****

**Project Advisor:**

**Ms. Rida Ayesha**

**Submitted By:**

**Nittasha s2024266092**

**Wasaf Zahra s2024266096**

**Maheen Hussain s2024266055**

**Session**

**2024\_2028**

**University of Management and Technology**

**C-II Johar Town Lahore Pakistan**

**Dedication**

This project is dedicated to all the students and staff members who have ever lost or found something valuable on campus, and to the campus community striving for better solutions through technology.

## 

**Final Approval**

* **Head of Department** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Department of Computer Science

School of Systems & Technology

UMT Lahore

* **Director (Final Year Projects-CS)** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Department of Computer Science

School of Systems & Technology

UMT Lahore

* **Supervisor** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Department of Computer Science

School of Systems & Technology

UMT Lahore

* **Co-Supervisor** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgment**

We are thankful to Almighty Allah for giving us the strength and knowledge to complete this project. Special thanks to our respected teacher Ms. Rida Ayesha for her continuous guidance and support throughout the project. We are also grateful to our classmates and families for their encouragement.

**Project Title Lost & Found Web App for Campus**

**Objective** To develop a centralized, user-friendly web application that helps university students and staff report and recover lost items within campus premises efficiently and securely.

**Undertaken by**  Nittasha

Wasaf Zahra

Maheen Hussain

**Supervised by Mrs.Rida Ayesha**

**Starting Date**

**Completion Date 4\20\2025**

**Tools Used**

* Visual studio code
  + - * ReactJS
      * Node.js
      * MongoDB
      * Firebase Authentication
      * Twilio / EmailJS (for notifications)
      * Figma (UI design)

**Operating System**

* + - * Windows 10
      * Ubuntu
      * Android
      * iOS

**Documentation** Microsoft Word

**Declaration Form**

**FYP Advisor Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Abstract**

The Lost & Found Web App for Campus is a digital platform designed for educational institutions to manage lost and found item reporting efficiently. It allows users to report lost or found items, view listings, claim ownership, and get real-time updates via email or SMS. The platform provides a systematic approach to reduce unclaimed lost items and ensures transparency and accountability through admin moderation and user authentication.

Revision Chart

| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| --- | --- | --- | --- |
| *Draft* | Nittasha  Wasaf  Maheen | Initial draft for Lost & Found Web App prepared for first internal review | (To be decided) TBD |
| *Preliminary* | Nittasha  Wasaf  Maheen | Added feedback from supervisor Ms. Rida Ayesha, refined problem/objectives | TBD |
| *Final* | Nittasha  Wasaf  Maheen | Fully structured and formatted report submitted for final evaluation | TBD |
| *Revision 1* | Nittasha | (If needed) Minor corrections made after final submission review | TBD |
| Revision 2 | Nittasha | Revised draft, revised according to the change control process and maintained under change control | TBD |
| *Etc.* | TBD | TBD | TBD |

Table of Content

Contents

[a. Definitions and Acronyms 3](#_Toc203376615)

[b. List of Figures 4](#_Toc203376616)

[c. List of Tables 4](#_Toc203376617)

[1. Introduction 5](#_Toc203376618)

[1.1 Motivations 5](#_Toc203376619)

[1.2 Project Overview 6](#_Toc203376620)

[1.3 Problem Statement 7](#_Toc203376621)

[1.4 Objectives 7](#_Toc203376622)

[2. Domain Analysis 9](#_Toc203376623)

[2.1 Customer 9](#_Toc203376624)

[2.2 Stakeholders 9](#_Toc203376625)

[2.3 Affected Groups with social or economic impact 9](#_Toc203376626)

[2.4 Dependencies/ External Systems 10](#_Toc203376627)

[2.5 Reference Documents 11](#_Toc203376628)

[2.5.1 Related Projects 11](#_Toc203376629)

[2.5.2 Feature Comparison 12](#_Toc203376630)

[3. REQUIREMENTS ANALYSIS 14](#_Toc203376631)

[3.2 List of Actors: 16](#_Toc203376632)

[3.3 List of use cases 17](#_Toc203376633)

[3.4 System use case diagram 19](#_Toc203376634)

[3.5 Extended use cases 19](#_Toc203376635)

[4. Project Scheduling & Prototyping 2](#_Toc203376636)

[4.1Work Breakdown Structure (WBS) 2](#_Toc203376637)

[4.2 Gantt-Chart for Work Breakdown Structure (WBS) 3](#_Toc203376638)

[4.3 Critical Path: 4](#_Toc203376639)

[4.4 User interfaces (Prototypes/mock screens) 5](#_Toc203376640)

[5. UML diagrams 2](#_Toc203376641)

[5.1 Data Flow Diagram Level 0 2](#_Toc203376642)

[5.2 SYSTEM DESIGN 3](#_Toc203376643)

[5.3 Class diagram 5](#_Toc203376644)

[5.4 Sequence Diagrams 6](#_Toc203376645)

[5.4.1 Sequence diagram 01 6](#_Toc203376646)

[5.4.2 Sequence Diagram 02 8](#_Toc203376647)

[5.4.3 Sequence Diagram 03 9](#_Toc203376648)

[7. Testing 10](#_Toc203376649)

[7.1 Extended Test Cases 10](#_Toc203376650)

[7.2 Decision Table 19](#_Toc203376651)

[7.2.1 Decision coverage table ( Submit Lost Item) 19](#_Toc203376652)

[5.4.1 20](#_Toc203376653)

[7.2.2 Decision coverage table (Submit Found Item) 20](#_Toc203376654)

[7.2.3 Decision coverage table ( Claim item) 21](#_Toc203376655)

[5.4.2 21](#_Toc203376656)

[7.2.4 Decision coverage table (Search and Filter Items) 21](#_Toc203376657)

[7.2.5 Decision coverage table (Login via firebase) 22](#_Toc203376658)

## Definitions and Acronyms

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
| UMT | University of Management and Technology |
| L&F | Lost and Found |
| UI | |  | | --- | | User Interface |  |  | | --- | |  | |
| UX | |  | | --- | |  |  |  | | --- | | User Experience | |
| DB | Database |
| CRUD | Create, Read, Update, Delete (basic operations in databases and backend systems) |
| OTP | One-Time Password |
| SMS | Short Message Service |
| API | |  | | --- | |  |  |  | | --- | | Application Programming Interface | |
| JWT | |  | | --- | |  |  |  | | --- | | JSON Web Token (used for secure user authentication and session management) | |
| VSC | Visual Studio Code |
| FOSS | |  | | --- | |  |  |  | | --- | | FoundIt Open Source System | |
| LIFS | |  | | --- | |  |  |  | | --- | | LUMS Lost & Found System | |
| LFWAC | Lost & Found Web App for Campus (our proposed system) |
| FCM | |  | | --- | |  |  |  | | --- | | Firebase Cloud Messaging (used for push notifications) | |
| Firebase | |  |  | | --- | --- | |  | A platform developed by Google for building mobile and web applications | |

## List of Figures

[Figure 1 20](#_Toc203378472)

[Figure 2 WBS 3](#_Toc203378473)

[Figure 3: Gantt Chart 4](#_Toc203378474)

[Figure 4: Critical path 5](#_Toc203378475)

[Figure 5 Prototype 1 (Login page) 6](#_Toc203378476)

[Figure 6 Prototype 2 (Home Screen) 7](#_Toc203378477)

[Figure 7Prototype 3 (Lost things) 8](#_Toc203378478)

[Figure 8 Prototype 4 (Submit item ) 9](#_Toc203378479)

[Figure 9 Prototype 5 (Admin Dashboard) 1](#_Toc203378480)

[Figure 10 Data flow diagram of lost and found web 2](#_Toc203378481)

[Figure 11 System architecture design for lost and found web 3](#_Toc203378482)

[Figure 12 Class Diagram of lost and found web 5](#_Toc203378483)

[Figure 13 Sequence Diagram 01 6](#_Toc203378484)

[Figure 14 Sequence diagram 02 8](#_Toc203378485)

[Figure 15 Sequence diagram 03 9](#_Toc203378486)

## List of Tables

[Table 1 Stakeholders 10](#_Toc203377393)

[Table 2 Feature Comparison 13](#_Toc203377394)

[Table 3 The Functional and Non-Functional Requirements 15](#_Toc203377395)

[Table 4 List of use cases 18](#_Toc203377396)

[Table 5 Login use case 20](#_Toc203377397)

[Table 6 Submit Lost Item 22](#_Toc203377398)

[Table 7: Submit Found Item 23](#_Toc203377399)

[Table 8 Approve Listings 26](#_Toc203377400)

[Table 9 Receive Notifications 28](#_Toc203377401)

[Table 10 Access Admin Dashboard 29](#_Toc203377402)

[Table 11 Update Profile 30](#_Toc203377403)

[Table 12 Generate Audit Reports 31](#_Toc203377404)

[Table 13: Handle Claim Requests 32](#_Toc203377405)

[Table 14 Test Case: Submit Lost Item 10](#_Toc203377406)

[Table 15: Submit Found Item 12](#_Toc203377407)

[Table 16 Test Case (Claim item) 14](#_Toc203377408)

[Table 17 Test Case (Search and Filter Items) 16](#_Toc203377409)

[Table 18 Test Case (Login via Firebase) 18](#_Toc203377410)

[Table 19 20](#_Toc203377411)

[Table 20: Decision coverage table (Found) 21](#_Toc203377412)

[Table 21 Decision coverage table (Claim) 22](#_Toc203377413)

[Table 22 Decision coverage table (Search and Filter Item) 22](#_Toc203377414)

[Table 23 Decision coverage table (Login via firebase) 22](#_Toc203377415)

# Introduction

In university campuses, thousands of students, faculty, and staff move between departments, lecture halls, libraries, cafeterias, and hostels daily. In this high-activity environment, it is very common for individuals to misplace or lose personal items, ranging from student ID cards and books to USB drives, wallets, and keys. Currently, most institutions lack a dedicated system for tracking, managing, or retrieving such items in a formal and secure manner.

Traditionally, students and staff rely on informal solutions such as posting about their lost belongings on WhatsApp groups, departmental notice boards, or in-person inquiries to reception desks and guards. While sometimes effective, these methods are often unreliable, prone to human error, and do not guarantee results. Additionally, there is no mechanism to validate ownership, record claims, or systematically match lost items with those that have been found.

Recognizing this gap, our team is developing a Lost & Found Web App specifically designed for campus use. This web-based platform aims to digitize and automate the lost and found process, making it secure, efficient, and accessible for the entire campus community. The app will allow university members to report lost or found items, upload photos, descriptions, and locations, and search existing listings to find matches. Furthermore, it will include admin moderation tools to verify entries, notification systems to alert users of potential matches, and a complete digital log of all transactions for transparency and accountability.

This solution addresses not only the operational challenges of item misplacement but also promotes a sense of trust, community, and responsibility among students and staff. By providing a centralized platform integrated with university systems, the Lost & Found Web App for Campus enhances student services,

## Motivations

University campuses are vibrant spaces filled with students, faculty, and staff constantly moving between classrooms, cafeterias, libraries, and other departments. In this busy environment, it is common for individuals to misplace personal belongings such as:

* Student ID cards
* USB flash drives
* Wallets or purses
* Stationery items
* Mobile accessories
* Books or notes
* Keys

Currently, there is no unified platform in most universities, including UMT, to manage and report these incidents. Students often rely on informal and unreliable methods such as:

* Posting in WhatsApp groups or Facebook pages
* Sticking physical notices on bulletin boards
* Asking friends or department staff to keep an eye out

These solutions are inefficient, fragmented, and have no mechanism to ensure accountability or effective matching between lost and found items. As a result, many valuable items are never returned to their rightful owners.

The motivation behind this project is to eliminate these challenges by introducing a centralized digital platform. This platform will make it easier to report, search, and claim lost or found items, saving time, increasing transparency, and encouraging responsibility among the campus community.

## Project Overview

The Lost & Found Web App for Campus is a dedicated, web-based application designed to simplify the process of handling lost and found items within university premises. The platform offers the following key features:

* **User Authentication:** Only registered university members (students, faculty, staff) can access the system through university-issued credentials.
* **Report Lost Items:** Users can post details about their lost items, including title, description, category, image, and location where it was lost.
* **Post Found Items:** Users can upload descriptions and pictures of items they found on campus, which will be stored in the system for others to browse.
* **Matching System:** The app uses matching logic based on keywords, categories, and time/location filters to help users identify potential matches.
* **Admin Moderation:** An admin panel allows authorized personnel to verify posts, prevent spam, and ensure legitimate use of the platform.
* **Notifications:** The system sends email or SMS notifications when a potential match is found or when updates occur related to a user's report.
* **Log and Tracking:** A digital trail is maintained for all entries, providing administrators a full history of reported and claimed items.

This centralized system ensures faster item recovery, improves transparency, and introduces a formal method of reporting which can be integrated into university operations and student services.

## Problem Statement

On large educational campuses, the absence of a structured lost and found management system leads to several problems:

1. **Frustration Among Students:** When students lose important items like ID cards, USBs, or notes before exams or assignments, they experience stress and difficulty in recovering them.
2. **Wasted Time and Resources:** A significant amount of time is wasted checking different departments or spreading messages across groups without any guaranteed results.
3. **Permanent Loss of Items:** In most cases, found items are either never reported or stored in security offices with no proper record, causing them to remain unclaimed.
4. **Lack of Accountability:** There is no system for verifying claims or tracking whether an item has been returned to the correct owner.
5. **Dependence on Informal Channels:** Informal communication channels like social media are not scalable, lack data protection, and often result in overlooked reports.

These issues collectively disrupt campus life, reduce productivity, and erode trust in the university’s ability to manage student services efficiently. Therefore, a dedicated, moderated, and digital solution is essential to address these concerns and provide a streamlined way to manage lost and found operations within the university.

## Objectives

The main goal of the Lost & Found Web App is to offer a comprehensive, automated, and secure solution for managing lost and found items on campus. The specific objectives of the project include:

**1. Centralized Digital Reporting System**  
To provide a centralized online portal where students and staff can easily report lost or found items, ensuring all data is in one place and accessible by the relevant stakeholders.

**2. Accurate Item Matching and Claim Process**  
To implement smart matching mechanisms that help users quickly identify if their item has been found or posted by someone else, and allow them to initiate a claim request securely.

**3. Real-Time Notifications**  
To ensure users are immediately notified when a match is detected or when their post status is updated (approved, denied, or claimed) via email or SMS.

**4. Secure Authentication and Access Control**To use verified university email accounts or student portals to log in, ensuring only legitimate university members can post and claim items.

**5. Admin Dashboard and Moderation Tools**  
To enable administrators or faculty members to monitor the system, approve posts, prevent abuse, and manage unresolved cases through a dedicated admin panel.

**6. Digital Logging and Transparency**  
To maintain a complete history log of all user activities related to item reporting, matching, and claiming—ensuring transparency, auditing, and system integrity.

**7. Enhanced User Experience and Accessibility**  
To offer a clean, mobile-friendly interface with filters, search options, and user guidance so anyone can use the system easily and efficiently.

# Domain Analysis

## Customer

University of Management and Technology (UMT) as the primary client. End users are students and faculty.

Table 1 Stakeholders

|  |  |
| --- | --- |
| **Stakeholder** | **Role in the System** |
| Students | Report and search for lost/found items using the web platform. |
| Faculty/Staff | Use the system to report or retrieve items and guide students in using the system efficiently. |
| Admins (Moderators) | Review, approve, and validate item reports and claims; ensure content authenticity and proper item handling. |
| IT Department | Maintain system functionality, handle technical issues, and ensure data security and backup. |
| Campus Security | Securely store found items and assist with the physical return of items after claim verification. |
| University Administration | |  | | --- | |  |  |  | | --- | | Set policies for system use, ensure integration with campus services, and monitor overall effectiveness. | |

## Affected Groups with social or economic impact

The deployment of the Lost & Found Web App for Campus will have a positive impact on multiple groups within the university, both socially and economically. These impacts align with the project’s core objectives of improving item recovery, streamlining reporting, and promoting responsibility.

* **Students**  
  Students are the primary beneficiaries of the system. They will experience reduced stress and quicker recovery of lost items, especially valuable ones like ID cards, mobile phones, and USBs. The platform will help save the cost of replacing lost items, improving economic well-being.
* **Faculty and Staff**  
  Faculty and staff who lose teaching materials or personal belongings will be able to report and recover items efficiently. Socially, this builds trust and responsibility within the campus. It also minimizes time lost in manually searching or informing admin offices.
* **Campus Security**  
  The system will reduce the manual burden on campus security personnel by providing a recorded, trackable system for lost and found. This leads to a more organized workflow, improves efficiency, and ensures safe storage and handover of items.
* **Administrative Departments**  
  Admins responsible for handling lost items will now be able to digitally log, validate, and approve claims. This reduces the chance of fraudulent claims and ensures auditability, leading to a more transparent and professional process.
* **Parents and Guardians**Although indirectly affected, parents benefit from the assurance that their child’s valuables can be recovered quickly. This saves them from the financial burden of frequently replacing items like phones, wallets, and ID cards.
* **IT Support Team**  
  The IT team will manage and maintain the system but benefit from streamlined support operations as the platform reduces helpdesk queries related to lost items. Their involvement ensures system security and continuity.
* **Finance Department**  
  If ID cards or official documents linked to campus payments are lost, this system helps ensure the item is verified before reissue, saving time and preventing fraud. This contributes to cost efficiency in reprinting and resource allocation.
* **Lost & Found Desk (if available)**  
  Staff at the physical Lost & Found desk will experience reduced paperwork and improved recordkeeping. The system will also help them prioritize verified claims, increasing user satisfaction.
* **University Image and Branding**  
  Deploying a modern digital system reflects the institution’s commitment to student well-being and technological advancement. Socially, it enhances the campus experience and builds a culture of responsibility and accountability.

## Dependencies/ External Systems

The successful implementation of the Lost & Found Web App depends on several external systems and technologies:

* **Firebase Authentication**  
  Used to authenticate users securely through university email accounts, ensuring that only authorized students and staff can access and interact with the system.
* **Email/SMS Services**Services like EmailJS or Twilio will be integrated to send real-time notifications to users about post approvals, claim status, or matching items.
* **Database (MongoDB / Firebase Realtime Database)**  
  A cloud-based NoSQL database is used to store user profiles, lost/found item records, timestamps, and system logs in a secure and scalable manner.
* **Cloud Hosting (Vercel / Firebase Hosting)**  
  The web app is deployed on cloud hosting platforms to ensure 24/7 access, smooth performance, and scalability for future usage by other departments or institutions.
* **Internet/Wi-Fi Connectivity**  
  As a web-based platform, consistent internet access is essential for users to report, view, and claim items.

## Reference Documents

The following systems and tools served as references and inspiration during the planning and design of this project:

* **UMT Feedback Portal**Studied for its simple and intuitive UI/UX structure, login mechanism, and integration with student records.
* **GitHub Open-Source Lost/Found Systems**  
  Reviewed several public repositories to understand backend logic, item matching features, and data structures used in similar community projects.
* **University Digital Reporting Tools**  
  Looked at existing reporting systems used for campus services to ensure our app aligns with user expectations and institutional requirements.

### Related Projects

To ensure that our project incorporates the best features from existing solutions and avoids common design flaws, we analyzed several related systems and platforms. Below is a list of reference projects along with features and insights adopted from them:

1. LUMS Lost & Found System

A basic internal system operated at LUMS for managing lost items.

* Inspiration: Admin moderation workflow and use of predefined categories (documents, electronics, etc.).
* Adopted Feature: Admin approval for all item submissions and post categorization to enhance searchability.

2. FoundIt (Open-source Project)

A public GitHub project offering a community-based lost and found system.

* Inspiration: Automated matching of reports based on keywords, date, and item type.
* Adopted Feature: Matching algorithm logic for cross-referencing lost and found reports using keyword filters.

3. UMT Feedback System

The university’s own system for gathering student feedback on courses and faculty.

* Inspiration: User authentication, simplicity of UI, and role-based access.
* Adopted Feature: Login system using university credentials and role separation (student, admin).

4. OLX (Item Listing Platform)

An online marketplace where users post and find products.

* **Inspiration**: Item listing structure, image upload, and advanced search filters.
* Adopted Feature: Clear item posting interface and search/filter functionalities for users to browse lost/found items.

5. Google Forms (Manual Reporting Tool)

Often used in universities for reporting lost items informally.

* Inspiration: Simplicity and form-based structure for data collection.
* Adopted Feature: Form-based approach in item reporting to keep the system user-friendly.

6. Roomi Lost & Found (Campus App Example)

A mobile app used by university dorms to manage lost items.

* Inspiration: Use of image-based reporting and real-time notifications.
* Adopted Feature: SMS/Email alerts for new matches and claims.

7. Thumbtack (Service Matching Platform)

A platform that matches users with local professionals, based on detailed profiles.

* Inspiration: Matching logic and user interaction flow.
* Adopted Feature: Structured data comparison for accuracy in item matching.

Table 2 Feature Comparison

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr**  **No.** | **Comparison Feature** | |  | | --- | |  |  |  | | --- | | **LUMS Lost & Found System (LLFS)** | | |  | | --- | |  |   **Foundlt Open Source System (FOSS)** | **Remarks** |
| **1.** | Admin Moderation | LLFS uses admin approval before displaying posts | Limited or no admin moderation | Our system uses a refined admin dashboard for post verification, improving trust and reducing spam. |
| **2.** | |  | | --- | |  |  |  | | --- | | Item Matching Algorithm | | Manual checking by users | |  | | --- | |  |  |  | | --- | | Basic keyword-based matching | | We enhance this using multiple filters (category, location, time) and structured descriptions for better accuracy |
| **3.** | |  | | --- | |  |  |  | | --- | | User Authentication | | May rely on general email or manual identity verification | |  | | --- | |  |  |  | | --- | | Basic email/password login | | We integrate Firebase Authentication with university credentials to ensure only authorized users can report or claim items. |
| **4.** | Notifications | |  | | --- | |  |  |  | | --- | | Not supported | | Limited email alerts | We implement real-time email and SMS alerts when a matching item is posted, improving user engagement. |
| **5.** | Image Upload and Attachment | Optional | |  | | --- | |  |  |  | | --- | | Supported | | |  | | --- | |  |  |  | | --- | | Our app allows image attachments and encourages visual evidence for both lost and found items, which improves claim verification. | |
| **6.** | |  | | --- | |  |  |  | | --- | | Mobile Responsive UI | | Basic | |  | | --- | |  |  |  | | --- | | Basic | | |  | | --- | |  |  |  | | --- | | Our interface is fully responsive, providing seamless experience across desktop, tablet, and mobile devices. | |
| **7.** | |  | | --- | |  |  |  | | --- | | Search and Filter Options | | Limited filters | |  | | --- | |  |  |  | | --- | | Text-based search only | | We provide category, time range, and location-based filters for faster and more accurate search results. |

# REQUIREMENTS ANALYSIS

Lost & Found Web App for Campus addresses a longstanding challenge faced in educational institutions—misplaced items with no effective recovery mechanism. Traditionally, students rely on outdated methods such as notice boards, WhatsApp groups, and manual announcements. This app offers a structured, efficient, and transparent solution that bridges this gap using modern technology.

With a user-friendly interface, university members can quickly post a report for a lost or found item, attach descriptions and images, and contact each other through verified profiles. The real-time search and filter options allow users to look up posts based on category, location, date, and keywords—streamlining the process of reconnecting items to their rightful owners.

Security is a key concern in any digital platform. Therefore, the Lost & Found Web App incorporates Firebase Authentication to ensure only authorized university members can report or claim items. Admins play a crucial role in moderating the platform, verifying each submission and managing claims to prevent abuse or spam.

Notifications via SMS or email enhance the user experience by immediately alerting them when a potential match is found or their post status changes. This reduces delays in communication and maximizes the likelihood of item recovery.

The app is designed to be mobile-responsive and cross-platform compatible, making it accessible on desktops, tablets, and mobile devices. Whether a student is in class, in the library, or walking across campus, they can conveniently report or search items in real time.

Furthermore, digital logs maintained by the system ensure transparency and accountability in the lost and found process. Admins can track the entire lifecycle of an item from reporting to recovery, generating reports for institutional use if needed.

By integrating modern authentication, responsive design, real-time notifications, and an admin-backed moderation system, the Lost & Found Web App transforms a chaotic process into an organized and user-friendly experience for the entire campus community.

Table 3 The Functional and Non-Functional Requirements

The following table lists the requirements for the Lost & Found Web App for Campus. These include functional services directly requested by users, supporting non-functional attributes, as well as system constraints and performance expectations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RID** | **Description** | **Attribute** | **Details & Boundary Constraints** | **Category** |
| R1.1 | Record a lost or found item | User Submission | Users submit item details including description and image | Functional |
| R1.2 | Browse listings | UI Responsiveness | Results should load within 2 seconds under normal load | Functional |
| R1.3 | Search/filter items | UI Performance | Search results appear within 3 seconds for up to 1000 entries | Functional |
| R1.4 | Notify on match/claim | Notification Response | Notification sent within 10 seconds of event trigger | Functional |
| R1.5 | Admin post approval | Moderation Workflow | Admin must approve/reject entries within 24 hours | Functional |
| R1.6 | Image attachment for item posts | Media Handling | Max image size: 2MB; Formats allowed: JPG, PNG | Functional |
| R2.1 | Secure login using Firebase Auth | Security | JWT & Firebase-based authentication for all users | Non-Functional |
| R2.2 | System available for use 24/7 | Uptime | Availability must be 99.9% per month | Non-Functional |
| R2.3 | Page load speed | Performance | Main pages should render in <3s on 4G network | Non-Functional |
| R2.4 | Cross-platform support | Compatibility | Fully functional on Chrome, Firefox, Safari, Edge | Non-Functional |
| R2.5 | Screen reader support | Accessibility | UI components must follow WCAG 2.1 standards | Non-Functional |
| R3.1 | Store item data in Firebase/MongoDB | Database Storage | Structured records with timestamps and image URLs | Data Requirement |
| R3.2 | Retain claim logs and history | Data Retention | Maintain records for 6 months minimum | Data Requirement |
| R4.1 | Admin moderation only via secure login | Access Control | Admin actions restricted to verified staff accounts | Constraint |
| R4.2 | Max 5MB upload limit per session | Storage Limit | Prevent bandwidth abuse via file size checks | Constraint |
| R5.1 | Integration with Twilio/Email JS | Notification Gateway | Must support fallback if one API fails | External Interface |
| R5.2 | Hosted via Firebase or Vercel | Deployment Target | CDN-based hosting for speed and reliability | External Interface |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RID** | **Description** | **Category** | **Attribute** | **Details & Boundary Constraints** |
| FR1 | Login | Functional | User Interface | Users (students/staff/admin) login using university email via Firebase Auth |
| FR2 | Submit Lost Item | Functional | User Interface | User fills form with item name, image, category, and location |
| FR3 | Submit Found Item | Functional | User Interface | Similar to Lost Item submission but marked as "found" |
| FR4 | View Listings | Functional | User Interface | All users can browse paginated item listings |
| FR5 | Search and Filter Items | Functional | User Interface | Search by keyword, date, location, category, or status |
| FR6 | Claim Item | Functional | User Interface | Request form sent to admin with user justification |
| FR7 | Admin Moderation Panel | Functional | Admin Interface | Admin can approve/reject/edit item entries |
| FR8 | Notifications | Functional | System Behavior | Email/SMS alerts for matches and updates |
| FR9 | Image Upload | Functional | User Interface | Image preview and upload functionality |
| NFR1 | Security | Non-Functional | Data Security | Firebase Auth + JWT; HTTPS for secure transfer |
| NFR2 | Responsiveness | Non-Functional | Usability | UI adapts to mobile, tablet, and desktop |
| NFR3 | Reliability | Non-Functional | Performance | 99.9% uptime with fallback error handling |
| NFR4 | Usability | Non-Functional | Interface Design | Simple UI/UX for all campus users |
| NFR5 | Maintainability | Non-Functional | Code Quality | Modular architecture; easily updatable |
| NFR6 | Compatibility | Non-Functional | System | Works on Chrome, Firefox, Edge, Safari |
| NFR7 | Accessibility | Non-Functional | Interface Design | Color contrast and readable text supported |
| NFR8 | Legal & Compliance | Non-Functional | Privacy | Follows GDPR and local data protection laws |

## 3.2 List of Actors:

The **system boundary** for the Lost & Found Web App for Campus includes all stakeholders who interact with the platform either directly or indirectly. Below is a categorized list of all actors and their roles in the system:

* **Student:** The primary actor who uses the system to report lost or found items. Students can also search listings, claim items, and receive notifications.
* **Faculty/Staff:** Members of the university who may report or retrieve items, assist students in using the system, or handle found items informally. Some may also receive notifications.
* **Admin (Moderator):** Responsible for approving/rejecting item listings, managing user claims, resolving disputes, and ensuring that the data in the system is accurate and free of spam.
* **Campus Security:** May physically handle and verify possession of found items. They can coordinate with the admin or provide manual verification during item return.
* **IT Support Team:** Maintains system integrity, ensures backend functionality, resolves bugs, and oversees data storage, authentication, and notification systems.
* **University Management:** Sets policies, oversees the system's implementation, and monitors its effectiveness. Also included in strategic integration with campus services.

These actors collectively define the operational and support framework for the platform and are all considered key stakeholders.

**Table 3.2: List of actors**

|  |  |
| --- | --- |
| **Actor** | **Description** |
| Student | Primary user who reports/searches/claims lost and found items |
| Admin | Moderator who approves entries, manages claims, and sends alerts |
| Staff | May assist in recovering items and verifying claimant authenticity |

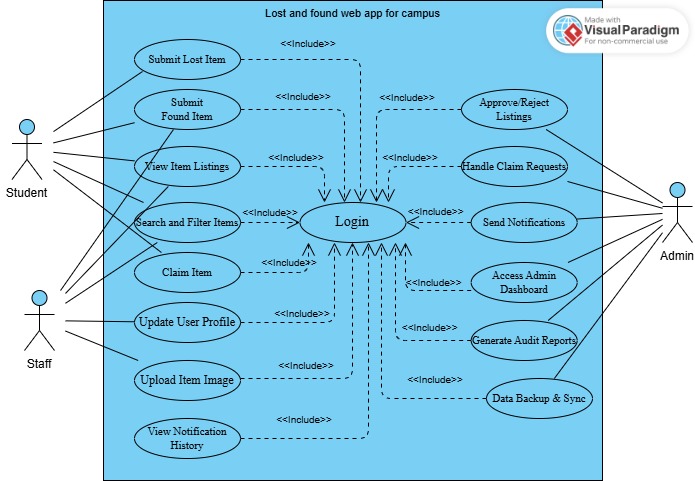
## List of use cases

Table 4 List of use cases

The following table outlines the main use cases for the **Lost & Found Web App for Campus**, describing the core functionalities provided to each system actor.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | **Use Case Name** | **Primary Actor** | **Brief Description** |
| UC-01 | Submit Lost Item | Student | Allows students to report lost items with description, location, and optional image. |
| UC-02 | Submit Found Item | Student/Staff | Enables users to report found items including time, place, and a photo. |
| UC-03 | View Item Listings | Student/Staff | Users can browse all lost and found posts on the platform. |
| UC-04 | Search and Filter Items | Student/Staff | Users filter/search listings by keywords, category, status, and date. |
| UC-05 | Claim Item | Student | Users request to claim an item by submitting identification or justification details. |
| UC-06 | Approve/Reject Listings | Admin | Admins verify and moderate item reports before publishing them. |
| UC-07 | Handle Claim Requests | Admin | Admins review and validate claim submissions before item handover. |
| UC-08 | Send Notifications | System | Automatically sends alerts (Email/SMS) on post approval, match, or claim status. |
| UC-09 | Login via Firebase | Student/Admin | Secure login via verified university credentials using Firebase Authentication. |
| UC-10 | Update User Profile | Student/Staff | Allows users to update personal information such as email, phone, or preferences. |
| UC-11 | Upload Item Image | Student/Staff | Upload and preview images while submitting lost or found reports. |
| UC-12 | View Notification History | Student | Displays a log of all notifications received for item status and updates. |
| UC-13 | Access Admin Dashboard | Admin | Admins access a dashboard to manage users, reports, and system settings. |
| UC-14 | Generate Audit Reports | Admin | Admin generates item logs and claim history for reporting or accountability. |
| UC-15 | Data Backup & Sync | IT System | Ensures secure backup of user data, item reports, and logs to cloud storage. |

## System use case diagram



**Figure 3.1**

Figure 1

**: Use case diagram**

## Extended use cases

Table 5 Login use case

This use case allows users (students, staff, and admin) to securely log into the Lost & Found web application using Firebase Authentication and access role-specific features.

|  |  |
| --- | --- |
| **Use Case ID** | UC-001 |
| **Use Case Name** | Login |
| **Created By** | Nittasha Zulfiqar |
| **Last Updated By** | Maheen Hussain |
| **Date Created** | 2025-5-14 |
| **Last Revision Date** | 2025-5-16 |
| **Actors** | Student (Primary), Staff, Admin (Secondary) |
| **Description** | This use case allows verified users (students, staff, or admins) to securely access their role-specific dashboard using Firebase Authentication with institutional email. |
| **Trigger** | User clicks on the “Login” button on the web or mobile app interface. |
| **Preconditions** | 1. User must have a valid institutional email registered in the system. 2. Internet connectivity must be available. 3. Firebase Authentication service must be active. |
| **Post Conditions** | 1. User is redirected to their role-specific dashboard. 2. Login event is logged for security and audit purposes. |
| **Normal Flow** | 1. User opens the Lost & Found Web App (web or mobile). 2. User clicks on the **“Login”** button. 3. The system displays input fields for email and password. 4. User enters valid credentials and submits. 5. Firebase validates credentials with stored records. 6. System determines the user’s role (Student, Staff, Admin). 7. User is redirected to their respective dashboard with personal interface. |
| **Alternative Flows** | **Alternative Flow 1 – Forgot Password**  3a. In **step 3** of the Normal Flow, if the user clicks “Forgot Password”:   1. System redirects to the password recovery page. 2. User enters their registered email. 3. System sends a password reset link via email. 4. User resets password using the link. 5. User logs in again. 6. Use Case resumes from step 4 of Normal Flow. |
| **Exceptions** | **Exception 1 – Invalid Credentials**  4a. In step 4, if the user enters incorrect login information:   1. System displays error message: *“Invalid email or password.”* 2. User retries. 3. Resumes from Step 3 of Normal Flow.   **Exception 2 – Excessive Login Attempts**  4b. If user exceeds maximum attempts (e.g., 5 tries):   1. System locks the account temporarily. 2. System notifies user of lock and prompts to reset password. 3. Resumes after password reset or lock duration expires. |
| **Includes** | * Sign up with Gmail * Sign up with Facebook * Institutional Email Registration |
| **Frequency of Use** | * Students: Daily * Staff: Weekly or as needed * Admin: Frequently for platform oversight and moderation |
| **Special Requirements** | * Secure HTTPS transmission * Firebase Authentication integration * Automatic session timeout on inactivity * Role-based UI redirection |
| **Assumptions** | All users have active institutional accounts and correct login details. Devices support the latest browsers or app versions. Firebase authentication infrastructure is functional. |
| **Notes and Issues** | **Notes:**   * System should log all login attempts with timestamps and IP for security auditing. * Ensure login screen is WCAG 2.1 accessible (screen reader support, color contrast).   **Issues:**   * Need to finalize max login attempts before lockout. * Decision required on enforcing mandatory Two-Factor Authentication (2FA) for Admins. |

Table 6 Submit Lost Item

This use case enables students to report items they have lost on campus by providing descriptions, categories, and optional images.

|  |  |
| --- | --- |
| **Use Case ID** | UC-002 |
| **Use Case Name** | Submit Lost Item |
| **Created By** | Wasaf Zahra |
| **Last Updated By** | Maheen Hussain |
| **Date Created** | 2025-05-2 |
| **Last Revision Date** | 2025-05-5 |
| **Actors** | Student (Primary), Admin (Secondary) |
| **Description** | This use case allows a student to report a lost item by entering relevant information and optionally attaching an image. |
| **Trigger** | User clicks the "Submit Lost Item" button on their dashboard. |
| **Preconditions** | 1. User is logged in. 2.The lost item form is accessible. 3. User has required item information. |
| **Post Conditions** | 1. The item is submitted to the system as a "Lost" entry. 2. Admin is notified for moderation. |
| **Normal Flow** | 1. Student logs into the application. 2. Student selects "Submit Lost Item" from the dashboard. 3. System displays the form. 4. Student fills in details. 5. Student optionally uploads an image. 6. Student submits the form. 7. System stores the record and notifies admin. |
| **Alternative Flows** | **Alternative Flow 1 – Image too large** 5a. If image exceeds 2MB:  1. System shows error.  2. Student uploads a smaller image.  3. Resumes at Step 6. |
| **Exceptions** | **Exception 1 – Incomplete Form** 4a. Required fields are empty:  1. System shows validation error.  2. Student fills missing fields.  3. Resumes at Step 5. **Exception 2 – Upload failure** 5a. Image fails to upload:  1. System shows retry.  2. Student retries or submits without image. |
| **Includes** | Login, Image Upload |
| **Frequency of Use** | Multiple times per day, especially during midterms/exams |
| **Special Requirements** | Form should autosave inputs. Image max size = 2MB. Fields must support mobile input. |
| **Assumptions** | Student knows the item details and location where it was lost. |
| **Notes and Issues** | • Ensure special characters are validated. • Add preview option for uploaded image. • Future update: support voice entry. |

Table 7: Submit Found Item

This use case allows students or staff to report items they’ve found, attaching optional images and specifying the found location to help return the item to its owner.

|  |  |
| --- | --- |
| **Use Case ID** | UC-003 |
| **Use Case Name** | Submit Found Item |
| **Created By** | Nittasha Zulfiqar |
| **Last Updated By** | Wasaf Zahra |
| **Date Created** | 2025-05-12 |
| **Last Revision Date** | 2025-05-14 |
| **Actors** | Staff (Primary), Student (Secondary), Admin (Secondary) |
| **Description** | This use case enables users to report an item they have found on campus, including item details, location, and optional image evidence. |
| **Trigger** | User clicks the "Submit Found Item" button on their dashboard or navigation menu. |
| **Preconditions** | 1. User must be logged in. 2. The found item form must be accessible. 3. User must have possession or full knowledge of the found item. |
| **Post Conditions** | 1. The found item is listed on the platform and tagged as “Found”. 2. The record is sent to the admin for moderation before being shown to other users. |
| **Normal Flow** | 1. User logs into the system. 2. Navigates to “Submit Found Item”. 3. Fills in item name, description, found location, and date. 4. Optionally uploads a photo. 5. Clicks the Submit button. 6. System stores the entry and notifies the admin for review. 7. Confirmation is shown to user. |
| **Alternative Flows** | **Alternative Flow 1 – No Image Provided** 4a. If user skips uploading image: 1. System uses default placeholder image. 2. Continues to Step 5. |
| **Exceptions** | **Exception 1 – Required Fields Missing** 3a. User leaves mandatory fields empty: 1. System shows error. 2. Highlights required fields. 3. Resumes from Step 3.  Exception 2 – Image Upload Fails 4a. Image file is not uploaded: 1. System shows retry or cancel option. 2. User retries or continues without image. |
| **Includes** | Login, Image Upload, Submit Confirmation |
| **Frequency of Use** | Several times per day depending on lost/found activity on campus. |
| **Special Requirements** | Form must be mobile-responsive. Image size should be limited to 2MB and allow only PNG/JPG. Show preview for uploaded images. |
| **Assumptions** | User has genuinely found the item and wants to return it. Admin will review and publish the post after validation. |
| **Notes and Issues** | • Allow description-based matching for admin validation. • Need fraud prevention logic (e.g. posting found item without actual possession). • Add optional contact number field. |

**Table 3.7: Claim Item**

This use case permits students to claim a listed found item by submitting a justification and, if necessary, additional proof of ownership.

|  |  |
| --- | --- |
| **Use Case ID** | UC-004 |
| **Use Case Name** | Claim Item |
| **Created By** | Maheen Hussain |
| **Last Updated By** | Nittasha Zulfiqar |
| **Date Created** | 2025-05-11 |
| **Last Revision Date** | 2025-05-12 |
| **Actors** | Student (Primary), Admin (Secondary) |
| **Description** | This use case allows a student to submit a claim request for a listed found item by providing necessary justification or identity proof. |
| **Trigger** | Student finds a matching found item and clicks on the "Claim" button. |
| **Preconditions** | 1. User must be logged in. 2. The item is visible and marked as "Found". 3. The item is not already claimed or under review. |
| **Post Conditions** | 1. Claim request is submitted successfully. 2. Admin is notified and claim enters review state. |
| **Normal Flow** | 1. Student logs into the application. 2. Browses or searches for the lost item. 3. Clicks on "Claim". 4. System displays claim form. 5. Student enters reason/justification and optional proof. 6. Clicks submit. 7. System stores the claim and notifies admin. 8. Status marked "Under Review". |
| **Alternative Flows** | Alternative Flow 1 – Attach Additional Evidence 5a. If student wants to provide more proof: 1. System allows multiple file uploads. 2. Resumes from Step 6. |
| **Exceptions** | Exception 1 – Incomplete Form 5a. Required fields left blank: 1. System shows error message. 2. Highlights required fields. 3. Resumes from Step 5.  Exception 2 – Claim Already Exists 2a. Another claim is already submitted: 1. System disables claim button. 2. Displays status "Already under review". |
| **Includes** | Login, Search/Filter Listings, View Item Details |
| **Frequency of Use** | Often used during busy periods (midterms, finals), typically 10–20 times per day. |
| **Special Requirements** | File size limit for attachments: 5MB Acceptable formats: JPG, PDF Form should auto-save drafts |
| **Assumptions** | Student is truthful and submits genuine proof of ownership. Item was correctly identified by the student. |
| **Notes and Issues** | • Admin decision may require manual verification by security or additional emails. • Delay in claim processing may frustrate users. • Future improvement: real-time chat with admin for claims. |

Table 8 Approve Listings

This use case enables the admin to review, approve, or reject newly submitted lost or found item posts before they appear publicly on the platform.

|  |  |
| --- | --- |
| **Use Case ID** | UC-005 |
| **Use Case Name** | Approve Listings |
| **Created By** | Wasaf Zahra |
| **Last Updated By** | Maheen Hussain |
| **Date Created** | 2025-05-9 |
| **Last Revision Date** | 2025-05-10 |
| **Actors** | Admin (Primary) |
| **Description** | This use case allows the admin to review, approve, or reject newly submitted lost or found item listings before they are made public. |
| **Trigger** | A new item listing is submitted by a user and enters the pending approval queue. |
| **Preconditions** | 1. Admin must be logged in. 2. There are pending listings in the moderation queue. |
| **Post Conditions** | 1. Listing is either published or rejected. 2. The item poster is notified. |
| **Normal Flow** | 1. Admin logs into the system. 2. Navigates to the Moderation Panel. 3. Views the list of pending listings. 4. Clicks on an entry to review details. 5. Approves or rejects the listing. 6. System updates the listing status. 7. Poster is notified of approval/rejection. |
| **Alternative Flows** | **Alternative Flow 1 – Edit Listing Before Approval:** 5a. Admin finds typos or missing info: 1. Admin edits the listing. 2. Then approves it. |
| **Exceptions** | **Exception 1 – Session Timeout:** 2a. Admin is logged out while reviewing: 1. System asks to log in again. 2. Unsaved changes may be lost. **Exception 2 – No Pending Listings:** 3a. If no entries exist: 1. System shows message: 'No items pending review'. |
| **Includes** | Login, View Item Details, Notification |
| **Frequency of Use** | Daily or as often as new listings are submitted. |
| **Special Requirements** | Moderation queue should support filters (e.g., date, type). Changes must be audit-logged. |
| **Assumptions** | Admin acts in good faith and follows institutional moderation policies. |
| **Notes and Issues** | • Future enhancement: Auto-flag listings with suspicious patterns. • Add option to temporarily hide listings instead of permanent rejection. |

Table 9 Receive Notifications

This use case handles the delivery of real-time notifications (email/SMS/in-app) to users when their post status changes or an item match is found.

|  |  |
| --- | --- |
| **Use Case ID** | UC-006 |
| **Use Case Name** | Receive Notifications |
| **Created By** | Wasaf Zahra |
| **Last Updated By** | Nittasha Zulfiqar |
| **Date Created** | 2025-05-01 |
| **Last Revision Date** | 2025-05-03 |
| **Actors** | Student (Primary), Staff (Secondary), Admin (Secondary) |
| **Description** | This use case enables users to receive automatic system notifications for updates like post approval, matching items, or claim responses. |
| **Trigger** | System detects a relevant event like item approval or a match found. |
| **Preconditions** | 1. User is registered and logged in. 2. User has notifications enabled. 3. Relevant system events are triggered. |
| **Post Conditions** | 1. Notification is delivered to user via email/SMS or in-app alert. 2. Notification is logged. |
| **Normal Flow** | 1. Event is triggered (e.g., claim response or item matched). 2. System identifies affected user. 3. Prepares notification message. 4. Sends notification via selected method (email/SMS/in-app). 5. User receives and views notification. |
| **Alternative Flows** | **Alternative Flow 1 – Notification Channel Failure:** 4a. If email fails to send: 1. System retries with alternate channel (e.g., SMS). 2. Logs error. |
| **Exceptions** | **Exception 1 – User Not Reachable:** 3a. Email/SMS delivery fails: 1. System queues retry. 2. Marks as undelivered if retry fails. |
| **Includes** | User Preferences, Email/SMS Gateway, Event Handling |
| **Frequency of Use** | Triggered dynamically by system events; varies per user activity. |
| **Special Requirements** | Ensure timely delivery (<10s after event). Fallback mechanism must be enabled. |
| **Assumptions** | Users have provided valid contact details. Email/SMS services are operational. |
| **Notes and Issues** | • Add preference management so users can select notification type. • Consider push notifications for future mobile app integration. |

Table 10 Access Admin Dashboard

This use case enables admins to log in and access a comprehensive dashboard for managing listings, user activity, and system reports.

|  |  |
| --- | --- |
| **Use Case ID** | UC-007 |
| **Use Case Name** | Access Admin Dashboard |
| **Created By** | Maheen Hussain |
| **Last Updated By** | Nittasha Zulfiqar |
| **Date Created** | 2025-04-28 |
| **Last Revision Date** | 2025-04-30 |
| **Actors** | Admin (Primary) |
| **Description** | This use case allows an admin to log in and view the administrative dashboard for managing users, items, and system configurations. |
| **Trigger** | Admin clicks on the 'Dashboard' link after successful login. |
| **Preconditions** | 1. Admin has valid credentials. 2. Admin is logged in. 3. Admin account has dashboard access rights. |
| **Post Conditions** | 1. Admin can view system statistics and controls. 2. Admin can manage listings, claims, and settings. |
| **Normal Flow** | 1. Admin logs into the system. 2. Clicks on the 'Dashboard' link in navigation. 3. System verifies admin role and permissions. 4. Dashboard interface loads with all modules and statistics. 5. Admin interacts with sections as required (e.g., review users, view logs). |
| **Alternative Flows** | Alternative Flow 1 – Limited Access: 3a. Admin account has restricted privileges: 1. Only specific sections of the dashboard are shown. |
| **Exceptions** | Exception 1 – Unauthorized Access: 3a. User tries to access admin dashboard without admin role: 1. System displays 'Access Denied' message. 2. Redirects to home page. |
| **Includes** | Login, Role Verification, System Overview |
| **Frequency of Use** | Typically once per session; multiple times per day during active moderation. |
| **Special Requirements** | Dashboard should be secure and load under 2 seconds. Modules must be collapsible for mobile responsiveness. |
| **Assumptions** | Admin users are trained and familiar with dashboard features. |
| **Notes and Issues** | • Add option to export reports from dashboard. • Future enhancement: Audit trail visualizations for each admin action. |

Table 11 Update Profile

This use case allows students and staff to update their personal information such as contact details and notification preferences.

|  |  |
| --- | --- |
| **Use Case ID** | UC-008 |
| **Use Case Name** | Update Profile |
| **Created By** | Maheen Hussain |
| **Last Updated By** | Wasaf Zahra |
| **Date Created** | 2025-05-04 |
| **Last Revision Date** | 2025-05-06 |
| **Actors** | Student (Primary), Staff (Secondary) |
| **Description** | This use case allows users to update their personal information such as name, contact details, and notification preferences. |
| **Trigger** | User clicks the 'Profile Settings' or 'Edit Profile' button from the dashboard. |
| **Preconditions** | 1. User must be logged in. 2. User profile must already exist. |
| **Post Conditions** | 1. Profile information is updated and saved in the database. 2. Changes take effect immediately for future actions. |
| **Normal Flow** | 1. User logs into the system. 2. Navigates to 'Profile Settings'. 3. System loads current profile data. 4. User edits required fields. 5. Clicks 'Save Changes'. 6. System validates and stores new data. 7. Confirmation message displayed. |
| **Alternative Flows** | **Alternative Flow 1 – Partial Update:** 4a. User only updates one or two fields: 1. System only updates changed fields. |
| **Exceptions** | **Exception 1 – Invalid Input:** 4a. User inputs invalid data (e.g., letters in phone number): 1. System shows validation error. 2. Prompts user to correct errors. **Exception 2 – Database Error:** 6a. If database update fails: 1. System shows error. 2. Retains unsaved changes in form. |
| **Includes** | Login, Form Validation, Notification |
| **Frequency of Use** | Occasionally; typically used during first login or when contact info changes. |
| **Special Requirements** | Data must be encrypted during transfer. Form should auto-fill existing values. |
| **Assumptions** | User is willing to provide accurate and up-to-date contact details. |
| **Notes and Issues** | • Add option to change password from profile page. • Future update: profile picture upload support. |

Table 12 Generate Audit Reports

This use case allows the admin to generate detailed reports of user activities, item submissions, claims, and moderation history for transparency and record-keeping.

|  |  |
| --- | --- |
| **Use Case ID** | UC-009 |
| **Use Case Name** | Generate Audit Reports |
| **Created By** | Nittasha Zulfiqar |
| **Last Updated By** | Wasaf Zahra |
| **Date Created** | 2025-05-13 |
| **Last Revision Date** | 2025-05-14 |
| **Actors** | Admin (Primary), University Management (Secondary) |
| **Description** | This use case enables admins to generate audit logs or summary reports of platform activity including item submissions, claims, and moderation actions. |
| **Trigger** | Admin selects the 'Generate Report' option from the dashboard. |
| **Preconditions** | 1. Admin is logged in. 2. System has relevant data stored. 3. Reporting module is functional. |
| **Post Conditions** | 1. A downloadable report is created. 2. Report is stored temporarily for admin access. |
| **Normal Flow** | 1. Admin logs into the system. 2. Navigates to 'Reports' section. 3. Selects report type (e.g., daily, weekly). 4. Applies filters (date, category, user actions). 5. Clicks 'Generate Report'. 6. System processes data and creates report. 7. Report is made available for download. |
| **Alternative Flows** | **Alternative Flow 1 – Email Report:** 6a. Admin chooses email delivery: 1. System emails report link to admin’s registered email. |
| **Exceptions** | **Exception 1 – No Data Found:** 5a. Filters return no records: 1. System shows 'No records found' message. **Exception 2 – Report Generation Timeout:** 6a. Report takes too long: 1. System shows timeout message. 2. Asks admin to narrow filters. |
| **Includes** | Login, Filter Data, Download File |
| **Frequency of Use** | Weekly or monthly depending on reporting policies. |
| **Special Requirements** | Export in PDF/CSV format. Must support graphs and tables. |
| **Assumptions** | Data is accurately logged and timestamped. Admin has appropriate permissions. |
| **Notes and Issues** | • Add automation option to schedule regular reports. • Ensure secure access to sensitive data during report generation. |

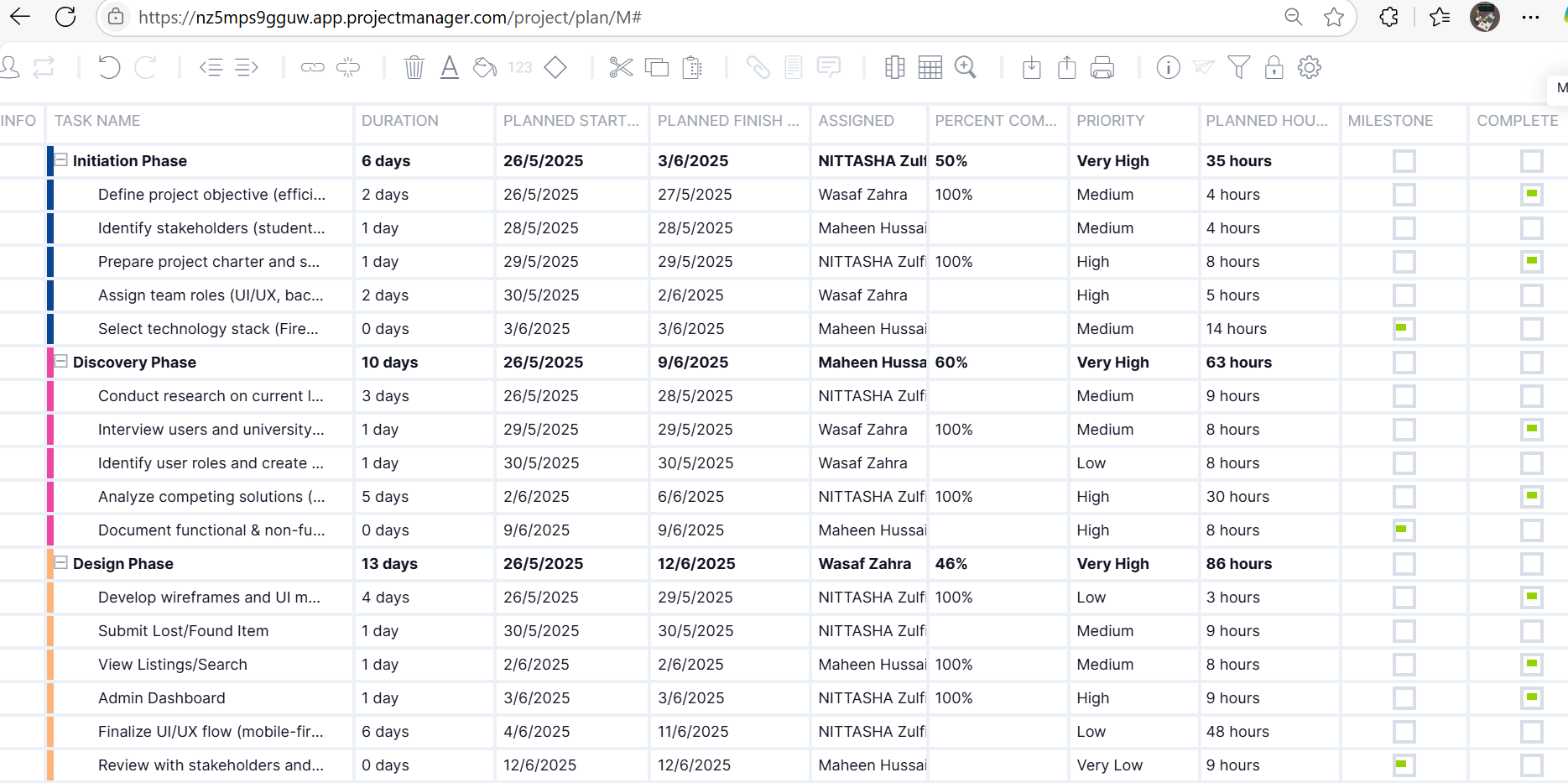
Table 13: Handle Claim Requests

This use case allows the admin to assess and process claim requests submitted by users, based on justification and evidence provided.

|  |  |
| --- | --- |
| **Use Case ID** | UC-010 |
| **Use Case Name** | Handle Claim Requests |
| **Created By** | Nittasha Zulfiqar |
| **Last Updated By** | Maheen Hussain |
| **Date Created** | 2025-05-12 |
| **Last Revision Date** | 2025-05-14 |
| **Actors** | Admin (Primary) |
| **Description** | This use case allows the Admin to review claim requests submitted by users and either approve or reject them. |
| **Trigger** | Admin navigates to the 'Claims' section upon receiving a new request notification. |
| **Preconditions** | 1. Admin is logged in. 2. There is at least one pending claim request. 3. Claim form has been filled and submitted by a student. |
| **Post Conditions** | 1. Claim status is updated in the database. 2. Claimant is notified about the decision. |
| **Normal Flow** | 1. Admin logs into the system. 2. Navigates to the 'Claim Requests' section. 3. Selects a pending request. 4. Reviews the claim details and attached proof. 5. Compares with item listing. 6. Approves or rejects the request. 7. System updates status and notifies claimant. |
| **Alternative Flows** | **Alternative Flow 1 – Incomplete Evidence:** 4a. Admin finds claim details insufficient: 1. Requests additional documentation. 2. Waits for student to re-submit. |
| **Exceptions** | **Exception 1 – No Pending Claims:** 2a. Admin opens empty claims queue: 1. System shows 'No pending requests'. **Exception 2 – System Timeout:** 5a. During processing system disconnects: 1. Admin is prompted to log in again. 2. Last session is restored. |
| **Includes** | Login, View Item Details, Notification |
| **Frequency of Use** | Used frequently, especially during peak times (exams, semester start). |
| **Special Requirements** | Decision must be logged with timestamp. Only verified admin should have this access. |
| **Assumptions** | Admin has full permissions and claim was made in good faith. |
| **Notes and Issues** | • Claim may involve coordination with campus security. • Future integration: auto-verification with item tags or QR codes. |

# Project Scheduling & Prototyping

## 4.1Work Breakdown Structure (WBS)



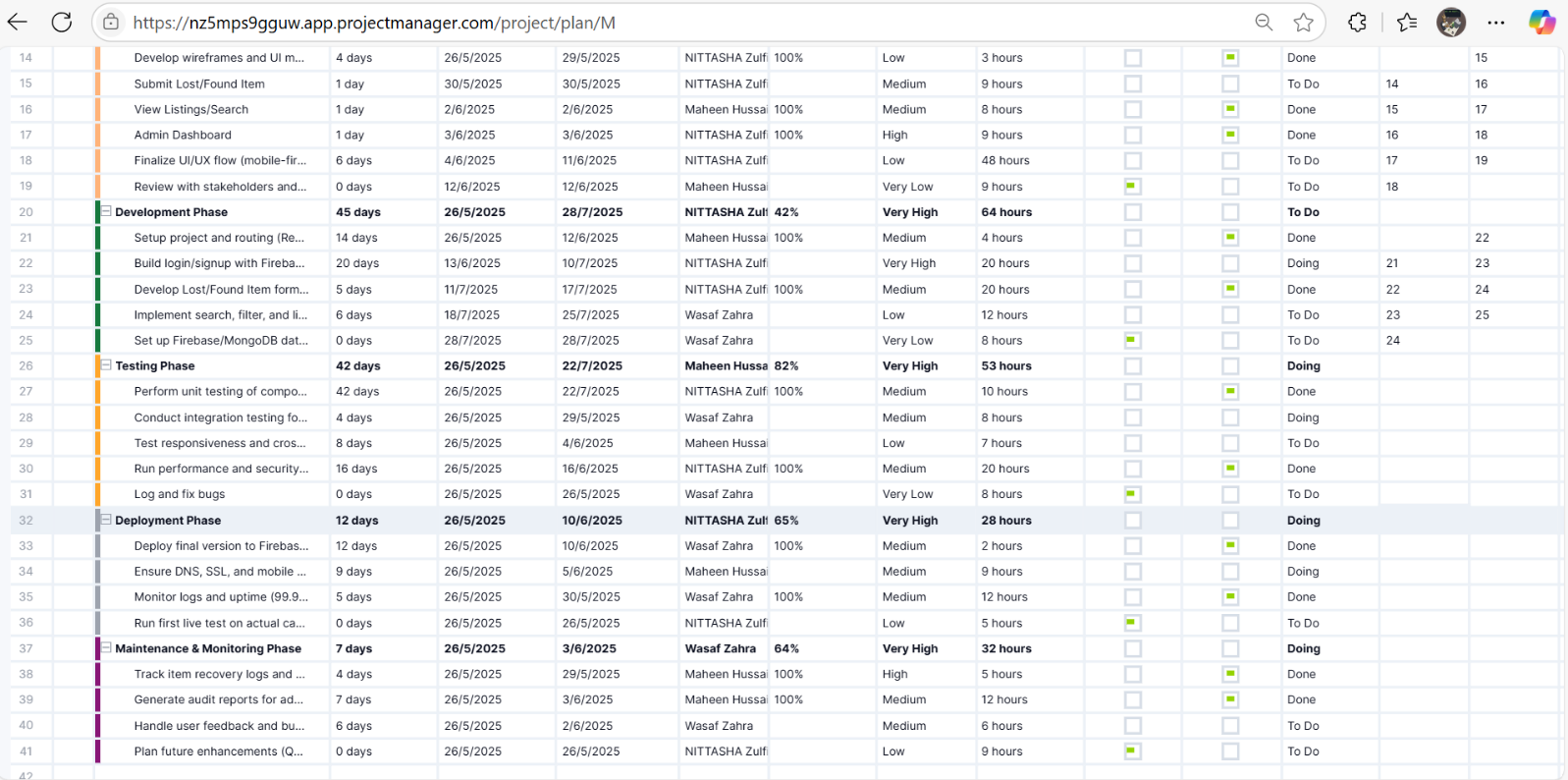
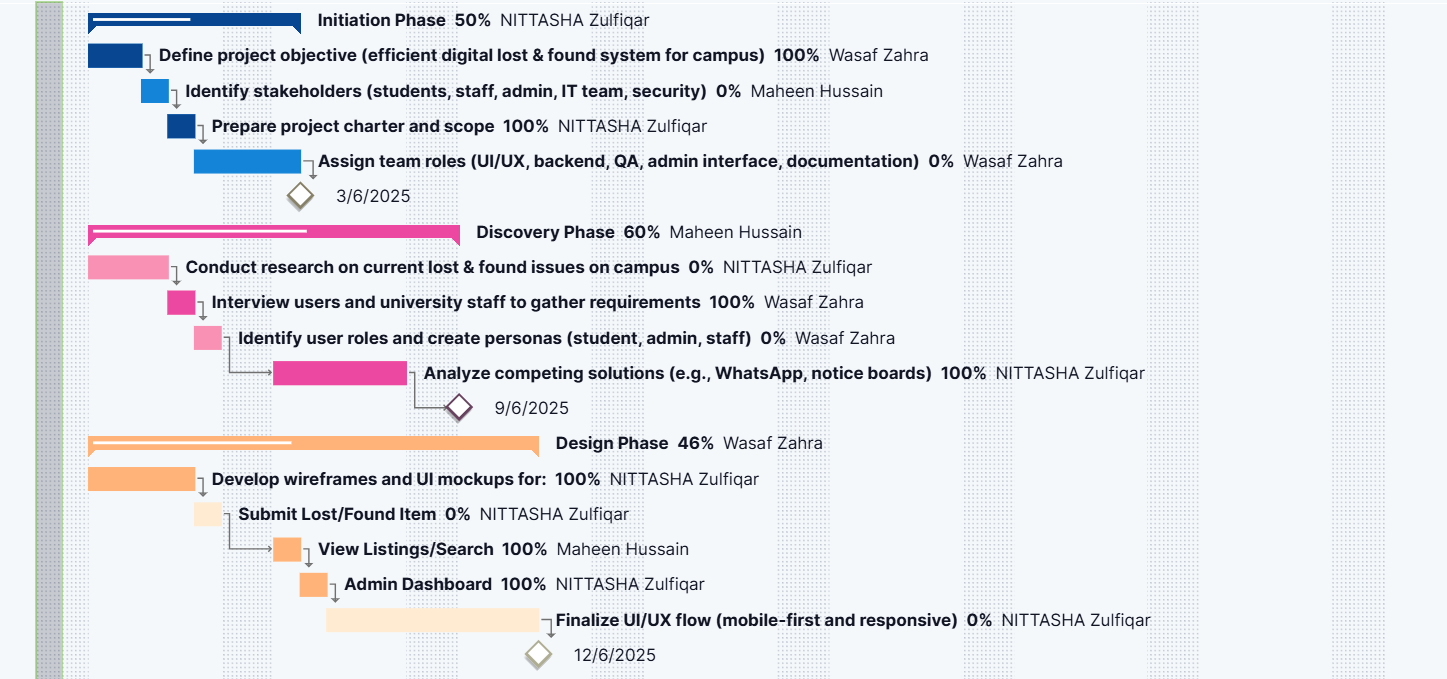


Figure 2 WBS

## Gantt-Chart for Work Breakdown Structure (WBS)



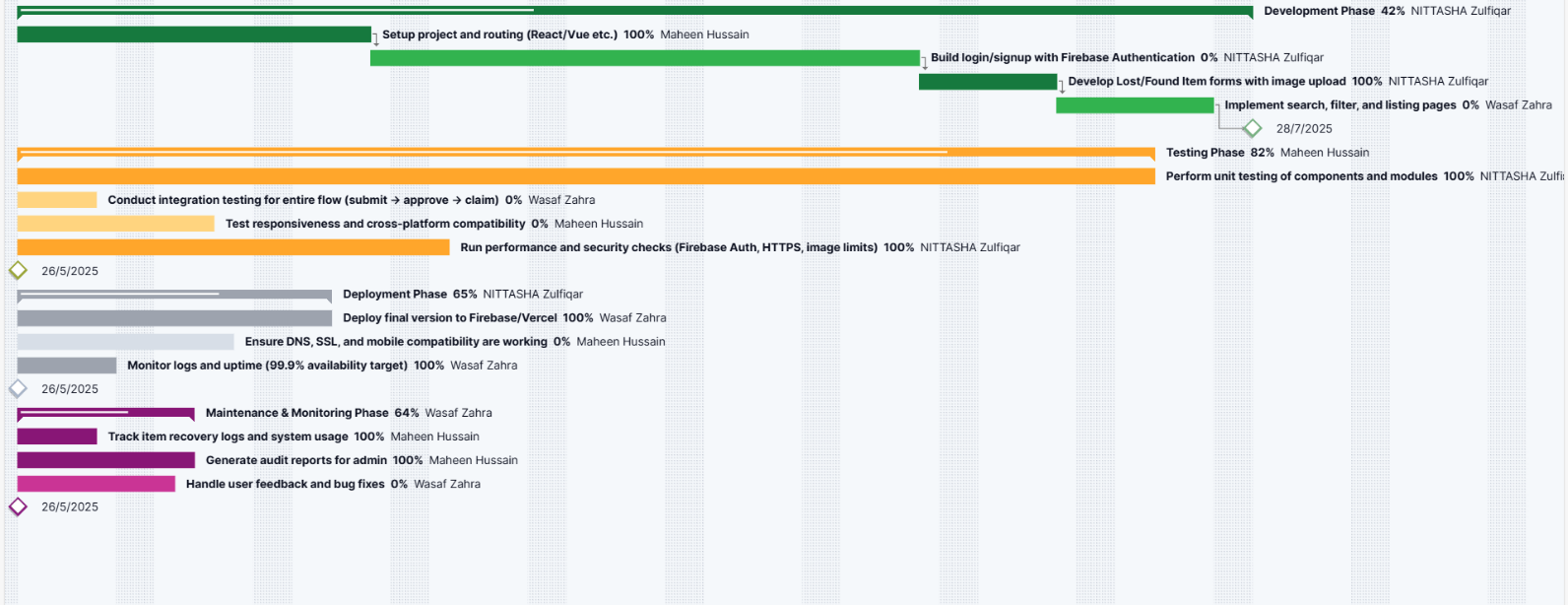
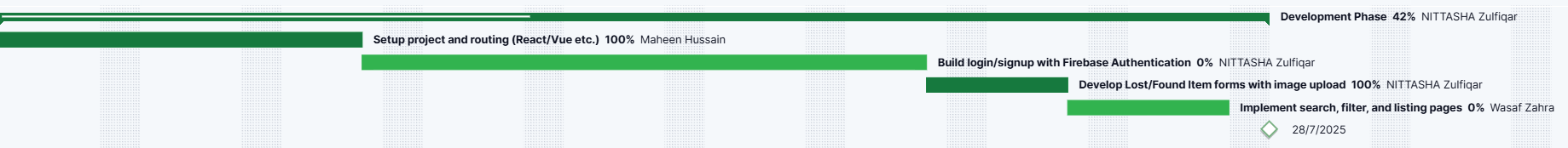


Figure 3: Gantt Chart

## Critical Path:



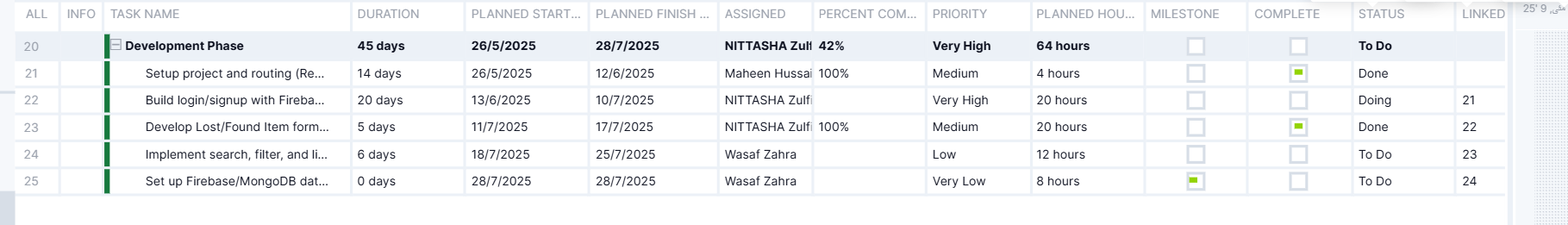


Figure 4: Critical path

## User interfaces (Prototypes/mock screens)



Figure 5 Prototype 1 (Login page)

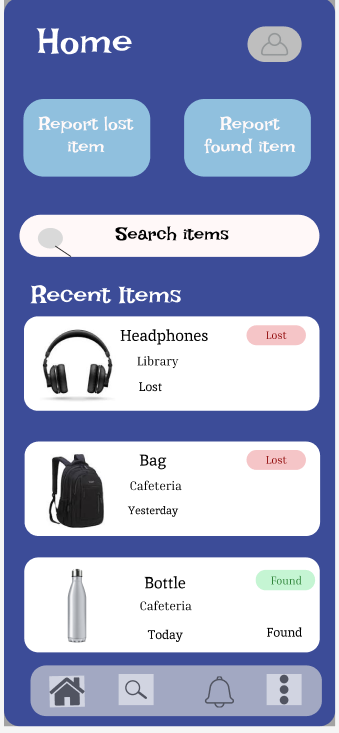


Figure 6 Prototype 2 (Home Screen)

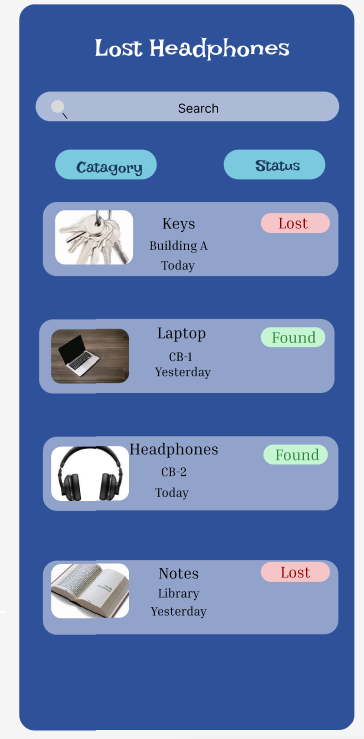


Figure 7Prototype 3 (Lost things)

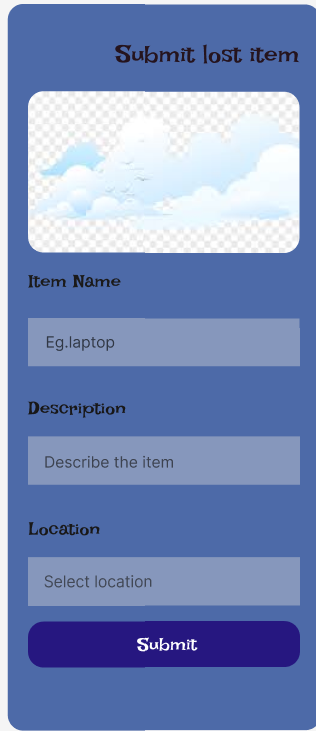


Figure 8 Prototype 4 (Submit item )

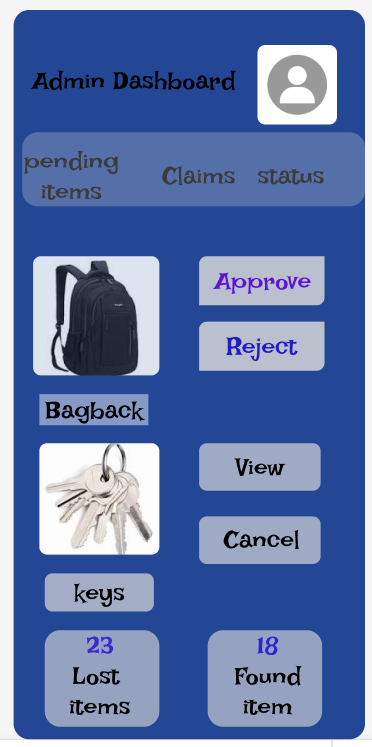
****

Figure 9 Prototype 5 (Admin Dashboard)

Link for figma:

<https://www.figma.com/design/h7vpWCdz7RrHgQBUxX26hv/Lost-and-found-web-app-for-campus?node-id=26-4&t=w0oBwtr5hA6FkUBM-1>

<https://www.figma.com/proto/h7vpWCdz7RrHgQBUxX26hv/Lost-and-found-web-app-for-campus?node-id=47-2&p=f&t=ovTbziddfcf4VAiL-1&scaling=scale-down&content-scaling=fixed&page-id=26%3A4>

# UML diagrams

## 5.1 Data Flow Diagram Level 0

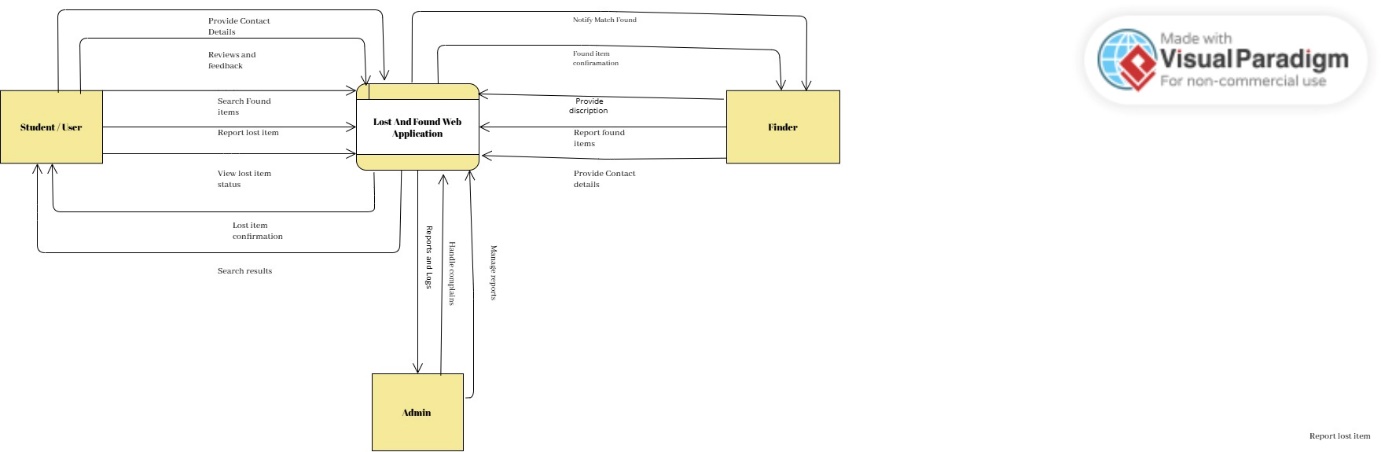


Figure 10 Data flow diagram of lost and found web

The diagram represents the workflow of a Lost and Found Web Application tailored for a university setting. It outlines the interactions between three primary users: the Student/User, the Finder, and the Admin. Students or users can report lost items, search through found items, view the status of their lost reports, and provide contact details and feedback. Finders, on the other hand, can report items they have found, describe them, submit their contact information, and receive notifications if a match is found with a lost item. The Admin oversees the entire system, managing reports, verifying item details, and ensuring the platform functions smoothly. This structured interaction helps streamline the process of reconnecting lost items with their rightful owners in an efficient and organized manner.

## SYSTEM DESIGN

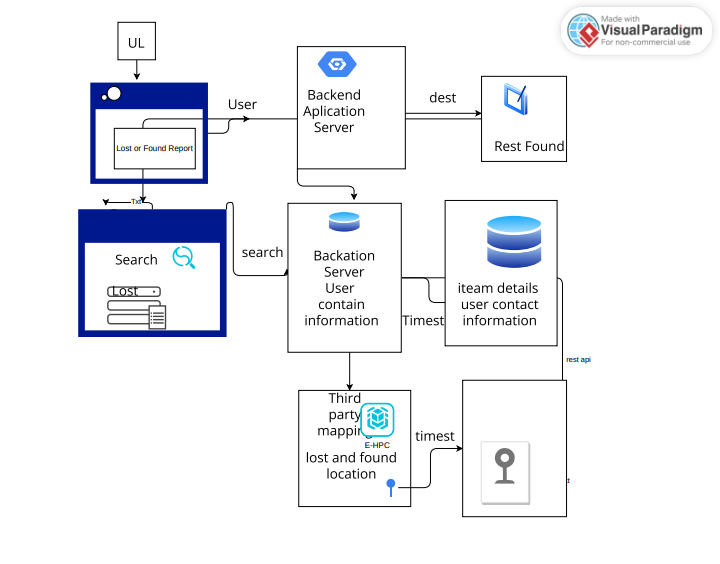


Figure 11 System architecture design for lost and found web

This figure represents the system architecture for a Lost and Found Web Application. The process begins when a user accesses the UI to submit a lost or found item report. This request is handled by the Backend Application Server, which communicates with the Rest Found service. The Backation Server stores user-related data and connects with the Item Details Database, which holds information about items and user contact details. Users can also search for items through the search interface, which queries the Backation Server. Additionally, third-party mapping services are integrated to show the location of lost and found items, enhancing item traceability with timestamped data.

## Class diagram

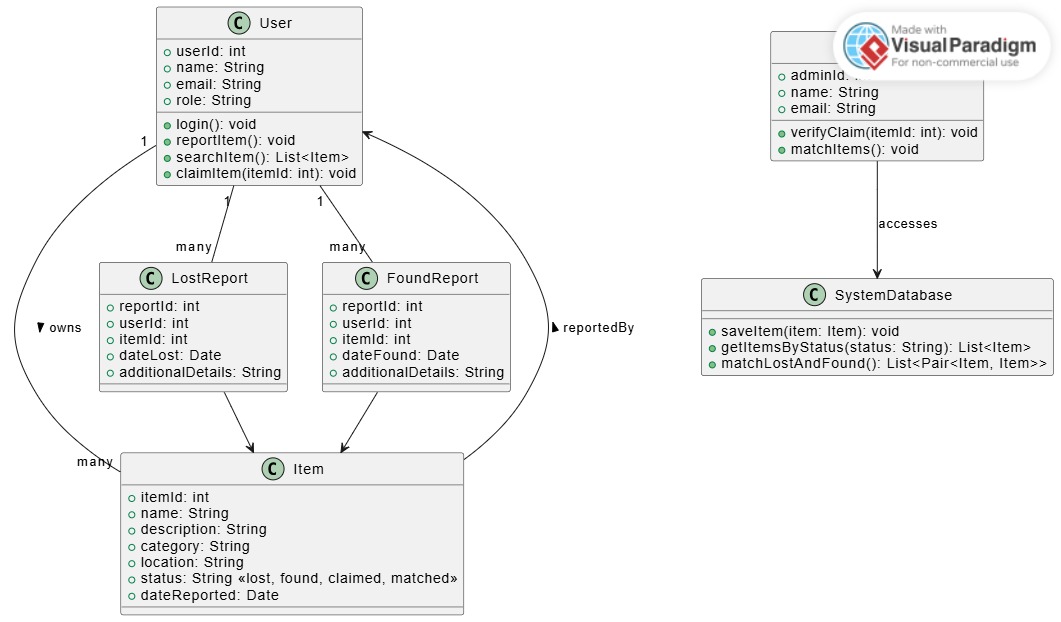


Figure 12 Class Diagram of lost and found web

This class diagram illustrates the object-oriented structure of a Lost and Found Management System. It includes key classes such as User, Admin, Item, LostReport, FoundReport, and SystemDatabase. Users can report lost or found items, search for items, and claim them. Each user can generate multiple lost or found reports, which are associated with individual Item entries containing detailed attributes like name, description, location, and status. The Admin class has the ability to verify claims and match lost items with found ones. The SystemDatabase serves as the backend repository, storing and retrieving item data based on their status and helping in matching lost and found items efficiently. This structure ensures organized data handling and streamlined functionality within the application.

## Sequence Diagrams

### 5.4.1 Sequence diagram 01

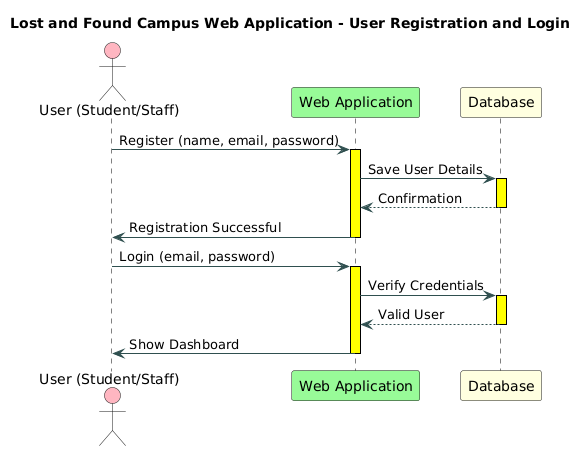


Figure 13 Sequence Diagram 01

It shows a user (student or staff) interacting with a web application, which communicates with a database. The registration process involves the user providing their name, email, and password, followed by the web application saving the details and sending a confirmation. For login, the user enters their email and password, and the web application verifies the credentials with the database, displaying the dashboard if valid. The diagram uses arrows and yellow-highlighted steps to clearly outline the workflow between the user, web application, and database.

### 5.4.2 Sequence Diagram 02

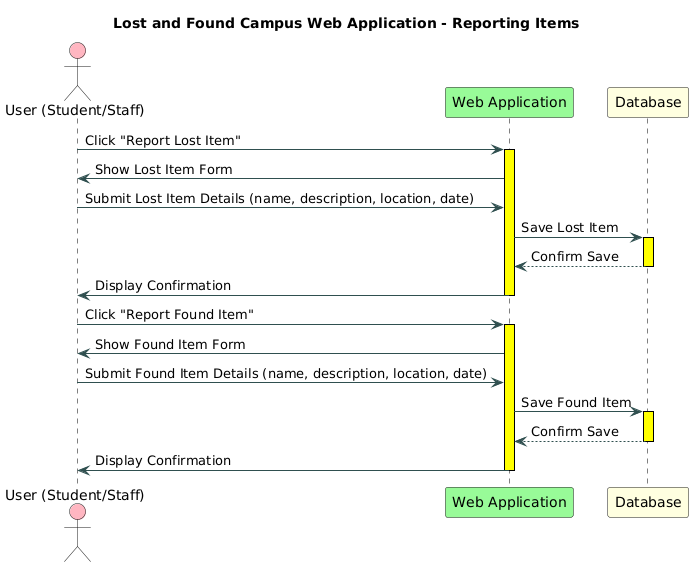


Figure 14 Sequence diagram 02

It illustrates a user (student or staff) interacting with a web application that connects to a database. The process begins with the user clicking "Report Lost Item" or "Report Found Item," filling out a form with details such as name, description, location, and date, and submitting it. The web application saves the details to the database and displays a confirmation to the user. This streamlined workflow ensures efficient reporting and management of lost and found items within the campus community.

### 5.4.3 Sequence Diagram 03

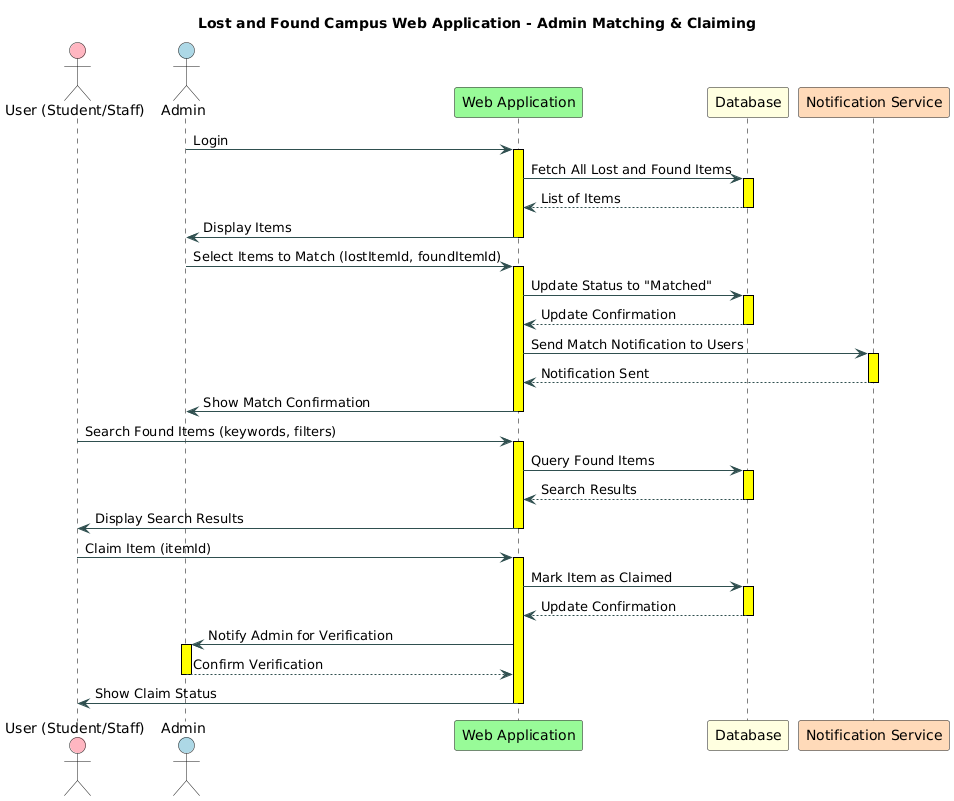


Figure 15 Sequence diagram 03

It depicts the interaction between users (students, staff, or admins), a web application, a database, and a notification service. The process begins with the admin fetching all lost and found items, displaying them to select matches, and updating the status to "Matched" in the database, followed by sending notifications to users. Additionally, users can search for found items using keywords or filters, view search results, and claim an item, which triggers an admin verification process with status updates and notifications. This workflow ensures efficient matching and claiming of lost and found items within the campus community.

# Testing

## Extended Test Cases

Table 14 Test Case: Submit Lost Item

This use case allows users to submit information about an item they have lost on campus. The system requires the user to be logged in and to provide all mandatory item details like name, description, and date lost.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID:** TC01   |  | | --- | |  | | | | | **Test Design By: Nittasha** | | | |
| **Test Module Name:**  Submit Lost Item | | | | **Test Design Date:** 15/07/2025 | | | |
| **Test Priority:** High | | | | **Test Executed By: Maheen** | | | |
| **Test Title/Name:**  To test submission of a lost item | | | | **Test Executed Date:** 16/07/2025 | | | |
| **Description:**  Ensure user can submit lost item details | | | | | | | |
| **Pre-Condition:**   1. User must be logged in. 2. Internet access must be active. | | | | | | | |
| **Dependencies:** None | | | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | |  | | --- | |  |   Open lost item form |  | Form is displayed | |  | Pass |  |
| **2** | Fill in item details | "Wallet", "Black, Leather” | Fields accept input | |  |  |
| **3** | Upload image (optional) | Image file uploaded | Image is previewed | |  |  |
| **4** | Submit form | Click submit | Lost item is recorded with status "Pending" | | Succesfully recorded |  |  |
| **Post-Condition:** Lost item entry added to the database for admin approval. | | | | | | | |

Table 15: Submit Found Item

This use case enables users to report any items they have found on campus. The item report should include a description and optional image, and is subject to admin approval.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID:** TC02 | | | | **Test Design By: Nittasha** | | | |
| **Test Module Name:** **Submit Found Item** | | | | **Test Design Date:** 15/07/2025 | | | |
| **Test Priority:** High | | | | **Test Executed By: Maheen** | | | |
| **Test Title/Name:**  To test submission of a found item | | | | **Test Executed Date:** 16/07/2025 | | | |
| **Description:** Ensure user can report a found item | | | | | | | |
| **Pre-Condition:**   1. User must be logged in. 2. Form must be accessible. | | | | | | | |
| **Dependencies:** None | | | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | Open Found Item Form |  | Form is displayed | |  | Pass |  |
| **2** | Enter item description | "Blue Notebook", "Eng Dept" | Inputs accepted | |  |  |
| **3** | Upload image (optional) | Upload image | Image previewed | |  |  |
| **4** | Submit form | Submit | Item saved to "Found Items" with pending | | Item saved succesfully |  |  |
| **Post-Condition:** Found item data stored and pending for approval. | | | | | | | |

Table 16 Test Case (Claim item)

**This use case lets users submit a claim for an item that has been reported as found. The user must provide a justification to support the claim, and be logged in to proceed.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID:** TC03 | | | | **Test Design By: Wasaf** | | | |
| **Test Module Name:** Claim item | | | | **Test Design Date:** 17/07/2025 | | | |
| **Test Priority:** Medium | | | | **Test Executed By: Nittasha** | | | |
| **Test Title/Name:**  To test the item claiming mechanism | | | | **Test Executed Date:** 18/07/2025 | | | |
| **Description:**  User submits a claim for an item | | | | | | | |
| **Pre-Condition:**   1. User must be logged in. 2. Item must exist in "Found" listings. | | | | | | | |
| **Dependencies:** None | | | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | Open found items page |  | Items listed | |  | Pass |  |
| **2** | Click "Claim" | Select item | Claim form opens | |  |  |
| **3** | Enter justification | “It’s my ID card” | Text accepted | |  |  |
| **4** | Submit claim | Submit | Claim request submitted for review | | Submited request |  |  |
| **Post-Condition:** Claim request stored for admin to verify and act. | | | | | | | |

Table 17 Test Case (Search and Filter Items)

**This use case allows users to search and filter items listed on the platform, either by keyword or category. This helps users easily find items they’ve lost or reported.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID:** TC04 | | | | **Test Design By:** Maheen | | | |
| **Test Module Name:** Search and Filter | | | | **Test Design Date:** 17/07/2025 | | | |
| **Test Priority:** Medium | | | | **Test Executed By: W**asaf | | | |
| **Test Title/Name:**  To test the search and filter options | | | | **Test Executed Date:** 18/07/2025 | | | |
| **Description:** Ensure user can search and filter | | | | | | | |
| **Pre-Condition:**   1. Listings must exist 2. User must be logged in | | | | | | | |
| **Dependencies:** None | | | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | Open list item |  | All items shown | |  | Pass |  |
| **2** | Search by keyword | Calculator" | Related items shown | |  |  |
| **3** | Filter by category | Electronics | Items of selected category displayed | | Item Displayed on screen |  |
| **Post-Condition:** Search/filter results shown accurately to user. | | | | | | | |

Table 18 Test Case (Login via Firebase)

**This use case manages user authentication via Firebase. Users must provide a valid email format and password to access the system securely.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID:** TC05 | | | | **Test Design By: Maheen** | | | |
| **Test Module Name:** Login | | | | **Test Design Date:** 17/07/2025 | | | |
| **Test Priority:** High | | | | **Test Executed By: Nittasha** | | | |
| **Test Title/Name:** To test Firebase login functionality | | | | **Test Executed Date:** 18/07/2025 | | | |
| **Description:**  Check if user can login using Firebase | | | | | | | |
| * **Pre-Condition:**  1. Valid Firebase credentials | | | | | | | |
| **Dependencies:** None | | | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | Open login form |  | |  |  | | --- | --- | | - | Form appears | | |  | Pass |  |
| **2** | Enter email/password | Email: abc@gmail.com  Password: | Inputs accepted | |  |  |
| **3** | Click on login Button |  | Redirect to user dashboard | |  |  |
| **Post-Condition:** User is logged in and session is active. | | | | | | | |

## Decision Table

### Decision coverage table ( Submit Lost Item)

Table 19

**Decision coverage table (Submit Lost Item)**

**This table describes the decision conditions and actions for the use case: Submit Lost Item.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** | **Rule 5** | **Rule 6** | **Rule 7** | **Rule 8** |
| User is logged in | T | T | F | F | T | T | F | F |
| Required fields filled | T | F | T | F | T | F | T | F |
| **Actions** |  |  |  |  |  |  |  |  |
| Lost item submitted | ✔ |  |  |  |  |  |  |  |
| Missing Information |  | ✔ |  |  |  |  |  |  |
| Login Required |  |  |  |  |  |  |  |  |

### 

### Decision coverage table (Submit Found Item)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** | **Rule 5** | **Rule 6** | **Rule 7** | **Rule 8** |
| User is logged in | T | T | F | F | T | T | F | F |
| Item description provided | T | F | T | F | T | F | T | F |
| **Actions** |  |  |  |  |  |  |  |  |
| Found item saved | ✔ |  |  |  |  |  |  |  |
| Missing Information |  | ✔ |  |  |  |  |  |  |
| Login Required |  |  |  |  |  |  |  |  |
| Submission Blocked |  |  |  |  |  |  |  |  |

Table 20: Decision coverage table (Found)

**This table describes the decision conditions and actions for the use case: Submit Found Item.**

### Decision coverage table ( Claim item)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** | **Rule 5** | **Rule 6** | **Rule 7** | **Rule 8** |
| User is logged in | T | T | F | F | T | T | F | F |
| Justification provided | T | F | T | F | T | F | T | F |
| **Actions** |  |  |  |  |  |  |  |  |
| Claim submitted | ✔ |  |  |  |  |  |  |  |
| Missing Justification |  | ✔ |  |  |  |  |  |  |
| Login Required |  |  |  |  |  |  |  |  |
| Claim Blocked |  |  |  |  |  |  |  |  |

Table 21 Decision coverage table (Claim)

**This table describes the decision conditions and actions for the use case: Claim Item**.

### 

### Decision coverage table (Search and Filter Items)

Table 22 Decision coverage table (Search and Filter Item)

**This table describes the decision conditions and actions for the use case:. Search and Filter Items**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** | **Rule 5** | **Rule 6** | **Rule 7** | **Rule 8** |
| Search keyword provided | T | T | F | F | T | T | F | F |
| Valid category selected | T | F | T | F | T | F | T | F |
| **Actions** |  |  |  |  |  |  |  |  |
| Filtered Results Shown | ✔ |  |  |  |  |  |  |  |
| Incomplete Filters |  | ✔ |  |  |  |  |  |  |
| No Matching Items |  |  |  |  |  |  |  |  |
| Prompt User Input |  |  |  |  |  |  |  |  |

### Decision coverage table (Login via firebase)

Table 23 Decision coverage table (Login via firebase)

**This table describes the decision conditions and actions for the use case:. Login via firebase**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** | **Rule 5** | **Rule 6** | **Rule 7** | **Rule 8** |
| Valid email format | T | T | F | F | T | T | F | F |
| Password provided | T | F | T | F | T | F | T | F |
| **Actions** |  |  |  |  |  |  |  |  |
| Login Successful | ✔ |  |  |  |  |  |  |  |
| Password Missing |  | ✔ |  |  |  |  |  |  |
| Email Invalid |  |  |  |  |  |  |  |  |
| Login Blocked |  |  |  |  |  |  |  |  |

**LINK OF GITHUB**

[Maheen-eng526/SE-project](https://github.com/Maheen-eng526/SE-project)