



CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G4

System Integration and Architecture

System Design Document (SDD)

Project Title: UniGear Tracker

Prepared By: Ybañez, Liezel, A.

Version: 1.0

Date: February 02, 2026

Status: Draft

REVISION HISTORY TABLE

Version	Date	Author	Changes Made	Status
0.1	02/02/26	Ybañez, Liezel, A.	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	4
1.0 INTRODUCTION	5
2.0 FUNCTIONAL REQUIREMENTS SPECIFICATION.....	5
3.0 NON-FUNCTIONAL REQUIREMENTS	9
4.0 SYSTEM ARCHITECTURE	11
5.0 API CONTRACT & COMMUNICATION	12
6.0 DATABASE DESIGN	17
7.0 UI/UX DESIGN.....	18
8.0 PLAN	21

EXECUTIVE SUMMARY

1.1 Project Overview

UniGear Tracker is a generalized inventory and borrowing system in the university that lets students borrow and reserve items and equipment from the university offices and laboratory that they need for events and academic purposes. The system includes a Spring Boot backend API, a React web application, and an Android mobile app, all integrated to provide a seamless transaction experience across platforms.

1.2 Objectives

1. Develop a fully functional e-commerce MVP with authentication, product catalog, shopping cart, and checkout
2. Implement a three-tier architecture using Spring Boot (backend), React (web), and Android (mobile)
3. Create RESTful APIs for communication between all system components
4. Design a responsive user interface that works consistently across web and mobile platforms
5. Deploy all system components to production-ready environments

1.3 Scope

Included Features:

- User registration and authentication (email/password)
- Product listing with search functionality
- Shopping cart management (add, remove, update quantities)
- Checkout process with shipping information collection
- Order placement with confirmation
- Admin panel for product management
- Responsive web interface
- Native Android mobile application
- PostgreSQL database with relational schema

Excluded Features:

- Real payment gateway integration

- Product reviews and ratings system
- Inventory management automation
- Email notification system
- Social media login integration
- Push notifications
- Advanced analytics dashboard

1.0 INTRODUCTION

1.1 Purpose

This document serves as the comprehensive design specification for the Marketplace MVP system. It provides detailed requirements, architectural decisions, API contracts, database design, and implementation roadmap to guide development and ensure all components integrate seamlessly.

2.0 FUNCTIONAL REQUIREMENTS SPECIFICATION

2.1 Project Overview

Project Name: Marketplace MVP

Domain: E-commerce/Retail

Primary Users: Customers and Administrators

Problem Statement: Users need a simple, fast way to browse and purchase products without the complexity of large e-commerce platforms.

Solution: A minimalist marketplace focusing on core shopping functionality with consistent experience across web and mobile.

2.2 Core User Journeys

Journey 1: First-time Customer Borrow

1. User visits web application
2. Clicks "Sign Up" and creates account
3. Browses product listing
4. Clicks product to view details

5. Adds product to cart
6. Views cart and proceeds to checkout
7. Enters shipping information
8. Places order and receives confirmation

Journey 2: Returning Customer Borrow

1. User logs in with existing credentials
2. Searches for specific product
3. Adds multiple products to cart
4. Updates cart quantities
5. Completes checkout using saved information
6. Receives order confirmation

Journey 3: Administrator Product Management

1. Admin logs in with admin credentials
2. Navigates to product management section
3. Adds new product with details
4. Edits existing product information
5. Views list of all orders
6. Manages product inventory

2.3 Feature List (MoSCoW)

MUST HAVE

1. User authentication (register, login, logout)
2. Product catalog (list, detail, search)
3. Shopping cart (add, remove, update, view)
4. Checkout process (shipping, order placement)
5. Admin panel (product CRUD, order view)

SHOULD HAVE

1. Product categories and filtering

2. Order history for users
3. Basic input validation feedback
4. Responsive design for all screen sizes

COULD HAVE

1. Product wishlist
2. Basic sales dashboard
3. Order status tracking

WON'T HAVE

1. Real payment processing
2. Email notifications
3. Social login
4. Advanced analytics
5. Multi-language support

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: Product Catalog

- **Screens:** Product Listing, Product Detail
- **Display:** Grid layout, product images, names, prices
- **Search:** By product name
- **API Endpoints:** GET /products, GET /products/{id}, GET /products/search
- **Admin Functions:** POST /products, PUT /products/{id}, DELETE /products/{id}

Feature: Borrowing Cart

- **Screens:** Cart View
- **Functions:** Add product, remove product, update quantity
- **Persistence:** Database storage per user
- **API Endpoints:** GET /cart, POST /cart/items, PUT /cart/items/{id}, DELETE /cart/items/{id}

Feature: Checkout Process

- **Screens:** Checkout Form, Order Confirmation
- **Data Collected:** Borrower address (name, address, city, zip)
- **Process:** Validate input, create order, clear cart
- **API Endpoints:** POST /orders, GET /orders/{id}

Feature: Admin Panel

- **Screens:** Product Management, Order List
- **Functions:** Add/edit/delete products, view orders
- **Access Control:** Admin role required
- **API Endpoints:** Admin-prefixed endpoints with role validation

2.5 Acceptance Criteria

AC-1: Successful User Registration

Given I am a new user

When I enter valid email and strong password

And confirm password matches

And click "Create Account"

Then my account should be created

And I should be automatically logged in

And redirected to the homepage

AC-2: Product Purchase Flow

Given I am logged in as a customer

When I add a product to my cart

And proceed to checkout
And enter valid shipping information
And place the order
Then I should see order confirmation
And my cart should be empty
And the order should appear in admin panel

AC-3: Admin Product Management

Given I am logged in as an administrator
When I add a new product with valid details
And save the product
Then the product should appear in customer product listing
And be available for purchase

3.0 NON-FUNCTIONAL REQUIREMENTS

3.1 Performance Requirements

- API response time: ≤ 2 seconds for 95% of requests
- Web page load time: ≤ 3 seconds on broadband
- Mobile app cold start: ≤ 3 seconds
- Support 100 concurrent users
- Database queries complete within 500ms

3.2 Security Requirements

- HTTPS for all communications
- JWT token authentication
- Password hashing with bcrypt (salt rounds = 12)
- SQL injection prevention

- XSS protection
- Rate limiting: 100 requests/minute per IP
- Admin endpoints require role verification

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+

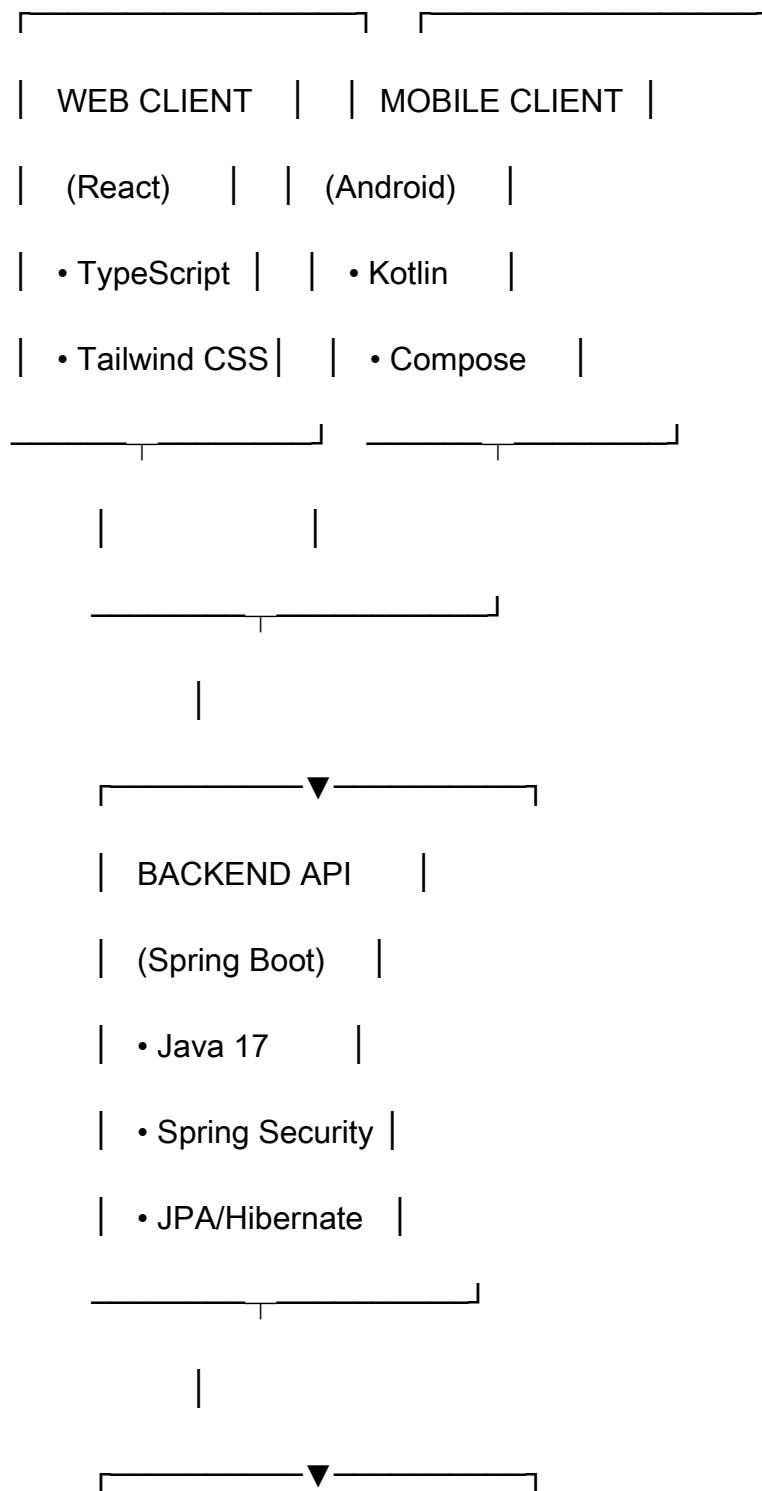
3.4 Usability Requirements

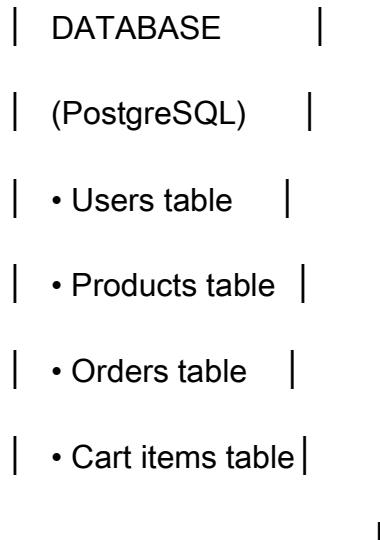
- Complete first purchase within 5 minutes for new users
- WCAG 2.1 Level AA compliance for web
- Consistent navigation across all pages
- Clear error messages with recovery options
- Touch targets minimum 44x44px on mobile
- Keyboard navigation support

4.0 SYSTEM ARCHITECTURE

4.1 Component Diagram

Note: This should be a component diagram





Technology Stack:

- **Backend:** Java 19, 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://api.marketplace.com/api/v1>
- **Format:** JSON for all requests/responses
- **Authentication:** Bearer token (JWT) in Authorization header
- **Response Structure:**

json

{

 "success": boolean,

```
"data": object|null,  
"error": {  
    "code": string,  
    "message": string,  
    "details": object|null  
},  
"timestamp": string  
}
```

5.2 Endpoint Specifications

Authentication Endpoints

POST /auth/register

Body: {email, password, confirmPassword, fullName?}

Response: {user: {id, email, name}, token, refreshToken}

POST /auth/login

Body: {email, password}

Response: {user: {id, email, name, role}, token, refreshToken}

POST /auth/logout

Headers: Authorization: Bearer {token}

Response: {message: "Logged out successfully"}

Product Endpoints

GET /products

Query: ?page=1&limit=20&search=keyword&category=electronics

Response: {products: [...], pagination: {page, limit, total, pages}}

GET /products/{id}

Response: {product: {id, name, description, price, stock, imageUrl, category}}

POST /products (Admin only)

Body: {name, description, price, stock, imageUrl, category}

Response: {product: {...}}

PUT /products/{id} (Admin only)

Body: {name?, description?, price?, stock?, imageUrl?, category?}

Response: {product: {...}}

Cart Endpoints

GET /cart (Authenticated)

Response: {cart: {id, items: [...], total, itemCount}}

POST /cart/items

Body: {productId, quantity}

Response: {message: "Added to cart", cartItem: {...}}

PUT /cart/items/{itemId}

Body: {quantity}

Response: {message: "Cart updated", cartItem: {...}}

DELETE /cart/items/{itemId}

Response: {message: "Removed from cart"}

Order Endpoints

POST /orders

Body: {shippingAddress: {fullName, address, city, zipCode, country}}

Response: {order: {id, orderNumber, total, status, items: [...], createdAt}}

GET /orders

Response: {orders: [...]}

GET /orders/{id}

Response: {order: {...}}

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

```
json
{
  "success": false,
  "data": null,
  "error": {
    "code": "AUTH-001",
```

```
"message": "Invalid credentials",
"details": "Email or password is incorrect"
},
"timestamp": "2024-01-28T10:30:00Z"
}

{
  "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

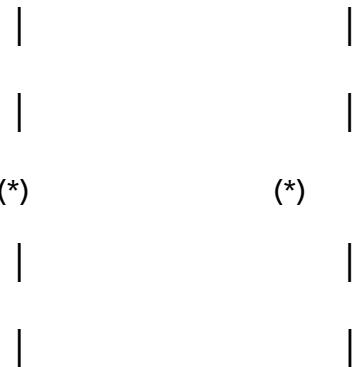
- SYSTEM-001: Internal server error

6.0 DATABASE DESIGN

6.1 Entity Relationship Diagram

Note: This should be an ERD

USERS (1) —— (1) CARTS (1) —— (*) CART_ITEMS (*) —— (1) PRODUCTS



ORDERS (1) —— (*) ORDER_ITEMS (*) ——

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