23CSE311

SOFTWARE ENGINEERING

SPRINT 1

TEAM – 7

Introduction

The L4 - Lost & Found Tracking System is a web-based platform built for college campuses to simplify how lost and found items are reported, tracked, and returned. Instead of relying on physical notice boards or manual registers, users can submit reports online with photos and location details, and the system does the heavy lifting of finding potential matches.

At the heart of the system is an AI matching engine powered by YOLOv8 object detection models. When a user uploads an image of a lost or found item, the system runs it through multiple YOLO models to identify the object type (phone, bag, laptop, stationery, etc.), and then compares it against existing reports using image similarity and text analysis to suggest probable matches.

To prevent false or fraudulent claims, the system includes a fraud detection module. It scores each claim based on the quality of ownership proof provided, the claimant's past history, and how closely their description matches the item — flagging suspicious submissions for admin review.

The platform supports four user roles — Student, Faculty, Visitor, and Admin — each with different levels of access controlled through Role-Based Access Control (RBAC). Students and faculty register with institutional IDs, while visitors use OTP-based temporary accounts. Admins manage users, configure AI settings, and approve or reject claims.

Other key features include:

- JWT-based authentication with access and refresh tokens for secure sessions
- OTP verification via Fast2SMS.
- Campus zone mapping using GeoJSON polygons to tag item locations on a map
- Multi-channel notifications — email, SMS, and in-app alerts for status updates
- PDF generation for handover letters and claim confirmations using PDFKit
- Audit logging that records every significant action for institutional accountability
- Background schedulers (via Bull queues and node-cron) for automated data retention, claim escalation, and reminder emails

The system follows ethical design principles: AI matching thresholds are fully configurable by admins, all AI decisions can be overridden manually, data retention policies support automatic deletion or anonymization of old records, and audit trails ensure full transparency across all operations.

STAKEHOLDERS

Primary Users

Students

The primary users who register using institutional IDs, report lost or found items with images and location data, submit ownership claims with proofs, receive AI-powered match notifications, and track claim status through a personal dashboard.

Teaching Faculty / Professors

Faculty members who register and use the system similarly to students, with the ability to report and claim items, and access notifications tailored to their role.

Campus Visitors

Temporary users who register via OTP-based verification (email/SMS), report or claim items during their visit, with accounts that auto-expire after a defined period.

Operational Users

Administrators

System controllers who manage user roles and permissions, configure AI matching thresholds and weights, review and approve or reject claims, manage campus zones, handle claim conflicts and escalations, and generate handover letters and claim confirmations.

Delegated Administrators

Users with scoped admin privileges delegated by primary administrators, with configurable permission scopes and expiry dates.

Management and Institutional Stakeholders

Campus Security / Lost & Found Office

Operational decision-makers who review AI match suggestions, process item handovers, monitor claim disputes, and use the admin dashboard for real-time item tracking and analytics.

Educational Institution

The high-level stakeholder concerned with audit logs for accountability, monitoring fairness in item recovery processes, ensuring compliance with data retention and privacy policies, and tracking long-term system adoption and effectiveness metrics.

INDIVIDUAL CONTRIBUTION

Praveen SB [CB.SC.U4CSE23136]

Led complete frontend redesign and user interface overhaul across the entire application. Implemented dark/light theme system with CSS custom properties and ThemeProvider integration. Redesigned landing page with scroll-driven animations, login and registration pages, visitor registration, user dashboard, item inventory and detail pages, report item and claim submission pages. Added GSAP effects library, animation hooks, and scroll utilities for enhanced user experience. Updated Tailwind configuration for theme support and managed frontend dependency updates throughout the project lifecycle.

Ankith [CB.SC.U4CSE23133]

Developed and deployed the complete frontend testing infrastructure and created comprehensive documentation for the system. Implemented Vitest unit testing framework covering all major frontend components including Login, Register, VisitorRegister, and ItemDetails. Developed frontend-backend integration features including admin dashboard components, claims management, live activity feeds with image support, and Cloudinary image upload functionality. Created extensive UML documentation including activity diagrams, class diagrams, sequence diagrams, use case diagrams, role-based access control diagrams, and use case stories. Implemented fixes for admin dashboard, claims data handling, notifications, and profile statistics alignment.

Adithya Monish Kumar K [CB.SC.U4CSE23103]

Developed and maintained the complete backend architecture including database operations, business logic implementation, and API functionality. Built comprehensive backend testing suite with 226 tests covering middleware, routes, services, models, and schemas achieving near-complete code coverage. Added TSDoc documentation with @module headers and inline documentation across the entire backend codebase for automated API documentation generation, including entry points, config, middleware, models, schemas, utils, services, routes, and scripts. Added JSDoc @module headers to all frontend source files for comprehensive code documentation. Implemented Fast2SMS integration for India-based SMS support, multi-origin CORS support for development environments, and production deployment scripts for Render hosting platform. Fixed notification preferences schema issues and test failures, re-enabled requireAdmin middleware functionality. Managed backend dependency updates, coordinated multiple pull request merges, and successfully merged updated frontend with backend branches.

R.D.Tarun [CB.SC.U4CSE23138]

Developed end-to-end testing infrastructure using Playwright and comprehensive frontend unit tests for core components including Login, Register, ReportItem, ItemInventory, Sidebar, and LogoutConfirmModal. Redesigned multiple admin pages including dashboard

with GlassCard layout, claims management, role and zone management, and AI configuration with theme support. Updated user-facing pages including profile with animated stats, notifications with type-based icons, and my claims with status filters. Implemented interactive navigation components including hoverable sidebar that expands on hover, admin sidebar with consistent navigation across all admin pages, and logout confirmation modal with Yes/No options. Added back to dashboard button on Report Item page for improved navigation flow. Updated sidebars and campus map component for theme compatibility with theme toggle and theme-aware styling. Revised project documentation including README for clarity and project overview, and DevDocs with comprehensive developer guidance including setup instructions, architecture overview, and technology stack. Updated backend dependencies and managed repository merges. Added E2E testing documentation and finalized test suite.

Vinaayak Kanagaraj [CB.SC.U4CSE23152]

Developed and trained multiple YOLO machine learning models for object detection including college bags, keyboards, mice, monitors, electronic screens, suitcases, mobile phones, and handbags. Implemented three-tier image classification system using YOLO, OpenClip, and FAISS for missing-found item matching. Created wrapper service for automated image pairing between reported missing items and found items. Developed suspicious request detection system to identify potentially fraudulent item claims from users based on behavioral patterns and request analysis. Assisted in E2E and Integration testing as well.

MEETINGS DOCUMENTATION

1. Project Initiation and Requirements Discussion
   Date: 30 December 2025

Agenda & Discussions:
Discussed the issue of lost and found items on the campus and the absence of a centralized tracking system.
Finalized the main objective of developing a digital lost and found management system for

campus use.
Identified primary user roles such as students, faculty, visitors, and administrators.
Outlined high-level system expectations and project scope.
Agreed to follow a modular and incremental development approach.

2. Epic Definition and User Story Planning
   Date: 30 December 2025

Agenda & Discussions:
Identified major system epics including authentication, item reporting, claims handling, notifications, and admin management.
Created user stories for different roles and system interactions.
Prioritized features based on importance and feasibility.
Discussed system constraints and assumptions.

3. Authentication and Login Strategy Discussion
   Date: 31 December 2025

Agenda & Discussions:
Finalized the authentication mechanism using access tokens and refresh tokens.
Discussed registration flow for students and faculty using institutional details.
Planned OTP-based temporary access for visitors.
Reviewed password security, failed login handling, and session tracking.
Assigned frontend and backend responsibilities for authentication.

4. Technology Stack Finalization
   Date: 31 December 2025

Agenda & Discussions:
Finalized frontend technologies for UI development and routing.
Finalized backend technologies using Node.js, Express, and TypeScript.
Selected MongoDB as the primary database.
Discussed use of third-party services for image storage, email, and SMS.
Decided on repository structure and version control workflow.

5. Initial Project Setup and Structure Review
   Date: 01 January 2026

Agenda & Discussions:
Reviewed initial project scaffolding for frontend and backend.
Discussed folder structures for routes, services, models, and components.
Verified environment variable setup.
Planned next steps for database schema creation and API development.

6. Database Design and Schema Review
   Date: 05 January 2026

Agenda & Discussions:
Designed and reviewed database schemas for users, items, claims, and campus zones.
Discussed indexing strategies and relationships between collections.
Planned expiry policies for temporary visitor accounts and sessions.

7. API Design and Validation Standards
   Date: 07 January 2026

Agenda & Discussions:
Standardized API response formats across services.
Finalized HTTP status code usage.
Reviewed request validation and centralized error handling strategy.

8. Frontend and Backend Integration Review
   Date: 11 January 2026

Agenda & Discussions:
Integrated frontend services with backend APIs.
Tested basic workflows such as registration, login, and dashboard access.
Reviewed token handling and protected routes.

9. Visitor Login and OTP Flow Discussion
   Date: 13 January 2026

Agenda & Discussions:
Finalized visitor registration using OTP verification.
Discussed OTP generation, expiry duration, and rate limiting.
Reviewed frontend flow and handled edge cases such as expired OTPs.

10. Claims Workflow and Admin Review Process
    Date: 17 January 2026

Agenda & Discussions:
Reviewed the complete claim submission and approval process.
Discussed handling of duplicate and conflicting claims.
Finalized claim status transitions and admin decision logic.

11. Admin Panel and Role-Based Access Review
    Date: 21 January 2026

Agenda & Discussions:
Reviewed admin dashboard features and controls.

Finalized role-based access permissions for all user roles.
Discussed audit logging and role change tracking.

12. User Interface and Usability Review
    Date: 25 January 2026

Agenda & Discussions:
Reviewed major user interfaces and forms.
Checked validation messages and error handling.
Discussed usability improvements and responsiveness.

13. Testing Strategy and Initial Test Setup
    Date: 02 February 2026

Agenda & Discussions:
Finalized tools for unit, integration, and end-to-end testing.
Set up test configurations for frontend and backend.
Planned test coverage for key user workflows.

14. Zone-Based Location System Review
    Date: 02 February 2026

Agenda & Discussions:
Reviewed campus zone design using geospatial data.
Discussed zone validation logic and admin zone management.
Demonstrated prototype flow for item reporting with zone selection.

15. Hosting and Deployment Discussion
    Date: 03 February 2026

Agenda & Discussions:
Discussed backend and frontend hosting options.
Reviewed production environment configuration and build process.
Planned deployment and service shutdown handling.

16. End-to-End Testing and Bug Review
    Date: 05 February 2026

Agenda & Discussions:
Reviewed results from end-to-end testing.
Resolved identified bugs and stability issues.
Verified notification delivery and background job execution.

17. AI Matching and Model Review
    Date: 08 February 2026

Agenda & Discussions:
Reviewed AI matching logic and similarity scoring.
Discussed confidence thresholds and match categories.
Planned backend integration for AI-generated suggestions.

18. Final Integration and Documentation Review
    Date: 09 February 2026

Agenda & Discussions:
Reviewed the complete system flow end-to-end.
Verified audit logging and document generation features.
Finalized project documentation.
Discussed future scope and possible enhancements.

PLANNED WORK FOR THE NEXT SPRINT

1. **Flutter Implementation**

Focused on a mobile version of the application using Flutter.

2. **Advanced AI Integration and Model Optimization**

The next sprint will focus on enhancing the AI matching pipeline with fine-tuned YOLO models specifically trained on common campus items. Existing YOLO fine-tuning notebooks for bags, phones, computer apparatus, screens, and suitcases will be integrated into an automated model training pipeline, enabling periodic retraining on newly collected data.

A feedback loop will be introduced where admin decisions on match suggestions (accepted, rejected, or overridden) are fed back into the system to continuously refine matching accuracy. A natural language description enhancement feature using large language models will be implemented to normalize and enrich user-submitted descriptions, improving TF-IDF text embeddings. Confidence calibration will be enhanced so that auto-approve thresholds dynamically adapt based on historical accuracy metrics.

3. **Cloud Hosting and Infrastructure Migration**

The application will be migrated from local development to a cloud-hosted production environment. Backend services will be deployed on a managed cloud platform such as AWS, GCP, or Azure. MongoDB Atlas will replace the local database, and a managed Redis service will be used for caching and job queue management.

Cloudinary, already integrated for image storage, will be configured with production-grade CDN delivery. The frontend will be deployed on a static hosting service such as Vercel or Netlify with proper environment variable management and domain configuration. Load balancing and auto-scaling policies will be implemented to handle peak campus traffic periods.

4. **DevOps Pipeline and Monitoring Infrastructure**

A comprehensive DevOps pipeline will be established using GitHub Actions. This includes Docker containerization for backend services, automated execution of unit tests with Vitest, integration tests with Supertest, and end-to-end tests with Playwright as mandatory checks before deployment.

A staging environment will be provisioned for pre-production validation. Monitoring infrastructure will be set up using Prometheus for metrics collection and Grafana for visualization. Centralized structured logging will be implemented for easier debugging and incident response. Nginx or Traefik will be configured as a reverse proxy with TLS termination, rate limiting, and request routing.

5. **Project Overview and Scope**

The project involves building a comprehensive campus Lost & Found management system that goes beyond simple item listing. Core objectives include AI-powered item matching, fraud-resistant claim processing, multi-role access control, and institutional audit compliance. The system scope covers the complete lifecycle of lost and found items, from reporting with image uploads and geo-tagged locations, through AI-driven matching and claim evaluation, to admin-approved handover with PDF documentation.

Problems addressed include inefficiency of physical lost and found offices, lack of intelligent matching between reports, susceptibility to fraudulent claims, and absence of accountability. Expected outcomes include reduced item recovery time, improved claim accuracy through AI, transparent operations, and a seamless user experience.

FINALIZED SYSTEM MODULES

Identity and Authentication

User registration, JWT access and refresh token management, OTP verification, login session tracking, password security with Argon2 hashing, and failed attempt lockout.

Role-Based Access Control
Role definitions for students, faculty, visitors, admins, and delegated admins, permission management, role change auditing, and delegated admin scopes with expiry.

Item Management
Lost and found item submission with multi-image uploads, item attributes, GeoJSON-based location mapping, item lifecycle management, and draft saving.

AI Matching Engine
YOLO object detection, OpenCLIP image embeddings, TF-IDF text embeddings, configurable similarity weights, and auto-match threshold management.

Fraud Detection
Claim confidence scoring, proof quality assessment, claimant history analysis, suspicious claim detection, and competing claim evaluation.

Claims Processing
Claim submission, ownership proof handling, AI confidence scoring, conflict detection, multi-level approval workflow, and admin override capability.

Notifications
Multi-channel notifications including email, SMS, in-app alerts, user preferences, quiet hours, and role or zone targeting.

Campus Zone Management
GeoJSON zone creation, geospatial indexing, zone activation and filtering.

Admin Dashboard and Analytics
AI performance metrics, claim analytics, user and item management, and AI configuration controls.

Background Jobs and Scheduling
Data retention, reminder scheduling, overdue claim escalation, and message queue processing using Bull and Redis.

Document Generation
PDF handover letters and claim confirmation documents.

Audit and Compliance
Comprehensive audit logging, data retention policies, anonymization rules, and role change audit trails.

TECH STACK

Frontend: React 18, Vite 5, JavaScript (JSX), Tailwind CSS, React Router DOM, Axios, GSAP, Framer Motion, Lucide React.

Backend: Node.js with Express, TypeScript, Vite-Node, Mongoose, JWT, Argon2, Zod, Bull, Redis, node-cron, PDFKit, Multer, SendGrid, Fast2SMS, Cloudinary, CORS.

AI and ML: YOLOv8, Roboflow API, OpenCLIP, TF-IDF, Non-Maximum Suppression, Cosine Similarity, YOLO fine-tuning notebooks.

Database and State: MongoDB, Redis, Bull queues, geospatial and TTL indexing.

TESTING AND QA

Unit testing using Vitest for frontend and backend. Component testing using React Testing Library. Integration testing using Supertest with MongoDB Memory Server. End-to-end testing using Playwright across complete user workflows. Code coverage using V8 provider. Linting with ESLint and Prettier.

DevOps and Deployment

Docker for containerization, GitHub Actions for CI/CD, Prometheus and Grafana for monitoring, Nginx or Traefik as reverse proxy, Cloudinary CDN, MongoDB Atlas, and managed Redis services.

Task Allocation

Frontend development covering UI components, animations, and responsiveness. Backend development including APIs, middleware, and validation. AI and ML pipeline development. Database schema and indexing. Authentication and security. DevOps, testing, and deployment.

TIMELINE AND MILESTONES

Phase 1: Core infrastructure
Phase 2: Feature development + Testing
Phase 3: AI model development
Phase 4: AI Integration & Testing
Phase 5: Deployment