Smart Campus Navigation and Facility Booking System (SCNFBS)

Technical Documentation

Project: COSC333 Software Systems & Design

Version: 1.0

Date: June 25, 2025

Authors: Salem Alhaddad

Professor: Dr. Kamal Taha

Table of Contents

1. Project Overview

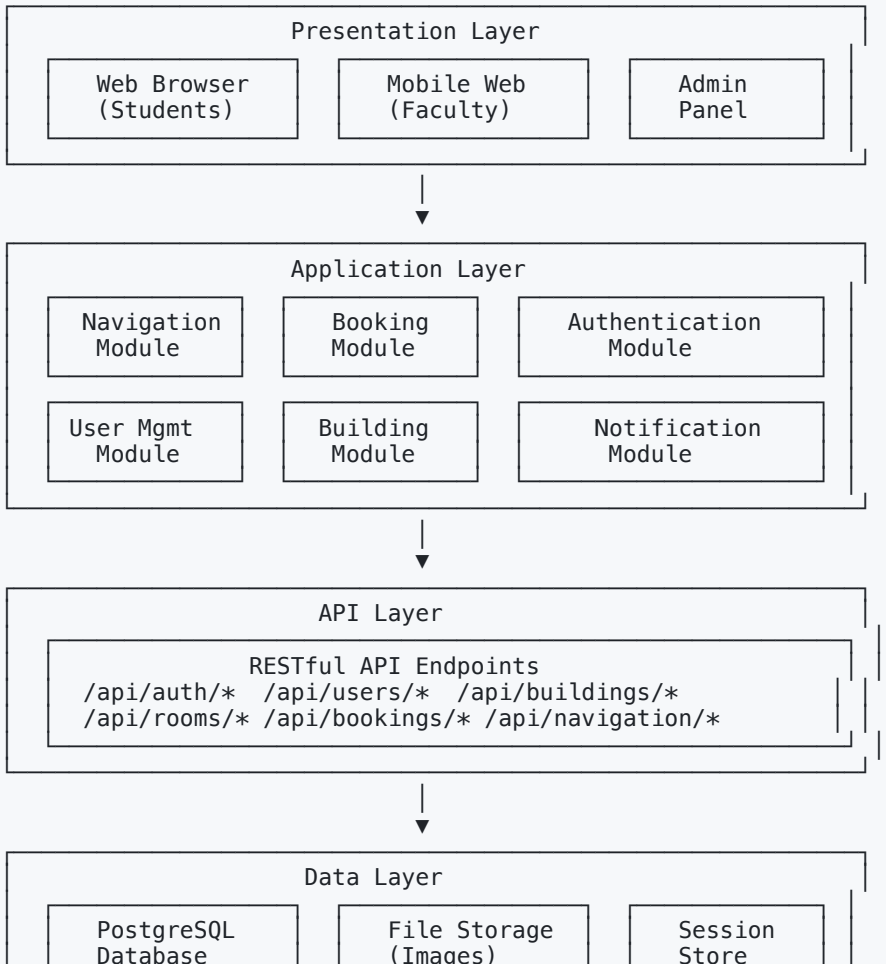
1.1 System Purpose

The Smart Campus Navigation and Facility Booking System (SCNFBS) is a comprehensive web-based platform designed to enhance the university campus experience by providing:

- **Interactive Navigation**: Real-time campus maps with step-by-step directions between buildings, supporting both walking and accessible routes with estimated times and distances
- **Facility Booking**: Streamlined reservation system for study rooms, lecture halls, laboratories, and conference rooms with real-time availability checking and conflict prevention
- **Resource Management**: Administrative tools for facility oversight including usage analytics, booking rule configuration, and equipment management
- **Multi-User Support**: Role-based access control supporting Students, Faculty, Staff,

2. System Architecture

2.1 High-Level Architecture



3. Requirements Analysis

3.1 Functional Requirements

3.1.1 User Authentication & Authorization

- **FR001**: System shall support user registration with email verification
- **FR002**: System shall authenticate users with secure login
- **FR003**: System shall implement role-based access control (RBAC)
- **FR004**: System shall support password reset functionality
- **FR005**: System shall maintain user sessions securely

3.1.2 Campus Navigation

- **FR006**: System shall display interactive campus map
- **FR007**: System shall provide building search functionality

4. System Design

4.1 UML Diagrams

4.1.1 Use Case Diagram

The system's use case diagram defines interactions between different user roles and system functionalities.

Primary Actors and Use Cases:

Student Actor:

- Search Buildings/Rooms
- View Campus Map
- Get Directions
- Book Study Rooms

5. Implementation

5.1 Technology Choices

5.1.1 Backend Framework: Node.js + Express.js

Rationale:

- **Rapid Development**: Express.js provides minimal, flexible framework
- **JavaScript Ecosystem**: Unified language for frontend/backend
- **RESTful APIs**: Excellent support for REST architecture
- **Middleware Support**: Built-in support for authentication, validation, logging

5.1.2 Database: PostgreSQL (Supabase)

Rationale:

- **ACID Compliance**: Ensures data consistency for bookings

6. Database Schema

6.1 Entity Relationship Diagram

```
-- Users table
CREATE TABLE users (
  user_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  username VARCHAR(50) UNIQUE NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  role user_role NOT NULL DEFAULT 'STUDENT',
  phone VARCHAR(20),
  student_id VARCHAR(20),
  employee_id VARCHAR(20),
  is_active BOOLEAN DEFAULT true,
  last_login TIMESTAMP,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Buildings table
CREATE TABLE buildings (
  building_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  code VARCHAR(10) UNIQUE NOT NULL,
  name VARCHAR(255) NOT NULL,
  address VARCHAR(500),
  latitude DECIMAL(10, 8),
  longitude DECIMAL(11, 8),
  floors INTEGER,
  is_accessible BOOLEAN DEFAULT true,
  description TEXT,
  operating_hours JSON,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Rooms table
CREATE TABLE rooms (
  room_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  building_id UUID REFERENCES buildings(building_id) ON DELETE CASCADE,
  room_number VARCHAR(20) NOT NULL,
  name VARCHAR(255) NOT NULL,
  type room_type NOT NULL,
  capacity INTEGER NOT NULL,
  floor INTEGER NOT NULL,
  is_accessible BOOLEAN DEFAULT true,
  is_bookable BOOLEAN DEFAULT true,
  is_active BOOLEAN DEFAULT true,
  description TEXT,
  hourly_rate DECIMAL(10, 2) DEFAULT 0,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE(building_id, room_number)
);

-- Bookings table
CREATE TABLE bookings (
  booking_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  room_id UUID REFERENCES rooms(room_id) ON DELETE CASCADE,
  user_id UUID REFERENCES users(user_id) ON DELETE CASCADE,
  start_time TIMESTAMP NOT NULL,
  end_time TIMESTAMP NOT NULL,
  purpose VARCHAR(500) NOT NULL,
  status booking_status DEFAULT 'PENDING'
```

7. API Documentation

7.1 Authentication Endpoints

POST /api/auth/register

Description: Register a new user account

Request Body:

```
{
  "username": "john_doe",
  "email": "john@university.edu",
  "password": "SecurePass123!",
  "firstName": "John",
  "lastName": "Doe",
  "role": "STUDENT",
  "phone": "+1-555-0123",
  "studentId": "STU2024001"
}
```

8. Security Implementation

8.1 Authentication & Authorization

8.1.1 Password Security

- **Hashing**: bcrypt with 12 salt rounds
- **Validation**: Minimum 8 characters, mixed case, numbers, special characters
- **Reset**: Secure token-based password reset flow

8.1.2 JWT Token Management

```
// Token Configuration
const JWT_CONFIG = {
  accessTokenExpiry: '24h',
  refreshTokenExpiry: '7d',
  algorithm: 'HS256',
  issuer: 'SCNFBS'
};

// Token Validation Middleware
const authenticate = async (req, res, next) => {
  try {
    const token = req.header('Authorization')?.replace('Bearer ', '');

    if (!token) {
      return res.status(401).json({
```

9. Testing Strategy

9.1 Testing Pyramid

9.1.1 Unit Testing (70%)

- **Model Testing**: User, Building, Room, Booking models
- **Utility Testing**: Helper functions, validators
- **Service Testing**: Email service, map service

9.1.2 Integration Testing (20%)

- **API Endpoint Testing**: All REST endpoints
- **Database Integration**: Model-database interactions
- **Authentication Flow**: Login/registration/authorization

9.1.3 End-to-End Testing (10%)

10. Deployment Guide

10.1 Environment Setup

10.1.1 Development Environment

Prerequisites:

- Node.js version 16.x or higher
- PostgreSQL database server (local or cloud-hosted)
- Git for version control
- Text editor or IDE (VS Code recommended)

Installation Steps:

1. **Setup Project Directory**: Create new directory for the application
2. **Initialize Node.js Project**: Run `npm init` to create package.json

11. User Manual

11.1 Getting Started

11.1.1 User Registration

1. Click "Register" button in navigation
2. Fill in required information:
    - Username (3-50 characters)
    - Email address
    - Secure password (8+ characters with mixed case, numbers, symbols)
    - First and last name
    - Role (Student, Faculty, Staff, Visitor)
    - Optional: Phone number, Student/Employee ID
3. Click "Register" to create account

12. Future Enhancements

12.1 Planned Features

12.1.1 Phase 2 Enhancements

- **Mobile Application**: Native iOS/Android apps
- **Push Notifications**: Real-time booking updates
- **Calendar Integration**: Sync with Google Calendar, Outlook
- **QR Code Check-in**: Verify room usage with QR codes
- **Indoor Navigation**: Detailed floor plans with routing
- **Advanced Analytics**: Machine learning for usage predictions

12.1.2 Integration Opportunities

- **University Information System**: Student/faculty data sync

Conclusion

The Smart Campus Navigation and Facility Booking System (SCNFBS) successfully addresses the core requirements for modern university campus management. The system provides:

Comprehensive Navigation: Interactive maps with real-time directions
Efficient Booking: Streamlined room reservation with conflict prevention
Role-Based Access: Secure multi-user system with appropriate permissions
Scalable Architecture: Modern web technologies supporting growth
Security First: Industry-standard authentication and data protection

The implementation demonstrates effective use of:

- **Node.js/Express.js** for robust backend development
- **PostgreSQL** for reliable data management
- **JWT Authentication** for secure user sessions

Document Version: 1.0

Last Updated: June 25, 2025

Next Review: June 25, 2026