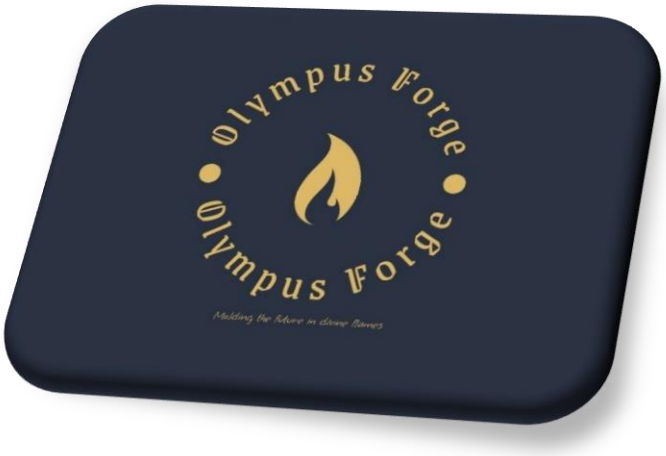
Lost and Found Management System

Team Name: Olympus Forge

Members:

- Zain Tahir (22i-0837)
- Awwab Aamir (22i-0971)
- Abdullah Waheed (22i-0765)



Deliverable:

- Deliverable 2
- System requirement specification document

Submission Date:
March 23, 2025

Introduction

The Lost and Found Management System is a full-stack web application developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack. The primary goal of the system is to facilitate efficient reporting, tracking, and management of lost and found items within an institutional or organizational setting. The application allows users to register and authenticate securely, post details about lost or found items, browse listings, and communicate with item owners or finders.

This project was built following software engineering principles and methodologies, with a focus on modularity, maintainability, and scalability. Agile practices were adopted throughout the development lifecycle, utilizing tools like Trello for sprint planning and task tracking. The backend is responsible for handling user authentication, data validation, and CRUD operations for items and users, while the frontend provides an intuitive and responsive user interface. MongoDB is used as the database to store user and item records efficiently.

The system is designed to operate on a local server and serves as a conceptual implementation of how modern web technologies can be integrated to address real-world problems through collaborative software development

System Requirements Specification

Functional Requirements

1. User Registration and Authentication
    - The system shall allow users to register an account by providing necessary credentials.
    - The system shall authenticate users before granting access to personalized features.
    - The system shall allow users to update and manage their account details securely.

2. Lost Item Reporting
    - The system shall allow users to submit reports of lost items, including item descriptions, category, and location.
    - The system shall provide an option for users to upload images of lost items.
    - The system shall store lost item reports in a structured database.

3. Found Item Reporting
    - The system shall allow users to submit reports of found items, including item descriptions, category, and location.
    - The system shall provide an option for users to upload images of found items.
    - The system shall store found item reports in a structured database.

4. Search for Lost and Found Items
    - The system shall allow users to search for lost and found items using filters such as category, location, and date.
    - The system shall display relevant search results in a structured manner.
    - The system shall provide item details, including images, location, and date reported.

5. Claiming Found Items
    - The system shall allow users to submit claim requests for found items.
    - The system shall provide an interface for claim verification and supporting information.
    - The system shall notify administrators to review and approve/reject claims.

6. Notification System
    - The system shall notify users when a reported lost item matches a found item.
    - The system shall notify users of updates regarding their submitted claims.

7. Report History
    - The system shall provide a report history section where users can view past lost and found reports.
    - The system shall allow users to track the status of their reported items.
8. Administrative Features
    - The system shall allow administrators to review and approve/reject reported lost and found items.
    - The system shall provide an interface for administrators to handle disputes over claimed items.
    - The system allows administrators to moderate user-generated content and flag inappropriate reports.

Non-Functional Requirements

Product Requirements

1. Usability

- The system should provide an intuitive and user-friendly interface with clear navigation.

2. Performance
   - The system shall support up to 50 concurrent users without performance degradation.

3. Scalability
   - The system's architecture shall allow for future enhancements without significant modifications.

4. Data Integrity
   - The system shall ensure data consistency by preventing duplicate records for lost and found items.

Organizational Requirements

1. Maintainability
   - The system's codebase shall be modular and well-documented to facilitate future development and debugging.

2. Developer Accessibility
   - The system shall be deployable on a local development environment with minimal setup.

3. Testing
   - The system shall undergo unit and integration testing to ensure stability and reliability.

External Requirements

1. Security
   - The system shall store user credentials securely using encryption techniques.
   - The system shall ensure data protection to prevent unauthorized access to user reports and claims.

2. Compliance
   - The system shall adhere to university guidelines regarding data privacy and ethical use of information.

3. Technology Constraints
   - The system shall be developed and hosted on a local server without dependencies on cloud-based services.

User stories

1. Report a Lost Item (Abdullah)

- User Role: User
- Goal: Report a lost item by providing relevant details.
- Reason: Enables users to submit lost item reports, increasing the chances of retrieval.
- Pre-conditions: User must be logged in.
- Post-conditions: The lost item report is stored in the database and confirmation is sent to the user.

2. Report a Found Item (Abdullah)

- User Role: User
- Goal: Report an item found with necessary details.
- Reason: Allows users to report found items, helping lost items be reclaimed by their rightful owners.
- Pre-conditions: User must be logged in.
- Post-conditions: The found item report is saved in the database, and confirmation is sent to the user.

3. Search for Lost Items (Abdullah)

- User Role: User
- Goal: Search for lost items using filters like category, location, and date.
- Reason: Helps users find lost items efficiently.
- Pre-conditions: The user must have access to the search page.
- Post-conditions: A list of matching lost items is displayed.

4. Search for Found Items (Abdullah)

- User Role: User
- Goal: Search for found items using filters like category, location, and date.
- Reason: Helps users check if someone has reported their lost item.
- Pre-conditions: The user must have access to the search page.
- Post-conditions: A list of matching items found is displayed.

5. Claim a Found Item (Abdullah)

- User Role: User
- Goal: Submit a claim request for a found item by verifying ownership.
- Reason: Ensures that only rightful owners retrieve lost items.
- Pre-conditions: A matching found item must be listed.
- Post-conditions: The claim request is submitted for verification.

6. User Registration (Zain)

- User Role: User
- Goal: Create an account to use the system.
- Reason: Allows users to report and claim items.
- Pre-conditions: User must provide valid registration details.
- Post-conditions: A new user account has been created.

7. User Login (Zain)

- User Role: User
- Goal: Log in to access the system.
- Reason: Ensures security and personalized access.
- Pre-conditions: The user must have a registered account.
- Post-conditions: The user is granted access to the system.

8. View Report History (Zain)

- User Role: User
- Goal: View a list of reported lost and found items.
- Reason: Helps users track their reported items.
- Pre-conditions: The user must be logged in.
- Post-conditions: A list of the user's reported items is displayed.

9. Receive Notifications on Matches (Zain)

- User Role: User
- Goal: Get notified when a found item matches a reported lost item.
- Reason: Keeps users updated about potential matches.
- Pre-conditions: The user must have reported a lost item.
- Post-conditions: A notification is sent to the user.

10. Generate Report (Admin) (Zain)

- User Role: Admin
- Goal: Get a standardized report of this month.
- Reason: To figure out similarities between lost cases.
- Pre-conditions: Admin is logged in and values exist.
- Post-conditions: Entire output is shown with graphs.

11. Manage Reported Items (Admin) (Awwab)

- User Role: Admin
- Goal: Review and validate lost and found item reports.
- Reason: Ensures data integrity and prevents spam or misuse.
- Pre-conditions: The system must contain reported lost and found items.
- Post-conditions: Items are either approved or flagged for review.

12.Send System Announcements (Admin) (Awwab)

- User Role: Admin
- Goal: Send system-wide notifications.
- Reason: Keeps users informed about updates, issues, or changes.
- Pre-conditions: The admin must have an announcement to make.
- Post-conditions: Users receive the system-wide notification.

13.Generate Reports on Lost & Found Items (Admin) (Awwab)

- User Role: Admin
- Goal: Generate reports summarizing lost and found trends.
- Reason: Helps administrators analyze data and improve the system.
- Pre-conditions: The system must contain lost and found data.
- Post-conditions: A report is generated and stored for analysis.

14.Allow Extensive Found Item Search (Awwab)

- User Role: User
- Goal: Search for desired item using keywords.
- Reason: Easy retrieval of items.
- Pre-conditions: User is logged in.
- Post-conditions: The closest items are flagged.

15.Moderate User-Generated Content (Admin)

- User Role: Admin
- Goal: Review and remove inappropriate or misleading posts.
- Reason: Ensures the platform remains trustworthy.
- Pre-conditions: The system must contain user-generated posts.
- Post-conditions: Inappropriate posts are flagged or removed.

Product Backlog

User Story 1: Report a Lost Item

User Level: Regular User

Priority: High

Tasks:

- Design the lost item report form.
- Implement the form submission functionality.
- Store lost item details in the database.
- Allow users to upload images of lost items.
- Send confirmation to the user upon reporting a lost item.

User Story 2: Report a Found Item

User Level: Regular User

Priority: High

Tasks:

- Design the found item report form.
- Implement the form submission functionality.
- Store found item details in the database.
- Allow users to upload images of found items.
- Send confirmation to the user upon reporting a found item.

User Story 3: Search for Lost Items

User Level: Regular User

Priority: High

Tasks:

- Implement search filters (category, location, date).
- Design the search results page.
- Fetch lost item data from the database.
- Display matching results.

User Story 4: Search for Found Items

User Level: Regular User

Priority: High

Tasks:

- Implement search filters (category, location, date).
- Design the search results page.
- Fetch found item data from the database.
- Display matching results.

User Story 5: Claim a Found Item

User Level: Regular User

Priority: High

Tasks:

- Implement claim request submission.
- Design claim verification process.
- Store claim request in the database.
- Notify admin for review.

User Story 6: User Registration

User Level: Regular User

Priority: High

Tasks:

- Design registration form.
- Validate user input.
- Store user details in the database.
- Send account confirmation email.

User Story 7: User Login

User Level: Regular User

Priority: High

Tasks:

- Design login page.
- Implement authentication logic.
- Validate user credentials.
- Redirect user upon successful login.

User Story 8: View Report History

User Level: Regular User

Priority: Medium

Tasks:

- Design report history page.
- Fetch user-reported items from the database.
- Display reports in a structured manner.

User Story 9: Receive Notifications on Matches

User Level: Regular User

Priority: High

Tasks:

- Implement notification service.
- Compare reported lost and found items.
- Notify users when a match is found.

User Story 10: Generate Report

User Level: Admin

Priority: Medium

Tasks:

- Parse the already acquired data.
- Import and use necessary libraries.
- Make graphs using parsed data

User Story 11: Manage Reported Items

User Level: Admin

Priority: High

Tasks:

- Create admin dashboard for item review.
- Implement approve/reject functionality.
- Flag suspicious reports for further verification.

Sprint 3

User Story 12: Allow Extensive search for found items

User Level: User

Priority: High

Tasks:

- Implement universal keyword search bar.
- Implement error checking methods.
- Ensure filtered items are filtered correctly.

User Story 13: Send System Announcements

User Level: Admin

Priority: Medium

Tasks:

- Create announcement module.
- Allow admin to send messages to users.
- Store announcements for future reference.

User Story 14: Report Suspicious Activity

User Level: Regular User

Priority: Medium

Tasks:

- Implement suspicious activity reporting feature.
- Allow users to submit complaints.
- Notify admin about reported activities.

User Story 15: Generate Reports on Lost & Found Items

User Level: Admin

Priority: Medium

Tasks:

- Create data analytics dashboard.
- Generate periodic reports on lost and found trends.
- Store generated reports for future reference.

User Story 16: Allow Anonymous Lost Item Reporting

User Level: Guest User

Priority: Low

Tasks:

- Design guest reporting form.
- Validate required fields (e.g., contact details).
- Store reports separately from registered user reports.

User Story 17: Moderate User-Generated Content

User Level: Admin

Priority: Medium

Tasks:

- Implement content moderation tools.
- Allow admins to flag or remove inappropriate posts.
- Store moderation logs for future reference.

Sprint Backlogs

Sprint 1

User Story 1: Report a Lost Item

User Level: Regular User

Priority: High

Tasks:

- Design the lost item report form.
- Implement the form submission functionality.
- Store lost item details in the database.
- Allow users to upload images of lost items.
- Send confirmation to the user upon reporting a lost item.

User Story 2: Report a Found Item

User Level: Regular User

Priority: High

Tasks:

- Design the found item report form.
- Implement the form submission functionality.
- Store found item details in the database.
- Allow users to upload images of found items.
- Send confirmation to the user upon reporting a found item.

Sprint 2

User Story 3: Search for Lost Items

User Level: Regular User

Priority: High

Tasks:

- Implement search filters (category, location, date).
- Design the search results page.
- Fetch lost item data from the database.
- Display matching results.

User Story 4: Search for Found Items

User Level: Regular User

Priority: High

Tasks:

- Implement search filters (category, location, date).
- Design the search results page.
- Fetch found item data from the database.
- Display matching results.

User Story 5: Claim a Found Item

User Level: Regular User

Priority: High

Tasks:

- Implement claim request submission.
- Design claim verification process.
- Store claim request in the database.
- Notify admin for review.

User Story 6: User Registration

User Level: Regular User

Priority: High

Tasks:

- Design registration form.
- Validate user input.
- Store user details in the database.
- Send account confirmation email.

User Story 7: User Login

User Level: Regular User

Priority: High

Tasks:

- Design login page.
- Implement authentication logic.
- Validate user credentials.
- Redirect user upon successful login.

User Story 8: View Report History

User Level: Regular User

Priority: Medium

Tasks:

- Design report history page.
- Fetch user-reported items from the database.
- Display reports in a structured manner.

User Story 9: Receive Notifications on Matches

User Level: Regular User

Priority: High

Tasks:

- Implement notification service.
- Compare reported lost and found items.
- Notify users when a match is found.

User Story 10: Receive Claim Status Updates

User Level: Regular User

Priority: High

Tasks:

- Implement status tracking for claims.
- Notify users about claim approval or rejection.
- Update claim status in the database.

User Story 11: Manage Reported Items

User Level: Admin

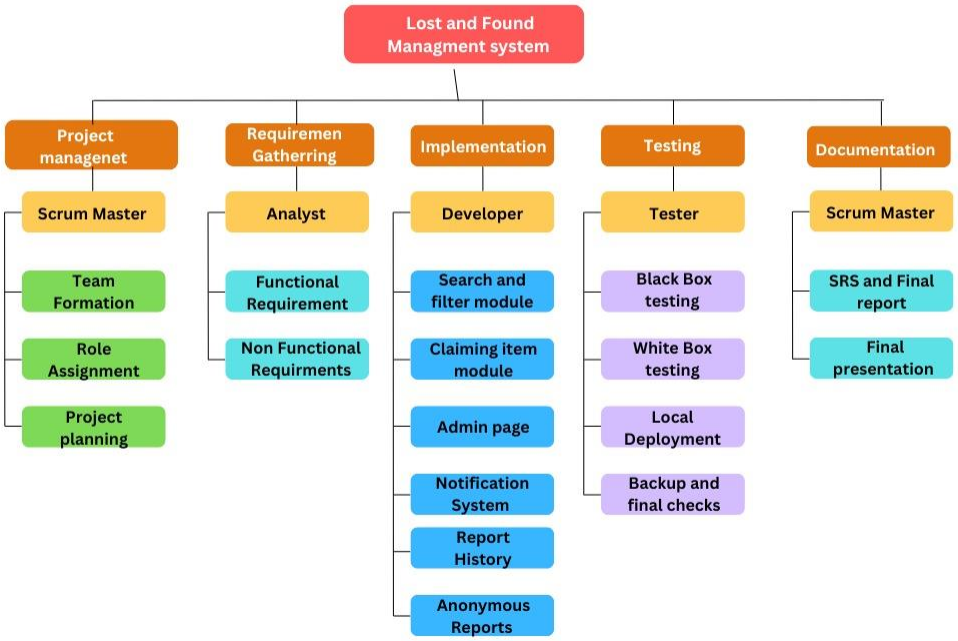
Priority: High

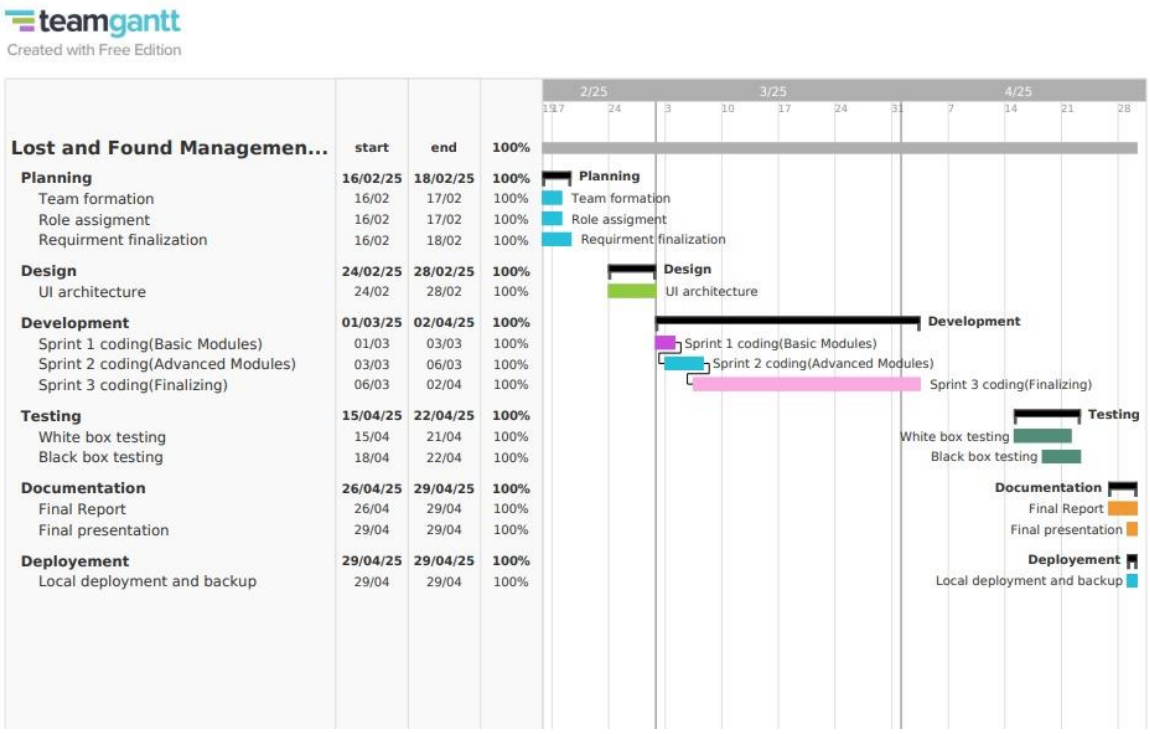
Tasks:

- Create admin dashboard for item review.
- Implement approve/reject functionality.
- Flag suspicious reports for further verification.

Project Plan

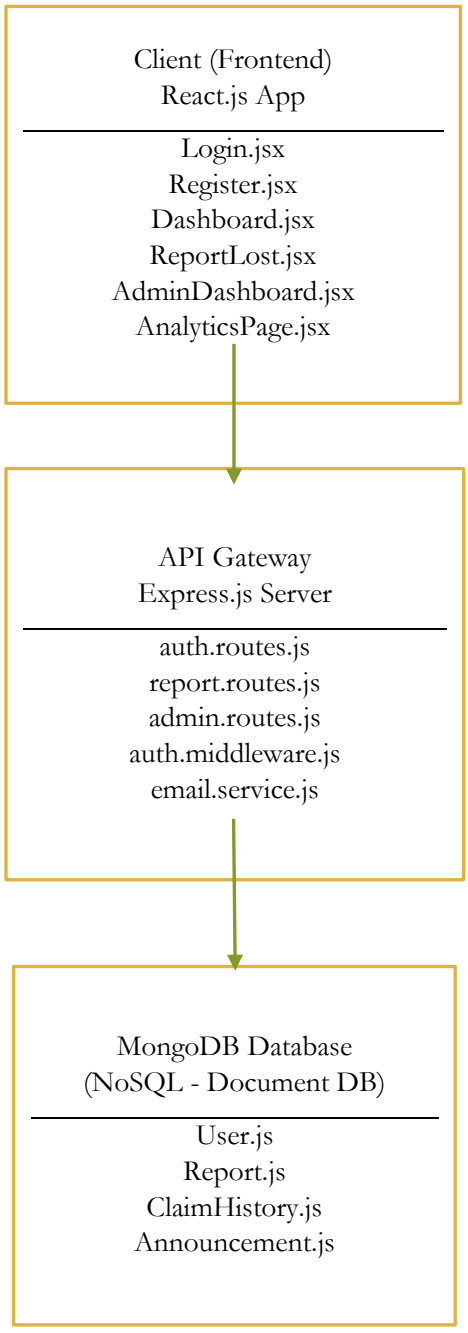
Work Breakdown Structure





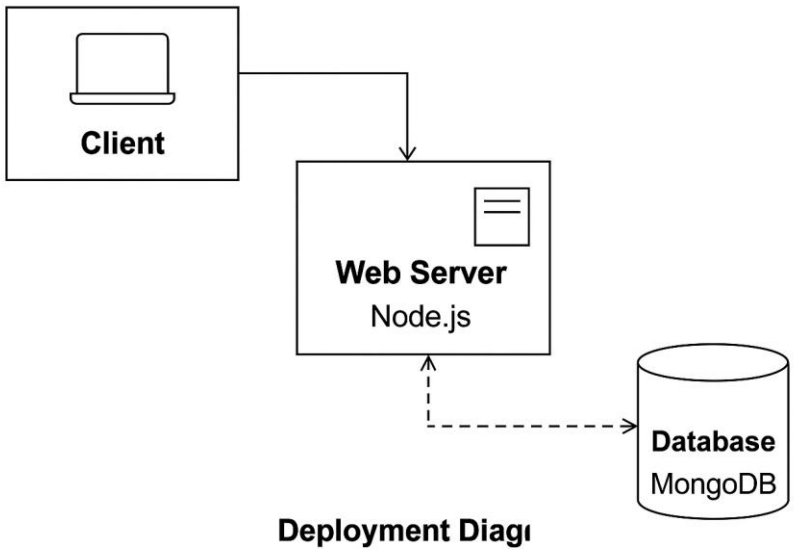
Architecture diagrams

1) UML Diagram

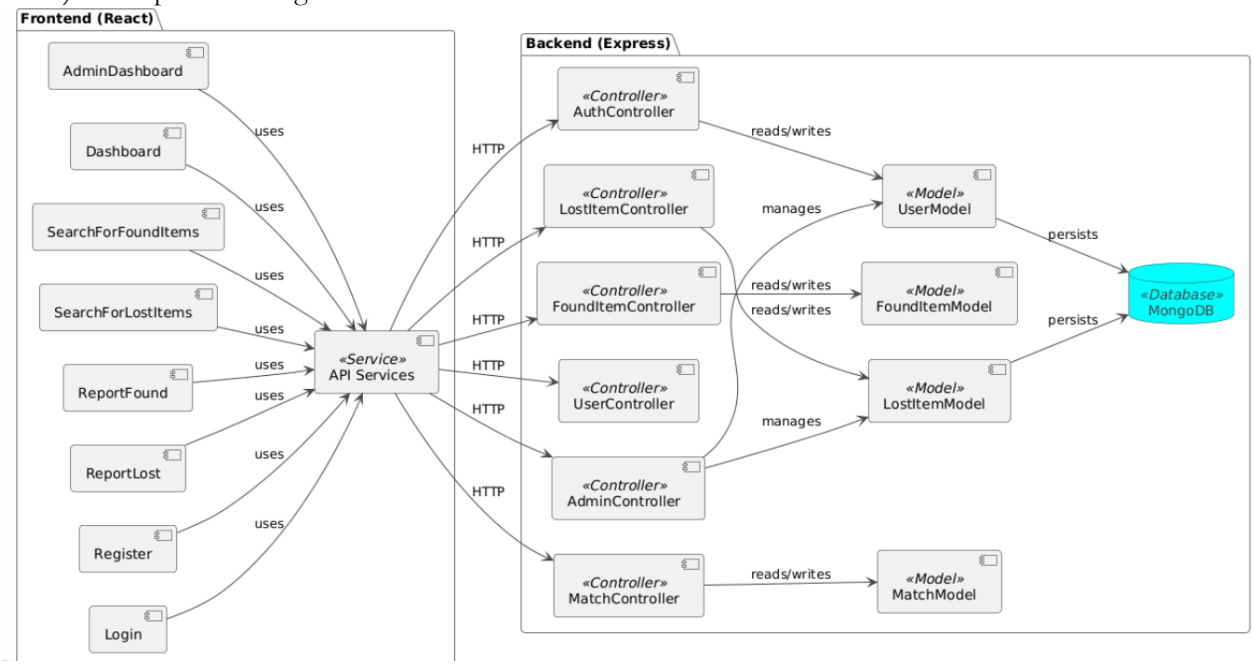


2) Architectural Styles Used:
   - Client: Component-Based (React Components)
   - Server: Layered (Routes → Controllers → Services → Models)
   - RESTful Communication (HTTP API between frontend & backend)
   - Repository Pattern (Mongoose for MongoDB Access)

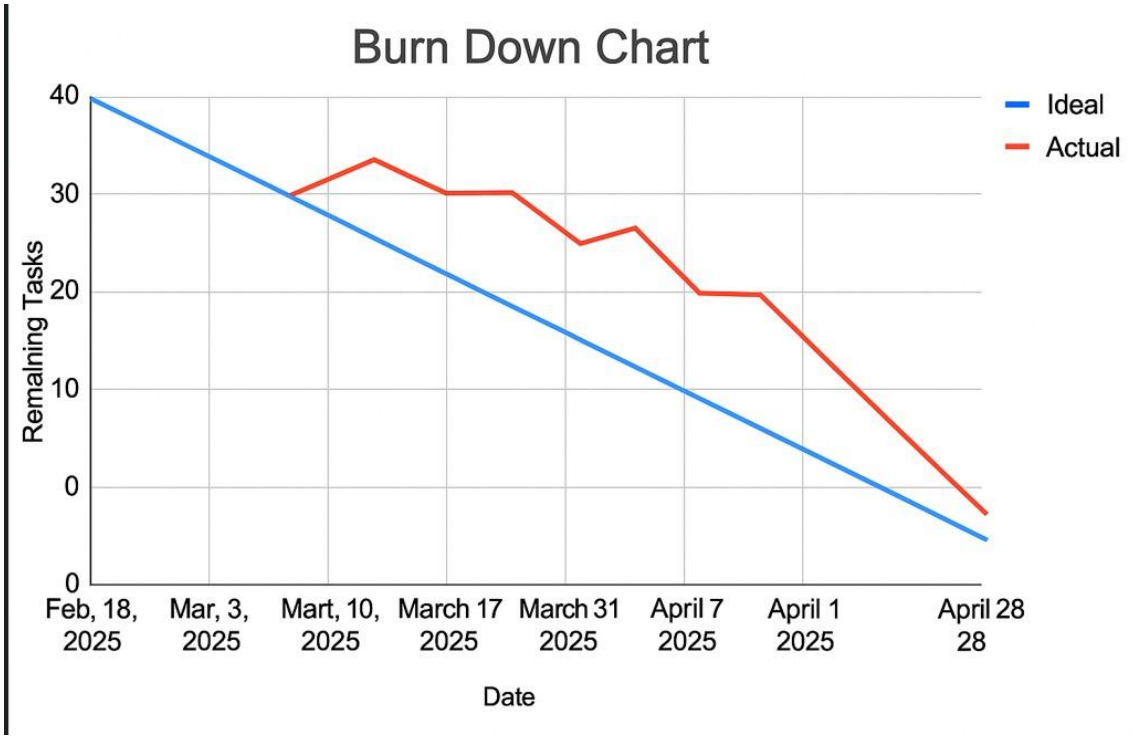
3) Deployment Diagram:



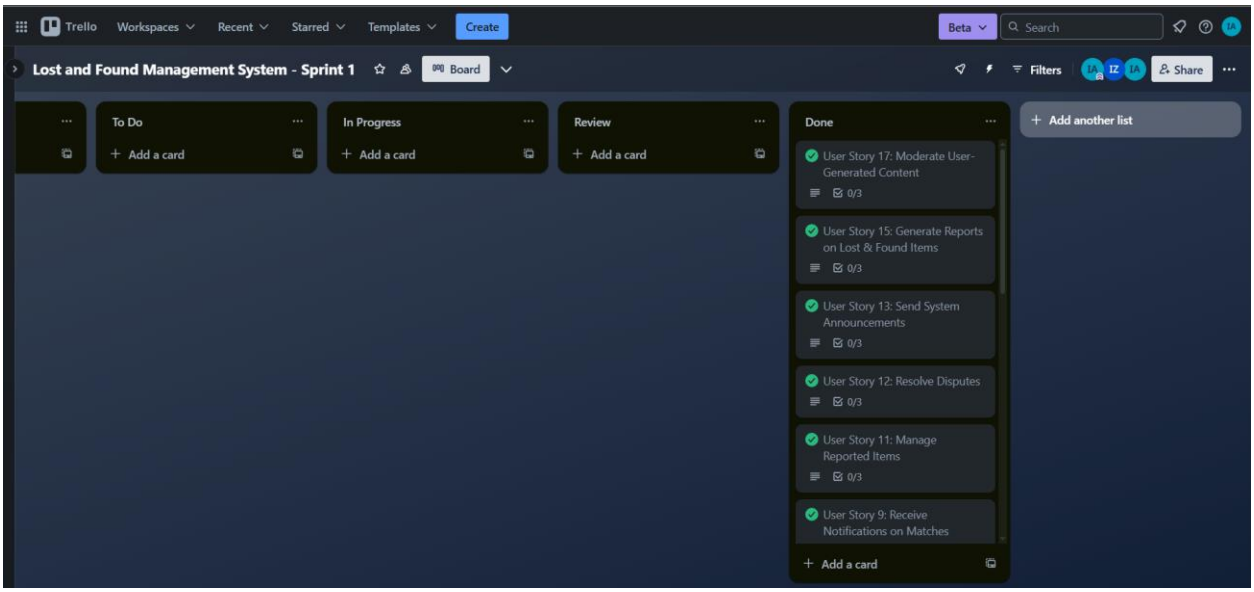
4) Components Diagram:



Project Burn Down Chart



Trello board



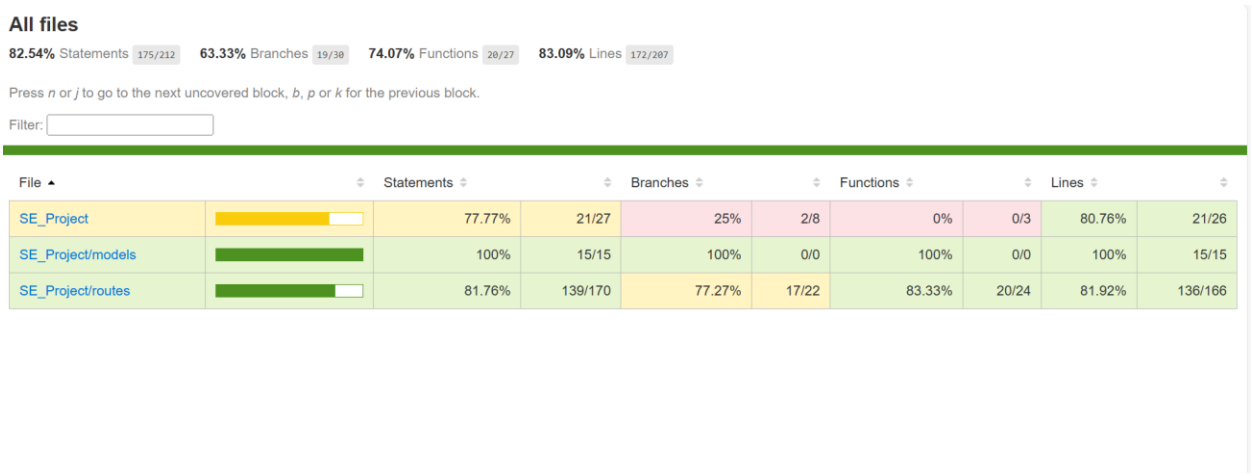
Black Box Testing

Class	Input Variable	Description
A1	Title (Lost Item)	Valid: 1–100 characters
A2	Title (Lost Item)	Invalid: >100 characters
B1	Password (User Reg)	Valid: 8–20 characters
B2	Password (User Reg)	Invalid: <8 or >20 characters
C1	Date Found (Item)	Valid: Past/current date
C2	Date Found (Item)	Invalid: Future date
D1	Location (Search)	Valid: Existing location
D2	Location (Search)	Invalid: Non-existent location
E1	File (Claim Item)	Valid: PDF/JPG/PNG ≤5MB
E2	File (Claim Item)	Invalid: Other formats or >5MB

Test Case ID	User Story	Test Data (Classes)	Type	Expected Result
TC-01	Report a Lost Item	A1	Weak	Accepted
TC-02	Report a Lost Item	A2	Weak	Error: Title too long
TC-03	User Registration	B1	Weak	Accepted
TC-04	User Registration	B2	Weak	Error: Password too short
TC-05	Report a Found Item	C1	Weak	Accepted
TC-06	Report a Found Item	C2	Weak	Error: Future date invalid
TC-07	Search for Lost Items	D1	Weak	Results displayed
TC-08	Search for Lost Items	D2	Weak	No results found
TC-09	Claim a Found Item	E1	Weak	Accepted
TC-10	Claim a Found Item	E2	Weak	Error: Invalid file
TC-11	Report a Lost Item	A1 + C1 + D1	Strong	Accepted
TC-12	Report a Lost Item	A2 + C1 + D1	Strong	Error: Title too long
TC-13	User Registration	B1 + C1 + D1	Strong	Accepted
TC-14	User Registration	B2 + C2 + D2	Strong	Multiple errors
TC-15	Claim a Found Item	E1 + A1 + B1	Strong	Accepted

White Box Testing

Backend:



All files

82.54% Statements 175/212 63.33% Branches 19/30 74.07% Functions 20/27 83.09% Lines 172/207

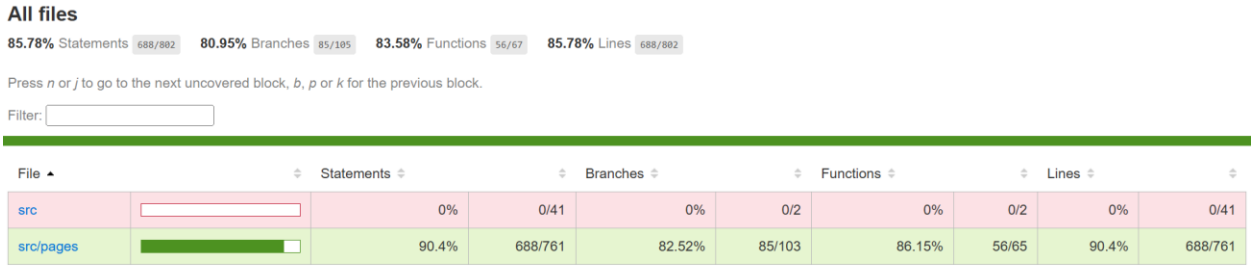
Press n or j to go to the next uncovered block, b, p or k for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
SE_Project	77.77%	21/27	25%	2/8	0%	0/3	80.76%	21/26
SE_Project/models	100%	15/15	100%	0/0	100%	0/0	100%	15/15
SE_Project/routes	81.76%	139/170	77.27%	17/22	83.33%	20/24	81.92%	136/166

Excluded: node_modules, env files, json files

Frontend:



All files

85.78% Statements 688/802 80.95% Branches 85/105 83.58% Functions 56/67 85.78% Lines 688/802

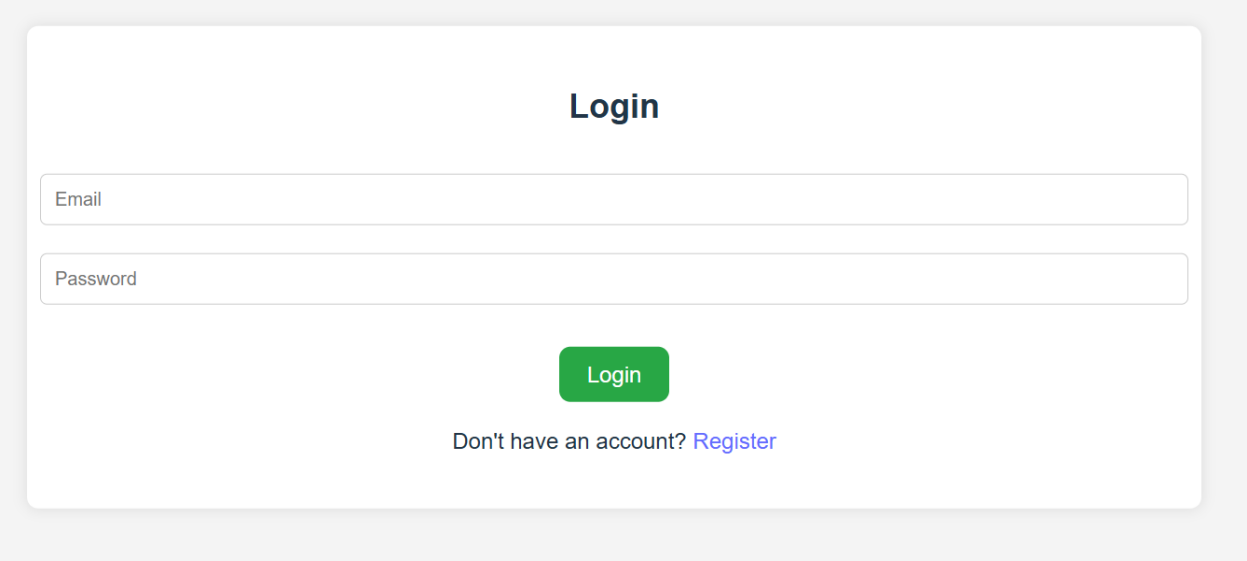
Press n or j to go to the next uncovered block, b, p or k for the previous block.

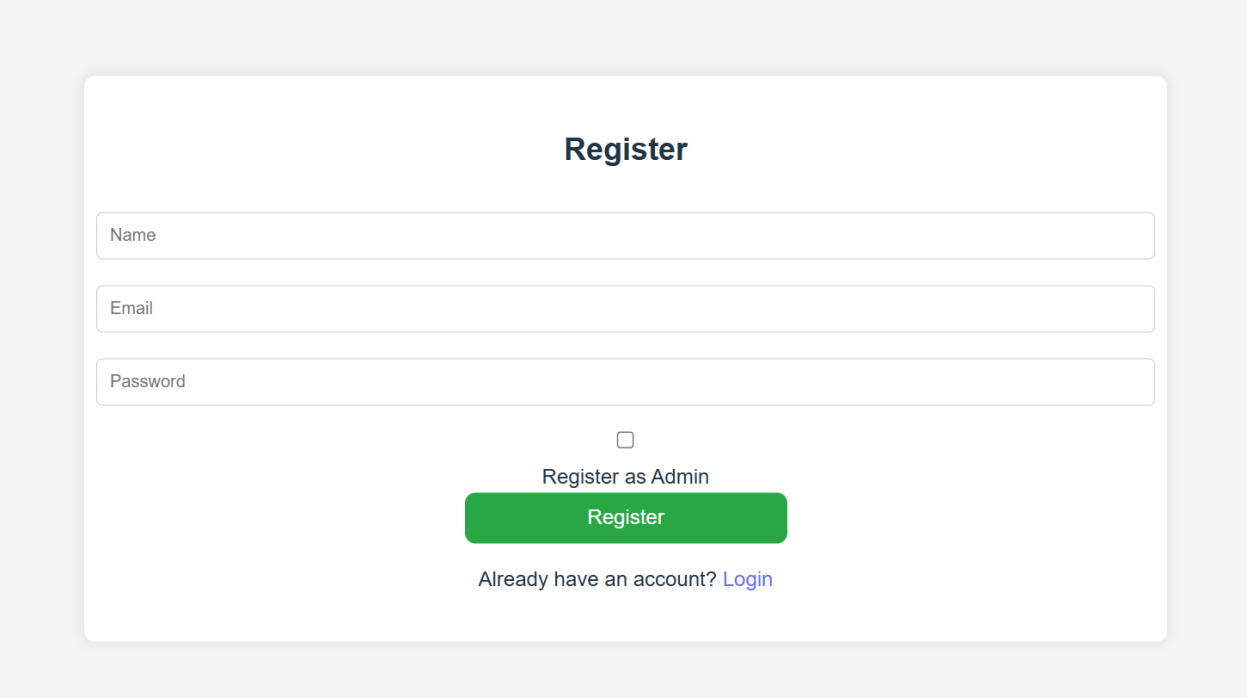
Filter:

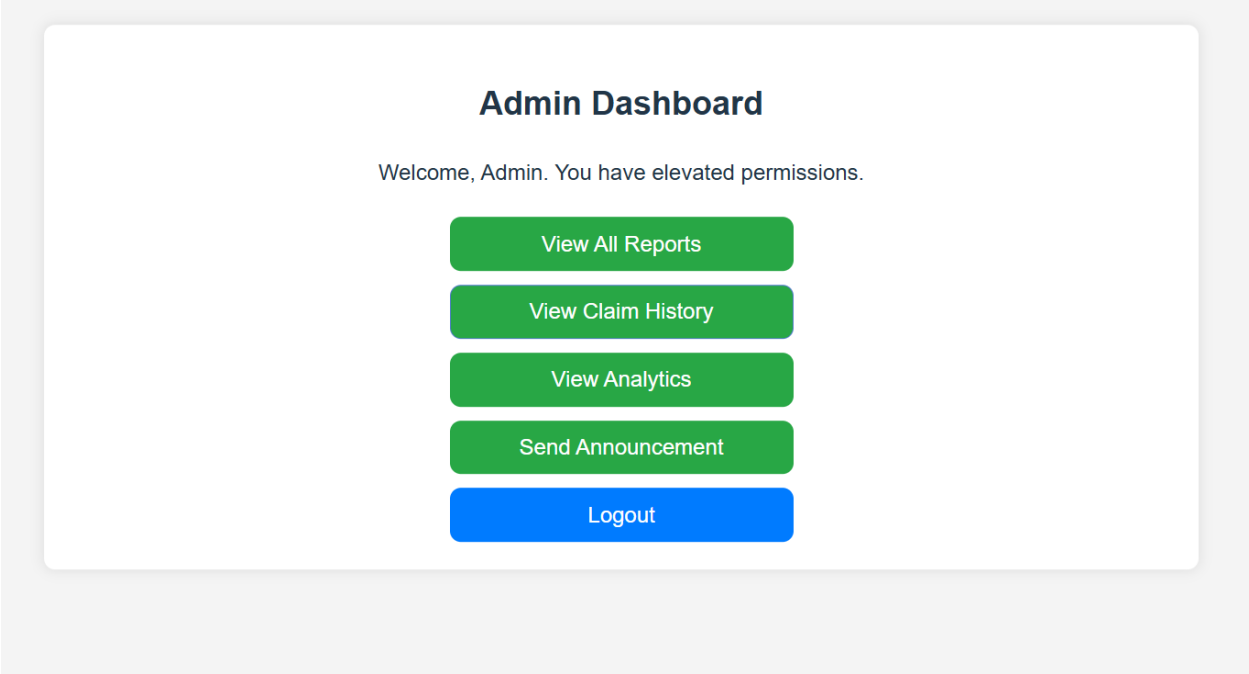
File	Statements		Branches		Functions		Lines	
src	0%	0/41	0%	0/2	0%	0/2	0%	0/41
src/pages	90.4%	688/761	82.52%	85/103	86.15%	56/65	90.4%	688/761

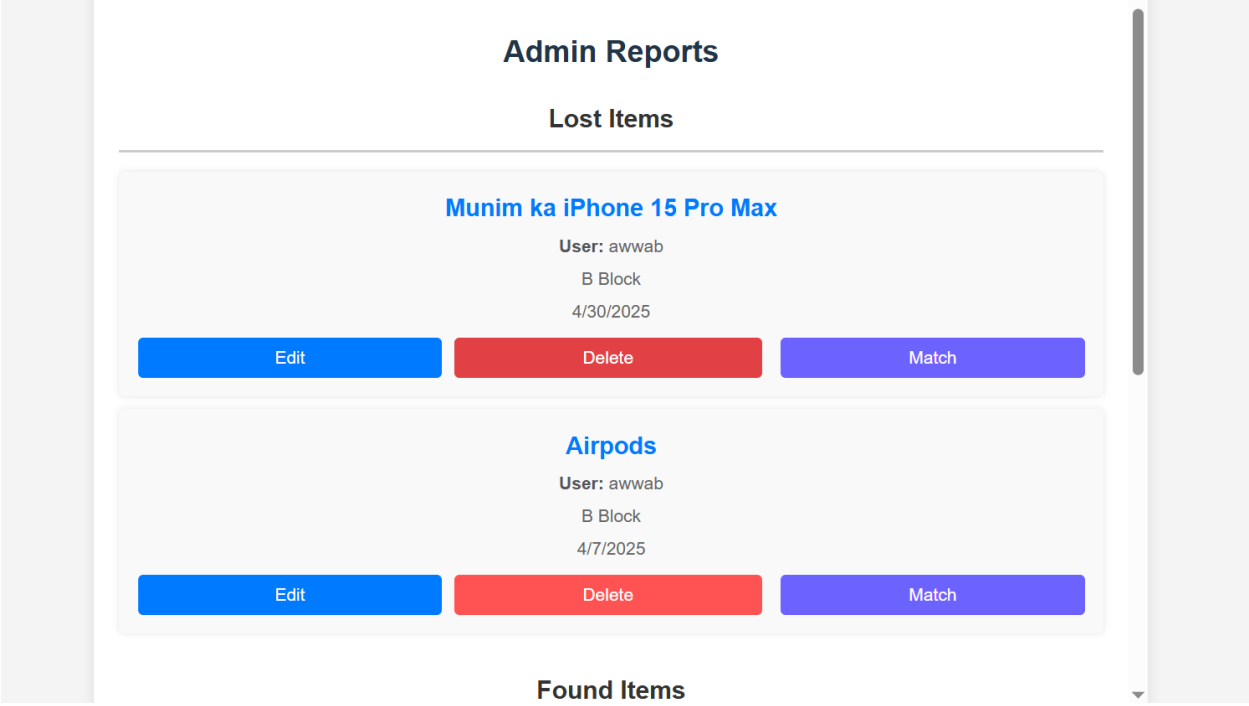
Excluded: App.jsx, main.jsx, setupTests.js

Screenshots of Application






Admin Dashboard
Welcome, Admin. You have elevated permissions.
View All Reports
View Claim History
View Analytics
Send Announcement
Logout


Admin Reports
Lost Items
Munim ka iPhone 15 Pro Max
User: awwab
B Block
4/30/2025
Edit
Delete
Match
Airpods
User: awwab
B Block
4/7/2025
Edit
Delete
Match
Found Items

Send System Announcement

Title

Message

Send Announcement

Report a Lost Item

Item Title

Description

Last Seen Location

Date Lost

mm / dd / yyyy

Choose File No file chosen

Submit

Back to Dashboard

Search for Lost Items

Search by title, description, or location

Select a lost item

Back to Dashboard

Announcements

No new announcements.

Work Division between Group Members

This project was developed collaboratively by three group members: Awwab, Zain, and Abdullah. Each member contributed to distinct areas of the MERN-based Lost & Found Management System to ensure modular and efficient development.

Awwab took the lead on frontend development. He was responsible for designing the user interface using React, implementing user-facing pages such as login, registration, dashboards, and item reporting. He also handled component routing, state management, and ensured that the UI was responsive and well-integrated with backend APIs.

Zain focused on the backend development using Express.js. He created and managed the API routes, controllers, and middleware. He also designed the database schema using Mongoose and connected the application to MongoDB. His responsibilities included implementing user authentication, report handling, and ensuring the server logic was robust and secure.

Abdullah handled administrative functionalities, analytics, and deployment. He developed admin-specific pages such as Admin Dashboard and Analytics, and implemented features for announcements and system monitoring. Additionally, he managed the deployment of both the frontend and backend using services like Netlify and Render, configured environment variables, and prepared project documentation.

The team collaborated closely using Git for version control and maintained regular coordination to ensure seamless integration across the full stack.

Lesson Learnt by Group

Developing a software engineering product using the Scrum methodology provided the group with valuable hands-on experience in agile project management. Working in sprints helped us break down complex tasks into manageable chunks and maintain consistent progress throughout the development cycle. Using Trello as a task management tool allowed us to visualize workflows, prioritize features, and track responsibilities transparently. We learned the importance of daily stand-ups, sprint reviews, and retrospectives in promoting accountability and adaptive planning. Most importantly, the iterative nature of Scrum enabled us to continuously refine both our code and communication, leading to a more organized, collaborative, and efficient development process.

GitHub Link

https://github.com/zaint4hir/SE_Project