**1. Introduction**

The objective of this User Requirements Document (URD) is to obtain agreement with all technical advisers & developers in Mobile App Directory team regarding the qualitative and quantitative characteristics of a proposed system. This URD is intended to unambiguously communicate our understanding and expectations for the “Lost-and-Found” (LAF) Android Application to technical advisers, as well as among ourselves. To this end, this URD avoids technical implementation language, and restricts notations used to those that express functionality from users’ viewpoints.

**1.1 Product**

The Lost & Found Mobile App is a comprehensive solution designed to help users report and recover lost and found items efficiently. This mobile application allows users to easily report items they have lost or found, search for items, and communicate with others to facilitate the return of lost possessions. The app leverages location-based services, image uploads, and real-time notifications to enhance the user experience and increase the chances of recovering lost items. Available on both iOS and Android platforms, the Lost & Found Mobile App aims to streamline the process of reconnecting people with their belongings through a user-friendly and secure interface.

**1.2 Scope**

The Android application “Lost-and-Found” system provides a platform for both users who lose properties searching the finder and users who find properties searching the owner. If a match is confirmed by the system, a link between the owner and the finder will be created by the system after a mutual identification authentication is passed. The Lost & Found Mobile App project encompasses the development, deployment, and maintenance of a mobile application designed to assist users in reporting and recovering lost and found items. The scope of this project includes the following key areas:

* User Interface and Experience:

- Develop a user-friendly and intuitive interface for both iOS and Android platforms.

- Implement design elements that facilitate easy navigation and interaction.

- Ensure accessibility features to accommodate users with disabilities.

* User Registration and Authentication:

- Allow users to register and create profiles using email, social media accounts, or phone numbers.

- Implement secure login mechanisms and password recovery options.

- Ensure user data protection and privacy compliance.

* Lost Item Reporting:

- Enable users to report lost items by providing detailed descriptions, uploading images, and specifying the last known location.

- Include features for categorizing items (e.g., electronics, clothing, pets).

- Store and manage lost item reports in a centralized database.

* Found Item Reporting:

- Allow users to report found items with similar details as lost item reporting.

- Provide options to upload images and specify the found location.

- Store and manage found item reports in the database.

* Search Functionality:

- Implement advanced search features that allow users to find lost or found items using keywords, categories, and location filters.

- Display search results with detailed information and images.

- Include sorting and filtering options to refine search results.

* Notifications:

- Integrate push notification services to alert users about potential matches for lost and found items.

- Allow users to customize notification preferences.

- Ensure timely and reliable delivery of notifications.

* Messaging System:

- Develop a secure messaging system for users to communicate about lost and found items.

- Include features such as message notifications, history, and attachment support.

- Ensure data privacy and security in all communications.

* Administrative Functions:

- Provide tools for administrators to manage user accounts, review reports, and moderate content.

- Implement reporting and analytics features to monitor app usage and performance.

- Ensure system maintenance and regular updates.

* Data Security and Privacy:

- Implement robust security measures to protect user data, including encryption and secure communication protocols.

- Comply with relevant data protection regulations and standards (e.g., GDPR, CCPA).

- Regularly audit and update security practices to address emerging threats.

* Deployment and Maintenance:

- Plan and execute the deployment of the app on iOS App Store and Google Play Store.

- Provide user support and handle feedback and bug reports.

- Regularly update the app to add new features, fix bugs, and improve performance.

* Marketing and Outreach:

- Develop a marketing strategy to promote the app and attract users.

- Utilize social media, online advertising, and partnerships to increase visibility.

- Gather user feedback to continuously improve the app.

By covering these areas, the Lost & Found Mobile App project aims to deliver a comprehensive and reliable solution for managing lost and found items, ultimately enhancing the user experience and increasing the chances of reuniting lost items with their owners.

**1.3 Business Goals**

The primary business goal of the Lost & Found Mobile App is to attract and retain a large user base through targeted marketing campaigns, a seamless onboarding experience, and engaging features such as notifications and gamification. Revenue generation is another key objective, with plans to implement in-app advertisements, offer premium features and subscriptions, and allow businesses to promote sponsored listings. Building a supportive and active community is essential, achieved by creating user profiles, organizing community events, and encouraging user-generated content. Establishing brand recognition and trust is also crucial, ensuring data security, providing excellent customer support, and maintaining transparency. Operational efficiency will be optimized through scalable infrastructure, automation of routine tasks, and real-time monitoring. The app aims to expand into new markets by localizing for different regions, conducting market research, and forming strategic partnerships. Customer satisfaction is a priority, with a focus on collecting user feedback, continuously improving the app, and offering personalized experiences. Sustainable growth will be pursued by diversifying revenue streams, reinvesting profits, and building strong relationships with stakeholders. Lastly, the app will stay ahead of the competition through continuous innovation, investing in research and development, updating based on user feedback, and analyzing competitors.

* User Acquisition and Retention

**Goal**: Attract and retain a large user base.

**Strategies:**

Targeted Marketing Campaigns: Utilize social media platforms, search engine marketing, and partnerships with influencers to reach potential users. Offer promotions and referral bonuses to incentivize new sign-ups.

Onboarding Experience: Develop a simple and engaging onboarding process that guides new users through the app’s features. Offer tutorial videos and interactive guides.

User Engagement: Send regular notifications and updates about new features, potential matches for lost items, and community events. Implement gamification elements such as badges and rewards for active users.

**Metrics:**

- Number of new sign-ups per month.

- User retention rates.

- Daily active users (DAU) and monthly active users (MAU).

* Revenue Generation
* **Goal**: Develop sustainable revenue streams.

**Strategies:**

In-App Advertisements: Partner with relevant brands and businesses to display targeted ads within the app.

Premium Features and Subscriptions: Offer advanced search filters, priority listing, ad-free experience, and enhanced notification options as part of a subscription model.

Sponsored Listings*:* Allow businesses to promote their lost and found services or products within the app.

**Metrics:**

- Monthly and annual revenue.

- Number of premium subscribers.

- Ad click-through rates and revenue.

* Community Building

**Goal**: Foster a supportive and active user community.

**Strategies:**

User Profiles and Social Features: Allow users to create detailed profiles, follow other users, and participate in community forums or groups.

Community Events: Organize virtual events, challenges, and contests to encourage user interaction and engagement.

User-Generated Content: Enable users to share tips, success stories, and advice on recovering lost items.

**Metrics:**

- User interaction rates (comments, likes, shares).

- Participation in community events and challenges.

- User feedback and satisfaction scores.

* Brand Recognition and Trust

**Goal**: Establish a trusted and recognized brand.

**Strategies**:

Data Security and Privacy: Implement robust security measures to protect user data, including end-to-end encryption and compliance with data protection regulations.

Customer Support: Provide timely and effective customer support through multiple channels, including in-app chat, email, and social media.

Transparency: Regularly communicate with users about app updates, security practices, and any changes to terms and conditions.

**Metrics:**

- User trust ratings and reviews.

- Number of resolved customer support queries.

- Brand mentions and media coverage.

* Operational Efficiency

**Goal**: Optimize internal processes for smooth operations.

**Strategies:**

Scalable Infrastructure: Use cloud-based solutions to ensure the app can handle increasing user load without performance degradation.

Automation: Automate routine tasks such as report moderation, user verification, and notification delivery to reduce manual effort.

Monitoring and Analytics: Implement real-time monitoring and analytics to track app performance, detect issues early, and make data-driven decisions.

**Metrics:**

- System uptime and reliability.

- Response times for automated processes.

- Efficiency improvements from automation.

* Market Expansion

**Goal**: Expand the app’s reach to new markets.

**Strategies:**

Localization: Adapt the app for different regions by supporting multiple languages and local currencies.

Market Research: Conduct research to identify potential growth areas and tailor marketing strategies accordingly.

Strategic Partnerships: Collaborate with local businesses, universities, and organizations to promote the app and reach new user groups.

**Metrics:**

- Number of new users from different regions.

- User engagement in targeted markets.

- Partnerships and collaborations formed.

* Customer Satisfaction

**Goal**: Achieve high levels of customer satisfaction and loyalty.

**Strategies:**

User Feedback: Regularly collect and analyze user feedback to identify areas for improvement and new feature requests.

Continuous Improvement: Implement a continuous improvement process to update the app based on user feedback and market trends.

Personalization: Offer personalized experiences and recommendations based on user behavior and preferences.

**Metrics:**

- Customer satisfaction scores (e.g., NPS, CSAT).

- User feedback volume and sentiment.

- Retention rates of satisfied users.

* Sustainable Growth

**Goal**: Ensure long-term sustainability and growth.

**Strategies:**

*Revenue Diversification*: Explore multiple revenue streams, such as partnerships, ads, subscriptions, and sponsored content.

*Reinvestment*: Reinvest profits into app development, marketing, and user acquisition to fuel growth.

*Stakeholder Engagement*: Build strong relationships with investors, partners, and key stakeholders to secure ongoing support and resources.

**Metrics:**

- Revenue growth rates.

- Investment and funding secured.

- Long-term financial projections.

* Innovation and Improvement

**Goal:** Stay ahead of the competition through continuous innovation.

**Strategies:**

Research and Development: Invest in R&D to explore new technologies and features that can enhance the app.

User-Centric Design: Continuously update the app’s design and functionality based on user feedback and emerging trends.

Competitive Analysis: Regularly analyze competitor apps to identify opportunities for differentiation and improvement.

**Metrics:**

- Number of new features released.

- User adoption rates of new features.

- Competitive positioning and market share.

By focusing on these detailed business goals and strategies, the Lost & Found Mobile App aims to establish itself as the premier platform for managing lost and found items, providing exceptional value to users while ensuring business growth and sustainability.

**1.4 Document Conventions**

This Software Requirements Specification (SRS) document adheres to specific standards and typographical conventions to ensure clarity, consistency, and ease of understanding for all readers. Adhering to these conventions is crucial for maintaining a professional and standardized approach throughout the document. The conventions are as follows:

* **Headings:**

Main Headings:

The main headings are styled in an 18-point, bold, Times New Roman font. This style is chosen to provide clear separation between major sections of the document, making it easier for readers to navigate and locate key sections quickly.

Subheadings:

Subheadings within each main section are presented in a 14-point, bold, Times New Roman font. This provides a hierarchical structure that helps in breaking down the content into manageable and easily identifiable parts.

* **Body Text:**

- The main body text of the document is written in an 11-point, regular Arial font. Arial is selected for its clean and modern appearance, which enhances readability, especially in lengthy documents. The use of a smaller font size for body text ensures that the document remains compact while maintaining legibility.

* **Emphasis and Special Formatting:**

Italicized Text:

Italicized text is used to represent external systems or important terms that require special attention. For example, names of external databases, systems, or critical terminologies within the context of the software are italicized to distinguish them from regular text.

Bold Text:

Bold text is utilized to highlight key terms and important concepts within the document. This makes it easier for readers to identify and focus on crucial information quickly.

* **Monospaced Text**:

Code Snippets:

Any code snippets included in the document are presented in a monospaced font. This font style mimics the appearance of text in a code editor, making it easier for developers to read and understand the code.

File Names, Commands, and API Endpoints:

Similarly, file names, system commands, and API endpoints are also shown in monospaced text to differentiate them from regular prose. This helps in avoiding confusion and ensures that these technical elements stand out.

* **Lists:**

Bulleted Lists:

Bulleted lists are used to present items or features in a non-sequential manner. This format is suitable for listing options, attributes, or any other information where the order is not important.

Numbered Lists:

Numbered lists are used for step-by-step instructions or sequences where the order of items is critical. This ensures that readers follow the steps in the correct order, which is essential for processes and procedures.

* **Tables and Figures:**

Tables:

Tables are employed to organize data into rows and columns, providing a clear and concise way to present complex information. Each table includes a title and, if necessary, a description to explain its contents.

Figures:

Diagrams, charts, and other figures are used to visually represent data and concepts. Each figure is labeled with a number and a caption to describe what it illustrates, aiding in comprehension.

By adhering to these document conventions, the SRS ensures a structured and professional presentation of information, facilitating better communication and understanding among stakeholders, developers, and users. These standards help in maintaining a consistent look and feel throughout the document, making it accessible and easy to navigate for all readers.

**1.5 References**

1. IEEE Standard 830-1998: IEEE Recommended Practice for Software Requirements Specifications. This standard provides guidelines for the structure and contents of SRS documents, ensuring that they are clear, complete, and verifiable.

2. ISO/IEC/IEEE 29148:2011: Systems and software engineering - Life cycle processes - Requirements engineering. This international standard offers comprehensive guidelines for requirements engineering, including the development of SRS documents.

3. "Software Requirements" by Karl E. Wiegers and Joy Beatty: this book provides in-depth insights into the best practices for gathering, documenting, and managing software requirements. It covers a wide range of techniques and tools used in the industry.

4. "Writing Effective Use Cases" by Alistair Cockburn: this book focuses on writing clear and effective use cases, which are an integral part of many SRS documents. It provides practical advice and templates for documenting use cases.

5. "The Elements of Style" by William Strunk Jr. and E.B. White: While not specific to software requirements, this classic book on writing provides timeless advice on clear and concise writing, which is essential for creating readable and effective SRS documents.

6. "The Unified Modeling Language User Guide" by Grady Booch, James Rumbaugh, and Ivar Jacobson: This guide to UML provides a comprehensive overview of the modeling language, which is often used to create diagrams and models within SRS documents.

7. "Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise" by Dean Leffingwell: This book provides insights into how agile methodologies can be applied to requirements gathering and documentation, offering a different perspective from traditional approaches.

8. "Managing Software Requirements: A Use Case Approach" by Dean Leffingwell and Don Widrig: This book emphasizes the use of use cases in managing software requirements and provides practical advice and examples for documenting them effectively.

9. Project Management Institute (PMI) Guidelines: PMI offers a range of guidelines and standards for project management, including aspects related to requirements management and documentation.

10. Company-Specific Standards and Guidelines: Many organizations have their own standards and guidelines for creating SRS documents, which may include specific templates, review processes, and documentation tools.

These references provide a solid foundation for understanding the principles and best practices involved in writing Software Requirements Specifications, ensuring that the document meets industry standards and effectively communicates the requirements to all stakeholders.

**2. Project Description**

**2.1 Overall Description**

The Lost & Found Mobile App is a comprehensive platform designed to assist users in reporting and recovering lost items. The app serves as a centralized hub where users can easily post details of lost and found items, browse through listings, and connect with others to facilitate the return of lost belongings. The primary purpose of the app is to reduce the frustration and inconvenience associated with lost items by providing a streamlined and efficient process for reporting and locating them. The platform leverages a user-friendly interface, advanced search filters, and community-driven support to ensure that users can quickly and easily find or return lost items.

The app includes several key features aimed at enhancing user experience and maximizing the chances of successful item recovery. Users can create detailed listings for lost and found items, including descriptions, photos, and location details. Advanced search functionalities allow users to filter listings by various criteria such as item type, date, and location, making it easier to find relevant matches. Social features like user profiles, messaging, and community forums foster a supportive environment where users can share tips, offer assistance, and celebrate successful recoveries.

To ensure broad accessibility, the app is designed to be compatible with various mobile operating systems, including iOS and Android. It also incorporates robust data security measures to protect user information and ensure privacy. The app’s scalable infrastructure can handle a growing user base and increasing activity, ensuring smooth and reliable performance even as usage expands.

The Lost & Found Mobile App aims to establish itself as a trusted and reliable platform in the community by prioritizing user trust and satisfaction. Regular updates and improvements are made based on user feedback to continually enhance the app’s functionality and user experience. By focusing on building strong user community and offering valuable features.

**2.1.1 Product Features**

The Lost & Found Android Application is designed to offer a comprehensive, user-friendly platform for individuals seeking to report, search for, and retrieve lost and found items. Key features of the application include robust user authentication mechanisms, allowing users to securely register, log in, and recover their passwords. Once registered, users can manage their profiles, including viewing and updating personal information. The core functionality revolves around item reporting, where users can report lost or found items by providing detailed information such as title, description, date, location, and images. This is complemented by a powerful search functionality, enabling users to search for items using keywords and filters, with results displayed in a clear and detailed manner.

The application also includes a secure messaging system that facilitates communication between users without exposing personal contact information, ensuring privacy and safety. Users receive notifications via email for important events such as successful item reports, new messages, and password reset requests. An admin panel allows administrators to manage reports and user accounts, ensuring the integrity and smooth operation of the platform. The design of the application is responsive, optimized for use on various devices, including smartphones and tablets, to provide a seamless user experience. Security is paramount, with data encryption and HTTPS protocol ensuring that all user data and communications are secure. Additionally, the application features an intuitive interface, making navigation easy and accessible for all users, and includes a feedback mechanism to continuously improve the user experience based on user input. These features collectively ensure that the Lost & Found Android Application is a reliable, secure, and efficient tool for users to report and find lost items.

The Lost & Found Android Application includes a comprehensive set of features designed to assist users in reporting, searching, and retrieving lost and found items efficiently. Below is a detailed list of the main features of the application:

* **User Authentication**

**Registration**

User Account Creation: Users can create an account by providing their name, email, and password.

Email Verification: Users receive an email with a verification link to validate their email address.

Secure Password Storage: Passwords are securely stored using encryption.

**Login**

User Login: Users can log in to their accounts using their registered email and password.

Session Management: Secure authentication mechanisms to maintain user sessions.

**Password Recovery**

Forgot Password: Users can reset their password if forgotten by providing their email address.

Password Reset Link: A password reset link is sent to the user's email.

* **User Profile**

**Profile Management**

View Profile: Users can view their profile information, including name, email, and contact details.

Edit Profile: Users can update their profile information.

Change Password: Users can change their password from the profile page.

* **Item Reporting**

**Report Lost Item**

Lost Item Details: Users can report lost items by providing details such as title, description, date lost, location, and an optional image.

Category Selection: Users can categorize the item (e.g., electronics, documents, accessories).

**Report Found Item**

Found Item Details: Users can report found items by providing details such as title, description, date found, location, and an optional image.

Category Selection: Users can categorize the item (e.g., electronics, documents, accessories).

* **Search Functionality**

**Search Lost Items**

Keyword Search: Users can search for lost items using keywords.

Filter Options: Users can filter search results by date, location, and category.

Search Results: Display relevant lost items matching the search criteria.

**Search Found Items**

Keyword Search: Users can search for found items using keywords.

Filter Options: Users can filter search results by date, location, and category.

Search Results: Display relevant found items matching the search criteria.

* **Item Listing**

**Lost Items Listing**

Lost Items Overview: A list of all reported lost items is available for users to browse.

Detailed View: Each item listing includes the title, description, date lost, location, and an image (if provided).

**Found Items Listing**

Found Items Overview: A list of all reported found items is available for users to browse.

Detailed View: Each item listing includes the title, description, date found, location, and an image (if provided).

**Communication System**

**Messaging**

User Messaging: Users can send and receive messages regarding lost and found items.

Secure Messaging: A secure messaging system allows users to communicate without exposing their personal contact information.

* **Notifications**

**Email Notifications**

Report Confirmation: Users receive email notifications for successful item reports.

Message Alerts: Users receive email notifications for new messages.

Password Reset: Users receive email notifications for password reset requests

* **Admin Panel**

**Manage Reports**

Report Overview: Admins can view and manage all reported lost and found items.

Edit/Delete Reports: Admins can edit or delete reports if necessary.

**User Management**

User Overview: Admins can view user profiles.

Account Management: Admins can deactivate user accounts if necessary.

* **Responsive Design**

**Mobile-Friendly**

Optimized Interface: The application is designed to be responsive and accessible on various devices, including smartphones and tablets.

* **Security**

**Data Encryption**

Encryption Standards: All sensitive user data, including passwords, are encrypted to ensure security.

**Secure Communication**

HTTPS Protocol: The application uses HTTPS to secure all communication between the client and server.

* **User Experience**

**Intuitive Interface**

User-Friendly Design: The application features a user-friendly and intuitive interface, making it easy for users to navigate and use the platform.

**Feedback Mechanism**

User Feedback: Users can provide feedback on the application, helping to improve future versions and address any issues.

These features work together to provide a robust and user-friendly Lost & Found Android Application, helping users to efficiently report and search for lost and found items.

**2.1.2 User Classes and Characteristics**

The Lost & Found Android Application is designed to accommodate two primary user classes: regular users and admin users, each with distinct characteristics and responsibilities. Regular users are the primary audience, typically individuals with basic to intermediate proficiency in using smartphones and mobile applications. They need quick and easy access to report or search for lost and found items, emphasizing the importance of a user-friendly and intuitive interface. These users are often security-conscious, concerned about the privacy and security of their personal information and communications. They may use the application occasionally when they lose or find an item and come from varied demographics, necessitating an inclusive design. Regular users engage in activities such as reporting lost items by providing detailed information and images, reporting found items to help others recover their possessions, searching for items using keywords and filters, communicating securely with other users, and managing their profiles by updating personal details and changing passwords.

Admin users, on the other hand, have administrative privileges and are responsible for managing the application's content and user base. They typically possess a higher level of technical proficiency and familiarity with administrative tools and functions. Admin users need to be detail-oriented to manage reports effectively and ensure the accuracy and integrity of information. They are entrusted with sensitive information and the responsibility to oversee user reports and accounts, making their role crucial in maintaining the application's reliability and user trust. Admins regularly use the application to monitor activities, manage content, and support users. Their activities include reviewing, editing, and deleting lost and found item reports to maintain quality and relevance, overseeing user accounts to handle issues like account activation and deactivation, monitoring the messaging system to ensure appropriate use, and maintaining the overall system to ensure smooth and efficient operation. Understanding these user classes and their specific needs allows the Lost & Found Android Application to be tailored to provide an optimal user experience, ensuring both regular and admin users can perform their tasks effectively and efficiently.

* Regular Users

Regular users are the primary users of the application. They include individuals who need to report or search for lost and found items.

**Characteristics:**

- Basic Smartphone Skills: Regular users typically have basic to intermediate proficiency with using smartphones and mobile applications.

- Need for Quick Access: These users require quick and easy access to the application to report lost or found items promptly and search for items efficiently.

- Security Conscious: Users are concerned about the privacy and security of their personal information and communications.

- Occasional Users: Regular users may use the application occasionally when they lose or find an item, rather than daily.

- Varied Demographics: Regular users can come from various age groups, backgrounds, and locations, necessitating a user-friendly and inclusive design.

**Roles and Activities:**

- Reporting Lost Items: Users report items they have lost by providing detailed information and possibly uploading images.

- Reporting Found Items: Users report items they have found, aiding others in recovering their lost possessions.

- Searching for Items: Users search the database for lost or found items using keywords and filters.

- Communication: Users communicate with other users who have reported lost or found items, using the secure messaging system.

- Profile Management: Users manage their profiles, including updating personal details

* Admin Users

Admin users have administrative privileges and are responsible for managing the application's content and user base.

**Characteristics:**

- Technical Proficiency: Admin users are typically more technically proficient and familiar with administrative tools and functions.

- Detail-Oriented: Admins need to be detail-oriented to effectively manage reports and ensure the accuracy and integrity of information.

- Responsible and Trustworthy: Admins are trusted with sensitive information and the responsibility to manage user reports and accounts.

- Regular Use: Admins use the application regularly to monitor activities, manage content, and support users.

**Roles and Activities:**

- Managing Reports: Admins review, edit, and delete lost and found item reports to maintain the quality and relevance of the information.

- User Management: Admins oversee user accounts, handling issues such as account activation, deactivation, and troubleshooting user problems.

- Monitoring Communications: Admins ensure that the messaging system is used appropriately and intervene if any misuse is detected.

- System Maintenance: Admins may be involved in maintaining the overall system, ensuring it runs smoothly and efficiently.

By understanding the distinct characteristics and needs of these user classes, the Lost & Found Android Application can be tailored to provide an optimal user experience, ensuring both regular and admin users can effectively perform their required tasks.

**2.1.3 Operating Environment**

The Lost & Found Android Application is designed to operate efficiently within a specific technical environment to ensure optimal performance, security, and user experience. This section outlines the key components of the operating environment, including hardware, software, network, and external systems.

* **Hardware Environment**

The application is intended to run on Android devices, which include smartphones and tablets. The minimum hardware specifications are as follows:

- Processor: Quad-core processor with a minimum clock speed of 1.5 GHz.

- RAM: At least 2 GB of RAM to handle the application’s functionalities smoothly.

- Storage: A minimum of 100 MB of free storage space for application installation and data storage.

- Camera: A functional camera for users to upload images of lost or found items.

- Internet Connectivity: Reliable internet connection (Wi-Fi or cellular) for data synchronization and communication.

* **Software Environment**

The application is built to be compatible with a range of Android operating system versions, ensuring accessibility for a broad user base. The software requirements include:

- Operating System: Android version 5.0 (Lollipop) and above. This ensures compatibility with a wide range of devices, including older models still in use.

- Development Framework: Android Studio for development, using Java or Kotlin as the programming language.

- Database: Firebase Realtime Database or a cloud-hosted relational database like MySQL or PostgreSQL for storing user data, item reports, and messaging information.

- APIs: Integration with third-party APIs for functionalities such as location services (Google Maps API) and push notifications (Firebase Cloud Messaging).

* **Network Environment**

The application requires a stable and secure network environment to function correctly. The network requirements include:

- Internet Access: Continuous internet access via Wi-Fi or cellular data for real-time data synchronization and communication.

- Security Protocols: Use of HTTPS for all data transmissions to ensure data security and privacy.

- Firewall and Network Security: Adequate firewall protection and network security measures on the server side to prevent unauthorized access and data breaches.

* **Server Environment**

The backend server environment plays a critical role in managing the application’s operations and data. The server requirements include:

- Cloud Hosting: Hosting on a reliable cloud platform such as AWS, Google Cloud, or Heroku to ensure scalability, reliability, and uptime.

- Server Specifications: Adequate server resources (CPU, RAM, and storage) to handle concurrent users and data processing efficiently.

- Database Management: Efficient database management practices to handle large volumes of data, including indexing and regular backups.

- Security Measures: Implementation of robust security measures, including SSL certificates, encryption of sensitive data, and regular security audits.

* **External Systems**

The application may integrate with several external systems to enhance functionality and user experience. These integrations include:

- Payment Gateways: For any in-app purchases or premium features, integration with secure payment gateways like PayPal or Stripe.

- Email Services: Integration with email services (e.g., SendGrid) for sending verification emails, notifications, and password reset links.

- Analytics Tools: Use of analytics tools like Google Analytics or Firebase Analytics to monitor application usage, user behavior, and performance metrics.

By ensuring compatibility and optimization within this operating environment, the Lost & Found Android Application can deliver a reliable, secure, and seamless experience to its users, fulfilling its core objectives effectively.

**2.1.4 Design and Implementation Constraints**

The Lost & Found Android Application is designed to operate within a set of technical, resource, regulatory, environmental, design, and implementation constraints that significantly influence its development and deployment. Technically, the application is exclusive to the Android platform, supporting a wide range of devices from older models running Android 5.0 (Lollipop) to the latest, ensuring broad compatibility and performance optimization even on devices with limited processing power and memory. Resource constraints include the availability of development resources such as the number of developers, their skill levels, and the time allocated for development, which may impact the scope and timeline of the project. Financial limitations further influence the choice of technologies, tools, and services. Regulatory constraints necessitate compliance with data privacy laws like GDPR and CCPA, ensuring secure storage, processing, and management of user data, while adhering to industry security standards to protect against unauthorized access and breaches. Environmentally, the application relies on stable internet connectivity for real-time data synchronization, with the backend infrastructure requiring high availability to prevent functionality disruptions due to server outages. Design constraints mandate an intuitive, user-friendly interface consistent with Android UI guidelines, catering to users of varying technical proficiency, and ensuring accessibility for users with disabilities through features like screen reader compatibility and appropriate contrast ratios. Implementation constraints involve dependencies on third-party libraries and APIs, introducing risks related to changes in those services and potential costs. Integration with existing systems may face compatibility and interfacing protocol challenges. Comprehensive testing is essential to ensure the application is bug-free and performs as expected, although constraints on testing resources and time may impact thoroughness. By recognizing and addressing these multifaceted constraints, the development team can make informed decisions that balance functionality, performance, and feasibility, ensuring the successful delivery of the Lost & Found Android Application.

**Technical Constraints**

1. Platform Specificity:

- The application is designed exclusively for the Android platform. It will not be available for other operating systems such as iOS, limiting its accessibility to Android users only.

2. Device Compatibility:

- The application must support a wide range of Android devices, from older models running Android 5.0 (Lollipop) to the latest devices. This requires careful consideration of backward compatibility and performance optimization.

3. Performance Limitations:

- The application should perform efficiently on devices with limited processing power and memory. Optimization is necessary to ensure smooth operation even on devices with minimal hardware specifications.

**Resource Constraints**

1. Development Resources:

- The project is constrained by the availability of development resources, including the number of developers, their skill levels, and the time allocated for development. This may impact the scope and timeline of the project.

2. Budget Limitations:

- The project has a limited budget for development, testing, and deployment. Financial constraints may affect the choice of technologies, tools, and services used in the project.

**Regulatory Constraints**

1. Data Privacy Regulations:

- The application must comply with data privacy laws and regulations, such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). This includes ensuring user data is securely stored, processed, and managed.

2. Security Standards:

- The application must adhere to industry security standards to protect user data from unauthorized access, breaches, and other security threats. This involves implementing robust encryption, secure authentication, and regular security audits

**Environmental Constraints**

1. Network Dependence:

- The application relies on stable internet connectivity for real-time data synchronization and communication. Users in areas with poor network coverage may experience reduced functionality or performance issues.

2. Server Availability:

- The backend infrastructure must ensure high availability and reliability. Downtime or server outages can disrupt the application’s functionality and negatively impact user experience.

**Design Constraints**

1. User Interface Design:

- The application must have an intuitive and user-friendly interface that accommodates users with varying levels of technical proficiency. The design should be consistent with Android UI guidelines and best practices.

2. Accessibility Requirements:

- The application must be accessible to users with disabilities. This includes implementing features such as screen reader compatibility, appropriate contrast ratios, and easily navigable interfaces.

**Implementation Constraints**

1. Third-Party Dependencies:

- The application may rely on third-party libraries and APIs (e.g., Google Maps API, Firebase). Dependence on external services introduces risks related to changes in those services, availability, and potential costs.

2. Integration with Existing Systems:

- If the application needs to integrate with existing systems or databases, there may be constraints related to compatibility, data formats, and interfacing protocols.

3. Testing and Quality Assurance:

- Comprehensive testing is required to ensure the application is free of bugs and performs as expected. Constraints on testing resources and time may impact the thoroughness of testing processes.

By identifying and addressing these design and implementation constraints, the development team can make informed decisions that balance functionality, performance, and feasibility, ultimately delivering a successful Lost & Found Android Application.

**2.1.5 Assumptions and Dependencies**

The development and implementation of the Rental App are based on several key assumptions and dependencies that could impact the project's success. It is assumed that the target users will have access to modern smartphones running iOS 11+ or Android 8.0+, and that these platforms will support the app's functionality without significant compatibility issues. The project also relies on the availability and stability of third-party components, including payment gateways like Stripe or PayPal, which are essential for secure transaction processing. Additionally, the app depends on reliable cloud hosting services for backend infrastructure and data storage, which must maintain high uptime and performance standards.

Another assumption is that the app will need to comply with evolving platform guidelines and security standards from both Apple and Google, which may require ongoing updates and adjustments. The project also assumes that the integration with third-party services for notifications and analytics will remain stable and compatible with the app's architecture. Any changes or disruptions in these external factors could impact the app's functionality or development timeline. Dependencies on reusable software components from other projects must be carefully managed to ensure compatibility and avoid potential integration issues. Addressing these assumptions and dependencies is crucial for the project's success and for maintaining the app's reliability and performance.

**2.2 Functional Requirements**

The system allows the finder to choose (if it has already been created) or create a tag (the category of the property) and create a new item using the location and time of the event. Also, he or she will be asked to upload 1-2 photos of the property. Finally, if the property is valuable that means it is in the predefined category of ‘valuable property’ by the system, the finder will be asked to design 1-3 questions for identification authentication. When some answers are sent back from a possible owner, the finder can check whether it is the right person and inform the system if his or her contact information can be shared or not. The module allows the owner to choose a tag and see all available found properties in the format of the location and time it is found. When one match is detected, the owner can request to contact the finder. If it is a valuable property, he or she has to answer the questions designed by the finder for identification authentication. Then the owner will receive a notification from the system informing the result after the finder checks the answers. If passes, the owner will receive the photos of the property and then be asked to confirm whether it is his or her belongings. If yes, the finder’s contact information will be shared. To maintain the efficiency of backend database, the system should have the capability of memory self-management. First, an item and all corresponding data should be deleted when an event is completed, that is - the finder’s contact information has been notified to the owner. Then, all information of this event will only store in the owner’s profile (as a ‘history event’) in the format of text. Second, as for the unmatched items, they should be cleaned by the system in a certain period of time, 2 months for valuable property and 1 month for common property, when permission is obtained from the finder

In this project, LaF requires a local instance serving as a database to store all backend data. A Servlet would be created and maintained stably and efficiently to guarantee multiple users logging-in, uploading data and searching information. Also, the system should support information interacting between users via sending notifications to and screening feedbacks from different users

The Lost & Found (LaF) application requires a comprehensive backend infrastructure to effectively manage data and facilitate user interactions. At the core of this infrastructure is a local database instance designed to store all backend data, including user profiles, lost and found reports, and messaging information. This database must support efficient storage and retrieval processes to ensure quick access to relevant content. A Servlet will be implemented to handle various server-side operations, including user authentication, data uploads, and information retrieval. The Servlet must be stable and capable of managing multiple users simultaneously to prevent performance issues. The system must provide secure user authentication, enabling multiple users to log in concurrently without degrading performance. Users should be able to upload data, such as detailed descriptions and images of lost or found items, with the backend handling these uploads seamlessly. To enhance user experience, the application must include powerful search functionality, allowing users to find relevant information quickly through keyword searches, filters, and sorting options. Additionally, a notification system should be in place to enable real-time communication between users, sending updates about lost and found items and managing user feedback. The backend infrastructure must be scalable to accommodate growing data volumes and user numbers, with performance optimizations ensuring responsiveness even under high load conditions. Security measures are crucial to protect user data and privacy, including encryption of sensitive information, secure data transmission protocols, and regular security audits to address potential vulnerabilities. By addressing these requirements, the Lost & Found application will offer a robust, user-friendly platform for managing lost and found items efficiently and securely.

In this project, the Lost & Found (LaF) application requires a robust and efficient backend infrastructure to manage all data and user interactions. The key requirements for this infrastructure include:

1. Local Database Instance:

- The application will utilize a local database instance to store all backend data. This database must support efficient data storage and retrieval, ensuring quick access to user-generated content such as lost and found reports, user profiles, and messages.

2. Servlet Management:

- A Servlet will be created and maintained to handle server-side operations. This Servlet must be stable and efficient, capable of supporting concurrent access by multiple users. It will manage user authentication, data upload processes, and information retrieval requests, ensuring seamless user experiences.

3. User Authentication and Data Upload:

- The system must support secure user authentication, allowing multiple users to log in simultaneously without performance degradation. Users should be able to upload data, such descriptions and images of lost or found items, with the database handling these uploads efficiently.

4. Data Search and Retrieval:

- The system should offer powerful search capabilities, enabling users to quickly find relevant information. This includes keyword searches, filtering options, and sorting mechanisms to help users locate lost or found items with ease.

5. User Interaction and Notification System:

- The application should support interaction between users through a notification system. This system will send real-time notifications to users about updates related to their lost or items.

8: Scalability and Performance

- The backend infrastructure must be scalable to handle increasing amounts of data and users. Performance optimizations should be in place to ensure the system remains responsive and efficient, even under high load conditions.

**2.1 Use-Case 1**

**(Reporting a Lost item)**

**Table 2.1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-1 | |
| **Purpose** | | To allow Service Providers to create and manage their digital service listings on the Rental App. | |
| **Priority** | | High – | |
| **Pre-conditions** | | User choose “Sign up” and sign-up page is available and waiting for user input. | |
| **Post-conditions** | | After step 2, the main user page appears. | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user logs into the application. | | The system authenticates the user and displays the main dashboard. |
| **2** | The user navigates to the "Report Lost Item" section. | | The system displays the "Report Lost Item" form. |
| **3** | The user fills out the form with item details and uploads a photo | | The system accepts the input and prepares to validate the submitted information. |
| **4** | The user submits the form. | | The system validates the submitted information. |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user is not logged in. | | The system prompts the user to log in before accessing the "Report Lost Item" section. |
| **2** | The user submits the form with invalid information. | | The system highlights the invalid fields and prompts the user to correct the errors. |
| **3** | The user attempts to upload a photo, but the upload fails. | | The system prompts the user to try uploading the photo again or to proceed without a photo. |
| **4** | The system encounters a database error while saving the report. | | The system displays an error message to the user and prompts them to try submitting the report again later. |

**2.2.2 Use-Case 2**

**(Searching for a Lost Item)**

**Table 2.2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-2 | |
| **Purpose** | | To enable users to search for items they have lost, increasing the chances of finding and retrieving those items. | |
| **Priority** | | High | |
| **Pre-conditions** | | The user must be registered and logged into the application The user must have access to a stable internet connection. The database and server must be operational. | |
| **Post-conditions** | | The user successfully retrieves a list of items matching the search criteria. The user can view details of specific items from the search results. | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user logs into the application. | | The system authenticates the user and displays the main dashboard. |
| **2** | The user navigates to the "Search Lost Items" section. | | The system displays the search form with various filter options (e.g., item name, date lost, location). |
| **3** | The user fills out the search form with criteria for the lost item. | | The system accepts the input and prepares to perform the search. |
| **4** | The user submits the search form. | | The system processes the search request based on the provided criteria. |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user is not logged in. | | The system prompts the user to log in before accessing the "Search Lost Items" section |
| **2** | The user submits the search form with invalid criteria. | | The system highlights the invalid fields and prompts the user to correct the errors. |
| **3** | The system encounters a database error while performing the search. | | The system displays an error message to the user and prompts them to try searching again later. |
| **4** | No items match the search criteria. | | The system displays a message indicating that no items were found matching the search criteria. |

**2.2.3 Use-Case 3**

**(Marking an Item as Found)**

**Table 2.3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-3 | |
| **Purpose** | | To allow users to mark items they have found, notifying the owner and helping facilitate the return of the item. | |
| **Priority** | | High | |
| **Pre-conditions** | | The user must be registered and logged into the application. The user must have access to a stable internet connection. The database and server must be operational. | |
| **Post-conditions** | | The item is marked as found in the database. The owner of the lost item is notified that the item has been found. The item is removed from the list of lost items and moved to a "found" status. | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user logs into the application. | | The system authenticates the user and displays the main dashboard. |
| **2** | The user navigates to the "Found Items" section. | | The system displays the list of items reported as found. |
| **3** | The user selects an item they have found from the list. | | The system displays a detailed form to mark the item as found, including item details and optional message. |
| **4** | The user fills out the form with any additional details about the found item. | | The system accepts the input and prepares to update the item status. |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user is not logged in. | | The system prompts the user to log in before accessing the "Found Items" section. |
| **2** | The user submits the form with invalid information. | | The system highlights the invalid fields and prompts the user to correct the errors. |
| **3** | The system encounters a database error while updating the item's status. | | The system displays an error message to the user and prompts |

Fig-2.2

## **2.3 Nonfunctional Requirements**

The Lost & Found (LaF) application must meet several nonfunctional requirements to ensure it is robust, secure, and user-friendly. The performance requirements dictate that the application should respond to user requests, such as logging in or submitting forms, within 2 seconds under normal conditions, and support up to 1000 concurrent users without any performance degradation. Scalability is essential, allowing the system to handle growing amounts of data and user numbers efficiently. Safety measures include daily backups of critical data to prevent loss and robust error handling to provide meaningful messages to users and log errors for further analysis. Security is paramount, with data encryption in transit and at rest, and strong authentication and authorization mechanisms to protect user data. The application must also comply with data privacy regulations like GDPR and CCPA.

The software quality attributes emphasize usability, reliability, maintainability, and portability. The user interface should be intuitive and user-friendly, ensuring ease of use for all users. The system should achieve 99.9% uptime, guaranteeing high availability and minimal downtime. A well-documented and modular codebase will facilitate maintenance and updates, while the application should be compatible with a wide range of Android devices, from older models to the latest versions. Other requirements include accessibility, ensuring the app meets WCAG 2.1 standards for users with disabilities, and localization, supporting multiple languages to cater to a diverse user base. Compatibility with popular web browsers and Android devices is also crucial for providing a consistent user experience across different platforms. By adhering to these nonfunctional requirements, the LaF application will deliver a reliable, secure, and user-friendly service for managing lost and found items.

**2.3.1 Performance Requirements**

The Lost & Found (LaF) application must meet stringent performance requirements to ensure it delivers a seamless and efficient user experience. Firstly, the application is designed to respond to user actions—such as logging in, submitting forms, and searching for items—within 2 seconds under normal operating conditions. This quick response time is crucial for maintaining user satisfaction, as users expect near-instant feedback from their interactions. A slow application can lead to frustration and decreased usage, so maintaining this response time is a top priority.

In addition to response time, the system must handle up to 1000 concurrent users without any performance degradation. This requirement ensures that the application can support a large user base simultaneously, which is essential for a widely-used service like LaF. Achieving this level of concurrency involves optimizing server-side processes, efficient database queries, and ensuring that the application architecture supports load balancing and resource allocation effectively.

Scalability is another critical performance requirement. As the user base grows and more data is accumulated in the system, the backend infrastructure must be able to scale dynamically. This scalability involves both vertical scaling (enhancing the capacity of existing servers) and horizontal scaling (adding more servers to distribute the load). Efficient data management practices, such as indexing and caching, play a significant role in maintaining performance as the data volume increases. Additionally, using cloud services can provide on-demand resources, allowing the system to handle spikes in usage without compromising performance.

To support scalability and high concurrency, the application must employ robust load balancing techniques. Load balancers distribute incoming network traffic across multiple servers to ensure no single server becomes a bottleneck. This distribution helps in maintaining fast response times and handling high volumes of requests. Furthermore, the application should use asynchronous processing for tasks that do not require immediate user feedback, such as sending notifications or performing background data analysis. This approach ensures that the main user interactions remain swift and responsive.

Monitoring and performance tuning are ongoing processes. The application should have monitoring tools in place to track performance metrics in real-time, such as response times, server load, and database query performance. These metrics help identify performance bottlenecks and allow for proactive optimization. Regular performance testing, including load testing and stress testing, should be conducted to ensure the system can handle expected and peak loads without degradation.

In summary, the performance requirements for the LaF application are designed to ensure it provides a fast, reliable, and scalable service. Quick response times, the ability to handle high concurrency, and dynamic scalability are essential components. Implementing efficient server-side processes, optimizing database queries, employing load balancing, and continuous monitoring are all critical strategies to meet these requirements. By adhering to these performance standards, the LaF application will be well-equipped to provide a superior user experience, even as its user base and data volume grow.

**2.3.2 Safety Requirements**

The Lost & Found (LaF) application must adhere to several safety requirements to ensure the integrity and reliability of the system. These requirements are designed to protect user data, maintain system stability, and ensure resilience in the face of unexpected issues.

**Data Backup:**

- Frequency: The system must perform daily backups of all critical data to prevent data loss. These backups should include user profiles, lost and found reports, and transaction logs.

- Backup Storage: Backups should be stored securely in a separate location from the primary database to protect against data loss due to server failure, corruption, or other unforeseen events.

- Backup Verification: Regularly verify the integrity and completeness of backups to ensure that they can be reliably restored in case of data loss.

**Error Handling:**

- Graceful Degradation: The system should handle errors and exceptions gracefully, providing users with meaningful error messages and guidance on how to proceed. This approach helps maintain a positive user experience even when issues occur.

- Error Logging: All errors and exceptions should be logged with sufficient detail to facilitate debugging and resolution. This includes capturing information about the error, user actions leading up to it, and system state at the time of the error.

- Incident Response: Implement an incident response plan to address and resolve critical issues promptly. This plan should include procedures for identifying, diagnosing, and fixing errors, as well as communication protocols for informing users and stakeholders about service disruptions.

**System Recovery:**

- Disaster Recovery Plan: Develop and maintain a disaster recovery plan that outlines procedures for restoring the system in the event of a catastrophic failure. This plan should include steps for data restoration, system reconfiguration, and service resumption.

- Redundancy: Implement redundancy for critical components of the system, such as servers, databases, and network infrastructure. This redundancy ensures that if one component fails, the system can continue operating using backup components.

**Data Integrity:**

- Data Validation: Ensure that all data inputs from users are validated to prevent corruption and ensure data accuracy. This includes checks for data format, completeness, and consistency.

- Transaction Management: Use transaction management techniques to ensure that data operations are completed successfully or rolled back in case of failure. This prevents partial updates that could lead to data inconsistencies.

**System Monitoring:**

- Performance Monitoring: Continuously monitor system performance to detect and address issues before they impact users. This includes tracking metrics such as response times, server load, and error rates.

- Security Monitoring: Implement security monitoring to detect and respond to potential threats and vulnerabilities. This includes monitoring for unauthorized access attempts, suspicious activity, and compliance with security policies.

By adhering to these safety requirements, the LaF application will maintain high levels of data integrity, system stability, and user trust. Ensuring regular backups, robust error handling, effective system recovery plans, data integrity measures, and continuous monitoring are essential for delivering a reliable and resilient service.

**2.3.3 Security Requirements**

The Lost & Found (LaF) application must meet stringent security requirements to ensure the protection of user data and the integrity of the system. First and foremost, all sensitive data, including user credentials, personal information, and transaction details, must be encrypted both in transit and at rest. Encryption in transit is achieved using industry-standard protocols such as TLS (Transport Layer Security) to protect data as it moves between the user's device and the server. Encryption at rest ensures that stored data, such as database entries, is protected from unauthorized access even if physical security is compromised. This involves using strong encryption algorithms and key management practices.

Authentication and authorization are critical components of the security framework. The application must implement robust authentication mechanisms to verify the identity of users before granting access to the system. This typically includes secure password policies, multi-factor authentication (MFA), and secure password storage practices such as hashing with salt. Authorization controls ensure that users can only access and modify data that they are permitted to, based on their roles and permissions. This involves defining user roles and permissions clearly and enforcing them rigorously throughout the application.

Data privacy regulations, such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA), must be adhered to, ensuring that user data is collected, processed, and stored in compliance with legal requirements. This includes obtaining user consent for data collection, providing users with access to their data, and ensuring that data is handled securely and responsibly.

In addition to encryption and access controls, the application must be designed to resist common security threats such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). This involves implementing secure coding practices, validating and sanitizing user inputs, and employing security libraries and frameworks to protect against these vulnerabilities.

Regular security assessments and updates are essential to maintaining the application's security posture. This includes conducting vulnerability scans, penetration testing, and code reviews to identify and address potential security issues. Keeping the system and its dependencies up to date with the latest security patches helps protect against known vulnerabilities.

By adhering to these security requirements, the LaF application will safeguard user data, ensure compliance with regulations, and maintain a secure environment against potential threats, ultimately fostering user trust and confidence in the system.

**2.3.4 Software Quality Attributes**

The Lost & Found (LaF) application is designed to excel in several software quality attributes to ensure it provides a superior user experience and maintains high operational standards. **Usability** is a key attribute, with the application featuring an intuitive and user-friendly interface. This means the layout and navigation are designed to be straightforward, allowing users of varying technical backgrounds to interact with the system effortlessly. Clear instructions, responsive design, and accessibility features, such as screen reader compatibility and keyboard navigation support, contribute to a positive user experience.

**Reliability** is another critical quality attribute, with the application designed to achieve a high level of uptime and availability. This involves robust error handling, automated system monitoring, and a disaster recovery plan to address and recover from unexpected failures. The system should be capable of performing its intended functions consistently without frequent disruptions, ensuring that users can rely on it for managing lost and found items.

**Maintainability** refers to the ease with which the application can be updated, fixed, and enhanced over time. The codebase should be well-documented, modular, and adhere to best coding practices, facilitating efficient maintenance and updates. This includes using version control systems to manage code changes and implementing a clear process for deploying updates and patches.

**Portability** ensures that the application can operate across a range of Android devices and versions, from older models running Android 5.0 (Lollipop) to the latest versions. This involves designing the application to be compatible with different screen sizes and hardware configurations, as well as optimizing performance to provide a consistent experience across devices.

Together, these software quality attributes—usability, reliability, maintainability, and portability—contribute to creating an application that is easy to use, dependable, and adaptable to changing needs and environments. By focusing on these aspects, the LaF application aims to deliver a high-quality service that meets user expectations and performs reliably across various scenarios.

**2.4 Other Requirements**

In addition to the core functional and non-functional requirements, the Lost and Found must address several supplementary needs.

**Database Requirements**

The Lost & Found (LaF) application relies on a local database to manage and store critical data such as user profiles, lost and found item reports, and transaction logs. The database must be designed to ensure high performance and reliability. It should support efficient querying and indexing to facilitate quick retrieval of information, especially for search functions and real-time updates. Additionally, the database must handle concurrent access by multiple users gracefully, maintaining data consistency and integrity through transaction management and locking mechanisms. Regular maintenance tasks, such as database backups and performance tuning, are essential to ensure the database remains robust and responsive.

**External Interface Requirements**

The application must interact with various external interfaces to provide a seamless user experience. These interfaces include:

* APIs: The application should integrate with external APIs for functionalities such as geolocation services to track item locations or third-party notification services to send alerts to users. API integration must be secure, using authentication tokens and encryption to protect data exchanged between systems.
* Web Services: If the application connects with other web services, it should adhere to relevant standards and protocols, such as REST or SOAP, to ensure compatibility and reliability.
* Payment Gateways: For any financial transactions or premium features, the application should interface with secure payment gateways, ensuring compliance with Payment Card Industry Data Security Standard (PCI DSS) requirements.

**Internationalization Requirements**

To cater to a global audience, the application must support internationalization, allowing it to be adapted for different languages and regional settings. This includes:

* Localization: The application should provide a mechanism for translating user interface elements, messages, and content into multiple languages. It should support language switching based on user preferences or regional settings.
* Date and Time Formats: The application must accommodate various date and time formats based on user locale, ensuring that all date and time information is displayed correctly.
* Currency and Units: For international users, the application should display currency and measurement units according to regional standards.

**Legal Requirements**

The application must comply with relevant legal and regulatory requirements to ensure the protection of user data and operational legality. Key considerations include:

* Data Privacy Regulations: Adhere to regulations such as GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act) to protect user data and provide users with rights regarding their personal information.
* Consumer Protection Laws: Ensure that the application complies with laws related to consumer rights, such as accurate representation of features, transparent pricing, and fair terms of service.
* Accessibility Standards: Comply with accessibility laws and standards, such as the Americans with Disabilities Act (ADA) and Web Content Accessibility Guidelines (WCAG), to ensure that the application is usable by individuals with disabilities.

**Reuse Objectives**

The application should aim to reuse existing components and systems to maximize efficiency and reduce development time. This includes:

* Code Reuse: Utilize reusable code libraries, frameworks, and modules for common functionalities, such as authentication, data validation, and user interface elements. This approach enhances consistency and accelerates development.
* Third-Party Components: Where appropriate, integrate third-party solutions or services for features like analytics, user authentication, or payment processing, avoiding the need to develop these components from scratch.
* Existing Systems: If integrating with existing systems or services, ensure that the interfaces and interactions are well-documented and adhere to established protocols to facilitate seamless integration and interoperability.

**3. Analysis Models**

**Use–Case Diagram**

A use case diagram visually represents the interactions between users (actors) and the system to achieve specific goals (use cases). For the Lost & Found (LaF) application, a use case diagram helps to illustrate the different functionalities available to users and the system's responses to user actions.

Here's how you can structure a use case diagram for the LaF application:

Actors:

* **Registered User**: A user who has created an account and logged into the application.
* **Guest User**: A user who is not logged in and can perform limited actions.
* **Admin**: An application administrator who manages user accounts and oversees the system's overall functionality.

Use Cases:

* **Register Account**: Allows a new user to create an account in the application.
* **Login**: Authenticates a registered user to access their account.
* **Post Lost Item**: Enables users to report a lost item, including details and images.
* **Post Found Item**: Allows users to report a found item, including details and images.
* **Search for Items**: Lets users search for lost or found items based on various criteria.
* **View Item Details**: Displays detailed information about a specific item.
* **Claim Item**: Allows a user to claim a found item they believe is theirs.
* **Send Notifications**: Sends notifications to users about relevant updates (e.g., matches for lost items).
* **Provide Feedback**: Enables users to provide feedback on items or interactions.
* **Manage Users**: Admin functionality to manage user accounts and permissions.
* **Generate Reports**: Admin functionality to generate reports on system usage, item statistics, etc.

**Diagram Representation**

* **Actors** are represented as stick figures or ovals on the left side of the diagram.
* **Use Cases** are represented as ovals or circles within the system boundary.
* **System Boundary** is shown as a rectangle encompassing the use cases.

Relationships are represented as lines connecting actors to use cases, indicating interactions.

Explanation

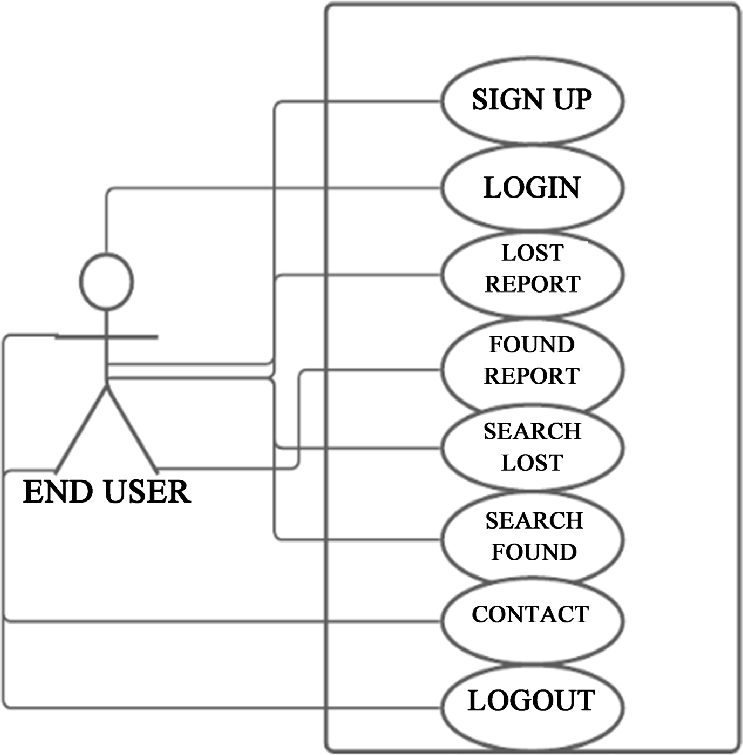
* **Guest User** can perform actions such as registering, logging in, and searching for items but has limited functionality compared to a registered user.
* **Registered User** can perform all actions available to guest users and additional actions like posting lost and found items, viewing item details, claiming items, and providing feedback.
* **Admin** has access to administrative functionalities, including managing users and generating reports.
* This use case diagram provides a high-level overview of how different actors interact with the Lost & Found application and highlights the core functionalities of the system.
*  **Use Case Diagram:**

Fig-3.1

**Diagram:** 

Fig-3.2

**Entity-Relationship Diagram (ERD) for Rental App**

An Entity-Relationship Diagram (ERD) is a visual representation of the data structure within a system, showing how entities (such as users, items, and reports) relate to each other. For the Lost & Found (LaF) application, the ERD will illustrate the primary data entities, their attributes, and the relationships between them.

Key Entities and Relationships

1. **User**

* **Attributes**: User ID (Primary Key), Username, Password Hash, Email, Phone Number, Role (Registered User, Admin), Date Joined
* **Relationships**:
* Posts Lost Item (One-to-Many): A user can post multiple lost items.
* Posts Found Item (One-to-Many): A user can post multiple found items.
* Provides Feedback (One-to-Many): A user can provide feedback on multiple items.

1. **Lose item**

* **Attributes**: Lost Item ID (Primary Key), User ID (Foreign Key), Item Description, Date Lost, Location Lost, Item Image, Status (Reported, Claimed, Removed)
* **Relationships**:
* Claimed By (Many-to-One): A lost item can be claimed by one user.
* Associated Feedback (One-to-Many): A lost item can have multiple feedback entries.

1. **Found item**

* **Attributes**: Found Item ID (Primary Key), User ID (Foreign Key), Item Description, Date Found, Location Found, Item Image, Status (Reported, Claimed, Removed)
* **Relationships**:
* Claimed By (Many-to-One): A found item can be claimed by one user.
* Associated Feedback (One-to-Many): A found item can have multiple feedback entries.

1. **Feedback**

* **Attributes**: Feedback ID (Primary Key), User ID (Foreign Key), Item ID (Foreign Key), Feedback Text, Date Submitted
* **Relationships**:

Belongs To User (Many-to-One): A feedback entry is associated with one user.

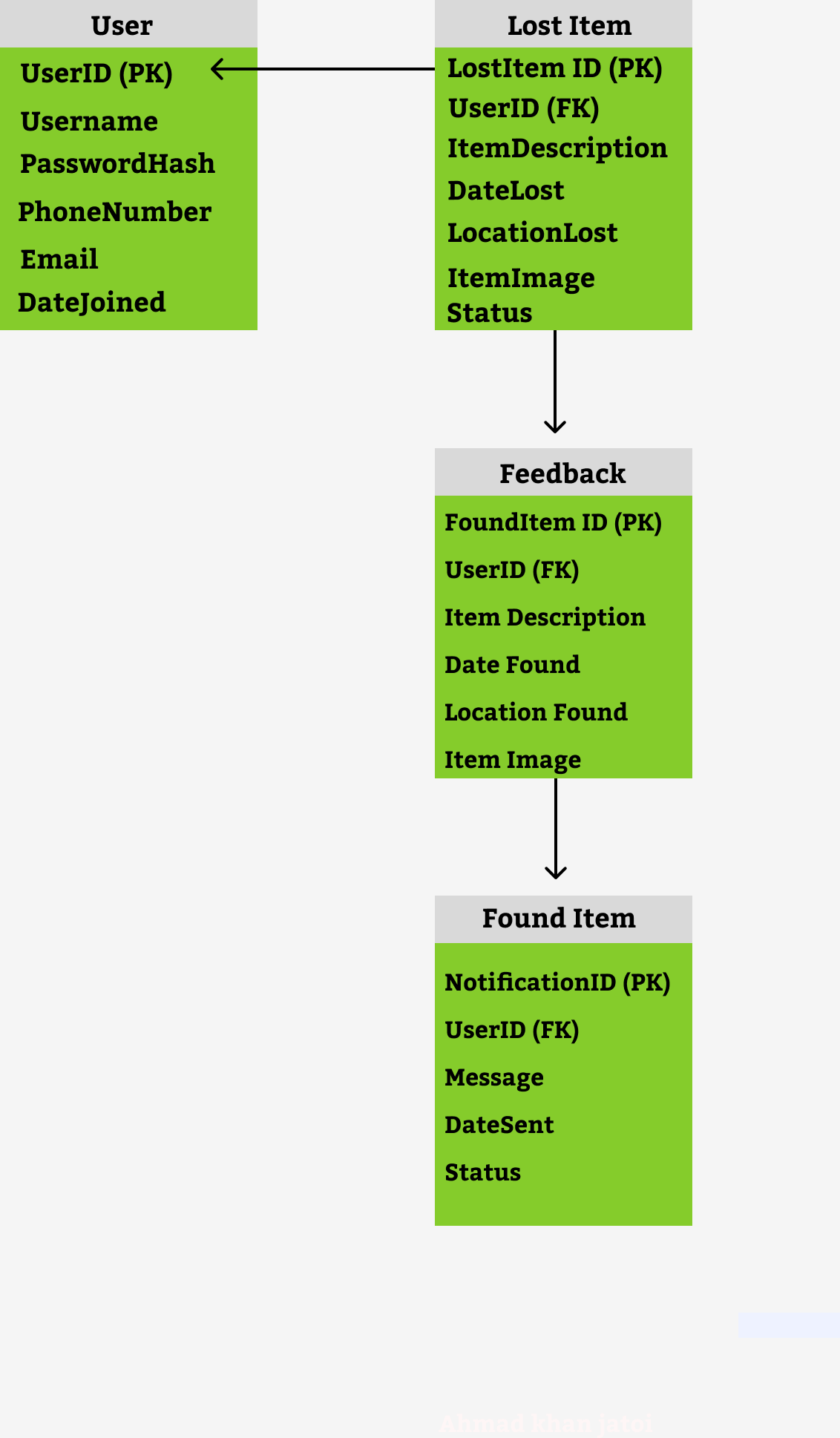
**Diagram**

Fig-3.3

**Class Diagram**

The Class Diagram illustrates the main classes in the Rental App, their attributes, methods, and the relationships between them. It helps in understanding the system's structure and how different components interact.

Explanation

* **User** class represents the individuals using the application, with methods for registration, login, and profile management.
* **Lost Item** and **Found Item** classes represent items reported as lost or found, respectively. They include methods for posting, updating, and claiming items.
* **Feedback** class holds user feedback related to items, with relationships linking it to both users and items.
* **Notification** class manages notifications sent to users, including methods for sending and marking notifications as read.

This class diagram provides a structured overview of the main components within the Lost & Found application, including their attributes, methods, and relationships.

**Class Diagram**

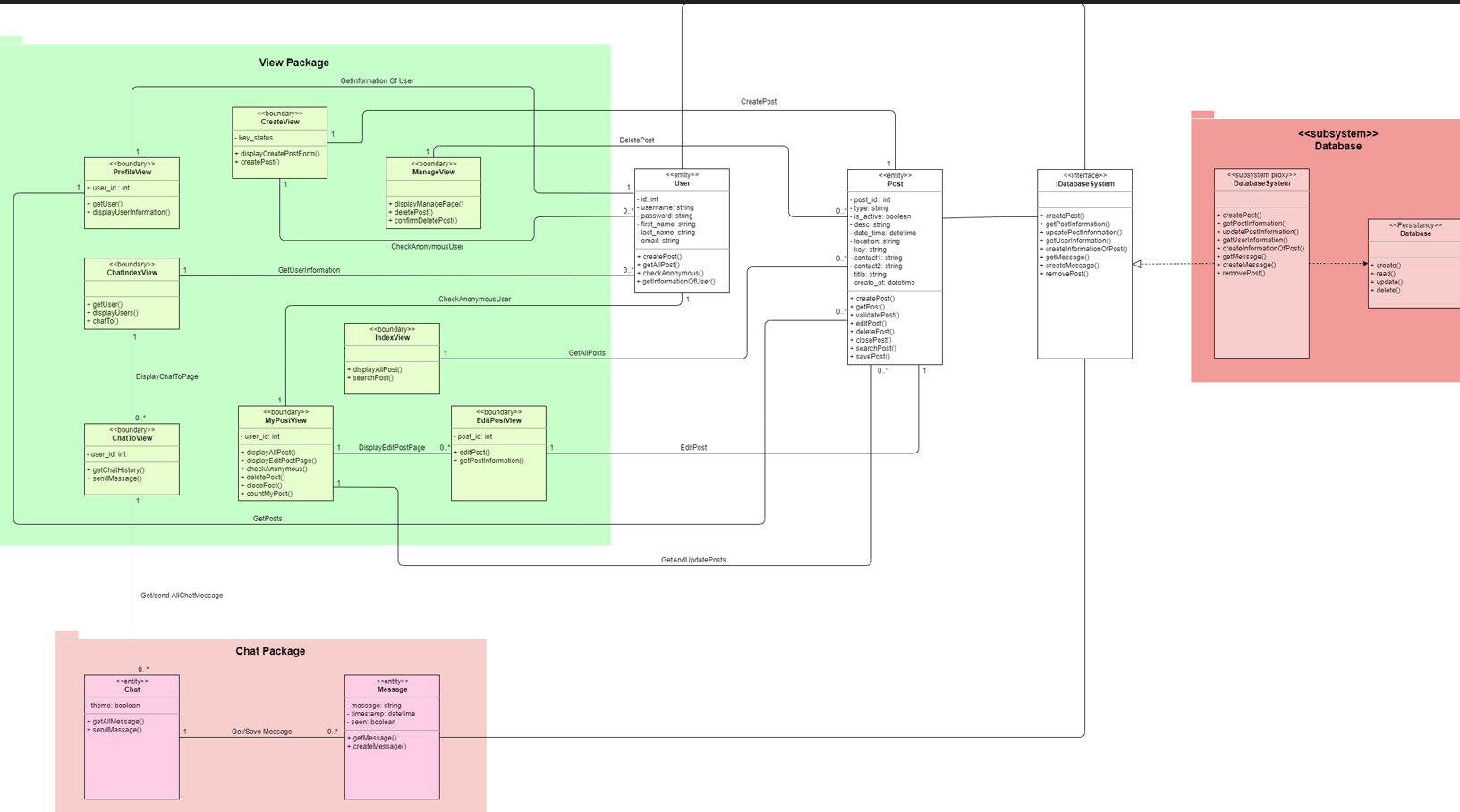


Fig-3.4

**Sequence Diagram**

A sequence diagram visually represents the sequence of interactions between objects or components in a system over time. For the Lost & Found (LaF) application, a sequence diagram can illustrate how different components interact to complete a specific use case.

Example Sequence Diagram: Posting a Lost Item

**Actors:**

* User
* Lost Item Class
* Database
* Notification System

**Scenario:** A registered user posts a lost item on the LaF application.

**Explanation**

1. **User** initiates the process by choosing to post a lost item. This action triggers a request to the Lost Item class to handle the item posting.
2. The **Lost Item** class processes the request and interacts with the **Database** to create a new record for the lost item. The create Lost Item () method is called to insert the new lost item information into the database.
3. The **Database** inserts the lost item record and confirms the insertion. The success or failure of this operation is communicated back to the Lost Item class.
4. The **Lost Item** class then initiates a notification to inform other users about the newly posted lost item. It interacts with the **Notification System** to send this notification.
5. The **Notification System** processes the request and sends the notification to users. It then returns a status update to the Lost Item class indicating whether the notification was successfully sent.
6. The **Lost Item** class receives the confirmation of the notification being sent and passes this confirmation back to the **User**, providing feedback that the item has been successfully posted.

Summary

The sequence diagram shows the flow of interactions from the user initiating the action to the system's responses, including updating the database and sending notifications.

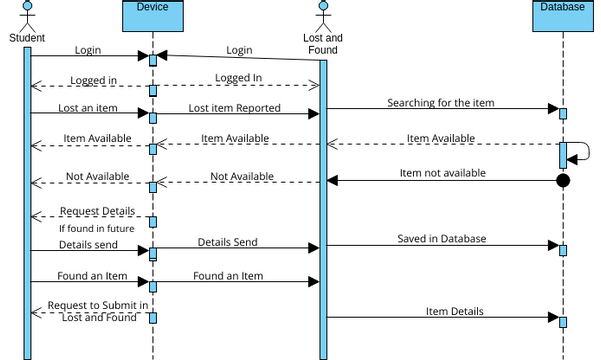
**Diagram**

Fig-3.5

**4 Design Models**

* + The component diagram of the Lost & Found (LaF) application illustrates the system's architectural structure by detailing the major components, their responsibilities, and how they interact with each other. The diagram is essential for understanding how the application is organized and how data flows through the system.

* + User Interface (UI) is the front-end component where users interact with the application. It is built using HTML, CSS, and JavaScript, providing an intuitive interface for users to post lost or found items, view item details, provide feedback, and manage notifications. The UI collects user input and sends requests to the Application Server, displaying the responses and updates to the user.

* + The Application Server acts as the central hub of the system, handling the business logic and processing user requests. It is implemented using technologies such as Java Servlets or Node.js with Express.js. This component manages user authentication, item postings, feedback processing, and data interactions with the Database. It ensures that user actions are appropriately processed and that the necessary data is retrieved or updated.

* + The Database is the persistent storage component of the application, using relational database management systems like MySQL, PostgreSQL, or SQLite. It stores all essential data, including user profiles, lost and found item records, feedback, and notifications. The Database supports CRUD (Create, Read, Update, and Delete) operations, enabling the Application Server to query, modify, and manage data as needed.
* The Notification System is responsible for managing and delivering notifications to users. Utilizing services such as Firebase Cloud Messaging, Twilio, or an SMTP server, this component sends alerts about new items, feedback responses, and other relevant updates. It handles the scheduling and delivery of notifications and manages their status, such as read or unread.

* + External APIs provide additional functionality by integrating third-party services. These APIs might include Google Maps for geolocation services, Stripe for payment processing (if applicable)
  + User Interface (UI) is the front-end component where users interact with the application. It is built using HTML, CSS, and JavaScript, providing an intuitive interface for users to post lost or found items, view item details, provide feedback, and manage notifications. The UI collects user input and sends requests to the Application Server, displaying the responses and updates to the user.
  + The Database is the persistent storage component of the application, using relational database management systems like MySQL, PostgreSQL, or SQLite. It stores all essential data, including user profiles, lost and found item records, feedback, and notifications. The Database supports CRUD (Create, Read, Update, and Delete) operations, enabling the Application Server to query, modify, and manage data as needed.

**Diagram**

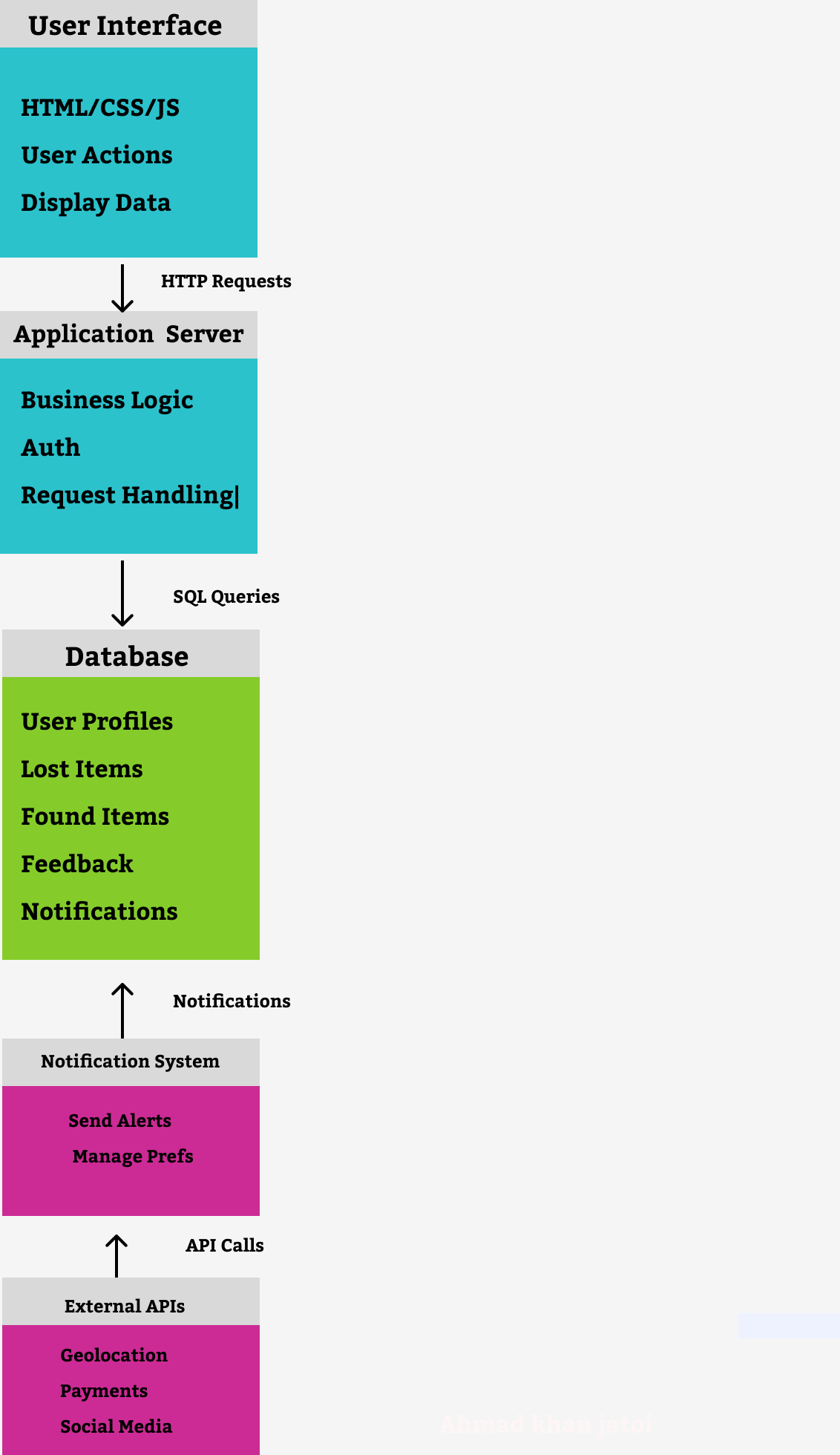


Fig 4.1

**Deployment Diagram**

A deployment diagram visualizes the physical deployment of artifacts (e.g., software components, databases) on nodes (e.g., servers, devices) in a system. For the Lost & Found (LaF) application, the deployment diagram will show how various components of the application are distributed across different hardware and network resources.

**Diagram**

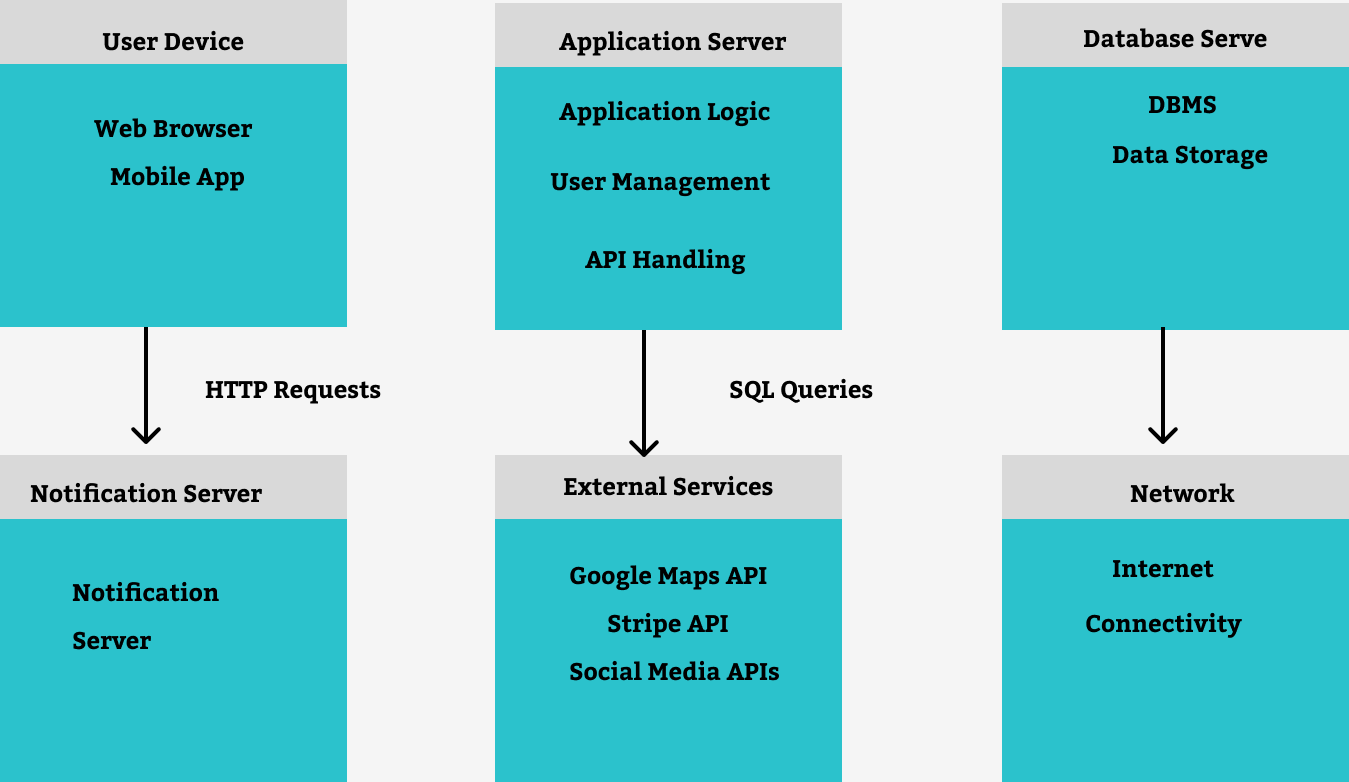


Fig-4.2

## **Data Flow Diagram:**

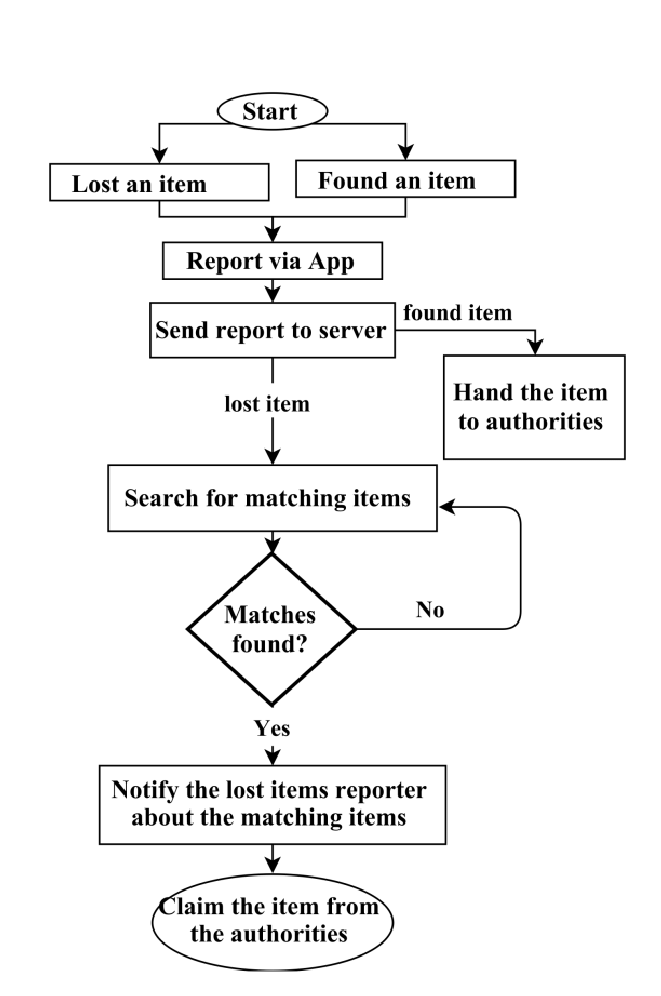


Fig 4.3: Data Flow Diagram

**5. Screenshots**

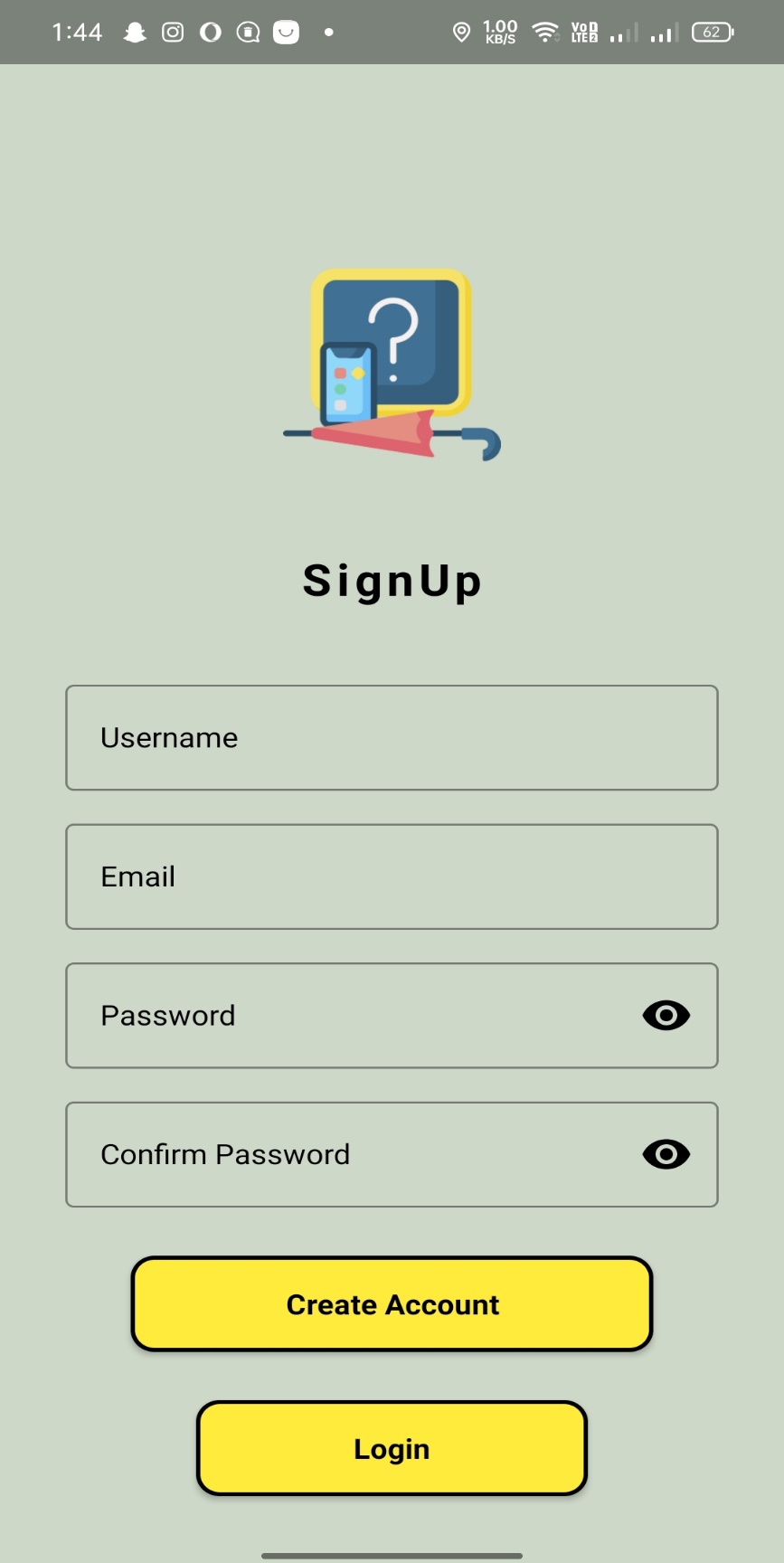
**

Fig 5.1: Signup

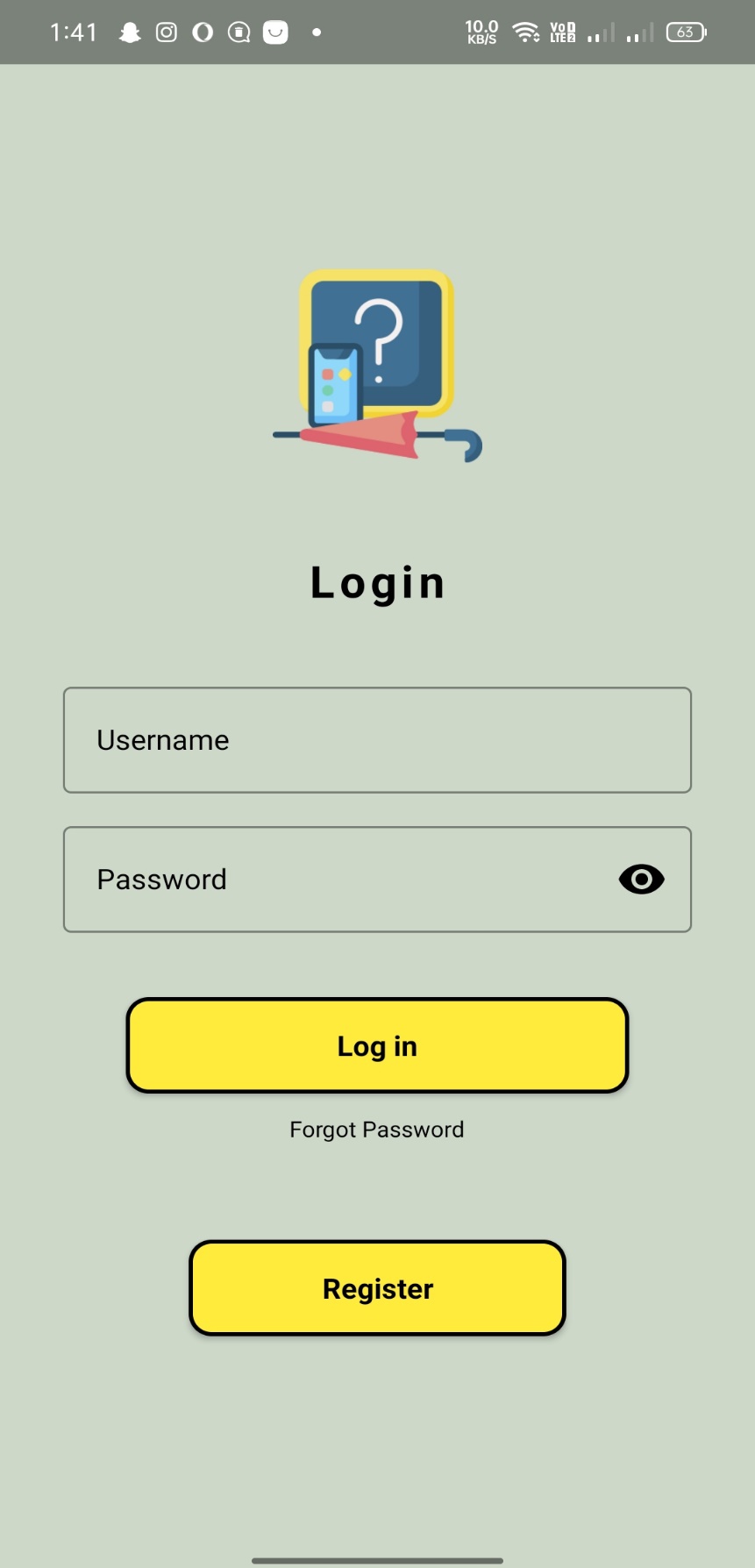
**

Fig 5.2: login

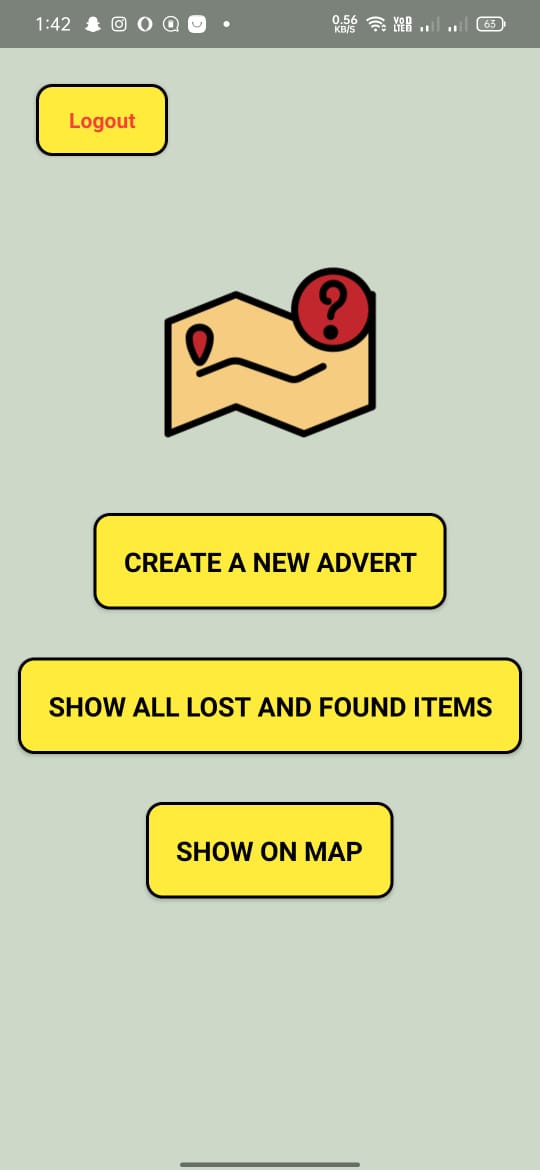
**

Fig 5.3: Home screen

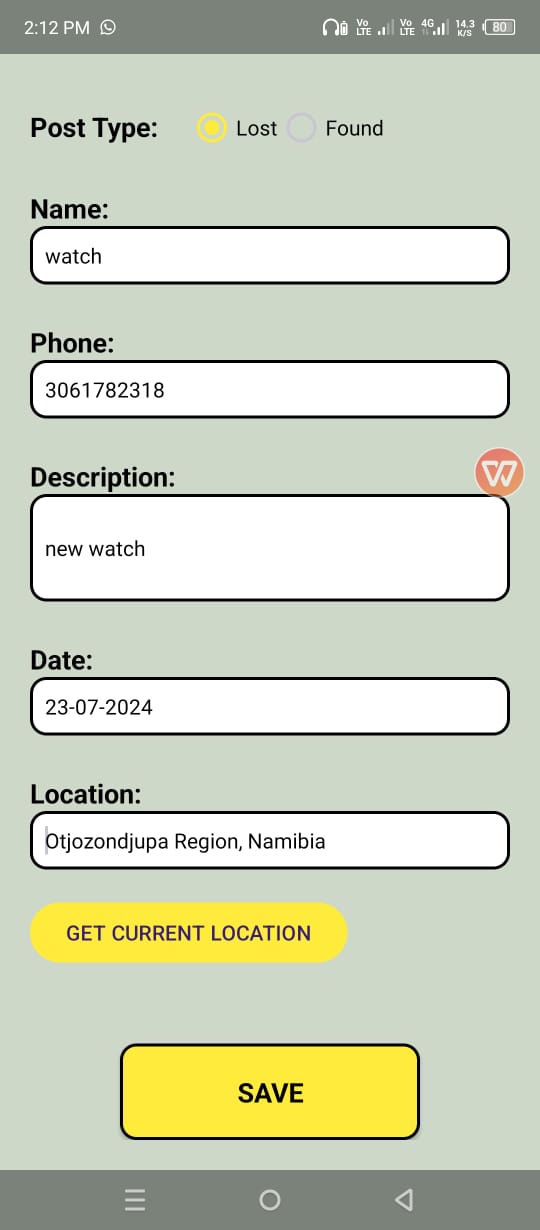
**

Fig 5.4: Lost

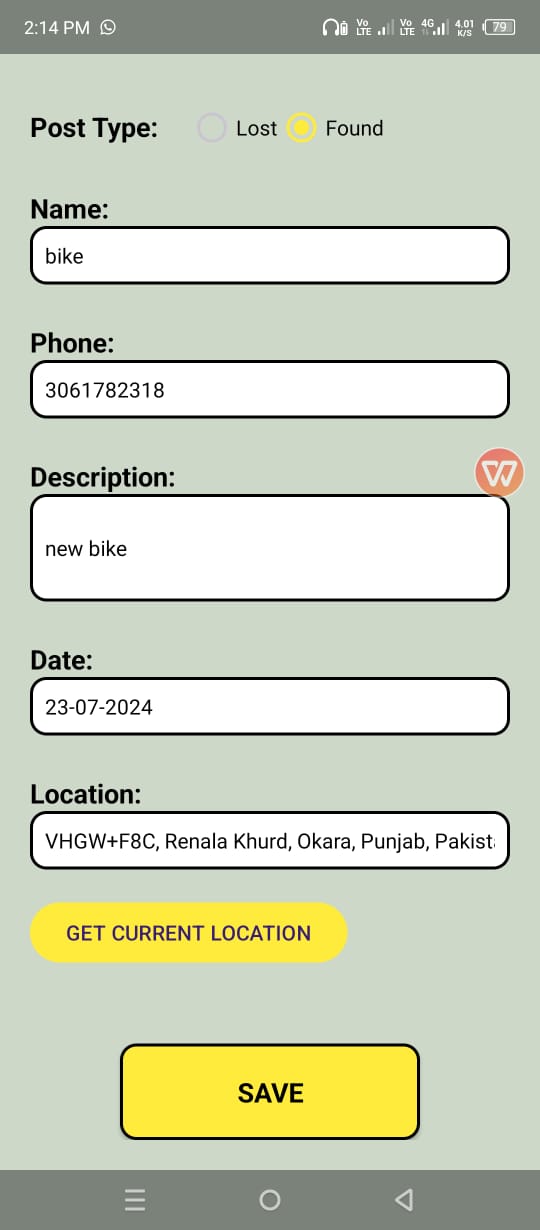
**

Fig 5.5: Found

***4***

Fig 5.6: Result

******

Fig 5.7: Lost and Found items

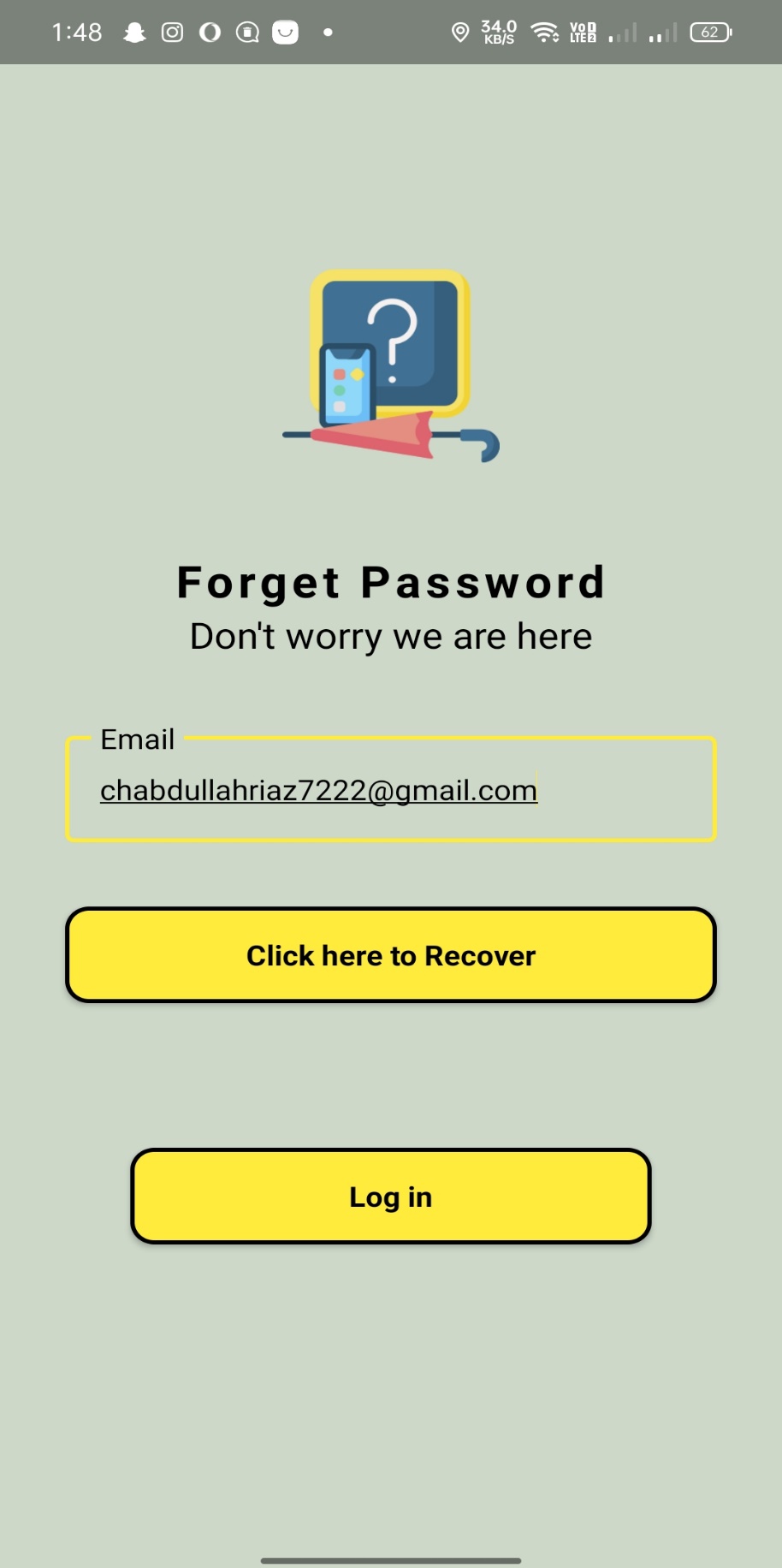
**

Fig 5.8: Forget Password

**

Fig 5.9: Location

**6. Test Cases**

**Test Cases 1**

**Table 6.1**

|  |  |
| --- | --- |
| **Identifier** | TC\_1 |
| **Priority** | high |
| **Related requirements(s)** | RQ\_LAF\_001 (Users must be able to post a lost item) |
| **Short description** | Verify user can successfully post a lost item. |
| **Pre-condition(s)** | User is registered and logged into the LaF application. <br> - The application is connected to the internet. |
| **Input data** | **Item Name:** "Black Wallet" **Description:** "A black leather wallet with several cards." **Location:** "Main Street Park "Date **Lost:** "2024-07-20" **Contact Information:** "user@example.com" |
| **Detailed steps** | 1. Navigate to the "Post Lost Item" page from the dashboard.2. Enter item details.3. Click the "Submit" button.4. Observe the response from the application. |
| **Expected result(s)** | Confirmation message: "Your lost item has been posted successfully." New item appears in the "Lost Items" list with correct details. User receives a confirmation email. |
| **Post-condition(s)** | Lost item is saved in the database. Item is visible in the "Lost Items" list. User receives a confirmation email. |

# 

# **Test Cases 2**

**Table 6.2**

|  |  |
| --- | --- |
| **Identifier** | TC\_2 |
| **Priority** | high |
| **Related requirements(s)** | RQ\_LAF\_002 (Users must be able to search for lost items) |
| **Short description** | Verify that a user can successfully search for a lost item. |
| **Pre-condition(s)** | User is registered and logged into the LaF application. There are existing lost item records in the database. |
| **Input data** | Search Keyword: "Wallet" |
| **Detailed steps** | 1. Log into the LaF application with valid credentials. 2. Navigate to the "Search Lost Items" page. 3. Enter "Wallet" in the search bar. 4. Click the "Search" button |
| **Expected result(s)** | The application displays a list of lost items matching the search keyword "Wallet". - Each item shows relevant details such as the item name, description, date lost, and location |
| **Post-condition(s)** | The search results are accurate and relevant to the keyword provided. |

**Test Cases 3**

**Table 6.3**

|  |  |
| --- | --- |
| **Identifier** | TC\_3 |
| **Priority** | high |
| **Related requirements(s)** | RQ\_LAF\_003 (Users must be able to receive notifications for new lost items) |
| **Short description** | Verify that users receive notifications for new lost items. |
| **Pre-condition(s)** | User is registered and logged into the LaF application. User has subscribed to notifications for lost items. The notification system is functional. |
| **Input data** | A new lost item record |
| **Detailed steps** | 1. An admin or another user posts a new lost item (e.g., "Black Wallet"). 2. Check the registered user’s notification inbox or notification area in the application. |
| **Expected result(s)** | The user receives a notification about the new lost item. The notification contains relevant details such as the item name and description. |
| **Post-condition(s)** | The notification is marked as read after the user views it. |