Campus Connect

Computer Science Project seminar

Technical Documentation

Mentor:
Prof. Dr. Klen Čopič

Students:
Elena Nenadić
Nemanja

Cvetić

Table of contents

Definition of the problem

Lost and found management systems in Slovenia are often outdated. Many businesses, institutions, and organizations still rely on email communication for handling lost-and-found inquiries and arranging in-person item collection. On one hand, email communication is simple, straightforward, and initially cost-effective, as it does not require dedicated software development. On the other hand, the time required to process requests manually, especially when dealing with important documents, electronics, or other valuable items, highlights the need for more modern and efficient solutions.

Specifically in the domain of education, educational institutions such as middle schools, high schools, and universities face numerous daily situations involving both lost and found items. Implementing a dedicated item management system would significantly save time for both students and administrative staff, considering the high level of digital literacy among students. In cases where no system is used, time management and user satisfaction become major concerns. Even in institutions where a system has already been implemented, there are still two key areas in which improvements can be made to address ongoing issues.

The first area is modernization. Implementing a modern interface, clear instructions, and an engaging application design makes daily usage easier for all users, including those with lower digital literacy. A well-designed system can reduce confusion, simplify navigation, and encourage users to report or claim items more frequently.

The second issue concerns privacy and user safety. In cases where valuable items are lost, there is a need for a secure process for verifying and claiming ownership. Without proper verification mechanisms, there is a risk of items being claimed by unauthorized individuals. A modern system should therefore include features that protect user information while ensuring that only the rightful owner can retrieve a lost item.

We aim to create a system that will help address these issues within educational institutions in Slovenia. The initial development is for the UP FAMNIT, where the system will serve as a centralized platform for managing lost and found items.

Although the system is initially intended for UP FAMNIT, its design will allow it to be easily adapted and generalized for use in other educational institutions, businesses, and organizations. This flexibility will enable future expansion and adoption in environments that face similar challenges in managing lost and found items.

Feasibility study

The proposed Lost & Found System is considered feasible because it can be developed using available technologies such as React, Vite, Node.js, and MariaDB. The development process will be supported by tools including MySQL Workbench, Visual Studio Code, GitHub, and Jira for project management and team communication. The system requires minimal financial resources and can be completed within the planned timeframe. It is expected to improve the process of reporting and recovering lost items while providing an easy-to-use platform for students and administrators. Therefore, the project is considered technically, economically, operationally feasible.

System Analysis and Design

Functional requirements

The system should enable the following functionalities:

1. The system enables the user to register. Registration is done by allowing users to provide their username, email, password, and password confirmation. Before registration, the user must agree to the terms and conditions. If one of the fields is missing, if the input is incorrect, if the password is too weak, or if the password confirmation does not match, the system notifies the user and asks them to correct the input.
2. The system allows the user to log in. Login is done by entering the already registered email address and matching password. The system gives feedback on whether the login was successful. In the case of an unregistered email or wrong password, the user is notified.
3. The system enables users to view their profile.
4. The system enables users to report a lost item. When creating a lost item report, the user must provide the item name, category, description, date and approximate time of loss, and the location where the item was lost. The user can also upload a picture of the item if available.
5. The system enables users to report a found item. When creating a found item report, the user must provide the item name, category, description, date and time when the item was found, and the location where it was found. The user can also upload an image of the found item.
6. The system enables users to select the location of a lost or found item on an interactive map. The map helps users mark the approximate place where the item was lost or found, making the search process easier and more accurate.
7. The system enables users to browse all reported lost and found items. Items are displayed in a list with basic information such as item name, category, date, status, and location.
8. The system enables users to search and filter lost and found items. Filters can include item category, date, location, item status, and whether the item is lost or found. This allows users to find relevant reports more quickly.
9. The system enables users to open a detailed page for each item. The detailed page displays the item name, description, category, image, date, location, map position, current status, and contact or claim option.
10. The system enables users to claim an item that they believe belongs to them. To claim an item, the user must provide additional information proving ownership, such as a detailed description, unique characteristics, or other verification details.

11. The system enables a secure ownership verification process. For valuable items such as documents, electronics, wallets, or keys, the system should allow employees or administrators to review the claim before approving the return of the item.
12. The system enables users to see the status of their reports and claims. The status can be, for example, pending, approved, rejected, returned, or closed. This allows users to follow the progress of their request.
13. The system enables users to receive notifications about changes in item status, claim approval or rejection, and possible matches between lost and found items.
14. The system allows employees or administrators to add, update, delete, approve, or reject item reports. They can manage reported items, review claims, change item status, and remove inappropriate or incorrect reports.
15. The system allows employees or administrators to manage official locations within the institution, such as faculty buildings, classrooms, library, cafeteria, reception, or student office. These locations can be used when reporting lost or found items.
16. The system allows employees or administrators to review reports and user activity to ensure safety and prevent misuse of the system. If necessary, administrators can block users or delete inappropriate content.
17. The system can be used by both registered and unregistered users. However, unregistered users can only browse publicly visible lost and found items, while registered users can create reports, claim items, receive notifications, and interact with the system.

Non-functional requirements

1. The system must be able to process a high number of requests at the same time, especially during busy periods at the institution.
2. Each page should load within 1.5 seconds so that users do not notice a delay while browsing, reporting, or claiming items.
3. The system should be available 24/7, with an uptime of at least 97%.
4. The system should be developed as a web application, without a separate Android or iOS application, in order to reduce development and maintenance costs.
5. The system must ensure the protection of users' personal information. Sensitive data such as email addresses, claim details, and ownership verification information should only be visible to authorized employees or administrators.
6. The system must store data securely and protect it from unauthorized access, modification, or deletion.
7. The system should keep item reports and claim records for at least 18 months, in case the information is needed for future reports, disputes, or verification.
8. The system must support different user roles, including unregistered users, registered users, and administrators.
9. The system should initially support use at UP FAMNIT, but it should be designed in a flexible way that allows later expansion to other educational institutions or organizations.
10. The system should include regular data updates and maintenance to ensure that item statuses, locations, and user information remain accurate.
11. The system should be easy to use and understandable for users with different levels of digital literacy.
12. The system should integrate with a map service in order to display and select the approximate location of lost and found items.
13. The system should include notifications that are delivered reliably and within a short period after a status change, claim decision, or possible item match.
14. The system should include documentation for installation, maintenance, administration, and future extension of the system.
15. The system should be developed and deployed within two months after the system design is finalized.
16. The system should be maintainable, meaning that future updates, new locations, new institutions, and additional features can be added without major changes to the whole system.

Matrix of user roles

Table 1: Matrix of user roles/functions

Functions	User	Admin
Register	Yes	No
Log in	Yes	Yes
View profile	Yes	No
View admin dashboard	No	Yes
Report a lost item	Yes	Yes
Report a found item	Yes	Yes
Upload item images	Yes	Yes
Select item location on map	Yes	Yes
Browse lost items	Yes	Yes
Browse found items	Yes	Yes
Search and filter items	Yes	Yes
View item details	Yes	Yes
Submit item claim	Yes	Yes
View claim status	Yes	Yes
Receive notifications	Yes	Yes
Review ownership claims	No	Yes
Approve/Reject claims	No	Yes
Delete item reports	No	Yes
Manage locations	No	Yes
Manage categories	No	Yes

Review reported content	No	Yes
Promote users to admin	No	Yes

Data Dictionary

Table 2: Data Dictionary

Entity	Description	Attribute	Type	Description of attribute
User	**User of the system**	id	int	Unique identificator
		name	varchar(100)	Name of the user
		student_number	varchar(20)	Unique student number of the user
		email	varchar(150)	Email of the user
		password	varchar(255)	Password of the user
		created_at	timestamp	Date of creation
		role	enum('student','admin')	User role (admin or student)
Item	**Item present in the system**	id	int	Unique identifier of the item
		user_id	int	User who created the listing
		type	enum('lost','found')	Indicates whether the item is lost or found
		title	varchar(150)	Title of the listing
		description	text	Detailed description of the item
		secret_question	varchar(500)	Verification question used to confirm ownership

		secret_answer	varchar(500)	Correct answer to the verification question
		category	enum	Category of the item
		location	varchar(255)	Location where the item was lost or found
		latitude	decimal	Latitude coordinate
		longitude	decimal	Longitude coordinatie
		status	enum('open','resolved')	Current status of the listing
		created_at	timestamp	Date and time when the listing was created
Message	**Resents a message exchanged between users in a conversation**	id	int	Unique identifier of the message
		conversation_id	int	Conversation to which the message belongs
		sender_id	int	User who sent the message
		body	text	Content of the message
		is_read	tinyint(1)	Is the message read
		sent_at	timestamp	Date and time when the message was sent
		id	int	Unique identifier of the image
		item_id	int	

Image	**Image present in the system**			Item to which the image belongs
		image_url	varchar(500)	Path or URL of the uploaded image
		uploaded_at	timestamp	Date and time when the image was uploaded
Conversation	**Conversation between two users present in the system**	id	int	Unique identifier of the conversation
		item_id	int	Item associated with the conversation
		user_1_id	int	First participant in the conversation
		user_2_id	int	Second participant in the conversation
		created_at	timestamp	Date and time when the conversation was created
Claim	**Claim of items present in the system**	id	int	Unique identifier of the claim
		item_id	int	Item for which the claim was submitted
		claimant_id	int	User who submitted the claim
		claimant_answer	text	User's answer to the ownership

				verification question
		status	enum	Current status of the claim
		created_at	timestamp	Date and time when the claim was created

Entity Relationship Diagram (ERD)

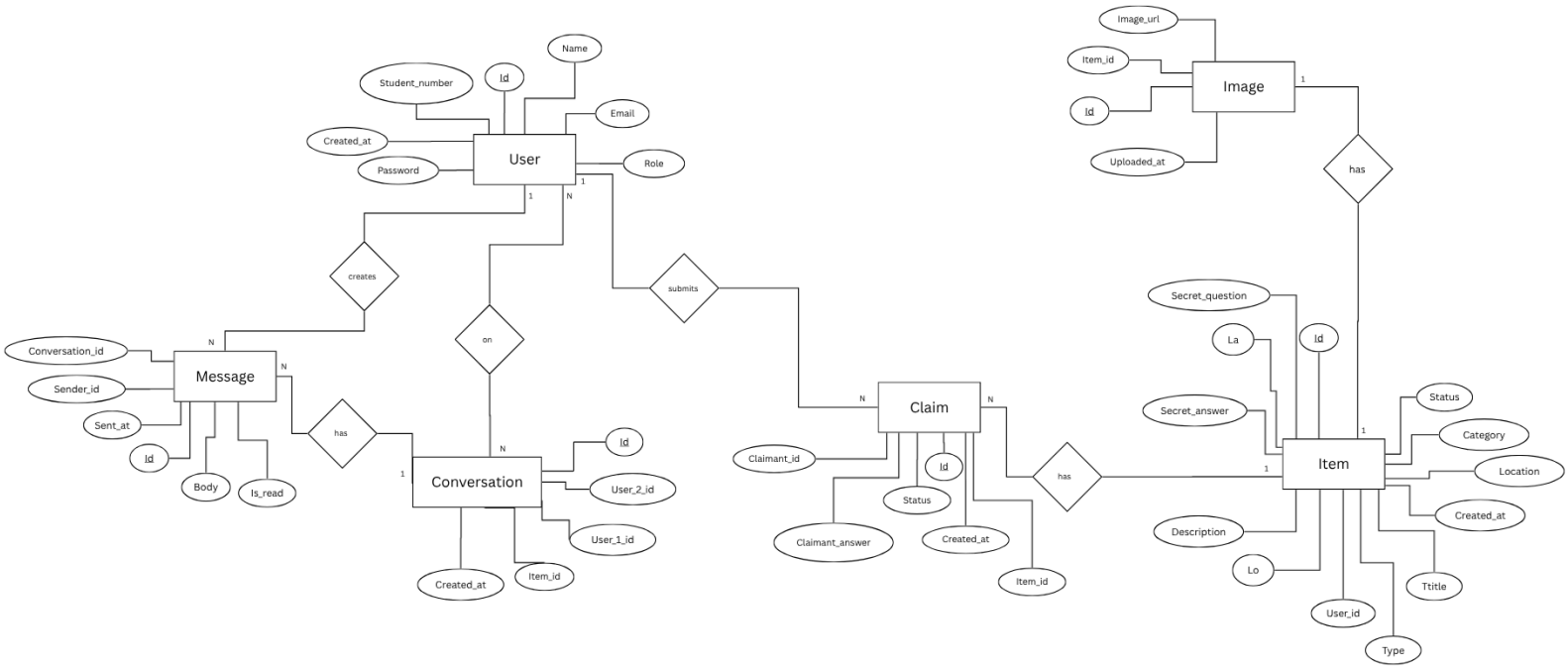

Figure 1: Entity Relationship Diagram

Relational Model

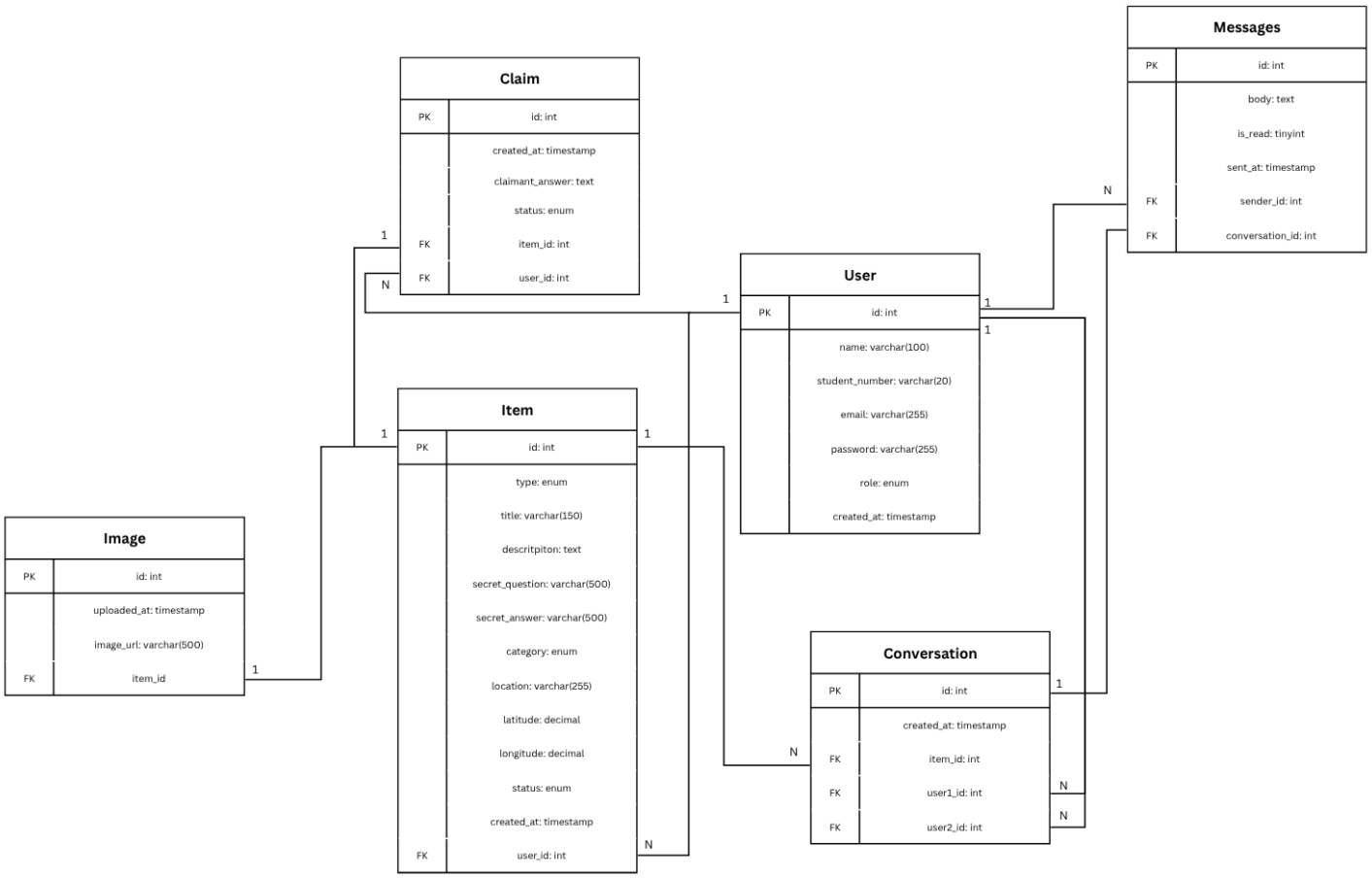

Figure 2: Relational Model

Physical Data Model

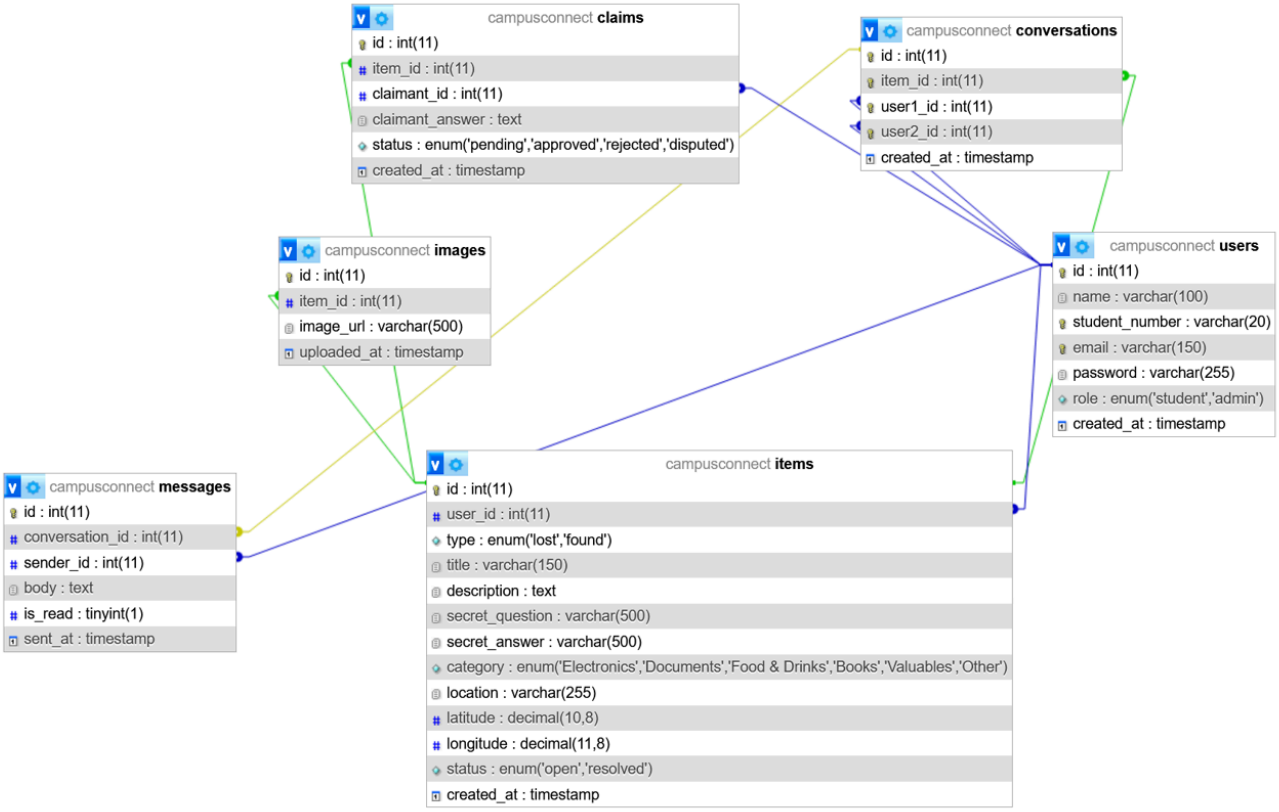

Figure 3: Physical data model

UML Sequence Diagram

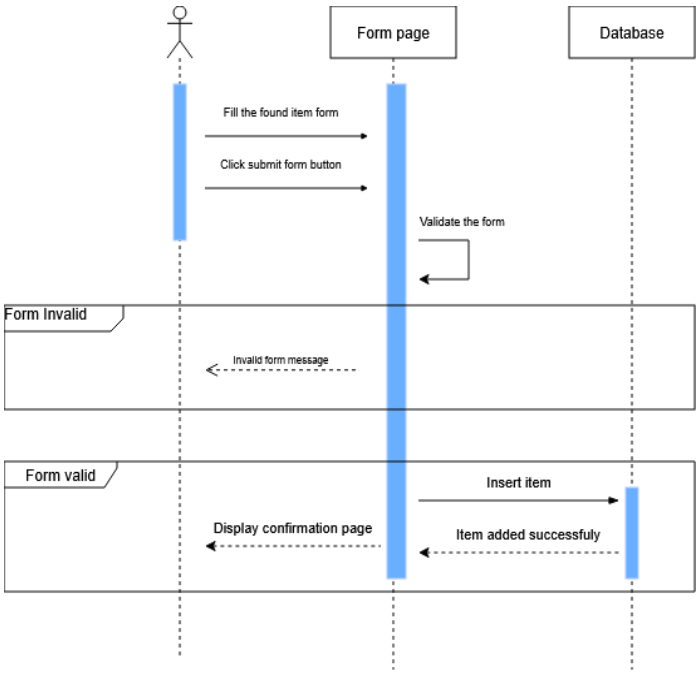

Figure 4: UML Sequence Diagram for reporting an item

Implementation

Technologies

The Lost & Found System was developed using React and Vite for the frontend, Node.js for the backend, and MariaDB as the database management system. These technologies were selected because they provide high performance, scalability, maintainability, and strong community support for modern web application development.

React is an open-source JavaScript library developed by Meta for building interactive user interfaces using reusable components[1]. Its component-based architecture simplifies application maintenance, promotes code reuse, and enables efficient updates through the Virtual DOM, resulting in improved rendering performance.

Vite was used as the frontend build tool and development server. Unlike traditional bundlers, Vite utilizes native ES modules during development, allowing significantly faster startup times

and instant hot module replacement (HMR)[2]. This greatly improves developer productivity and reduces waiting time during testing and implementation.

The backend was implemented using Node.js, a JavaScript runtime environment built on Google's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O architecture that enables efficient handling of multiple simultaneous requests while maintaining good performance[3]. Using JavaScript on both the frontend and backend also simplified development by allowing the same programming language to be used throughout the entire application.

MariaDB was selected as the relational database management system because it is open-source, reliable, and fully compatible with MySQL. It provides high performance, strong security features, and efficient management of relational data[4]. The database stores user accounts, item reports, messages, claims, and administrative information while ensuring data integrity through relational constraints.

Together, these technologies provide a modern full-stack architecture capable of supporting future extensions and deployment in a production environment.

Development Tools

Several development tools were used throughout the project. Visual Studio Code served as the primary code editor due to its lightweight design, integrated debugging capabilities, and extensive extension ecosystem that supports JavaScript, React, Node.js, and database development[6].

MySQL Workbench was used for database design, schema creation, and administration. Although the project uses MariaDB, MySQL Workbench provides full compatibility for designing and managing MariaDB databases through SQL queries and graphical modeling tools[9]. GitHub was used for version control and collaborative development. Git enabled team members to track changes, create separate development branches, merge completed features, and maintain a complete history of project modifications[7].

Jira supported project planning and team communication by organizing development tasks, monitoring progress, assigning responsibilities, and tracking project milestones[8]. Using Jira helped the team coordinate development activities and manage the project more efficiently. These tools improved collaboration, organization, code quality, and tracking of project progress throughout development.

Frontend Implementation

Frontend development focused on creating a user-friendly and responsive interface that allows users to easily interact with the system. The application was built using React.

Several pages were developed, including user authentication pages, lost and found item pages, item detail pages, messaging interfaces, map pages, claim management pages, profile pages, and administrative views. In addition, two informational pages were created using HTML and CSS to explain the purpose of the project and provide users with a built-in user manual and usage guidelines.

CSS styling was applied throughout the application to improve its visual appearance and ensure compatibility across different screen sizes and devices. Responsive layouts allow the application to function correctly on desktop computers, tablets, and mobile devices. The component-based architecture of React also simplified the implementation of reusable interface elements such as navigation bars, forms, buttons, and cards.

Interactive maps were implemented using the Leaflet JavaScript library, allowing users to visualize lost and found items geographically[5]. The map is centered around the Koper area, with particular emphasis on the UP FAMNIT campus and nearby locations where students frequently lose or find personal belongings. Users can interact with markers to obtain additional information about reported items.

The frontend development phase began in early April after the initial project planning and system design stages had been completed. Development continued throughout April and May, with additional improvements, bug fixes, interface refinements, and usability enhancements implemented during the final stages of the project.

Backend Implementation

The backend was developed using Node.js and is responsible for processing requests, managing application logic, and handling communication with the database. All major system functionalities were implemented on the backend, including user authentication, item management, image handling, messaging, claim processing, and administrative operations.

Authentication mechanisms were implemented to ensure that only authorized users (students and administrators) can access protected system features and perform actions according to their assigned permissions. Passwords are securely stored, and user sessions are managed to prevent unauthorized access. Input validation is performed before processing requests to reduce the risk of invalid data and common web security vulnerabilities.

The backend communicates with the MariaDB database through SQL queries, retrieving and updating application data whenever users interact with the system. Database operations were designed to maintain consistency and reliability while minimizing redundant data storage. Backend development began with the design of the database structure, including the creation of database schemas, tables, relationships, and constraints between entities. Development continued throughout April and May alongside frontend implementation.

The future plan for the project is to deploy the application on faculty-hosted servers, making it accessible to students and staff through a centralized platform for reporting and recovering lost items. Deployment would allow the system to operate continuously while supporting real-world usage within the university community.

Future Work

This project serves as a foundation for a modern and efficient system for managing lost and found items. Although it was primarily developed with UP FAMNIT in mind, the system can be expanded and adapted for use at other educational institutions throughout Slovenia.

Several future improvements are planned. These include support for multiple languages to make the platform accessible to a wider range of users, the development of more advanced and interactive map features, and the creation of dedicated mobile applications for Android and iOS devices.

Future versions may also include push notifications to inform users when matching items are reported, image recognition algorithms to assist in identifying similar objects automatically, and integration with university authentication systems for simplified login.

Additional enhancements include the integration of higher-quality images, icons, animations, and visual assets to provide a more modern and professional user experience. Performance optimization, accessibility improvements, and additional administrative tools are also planned to further improve usability and system maintainability.

References

[1] React, "React Documentation," Meta Platforms, Inc. [Online]. Available: https://react.dev/. Accessed: June. 2, 2026.

[2] Vite, "Vite Documentation." [Online]. Available: https://vite.dev/. Accessed: June. 2, 2026.

[3] Node.js Foundation, "Node.js Documentation." [Online]. Available: https://nodejs.org/. Accessed: June. 12, 2026.

[4] MariaDB Foundation, "MariaDB Documentation." [Online]. Available: https://mariadb.org/. Accessed: June. 21, 2026.

[5] Leaflet Contributors, "Leaflet Documentation." [Online]. Available: https://leafletjs.com/. Accessed: June. 21, 2026.

[6] Microsoft, "Visual Studio Code Documentation." [Online]. Available: https://code.visualstudio.com/docs. Accessed: June. 12, 2026.

[7] GitHub, "GitHub Documentation." [Online]. Available: https://docs.github.com/. Accessed: June. 13, 2026.

[8] Atlassian, "Jira Software Documentation." [Online]. Available: https://support.atlassian.com/jira-software-cloud/. Accessed: June. 15, 2026.

[9] Oracle Corporation, "MySQL Workbench Manual." [Online]. Available: https://dev.mysql.com/doc/workbench/en/. Accessed: June. 18, 2026.

Appendices

Github repository link: https://github.com/NemanjaCvetic/CampusConnect

Jira board link:
https://lost-and-found-system.atlassian.net/?continue=https%3A%2F%2Flost-and-found-system.atlassian.net%2Fwelcome%2Fsoftware%3FprojectId%3D10033&atlOrigin=eyJpIjoiMTNkNzA5YjQzZGY2NDEyYzliMzlhOWFkMzM4ODBjOTgiLCJwIjoiamlyYS1zb2Z0d2FyZSJ9