian ini meliputi Rekayasa Perangkat Lunak, *Software Process Model, Unified Modelling Language,* Pengujian Perangkat Lunak, Sistem Operasi Android, *Service Oriented Architecture* (SOA), *Web Services, Web Service Definition Language* (WSDL), *Simple Object Access Protocol* (SOAP), *eXtensible Markup Language* (XML), KSOAP *Library,* AChartEngine dan *Flood Early Warning System* (Sistem Informasi Deteksi Dini Bahaya Banjir).

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini adalah membahas penelitian sebelumnya yang berjudul "*Prototype of the Real Time Decision Support System for Flood Early Warning at Brantas River Basin*". Penelitian ini membahas tentang pengolahan dan penerimaan data yang dikirim oleh modul sensor pada stasiun pemantau kondisi DAS hulu dan hilir. Parameter yang diterima oleh modul sensor di setiap stasiun pemantau kondisi DAS berupa tingkat ketinggian air sungai, suhu dan kelembaban. Proses pengolahan data menghasilkan sebuah deteksi dini bahaya banjir kemudian ditampilkan ke dalam aplikasi *web* [SSE-13].

Perbedaan yang dibuat penulis pada penelitian ini adalah pada pengembangan *prototype* penelitian sebelumnya ke dalam *client side*. Pengembangan *prototype* tersebut merupakan proses pembuatan sebuah aplikasi perangkat bergerak di dalam sistem operasi android. Pembuatan aplikasi pada penelitian ini menggunakan teknologi *web service* dengan melakukan pertukaran data dari *server side* berupa data XML.

2.2 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak merupakan disiplin ilmu yang berkaitan dengan semua aspek produksi perangkat lunak dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah sistem digunakan. Dalam definisi rekayasa perangkat lunak terdapat dua frase kunci:

1. Disiplin rekayasa

   Perekayasa membuat suatu alat bekerja. Mereka menerapkan teori, metode dan alat-alat dimana dapat digunakan secara tepat. Mereka menggunakan secara selektif dan selalu mencoba untuk menemukan solusi dari suatu masalah bahkan ketika tidak terdapat dalam teori maupun metode. Perekayasa juga mengakui bahwa mereka harus bekerja dalam tekanan organisasi dan finansial sehingga mereka mencari solusi dalam kondisi tersebut.

2. Semua aspek dari produksi perangkat lunak

   Perangkat lunak tidak hanya peduli dengan proses teknis dari pengembangan perangkat lunak. Hal ini juga mencakup kegiatan seperti manajemen proyek perangkat lunak dan pengembangan alat, metode serta teori untuk mendukung produksi perangkat lunak.

Secara umum, rekayasa perangkat lunak mengadopsi pendekatan yang sistematis dan terorganisir untuk bekerja karena hal tersebut merupakan cara yang paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Namun, rekayasa adalah segala sesuatu tentang memilih metode yang paling sesuai untuk suatu set keadaan dan pendekatan yang lebih kreatif serta informal terhadap pengembangan yang mungkin efektif pada beberapa keadaan [SOM-11:7-10].

2.3 *Software Process Model*

Model proses perangkat lunak adalah representasi yang sederhana dari proses perangkat lunak. Setiap model proses merupakan proses dari perspektif tertentu yang hanya menyediakan informasi parsial tentang proses tersebut. Sebuah model proses untuk rekayasa perangkat lunak dipilih berdasarkan sifat proyek dan aplikasi, metode dan alat-alat yang akan digunakan serta kontrol *deliverable* yang diperlukan. Beberapa model proses perangkat lunak yang sering digunakan para pengembang perangkat lunak adalah *waterfall model, incremental development,* dan *reuse-oriented software engineering* [SOM-11:29-35].

Reuse-oriented software engineering adalah penggunaan kembali perangkat lunak yang telah ada. Hal ini sering terjadi ketika orang-orang yang bekerja pada proyek mengetahui desain atau kode yang mirip dengan apa yang dibutuhkan. Mereka mencari dan memodifikasi sesuai kebutuhan lalu menggabungkan ke

dalam sistem. Pendekatan *reuse-oriented software engineering* mengandalakan komponen perangkat lunak yang dapat digunakan kembali dan mengintegrasikan kerangka untuk komposisi komponen sistem [SOM-11:35].

Tahapan utama dari *reuse-oriented software engineering* secara langsung mencerminkan dasar pembangunan kegiatan [SOM-11:35]:

1. Analisis komponen

   Dari spesifikasi kebutuhan yang telah ditentukan, dicari komponen yang dapat mengimplementasi spesifikasi tersebut. Biasanya, tidak ada komponen yang benar-benar mampu memenuhi spesifikasi dan mungkin hanya beberapa komponen yang memenuhi persyaratan fungsional dari spesifikasi kebutuhan.

2. Modifikasi Kebutuhan

   Selama tahap ini kebutuhan dianalisis menggunakan informasi tentang komponen yang telah ditentukan. Kemudian dimodifikasi untuk mencerminkan komponen yang tersedia. Ketika satu kondisi tidak memungkinkan untuk memodifikasi kebutuhan maka kegiatan analisis komponen dapat dilakukan kembali untuk mencari solusi alternatif.

3. Desain sistem dengan *reuse*

   Tahap ini merupakan proses merancang *framework* pada sistem atau *framework* yang tersedia akan digunakan kembali. Para desainer memperhitungkan komponen yang digunakan kembali dan mengatur *framework* untuk mengembangkannya.

4. Pengembangan dan integrasi perangkat lunak

   Perangkat lunak yang tidak dapat diperoleh secara eksternal maka akan dikembangkan dan komponen diintegrasikan untuk menciptakan sistem baru.

2.4 ***Unified Modelling Language*** (UML)

UML adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap

bentuk memiliki makna tertentu. UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering) [DSA-03].

Adapun sejarah UML dimulai dari banyaknya metodologi pemodelan berorientasi objek yang telah bermunculan di dunia di era tahun 1990. Diantaranya metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dan sebagainya. Dikarenakan metodologi yang ada membawa notasi sendiri-sendiri yang mengakibatkan timbul masalah baru apabila ada sebuah kerjasama dengan *group*/perusahaan lain yang menggunakan metodologi yang berlainan sehingga pada tahun 1996 UML dijadikan standar bahasa pemodelan untuk aplikasi berorientasi objek [DSA-03].

Pada UML versi 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori [DSA-03]. Berikut ini penjelasan singkat dari pembagian kategori tersebut.

- *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
- *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
- *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

2.4.1 *Use Case* Diagram

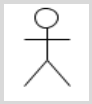
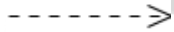
Diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Diagram *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

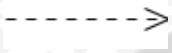
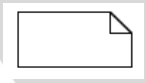
Syarat penamaan pada diagram *use case* adalah nama didefinisikan sesederhana mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu definisi dari aktor dan *use case.*

1. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Simbol aktor adalah gambar orang. Namun aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor.

Tabel 2.1 menjelaskan simbol-simbol yang digunakan dalam *use case* diagram.

Tabel 2.1 Keterangan Simbol *Use Case* Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		Aktor / *Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri. Jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang. Biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
2		Generalisasi/ *Generalization*	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
3		Menggunakan / *include / uses* <<include>>  <<uses>>	Fungsinya atau sebagai syarat dijalankan *use case* ini. Ada dua sudut pandang yang cukup besar mengenai include di dalam *use case* : Include berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan. Include berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan. Kedua interpretasi di atas dapat

			dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.
4		Ekstensi/ *Extend* <<extend>>	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.
5		Asosiasi/ *Association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
6		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
7		*Use Case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case.*
8		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
9		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Sumber : [RJB-99:42-46]

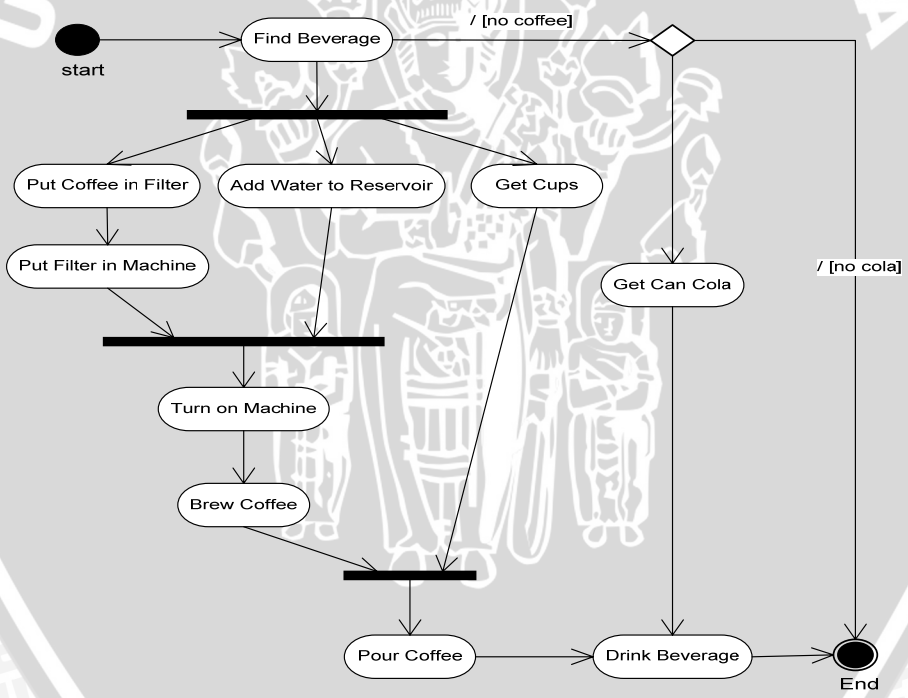
2.4.2 *Activity* Diagram

Activity diagram adalah representasi grafis dari alur kerja tahapan aktifitas. Diagram ini mendukung pilihan tindakan, iterasi dan *concurrency*. Pada pemodelan UML, *activity diagram* dapat digunakan untuk menjelaskan bisnis dan

alur kerja operasional secara *step-by-step* dari komponen suatu sistem. *Activity diagram* menunjukkan keseluruhan dari aliran kontrol. *Activity* diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir [DSA-03].

Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity* diagram merupakan *state* diagram khusus di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas [DSA-03].

Gambar 2.1 merupakan contoh *activity* diagram.



Gambar 2.1 *Activity* Diagram
Sumber : [DSA-03]

2.4.3 *Class* Diagram

Class diagram adalah suatu diagram yang memperlihatkan atau menampilkan struktur dari sebuah sistem yang akan menampilkan sistem kelas, atribut dan hubungan antara kelas. Objek diagram adalah suatu diagram yang berfungsi untuk

mengatur atribut, objek dan hubungan antara *class*. Objek diagram juga dapat menampilkan struktur model sistem dalam waktu tertentu. *Class* dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time* [DSA-03].

Class memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

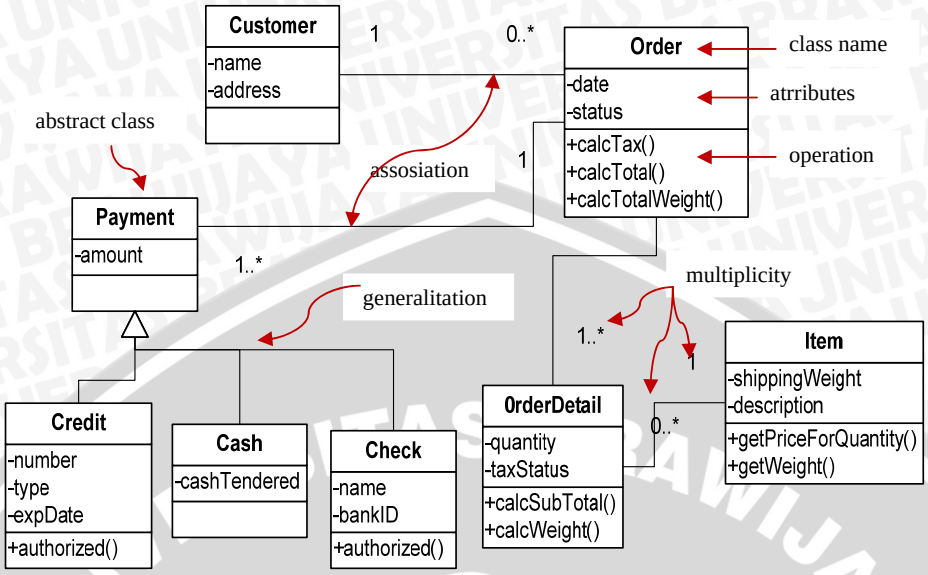
Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private,* tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected,* hanya dapat dipanggil oleh *class* yang bersangkutan dan anak - anak yang mewarisinya
3. *Public,* dapat dipanggil oleh siapa saja

Hubungan antara *class* dibagi menjadi empat bagian antara lain.

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas”).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain, mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang dikirim dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence* diagram.

Gambar 2.2 merupakan sebuah *class* diagram yang memiliki beberapa hubungan antar *class.*



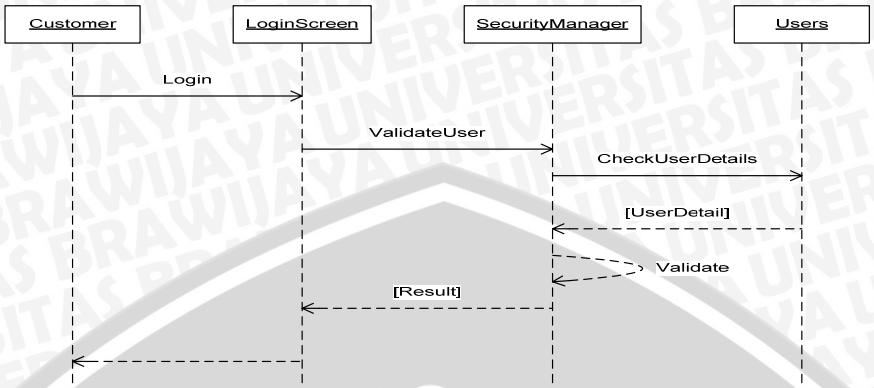
Gambar 2.2 *Class* Diagram
Sumber : [DSA-03]

2.4.4 *Sequence* Diagram

Sequence diagram adalah suatu diagram yang menggambarkan interaksi antara objek dan mengindikasikan komunikasi diantara objek-objek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh objek-objek yang melakukan suatu tugas atau aksi tertentu.

Objek-objek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya diletak di paling kiri dari diagram. Pada diagram ini, dimensi vertikal merepresentasikan waktu [DSA-03].

Bagian paling atas dari diagram menjadi titik awal dan waktu berjalan ke bawah sampai dengan bagian dasar dari diagram. Garis vertical pada *sequence* diagram disebut *lifeline,* dilekatkan pada setiap objek atau aktor. Kemudian, *lifeline* tersebut digambarkan menjadi kotak ketika objek melakukan suatu operasi, kotak tersebut disebut *activation box*. Objek dikatakan mempunyai *live activation* pada saat tersebut. Pesan yang dipertukarkan antar objek digambarkan sebagai sebuah anak panah antara *activation box* pengirim dan penerima. Kemudian diatasnya diberikan label pesan [DSA-03]. Gambar 2.3 merupakan contoh *sequence* diagram.



Gambar 2.3 *Sequence* Diagram
Sumber : [DSA-03]

2.5 Pengujian Perangkat Lunak

Pengujian perangkat lunak dilakukan untuk mengetahui bahwa program dapat melakukan apa yang seharusnya dilakukan dan untuk menemukan sebuah kesalahan pada program sebelum program tersebut mulai digunakan. Ketika melakukan pengujian perangkat lunak, program harus dijalankan dengan menggunakan data buatan. Tahap selanjutnya, memeriksa hasil kasus uji untuk beberapa kesalahan, anomali atau informasi terkait atibut non-fungsional program.

Proses pengujian perangkat lunak memiliki dua tujuan [SOM-11:206]:

1. Untuk menunjukkan kepada pengembang dan pengguna bahwa perangkat lunak memenuhi spesifikasi kebutuhan.
2. Untuk menemukan situasi di mana perilaku dari perangkat lunak terdapat sebuah kesalahan, hal yang tidak diinginkan atau tidak sesuai dengan spesifikasi kebutuhan. Perilaku tersebut seperti *system crashes*, interaksi antar sistem, kesalahan dalam komputasi dan perubahan data.

2.5.1 Teknik Pengujian

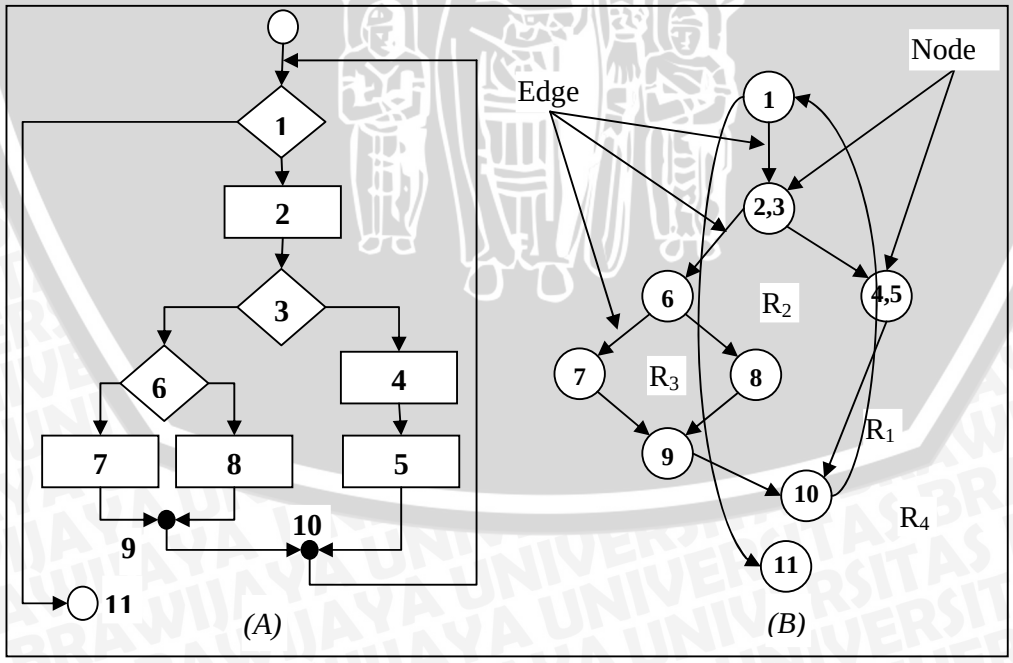
Pengujian perangkat lunak memerlukan perancangan kasus uji (*test case*) agar dapat menemukan kesalahan dalam waktu singkat dan usaha minimum. Berbagai macam metode perancangan kasus uji telah berevolusi. Metode-metode tersebut dilakukan dengan pendekatan sistematis dalam melakukan pengujian. Terlebih lagi metode-metode ini menyediakan mekanisme yang dapat membantu memastikan kelengkapan dari pengujian dan menyediakan kemungkinan tertinggi untuk menemukan kesalahan-kesalahan dalam perangkat lunak [PRR-01:443].

Teknik atau metode perancangan kasus uji yang digunakan adalah *black-box testing* dan *white-box testing*.

2.5.1.1 *White-Box Testing*

White-box testing atau *glass-box testing* merupakan sebuah metode perancangan kasus uji yang menggunakan struktur kontrol dari perancangan prosedural untuk memperoleh kasus uji [PRR-01:444]. Ada dua jenis pengujian yang termasuk *white-box testing* yaitu *basis path testing* dan *control structure testing.*

Basis path testing ini memungkinkan perancangan kasus uji memperoleh ukuran kompleksitas logis dari sebuah perancangan prosedural dan menggunakan pengukuran ini sebagai pedoman untuk mendefinisikan *basis set* dari jalur eksekusi (*execution path*). *Test case* yang dilakukan dengan menggunakan *basis set* yang akan membuktikan bahwa setiap *statement* di dalam program akan dieksekusi paling tidak sekali selama pengujian. Setiap representasi desain prosedural yang berupa *flow chart* dapat diterjemahkan ke dalam *flow graph*. Gambar 2.4 menunjukkan transformasi *flow chart (A)* ke *flow graph (B).* Setelah *flow graph* didefinisikan maka harus ditentukan ukuran kompleksitas (*cyclomatic complexity*).



Gambar 2.4 Transformasi *Flow Chart* ke *Flow Graph*
Sumber : [PPR-01:447]

Cyclomatic complexity adalah metriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila metriks ini digunakan dalam konteks metode pengujian *basis path*, maka nilai yang terhitung untuk *cyclomatic complexity* menentukan jumlah jalur independen (*independent path*) dalam *basis set* suatu program dan memberi batas terhadap jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua *statement* telah dieksekusi sedikitnya satu kali.

Jalur independen adalah jalur dari sebuah program yang mengenalkan sedikitnya satu rangkaian *statement* proses baru atau suatu kondisi baru. Untuk menentukan *cyclomatic complexity* bisa dilakukan dengan beberapa cara, diantaranya [PRR-01:448].

1. Jumlah *region* pada *flow graph* sesuai dengan *cyclomatic complexity*.
2. Cyclomatic complexity V(G), untuk grafik G adalah V(G) = E – N + 2, dimana E adalah jumlah *edge*, dan N adalah jumlah *node.*
3. V(G) = P + 1, dimana P adalah jumlah *predicate node* yaitu *node* yang merupakan kondisi (ada 2 atau lebih *edge* akan keluar *node*).

2.5.1.2 *Black-Box Testing*

Black-box testing atau pengujian perilaku berfokus pada persyaratan fungsional perangkat lunak. *Black-box testing* memungkinkan pengembang perangkat lunak untuk mengatur kondisi input untuk melaksanakan semua persyaratan fungsional program. *Black-box testing* menggunakan pendekatan komplementer yang memungkinkan untuk mengungkap kesalahan [PRR-01:459]. *Black-box testing* dapat mengungkap kesalahan dalam kategori berikut.

1. Fungsi yang hilang atau tidak benar
2. Kesalahan antarmuka
3. Kesalahan dalam struktur data
4. Kesalahan perilaku atau kinerja
5. Kesalahan inisialisasi dan terminasi

Ada banyak keuntungan dalam penggunaan metode *black-box testing*. Berikut adalah beberapa keuntungan penggunaan metode *black-box testing*.

1. Kemudahan dalam penggunaan. Karena penguji hanya berfokus terhadap beberapa kasus uji yang bekerja dalam aplikasi.

2. Pengembangan uji kasus yang lebih cepat. Karena penguji hanya berfokus terhadap perilaku sistem melalui antarmuka pengguna aplikasi.
3. Pengujian dengan cara berfokus pada masukan valid dan tidak valid serta memastikan output yang diterima sesuai dengan tujuan.

2.5.2 Strategi Pengujian

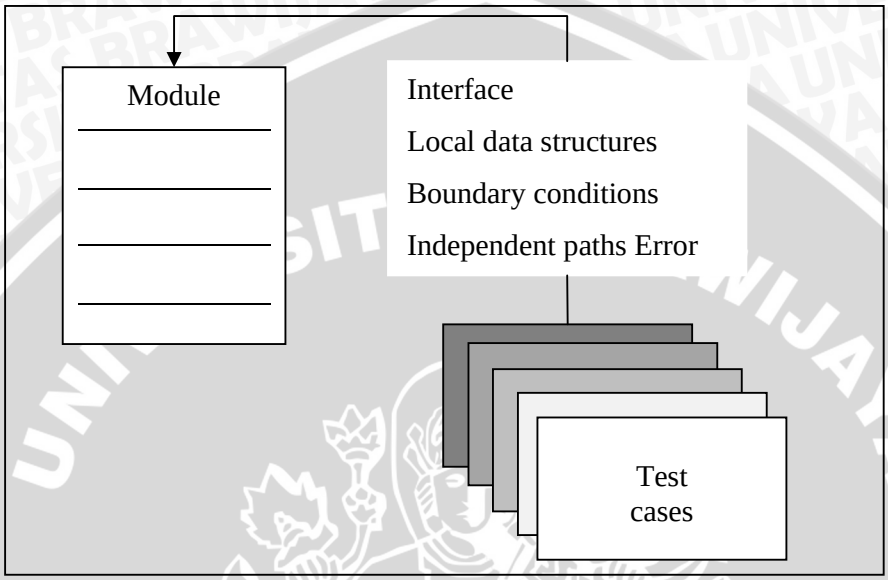
Strategi untuk pengujian perangkat lunak mengintegrasikan metode perancangan *test case* perangkat lunak ke dalam sederetan langkah yang direncanakan dengan baik, dan hasilnya adalah konstruksi perangkat lunak yang berhasil [PRR-01:477]. Sejumlah strategi pengujian perangkat lunak telah diusulkan di dalam literatur. Strategi pengujian harus mengakomodasi pengujian tingkat rendah yang diperlukan untuk membuktikan bahwa segmen kode sumber yang kecil telah diimplementasikan dengan tepat. Demikian juga pengujian tingkat tinggi yang memvalidasi fungsi-fungsi sistem mayor yang berlawanan dengan kebutuhan pelanggan. Proses pengujian dimulai dengan pengujian yang berfokus pada setiap modul secara individual (*unit testing*), dilanjutkan dengan pengujian integrasi (*integration testing*) dan pengujian validasi (*validation testing*) [PRR-01:481].

2.5.2.1 Pengujian Unit

Pengujian unit berfokus pada usaha verifikasi pada inti terkecil dari desain perangkat lunak, yakni modul. Dengan menggunakan gambaran desain prosedural sebagai panduan, jalur kontrol perlu diuji untuk mengungkap kesalahan di dalam batas modul tersebut. Kompleksitas relatif dari pengujian dan kesalahan yang diungkap dibatasi oleh ruang lingkup batasan yag dibangun untuk pengujian unit. Pengujian unit biasanya berorientasi pada *white-box*, dan langkahnya dapat dilakukan secara paralel untuk model bertingkat [PRR-01:485].

Pengujian yang terjadi sebagai bagian dari inti digambarkan secara skematis pada gambar 2.5. *Interface* modul diuji untuk memastikan bahwa informasi secara tepat mengalir masuk dan keluar dari inti program yang diuji. Struktur data lokal diuji untuk memastikan bahwa data yang tersimpan secara temporal dapat tetap menjaga integritasnya selama semua langkah di dalam suatu algoritma dieksekusi.

Kondisi batas diuji untuk memastikan bahwa modul beroperasi dengan tepat pada batas yang ditentukan untuk membatasi pemrosesan. Semua jalur independen (jalur dasar) yang melalui struktur kontrol dipakai sedikitnya satu kali. Dan akhirnya, penanganan kesalahan uji [PRR-01:485].

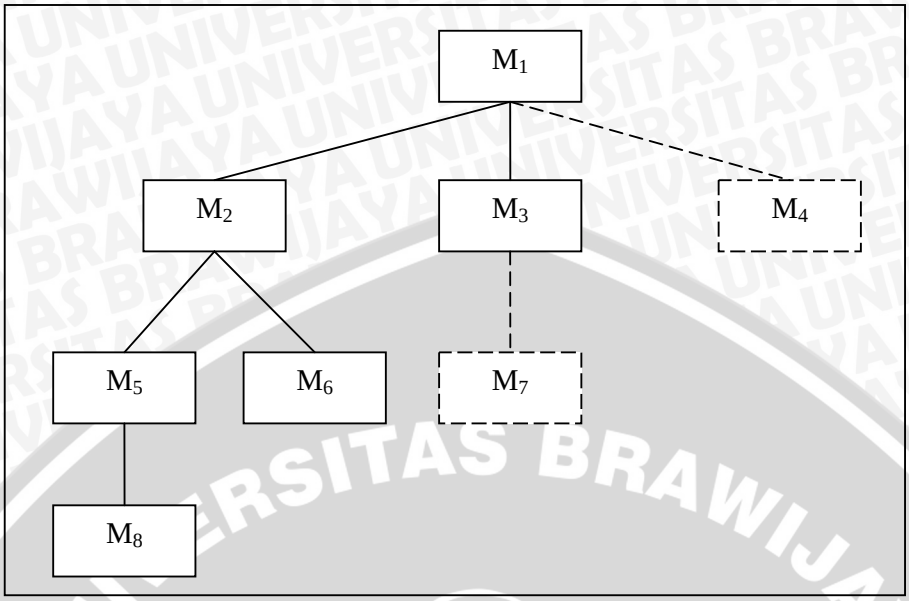


Gambar 2.5 Pengujian Unit
Sumber : [PPR-01:487]

2.5.2.2 Pengujian Integrasi

Pengujian integrasi adalah teknik sistematis untuk mengkonstruksi struktur program dengan melakukan pengujian untuk mengungkap kesalahan sehubungan dengan *interfacing*. Sasarannya adalah untuk mengambil modul yang sudah dilakukan pengujian unit dan membangun struktur program yang telah ditentukan terhadap desain sistem. *Integration testing* berorientasi *black box* dan mempunyai dua pola pengujian yaitu integrasi *top-down* (*top-down integration)* dan integrasi *bottom-up* (*bottom-up integration)* [PRR-01:488].

Integrasi *top-down* adalah pendekatan inkremental terhadap struktur program. Modul diintegrasikan dengan menggerakkan ke bawah melalui hirarki kontrol, dimulai dengan modul kontrol utama (program utama). Subordinat program terhadap modul kontrol utama digabungkan ke dalam struktur dengan cara *depth-first* atau *breadth-first* [PRR-01:489]. Gambar 2.6 menunjukkan pola pengujian integrasi *top-down*.

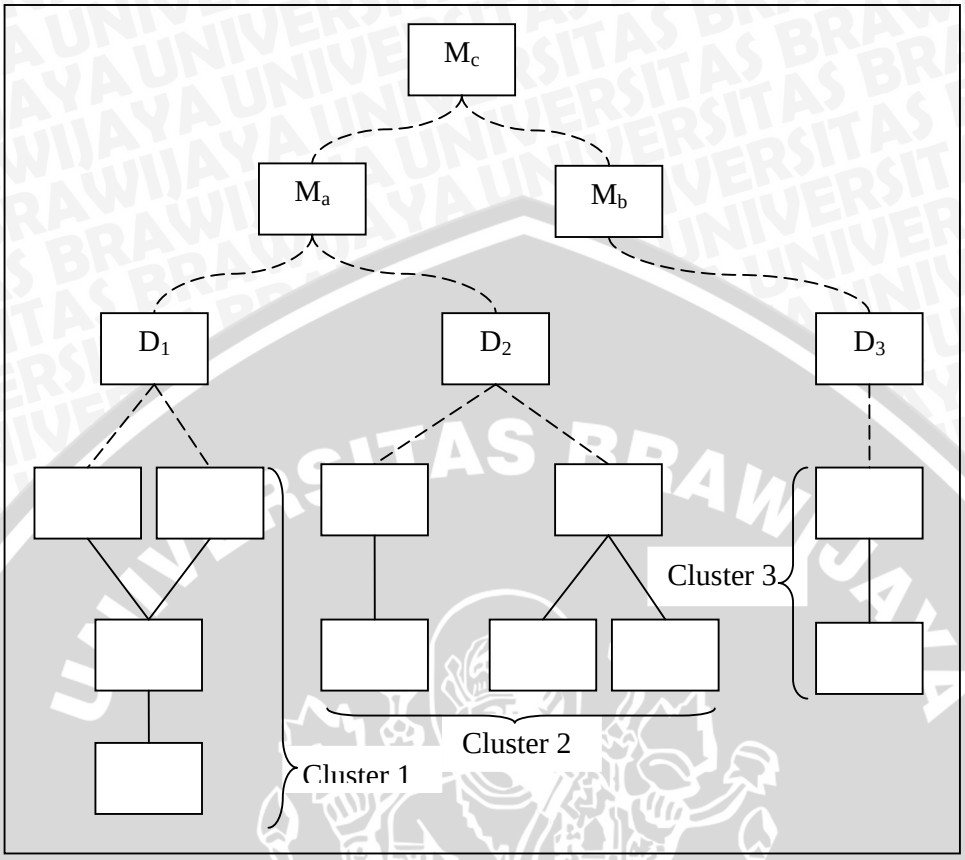


Gambar 2.6 Integrasi *Top-Down*
Sumber : [PPR-01:489]

Proses integrasi *top-down* dilakukan dalam lima langkah [PRR-01:489]:

- Modul kontrol utama digunakan sebagai *test driver* dan *stub* digunakan untuk menggantikan semua komponen dibawahnya.
- Pemilihan pendekatan integrasi yang diinginkan (*depth* atau *breadth first*).
- Pengujian dikerjakan untuk setiap komponen yang diintegrasikan.
- *Stub* digantikan dengan komponen yang sebenarnya setelah menyelesaikan serangkaian pengujian.
- Proses akan terus dilakukan sampai membentuk sebuah perangkat lunak yang utuh.

Pengujian integrasi *bottom-up* memulai konstruksi dan pengujian dengan modul atomik (modul pada tingkat paling rendah pada struktur program). Hal ini terlihat pada gambar 2.7. Karena modul diintegrasikan dari bawah ke atas, maka pemrosesan yang diperlukan untuk modul subordinat ke suatu tingkat yang diberikan akan selalu tersedia dan kebutuhan akan *stub* dapat dieliminasi [PRR-01:490].



Gambar 2.7 Integrasi *Bottom-Up*
Sumber : [PPR-01:491]

Integrasi *bottom-up* dapat diimplementasi dengan langkah-langkah berikut [PRE-01:490] :

- Komponen pada level terendah digabung ke dalam sebuah sub fungsi (*cluster*).
- Driver akan dibuat untuk menguji setiap *cluster*.
- Driver akan diganti dengan modul sesungguhnya setelah sub fungsi teruji.
- Proses akan terus dilakukan sampai membentuk sebuah perangkat lunak yang utuh.

2.5.2.3 Pengujian Validasi

Pada kulminasi pengujian terintegrasi, perangkat lunak secara lengkap dirakit sebagai suatu paket, kesalahan *interfacing* telah diungkap dan dikoreksi serta seri akhir dari pengujian perangkat lunak, yaitu pengujian validasi dapat dilakukan. Validasi dapat ditentukan dengan berbagai cara, tetapi definisi yang sederhana adalah bahwa validasi berhasil bila perangkat lunak berfungsi sesuai dengan yang

diharapkan pengguna. Validasi perangkat lunak dicapai melalui sederetan pengujian *black-box* yang memperlihatkan konformitas dengan spesifikasi kebutuhan. Rencana pengujian menguraikan kelas-kelas pengujian yang akan dilakukan dan prosedur pengujian menentukan *test case* spesifik yang akan digunakan untuk mengungkap kesalahan dalam konformitas dengan spesifikasi kebutuhan. Baik rencana dan prosedur dirancang untuk memastikan apakah semua persyaratan fungsional dipenuhi, semua persyaratan kinerja dicapai, dokumentasi benar dan direkayasa oleh manusia serta persyaratan lainnya dipenuhi (transportabilitas, kompatibilitas, pembetulan kesalahan, maintanabilitas) [PRR-01:495].

2.5.2.4 *User Acceptance Testing* (UAT)

UAT adalah salah satu prosedur proyek perangkat lunak akhir dan kritis yang harus terjadi sebelum perangkat lunak dikembangkan atau tergulir ke pasar (pengguna). Selama UAT, pengguna perangkat lunak sebenarnya menguji perangkat lunak untuk memastikan bahwa perangkat lunak sudah dapat menangani tugas-tugas yang diperlukan dalam skenario dunia nyata dan sesuai dengan spesifikasi [MAX-13].

2.6 Sistem Operasi Android

Sistem operasi Android adalah *platform* terbuka dari Google yang dirancang untuk perangkat mobile. Tujuan dari *platform* terbuka tersebut adalah untuk mempercepat inovasi dalam konsumen perangkat bergerak dan menawarkan pengalaman dalam perangkat bergerak yang lebih kaya, lebih murah dan lebih baik [MAR-11].

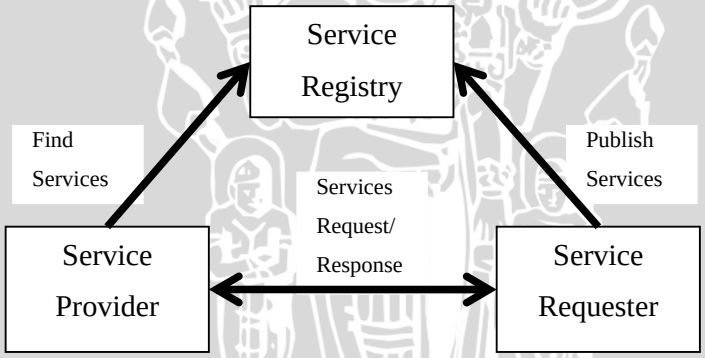
Android adalah *platform open source.* Mulai dari *lob-level* Linux *modul,* semua *native library,* sampai *application framework* untuk semua aplikasi bersifat terbuka. Android dilisensikan di bawah lisensi *Business-friendly licences* (Apache/MIT) sehingga orang lain dapat memperpanjang dan menggunakan untuk berbagai keperluan secara bebas. Pengembang memiliki akses untuk sumber kode pada seluruh *platform*. Sedangkan produsen dapat dengan mudah menanamkan sistem operasi Android pada perangkat keras tertentu.

Tujuan utama diciptakannya Android adalah untuk perangkat *mobile.* Ketika merancang Android terdapat kendala di mana perangkat *Mobile* kemungkinan tidak akan ada perubahan yang signifikan di masa mendatang. Faktor yang

pertama, sumber tenaga dari perangkat *mobile* adalah baterai dan performa baterai kemungkinan tidak akan berkembang dengan pesat dalam waktu dekat. Faktor yang kedua, perangkat *mobile* berukuran kecil, berarti bahwa perangkat *mobile* akan selalu memiliki banyak keterbatasan dalam hal memori dan kecepatan. Android dirancang untuk berjalan pada segala macam jenis perangkat keras. Android tidak membuat asumsi tentang ukuran layar perangkat, resolusi, *chipset*, dan sebagainya.

2.7 *Service Oriented Architecture* (SOA)

SOA merupakan sebuah model arsitektur untuk membangun aplikasi perangkat lunak yang menggunakan beberapa *service* yang tersedia di sebuah jaringan seperti Web. SOA merupakan konsep dasar yang melandasi implementasi dari sebuah *web services* [SKG-08]. Model sistem dengan menerapkan SOA memiliki berbagai macam layanan yang disediakan untuk pengguna. *Service* merupakan sekumpulan aktifitas yang disusun sedemikian rupa dan berfungsi untuk sebuah tujuan.



Gambar 2.8 Operasi Dasar SOA
Sumber : [SKG-08]

Terdapat 3 bagian pada konsep SOA, yaitu *Service Provider*, *Service Registry*, dan *Service Requester*. *Service Provider* adalah penyedia *service*, *Service Registry* adalah bagian yang menyimpan informasi tentang *service* yang tersedia, dan *Service Requester* adalah *user* atau pengguna *service*.

Service Provider mendefinisikan *service* yang disediakan dengan menggunakan WSDL (Web Service Description Language) dan mendaftarkan pada *Service Registry*. *Service Requester* mencari informasi *service* pada *Service Registry*, informasi yang disediakan adalah daftar *service*, lokasi *service* dan tata

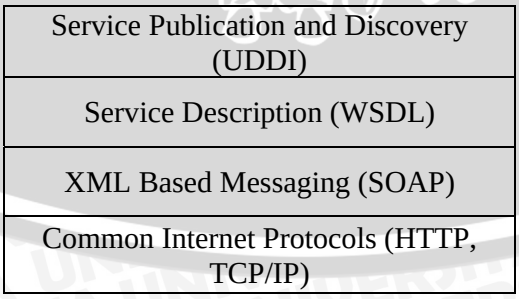
cara berkomunikasi. Setelah mendapatkan informasi dari *Service Registry*, *Service Requester* melakukan koneksi langsung ke *Service Provider*.

2.8 *Web Services*

Web service merupakan suatu komponen software yang merupakan *self-containing*, aplikasi modular *self-describing* yang dapat dipublikasikan, dialokasikan, dan dilaksanakan pada web. *Web service* adalah teknologi yang mengubah kemampuan internet dengan menambahkan kemampuan *transactional web*, yaitu kemampuan web untuk saling berkomunikasi dengan pola *program-to-program* (P2P). Fokus web selama ini didominasi oleh komunikasi *program-to-user* dengan interaksi *business-toconsumer* (B2C), sedangkan *transactional web* akan didominasi oleh *program-to-program* dengan interaksi *business-to-business* [DEV-11].

Gambar 2.9 merupakan blok bangunan *web service* yang menyediakan fasilitas komunikasi jarak jauh antara dua aplikasi.

a. Layer 1 : protokol internet standar yang digunakan sebagai sarana transportasi adalah HTTP dan TCP/IP.
b. Layer 2 : *Simple Object Access Protocol* (SOAP) berbasiskan XML dan digunakan untuk pertukaran informasi antar sekelompok layanan.
c. Layer 3 : *Web service Definition Language* (WSDL) digunakan untuk mendiskripsikan *attribute* layanan.
d. Layer 4 : *Universal Description, Discovery and Integration*, yang mana merupakan direktori pusat untuk deskripsi layanan.



Gambar 2.9 Blok Bangunan *Web Service*
Sumber : [DEV-11]

2.9 *Web Service Definition Language* (WSDL)

WSDL merupakan sebuah bahasa berbasis XML yang digunakan untuk mendefinisikan *web service* dan menggambarkan bagaimana cara untuk mengakses *web service* tersebut. Fungsi utama WSDL dalam *web service* adalah untuk mengotomasi mekanisme komunikasi business-to-business dalam *web service* melalui protokol internet. WSDL merupakan representasi kontrak antara requestor dan providernya. Secara teknis merupakan representasi kontrak antara kode klien dan kode di server. Dengan menggunakan WSDL klien dapat memanfaatkan fungsi-fungsi publik yang disediakan oleh server [DEV-11].

Didalam dokumen WSDL terdapat lima elemen utama antara lain.

1. Elemen <type>, berfungsi untuk mendefinisikan tipe data-tipe data yang digunakan dalam pesan.
2. Elemen <message>, berfungsi untuk mendefinisikan format dari sebuah pesan. Pesan digunakan sebagai struktur masukan (input) atau keluaran (output) bagi operasi.
3. Elemen <portType>, berfungsi untuk mendefinisikan sekumpulan operasi-operasi. Tiap-tiap elemen <operation> mendefinisikan sebuah operasi dan pesan masukan atau keluaran yang berkaitan dengan operasi tersebut.
4. Elemen <binding>, berfungsi untuk memetakan operasi-operasi dan pesan yang terdefinisikan pada port type ke protokol tertentu.
5. Elemen <service>, berfungsi untuk mendefinisikan sekumpulan port-port yang saling berhubungan. Elemen <port> memetakan binding ke lokasi dari sebuah web-service [PTK-05].

2.10 *Simple Object Access Protocol* (SOAP)

SOAP merupakan protokol untuk pertukaran informasi dengan desentralisasi dan terdistribusi. SOAP merupakan gabungan antara HTTP dengan XML karena SOAP umumnya menggunakan protocol HTTP sebagai sarana transport datanya dan data akan dipertukarkan ditulis dalam format XML. Karena SOAP mengunakan HTTP dan XML maka SOAP memungkinkan pihak-pihak yang mempunyai platform, sistem operasi dan perangkat lunak yang berbeda dapat

saling mempertukarkan datanya. SOAP mengatur bagaimana request dan respon dari suatu *web service* bekerja [DEV-11].

2.11 *eXtensible Markup Language* (XML)

XML merupakan dasar terbentuknya *web service* yang digunakan untuk mendeskripsikan data. Pada level paling detail *web service* secara keseluruhan dibentuk diatas XML. Fungsi utama dari XML adalah komunikasi antar aplikasi, integrasi data, dan komunikasi aplikasi eksternal dengan partner luaran. Dengan standarisasi XML, aplikasi-aplikasi yang berbeda dapat dengan mudah berkomunikasi antar satu dengan yang lain [DEV-11].

2.12 KSOAP *Library*

KSOAP adalah *library web service* dari klien yang terbatas untuk program-program Java seperti Applets atau aplikasi J2ME (CLDC/CDC/MIDP). KSOAP yang digunakan pada aplikasi ini adalah KSOAP versi 2 yang merupakan rancangan ulang yang lengkap dengan mengambil catatan dari kinerja KSOAP versi 1.x [KSP-06]. Beberapa perbedaan antara KSOAP versi 1.x dengan kSOAP versi 2 antara lain.

1. Struktur KSOAP versi 2 lebih bebas daripada KSOAP versi 1.
2. KSOAP versi 2 telah mendukung *literal coding*.
3. *Support* SOAP *Serialization* tidak lagi menjadi kebutuhan penting dan terpisah pada *package* yang lain.
4. Beberapa *class-class* yang berbeda telah diintegrasikan ke *class* SoapSerializationEnvelope, menyediakan dukungan SOAP Serialization SoapSerializationEnvelope dan memperluas *class* dasar SoapEnvelope.
5. *Flag* .Net dapat digunakan untuk mengganti SoapSerializationEnvelope dari penggunaan standar menjadi *namespace handling* yang menjadi default di .NET.

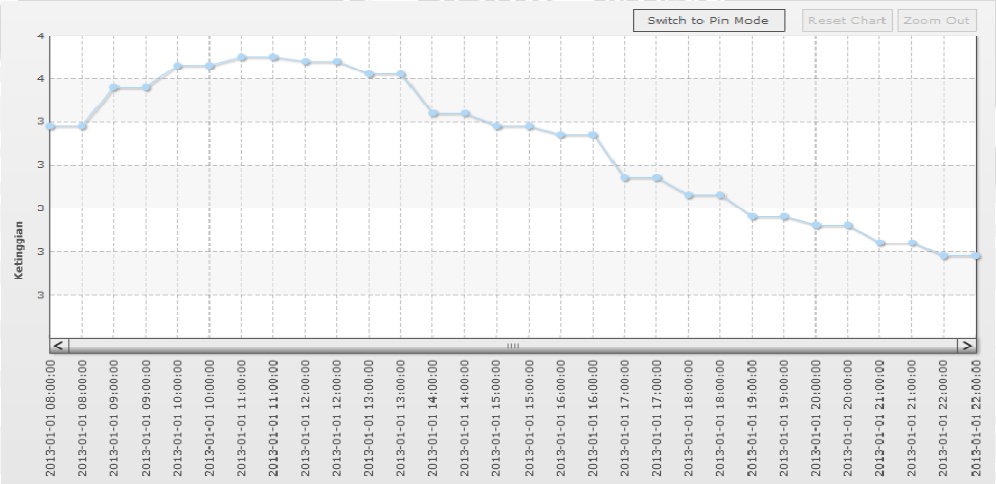
2.13 AChartEngine

AChartEngine merupakan sebuah *charting library* untuk aplikasi di sistem operasi android [ACE-09]. AChartEngine mendukung beberapa jenis grafik meliputi *line chart, area chart, scatter chart, time chart, bar chart, pie chart, bubble chart, doughnut chart, range bar chart, dial chart, combined chart dan*

cubic line chart. Jenis-jenis tersebut merupakan jenis grafik yang didukung berupa *multiple series*, dapat ditampilkan dengan sumbu X horizontal (*default*) atau vertical dan mendukung beberapa fitur lainnya. AChartEngine versi 1.0.0 merupakan AChartEngine yang dirilis pertama kali.

2.14 *Flood Early Warning System* (Sistem Informasi Deteksi Dini Bahaya Banjir)

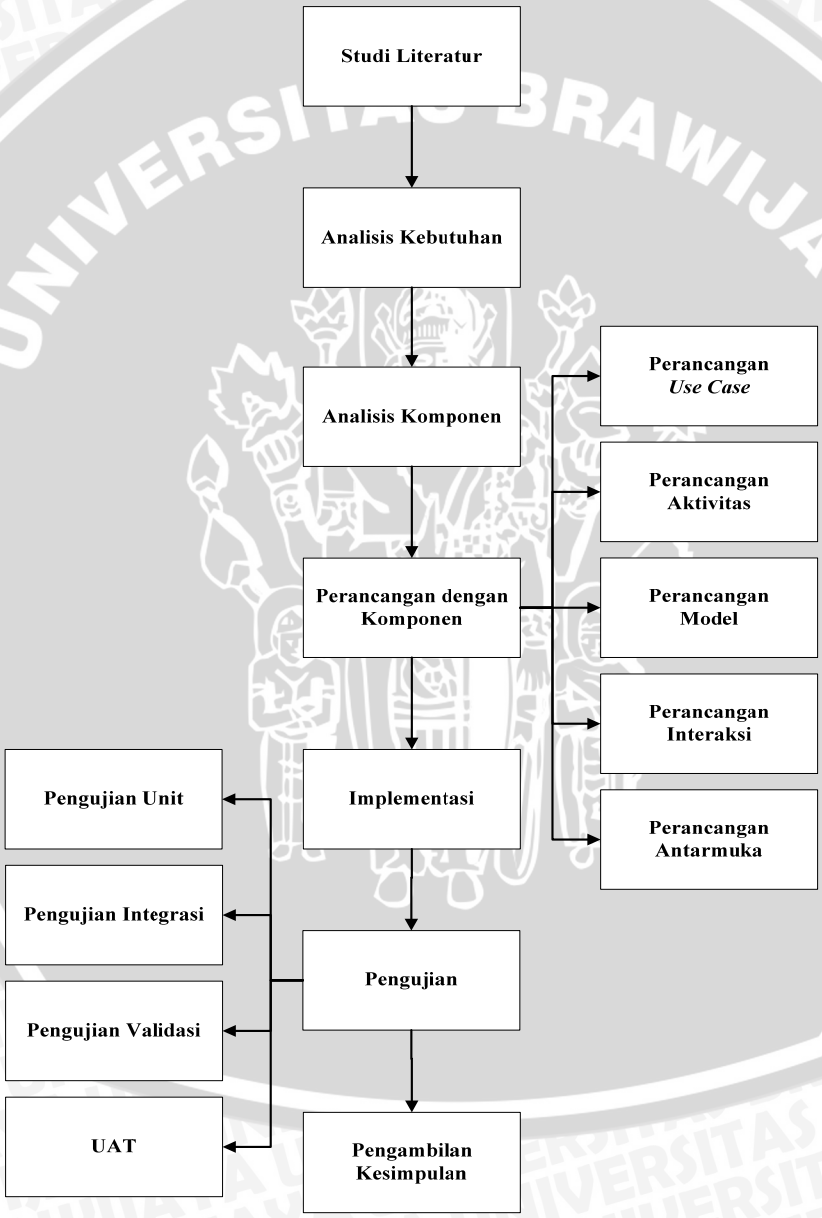
Flood Early Warning System (FEWS) merupakan sebuah *prototype* sistem pendukung keputusan penanda dini bahaya banjir secara *real time* menggunakan sebuah *datalogger* berbasis *Global System for Mobile Communications* (GSM). FEWS memantau kondisi DAS (Daerah Aliran Sungai) disepanjang hulu dan hilir. Proses pemantauan dilakukan dengan memasang sebuah perangkat keras yang digunakan sebagai stasiun pemantauan kondisi DAS. Di setiap stasiun pemantauan kondisi DAS memiliki beberapa modul sensor yang terhubung dengan *datalogger* pada GSM. Modul sensor akan memantau tiga parameter yaitu tingkat ketinggian air sungai, suhu dan kelembaban yang dilakukan secara terus-menerus. Hasil deteksi setiap data dari modul sensor diproses oleh *datalogger* kemudian dikirim ke *server* stasiun pusat melalui SMS *Gateway*. Hasil penerimaan dan pengolahan data akan ditampilkan ke dalam sebuah aplikasi *web* dengan beberapa fitur yaitu status pemantauan banjir, pemantauan tingkat air, peta wilayah, tanya jawab, manajemen berita, pemantauan grafik, dan dokumentasi pencetakan *log* kondisi air [SSE-13].



Gambar 2.10 Tampilan Aplikasi *Web* FEWS
Sumber : [SSE-13]

BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan mengenai prosedur-prosedur dan kegiatan-kegiatan yang akan dilakukan dalam pengerjaan skripsi, yaitu studi literatur, analisa dan perancangan, implementasi dan pengujian dalam pembuatan sistem informasi yang akan dibuat.



Gambar 0.1 Diagram Pohon Metodologi Penelitian
Sumber : [Perancangan]

3.1 Studi Literatur

Studi literatur merupakan penelusuran literatur yang bertujuan dalam menyusun dasar teori yang digunakan untuk menunjang skripsi. Penelusuran literatur dapat bersumber dari buku, media, pakar maupun hasil penelitian orang lain. Teori-teori pendukung tersebut meliputi:

1. Rekayasa Perangkat Lunak
2. *Software Proses Model*
3. *Unified Modelling Language*
    - *Use Case* Diagram
    - *Activity* Diagram
    - *Class* Diagram
    - *Sequence* Diagram
4. Pengujian Perangkat Lunak
5. Sistem Operasi Android
6. *Service Oriented Architecture* (SOA)
7. *Web Services*
8. *Web Service Definition Language* (WSDL)
9. *Simple Object Access Protocol* (SOAP)
10. *eXtensible Markup Language* (XML)
11. KSOAP *Library*
12. AChartEngine
13. *Flood Early Warning System*

3.2 Analisis Kebutuhan

Analisa kebutuhan merupakan sebuah proses awal untuk mendapatkan informasi, model, spesifikasi tentang perangkat lunak yang diinginkan oleh pengguna (*client*). Sistem yang baik dan sesuai dengan kebutuhan pengguna sangat bergantung kepada keberhasilan dalam analisa kebutuhan.

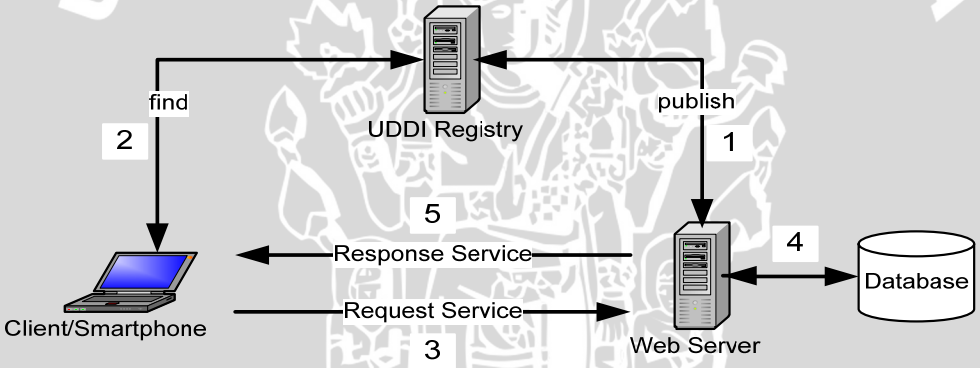
3.2.1 Kebutuhan Sistem

Dalam menganalisis sistem dibutuhkan dua jenis kebutuhan. Dua jenis kebutuhan tersebut adalah kebutuhan fungsional dan kebutuhan nonfungsional. Berikut penjelasan dari kebutuhan fungsional dan kebutuhan nonfungsional.

1. Kebutuhan Fungsional
   - Sistem dapat memberikan informasi tentang berita terkini mengenai bencana banjir.
   - Sistem mampu memberikan informasi tentang keadaan sungai di daerah pengguna.
   - Sistem dapat menampilkan interval ketinggian air dari sungai tertentu dalam waktu tertentu.
2. Kebutuhan Nonfungsional
   - Fitur-fitur sistem dapat berjalan dengan baik melalui telepon cerdas berbasis sistem operasi android versi 2.3.

3.2.2 Arsitektur Sistem

Arsitektur sistem penanda dini bahaya banjir di dalam sistem operasi android dijelaskan pada gambar berikut ini.

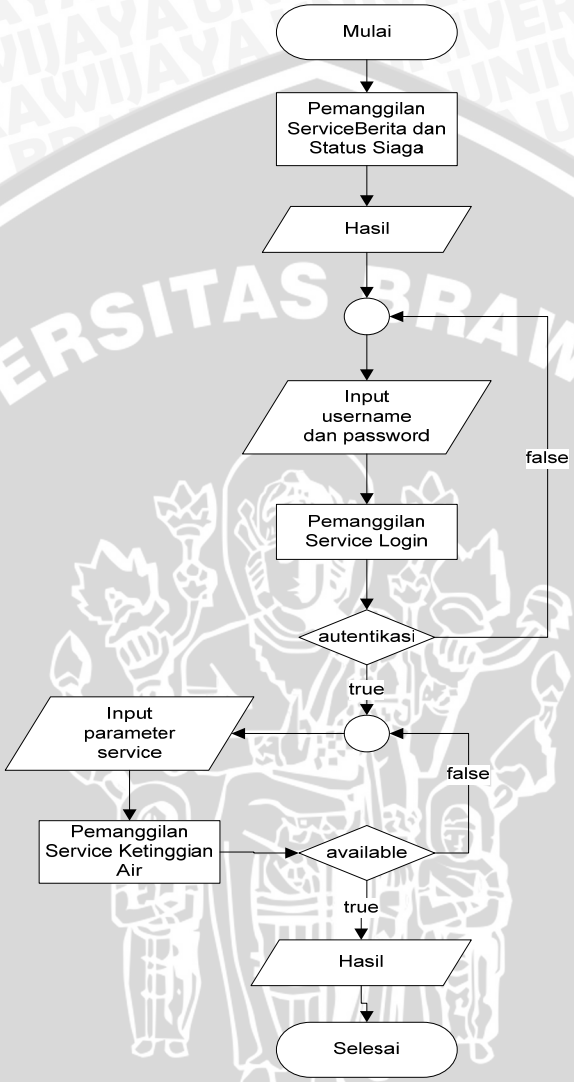


Gambar 3.2 Arsitektur Sistem
Sumber : [Perancangan]

Gambar 3.2 mengilustrasikan sebuah siklus sistem. Pada tahap pertama *web server* akan mempublikasikan *service* yang dimilikinya ke UDDI *registry*. Tahap kedua *client*/*smartphone* mencari *service* yang dibutuhkan di dalam UDDI *registry*. Tahap ketiga dan kelima merupakan proses pemanggilan operasi atau *method* yang melibatkan sebuah service melakukan suatu proses untuk memproses sebuah permintaan dari *client* kemudian mengirimkan hasil pemrosesan kepada *client.* Tahap keempat mengilustrasikan hubungan *web server* dengan *database* dalam hal penyimpanan sebuah data.

3.2.3 Diagram Alir Sistem

Diagram alir menggunakan notasi-notasi untuk menggambarkan arus data yang membantu dalam proses memahami jalannya aplikasi.



Gambar 3.3 Diagram Alir Sistem
Sumber : [Perancangan]

Program dimulai dari proses pemanggilan dan pemrosesan *service* berita dan status siaga yang dilakukan oleh sistem. Kemudian pengguna dapat melakukan proses input *username* dan *password. Username* dan *password* tersebut digunakan sebagai parameter dalam pemanggilan sebuah *service* login. Proses ini akan terus diulang selama hasil dari pemanggilan *service* bernilai *true* (benar). Setelah proses identifikasi pengguna selesai, pengguna dapat melihat hasil dari pemrosesan

pemanggilan *service* ketinggian air dengan cara memasukkan parameter yang sudah di sediakan oleh sistem.

3.3 Analisis Komponen

Tahap ini bertujuan untuk menentukan komponen-komponen yang akan dipakai pada aplikasi. Diawali dengan mengumpulkan seluruh komponen yang berhubungan dengan kebutuhan yang ada. Dilanjutkan dengan proses pemilihan komponen-komponen yang dapat digunakan untuk memenuhi kebutuhan. Proses pengembangan *Flood Early Warning System* di sisi *client* akan menggunakan sebuah *library* KSOAP. *Library* KSOAP ini berfungsi untuk mengkonversi data XML menjadi sebuah *object*. *Object* tersebut akan diolah di dalam aplikasi menjadi sebuah informasi. Informasi yang dimaksud meliputi informasi tentang berita terkini mengenai bahaya banjir, status siaga banjir dan ketinggian air sungai suatu daerah. Aplikasi ini akan dilengkapi dengan sebuah grafik yang menampilkan ketinggian air sungai suatu daerah. AChartEngine merupakan sebuah *library* yang dapat digunakan untuk membuat grafik di sistem operasi android.

3.4 Perancangan dengan Komponen

Selama tahap ini komponen-komponen yang telah ditentukan akan dilakukan penambahan atau perubahan ke dalam sistem. Kemudian dimodelkan ke dalam bentuk diagram *use case*, diagram aktivitas, diagram kelas dan diagram interaksi. Perancangan dilakukan dengan merancang *use case* dan aktivitas untuk menampilkan berita terkini banjir, status siaga banjir dan ketinggian air sungai suatu daerah. Tahap selanjutnya adalah merancang model dan interaksi. Tahap yang terakhir adalah merancang antarmuka sistem.

3.4.1 Perancangan *Use Case*

Sistem dapat dipandang dari tiga aspek yaitu: siapa pengguna sistem, apa kegiatan yang dapat dilakukan dalam sistem, dan bagaimana pengguna dapat melakukan kegiatan tersebut. Adapun kegiatan yang dapat dilakukan oleh sistem dan pengguna adalah sebagai berikut.

- Menampilkan dan melihat informasi tentang berita terkini bahaya banjir.

- Menampilkan dan melihat informasi tentang status siaga banjir suatu daerah.
- Menampilkan dan melihat informasi tentang ketinggian air sungai suatu daerah.

3.4.2 Perancangan Aktivitas

Alur kerja sistem secara berurutan digambarkan melalui *activity diagram.* Diagram aktivitas bermanfaat untuk menganalisis *use case* melalui penggambaran aktivitas-aktivitas yang dibutuhkan, penggambaran algoritma berurutan yang kompleks, dan pemodelan aplikasi dengan proses paralel. Adapun perancangan alur kerja sistem meliputi aktivitas untuk menampilkan berita terkini mengenai bencana banjir, aktivitas untuk menampilkan status siaga banjir suatu daerah dan aktivitas untuk menampilkan ketinggian air sungai suatu daerah.

3.4.3 Perancangan Kelas / Model

Perancangan kelas-kelas dan *interface-interface* dalam sistem dimodelkan dalam *class diagram.* Diagram kelas digunakan untuk mengidentifikasi kelas-kelas serta paket-paket yang terdapat dalam sistem. Kelas-kelas yang telah teridentifikasi kemudian mengindentifikasi hubungan antar kelas. Perancangan kelas dan *interface* dikategorikan menjadi beberapa bagian, antara lain:

- Kelas yang berfungsi untuk mengatur koneksi internet.
- Kelas yang berfungsi untuk mengolah data XML menjadi object SOAP.
- Kelas yang berfungsi untuk mengatur tampilan sistem.
- Kelas yang berfungsi untuk menampilkan informasi ke dalam bentuk grafik.

3.4.4 Perancangan Interaksi

Perancangan interaksi antar elemen (objek) dalam sistem yang telah teridentifikasi akan dimodelkan dalam *sequence diagram* yang menggambarkan interaksi antar objek yang disusun dalam urutan waktu. *Sequence diagram* berupa skenario atau rangkaian langkah-langkah yang dilakukan dari sebuah kejadian untuk menghasilkan *output* tertentu. Skenario yang akan dirancang meliputi:

- Skenario untuk menampilkan berita terkini bahaya banjir.
- Skenario untuk menampilkan status siaga banjir suatu daerah.

- Skenario untuk menampilkan ketinggian air sungai suatu daerah.

3.4.5 Perancangan Antarmuka

Kebutuhan-kebutuhan desain antarmuka untuk pembuatan sistem informasi ini sebagai berikut.

1. Sistem ini akan dibangun dengan tampilan yang sesuai dengan telepon cerdas berbasis sistem operasi android.
2. Performa antarmuka sistem harus disesuaikan dengan minimalnya sumber daya telepon cerdas berbasis sistem operasi android.

3.5 Implementasi

Implementasi perangkat lunak mengacu kepada perancangan perangkat lunak. Implementasi perangkat lunak dilakukan menggunakan bahasa pemrograman java di lingkungan sistem operasi android. Implementasi aplikasi ini antara lain.

- Penjabaran spesifikasi lingkungan pengembangan perangkat lunak.
- Implementasi data model *web services.*
- Pembuatan *class* dan *interface* yang merupakan komponen dari *file* program dengan ekstensi .java yang berfungsi untuk mengolah data *web services.*
- Pembuatan *user interface* di dalam sistem operasi android.

3.6 Pengujian

Pengujian perangkat lunak dilakukan untuk mengetahui apakah kinerja dan performa sistem perangkat lunak *Flood Early Warning System* berbasis perangkat bergerak telah memenuhi spesifikasi kebutuhan yang melandasinya. Strategi pengujian perangkat lunak yang akan digunakan yaitu pengujian unit (*unit testing*), pengujian integrasi (*integration testing*), pengujian validasi (*validation testing*) dan *user acceptance testing*. Metode pengujian yang akan digunakan adalah *white-box testing* dan *black-box testing*. Proses pengujian perangkat lunak dimulai dari pengujian unit, kemudian dilanjutkan dengan pengujian integrasi, dan berakhir pada pengujian validasi. Pada tahap pengujian unit dan pengujian integrasi digunakan metode *white-box testing* dengan teknik *basis path.* Kemudian pada tahap pengujian validasi digunakan metode *black-box testing.* Analisis juga

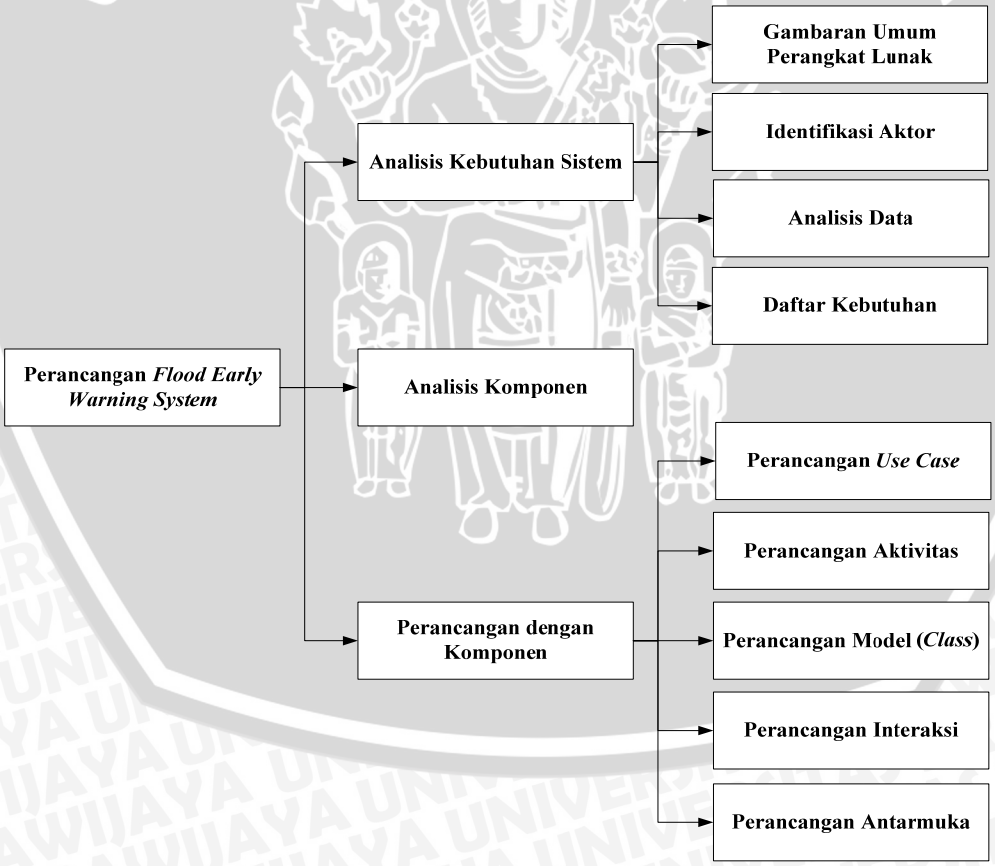
dilakukan untuk mengetahui hasil dari pengujian perangkat lunak sehingga dapat didapatkan kesimpulan dari pengembangan perangkat lunak yang telah dilakukan.

3.7 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan dan didasarkan pada kesesuaian antara teori dan praktik. Kesimpulan diambil untuk menjawab rumusan masalah yang telah ditetapkan sebelumnya. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

BAB IV
PERANCANGAN

Bab ini membahas mengenai proses analisis kebutuhan, proses analisis komponen dan proses perancangan perangkat lunak. Analisis kebutuhan terdiri dari empat tahap yaitu gambaran umum perangkat lunak, melakukan proses identifikasi aktor yang terlibat dalam sistem perangkat lunak, melakukan proses analisis data yang diperlukan, membuat daftar kebutuhan pengguna meliputi kebutuhan fungsional dan kebutuhan nonfungsional. Analisis komponen membahas perancangan komponen-komponen yang telah diperoleh sesuai dengan spesifikasi kebutuhan. Proses perancangan perangkat lunak meliputi perancangan *use case*, perancangan aktivitas, perancangan model (*class*), perancangan interaksi, perancangan antarmuka. Gambar 4.1 menjelaskan tahap-tahap dari perancangan sistem penanda dini bahaya banjir.



Gambar 4.1 Diagram Pohon Perancangan Penanda Dini Bahaya Banjir
Sumber : [Perancangan]

4.1 Analisis Kebutuhan

Analisis kebutuhan ini bertujuan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan pengguna. Proses analisis kebutuhan diawali dengan deskripsi umum perangkat lunak, identifikasi aktor yang terlibat, analisis data yang digunakan di dalam perangkat lunak dan penjabaran tentang daftar kebutuhan.

4.1.1 Gambaran Umum Perangkat Lunak

Perangkat lunak yang akan dikembangkan merupakan sebuah aplikasi perangkat bergerak penanda dini bahaya banjir di dalam sistem operasi android. Aplikasi ini merupakan proses pengembangan *client side* dari sebuah aplikasi *web FloodEarly Warning System.* Proses pengembangan tersebut dilakukan dengan memanfaatkan *web service* yang disediakan oleh *FloodEarly Warning System.* Aplikasi ini berfungsi memberikan informasi tentang berita terkini bencana banjir, ketinggian air dan status siaga keadaan sungai.

4.1.2 Identifikasi Aktor

Tahap ini mempunyai tujuan untuk melakukan identifikasi terhadap aktor yang akan berinteraksi dengan sistem. Penjelasan dari masing-masing identifikasi aktor dapat dilihat pada tabel 4.1.

Tabel 0.1 Identifikasi Aktor

Aktor	Deskripsi
Pengguna Biasa	Pengguna hanya dapat melihat berita terkini bahaya banjir dan status siaga suatu daerah.
Pengguna Utama	Dapat memanfaatkan seluruh fitur dari aplikasi mulai dari melihat berita terkini, status siaga banjir sampai melihat ketinggian air sungai suatu daerah
Pengelola Aplikasi Banjir *(*PAB*)*	Berperan sebagai aktor dalam memantau perkembangan aplikasi dan memberikan akses kepada pengguna lain

Sumber : [Perancangan]

4.1.3 Analisis Data

Sistem yang akan dibangun merupakan pengembangan *FloodEarly Warning System* dari *server side* menuju sisi pengguna (*client side*) dengan menggunakan teknologi *web service*. Pertukaran data antara *server side* dan *client side* berupa

data XML yang direpresentasikan di dalam WSDL. Berikut ini merupakan bagian data XML FEWS yang dikategorikan dalam lima elemen, antara lain:

1. Elemen <type>

Elemen ini berfungsi untuk mendefinisikan tipe data yang akan digunakan dalam pesan.

```
<types>
   <xsd:schema
      targetNamespace="https://fews.ub.ac.id/index.php/se
      rvicefews">
   <xsd:import
      namespace="http://schemas.xmlsoap.org/soap/encoding
      /"/>
   <xsd:import
      namespace="http://schemas.xmlsoap.org/wsdl/"/>

   <xsd:complexType name="FewsData">
      <xsd:complexContent>
         <xsd:restriction base="SOAP-ENC:Array">
         <xsd:element name="tempat" type="xsd:string"/>
         <xsd:element name="status" type="xsd:string"/>
         <xsd:element name="tinggiair"
         type="xsd:string"/>
         </xsd:restriction>
      </xsd:complexContent>
   </xsd:complexType>
   </xsd:schema>
</types>
```

Gambar 4.2 Elemen <type> XML FEWS
Sumber : [Perancangan]

Elemen *type* pada gambar 4.2 merupakan pendefinisian sebuah tipe data yang digunakan dalam *web service.* Atribut *TargetNamespace* pada elemen *schema* merupakan konvensi skema XML yang memungkinkan dokumen WSDL menunjuk ke dirinya sendiri. Atribut *complexType* merepresentasikan nama dari tipe data sedangkan atribut *element* berfungsi sebagai parameter untuk menerima masukan sesuai dengan tipe data masing-masing.

2. Elemen <message>

Elemen ini berfungsi untuk mendefinisikan format dari sebuah pesan. Pesan digunakan sebagai struktur masukan (*input*) atau keluaran (*output*) bagi operasi.

```
1  <message name="getBeritaRequest">
2  <part name="param" type="xsd:string"/>
3  </message>
4  <message name="getBeritaResponse">
5  <part name="return" type="tns:AllBerita"/>
6  </message>
```

Gambar 4.3 Elemen <message> XML FEWS
Sumber : [Perancangan]

Baris 1 pada gambar 4.3 menunjukkan sebuah *message request* dari sebuah *web service* yang memiliki parameter param bertipe data string untuk menerima masukan. Baris 4 menunjukkan sebuah *message response* dengan parameter *multiple* data kompleks yang telah didefinisikan bernama AllBerita.

3. Elemen <portType>

Elemen ini berfungsi untuk mendefinisikan sekumpulan operasi-operasi. Setiap elemen <operasi> mendefinisikan sebuah operasi dan pesan masukan atau keluaran yang berkaitan dengan operasi tersebut.

```
1  <portType name="Fews ServicePortType">
2  <operation name="getBerita">
3  <documentation>This is BTV Webservice</documentation>
4  <input message="tns:getBeritaRequest"/>
5  <output message="tns:getBeritaResponse"/>
6  </operation>
7  </portType>
```

Gambar 4.4 Elemen <portType> XML FEWS
Sumber : [Perancangan]

Elemen portType pada gambar 4.4 menunjukkan operasi request-response. Operasi tersebut terdiri dari satu pesan *input* yaitu getBeritaRequest dan satu pesan *output* yaitu getBeritaResponse.

4. Elemen <binding>

Elemen ini berfungsi untuk memetakan operasi-operasi dan pesan yang terdefinisikan pada port type ke protokol tertentu.

```
<binding name="Fews ServiceBinding" type="tns:Fews
ServicePortType">
    <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getBerita">
        <soap:operation
        soapAction="https://fews.ub.ac.id/index.php/servi
        cefews/getBerita" style="rpc"/>
        <input>
            <soap:body use="encoded"
            namespace="https://fews.ub.ac.id/index.php/ser
            vicefews"
            encodingStyle="http://schemas.xmlsoap.org/soap
            /encoding/"/>
        </input>
        <output>
            <soap:body use="encoded"
            namespace="https://fews.ub.ac.id/index.php/ser
            vicefews"
            encodingStyle="http://schemas.xmlsoap.org/soap
            /encoding/"/>
        </output>
    </operation>
</binding>
```

Gambar 4.5 Elemen <binding> XML FEWS
Sumber : [Perancangan]

Elemen binding menyediakan spesifikasi detail bagaimana sebuah operasi portType dijalankan. Binding dapat tersedia melalui berbagai transport, termasuk HTTP GET, HTTP POST, atau SOAP. Elemen binding mempunyai atribut name dan type. Web service pada FEWS menggunakan transport SOAP.

5. Elemen <service>

Elemen ini berfungsi untuk mendefinisikan sekumpulan port-port yang saling berhubungan.

```
1  <service name="Fews Service">
2      <port name="Fews ServicePort" binding="tns:Fews
3      ServiceBinding">
4          <soap:address
5          location="https://fews.ub.ac.id:443/index.php/ser
6          vicefews"/>
7      </port>
8  </service>
```

Gambar 4.6 Elemen <service> XML FEWS
Sumber : [Perancangan]

Elemen service pada gambar 4.6 menentukan operasi layanan. Elemen service memiliki atribut name yang telah didefinisikan yaitu Fews Service. Baris 2 menunjukkan port yang digunakan di dalam *web service* sedangkan Baris 4 menunjukkan alamat dari web service.

4.1.4 Daftar Kebutuhan

Daftar kebutuhan terdiri dari kebutuhan fungsional dan kebutuhan non-fungsional yang harus disediakan oleh sistem. Daftar kebutuhan ini disebut dengan *Software Requirement Specification* (SRS). Daftar kebutuhan fungsional dapat dilihat pada tabel 4.2.

Tabel 0.2 Daftar Kebutuhan Fungsional

Nomor SRS	Kebutuhan	*Use Case*
SRS_001_01	Sistem mampu menampilkan dan mengolah data atau informasi tentang berita terkini bahaya banjir.	Melihat berita terkini
SRS_001_02	Sistem mampu menampilkan dan mengolah data tentang status siaga banjir.	Melihat status siaga banjir
SRS_001_03	Sistem mampu menampilkan dan mengolah data tentang ketinggian air sungai berdasarkan satuan waktu.	Melihat ketinggian air

Sumber : [Perancangan]

Kebutuhan non-fungsional juga harus disediakan oleh sistem. Kebutuhan non-fungsional merupakan batasan layanan atau fungsi yang ditawarkan sistem seperti batasan waktu, batasan pengembangan proses, dan standarisasi. Daftar kebutuhan non-fungsional dapat dilihat pada tabel 4.3.

Tabel 0.3 Daftar Kebutuhan Nonfungsional

Parameter	Deskripsi Kebutuhan
Interoperability	Fungsi-fungsi semua operasi dalam perangkat lunak harus dapat berjalan dengan baik.

Sumber : [Perancangan]

4.2 Analisis Komponen

Proses analisis komponen diawali dengan mencari semua komponen yang berhubungan dengan pokok bahasan skripsi. Langkah selanjutnya adalah memilih beberapa komponen yang sesuai dengan kebutuhan yang diperlukan dalam aplikasi. Komponen yang dapat memenuhi kebutuhan akan langsung dipakai dalam aplikasi, sedangkan komponen yang tidak sesuai akan mengalami tahap modifikasi kebutuhan. Daftar komponen yang digunakan pada aplikasi disajikan dalam bentuk tabel 4.4.

Tabel 4.4 Daftar Komponen Aplikasi

Nama Library	Nama Komponen	Deskripsi Komponen
KSOAP 2.5.8	SOAPObject	Merupakan sebuah obyek dinamis yang sederhana yang dapat digunakan untuk membangun proses pemanggilan SOAP tanpa mengimplementasikan KvmSerializable.
	SoapSerializationEnvelope	Merupakan sebuah *class* yang menggeneralisasi *class* SOAP Envelope. Berfungsi untuk memastikan dokumen SOAP.
	HttpTransportSE	Merupakan sebuah *Java Standard Edition* (J2SE) berbasis HTTP *Transport Layer*.
AChartEngine	XYSeriesRenderer	Berfungsi untuk membuat sebuah jenis seri XY.
	XYMultipleSeriesRenderer	Berfungsi untuk membuat *multiple* jenis seri XY.
	PointStyle	Berfungsi untuk membuat tipe *chart point*.
	XYMultipleSeriesDataSet	Berfungsi untuk menampung nilai

		dataset XYSeries
	XYSeries	Berfungsi untuk menginisialisasi nilai-nilai koordinat XY pada sebuah grafik

Sumber : [Perancangan]

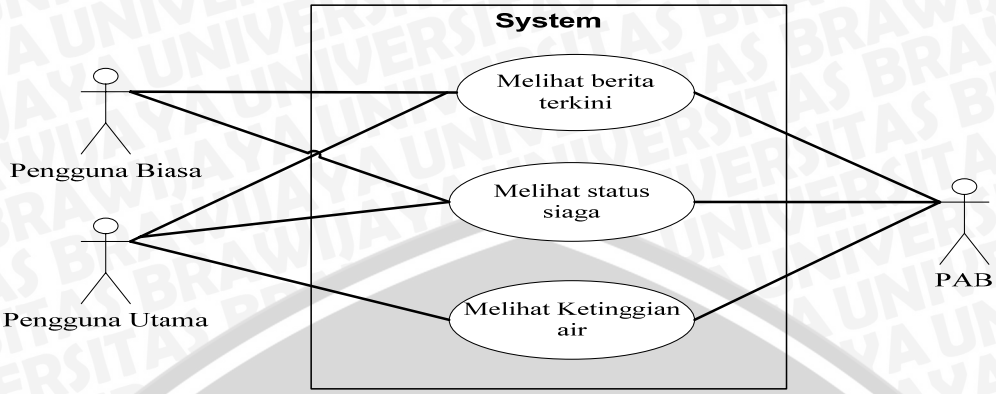
Salah satau komponen achartengine yang digunakan dalam aplikasi untuk menampilkan grafik ketinggian air suatu daerah adalah komponen XYSeries. Komponen XYSeries digunakan untuk menginisialisasi sebuah nilai bertipe data double pada koordinat X dan Y. Komponen yang seharusnya lebih tepat digunakan untuk menampilkan ketinggian air suatu daerah adalah komponen TimeSeries karena nilai pada koordinat X bertipe data date dan koordinat Y bertipe data double. Kekurangan pada komponen TimeSeries terletak pada segi tampilan koordinat X yang disebabkan oleh tidak ada kesesuaian dengan nilai yang sudah diinisialisasi. Jadi, penggunaan komponen TimeSeries akan digantikan dengan komponen XYSeries.

4.3 Perancangan dengan Komponen

Perancangan dengan komponen bertujuan untuk merancang komponen-komponen yang telah diperoleh sesuai dengan spesifikasi kebutuhan. Perancangan tersebut mencakup penambahan maupun perubahan pada komponen-komponen yang telah ditentukan. Perancangan dengan komponen terdiri dari lima tahap yaitu antara lain perancangan *use case*, perancangan aktivitas, perancangan model, perancangan interaksi dan perancangan antarmuka.

4.3.1 Perancangan *Use Case*

Diagram *use case* adalah salah satu diagram untuk memodelkan aspek perilaku sistem. Diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. *Use case* merupakan fungsionalitas dari sistem yang diinisialisasi oleh aktor. Fungsi atau fitur yang akan dibuat di dalam sistem penanda dini bahaya banjir berbasis perangkat bergerak antara lain yaitu menampilkan berita terkini mengenai bencana banjir, status siaga keadaan sungai dan ketinggian air sungai di daerah tertentu. Gambar 4.7 merupakan hasil analisis *use case* pada sistem informasi penanda dini bahaya banjir di dalam sistem operasi android.



Gambar 4.7 *Use Case Diagram*
Sumber : [Perancangan]

4.3.1.1 Skenario *Use Case*

Skenario *use case* memuat nama *use case*, aktor di dalam *use case* tersebut, tujuan dari *use case*, deskripsi global tentang *use case*, kondisi awal yang harus dipenuhi, dan kondisi akhir yang diharapkan setelah berjalannya fungsional *use case*. Skenario *use case* juga memuat beberapa penjelasan berkaitan dengan tanggapan dari sistem atas suatu tindakan yang diberikan oleh aktor. Tabel 4.5 merupakan skenario *use case* untuk melihat berita terkini tentang bahaya banjir.

Tabel 4.5 Skenario *Use Case* Melihat Berita Terkini

Skenario *Use Case*	
Nama	Melihat berita terkini
Tujuan	Untuk melihat berita terkini yang berkaitan dengan bencana banjir
Deskripsi	*Use case* ini menjelaskan pengguna dapat melihat berita terkini yang berkaitan dengan bencana banjir
Aktor	Pengguna Biasa, Pengguna Utama, PAB
Skenario Utama	
Kondisi Awal	Aplikasi menampilkan proses pertukaran data *web service*
Aksi Aktor	**Reaksi Sistem**
1. Aktor memilih parameter untuk melihat berita terkini bencana banjir 6. Aktor melihat informasi berita terkini bahaya banjir	2. Sistem melakukan pengecekan koneksi internet 3. Sistem melakukan pemanggilan *web service* 4. Sistem mengolah data *response web service* 5. Sistem menampilkan berita terkini bencana banjir
Kondisi Akhir	Sistem menampilkan berita terkini bencana banjir

Sumber : [Perancangan]

Tabel 4.6 menjelaskan skenario *use case* untuk melihat status siaga banjir suatu daerah.

Tabel 4.6 Skenario *Use Case* Melihat Status Siaga

Skenario *Use Case*	
Nama	Melihat status siaga
Tujuan	Untuk melihat status siaga keadaan sungai di sekitar daerah tempat tinggal pengguna
Deskripsi	*Use case* ini menjelaskan pengguna dapat melihat status siaga keadaan sungai di sekitar daerah tempat tinggal pengguna
Aktor	Pengguna Biasa, Pengguna Utama, PAB
Skenario Utama	
Kondisi Awal	Aplikasi menampilkan proses pertukaran data *web service*
Aksi Aktor	**Reaksi Sistem**
1. Aktor memilih parameter untuk melihat status siaga keadaan sungai 6. Aktor melihat informasi status siaga keadaan sungai	2. Sistem melakukan pengecekan koneksi internet 3. Sistem melakukan pemanggilan web service 4. Sistem mengolah data *response web service* 5. Sistem menampilkan status siaga keadaan sungai
Kondisi Akhir	Sistem menampilkan status siaga keadaan sungai

Sumber : [Perancangan]

Tabel 4.7 menjelaskan skenario *use case* untuk melihat ketinggian air sungai suatu daerah.

Tabel 4.7 Skenario *Use Case* Melihat Ketinggian Air

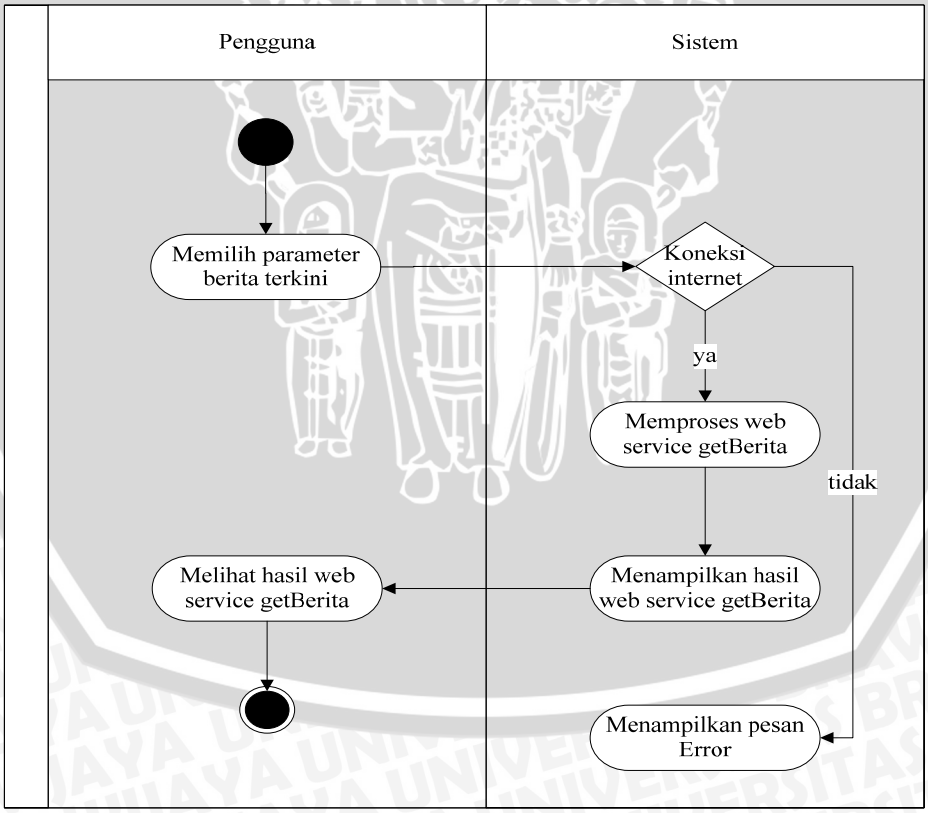
Skenario *Use Case*	
Nama	Melihat ketinggian air sungai suatu daerah
Tujuan	Untuk melihat ketinggian air sungai di daerah tertentu berdasarkan satuan waktu
Deskripsi	*Use case* ini menjelaskan pengguna dapat melihat ketinggian air sungai di daerah tertentu berdasarkan satuan waktu
Aktor	Pengguna Utama, PAB
Skenario Utama	
Kondisi Awal	Aplikasi menampilkan antarmuka utama aplikasi
Aksi Aktor	**Reaksi Sistem**
1. Aktor memasukkan *username* dan *password* 2. Aktor memilih daerah sungai 3. Aktor memilih jenis satuan waktu	4. Sistem melakukan pengecekan koneksi internet 5. Sistem melakukan autentikasi identitas pengguna

	6. Sistem menampilkan daftar sungai 7. Sistem menampilkan *form input* kalender. 8. Sistem melakukan pemanggilan web service 9. Sistem mengolah data *response web service*
Kondisi Akhir	Sistem menampilkan ketinggian air sungai suatu daerah

Sumber : [Perancangan]

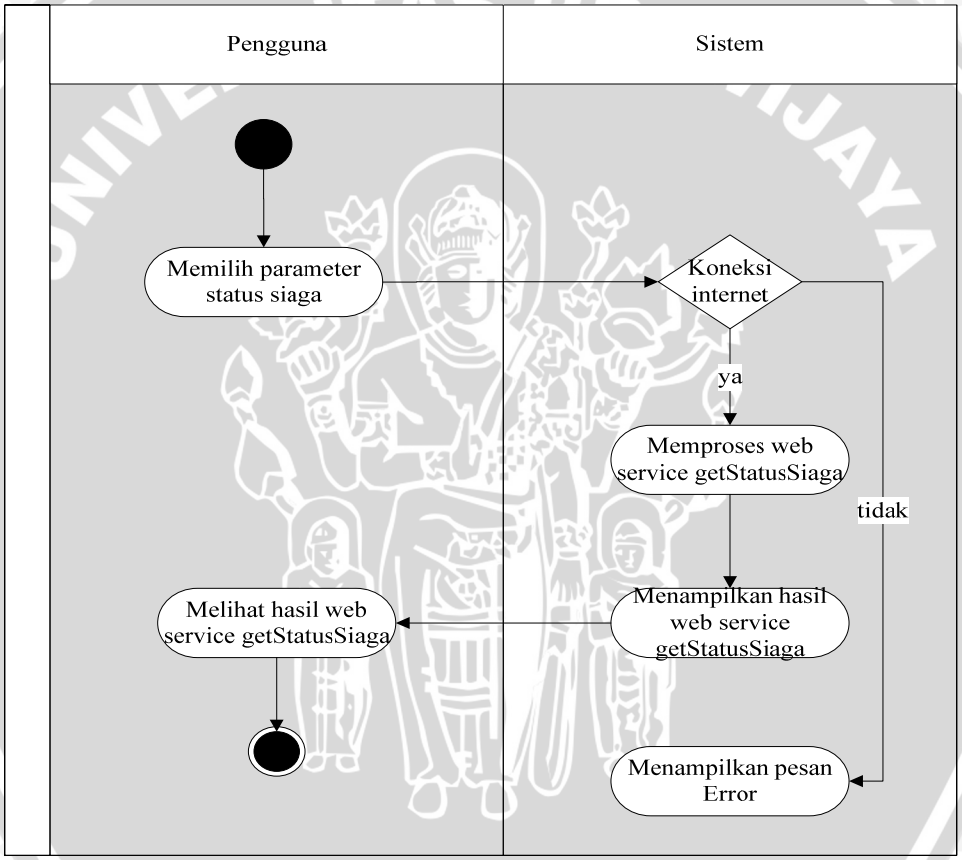
4.3.2 Perancangan Aktivitas

Activity diagram digunakan untuk menjelaskan alur proses dari penggunaan sistem. *Activity diagram* pada aplikasi disesuaikan dengan banyaknya kebutuhan fungsional dari aplikasi. Adapun *activity diagram* yang akan dibuat adalah *activity diagram* menampilkan berita terkini (Gambar 4.8), *activity diagram* menampilkan status siaga (Gambar 4.9), *activity diagram* menampilkan ketinggian air berdasarkan satuan waktu (Gambar 4.10).



Gambar 4.8 *Activity Diagram* Berita Terkini
Sumber : [Perancangan]

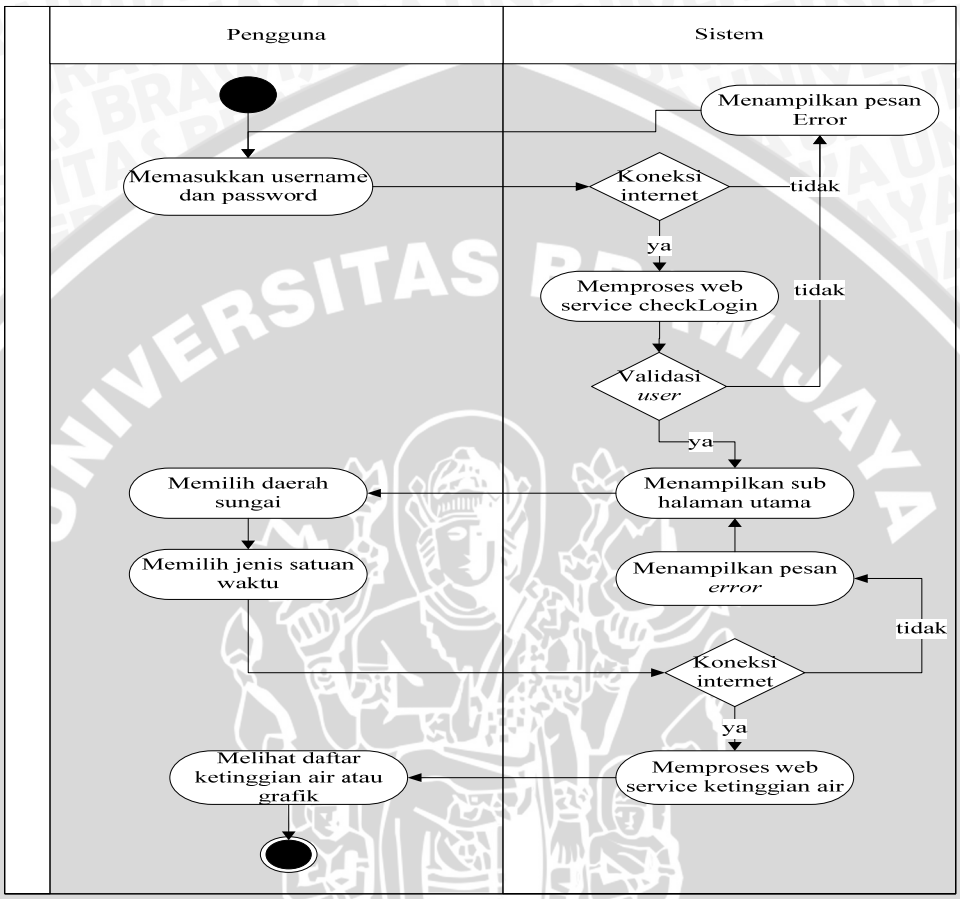
Gambar 4.8 menunjukkan proses yang dilakukan oleh sistem untuk menampilkan berita terkini mengenai bahaya banjir. Proses yang pertama dilakukan adalah pengguna memilih parameter untuk melihat berita terkini bahaya banjir kemudian sistem melakukan pengecekan terhadap koneksi internet. Apabila tidak ada koneksi internet maka sistem akan menampilkan pesan *error/*kesalahan. Jika sistem mendeteksi adanya koneksi internet maka sistem akan memanggil *web service* getBerita. Hasil dari pemanggilan tersebut kemudian akan diproses dan ditampilkan kepada pengguna.



Gambar 4.9 *Activity Diagram* Status Siaga Banjir
Sumber : [Perancangan]

Gambar 4.9 merupakan *activity diagram* status siaga banjir yang berfungsi untuk menunjukkan proses menampilkan informasi status siaga banjir sungai di daerah tertentu. Pengguna memilih parameter untuk melihat status siaga banjir. Tahap selanjutnya adalah sistem melakukan proses pengecekan koneksi internet. Jika tidak ada koneksi internet maka sistem akan menampilkan pesan

error/kesalahan. Sedangkan jika terdeteksi adanya koneksi internet maka sistem akan memanggil *web service* getStatusSiaga. Hasil pemanggilan tersebut akan diproses dan ditampilkan kepada pengguna.



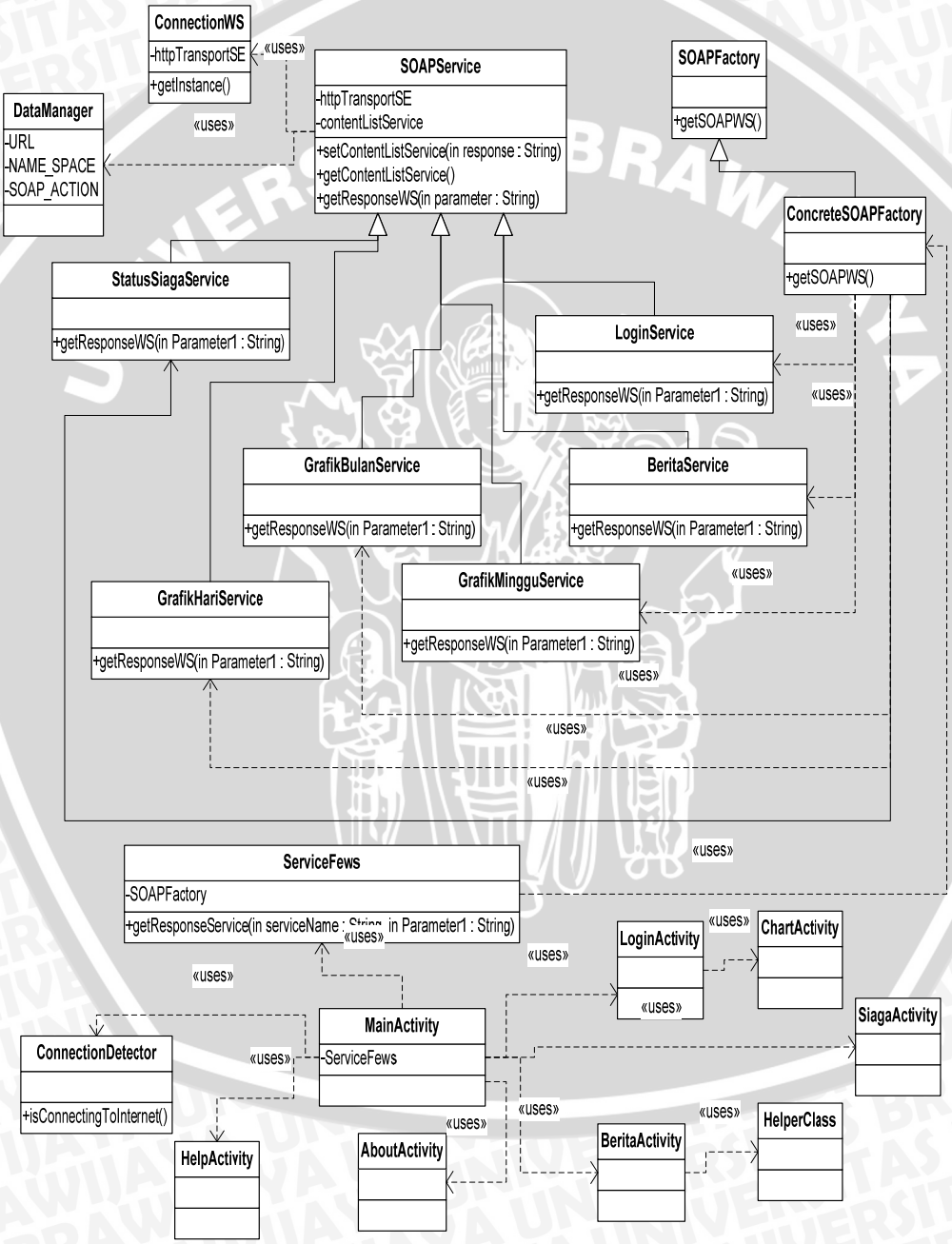
Gambar 4.10 *Activity Diagram* Ketinggian Air
Sumber : [Perancangan]

Gambar 4.10 merupakan *activity diagram* ketinggian air yang digunakan sebagai proses untuk menampilkan ketinggian air sungai suatu daerah. Sistem akan meminta pengguna untuk memasukkan *username* dan *password.* Proses selanjutnya adalah sistem akan melakukan pengecekan koneksi internet jika tidak ada koneksi maka sistem akan menampilkan pesan *error.* Apabila sistem mendeteksi adanya koneksi internet maka sistem akan memanggil *web service* checkLogin dimana *username* dan *password* sebagai parameternya. Hasil pemanggilan *web service* digunakan untuk proses validitas pengguna. Kemudian sistem akan menerima masukan dari pengguna berupa nama daerah dan satuan waktu. Masukan tersebut digunakan sebagai parameter dari proses pemanggilan

web service. Hasil dari proses pemanggilan tersebut akan ditampilkan ke pengguna berupa informasi ketinggian air suatu daerah.

4.3.3 Perancangan Model (*Class*)

Class diagram merupakan sebuah gambaran tentang sistem dan relasi-relasi yang terdapat di dalamnya.



Gambar 4.11 *Class Diagram* Penanda Dini Bahaya Banjir
Sumber : [Perancangan]

Terlihat dari gambar 4.11 bahwa *class* yang dirancang dalam aplikasi ini terdiri dari 21 *class* meliputi ConnectionWS, SOAPService, StatusSiagaService, SOAPFactory, ConcreateSOAPFactory, ServiceFews, GrafikHariService, GrafikMingguService, GrafikBulanService, LoginService, BeritaService, ConnectionDetector, MainActivity, LoginActivity, ChartActivity, SiagaActivity, AboutActivity, HelpActivity, BeritaActivity, HelperClass dan DataManager. Tabel 4.8 merupakan deskripsi dari *class* ConnectionWS beserta penjelasan dari masing-masing *method.*

Tabel 4.8 Deskripsi *Class* ConnectionWS

Nama *Class* : ConnectionWS
Deskripsi : *Class* connectionWS merupakan sebuah *class* yang digunakan untuk mengolah koneksi *client side* ke *server side.*
Nama Method : getInstance() **Fungsi :** *Method* ini berfungsi untuk menginisialisasi sebuah object yang mengatur koneksi ke *server side.*

Sumber : [Perancangan]

Tabel 4.9 menjelaskan deskripsi dari *class* SOAPService yang dilengkapi dengan penjelasan method yang dimiliki *class* tersebut beserta penjelasan sebuah *class* generalisasi dari *class* SOAPService.

Tabel 4.9 Deskripsi *Class* SOAPService

Nama *Class* : SOAPService
Deskripsi : *Class* ini merupakan *abstract class* yang digunakan untuk mendefinisikan *prototype method-method* dalam proses memanipulasi pemanggilan *web service.*
Nama Method : getResponseWS(Hashmap<String,String> parameter) **Fungsi :** *Prototype method* ini nantinya akan diimplementasikan sebagai *method* yang berfungsi untuk melakukan inisialisasi parameter dalam pemanggilan *web service.* Hasil dari pemrosesan berupa object SOAP kemudian dikonversi ke dalam bentuk String. *Method* ini mengembalikan nilai berupa tipe data String.
Nama Method : setContentListService(String response) **Fungsi :** *Prototype method* ini nantinya akan diimplementasikan sebagai *method* yang berfungsi untuk memasukkan *response* bertipe data String ke dalam sebuah object bertipe data ArrayList.
Nama Method : getContentListService() **Fungsi :** Method ini berfungsi untuk mengembalikan sebuah nilai bertipe data ArrayList.

Nama *Class Child*	Deskripsi
StatusSiagaService	*Class* ini merupakan generalisasi *class* SOAPService. *Class* ini berfungsi untuk menangani proses pemanggilan *web service* getStatusSiaga.
GrafikHariService	*Class* ini merupakan generalisasi *class* SOAPService. *Class* ini berfungsi untuk menangani proses pemanggilan *web service* getGrafikHari.
GrafikMingguService	*Class* ini merupakan generalisasi *class* SOAPService. *Class* ini berfungsi untuk menangani proses pemanggilan *web service* getGrafikMinggu.
GrafikBulanService	*Class* ini merupakan generalisasi *class* SOAPService. *Class* ini berfungsi untuk menangani proses pemanggilan *web service* getGrafikBulan.
LoginService	*Class* ini merupakan generalisasi *class* SOAPService. *Class* ini berfungsi untuk menangani proses pemanggilan *web service* checkLogin.
BeritaService	*Class* ini merupakan generalisasi *class* SOAPService. *Class* ini berfungsi untuk menangani proses pemanggilan *web service* getBerita.

Sumber : [Perancangan]

Tabel 4.10 merupakan deskripsi dari *class* SOAPFactory yang dilengkapi dengan penjelasan method yang dimiliki *class* tersebut beserta penjelasan dari *class* generalisasi *class* SOAPFactory.

Tabel 4.10 Deskripsi *Class* SOAPFactory

Nama *Class* : SOAPService	
Deskripsi : *Class* ini merupakan *abstract class* yang digunakan untuk mendefinisikan *prototype method-method* dalam pembuatan sebuah object dari SOAPService.	
Nama Method : getSOAPWS(String serviceName) **Fungsi :** *Prototype method* ini nantinya akan diimplementasikan sebagai *method* yang berfungsi untuk melakukan instansiasi dari sebuah class SOAPService. Prototype method ini mengembalikan sebuah object bertipe data SOAPService.	
Nama *Class Child*	**Deskripsi**
ConcreateSOAPFactory	*Class* ini merupakan generalisasi *class* SOAPFactory. Class ini mengimplementasi *method* getSOAPWS().

Sumber : [Perancangan]

ServiceFews merupakan sebuah *class* yang digunakan untuk proses inisialisasi pemanggilan web service. Tabel 4.11 menunjukkan deskripsi dari *class* ServiceFews dan penjelasan masing-masing *method.*

Tabel 4.11 Deskripsi *Class* ServiceFews

Nama *Class* : ServiceFews
Deskripsi : *Class* ServiceFews merupakan sebuah *class* yang digunakan untuk proses inisialisasi pemanggilan *web service.*
Nama Method : getResponseService(String serviceName, Hashmap<String,String> parameter) **Fungsi :** *Method* ini berfungsi untuk melakukan proses pemanggilan *web service*. Method ini akan mengembalikan sebuah nilai bertipe data ArrayList<Hashmap <String,String>>

Sumber : [Perancangan]

ConnectionDetector merupakan sebuah *class* yang digunakan untuk mendeteksi koneksi internet. Tabel 4.12 menunjukkan deskripsi dari *class* ConnectionDetector dan penjelasan masing-masing *method.*

Tabel 4.12 Deskripsi *Class* ConnectionDetector

Nama *Class* : ConnectionDetector
Deskripsi : *Class* ConnectionDetector merupakan sebuah *class* yang digunakan untuk mendeteksi koneksi internet.
Nama Method : isConnectingToInternet(Context context) **Fungsi :** *Method* ini berfungsi untuk mendeteksi koneksi sistem ke internet. Method ini mengembalikan sebuah nilai bertipe data boolean yaitu salah (0) dan benar (1)

Sumber : [Perancangan]

Tabel 4.13 merupakan deskripsi dari *class* MainActivity yang dilengkapi dengan penjelasan masing-masing *method. Class* MainActivity merupakan sebuah main *class* dimana BeritaActivity, SiagaActivity, LoginActivity, AboutActivity dan HelpActivity memiliki hubungan asosiasi.

Tabel 4.13 Deskripsi *Class* MainActivity

Nama *Class* : MainActivity
Deskripsi : *Class* MainActivity merupakan sebuah *class* utama yang digunakan sebagai *class* main. *Class* ini digunakan untuk mengatur tampilan dan proses input parameter.
Nama Method : onCreate(Bundle savedInstanceState) **Fungsi :** *Method* ini merupakan proses *overriding* dari *class* Activity di lingkungan operasi

sistem android sebagai proses inisialisasi.
Nama Method : onTabChanged(String tabId) **Fungsi :** *Method* ini merupakan proses overriding dari *class* Activity di lingkungan operasi sistem android yang digunakan untuk mengatasi *event* pergantian tab/menu.
Nama Method : onCreateOptionsMenu(Menu menu) **Fungsi :** *Method* ini merupakan proses overriding dari *class* Activity di lingkungan operasi sistem android yang digunakan untuk menginisialisasi sebuah menu.
Nama Method : onOptionsItemSelected(MenuItem item) **Fungsi :** *Method* ini merupakan proses overriding dari *class* Activity di lingkungan operasi sistem android yang digunakan untuk mengatasi *event* menu.
Nama Method : getErrorConnection() **Fungsi :** *Method* ini berfungsi untuk menampilkan pesan *error* ketika tidak ada koneksi internet.

Sumber : [Perancangan]

Class LoginActivity merupakan sebuah *class* yang digunakan untuk meng-*handle* proses validitas pengguna. *Class* ini berfungsi untuk menginstansiasi *object* ChartActivity. Tabel 4.14 menunjukkan deskripsi dari *class* LoginActivity yang dilengkapi dengan penjelasan masing-masing *method.*

Tabel 4.14 Deskripsi *Class* LoginActivity

Nama *Class* : LoginActivity
Deskripsi : *Class* LoginActivity merupakan sebuah *class* yang digunakan untuk meng-*handle* proses validitas pengguna.
Nama Method : onCreate(Bundle savedInstanceState) **Fungsi :** *Method* ini merupakan proses *overriding* dari *class* Activity di lingkungan operasi sistem android sebagai proses inisialisasi.
Nama Method : onClick(View arg0) **Fungsi :** *Method* ini merupakan proses overriding dari *class* Activity di lingkungan operasi sistem android yang digunakan untuk mengatasi event *click* sebuah object.

Sumber : [Perancangan]

Tabel 4.15 merupakan deskripsi dari *class* BeritaActivity yang dilengkapi dengan penjelasan masing-masing *method.*

Tabel 4.15 Deskripsi *Class* BeritaActivity

Nama *Class* : BeritaActivity
Deskripsi : *Class* BeritaActivity merupakan sebuah *class* yang digunakan untuk mengatur tampilan sistem dalam menampilkan berita terkini mengenai bahaya banjir.

Nama Method : refreshContentInfo(HelperClass helper) **Fungsi :** *Method* ini berfungsi untuk memperbaharui tampilan untuk menampilkan berita terkini bahaya banjir.
Nama Method : getErrorConnection() **Fungsi :** *Method* ini berfungsi untuk merubah tampilan ketika terjadi kesalahan koneksi internet. Method ini akan dipanggil oleh sistem ketika sistem mendeteksi tidak adanya koneksi internet.
Nama Method : getBeritaView() **Fungsi :** *Method* ini berfungsi untuk mengembalikan sebuah nilai view.

Sumber : [Perancangan]

Tabel 4.16 merupakan deskripsi dari *class* SiagaActivity yang dilengkapi dengan penjelasan masing-masing *method.*

Tabel 4.16 Deskripsi *Class* SiagaActivity

Nama *Class* : SiagaActivity
Deskripsi : *Class* SiagaActivity merupakan sebuah *class* yang digunakan untuk mengatur tampilan sistem dalam menampilkan status siaga banjir suatu daerah.
Nama Method : refreshContentSiaga(ArrayList<HashMap<String,String>> list) **Fungsi :** *Method* ini berfungsi untuk memperbaharui tampilan dalam menampilkan status siaga banjir suatu daerah.
Nama Method : getErrorConnection() **Fungsi :** *Method* ini berfungsi untuk merubah tampilan ketika terjadi kesalahan koneksi internet.
Nama Method : getSiagaView() **Fungsi :** *Method* ini berfungsi untuk mengembalikan sebuah nilai view.
Nama Method : setColorStatus(ArrayList<HashMap<String,String>> list) **Fungsi :** *Method* ini berfungsi untuk mengganti warna pada *field* status.
Nama Method : selectItemStatus(int index, String color) **Fungsi :** *Method* ini berfungsi untuk melakukan seleksi kondisi pada *field* status.

Sumber : [Perancangan]

Class ChartActivity sebuah *class* yang digunakan untuk meng-*handle request* dari pengguna untuk menampilkan ketinggian air sungai suatu daerah. Tabel 4.17 menjelaskan deskripsi dari *class* ChartActivity beserta penjelasan masing-masing *method*.

Tabel 4.17 Deskripsi *Class* ChartActivity

Nama *Class* : ChartActivity
Deskripsi : *Class* ChartActivity merupakan sebuah *class* yang digunakan untuk meng-*handle request* dari pengguna untuk menampilkan ketinggian air sungai suatu daerah.
Nama Method : onCreate(Bundle savedInstanceState) **Fungsi :** *Method* ini merupakan proses *overriding* dari *class* Activity di lingkungan operasi sistem android sebagai proses inisialisasi.
Nama Method : onClick(View arg0) **Fungsi :** *Method* ini merupakan proses overriding dari *class* Activity di lingkungan operasi sistem android yang digunakan untuk mengatasi event *click* sebuah object.

Sumber : [Perancangan]

Tabel 4.18 merupakan deskripsi dari *class* HelperClass yang dilengkapi dengan penjelasan masing-masing *method.*

Tabel 4.18 Deskripsi *Class* HelperClass

Nama *Class* : HelperClass
Deskripsi : *Class* HelperClass merupakan sebuah *class* generalisasi dari *class* ArrayAdapter yang berfungsi untuk mengatur tampilan dalam bentuk *list* di sistem operasi android
Nama Method : getView(int position, View convertView, ViewGroup parent) **Fungsi :** *Method* ini merupakan proses *overriding* dari *class* ArrayAdapter di lingkungan operasi sistem android sebagai proses inisialisasi.

Sumber : [Perancangan]

Tabel 4.19 merupakan deskripsi dari class AboutActivity yang dilengkapi dengan penjelasan masing-masing *method.* Class tersebut berfungsi untuk memuat informasi tentang aplikasi.

Tabel 4.19 Deskripsi Class AboutActivity

Nama *Class* : AboutActivity
Deskripsi : *Class* AboutActivity merupakan sebuah *class* berfungsi untuk memuat informasi tentang aplikasi.
Nama Method : onCreate(Bundle savedInstanceState) **Fungsi :** *Method* ini merupakan proses *overriding* dari *class* Activity di lingkungan operasi sistem android sebagai proses inisialisasi.

Sumber : [Perancangan]

Tabel 4.20 merupakan deskripsi dari class HelpActivity yang dilengkapi dengan penjelasan masing-masing *method.*

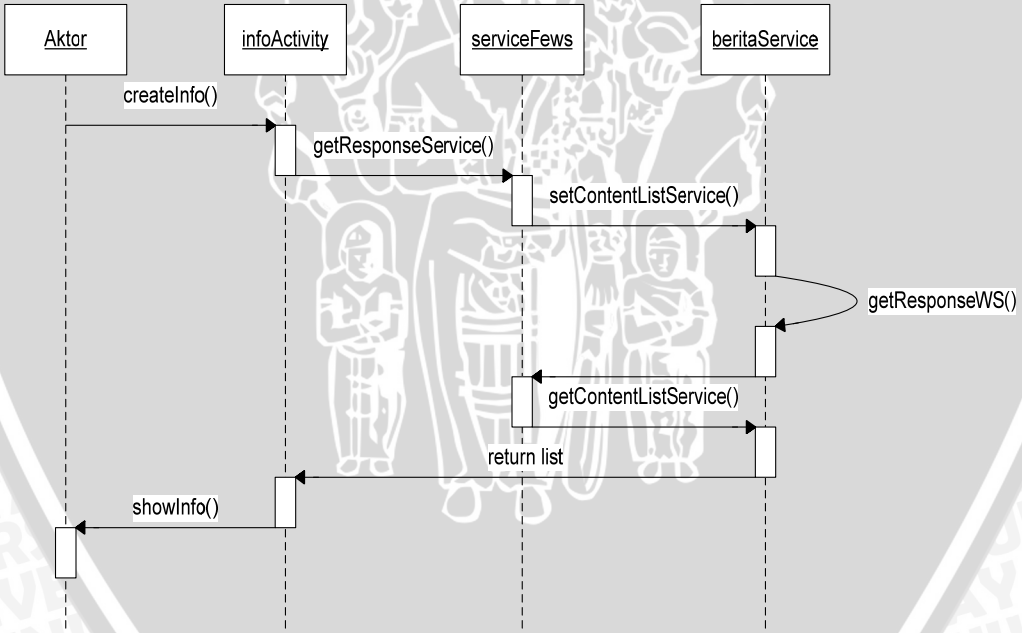
Tabel 4.20 Deskripsi Class HelpActivity

Nama *Class* : HelpActivity
Deskripsi : *Class* HelpActivity merupakan sebuah *class* berfungsi untuk memuat informasi petunjuk penggunaan aplikasi.
Nama Method : onCreate(Bundle savedInstanceState) **Fungsi :** *Method* ini merupakan proses *overriding* dari *class* Activity di lingkungan operasi sistem android sebagai proses inisialisasi.

Sumber : [Perancangan]

4.3.4 Perancangan Interaksi

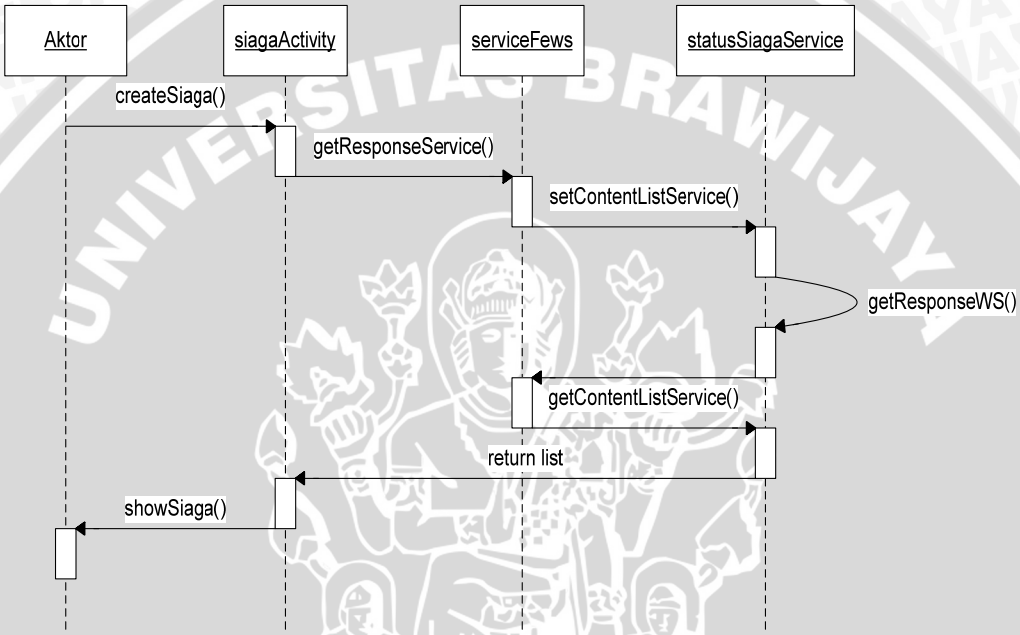
Sequence diagram digunakan untuk menggambarkan interaksi antar objek yang disusun dalam urutan waktu. *Sequence diagram* berupa skenario atau rangkaian langkah-langkah yang dilakukan dari sebuah kejadian untuk menghasilkan keluaran tertentu.



Gambar 4.12 *Sequence Diagram* Berita Terkini
Sumber : [Perancangan]

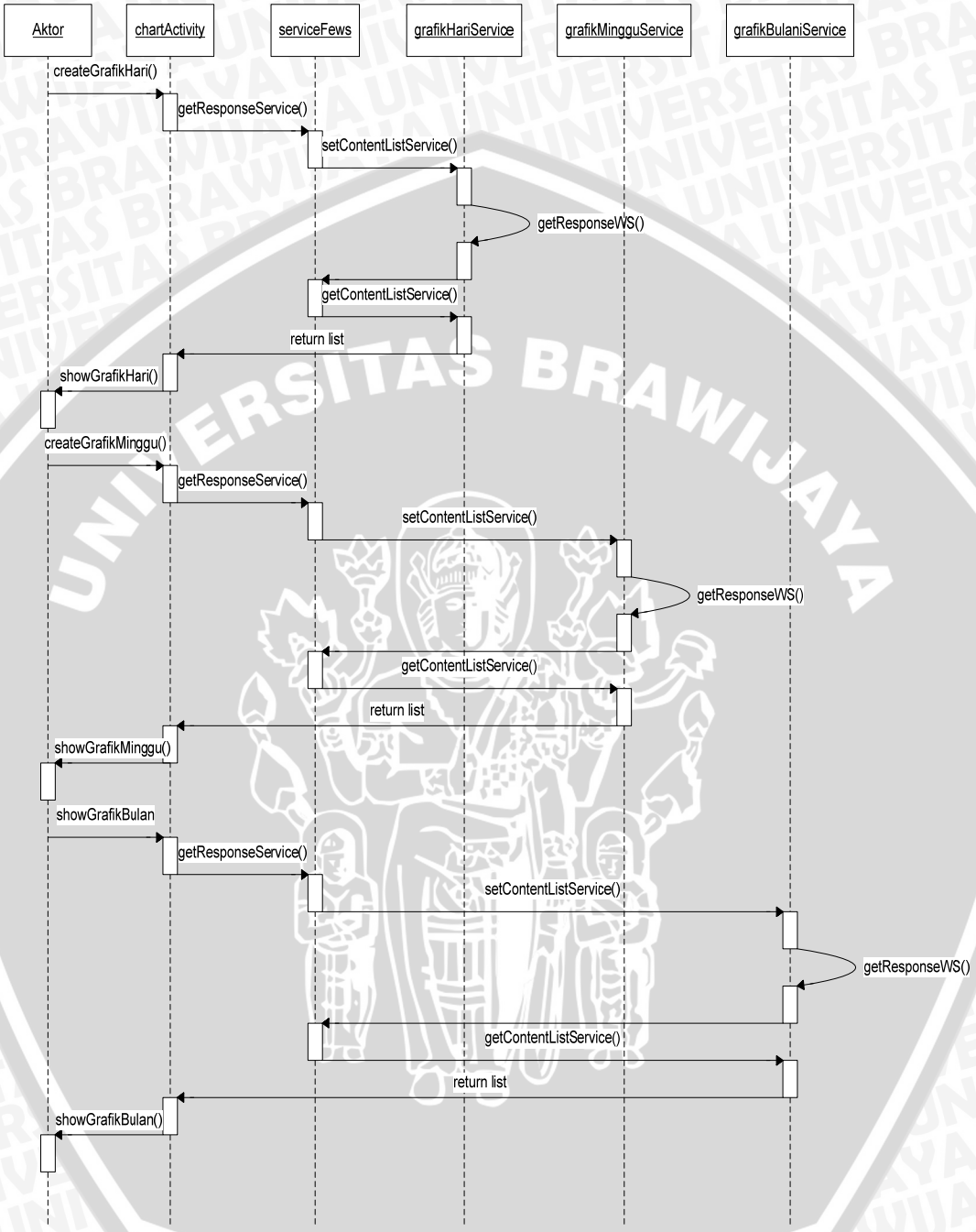
Gambar 4.12 menjelaskan interaksi antar objek dalam proses menampilkan berita terkini terkait bencana banjir. Pertama, aktor akan memanggil sebuah *method* createInfo() yang berfungsi sebagai inisialisasi dari objek infoActivity.

Objek infoActivity akan memproses sebuah *service* dengan memanfaatkan *method* getResponseService() pada *class* ServiceFews. Proses akan dilanjutkan dengan memanggil *method* setContentListService() pada *class* StatusSiagaService. Objek beritaService akan memproses *method* getResponseWS() pada objek itu sendiri. Kemudian hasil pemrosesan akan ditampilkan melalui *method* showInfo().



Gambar 4.13 *Sequence Diagram* Status Siaga
Sumber : [Perancangan]

Gambar 4.13 menjelaskan interaksi antar objek dalam proses menampilkan status siaga banjir suatu daerah. Pertama, aktor akan memanggil sebuah *method* createSiaga() yang berfungsi sebagai inisialisasi dari objek siagaActivity. Objek siagaActivity akan memproses sebuah *service* dengan memanfaatkan *method* getResponseService() pada *class* ServiceFews. Proses akan dilanjutkan dengan memanggil *method* setContentListService() pada *class* StatusSiagaService. Objek statusSiagaService akan memproses *method* getResponseWS() pada objek itu sendiri. Kemudian hasil pemrosesan akan ditampilkan melalui *method* showSiaga().



Gambar 4.14 *Sequence Diagram* Ketinggian Air
Sumber : [Perancangan]

Gambar 4.14 merupakan interaksi antar objek dalam proses menampilkan ketinggian air sungai suatu daerah. Ada 3 jenis tampilan ketinggian air sungai yaitu berdasarkan satuan waktu harian, mingguan dan bulanan. Satuan waktu

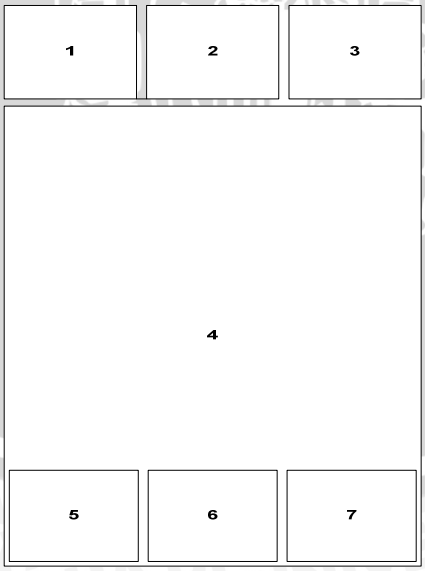
tersebut merupakan sebuah parameter dalam pemrosesan sebuah *service*. Pertama, aktor akan memanggil sebuah *method* createGrafikHari() yang berfungsi sebagai inisialisasi dari objek chartActivity. Objek chartActivity akan memproses sebuah *service* dengan memanfaatkan *method* getResponseService() pada *class* ServiceFews. Proses akan dilanjutkan dengan memanggil *method* setContentListService() pada *class* GrafikHariService. Objek grafikHariService akan memproses *method* getResponseWS() pada objek itu sendiri. Kemudian hasil pemrosesan akan ditampilkan melalui *method* showGrafikHari(). Proses kedua dan ketiga hampir sama dengan proses pertama. Perbedaannya terletak pada pemrosesan pemanggilan service.

4.3.5 Perancangan Antarmuka

Pembuatan kerangka aplikasi dilakukan dengan merancang sebuah antarmuka. Tujuan dari perancangan antarmuka adalah untuk membuat sebuah tampilan dari aplikasi yang sederhana agar memudahkan pengguna dalam mengoperasikan aplikasi.

4.3.5.1 Perancangan Halaman Utama

Gambar 4.15 merupakan antarmuka halaman utama dari aplikasi Flood Early Warning System pada perangkat bergerak di dalam sistem operasi android.



Gambar 4.15 Antarmuka Halaman Utama
Sumber : [Perancangan]

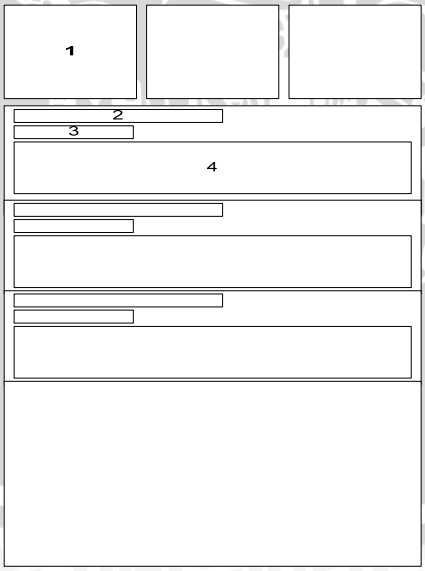
Halaman utama menampilkan beberapa menu yaitu menu untuk menampilkan berita terkini, status siaga banjir, ketinggian air, memperbaharui tampilan, tentang perangkat lunak dan bantuan penggunaan perangkat lunak. Tabel 4.21 merupakan keterangan dari antarmuka halaman utama.

Tabel 4.21 Keterangan Antarmuka Halaman Utama

Nomor	Nama	Keterangan
1	Menu berita	Menu yang dapat digunakan pengguna untuk melihat berita terkini bahaya banjir.
2	Menu siaga	Menu yang dapat digunakan pengguna untuk melihat status siaga banjir.
3	Menu ketinggian air	Menu yang dapat digunakan pengguna untuk melihat ketinggian air sungai suatu daerah.
4	Isi	Bagian ini memuat isi dari masing-masing menu.
5	Menu pembaharuan	Menu yang digunakan untuk memperbaharui masing-masing *content* (isi).
6	Menu tentang	Menu yang digunakan untuk melihat informasi tentang aplikasi.
7	Menu bantuan	Menu yang digunakan pengguna sebagai petunjuk untuk menggunakan aplikasi.

Sumber : [Perancangan]

4.3.5.2 Perancangan Halaman Berita Terkini



Gambar 4.16 Antarmuka Halaman Berita Terkini
Sumber : [Perancangan]

Halaman berita terkini menampilkan beberapa informasi mengenai berita terkini mengenai bahaya banjir. Penjelasan bagian-bagian antarmuka halaman berita terkini dijelaskan pada tabel 4.22.

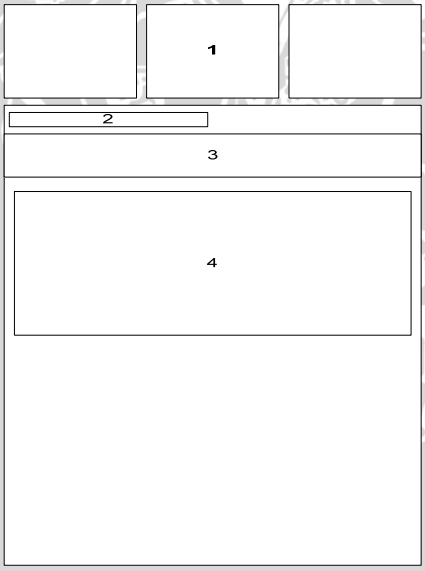
Tabel 4.22 Keterangan Antarmuka Halaman Berita Terkini

Nomor	Nama	Keterangan
1	*Header*	Bagian ini merupakan sebuah menu yang akan digunakan oleh pengguna untuk melihat berita terkini mengenai bahaya banjir dan dilengkapi dengan *icon* gambar.
2	Judul Berita	Bagian ini berupa judul dari sebuah berita.
3	Tanggal	Bagian ini menunjukkan waktu dari sebuah berita.
4	Isi	Bagian ini memuat isi dari sebuah berita.

Sumber : [Perancangan]

4.3.5.3 Perancangan Halaman Status Siaga

Halaman status siaga menampilkan beberapa informasi mengenai status siaga dan tinggi muka air keadaan sungai di daerah tertentu.



Gambar 4.17 Antarmuka Halaman Status Siaga
Sumber : [Perancangan]

Gambar 4.17 merupakan rancangan antarmuka halaman status siaga banjir suatu daerah. Penjelasan bagian-bagian antarmuka halaman status siaga dijelaskan pada tabel 4.23.

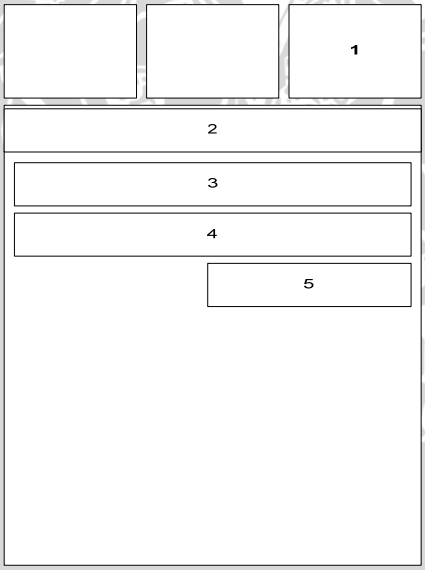
Tabel 4.23 Keterangan Antarmuka Halaman Status Siaga

Nomor	Nama	Keterangan
1	*Header*	Bagian ini merupakan sebuah menu yang akan digunakan oleh pengguna untuk melihat status siaga keadaan sungai di daerah tertentu dan dilengkapi dengan *icon* gambar.
2	Tanggal	Bagian ini menunjukkan waktu pengiriman data dari *server side*.
3	Judul	Bagian ini merupakan judul dari sebuah tabel yang berisi status siaga keadaan sungai masing-masing daerah.
4	Tabel Status Siaga	Bagian ini memuat daftar daerah, tinggi muka air dan status siaga suatu daerah.

Sumber : [Perancangan]

4.3.5.4 Perancangan Halaman Ketinggian Air

Antarmuka halaman ketinggian air dibagi menjadi tiga tampilan yaitu antarmuka proses validitas pengguna, antarmuka pencarian ketinggian air sungai suatu wilayah dan antarmuka yang digunakan untuk menampilkan hasil pencarian.



Gambar 4.18 Antarmuka Halaman Validitas Pengguna
Sumber : [Perancangan]

Penjelasan bagian-bagian antarmuka halaman validitas pengguna dijelaskan pada tabel 4.24.

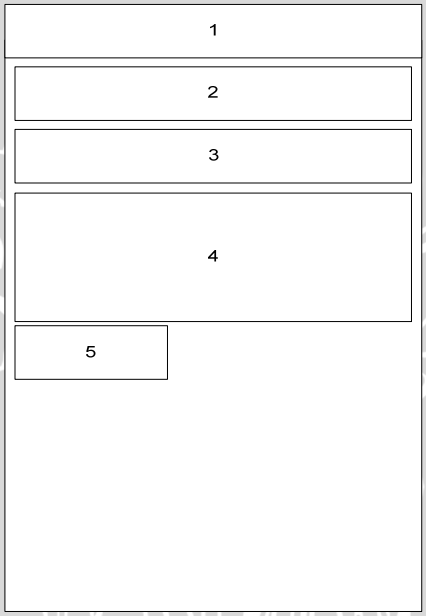
Tabel 4.24 Keterangan Antarmuka Halaman Validitas Pengguna

Nomor	Nama	Keterangan
1	*Header*	Bagian ini merupakan sebuah menu yang akan

		digunakan oleh pengguna untuk verifikasi identitas pengguna dan dilengkapi dengan *icon* gambar.
2	Judul	Bagian ini merupakan judul *form input* validitas pengguna.
3	*Field Username*	Digunakan untuk menerima masukan *username* pengguna.
4	*Field Password*	Digunakan untuk menerima masukan *password* pengguna.
5	Tombol Navigasi	Digunakan untuk memproses validitas pengguna.

Sumber : [Perancangan]

Gambar 4.19 menjelaskan rancangan antarmuka untuk proses pencarian ketinggian air sungai suatu daerah.



Gambar 4.19 Antarmuka Proses Pencarian Ketinggian Air
Sumber : [Perancangan]

Penjelasan mengenai bagian-bagian antarmuka pada gambar 4.19 akan dijelaskan pada tabel 4.25.

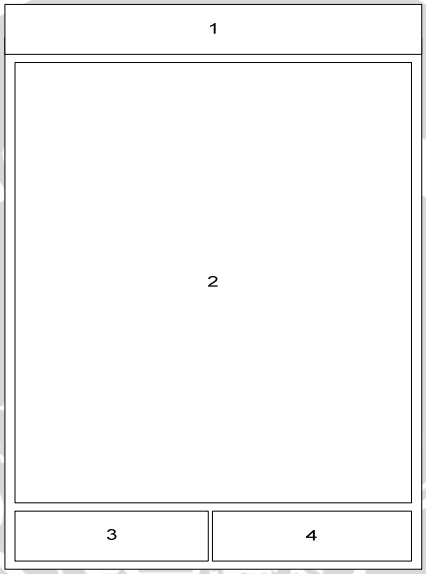
Tabel 4.25 Keterangan Antarmuka Proses Pencarian Ketinggian Air

Nomor	Nama	Keterangan
1	*Header*	Berisi judul dari menu ketinggian air.
2	Daerah	Bagian ini berisi daftar daerah sungai.
3	Waktu	Bagian ini berisi jenis satuan waktu
4	*Form Input*	Digunakan untuk memasukkan sebuah tanggal yang akan diproses sebagai parameter dalam

		pencarian ketinggian air sungai suatu daerah.
5	Tombol Navigasi	Digunakan untuk memproses pencarian ketinggian air sungai suatu daerah.

Sumber : [Perancangan]

Gambar 4.20 merupakan halaman antarmuka yang akan ditampil ketika pengguna sudah melakukan proses validitas pengguna.



Gambar 4.20 Antarmuka Hasil Pencarian Ketinggian Air
Sumber : [Perancangan]

Penjelasan mengenai bagian-bagian antarmuka pada gambar 4.20 akan dijelaskan pada tabel 4.26.

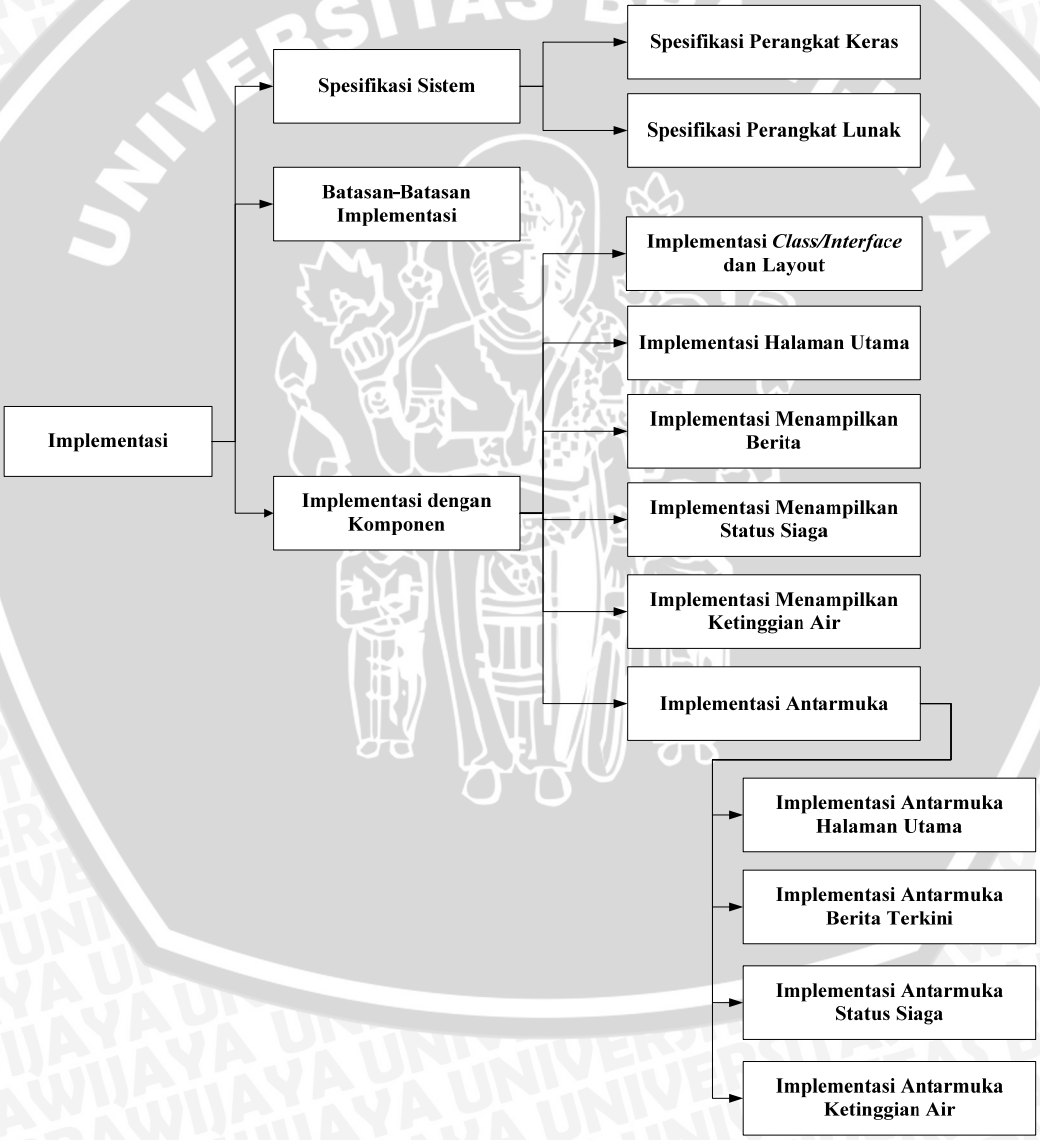
Tabel 4.26 Keterangan Antarmuka Hasil Pencarian Ketinggian Air

Nomor	Nama	Keterangan
1	*Header*	Berisi judul dari menu ketinggian air.
2	Hasil	Bagian ini berisi daftar ketinggian air yang dilengkapi dengan tinggi air, tanggal dan daerah.
3	Tombol Navigasi	Digunakan untuk kembali ke tampilan sebelumnya
4	Tombol Navigasi	Digunakan untuk melihat ketinggian air dalam bentuk grafik.

Sumber : [Perancangan]

BAB V
IMPLEMENTASI

Bab ini membahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak yang dibuat. Pembahasan terdiri atas penjelasan tentang spesifikasi sistem, batasan – batasan dalam implementasi, implementasi tiap *class* pada *file* program, implementasi algoritma, dan implementasi antarmuka. Tahap-tahap pembahasan implementasi yang dikerjakan digambarkan pada gambar 5.1 berikut ini.



Gambar 5.1 Diagram Pohon Implementasi
Sumber : [Implementasi]

5.1 Spesifikasi Sistem

Hasil analisis kebutuhan yang telah diuraikan pada bab 4 menjadi acuan untuk melakukan implementasi menjadi sebuah sistem yang dapat berfungsi sesuai dengan kebutuhan. Spesifikasi sistem diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras meliputi spesifikasi prosesor, memori, hardisk, mother board, dan kartu grafis dijelaskan pada tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras Telepon Cerdas Android

Nama Komponen	Spesifikasi
Prosesor	1 GHz Cortex-A9
Memori (RAM)	512 MB
Memori	Internal : 512 MB Eksternal : 8 GB
Chipset	MTK 6575
Kartu Grafis	PowerVR SGX531
Ukuran layar	320 x 480 piksel

Sumber : [Implementasi]

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses pengembangan perangkat lunak *Flood Early Warning System* dijelaskan pada tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak Telepon Cerdas Android

Nama Komponen	Spesifikasi
Sistem operasi	Android OS, v2.3.6 (Gingerbread)

Sumber : [Implementasi]

5.2 Batasan-Batasan Implementasi

Beberapa batasan-batasan dalam mengimplementasikan sistem adalah sebagai berikut:

a. Pembuatan aplikasi *Flood Early Warning System* berbasis perangkat bergerak ini dikerjakan dengan bahasa Pemrograman Java di lingkungan sistem operasi android.
b. Aplikasi ini dibangun dengan menggunakan teknologi *web service* berbasis SOAP.

c. Tingkat akurasi penerimaan data dari *server side* tergantung kecepatan akses internet telepon cerdas pengguna.

5.3 Implementasi dengan Komponen

Perangkat lunak *Flood Early Warning System* berbasis perangkat bergerak memiliki beberapa fitur yang terbagi dalam beberapa fungsi. Beberapa fitur tersebut diproses menggunakan *resource* pada komponen-komponen yang telah ditentukan. Pada penulisan skripsi ini mencantumkan beberapa *class* hasil implementasi dari perancangan pada bab IV sub bab Perancangan Model (*Class*). Implementasi dengan komponen ini akan direpresentasikan dalam bentuk *code* dengan bahasa pemrograman Java di lingkungan sistem operasi android. Implementasi dengan komponen juga menjelaskan beberapa *class* yang merepresentasikan masing-masing fitur terkait dengan hasil implementasi antarmuka sistem dari perancangan pada bab IV sub bab Perancangan Antarmuka.

5.3.1 Implementasi *Class*/*Interface* dan *Layout*

Implementasi *class*/*interface* merupakan proses implementasi dari hasil perancangan pada Bab IV halaman 48 yang direalisasikan pada sebuah *file* program dengan ekstensi *.java. Tabel 5.3 menjelaskan mengenai pasangan antara *class* dengan *file* program yang digunakan untuk mengimplementasikannya.

Tabel 5.3 Implementasi *Class* atau *Interface*

No.	*Package*	Nama *Class* atau *Interface*	Nama *File* Program
1	fews.constant	DataManager	DataManager.java
2	fews.factory	BeritaService	BeritaService.java
3	fews.factory	ConcreateSOAPFactory	ConcreateSOAPFactory.java
4	fews.factory	GrafikBulanService	GrafikBulanService.java
5	fews.factory	GrafikHariService	GrafikHariService.java
6	fews.factory	GrafikMingguService	GrafikMingguService.java
7	fews.factory	LoginService	LoginService.java
8	fews.factory	ServiceFews	ServiceFews.java
9	fews.factory	SOAPFactory	SOAPFactory.java
10	fews.factory	SOAPService	SOAPService.java
11	fews.factory	StatusSiagaService	StatusSiagaService.java
12	fews.main	SiagaActivity	SiagaActivity.java
13	fews.main	BeritaActivity	BeritaActivity.java
14	fews.main	ChartActivity	ChartActivity.java
15	fews.main	LoginActivity	LoginActivity.java
16	fews.main	MainActivity	MainActivity.java

17	fews.main	AboutActivity	AboutActivity.java
18	fews.main	HelpActivity	HelpActivity.java
19	fews.singleton	ConnectionWS	ConnectionWS.java
20	fews.util	HelperClass	HelperClass.java
21	fews.util	ConnectionDetector	ConnectionDetector.java

Sumber : [Implementasi]

Tabel 5.4 merupakan implementasi *layout* yang akan digunakan di dalam aplikasi. *Layout* ini berfungsi untuk mengatur tampilan aplikasi di dalam sistem operasi android yang akan digunakan pengguna untuk berinteraksi dengan sistem. Pada operasi sistem android, *layout* dibuat menggunakan *tag-tag* XML dengan ekstensi *.xml.

Tabel 5.4 Implementasi *Layout*

No.	Folder	Nama *Layout*	Nama File Program
1	/res/layout	chart_main	chart_main.xml
2	/res/layout	grafik_day	grafik_day.xml
3	/res/layout	grafik_listitem	grafik_listitem.xml
4	/res/layout	grafik_month	grafik_month.xml
5	/res/layout	grafik_week	grafik_week.xml
6	/res/layout	icon_chart	icon_chart.xml
7	/res/layout	icon_info	icon_info.xml
8	/res/layout	icon_siaga	icon_siaga.xml
9	/res/layout	login_form	login_form.xml
10	/res/layout	main	main.xml
11	/res/layout	subinfo	subinfo.xml
12	/res/layout	about	about.xml
13	/res/layout	help	help.xml
14	/res/layout	info	info.xml
15	/res/layout	siaga	siaga.xml

Sumber : [Implementasi]

5.3.2 Implementasi Halaman Utama

Halaman utama pada perangkat lunak *Flood Early Warning System* berbasis perangkat bergerak menyediakan beberapa menu informasi. Menu informasi tersebut antara lain berita terkini bahaya banjir, status siaga banjir dan ketinggian air sungai suatu daerah. Serta memiliki beberapa menu penunjang seperti bantuan penggunaan perangkat, tentang perangkat dan update informasi. Implementasi cuplikan *code* halaman utama dapat dilihat pada gambar 5.2.

```
1  public void onCreate(Bundle savedInstanceState) {
2        changedTab();
3        if (ConnectionDetector.isConnectingToInternet
4        (getApplicationContext())) {
5              new LoadService().execute();
6        }
7        else
8        {
9              berita.getErrorConnection();
10             siaga.getErrorConnection();
11             getErrorConnection();
12             isConnected = false;
13       }
14 }
15 public void changedTab() {
16       if (ConnectionDetector.isConnectingToInternet
17       (getApplicationContext())) {
18
19       } else {
20             if(!isConnected){
21                   getErrorConnection();
22                   berita.getErrorConnection();
23                   siaga.getErrorConnection();
24             }
25       }
26       for (int i = 0; i < getTabHost().getTabWidget()
27       .getChildCount(); i++) {
28             getTabHost().getTabWidget().getChildAt(i)
29             .setBackgroundColor(Color.WHITE);
30       }
31
32       getTabHost().getTabWidget().setBackgroundColor(Color.GRAY);
33       getTabHost().getTabWidget().setStripEnabled(false);
34 }
35
36 private class LoadService extends AsyncTask<Void, Void, Void>
37 {
38 protected void onPreExecute()
39 {
40       progressDialog = ProgressDialog.show(MainActivity.this, "",
41       "Silakan tunggu...", true, false);
42 }
43 protected Void doInBackground(Void... params)
44 {
45       synchronized (this) {
46             serviceBerita = serviceFews.getResponseService
47             ("Berita", null);
48             serviceSiaga = serviceFews.getResponseService
49             ("StatusSiaga",null);
50       }
51       return null;
52 }
53 protected void onPostExecute(Void result)
54 {
55       progressDialog.dismiss();
56       berita.refreshContentInfo(new HelperClass
57       (getApplicationContext(),R.layout.subinfo,serviceBerita,
58       "Berita"));
59       pBerita.removeAllViewsInLayout();
```

```
60          pBerita.addView(berita.beritaView);
61          siaga.refreshContentSiaga(serviceSiaga);
62          pSiaga.removeAllViewsInLayout();
63          pSiaga.addView(siaga.getSiagaView());
64          }
65  }
66  public void getErrorConnection()
67  {
68          Toast.makeText(tabHost.getCurrentTabView().getContext(),
69          "Tidak ada sambungan. Silakan coba lagi nanti.",
70          Toast.LENGTH_SHORT).show();
71  }
```

Gambar 5.2 Implementasi Cuplikan *Code* Halaman Utama
Sumber : [Implementasi]

Gambar 5.2 merupakan cuplikan *code* dari *class* MainActivity yang berfungsi sebagai *main class* aplikasi. Adapun Hasil implementasi antarmuka untuk menampilkan halaman utama dapat dilihat pad sub bab 5.3.6.1 antarmuka halaman utama. Penjelasan implementasi cuplikan *code* halaman utama pada gambar 5.2 yaitu:

1. Baris 3-13 digunakan untuk mendeteksi koneksi internet ketika aplikasi pertama kali dijalankan.
2. Baris 15-34 digunakan untuk mengatur warna dan isi dari masing-masing tab menu ketika terjadi sebuah *event*.
3. Baris 36-65 merupakan sebuah *class* yang berfungsi untuk melakukan pemanggilan operasi *web service* yang memiliki tiga buah *method* yaitu *method* onPreExecute() berfungsi untuk inisialisasi sebuah *progress* pemanggilan operasi *web service*. *Method* doInBackground(Void… params) berfungsi untuk melakukan pemanggilan operasi *web service*. *Method* onPostExecute(Void result) berfungsi untuk mengolah data dari hasil pemanggilan operasi *web service*.
4. Baris 66-71 merupakan sebuah *method* yang berfungsi untuk menampilkan pesan *error* ketika tidak ada koneksi internet.

5.3.3 Implementasi Menampilkan Berita Terkini

Proses menampilkan berita terkini dilakukan dengan memanggil operasi dari *web service*. Operasi menampilkan berita terkini merupakan proses implementasi dari perancangan dengan komponen. Implementasi cuplikan *code* menampilkan berita terkini dapat dilihat pada gambar 5.3.

```
1  public void refreshContentInfo(HelperClass helper)
2  {
3        titleInfo.setVisibility(View.VISIBLE);
4        errorInfo.setVisibility(View.INVISIBLE);
5        listInfo.setAdapter(helper);
6  }
7  public View getBeritaView() {
8        return beritaView;
9  }
10 public void getErrorConnection(){
11       errorInfo.setVisibility(View.VISIBLE);
12 }
```

Gambar 5.3 Implementasi Cuplikan *Code* Menampilkan Berita Terkini
Sumber : [Implementasi]

Gambar 5.3 merupakan cuplikan *code* dari *class* BeritaActivity yang berfungsi untuk mengolah dan menampilkan data *web service* mengenai berita terkini bahaya banjir. Hasil implementasi antarmuka untuk menampilkan berita terkini dapat dilihat pad sub bab 5.3.6.2 antarmuka berita terkini. Penjelasan implementasi cuplikan *code* menampilkan berita terkini pada gambar 5.3 yaitu:

1. Baris 1-6 merupakan sebuah *method* yang digunakan untuk menampilkan hasil pemanggilan operasi *web service* ke dalam sebuah *list* dengan parameter *class* HelperClass.
2. Baris 7-9 merupakan sebuah *method* yang mengembalikan sebuah nilai berupa View.
3. Baris 10-12 berfungsi untuk menampilkan pesan *error* ketika tidak ada koneksi internet.

5.3.4 Implementasi Menampilkan Status Siaga

Proses menampilkan status siaga dilakukan dengan memanggil operasi dari *web service*. Operasi menampilkan status siaga merupakan proses implementasi dari perancangan dengan komponen. Implementasi cuplikan *code* menampilkan status siaga dapat dilihat pada gambar 5.4.

```
1  public void refreshContentSiaga(ArrayList<HashMap<String,
2  String>> listSiaga) {
3        errorSiaga.setVisibility(View.INVISIBLE);
4        siagaContent.setVisibility(View.VISIBLE);
5
6        wilayah1.setText(listSiaga.get(0).get("Wilayah"));
7        status1.setText(listSiaga.get(0).get("Status"));
8        tinggiAir1.setText(listSiaga.get(0).get("TinggiAir"));
9
10       wilayah2.setText(listSiaga.get(1).get("Wilayah"));
11       status2.setText(listSiaga.get(1).get("Status"));
12       tinggiAir2.setText(listSiaga.get(1).get("TinggiAir"));
```

```
13
14          wilayah3.setText(listSiaga.get(2).get("Wilayah"));
15          status3.setText(listSiaga.get(2).get("Status"));
16          tinggiAir3.setText(listSiaga.get(2).get("TinggiAir"));
17
18          wilayah4.setText(listSiaga.get(3).get("Wilayah"));
19          status4.setText(listSiaga.get(3).get("Status"));
20          tinggiAir4.setText(listSiaga.get(3).get("TinggiAir"));
21
22          timeStamp = new Timestamp(Calendar.getInstance()
23          .getTime().getTime());
24          tanggal.setText(timeStamp.toString());
25  }
26
27  public View getSiagaView()
28  {
29          return siagaView;
30  }
31
32  public void getErrorConnection()
33  {
34          errorSiaga.setVisibility(View.VISIBLE);
35  }
```

Gambar 5.4 Implementasi Cuplikan *Code* Menampilkan Status Siaga
Sumber : [Implementasi]

Gambar 5.4 merupakan cuplikan *code* dari *class* SiagaActivity yang berfungsi untuk mengolah dan menampilkan data *web service* mengenai status siaga suatu daerah. Hasil implementasi antarmuka untuk menampilkan status siaga suatu daerah dapat dilihat pad sub bab 5.3.6.3 antarmuka status siaga. Penjelasan implementasi cuplikan *code* menampilkan status siaga suatu daerah pada gambar 5.4 yaitu:

1. Baris 1-25 merupakan sebuah *method* yang digunakan untuk menampilkan data hasil pemanggilan operasi *web service* mengenai status siaga suatu daerah. Data tersebut akan ditampilkan ke dalam sebuah TextView.
2. Baris 27-30 merupakan *method* yang mengembalikan sebuah nilai berupa View.
3. Baris 32-35 berfungsi untuk mengatur tampilan ketika terjadi kesalahan koneksi internet.

5.3.5 Implementasi Menampilkan Ketinggian Air

Proses menampilkan berita terkini dilakukan dengan memanggil operasi dari *web service*. Operasi menampilkan ketinggian air merupakan proses implementasi dari perancangan dengan komponen. Gambar 5.5 menunjukkan implementasi cuplikan *code* validitas pengguna yang merupakan bagian dari implementasi

menampilkan ketinggian air. Adapun Hasil implementasi antarmuka untuk menampilkan ketinggian air sungai suatu daerah dapat dilihat pad sub bab 5.3.6.4 antarmuka ketinggian air.

```
1  public void onClick(View arg0)
2  {
3     if (arg0.equals(login))
4     {
5         if(!username.getText().toString().equals("")&&
6         !password.getText()
7         .toString().equals(""))
8         {
9               mapLogin.put("username",username
10              .getText().toString());
11              mapLogin.put("password",password
12              .getText().toString());
13              if(ConnectionDetector
14              .isConnectingToInternet
15              (getApplicationContext()))
16              {
17                    new LoadView().execute();
18              }
19              else
20              {
21                    username.setText("");
22                    password.setText("");
23                    Toast.makeText(LoginActivity.this, "Tidak
24                    ada sambungan. Silakan coba lagi
25                    nanti.",Toast.LENGTH_SHORT) .show();
26              }
27        }
28        else
29        {
30             Toast.makeText(this, "Silakan, isikan username and
31             password",Toast.LENGTH_LONG) .show();
32        }
33    }
34 }
35 private class LoadView extends AsyncTask<Void, Void, Void>
36 {
37 protected void onPreExecute()
38 {
39       progressDialog=ProgressDialog.show (LoginActivity.this,
40       "","Please wait...");
41 }
42 protected Void doInBackground(Void... params)
43 {
44       responseService = serviceFews
45       .getResponseService("Login",mapLogin);
46       return null;
47 }
48 protected void onPostExecute(Void result)
49 {
50       progressDialog.dismiss();
51       if (!responseService.get(0) .get("result").equals(""))
52       {
53            username.setText("");
```

```
54              password.setText("");
55              if(responseService.get(0).get("result")
56              .equals("1"))
57              {
58                   startActivity(new Intent(LoginActivity .this,
59                   ChartActivity.class));
60              }
61              else
62              {
63                   Toast.makeText(LoginActivity.this, "Username
64                   and password failed!",Toast.LENGTH_LONG)
65                   .show();
66              }
67         }
68         else
69               {
70               username.setText("");
71               password.setText("");
72               Toast.makeText(LoginActivity.this,"Tidak ada
73               sambungan. Silakan coba lagi nanti.",
74               Toast.LENGTH_LONG).show();
75         }
76   }
```

Gambar 5.5 Implementasi Cuplikan *Code* Validitas Pengguna
Sumber : [Implementasi]

Gambar 5.5 merupakan cuplikan *code* dari *class* LoginActivity yang berfungsi untuk memproses validitas pengguna. Adapun penjelasan dari gambar 5.5 sebagai berikut:

1. Baris 1-34 merupakan sebuah *method* yang digunakan ketika ada sebuah *event* pada tombol login. Baris 3-33 merupakan sebuah kondisi yang digunakan untuk mengkonfirmasi pengguna ketika *username* dan *password* belum diisikan.
2. Baris 35-76 merupakan sebuah *class* yang berfungsi untuk melakukan pemanggilan operasi *web service* yang memiliki tiga buah *method* yaitu *method* onPreExecute() berfungsi untuk inisialisasi sebuah *progress* pemanggilan operasi *web service. Method* doInBackground(Void… params) berfungsi untuk melakukan pemanggilan operasi *web service*. *Method* onPostExecute(Void result) berfungsi untuk melakukan pencocokan *username* dan *password* dengan hasil pemanggilan operasi *web service.*

Gambar 5.6 merupakan cuplikan code *class* ChartActivity yang berfungsi untuk mengolah dan menampilkan data *web service* mengenai ketinggian air sungai suatu daerah.

```
1  public void onClick(View v) {
2        if (v == buttonDay || v == buttonMonth ||
3        v == buttonWeek) {
4              requestParam.put("daerah",daerahSpin
5              .getSelectedItem().toString());
6              requestParam.put("awal", mulai.getText()
7              .toString());
8              requestParam.put("akhir", sampai.getText()
9              .toString());
10             requestParam.put("tahunMingguan", tahunSpinWeek
11             .getSelectedItem().toString());
12             requestParam.put("bulanMingguan",
13             Integer.toString(bulanSpinWeek
14             .getSelectedItemPosition() + 1));
15             requestParam.put("mingguMingguan",
16             Integer.toString(mingguSpinWeek
17             .getSelectedItemPosition() + 1));
18             requestParam.put("tahunBulanan", tahunSpinMonth
19             .getSelectedItem().toString());
20             requestParam.put("bulanBulanan",
21             Integer.toString(bulanSpinMonth
22             .getSelectedItemPosition() + 1));
23             if (ConnectionDetector.isConnectingToInternet
24             (getApplicationContext()))
25             {
26                   new LoadView().execute();
27             }
28             else
29             {
30                   Toast.makeText(ChartActivity.this,"Tidak
31                   ada sambungan. Silakan coba lagi nanti.",
32                   Toast.LENGTH_SHORT).show();
33             }
34       }
35       if (v == buttonBack)
36       {
37             showDefaultView();
38             viewFlipper.showNext();
39       }
40       if (v == buttonChart)
41       {
42             if (!responseService.isEmpty())
43             {
44                  chart.setDataPoint(responseService);
45                  startActivity(chart
46                  .execute(getApplicationContext()));
47             }
48       }
49 }
50 private class LoadView extends AsyncTask<Void, Void, Void>
51 {
52 protected void onPreExecute()
53 {
54       progressDialog = ProgressDialog
55       .show(ChartActivity.this, "","Please wait...", false,
56       false);
57 }
58 protected Void doInBackground(Void... params)
59 {
```

```
60          responseService = serviceFews.getResponseService
61          (intervalSpin.getSelectedItem().toString(),
62          requestParam);
63          return null;
64  }
65  protected void onPostExecute(Void result)
66  {
67  progressDialog.dismiss();
68  if (!responseService.isEmpty())
69  {
70          ListAdapter adapter = new
71          SimpleAdapter(getApplicationContext(),
72          responseService, R.layout.grafik_listitem, new
73          String[] {"id","Daerah","Tgl","Tinggi"}, new
74          int[]{R.id.id_grafik,R.id.daerah,R.id.tglgrafik,
75          R.id.tinggiair});
76          setListAdapter(adapter);
77          viewFlipper.showNext();
78  }
79  else
80  {
81          Toast.makeText(ChartActivity.this, "Data tidak
82          ditemukan",Toast.LENGTH_SHORT).show();
83  }
84  }
85  }
```

Gambar 5.6 Implementasi Cuplikan *Code* Menampilkan Ketinggian Air
Sumber : [Implementasi]

Operasi untuk menampilkan ketinggian air sungai suatu daerah akan diproses ketika sistem sudah melakukan proses validitas pengguna. Adapun penjelasan dari gambar 5.6 sebagai berikut:

1. Baris 1-49 merupakan sebuah *method* yang digunakan ketika ada sebuah *event* pada tombol buttonDay, buttonWeek, buttonMonth, buttonBack dan buttonChart. Tombol buttonDay, buttonWeek, buttonMonth digunakan untuk memanggil operasi web service mengenai ketinggian air sungai berdasarkan waktu harian, mingguan dan bulanan. Tombol buttonBack digunakan untuk mengkonfirmasi pengguna ketika ingin keluar dari kondisi login. Tombol buttonChart digunakan untuk menampilkan data ketinggian air sungai dalam bentuk grafik. Baris 23-33 merupakan sebuah kondisi untuk mendeteksi koneksi internet. Jika bernilai benar maka *object* hasil instansiasi dari *class* LoadView akan melakukan proses pemanggilan operasi *web service*. Jika salah maka akan menampilkan pesan kesalahan/*error*.

2. Baris 50-85 merupakan sebuah *class* yang berfungsi untuk melakukan pemanggilan operasi *web service* sesuai dengan parameter masukan pengguna.*Class* tersebut memiliki tiga buah *method* yaitu *method* onPreExecute() berfungsi untuk inisialisasi sebuah *progress* pemanggilan operasi *web service. Method* doInBackground(Void… params) berfungsi untuk melakukan pemanggilan operasi *web service* sesuai dengan parameter masukan dari pengguna. *Method* onPostExecute(Void result) berfungsi untuk menampilkan hasil pemanggilan operasi *web service* ketinggian air sungai suatu daerah ke dalam bentuk *list.*

5.3.6 Implementasi Antarmuka

Antarmuka perangkat lunak ini dibagi menjadi empat, yaitu antarmuka halaman utama, antarmuka berita terkini, antarmuka status siaga dan antarmuka ketinggian air.

5.3.6.1 Antarmuka Halaman Utama

Antarmuka halaman utama memuat beberapa menu yaitu berita terkini bahaya banjir, status siaga suatu daerah dan ketinggian air sungai suatu daerah. Antarmuka halaman utama juga menyediakan *tools* seperti bantuan penggunaan perangkat, tentang perangkat dan perbaharui informasi.

Gambar 5.7 Antarmuka Halaman Utama
Sumber : [Implementasi]

Gambar 5.7 merupakan antarmuka halaman utama aplikasi *Flood Early Warning System* di dalam sistem operasi android. Terlihat pada gambar bahwa aplikasi memiliki beberapa menu tambahan selain fitur yang disediakan oleh sistem. Menu perbaharui digunakan untuk memperbaharui isi dari fitur berita terkini dan status siaga. Menu tentang digunakan untuk mengetahui informasi mengenai aplikasi. Menu bantuan berfungsi sebagai petunjuk dalam penggunaan aplikasi.

5.3.6.2 Antarmuka Berita Terkini

Antarmuka berita terkini merupakan halaman yang dapat digunakan pengguna untuk melihat berita terkini terkait bahaya banjir. Antarmuka ini memuat judul dari sebuah berita, tanggal pembuatan berita dan isi dari berita.

Gambar 5.8 Antarmuka Berita Terkini
Sumber : [Implementasi]

Gambar 5.8 menunjukkan antarmuka berita terkini terkait bencana banjir. Sistem akan langsung menampilkan informasi tentang bencana banjir kepada pengguna ketika aplikasi dijalankan tanpa meminta masukan kepada pengguna.

5.3.6.3 Antarmuka Status Siaga

Antarmuka status siaga merupakan halaman yang dapat digunakan pengguna untuk status siaga banjir suatu daerah. Antarmuka ini memuat tanggal penerimaan data, status siaga banjir tiap-tiap daerah dan tinggi muka air tiap-tiap daerah.

Gambar 5.9 Antarmuka Status Siaga
Sumber : [Implementasi]

Gambar 5.9 menunjukkan antarmuka status siaga banjir suatu daerah. Sistem akan langsung menampilkan informasi status siaga banjir kepada pengguna ketika aplikasi dijalankan tanpa meminta masukan kepada pengguna.

5.3.6.4 Antarmuka Ketinggian Air

Antarmuka ketinggian air merupakan sebuah halaman yang dapat digunakan oleh pengguna untuk mengetahui ketinggian air sungai suatu daerah ketika pengguna sudah melakukan validitas pengguna. Antarmuka ketinggian air terdiri atas halaman utama, halaman hasil pemrosesan ketinggian air dalam bentuk *list* dan halaman hasil pemrosesan ketinggian air dalam bentuk grafik.

Gambar 5.10 Antarmuka Validitas Pengguna (*Login*)
Sumber : [Implementasi]

Gambar 5.10 merupakan implementasi antarmuka validitas pengguna (*login*). Antarmuka validitas pengguna (*login*) merupakan halaman yang dapat digunakan pengguna untuk melakukan proses validitas pengguna. Proses tersebut dilakukan dengan memasukkan *username* dan *password*.

Gambar 5.11 merupakan halaman utama menu ketinggian air. Halaman ini akan ditampilkan ketika pengguna sudah melalui tahap validitas pengguna (*login*). Untuk mengatasi kesalahan *request* pengguna maka halaman ini dirancang sedemikian rupa secara bertahap.

Gambar 5.11 Halaman Utama Ketinggian Air
Sumber : [Implementasi]

Antarmuka pada gambar 5.11 memungkinkan pengguna harus memilih daerah sebelum ke proses selanjutnya. Gambar 5.12 menunjukkan antarmuka ketika pengguna sudah memilih daerah.

Gambar 5.12 Halaman Utama ketika Pengguna Memilih Daerah
Sumber : [Implementasi]

Terlihat pada gambar 5.12 bahwa sistem akan meminta sebuah masukan kepada pengguna berupa interval sebuah tanggal yaitu tanggal awal dan akhir. Gambar 5.13 menunjukkan bagian dari halaman utama ketika pengguna memilih interval mingguan.

Gambar 5.13 Halaman Utama ketika Pengguna Memilih Interval Mingguan
Sumber : [Implementasi]

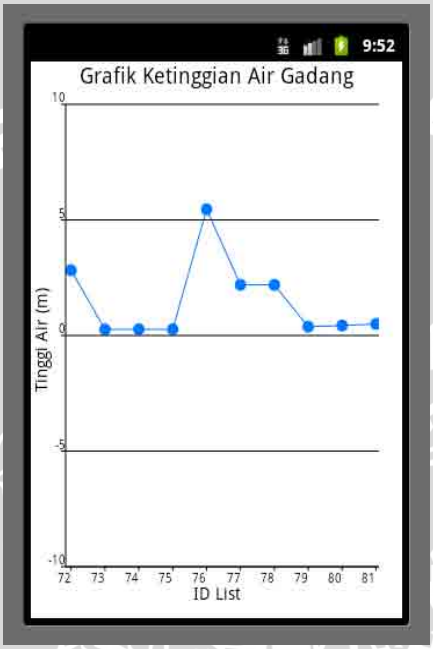
Terlihat pada gambar 5.13 bahwa pengguna dapat mengatur parameter dalam melakukan *request* untuk menampilkan ketinggian air sungai di suatu daerah. Pengguna dapat mengatur parameter bulan, tahun dan minggu sesuai dengan yang diinginkan. Gambar 5.14 menunjukkan bagian dari halaman utama ketika pengguna memilih interval bulanan.

Gambar 5.14 Halaman Utama ketika Pengguna Memilih Interval Bulanan
Sumber : [Implementasi]

Terlihat pada gambar 5.14 bahwa pengguna dapat mengatur parameter dalam melakukan *request* untuk menampilkan ketinggian air sungai di suatu daerah. Pengguna dapat mengatur parameter bulan dan tahun sesuai dengan yang diinginkan.

Gambar 5.15 Halaman Hasil Pemrosesan Ketinggian Air dalam Bentuk *List*
Sumber : [Implementasi]

Gambar 5.15 merupakan sebuah halaman yang berfungsi untuk menampilkan hasil pemrosesan ketinggian air. Halaman ini akan tampil ketika pengguna sudah memasukkan parameter dari masing-masing interval waktu. Hasil tersebut akan ditampilkan dalam bentuk *list.* Di dalam halaman tersebut terdapat dua tombol yaitu tombol Kembali dan Grafik. Fungsi tombol Kembali adalah untuk mengembalikan halaman utama pada posisi awal sedang tombol Grafik berfungsi untuk melihat hasil pemrosesan dalam bentuk grafik.



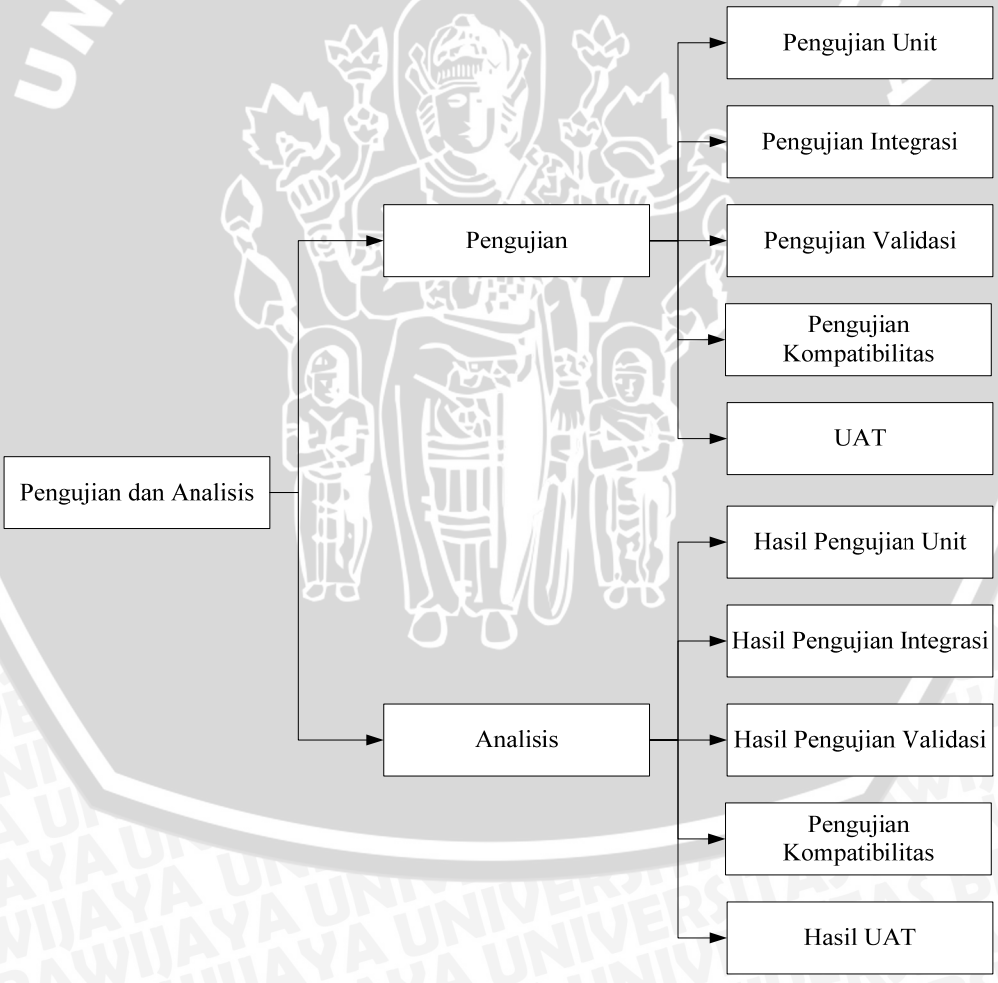
Gambar 5.16 Halaman Hasil Pemrosesan Ketinggian Air dalam Bentuk Grafik

Sumber : [Implementasi]

Gambar 5.16 merupakan sebuah halaman yang digunakan untuk menampilkan informasi ketinggian air sungai suatu daerah dalam bentuk grafik. Grafik tersebut mempunyai dua indikator yaitu tinggi air dan id *list.* Indikator tinggi air digunakan sebagai ukuran dari ketinggian air sedangkan indikator id *list* digunakan untuk menunjukkan informasi yang dimuat pada gambar 5.15.

BAB VI
PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai tahapan pengujian dan analisis Aplikasi Perangkat Bergerak Penanda Dini Bahaya Banjir Menggunakan *Web service* Berbasis SOAP. Proses pengujian dilakukan melalui empat tahapan (strategi) yaitu pengujian unit, pengujian integrasi, pengujian validasi dan UAT. Pada pengujian unit dan integrasi akan digunakan teknik pengujian *White-Box Testing.* Pada pengujian validasi dan pengujian kompatibilitas akan digunakan teknik pengujian *Black-Box Testing.* UAT dilakukan dengan cara memberikan kuesioner kepada pengguna sistem untuk mendapatkan timbal balik pengguna terhadap sistem.



Gambar 6.1 Diagram Pohon Pengujian dan Analisis
Sumber : [Pengujian dan Analisis]

6.1 Pengujian

Proses pengujian dilakukan melalui empat tahapan (strategi) yaitu pengujian unit, pengujian integrasi, pengujian validasi dan UAT.

6.1.1 Pengujian Unit

Perangkat lunak yang dikembangkan dengan konsep *object-oriented programming* menerapkan pengujian unit untuk suatu *method* (operasi) dari suatu *class.* Pada pengujian unit digunakan teknik *White-Box Testing* dengan teknik *Basis Path Testing.* Proses pengujian pada teknik *Basis Path Testing* dilakukan dengan memodelkan operasi dari suatu *class* pada suatu *flow graph,* menentukan jumlah *cyclomatic complexity,* menentukan sebuah basis set dari jalur independen dan memberikan kasus uji (*test case*) pada setiap basis set yang telah ditentukan. Pengujian unit pada Aplikasi Perangkat Bergerak Penanda Dini Bahaya Banjir Menggunakan *Web service* Berbasis SOAP digunakan untuk memastikan unit terkecil dari program atau modul beroperasi dengan tepat ditinjau dari jalur kontrol dari masing-masing modul dan menjamin setiap modul menjalankan fungsinya dengan baik. Pengujian unit aplikasi ini dilakukan pengujian pada *method* getResponseWS(HashMap<String, String> parameter) pada *class* BeritaService dan *method* getSOAPWS(String serviceName) pada *class* ConcreateSOAP Factory.

6.1.1.1 Pengujian Unit pada *Class* BeritaService

Class BeritaService merupakan sebuah class yang berfungsi untuk berfungsi untuk menangani proses pemanggilan operasi *web service* getBerita yang terdapat pada *method* getResponseWS(HashMap<String, String> parameter). Gambar 6.2 menunjukkan proses pengujian pada *method* getResponseWS(HashMap<String, String> parameter).

```
 1  public String getResponseWS(HashMap<String, String>
 2  parameter) {
 3  String response = "";
 4  SoapObject request = new SoapObject(DataManager
 5  .NAMESPACE, "getBerita");
 6  request.addProperty("fews", "fewsmobile");
 7  SoapSerializationEnvelope envelope = new
 8  SoapSerializationEnvelope(SoapEnvelope.VER11);
 9  envelope.dotNet = true;
10  envelope.setOutputSoapObject(request);
11  TrustManagerManipulator.allowAllSSL();
```

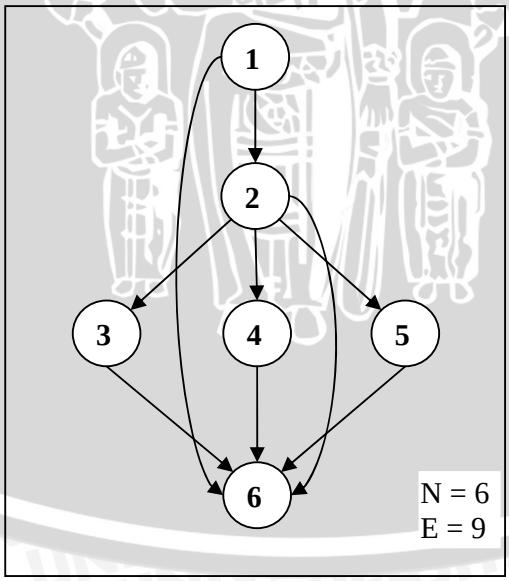
(1)

```
12  try {
13      httpTransportSE.call(DataManager.SOAP_ACTION +
14      "/" + "getBerita",envelope);
15      httpTransportSE.debug = true;
16      if (envelope.bodyIn instanceof SoapObject) {
17          result = (SoapObject) envelope.bodyIn;
18          response = result.getProperty(0).toString();
19          return response;
20      }
21  } catch (IOException e) {
22      return response;
23  }
24  catch (XmlPullParserException e) {
25      return response;
26  }
27  catch (Exception e) {
28      return response;
29  }
30  return response;
31  }
```

2 3 4 5 6

Gambar 6.2 Pengujian Unit pada *Class* BeritaService
Sumber : [Pengujian dan Analisis]

Terlihat pada gambar 6.2 bahwa *method* getResponseWS(HashMap<String, String> parameter) memiliki beberapa proses dan jalur kontrol. Penomoran pada setiap proses dan jalur kontrol digunakan untuk memodelkannya ke dalam *flow graph*. *Gambar* 6.3 menjelaskan *flow graph method* getResponseWS(HashMap <String , String> parameter) pada *class* BeritaService.



Gambar 6.3 *Flow Graph Class* BeritaService
Sumber : [Pengujian dan Analisis]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* getResponseWS(HashMap <String , String> parameter) pada *class* BeritaService

menghasilkan jumlah kompleksitas siklomatis (*cyclomatic complexity*) melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul (*node*).

V(G) = E – N + 2
= 9 – 6 + 2
= 5

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka ditentukan lima buah basis set dari jalur *independent*, yaitu:

Jalur 1 : 1 – 6
Jalur 2 : 1 – 2 – 6
Jalur 3 : 1 – 2 – 3 – 6
Jalur 4 : 1 – 2 – 4 – 6
Jalur 5 : 1 – 2 – 5 – 6

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada tabel 6.1.

Tabel 6.1 Kasus Uji Pengujian Unit *Class* BeritaService

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Memberikan nilai pada atribut *response* dengan nilai kosong dan memberi nilai parameter berupa string ke dalam atribut *request*	*Method* memberikan nilai kembalian dengan nilai kosong	*Method* memberikan nilai kembalian dengan nilai kosong
2	Memberikan nilai pada atribut *response* dengan nilai kosong dan memberi nilai parameter berupa string ke dalam atribut *request*	*Method* memberikan nilai kembalian dengan nilai kosong	*Method* memberikan nilai kembalian dengan nilai kosong
3	Memberikan nilai pada atribut *response* dengan nilai kosong dan memberi nilai parameter berupa string ke dalam atribut *request*	*Method* memberikan nilai kembalian dengan nilai kosong	*Method* memberikan nilai kembalian dengan nilai kosong
4	Memberikan nilai pada atribut *response* dengan nilai kosong dan memberi nilai parameter berupa string ke dalam atribut *request*	*Method* memberikan nilai kembalian dengan nilai SoapObject yang	*Method* memberikan nilai kembalian dengan nilai SoapObject yang

		dikonversikan ke dalam string	dikonversikan ke dalam string
5	Memberikan nilai pada atribut *response* dengan nilai kosong dan memberi nilai parameter berupa string ke dalam atribut *request*	*Method* memberikan nilai kembalian dengan nilai kosong	*Method* memberikan nilai kembalian dengan nilai kosong

Sumber : [Pengujian dan Analisis]

6.1.1.2 Pengujian Unit pada *Class* ConcreateSOAPFactory

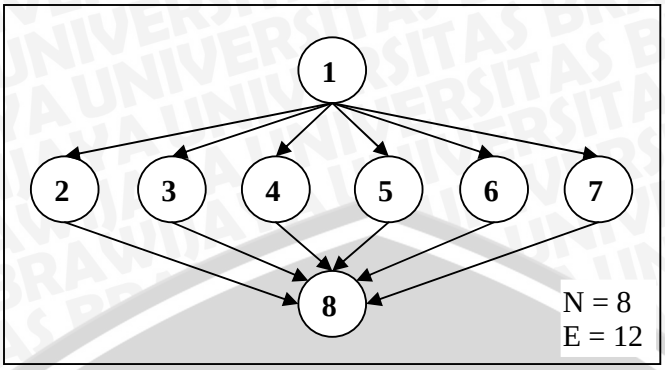
Class ConcreateSOAPFactory merupakan turunan dari *class* SOAPFactory yang digunakan untuk menginstansiasi sebuah *object* dari *class* turunan SOAPService yang terdapat pada *method* getSOAPWS(String serviceName). Gambar 6.4 menunjukkan proses pengujian pada *method* getSOAPWS(String serviceName).

```
1  public SOAPService getSOAPWS(String serviceName) {
2
3        String parameter = serviceName;                          } 1
4
5        if (parameter.equals("Berita")) {                        }
6              return new BeritaService();                        } 2
7        }
8        else if (parameter.equals("Login")) {                    }
9              return new LoginService();                         } 3
10       }
11       else if (parameter.equals("StatusSiaga")) {              }
12             return new StatusSiagaService();                   } 4
13       }
14       else if (parameter.equals("Hari")) {                     }
15             return new GrafikHariService();                    } 5
16       }
17        else if (parameter.equals("Minggu")) {                  }
18             return new GrafikMingguService();                  } 6
19       }
20        else if (parameter.equals("Bulan")) {                   }
21             return new GrafikBulanService();                   } 7
22       }
23
24       return null;                                             } 8
25 }
```

Gambar 6.4 Pengujian Unit pada *Class* ConcreateSOAPFactory
Sumber : [Pengujian dan Analisis]

Terlihat pada gambar 6.4 bahwa *method* getSOAPWS(String serviceName) memiliki beberapa proses dan jalur kontrol. Penomoran pada setiap proses dan jalur kontrol digunakan untuk memodelkannya ke dalam *flow graph*.

Gambar 6.5 merupakan *flow graph method* getSOAPWS(String serviceName) pada class ConcreateSOAPFactory.



Gambar 6.5 *Flow Graph Class* ConcreateSOAPFactory
Sumber : [Pengujian dan Analisis]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap method getSOAPWS(String serviceName) menghasilkan jumlah kompleksitas siklomatis (*cyclomatic complexity*) melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2
= 12 – 8 + 2
= 6

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka ditentukan empat buah basis set dari jalur *independent*, yaitu:

Jalur 1 : 1 – 2 – 8
Jalur 2 : 1 – 3 – 8
Jalur 3 : 1 – 4 – 8
Jalur 4 : 1 – 5 – 8
Jalur 5 : 1 – 6 – 8
Jalur 6 : 1 – 7 – 8

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada tabel 6.2.

Tabel 6.2 Kasus Uji Pengujian Unit *Class* ConcreateSOAPFactory

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Memberikan nilai pada atribut parameter dengan nilai string “Berita”	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* BeritaService	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* BeritaService

2	Memberikan nilai pada atribut parameter dengan nilai string “Login”	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* LoginService	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* LoginService
3	Memberikan nilai pada atribut parameter dengan nilai string “StatusSiaga”	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* StatusSiagaService	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* StatusSiagaService
4	Memberikan nilai pada atribut parameter dengan nilai string “Hari”	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* GrafikHariService	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* GrafikHariService
5	Memberikan nilai pada atribut parameter dengan nilai string “Minggu”	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* GrafikMingguService	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* GrafikMingguService
6	Memberikan nilai pada atribut parameter dengan nilai string “Bulan”	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* GrafikBulanService	*Method* memberikan nilai kembalian berupa *object* hasil instansiasi *class* GrafikBulanService

Sumber : [Pengujian dan Analisis]

6.1.2 Pengujian Integrasi

Pengujian integrasi diterapkan pada proses yang mengintegrasikan fungsionalitas dari beberapa *class* untuk melakukan sebuah operasi tertentu. Pada pengujian integrasi yang dijadikan sebagai obyek uji adalah *class* inti yang menggabungkan kinerja dari *class* operasi, *entity class*, *class-class* antarmuka, dan *class-class* lain. Pengujian integrasi yang dilakukan terhadap *class* inti ini menggunakan strategi *bottom-up,* dimana modul-modul yang diintegrasikan masing-masing diuji terlebih dahulu dalam pengujian unit.

6.1.2.1 Pengujian Integrasi untuk Menampilkan Berita Terkini

Proses menampilkan berita terkini mengenai bahaya banjir melalui pemanggilan operasi *web service* dilakukan di dalam *class* MainActivity. Selanjutnya hasil pemrosesan diolah di dalam *class* BeritaActivity untuk ditampilkan ke dalam sebuah antarmuka pengguna. Gambar 6.6 menunjukkan

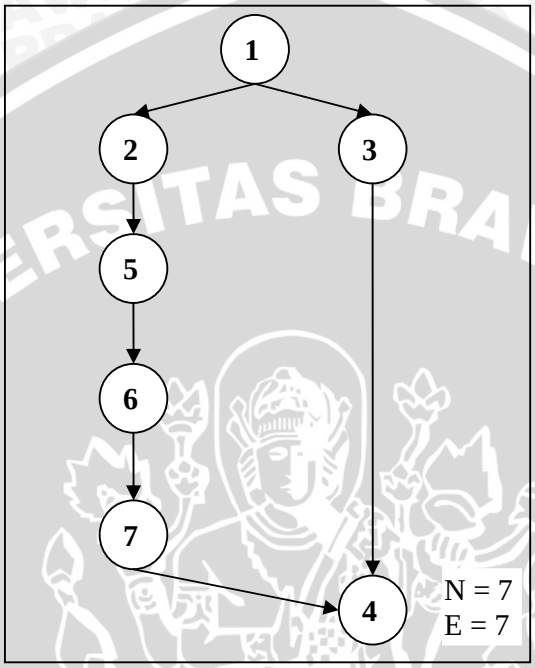
proses pengujian pada *method* onCreate(Bundle savedInstanceState) untuk menampilkan berita terkini mengenai bahaya banjir.

```
1  public void onCreate(Bundle savedInstanceState) {
2  super.onCreate(savedInstanceState);
3  requestWindowFeature(Window.FEATURE_NO_TITLE);
4  setContentView(R.layout.main);
5  setRequestedOrientation(ActivityInfo
6  .SCREEN_ORIENTATION_PORTRAIT);
7
8  berita = new BeritaActivity(getLayoutInflater()
9  .inflate(R.layout.info,null));
10
11 if (ConnectionDetector
12 .isConnectingToInternet(getApplicationContext())) {
13       new LoadService().execute();
14 }
15 else {
16       getErrorConnection();
17       isConnected = false;
18 }
19
20 changeTab();
21
22 }
23 private class LoadService extends AsyncTask<Void,
24 Void, Void> {
25 protected void onPreExecute() {
26       progressDialog = ProgressDialog
27       .show(MainActivity.this, "","Silakan
28       tunggu...",true, false);
29 }
30 protected Void doInBackground(Void... params) {
31       synchronized (this) {
32            serviceBerita = serviceFews
33            .getResponseService("Berita", null);
34       }
35 return null;
36 }
37 protected void onPostExecute(Void result) {
38       progressDialog.dismiss();
39       if (!serviceBerita.get(0).get("Judul")
40       .equals("")){
41            berita.refreshContentInfo
42            (new HelperClass(getApplicationContext(),
43            R.layout.subinfo, serviceBerita,
44            "Berita"));
45            pBerita.removeAllViewsInLayout();
46            pBerita.addView(berita.beritaView);
47       }
48 }
```

1 2 3 4 5 6 7

Gambar 6.6 Pengujian Integrasi untuk Menampilkan Berita Terkini
Sumber : [Pengujian dan Analisis]

Terlihat pada gambar 6.6 bahwa *method* onCreate(Bundle savedInstance State) memiliki beberapa proses dan jalur kontrol. Penomoran pada setiap proses dan jalur kontrol digunakan untuk memodelkannya ke dalam *flow graph*. Gambar 6.7 merupakan *flow graph* untuk menampilkan berita terkini mengenai bahaya banjir.



Gambar 6.7 *Flow Graph* untuk Menampilkan Berita Terkini
Sumber : [Pengujian dan Analisis]

Pemodelan ke dalam *flow graph* yang telah dilakukan menghasilkan jumlah kompleksitas siklomatis (*cyclomatic complexity*) melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2
= 7 – 7 + 2
= 2

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka ditentukan empat buah basis set dari jalur *independent*, yaitu:

Jalur 1 : 1 – 2 – 5 – 6 – 7 – 4
Jalur 2 : 1 – 3 – 4

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada tabel 6.3.

Tabel 6.3 Kasus Uji Pengujian Integrasi untuk Menampilkan Berita Terkini

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Mendeteksi adanya sebuah koneksi internet	Data hasil pemrosesan pemanggilan operasi *web service* tentang status siaga banjir suatu daerah akan ditampilkan ke dalam antarmuka	Data hasil pemrosesan pemanggilan operasi *web service* tentang status siaga banjir suatu daerah akan ditampilkan ke dalam antarmuka
2	Tidak terdeteksi adanya koneksi internet	Menampilkan pesan kesalahan pada antarmuka	Menampilkan pesan kesalahan pada antarmuka

Sumber : [Pengujian dan Analisis]

6.1.2.2 Pengujian Integrasi untuk Menampilkan Status Siaga

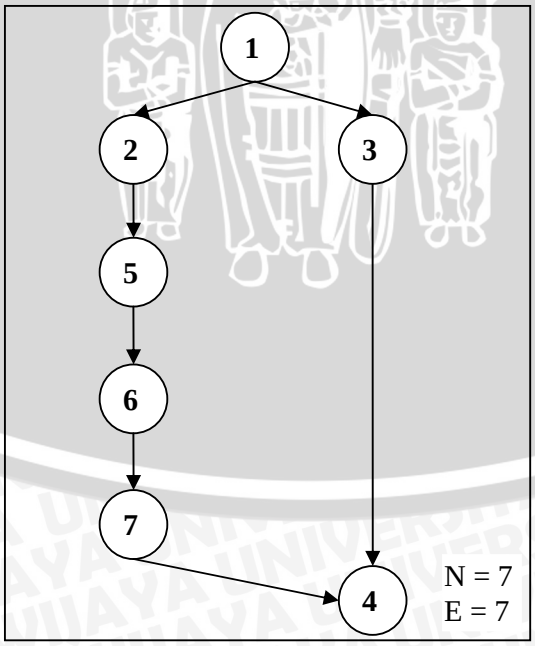
Proses menampilkan status siaga banjir suatu daerah melalui pemanggilan operasi *web service* dilakukan di dalam *class* MainActivity. Selanjutnya hasil pemrosesan diolah di dalam *class* StatusSiagaActivity untuk ditampilkan ke dalam sebuah antarmuka pengguna. Gambar 6.8 menunjukkan proses pengujian pada *method* onCreate(Bundle savedInstanceState) untuk menampilkan status siaga banjir suatu daerah.

```
 1  @Override
 2  public void onCreate(Bundle savedInstanceState) {
 3
 4  super.onCreate(savedInstanceState);                    } 1
 5  requestWindowFeature(Window.FEATURE_NO_TITLE);         }
 6  setContentView(R.layout.main);                         }
 7  setRequestedOrientation(ActivityInfo                   }
 8  .SCREEN_ORIENTATION_PORTRAIT);                         }
 9
10  siaga = new SiagaActivity(getLayoutInflater()          }
11  .inflate(R.layout.siaga, null));                       }
12
13  if (ConnectionDetector                                 } 2
14  .isConnectingToInternet(getApplicationContext())){     }
15        new LoadService().execute();                     }
16  }
17  else {                                                 } 3
18        getErrorConnection();                            }
19        isConnected = false;                             }
20  }
21
22  changeTab();                                           } 4
23
24  }
25
26  private class LoadService extends AsyncTask<Void,
27  Void, Void> {
```

```
28  protected void onPreExecute() {                                  5
29        progressDialog = ProgressDialog
30        .show(MainActivity.this, "","Silakan
31        tunggu...",true, false);
32  }
33  protected Void doInBackground(Void... params) {
34
35        synchronized (this) {                                      6
36            serviceSiaga = serviceFews
37            .getResponseService("StatusSiaga", null);
38        }
39  return null;
40  }
41  protected void onPostExecute(Void result) {
42        progressDialog.dismiss();
43        if(!serviceSiaga.get(0).get("Wilayah")                     7
44        .equals("")) {
45              siaga.refreshContentSiaga(serviceSiaga);
46              pSiaga.removeAllViewsInLayout();
47              pSiaga.addView(siaga.getSiagaView());
48        }
49  }
```

Gambar 6.8 Pengujian Integrasi untuk Menampilkan Status Siaga
Sumber : [Pengujian dan Analisis]

Terlihat pada gambar 6.8 bahwa *method* onCreate(Bundle savedInstance State) memiliki beberapa proses dan jalur kontrol. Penomoran pada setiap proses dan jalur kontrol digunakan untuk memodelkannya ke dalam *flow graph*. Gambar 6.9 merupakan *flow graph* untuk menampilkan informasi tentang status siaga banjir suatu daerah.



Gambar 6.9 *Flow Graph* untuk Menampilkan Status Siaga
Sumber : [Pengujian dan Analisis]

Pemodelan ke dalam *flow graph* yang telah dilakukan menghasilkan jumlah kompleksitas siklomatis (*cyclomatic complexity*) melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2

= 7 – 7 + 2

= 2

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka ditentukan empat buah basis set dari jalur *independent*, yaitu:

Jalur 1 : 1 – 2 – 5 – 6 – 7 – 4

Jalur 2 : 1 – 3 – 4

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada tabel 6.4.

Tabel 6.4 Kasus Uji Pengujian Integrasi untuk Menampilkan Status Siaga

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Mendeteksi adanya sebuah koneksi internet	Data hasil pemrosesan pemanggilan operasi *web service* tentang status siaga banjir suatu daerah akan ditampilkan ke dalam antarmuka	Data hasil pemrosesan pemanggilan operasi *web service* tentang status siaga banjir suatu daerah akan ditampilkan ke dalam antarmuka
2	Tidak terdeteksi adanya koneksi internet	Menampilkan pesan kesalahan pada antarmuka	Menampilkan pesan kesalahan pada antarmuka

Sumber : [Pengujian dan Analisis]

6.1.2.3 Pengujian Integrasi untuk Menampilkan Ketinggian Air

Proses pengujian untuk menampilkan ketinggian air sungai suatu daerah dilakukan dengan dua obyek uji yaitu pengujian integrasi untuk validitas pengguna dan pengujian integrasi untuk menampilkan ketinggian air berdasarkan satuan waktu tertentu. Pengujian integrasi untuk validitas pengguna dilakukan dengan menggunakan *method* onClick(View arg0) yang terdapat pada *class* LoginActivity. Gambar 6.10 menunjukkan pengujian integrasi untuk validitas pengguna.

```
1   public void onClick(View arg0) {
2   View event = arg0;
3
4   if (event.equals(login)) {
5       if (!username.getText().toString().equals("")&&
6       !password.getText().toString().equals("")) {
7           mapLogin.put("username",username.getText()
8           .toString());
9           mapLogin.put("password",password.getText()
10          .toString());
11          if (ConnectionDetector
12          .isConnectingToInternet
13          (getApplicationContext())){
14              new LoadView().execute();
15          }
16          else {
17              Toast.makeText(LoginActivity.this,"Tidak
18              ada sambungan. Silakan coba lagi
19              nanti.",Toast.LENGTH_SHORT).show();
20          }
21      }
22      else {
23          Toast.makeText(this,"Silakan, "masukkan
24          username dan password",Toast.LENGTH_LONG)
25          .show();
26      }
27      username.setText("");
28      password.setText("");
29  }}
30  private class LoadView extends AsyncTask<Void, Void,
31  Void> {
32  protected void onPreExecute() {
33      progressDialog = ProgressDialog
34      .show(LoginActivity.this, "","Please wait...");
35  }
36
37  protected Void doInBackground(Void... params) {
38      responseService =
39      serviceFews.getResponseService("Login",
40      mapLogin);
41      return null;
42  }
43
44  protected void onPostExecute(Void result) {
45      if (!responseService.get(0).get("result")
46      .equals("")) {
47           username.setText("");
48           password.setText("");
49           if(responseService.get(0).get("result")
50           .equals("1")){
51               startActivity(new Intent(LoginActivity
52               .this,ChartActivity.class));
53           }
54           else{
55               Toast.makeText(LoginActivity.this,
56               "Username and password tidak valid!",
57               Toast.LENGTH_LONG).show();
58           }
59      }
```

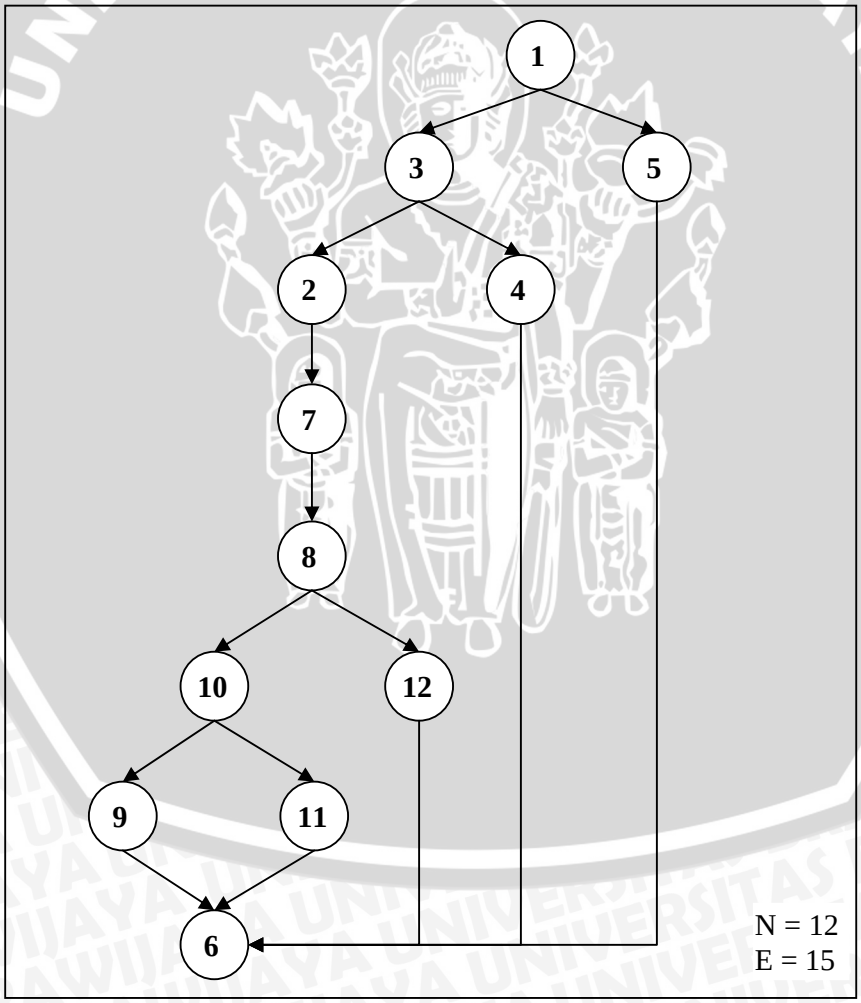
1
2
3
4
5
6
7
8
9
10
11

```
60      else {
61           username.setText("");
62           password.setText("");
63           Toast.makeText(LoginActivity.this,"Tidak ada
64           sambungan. Silakan coba lagi nanti.",
65           Toast.LENGTH_LONG).show();
66      }
67  }
68  }
```

12

Gambar 6.10 Pengujian Integrasi untuk Validitas Pengguna
Sumber : [Pengujian dan Analisis]

Terlihat pada gambar 6.10 bahwa *method* onClick(View arg0) memiliki beberapa proses dan jalur kontrol. Penomoran pada setiap proses dan jalur kontrol digunakan untuk memodelkannya ke dalam *flow graph*. Gambar 6.11 merupakan *flow graph* untuk validitas pengguna yang digunakan untuk melihat informasi ketinggian air sungai suatu daerah.



Gambar 6.11 *Flow Graph* untuk Validitas Pengguna
Sumber : [Pengujian dan Analisis]

Pemodelan ke dalam *flow graph* yang telah dilakukan menghasilkan jumlah kompleksitas siklomatis (*cyclomatic complexity*) melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2

= 15 – 12 + 2

= 5

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka ditentukan empat buah basis set dari jalur *independent*, yaitu:

Jalur 1 : 1 – 5 – 6

Jalur 2 : 1 – 3 – 4 – 6

Jalur 3 : 1 – 3 – 2 – 7 – 8 – 12 – 6

Jalur 4 : 1 – 3 – 2 – 7 – 8 – 10 – 11 – 6

Jalur 5 : 1 – 3 – 2 – 7 – 8 – 10 – 9 – 6

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada tabel 6.5.

Tabel 6.5 Kasus Uji Pengujian Integrasi untuk Validitas Pengguna

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Memberikan nilai atribut *username* dan *password* dengan nilai kosong	Menampilkan pesan kesalahan pada antarmuka	Menampilkan pesan kesalahan pada antarmuka
2	Tidak terdeteksi adanya koneksi internet	Menampilkan pesan kesalahan pada antarmuka	Menampilkan pesan kesalahan pada antarmuka
3	Tidak terdeteksi adanya koneksi internet	Menampilkan pesan kesalahan pada antarmuka	Menampilkan pesan kesalahan pada antarmuka
4	Memberikan nilai atribut *username* dan *password* dengan nilai “username” dan “password”	Data hasil pemanggilan operasi *web service* bernilai salah dan menampilkan pesan kesalahan pada antarmuka	Data hasil pemanggilan operasi *web service* bernilai salah dan menampilkan pesan kesalahan pada antarmuka
5	Memberikan nilai atribut *username* dan *password* dengan	Data hasil pemanggilan operasi *web service* bernilai	Data hasil pemanggilan operasi *web service* bernilai

	nilai “pjt1a” dan “pjt1a”	benar dan menampilkan antarmuka ketinggian air sungai yang ditampilkan dalam bentuk *form* masukan.	benar dan menampilkan antarmuka ketinggian air sungai yang ditampilkan dalam bentuk *form* masukan.

Sumer : [Pengujian dan Analisis]

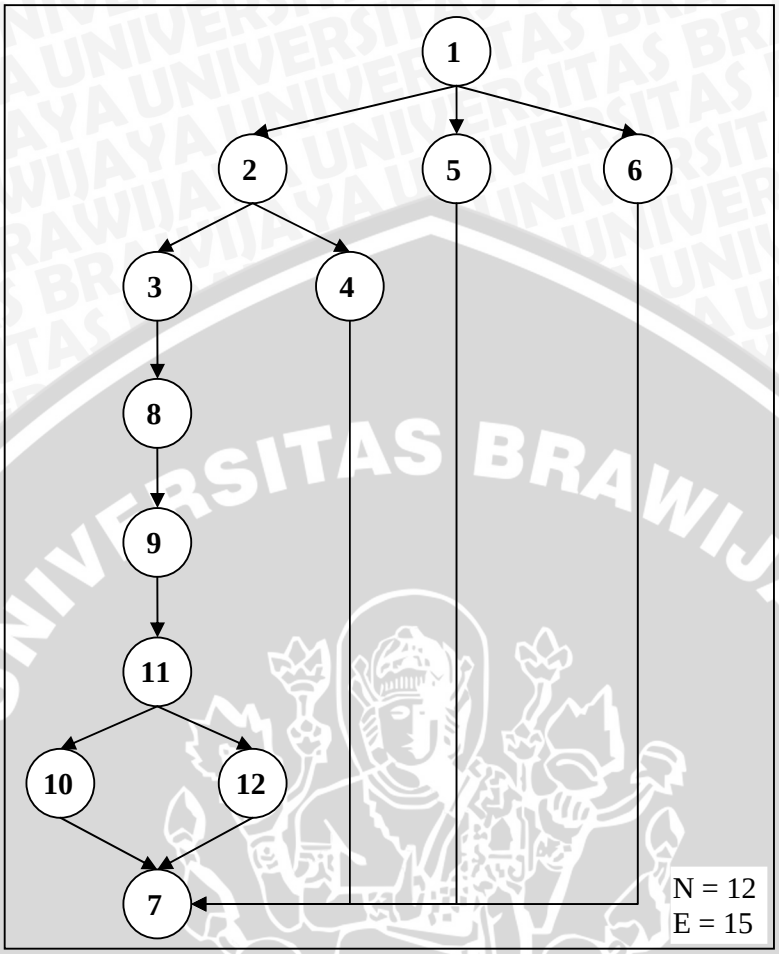
Pengujian integrasi untuk menampilkan ketinggian air sungai suatu daerah dilakukan dengan menggunakan *method* onClick(View arg0) yang terdapat pada *class* ChartActivity. Gambar 6.12 menunjukkan pengujian integrasi untuk menampilkan ketinggian air.

```
 1  public void onClick(View v) {                                    1
 2  View event = v;                                                  1
 3
 4  if (v == buttonDay || v == buttonMonth ||                        2
 5  v == buttonWeek) {                                               2
 6       requestParam.put("daerah",daerahSpin                        2
 7       .getSelectedItem().toString());                             2
 8       requestParam.put("awal", mulai.getText()                    2
 9       .toString());                                               2
10       requestParam.put("akhir", sampai.getText()                  2
11       .toString());                                               2
12       requestParam.put("tahunMingguan", tahunSpin                 2
13       Week.getSelectedItem().toString());                         2
14       requestParam.put("bulanMingguan",                           2
15       Integer.toString(bulanSpinWeek                              2
16       .getSelectedItemPosition() + 1));                           2
17       requestParam.put("mingguMingguan",                          2
18       Integer.toString(mingguSpinWeek                             2
19       .getSelectedItemPosition() + 1));                           2
20       requestParam.put("tahunBulanan", tahunSpin                  2
21       Month.getSelectedItem().toString());                        2
22       requestParam.put("bulanBulanan",                            2
23       Integer.toString(bulanSpinMonth                             2
24       .getSelectedItemPosition() + 1));                           2
25       if (ConnectionDetector.isConnectingToInternet               2
26       (getApplicationContext())) {                                3
27            new LoadView().execute();                              3
28       }                                                           3
29       else {                                                      4
30            Toast.makeText(ChartActivity.this,                     4
31            "Tidak ada sambungan.                                  4
32            Silakan coba lagi nanti.",                             4
33            Toast.LENGTH_SHORT).show();                            4
34       }                                                           4
35  }
36  if (event == buttonChart) {                                      5
37       if (!responseService.isEmpty()) {                           5
38       chart.setDataPoint(responseService);                        5
39       startActivity(chart                                         5
40       .execute(getApplicationContext()));}                        5
41  }
```

```
42  if (event == buttonBack) {                                        6
43       showDefaultView();
44       viewFlipper.showNext();
45    }
46                                                                     7
47  showDefaultView();
48
49  }
50  private class LoadView extends AsyncTask<Void, Void,
51  Void> {
52  protected void onPreExecute() {                                    8
53       progressDialog = ProgressDialog
54       .show(ChartActivity.this, "","Please wait...",
55       false, false);
56  }
57  protected Void doInBackground(Void... params) {                    9
58       responseService = serviceFews
59       .getResponseService(intervalSpin
60       .getSelectedItem().toString(), requestParam);
61  return null
62  }
63  protected void onPostExecute(Void result) {
64  progressDialog.dismiss();
65       if (!responseService.isEmpty()) {                            10
66            ListAdapter adapter = new
67            SimpleAdapter(getApplicationContext(),
68            responseService, R.layout
69            .grafik_listitem,new String[]
70            {"id","Daerah","Tgl","Tinggi"}, new
71            int[]{R.id.id_grafik,R.id.daerah,                        11
72            R.id.tglgrafik, R.id.tinggiair});
73            setListAdapter(adapter);
74            viewFlipper.showNext();
75       }
76       else {                                                        12
78           Toast.makeText(ChartActivity.this, "Data
79           tidak ditemukan",
80           Toast.LENGTH_SHORT).show();
81       }
82  }
83  }
```

Gambar 6.12 Pengujian Integrasi untuk Menampilkan Ketinggian Air
Sumber : [Pengujian dan Analisis]

Terlihat pada gambar 6.12 bahwa *method* onClick(View arg0) memiliki beberapa proses dan jalur kontrol. Penomoran pada setiap proses dan jalur kontrol digunakan untuk memodelkannya ke dalam *flow graph*. Gambar 6.13 merupakan *flow graph* untuk menampilkan informasi tentang ketinggian air sungai suatu daerah.



Gambar 6.13 *Flow Graph* untuk Menampilkan Ketinggian Air
Sumber : [Pengujian dan Analisis]

Pemodelan ke dalam *flow graph* yang telah dilakukan menghasilkan jumlah kompleksitas siklomatis (*cyclomatic complexity*) melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2
= 15 – 12 + 2
= 5

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka ditentukan empat buah basis set dari jalur *independent*, yaitu:

Jalur 1 : 1 – 6 – 7
Jalur 2 : 1 – 5 – 7
Jalur 3 : 1 – 2 – 4 – 7

Jalur 4 : 1 – 2 – 3 – 8 – 9 – 11 – 12 – 7

Jalur 5 : 1 – 2 – 3 – 8 – 9 – 11 – 10 – 7

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada tabel 6.6.

Tabel 6.6 Kasus Uji Pengujian Integrasi untuk Menampilkan Ketinggian Air

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Memberikan sebuah *event* pada tombol buttonBack dengan menekan tombol buttonBack tersebut	Tampilan aplikasi akan berubah ke dalam tampilan awal	Tampilan aplikasi akan berubah ke dalam tampilan awal
2	Memberikan sebuah *event* pada tombol buttonChart dengan menekan tombol buttonChart tersebut	Menampilkan data hasil pemanggilan operasi *web service* ke dalam bentuk grafik	Menampilkan data hasil pemanggilan operasi *web service* ke dalam bentuk grafik
3	Tidak terdeteksi adanya koneksi internet	Menampilkan pesan kesalahan pada antarmuka	Menampilkan pesan kesalahan pada antarmuka
4	Memberikan nilai atribut parameter dengan nama daerah “” tahun “” dan bulan “”	Menampilkan pesan kesalahan pada antarmuka bahwa data tidak dapat ditemukan	Menampilkan pesan kesalahan pada antarmuka bahwa data tidak dapat ditemukan
5	Memberikan nilai atribut parameter dengan nama daerah “Gadang” tahun “2012” dan bulan “Oktober”	Data hasil pemanggilan operasi *web service* ditampilkan dalam bentuk *list.*	Data hasil pemanggilan operasi *web service* ditampilkan dalam bentuk *list.*

Sumer : [Pengujian dan Analisis]

6.1.3 Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. *Item-item* yang telah dirumuskan dalam daftar kebutuhan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Pengujian validasi menggunakan metode pengujian *Black-Box* karena tidak diperlukan konsentrasi terhadap alur jalannya program/aplikasi.

6.1.3.1 Kasus Uji Validasi

Kasus uji validasi memuat nama kasus uji, obyek uji, tujuan pengujian, prosedur uji dan hasil yang diharapkan. Terdapat tujuh kasus uji dalam pengujian validasi diantaranya adalah kasus uji menampilkan berita terkini, kasus uji menampilkan status siaga, kasus uji login sah, kasus uji login tidak sah, kasus uji menampilkan ketinggian air berdasarkan harian, kasus uji menampilkan ketinggian air berdasarkan mingguan dan kasus uji menampilkan ketinggian air berdasarkan bulanan.

a. Kasus Uji Menampilkan Berita Terkini

Proses menampilkan berita terkini mengenai bahaya banjir dilakukan dengan memanggil operasi *web service* tanpa parameter. Tabel 6.7 menjelaskan kasus uji untuk menampilkan berita terkini mengenai bahaya banjir.

Tabel 6.7 Kasus Uji untuk Pengujian Validasi Menampilkan Berita Terkini

Nama Kasus Uji	Kasus Uji Menampilkan Berita Terkini
Objek Uji	Kebutuhan Fungsional (SRS_001_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan informasi berita terkini mengenai bahaya banjir melalui *web service*
Prosedur Uji	1. Sistem dijalankan saat program dieksekusi 2. Aplikasi mendeteksi koneksi internet 3. Aplikasi menampilkan data melalui pemanggilan operasi *web service*
Hasil yang diharapkan	Aplikasi dapat menampilkan informasi berita terkini mengenai bahaya banjir melalui *web service*

Sumber : [Pengujian dan Analisis]

Hasil kasus uji menampilkan berita terkini mengenai bahaya banjir dapat dilihat pada gambar 6.14. Gambar 6.14 menjelaskan bahwa hasil pemanggilan operasi *web service* berupa informasi berita terkini mengenai bahaya banjir ditampilkan dalam bentuk *list.*

Gambar 6.14 Hasil Kasus Uji Menampilkan Berita Terkini
Sumber : [Pengujian dan Analisis]

b. Kasus Uji Menampilkan Status Siaga

Proses menampilkan status siaga banjir suatu daerah dilakukan dengan memanggil operasi *web service* tanpa parameter. Tabel 6.8 menjelaskan kasus uji untuk menampilkan status siaga banjir suatu daerah.

Tabel 6.8 Kasus Uji untuk Pengujian Validasi Menampilkan Status Siaga

Nama Kasus Uji	Kasus Uji Menampilkan Status Siaga
Objek Uji	Kebutuhan Fungsional (SRS_001_02)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan informasi status siaga banjir suatu daerah melalui *web service*
Prosedur Uji	1. Sistem dijalankan saat program dieksekusi 2. Aplikasi mendeteksi koneksi internet 3. Aplikasi menampilkan data melalui pemanggilan operasi *web service*
Hasil yang diharapkan	Aplikasi dapat menampilkan informasi status siaga banjir suatu daerah melalui *web service*

Sumber : [Pengujian dan Analisis]

Hasil kasus uji menampilkan status siaga banjir suatu daerah dapat dilihat pada gambar 6.15. Gambar 6.15 menunjukkan bahwa hasil pemanggilan operasi *web service* berupa informasi status siaga banjir suatu daerah ditampilkan dalam bentuk tabel.

Gambar 6.15 Hasil Kasus Uji Menampilkan Status Siaga
Sumber : [Pengujian dan Analisis]

c. Kasus Uji Login Sah

Informasi ketinggian air sungai suatu daerah dapat dilihat oleh pengguna ketika pengguna sudah dalam kondisi login. Kondisi login yang dimaksud ketika *username* dan *password* masukan pengguna sebagai parameter operasi *web service* bernilai benar. Gambar 6.9 menunjukkan kasus uji login sah yang dapat dilakukan oleh pengguna.

Tabel 6.9 Kasus Uji untuk Pengujian Validasi Login Sah

Nama Kasus Uji	Kasus Uji Login Sah
Objek Uji	Kebutuhan Fungsional (SRS_001_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menyediakan fasilitas login bagi PAB dan pengguna utama
Prosedur Uji	1. Sistem dijalankan saat program dieksekusi 2. Form login ditampilkan 3. PAB dan pengguna utama memasukkan *username* dan *password* di dalam *form* login 4. PAB dan pengguna utama menekan tombol login
Hasil yang diharapkan	Aplikasi dapat melakukan penyeleksian kondisi login berdasarkan data yang dimasukkan dan jika penyeleksian kondisi bernilai benar, maka pengguna dapat masuk ke dalam sistem untuk melihat ketinggian air sungai suatu daerah.

Sumber : [Pengujian dan Analisis]

Hasil kasus uji login sah dapat dilihat pada gambar 6.16. Gambar 6.16 menunjukkan bahwa antarmuka akan dialihkan ke dalam sebuah antarmuka yang digunakan untuk menampilkan ketinggian air sungai suatu daerah. Antarmuka tersebut akan ditampilkan ketika pengguna dalam kondisi login.

Gambar 6.16 Hasil Kasus Uji Login Sah
Sumber : [Pengujian dan Analisis]

d. Kasus Uji Login Tidak Sah

Informasi ketinggian air sungai suatu daerah dapat dilihat oleh pengguna ketika pengguna sudah dalam kondisi login. Kondisi login yang dimaksud ketika *username* dan *password* masukan pengguna sebagai parameter operasi *web service* bernilai salah. Gambar 6.10 menunjukkan kasus uji login tidak sah yang dapat dilakukan oleh pengguna.

Tabel 6.10 Kasus Uji untuk Pengujian Validasi Login Tidak Sah

Nama Kasus Uji	Kasus Uji Login Tidak Sah
Objek Uji	Kebutuhan Fungsional (SRS_001_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menyediakan fasilitas login bagi PAB dan pengguna utama
Prosedur Uji	1. Sistem dijalankan saat program dieksekusi 2. Form login ditampilkan 3. PAB dan pengguna utama memasukkan *username* dan *password* di dalam *form* login 4. PAB dan pengguna utama menekan tombol login

Hasil yang diharapkan	Aplikasi dapat melakukan penyeleksian kondisi login berdasarkan data yang dimasukkan dan jika penyeleksian kondisi bernilai salah, maka pengguna tidak dapat masuk ke dalam sistem untuk melihat ketinggian air sungai suatu daerah dan aplikasi akan menampilkan pesan kesalahan.

Sumber : [Pengujian dan Analisis]

Hasil kasus uji login tidak sah dapat dilihat pada gambar 6.17. Gambar 6.17 menunjukkan bahwa sistem akan menampilkan pesan kesalahan/*error* ketika username dan password yang dimasukkan oleh pengguna tidak valid.

Gambar 6.17 Hasil Kasus Uji Login Tidak Sah
Sumber : [Pengujian dan Analisis]

e. Kasus Uji Menampilkan Ketinggian Air Berdasarkan Harian

Proses menampilkan ketinggian air sungai suatu daerah dilakukan dengan memanggil operasi *web service*. Operasi tersebut membutuhkan beberapa parameter yaitu tanggal mulai, sampai dan nama daerah. Tabel 6.11 menjelaskan kasus uji untuk menampilkan ketinggian air sungai suatu daerah berdasarkan satuan waktu hari.

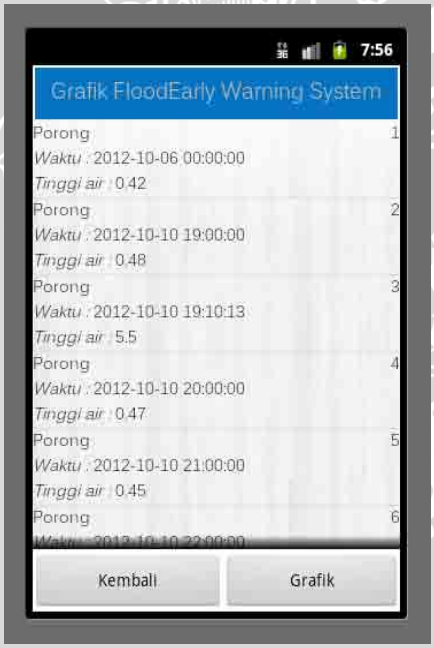
Tabel 6.11 Kasus Uji untuk Pengujian Validasi Menampilkan Ketinggian Air Berdasarkan Harian

Nama Kasus Uji	Kasus Uji Menampilkan Ketinggian Air Berdasarkan Harian
Objek Uji	Kebutuhan Fungsional (SRS_001_03)

Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu hari melalui *web service*
Prosedur Uji	1. Sistem dijalankan pada keadaan sudah login 2. PAB dan pengguna utama memasukkan parameter operasi *web service* tanggal mulai dan sampai serta nama daerah 3. PAB dan pengguna utama menekan tombol kirim
Hasil yang diharapkan	Aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu hari

Sumber : [Pengujian dan Analisis]

Hasil kasus uji menampilkan ketinggian air sungai suatu daerah dapat dilihat pada gambar 6.18. Gambar 6.18 menunjukkan bahwa informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu hari ditampilkan dalam bentuk *list*.



Gambar 6.18 Hasil Kasus Uji Menampilkan Ketinggian Air Berdasarkan Harian

Sumber : [Pengujian dan Analisis]

f. Kasus Uji Menampilkan Ketinggian Air Berdasarkan Mingguan

Proses menampilkan ketinggian air sungai suatu daerah dilakukan dengan memanggil operasi *web service*. Operasi tersebut membutuhkan beberapa parameter yaitu nama daerah, tahun, bulan dan minggu. Tabel 6.12 menjelaskan

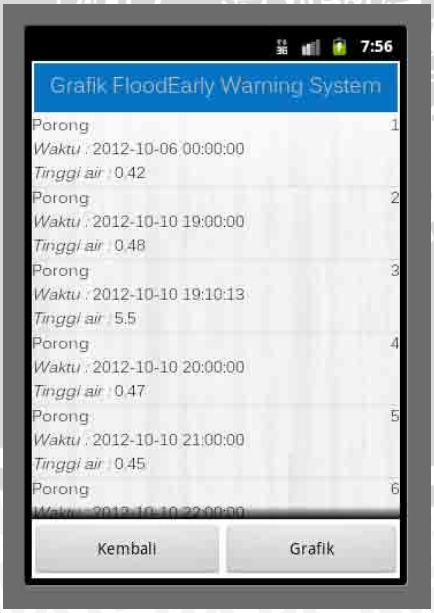
kasus uji untuk menampilkan ketinggian air sungai suatu daerah berdasarkan satuan waktu minggu.

Tabel 6.12 Kasus Uji untuk Pengujian Validasi Menampilkan Ketinggian Air Berdasarkan Mingguan

Nama Kasus Uji	Kasus Uji Menampilkan Ketinggian Air Berdasarkan Mingguan
Objek Uji	Kebutuhan Fungsional (SRS_001_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu minggu melalui *web service*
Prosedur Uji	1. Sistem dijalankan pada keadaan sudah login 2. PAB dan pengguna utama memasukkan parameter operasi *web service* daerah, tahun, bulan dan minggu 3. PAB dan pengguna utama menekan tombol kirim
Hasil yang diharapkan	Aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu minggu

Sumber : [Pengujian dan Analisis]

Hasil kasus uji menampilkan ketinggian air sungai suatu daerah dapat dilihat pada gambar 6.19. Gambar 6.19 menunjukkan bahwa informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu minggu ditampilkan dalam bentuk *list*.



Gambar 6.19 Hasil Kasus Uji Menampilkan Ketinggian Air Berdasarkan Mingguan

Sumber : [Pengujian dan Analisis]

g. Kasus Uji Menampilkan Ketinggian Air Berdasarkan Bulanan

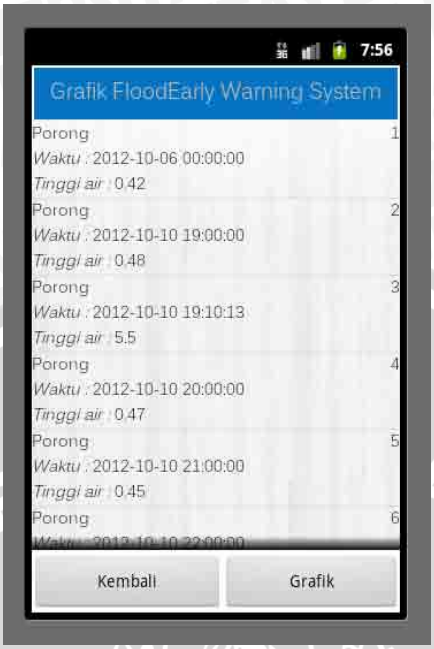
Proses menampilkan ketinggian air sungai suatu daerah dilakukan dengan memanggil operasi *web service*. Operasi tersebut membutuhkan beberapa parameter yaitu nama daerah, tahun dan bulan. Tabel 6.13 menjelaskan kasus uji untuk menampilkan ketinggian air sungai suatu daerah berdasarkan satuan waktu bulan.

Tabel 6.13 Kasus Uji untuk Pengujian Validasi Menampilkan Ketinggian Air Berdasarkan Bulanan

Nama Kasus Uji	Kasus Uji Menampilkan Ketinggian Air Berdasarkan Bulanan
Objek Uji	Kebutuhan Fungsional (SRS_001_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu bulan melalui *web service*
Prosedur Uji	1. Sistem dijalankan pada keadaan sudah login 2. PAB dan pengguna utama memasukkan parameter operasi *web service* daerah, tahun dan bulan 3. PAB dan pengguna utama menekan tombol kirim
Hasil yang diharapkan	Aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu bulan

Sumber : [Pengujian dan Analisis]

Hasil kasus uji menampilkan ketinggian air sungai suatu daerah dapat dilihat pada gambar 6.20. Gambar 6.20 menunjukkan bahwa informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu hari ditampilkan dalam bentuk *list*.



Gambar 6.20 Hasil Kasus Uji Menampilkan Ketinggian Air Berdasarkan Bulanan

Sumber : [Pengujian dan Analisis]

6.1.3.2 Hasil Pengujian Validasi

Hasil pengujian validasi didapatkan dari kasus uji yang sudah dilakukan. Hasil pengujian validasi memuat beberapa kategori antara lain yaitu nama kasus uji, hasil yang diharapkan, hasil yang didapatkan dan status validitas. Tabel 6.14 menunjukkan hasil pengujian validasi.

Tabel 6.14 Hasil Pengujian Validasi

No.	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validitas
1	Kasus Uji Menampilkan Berita Terkini	Aplikasi dapat menampilkan informasi berita terkini mengenai bahaya banjir melalui *web service*	Aplikasi dapat menampilkan informasi berita terkini mengenai bahaya banjir melalui *web service*	Valid
2	Kasus Uji Menampilkan Status Siaga	Aplikasi dapat menampilkan informasi status siaga banjir suatu daerah melalui *web service*	Aplikasi dapat menampilkan informasi status siaga banjir suatu daerah melalui *web service*	Valid
3	Kasus Uji Login Sah	Aplikasi dapat melakukan	Aplikasi dapat melakukan	Valid

		penyeleksian kondisi login berdasarkan data yang dimasukkan dan jika penyeleksian kondisi bernilai benar, maka pengguna dapat masuk ke dalam sistem untuk melihat ketinggian air sungai suatu daerah.	penyeleksian kondisi login berdasarkan data yang dimasukkan dan jika penyeleksian kondisi bernilai benar, maka pengguna dapat masuk ke dalam sistem untuk melihat ketinggian air sungai suatu daerah.	
4	Kasus Uji Login Tidak Sah	Aplikasi dapat melakukan penyeleksian kondisi login berdasarkan data yang dimasukkan dan jika penyeleksian kondisi bernilai salah, maka pengguna tidak dapat masuk ke dalam sistem untuk melihat ketinggian air sungai suatu daerah dan aplikasi akan menampilkan pesan kesalahan.	Aplikasi dapat melakukan penyeleksian kondisi login berdasarkan data yang dimasukkan dan jika penyeleksian kondisi bernilai salah, maka pengguna tidak dapat masuk ke dalam sistem untuk melihat ketinggian air sungai suatu daerah dan aplikasi akan menampilkan pesan kesalahan.	Valid
5	Kasus Uji Menampilkan Ketinggian Air Berdasarkan Harian	Aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu hari	Aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu hari	Valid
6	Kasus Uji Menampilkan Ketinggian Air Berdasarkan Mingguan	Aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu minggu	Aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu minggu	Valid

7	Kasus Uji Menampilkan Ketinggian Air Berdasarkan Bulanan	Aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu bulan	Aplikasi dapat menampilkan informasi mengenai ketinggian air sungai suatu daerah berdasarkan satuan waktu bulan	Valid

Sumber : [Pengujian dan Analisis]

6.1.4 Pengujian Kompatibilitas

Pengujian kompatibilitas digunakan untuk mengetahui kompatibilitas antarmuka sistem pada tiga sistem operasi android yaitu sistem operasi android versi 2.3, sistem operasi android versi 4.0 dan sistem operasi android 4.1.

6.1.4.1 Pengujian Kompatibilitas Sistem Operasi Android Versi 2.3

Pengujian kompatibilitas sistem operasi android versi 2.3 dilakukan untuk mengetahui validitas kinerja fitur-fitur yang disediakan oleh sistem terkait dengan antarmuka sistem. Tabel 6.15 menjelaskan prosedur dan hasil kasus uji pengujian kompatibilitas sistem operasi android versi 2.3.

Tabel 6.15 Pengujian Kompatibilitas Sistem Operasi Android Versi 2.3

Nama Kasus Uji	Pengujian kompatibilitas sistem operasi android versi 2.3
Objek Uji	Kebutuhan Nonfungsional
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui validitas kinerja dari sistem dalam menyediakan fitur-fitur dan antarmuka
Prosedur Uji	Membuka setiap halaman sesuai dengan spesifikasi kebutuhan sistem
Hasil yang diharapkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem
Hasil yang didapatkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem
Status Validitas	Valid

Sumber : [Pengujian dan Analisis]

6.1.4.2 Pengujian Kompatibilitas Sistem Operasi Android Versi 4.0

Pengujian kompatibilitas sistem operasi android versi 4.0 dilakukan untuk mengetahui validitas kinerja fitur-fitur yang disediakan oleh sistem terkait dengan

antarmuka sistem. Tabel 6.16 menunjukkan prosedur dan hasil kasus uji pengujian kompatibilitas sistem operasi android versi 4.0.

Tabel 6.16 Pengujian Kompatibilitas Sistem Operasi Android Versi 4.0

Nama Kasus Uji	Pengujian kompatibilitas sistem operasi android versi 4.0
Objek Uji	Kebutuhan Nonfungsional
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui validitas kinerja dari sistem dalam menyediakan fitur-fitur dan antarmuka
Prosedur Uji	Membuka setiap halaman sesuai dengan spesifikasi kebutuhan sistem
Hasil yang diharapkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem
Hasil yang didapatkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem
Status Validitas	Valid

Sumber : [Pengujian dan Analisis]

6.1.4.3 Pengujian Kompatibilitas Sistem Operasi Android Versi 4.1

Pengujian kompatibilitas sistem operasi android versi 4.1 dilakukan untuk mengetahui validitas kinerja fitur-fitur yang disediakan oleh sistem terkait dengan antarmuka sistem. Tabel 6.17 menunjukkan prosedur dan hasil kasus uji pengujian kompatibilitas sistem operasi android versi 4.1.

Tabel 6.17 Pengujian Kompatibilitas Sistem Operasi Android Versi 4.1

Nama Kasus Uji	Pengujian kompatibilitas sistem operasi android versi 4.1
Objek Uji	Kebutuhan Nonfungsional
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui validitas kinerja dari sistem dalam menyediakan fitur-fitur dan antarmuka
Prosedur Uji	Membuka setiap halaman sesuai dengan spesifikasi kebutuhan sistem
Hasil yang diharapkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem
Hasil yang didapatkan	Sistem dapat menampilkan fitur-fitur dan antarmuka sesuai dengan implementasi antarmuka sistem
Status Validitas	Valid

Sumber : [Pengujian dan Analisis]

6.1.5 UAT

UAT digunakan untuk mengetahui apakah sistem yang dibangun atau dikembangkan telah dapat diterima atau belum oleh pengguna sistem. Segi antarmuka sistem, kejelasan menu atau navigasi sistem, detail data berita terkini mengenai bahaya banjir, detail data status siaga banjir suatu daerah, detail data ketinggian air sungai suatu daerah dan kualitas keseluruhan sistem harus dapat diterima atau tidak oleh pengguna. UAT dilakukan dengan memberikan kuesioner kepada satu orang PAB, dua orang pengguna utama dan sepuluh orang pengguna biasa untuk menilai keseluruhan sistem dan memberikan komentar maupun saran terhadap sistem.

Tabel 6.18 Hasil Kuisioner *User Acceptance Testing*

No.	Pertanyaan	Jumlah Penilai				
		Sangat Kurang	Kurang	Cukup	Bagus	Sangat Bagus
1.	Bagaimana kemudahan navigasi atau menu untuk menemukan informasi yang Anda butuhkan?	-	-	1	10	2
2.	Bagaimana penampilan visual aplikasi?	-	-	1	8	4
3.	Bagaimana relevansi informasi terkait dengan bahaya banjir di dalam aplikasi?	-	-	-	7	6
4.	Bagaimana relevansi menu terkait dengan informasi bahaya banjir di dalam aplikasi?	-	1	-	9	3
5.	Bagaimana kejelasan informasi tentang bahaya banjir di dalam aplikasi?	-	-	2	9	2
6.	Bagaimana kualitas keseluruhan aplikasi?	-	-	1	8	4
Jumlah		-	1	5	51	21

Sumber : [Pengujian dan Analisis]

Tabel 6.18 menjelaskan bahwa aplikasi yang sudah dibuat memiliki tingkat kelayakan dalam kriteria bagus sesuai dengan tanggapan dari beberapa responden atau pengguna aplikasi.

6.2 Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian Aplikasi Perangkat Bergerak Penanda Dini Bahaya Banjir Menggunakan *Web service* Berbasis SOAP yang telah dilakukan. Proses analisis mengacu pada dasar teori sesuai dengan hasil pengujian yang didapatkan. Analisis dilakukan terhadap hasil pengujian di setiap tahap pengujian. Proses analisis yang dilakukan meliputi analisis hasil pengujian unit, analisis hasil pengujian integrasi, analisis hasil pengujian validasi dan analisis hasil UAT.

6.2.1 Analisis Hasil Pengujian Unit

Proses analisis terhadap hasil pengujian unit dilakukan dengan melihat kesesuaian fungsi dari implementasi unit modul yang diuji dengan hasil perancangan perangkat lunak yang telah dirancang sebelumnya. Berdasarkan hal tersebut, maka dapat diambil kesimpulan bahwa unit modul dari program sudah dapat memenuhi kebutuhan fungsional yang telah dirancang pada tahap perancangan.

6.2.2 Analisis Hasil Pengujian Integrasi

Proses analisis terhadap hasil pengujian integrasi dilakukan dengan melihat kesesuaian beberapa unit modul yang menyusun satu blok fungsi dalam Aplikasi Perangkat Bergerak Penanda Dini Bahaya Banjir Menggunakan *Web service* Berbasis SOAP. Berdasarkan hal tersebut, maka dapat diambil kesimpulan bahwa integrasi dari beberapa unit modul dari program sudah memenuhi kebutuhan fungsional yang telah dirancang pada tahap perancangan.

6.2.3 Analisis Hasil Pengujian Validasi

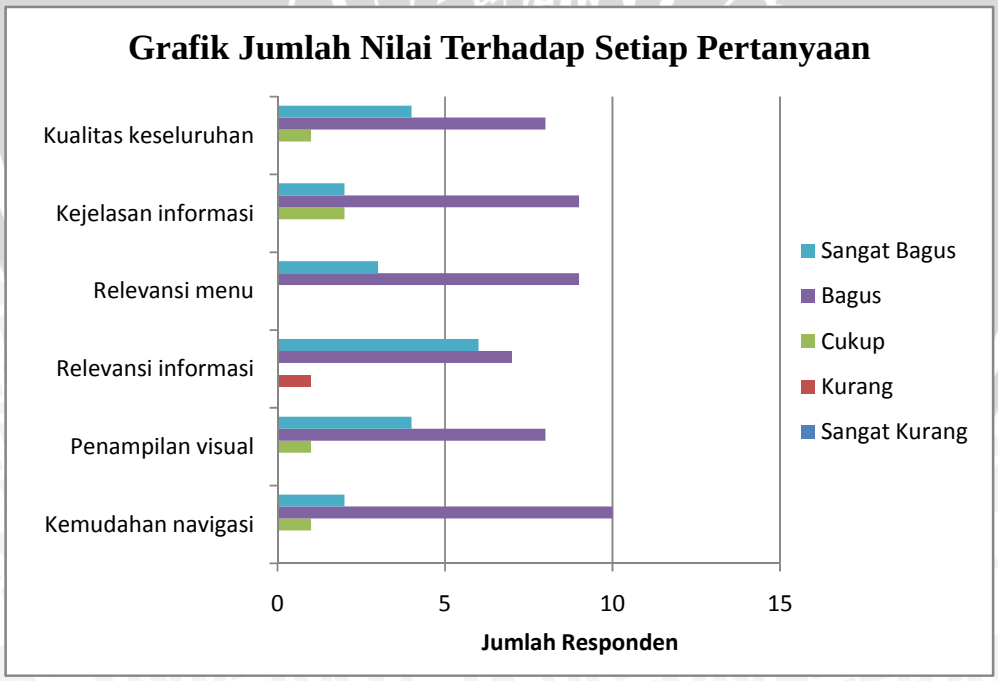
Proses analisis terhadap hasil pengujian validasi dilakukan dengan melihat kesesuaian antara hasil kinerja sistem dengan daftar kebutuhan. Berdasarkan hasil pengujian validasi dapat disimpulkan bahwa implementasi dan fungsionalitas Aplikasi Perangkat Bergerak Penanda Dini Bahaya Banjir Menggunakan *Web service* Berbasis SOAP telah memenuhi kebutuhan yang telah dijabarkan pada tahap analisis kebutuhan.

6.2.4 Analisis Hasil Pengujian Kompatibilitas

Proses analisis terhadap hasil pengujian kompatibilitas pada beberapa sistem operasi android yaitu sistem operasis android versi 2.3, sistem operasis android versi 4.0 dan sistem operasis android versi 4.1 dilakukan dengan melihat kesesuaian antara hasil kinerja sistem dengan daftar kebutuhan. Berdasarkan hasil pengujian kompatibilitas dapat disimpulkan bahwa implementasi dan fungsionalitas Aplikasi Perangkat Bergerak Penanda Dini Bahaya Banjir Menggunakan *Web service* Berbasis SOAP telah memenuhi kebutuhan yang telah dijabarkan pada tahap analisis kebutuhan.

6.2.5 Analisis Hasil Pengujian UAT

Proses analisis terhadap hasil pengujian sistem terhadap pengguna dilakukan dengan menghitung jumlah tiap nilai dari semua tanggapan responden. Pengujian sistem terhadap pengguna memiliki lima tingkatan penilaian dengan mengajukan enam pertanyaan terhadap sistem kepada responden. Gambar 6.21 menjelaskan tentang tanggapan dari beberapa responden yang direpresentasikan ke dalam sebuah tingkatan penilaian yaitu sangat bagus, bagus, cukup, kurang dan sangat kurang.



Gambar 6.21 Grafik Jumlah Nilai Terhadap Setiap Pertanyaan
Sumber : [Pengujian dan Analisis]

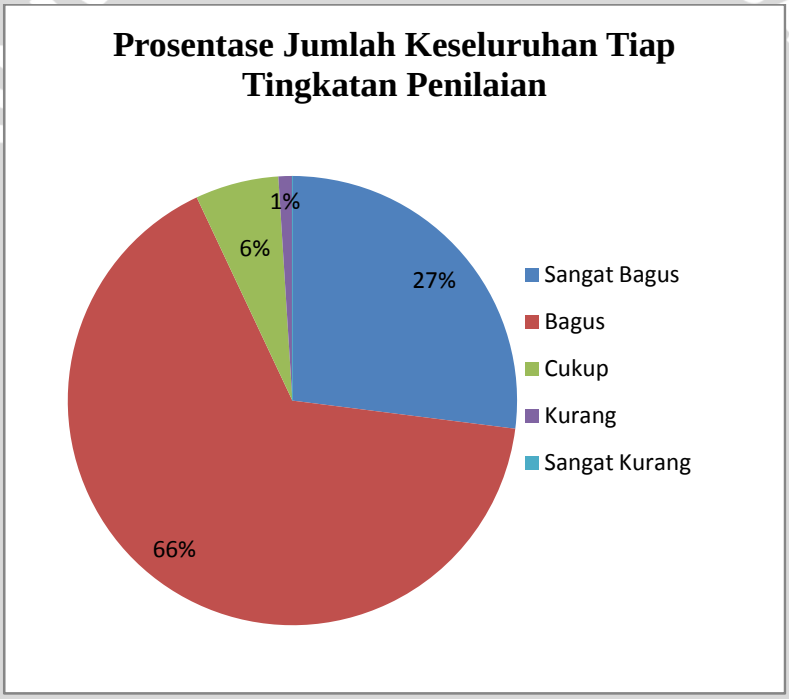
Adapun tabel 6.19 menjelaskan tentang prosentase dari jumlah keseluruhan dari setiap penilaian yang diberikan oleh responden.

Tabel 6.19 Prosentase Jumlah Keseluruhan Tiap Tingkatan Penilaian

Tingkatan Penilaian	Jumlah Keseluruhan Penilaian	Prosentase
Sangat Bagus	21	27 %
Bagus	51	66%
Cukup	5	6%
Kurang	1	1%
Sangat Kurang	0	0%

Sumber : [Pengujian dan Analisis]

Setiap nilai dari masing-masing pertanyaan akan dijumlahkan. Penjumlahan tersebut dimaksudkan untuk mengetahui rata-rata keseluruhan penilaian yang diberikan oleh responden.



Gambar 6.22 Grafik Prosentase Jumlah Keseluruhan Tiap Tingkatan Penilaian

Sumber : [Pengujian dan Analisis]

Gambar 6.22 menunjukkan bahwa perangkat lunak yang sudah dibuat memiliki tanggapan yang bagus dari responden. 66% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori bagus, 6% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori cukup, 27% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori

sangat bagus dan 1% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori kurang.

Sistem pendukung keputusan penanda dini bahaya banjir masih memerlukan beberapa pembenahan dari segi antarmuka pengguna. Proses pembenahan tersebut dilakukan untuk merepresentasikan tanggapan tentang komentar dan saran yang diberikan oleh beberapa responden. Antarmuka pengguna yang masih perlu dibenahi adalah antarmuka untuk menampilkan ketinggian air sungai suatu daerah ke dalam bentuk grafik. Grafik yang dimaksud adalah menampilkan data waktu pada koordinat sumbu X dengan format “yyyy-MM-dd HH:mm:ss”.

BAB VII
PENUTUP

7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang telah dilakukan, maka diambil kesimpulan sebagai berikut.

1. Perancangan sistem informasi pendukung keputusan tentang bahaya banjir telah dibuat sesuai dengan spesifikasi kebutuhan yang telah dianalisa.
2. Model pengembangan perangkat lunak sistem informasi pendukung keputusan tentang bahaya banjir menggunakan model *reuse-oriented software engineering*. Dengan model tersebut mudah dalam proses pengembangan sistem dengan menambahkan atau mengubah beberapa komponen dalam sistem maupun menggunakan beberapa komponen lainnya yang telah tersedia.
3. Sistem informasi pendukung keputusan tentang bahaya banjir telah dibuat sesuai perancangan dan diimplementasikan dari komponen-komponen yang telah ditentukan.
4. Berdasarkan dari hasil pengujian unit dan integrasi dengan metode *White-Box Testing* didapatkan hasil yang sangat baik dengan ditunjukkannya jalur kontrol dan beberapa proses yang dilakukan sistem sudah sesuai dengan perancangan sistem.
5. Hasil pengujian validasi dengan metode *Black-Box Testing* pada sistem menunjukkan nilai dengan prosentase 100%. Sistem sudah memenuhi spesifikasi kebutuhan yang telah dianalisa.
6. Berdasarkan dari hasil *User Acceptance Testing*, sistem telah dapat diterima oleh pengguna. Hasil yang diperoleh menunjukkan 66% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori bagus, 6% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori cukup, 27% tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori sangat bagus dan 1%

tanggapan responden memberikan penilaian terhadap aplikasi dengan kategori kurang.

7.2 Saran

Saran yang diberikan untuk pengembangan penelitian selanjutnya antara lain sebagai berikut.

1. Perlu adanya pengembangan aplikasi ke dalam *platform* atau lingkungan sistem operasi yang berbeda agar dapat membedakan pengembangan teknologi *web service*.
2. Untuk pengembangan lebih lanjut aplikasi ini dapat dilakukan dengan mengotomatisasi pembaharuan informasi yang disediakan oleh sistem dengan melakukan pemanggilan operasi *web service*.
3. Perlu adanya pengembangan dan pembenahan pada tampilan antarmuka pengguna yaitu antarmuka untuk menampilkan ketinggian air sungai suatu daerah dalam bentuk grafik. Grafik yang dimaksud adalah menampilkan data waktu pada koordinat sumbu X dengan format “yyyy-MM-dd HH:mm:ss”.
4. Perangkat lunak ini bisa terealisasikan atau bukan dalam bentuk *prototype* lagi sehingga pengguna bisa menggunakan dan mendapatkan informasi tentang bahaya banjir.

DAFTAR PUSTAKA

[ACE-09] AChartEngine, diakses pada 24 Juli 2013, http://www.achartengine.org/content/news.html.

[DEV-11] Deviana, Hartati. 2011, ”Penerapan Xml Web Service pada Sistem Distribusi Barang”, Generic, Vol. 6, No. 2, hal. 55-62.

[DSA-03] Dharwiyanti, Sri., Satria Wahono, Romi. 2003, “Pengantar Unified Modeling Language (UML)”, Kuliah Umum IlmuKomputer.com.

[KSP-06] kSOAP 2, diakses pada 04 Juni 2013, http://ksoap2.sourceforge.net.

[MAR-11] Gargeta, Marko. 2011, “Learning Android”, O’Reilly Media, Inc. Sebastopol.

[MBR-13] Maxim, Bruce R. 2013, “Software Testing Strategies”. Diakses pada 24 Juli 2013, http://www.learningace.com/doc/2027567/0b918c897e643de4e7447f663a111bc7/lec25

[PRR-01] Pressman, Roger S.. 2001, “Software Engineering A Practitioner’s Approach fifth edition”, McGraw-Hill, New York.

[PTK-05] Priyambodo, Tri Kuntoro. 2005, “Implementasi Web-Service Untuk Pengembangan Layanan Pariwisata Terpadu”, TEKNOIN, Vol.10, No.2, Yogyakarta.

[RJB-98] Rumbaugh, James, Jacobson, Ivar, Booch, Grady. 199, “The Unified Modeling Language Reference Manual”, Addison Wesley, Canada.

[SKG-08] Sahin,K., Gumusay, M.U.. 2008, “Service Oriented Architecture (SOA) Based Web Service For Geograpic Information Systems”, The Internasional Archives of The Photogrammetry Remote Sensing and Spatial Information Sciences Vol. XXXVII Part B2.

[SOM-11] Sommerville, Ian. 2011, “Software Engineering, 9th edition”, Addison-Wesley, New York.

[SSE-13] Soebroto, Andy Arief, Soekotjo, Harry, Suhartanto, Ery, 2013, “Prototype of the Real Time Decision Support System for Flood Early Warning at Brantas River Basin”, IJEIR, Vol. 2, Issue 3, ISSN: 2277 – 5668.

LAMPIRAN

Lampiran 1 Kuisioner *User Acceptance Testing*

Kuesioner Pengujian Penerimaan Terhadap Pengguna Rancang Bangun Aplikasi Perangkat Bergerak Penanda Dini Bahaya Banjir Menggunakan Web Service Berbasis SOAP

Nama : _____

Jenis Kelamin : _____

Umur : _____

Pekerjaan : _____

Berikan tanda cek (v) pada nilai yang Anda anggap paling sesuai!

No.	Pertanyaan	Nilai				
		Sangat Kurang	Kurang	Cukup	Bagus	Sangat Bagus
1.	Bagaimana kemudahan navigasi atau menu untuk menemukan informasi yang Anda butuhkan?					
2.	Bagaimana penampilan visual aplikasi?					
3.	Bagaimana relevansi informasi terkait dengan bahaya banjir di dalam aplikasi?					
4.	Bagaimana relevansi menu terkait dengan informasi bahaya banjir di dalam aplikasi?					
5.	Bagaimana kejelasan informasi tentang bahaya banjir di dalam aplikasi?					
6.	Bagaimana kualitas keseluruhan aplikasi?					

7.	Komentar dan saran	

Malang,

Responden

_____