

Names : Kalisa Mutabazi - ID: 25863

**Auca Lost & Found Item System**

**Problem Description**

In Auca Students and staff often misplace personal belongings such as books, bags, phones, and ID cards on campus. Currently, there is no structured or centralized way to report and track lost or found items. Information about lost or found items is usually spread by word of mouth, posters, or informal messages, which is inefficient and unreliable. As a result:

* Owners struggle to recover their lost belongings quickly.
* Found items may remain unclaimed for long periods or get permanently lost.
* There is no proper accountability or record of lost-and-found transactions.

The **Lost & Found Item System** aims to provide a centralized, digital platform where students and staff can easily report, search, claim, and track lost or found items, ensuring a faster and more reliable recovery process.

**Key Business Requirements**

The system must:

1. Allow users (students and staff) to **report lost items** with details such as item name, description, date, and location.
2. Allow users to **report found items** with similar details and an option to upload a photo.
3. Provide a **search function** so users can browse or filter items based on categories, date, or location.
4. Enable a **claiming process**, where the rightful owner can request to claim a found item.
5. Include an **admin role** to manage, verify, and approve lost or found item reports.
6. Notify users (via email or in-app alerts) when a matching lost or found item is reported.
7. Keep a **record/history** of items reported, claimed, and resolved for accountability.
8. Be accessible to both students and staff via a simple and user-friendly web interface.

**Quality Attributes (Non-Functional Requirements)**

The system should exhibit the following qualities:

1. **Usability** – Simple, intuitive, and accessible to non-technical users.
2. **Reliability** – Ensure accurate recording and retrieval of item details with minimal downtime.
3. **Security** – Protect user accounts and prevent unauthorized access to sensitive information.
4. **Scalability** – Support an increasing number of users and item records as the campus grows.
5. **Maintainability** – Easy to update, fix bugs, and add new features in the future.
6. **Performance** – Fast search and retrieval of items without long delays.
7. **Data Integrity** – Prevent duplicate or false reports and ensure correct matching of lost and found items.

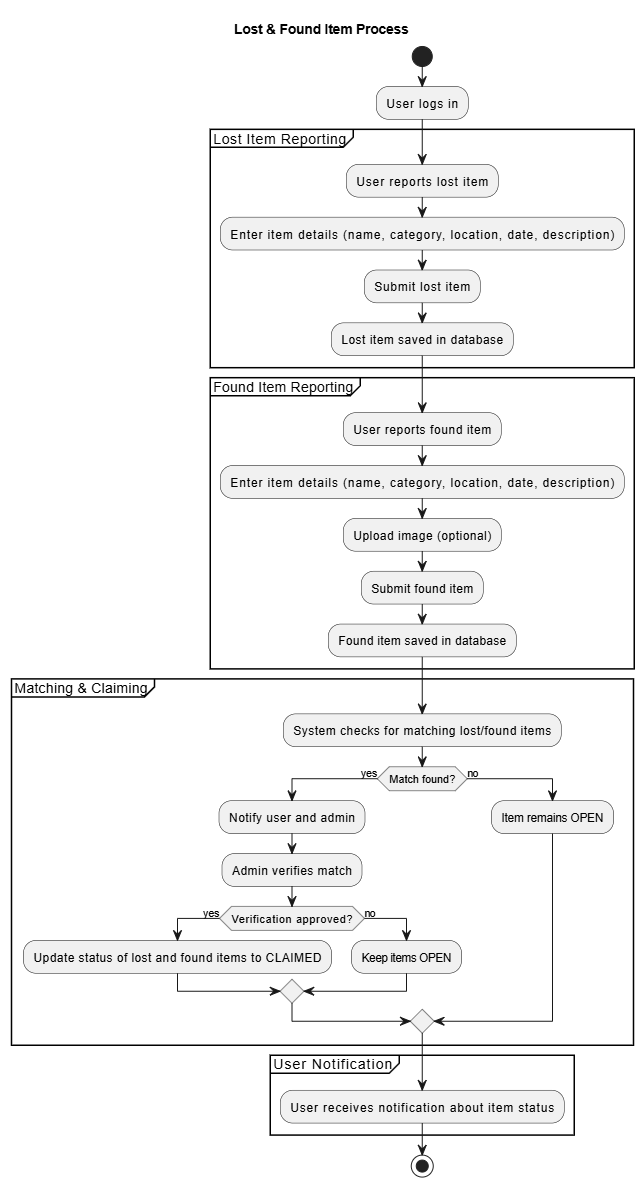
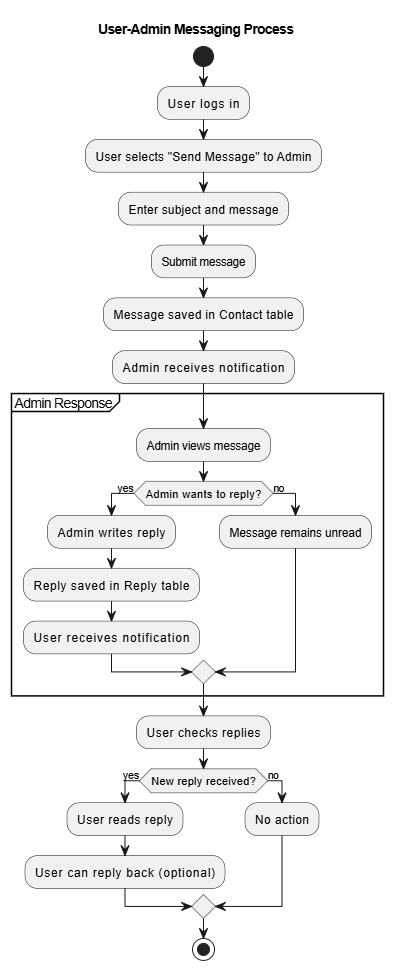
**Domain & Data Modeling**

1. **ER / UML Diagram**

**Relationships Summary:**

* Users → Match: 1-to-many (a user can verify multiple matches)
* LostItem → Match: 1-to-many (a lost item can have multiple potential matches)
* FoundItem → Match: 1-to-many (a found item can have multiple potential matches)
* Category → LostItem / FoundItem: 1-to-many
* Contact → Reply: 1-to-many (a message can have multiple replies)

**UML Activity Diagram:**

**  
**

**b. Principal Domain Concepts**

**Actors:**

* **Students/Staff** – report lost/found items, claim items, send messages to admin**.**
* **Admin –** manage items, approve matches, respond to messages.

**Processes:**

* **Report Lost/Found Item –** users submit item details.
* **Match/Claim –** system or admin matches lost and found items**.**
* **Messaging –** users communicate with admins via Contact and Reply.
* **Categorization –** items are grouped by category for better search.

**Data Objects:**

* **Users –** represent all users and admins
* **Category** – groups items
* **LostItem / FoundItem –** reported items
* **Match** – tracks pairing between lost and found items
* **Contact / Reply** – handles user-admin communication

**c. Technology-Oriented Schema (Relational)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Primary Key** | **Foreign Keys** | **Constraints / Indexes** |
| Users | userId | – | username, email UNIQUE |
| Category | categoryId | – | categoryName UNIQUE |
| LostItem | lostId | categoryId → Category.categoryId | status ENUM, index on categoryId |
| FoundItem | foundId | categoryId → Category.categoryId | status ENUM, index on categoryId |
| Match | matchId | lostId → LostItem.lostId foundId → FoundItem.foundId matchedBy → Users.userId | status ENUM, index on lostId, foundId |
| Contact | id | sender\_id → Users.userId receiver\_id → Users.userId | readStatus BOOLEAN, index on sender/receiver |
| Reply | id | contact\_id → Contact.id | index on contact\_id |

**Notes:**

* status fields in LostItem, FoundItem, and Match use the **ItemStatus** enum.
* Indexes are added for quick search and lookups on categoryId, lostId, foundId, sender\_id, and receiver\_id.
* Match serves as a junction table between LostItem and FoundItem, including verification tracking.

**3. System Implementation**

**b. Front-End Design and Prototype**

**Framework/Library:** React.js with TypeScript

**Design Principles:**

* Responsive layout: Adapts to desktops, tablets, and mobile screens using CSS Flexbox/Grid and media queries.
* **Components:**
  + Login / Registration
  + Dashboard with navigation for lost items, found items, matches, and messages
  + Item Reporting Forms for lost and found items
  + Match & Claim Management
  + Messaging Module (Contact & Reply)
* **Mockups / Wireframes:**
  + Desktop: Multi-column layout for dashboard and item management.
  + Tablet/Mobile: Collapsible sidebar, stacked card views for items and messages.

**Back-End Architecture**

**Framework:** Spring Boot (Java)

**Architecture Style:** Layered (Monolithic)

**Layers:**

1. **Controller Layer:** Handles HTTP requests from React front-end.
2. **Service Layer:** Business logic for items, matches, users, and messaging.
3. **Repository Layer:** Interfaces with PostgreSQL for persistence using Spring Data JPA.
4. **Security Layer:** OAuth2 authentication, JWT token handling, role-based authorization.

**Endpoints:**

* POST /lost-items – Submit lost item
* POST /found-items – Submit found item
* GET /matches – Fetch item matches
* POST /contacts – Send message
* GET /contacts – Retrieve messages

**Database Design**

**Relational Database:** PostgreSQL

**Tables:**

* Users
* Category
* LostItem
* FoundItem
* Match
* Contact
* Reply

**Indexes & Constraints:**

* Primary keys on id fields
* Foreign keys for relationships (e.g., lostId, foundId, categoryId)
* Unique constraints on Users.email and Users.username

**e. Security Implementation**

**Authentication:**

* OAuth2 login (Google, Github and institutional login)
* JWT-based session management

**Authorization:**

* Role-based access control (RBAC)
* Admin: full access to matches, messages, and item verification
* Users: limited access to own items and messages

**Optimization:**

* Use pagination for lists of items
* Cache frequent queries (e.g., categories, approved matches)

**f. Version Control**

* **Git** used for version control
* **Feature branches** for new modules
* **Pull requests** submitted to main/master branch upon feature completion
* Commit messages follow clear semantic style: feat: add lost item reporting, fix: resolve match status bug