

**Laporan Akhir
Sertifikasi**



Trisha Alexis Likorawung
0706022110037

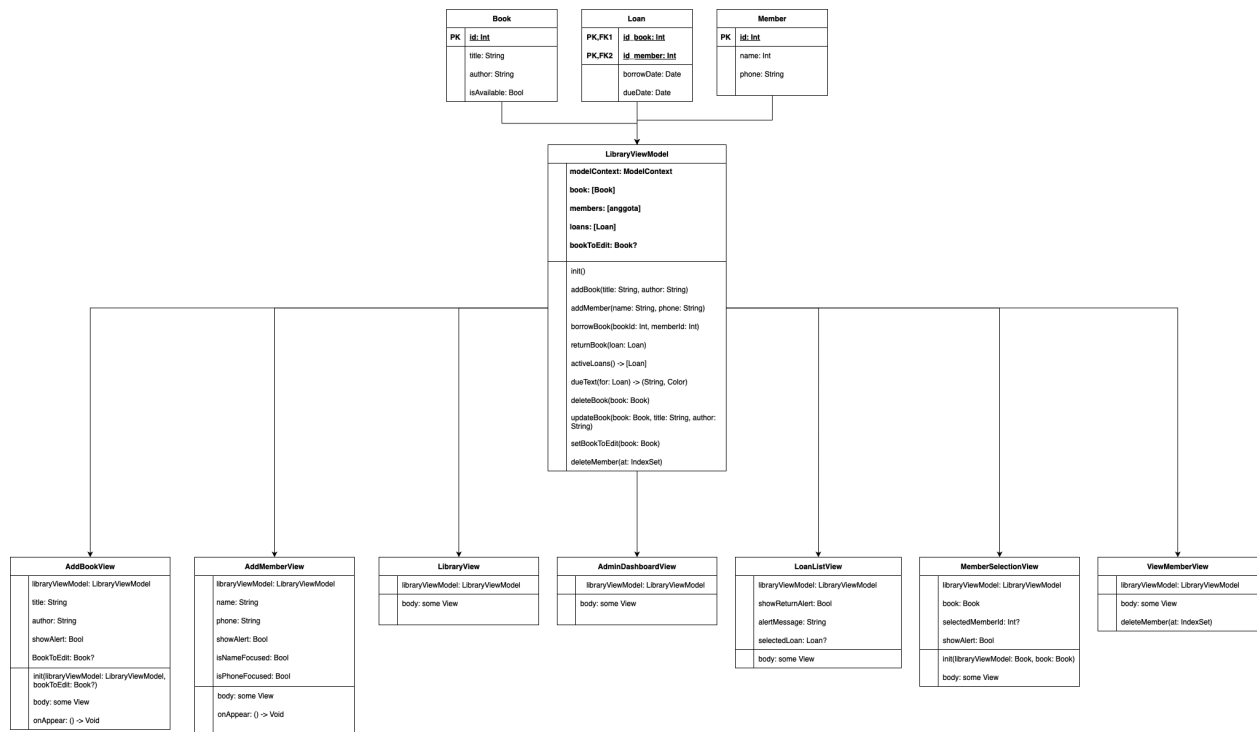
**Information Systems for Business
Universitas Ciputra Surabaya
2024/2025**

Daftar Isi

LAPORAN AKHIR.....	1
Daftar Isi.....	2
I. Design System.....	3
A. Class Diagram.....	3
B. ERD.....	4
C. Wireframe.....	5
II. Langkah Implementasi.....	6
III. Hasil Pengujian.....	7

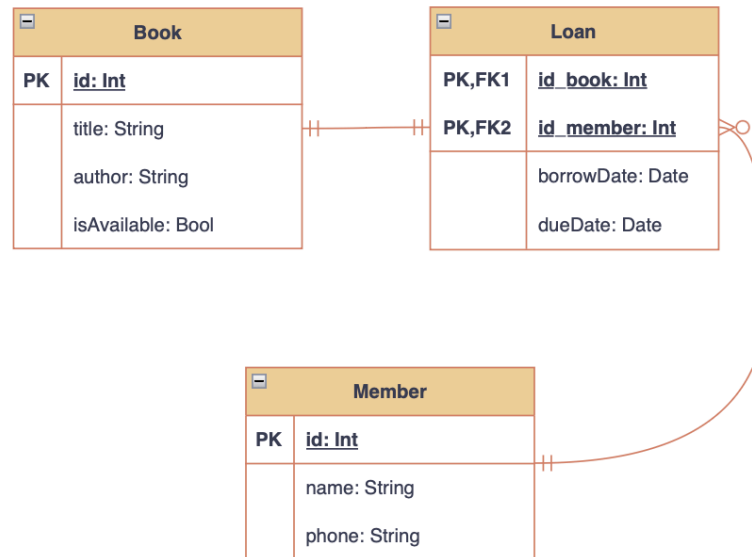
I. Design System

A. Class Diagram



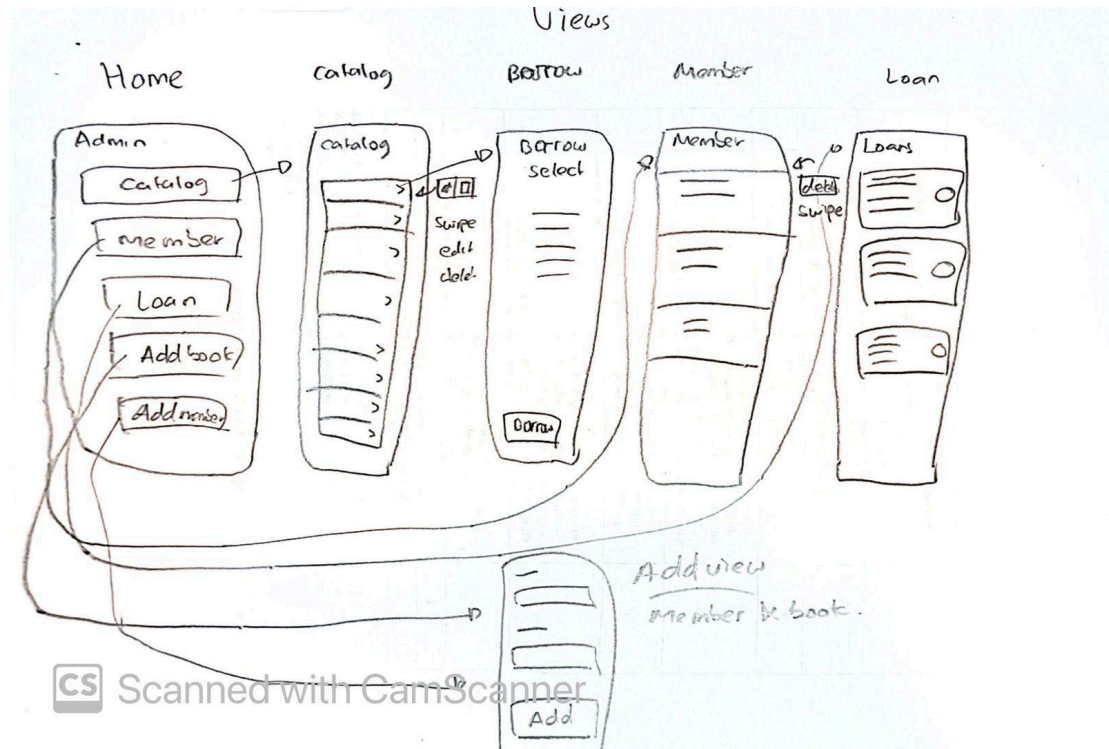
Gambar diatas merupakan hasil *Class Diagram* untuk objek yang ada pada aplikasi. *Class Diagram* di atas adalah hasil pembuatan *Class Diagram* untuk objek-objek yang tersedia dalam aplikasi. Diagram kelas ini menggambarkan struktur data dari aplikasi perpustakaan admin yakni aplikasi iOS. Saya menggunakan Swift Data. Kelas utama meliputi *Book*, *Member*, dan *Loan*. Selain itu, ada *View* yang mewakili data yang relevan dalam antarmuka pengguna seperti *LibraryView*, *LoanListView*, *ViewMemberView*, dan *AdminDashBoardView*. Ada pula antarmuka lain seperti *MemberSelectionView*, *AddMemberView*, dan *AddBookView*. Serta, ada *LibraryViewModel*.

B. ERD



Gambar diatas merupakan ERD diagram aplikasi dimana table *book* merupakan *one to one* ke tabel *loan* karena hanya ada satu buku per judul kalau buku dipinjamkan jadi tidak *available* lagi. Lalu hubungan dengan *member* dan *loan one to many* karena satu member bisa pinjam lebih dari satu buku (bukan buku yang sama karena adanya hanya satu buku per judulnya). Buku dibuat tabel sederhana tapi bisa ditambahkan isbn, tahun terbit, penerbit, gambar, dan lain-lain bila ingin lebih lengkap. Begitu pula dengan member bisa ditambahkan tanggal lahir, asal, alamat, dan sebagainya kalau mau lebih lengkap datanya.

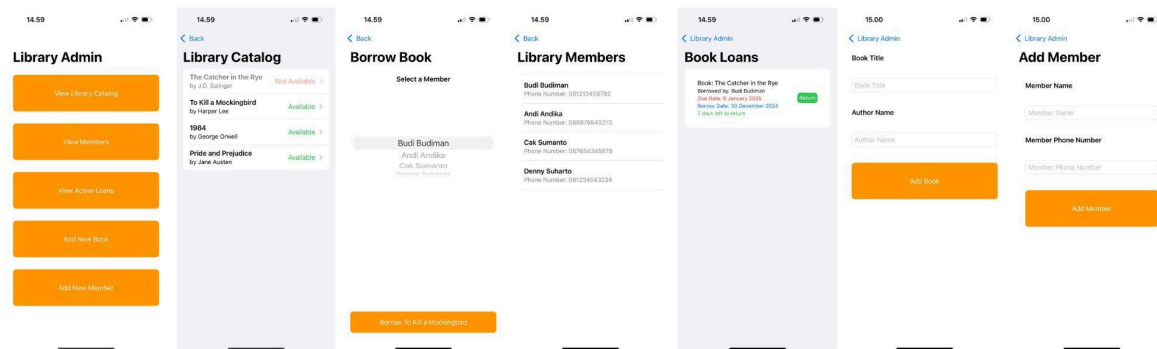
C. WireFrame



Gambar diatas merupakan hasil wireframing yang dibuat di kerta untuk merancang sketsa awal aplikasi berupa gambaran besar dari aplikasi yang akan dibuat. Terdapat halaman *home/admin dashboard* untuk menampilkan tombol-tombol yang dapat dilakukan. Terdapat katalog bisa lihat *list* judul buku, penulis dan *status* bisa dipinjam atau tidak. Kalau tidak maka akan tidak bisa dipinjam atau *not available*. Bisa edit buku, bisa hapus buku dengan di *swipe*. Kalau di tekan bisa ke *borrow book view* yang bisa pilih member mana yang akan pinjam dan tombol dipinjam maka akan berubah status dan *available* jadi *not available*.

Bisa lihat member isinya *list* nama dan nomor telepon bisa *delete member*. Ada view *loans* bisa lihat *active loans* isinya judul buku, dipinjam oleh nama member, tanggal harus dikembalikan, tanggal dipinjam, berapa hari lagi di kembalikan, ada tombol *return* jadi kalau sudah kembali jadi *available* lagi *statusnya* lalu akan hilang dari *active loans*. Lalu, ada juga *add book* dan *add member* jadi bisa menambahkan member dan menambahkan buku.

Preview Aplikasi



II. Langkah Implementasi

1. Membuat *wireframe*

Pertama-tama sebelum aplikasi dibuat saya membuatkan wireframing untuk merencanakan apa saja yang dibutuhkan di lembar kertas.

2. Membuat *design ERD*

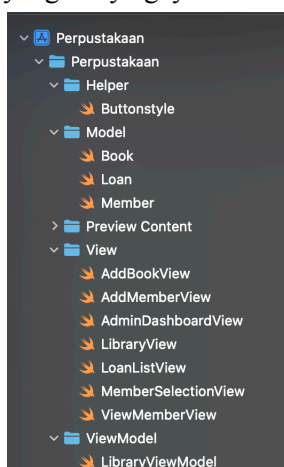
Setelah itu, saya membuat desain ERD untuk *database* aplikasi, yang kemudian diimplementasikan sebagai *swift data*.

3. Membuat *project* di *xcode* dan membuat *git* yang *public*

Saya lalu membuat projectnya di xcode menggunakan *swiftui* untuk tampilan, *swift data* untuk lokal database, dan bisa *add*, *edit* dan *delete* pakai *function-function*.

4. Membuat MVVM

Saya merapikan arsitektur dengan MVVM (model-view-viewmodel). Memisahkan jadi folder-folder, ada view yang isinya tampilan saja namanya diakhirinya dengan view seperti memberview, libraryview, dan lain-lain. Folder view model isinya yang ada function di masukkan disana. Kemudian databasenya di folder model ada *book*, *loan*, dan *member*. Ada pula helper yang isinya gaya tombol atau *button style*.



5. Membuat *error handling* dan *clean code*

Saya membuat clean code dan error handling untuk tahu bug dimana dan pakai print, else, dan sebagainya.

6. Membuat Class Diagram

Selanjutnya saya membuat *class diagram* yakni hubungan antar objek pada aplikasi

7. Pengujian User Testing

Terakhir, saya melakukan unit testing dengan user testing untuk tahu ada error apa dan hasilnya berjalan lancar saya menggunakan sepuluh test case yakni menambahkan member baru. Selain itu, testing dilakukan secara manual dan berjalan lancar.

III. Hasil Pengujian

Test Case	Steps	Ekspektasi	Hasil
User membuat member baru	<ol style="list-style-type: none"> 1. Pilih tombol add member 2. Ketikkan nama "Trisha" di textfield name 3. Ketikkan nomor telepon "081325634789" 4. Tekan add member 	Akan melakukan semua perintah dan muncul alert member added successfully	Pass (aplikasi juga melakukan serangkaian percobaan seperti dengan orientasi landscape, darkmode, dsb)
User mengembalikan buku	<ol style="list-style-type: none"> 1. Pilih tombol view active loans 2. Di buku the catcher in the rye tekan tombol return 3. Muncul alert tekan yes maka buku akan dikembalikan 	Akan melakukan semua perintah dan terakhir buku hilang dari book loans	Pass
User melihat pinjaman aktif	<ol style="list-style-type: none"> 1. Pilih tombol view active loans 	Akan melakukan perintah dan bila lihat pinjaman aktif	Pass
User melihat member	<ol style="list-style-type: none"> 1. Pilih tombol view members 	Akan melakukan perintah bisa lihat member	Pass
User melihat katalog perpustakaan	<ol style="list-style-type: none"> 1. Pilih tombol view library catalog 	Akan melakukan perintah dan bisa lihat katalog	Pass
User meminjam buku	<ol style="list-style-type: none"> 1. Pilih tombol view library catalog 2. Pilih buku 1984 tekan 3. Akan ke halaman borrow book select a member pilih nama 	Akan melakukan semua perintah dan terakhir buku dari available jadi not available statusnya	Pass

	budi budiman 4. Tekan tombol borrow 1984 5. Muncul alert tekan borrow		
User membuat buku baru	1. Pilih tombol view add new book 2. Ketikkan "Harry Potter" di textfield title 3. Ketikkan author name "JK Rowling" 4. Tekan add book	Akan melakukan semua perintah dan muncul alert book added successfully	Pass
User mengedit buku di katalog	1. Pilih tombol view library catalog 2. Geser ke kiri buku To Kill a Mockingbird 3. Tekan tombol edit 4. Ganti title jadi "To Kill" dan author jadi "Harper" 5. Tekan tombol save changes	Akan melakukan semua perintah dan judul dan author di katalog berubah sesuai yang diubah	Pass
User menghapus member	1. Pilih tombol view member 2. Geser ke kiri member bernama Budi Budiman 3. Tekan tombol delete	Akan menghapus Budi Budiman dari list nama member	Pass
User menghapus buku dari katalog	1. Pilih tombol view library catalog 2. Geser ke kiri buku The Catcher in the Rye 3. Tekan tombol delete	Akan menghapus buku The Catcher in the Rye	Pass


```

import XCTest

final class PerpustakaanUITests: XCTestCase {
|
    override class var runsForEachTargetApplicationUIConfiguration: Bool {
        true
    }

    override func setUpWithError() throws {
        continueAfterFailure = false
    }

    @MainActor
    func testAddMemberForm() throws {
        let app = XGUIApplication()
        app.launch()

        let launchScreenAttachment = XCTAttachment(screenshot: app.screenshot())
        launchScreenAttachment.name = "Launch Screen"
        launchScreenAttachment.lifetime = .keepAlways
        add(launchScreenAttachment)

        let addNewMemberButton = app.buttons["Add New Member"]
        XCTAssertTrue(addNewMemberButton.exists, "The 'Add New Member' button should exist")
        addNewMemberButton.tap()

        let addMemberTitle = app.staticTexts["Add Member"]
        XCTAssertTrue(addMemberTitle.exists, "The AddMemberView screen should be displayed after tapping 'Add New Member'")

        let nameTextField = app.textFields["Member Name"]
        XCTAssertTrue(nameTextField.exists, "The 'Member Name' text field should exist")
        nameTextField.tap()
        nameTextField.typeText("Trisha")

        let phoneTextField = app.textFields["Member Phone"]
        XCTAssertTrue(phoneTextField.exists, "The 'Member Phone' text field should exist")
        phoneTextField.tap()
        phoneTextField.typeText("081325634789")

        let addMemberButton = app.buttons["Add Member"]
        XCTAssertTrue(addMemberButton.exists, "The 'Add Member' button should exist")
        addMemberButton.tap()

        let successAlert = app.alerts["Member Added"]
        XCTAssertTrue(successAlert.exists, "The success alert should appear after adding a member")
        successAlert.buttons["OK"].tap()

        let addMemberFormScreenshot = XCTAttachment(screenshot: app.screenshot())
        addMemberFormScreenshot.name = "Add Member Form After Success"
        addMemberFormScreenshot.lifetime = .keepAlways
        add(addMemberFormScreenshot)
    }
}

```