RANCANG BANGUN APLIKASI *MOBILE* PENGELOLA KEUANGAN PRIBADI

SKRIPSI

LABORATORIUM PERANGKAT BERGERAK

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

AKHMAD SYAIFUL YAMANG

NIM. 105060800111070

KEMENTERIAN RISET TEKNOLOGI DAN PENDIDIKAN TINGGI

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER

MALANG

2015

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI *MOBILE* PENGELOLA KEUANGAN PRIBADI

SKRIPSI

LABORATORIUM PERANGKAT BERGERAK

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh:

AKHMAD SYAIFUL YAMANG

NIM. 105060800111070

Skripsi ini telah disetujui oleh dosen pembimbing pada tanggal 22 Desember 2014

Dosen Pembimbing I	**Dosen Pembimbing II**
Aryo Pinandito, ST, M.MT	**Herman Tolle, Dr. Eng, ST., MT.**
NIK. 83051916110374	**NIP. 19740823 200012 1 001**

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI *MOBILE* PENGELOLA KEUANGAN PRIBADI

SKRIPSI

LABORATORIUM PERANGKAT BERGERAK

Untuk memenuhi persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh :

AKHMAD SYAIFUL YAMANG

NIM. 105060800111070

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 9 Januari 2015
dan dinyatakan memenuhi syarat untuk memperoleh
gelar Sarjana dalam bidang Ilmu Komputer

Penguji I,	**Penguji II,**
Agi Putra Kharisma, S.T., M.T.	**Fajar Pradana, S.ST, M.Eng**
NIK. -	**NIK. 87112116110371**

Penguji III,

Aswin Suharsono, ST., MT.
NIK. 840919 06 1 1 0251

Mengetahui,
Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, MT
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, 9 Januari 2015

Penulis,

Akhmad Syaiful Yamang

NIM. 105060800111070

KATA PENGANTAR

Puji dan syukur Penulis panjatkan kehadirat Allah SWT, karena hanya dengan rahmat dan bimbingannya Penulis dapat menyelesaikan tugas akhir dengan judul "Rancang Bangun Aplikasi *Mobile* Pengelola Keuangan Pribadi" dengan baik. Tanpa rahmat dan bimbingan dari Tuhan Yang Maha Esa, maka niscaya Penulis tidak akan dapat menyelesaikan tugas akhir ini dengan baik dan tepat waktu.

Penulisan dan penyusunan laporan skripsi ini dapat terlaksana dengan baik karena adanya bantuan secara langsung maupun tidak langsung dari pihak tertentu diantaranya:

1. Bapak Aryo Pinandito, ST, M.MT selaku dosen pembimbing I yang telah banyak memberikan ilmu dan saran untuk laporan skripsi ini.
2. Bapak Herman Tolle, Dr. Eng, ST., MT. selaku dosen pembimbing II yang juga banyak memberikan ilmu dan saran untuk laporan skripsi ini.
3. Kedua orang tua Ali Yamang Hasan S.H., M.M. dan Dra. Hamsinah B. yang telah memberi motivasi, kasih sayang serta dukungan moril dan materil. Kakak Syahrul Mubarak Y., Kakak Syahriani Rezki A.Y., Adek Sri Rochmani P.Y. serta Dita Oktaria yang telah memberikan semangat dari awal sampai akhir pengerjaan skripsi ini.
4. Seluruh rekan kerja dan supervisor BPTIK PTIIK UB, terima kasih untuk segala pengalaman dan pelajaran selama bergabung bersama kalian.
5. Semua teman-teman PTIIK, khususnya Informatika/Ilmu Komputer 2010 terima kasih atas segala bantuan dan dukungannya selama ini.
6. Segenap dosen dan karyawan PTIIK Universitas Brawijaya yang telah membantu pelaksanaan skripsi ini.
7. Semua teman-teman kontrakan manggar yang telah membantu dalam hal dukungan moril.
8. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT. Dengan segala kerendahan hati, penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari sempurna karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Akhirnya semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa PTIIK Universitas Brawijaya.

Malang, Januari 2015

Penulis

ABSTRAK

Akhmad Syaiful Yamang. 2014. Rancang Bangun Aplikasi *Mobile* Pengelola Keuangan Pribadi. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing Aryo Pinandito, ST, M.MT dan Herman Tolle, Dr. Eng, ST., MT.

Keuangan dapat dikatakan sehat apabila seseorang mampu menyelaraskan antara penggunaan dana dan pemasukan. Kesalahan dalam mengelola keuangan mengakibatkan memburuknya kesehatan keuangan. Pengelolaan keuangan yang baik dapat membantu mewujudkan suatu keuangan yang sehat. Dalam mengelola keuangan, diperlukan suatu pengalokasian yang dapat membantu dalam memastikan penerimaan dan pengeluaran. Selain pengalokasian, dalam mengelola keuangan juga diperlukan pencatatan transaksi dan informasi lain seperti rekening dan jadwal pembayaran. Hal ini penting, terutama bagi orang-orang yang memiliki kesibukan yang padat.

Penelitian ini bermaksud untuk menyediakan media yang dapat membantu dalam pengalokasian dana dan mencatat informasi-informasi terkait proses transaksi. Media tersebut adalah sebuah aplikasi pada perangkat *mobile* yang dapat digunakan dimana saja tanpa perlu koneksi internet. Dalam penelitian ini sistem dibangun menggunakan sebuah konsep pengembangan dengan bahasa pemrograman HTML5, CSS3 serta *Javascript*, dan dimaksimalkan pada *layout interface* menggunakan Jquerymobile, atau yang sering disebut konsep *hybrid*, dengan *framework* Phonegap yang diterapkan pada perangkat Android dan iOS. Teknik penyimpanan data menggunakan *local storage* dengan DBMS SQLite sehingga data tidak dapat diakses dari luar.

Dari penelitian yang dilakukan, aplikasi yang dibuat telah sesuai dengan spesifikasi kebutuhan dan perancangan yang telah dianalisis.

Kata kunci*:* Keuangan, pengelolaan, pengalokasian, pencatatan, *hybrid*.

ABSTRACT

Akhmad Syaiful Yamang. 2014. Rancang Bangun Aplikasi *Mobile* Pengelola Keuangan Pribadi. Information Technology and Computer Science Program, Brawijaya University, Malang. *Advisor:* Aryo Pinandito, ST, M.MT and Herman Tolle, Dr. Eng, ST., MT.

Financial Health is called when someone is able to harmonize the use of funds and income. Mistakes in financial management resulted in deteriorating financial health. Good financial management can help realize a healthy financial. In the financial management, allocation is required to help in ensuring revenues and expenditures. In addition to the allocation, the recording is also required in financial management, both transaction or other information such as account info and payment schedule. This is important, especially for people who have a solid rushing.

This study intends to provide an offline mobile application for helping people allocate the funds and record all financial transaction information anytime anywhere. System built with HTML5, CSS3, Javascript, JqueryMobile and PhoneGap as a hybrid application that implemented on Android and iOS devices. System also use SQLite DBMS as local storage that only accessed through the application.

From the research conducted, the application has fulfilled the analyzed requirements and design.

Keywords: Financial, management, allocation, records, hybrid.

DAFTAR ISI

KATA PENGANTAR i
ABSTRAK iii
ABSTRACT iv
DAFTAR ISI v
DAFTAR GAMBAR viii
DAFTAR TABEL xi
DAFTAR KODE xiii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan 3
1.5 Manfaat 4
1.6 Sistematika Penulisan 4
BAB II KAJIAN PUSTAKA DAN DASAR TEORI 6
2.1 HTML5 6
2.2 CSS 6
2.3 Javascript 7
2.4 jQuery Mobile 8
2.5 PhoneGap 9
2.6 SQLite 11
2.6.1 Kelas Penyimpanan dan Jenis Data 13

BAB III METODOLOGI PENELITIAN.....13
3.1 Studi Literatur.....14
3.2 Analisis Kebutuhan.....14
3.3 Perancangan.....14
3.4 Implementasi.....15
3.5 Pengujian dan Analisis.....16
3.6 Pengambilan Kesimpulan dan Saran.....16
BAB IV ANALISIS DAN PERANCANGAN.....17
4.1 Analisis Kebutuhan.....17
4.1.1 Gambaran Umum Aplikasi.....17
4.1.2 Identifikasi Aktor.....18
4.1.3 Analisis Kebutuhan Fungsional.....18
4.1.4 Analisis Kebutuhan Non-Fungsional.....27
4.2 Perancangan.....27
4.2.1 Perancangan Umum Sistem.....28
4.2.2 Perancangan Perangkat Lunak.....28
BAB V IMPLEMENTASI DAN PENGUJIAN.....44
5.1 Implementasi.....44
5.1.1 Spesifikasi Perangkat Keras dan Perangkat Lunak Sistem.....45
5.1.2 Batasan-batasan Implementasi.....45
5.1.3 Implementasi Basis Data.....45
5.1.4 Implementasi Class dan Assets Pada File Program.....48
5.1.5 Implementasi Kode Program.....56
5.1.6 Implementasi Antarmuka Aplikasi.....60
5.2 Pengujian.....68
5.2.1 Pengujian Validasi.....68

5.2.2 Pengujian Unit.....87
5.2.3 Pengujian Kompatibilitas.....96
5.3 Analisis Hasil Pengujian.....96
5.3.1 Analisis Hasil Pengujian Validasi.....97
5.3.2 Analisis Hasil Pengujian Unit.....97
5.3.3 Analisis Hasil Pengujian Kompatibilitas.....97
BAB VI PENUTUP.....98
6.1 Kesimpulan.....98
6.2 Saran.....98
DAFTAR PUSTAKA.....100

DAFTAR GAMBAR

Gambar 2.1 Tradisional RDBMS client/server arsitektur....12
Gambar 2.2 Arsitektur SQLite....13
Gambar 3.1 Diagram Alir Penelitian....13
Gambar 3.2 Diagram Pohon Perancangan....15
Gambar 4.1 Diagram *Use case*....20
Gambar 4.2 Diagram Blok Perancangan Umum Sistem....28
Gambar 4.3 Perancangan Arsitektur Sistem Aplikasi....29
Gambar 4.4 *Entity Relational Diagram*....31
Gambar 4.5 *Activity Diagram* Mengelola Transaksi....35
Gambar 4.6 *Activity Diagram* Mengelola *Reminder* (Pengingat)....36
Gambar 4.7 *Activity Diagram* Mengelola Memo....37
Gambar 4.8 *Activity Diagram* Mengelola Kategori....38
Gambar 4.10 Antarmuka Halaman *Welcome Screen*....40
Gambar 4.11 Antarmuka Halaman Home....40
Gambar 4.12 Antarmuka Halaman Sub Menu Transaksi....41
Gambar 4.13 Antarmuka Halaman Mengelola Transaksi....41
Gambar 4.14 Antarmuka Halaman Mengelola Kategori....42
Gambar 4.15 Antarmuka Halaman Mengelola Reminder....42
Gambar 4.16 Antarmuka Halaman Mengelola Memo....43
Gambar 5.1 Diagram Pohon Implementasi dan Pengujian....44
Gambar 5.2 *Physical Diagram*....46
Gambar 5.3 Tampilan Antarmuka Halaman *Welcome Screen*....61
Gambar 5.4 Tampilan Antarmuka Halaman *Home*....63

Gambar 5.5 Tampilan Antarmuka Sub Menu Transaksi 64
Gambar 5.6 Tampilan Antarmuka Menu Daftar Transaksi 65
Gambar 5.7 Tampilan Antarmuka Menu Kategori Pengeluaran 66
Gambar 5.8 Tampilan Antarmuka Menu *Reminder*.....67
Gambar 5.9 Tampilan Antarmuka Menu Memo.....68
Gambar 5.10 Kasus uji tambah transaksi.....70
Gambar 5.11 Kasus uji edit transaksi.....71
Gambar 5.12 Kasus uji ubah transaksi.....72
Gambar 5.13 Kasus uji hapus transaksi 73
Gambar 5.14 Kasus uji tambah kategori pengeluaran 74
Gambar 5.15 Kasus uji edit kategori pengeluaran 76
Gambar 5.16 Kasus uji ubah kategori pengeluaran 77
Gambar 5.17 Kasus uji hapus kategori pengeluaran.....78
Gambar 5.18 Kasus uji tambah reminder.....80
Gambar 5.19 Kasus uji ubah status reminder.....81
Gambar 5.20 Kasus uji hapus reminder 82
Gambar 5.21 Kasus uji tambah memo 83
Gambar 5.22 Kasus uji edit memo.....84
Gambar 5.23 Kasus uji ubah memo 86
Gambar 5.24 Kasus uji hapus memo.....87
Gambar 5.25 Gambar keadaan awal database tabel tbl_setting_anggaran 88
Gambar 5.26 Hasil pengujian unit kasus uji pertama 89
Gambar 5.27 Hasil pengujian unit kasus uji kedua.....91
Gambar 5.28 Keadaan database setelah pengujian kasus uji kedua.....91
Gambar 5.29 Hasil pengujian unit kasus uji ketiga.....93

Gambar 5.30 Keadaan database setelah pengujian kasus uji ketiga 93
Gambar 5.31 Hasil pengujian unit kasus uji keempat.....95
Gambar 5.32 Keadaan database setelah pengujian kasus uji keempat.....96

DAFTAR TABEL

Tabel 2.1 Fitur-fitur PhoneGap 10
Tabel 2.2 Tipe Data SQLite 13
Tabel 4.1 Informasi Kelebihan dan Kekurangan Sistem Terkait 17
Tabel 4.2 Identifikasi Aktor 18
Tabel 4.3 Spesifikasi Kebutuhan Fungsional 19
Tabel 4.4 Skenario *Use Case* Mengelola Transaksi 21
Tabel 4.5 Skenario *Use Case* Mengelola Kategori Pemasukan 23
Tabel 4.6 Skenario *Use Case* Mengelola *Reminder* (Pengingat) 24
Tabel 4.7 Skenario *Use Case* Mengelola Memo 26
Tabel 4.8 Spesifikasi Kebutuhan Non-Fungsional 27
Tabel 4.9 Struktur Tabel tbl_user 32
Tabel 4.10 Struktur Tabel tbl_jenis_transaksi 32
Tabel 4.11 Struktur Tabel tbl_jenis_pembayaran 32
Tabel 4.12 Struktur Tabel Memo 33
Tabel 4.13 Struktur Tabel tbl_mata_uang 33
Tabel 4.14 Struktur Tabel tbl_reminder 33
Tabel 4.15 Struktur Tabel tbl_setting_anggaran 34
Tabel 4.16 Struktur Tabel tbl_transaksi 34
Tabel 5.1 Implementasi class pada kode program *.java 48
Tabel 5.2 Implementasi assets pada kode program *.html 49
Tabel 5.3 Implementasi assets pada kode program *.js 56
Tabel 5.4 Kasus Uji Validasi Tambah Transaksi 69
Tabel 5.5 Kasus Uji Validasi Edit Transaksi 70
Tabel 5.6 Kasus Uji Validasi Ubah Transaksi 71

Tabel 5.7 Kasus Uji Validasi Hapus Transaksi.....72
Tabel 5.8 Kasus Uji Validasi Tambah Kategori Pengeluaran.....74
Tabel 5.9 Kasus Uji Validasi Edit Kategori Pengeluaran.....75
Tabel 5.10 Kasus Uji Validasi Ubah Kategori Pengeluaran.....76
Tabel 5.11 Kasus Uji Validasi Hapus Kategori Pengeluaran.....77
Tabel 5.12 Kasus Uji Validasi Tambah Reminder.....79
Tabel 5.13 Kasus Uji Validasi Ubah Status Reminder.....80
Tabel 5.14 Kasus Uji Validasi Hapus Reminder.....81
Tabel 5.15 Kasus Uji Validasi Tambah Memo.....82
Tabel 5.16 Kasus Uji Validasi Edit Memo.....84
Tabel 5.17 Kasus Uji Validasi Ubah Memo.....85
Tabel 5.18 Kasus Uji Validasi Hapus Memo.....86
Tabel 5.19 Kasus Uji dan Hasil Pengujian Kompatibilitas Android dan iOS.....96

DAFTAR KODE

Kode 2.1 Penggunaan localStorage.....6
Kode 2.2 Penggunaan CSS 7
Kode 2.3 Penggunaan Javascript Internal 7
Kode 2.4 Penggunaan Javascript Eksternal 8
Kode 2.6 Pemanggilan javascript dan CSS JQueryMobile.....9
Kode 2.7 Kode pengaturan fitur *file* pada config.xml.....10
Kode 2.8 Kode pengaturan fitur *file* pada AndroidManifest.xml 11
Kode 2.9 Kode pengaturan fitur *storage* pada config.xml.....11
Kode 2.10 Kode pengaturan fitur *notification* pada config.xml 11
Kode 5.1 Kode Implementasi Basis Data Tabel Transaksi.....47
Kode 5.2 Kode Implementasi Basis Data Tabel Setting Anggaran 47
Kode 5.3 Kode Implementasi Basis Data Tabel Memo.....47
Kode 5.4 Kode Implementasi Basis Data Tabel Reminder 48
Kode 5.5 MainActivity.java.....48
Kode 5.6 Kode Konten Halaman Awal 50
Kode 5.7 Kode Footer Halaman Awal.....50
Kode 5.8 Kode Header Halaman Utama.....51
Kode 5.9 Kode Konten Halaman Utama 51
Kode 5.10 Kode Konten Halaman Sub Menu Transaksi 52
Kode 5.11 Kode Konten Daftar Transaksi.....53
Kode 5.12 Kode Konten Kategori.....54
Kode 5.13 Kode Konten Reminder.....55

Kode 5.14 Kode Konten Memo 55
Kode 5.15 Kode Perulangan Menampilkan Daftar Transaksi 57
Kode 5.16 Kode Perulangan Menampilkan Daftar Kategori 58
Kode 5.17 Kode Menampilkan Progress Bar Kategori Pengeluaran 58
Kode 5.18 Kode Perulangan Menampilkan Daftar Memo.....59
Kode 5.19 Kode Tambah dan Edit Memo 59
Kode 5.20 Kode Hapus Memo.....60
Kode 5.21 Kode Perulangan Menampilkan Daftar Reminder 60
Kode 5.22 Kode *Welcome Screen* 61
Kode 5.23 Kode Halaman *Home* 62
Kode 5.24 Kode Halaman Sub Menu Transaksi 63
Kode 5.25 Kode Halaman Menu Daftar Transaksi 64
Kode 5.26 Kode Halaman Menu Kategori Pengeluaran 65
Kode 5.27 Kode Halaman Menu Reminder 67
Kode 5.28 Kode Halaman Menu Memo 68
Kode 5.29 Kode inisialisasi variable yang digunakan 88
Kode 5.30 Kode pengujian QUnit kasus uji pertama.....88
Kode 5.31 Kode pengujian QUnit kasus uji kedua 90
Kode 5.32 Kode pengujian QUnit kasus uji ketiga.....92
Kode 5.33 Kode pengujian QUnit kasus uji keempat 94

BAB I
PENDAHULUAN

1.1 Latar Belakang

Keuangan merupakan salah satu indikator paling umum yang digunakan untuk menilai sukses atau tidaknya kehidupan seseorang. Seseorang dikatakan sukses dan memiliki keuangan yang sehat, apabila telah mampu menghindari praktik *besar pasak daripada tiang* atau mampu menyelaraskan antara penggunaan dana dan pemasukan [WAR-10]. Untuk mewujudkannya, tentu diperlukan suatu pengelolaan yang baik. Dalam mengelola keuangan, diperlukan suatu penganggaran atau pengalokasian yang dapat membantu dalam memastikan penerimaan dan pengeluaran dalam rentan waktu tertentu [GT-12]. Pengalokasian dana umumnya dapat digolongkan menjadi tiga hal pokok, yaitu konsumsi, tabungan dan investasi. Perencanaan keuangan dengan melakukan pengalokasian terhadap dana, membutuhkan suatu kedisiplinan dalam pengaplikasiannya. Sehingga penggunaan dana dapat sesuai dengan perencanaan dan pengalokasian yang telah ditetapkan sebelumnya. Kesalahan dalam mengelola keuangan dapat berakibat pada memburuknya kesehatan keuangan seseorang [GT-12]. Kesehatan keuangan yang buruk, yang diakibatkan karena tidak terkontrolnya pemakaian dan penggunaan dana, dapat berpengaruh terhadap kondisi kehidupan seseorang dan menjadi tidak sejahtera [RR-12]. Hal ini diperkuat dengan fakta bahwa sebanyak 19,92% dari 198.170 rumah dinas TNI yang disediakan negara masih dihuni para purnawirawan, mayoritas dikarenakan mereka tidak memiliki rumah. Hal ini menimbulkan pertanyaan, bagaimana pengelolaan dan pengalokasian sumberdaya keuangan pribadi mereka selama masih aktif bekerja [WAR-10]. Selain pengalokasian, hal yang tidak kalah pentingnya dalam mengelola keuangan adalah pencatatan. Pencatatan dalam mengelola keuangan, umumnya hanya diartikan pencatatan transaksi saja. Padahal pencatatan disini bukan hanya pencatatan transaksi, melainkan pencatatan informasi-informasi keuangan yang dapat digunakan dalam proses transaksi seperti akun-akun keuangan, rekening, jadwal

pembayaran dan lain-lain juga perlu dilakukan. Beberapa orang berpikir bahwa mencatat informasi keuangan itu tidak penting karena mampu untuk mengingatnya, namun hal tersebut keliru dan tidak akan pernah bisa terjadi [NAT-00]. Umumnya manusia tidak akan bisa mengingat secara baik apa yang telah dilakukan sebelumnya dalam waktu yg lama, terutama bagi orang-orang yang memiliki kesibukan yang sangat padat, banyaknya hal yang perlu dipikirkan tentu akan menyita waktu dan tenaga. Pencatatan informasi-informasi keuangan, dapat membantu dalam proses transaksi keuangan. Akan tetapi, pencatatan yg umumnya dilakukan secara fisik seperti pada buku dan kertas seringkali tertinggal atau bahkan hilang. Hal ini tentu cukup merepotkan, karena diperlukan untuk melakukan pencatatan kembali. Dengan memanfaatkan perkembangan teknologi *smartphone (mobile)* saat ini, masalah-masalah dalam mengelola keuangan tersebut dapat diatasi dengan cara membangun sebuah aplikasi *mobile* pengelola keuangan pribadi.

Pemanfaatan teknologi *mobile* ini didukung oleh jumlah pengguna *smartphone* yang semakin meningkat terutama *smartphone* Android, dimana berdasarkan data yang diperoleh dari *website* http://gs.statcounter.com/ terhitung pada bulan April 2014, pengguna *smartphone* dengan OS Android mendominasi dengan perolehan angka 44.87% di Indonesia dan 53.22% di dunia, diikuti iOS dengan angka 32.9% [STA-14]. Pemanfaatan teknologi *mobile* dalam pengelolaan keuangan sebenarnya telah dilakukan. Dibuktikan dengan banyaknya aplikasi pengelola keuangan di *Google Play Store* untuk Android dan *App Store* untuk iOS. Akan tetapi, aplikasi yang ada hanya fokus pada pencatatan transaksi keuangan dan hanya menyediakan klasifikasi berdasarkan kategori, belum dapat menyelesaikan permasalahan pengalokasian penggunaan dana pengguna. Selain itu, aplikasi yang ada juga belum mampu menyediakan fitur yang berfungsi untuk mengingatkan dan mencatat informasi terkait proses transaksi keuangan yang dilakukan pengguna.

Berdasarkan masalah diatas, diperlukan suatu solusi berupa aplikasi *mobile* yang dapat menyelesaikan permasalahan pengelolaan keuangan. Aplikasi yang dibangun nantinya diharapkan dapat membantu pengguna dalam melakukan pengalokasian dana pribadinya ke dalam pos pemasukan, pengeluaran dan tabungan/investasi. Selain itu, aplikasi yang dibangun nantinya juga diharapkan

dapat membantu pengguna dalam melakukan pencatatan transaksi keuangan yang dilakukan pengguna serta pencatatan informasi-informasi terkait proses transaksi keuangan dalam bentuk pengingat (*reminder*) dan memo. Pada penelitian ini, aplikasi dirancang untuk digunakan pada perangkat bergerak sistem operasi Android dan iOS. Aplikasi juga dirancang dengan menggunakan *framework* PhoneGap dan JQuery *Mobile* serta dibuat dengan menggunakan model pengembangan *hybrid app* dengan alasan untuk mempermudah pengembangan karena kompatibilitasnya terhadap berbagai macam *mobile OS* seperti Android dan iOS.

1.2 Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah dapat disusun sebagai berikut :

1. Bagaimana merancang dan mengimplementasikan suatu aplikasi *mobile* yang dapat mengalokasikan penggunaan dana pribadi ke dalam pos kategori.
2. Bagaimana merancang dan mengimplementasikan suatu aplikasi *mobile* yang berfungsi sebagai pengingat dan memo yang dapat mencatat informasi terkait proses transaksi keuangan.
3. Bagaimana kompatibilitas aplikasi *hybrid* yang dibangun untuk *platform* Android dan iOS.

1.3 Batasan Masalah

Untuk menghindari kemungkinan semakin berkembangnya masalah, maka batasan masalah dalam skripsi ini antara lain adalah:

1. Pengingat (*reminder*) tidak menggunakan *push notification*, dan hanya muncul pada saat aplikasi dijalankan.
2. Pengalokasian dana dikelompokkan menjadi tiga pos, yaitu : pemasukan, pengeluaran dan tabungan/investasi.
3. Kategori tabungan/investasi tidak memperhitungkan bunga.

1.4 Tujuan

Tujuan dari penelitian skripsi ini adalah merancang suatu aplikasi *mobile* yang dapat mengalokasikan penggunaan dana pribadi ke dalam pos kategori,

berfungsi sebagai pengingat dan memo yang dapat mencatat informasi terkait proses transaksi keuangan serta kompatibel terhadap berbagai macam OS terutama Android dan iOS.

1.5 Manfaat

Manfaat yang dapat diperoleh dari penelitian skripsi ini antara lain:

1. Membantu pengguna dalam mengalokasikan penggunaan sumberdaya keuangan pribadi sehingga dapat terhindar dari pemborosan yang tidak perlu.
2. Membantu pengguna dalam aktivitas keuangannya dengan adanya catatan informasi yang dapat membantu dalam proses transaksi.

1.6 Sistematika Penulisan

Penyusunan skripsi ini menggunakan kerangka pembahasan yang tersusun sebagai berikut:

BAB I PENDAHULUAN

Bab ini merupakan dasar dari penyusunan skripsi ini yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II KAJIAN PUSTAKA DAN DASAR TEORI

Bab ini berisi kajian pustaka, referensi atau sumber-sumber yang berhubungan dengan permasalahan dalam skripsi yang meliputi : html, css, javascript, jquery dan lain-lain.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan bagaimana metodologi untuk perancangan dan pembuatan aplikasi *mobile* pengelola keuangan pribadi.

BAB IV ANALISIS DAN PERANCANGAN

Bab ini menjelaskan analisis dan perancangan aplikasi *mobile* pengelola keuangan pribadi yang dapat menjawab permasalahan yang diuraikan pada rumusan masalah.

BAB V IMPLEMENTASI DAN PENGUJIAN

Bab ini menjelaskan implementasi aplikasi *mobile* pengelola keuangan pribadi, menjawab permasalahan yang diuraikan pada rumusan masalah, serta menjelaskan hasil pengujian aplikasi berdasarkan paramater-parameter yang telah ditetapkan.

BAB VI PENUTUP

Bab ini menjelaskan kesimpulan yang dapat diambil dari pembuatan dan pengujian aplikasi *mobile* pengelola keuangan pribadi dan disertai saran yang mendukung untuk pengembangan selanjutnya.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini akan diuraikan mengenai kajian pustaka terhadap penelitian sebelumnya dan teori-teori dasar pembuatan perangkat lunak diantaranya tentang HTML, CSS, Javascript, jQuery Mobile, PhoneGap dan SQLite.

2.1 HTML5

HTML atau *Hyper Text Markup Language* adalah suatu bahasa yang digunakan untuk membangun dan mendeskripsikan sebuah halaman web. HTML5 adalah standar terbaru dari HTML. HTML versi 5 ini dapat memberikan mulai dari animasi grafis, music, video dan dapat digunakan untuk membangun sebuah aplikasi web. Salah satu dari fitur terbaru dari HTML5 yang digunakan dalam penelitian ini adalah *local storage* / *web storage* dan *local SQL Database*. Kode 2.1 memperlihatkan cara penggunaan *localStorage*.

```
<html>
<head>
<script type=”text/javascript”>
//kode javascript set dan get local storage
localStorage.setItem("lastname", "Smith");
localStorage.getItem("lastname");
//kode javascript set dan get session storage
sessionStorage.setItem("lastname", "Smith");
sessionStorage.getItem("lastname");
</script>
</head>
<body>
</body>
```

Kode 2.1 Penggunaan localStorage

Perbedaan dari *localStorage* dan *sessionStorage* adalah *localStorage* tidak memiliki batas waktu penyimpanan, sedangkan *sessionStorage* hanya tersimpan dalam satu *session* [WS-14].

2.2 CSS

CSS atau *Cascading Style Sheets* dibuat oleh *World Wide Web Consortium* (W3C) dengan tujuan mempermudah mengubah tampilan web hanya dengan menggunakan satu file saja. CSS3 adalah standar terbaru dari CSS. Beberapa fitur

dari CSS3 yang digunakan dalam penelitian ini diantaranya *web fonts* dan *text effects*. Kode 2.2 adalah cara penggunaannya dari CSS3.

```
//web fonts
@font-face{
font-family: myFirstFont;
src: url(sansation_light.woff);
}
div{
font-family:myFirstFont;
}

//text effects
h1{
text-shadow: 5px 5px 5px #FF0000;
}
```

Kode 2.2 Penggunaan CSS

Web fonts mempermudah dalam penggunaan font yang belum terpasang di dalam PC. Sedangkan *text effects* mempermudah dalam pengaturan efek dari teks yang ditampilkan [WS-14].

2.3 Javascript

Javascript adalah bahasa yang dapat ditambahkan pada halaman HTML untuk membuatnya menjadi lebih interaktif dan nyaman bagi pengguna. Walaupun memiliki nama yang mirip, javascript bukanlah bagian dari bahasa java, javascript lebih mudah untuk dipelajari dibandingkan java. Javascript dapat dituliskan dengan menggunakan tag <script>. Struktur penulisannya dapat dituliskan pada internal halaman web sesuai dengan Kode 2.3.

```
<html><head>
<script type=”text/javascript”>
//kode javascript
</script>
</head><body></body>
```

Kode 2.3 Penggunaan Javascript Internal

Atau menghubungkan halaman web dengan file eksternal javascript dengan cara sesuai Kode 2.4.

```
<html><head>
<script type=”text/javascript” src=”coba.js”>
</script></head>
<body></body>
```

Kode 2.4 Penggunaan Javascript Eksternal

Cara yang kedua, merupakan cara yang efektif dan yang akan digunakan pada penelitian ini. Hal ini dikarenakan kita dapat memodifikasi hanya pada satu file javscript eksternal, tp dapat digunakan pada semua halaman web yang memanggilnya [JS-10].

2.4 jQuery Mobile

JQuery Mobile merupakan suatu *framework* yang menggunakan prinsip dasar JQuery "*write less, do more*" dan mengembangkannya dengan menyediakan sebuah *User Interface* dalam pengembangan aplikasi perangkat *mobile* pada beberapa *platform* [RY-12].

Beberapa fitur yang ada pada jQuery Mobile antara lain [RY-12] :

1. Kompatibel terhadap berbagai macam platform mobile seperti iOS, Android, Symbian, Blackberry, WebOS, Windows Phone 7, Samsung Bada dan MeeGo.
2. Dibangun di atas library jQuery. Hal ini akan mempermudah programmer untuk memahami sintak-sintak di dalamnya karena telah familier atau hampir sama dengan jQuery.
3. Menggunakan theme tertentu sehingga memudahkan dalam mengkustomasi tampilan sesuai keinginan.
4. Support *multiple page* yang dapat mengurangi jumlah file pada aplikasi, sehingga lebih efisien dalam penggunaan file.

Struktur *Multiple page* dari jQuery *Mobile* dapat dilihat pada Kode 2.5 [RY-12]:

```
<div data-role="page" id="main">
 <div data-role="header">
   <h1> Main Page </h1>
 </div>
 <div data-role="content">
   <h1> Page Nav and Dialog Example </h1>
   <a data-role="button" href="#page2">Page Navigation</a>
 </div>
 <div data-role="footer">
   <h4> Main Page Footer </h4>
 </div>
</div>

<div data-role="page" id="page2" data-add-back-btn=”true”>
 <div data-role="header">
  <h1> Second Page </h1>
 </div>
 <div data-role="content">
  <h1> Second Page </h1>
 </div>
 <div data-role="footer">
  <h4> Click back to go back to main page </h4>
 </div>
```

Kode 2.5 Penggunaan *Multipage* JQueryMobile

Untuk menggunakan JQuery mobile, diperlukan *javascript* dan CSS yang harus di panggil pada halaman HTML. Kode 2.6 adalah kode pemanggilannya [RY-12]:

```
<link rel="stylesheet" href=" jquery.mobile-1.4.0.min.css" />
<script src="jquery-1.9.1.min.js"></script>
<script src=" jquery.mobile-1.4.0.min.js"></script>
```

Kode 2.6 Pemanggilan javascript dan CSS JQueryMobile

2.5 PhoneGap

PhoneGap adalah sebuah *framework* yang digunakan untuk mengembangkan aplikasi *mobile* menggunakan teknologi web yaitu HTML, CSS dan javascript. Aplikasi yang dikembangkan menggunakan PhoneGap adalah aplikasi *hybrid*. Aplikasi *hybrid* bukan merupakan aplikasi web base, dan bukan juga aplikasi *native.* PhoneGap menyediakan jembatan antara bahasa *javascript* ke bahasa *native* dan membuat javascript bisa mengakses dan mengontrol *device* seperti camera, GPS dan lain-lain [RY-12]. Tabel 2.1 merupakan compabilitas fitur-fitur PhoneGap terhadapa berbagai OS.

Tabel 2.1 Fitur-fitur PhoneGap

Fitur PhoneGap	amazon-fireos	android	blackberry 10	Firefox OS	ios	Ubuntu	Windows Phone 7	Windows Phone 8	Windows 8	tizen
Accelerometer	√	√	√	√	√	√	√	√	√	√
Camera	√	√	√	√	√	√	√	√	√	√
Capture	√	√	√	X	√	√	√	√	X	X
Compass	√	√	√	X	√	√	√	√	√	√
Connection	√	√	√	X	√	√	√	√	√	√
Contacts	√	√	√	√	√	√	√	√	X	X
Device	√	√	√	√	√	√	√	√	√	√
Events	√	√	√	X	√	√	√	√	√	√
File	√	√	√	X	√	√	√	√	√	X
Geolocation	√	√	√	√	√	√	√	√	√	√
Globalization	√	√	X	X	√	√	√	√	X	X
InAppBrowser	√	√	√	X	√	√	√	√	X	X
Media	√	√	√	X	√	√	√	√	√	√
Notification	√	√	√	X	√	√	√	√	√	√
Splashscreen	√	√	√	X	√	√	√	√	√	X
Storage	√	√	√	X	√	√	√	√	√	√

Keterangan :

√ - *supported feature*

× - *unsupported feature due to hardware or software restrictions*

Sumber: [PGP-14]

Fitur *File* digunakan untuk memberikan akses kepada aplikasi dalam membuat , membaca, serta menampilkan daftar direktori dan *File Systems.* Fitur ini sangat cocok untuk aplikasi yang memiliki fitur untuk mengganti isi file yang ada dalam direktori dan *File Systems* pada *Smartphone* [RY-12]. Konfigurasi untuk menggunakan fitur *File* [PGP-14]:

1. Pengaturan pada **app/res/xml/config.xml** dapat dilihat pada Kode 2.7:

```
<feature name="File">
<param name="android-package"
value="org.apache.cordova.file.FileUtils"/>
</feature>
<feature name="FileTransfer">
    <param name="android-package"
value="org.apache.cordova.filetransfer.FileTransfer" />
</feature>
```

Kode 2.7 Kode pengaturan fitur *file* pada config.xml

2. Pengaturan pada **app/AndroidManifest.xml** dapat dilihat pada Kode 2.8:

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Kode 2.8 Kode pengaturan fitur *file* pada AndroidManifest.xml

Fitur *Storage* terdiri dari *Web Storage* dan *Web SQL Database*. *Web Storage*, yang terdiri dari *Local Storage* dan *Session Storage*, digunakan untuk mengakses data dengan *key/value* tertentu. Sedangkan *Web SQL Database* memberikan fitur lengkap database dengan menggunakan SQL query, yang dalam penerapannya dalam aplikasi *mobile* akan dikonversikan menjadi SQLite oleh PhoneGap. Konfigurasi untuk menggunakan fitur *Storage* [PGP-14]:

1. Pengaturan pada **app/res/xml/config.xml** dapat dilihat pada Kode 2.9:

```
<feature name="Storage">
<param name="android-package"
value="org.apache.cordova.Storage" />
</feature>
```

Kode 2.9 Kode pengaturan fitur *storage* pada config.xml

Fitur *Notification* digunakan untuk memberi informasi kepada user tentang sesuatu yang telah terjadi [RY-12]. Fitur ini terdiri dari *Alert*, *Confirm*, *Prompt*, *Beep*, dan *Vibrate* [PGP-14]. Dan yang digunakan dalam aplikasi ini adalah *Notification Confirm*. Konfigurasi untuk menggunakan fitur *Notification* [PGP-14]:

1. Pengaturan pada **app/res/xml/config.xml** dapat dilihat pada Kode 2.10:

```
<feature name="Notification">
<param name="android-package"
value="org.apache.cordova.dialogs.Notification" />
</feature>
```

Kode 2.10 Kode pengaturan fitur *notification* pada config.xml

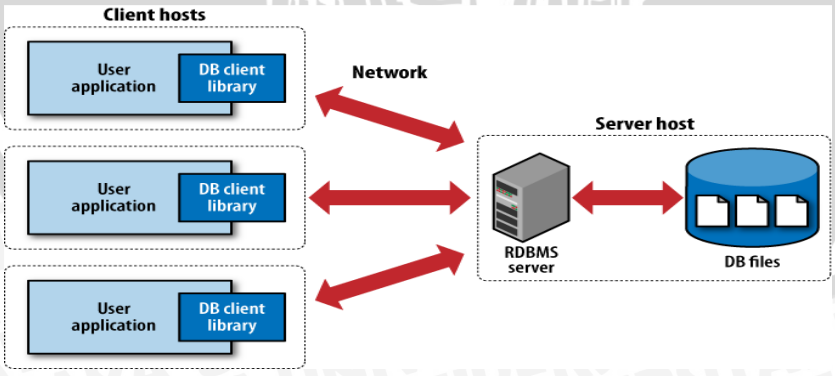
2.6 SQLite

SQLite adalah perangkat lunak yang menyediakan sistem manajemen database relational (RDBMS), yang menyimpan data yang dimasukkan pengguna dalam table yang besar, seperti RDBMS lainnya (Oracle, MySQL, PostgreSQL, dll). Kata “Lite” tidak melambangkan kemampuan dari RDBMS ini, tetapi melambangkan pada kemudahan dalam instalasi, administrasi dan penggunaan.

Berikut fitur-fitur yang disediakan SQLite :

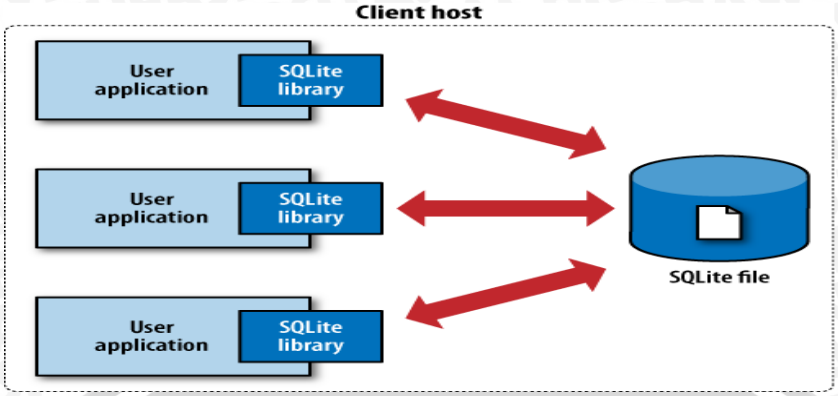
- ***Serverless***, SQLite tidak memerlukan proses pada server atau sistem, melainkan dengan mengakses file penyimpanan secara langsung.
- ***Zero Configuration***, Tidak ada server berarti tidak perlu setup (instalasi). Membuat sebuah database pada SQLite semudah membuat file biasa.
- ***Cross Platform***, Semua database berada pada satu file cross-platform tanpa perlu administrasi.
- ***Self-Contained***, Sebuah library berisi seluruh sistem database yang terintegrasi langsung dengan aplikasi.
- ***Small Runtime Footprint***, Pembuatan sebuah database pada SQLite kurang dari satu *megabyte* kode, dan membutuhkan beberapa *megabyte* memori. Dengan melakukan sedikit pengaturan, ukuran dan memori yang digunakan dapat berkurang secara signifikan.
- ***Transactional***, Transaksi pada SQLite memperbolehkan aksi penyimpanan melalui beberapa proses/thread.
- ***Full Featured***, SQLite support pada hampir sebagai besar query standar SQL92 (SQL2)
- ***Highly Reliable***, Tim pengembang SQLite melakukan pengujian dan verifikasi pada kode secara sangat serius.

Perbedaan arsitektur SQLite dengan RDBMS lainnya dapat dilihat pada Gambar 2.1 dan Gambar 2.2 :



Gambar 2.1 Tradisional RDBMS client/server arsitektur

Sumber : [KRE-10]



Gambar 2.2 Arsitektur SQLite

Sumber : [KRE-10]

2.6.1 Kelas Penyimpanan dan Jenis Data

Setiap nilai yang disimpan dalam database SQLite (atau dimanipulasi oleh mesin database) memiliki salah satu kelas penyimpanan, hal ini dapat dilihat pada Tabel 2.2 :

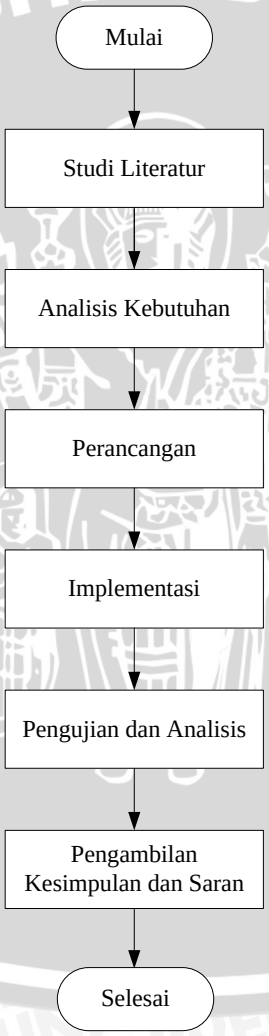
Tabel 2.2 Tipe Data SQLite

Tipe Data	Value
NULL	NULL
INTEGER	signed integer, disimpan dalam 1, 2, 3, 4, 6, atau 8 byte tergantung pada besarnya nilai.
FLOAT	Nilai adalah nilai floating point, disimpan sebagai 8-byte IEEE nomor floating point.
TEXT	Nilai adalah string teks, disimpan menggunakan database encoding (UTF-8, UTF-16BE atau UTF-16LE).
BLOB	Nilai adalah blob data, disimpan persis seperti masukan.

Sumber : [KRE-10]

BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah - langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dikembangkan. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan perangkat lunak selanjutnya. Gambar 3.1 merupakan diagram alir runtutan pengerjaan penelitian ini:



Gambar 3.1 Diagram Alir Penelitian

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

a. HTML
b. CSS
c. Javascript
d. jQuery Mobile
e. PhoneGap
f. SQLite

3.2 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan dibangun. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) sistem dan siapa saja yang terlibat di dalamnya. Analisis juga dilakukan untuk memanfaatkan informasi tentang kelebihan dan kekurangan dari sistem terkait yang sudah ada. Kemudian menentukan hal-hal apa saja yang akan diterapkan pada sistem sehingga nantinya sistem yang dibuat memiliki keunggulan dibandingkan dengan sistem yang sudah ada.

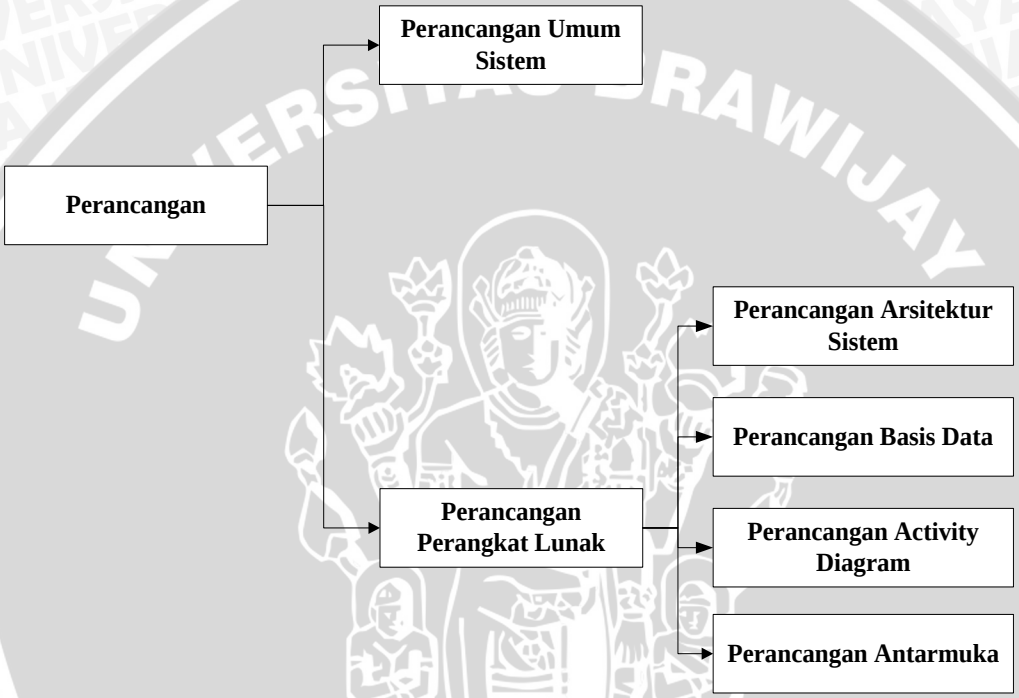
Analisis kebutuhan sistem digambarkan menggunakan bahasa pemodelan UML (*Unified Modeling Language*). *Use case diagram* digunakan untuk mendeskripsikan kebutuhan-kebutuhan dan fungsionalitas sistem dari perspektif *end-user*. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) perangkat lunak yang kemudian akan dimodelkan dalam *use case diagram*. Tahap analisis kebutuhan terdiri atas empat langkah yaitu melakukan penjabaran tentang gambaran umum aplikasi, melakukan proses identifikasi aktor yang terlibat dalam aplikasi, membuat daftar kebutuhan pengguna dengan menganalisis kebutuhan fungsional dan kebutuhan non fungsional.

3.3 Perancangan

Pada proses perancangan aplikasi terdapat 2 (dua) tahapan yaitu, tahap pertama adalah perancangan umum sistem, tahap kedua adalah proses perancangan

perangkat lunak. Tahap perancangan umum sistem menjelaskan mengenai gambar proses kerja sistem secara umum. Sedangkan pada proses perancangan perangkat lunak memiliki empat langkah, yaitu perancangan arsitektural, pemodelan *activity diagram*, perancangan basis data yang direpresentasikan menggunakan *Entity Relationship Diagram* (ERD) dan perancangan antarmuka pengguna dari aplikasi.

Gambar 3.2 akan menjelaskan tahap-tahap dari perancangan sistem aplikasi *mobile.*



Gambar 3.2 Diagram Pohon Perancangan

3.4 Implementasi

Implementasi perangkat lunak mengacu kepada perancangan perangkat lunak. Implementasi perangkat lunak dilakukan dengan menggunakan bahasa pemrograman HTML, CSS dan Javascript. Dalam tahap implementasi pembangunan aplikasi *mobile* yang *cross-platform* dilakukan dengan menggunakan *framework* PhoneGap dan model pengembangan *hybrid*. Dari segi database, implementasi akan dilakukan dengan menggunakan SQLite. Sedangkan implementasi antarmuka berdasarkan perancangan antarmuka yang telah dilakukan dengan menggunakan *framework* jQuery Mobile. Pada tahap akhir dilakukan

implementasi simulasi pada *hardware* secara langsung dari Eclipse menggunakan *Android Develepmoent Tools* (ADT).

3.5 Pengujian dan Analisis

Aplikasi yang sudah dibuat akan dilakukan pengujian, pengujian perangkat lunak dilakukan untuk mengetahui kesesuaian aplikasi *mobile* pengelola keuangan pribadi telah sesuai dengan spesifikasi kebutuhan yang melandasinya. Pengujian yang dilakukan pada aplikasi ini dengan metode pengujian validasi, pengujian unit dan pengujian kompatibilitas. Pengujian validasi menggunakan metode black-box testing, sedangkan pengujian unit menggunakan *tools* QUnit. Pengujian kompatibilitas dilakukan untuk mengetahui bagaimana kompatibilitas perangkat lunak ketika berjalan pada Android dan iOS.

Setelah tahap pengujian selesai maka dilakukan analisis untuk mengetahui hasil dari pengujian perangkat lunak. Analisis pengujian merupakan tahapan untuk menganalisis hasil dari pengujian aplikasi. Analisis pengujian ini mengacu dari metode pengujian yang dilakukan sehingga mendapatkan kesimpulan dari aplikasi *mobile* pengelola keuangan pribadi yang sudah dibuat.

3.6 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan perangkat lunak, implementasi perangkat lunak, dan pengujian perangkat lunak telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan atau kekurangan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan perangkat lunak lebih lanjut.

BAB IV
ANALISIS DAN PERANCANGAN

Bab ini membahas mengenai analisis dan perancangan aplikasi. Tahap analisis kebutuhan terdiri atas empat langkah yaitu melakukan penjabaran tentang gambaran umum aplikasi, melakukan proses identifikasi aktor yang terlibat dalam aplikasi, membuat daftar kebutuhan pengguna dengan menganalisis kebutuhan fungsional dan kebutuhan non fungsional. Sedangkan perancangan yang dilakukan meliputi 2 (dua) tahap yaitu perancangan umum sistem dan perancangan perangkat lunak.

4.1 Analisis Kebutuhan

Proses analisis kebutuhan ini diawali dengan penjabaran gambaran umum aplikasi, identifikasi aktor, penjabaran tentang kebutuhan fungsional yang akan di modelkan dalam bentuk *use case diagram* serta kebutuhan non fungsional. Analisis kebutuhan ini bertujuan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan pengguna. Pada analisis kebutuhan juga memanfaatkan informasi kelebihan dan kekurangan dari beberapa sistem terkait yang sudah ada, informasi tersebut dapat dilihat pada Tabel 4.1.

Tabel 4.1 Informasi Kelebihan dan Kekurangan Sistem Terkait

Aplikasi	Kelebihan	Kekurangan
MoneyWise (Android)	Memiliki fitur repeat pada saat tambah transaksi	Tidak memiliki alokasi/budgeting pada tiap kategori
Spending (iOS)	Memiliki fitur manajemen akun	Tidak memiliki alokasi/budgeting pada tiap kategori
BudgetWise (iOS)	Memiliki fitur statistik	Tidak memiliki alokasi/budgeting pada tiap kategori

4.1.1 Gambaran Umum Aplikasi

Pembahasan gambaran umum aplikasi *mobile* pengelola keuangan pribadi terdiri atas dua bagian, yaitu deskripsi umum aplikasi *mobile* pengelola keuangan

pribadi dan lingkungan aplikasi *mobile* pengelola keuangan pribadi. Berikut penjabarannya :

1. **Deskripsi Aplikasi**

Aplikasi *mobile* pengelola keuangan pribadi adalah aplikasi yang digunakan untuk mengelola sumberdaya keuangan secara personal. Pengguna diharapkan dapat mencatat dan melihat transaksi keuangan yang telah dilakukan. Selain itu pengguna juga diharapkan dapat mengatur pengalokasian terhadap sumberdaya keuangan yang dibagi menjadi 3 kategori, yaitu pemasukan, pengeluaran dan investasi/tabungan. Pengguna juga diharapkan dapat mencatat informasi-informasi yang terkait dengan proses transaksi keuangan dalam bentuk memo dan pengingat (*reminder*).

2. **Lingkungan Aplikasi**

Aplikasi *mobile* pengelola keuangan pribadi ini membutuhkan suatu lingkungan yang digunakan sebagai tempat berjalannya aplikasi. Secara keseluruhan aplikasi *mobile* ini berbasis *hybrid mobile application*, sehingga membutuhkan sebuah *device* untuk menjalankan aplikasi tersebut. Dengan penggunaan teknologi *mobile smartphone* Android dan iOS, aplikasi *mobile* ini diharapkan bisa dijalankan di semua lingkungan *device* Android dan iOS.

4.1.2 Identifikasi Aktor

Tahap ini adalah tahap untuk melakukan identifikasi terhadap aktor-aktor yang akan berinteraksi dengan aplikasi. Pada Tabel 4.2 memperlihatkan aktor-aktor yang terlibat beserta penjelasannya.

Tabel 4.2 Identifikasi Aktor

Aktor	Deskripsi
Pengguna	Pengguna adalah orang yang dapat menggunakan aplikasi untuk mencatat dan mengalokasikan sumberdaya keuangan, mencatat informasi keuangan seperti akun keuangan, mencatat pengingat yang digunakan dalam proses transaksi keuangan.

4.1.3 Analisis Kebutuhan Fungsional

Daftar kebutuhan terdiri dari kebutuhan fungsional dan kebutuhan non-fungsional. Pada daftar kebutuhan fungsional akan dispesifikasikan yaitu

spesifikasi kebutuhan fungsional pengguna yang di tunjukkan pada Tabel 4.3 dengan pernomer mengunakan SRS (*Software Requirement Spesification*).

Tabel 4.3 Spesifikasi Kebutuhan Fungsional

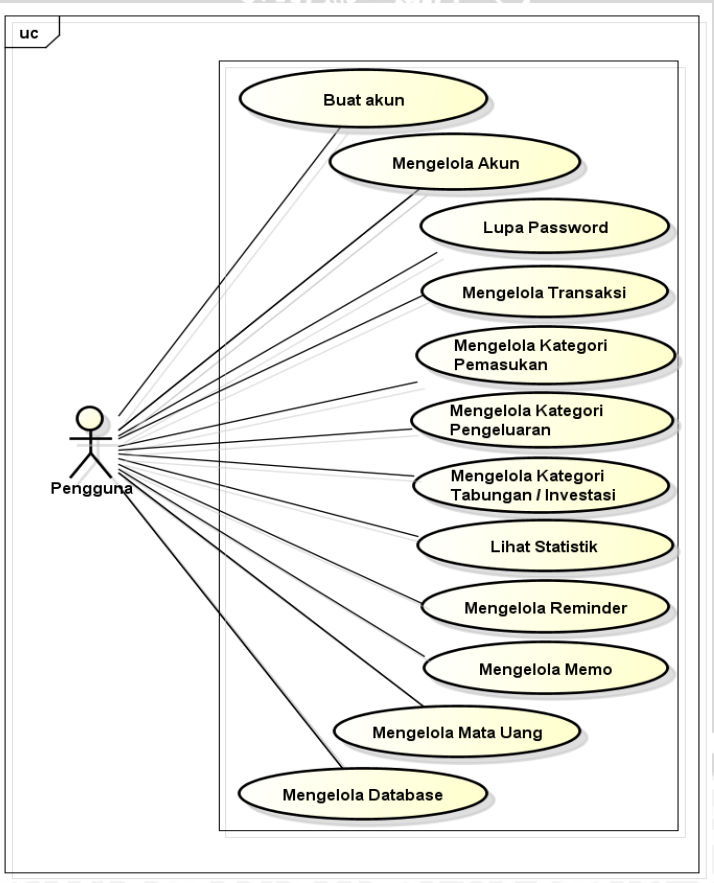
Nomor SRS	Kebutuhan	*Use case*
SRS_001_01	Aplikasi harus dapat menyediakan fasilitas buat akun untuk pengguna, sehingga pengguna dapat masuk ke dalam aplikasi dan menikmati fitur pada aplikasi ini.	Buat Akun
SRS_001_02	Aplikasi harus dapat menyediakan fasilitas lihat data akun pengguna, ubah data akun pengguna dan *password* akun, untuk mengelola data akun pengguna.	Mengelola Akun
SRS_001_03	Aplikasi harus dapat menyediakan fasilitas lupa password untuk pengguna, sehingga pengguna dalam mengingat kembali password	Lupa Password
SRS_002_01	Aplikasi harus dapat menyediakan fasilitas tambah, lihat, ubah dan hapus data transaksi, sehingga pengguna dapat mengelola (menambah, mengubah, melihat, dan menghapus) transaksi yang dilakukan pengguna.	Mengelola Transaksi
SRS_002_02	Aplikasi harus dapat menyediakan fasilitas lihat, tambah, ubah dan hapus kategori pemasukan, sehingga pengguna dapat mengelola (menambah, mengubah, melihat, dan menghapus) dan mengkategorikan pemasukan pengguna.	Mengelola Kategori Pemasukan
SRS_002_03	Aplikasi harus dapat menyediakan fasilitas lihat, tambah, ubah dan hapus kategori pengeluaran, sehingga pengguna dapat mengelola (menambah, mengubah, melihat, dan menghapus) dan mengalokasikan pengeluaran pengguna.	Mengelola Kategori Pengeluaran
SRS_002_04	Aplikasi harus dapat menyediakan fasilitas lihat, tambah, ubah dan hapus kategori investasi, sehingga pengguna dapat mengelola (menambah, mengubah, melihat, dan menghapus) dan mengalokasikan investasi pengguna.	Mengelola Kategori Investasi
SRS_002_05	Aplikasi harus dapat menyediakan fasilitas lihat statistik data transaksi berupa total transaksi sehari-hari perbulan, dan pertumbuhan kas perbulan pengguna.	Lihat Statistik
SRS_003_01	Aplikasi harus dapat menyediakan fasilitas lihat dan hapus pengingat, sehingga pengguna dapat diingatkan dalam melakukan kegiatan transaksi pengguna sesuai waktunya.	Mengelola Pengingat (*Reminder)*
SRS_004_01	Aplikasi harus dapat menyediakan fasilitas lihat, tambah, ubah dan hapus memo, sehingga pengguna dapat menyimpan informasi-informasi keuangan yang dibutuhkan ketika proses kegiatan transaksi pengguna.	Mengelola Memo

SRS_005_01	Aplikasi harus dapat menyediakan fasilitas lihat dan ubah mata uang, sehingga pengguna dapat memilih mata uang yang akan digunakan.	Mengelola Mata Uang
SRS_006_01	Aplikasi harus dapat menyediakan fasilitas *backup* dan *restore* database, sehingga pengguna dapat mengamankan data transaksi pengguna yang telah disimpan.	Mengelola Database

Selanjutnya daftar kebutuhan fungsional akan lebih dijabarkan menggunakan diagram *use case*.

1. Diagram *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, Diagram *use case* untuk aplikasi ditunjukkan dalam Gambar 4.1.



Gambar 4.1 Diagram *Use case*

Diagram *Use Case* pada Gambar 4.1 menunjukkan fungsi-fungsi yang harus dapat dilakukan pengguna dalam menggunakan aplikasi pengelola keuangan

pribadi. Dalam penelitian ini, hanya beberapa fungsi yang akan dibahas. Fungsi-fungsi tersebut antara lain pengguna harus dapat melakukan pencatatan transaksi keuangan sehingga dibutuhkan fungsi mengelola transaksi, pengguna harus dapat melakukan pengalokasian penggunaan dana ke dalam pos kategori sehingga dibutuhkan fungsi mengelola kategori (pemasukan, pengeluaran dan investasi/tabungan), pengguna harus dapat melakukan pencatatan informasi-informasi terkait proses transaksi keuangan sehingga dibutuhkan fungsi mengelola reminder dan fungsi mengelola memo.

2. Skenario *Use case*

Secara lebih mendetail, masing-masing *use case* yang terdapat pada diagram *use case*, dijabarkan dalam skenario *use case*. Namun dalam penelitian ini, hanya fungsi mengelola transaksi, mengelola kategori, mengelola *reminder* dan mengelola memo yang akan dibahas. Di dalam skenario *use case*, akan diberikan uraian nama *use case*, aktor yang berhubungan dengan *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*. Selain itu juga akan diberikan ulasan yang berkaitan dengan tanggapan dari sistem atas suatu aksi yang diberikan oleh aktor.

a. Skenario *Use Case* Mengelola Transaksi

Kebutuhan fungsional yang harus disediakan oleh sistem adalah kebutuhan untuk mengelola transaksi keuangan di dalam sistem. Kebutuhan tersebut direpresentasikan oleh *use case* mengelola transaksi. Tabel 4.4 merupakan skenario *use case* mengelola transaksi.

Tabel 4.4 Skenario *Use Case* Mengelola Transaksi

Nama	Mengelola Transaksi
Kode SRS	SRS_002_01
Tujuan	Mengelola transaksi keuangan
Deskripsi *(Brief Description)*	*Use case* ini memungkinkan pengguna untuk mengelola transaksi keuangan. Mulai dari tambah transaksi, lihat transaksi, ubah transaksi dan hapus transaksi.
Aktor	Pengguna

<table>
<tr><td>Kondisi Awal
(Pre-Conditions)</td><td>Aktor harus menjalankan dan login aplikasi terlebih dahulu sebelum use case dimulai. Aktor harus masuk pada menu transaksi.</td></tr>
<tr><td colspan="2">Flow of Events</td></tr>
<tr><td colspan="2">Alur Utama (Basic Flow)</td></tr>
<tr><td colspan="2">Use case ini dimulai ketika aktor ingin menambah, melihat, mengubah dan/atau menghapus transaksi keuangan dari sistem
1. Sistem akan meminta aktor untuk memilih fungsi yang akan dilakukan, tambah transaksi atau fungsi lain dari mengelola transaksi (daftar transaksi).
2. Jika aktor memilih fungsi tambah transaksi, maka sub flow tambah transaksi dijalankan.
3. Jika aktor memilih fungsi lain dari mengelola transaksi (menu daftar transaksi), maka lanjut ke langkah selanjutnya.
4. Sistem akan menampilkan daftar transaksi bulan ini, dan menunggu aktor untuk menentukan tahun dan bulan transaksi yang ingin di tampilkan.
5. Setelah aktor memilih tahun dan bulan, sistem akan menampilkan data transaksi yang sesuai dengan data masukan aktor
6. Sistem akan meminta aktor untuk memilih fungsi yang akan dilakukan selanjutnya. (ubah transaksi atau hapus transaksi)
7. Setelah aktor memilih pilihan informasi maka salah satu sub flow dijalankan.
a. Jika aktor memilih “Ubah Transaksi” maka sub flow ubah transaksi dijalankan.
b. Jika aktor memilih “Hapus Transaksi” maka sub flow hapus transaksi dijalankan.</td></tr>
<tr><td colspan="2">Alur Bagian (Sub Flow)</td></tr>
<tr><td>Tambah Transaksi</td><td>1. Sistem akan meminta aktor memasukkan informasi data transaksi. Informasi yang dimasukkan berupa jenis transaksi, kategori transaksi, jenis pembayaran, dan beberapa data yang lain.
2. Setelah aktor memberikan informasi yang diminta, data transaksi akan ditambahkan ke dalam sistem.</td></tr>
<tr><td>Ubah Transaksi</td><td>1. Sistem akan menampilkan informasi data transaksi sesuai dengan transaksi yang dipilih untuk diubah
2. Aktor membuat pengubahan yang diinginkan pada informasi data transaksi. Informasi sesuai dengan yang ada pada sub flow tambah transaksi.
3. Setelah aktor mengubah informasi yang diperlukan, sistem akan mengubah data transaksi.</td></tr>
<tr><td>Hapus Transaksi</td><td>1. Sistem akan meminta aktor untuk mengkonfirmasi penghapusan data transaksi yang telah dipilih untuk dihapus.
2. Aktor memverifikasi penghapusan.
3. Sistem akan menghapus data transaksi dari system.</td></tr>
<tr><td colspan="2">Alur Alternatif (Alternative Flows)</td></tr>
</table>

Hapus Transaksi Dibatalkan	Jika pada *sub flow* "Hapus Transaksi" aktor memutuskan untuk tidak menghapus data transaksi, penghapusan akan dibatalkan, dan kembali ke awal *basic flow*
Kondisi Akhir *(Post-Conditions)*	Jika *use case* berhasil, informasi data transaksi ditambahkan, diperbarui, atau dihapus dari sistem. Jika tidak, keadaan sistem tidak akan berubah.

b. Skenario *Use Case* Mengelola Kategori

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mengelola kategori dalam sistem. Kebutuhan tersebut direpresentasikan oleh *use case* mengelola kategori pemasukan, pengeluaran dan investasi. Tabel 4.5 merupakan skenario *use case* mengelola kategori pemasukan yang mewakili *use case* mengelola kategori pengeluaran dan investasi karena memiliki alur yang sama.

Tabel 4.5 Skenario *Use Case* Mengelola Kategori Pemasukan

Nama	Mengelola Kategori Pemasukan
Kode SRS	SRS_002_03
Tujuan	Mengelola kategori pemasukan
Deskripsi *(Brief Description)*	*Use case* ini memungkinkan pengguna untuk mengelola kategori pemasukan. Mulai dari tambah, ubah, dan hapus kategori.
Aktor	Pengguna
Kondisi Awal *(Pre-Conditions)*	Aktor harus menjalankan dan login aplikasi terlebih dahulu sebelum *use case* dimulai. Aktor harus masuk pada menu kategori.
Flow of Events	
Alur Utama *(Basic Flow)*	
Use case ini dimulai ketika aktor ingin melihat, mengubah dan menghapus kategori. 1. Sistem akan menampilkan kategori. 2. Sistem akan meminta aktor menentukan fungsi yang akan dijalankan selanjutnya. (tambah, ubah, atau hapus kategori) 3. Setelah aktor memilih pilihan informasi maka salah satu *sub flow* akan dijalankan. a. Jika aktor memilih "Tambah Kategori" maka *sub flow* tambah kategori dijalankan. b. Jika aktor memilih "Ubah Kategori" maka *sub flow* ubah kategori dijalankan. c. Jika aktor memilih "Hapus Kategori" maka *sub flow* hapus kategori dijalankan.	
Alur Bagian *(Sub Flow)*	
Tambah Kategori	1. Sistem akan meminta aktor untuk memasukkan informasi kategori. Mulai dari nama kategori dan total anggaran.

	2. Setelah aktor memberikan informasi yang diminta, data akan disimpan ke dalam sistem
Ubah Kategori	1. Sistem akan menampilkan informasi kategori sesuai dengan kategori yang dipilih untuk diubah 2. Aktor membuat pengubahan yang diinginkan pada informasi kategori. Informasi sesuai dengan yang ada pada *sub flow* tambah kategori. 3. Setelah aktor mengubah informasi yang diperlukan, sistem akan mengubah kategori.
Hapus Kategori	1. Sistem akan meminta aktor untuk mengkonfirmasi penghapusan kategori yang telah dipilih untuk dihapus. 2. Aktor memverifikasi penghapusan. 3. Sistem akan meminta aktor untuk mengkonfirmasi penghapusan data transaksi kategori yang telah dipilih untuk dihapus. 4. Sistem akan menghapus kategori dari system.
Alur Alternatif *(Alternative Flows)*	
Hapus Kategori Pemasukan Dibatalkan	Jika pada *sub flow* "Hapus Kategori" aktor memutuskan untuk tidak menghapus kategori, penghapusan akan dibatalkan, dan kembali ke awal *basic flow*
Hapus Data Transaksi Kategori Pemasukan Dibatalkan	Jika pada *sub flow* "Hapus Kategori" aktor memutuskan untuk tidak menghapus data transaksi kategori, status kategori akan diupdate menjadi *not available*, sehingga tidak dapat digunakan dalam transaksi lagi. Tetapi data transaksi kategori tidak akan dihapuskan.
Kondisi Akhir *(Post-Conditions)*	Jika *use case* berhasil, informasi kategori akan ditambahkan, diubah, atau dihapus dari sistem. Jika tidak, keadaan sistem tidak akan berubah.

c. Skenario *Use Case* Mengelola *Reminder* (Pengingat)

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mengelola pengingat tentang aktivitas keuangan pengguna yang harus dilakukan di dalam sistem. Kebutuhan tersebut direpresentasikan oleh *use case* mengelola *reminder*. Tabel 4.6 merupakan skenario *use case* mengelola *reminder*.

Tabel 4.6 Skenario *Use Case* Mengelola *Reminder* (Pengingat)

Nama	Mengelola *Reminder*
Kode SRS	SRS_003_01
Tujuan	Mengelola pengingat aktivitas keuangan

<table>
<tr><td>Deskripsi
(Brief Description)</td><td>Use case ini memungkinkan pengguna untuk mengelola reminder pada sistem. Termasuk menambah, melihat, ubah status dan hapus reminder dari sistem.</td></tr>
<tr><td>Aktor</td><td>Pengguna</td></tr>
<tr><td>Kondisi Awal
(Pre-Conditions)</td><td>Aktor harus menjalankan dan login aplikasi terlebih dahulu sebelum use case dimulai. Aktor harus masuk pada menu reminder.</td></tr>
<tr><td colspan="2">Flow of Events</td></tr>
<tr><td colspan="2">Alur Utama (Basic Flow)</td></tr>
<tr><td colspan="2">Use case ini dimulai ketika aktor ingin melihat, menambah, mengubah status dan menghapus reminder.
1. Sistem akan menampilkan reminder.
2. Sistem akan meminta aktor untuk menentukan fungsi yang akan dilakukan selanjutnya. (tambah, ubah status, atau hapus reminder).
3. Setelah aktor memilih pilihan informasi maka salah satu sub flow dijalankan.
a. Jika aktor memilih “Tambah Reminder” maka sub flow tambah reminder dijalankan.
b. Jika aktor memilih “Ubah Status Reminder” maka sub flow ubah status reminder dijalankan.
c. Jika aktor memilih “Hapus Reminder” maka sub flow hapus reminder dijalankan.</td></tr>
<tr><td colspan="2">Alur Bagian (Sub Flow)</td></tr>
<tr><td>Tambah Reminder</td><td>1. Sistem akan meminta aktor untuk memasukkan informasi reminder. Mulai dari keterangan reminder dan tanggal reminder.
2. Setelah aktor memberikan informasi yang diminta, data akan disimpan ke dalam sistem</td></tr>
<tr><td>Ubah Status Reminder</td><td>1. Aktor membuat pengubahan yang diinginkan pada informasi status reminder.
2. Setelah aktor mengubah informasi yang diperlukan, sistem akan mengubah status reminder.</td></tr>
<tr><td>Hapus Reminder</td><td>1. Sistem akan meminta aktor untuk mengkonfirmasi penghapusan reminder yang telah dipilih untuk dihapus.
2. Aktor memverifikasi penghapusan.
3. Sistem akan menghapus kategori investasi dari system.</td></tr>
<tr><td colspan="2">Alur Alternatif (Alternative Flows)</td></tr>
<tr><td>Hapus Reminder Dibatalkan</td><td>Jika pada sub flow “Hapus Reminder” aktor memutuskan untuk tidak menghapus reminder, penghapusan akan dibatalkan, dan kembali ke awal basic flow</td></tr>
<tr><td>Kondisi Akhir
(Post-Conditions)</td><td>Jika use case berhasil, informasi reminder akan ditambahkan, diubah status, atau dihapus dari sistem. Jika tidak, keadaan sistem akan tidak berubah.</td></tr>
</table>

d. Skenario *Use Case* Mengelola Memo

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mengelola catatan informasi penting yang terkait transaksi keuangan dalam bentuk memo di dalam sistem. Kebutuhan tersebut direpresentasikan oleh *use case* mengelola memo. Tabel 4.7 merupakan skenario *use case* mengelola memo.

Tabel 4.7 Skenario *Use Case* Mengelola Memo

Nama	Mengelola Memo
Kode SRS	SRS_004_01
Tujuan	Mengelola catatan informasi penting transaksi keuangan
Deskripsi *(Brief Description)*	*Use case* ini memungkinkan pengguna untuk mengelola memo pada sistem. Termasuk menambahkan, mengubah dan menghapus memo dari sistem.
Aktor	Pengguna
Kondisi Awal *(Pre-Conditions)*	Aktor harus menjalankan dan login aplikasi terlebih dahulu sebelum *use case* dimulai. Aktor harus masuk pada menu memo.
Flow of Events	
Alur Utama *(Basic Flow)*	
Use case ini dimulai ketika aktor ingin menambah, mengubah, dan menghapus memo. 1. Sistem akan meminta aktor untuk memasukkan *password* aktor sebagai pengaman dari memo. 2. Setelah *password* sesuai, sistem akan menampilkan daftar memo. 3. Sistem akan meminta aktor menentukan fungsi yang akan dijalankan selanjutnya. (lihat, tambah, ubah atau hapus memo) 4. Setelah aktor memilih pilihan informasi maka salah satu *sub flow* dijalankan. a. Jika aktor memilih “Lihat Memo” maka *sub flow* lihat memo dijalankan. b. Jika aktor memilih “Tambah Memo” maka *sub flow* tambah memo dijalankan. c. Jika aktor memilih “Ubah Memo” maka *sub flow* ubah memo dijalankan. d. Jika aktor memilih “Hapus Memo” maka *sub flow* hapus memo dijalankan.	
Alur Bagian *(Sub Flow)*	
Lihat Memo	1. Sistem akan meminta aktor untuk memilih memo yang ingin ditampilkan. 2. Setelah aktor memilih, sistem akan menampilkan memo.
Tambah Memo	1. Sistem akan meminta aktor untuk memasukkan informasi memo. Mulai dari judul dan keterangan memo. 2. Setelah aktor memberikan informasi yang diminta, data akan disimpan ke dalam sistem
Ubah Memo	1. Sistem akan menampilkan informasi memo sesuai dengan memo yang dipilih untuk diubah

	2. Aktor membuat pengubahan yang diinginkan pada informasi memo. Informasi sesuai dengan yang ada pada *sub flow* tambah memo. 3. Setelah aktor mengubah informasi yang diperlukan, sistem akan mengubah memo.
Hapus Memo	1. Sistem meminta aktor untuk mengkonfirmasi penghapusan memo yang telah dipilih untuk dihapus. 2. Aktor akan memverifikasi penghapusan. 3. Sistem menghapus memo dari sistem.
Alur Alternatif *(Alternative Flows)*	
Hapus Memo Dibatalkan	Jika pada *sub flow* "Hapus Memo" aktor memutuskan untuk tidak menghapus memo, penghapusan akan dibatalkan, dan kembali ke awal *basic flow*
Kondisi Akhir *(Post-Conditions)*	Jika *use case* berhasil, informasi memo akan ditampilkan, ditambahkan, diubah atau dihapus dari sistem. Jika tidak, keadaan sistem tidak akan berubah.

4.1.4 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non fungsional adalah analisis untuk mengetahui spesifikasi yang dibutuhkan oleh sistem. Pada Tabel 4.8 ada beberapa parameter dan deskripsi kebutuhan yang akan digunakan dalam pengembangan, yaitu *Compatibility*.

Tabel 4.8 Spesifikasi Kebutuhan Non-Fungsional

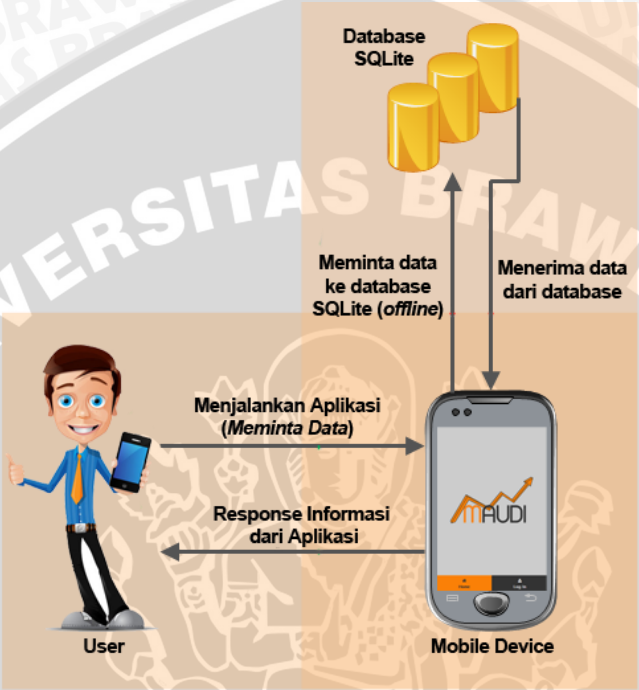
Parameter	Deskripsi Kebutuhan
Compatibility	Aplikasi harus dapat dijalankan di berbagai sistem operasi seperti Android dan iOS.

4.2 Perancangan

Proses perancangan diawali dengan perancangan umum sistem yang menjelaskan mengenai gambar proses kerja sistem secara umum. Selanjutnya adalah perancangan perangkat lunak yang berdasarkan pada hasil analisis kebutuhan yang dilakukan. Proses perancangan perangkat lunak dilakukan empat langkah, yaitu perancangan arsitektural, pemodelan *activity diagram*, perancangan basis data yang direpresentasikan menggunakan *Entity Relationship Diagram* (ERD) dan perancangan antarmuka pengguna dari aplikasi.

4.2.1 Perancangan Umum Sistem

Perancangan umum sistem merupakan tahapan awal dari perancangan perangkat lunak. Perancangan sistem dilakukan untuk merepresentasikan arsitektur sistem yang akan dibuat secara umum. Gambar 4.2 berikut menunjukan perancangan umum sistem.



Gambar 4.2 Diagram Blok Perancangan Umum Sistem

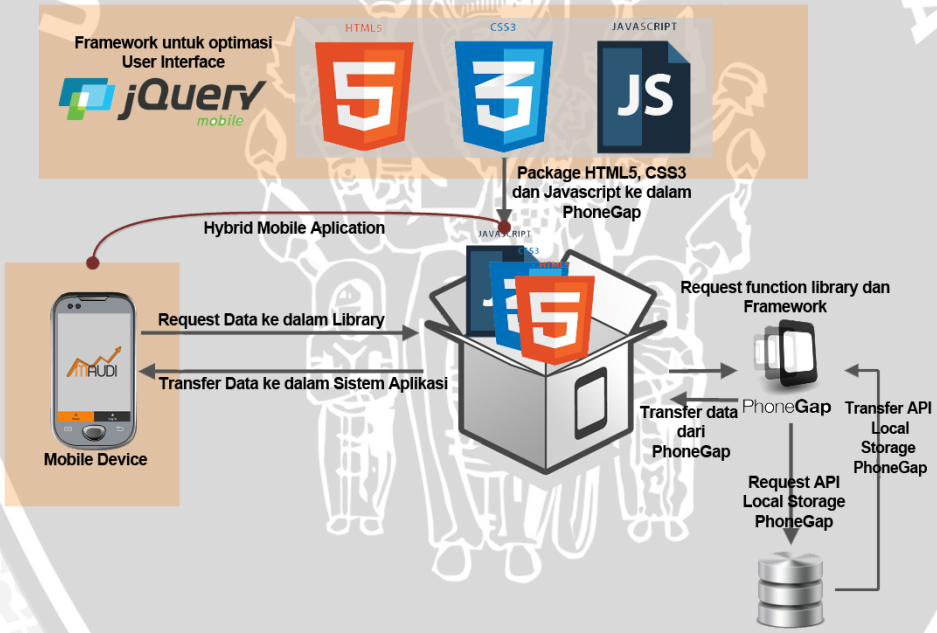
Berdasarkan Gambar 4.2, perancangan umum sistem terdiri dari 3 objek utama, yaitu pengguna *(user)*, aplikasi *(mobile device)*, dan *database* (SQLite). Pengguna akan menjalankan aplikasi dan meminta data ke database. Kemudian database akan mengirimkan data yang diminta kepada aplikasi yang selanjutkan akan ditampilkan sebagai informasi untuk pengguna.

4.2.2 Perancangan Perangkat Lunak

Perancangan aplikasi dilakukan dalam empat tahap, yaitu perancangan *activity diagram*, perancangan arsitektural, perancangan basis data, dan perancangan antarmuka pengguna dari aplikasi. Perancangan aplikasi pada skripsi ini menggunakan pendekatan desain berorientasi objek yang direpresentasikan dengan menggunakan UML (*Unified Modelling Language*).

4.2.2.1 Perancangan Arsitektur Sistem

Berdasarkan Gambar 4.3, perancangan perangkat lunak aplikasi *mobile* pengelola keuangan pribadi dibangun dengan menggunakan konsep *hybrid mobile application* pada perangkat Android dan iOS. Konsep *hybrid* sendiri dibangun menggunakan bahasa pemrograman HTML5, CSS3 serta *javascript*, dan dimaksimalkan pada *layout interface* menggunakan jQuery mobile. Pada implementasi konsep *hybrid mobile application* ini, penulis menggunakan *framework mobile* PhoneGap karena mampu menyediakan jembatan antara bahasa *javascript* ke bahasa *native* dan membuat *javascript* bisa mengakses dan mengontrol *device*. Teknik penyimpanan data menggunakan *local storage* dengan DBMS SQLite. Fitur SQLite sendiri telah disediakan oleh *framework* PhoneGap dengan API *Local Storage*.



Gambar 4.3 Perancangan Arsitektur Sistem Aplikasi

Berdasarkan Gambar 4.3, dapat dilihat aplikasi pengelola keuangan pribadi dalam proses request data dalam bentuk bahasa HTML, CSS, dan javascript akan di proses oleh library PhoneGap agar bisa mengakses database yang ada pada *device* kemudian PhoneGap mengambil data yang dibutuhkan ke database (SQLite) dengan menggunakan API Local Storage. Selanjutnya data ditransferkan kembali

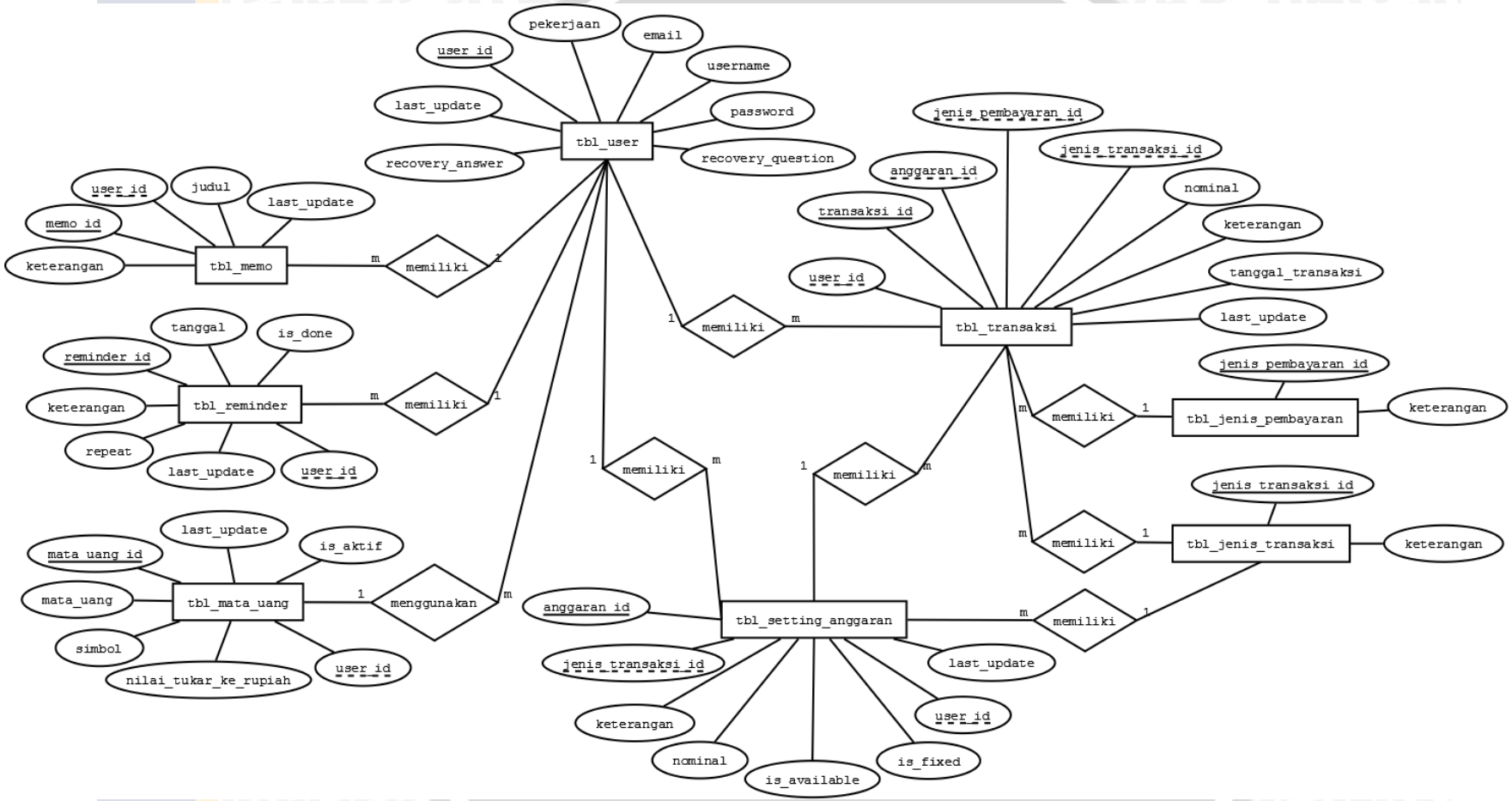
kepada PhoneGap dan ditampilkan sebagai informasi oleh aplikasi pengelola keuangan pribadi.

4.2.2.2 Perancangan Basis Data

Basis data berfungsi sebagai tempat menyimpan data. Pada skripsi ini perancangan basis data direpresentasikan dalam bentuk *Entity Relationship Diagram* (ERD). ERD menunjukkan hubungan yang terjadi diantara objek (entitas) yang terlibat dalam suatu database. ERD berisi komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan beberapa atribut yang mempresentasikan seluruh fakta yang ditinjau dari keadaan yang nyata dan aplikasi-aplikasi pengelola keuangan yang sudah ada. Aplikasi yang dimaksud adalah "*MoneyWise"* yang dapat di unduh melalui *Google Play Store*.

Pada perancangan basis data sistem ini terdapat delapan buah tabel yaitu tabel user, tabel jenis transaksi, tabel jenis pembayaran, tabel memo, tabel reminder, tabel mata uang, tabel setting anggaran dan tabel transaksi. Berdasarkan aplikasi acuan, pada tabel transaksi dibutuhkan beberapa atribut diantaranya keterangan, tanggal transaksi, kategori transaksi, jenis transaksi, dan nominal. Untuk tabel setting anggaran merupakan tabel yang berisi tentang alokasi dari kategori-kategori transaksi. Pada aplikasi acuan, kategori transaksi hanya berisi nama kategori saja, sehingga diperlukan atribut tambahan seperti nominal dan jenis transaksi dari kategori yang dibuat. Pada skripsi ini, aplikasi yang dibuat adalah multi-user, sehingga perlu ditambahkan atribut id user pada tabel tertentu.

Dalam Gambar 4.4 terdapat relasi yang menghubungkan antar tabel. Tabel tbl_user dengan 5 tabel (tbl_transaksi, tbl_setting_anggaran, tbl_memo, tbl_reminder, dan tbl_ mata_uang) mempunyai hubungan dimana satu user dapat memiliki banyak data dalam kelima tabel tersebut, sehingga relasi yang digunakan adalah *one to many*. Begitu juga dengan 3 tabel (tbl_setting_anggaran, tbl_jenis_pembayaran dan tbl_jenis_transaksi) dengan tabel tbl_transaksi yang mempunyai hubungan dimana satu jenis setting anggaran, satu jenis transaksi dan satu jenis pembayaran dapat dipakai pada banyak transaksi, sehingga relasi yang digunakan adalah *one to many*.



Gambar 4.4 *Entity Relational Diagram*

Berikut ini merupakan struktur tabel serta keterangan masing-masing tabel dan field yang ada pada database. Entitas tbl_user merepresentasikan tabel tbl_user yang berisi data-data dari user yang berperan sebagai pengguna aplikasi *mobile* seperti kode user, username, password, email, pekerjaan, recovery question, recovery answer dan last update. Struktur tabel user ditunjukkan pada Tabel 4.9.

Tabel 4.9 Struktur Tabel tbl_user

No.	Nama Field	Tipe Data	Keterangan
1	user_id (pk)	Integer	Kode dari user (1,2,3)
2	pekerjaan	Text	Pekerjaan dari user (wirausaha)
3	email	Text	Email dari user (tes@gmail.com)
4	username	Text	Username dari user (ani,akbar)
5	password	Text	Password dari user (p4ssword)
6	recovery Question	Text	Recovery Question dari user (Ani)
7	recovery Answer	Text	Recovery Answer dari user (syaiful)
8	last_update	Datetime	Waktu perubahan terakhir dari table tbl_user (2014-08-01 08:08:08)

Entitas tbl_jenis_transaksi merepresentasikan tabel tbl_jenis_transaksi yang berisi data-data dari jenis transaksi seperti kode jenis transaksi dan keterangan. Struktur tabel tbl_jenis_transaksi ditunjukkan pada Tabel 4.10.

Tabel 4.10 Struktur Tabel tbl_jenis_transaksi

No.	Nama Field	Tipe Data	Keterangan
1	jenis_transaksi_id (PK)	Integer	Kode dari jenis transaksi (1,2,3)
2	Keterangan	Text	Jenis transaksi (pemasukan, pengeluaran)

Entitas tbl_jenis_pembayaran merepresentasikan tabel tbl_jenis_pembayaran yang berisi data-data dari jenis pembayaran yang terdiri dari kode jenis pembayaran, dan keterangan. Struktur tabel tbl_jenis_pembayaran ditunjukkan pada Tabel 4.11.

Tabel 4.11 Struktur Tabel tbl_jenis_pembayaran

No.	Nama Field	Tipe Data	Keterangan
1	jenis_pembayaran_id (PK)	Integer	Kode dari jenis pembayaran (1,2,3)
2	Keterangan	Text	Jenis pembayaran (cash, kredit)

Entitas tbl_memo merepresentasikan tabel tbl_memo di dalam database. Tabel tbl_memo berisi mengenai catatan informasi penting yang berisi kode memo,

judul memo, keterangan memo, kode user, dan last_update. Struktur tabel tbl_memo ditunjukkan pada Tabel 4.12.

Tabel 4.12 Struktur Tabel Memo

No.	Nama Field	Tipe Data	Keterangan
1	memo_id (PK)	Integer	Kode dari memo (1,2,3)
2	Judul	Text	Judul memo (rekening bca,…)
3	Keterangan	Text	Keterangan / Isi dari memo (no rekening 123123123,….)
4	user_id	Integer	Kode dari user (1,2,3)
5	last_update	Datetime	Waktu perubahan terakhir dari table tbl_memo (2014-08-01 08:08:08,…)

Entitas tbl_mata_uang merepresentasikan tabel tbl_mata_uang di dalam *database*. Tabel tbl_mata_uang berisi mengenai daftar mata uang yang berisi mengenai kode mata uang, nama mata uang, simbol, nilai tukar ke rupiah, is_aktif, kode user dan last_update. Struktur tabel tbl_mata_uang ditunjukkan pada Tabel 4.13.

Tabel 4.13 Struktur Tabel tbl_mata_uang

No.	Nama Field	Tipe Data	Keterangan
1	mata_uang_id (PK)	Integer	Kode dari mata uang (1,2,3)
2	mata_uang	Text	Nama mata uang (rupiah,dollar)
3	Symbol	Text	Simbol mata uang (Rp,$)
4	nilai_tukar_ke_rupiah	Real	Nilai tukar mata uang ke rupiah (100,20000)
5	is_aktif	Integer	Status keaktifan mata uang, bernilai 1 bila status mata uang aktif (1,0)
6	user_id	Integer	Kode dari user (1,2,3)
7	last_update	Datetime	Waktu perubahan terakhir dari table tbl_mata_uang (2014-08-01 08:08:08)

Entitas tbl_reminder merepresentasikan tabel tbl_reminder di dalam database. Struktur tabel tbl_reminder ditunjukkan pada Tabel 4.14.

Tabel 4.14 Struktur Tabel tbl_reminder

No.	Nama Field	Tipe Data	Keterangan
1	reminder_id (PK)	Integer	Kode dari reminder (1,2,3)
2	Keterangan	Text	Keterangan / Isi dari reminder (Pergi ke pasar)
3	Tanggal	Date	Tanggal reminder (2014-08-01)
4	Repeat	Text	Status perulangan reminder (bulanan, mingguan)
5	is_done	Integer	Status pengerjaan reminder bernilai 1 bila status telah dikerjakan (1,0)

6	user_id	Integer	Kode dari user (1,2,3)
7	last_update	Datetime	Waktu perubahan terakhir dari table tbl_reminder (2014-08-01 08:08:08)

Entitas tbl_setting_anggaran merepresentasikan tabel tbl_setting_anggaran di dalam database. Struktur tabel tbl_setting_anggaran ditunjukkan pada Tabel 4.15.

Tabel 4.15 Struktur Tabel tbl_setting_anggaran

No.	Nama Field	Tipe Data	Keterangan
1	anggaran_id (PK)	Integer	Kode dari anggaran (1,2,3)
2	Keterangan	Text	Keterangan / nama anggaran (Perlengkapan rumah)
3	Nominal	Real	Nominal anggaran (10000, 100000)
4	jenis_transaksi_id	Integer	Kode dari jenis transaksi (1,2,3)
5	is_available	Integer	Status keaktifan dari anggaran bernilai 1 bila status anggaran available (1,0)
6	is_fixed	Integer	Status ketetapan anggaran bernilai 1 bila status anggaran tetap (1,0)
7	user_id	Integer	Kode dari user (1,2,3)
8	last_update	Datetime	Waktu perubahan terakhir dari table tbl_setting_anggaran (2014-08-01 08:08:08)

Entitas tbl_transaksi merepresentasikan tabel tbl_transaksi di dalam database. Struktur tabel tbl_transaksi ditunjukkan pada Tabel 4.16.

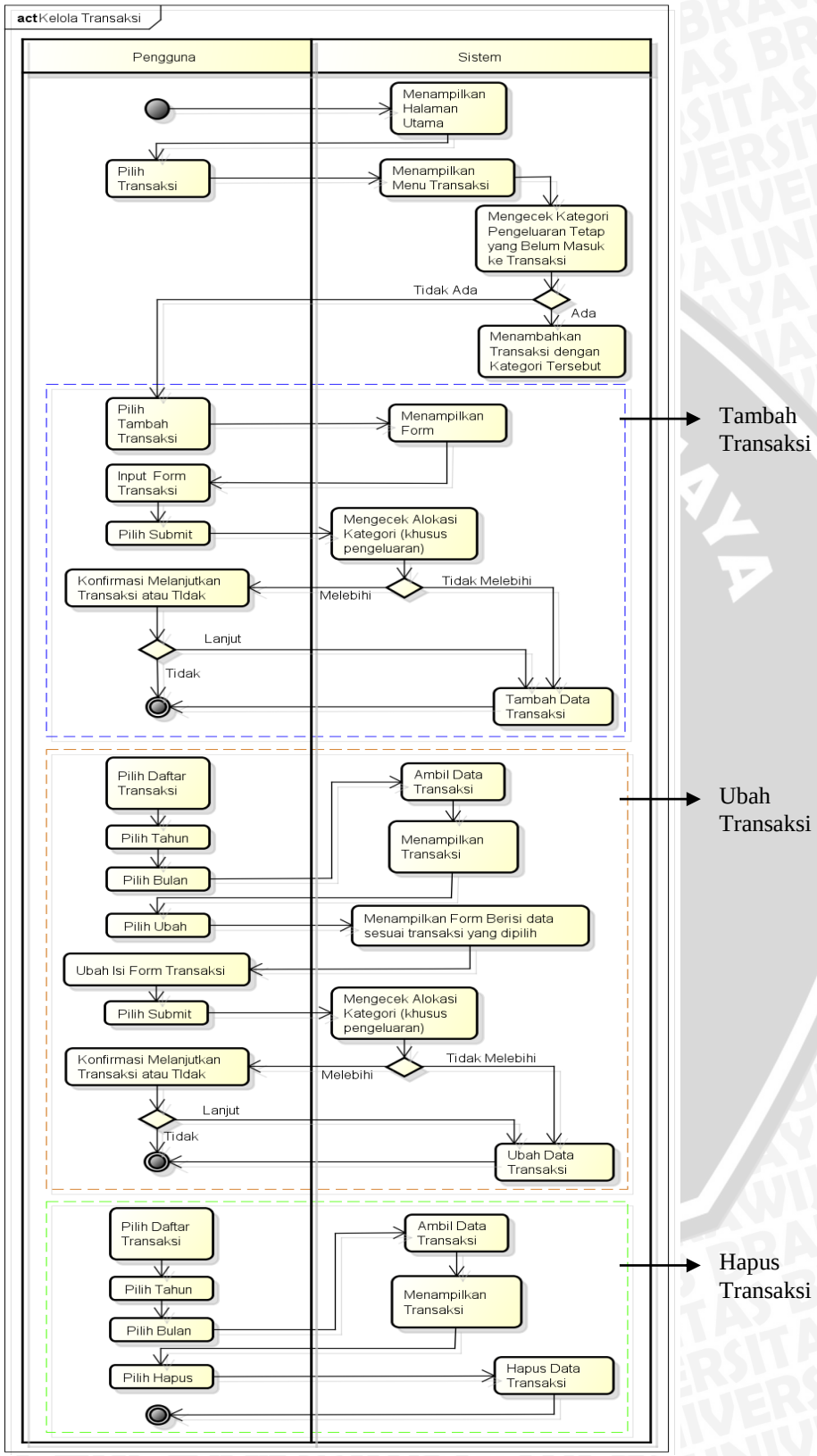
Tabel 4.16 Struktur Tabel tbl_transaksi

No.	Nama Field	Tipe Data	Keterangan
1	transaksi_id (PK)	Integer	Kode dari transaksi (1,2,3)
2	Keterangan	Text	Keterangan transaksi (Beli ikan, bayar listrik)
3	Nominal	Real	Nominal transaksi (1000, 100000)
4	tanggal_transaksi	Date	Tanggal dari transaksi (2014-08-01)
5	jenis_transaksi_id	Integer	Kode dari jenis transaksi (1,2,3)
6	anggaran_id	Integer	Kode dari anggaran (1,2,3)
7	jenis_pembayaran_id	Integer	Kode dari jenis pembayaran (1,2,3)
8	user_id	Integer	Kode dari user (1,2,3)
9	last_update	Datetime	Waktu perubahan terakhir dari table tbl_transaksi (2014-08-01 08:08:08)

4.2.2.3 Perancangan *Activity Diagram*

Diagram aktivitas *(Activity Diagram)* adalah diagram untuk memodelkan aktivitas antara pengguna dan sistem yang berjalan berdasarkan pada skenario *use case*.

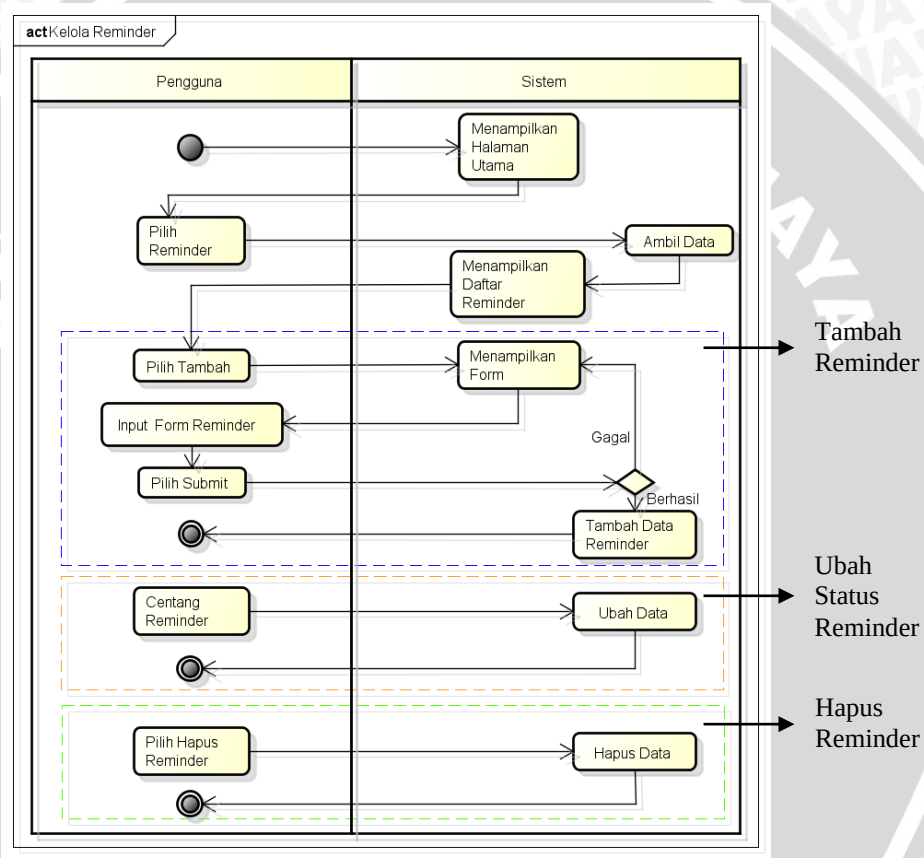
1. ***Activity Diagram*** **Mengelola Transaksi**



Gambar 4.5 *Activity Diagram* Mengelola Transaksi

Dalam Gambar 4.5 ditunjukkan aktivitas yang dilakukan oleh pengguna dan sistem. Deskripsi *activity diagram* sesuai dengan skenario pada *use case* mengelola transaksi yang ditunjukkan pada Tabel 4.4. Pengguna menjalankan aplikasi, selanjutnya sistem menampilkan halaman utama. Pengguna memilih menu transaksi dan melakukan instruksi selanjutnya. Terdapat tiga fungsi masing-masing yang digambarkan yaitu tambah transaksi, ubah transaksi, dan hapus transaksi.

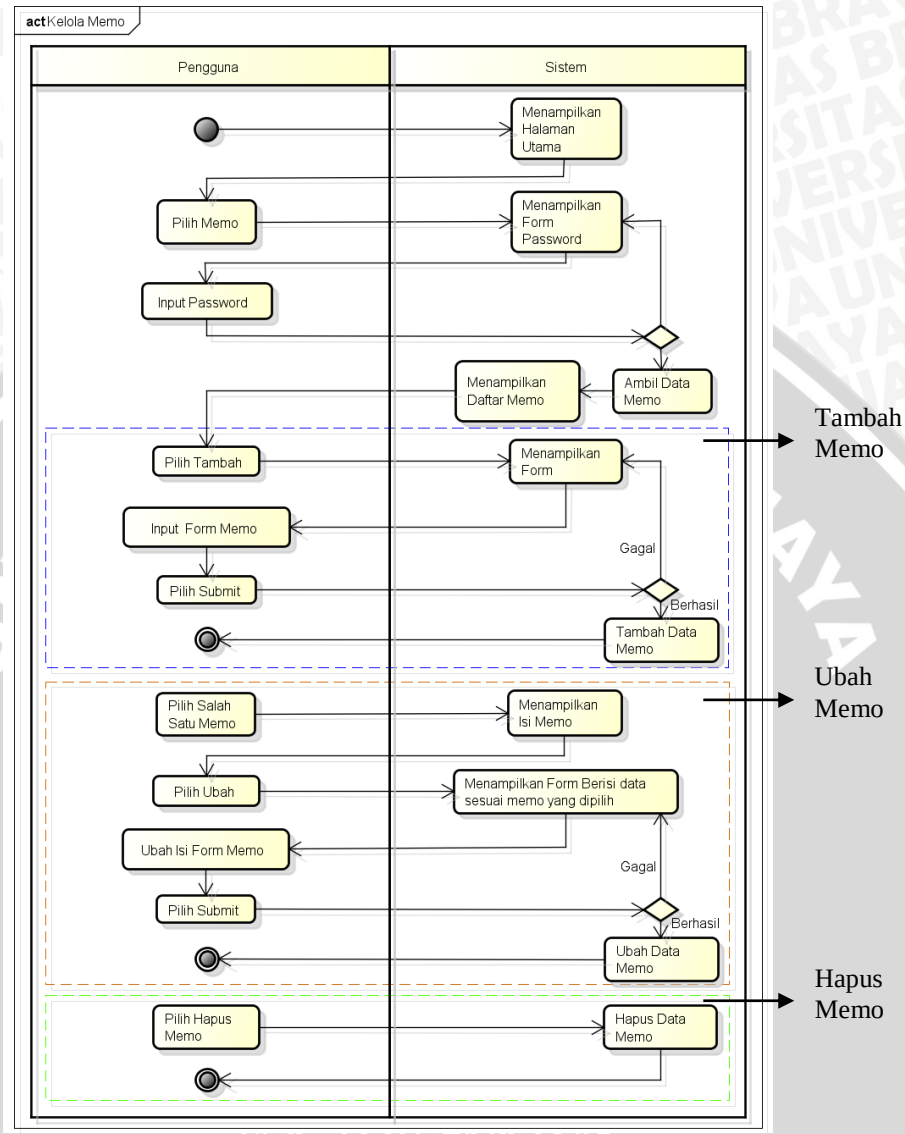
2. *Activity Diagram* Mengelola *Reminder* (Pengingat)



Gambar 4.6 *Activity Diagram* Mengelola *Reminder* (Pengingat)

Dalam Gambar 4.6 ditunjukkan aktivitas yang dilakukan oleh pengguna dan sistem. Deskripsi *activity diagram* sesuai dengan skenario pada *use case* mengelola *reminder* (pengingat) yang ditunjukkan pada Tabel 4.6. Pengguna menjalankan aplikasi, selanjutnya sistem menampilkan halaman utama. Pengguna memilih menu *reminder* dan melakukan instruksi selanjutnya. Terdapat tiga fungsi masing-masing yang digambarkan yaitu tambah reminder, ubah status reminder, dan hapus reminder.

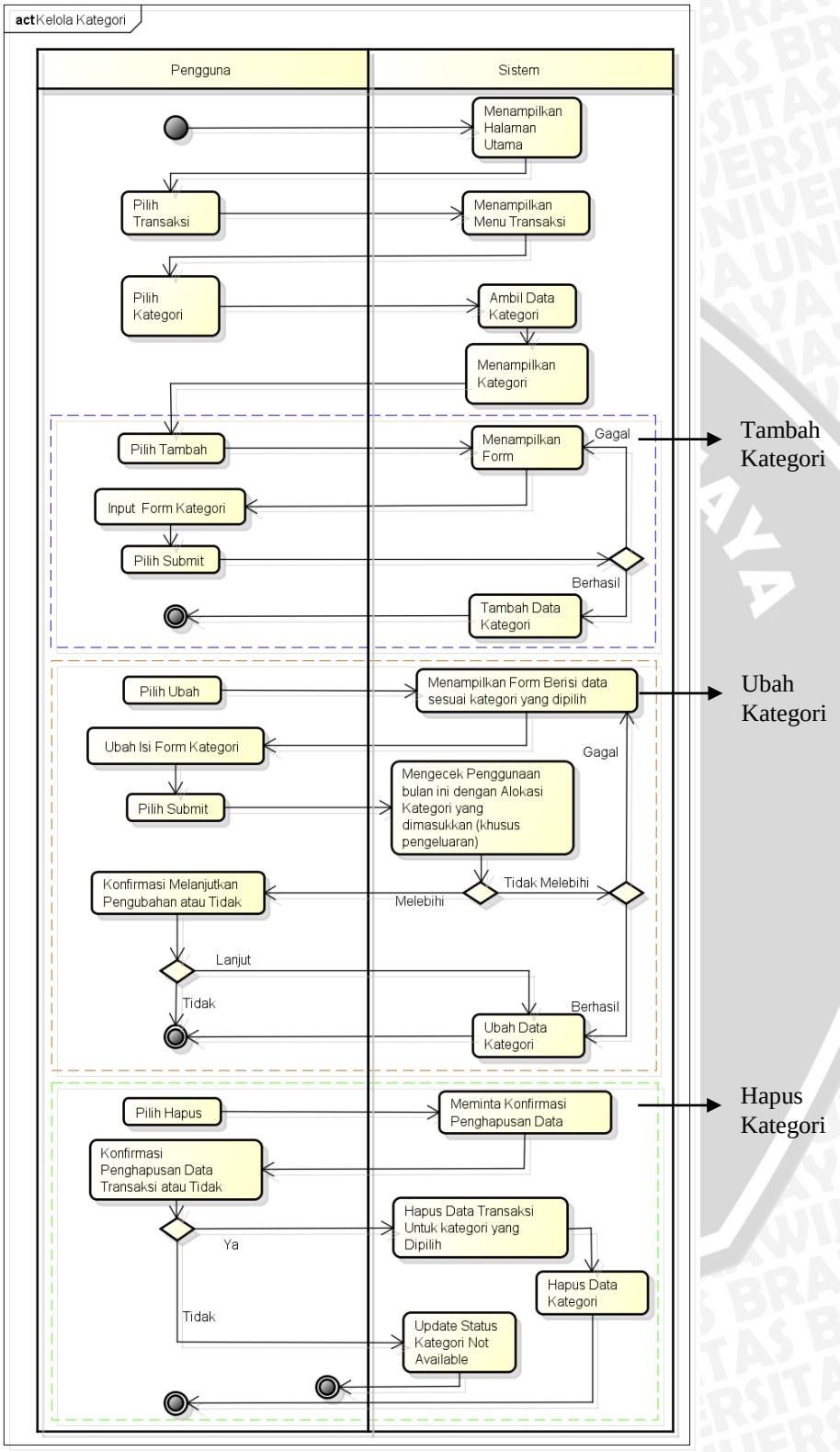
3. *Activity Diagram* Mengelola Memo



Gambar 4.7 *Activity Diagram* Mengelola Memo

Dalam Gambar 4.7 ditunjukkan aktivitas yang dilakukan oleh pengguna, antarmuka pengguna dan sistem basis data. Deskripsi *activity diagram* sesuai dengan skenario pada *use case* mengelola memo yang ditunjukkan pada Tabel 4.7. Pengguna menjalankan aplikasi, selanjutnya sistem menampilkan halaman utama. Pengguna memilih menu memo dan melakukan instruksi selanjutnya. Terdapat tiga fungsi masing-masing yang digambarkan yaitu tambah memo, ubah memo, dan hapus memo.

4. *Activity Diagram* Mengelola Kategori

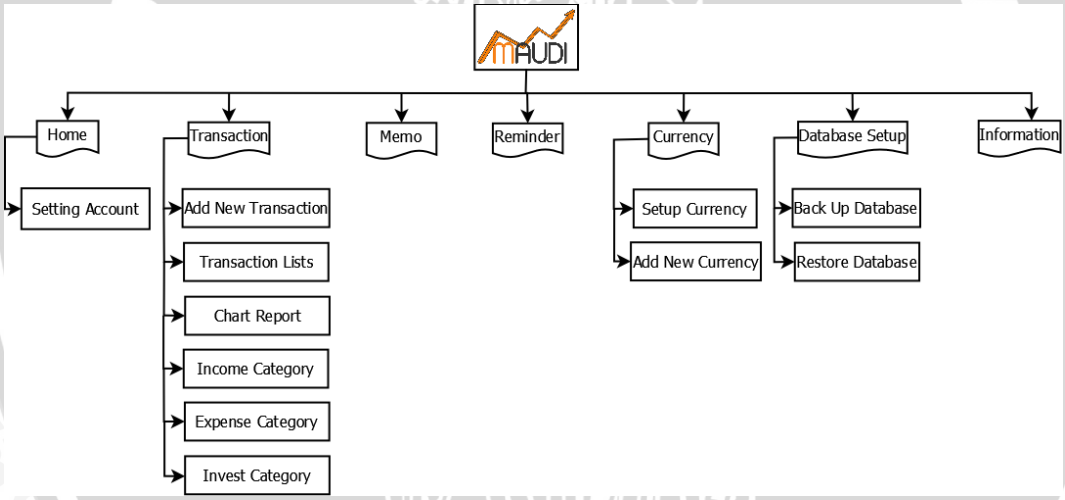


Gambar 4.8 *Activity Diagram* Mengelola Kategori

Dalam Gambar 4.8 ditunjukkan aktivitas yang dilakukan oleh pengguna dan sistem. Deskripsi *activity diagram* sesuai dengan skenario pada *use case* mengelola kategori yang ditunjukkan pada Tabel 4.5. Pengguna menjalankan aplikasi, selanjutnya sistem menampilkan halaman utama. Pengguna memilih menu transaksi dan melakukan instruksi selanjutnya. Terdapat tiga fungsi masing-masing yang digambarkan yaitu tambah kategori, ubah kategori, dan hapus kategori.

4.2.2.4 Perancangan Antarmuka

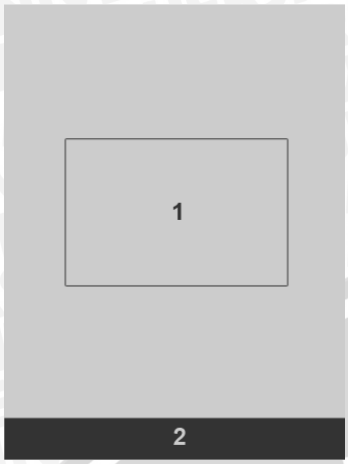
Pada bagian ini akan dijelaskan mengenai perancangan antarmuka aplikasi *mobile* pengelola keuangan pribadi. Aplikasi ini akan digunakan oleh pengguna untuk mengelola keuangan secara pribadi termasuk pengalokasian penggunaan dana serta untuk mengingatkan dan mencatat informasi penting terkait proses transaksi. Gambar struktur pohon menu aplikasi ditunjukkan pada Gambar 4.9.



Gambar 4.9 Struktur Pohon Menu Aplikasi

a. Halaman *Welcome Screen*

Halaman *welcome screen* merupakan tampilan awal ketika pengguna membuka aplikasi sebelum melakukan login. *Welcome screen* adalah halaman dengan tampilan berupa logo aplikasi dimana aplikasi dibelakang layar menjalankan proses pengecekan dan pembuatan tabel pada aplikasi. Halaman terdiri dari logo aplikasi dan menu *footer*. Gambar 4.10 adalah gambar rancangan antarmuka *welcome screen*.



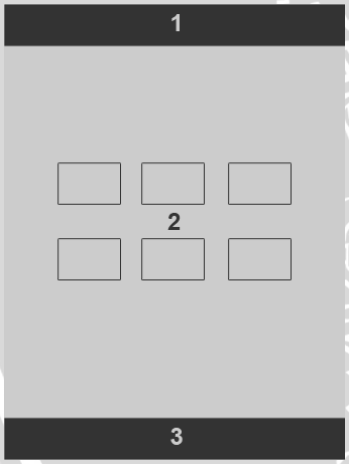
Keterangan :

1. Nama Aplikasi.
2. Footer aplikasi yang berisi menu navigasi ke menu log in dan sign up.

Gambar 4.10 Antarmuka Halaman *Welcome Screen*

b. Halaman Menu Utama *(Home)*

Halaman home adalah halaman awal setelah login. Halaman home terdiri dari menu-menu utama yang disediakan oleh aplikasi. Gambar 4.11 adalah gambar rancangan antarmuka halaman *home*.



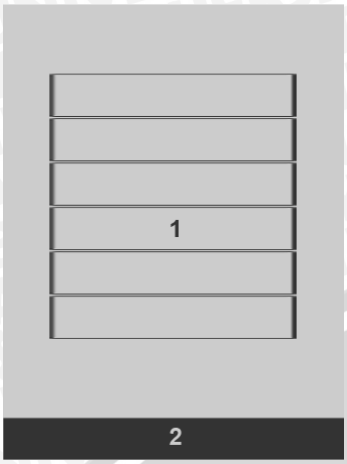
Keterangan :

1. Header aplikasi yang berisi logo aplikasi dan menu setting akun.
2. Daftar menu-menu utama, diantaranya adalah transaksi, memo, reminder, mata uang, database, dan about.
3. Footer aplikasi yang berisi menu navigasi log out.

Gambar 4.11 Antarmuka Halaman Home

c. Halaman Sub Menu

Halaman sub menu akan ditampilkan ketika pengguna memilih untuk melakukan transaksi dengan memilih menu transaksi. Gambar 4.12 adalah gambar rancangan antarmuka sub menu transaksi.



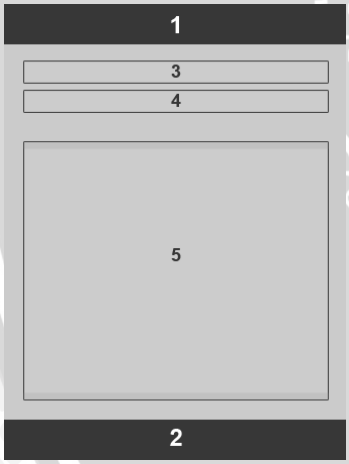
Keterangan :

1. Daftar sub menu transaksi.
2. Footer aplikasi yang berisi menu navigasi back.

Gambar 4.12 Antarmuka Halaman Sub Menu Transaksi

d. Halaman Mengelola Transaksi

Halaman form akan ditampilkan ketika pengguna memilih untuk melakukan kelola transaksi sesuai skenario mengelola transaksi SRS_002_01. Gambar 4.13 adalah gambar rancangan antarmuka mengelola transaksi.



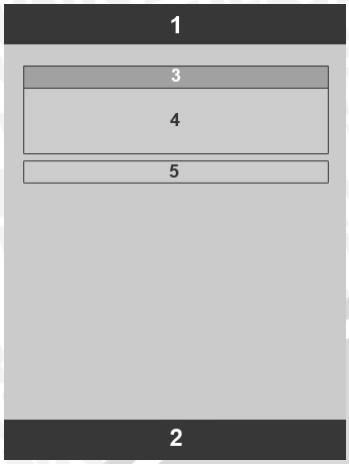
Keterangan :

1. Header aplikasi yang berisi logo aplikasi dan menu setting akun.
2. Footer aplikasi yang berisi menu navigasi back.
3. Filter tahun.
4. Filter bulan.
5. Daftar transaksi sesuai filter tahun dan bulan yang dipilih.

Gambar 4.13 Antarmuka Halaman Mengelola Transaksi

e. Halaman Mengelola Kategori

Halaman form akan ditampilkan ketika pengguna memilih untuk melakukan kelola kategori sesuai skenario mengelola kategori (pemasukan, pengeluaran, investasi) SRS_002_03. Gambar 4.14 adalah gambar rancangan antarmuka mengelola kategori.



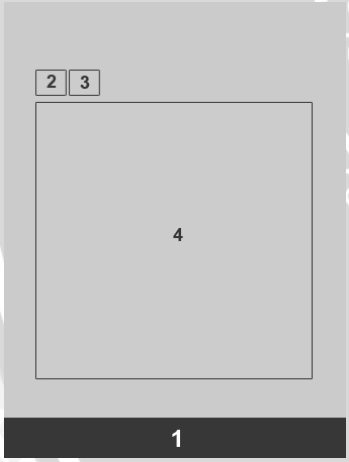
Keterangan :

1. Header aplikasi yang berisi logo aplikasi dan menu setting akun.
2. Footer aplikasi yang berisi menu navigasi back.
3. Judul kategori (kategori pemasukan, kategori pengeluaran, kategori investasi).
4. Daftar kategori dan jumlah alokasinya.
5. Button tambah kategori baru.

Gambar 4.14 Antarmuka Halaman Mengelola Kategori

f. Halaman Mengelola Reminder

Halaman form akan ditampilkan ketika pengguna memilih untuk melakukan kelola reminder sesuai skenario mengelola reminder SRS_003_01. Gambar 4.15 adalah gambar rancangan antarmuka mengelola reminder.



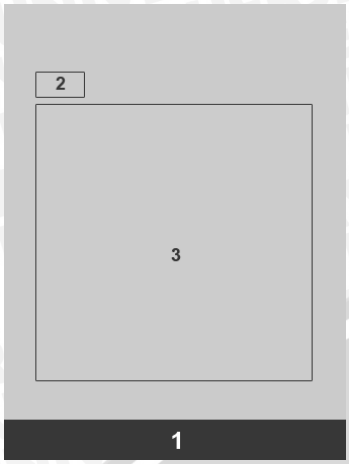
Keterangan :

1. Footer aplikasi yang berisi menu navigasi back.
2. Button Today yang berfungsi menampilkan reminder untuk hari ini.
3. Button tambah reminder baru.
4. Daftar reminder yang telah dibuat.

Gambar 4.15 Antarmuka Halaman Mengelola Reminder

g. Halaman Mengelola Memo

Halaman form akan ditampilkan ketika pengguna memilih untuk melakukan kelola reminder sesuai skenario mengelola reminder SRS_004_01. Gambar 4.16 adalah gambar rancangan antarmuka mengelola memo.



Keterangan :

1. Footer aplikasi yang berisi menu navigasi back.
2. Button tambah memo baru.
3. Daftar memo yang telah dibuat.

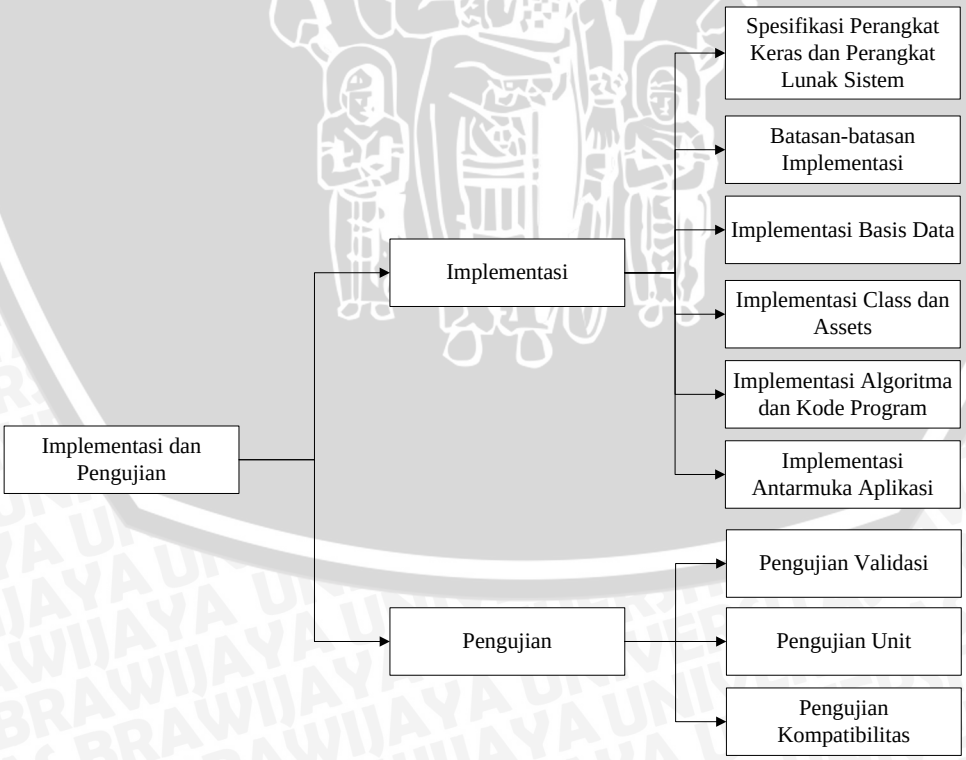
Gambar 4.16 Antarmuka Halaman Mengelola Memo

BAB V
IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas mengenai implementasi dan pengujian aplikasi. Pembahasan implementasi terdiri atas penjelasan tentang spesifikasi lingkungan implementasi, batasan-batasan dalam implementasi, implementasi basis data, implementasi *class* dan *assets* pada file program, implementasi algoritma dan implementasi antarmuka aplikasi.

5.1 Implementasi

Pada bab ini akan dibahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan perancangan perangkat lunak. Implementasi terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, implementasi tiap class pada file program, implementasi database, implementasi kode program, dan implementasi antarmuka perangkat lunak. Gambar 5.1 adalah diagram pohon implementasi dan pengujian.



Gambar 5.1 Diagram Pohon Implementasi dan Pengujian

5.1.1 Spesifikasi Perangkat Keras dan Perangkat Lunak Sistem

Dalam pengembangan aplikasi pengelola keuangan pribadi ini menggunakan sebuah PC dengan spesifikasi Sistem Operasi Windows 8 dan perangkat lunak yang digunakan untuk membuat aplikasi tersebut yaitu Eclipse Juno dengan ADT (Android Development Tools) plug in serta untuk uji coba aplikasi menggunakan Smartphone Smartfren Andromax V ZTE N986 dengan Sistem Operasi Android 4.2.1 (Jelly Bean) dan iPad Air dengan Sistem Operasi iOS 7.1.

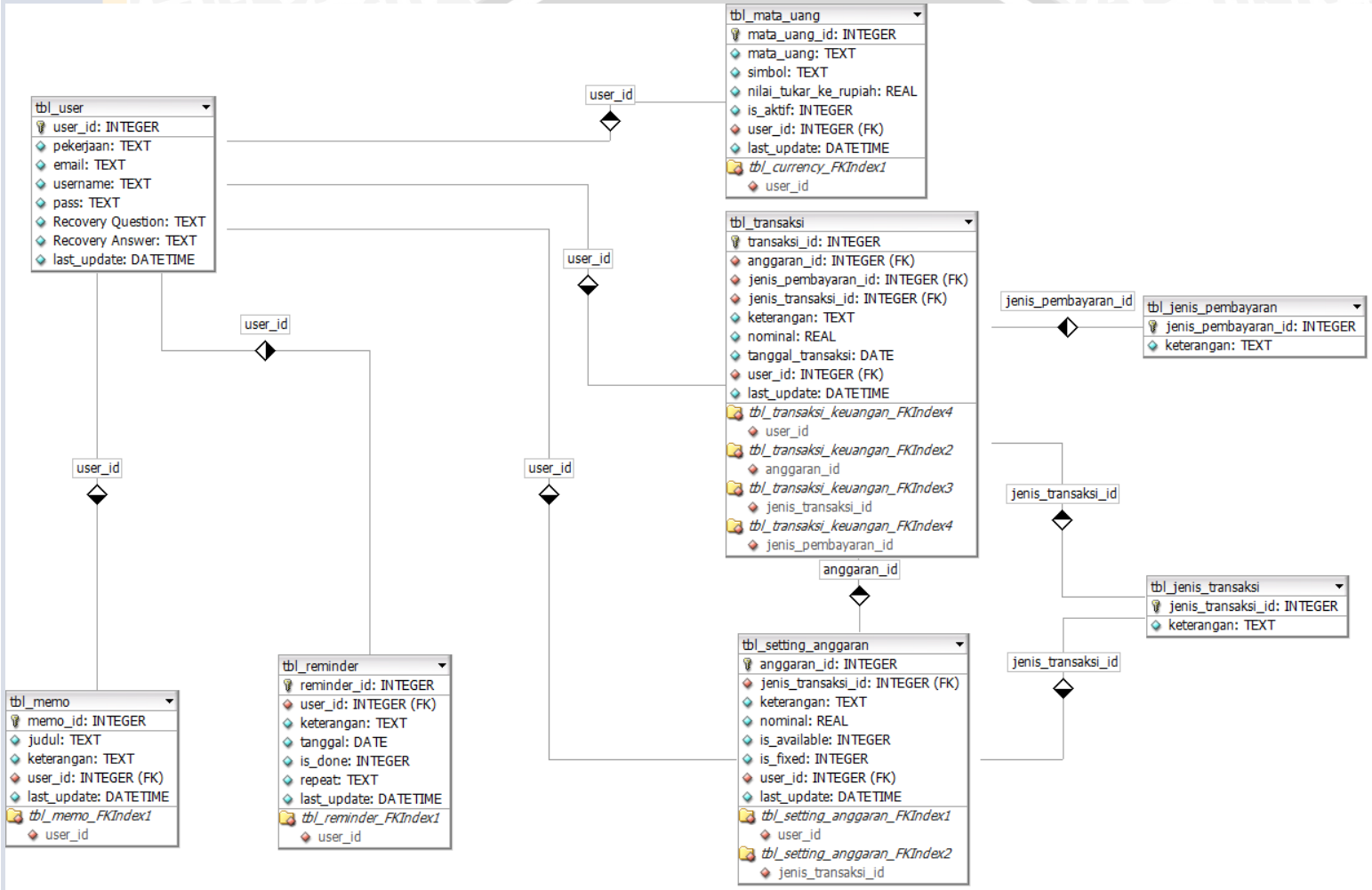
5.1.2 Batasan-batasan Implementasi

Pada implementasi perangkat lunak aplikasi *mobile* pengelola keuangan pribadi terdapat batasan-batasan dalam proses yaitu sebagai berikut :

1. Aplikasi mobile pengelola keuangan pribadi dirancang untuk dijalankan pada *smartphone* Android dan iOS menggunakan konsep hybrid.
2. Aplikasi dapat dijalankan tanpa menggunakan koneksi internet karena menggunakan konsep *offline (local storage).*
3. Aplikasi menggunakan framework jQuery *Mobile* dan PhoneGap dalam pembuatan aplikasi mobile *hybrid* dan menggunakan bahasa pemrograman HTML, CSS, dan Javascript.
4. Komunikasi data antara aplikasi dengan *local storage* diimplementasikan menggunakan pertukaran data dengan bahasa pemrograman javascript.
5. Penyimpanan data yang digunakan pada *local storage* adalah menggunakan SQLite pada *framework* PhoneGap.
6. Pengingat (*reminder*) tidak menggunakan *push notification*, dan hanya muncul pada saat aplikasi dijalankan.

5.1.3 Implementasi Basis Data

Implementasi pada penyimpanan data dilakukan dengan DBMS SQLite. Hasil implementasi SQLite pada database ini dimodelkan dalam *physical diagram*. Pada *physical diagram* terdapat hubungan relasi antar tabel. Dalam Gambar 5.2 menggambarkan *physical diagram* dari aplikasi mobile pengelola keuangan pribadi.



Gambar 5.2 *Physical Diagram*

5.1.3.1 Implementasi Kode Basis Data

Setiap tabel yang telah dirancang, akan direalisasikan dengan menggunakan bahasa yang dimengerti oleh sistem manajemen basis data dan dalam aplikasi ini menggunakan SQLite. Berikut kode implementasi basis data untuk tabel transaksi pada Kode 5.1 mengacu pada Tabel 4.16, tabel setting anggaran pada Kode 5.2 mengacu pada Tabel 4.15, tabel memo pada Kode 5.3 mengacu pada Tabel 4.12 dan tabel reminder pada Kode 5.4 mengacu pada Tabel 4.14.

a. Tabel Transaksi (tbl_transaksi)

```
CREATE TABLE IF NOT EXISTS tbl_transaksi
(
transaksi_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
jenis_transaksi_id INTEGER NOT NULL, anggaran_id INTEGER NOT
NULL,
jenis_pembayaran_id INTEGER NOT NULL,
keterangan TEXT NOT NULL,
nominal REAL NOT NULL,
tanggal_transaksi DATE NOT NULL,
user_id INTEGER NOT NULL,
last_update DATETIME NOT NULL
);
```

Kode 5.1 Kode Implementasi Basis Data Tabel Transaksi

b. Tabel Setting Anggaran (tbl_setting_anggaran)

```
CREATE TABLE IF NOT EXISTS tbl_setting_anggaran
(
anggaran_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
jenis_transaksi_id INTEGER NOT NULL,
keterangan TEXT NOT NULL,
nominal REAL NOT NULL,
is_fixed INTEGER,
is_available INTEGER NOT NULL,
user_id INTEGER NOT NULL,
last_update DATETIME NOT NULL
);
```

Kode 5.2 Kode Implementasi Basis Data Tabel Setting Anggaran

c. Tabel Memo (tbl_memo)

```
CREATE TABLE IF NOT EXISTS tbl_memo
(
memo_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
judul TEXT NOT NULL,
keterangan TEXT NOT NULL,
user_id INTEGER NOT NULL,
last_update DATETIME NOT NULL
);
```

Kode 5.3 Kode Implementasi Basis Data Tabel Memo

d. Tabel Reminder (tbl_reminder)

```
CREATE TABLE IF NOT EXISTS tbl_reminder
(
reminder_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
keterangan TEXT NOT NULL,
tanggal DATE NOT NULL,
is_done INTEGER NOT NULL,
repeat TEXT NOT NULL,
user_id INTEGER NOT NULL,
last_update DATETIME NOT NULL
);
```

Kode 5.4 Kode Implementasi Basis Data Tabel Reminder

5.1.4 Implementasi Class dan Assets Pada File Program

Setiap class dan assets yang telah dirancang pada proses perancangan direalisasikan/diimplementasikan pada sebuah file program. Hal ini bertujuan untuk membangun sebuah aplikasi yang sesuai dengan yang telah dirancang pada proses perancangan. Class dan assets yang dimaksud diantaranya adalah implementasi class default dengan menggunakan format java (.java) dan implementasi assets dengan menggunakan format HTML (.html) serta javascript (.js). Pada Tabel 5.1, Tabel 5.2, Tabel 5.3 menjelaskan mengenai hubungan antara class dan assets dengan file program yang digunakan pada implementasi.

Tabel 5.1 Implementasi class pada kode program *.java

Folder	Package	Nama	Nama File Program
Src	com.maudi.application	Main Activity	MainActivity.java

Berikut implementasi dari class MainActivity.java

```
package com.maudi.application;
import org.apache.cordova.DroidGap;
import android.os.Bundle;
public class MainActivity extends DroidGap {
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      super.setIntegerProperty("loadUrlTimeoutValue",20000);
      super.loadUrl("file:///android_asset/www/view/index.html");
    }
}
```

Kode 5.5 MainActivity.java

MainActivity.java digunakan untuk memberikan hak akses pada PhoneGap dalam mengakses fitur *native* pada perangkat Android yaitu dengan mengimport org.apache.cordova.DroidGap dan mengeset class activity ke dalam DroidGap. Kemudian di dalam class tersebut terdapat baris kode super.

loadUrl("file:///android_asset/www/view/index.html"); yang digunakan aplikasi untuk bisa mengakses file HTML dengan baik.

Tabel 5.2 Implementasi assets pada kode program *.html

Folder	Nama	Nama File Program	Keterangan
Folder Utama : assets/www/view			
/	Index	index.html	Halaman awal, berisi fungsi login, register, dan forgot password
/	Home	home.html	Halaman utama setelah login, berisi fungsi setting account dan menu lainnya
/	Memo	memo.html	Impementasi dari memo (pencatatan informasi penting keuangan)
/	Reminder	reminder.html	Impementasi dari reminder (pengingat)
/	Database Setup	export.html	Impementasi dari setting database
/	Currency	setting.html	Impementasi dari setting currency (mata uang)
/keuangan	Transaction	keuangan.html	Impementasi dari sub menu transaksi
/keuangan	Add New Transaction	add.html	Impementasi dari tambah dan edit transaksi
/ keuangan	Transaction List	transaksi.html	Impementasi dari daftar transaksi
/ keuangan	Statistik	statistik.html	Impementasi dari laporan daftar transaksi
/keuangan	Pemasukan	pemasukan.html	Impementasi dari category transaksi (pemasukan, pengeluaran, invest)
/keuangan	Pengeluaran	pengeluaran.html	
/keuangan	Invest	invest.html	

Kode html tersebut digunakan untuk membuat tampilan antarmuka dari aplikasi pengelola keuangan pribadi. Implementasi kode .html mengacu pada perancangan antarmuka. Berikut implementasi dari kode program .html

a. Index.html

File index.html digunakan untuk membuat antarmuka dari halaman awal sebelum login aplikasi pengelola keuangan pribadi. Berikut keterangan dari kode yang digunakan:

1. Kode untuk konten yang berisi nama aplikasi

```
<div class="form-login login-container" id="page-welcome">
  <div class="logo-welcome">
    <img src="../img/LogoBig.png" />
  </div>
</div>
```

Kode 5.6 Kode Konten Halaman Awal

Kode 5.6 merupakan kode untuk menampilkan nama aplikasi berupa gambar pada tampilan halaman awal. Untuk pengaturan style telah ditambahkan pada CSS sehingga untuk menggunakannya cukup memanggil class atau id style saja.

2. Kode untuk footer yang berisi menu navigasi

```
<div data-role="footer" data-position="fixed" id="footer-cust"
class="footer" data-tap-toggle="false">
      <div data-role="navbar">
        <ul>
          <li><a class="ui-btn-active ui-state-persist
nav_active"><i class="fa fa-home"></i><br><span class="font-
tab-bottom">Home</span></a></li>
          <li><a href="#login-container" data-ajax="false"
class="bg_tab_footer"><i class="fa fa-user"></i><br><span
class="font-tab-bottom">Log In</span></a></li>
       </ul>
      </div>
</div>
```

Kode 5.7 Kode Footer Halaman Awal

Kode 5.7 merupakan kode untuk menampilkan footer halaman yang berisi menu navigasi. Setiap halaman menggunakan kode ini sebagai navigasi, yang membedakan hanya link/tujuan halaman saja.

b. Home.html

File Home.html digunakan untuk membuat antarmuka dari halaman utama setelah login aplikasi pengelola keuangan pribadi. Berikut keterangan dari kode yang digunakan:

1. Kode untuk header yang berisi logo aplikasi dan menu setting akun

```
<div data-role="header" class="ui-grid-a header" id="header-
cust" data-position="fixed" style="overflow:hidden;" data-tap-
toggle="false">
    <div class="ui-block-a" style="float:right;">
      <div class="option-top">
        <a href="#option-page" id="option-btn" data-
            ajax="false" data-rel="external"><i class="fa fa-
            gear"></i></a>
      </div>
    </div>
    <div class="ui-block-a logo-header-home">
      <img src="../img/LogoSmall.png" />
    </div>
</div>
```

Kode 5.8 Kode Header Halaman Utama

Kode 5.8 merupakan kode untuk menampilkan header aplikasi yang berisi logo aplikasi dan menu setting akun. Untuk pengaturan style telah ditambahkan pada CSS sehingga untuk menggunakannya cukup memanggil class atau id style saja.

2. Kode untuk konten yang berisi daftar menu utama aplikasi

```
<div class="ui-grid-b menu" id="menu-home">
  //MENU TRANSACTION
  <div class="ui-block-a ui-block-menu">
    <a href="keuangan/keuangan.html" data-ajax="false" data-
      rel="external">
      <span class="icon-menu">
       <i class="fa fa-book"></i><br>
       <span class="font-menu">Transaction</span>
      </span>
    </a>
  </div>

  //MENU REMINDER
  //MENU MEMO
  //MENU CURRENCY
  //MENU DATABASE
  //MENU INFORMATION
</div>
```

Kode 5.9 Kode Konten Halaman Utama

Kode 5.9 merupakan kode untuk menampilkan konten halaman utama yang berisi menu utama aplikasi. Kode yang digunakan untuk tiap menu pada halaman utama hampir sama, yang membedakan hanya link/tujuan, ikon dan nama menu saja. Untuk pengaturan style telah ditambahkan pada CSS sehingga untuk menggunakannya cukup memanggil class atau id style saja.

3. Kode untuk footer yang berisi menu navigasi

Kode untuk menampilkan footer pada halaman utama sama seperti menu kode footer halaman awal yang membedakan hanya link/tujuan halaman yang menuju untuk fungsi logout.

c. Keuangan.html

File Keuangan.html digunakan untuk membuat antarmuka dari halaman sub menu transaksi setelah memilih menu transaksi pada aplikasi pengelola keuangan pribadi. Berikut keterangan dari kode yang digunakan:

1. Kode untuk konten yang berisi daftar sub menu transaksi

```
<div class="keu-container" id="page-keuangan">
   <ul data-role="listview" data-filter="false">
     //MENU ADD NEW TRANSACTION
     <li data-icon="carat-r">
      <a href="add.html" onclick="link_transaksi('')" data-
        transition="slide" data-ajax="false" data-
        rel="external">
       <i class="fa fa-plus"></i>
       <h2>Add New Transaction</h2>
      </a>
     </li>

     //MENU TRANSACTION LIST
     //MENU CHART REPORT
     //MENU INCOME CATEGORY
     //MENU EXPENSE CATEGORY
     //MENU INVEST CATEGORY
   </ul>
</div>
```

Kode 5.10 Kode Konten Halaman Sub Menu Transaksi

Kode 5.10 merupakan kode untuk menampilkan konten halaman sub menu transaksi yang berisi sub menu dari menu transaksi. Kode yang digunakan untuk tiap menu pada halaman sub menu transaksi hampir sama, yang membedakan hanya link/tujuan, ikon dan nama menu saja. Untuk pengaturan style telah ditambahkan pada CSS sehingga untuk menggunakannya cukup memanggil class atau id style saja.

2. Kode untuk footer yang berisi menu navigasi

Kode untuk menampilkan footer pada halaman sub menu transaksi sama seperti menu kode footer halaman utama yang membedakan hanya link/tujuan halaman yang menuju kembali ke halaman utama.

d. Transaksi.html

File Transaksi.html digunakan untuk membuat antarmuka dari halaman daftar transaksi setelah memilih daftar transaksi pada aplikasi pengelola keuangan pribadi. Berikut keterangan dari kode yang digunakan

1. Kode untuk header yang berisi logo aplikasi dan menu setting akun

Kode untuk menampilkan header pada halaman daftar transaksi sama seperti menu kode header halaman utama yang membedakan hanya logo aplikasi pada header terdapat link menuju kembali ke halaman utama.

2. Kode untuk footer yang berisi menu navigasi

Kode untuk menampilkan footer pada halaman daftar transaksi sama seperti menu kode footer halaman utama yang membedakan hanya link/tujuan halaman yang menuju kembali ke halaman sub menu transaksi.

3. Kode untuk konten yang berisi filter tahun, bulan dan daftar transaksi

```
<span id="list-transaksi">
   <form>
    <div class="ui-field-contain transaksi-filter">
     <select name="tahun" id="select_tahun" data-mini="true">
      <option value="0">Select year</option>
     </select>
     <select name="bulan" id="select_bulan" data-mini="true">
      <option value="0">Select month</option>
     </select>
    </div>
   </form>
   <div class="transaksi-container" id="daftar-transaksi">
     //========daftar transaksi========//
   </div>
   <div class="kosong" id="kosong">
   </div>
</span>
```

Kode 5.11 Kode Konten Daftar Transaksi

Kode 5.11 merupakan kode untuk menampilkan konten halaman daftar transaksi yang berisi filter tahun, bulan dan daftar transaksi yang ditampilkan melalui javascript. Untuk pengaturan style telah ditambahkan pada CSS sehingga untuk menggunakannya cukup memanggil class atau id style saja.

e. Pengeluaran.html

File Pengeluaran.html digunakan untuk membuat antarmuka dari halaman kategori (pengeluaran) setelah memilih kategori pengeluaran pada aplikasi pengelola keuangan pribadi. Berikut keterangan dari kode yang digunakan

1. Kode untuk header yang berisi logo aplikasi dan menu setting akun

Kode untuk menampilkan header pada halaman kategori sama seperti menu kode header halaman utama yang membedakan hanya logo aplikasi pada header terdapat link menuju kembali ke halaman utama.

2. Kode untuk footer yang berisi menu navigasi

Kode untuk menampilkan footer pada halaman kategori sama seperti menu kode footer halaman utama yang membedakan hanya link/tujuan halaman yang menuju kembali ke halaman sub menu transaksi.

3. Kode untuk konten yang berisi judul, daftar kategori, dan tombol tambah kategori

```
<div class="transaksi-container">
  <ul    data-role="listview"    id="list-pengeluaran"    data-
      inset="true" data-divider-theme="a">
       //=======judul kategori=======//
       //=======daftar kategori=======//
  </ul>
<button id="tambah_pengeluaran" class="button">Add new</button>
<br><br>
```

Kode 5.12 Kode Konten Kategori

Kode 5.12 merupakan kode untuk menampilkan konten halaman kategori yang berisi judul, daftar kategori, dan tombol tambah kategori yang ditampilkan melalui javascript. Untuk pengaturan style telah ditambahkan pada CSS sehingga untuk menggunakannya cukup memanggil class atau id style saja.

f. Reminder.html

File Reminder.html digunakan untuk membuat antarmuka dari halaman reminder setelah memilih menu reminder pada aplikasi pengelola keuangan pribadi. Berikut keterangan dari kode yang digunakan

1. Kode untuk footer yang berisi menu navigasi

Kode untuk menampilkan footer pada halaman reminder sama seperti menu kode footer halaman utama yang membedakan hanya link/tujuan halaman yang menuju kembali ke halaman utama.

2. Kode untuk konten yang berisi tombol today, tambah reminder, dan daftar reminder

Kode untuk menampilkan konten halaman reminder yang berisi tombol today, tambah dan daftar reminder. Untuk pengaturan style telah ditambahkan pada

CSS sehingga untuk menggunakannya cukup memanggil class atau id style saja. Kode ini ditunjukkan pada Kode 5.13.

```
<div class="reminder-page-container">
  <div class="btn-reminder-container">
    <div class="btn-reminder-today"><a href="#" id="reminder-
      today" class="ui-btn">Today</a></div>
    <div class="btn-reminder-add-new"><a href="#reminder-new"
      class="ui-btn" data-ajax="false" data-rel="external"><i
      class="fa fa-plus"></i></a></div>
  </div><br>

  <div class="reminder-list">
    <ul data-role="listview" data-icon="false" id="list-
      reminder" data-split-theme="a" data-filter="true" data-
      filter-placeholder="Search reminder ..." data-
      inset="true">
       //==========daftar reminder==========//
    </ul>
    <div id="div-kosong"></div>
  </div>
</div>
```

Kode 5.13 Kode Konten Reminder

g. Memo.html

File Memo.html digunakan untuk membuat antarmuka dari halaman memo setelah memilih menu memo dan melakukan verifikasi password pada aplikasi pengelola keuangan pribadi. Berikut keterangan dari kode yang digunakan

1. Kode untuk footer yang berisi menu navigasi

Kode untuk menampilkan footer pada halaman memo sama seperti menu kode footer halaman utama yang membedakan hanya link/tujuan halaman yang menuju kembali ke halaman utama.

2. Kode untuk konten yang berisi tambah memo, dan daftar memo

```
<div class="memo-page-container">
 <div class="btn-memo-container">
  <div class="btn-memo-add-new"><a href="#memo-new" class="ui-
    btn" data-ajax="false" data-rel="external"><i class="fa fa-
    plus"></i></a></div>
 </div><br>
 <div class="memo-list">
   <ul data-role="listview" data-split-icon="delete" id="memo-
    list" data-split-theme="b" data-filter="true" data-filter-
    placeholder="Search memo ..." data-inset="true">
     //=====daftar memo======//
   </ul>
   <div id="div-kosong"></div>
  </div>
</div>
```

Kode 5.14 Kode Konten Memo

Kode 5.14 merupakan kode untuk menampilkan konten halaman memo yang berisi tombol tambah dan daftar memo. Untuk pengaturan style telah ditambahkan pada CSS sehingga untuk menggunakannya cukup memanggil class atau id style saja.

Tabel 5.3 Implementasi assets pada kode program *.js

Folder	Nama	Nama File Program
Folder Utama : assets/www/js/proses		
/	Add New Transaction Transaction Lists Income Category Expense Category Invest Category	transaksi.js
/	Memo	memo.js
/	Reminder	reminder.js
/	Database Setup	back-up-restore-database.js
/	Setting Currency	currency.js
/	Chart Report	statistik.js
/	Login Register Forgot Password	login.js

Tabel 5.3 merupakan Javascript yang berisi proses-proses pada aplikasi pengelola keuangan pribadi. Kemudian hasil dari proses-proses tersebut ditampilkan pada kode .html. File transaksi.js mempunyai fungsi utama untuk melakukan proses dari dari menu transaksi termasuk mengelola transaksi dan mengelola kategori (pemasukan, pengeluaran dan invest/tabungan), memo.js mempunyai fungsi utama untuk mengelola memo, reminder.js mempunyai fungsi utama untuk mengelola reminder, back-up-restore-database.js mempunyai fungsi utama untuk back-up dan restore database, currency.js berfungsi untuk mengelola mata uang, statistic.js mempunyai fungsi utama untuk menampilkan statistik keuangan berdasarkan data transaksi, dan yang terakhir login.js mempunyai fungsi utama untuk mengelola akun.

5.1.5 Implementasi Kode Program

Aplikasi *mobile* ini mempunyai beberapa proses (*function*) utama yang terbagi dalam beberapa file javascript. Pada penulisan skripsi ini hanya dicantumkan algoritma dari beberapa proses saja sehingga tidak tidak semua

algoritma *function* akan dicantumkan. Algoritma proses yang dicantumkan antara lain adalah mengelola transaksi, mengelola memo, mengelola reminder, dan mengelola kategori.

5.1.5.1 Implementasi Kode Mengelola Transaksi

Kode mengelola transaksi dijalankan saat penguna memilih sub menu tambah transaksi dan daftar transaksi pada menu transaksi. Kode tersebut akan menyimpan data transaksi ketika pengguna menambahkan transaksi pada sub menu tambah transaksi dan akan menampilkan daftar transaksi yang sudah dibuat oleh pengguna ketika memilih sub menu daftar transaksi. Proses ubah dan hapus transaksi juga ada pada kode ini. Kode ini terdapat dalam file transaksi.js. Dibutuhkan suatu perulangan untuk menampilkan daftar transaksi, karena data yang ditampilkan bisa lebih dari satu. Kode perulangan ditunjukkan pada Kode 5.15.

```
for (j = 0; j < len_data; j++){
 //contoh inisialisasi variable
  var transaksi_id = results.rows.item(j).transaksi_id;

  //fungsi untuk menampilkan daftar transaksi
   if(tgl==hari_data){
     msg2 = "<li class='delete_trans"+transaksi_id+"'>";
     .......
     msg2 += "</li>";
     document.querySelector('.data_trans'+hari_data).innerHTML
           +=  msg2;
   }

  //fungsi untuk perhitungan total transaksi
  if(jenis_transaksi_id==1){
    total += nominal;
  }else if(jenis_transaksi_id==2){
    total -= nominal;
    $('.nominal'+trans_id).attr("style","color:red;
      float:right;");
  }else if(jenis_transaksi_id==3){
    total -= nominal;
    $('.nominal'+trans_id).attr("style","color:#fa8007;
     float:right;");
  }
  if(jenis_pembayaran_id == 2){
    $('.attr'+trans_id).append("(Credit Card)");
    $('.nominal'+trans_id).attr("style","color:#0781FA
      !important; float:right;");
  }
}
```

Kode 5.15 Kode Perulangan Menampilkan Daftar Transaksi

5.1.5.2 Implementasi Kode Mengelola Kategori

Kode mengelola kategori dijalankan saat penguna memilih sub menu kategori pada menu transaksi. Kategori terdiri dari 3, yaitu pemasukan, pengeluaran dan invest/tabungan. Dalam implementasinya, ketiga kategori ini hampir sama, sehingga akan diambil salah satu kode implementasi dari ketiga kategori tersebut, yaitu kategori pengeluaran. Kode mengelola kategori pengeluaran ini menampilkan daftar kategori pengeluaran yang telah dibuat pengguna dan akan menyimpan data kategori yang dimasukkan pengguna apabila pengguna memilih tambah kategori baru. Kode ini terdapat dalam file transaksi.js dan dapat dilihat di Kode 5.16.

```
for (i = 0; i < len; i++){
 //contoh inisialisasi variable
 var available = results.rows.item(i).is_available;

 //fungsi untuk menampilkan daftar kategori
 msg = "<li id='delete"+anggaran_id+"'>";
 .....
 msg += "</li>";
 document.querySelector('#list-pengeluaran').innerHTML +=
 msg;

 //fungsi untuk menghitung dan menampilkan progress bar
 view_progressbar(nilai,anggaran_id);
 if(available == 1){
  nilai_total += nilai;
 }
}
```

Kode 5.16 Kode Perulangan Menampilkan Daftar Kategori

Berbeda dengan kategori lain, kategori pengeluaran dalam menampilkannya juga dipanggil sebuah fungsi view_progressbar yang berisi pemanggilan fungsi percentage_proses berguna menampilkan prosentase penggunaan dana pada kategori pengeluaran tersebut. Kode tersebut ditunjukkan pada Kode 5.17.

```
function percentage_proses(percentage,id){
      var pbar=jQMProgressBar(id)
        .setOuterTheme('b')
        .setInnerTheme('e')
        .isMini(true)
        .setMax(100)
        .setStartFrom(0)
        .setInterval(10)
        .showCounter(false)
        .build();
      pbar.setValue(percentage);
};
```

Kode 5.17 Kode Menampilkan Progress Bar Kategori Pengeluaran

5.1.5.3 Implementasi Kode Mengelola Memo

Kode mengelola memo dijalankan saat penguna memilih menu memo. Kode tersebut akan menyimpan data memo ketika pengguna menambahkan memo dan akan menampilkan daftar memo yang sudah dibuat oleh pengguna. Proses ubah dan hapus memo juga ada pada kode ini. Kode 5.18, Kode 5.19 dan Kode 5.20 ini terdapat dalam file memo.js. Dibutuhkan suatu perulangan untuk menampilkan daftar memo, karena data yang ditampilkan bisa lebih dari satu. Kode perulangan ditunjukkan pada Kode 5.18.

```
for (i = 0; i < len; i++){
  //contoh inisialisasi variable
  var id = results.rows.item(i).memo_id;

  //fungsi menampilkan daftar memo
  msg += '<li class="memo-rows" id="delete'+id+'">';
  .....
  msg += '</li>';
}
$("#memo-list").html(msg);
```

Kode 5.18 Kode Perulangan Menampilkan Daftar Memo

```
if(id_memo){
  var action = "Update ";
  tx.executeSql('UPDATE tbl_memo SET judul = "'+judul+'",
     keterangan = "'+keterangan+'", last_update =
     "'+last_update+'" WHERE memo_id = "'+id_memo+'" AND user_id
     = "'+user+'"');
}
else{
  var action = "Add new ";
  tx.executeSql('INSERT INTO tbl_memo
     (judul,keterangan,user_id,last_update) VALUES
     ("'+judul+'","'+keterangan+'","'+user+'","'+last_update+'")
     ');
}
tx.executeSql('SELECT * FROM tbl_memo WHERE judul = "'+judul+'"
   AND keterangan="'+keterangan+'" AND user_id= "'+user+'" AND
   last_update="'+last_update+'"', [], function (tx, results) {
   var len = results.rows.length;
   if(len>0){
      alert(action+"memo success!");
      window.location.href = "memo.html";
   }
   else{
      alert(action+"memo failed!");
   }
});
```

Kode 5.19 Kode Tambah dan Edit Memo

```
var x = confirm("Are you sure?");
```

```
if(x){
  db.transaction(function (tx) {
    tx.executeSql('DELETE FROM tbl_memo WHERE memo_id =
      "'+id+'" AND user_id = "'+user+'"');

    tx.executeSql("SELECT * FROM tbl_memo WHERE user_id =
      '"+user+"'", [], function(tx,results){
      var len = results.rows.length, i;
      if(len > 0){
        $("#delete"+id).fadeOut();
      }
      else{
        location.reload();
      }
    });
  });
}
```

Kode 5.20 Kode Hapus Memo

5.1.5.4 Implementasi Kode Mengelola Reminder

Kode mengelola reminder dijalankan saat penguna memilih menu reminder. Kode tersebut akan menyimpan data reminder ketika pengguna menambahkan reminder dan akan menampilkan daftar reminder yang sudah dibuat oleh pengguna. Proses ubah status dan hapus reminder juga ada pada kode ini. Kode ini terdapat dalam file reminder.js. Dibutuhkan suatu perulangan untuk menampilkan daftar reminder, karena data yang ditampilkan bisa lebih dari satu. Kode perulangan ditunjukkan pada Kode 5.21.

```
for (i = 0; i < len; i++){
 //contoh inisialisasi variable
 var id = results.rows.item(i).reminder_id;

 //fungsi untuk menampilkan daftar reminder
 msg += "<li class='bottom-space' id='delete"+id+"'>";
 ......
 msg += "</li>";
 $('#list-reminder').html(msg);
}
```

Kode 5.21 Kode Perulangan Menampilkan Daftar Reminder

5.1.6 Implementasi Antarmuka Aplikasi

Pada implementasi antarmuka aplikasi akan ditampilkan hasil implementasi antarmuka aplikasi *mobile* pengelola keuangan pribadi.

5.1.6.1 Halaman *Welcome Screen*

Halaman *welcome screen* akan ditampilkan setelah pengguna membuka aplikasi. Halaman ini berfungsi untuk memberikan tampilan awal dimana pada saat

yang bersamaan terjadi proses pengecekan session dan pembuatan tabel dalam database diawal pemasangan aplikasi ke *device* di *background process*. Kode halaman *Welcome Screen* ini diimplementasikan pada index.html. Kode halaman *Welcome Screen* ditunjukkan pada Kode 5.22.

```
<body>
  <div id="welcome-container" data-role="page"
  class="main_page" data-title="MAUDI">
   <div class="form-login login-container" id="page-welcome">
     <div class="logo-welcome">
       <img src="../img/LogoBig.png" />
     </div>
   </div>
   <div data-role="footer" data-position="fixed" id="footer-
   cust" class="footer" data-tap-toggle="false">
    <div data-role="navbar">
     <ul>
       <li><a class="ui-btn-active ui-state-persist
           nav_active"><i class="fa fa-home"></i><br><span
           class="font-tab-bottom">Home</span></a></li>
       <li><a href="#login-container" data-ajax="false"
           class="bg_tab_footer"><i class="fa fa-
           user"></i><br><span class="font-tab-bottom">Log
           In</span></a></li>
     </ul>
    </div>
   </div>
  </div>
</body>
```

Kode 5.22 Kode *Welcome Screen*

Tampilan halaman *Welcome Screen* ditunjukkan Gambar 5.3.

Gambar 5.3 Tampilan Antarmuka Halaman *Welcome Screen*

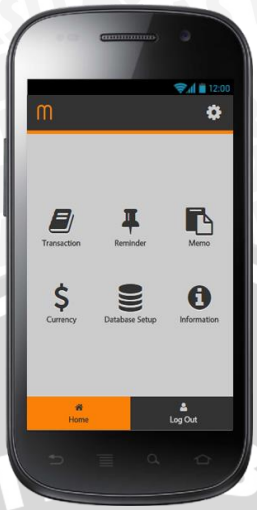
5.1.6.2 Halaman Utama (*Home*)

Halaman *home* ditampilkan setelah aplikasi dibuka dan pengguna melakukan log in. Halaman ini berisi menu-menu yang dapat dinikmati pengguna dalam melakukan pengelola keuangan pribadi. Kode halaman *Home* ini diimplementasikan pada Home.html. Kode halaman *Home* ditunjukkan pada Kode 5.23.

```
<body>
<div data-role="page" id="home_page" class="main_page" data-
title="HOME">
  <div data-role="header" class="ui-grid-a header" id="header-
  cust" data-position="fixed" style="overflow:hidden;" data-
  tap-toggle="false">
    <div class="ui-block-a" style="float:right;">
     <div class="option-top">
      <a href="#option-page" id="option-btn" data-ajax="false"
         data-rel="external"><i class="fa fa-gear"></i></a>
     </div>
    </div>
    <div class="ui-block-a logo-header-home">
     <img src="../img/LogoSmall.png" />
    </div>
  </div>
  <div class="ui-grid-b menu" id="menu-home">
    <div class="ui-block-a ui-block-menu">
     <a href="keuangan/keuangan.html" data-ajax="false" data-
        rel="external">
        <span class="icon-menu">
         <i class="fa fa-book"></i><br>
         <span class="font-menu">Transaction</span>
        </span>
     </a>
    </div>
    ......
  </div>
  //======KODE FOOTER======//
</div>
</body>
```

Kode 5.23 Kode Halaman *Home*

Tampilan halaman *Welcome Screen* ditunjukkan Gambar 5.4.



Gambar 5.4 Tampilan Antarmuka Halaman *Home*

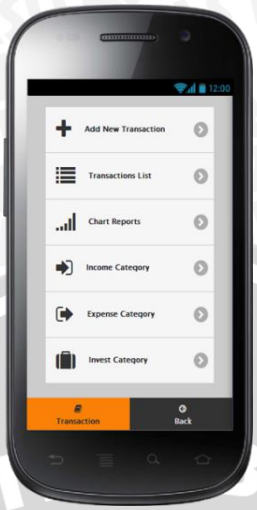
5.1.6.3 Halaman Sub Menu Transaksi

Halaman sub menu transaksi ini ditampilkan ketika pengguna memilih menu *transaction* pada halaman *home*. Pada menu *transaction* terdapat 6 (enam) sub menu, yaitu *add new transaction*, *transaction list*, *chart report*, *income category*, *expense category*, dan *invest category*. Kode halaman sub transaksi ini diimplementasikan pada keuangan.html. Kode halaman sub transaksi ditunjukkan pada Kode 5.24.

```
<body>
<div data-role="page" class="main_page" id="menu-keuangan"
data-title="KEUANGAN">
 <div class="keu-container" id="page-keuangan">
  <ul data-role="listview" data-filter="false">
   <li data-icon="carat-r">
    <a href="add.html" onclick="link_transaksi('')" data-
     transition="slide" data-ajax="false" data-rel="external">
     <i class="fa fa-plus"></i>
     <h2>Add New Transaction</h2>
    </a>
   </li>
   ......
  </ul>
 </div>
 //======KODE FOOTER=====//
</body>
```

Kode 5.24 Kode Halaman Sub Menu Transaksi

Tampilan halaman Sub Menu Transaksi ditunjukkan Gambar 5.5.



Gambar 5.5 Tampilan Antarmuka Sub Menu Transaksi

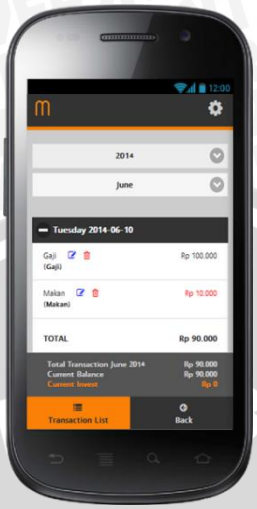
5.1.6.4 Menu Daftar Transaksi

Menu daftar transaksi ditampilkan ketika pengguna memilih menu daftar transaksi pada halaman sub menu transaksi. Pada menu daftar transaksi, aplikasi akan menampilkan daftar transaksi yang telah pengguna buat. Pengguna juga dapat melakukan pengubahan dan menghapus daftar transaksi dari halaman ini. Kode halaman daftar transaksi ini diimplementasikan pada transaksi.html. Kode halaman daftar transaksi ditunjukkan pada Kode 5.25.

```
<body>
<div data-role="page" id="transaksi-page" class="main_page"
data-title="KEUANGAN">
//==========KODE HEADER=========//
<span id="list-transaksi">
 <form>
  <div class="ui-field-contain transaksi-filter">
   <select name="tahun" id="select_tahun" data-mini="true">
     <option value="0">Select year</option>
   </select>
   <select name="bulan" id="select_bulan" data-mini="true">
     <option value="0">Select month</option>
   </select>
  </div>
 </form>
 <div class="transaksi-container" id="daftar-transaksi">
 //==========daftar=========//
 </div>
 <div class="kosong" id="kosong"></div>
</span>
//==========KODE FOOTER=========//
</div>
</body>
```

Kode 5.25 Kode Halaman Menu Daftar Transaksi

Tampilan halaman Menu Daftar Transaksi ditunjukkan Gambar 5.6.



Gambar 5.6 Tampilan Antarmuka Menu Daftar Transaksi

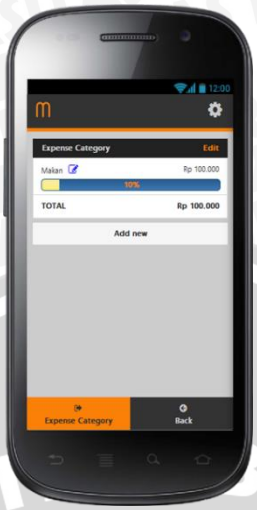
5.1.6.5 Menu Kategori

Menu Kategori ditampilkan ketika pengguna memilih salah satu dari menu kategori (pemasukan, pengeluaran dan invest/tabungan) pada halaman sub menu transaksi, karena implementasinya yang hampir sama maka akan diambil salah satu sebagai contoh yaitu kategori pengeluaran. Pada menu kategori pengeluaran, aplikasi akan menampilkan daftar kategori pengeluaran yang telah pengguna buat. Pengguna juga dapat melakukan pengubahan dan menghapus daftar kategori pengeluaran dari halaman ini. Kode halaman daftar kategori pengeluaran ini diimplementasikan pada pengeluaran.html. Kode halaman daftar kategori pengeluaran ditunjukkan pada Kode 5.26.

```
<div data-role="page" class="main_page" id="pengeluaran-page"
data-title="KEUANGAN">
//==========KODE HEADER=========//
<div class="transaksi-container">
 <ul data-role="listview" id="list-pengeluaran" data-
 inset="true" data-divider-theme="a">
  //==========daftar=========//
 </ul>
 <button id="tambah_pengeluaran" class="button">Add
   new</button><br><br>
 ......
</div>
//==========KODE FOOTER=========//
</div>
```

Kode 5.26 Kode Halaman Menu Kategori Pengeluaran

Tampilan halaman Menu Kategori Pengeluaran ditunjukkan Gambar 5.7.



Gambar 5.7 Tampilan Antarmuka Menu Kategori Pengeluaran

5.1.6.6 Menu *Reminder*

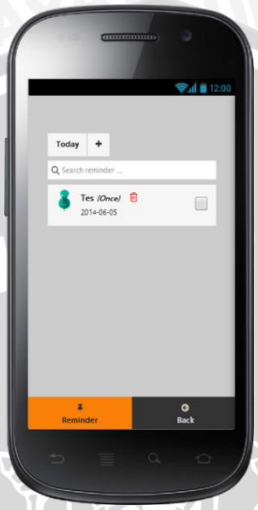
Menu *reminder* ditampilkan ketika pengguna memilih menu *reminder* pada halaman *home*. Pada menu reminder, aplikasi akan menampilkan daftar reminder yang telah pengguna buat. Pengguna juga dapat melihat daftar reminder hari ini dengan menekan tombol today pada menu reminder ini. Kode halaman daftar reminder ini diimplementasikan pada reminder.html. Berikut kode halaman daftar reminder ditunjukkan pada Kode 5.27.

```
<!DOCTYPE html>
<html>
<head>...</head>
<body>
<div data-role="page" class="main_page" id="reminder-index"
data-title="REMINDER">
 <div class="reminder-page-container">
  <div class="btn-reminder-container">
    <div class="btn-reminder-today"><a href="#" id="reminder-
       today" class="ui-btn">Today</a></div>
    <div class="btn-reminder-add-new"><a href="#reminder-new"
       class="ui-btn" data-ajax="false" data-
       rel="external"><i class="fa fa-plus"></i></a></div>
  </div><br>
  <div class="reminder-list">
    <ul data-role="listview" data-icon="false" id="list-
      reminder" data-split-theme="a" data-filter="true" data-
      filter-placeholder="Search reminder ..." data-
      inset="true">
    //==========daftar=========//
    </ul>
    <div id="div-kosong"></div>
  </div>
 </div>
//==========KODE FOOTER=========//
```

```
</div>
</body>
</html>
```

Kode 5.27 Kode Halaman Menu Reminder

Tampilan halaman Menu Kategori Pengeluaran ditunjukkan Gambar 5.8.



Gambar 5.8 Tampilan Antarmuka Menu *Reminder*

5.1.6.7 Menu Memo

Menu memo ditampilkan ketika pengguna memilih menu memo pada halaman *home*. Untuk memasuki menu memo, terlebih dahulu aplikasi akan meminta verifikasi berupa masukan *password* pengguna demi keamanan data pengguna. Kode halaman daftar memo ini diimplementasikan pada memo.html. Berikut kode halaman daftar memo ditunjukkan pada Kode 5.28.

```
<body>
<div id="memo-index" data-role="page"  class="main_page" data-
title="MAUDI">
  <div class="memo-page-container">
    <div class="btn-memo-container">
      <div class="btn-memo-add-new"><a href="#memo-new"
          class="ui-btn" data-ajax="false" data-rel="external"><i
          class="fa fa-plus"></i></a></div>
    </div><br>
    <div class="memo-list">
      <ul data-role="listview" data-split-icon="delete" id="memo-
          list" data-split-theme="b" data-filter="true" data-
          filter-placeholder="Search memo ..." data-inset="true">
      //==========daftar=========//
      </ul>
      <div id="div-kosong"></div>
    </div>
    ......
  </div>
//==========KODE FOOTER=========//
```

```
</div>
</body>
```

Kode 5.28 Kode Halaman Menu Memo

Tampilan halaman Menu Memo ditunjukkan Gambar 5.9.

Gambar 5.9 Tampilan Antarmuka Menu Memo

5.2 Pengujian

Pengujian yang dilakukan pada aplikasi ini dilakukan dengan tiga tahapan, yaitu pengujian validasi, pengujian unit dengan menggunakan *tools* QUnit, dan pengujian kompatibilitas. Untuk pengujian unit dengan menggunakan QUnit dikarenakan QUnit dapat digunakan pada jQuery, jQuery UI dan jQuery mobile.

5.2.1 Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. Daftar kebutuhan yang telah dirumuskan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Pengujian ini menggunakan metode pengujian *black-box*, karena tidak diperlukan konsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan kesesuaian antara kinerja sistem dengan daftar kebutuhan.

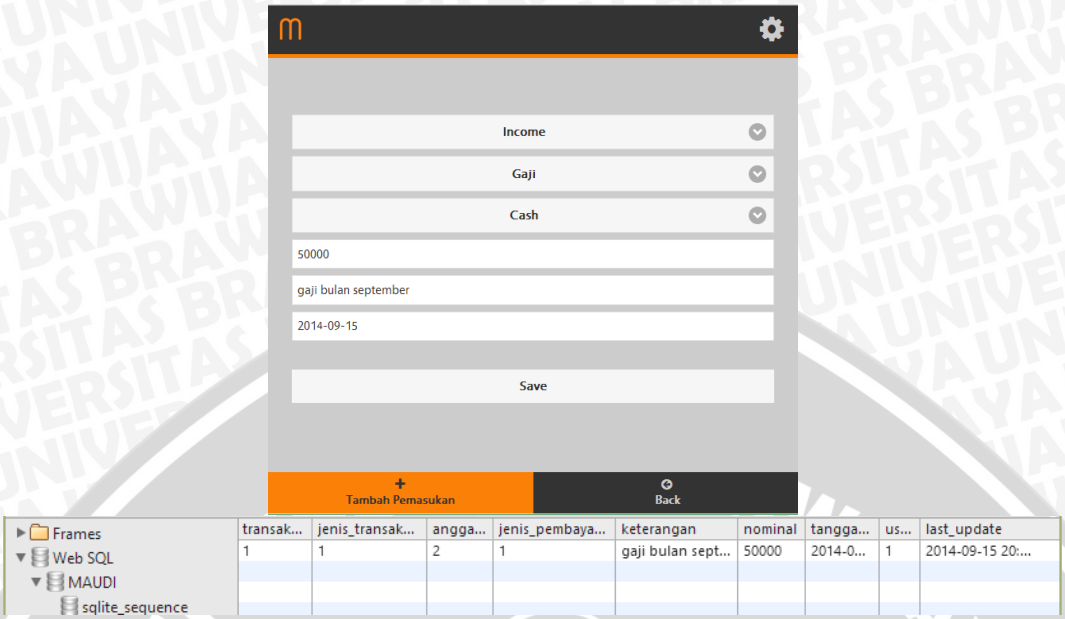
5.2.1.1 Kasus Uji

1. Kasus uji Mengelola Transaksi

Kasus uji mengelola transaksi terdiri dari beberapa kasus uji validasi yaitu tambah transaksi pada Tabel 5.4, edit transaksi pada Tabel 5.5, ubah transaksi pada Tabel 5.6, dan hapus transaksi pada Tabel 5.7.

Tabel 5.4 Kasus Uji Validasi Tambah Transaksi

Nama Kasus Uji	Kasus uji Tambah Transaksi
Objek Uji	Kebutuhan Fungsional (SRS_002_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat mencatat transaksi yang dimasukkan pengguna dengan benar
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Transaction 3. Memilih menu Add New Transaction 4. Sistem menampilkan form transaksi 5. Memasukkan data transaksi baru yang terdiri dari jenis transaksi *income*, kategori pemasukan gaji, jenis pembayaran *cash*, nominal 50000, keterangan gaji bulan September dan tanggal transaksi 2014-09-15 pada form yang tampil 6. Memilih tombol Save 7. Sistem akan menyimpan informasi data yang dimasukkan
Hasil yang Diharapkan	Aplikasi dapat menyimpan informasi transaksi sama dengan data yang dimasukkan atau ditambahkan pengguna.
Status Validitas (Android)	Valid, karena data yang dimasukkan pengguna sama dengan data yang tersimpan di dalam database dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data yang dimasukkan pengguna sama dengan data yang tersimpan di dalam database dan dapat berjalan dengan baik.



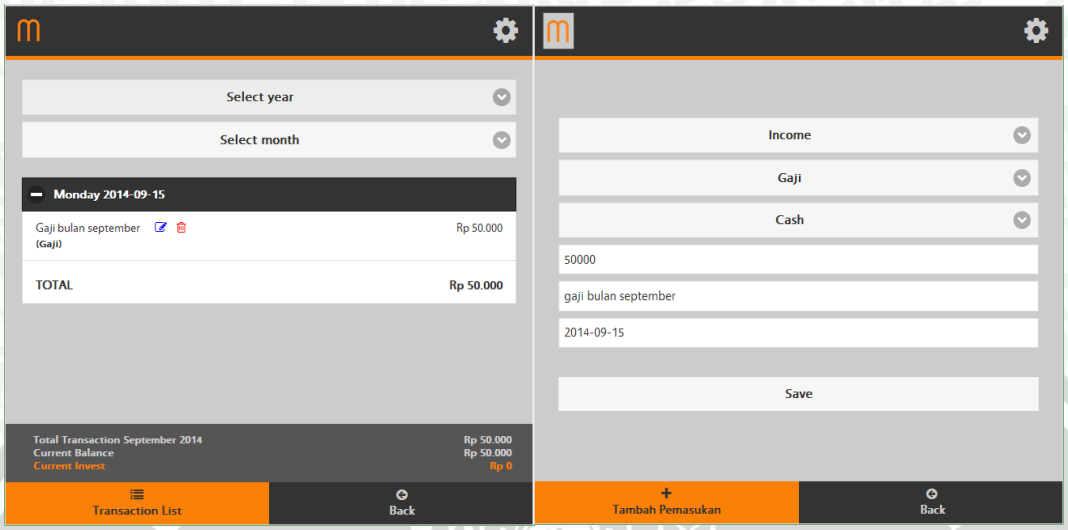
Gambar 5.10 Kasus uji tambah transaksi

Tabel 5.4 merupakan kasus uji tambah transaksi dengan kebutuhan fungsional SRS_002_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat mencatat transaksi sama dengan yang dimasukkan pengguna. Karena data yang dimasukkan pada aplikasi dan data yang tersimpan di database sama, sesuai Gambar 5.10, dan dapat berjalan baik maka status validitas pada kasus uji tambah transaksi dinyatakan valid.

Tabel 5.5 Kasus Uji Validasi Edit Transaksi

Nama Kasus Uji	Kasus uji Edit Transaksi
Objek Uji	Kebutuhan Fungsional (SRS_002_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat menampilkan data transaksi pada form sama dengan transaksi yang dipilih
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Transaction 3. Memilih menu Transaction List 4. Sistem menampilkan daftar transaksi bulan September 5. Memilih transaksi yang ingin diubah 6. Sistem menampilkan form edit transaksi yang berisi data yang sama dengan transaksi yang dipilih
Hasil yang Diharapkan	Aplikasi dapat menampilkan data transaksi yang sama dengan transaksi yang dipilih pada form
Status Validitas (Android)	Valid, karena data yang ditampilkan sama dengan data transaksi yang dipilih pada form dan dapat berjalan dengan baik.

Status Validitas (iOS)	Valid, karena data yang ditampilkan sama dengan data transaksi yang dipilih pada form dan dapat berjalan dengan baik.



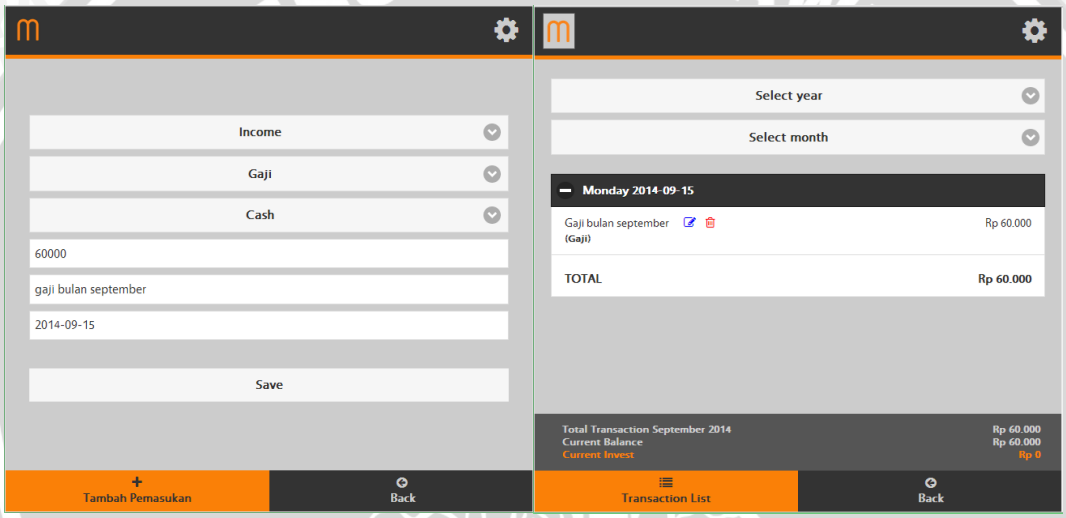
Gambar 5.11 Kasus uji edit transaksi

Tabel 5.5 merupakan kasus uji validasi edit transaksi dengan kebutuhan fungsional SRS_002_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat menampilkan data transaksi pada form sama dengan transaksi yang dipilih. Karena data yang dipilih pada aplikasi dan data yang ditampilkan pada form sama, sesuai Gambar 5.11, dan dapat berjalan baik maka status validitas pada kasus uji edit transaksi dinyatakan valid.

Tabel 5.6 Kasus Uji Validasi Ubah Transaksi

Nama Kasus Uji	Kasus uji Ubah Transaksi
Objek Uji	Kebutuhan Fungsional (SRS_002_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat mengubah transaksi yang dipilih pengguna sama dengan data yang dimasukkan pengguna
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Transaction 3. Memilih menu Transaction List 4. Sistem menampilkan daftar transaksi bulan September 5. Memilih transaksi yang ingin diubah 6. Sistem menampilkan form edit transaksi yang berisi data dari transaksi yang dipilih 7. Memasukkan data transaksi yang terdiri dari jenis transaksi *income*, kategori pemasukan gaji, jenis pembayaran *cash*,

	nominal 60000, keterangan gaji bulan September dan tanggal transaksi 201-09-15 pada form yang tampil 8. Memilih tombol Save 9. Sistem akan menyimpan perubahan informasi data yang dimasukkan
Hasil yang Diharapkan	Aplikasi dapat mengubah data informasi transaksi yang sama dengan data yang dimasukkan pengguna.
Status Validitas (Android)	Valid, karena transaksi telah berubah dan sama dengan data yang dimasukkan pengguna dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena transaksi telah berubah dan sama dengan data yang dimasukkan pengguna dan dapat berjalan dengan baik.



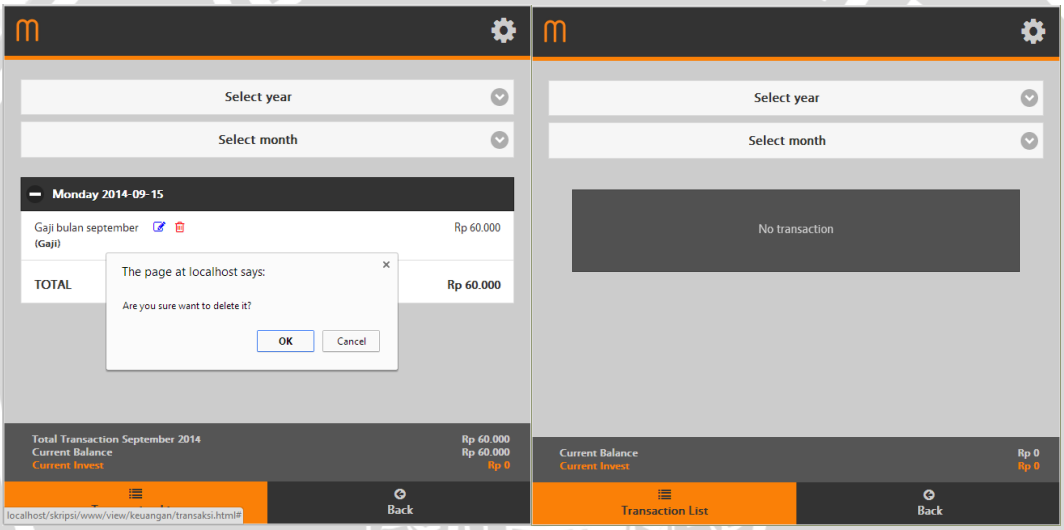
Gambar 5.12 Kasus uji ubah transaksi

Tabel 5.6 merupakan kasus uji validasi ubah transaksi dengan kebutuhan fungsional SRS_002_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat mengubah transaksi yang dipilih pengguna. Karena data yang diubah dan data yang ditampilkan pada daftar transaksi sama, sesuai Gambar 5.12, dan dapat berjalan baik maka status validitas pada kasus uji ubah transaksi dinyatakan valid.

Tabel 5.7 Kasus Uji Validasi Hapus Transaksi

Nama Kasus Uji	Kasus uji Hapus Transaksi
Objek Uji	Kebutuhan Fungsional (SRS_002_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat menghapus transaksi yang sama dengan yang dipilih pengguna
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Transaction

	3. Memilih menu Transaction List 4. Sistem menampilkan daftar transaksi bulan September 5. Memilih transaksi yang ingin dihapus 6. Sistem meminta konfirmasi penghapusan 7. Memilih tombol OK 8. Sistem akan menghapus informasi data transaksi yang dipilih
Hasil yang Diharapkan	Aplikasi dapat menghapus data transaksi yang sama dengan yang dipilih pengguna.
Status Validitas (Android)	Valid, karena transaksi yang dihapus sama dengan transaksi yang dipilih pengguna dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena transaksi yang dihapus sama dengan transaksi yang dipilih pengguna dan dapat berjalan dengan baik.



Gambar 5.13 Kasus uji hapus transaksi

Tabel 5.7 merupakan kasus uji validasi hapus transaksi dengan kebutuhan fungsional SRS_002_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat menghapus transaksi yang dipilih pengguna. Karena data yang dipilih dan data yang dihapus pada daftar transaksi sama, sesuai Gambar 5.13, dan dapat berjalan baik maka status validitas pada kasus uji hapus transaksi dinyatakan valid.

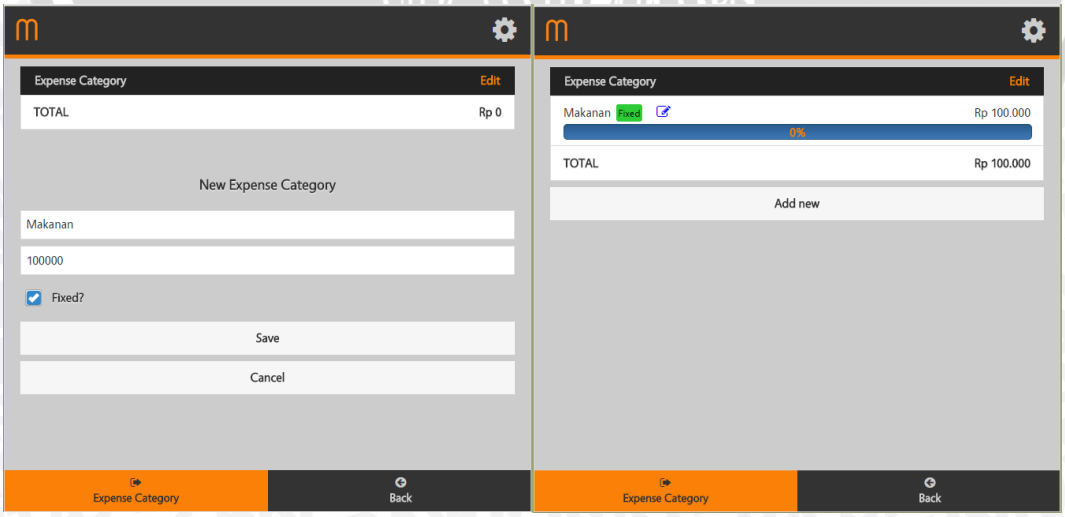
2. Kasus uji Mengelola Kategori (Pengeluaran)

Pengujian validasi untuk kasus uji mengelola kategori terdiri dari 3 kategori, pemasukan, pengeluaran dan invest/tabungan. Untuk pengujian validasi pada kategori pemasukan dan invest/tabungan telah dilakukan dengan prosedur yang sama dan mendapatkan hasil valid. Kasus uji mengelola kategori pengeluaran

sendiri terdiri dari beberapa kasus uji validasi yaitu tambah kategori pengeluaran pada Tabel 5.8, edit kategori pengeluaran pada Tabel 5.9, ubah kategori pengeluaran pada Tabel 5.10, dan hapus kategori pengeluaran pada Tabel 5.11.

Tabel 5.8 Kasus Uji Validasi Tambah Kategori Pengeluaran

Nama Kasus Uji	Kasus uji Tambah Kategori Pengeluaran
Objek Uji	Kebutuhan Fungsional (SRS_002_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat mencatat kategori pengeluaran sama dengan yang dimasukkan pengguna.
Prosedur Uji	1. Masuk Aplikasi 2. Memilih menu Transaction 3. Memilih menu Expense Category 4. Sistem menampilkan daftar kategori pengeluaran 5. Memilih tombol Add New 6. Sistem menampilkan form kategori pengeluaran 7. Memasukkan data kategori pengeluaran baru yang terdiri dari nama kategori dan nominal 100000 pada form yang tampil 8. Memilih tombol Save 9. Sistem akan menyimpan informasi data yang dimasukkan
Hasil yang Diharapkan	Aplikasi dapat menyimpan kategori pengeluaran sama dengan data yang dimasukkan atau ditambahkan pengguna.
Status Validitas (Android)	Valid, karena data yang dimasukkan pengguna sama dengan data yang tersimpan di database dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data yang dimasukkan pengguna sama dengan data yang tersimpan di database dan dapat berjalan dengan baik.

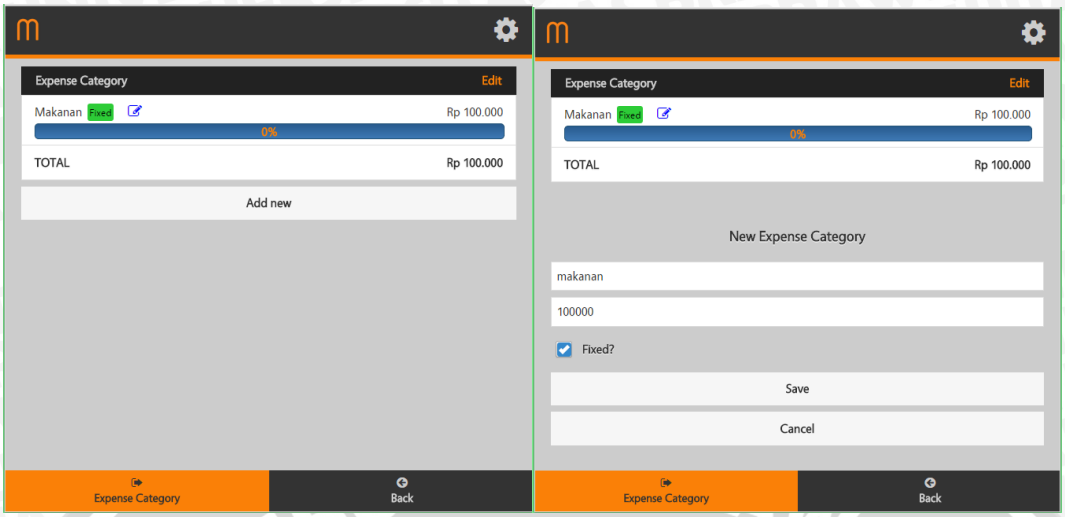


Gambar 5.14 Kasus uji tambah kategori pengeluaran

Tabel 5.8 merupakan kasus uji validasi tambah kategori pengeluaran dengan kebutuhan fungsional SRS_002_03 yang bertujuan untuk memastikan aplikasi yang dibuat dapat mencatat kategori pengeluaran sama dengan yang dimasukkan pengguna. Karena data yang dimasukkan pada aplikasi dan data yang tersimpan di database sama, sesuai Gambar 5.14, dan dapat berjalan baik maka status validitas pada kasus uji tambah kategori pengeluaran dinyatakan valid.

Tabel 5.9 Kasus Uji Validasi Edit Kategori Pengeluaran

Nama Kasus Uji	Kasus uji Edit Kategori Pengeluaran
Objek Uji	Kebutuhan Fungsional (SRS_002_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat menampilkan data kategori pengeluaran sama dengan yang dipilih pengguna pada form .
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Transaction 3. Memilih menu Expense Category 4. Sistem menampilkan daftar kategori pengeluaran 5. Memilih kategori pengeluaran yang ingin diubah 6. Sistem menampilkan form kategori pengeluaran berisi data kategori yang dipilih
Hasil yang Diharapkan	Aplikasi dapat menampilkan data kategori pengeluaran yang sama dengan yang dipilih pengguna pada form.
Status Validitas (Android)	Valid, karena data yang ditampilkan pada form sama dengan data yang dipilih pengguna dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data yang ditampilkan pada form sama dengan data yang dipilih pengguna dan dapat berjalan dengan baik.



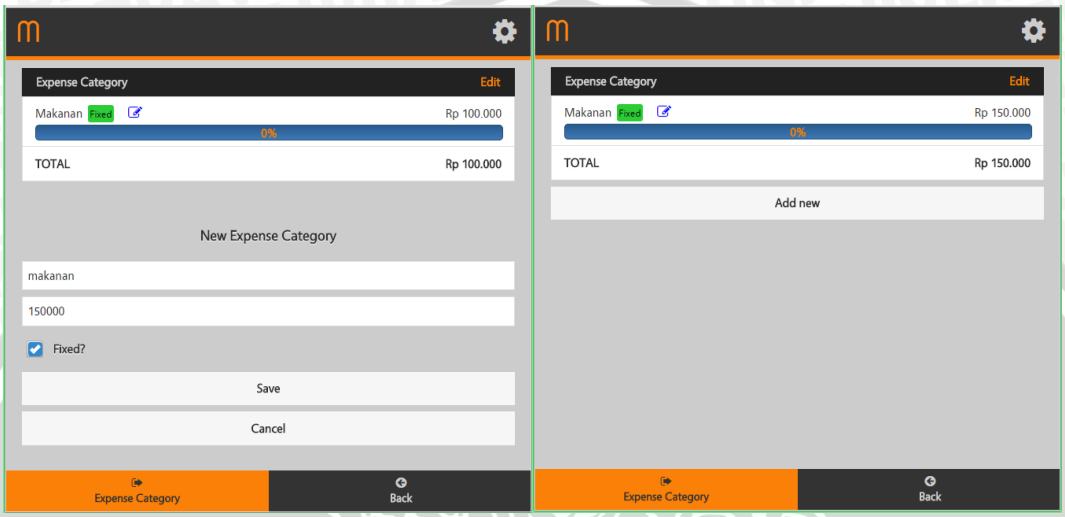
Gambar 5.15 Kasus uji edit kategori pengeluaran

Tabel 5.9 merupakan kasus uji validasi edit kategori pengeluaran dengan kebutuhan fungsional SRS_002_03 yang bertujuan untuk memastikan aplikasi yang dibuat dapat menampilkan data kategori pengeluaran pada form sama dengan yang dipilih pengguna. Karena data yang dipilih pada aplikasi dan data yang ditampilkan pada form sama, sesuai Gambar 5.15, dan dapat berjalan baik maka status validitas pada kasus uji edit kategori pengeluaran dinyatakan valid.

Tabel 5.10 Kasus Uji Validasi Ubah Kategori Pengeluaran

Nama Kasus Uji	Kasus uji Ubah Kategori Pengeluaran
Objek Uji	Kebutuhan Fungsional (SRS_002_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat mengubah kategori pengeluaran yang dipilih pengguna sama dengan data yang dimasukkan pengguna.
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Transaction 3. Memilih menu Expense Category 4. Sistem menampilkan daftar kategori pengeluaran 5. Memilih kategori pengeluaran yang ingin diubah 6. Sistem menampilkan form kategori pengeluaran berisi data kategori yang dipilih pada form 7. Mengubah data kategori pengeluaran pada form yang tampil yang terdiri dari nama kategori makanan dan nominal 150000 8. Memilih tombol Save 9. Sistem akan menyimpan informasi data yang dimasukkan
Hasil yang Diharapkan	Aplikasi dapat mengubah data kategori pengeluaran sama dengan data yang dimasukkan pengguna.

Status Validitas (Android)	Valid, karena data yang tersimpan di database dan data yang dimasukkan pengguna sama dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data yang tersimpan di database dan data yang dimasukkan pengguna sama dan dapat berjalan dengan baik.



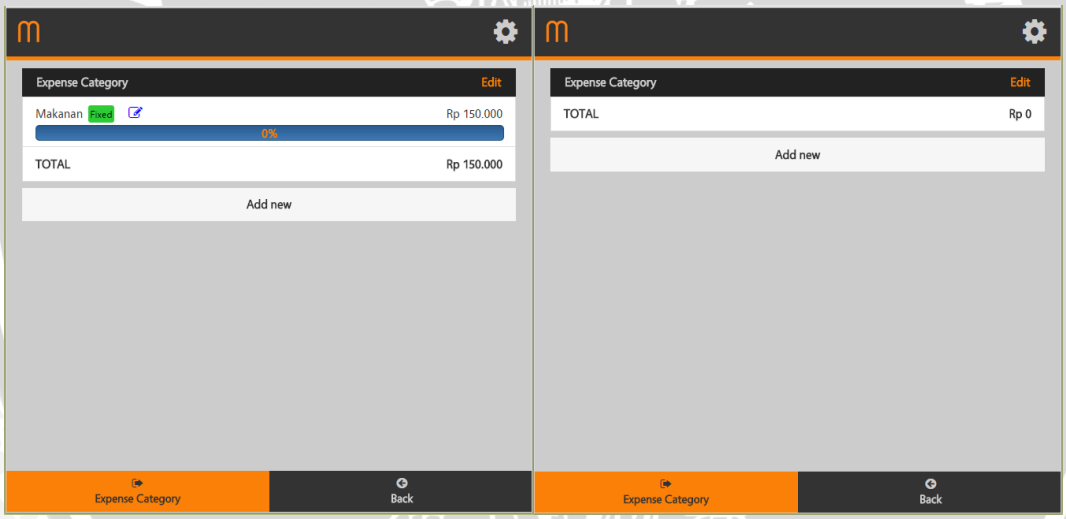
Gambar 5.16 Kasus uji ubah kategori pengeluaran

Tabel 5.10 merupakan kasus uji validasi ubah kategori pengeluaran dengan kebutuhan fungsional SRS_002_03 yang bertujuan untuk memastikan aplikasi yang dibuat dapat mengubah kategori pengeluaran yang dipilih pengguna sama dengan data yang dimasukkan pengguna. Karena data yang diubah dan data yang ditampilkan pada daftar kategori pengeluaran sama, sesuai Gambar 5.16, dan dapat berjalan baik maka status validitas pada kasus uji ubah kategori pengeluaran dinyatakan valid.

Tabel 5.11 Kasus Uji Validasi Hapus Kategori Pengeluaran

Nama Kasus Uji	Kasus uji Hapus Kategori Pengeluaran
Objek Uji	Kebutuhan Fungsional (SRS_002_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat menghapus kategori pengeluaran sama dengan yang dipilih pengguna.
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Transaction 3. Memilih menu Expense Category 4. Sistem menampilkan daftar kategori pengeluaran 5. Memilih tombol Edit pada kanan atas 6. Memilih kategori pengeluaran yang ingin dihapus

	7. Sistem meminta konfirmasi penghapusan 8. Memilih tombol OK 9. Sistem meminta konfirmasi penghapusan seluruh data transaksi untuk kategori pengeluaran tersebut 10. Jika memilih tombol Yes, Sistem akan menghapus kategori penghapusan yang dipilih beserta seluruh data transaksi untuk kategori penghapusan tersebut. 11. Jika memilih tombol No, Sistem akan menghapus (mengubah status menjadi not available) kategori penghapusan yang dipilih.
Hasil yang Diharapkan	Aplikasi dapat menghapus data kategori pengeluaran sama dengan yang dipilih pengguna.
Status Validitas (Android)	Valid, karena data yang dihapus sama dengan data yang dipilih pengguna dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data yang dihapus sama dengan data yang dipilih pengguna dan dapat berjalan dengan baik.



Gambar 5.17 Kasus uji hapus kategori pengeluaran

Tabel 5.11 merupakan kasus uji validasi hapus kategori pengeluaran dengan kebutuhan fungsional SRS_002_03 yang bertujuan untuk memastikan aplikasi yang dibuat dapat menghapus kategori pengeluaran sama dengan yang dipilih pengguna. Karena data yang dipilih dan data yang dihapus pada daftar kategori pengeluaran sama, sesuai Gambar 5.17, dan dapat berjalan baik maka status validitas pada kasus uji hapus kategori pengeluaran dinyatakan valid.

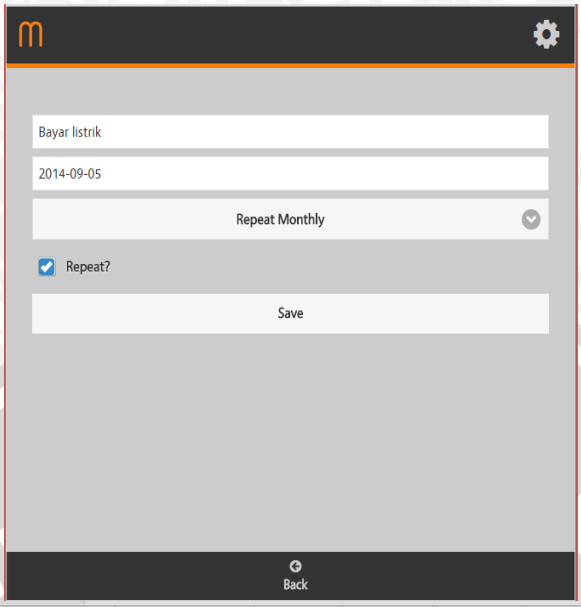
3. Kasus uji Mengelola Reminder

Kasus uji mengelola *reminder* terdiri dari beberapa kasus uji validasi yaitu tambah reminder pada Tabel 5.12, ubah status reminder pada Tabel 5.13, dan hapus reminder pada Tabel 5.14.

Tabel 5.12 Kasus Uji Validasi Tambah Reminder

Nama Kasus Uji	Kasus uji Tambah Reminder
Objek Uji	Kebutuhan Fungsional (SRS_003_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat mencatat reminder yang dimasukkan pengguna.
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Reminder 3. Sistem menampilkan daftar reminder 4. Memilih menu Add New Reminder 5. Sistem menampilkan form reminder 6. Memasukkan data reminder baru yang terdiri dari judul reminder bayar listrik, tanggal reminder 201-09-05,is_repeat iya dan repeat tiap bulan pada form yang tampil 7. Memilih tombol Save 8. Sistem akan menyimpan informasi data yang dimasukkan
Hasil yang Diharapkan	Aplikasi dapat menyimpan reminder sama dengan data yang dimasukkan atau ditambahkan pengguna.
Status Validitas (Android)	Valid, karena data yang tersimpan di database sama dengan yang dimasukkan pengguna dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data yang tersimpan di database sama dengan yang dimasukkan pengguna dan dapat berjalan dengan baik.

Tabel 5.12 merupakan kasus uji validasi tambah reminder dengan kebutuhan fungsional SRS_003_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat mencatat reminder sama dengan yang dimasukkan pengguna. Karena data yang dimasukkan pada aplikasi dan data yang tersimpan di database sama, sesuai Gambar 5.18, dan dapat berjalan baik maka status validitas pada kasus uji tambah reminder dinyatakan valid.



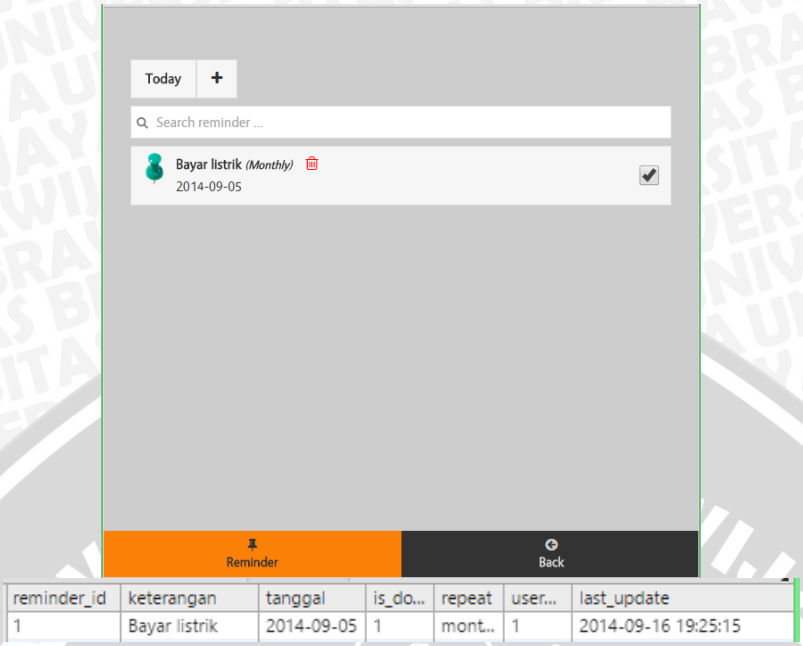
reminder_id	keterangan	tanggal	is_done	repeat	user_id	last_update
1	Bayar listrik	2014-09-05	0	monthly	1	2014-09-16 19:13:33

Gambar 5.18 Kasus uji tambah reminder

Tabel 5.13 Kasus Uji Validasi Ubah Status Reminder

Nama Kasus Uji	Kasus uji Ubah Reminder
Objek Uji	Kebutuhan Fungsional (SRS_003_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat mengubah status reminder yang sama dengan yang dipilih pengguna .
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Reminder 3. Sistem menampilkan daftar reminder 4. Mencentang reminder yang ingin diubah statusnya 5. Sistem akan menyimpan informasi data yang dimasukkan
Hasil yang Diharapkan	Aplikasi dapat mengubah status reminder yang sama dengan data yang dipilih pengguna.
Status Validitas (Android)	Valid, karena data reminder di database yang diubah sama dengan yang dipilih pengguna dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data reminder di database yang diubah sama dengan yang dipilih pengguna dan dapat berjalan dengan baik.

reminder_id	keterangan	tanggal	is_do...	repeat	user...	last_update
1	Bayar listrik	2014-09-05	0	mont...	1	2014-09-16 19:13:33



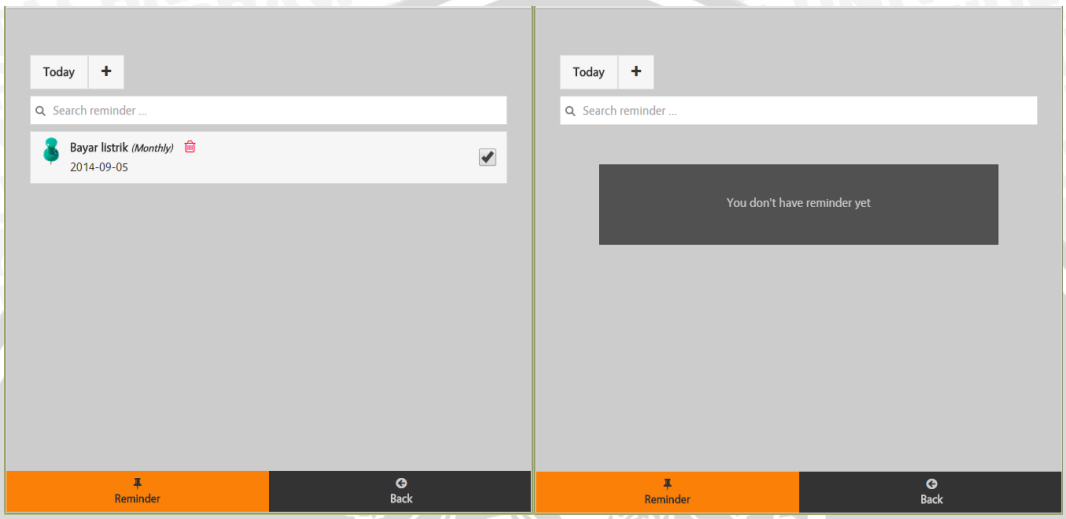
Gambar 5.19 Kasus uji ubah status reminder

Tabel 5.13 merupakan kasus uji validasi ubah status reminder dengan kebutuhan fungsional SRS_003_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat mengubah status reminder yang sama dengan yang dipilih pengguna. Karena data yang diubah dan data yang ada pada database sama, sesuai Gambar 5.19, dan dapat berjalan baik maka status validitas pada kasus uji ubah status reminder dinyatakan valid.

Tabel 5.14 Kasus Uji Validasi Hapus Reminder

Nama Kasus Uji	Kasus uji Hapus Reminder
Objek Uji	Kebutuhan Fungsional (SRS_003_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat menghapus reminder yang sama dengan yang dipilih pengguna.
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Reminder 3. Sistem menampilkan daftar reminder 4. Memilih reminder yang ingin dihapus 5. Sistem meminta konfirmasi penghapusan data reminder 6. Memilih tombol OK 7. Sistem akan menghapus informasi data yang dipilih
Hasil yang Diharapkan	Aplikasi dapat menghapus data reminder yang sama dengan yang dipilih pengguna.

Status Validitas (Android)	Valid, karena data reminder yang dihapus sama dengan yang dipilih pengguna dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data reminder yang dihapus sama dengan yang dipilih pengguna dan dapat berjalan dengan baik.



Gambar 5.20 Kasus uji hapus reminder

Tabel 5.14 merupakan kasus uji validasi hapus reminder dengan kebutuhan fungsional SRS_003_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat menghapus reminder yang sama dengan yang dipilih pengguna. Karena data yang dipilih dan data yang dihapus pada daftar reminder sama, sesuai Gambar 5.20, dan dapat berjalan baik maka status validitas pada kasus uji hapus reminder dinyatakan valid.

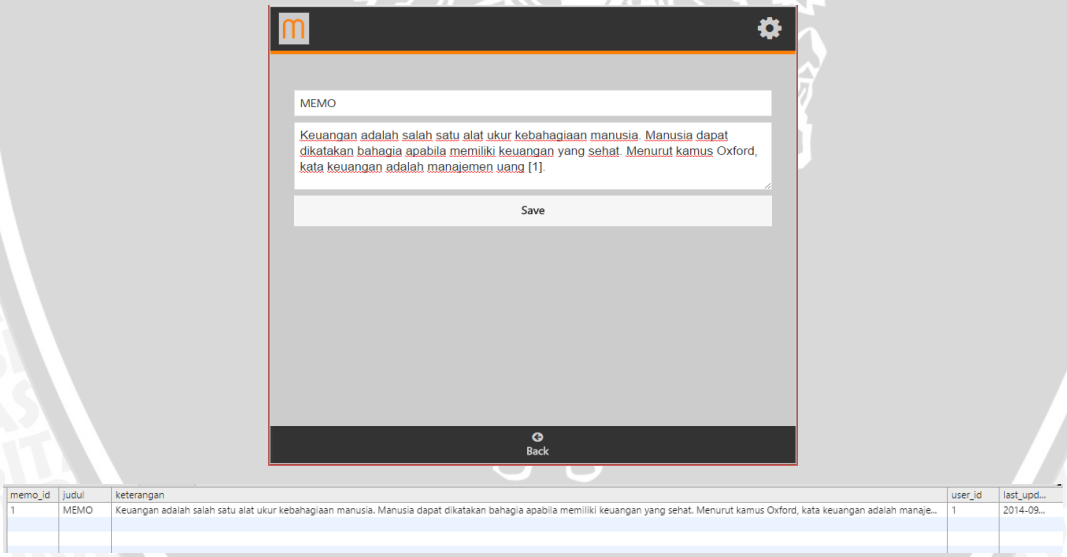
4. Kasus uji Mengelola Memo

Kasus uji mengelola memo terdiri dari beberapa kasus uji validasi yaitu tambah memo pada Tabel 5.15, edit memo pada Tabel 5.16, ubah memo pada Tabel 5.17, dan hapus memo pada Tabel 5.18.

Tabel 5.15 Kasus Uji Validasi Tambah Memo

Nama Kasus Uji	Kasus uji Tambah Memo
Objek Uji	Kebutuhan Fungsional (SRS_004_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat mencatat memo sama dengan yang dimasukkan pengguna.

Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Memo 3. Sistem meminta pengguna memasukkan password untuk keamanan data 4. Sistem menampilkan daftar memo 5. Memilih menu Add New Memo 6. Sistem menampilkan form memo 7. Memasukkan data memo baru yang terdiri dari judul memo Memo dan keterangan keuangan adalah … pada form yang tampil 8. Memilih tombol Save 9. Sistem akan menyimpan informasi data yang dimasukkan
Hasil yang Diharapkan	Aplikasi dapat menyimpan memo sama dengan data yang dimasukkan atau ditambahkan pengguna.
Status Validitas (Android)	Valid, karena data yang tersimpan di database sama dengan yang dimasukkan pengguna dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data yang tersimpan di database sama dengan yang dimasukkan pengguna dan dapat berjalan dengan baik.

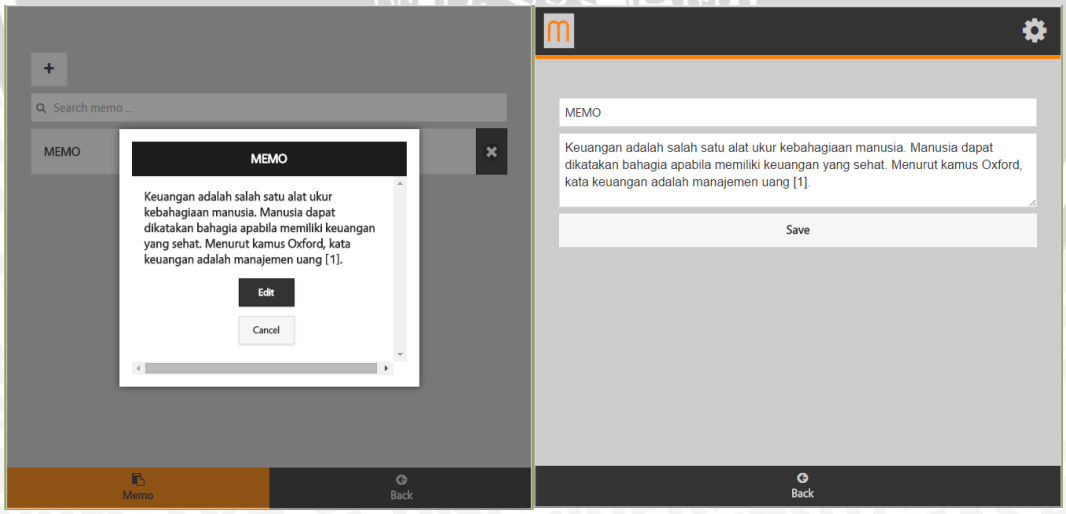


Gambar 5.21 Kasus uji tambah memo

Tabel 5.15 merupakan kasus uji validasi tambah memo dengan kebutuhan fungsional SRS_004_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat mencatat memo sama dengan yang dimasukkan pengguna. Karena data yang dimasukkan pada aplikasi dan data yang tersimpan di database sama, sesuai Gambar 5.21, dan dapat berjalan baik maka status validitas pada kasus uji tambah memo dinyatakan valid.

Tabel 5.16 Kasus Uji Validasi Edit Memo

Nama Kasus Uji	Kasus uji Edits Memo
Objek Uji	Kebutuhan Fungsional (SRS_004_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat menampilkan data memo yang sama dengan yang dipilih pengguna pada form .
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Memo 3. Sistem meminta pengguna memasukkan password untuk keamanan data 4. Sistem menampilkan daftar memo 5. Memilih memo yang ingin diubah 6. Sistem menampikan detail memo yang dipilih 7. Memilih tombol Edit 8. Sistem menampilkan form memo berisi data yang sama dengan memo yg dipilih
Hasil yang Diharapkan	Aplikasi dapat menampilkan data memo yang sama dengan yang dipilih pengguna.
Status Validitas (Android)	Valid, karena data yang ditampilkan pada form sama dengan data memo yang dipilih dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data yang ditampilkan pada form sama dengan data memo yang dipilih dan dapat berjalan dengan baik.



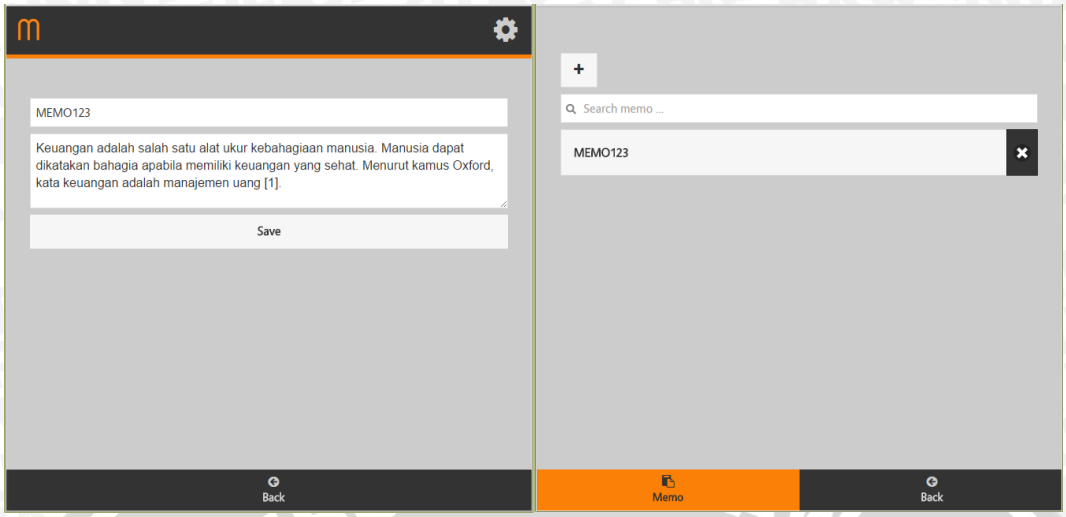
Gambar 5.22 Kasus uji edit memo

Tabel 5.16 merupakan kasus uji validasi edit memo dengan kebutuhan fungsional SRS_004_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat menampilkan data memo yang sama dengan yang dipilih pengguna pada

form. Karena data yang dipilih dan data yang ditampilkan sama, sesuai Gambar 5.22, dan dapat berjalan baik maka status validitas pada kasus uji edit memo dinyatakan valid.

Tabel 5.17 Kasus Uji Validasi Ubah Memo

Nama Kasus Uji	Kasus uji Ubah Memo
Objek Uji	Kebutuhan Fungsional (SRS_004_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat mengubah memo yang dipilih pengguna sama dengan data yang dimasukkan pengguna.
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Memo 3. Sistem meminta pengguna memasukkan password untuk keamanan data 4. Sistem menampilkan daftar memo 5. Memilih memo yang ingin diubah 6. Sistem menampikan detail memo yang dipilih 7. Memilih tombol Edit 8. Sistem menampilkan form memo berisi data yang sama dengan memo yg dipilih 9. Mengubah data memo yang terdiri dari judul memo MEMO123 dan keterangan keuangan adalah … pada form yang ditampilkan 10. Memilih tombol Save 11. Sistem akan menyimpan informasi data yang dimasukkan
Hasil yang Diharapkan	Aplikasi dapat mengubah memo sama dengan data yang dimasukkan pengguna.
Status Validitas (Android)	Valid, data yang diubah sama dengan data yang dimasukkan pengguna dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, data yang diubah sama dengan data yang dimasukkan pengguna dan dapat berjalan dengan baik.

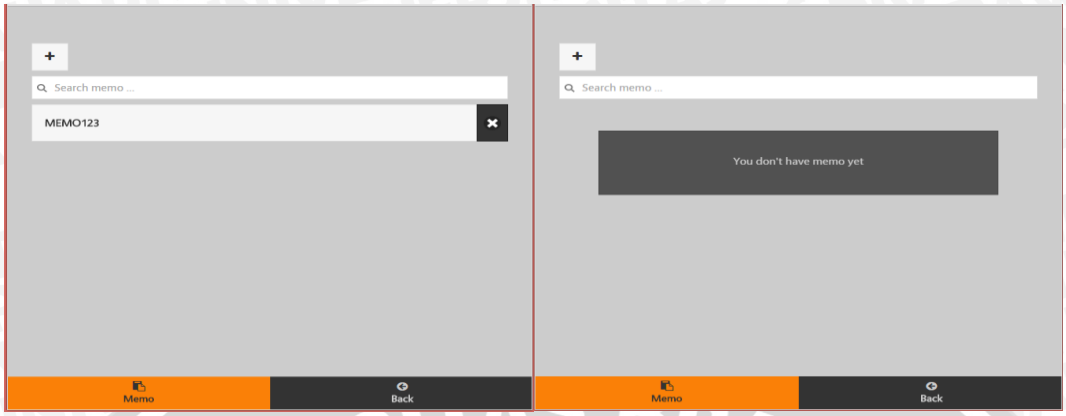


Gambar 5.23 Kasus uji ubah memo

Tabel 5.17 merupakan kasus uji validasi ubah memo dengan kebutuhan fungsional SRS_004_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat mengubah memo yang dipilih pengguna. Karena data yang diubah dan data yang ditampilkan pada daftar memo sama, sesuai Gambar 5.23, dan dapat berjalan baik maka status validitas pada kasus uji ubah memo dinyatakan valid.

Tabel 5.18 Kasus Uji Validasi Hapus Memo

Nama Kasus Uji	Kasus uji Hapus Transaksi
Objek Uji	Kebutuhan Fungsional (SRS_004_01)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi yang dibuat dapat menghapus memo yang dipilih pengguna.
Prosedur Uji	1. Menjalankan Aplikasi 2. Memilih menu Memo 3. Sistem meminta pengguna memasukkan password untuk keamanan data 4. Sistem menampilkan daftar memo 5. Memilih memo yang ingin dihapus 6. Sistem meminta konfirmasi penghapusan memo yang dipilih 7. Memilih tombol OK 8. Sistem akan menghapus memo yang dipilih
Hasil yang Diharapkan	Aplikasi dapat menghapus data memo yang sama dengan yang dipilih pengguna.
Status Validitas (Android)	Valid, karena data yang dihapus sama dengan data memo yang dipilih pengguna dan dapat berjalan dengan baik.
Status Validitas (iOS)	Valid, karena data yang dihapus sama dengan data memo yang dipilih pengguna dan dapat berjalan dengan baik.



Gambar 5.24 Kasus uji hapus memo

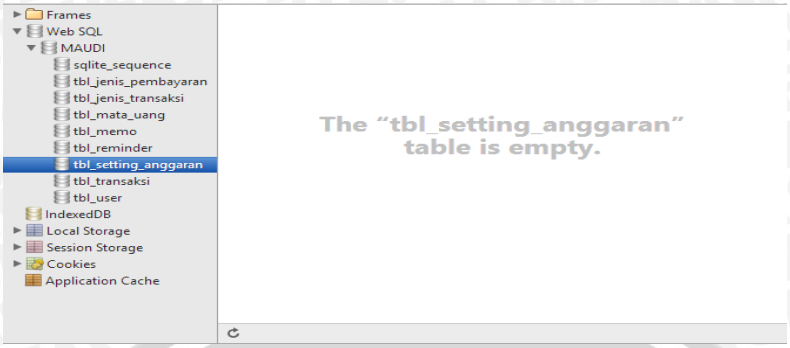
Tabel 5.18 merupakan kasus uji validasi hapus memo dengan kebutuhan fungsional SRS_004_01 yang bertujuan untuk memastikan aplikasi yang dibuat dapat menghapus memo yang sama dengan yang dipilih pengguna. Karena data yang dipilih dan data yang dihapus pada daftar memo sama, sesuai Gambar 5.24, dan dapat berjalan baik maka status validitas pada kasus uji hapus memo dinyatakan valid.

5.2.2 Pengujian Unit

Pengujian unit dilakukan untuk melihat apakah algoritma berjalan dengan benar. Pengujian dilakukan dengan menggunakan *tools* QUnit dengan melakukan pengujian kode program javascipt. Untuk kasus uji pada pengujian unit ini, dibatasi pada fungsi mengelola kategori pemasukan. Pengujian dilakukan pada proses lihat, tambah, ubah dan hapus data kategori pemasukan. Dalam penelitian ini juga telah dilakukan pengujian unit pada semua fungsi dengan prosedur yang sama dan mendapatkan hasil *passed*.

5.2.2.1 Kasus Uji

Kasus uji yang pertama adalah membaca atau melihat data kategori pemasukan, apakah sesuai dengan keadaan database atau tidak. Gambar 5.25 adalah keadaan awal database sebelum dilakukan pengujian.



Gambar 5.25 Gambar keadaan awal database tabel tbl_setting_anggaran

```
var db = openDatabase('MAUDI', '1.0', 'MAUDI DB', 2000000);
var kurs = window.localStorage.getItem('kurs');
var user = window.localStorage.getItem('user_id');
var nilai_tukar =
      window.localStorage.getItem('nilai_tukar_ke_rupiah');
var current = new Date();
var last_update = current.getFullYear()+"-
"+("0"+(current.getMonth()+1)).slice(-2)+"-
"+("0"+current.getDate()).slice(-2)+"
"+("0"+current.getHours()).slice(-
2)+":"+("0"+current.getMinutes()).slice(-
2)+":"+("0"+current.getSeconds()).slice(-2);
```

Kode 5.29 Kode inisialisasi variable yang digunakan

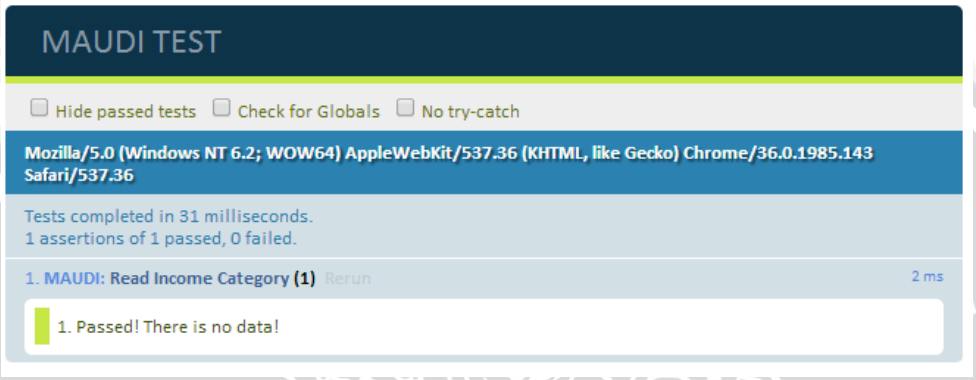
```
db.transaction(function (tx) {
   var jenis_transaksi_id = 1;
   var id=1;
   var keterangan = "Testing Data";
   var jumlah_pemasukan = 69;
   var is_tersedia = 1;
   //===read===//
   tx.executeSql('SELECT anggaran_id FROM
   tbl_setting_anggaran WHERE keterangan = "'+keterangan+'"
   AND jenis_transaksi_id = "'+jenis_transaksi_id+'" AND
   user_id = "'+user+'"', [], function (tx, results) {
      var row = results.rows.length;
      test_read_category(row);
   });
});
function test_read_category(row){
  QUnit.module("MAUDI");
  QUnit.test( "Read Income Category", function( assert ){
      assert.ok( row == 0 , "Passed! There is no data!");
  });
}
```

Kode 5.30 Kode pengujian QUnit kasus uji pertama

Pada kasus uji pertama pengujian QUnit untuk proses membaca data dari kategori pemasukan.

a. Kode 5.29 adalah inisialisasi dari variable yang digunakan dalam membaca data kategori pemasukan.
b. Kode 5.30 adalah proses testing atau pengujian. Baris 19 adalah proses perbandingan, antara jumlah data yang terbaca dengan angka yang diharapkan. Apabila didapatkan hasil yang sama, maka hasil yang diharapkan tercapai.

Hasil pengujian unit pada kasus uji pertama dijelaskan pada Gambar 5.26 :



Gambar 5.26 Hasil pengujian unit kasus uji pertama

Dalam Gambar 5.26 di atas menjelaskan hasil pengujian terhadap kasus uji yang dilakukan pada Kode 5.29. Pengujian pada kasus uji pertama didapatkan hasil alokasi waktu test sebesar 31 ms dengan 1 dari 1 pernyataan dinyatakan diterima *(passed)* dan 0 dinyatakan ditolak *(failed)*.

Kasus uji yang kedua adalah menambahkan data kategori pemasukan, apakah berhasil atau tidak.

```
1   db.transaction(function (tx) {
2     var jenis_transaksi_id = 1;
3     var id=1;
4     var keterangan = "Testing Data"; //insert
5     var jumlah_pemasukan = 69;
6     var is_tersedia = 1;
7     //===insert / update===//
8     tx.executeSql('SELECT anggaran_id FROM
9     tbl_setting_anggaran WHERE keterangan = "'+keterangan+'"
10    AND jenis_transaksi_id = "'+jenis_transaksi_id+'" AND
11    user_id = "'+user+'"', [], function (tx, results) {
12      var cek = results.rows.length;
13      if(cek == 0){
14        if(id){
15          var action = "Update ";
16        }else{
17          var action = "Add new ";
18
```

```
19            tx.executeSql('INSERT INTO tbl_setting_anggaran
20              (jenis_transaksi_id, keterangan, nominal,
21              is_available, user_id, last_update) VALUES
22              ("'+jenis_transaksi_id+'", "'+keterangan+'",
23              "'+(jumlah_pemasukan*nilai_tukar)+'", "1",
24              "'+user+'", "'+last_update+'")');
25          }
26        }else{
27            var action = "duplicate";
28        }
29        if(action !== "duplicate"){
30          tx.executeSql('SELECT anggaran_id FROM
31            tbl_setting_anggaran WHERE user_id = "'+user+'" AND
32            jenis_transaksi_id = "'+jenis_transaksi_id+'" AND
33            keterangan = "'+keterangan+'" AND nominal =
34            "'+(jumlah_pemasukan*nilai_tukar)+'"', [], function
35            (tx, results) {
36              var len = results.rows.length;
37              if(len>0){
38                var status = action+"income category success!";
39              }else{
40                var status = action+"income category failed!";
41              }
42              test_insert_category(len, status); //insert
43            });
44        }
45      });
46  });
47  //insert
48  function test_insert_category(row, status){
49    QUnit.module("MAUDI");
50    QUnit.test( "Insert Income Category", function( assert ){
51      assert.ok( row > 0 , status);
52    });
53  }
```

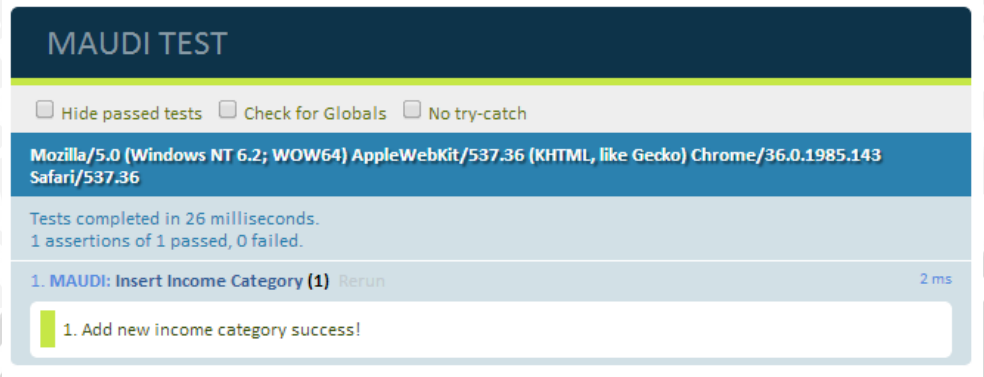
Kode 5.31 Kode pengujian QUnit kasus uji kedua

Pada kasus uji kedua pengujian QUnit untuk proses menambahkan data kategori pemasukan.

a. Pada Kode 5.31 baris 1-27 adalah kode untuk menambahkan dan mengubah data kategori pemasukan. Baris 4 adalah data yang ditambahkan.
b. Kode baris 27-43 adalah kode untuk pengecekan data, apakah berhasil atau tidak. Kode baris 41 adalah kode untuk pemanggilan testing kasus uji kedua dengan melemparkan jumlah data yang terbaca.
c. Baris 47-52 adalah kode untuk proses testing kasus uji kedua, dimana baris 50 adalah proses perbandingan, antara jumlah data yang terbaca

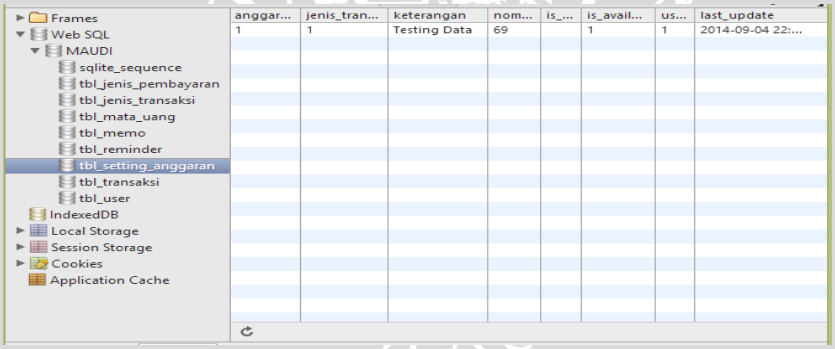
dengan angka yang diharapkan. Apabila didapatkan hasil yang sama, maka hasil yang diharapkan tercapai.

Hasil pengujian unit pada kasus uji kedua dijelaskan pada Gambar 5.27 dan keadaan database setelah dilakukan kasus uji kedua pada Gambar 5.28 :



Gambar 5.27 Hasil pengujian unit kasus uji kedua

Dalam Gambar 5.27 di atas menjelaskan hasil pengujian terhadap kasus uji yang dilakukan pada Kode 5.31. Pengujian pada kasus uji kedua didapatkan hasil alokasi waktu test sebesar 26 ms dengan 1 dari 1 pernyataan dinyatakan diterima *(passed)* dan 0 dinyatakan ditolak *(failed)*.



Gambar 5.28 Keadaan database setelah pengujian kasus uji kedua

Kasus uji yang ketiga adalah mengubah data kategori pemasukan, apakah berhasil atau tidak.

```
1  db.transaction(function (tx) {
2     var jenis_transaksi_id = 1;
3     var id=1;
4     var keterangan = "Testing Update Data"; //update
5     var jumlah_pemasukan = 69;
6     var is_tersedia = 1;
7     //===insert / update===//
8     tx.executeSql('SELECT anggaran_id FROM
9     tbl_setting_anggaran WHERE keterangan = "'+keterangan+'"
10
```

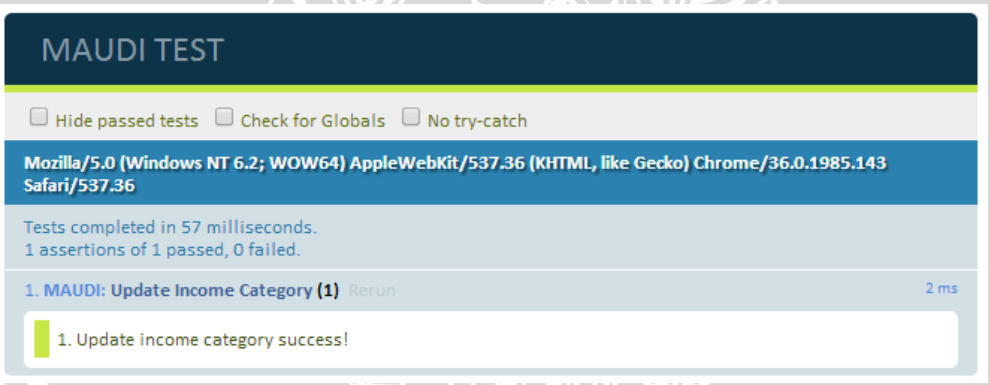
```
11    AND jenis_transaksi_id = "'+jenis_transaksi_id+'" AND
12    user_id = "'+user+'"', [], function (tx, results) {
13      var cek = results.rows.length;
14      if(cek == 0){
15        if(id){
16          var action = "Update ";
17          tx.executeSql('UPDATE tbl_setting_anggaran SET
18            is_available = "'+is_tersedia+'", keterangan =
19            "'+keterangan+'", nominal =
20            "'+(jumlah_pemasukan*nilai_tukar)+'",
21            last_update = "'+last_update+'" WHERE user_id =
22            "'+user+'" AND anggaran_id = "'+id+'"');
23        }else{
24          var action = "Add new ";
25        }
26      }else{
27        for (j = 0; j < cek; j++){
28          var anggaran_id = results.rows.item(j).anggaran_id;
29        }
30        if(id && id == anggaran_id){
31          var action = "Update ";
32          tx.executeSql('UPDATE tbl_setting_anggaran SET
33              is_available = "'+is_tersedia+'", keterangan =
34              "'+keterangan+'", nominal =
35              "'+(jumlah_pemasukan*nilai_tukar)+'",
36              last_update = "'+last_update+'" WHERE user_id =
37              "'+user+'" AND anggaran_id = "'+id+'"');
38        }else{
39          var action = "duplicate";
40        }
41      }
42      if(action !== "duplicate"){
43        tx.executeSql('SELECT anggaran_id FROM
44          tbl_setting_anggaran WHERE user_id = "'+user+'" AND
45          jenis_transaksi_id = "'+jenis_transaksi_id+'" AND
46          keterangan = "'+keterangan+'" AND nominal =
47          "'+(jumlah_pemasukan*nilai_tukar)+'"', [], function
48          (tx, results) {
49            var len = results.rows.length;
50            if(len>0){
51              var status = action+"income category success!";
52            }else{
53              var status = action+"income category failed!";
54            }
55            test_update_category(len, status); //update
56          });
57      }
58    });
59  });
60  //update
61  function test_update_category(row, status){
62    QUnit.module("MAUDI");
63    QUnit.test( "Update Income Category", function( assert ){
64      assert.ok( row > 0 , status);
65    });
    }
```

Kode 5.32 Kode pengujian QUnit kasus uji ketiga

Pada kasus uji ketiga pengujian QUnit untuk proses mengubah data kategori pemasukan.

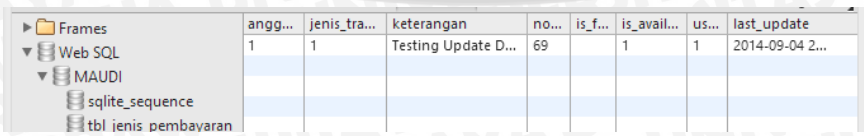
a. Pada Kode 5.32 baris 1-40 adalah kode untuk mengubah data kategori pemasukan. Kode baris 5 adalah data yang dipakai untuk mengubah data sebelumnya.
b. Pada baris 41-56 adalah kode untuk melakukan pengecekan data, apakah berhasil atau tidak. Dimana baris 54 adalah kode untuk pemanggilan testing kasus uji ketiga dengan melemparkan jumlah data yang terbaca.
c. Sedangkan baris 60-65 adalah kode untuk proses testing kasus uji ketiga, dimana baris 63 adalah proses perbandingan, antara jumlah data yang terbaca dengan angka yang diharapkan. Apabila didapatkan hasil yang sama, maka hasil yang diharapkan tercapai.

Hasil pengujian unit pada kasus uji ketiga dijelaskan pada Gambar 5.29 dan keadaan database setelah dilakukan kasus uji ketiga pada Gambar 5.30 berikut:



Gambar 5.29 Hasil pengujian unit kasus uji ketiga

Dalam Gambar 5.29 di atas menjelaskan hasil pengujian terhadap kasus uji yang dilakukan pada Kode 5.32. Pengujian pada kasus uji ketiga didapatkan hasil alokasi waktu test sebesar 57 ms dengan 1 dari 1 pernyataan dinyatakan diterima *(passed)* dan 0 dinyatakan ditolak *(failed)*.



Gambar 5.30 Keadaan database setelah pengujian kasus uji ketiga

Kasus uji yang keempat adalah menghapus data kategori pemasukan, apakah berhasil atau tidak.

```
1  db.transaction(function (tx) {
2     seleksi_delete_category(1,id,jenis_transaksi_id); //if
3     transaction data not delete
4     seleksi_delete_category(2,id,jenis_transaksi_id); //if
5     transaction data delete
6  });
7  function seleksi_delete_category(button,id,jenis_transaksi){
8        if(button==2){
9              //alert("delete");
10             delete_category_proses(id,jenis_transaksi);
11       }
12       else if(button==1){
13             //alert("update");
14             update_category_available(id,jenis_transaksi);
15       }
16 }
17 function delete_category_proses(id,jenis_transaksi){
18  db.transaction(function (tx) {
19   tx.executeSql('DELETE FROM tbl_setting_anggaran WHERE
20     anggaran_id = "'+id+'" AND user_id = "'+user+'"');
21   tx.executeSql('DELETE FROM tbl_transaksi WHERE anggaran_id
22       = "'+id+'" AND user_id = "'+user+'"');
23   tx.executeSql("SELECT * FROM tbl_setting_anggaran WHERE
24      anggaran_id = '"+id+"' AND user_id = '"+user+"' AND
25      jenis_transaksi_id ='"+jenis_transaksi+"'", [],
26      function(tx,results){
27        var len = results.rows.length, i;
28        if(len > 0){
29         var status = "Delete income category with all
30          transaction data with that category failed!";
31        }
32        else{
33         var status = "Delete income category with all
34          transaction data with that category success!";
35        }
36        test_delete_category(len, status);
37      });
38 });
39 };
40 //delete
41 function test_delete_category(row, status){
42    QUnit.module("MAUDI");
43    QUnit.test( "Delete Income Category with all transaction
44 data for that category", function( assert ){
45        assert.ok( row == 1 , status); //if transaction data
46 not delete
47        assert.ok( row == 0 , status); //if transaction data
48 delete
49    });
50 }
```

Kode 5.33 Kode pengujian QUnit kasus uji keempat

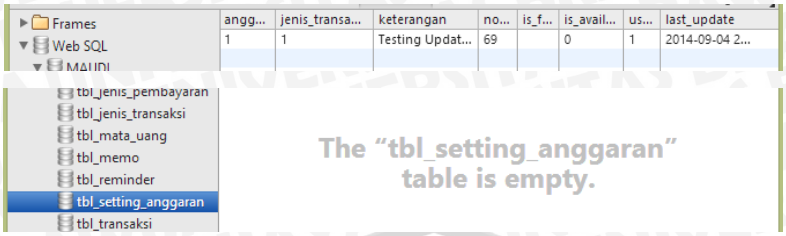
Pada kasus uji kedua pengujian QUnit untuk proses mengubah data kategori pemasukan.

a. Pada Kode 5.33 baris 1-39 adalah kode untuk menghapus data kategori pemasukan. Kode baris 36 adalah kode pemanggilan proses testing.
b. Sedangkan baris 40-50 adalah kode untuk proses testing kasus uji keempat, dimana baris 45 dan 47 adalah proses perbandingan, antara jumlah data yang terbaca dengan angka yang diharapkan. Apabila didapatkan hasil yang sama, maka hasil yang diharapkan tercapai.

Hasil pengujian unit pada kasus uji keempat dijelaskan pada Gambar 5.31 dan keadaan database setelah dilakukan kasus uji ketiga pada Gambar 5.32 berikut:

Gambar 5.31 Hasil pengujian unit kasus uji keempat

Dalam Gambar 5.31 di atas menjelaskan hasil pengujian terhadap kasus uji yang dilakukan pada Kode 5.33. Pengujian pada kasus uji keempat didapatkan hasil alokasi waktu test sebesar 31 ms dan 49 ms dengan 1 dari 1 pernyataan dinyatakan diterima *(passed)* dan 0 dinyatakan ditolak *(failed)*.



Gambar 5.32 Keadaan database setelah pengujian kasus uji keempat

5.2.3 Pengujian Kompatibilitas

Pengujian kompatibilitas digunakan untuk mengetahui kompatibilitas antarmuka sistem pada perangkat bergerak Android dan iOS. Pengujian kompatibilitas dilakukan untuk mengetahui validitas kinerja fitur-fitur yang disediakan oleh sistem terkait dengan antarmuka sistem. Tabel 5.19 menjelaskan prosedur dan hasil kasus uji pengujian kompatibilitas pada sistem operasi Android dan iOS.

Tabel 5.19 Kasus Uji dan Hasil Pengujian Kompatibilitas Android dan iOS

Nama Kasus Uji	Pengujian kompatibilitas Android dan iOS
Objek Uji	Kebutuhan Nonfungsional
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui validitas kinerja dari sistem dalam menyediakan fitur-fitur dan antarmuka pengguna
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 (Android)	Valid
Status Validitas (iOS)	Valid

5.3 Analisis Hasil Pengujian

Proses analisis terhadap hasil pengujian dilakukan untuk mendapatkan kesimpulan dari hasil pengujian aplikasi pengelola keuangan pribadi yang telah dilakukan. Proses analisis mengacu pada hasil pengujian yang didapatkan. Analisis dilakukan terhadap hasil pengujian di setiap tahap pengujian. Proses analisis yang dilakukan meliputi analisis hasil pengujian validasi, dan analisis hasil pengujian unit.

5.3.1 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 dengan kasus uji mengelola transaksi, mengelola kategori (pemasukan, pengeluaran, dan tabungan/investasi), mengelola memo dan mengelola reminder, dapat disimpulkan bahwa implementasi dan fungsionalitas aplikasi pengelola keuangan pribadi telah memenuhi kebutuhan yang dijabarkan pada Gambar 4.1 tahap analisis kebutuhan yang diantaranya adalah kebutuhan pengguna yang harus dapat melakukan pencatatan transaksi keuangan, pengguna yang harus dapat melakukan pengalokasian penggunaan dana ke dalam pos kategori dan pengguna yang harus dapat melakukan pencatatan informasi-informasi terkait proses transaksi keuangan. Oleh karena kebutuhan mengelola transaksi, mengelola kategori, mengelola memo dan mengelola reminder telah terpenuhi dengan baik berdasarkan hasil pengujian validasi maka dapat disimpulkan aplikasi yang telah dibangun telah berjalan/berfungsi dengan baik pada perangkat *mobile* Android dan iOS.

5.3.2 Analisis Hasil Pengujian Unit

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

5.3.3 Analisis Hasil Pengujian Kompatibilitas

Proses analisis terhadap hasil pengujian kompatibilitas yang dilakukan pada aplikasi mobile terhadap sistem operasi Android dan iOS dilakukan untuk melihat kesesuaian antara hasil kinerja sistem dengan daftar kebutuhan. Berdasarkan hasil pengujian kompatibilitas dapat disimpulkan bahwa implementasi dan fungsionalitas aplikasi *mobile* pengelola keuangan pribadi telah memenuhi kebutuhan yang telah dijabarkan pada tahap analisis kebutuhan.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil analisis perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut :

1. Perancangan aplikasi mobile pengelola keuangan pribadi telah dibuat sesuai dengan spesifikasi kebutuhan dan perancangan yang telah dianalisis, serta dapat digunakan untuk mengalokasikan penggunaan dana pribadi ke dalam kategori pos yang terdiri dari pemasukan, pengeluaran dan investasi/tabungan.
2. Perancangan aplikasi mobile pengelola keuangan pribadi telah dibuat sesuai dengan spesifikasi kebutuhan dan perancangan yang telah dianalisis, serta dapat digunakan sebagai pengingat (*reminder*) dan memo yang dapat mencatat informasi terkait proses transaksi keuangan dengan menggunakan *local storage* pada fitur PhoneGap.
3. Berdasarkan hasil pengujian validasi, unit dan kompatibilitas yang dilakukan, aplikasi dapat dinyatakan berjalan baik pada perangkat *mobile* dengan sistem operasi Android dan iOS dan telah memenuhi spesifikasi kebutuhan yang telah dianalisis tanpa perlu melakukan perubahan pada kode program.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan aplikasi *mobile* pengelola keuangan pribadi selanjutnya antara lain adalah:

1. Perlu dilakukan pengembangan lebih lanjut pada aplikasi dengan menambahkan *push notification* menggunakan *Google Cloud Messaging* pada fitur *reminder* sehingga fitur menjadi lebih *real-time* dan tidak perlu membuka aplikasi terlebih dahulu untuk menggunakan fitur *reminder*.
2. Untuk pengembangan lebih lanjut, dibutuhkan fitur setting database (*backup restore*) yang melalui proses enkripsi sehingga data lebih aman apabila terjadi sesuatu pada *device* seperti hilang atau perlu install ulang OS.

3. Dilakukan pengembangan untuk beberapa *mobile os* lain yaitu windows phone dan Symbian agar dapat menjangkau pengguna yang menggunakan perangkat selain Android dan iOS.

DAFTAR PUSTAKA

[WAR-10] Warsono. 2010, "Prinsip-prinsip dan praktik keuangan pribadi.", Journal of science, Volume 13 Nomor 2 Juli - Desember 2010.

[GT-12] Garg, Taruna. 2012, "Personal Financial Helath.", IJRMEC, Volume 2, Issue 11 (November- 2012).

[RR-12] Rodhiyah, Rodhiyah. 2012, "Manajemen Keuangan Keluarga Guna Menuju Keluarga Sejahtera." FORUM: Majalah Pengembangan Ilmu Sosial, 40 (1). hal. 28-33.

[NAT-00] National Department of Agriculture. 2000, "Financial Management", National Department of Agriculture. Pretoria.

[STA-14] Statcounter. 2014. StatCounter Global Stats. WWW [terhubung berkala]. http://gs.statcounter.com [diakses pada tanggal 10 April 2014].

[WS-14] W3schools. 2014. W3Schools Online Web Tutorials. WWW [terhubung berkala]. http://www.w3schools.com [diakses pada tanggal 10 April 2014].

[JS-10] Wilton, Paul dan McPeak, Jeremy. 2010, "Beginning Javascript.", 4th edition, Wiley Publishing, Inc., Canada.

[RY-12] Ghatol, Rohit dan Patel, Yogesh. 2012, "Beginning PhoneGap.", Apress., New York.

[JQM-14] JQueryMobile. 2014. JQuery mobile framework. WWW [terhubung berkala]. http://jquerymobile.com [diakses pada tanggal 10 April 2014].

[PGP-14] PhoneGap. 2014. PhoneGap framework documentation. WWW [terhubung berkala]. http://docs.phonegap.com/en/3.1.0 [diakses pada tanggal 10 April 2014].

[KRE-10] Kreibich, Jay A. 2010. "Using SQLite - Small, Fast, Reliable, Choose Any Three". O'Reilly.