

Lost And Found

Mohamed hanif 324056886, Mohamed hanif 212924211

SDD SUBMITTED FOR THE COURSE
Methods in software engineering

Shenkar College
Faculty of Engineering

Content

1. Introduction.....	2
.1.1 System Overview	2
1.2. Purpose	2
1.3. Scope	2
1.4. Constraints.....	2
2. System Architecture – System Context Diagram	3
2.1. Architectural Description and Design: Roles, Activities and Data.....	3
2.2. The Life Cycle of the System	5
3. Design.....	7
3.1. Data Design - Database Description.....	7
3.2. Structural Design - Class Diagram.....	8
3.3. Interactions Design	9
3.3.1. Use Cases.....	9
3.3.2. Sequence Diagram.....	10
3.3.3. Activity Diagram / State / Processes canceled 2024.....	10
3.4. Description of Algorithmic Components	10
3.5. Software Architecture Pattern	12
4. Verification	15
4.1. Validation and Evaluation Plan	15
4.2. Testing Platform	18
5. Project Management	20
5.1. Schedule or Gantt Plan	20
5.2. Team Roles.....	21
6. Main Functional Screens.....	26
7. Appendix-A: POC Discussion	30

1. Introduction

1.1. System Overview

The "Lost and Found" system is designed to streamline the process of reporting and finding lost items. Users can sign up, log in, report lost or found items, and claim them. Administrators verify claims and manage the system. The system aims to enhance the efficiency and accuracy of lost and found processes.

1.2. Purpose

The purpose of this document is to provide a detailed design of the "Lost and Found" system. It outlines the system's architecture, interfaces, and components, ensuring that developers have a clear understanding of the system's requirements and how to implement them. This document serves as a blueprint for developing the system.

1.3. Scope

This document covers the design aspects of the "Lost and Found" system, including:

- User interaction design (e.g., login, report, claim)
- Administrative functions (e.g., verify claims, manage reports)
- Data storage and management
- System interfaces and modules
- Constraints and assumptions

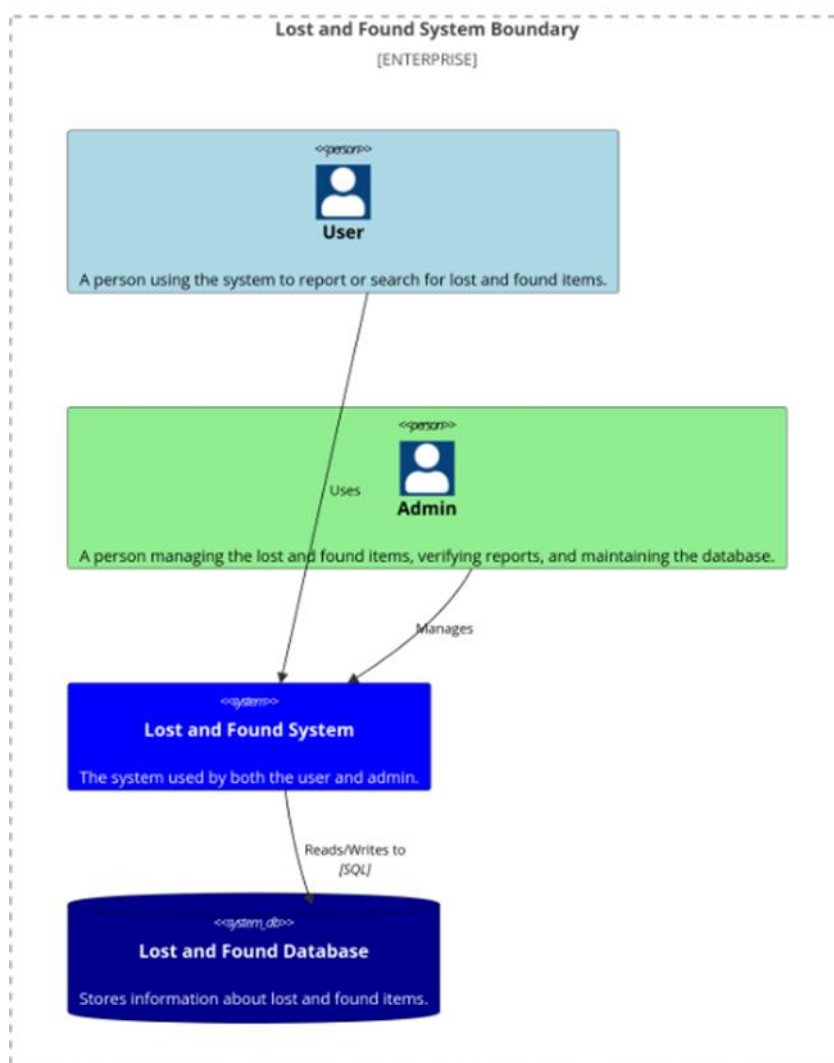
1.4. Constraints

The design of the "Lost and Found" system is subject to the following constraints:

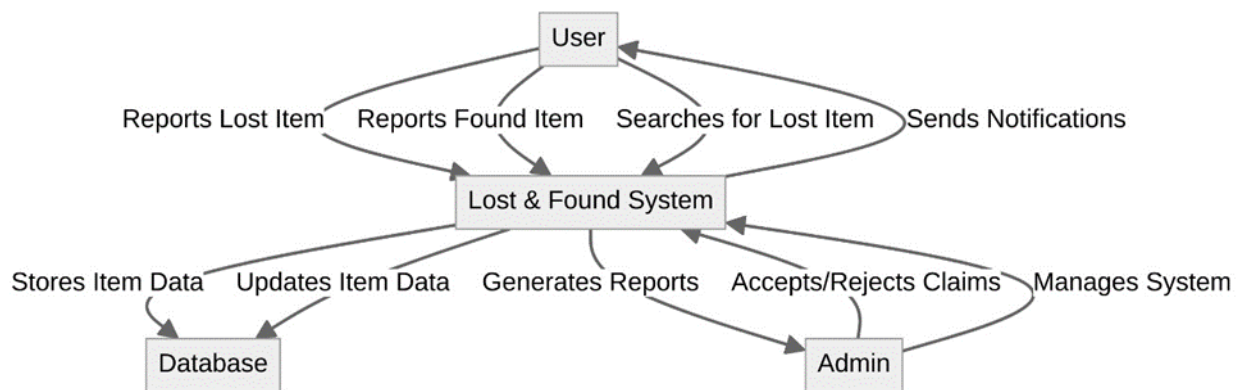
- The system must be accessible via web browsers.
- It should support concurrent users without performance degradation.
- Data must be stored securely to protect user privacy.
- The system should be scalable to handle an increasing number of users and reports.
- The system should adhere to applicable data protection regulations (e.g., GDPR).

2. System Architecture – System Context Diagram

2.1. Architectural Description and Design: Roles, Activities and Data



System Context Diagram



Roles:

1. User:

- **Description:** A person using the system to report or search for lost and found items.
- **Activities:**
 - Reports Lost Item
 - Reports Found Item
 - Searches for Lost Item
 - Receives Notifications

2. Admin:

- **Description:** A person managing the lost and found items, verifying reports, and maintaining the database.
- **Activities:**
 - Accepts/Rejects Claims
 - Manages System
 - Generates Reports

3. Lost and Found System:

- **Description:** The system used by both the user and admin.
- **Activities:**
 - Updates Item Data
 - Stores Item Data

4. **Database:**

- **Description:** Stores information about lost and found items.
- **Activities:**
 - Reads/Writes Item Data

Data:

- **User Data:** Information related to users, including credentials, contact information, and user roles.
- **Item Reports:** Details of lost and found items, including descriptions, locations, and timestamps.
- **Claims:** Records of users claiming found items, including verification status.
- **Notifications:** Messages sent to users regarding their reports and claims.

System Context Diagram:

Below is a more detailed context diagram showing the interactions among different roles and the system components:

2.2. The Life Cycle of the System

The "Lost and Found" system follows a life cycle that includes the following stages:

1. **Initiation:**
 - Requirement analysis and feasibility study.
 - Project planning and resource allocation.
2. **Design:**
 - High-level architectural design.
 - Detailed design of modules and interfaces (as outlined in the SDD).

3. Implementation:

- Development of the system's components and integration.
- Database setup and configuration.
- Implementation of user and admin interfaces.

4. Testing:

- Unit testing of individual components.
- Integration testing to ensure modules work together.
- User acceptance testing to validate the system meets requirements.

5. Deployment:

- Installation of the system on production servers.
- Configuration of the environment and initial data load.
- User training and documentation.

6. Maintenance:

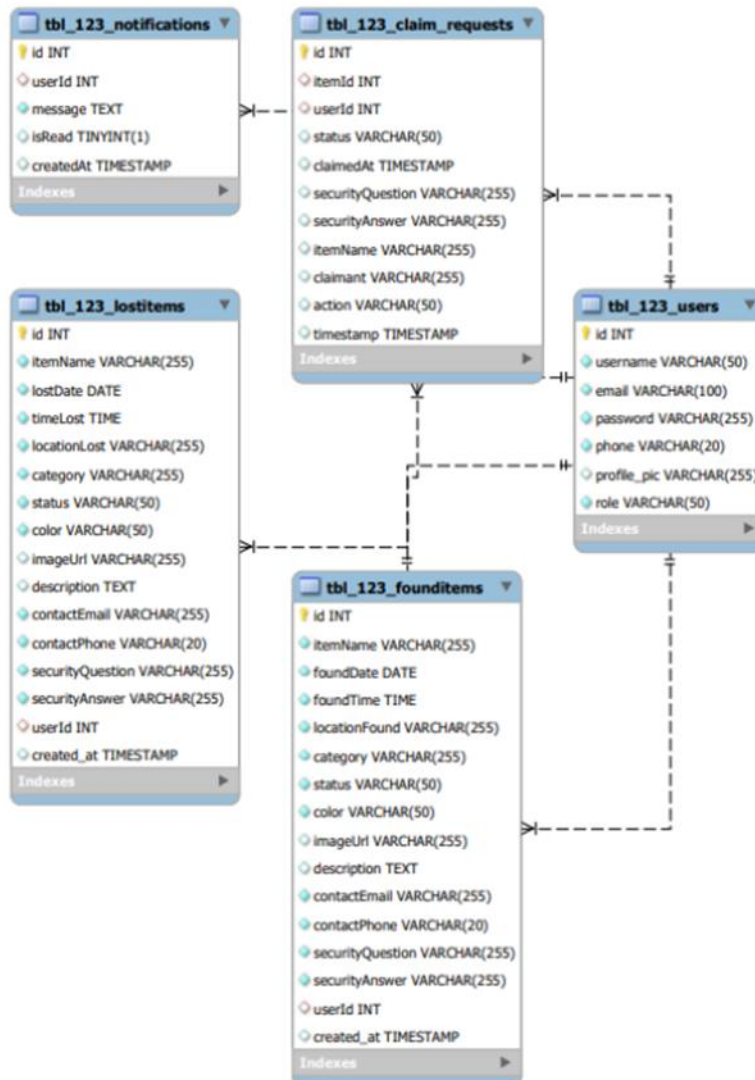
- Monitoring system performance and user feedback.
- Bug fixing and updates to enhance functionality.
- Periodic reviews and improvements based on user needs and technology advancements.

7. Decommissioning:

- Archiving of data and user information.
- Shutdown of the system and resources reallocation.

3. Design

3.1. Data Design - Database Description

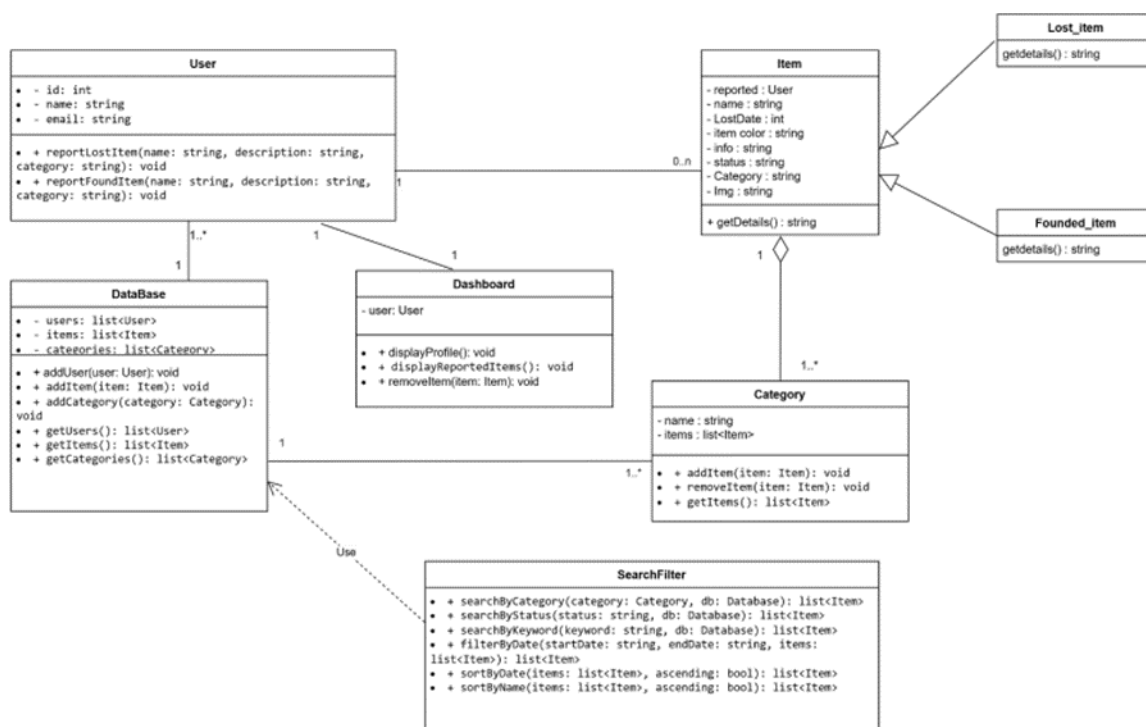


Relationships Summary

- **Users Table (tbl_123_users):**
 - **One-to-Many:** Each user can report multiple lost items (tbl_123_lostitems) and found items (tbl_123_founditems).
 - **One-to-Many:** Each user can make multiple claim requests (tbl_123_claim_requests).
 - **One-to-Many:** Each user can receive multiple notifications (tbl_123_notifications).
- **Lost Items Table (tbl_123_lostitems) and Found Items Table (tbl_123_founditems):**
 - **Many-to-One:** Each lost/found item is reported by one user (tbl_123_users).

- **One-to-Many:** Each lost/found item can have multiple claim requests (tbl_123_claim_requests).
- **Claim Requests Table (tbl_123_claim_requests):**
 - **Many-to-One:** Each claim request is made by one user (tbl_123_users).
 - **Many-to-One:** Each claim request is associated with one lost/found item (tbl_123_lostitems or tbl_123_founditems).
- **Notifications Table (tbl_123_notifications):**
 - **Many-to-One:** Each notification is sent to one user (tbl_123_users).

3.2. Structural Design - Class Diagram

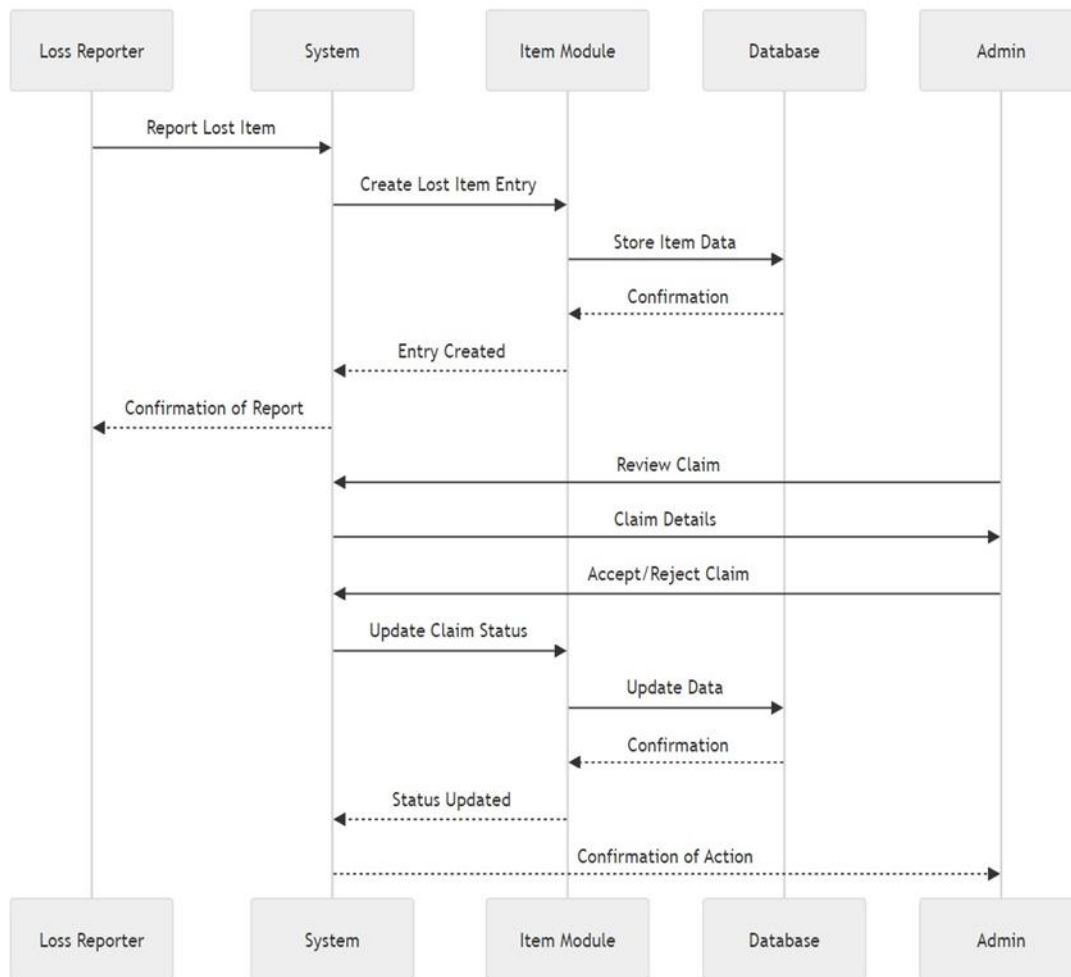


3.3. Interactions Design

3.3.1. Use Cases



3.3.2. Sequence Diagram



3.3.3. ~~Activity Diagram / State / Processes~~ canceled 2024

3.4. Description of Algorithmic Components

In the "Lost and Found" system, several algorithmic components are essential for the proper functioning of the application. These components handle various operations such as searching, matching items, notifying users, and managing claims. Below is a description of the key algorithmic components.

1. Item Matching Algorithm

Purpose: To match found items with reported lost items based on their attributes.

Steps:

1. Retrieve all lost and found items from the database.
2. Compare attributes such as itemName, location, category, color, and date.
3. Calculate a similarity score for each potential match.

4. If the similarity score exceeds a predefined threshold, mark the items as potential matches.

Notify users of potential matches.

2. Notification Algorithm

Purpose: To send notifications to users about the status of their reports and potential matches.

Steps:

1. Identify events that trigger notifications (e.g., new item reported, item matched, claim status changed).
2. Generate notification messages based on the event.

Send notifications to the relevant users.

3. Search Algorithm

Purpose: To enable users to search for lost or found items using various filters.

Steps:

1. Retrieve user search inputs (e.g., keywords, location, category).
2. Query the database using the search inputs.
3. Rank the search results based on relevance.
4. Return the ranked search results to the user.

4. Claim Verification Algorithm

Purpose: To verify the validity of claims made by users for found items.

Steps:

1. Retrieve the claim request and associated item details.
2. Verify the claimant's details (e.g., security question and answer).
3. Compare the claimant's details with the item's report details.
4. Approve or reject the claim based on verification results.

Update the claim status and notify the claimant.

3.5. Software Architecture Pattern

The "Lost and Found" system employs an N-tier architecture pattern combined with the MVC (Model-View-Controller) design pattern for the Logic tier. The implementation uses HTML, CSS, and JavaScript for the frontend, with Express and Node.js for the backend, and SQL for database operations.

N-Tier Architecture

1. Presentation Tier (Client Layer)

- **Description:** This layer handles the user interface and user interaction.
- **Technologies:** HTML, CSS, JavaScript
- **Responsibilities:**
 - Displaying forms for reporting lost and found items.
 - Sending user inputs to the backend.
 - Displaying results and notifications to the users.

2. Logic Tier (Application Layer)

- **Description:** This layer contains the application's business logic.
- **Technologies:** Express, Node.js
- **Responsibilities:**
 - Handling incoming requests from the Presentation tier.
 - Processing business logic and interacting with the database.
 - Sending responses back to the Presentation tier.

3. Data Tier (Database Layer)

- **Description:** This layer is responsible for data storage and management.
- **Technologies:** SQL
- **Responsibilities:**
 - Storing user information, item reports, claims, and notifications.
 - Performing CRUD operations requested by the Logic tier.

MVC Architecture within the Logic Tier

The Logic tier follows the MVC architecture pattern to organize the application into three interconnected components: Model, View, and Controller.

1. **Model**

- **Description:** Represents the data and business logic.
- **Responsibilities:**
 - Defining data schemas and models.
 - Interacting with the database to retrieve and store data.
 - Implementing business logic.

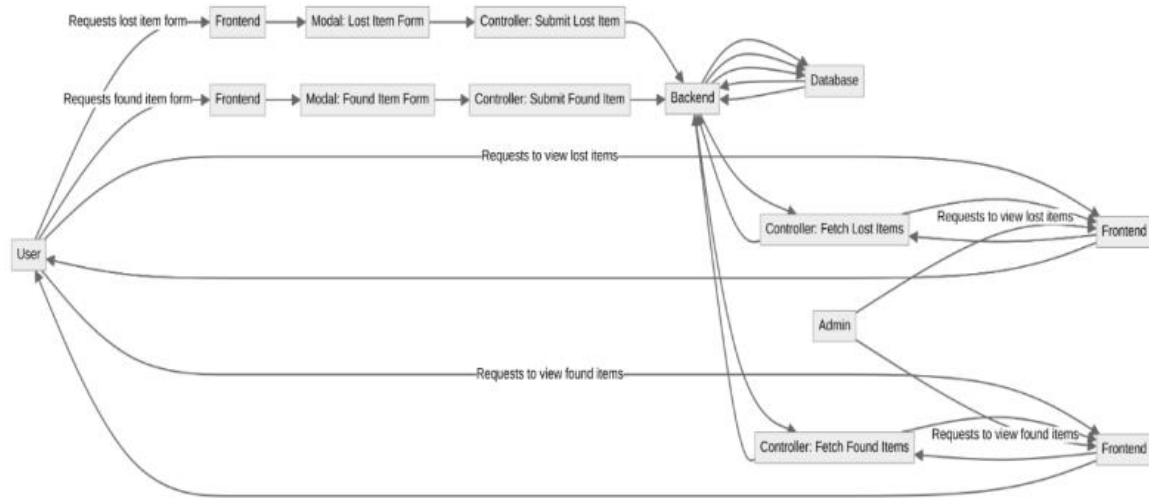
2. **View**

- **Description:** Represents the presentation layer within the Logic tier.
- **Responsibilities:**
 - Rendering the UI components.
 - Displaying data received from the Controller.
 - Capturing user interactions and passing them to the Controller.

3. **Controller**

- **Description:** Acts as an intermediary between the Model and the View.
- **Responsibilities:**
 - Handling user inputs from the View.
 - Interacting with the Model to process data.
 - Updating the View with the processed data.

Diagram of N-Tier Architecture with MVC



Description of the Diagram:

- **User Interactions:**
 - Users request lost or found item forms via the frontend.
 - The frontend displays the appropriate form (Lost Item Form, Found Item Form).
- **Form Submissions:**
 - Users submit the forms.
 - The frontend sends the form data to the backend controller (Controller: Submit Lost Item, Controller: Submit Found Item).
- **Backend Processing:**
 - The backend controller processes the form submissions.
 - The controller interacts with the database to store the item data.
- **Requests to View Items:**
 - Users request to view lost or found items.

- The controller fetches the requested items from the database (Controller: Fetch Lost Items, Controller: Fetch Found Items).
- The controller sends the fetched data to the frontend.
- **Admin Interactions:**
 - Admins manage the system, including verifying claims and managing reports.

Technologies Used:

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Node.js, Express
- **Database:** SQL

4. Verification

4.1. Validation and Evaluation Plan

The validation and evaluation plan for the "Lost and Found" system outlines the strategies and methods to ensure that the system meets its requirements and functions correctly. This plan includes identifying key factors to test, designing repeated experiments, and measurements to improve the final result. The outcomes of these evaluations will be documented in the course project report.

Validation Objectives:

1. Ensure that the system meets all functional requirements.
2. Verify that the system performs reliably under expected and peak loads.
3. Confirm that the user interface is intuitive and user-friendly.
4. Validate data accuracy and integrity in the database.
5. Ensure secure handling of user data and system operations.

Evaluation Criteria:

- **Functionality:** Does the system perform the required tasks correctly?
- **Performance:** How does the system perform under different loads?
- **Usability:** Is the system easy to use and navigate?

- **Security:** Are the security measures effective in protecting user data?
- **Compatibility:** Does the system work across different devices and browsers?

Validation Methods:

1. Unit Testing:

- **Objective:** Verify that individual components (functions, methods, classes) work as intended.
- **Tools:** Mocha, Chai (JavaScript testing frameworks).
- **Process:** Write and execute tests for each function and method in isolation.
- **Frequency:** Continuous during development.

2. Integration Testing:

- **Objective:** Ensure that different components of the system work together as expected.
- **Tools:** Postman (for API testing), Supertest.
- **Process:** Test the integration points between modules, such as the interaction between the frontend and backend.
- **Frequency:** After significant development milestones.

3. System Testing:

- **Objective:** Validate the complete and integrated system to ensure it meets the specified requirements.
- **Tools:** Selenium (for automated browser testing), Manual Testing.
- **Process:** Execute end-to-end scenarios covering all functional requirements.
- **Frequency:** Before each release.

4. Performance Testing:

- **Objective:** Assess the system's performance under various conditions, including stress and load testing.
- **Tools:** JMeter, LoadRunner.
- **Process:** Simulate different user loads and measure response times, throughput, and resource utilization.
- **Frequency:** During pre-release and after significant changes.

5. Usability Testing:

- **Objective:** Evaluate the user interface and overall user experience.
- **Tools:** User surveys, usability testing sessions.
- **Process:** Conduct usability tests with real users to gather feedback on the UI/UX.
- **Frequency:** After initial development and periodically during development.

6. Security Testing:

- **Objective:** Identify and fix security vulnerabilities.
- **Tools:** OWASP ZAP, Burp Suite.
- **Process:** Perform security assessments including vulnerability scanning, penetration testing, and code reviews.
- **Frequency:** Before deployment and regularly during maintenance.

7. Database Testing:

- **Objective:** Ensure data integrity, consistency, and performance of the database.
- **Tools:** SQL queries, automated database testing tools.
- **Process:** Validate data storage, retrieval operations, and ensure that data relationships and constraints are correctly enforced.
- **Frequency:** During integration testing and system testing.

8. Compatibility Testing:

- **Objective:** Verify that the system works across different devices and browsers.
- **Tools:** BrowserStack, CrossBrowserTesting.
- **Process:** Test the application on various devices, operating systems, and browsers to ensure compatibility.
- **Frequency:** Before each major release.

Metrics and Measurements:

- **Bug Count:** Number of bugs identified and resolved.
- **Test Coverage:** Percentage of code covered by tests.
- **Response Time:** Average time taken to respond to user actions.
- **Throughput:** Number of transactions processed in a given time period.

- **User Satisfaction:** Feedback from usability tests and surveys.
- **Security Incidents:** Number and severity of security vulnerabilities found.

Reporting and Documentation:

- Document all test cases, test results, and identified issues.
- Regularly update the project report with findings from the validation and evaluation processes.
- Include summaries of test results, performance metrics, and user feedback in the final project report.

4.2. Testing Platform

The "Lost and Found" system's testing platform includes a combination of automated and manual testing tools and environments. These tools and methods will ensure the system meets its functional, performance, security, and usability requirements.

Automated Testing Tools

1. Unit Testing

- **Tools:** Mocha, Chai
- **Purpose:** Verify that individual components work as intended.
- **Usage:** Test all modules, including user authentication, item reporting, and notification functionalities.

2. Integration Testing

- **Tools:** Postman, Supertest
- **Purpose:** Ensure different system components work together correctly.
- **Usage:** Test API endpoints for item reporting, user login, and claim processing.

3. System Testing

- **Tools:** Selenium
- **Purpose:** Validate the complete and integrated system to ensure it meets specified requirements.
- **Usage:** Execute end-to-end scenarios covering all functional requirements.

4. Performance Testing

- **Tools:** JMeter
- **Purpose:** Assess the system's performance under various conditions, including stress and load testing.
- **Usage:** Simulate different user loads and measure response times, throughput, and resource utilization.

5. Security Testing

- **Tools:** OWASP ZAP
- **Purpose:** Identify and mitigate security vulnerabilities.
- **Usage:** Perform automated scans and manual testing for common security issues like SQL injection and XSS.

Manual Testing Tools

1. Usability Testing

- **Tools:** User surveys, usability testing sessions
- **Purpose:** Evaluate the user interface and overall user experience.
- **Usage:** Conduct usability tests with real users to gather feedback on the UI/UX.

2. Compatibility Testing

- **Tools:** BrowserStack
- **Purpose:** Verify that the system works across different devices and browsers.
- **Usage:** Test the application on various devices, operating systems, and browsers to ensure compatibility.

3. Manual Exploratory Testing

- **Purpose:** Uncover issues that automated tests might miss by performing manual testing without predefined test cases.
- **Usage:** Test various user scenarios and edge cases manually to identify unexpected issues.

Testing Environments

1. Development Environment

- **Purpose:** Used by developers for writing and running unit tests during development.
- **Tools:** Local development setup with Node.js, Express, and SQL.

2. Staging Environment

- **Purpose:** Used for integration and system testing before deploying to production.
- **Tools:** Hosted environment similar to production, with automated deployment and testing pipelines.

3. Production Environment

- **Purpose:** Used for performance and security testing to ensure the system meets real-world demands.
- **Tools:** Live production environment with monitoring and logging tools.

Testing Strategy Summary

- **Automated Testing:** Focus on unit testing, integration testing, system testing, performance testing, and security testing using tools like Mocha, Chai, Postman, Supertest, Selenium, JMeter, and OWASP ZAP.
- **Manual Testing:** Emphasize usability testing, compatibility testing, and manual exploratory testing to complement automated tests.
- **Testing Environments:** Utilize development, staging, and production environments to ensure comprehensive testing across different stages of development.

By implementing this robust testing platform, the "Lost and Found" system will be thoroughly validated and evaluated, ensuring it meets the highest standards of quality and reliability.

5. Project Management

5.1. Schedule or Gantt Plan

6. Below is the updated Gantt chart for the "Lost and Found" system project, covering the period from April to August.

Task	Start Date	End Date	Duration	Dependencies
Initiation	01/04/2024	05/04/2024	5 days	-
Requirement Analysis	01/04/2024	03/04/2024	3 days	-
Feasibility Study	04/04/2024	05/04/2024	2 days	Requirement Analysis
Design	06/04/2024	15/04/2024	10 days	Initiation
High-Level Design	06/04/2024	08/04/2024	3 days	Feasibility Study
Detailed Design	09/04/2024	15/04/2024	7 days	High-Level Design
Implementation	16/04/2024	30/06/2024	45 days	Design
Frontend Development	16/04/2024	06/05/2024	21 days	Detailed Design
Backend Development	16/04/2024	06/05/2024	21 days	Detailed Design
Database Setup	07/05/2024	10/05/2024	4 days	Detailed Design
Integration	11/05/2024	20/05/2024	10 days	Frontend, Backend, DB
Testing	01/07/2024	15/07/2024	15 days	Implementation
Unit Testing	01/07/2024	05/07/2024	5 days	Implementation
Integration Testing	06/07/2024	10/07/2024	5 days	Unit Testing
System Testing	11/07/2024	15/07/2024	5 days	Integration Testing
Deployment	16/07/2024	20/07/2024	5 days	Testing
Production Deployment	16/07/2024	20/07/2024	5 days	System Testing
Maintenance	21/07/2024	31/08/2024	Ongoing	Deployment

6.1. Team Roles

Project Manager

- **Role:** Oversee the entire project, ensure timelines are met, and manage resources.
- **Responsibilities:**
 - Create and maintain the project plan.
 - Coordinate team activities and communications.
 - Manage risks and issues.
 - Liaise with stakeholders.

Frontend Developer

- **Role:** Develop and maintain the user interface.
- **Responsibilities:**
 - Implement UI components using HTML, CSS, and JavaScript.
 - Ensure the frontend is responsive and user-friendly.

- Integrate frontend with backend APIs.
- Perform unit testing of frontend components.

Backend Developer

- **Role:** Develop and maintain server-side logic.
- **Responsibilities:**
 - Implement APIs using Node.js and Express.
 - Integrate with the database.
 - Ensure security and data integrity.
 - Perform unit testing of backend components.

Database Administrator (DBA)

- **Role:** Manage and maintain the database.
- **Responsibilities:**
 - Design and implement the database schema.
 - Optimize database performance.
 - Ensure data security and backup.
 - Support integration with backend services.

Quality Assurance (QA) Engineer

- **Role:** Ensure the system meets quality standards.
- **Responsibilities:**
 - Develop and execute test plans.
 - Perform unit, integration, system, and performance testing.
 - Report and track bugs.
 - Verify bug fixes and perform regression testing.

UX/UI Designer

- **Role:** Design the user interface and experience.
- **Responsibilities:**

- Create wireframes, mockups, and prototypes.
- Conduct user research and usability testing.
- Ensure the design is user-friendly and meets user requirements.
- Work closely with frontend developers to implement designs.

DevOps Engineer

- **Role:** Manage deployment and continuous integration.
- **Responsibilities:**
 - Set up and maintain CI/CD pipelines.
 - Manage cloud infrastructure and environments.
 - Monitor system performance and reliability.
 - Ensure smooth deployment processes.

Business Analyst

- **Role:** Gather and analyze requirements.
- **Responsibilities:**
 - Conduct stakeholder interviews and workshops.
 - Document requirements and create user stories.
 - Ensure requirements are clear and understood by the team.
 - Support the team in understanding business needs.

Support Engineer

- **Role:** Provide technical support post-deployment.
- **Responsibilities:**
 - Monitor the system for issues.
 - Respond to user queries and issues.
 - Perform routine maintenance tasks.
 - Liaise with development team for bug fixes and updates.

By clearly defining these roles and responsibilities, the project team can work efficiently and collaboratively, ensuring the successful development and deployment of the "Lost and Found" system.

Project Manager

- **Role:** Oversee the entire project, ensure timelines are met, and manage resources.
- **Responsibilities:**
 - Create and maintain the project plan.
 - Coordinate team activities and communications.
 - Manage risks and issues.
 - Liaise with stakeholders.

Frontend Developer

- **Role:** Develop and maintain the user interface.
- **Responsibilities:**
 - Implement UI components using HTML, CSS, and JavaScript.
 - Ensure the frontend is responsive and user-friendly.
 - Integrate frontend with backend APIs.
 - Perform unit testing of frontend components.

Backend Developer

- **Role:** Develop and maintain server-side logic.
- **Responsibilities:**
 - Implement APIs using Node.js and Express.
 - Integrate with the database.
 - Ensure security and data integrity.
 - Perform unit testing of backend components.

Database Administrator (DBA)

- **Role:** Manage and maintain the database.
- **Responsibilities:**
 - Design and implement the database schema.
 - Optimize database performance.
 - Ensure data security and backup.
 - Support integration with backend services.

Quality Assurance (QA) Engineer

- **Role:** Ensure the system meets quality standards.
- **Responsibilities:**
 - Develop and execute test plans.
 - Perform unit, integration, system, and performance testing.
 - Report and track bugs.
 - Verify bug fixes and perform regression testing.

UX/UI Designer

- **Role:** Design the user interface and experience.
- **Responsibilities:**
 - Create wireframes, mockups, and prototypes.
 - Conduct user research and usability testing.
 - Ensure the design is user-friendly and meets user requirements.
 - Work closely with frontend developers to implement designs.

DevOps Engineer

- **Role:** Manage deployment and continuous integration.
- **Responsibilities:**
 - Set up and maintain CI/CD pipelines.
 - Manage cloud infrastructure and environments.
 - Monitor system performance and reliability.
 - Ensure smooth deployment processes.

Business Analyst

- **Role:** Gather and analyze requirements.
- **Responsibilities:**
 - Conduct stakeholder interviews and workshops.
 - Document requirements and create user stories.
 - Ensure requirements are clear and understood by the team.
 - Support the team in understanding business needs.

Support Engineer

- **Role:** Provide technical support post-deployment.
- **Responsibilities:**
 - Monitor the system for issues.
 - Respond to user queries and issues.
 - Perform routine maintenance tasks.
 - Liaise with the development team for bug fixes and updates.

7. Main Functional Screens

The "Lost and Found" system will feature several key functional screens to facilitate the core functionalities of the application. Below are descriptions of the main functional screens, their purposes, and the primary elements they will contain.

1. Home Screen

Purpose: Provide an overview of the system and allow users to navigate to different sections.

Key Elements:

- Welcome message
- Navigation bar (links to Login, Sign Up, Report Lost Item, Report Found Item, Search)
- Brief instructions on how to use the system
- Latest lost and found items summary

2. User Registration Screen

Purpose: Allow new users to create an account.

Key Elements:

- Input fields for username, email, password, phone number
- Profile picture upload
- Submit button
- Link to Login screen

3. User Login Screen

Purpose: Allow existing users to log into their account.

Key Elements:

- Input fields for username and password
- Submit button
- Link to Registration screen

4. Dashboard Screen

Purpose: Provide users with an overview of their account activities and quick access to main functions.

Key Elements:

- User greeting
- Summary of user's reported items (lost and found)
- Notifications summary
- Quick links to report lost/found items, search items, view claims

5. Report Lost Item Screen

Purpose: Allow users to report a lost item.

Key Elements:

- Input fields for item name, lost date, lost time, location, category, color, description
- Upload button for item image
- Contact information fields (email, phone)
- Security question and answer fields
- Submit button

6. Report Found Item Screen

Purpose: Allow users to report a found item.

Key Elements:

- Input fields for item name, found date, found time, location, category, color, description
- Upload button for item image
- Contact information fields (email, phone)
- Security question and answer fields
- Submit button

7. Search Items Screen

Purpose: Allow users to search for lost or found items.

Key Elements:

- Input fields for search criteria (keywords, location, category, date range)
- Filter options (lost/found, status)
- Search button
- Display area for search results (list of items with basic details and links to full details)

8. Item Details Screen

Purpose: Display detailed information about a specific lost or found item.

Key Elements:

- Item name, description, and image
- Details (date, time, location, category, color)
- Contact information
- Link to claim the item
- Link to return to search results

9. Claim Item Screen

Purpose: Allow users to claim a found item.

Key Elements:

- Display of item details
- Input fields for user's security question and answer
- Submit button for claim request
- Confirmation message after submission

10. Admin Dashboard Screen

Purpose: Provide administrators with an overview of system activities and management functions.

Key Elements:

- Summary of system statistics (number of users, items reported, claims made)
- List of pending claims for verification
- Links to manage user accounts, view reports, and system settings

11. Verify Claim Screen (Admin)

Purpose: Allow administrators to verify and process item claims.

Key Elements:

- List of pending claims
- Detailed view of each claim (item details, claimant details, security question and answer)
- Approve and reject buttons
- Input field for verification notes
- Confirmation message after processing

12. Notifications Screen

Purpose: Display notifications to the user.

Key Elements:

- List of notifications with status (read/unread)
- Details of each notification (message content, date, and time)
- Mark as read button

Link to related item or action

8. Appendix-A: POC Discussion

Objective

The primary objective of developing a Proof of Concept (POC) was to mitigate the biggest risk identified during the project planning phase: ensuring the seamless integration and functionality of the core components of the "Lost and Found" system. The POC aimed to validate our ability to implement the key features of the system, including item reporting, user authentication, data storage, and the additional security question feature for item claims.

Developed Components

1. User Authentication

- **Description:** Implemented a basic user registration and login system.
- **Purpose:** To ensure we can securely manage user accounts and authenticate users.
- **Components:**
 - Registration form
 - Login form
 - Backend API for handling authentication
 - Database integration for storing user credentials

2. Item Reporting

- **Description:** Developed forms for reporting lost and found items.
- **Purpose:** To verify that users can submit item details and that the system can store and retrieve these reports.
- **Components:**
 - Lost Item Report form
 - Found Item Report form
 - Backend API for submitting item reports
 - Database tables for storing item data

3. Data Storage

- **Description:** Set up a basic SQL database.
- **Purpose:** To confirm that our data schema is effective and that we can perform CRUD (Create, Read, Update, Delete) operations.
- **Components:**

- Database tables for users, lost items, and found items
- Integration with the backend API

4. Security Question for Item Claims

- **Description:** Added a security question feature that users must answer correctly to claim a found item.
- **Purpose:** To enhance the security of the item claiming process and ensure only rightful owners can claim their items.
- **Components:**
 - Security question and answer fields in the lost and found item reporting forms
 - Claim Item form that includes the security question
 - Backend API for verifying the security question answer during the claim process
 - Database integration to store and retrieve security questions and answers

Discoveries and Findings

1. User Authentication:

- **Findings:** Successfully implemented user registration and login. Identified the need for enhanced security measures, such as password hashing and input validation.
- **Challenges:** Initial issues with session management were resolved by implementing JWT (JSON Web Token) for stateless authentication.

2. Item Reporting:

- **Findings:** Successfully created forms and APIs for item reporting. Validated that the frontend can send data to the backend, and the backend can store and retrieve this data.
- **Challenges:** Encountered challenges with file uploads for item images. Resolved by integrating a file storage solution (e.g., AWS S3).

3. Data Storage:

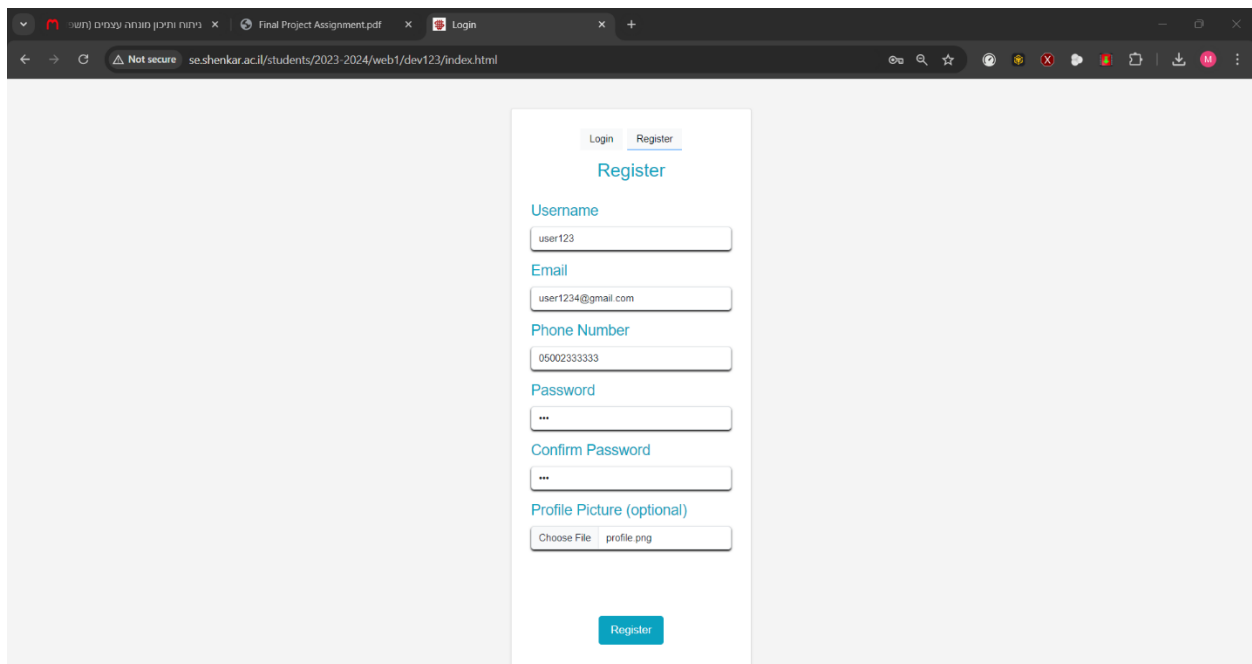
- **Findings:** Successfully set up the SQL database and performed basic CRUD operations. Confirmed that the data schema supports the required operations.
- **Challenges:** Required optimization of database queries to improve performance.

4. Security Question for Item Claims:

- **Findings:** Successfully implemented the security question feature for claiming items. Validated that the security question and answer are correctly stored and retrieved, and that the claim process is secure.
- **Challenges:** Ensuring the security question process is user-friendly while maintaining high security. Resolved by testing different question formats and providing clear instructions to users.

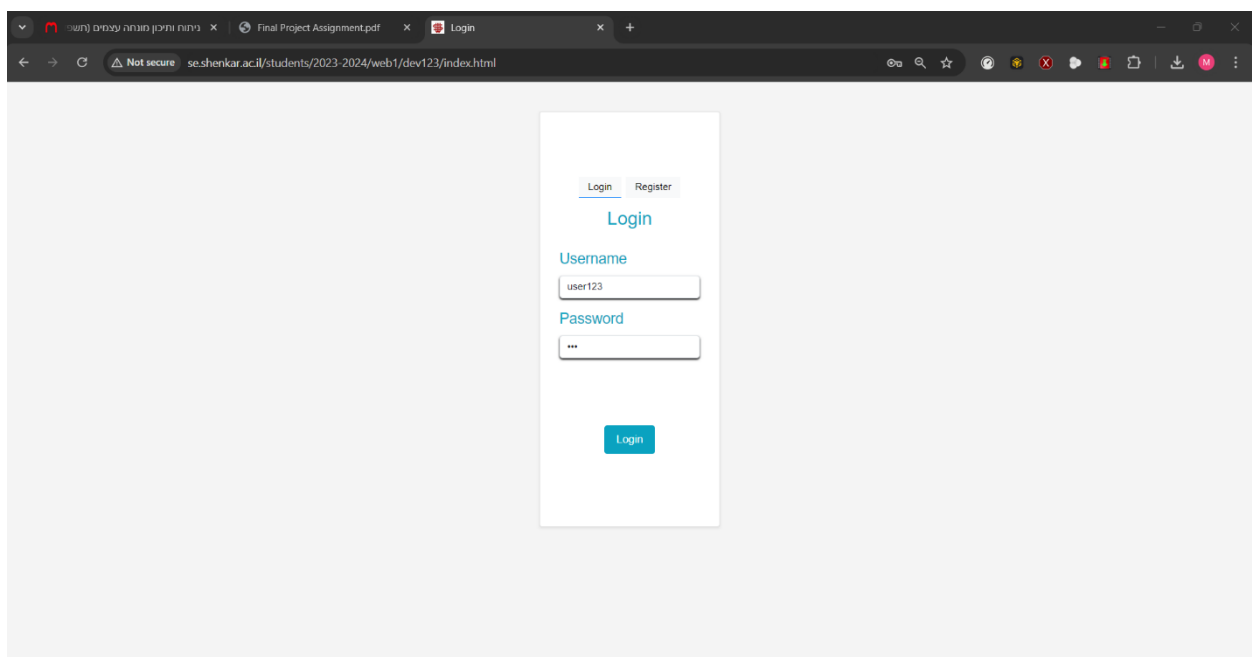
Screenshot of the POC

Below is a screenshot of the POC demonstrating the user registration, item reporting, and security question feature for item claims.



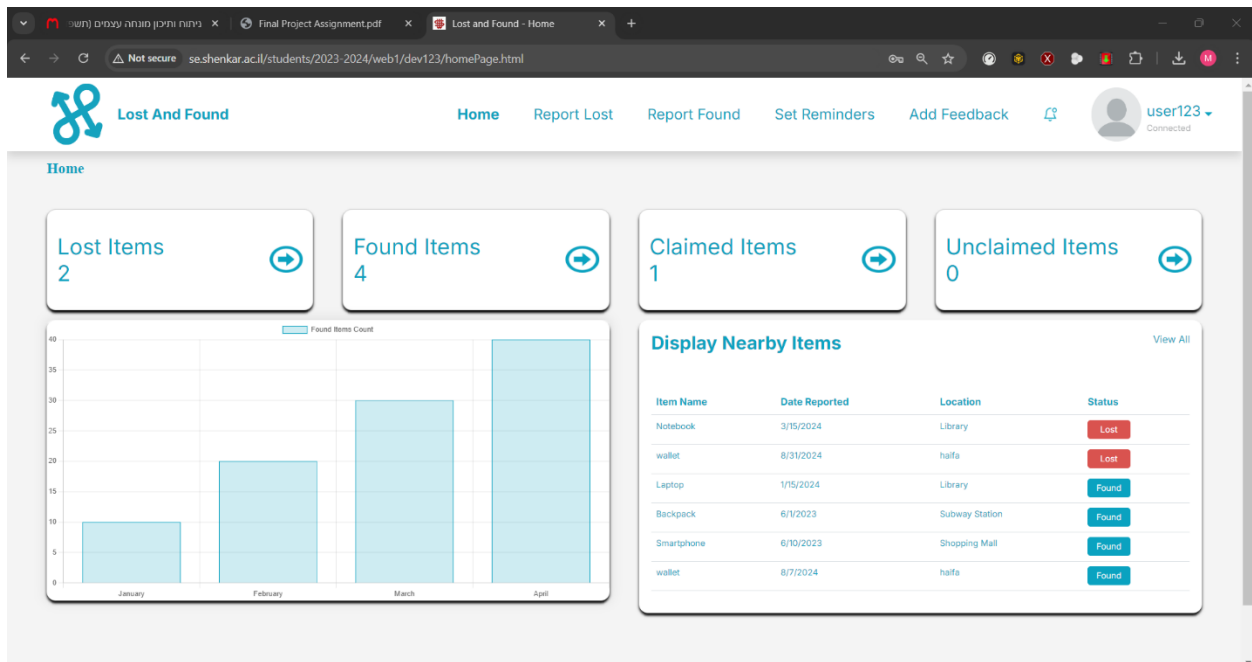
The screenshot shows a web browser window with the URL `se.shenkar.ac.il/students/2023-2024/web1/dev123/index.html`. The page displays a registration form with the following fields and labels:

- Register** (tab)
- Username**: `user123`
- Email**: `user1234@gmail.com`
- Phone Number**: `0500233333`
- Password**: `***`
- Confirm Password**: `***`
- Profile Picture (optional)**: `Choose File` (button) and `profile.png` (text)
- Register** (button)



The screenshot shows the same web browser window, but the page displays a login form with the following fields and labels:

- Login** (tab)
- Username**: `user123`
- Password**: `***`
- Login** (button)



Lost And Found

Home Report Lost Report Found Set Reminders Add Feedback user123 Connected

Home / Report Lost Item

Report Lost Item

Item Name Location Lost Date Time Lost Category Description (Optional) Contact Information

galaxy phone haifa 02/08/2024 10:00 AM Personal Items Please Enter Description user1234@gmail.com 05099999

Add Image

Choose File phone.jpeg

Accepted formats: JPEG, PNG, GIF

Submit

Item reported successfully!

OK

Report Found Item

Item Name

Location Found

Found Date

Found Time

Category

Item Color

Description (Optional)

Contact Information

Add Image

 phone.jpeg

Choose file

Accepted formats: JPEG, PNG, GIF

Security Question

Answer

[Report Found](#)[Set Reminders](#)[Add Feedback](#)

user123

Connected

Your lost item "galaxy phone" might have been found. Check the found items list. [View Item](#)

Items



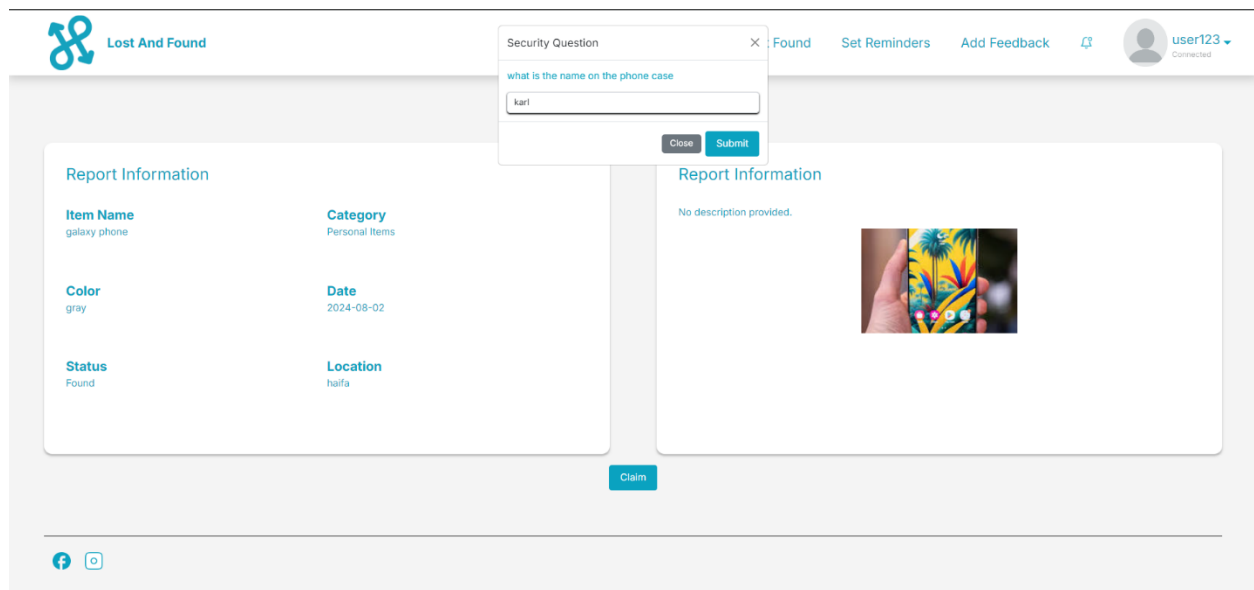
Unclaimed Items

0



nearby Items

[View All](#)



Conclusion

The POC development provided valuable insights and confidence in our ability to implement the "Lost and Found" system. It validated the core functionalities, including the new security question feature for item claims, and highlighted areas for improvement, such as security enhancements and performance optimizations. The successful completion of the POC mitigates the primary risks and paves the way for the full-scale development of the system.