LabLink Documentation

A. Project Planning

1. Project Proposal

Overview

LabLink aims to streamline the process of medical test booking by connecting patients with accredited laboratories through a single mobile platform. The application enables patients to browse available labs, select branches, choose specific tests or upload prescriptions, and schedule appointments easily. It also provides laboratories and administrators with tools for managing tests, branches, bookings, and performance analytics, improving the overall patient experience.

Objectives

- Simplify the process of finding and booking laboratory tests for patients.
- Enable labs to efficiently manage test offerings, branches, and appointments.
- Provide a clear and secure platform for uploading and viewing patient test results.
- Allow patients to rate and review laboratories based on their experience.
- Equip lab administrators and the super admin with dashboards and reports for better operational insights.
- Ensure data protection, authentication, and smooth performance across all user roles.

Scope

The current version of LabLink supports three main user roles:

Patients: Can register and log in, explore available laboratories and branches, select desired tests or upload a prescription, schedule appointments, view booking history, access uploaded test results, and rate or review labs. Lab details such as contact information are provided to facilitate external communication if needed.

Lab Admins: Their accounts are created by the Super Admin. Lab Admins can add and manage branches, define available tests and their prices, review and respond to booking requests, upload test results for patients, view patient ratings and reviews,

and access statistical dashboards and reports for performance tracking.

Super Admin: Has the highest level of control in the system. The Super Admin can manage laboratories by creating, editing, or removing lab accounts, overseeing their performance, and monitoring the system's overall activity through comprehensive dashboards and reports.

Planned future enhancements:

Include the integration of a notification system, as well as in-app payment and chat functionalities.

2. Task Assignments and Roles

The LabLink development team consists of three Flutter developers, with one serving as the Team Leader.

Team Leader – *Habiba Mahmoud*

- Designed the user interface and experience using Figma.
- Collected and organized project documentation and coordinated team efforts.
- Contributed equally to Flutter development and implementation tasks.

Developers – *Habiba Mahmoud, Yomna Mohamed, Abdulrahman Mahmoud*

- Implemented assigned screens and functionalities using Flutter and Firebase.
- Collaborated on creating system diagrams (use case, sequence, and class diagrams).
- Participated in testing, debugging, and feature integration to ensure stable operation.

All members shared equal responsibility for development, testing, and code integration. Collaboration and version control were maintained through shared repositories and regular weekly meetings to monitor progress and resolve challenges.

3. Risk Assessment and Mitigation Plan

Risk Type	Description	Mitigation Strategy
Technical Risk	Potential Firebase integration or data synchronization issues.	Conduct regular testing, enable offline persistence, and apply

		robust error handling and validation.
Schedule Risk	Delays due to overlapping tasks or feature changes during development.	Use project tracking tools, set realistic milestones, and maintain continuous team communication.
Security and Privacy Risk	Unauthorized access or exposure of sensitive user data.	Apply Firebase Authentication and Firestore Security Rules.

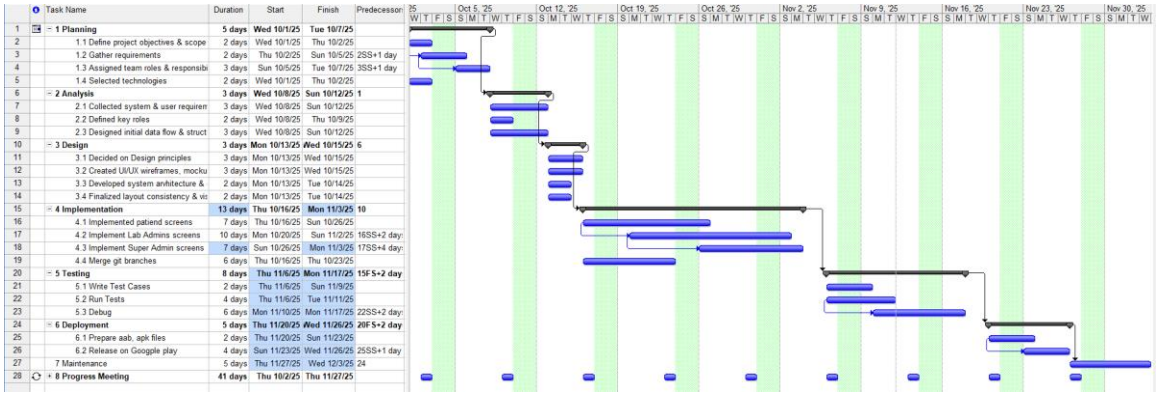
4. Project Timeline

The development of LabLink followed the Software Development Life Cycle (SDLC) methodology over a period of two months, divided into six main phases: Planning, Analysis, Design, Development, Testing, and Future Deployment & Maintenance. Weekly Thursday meetings were held to review progress, identify challenges, and plan next steps for the upcoming phase.

SDLC Phase	Duration	Key Activities	Milestones & Meetings
Planning Phase	Week 1	- Defined project objectives and scope. - Assigned team roles and responsibilities. - Selected technologies (Flutter and Firebase).	Week 1 Meeting: Confirmed project goals and development plan.
Analysis Phase	Week 2	- Collected system and user requirements. - Defined key roles: Patient, Lab Admin, and Super Admin. - Designed initial data flow and structure.	Week 2 Meeting: Reviewed requirements and finalized database design.
Design Phase	Week 3	- Created UI/UX mockups using Figma. - Developed system architecture and diagrams. - Finalized layout consistency and visual guidelines.	Week 3 Meeting: Approved UI mockups and system design.

Development Phase	Week 4–7	- Implemented authentication, booking management, and dashboard functionalities. - Integrated Firebase Firestore and Storage. - Conducted internal code reviews.	Weekly Meetings (Weeks 4–7): Showcased progress and addressed implementation challenges.
Testing Phase	Week 8	- Conducted unit and user acceptance testing. - Validated form inputs, role-based access, and data synchronization. - Fixed identified issues and optimized performance.	Week 8 Meeting: Reviewed test results and marked project as ready for presentation.
Future Deployment & Maintenance (Planned)	—	- Prepare for app publication and hosting setup. - Plan continuous monitoring and feedback collection. - Schedule future updates and feature enhancements.	Future Milestone: Deployment planned after final client approval and additional testing.

Weekly Thursday milestone meetings supported coordination, accountability, and consistent progress toward each SDLC goal.



	Task Name	Duration	Start	Finish	Predecessors
1	1 Planning	5 days	Wed 10/1/25	Tue 10/7/25	
2	1.1 Define project objectives & scope	2 days	Wed 10/1/25	Thu 10/2/25	
3	1.2 Gather requirements	2 days	Thu 10/2/25	Sun 10/5/25	2SS+1 day
4	1.3 Assigned team roles & responsibi	3 days	Sun 10/5/25	Tue 10/7/25	3SS+1 day
5	1.4 Selected technologies	2 days	Wed 10/1/25	Thu 10/2/25	
6	2 Analysis	3 days	Wed 10/8/25	Sun 10/12/25	1
7	2.1 Collected system & user requiren	3 days	Wed 10/8/25	Sun 10/12/25	
8	2.2 Defined key roles	2 days	Wed 10/8/25	Thu 10/9/25	
9	2.3 Designed initial data flow & struct	3 days	Wed 10/8/25	Sun 10/12/25	
10	3 Design	3 days	Mon 10/13/25	Wed 10/15/25	6
11	3.1 Decided on Design principles	3 days	Mon 10/13/25	Wed 10/15/25	
12	3.2 Created UI/UX wireframes, mocku	3 days	Mon 10/13/25	Wed 10/15/25	
13	3.3 Developed system architecture &	2 days	Mon 10/13/25	Tue 10/14/25	
14	3.4 Finalized layout consistency & vi	2 days	Mon 10/13/25	Tue 10/14/25	
15	4 Implementation	13 days	Thu 10/16/25	Mon 11/3/25	10
16	4.1 Implemented patiend screens	7 days	Thu 10/16/25	Sun 10/26/25	
17	4.2 Implement Lab Admins screens	10 days	Mon 10/20/25	Sun 11/2/25	16SS+2 day
18	4.3 Implement Super Admin screens	7 days	Sun 10/26/25	Mon 11/3/25	17SS+4 day
19	4.4 Merge git branches	6 days	Thu 10/16/25	Thu 10/23/25	
20	5 Testing	8 days	Thu 11/6/25	Mon 11/17/25	15F S+2 day
21	5.1 Write Test Cases	2 days	Thu 11/6/25	Sun 11/9/25	
22	5.2 Run Tests	4 days	Thu 11/6/25	Tue 11/11/25	
23	5.3 Debug	6 days	Mon 11/10/25	Mon 11/17/25	22SS+2 day
24	6 Deployment	5 days	Thu 11/20/25	Wed 11/26/25	20F S+2 day
25	6.1 Prepare aab, apk files	2 days	Thu 11/20/25	Sun 11/23/25	
26	6.2 Release on Goggple play	4 days	Sun 11/23/25	Wed 11/26/25	25SS+1 day
27	7 Maintenance	5 days	Thu 11/27/25	Wed 12/3/25	24
28	8 Progress Meeting	41 days	Thu 10/2/25	Thu 11/27/25	

5. KPIs (Key Performance Indicators)

- **User Retention Rate** : percentage of active returning users
- **Monthly Booking Volume** : total number of bookings per month
- **Error Rate** : number of failed operations (upload failures, Firestore sync issues)
- **Average Booking Completion Time** : time from selecting a lab to completing the booking
- **Crash-Free Sessions** *:* percentage of sessions without app crashes
- **Data Synchronization Success Rate** *:* percentage of Firestore operations completed successfully

B. Literature Review

1. Related Applications & Research

LabLink builds upon concepts found in existing healthcare booking systems:

- **Vezeeta** – Appointment booking platform for clinics and hospitals
- **Laboratory Information Systems (LIS)** – Used by labs for managing test workflows
- **Online Medical Appointment Systems** – Research emphasizes usability, security, and reliability

LabLink differentiates itself by integrating **multi-role management (patients, lab admins, super admin)** into a single mobile interface using Flutter and Firebase.

2. Lecturer Feedback & Evaluation

During development, the instructor provided feedback in two main areas:

Firebase Integration Issues

- The team encountered difficulties uploading images to Firebase Storage.
- The instructor suggested trying Supabase as an alternative, but after further debugging, Firebase Storage was successfully used.

Testing & Provider Issues

- Unit test cases involving fake providers were not functioning correctly.
- The instructor advised replacing the fake provider with the real provider during widget testing, which resolved the issue.

This feedback improved reliability and highlighted common pitfalls with asynchronous testing and third-party storage services.

3. Suggested Improvements

- Introduce caching for test lists and branches to reduce Firestore reads.
- Add in-app notifications for booking updates.
- Implement online payments for improved convenience.
- Enhance test result uploads with template support.
- Add offline mode for viewing past bookings.

C. Requirements Gathering

1. Stakeholder Analysis

Patients: Patients are the end-users of the application. Their main interest is in a convenient, secure, and transparent process for booking laboratory tests, viewing results, and providing feedback. Their satisfaction is achieved through a user-friendly design, reliable performance, and the ability to rate labs to improve service quality.

Lab Administrators: Lab Administrators play a critical role in the system's daily operation. They manage branches, define test services and prices, handle bookings, upload results, and monitor patient reviews. Their effectiveness directly influences the user experience. Providing intuitive management tools and clear dashboards enhances their productivity and accuracy.

Super Administrator: The Super Administrator acts as the system overseer and ensures the overall platform integrity. This role manages lab admin accounts,

monitors lab activities, and ensures compliance with system standards. Their influence is high, as they control data consistency and operational efficiency.

Development Team: The development team is responsible for building, testing, and maintaining the application. Their interest lies in ensuring performance, scalability, and a high-quality user experience. They continue to refine the system based on stakeholder feedback and technical evaluations.

Future Stakeholders: Future stakeholders may include healthcare authorities or data compliance bodies, especially as the application scales. Their involvement would focus on verifying data privacy standards, ethical handling of patient information, and potential integration with healthcare systems.

2. User Stories

Patient

- As a patient, I want to browse available laboratories so I can find nearby options.
- As a patient, I want to upload a prescription so the lab can identify required tests.
- As a patient, I want to book an appointment so I can visit the lab at a suitable time.
- As a patient, I want to rate the lab so I can share feedback.

Lab Admin

- As a lab admin, I want to manage branches so I can keep location data updated.
- As a lab admin, I want to add/edit available tests so patients can see accurate prices.
- As a lab admin, I want to accept or reject bookings so I can control workflow.
- As a lab admin, I want to upload test results to inform patients of outcomes.
- As a lab admin, I want to view ratings to monitor service quality.

Super Admin

- As a super admin, I want to manage lab accounts so I can control system access.
- As a super admin, I want to monitor labs' performance so I can ensure service quality.
- As a super admin, I want to view system-wide metrics for decision-making.

3. Functional Requirements

Patient

- Register & log in
- Browse labs and branches
- Upload prescriptions
- Book appointments
- View results
- View booking history
- Submit ratings and reviews

Lab Admin

- Manage branch information
- Add/edit tests and prices
- View/approve/reject bookings
- Upload results
- View patient feedback
- Access dashboard metrics

Super Admin

- Add, edit, or remove lab accounts
- Monitor lab performance
- View system-wide reports

4. Non-Functional Requirements

- **Performance:** App should load core screens within 3 seconds.
- **Security:** Firebase Authentication + Firestore Security Rules for role-based access.
- **Usability:** Clean UI with consistent navigation across roles.

- **Reliability:** Firestore data sync should succeed ≥ 95% of the time.
- **Scalability:** Must support adding new labs without app updates.
- **Accessibility:** Readable text, proper contrast, and large tap areas.

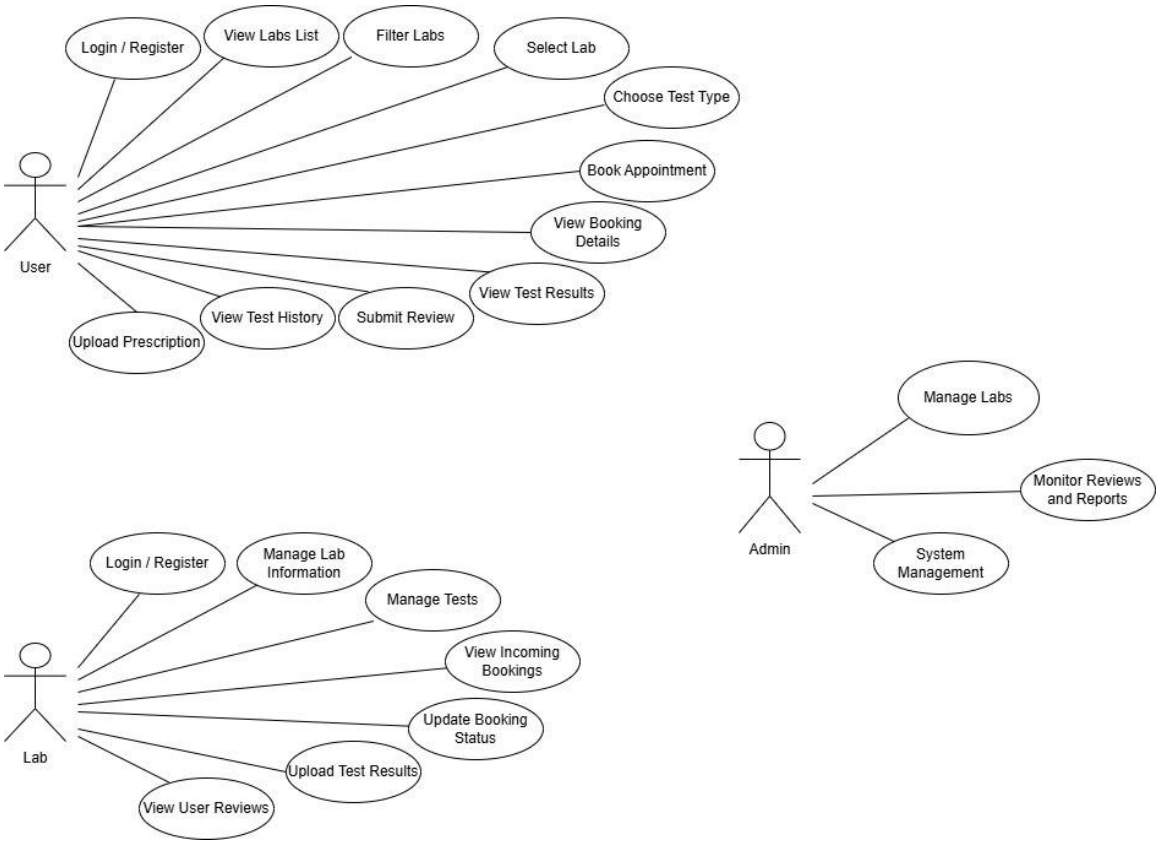
D. System Analysis & Design

1. Problem Statement & Objectives

LabLink solves the fragmented process of booking lab tests by centralizing access to accredited laboratories, providing transparency, convenience, and efficiency.
The objective is to create a unified mobile platform for patients and admins with automated booking workflows, secure data handling, and clear operational oversight.

2. Use Case Diagram & Descriptions

Use Case Diagram



Descriptions

Use Case: Book Test
Actor: Patient
Steps: Browse => Select test => Choose branch => Schedule => Confirm booking

Use Case: Upload Test Results
Actor: Lab Admin
Steps: Select patient booking => Upload PDF => Save => Patient views results

Use Case: Manage Lab Accounts
Actor: Super Admin
Steps: Create/edit/delete lab => Assign credentials => Monitor performance

3. Software Architecture (High-Level)

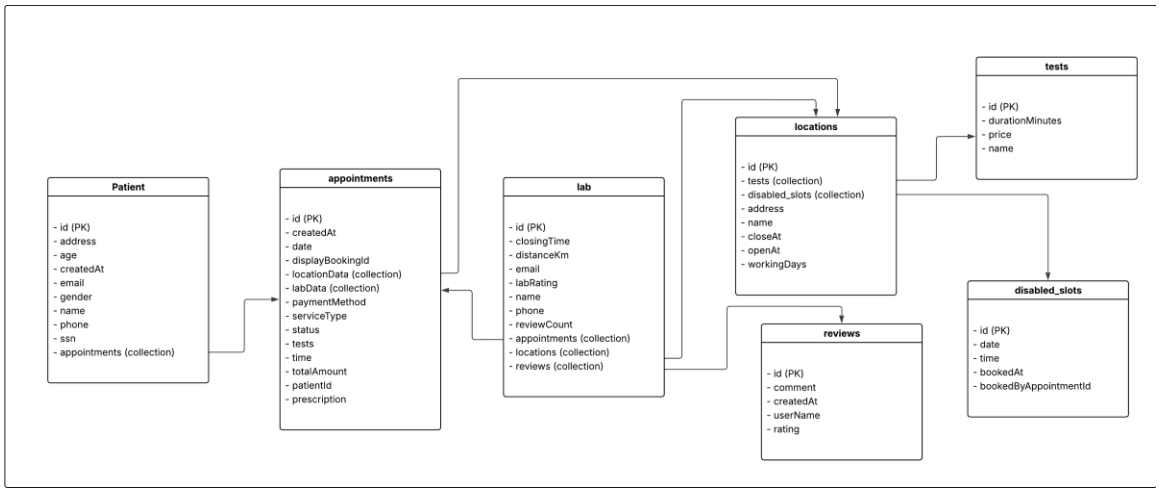
Architecture Style: Role-Based Layered Architecture

LabLink is structured as:

- Presentation Layer (per role)
  Screens, widgets (Flutter)
- Logic Layer (per role)
  ChangeNotifier Providers
  Controllers
  Utilities
- Data Access Layer (per role)
  Firebase Auth
  Firebase Firestore
  Firebase Storage
  (All used directly inside role folders)
- Shared Data Models (global)
  /Models directory shared by all roles

This architecture ensures separation by responsibility while keeping role-specific workflows independent.

4. Database Design & Data Modeling

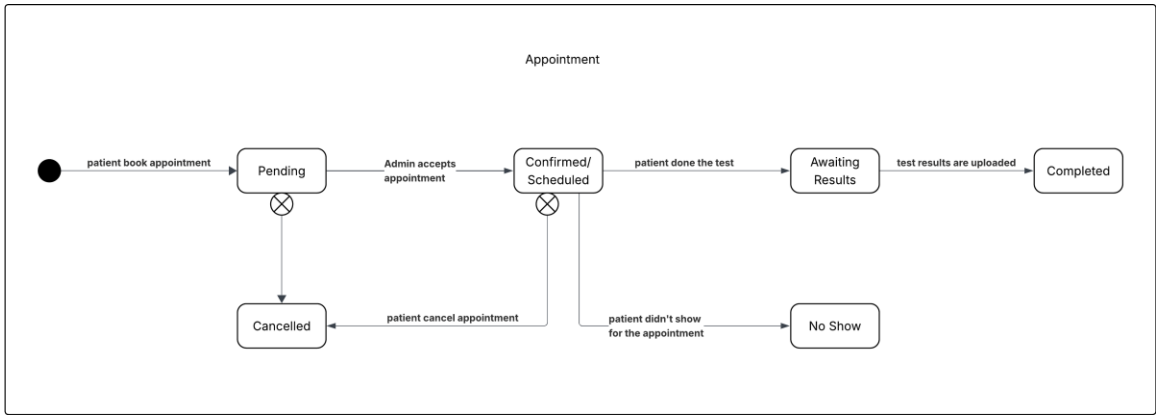


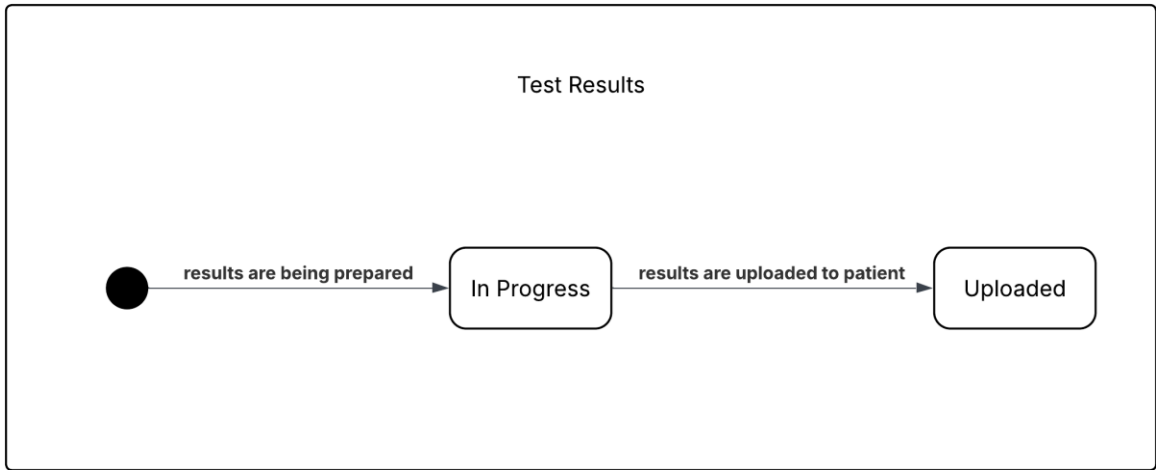
Logical & Physical Schema

- Patient
- Appointments
- Lab
- Locations
- Reviews
- Tests
- Disabled_slots

5. Diagrams & System Behavior

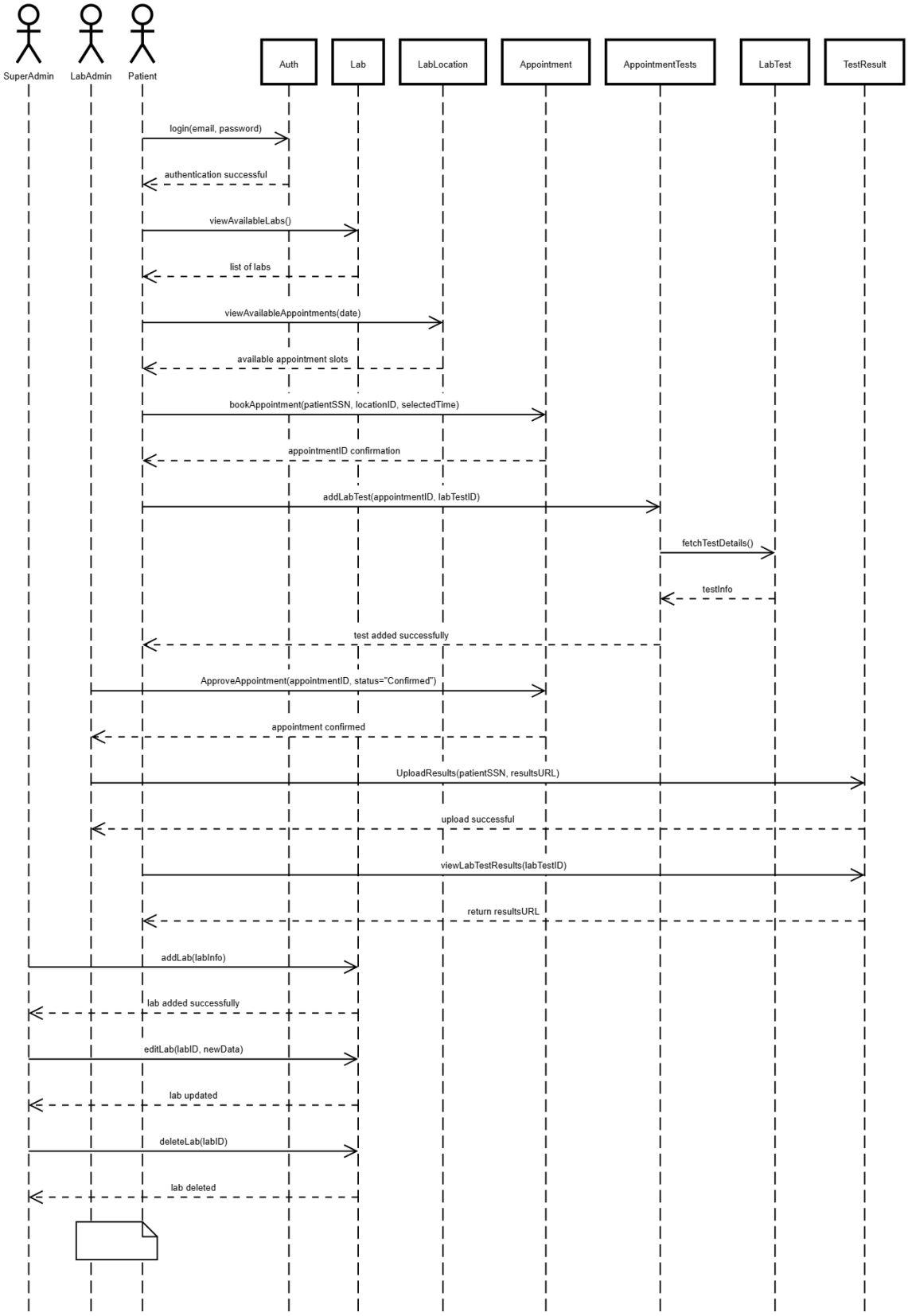
State Machine Diagram Placeholder



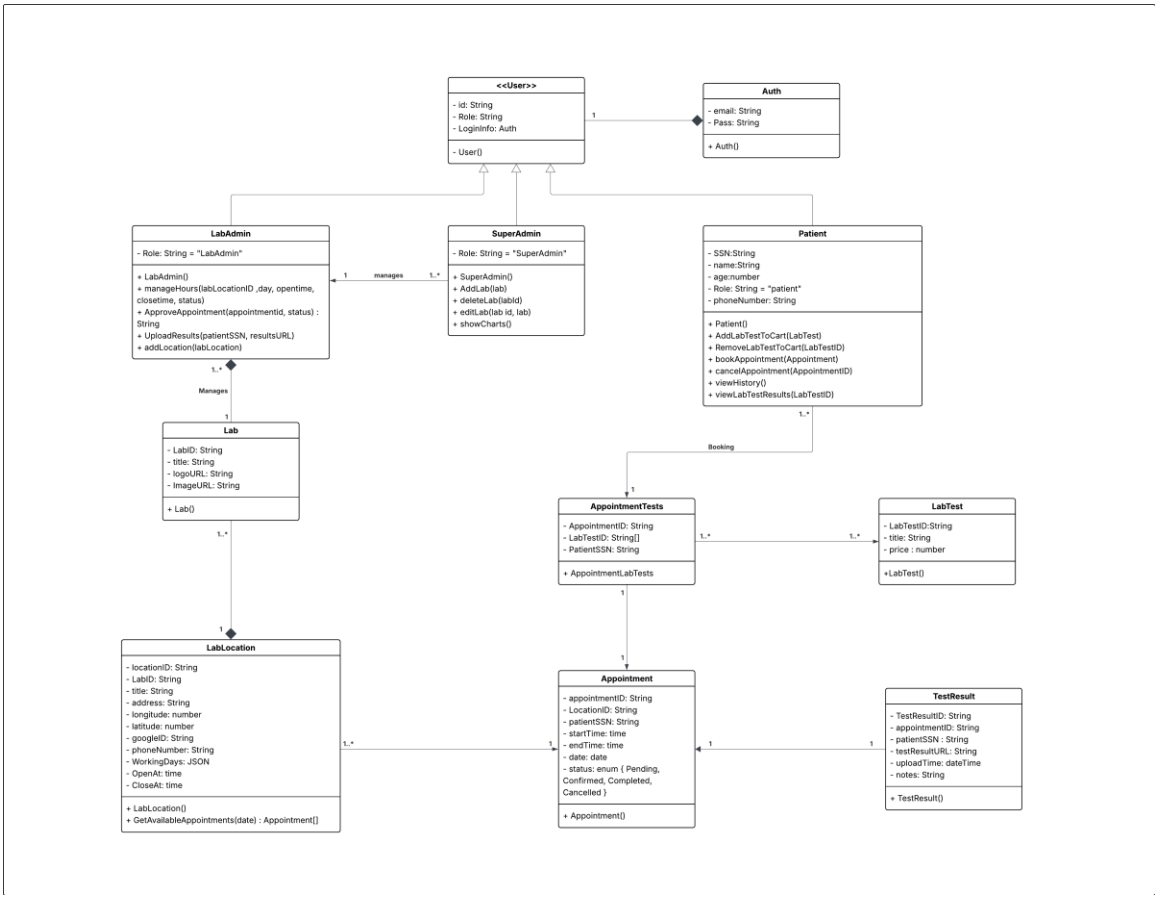


Sequence Diagrams Placeholder

LabLink System Sequence Diagram

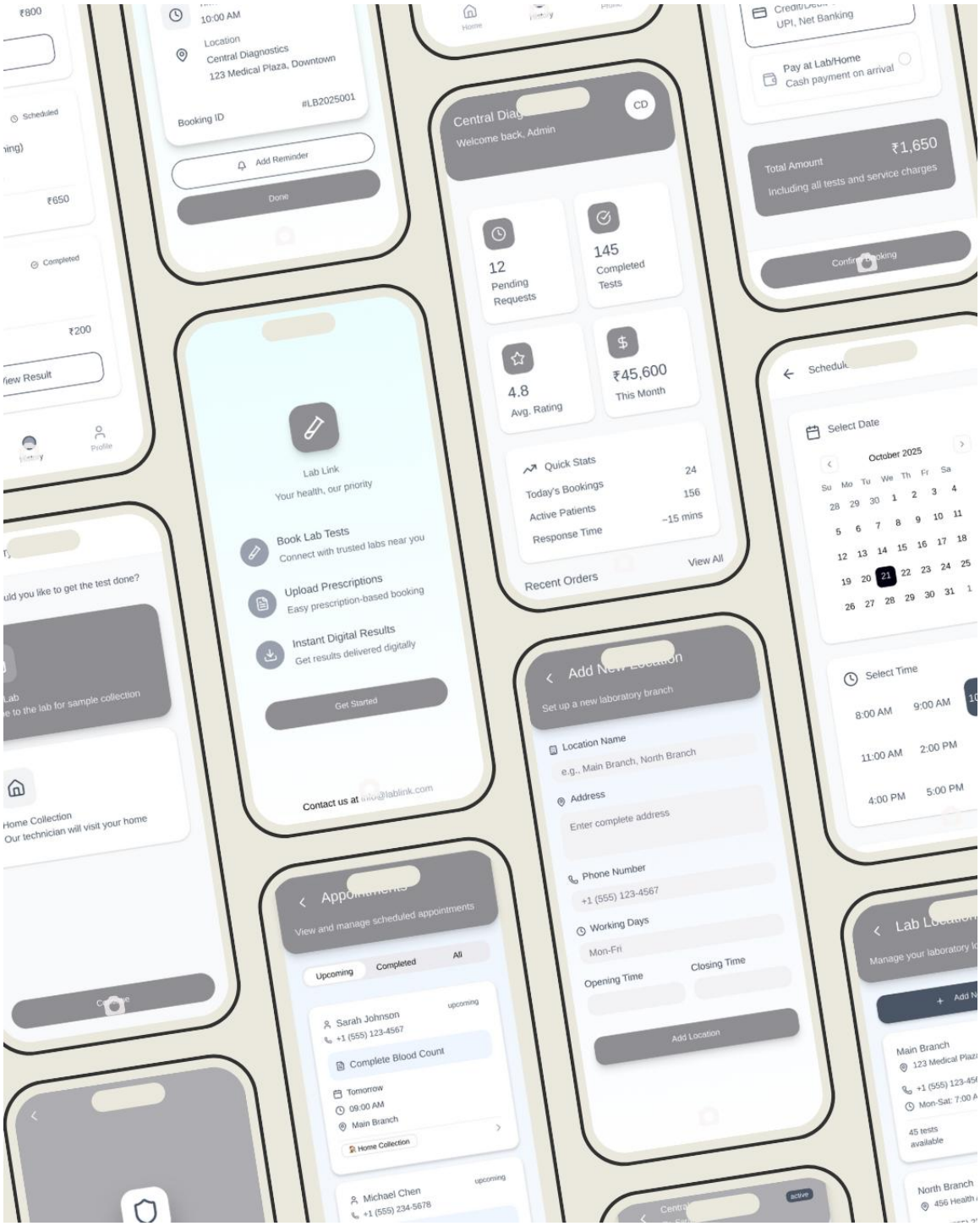


Class Diagram Placeholder



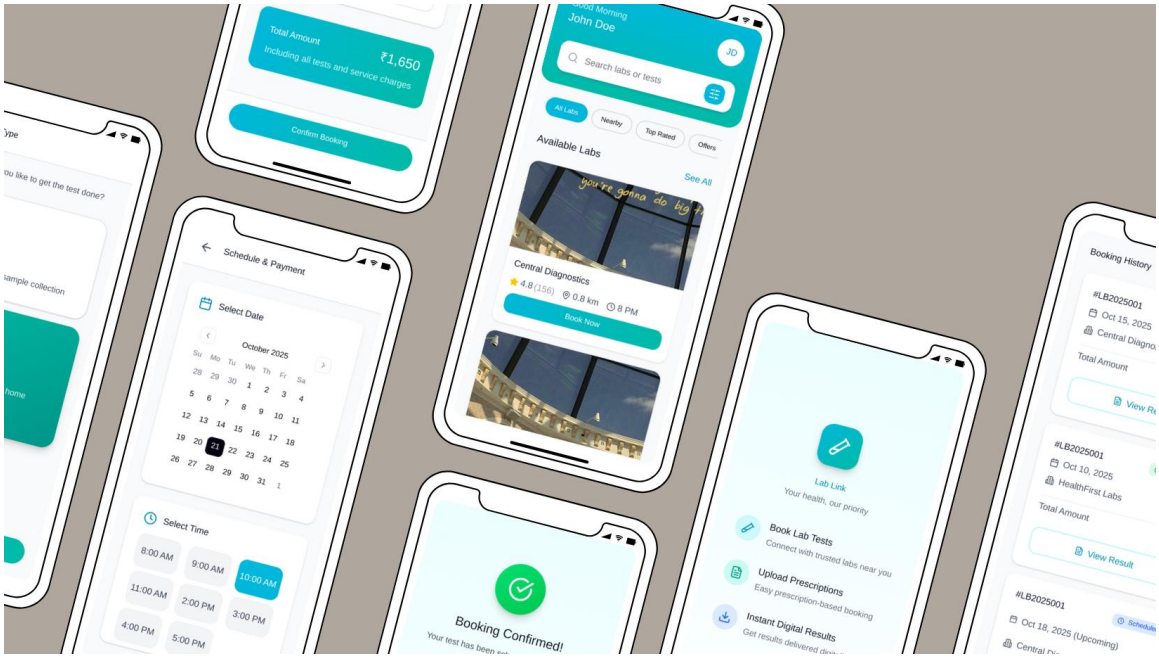
6. UI / UX Design

1. Wireframes

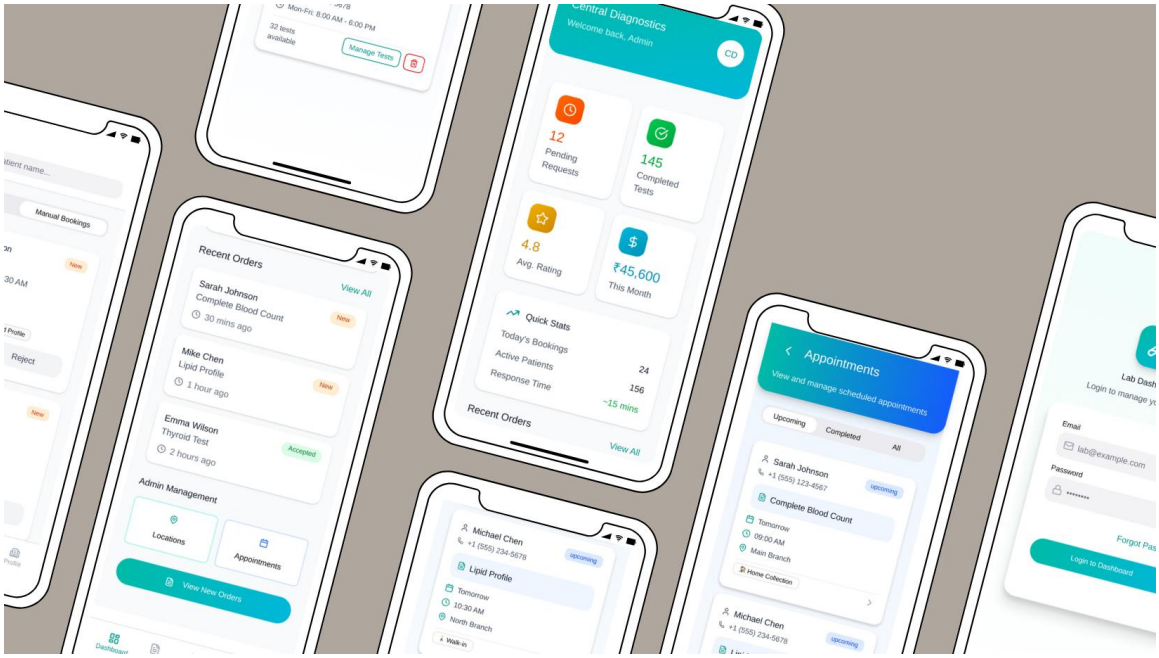


2. Mockups

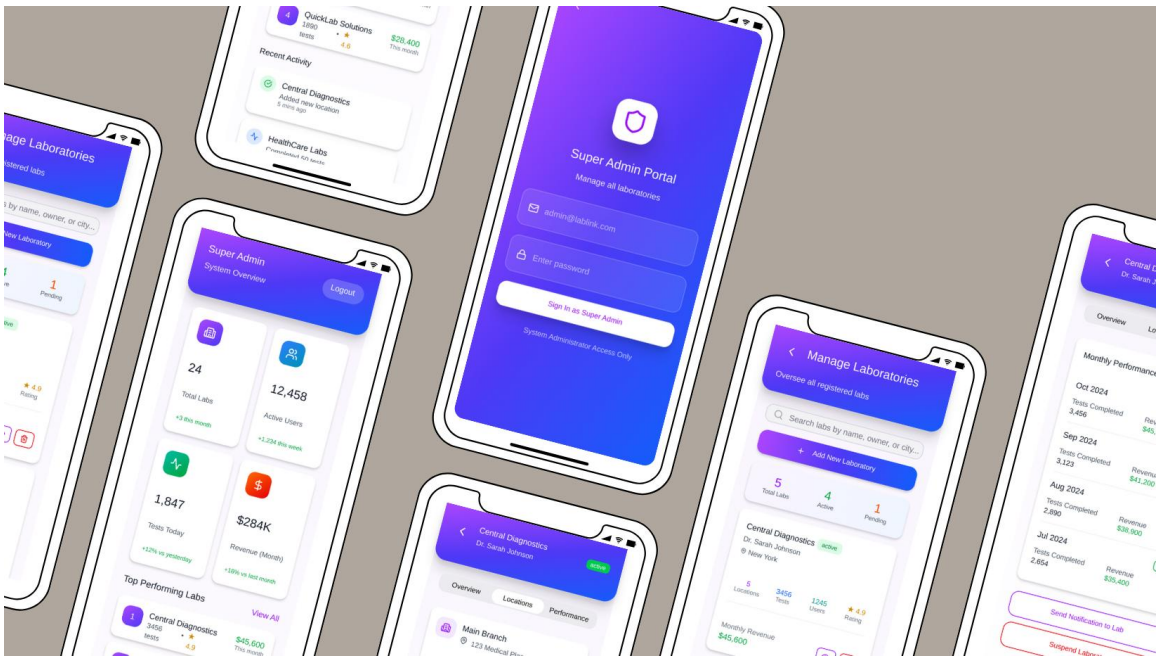
Patient Screens



Lab Admin Screens



Super Admin Screens



3. UI/UX Guidelines

Design Principles

The LabLink application follows a user-centered design approach focused on simplicity, clarity, and accessibility. Each interface is designed to guide users intuitively through key tasks such as booking tests, managing appointments, and monitoring analytics. Consistent spacing, visual hierarchy, and icon usage ensure that users can interact with the application efficiently, regardless of their technical background. The design emphasizes minimal cognitive load by reducing clutter, using clean card-based layouts, and prioritizing essential information on each screen.

Color Scheme

The color palette of LabLink reflects professionalism and trust within the healthcare domain while maintaining a modern and friendly appearance.

- **Primary Colors:**
  - Teal Green `#00BBA7` : used for buttons, highlights, and key interactive components.
  - Aqua Blue `#00B8DB` : applied in gradients and accent areas to enhance visual continuity.
- **Super Admin Accent Colors:**
  - Violet `#AD46FF` and Deep Indigo `#4F39F6` : utilized in administrative interfaces to differentiate higher-level control panels.

The consistent use of gradients, typically blending teal and aqua tones, creates a dynamic visual flow and reinforces the brand identity. Neutral backgrounds with light grays and whites ensure that interface elements remain the focal point while maintaining readability.

Typography

LabLink uses the **Roboto** typeface, the default font from Material Design. Roboto provides excellent legibility across different screen sizes and densities, ensuring that both patients and administrators can easily navigate the app.

- **Headings (H1–H4):** Bold or Medium weights, primarily used for section titles and key metrics.
- **Body Text:** Regular weight, used for descriptions, booking details, and system feedback.
- **Labels and Buttons:** Medium or Semi-Bold, ensuring strong visual distinction for interactive elements.

Iconography

The application incorporates both **Material Icons** and **Font Awesome** icon sets to represent actions and categories clearly.

- Material Icons are primarily used for navigation and functional buttons (e.g., Dashboard, Orders, Reports).
- Font Awesome icons complement the interface by enhancing visual communication in cards, lists, and status indicators (e.g., accepted orders, new bookings).

All icons follow a consistent line style and are aligned with the overall minimalist aesthetic. Their uniform stroke width and spacing contribute to a cohesive user experience.

Animations and Interactivity

To enhance user engagement and provide positive visual feedback, the LabLink app incorporates **Lottie animations**.
A subtle Lottie animation is displayed on the **booking confirmation screen**, signaling the successful completion of an appointment. This addition provides a modern and interactive touch, helping users associate the action with a sense of completion and satisfaction.
Animations are used sparingly throughout the application to maintain performance efficiency and ensure they do not distract from primary actions.

Accessibility Considerations

Accessibility is a core design priority in LabLink. The interface uses high-contrast text and icon colors to ensure visibility against light backgrounds. Buttons and touch targets maintain sufficient size for easy interaction on mobile devices. Font sizes and padding adhere to Material Design guidelines, ensuring clarity and readability across various screen resolutions.
In addition, visual cues such as color-coded badges (e.g., *New*, *Accepted*) and icons improve the interpretability of booking statuses without relying solely on text.

7. System Deployment & Integration

Technology Stack (High-Level)

- **Frontend:** Flutter (Dart)
- **State Management:** Provider + ChangeNotifier

- **Backend Services:** Firebase Auth, Firestore, Firebase Storage
- **Platform:** Android mobile application

Deployment

Since LabLink is not server-hosted, the deployment components will include:

- Android Device
- Firebase Cloud (Auth / Firestore / Storage)

Components

Components include:

- Patient Module
- Lab Admin Module
- Super Admin Module
- Shared Models
- Firebase backend

E. Implementation

1. Source Code

- Code follows meaningful naming conventions
- Uses reusable widgets and providers
- Firebase calls encapsulated per role
- Validation and error handling included in forms and workflows
- Shared models used across roles

2. Version Control & Collaboration

Repository

GitHub: https://github.com/Habiba04/LabLink

Branching Strategy

- **main** => stable production version
- **feature branches** => one feature per branch
- Pull requests required before merging
- Team reviewed code before merging into main

3. Deployment & Execution

README file covers:

- Project Description
- Features
- User Roles
- Installation
- Usage

Executable files are generated through the Flutter build process (APK).

F. Testing & Quality Assurance

Summary

Testing was conducted using unit, widget and manual tests.
Main issues encountered:

- Unsynced Firestore writes
- Missing data in collections
  Resolved through improved validation and consistent Firestore structure.

G. Final Presentation & Reports

1. User Manual

A user manual includes:

For Patients

- Register/Login
- Browsing labs
- Selecting tests or uploading prescriptions
- Booking appointments
- Viewing results
- Rating labs

For Lab Admin

- Adding branches
- Managing tests
- Viewing bookings
- Uploading results
- Viewing reviews

For Super Admin

- Creating lab accounts
- Editing or deleting accounts
- Monitoring system-wide activity

2. Technical Documentation

Include:

- System architecture
- Data models and Firestore collections
- API interactions with Firebase (Auth, Firestore, Storage)