REPORT SOFTWARE ENGINEERING
KELOMPOK 11: PET STOP

PET STOP

Link download : https://drive.google.com/drive/folders/1hrvV0D6aOoql6-6zEGqoR4yrC9pLTUS0?usp=sharing

Kelompok 11 :

FATIH ACHMAD AL-HARITZ - 2602097132
GRIVENTH GRIFFITH AGUSTIN - 2602103021
ARIFIAN ZAKI AZ ZUHDI - 2602113451
AHMAD AFIF DHOVIEN PRABASWARA - 2602097933
FINDA ANGELICA GERALDINE DE WANNA - 2602118521

BAB 1: Introduction

1.1 Deskripsi

Pet Stop merupakan aplikasi untuk kalian pecinta binatang dan yang memiliki hewan peliharaan.

Aplikasi ini menyediakan fitur-fitur yang memudahkan kalian dalam dunia pemeliharaan binatang seperti *Pet Groomer,* konsultasi ke dokter hewan berlisensi, penitipan hewan peliharaan di pet shop terdekat, dll.

Kami juga menyediakan fitur Komunitas untuk kalian yang baru berkecimpung di dunia pemeliharaan hewan, kalian bisa bertanya dan berbincang-bincang bersama orang banyak untuk *sharing* ilmu.

1.2 Permasalahan

1. Pemilik peliharaan biasanya tidak ada waktu untuk membawa peliharaannya ke dokter hewan, karena faktor jarak maupun sibuk karir.
2. Karena faktor jarak dan sibuk karir, maka untuk konsultasi dokter saja tidak ada waktu. Dan untuk mendapatkan dokter hewan terpercaya diluar rumah sakit hewan lumayan sulit.
3. Untuk pemula yang baru memelihara hewan kadang bingung bertanya kepada siapa (apalagi tidak ada teman yang memelihara hewan) masalah peliharaan seperti cara merawat dll.
4. Saat ingin berpergian dan tidak membawa peliharaan kadang lumayan memakan waktu untuk cari pet shop yang buka dan menerima penitipan hewan.

1.3 Ruang Lingkup

Aplikasi ini akan fokus pada pengembangan sistem informasi penjualan berbasis Mobile untuk pet shop yang berfokus pada layanan pet groomer, penitipan, dan penyedia dokter hewan. Berikut adalah aspek-aspek yang termasuk dalam ruang lingkup laporan:

1. Aplikasi Penjualan Berbasis APP dengan Firebase: Laporan akan membahas pengembangan aplikasi penjualan yang dapat diakses melalui web dengan menggunakan Firebase sebagai platform backend. Fitur-fitur yang terkait dengan transaksi penjualan, inventaris, dan pelaporan akan menjadi bagian dari ruang lingkup ini.

2. Database Firebase: Firebase adalah layanan database cloud yang menyimpan dan mengakses data secara real-time. Aplikasi akan menggunakan Firebase sebagai database untuk menyimpan data pelanggan, produk, dan transaksi penjualan. Pengelolaan data melalui Firebase akan menjadi bagian penting dalam laporan.

3. Layanan Pet Groomer, Penitipan, dan Penyedia Dokter Hewan: Laporan akan membahas proses dan fitur-fitur yang terkait dengan layanan pet groomer (perawatan hewan), penitipan hewan, dan penyedia dokter hewan. Ini mencakup manajemen jadwal, pemesanan layanan, dan komunikasi antara pemilik hewan dan penyedia layanan.

4. Batasan Masalah: Laporan tidak akan membahas aspek lain yang tidak relevan dengan layanan yang disebutkan di atas.

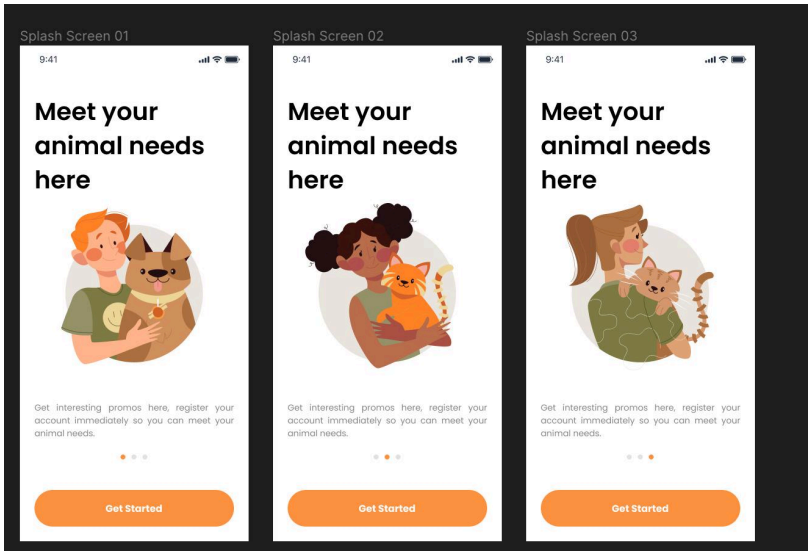
1.4 Solusi

Berdasarkan masalah yang dipaparkan, aplikasi kami menawarkan konsultasi virtual dengan dokter hewan terpercaya, sehingga pemilik hewan dapat berkonsultasi kapan saja dan dari mana saja tanpa terhalang oleh jarak atau kesibukan karir. Selain itu, aplikasi dapat menyertakan fitur direktori dokter hewan yang memudahkan pencarian dokter hewan terdekat yang telah diverifikasi keahliannya. Untuk pemilik hewan pemula, aplikasi bisa menyediakan forum tanya jawab di mana mereka dapat bertanya dan mendapatkan tips perawatan dari komunitas pemilik hewan peliharaan lainnya. Terakhir, aplikasi tersebut dapat memuat informasi tentang pet shop dan pusat penitipan hewan yang terpercaya dan tersedia untuk menerima penitipan hewan, lengkap dengan ulasan dan rating dari pengguna lain, memudahkan pemilik hewan saat perlu bepergian. Dengan solusi ini, pemilik hewan peliharaan dapat mengatasi kendala yang mereka hadapi dengan lebih efisien dan efektif.

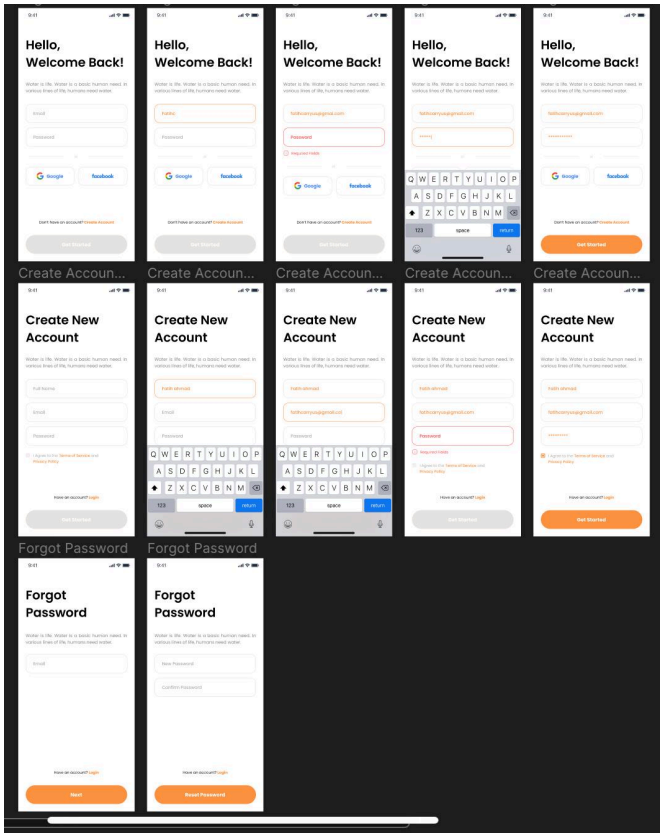
BAB 2: Design

2.1 Prototype

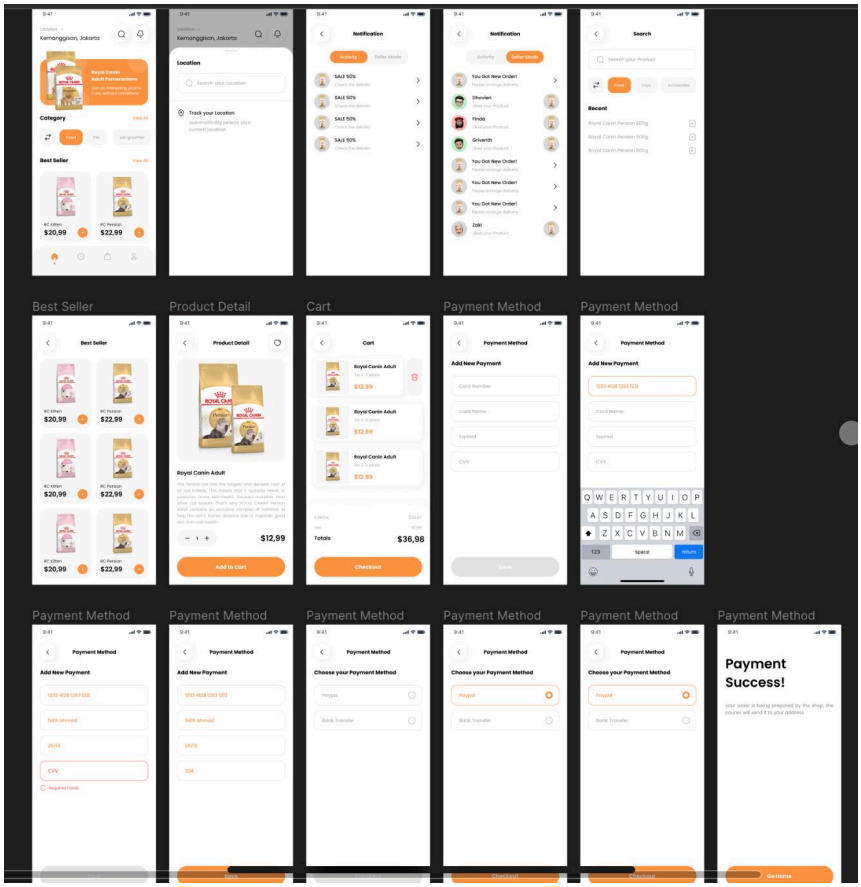
1. Splash screen



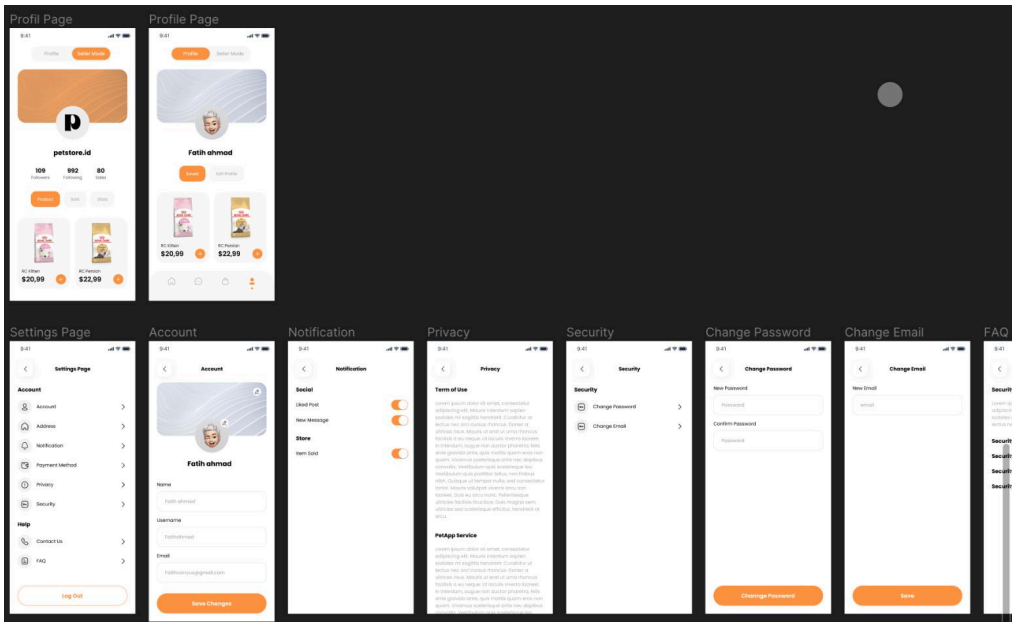
2. Login, Register, and Forgot Password



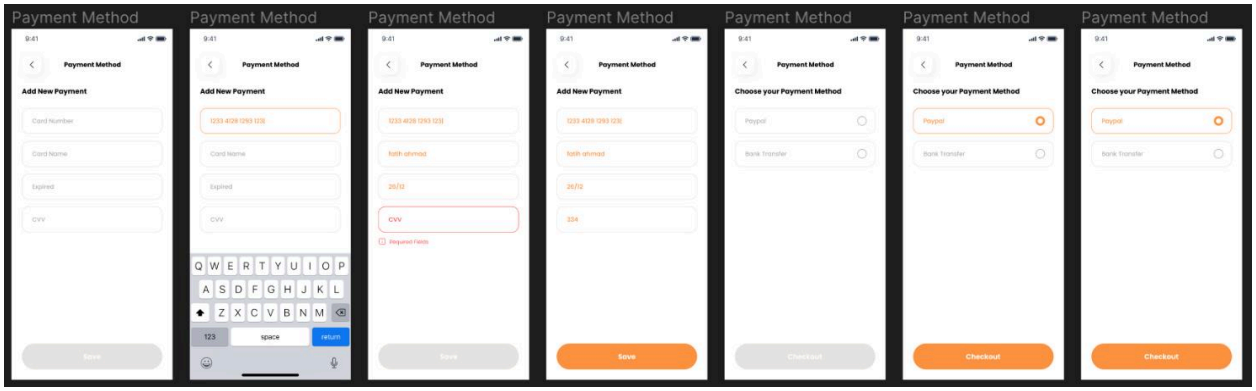
3. Home Page, Filter, and Payment



4. Profile and Account Details



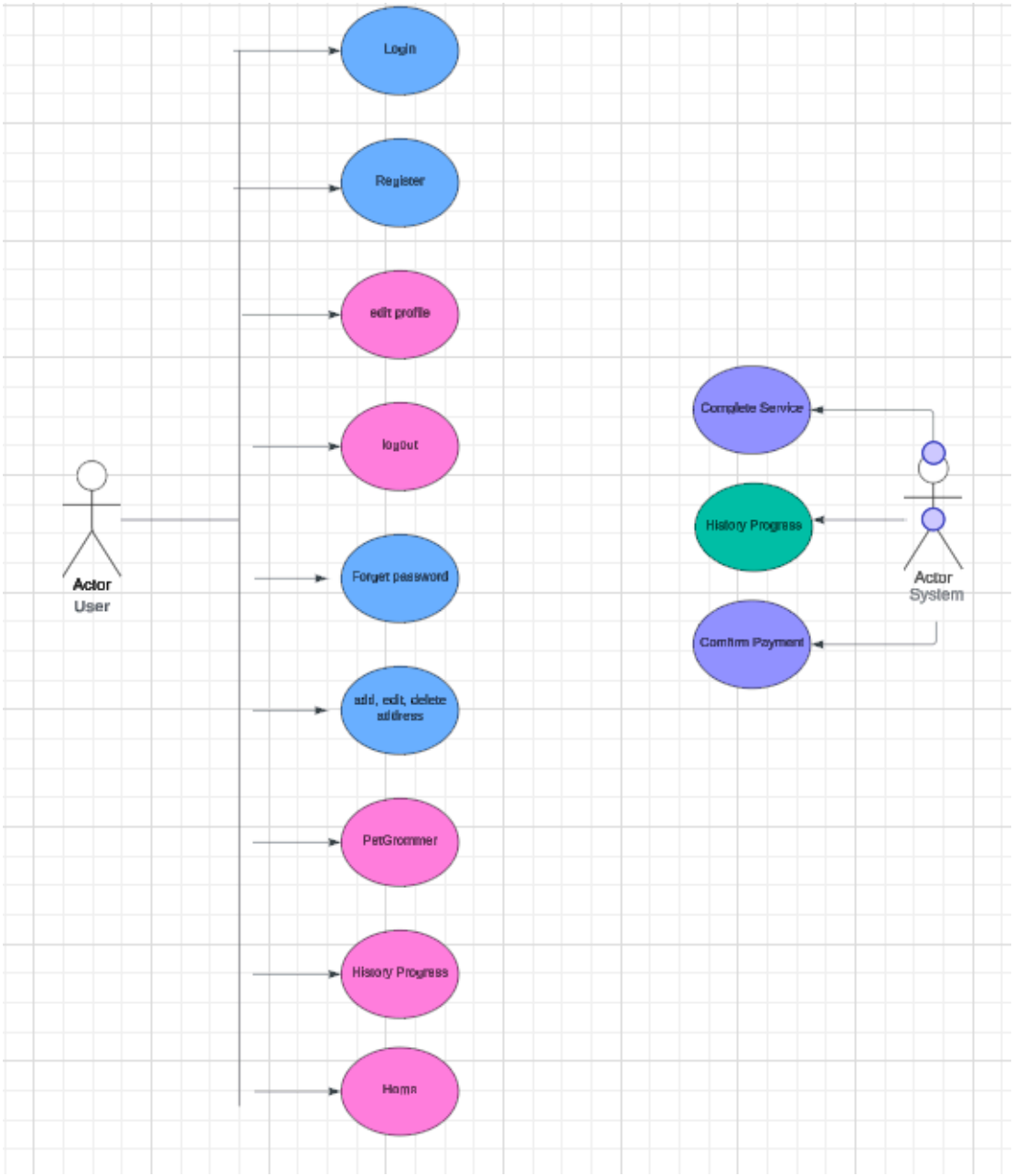
5. Payment Method



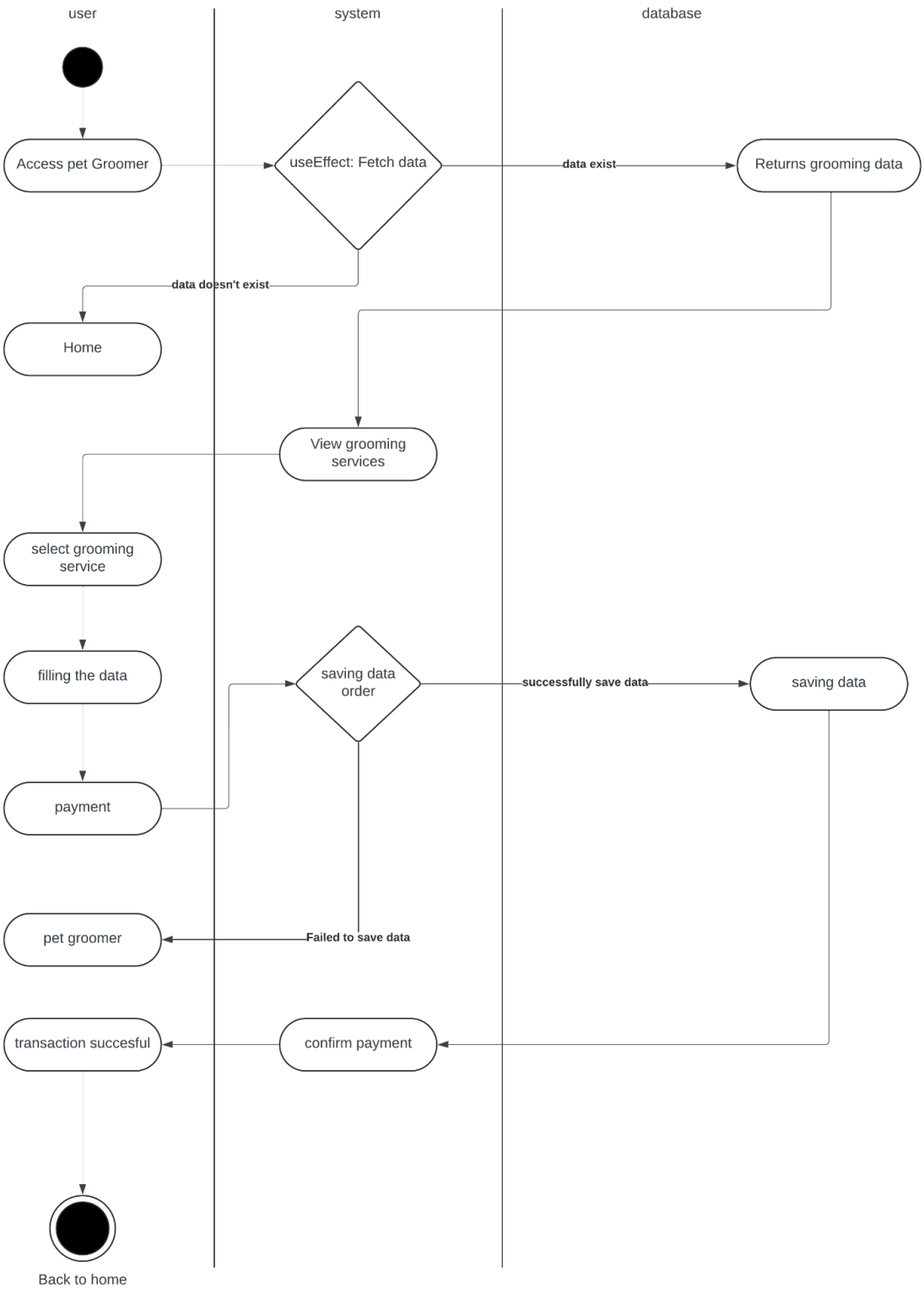
*Llink Prototype:
https://www.figma.com/file/nF5HXnXSo0gGLWsBRaOCJi/Pet-App?type=design&node-id=13%3A5238&mode=design&t=PDZN2bJTwtzE2yqo-1

2.2 UML Design (Usecase, Activity Diagram, CRC Card, Class Diagram)

- **Usecase Diagram**



- **Activity Diagram**



- **CRC Card**

PetGroomer

User Display	(Proses Bisnis)
Responsibility	Collaborator
• Display pet groomer data (features, description, price) • Select service • Upload payment proof • Confirm payment	• Features description

Display Database	(Database)
Responsibility	Collaborator
• Giving Display (features, description, price) • Validate payment	• PetGroomer Database • Payment Database

Login

User Login	(Proses Bisnis)
Responsibility	Collaborator
• Cek email • Cek password • Create session	• User Login feature

User Register Model	(Database)
Responsibility	Collaborator
• Cek format input • Validate email & password	• User Database

Register

User Controller	(Proses Bisnis)
Responsibility	Collaborator

User Register Model	(Database)
Responsibility	Collaborator

User Controller	(Proses Bisnis)
• Input user data to profile/database • Save registered user	• User

User Register Model	(Database)
• Check email format • Check registration format	• User Database

Edit Profile

Profile	(Proses Bisnis)
Responsibility	Collaborator
• Input user data to profile / database • Upload profile pict	• Update profile feature

Profile Database	(Database)
Responsibility	Collaborator
• Current user data • Cek update format	• User Database

Logout

User Logout	(Proses Bisnis)
Responsibility	Collaborator
• Logout current user	• Session Management

User Register Model	(Database)
Responsibility	Collaborator
• Logout user data • Stopping session	• Session end

Forget Password

Forget Password	(Proses Bisnis)
Responsibility	Collaborator
• Current user • New password	• Add new password

User Register Model	(Database)
Responsibility	Collaborator
• Validate user	• User Database

Add, Edit, Delete, Address

User Account	(Proses Bisnis)
Responsibility	Collaborator
• Add new address • Edit existing address • Delete existing address	• Update address

Account Database	(Database)
Responsibility	Collaborator
• Cek format input • Validate new address	• User Database

History

Transaction	(Proses Bisnis)
Responsibility	Collaborator
• Current user • Cek transaction history	• History transaction feature

Account Database	(Database)
Responsibility	Collaborator
• Giving history transaction display	• Transaction Database

Progress

Transaction	(Proses Bisnis)
Responsibility	Collaborator
• Current user • Do ongoing transactions	• Transaction feature

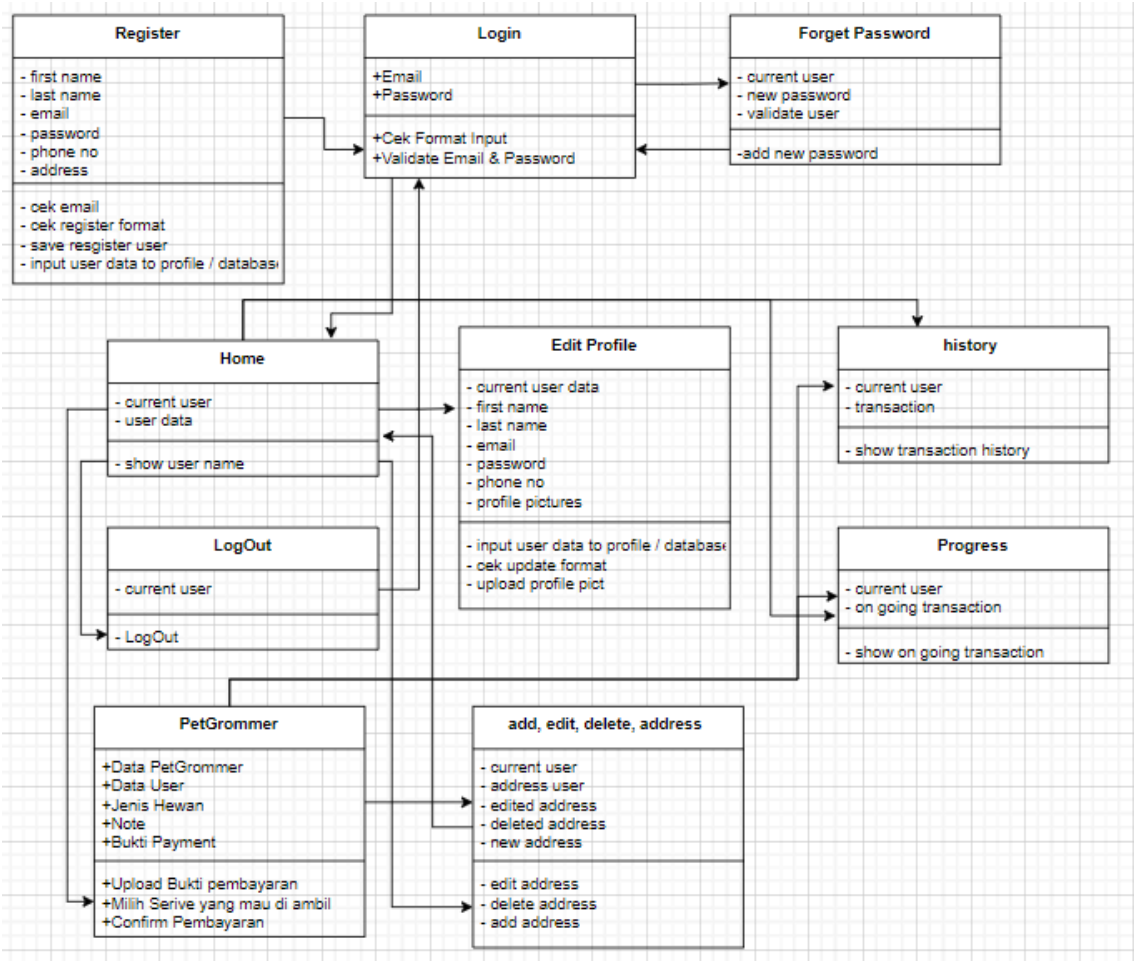
Account Database	(Database)
Responsibility	Collaborator
• Cek format input • Validate transaction	• Transaction Database

Home

User Filter	(Proses Bisnis)
Responsibility	Collaborator
• Current user • Display homepage	• Search feature • Notification feature • Tracking location feature

Home Database	(Database)
Responsibility	Collaborator
• Show best seller • Track location • Show notification (Activity & Seller Mode) • Show Search	• User Database

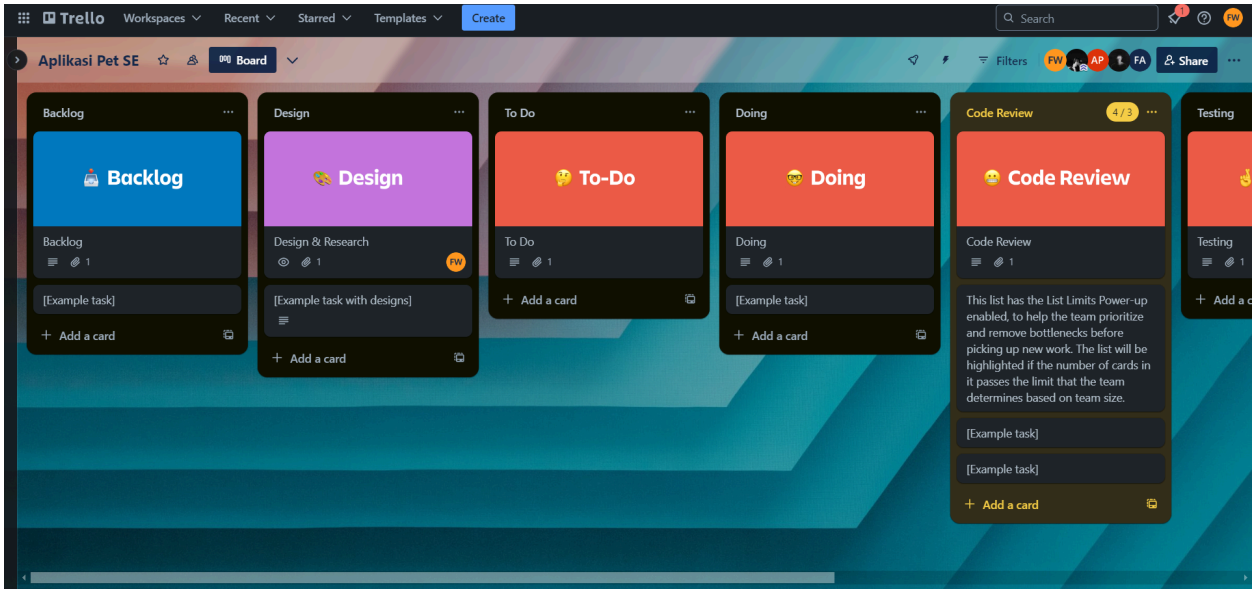
- Class Diagram

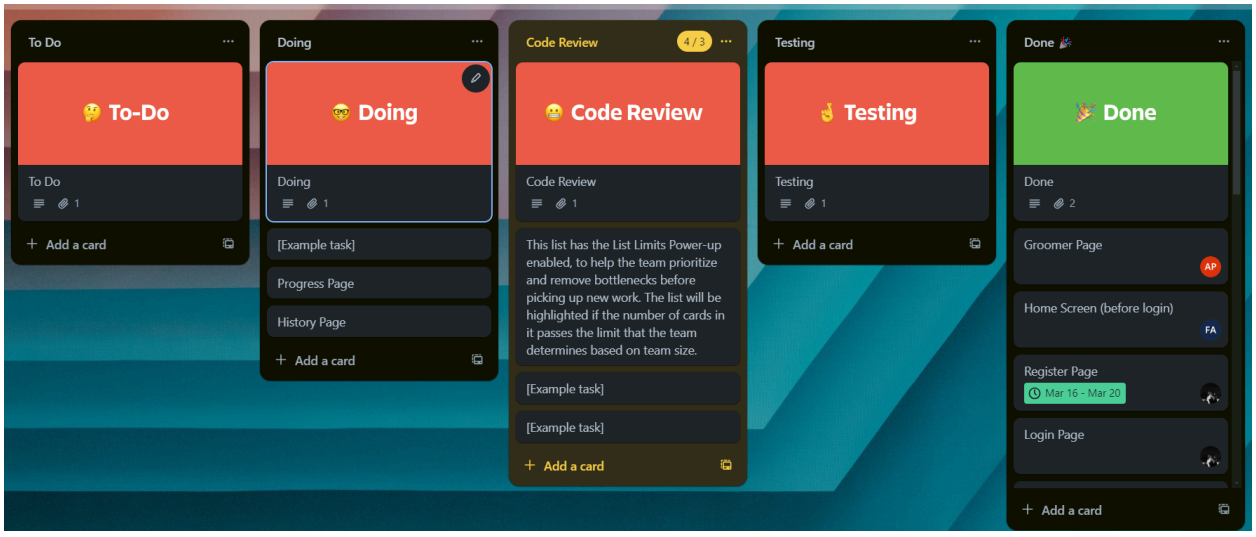


BAB 3: Development Activity

3.1 Screenshot Trello

Link Trello:
https://trello.com/invite/b/65f7b876a37c8b29c1fc3010/ATTIa2c458d6972160a1bdf6aa37b25ead662786F792/Aplikasi%20Pet%20SE





3.2 GitHub Link

Berikut merupakan link GitHub se;ama pengerjaan:
https://github.com/fatihalharitz/AOL-SE/blob/main/README.md

Berikut adalah link download app yang sudah jadi :
https://drive.google.com/drive/folders/1hrvV0D6aOoql6-6zEGqoR4yrC9pLTUS0?usp=sharing

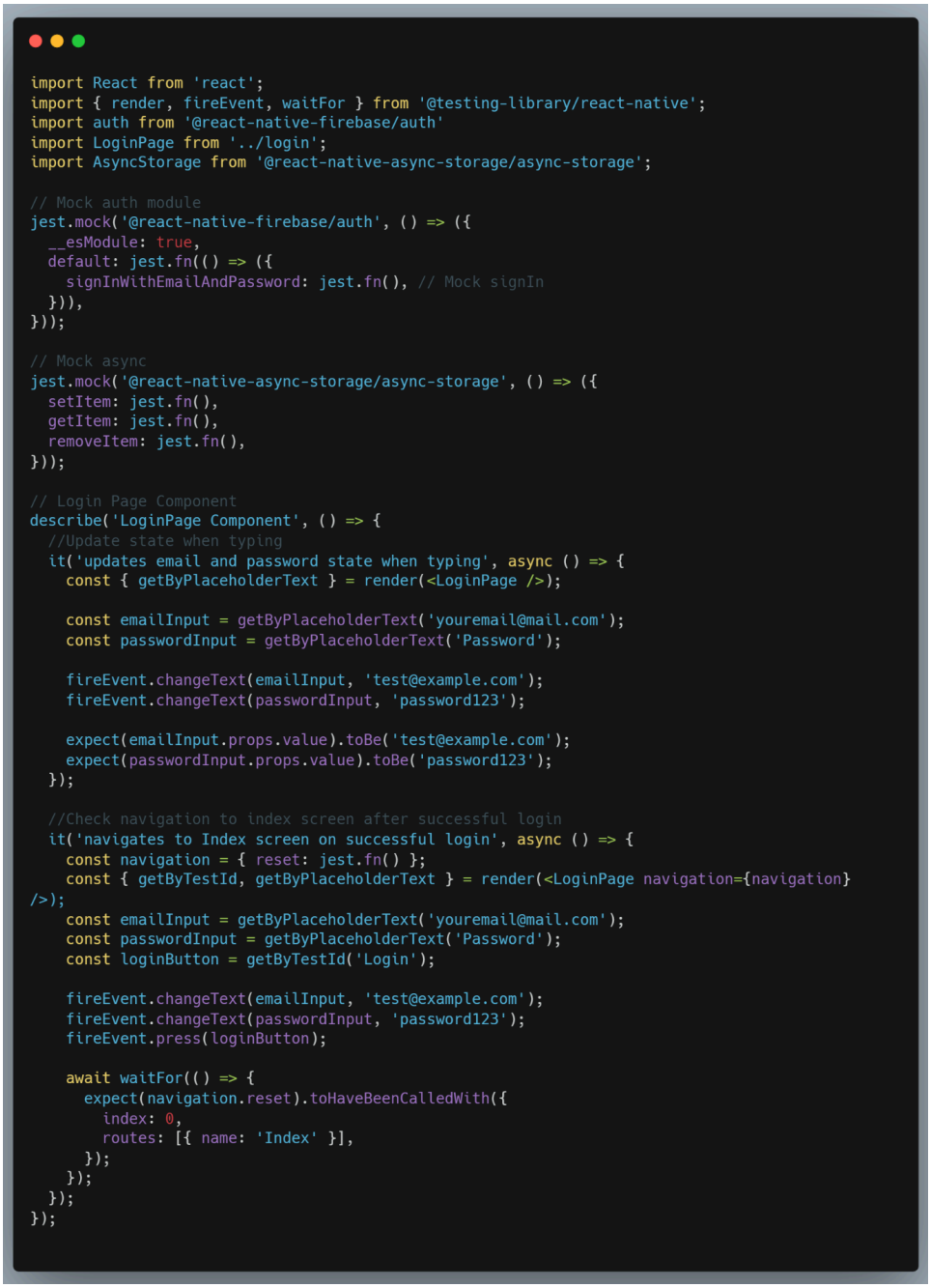
3.3 **Perhitungan Cost Development** (Salary & Server Infrastructure)

Information Domain Value	Count	Simple	Average	Complex	Total
External Inputs	23	3	4	6	69
External Outputs	17	4	5	7	68
External Inquiries	16	3	4	6	48
Internal Logical Files	7	7	10	15	49
External Interface Files	7	5	7	10	35
				Count Total	269

Total	269
FP-e	309,35
Productivity	5
Cost Total	Rp 1.345.000.000,00

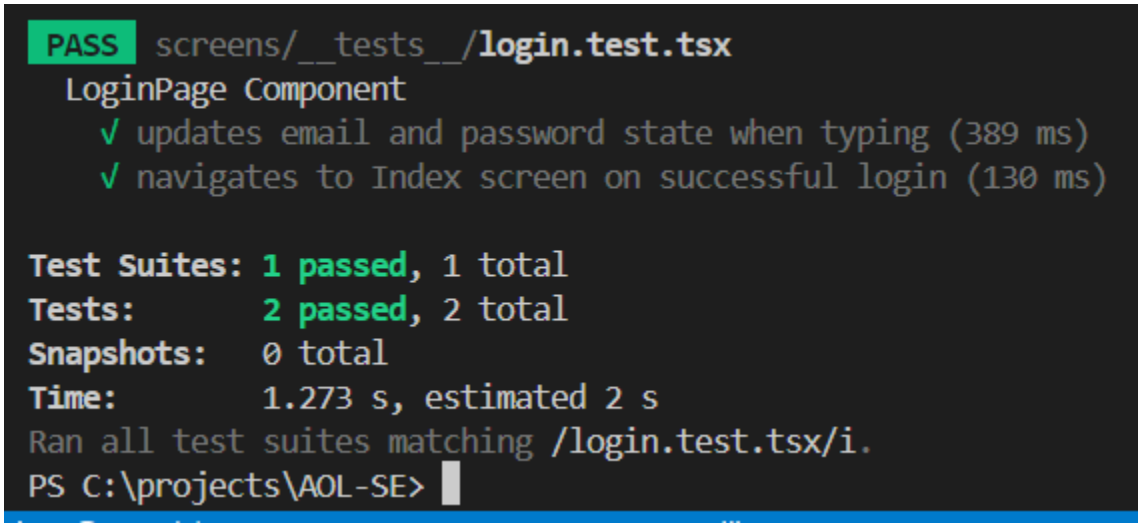
3.4 Unit Testing

3.4.1 Login Unit Testing (Backend)



```
import React from 'react';
import { render, fireEvent, waitFor } from '@testing-library/react-native';
import auth from '@react-native-firebase/auth'
import LoginPage from '../login';
import AsyncStorage from '@react-native-async-storage/async-storage';

// Mock auth module
jest.mock('@react-native-firebase/auth', () => ({
  __esModule: true,
  default: jest.fn(() => ({
    signInWithEmailAndPassword: jest.fn(), // Mock signIn
  })),
}));

// Mock async
jest.mock('@react-native-async-storage/async-storage', () => ({
  setItem: jest.fn(),
  getItem: jest.fn(),
  removeItem: jest.fn(),
}));

// Login Page Component
describe('LoginPage Component', () => {
  //Update state when typing
  it('updates email and password state when typing', async () => {
    const { getByPlaceholderText } = render(<LoginPage />);

    const emailInput = getByPlaceholderText('youremail@mail.com');
    const passwordInput = getByPlaceholderText('Password');

    fireEvent.changeText(emailInput, 'test@example.com');
    fireEvent.changeText(passwordInput, 'password123');

    expect(emailInput.props.value).toBe('test@example.com');
    expect(passwordInput.props.value).toBe('password123');
  });

  //Check navigation to index screen after successful login
  it('navigates to Index screen on successful login', async () => {
    const navigation = { reset: jest.fn() };
    const { getByTestId, getByPlaceholderText } = render(<LoginPage navigation={navigation}
/>);
    const emailInput = getByPlaceholderText('youremail@mail.com');
    const passwordInput = getByPlaceholderText('Password');
    const loginButton = getByTestId('Login');

    fireEvent.changeText(emailInput, 'test@example.com');
    fireEvent.changeText(passwordInput, 'password123');
    fireEvent.press(loginButton);

    await waitFor(() => {
      expect(navigation.reset).toHaveBeenCalledWith({
        index: 0,
        routes: [{ name: 'Index' }],
      });
    });
  });
});
```

Result :


PASS screens/__tests__/login.test.tsx
LoginPage Component
√ updates email and password state when typing (389 ms)
√ navigates to Index screen on successful login (130 ms)
Test Suites: 1 passed, 1 total
Tests: 2 passed, 2 total
Snapshots: 0 total
Time: 1.273 s, estimated 2 s
Ran all test suites matching /login.test.tsx/i.
PS C:\projects\AOL-SE>

3.4.2 Login Testing (Frontend)

```
import React from 'react';
import { render, fireEvent, waitFor } from '@testing-library/react-native';
import auth from '@react-native-firebase/auth'
import LoginPage from '../login';
import AsyncStorage from '@react-native-async-storage/async-storage';

export default () => ({
  signInWithEmailAndPassword: jest.fn(() => Promise.resolve({ user: { uid: 'test-uid' } })),
});

jest.mock('@react-native-firebase/auth', () => ({
  __esModule: true,
  default: jest.fn(() => ({
    signInWithEmailAndPassword: jest.fn(), // Mock the signInWithEmailAndPassword method
  })),
}));

jest.mock('@react-native-async-storage/async-storage', () => ({
  setItem: jest.fn(),
  getItem: jest.fn(),
  removeItem: jest.fn(),
}));

describe('Login Component', () => {
  it('should render email and password inputs and login button', () => {
    const { getByTestId } = render(<LoginPage />);

    expect(getByTestId('emailInput')).toBeTruthy();
    expect(getByTestId('passwordInput')).toBeTruthy();
    expect(getByTestId('Login')).toBeTruthy();
  })
})
```

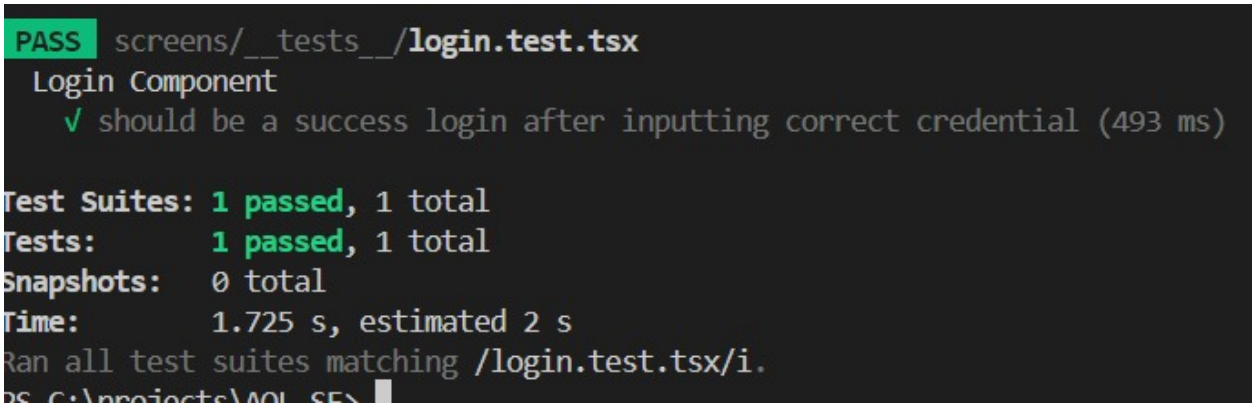
Result :

```
PASS screens/__tests__/login.test.tsx
  Login Component
    √ should render email and password inputs and login button (389 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.675 s, estimated 2 s
Ran all test suites matching /login.test.tsx/i.
PS C:\projects\AOL-SE>
```

3.4.3 verify credential(frontend)

```
import React from 'react';
import { render, fireEvent, waitFor } from '@testing-library/react-native';
import auth from '@react-native-firebase/auth'
import LoginPage from '../login';
import AsyncStorage from '@react-native-async-storage/async-storage';

export default () => ({
  signInWithEmailAndPassword: jest.fn(() => Promise.resolve({ user: { uid: 'test-uid' } })),
});

jest.mock('@react-native-firebase/auth', () => ({
  __esModule: true,
  default: jest.fn(() => ({
    signInWithEmailAndPassword: jest.fn(), // Mock the signInWithEmailAndPassword method
  })),
}));

jest.mock('@react-native-async-storage/async-storage', () => ({
  setItem: jest.fn(),
  getItem: jest.fn(),
  removeItem: jest.fn(),
}));

describe('Login Component', () => {
    it('should be a success login after inputting correct credential', async () => {
          const { getByTestId, getByPlaceholderText , getByText} = render(<LoginPage  />);

          const emailInput = getByPlaceholderText('youremail@mail.com');
          const passwordInput = getByPlaceholderText('Password');
          const loginButton = getByTestId('Login');

          fireEvent.changeText(emailInput, 'test@example.com');
          fireEvent.changeText(passwordInput, 'password123');
          fireEvent.press(loginButton);

          await waitFor(() => {
            expect(getByText('Login Successful')).toBeTruthy();
          });
        });
  });
```

Result :


PASS screens/__tests__/login.test.tsx
Login Component
√ should be a success login after inputting correct credential (493 ms)
Test Suites: 1 passed, 1 total
Tests: 1 passed, 1 total
Snapshots: 0 total
Time: 1.725 s, estimated 2 s
Ran all test suites matching /login.test.tsx/i.