CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-[Section]
System Integration and Architecture

System Design Document (SDD)

Project Title: Lost & Found Hub

Prepared By: Escalante, John Carlo C.

Version: 0.1

Date: 2/21/2026
Status: Draft

REVISION HISTORY TABLE

Version	Date	Author	Changes Made	Status
0.1	2/21/2026	John Carlo C. Escalante	Initial draft	Draft
0.2	[Date]	[Your Name]	Added API specifications	Review
0.3	[Date]	[Your Name]	Updated database design	Review
0.4	[Date]	[Your Name]	Added UI/UX designs	Review
0.5	[Date]	[Your Name]	Incorporated feedback	Revised
0.6	[Date]	[Your Name]	Final review and corrections	Final
1	[Date]	[Your Name]	Baseline version for development	Approved

TABLE OF CONTENTS

Contents

EXECUTIVE SUMMARY

1.1 Project Overview

Lost & Found Hub is a cross-platform campus application that enables students and staff to report lost items, post found items, and submit claim requests. The system consists of a backend API, web application, and mobile application that work together to provide a centralized lost-and-found management solution.

1.2 Objectives

1. Develop a functional lost and found management system
2. Implement user authentication (register, login, logout)
3. Allow users to post lost and found item reports
4. Enable claim request submission and approval process
5. Provide admin monitoring and verification
6. Ensure consistent experience across web and mobile platforms

1.3 Scope

Included Features:

- User registration and authentication
- Lost item reporting
- Found item reporting
- Claim request submission
- Admin approval/rejection of claims
- User dashboard
- Cross-platform support (web and mobile)
- Centralized relational database

Excluded Features:

- Payment processing
- Delivery services
- SMS/email notifications
- Social media login
- AI image recognition

1.0 INTRODUCTION

1.1 Purpose

This System Design Document (SDD) provides a comprehensive design specification for the Lost & Found Hub system. It describes the overall architecture, functional and non-functional requirements, API structure, database design, and implementation plan.

The purpose of this document is to guide the development team in building a consistent, scalable, and secure cross-platform system. It also serves as a reference for instructors, stakeholders, and reviewers to understand how the system components integrate and operate together.

2.0 FUNCTIONAL REQUIREMENTS SPECIFICATION

2.1 Project Overview

Project Name: Lost & Found Hub
Domain: Campus Utility System
Primary Users:

1. Students
2. Campus Staff
3. Administrators

Problem Statement: Students and staff often struggle to recover lost items due to lack of centralized reporting.
Solution: A digital platform that centralizes lost and found reports, allowing efficient tracking and claiming of items.

2.2 Core User Journeys

Journey 1: Reporting a Lost Item

1. User registers or logs in
2. User selects “Report Lost Item”
3. User enters item details (name, description, date, location)
4. User submits report
5. Item appears in dashboard as ACTIVE

Journey 2: Claiming a Found Item

1. User logs in

2. User browses found items
3. User selects item
4. User submits claim request
5. Admin reviews and approves/rejects claim
6. System updates item status

Journey 3: Admin Review Process

1. Admin logs in
2. Admin views pending claims
3. Admin verifies details
4. Admin approves or rejects claim
5. System updates item status

2.3 Feature List (MoSCoW)

MUST HAVE

1. User registration/login/logout
2. Lost item submission
3. Found item submission
4. Claim request submission
5. Admin claim approval/rejection system
6. Dashboard

SHOULD HAVE

1. Image upload for found items
2. Search and filtering by date or keyword
3. Claim history tracking

COULD HAVE

1. Push notifications
2. Real-time status tracking (Pending, Approved, Rejected)

WON'T HAVE

1. Online payments
2. GPS-based tracking
3. Third-party integrations

2.4 Detailed Feature Specifications

Feature: User Authentication

- **Screens:** Registration, Login, Forgot Password
- **Fields:** Email, Password, Confirm Password
- **Validation:** Email format, password strength, uniqueness
- **API Endpoints:** POST /auth/register, POST /auth/login, POST /auth/logout
- **Security:** JWT tokens, password hashing with bcrypt

Feature: Item Reporting

- **Screens:** Report Lost, Report Found
- **Display:** Item name, Description, Date lost/found, Location, Optional image
- **API Endpoints:** POST/items/lost, POST/items/found, GET/items, GET/items/{id}

Feature: Claim Management

- **Screens:** Item Detail, Claim Form
- **API Endpoints:** POST/claims, GET/claims, PUT/claims/{id}/approve, PUT/claims/{id}/reject
- **Access Control:** Only admins can approve/reject claims

2.5 Acceptance Criteria

AC-1: Successful User Registration

- ➢ Given I am a new user
- ➢ When I enter valid personal details and a strong password
- ➢ And click “Create Account”
- ➢ Then my account should be created
- ➢ And I should be redirected to the dashboard

AC-2: Successful Lost Item Submission

- ➢ Given I am logged in
- ➢ When I submit a lost item report with valid details
- ➢ Then the item should be saved in the system
- ➢ And it should appear in the dashboard as ACTIVE

AC-3: Successful Claim Approval

- ➢ Given I am logged in as an administrator
- ➢ When I review a pending claim request
- ➢ And approve the claim
- ➢ Then the claim status should change to APPROVED
- ➢ And the item status should change to CLAIMED

3.0 NON-FUNCTIONAL REQUIREMENTS

3.1 Performance Requirements

- API response time ≤ 2 seconds
- System supports at least 100 concurrent users

3.2 Security Requirements

- HTTPS communication
- Password hashing
- JWT-based authentication
- Role-based access control

3.3 Compatibility Requirements

- **Web Browsers:** Chrome, Firefox, Safari, Edge (latest 2 versions)
- **Android:** API Level 24+ (Android 7.0+)
- **Screen Sizes:** Mobile (360px+), Tablet (768px+), Desktop (1024px+)
- **Operating Systems:** Windows 10+, macOS 10.15+, Linux Ubuntu 20.04+

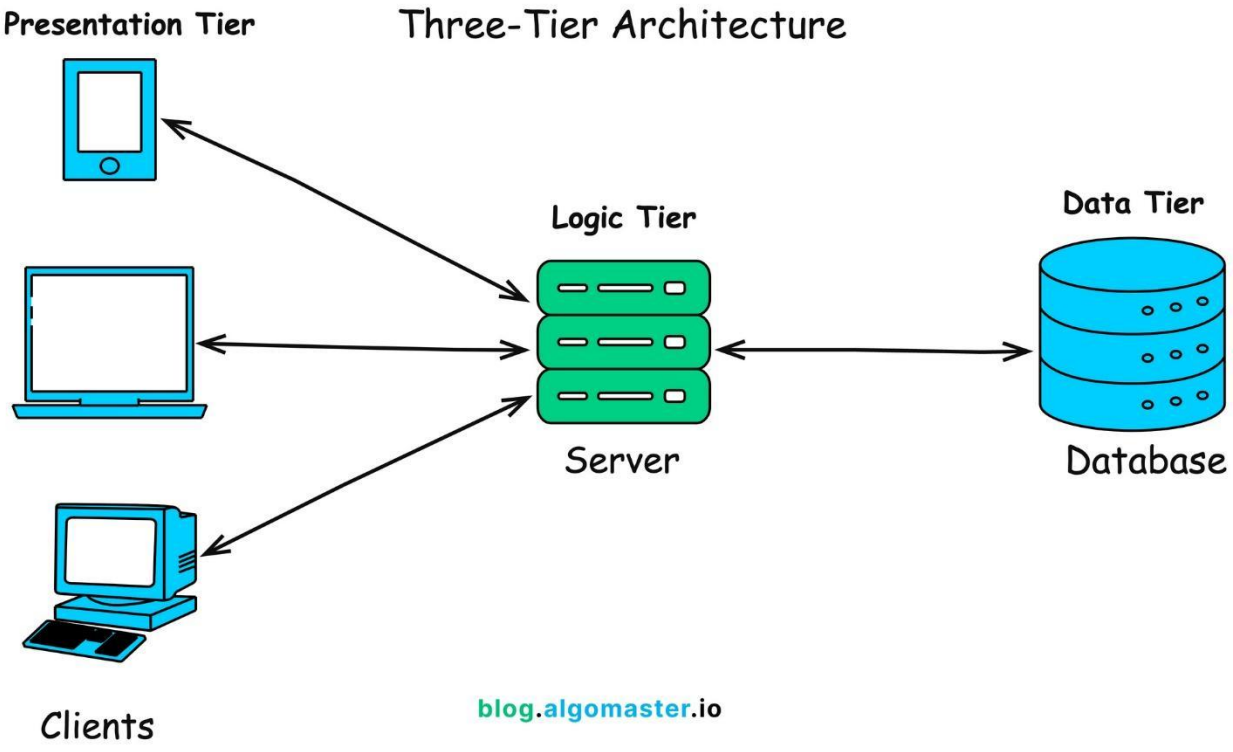
3.4 Usability Requirements

- Simple navigation
- Clear status indicators (Pending, Approved, Rejected)
- Mobile-friendly interface

4.0 SYSTEM ARCHITECTURE

4.1 Component Diagram

Note: This should be a component diagram



Technology Stack:

- **Backend:** Java 17, Spring Boot 3.x, Spring Security, Spring Data JPA
- **Database:** PostgreSQL 14+
- **Web Frontend:** React 18, TypeScript, Tailwind CSS, Axios
- **Mobile:** Kotlin, Jetpack Compose, Retrofit, Room
- **Build Tools:** Maven (Backend), npm/yarn (Web), Gradle (Android)
- **Deployment:** Railway/Heroku (Backend), Vercel/Netlify (Web), APK (Mobile)

5.0 API CONTRACT & COMMUNICATION

5.1 API Standards

Base URL	https://[server_hostname]:[port]/api/v1
Format	JSON for all requests/responses
Authentication	Bearer token (JWT) in Authorization header

Response Structure	{
	"success": boolean,
	"data": object|null,
	"error": {
	"code": string,
	"message": string,

	"details": object|null }, "timestamp": string }

5.2 Endpoint Specifications

Authentication Endpoints

User Registration

Description	User Registration
API URL	/auth/register
HTTP Request Method	POST
Format	JSON for all requests/responses
Authentication	None

Request Payload	{
	"email": <email>,
	"password": <password>,
	"firstname": <firstname>,
	"lastname": <lastname>,
	}

Response Structure	{
	"success": boolean,
	"data": {
	user {
	"email": <email>,
	"firstname": <firstname>,
	"lastname": <lastname>,
	},
	"accessToken": <token>,
	"refreshToken": <token>,
	},
	"error": {
	"code": string,
	"message": string,
	"details": object|null
	},
	"timestamp": string
	}

User Login

Description	User Login
API URL	/auth/login
HTTP Method	POST

Format	JSON for all requests/responses
Authentication	None

Request Payload	{
	“email”: <email>,
	“password”: <password>,
	}

Response Structure	{
	"success": boolean,
	"data": {
	“user”: {
	“email”: <email>,
	“firstname”: <firstname>,
	“lastname”: <lastname>,
	“role”: <role>,
	},
	“accessToken”: <token>,
	“refreshToken”: <token>,
	},
	"error": {
	"code": string,
	"message": string,
	"details": object|null
	},
	"timestamp": string
	}

5.3 Error Handling

HTTP Status Codes

- 200 OK - Successful request
- 201 Created - Resource created
- 400 Bad Request - Invalid input
- 401 Unauthorized - Authentication required/failed
- 403 Forbidden - Insufficient permissions
- 404 Not Found - Resource doesn't exist
- 409 Conflict - Duplicate resource
- 500 Internal Server Error - Server error

Error Code Examples

Example 1	json { "success": false,

	"data": null,
	"error": {
	"code": "AUTH-001",
	"message": "Invalid credentials",
	"details": "Email or password is incorrect"
	},
	"timestamp": "2024-01-28T10:30:00Z"
	}

Example 2	{
	"success": false,
	"data": null,
	"error": {
	"code": "VALID-001",
	"message": "Validation failed",
	"details": {
	"email": "Email is required",
	"password": "Must be at least 8 characters"
	}
	},
	"timestamp": "2024-01-28T10:30:00Z"
	}

Common Error Codes

- AUTH-001: Invalid credentials
- AUTH-002: Token expired
- AUTH-003: Insufficient permissions
- VALID-001: Validation failed
- DB-001: Resource not found
- DB-002: Duplicate entry
- BUSINESS-001: Insufficient stock
- SYSTEM-001: Internal server error

6.0 DATABASE DESIGN

6.1 Entity Relationship Diagram

Note: This should be an ERD (Database Schema)

Detailed Relationships:

- **One-to-One:** User ↔ Cart (Each user has exactly one cart)
- **One-to-Many:** User → Orders (User can have multiple orders)
- **One-to-Many:** Cart → CartItems (Cart contains multiple items)
- **One-to-Many:** Order → OrderItems (Order contains multiple items)
- **Many-to-One:** CartItems → Product (Items reference products)
- **Many-to-One:** OrderItems → Product (Items reference products)

Key Tables:

1. **users** - User accounts and authentication
2. **products** - Product catalog information
3. **carts** - Shopping cart per user
4. **cart_items** - Items in shopping cart
5. **orders** - Customer orders
6. **order_items** - Items in each order
7. **refresh_tokens** - JWT refresh tokens

Table Structure Summary:

- **users:** id, email, password_hash, full_name, role, created_at
- **products:** id, name, description, price, stock, image_url, category
- **carts:** id, user_id, created_at
- **cart_items:** id, cart_id, product_id, quantity
- **orders:** id, order_number, user_id, total, status, shipping_address
- **order_items:** id, order_id, product_id, product_name, quantity, price

7.0 UI/UX DESIGN

7.1 Web Application Wireframes

Note: This should be wireframes from Figma

Homepage (Product Listing)

Header: [Logo] [Search Bar] [Cart Icon] [User Menu]

Content: Product Grid (3 columns desktop)

Each Product Card: Image, Name, Price, "Add to Cart" button

Footer: Links, Copyright

Product Detail Page

Back Button

Product Image (large)

Product Name and Price

Description

Quantity Selector (1-10)

"Add to Cart" and "Buy Now" buttons

Product Specifications

Shopping Cart Page

Cart Title

List of Cart Items (Image, Name, Quantity, Price, Remove)

Order Summary: Subtotal, Shipping, Tax, Total

"Continue Shopping" and "Proceed to Checkout" buttons

Checkout Page

Shipping Address Form

Order Review (Items, Prices, Totals)

"Place Order" button

Terms and Conditions note

Admin Dashboard

Sidebar Navigation: Dashboard, Products, Orders, Users

Product Management: Add New button, Product list with Edit/Delete

Order Management: Order list with status filters

7.2 Mobile Application Wireframes

Note: This should be wireframes from Figma

Bottom Navigation

[🏠 Home] [🔍 Search] [🛒 Cart] [👤 Profile]

Home Screen

Search Bar

Product Grid (2 columns)

Swipe gestures for quick actions

Pull to refresh

Product Detail Screen

Back arrow

Product image (swipeable gallery)

Product info

Quantity selector

"Add to Cart" fixed bottom button

Cart Screen

Edit mode for quantity updates

Swipe to remove items

Order summary sticky bottom

Checkout button

Checkout Flow

Step indicator: Cart → Shipping → Payment → Confirm

Address form (auto-complete)

Order summary

Place order button

Mobile-Specific Features:

- Touch-optimized buttons (min 44x44px)
- Gesture support (swipe, pull-to-refresh)
- Offline caching for product images
- Bottom navigation for main actions
- Simplified forms for mobile input

Design System:

- **Colors:** Primary (#2563EB), Secondary (#7C3AED), Success (#10B981), Error (#EF4444)
- **Typography:** Inter font family, responsive sizing
- **Spacing:** 8px grid system
- **Components:** Consistent buttons, inputs, cards, modals
- **Responsive:** Mobile-first approach, breakpoints at 640px, 768px, 1024px

8.0 PLAN

8.1 Project Timeline

Phase 1: Planning & Design (Week 1-2)

Week 1: Requirements & Architecture

Day 1-2: Project setup and documentation

Day 3-4: Complete FRS and NFR

Day 5-7: System architecture design

Week 2: Detailed Design

Day 1-2: API specification

Day 3-4: Database design

Day 5-6: UI/UX wireframes

Day 7: Implementation plan finalization

Phase 2: Backend Development (Week 3-4)

Week 3: Foundation

Day 1: Spring Boot setup with dependencies

Day 2: Database configuration and entities

Day 3: JWT authentication implementation

Day 4: User management endpoints

Day 5: Product CRUD operations

Week 4: Core Features

Day 1: Cart functionality

Day 2: Order management

Day 3: Search and filtering

Day 4: Error handling and validation

Day 5: API documentation and testing

Phase 3: Web Application (Week 5-6)

Week 5: Frontend Foundation

Day 1: React setup with TypeScript

Day 2: Authentication pages (login, register)

Day 3: Product listing page

Day 4: Product detail page

Day 5: Shopping cart implementation

Week 6: Complete Web Features

Day 1: Checkout flow

Day 2: Order history and confirmation

Day 3: Admin dashboard

Day 4: Responsive design polish

Day 5: API integration and testing

Phase 4: Mobile Application (Week 7-8)

Week 7: Android Foundation

Day 1: Android Studio setup and project structure

Day 2: Authentication screens

Day 3: Product browsing

Day 4: Shopping cart

Day 5: API service layer

Week 8: Complete Mobile App

Day 1: Checkout flow

Day 2: Order management

Day 3: UI polish and animations

Day 4: Testing on emulator/device

Day 5: APK generation and documentation

Phase 5: Integration & Deployment (Week 9-10)

Week 9: Integration Testing

Day 1: End-to-end testing across platforms

Day 2: Bug fixes and optimization

Day 3: Security review

Day 4: Performance testing

Day 5: Documentation updates

Week 10: Deployment

Day 1: Backend deployment (Railway/Heroku)

Day 2: Web app deployment (Vercel/Netlify)

Day 3: Mobile APK distribution

Day 4: Final testing

Day 5: Project submission

Milestones:

- **M1 (End Week 2):** All design documents complete
- **M2 (End Week 4):** Backend API fully functional
- **M3 (End Week 6):** Web application complete
- **M4 (End Week 8):** Mobile application complete
- **M5 (End Week 10):** Full system deployed and integrated

Critical Path:

1. Authentication system (Week 3)
2. Product catalog API (Week 3-4)
3. Shopping cart functionality (Week 4)
4. Checkout process (Week 6)
5. Cross-platform testing (Week 9)

Risk Mitigation:

- Start with simplest working version of each feature
- Test integration points early and often
- Keep backup of working versions
- Focus on core functionality before enhancements