|  |  |  |
| --- | --- | --- |
| **Batch.** | **2301C1** |  |
| Serial No. | Enrollment Number | Student Name |
| 1. | Student1441770 | Umm-e-Aiman Hassan |
| 2. | Student1447028 | Mahnoor Zia |
| 3. | Student1416352 | MUHAMMAD Misbah Hassan |
| 4. | Student1416324 | TAHA MAHMOOD . |
| 5. | Student1416361 | M Daniyal Ghouri |
| 6. | Student1400093 | ALI ANSARI |
| 7. | Student1449711 | MUHAMMAD USMAN SIDDIQUI |

luXURYSTAY HOSPITALITY

# ****Developer Guide: LuxuryStay HMS****

**Version:** 1.0  
**Project:** Hotel Management System (HMS)  
**Client:** LuxuryStay Hospitality  
**Date:** October 26, 2023  
**Status:** Approved for Development  
**Confidentiality:** Internal Use Only

## **Table of Contents**

Contents

[Developer Guide: LuxuryStay HMS 1](#_Toc208978455)

[Table of Contents 1](#_Toc208978456)

[1.0 Introduction & Objectives 2](#_Toc208978457)

[1.1 Project Vision 2](#_Toc208978458)

[1.2 Document Purpose 2](#_Toc208978459)

[1.3 Target Audience 2](#_Toc208978460)

[2.0 Technology Stack & Architecture 2](#_Toc208978461)

[2.1 Approved Technology Stack 2](#_Toc208978462)

[2.2 System Architecture 3](#_Toc208978463)

[3.0 Core Modules & API Design 4](#_Toc208978464)

[3.1 Authentication & Authorization Module 4](#_Toc208978465)

[3.2 Room Management Module 5](#_Toc208978466)

[3.3 Booking & Reservation Module 5](#_Toc208978467)

[3.4 Billing & Payment Module 5](#_Toc208978468)

[3.5 Reporting & Dashboard Module 5](#_Toc208978469)

[4.0 Development Workflow & Standards 6](#_Toc208978470)

[4.1 Version Control Strategy 6](#_Toc208978471)

[4.2 Code Quality Standards 6](#_Toc208978472)

[4.3 API Design Standards 6](#_Toc208978473)

[4.4 Security Requirements 7](#_Toc208978474)

[5.0 Getting Started (Local Development) 7](#_Toc208978475)

[5.1 Prerequisites 7](#_Toc208978476)

[5.2 Development Environment Setup 8](#_Toc208978477)

[5.3 Testing 8](#_Toc208978478)

[6.0 Deployment Strategy 9](#_Toc208978479)

[6.1 Environments 9](#_Toc208978480)

[6.2 CI/CD Pipeline 9](#_Toc208978481)

[6.3 Monitoring & Logging 9](#_Toc208978482)

[7.0 Future Considerations & Scalability 10](#_Toc208978483)

[7.1 Short-term Roadmap (Next 6 months) 10](#_Toc208978484)

[7.2 Medium-term Roadmap (6-12 months) 10](#_Toc208978485)

[7.3 Long-term Vision (1-2 years) 10](#_Toc208978486)

[8.0 Support & Contact Information 10](#_Toc208978487)

[8.1 Technical Support 10](#_Toc208978488)

[8.2 Key Contacts 11](#_Toc208978489)

[8.3 Emergency Procedures 11](#_Toc208978490)

## **1.0 Introduction & Objectives**

### **1.1 Project Vision**

To develop a modern, scalable, and user-centric Hotel Management System that centralizes operations for LuxuryStay Hospitality, enhancing efficiency, guest satisfaction, and data-driven decision-making across all properties.

### **1.2 Document Purpose**

This guide serves as the primary technical reference for the development team, outlining system architecture, technology stack, feature specifications, and development standards to ensure a consistent, high-quality, and maintainable codebase.

### **1.3 Target Audience**

* Back-end and Front-end Developers
* Database Administrators & DevOps Engineers
* Quality Assurance Engineers
* Project Technical Leads & Architects

## **2.0 Technology Stack & Architecture**

### **2.1 Approved Technology Stack**

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Technology** | **Version** | **Rationale** |
| Frontend | React with TypeScript | 18.x | Type safety, component architecture, rich ecosystem |
| Backend | Node.js with NestJS framework | 20.x | Structured, scalable, TypeScript support |
| Database | PostgreSQL | 14.x | ACID compliance, JSON support, reliability |
| Authentication | JWT with Passport.js | Latest | Industry standard for stateless authentication |
| Caching | Redis | 7.x | High-performance session and data caching |
| API Documentation | Swagger/OpenAPI | 3.0 | Standardized API documentation |
| Containerization | Docker | Latest | Environment consistency |

### **2.2 System Architecture**

The system will follow a **modular monolith** pattern with clear separation of concerns to facilitate future microservices migration.

**Architecture Overview:**

text

[React Web Client]

⇅

[API Gateway (NestJS)]

⇅

[Core Application Modules]

⇅

[Data Access Layer]

⇅

[PostgreSQL Database] + [Redis Cache]

**Key Components:**

* **API Gateway:** Unified entry point handling authentication, routing, and request validation
* **Service Modules:** Independently developed modules with defined interfaces
* **Data Layer:** PostgreSQL for persistent storage, Redis for caching and sessions
* **Client Layer:** Responsive React application with role-based UI components

## **3.0 Core Modules & API Design**

### **3.1 Authentication & Authorization Module**

**Purpose:** Secure user authentication and role-based access control

**Key Endpoints:**

* POST /api/v1/auth/login - User authentication
* POST /api/v1/auth/register - User registration (Admin only)
* GET /api/v1/auth/me - Current user profile
* POST /api/v1/auth/refresh-token - Token refresh

**Data Models:**

typescript

interface User {

id: string;

email: string;

passwordHash: string;

role: UserRole;

hotelId?: string;

isActive: boolean;

lastLogin: Date;

}

### **3.2 Room Management Module**

**Purpose:** Comprehensive management of room inventory and status

**Key Endpoints:**

* GET /api/v1/rooms - Retrieve rooms with filtering
* POST /api/v1/rooms - Create new room (Admin)
* PUT /api/v1/rooms/{id} - Update room details
* GET /api/v1/room-availability - Check room availability

### **3.3 Booking & Reservation Module**

**Purpose:** End-to-end booking management from creation to completion

**Key Endpoints:**

* POST /api/v1/bookings - Create new booking
* GET /api/v1/bookings - Retrieve bookings with filters
* PUT /api/v1/bookings/{id}/checkin - Process guest check-in
* PUT /api/v1/bookings/{id}/checkout - Process guest check-out

### **3.4 Billing & Payment Module**

**Purpose:** Invoice generation and payment processing

**Key Endpoints:**

* POST /api/v1/bookings/{id}/invoice - Generate booking invoice
* GET /api/v1/invoices/{id} - Retrieve invoice details
* POST /api/v1/invoices/{id}/payment - Process payment

### **3.5 Reporting & Dashboard Module**

**Purpose:** Business intelligence and operational reporting

**Key Endpoints:**

* GET /api/v1/reports/occupancy - Occupancy rate reports
* GET /api/v1/reports/revenue - Revenue analytics
* GET /api/v1/reports/guest-stats - Guest demographic reports

## **4.0 Development Workflow & Standards**

### **4.1 Version Control Strategy**

* **Branching Model:** GitFlow workflow
* **Main Branches:**
  + main (production releases)
  + develop (integration branch)
* **Supporting Branches:**
  + feature/\* (new functionality)
  + release/\* (release preparation)
  + hotfix/\* (production fixes)

### **4.2 Code Quality Standards**

* **TypeScript:** Strict mode enabled
* **ESLint:** Airbnb style guide with custom modifications
* **Prettier:** Consistent code formatting
* **Pre-commit Hooks:** Automated linting and testing
* **Testing Requirements:**
  + Minimum 80% test coverage
  + Unit tests for all services and utilities
  + Integration tests for API endpoints
  + E2E tests for critical user journeys

### **4.3 API Design Standards**

* **RESTful conventions** with proper HTTP methods and status codes
* **Versioning:** API version in URL path (/api/v1/)
* **Response Format:**

json

{

"success": true,

"data": {},

"metadata": {}

}

* **Error Response Format:**

json

{

"success": false,

"error": {

"code": "ERROR\_CODE",

"message": "Human readable message",

"details": {}

}

}

### **4.4 Security Requirements**

* Password hashing with bcrypt (12 rounds)
* JWT tokens with appropriate expiration
* Role-based access control on all endpoints
* SQL injection prevention through parameterized queries
* XSS protection through input sanitization
* CORS configuration for trusted domains only
* Environment variables for all sensitive configuration

## **5.0 Getting Started (Local Development)**

### **5.1 Prerequisites**

* Node.js 18.x or higher
* PostgreSQL 14.x or higher
* Git
* Docker (optional)

### **5.2 Development Environment Setup**

1. **Clone repository:**

bash

git clone https://github.com/luxurystay/hms.git

cd hms

1. **Install dependencies:**

bash

npm install

1. **Environment configuration:**

bash

cp .env.example .env

*# Configure database and other environment variables*

1. **Database setup:**

bash

npm run db:migrate

npm run db:seed

1. **Start development servers:**

bash

*# Backend server*

npm run dev:server

*# Frontend client*

npm run dev:client

### **5.3 Testing**

bash

*# Unit tests*

npm test

*# E2E tests*

npm run test:e2e

*# Test coverage*

npm run test:cov

## **6.0 Deployment Strategy**

### **6.1 Environments**

* **Development:** Feature branch deployments for testing
* **Staging:** Mirror of production for final validation
* **Production:** Live customer environment

### **6.2 CI/CD Pipeline**

* **GitHub Actions** for automated testing and deployment
* **Automated testing** on all pull requests
* **Containerization** with Docker for consistent environments
* **AWS ECS/EKS** for container orchestration
* **Blue-green deployment** strategy for zero downtime

### **6.3 Monitoring & Logging**

* **Application monitoring** with Prometheus/Grafana
* **Log aggregation** with ELK stack
* **Error tracking** with Sentry
* **Performance monitoring** with New Relic

## **7.0 Future Considerations & Scalability**

### **7.1 Short-term Roadmap (Next 6 months)**

* Mobile application for guests
* Integration with payment gateways
* Housekeeping management module
* Multi-language support

### **7.2 Medium-term Roadmap (6-12 months)**

* Microservices architecture migration
* Real-time features with WebSockets
* Advanced analytics with data warehouse integration
* CRM integration

### **7.3 Long-term Vision (1-2 years)**

* AI-powered pricing and recommendations
* IoT integration for smart rooms
* Multi-property management support
* White-label solution for other hotel chains

## **8.0 Support & Contact Information**

### **8.1 Technical Support**

* **GitHub Issues:** For bug reports and feature requests
* **Slack Channel:** #luxurystay-hms-dev for real-time discussions
* **Wiki:** Comprehensive documentation and tutorials

### **8.2 Key Contacts**

* **Technical Lead:** [Name] - [email]
* **Project Manager:** [Name] - [email]
* **Database Administrator:** [Name] - [email]
* **DevOps Engineer:** [Name] - [email]

### **8.3 Emergency Procedures**

For critical production issues:

1. Notify technical lead immediately
2. Create hotfix branch from main
3. Follow emergency deployment procedures
4. Post-mortem analysis within 24 hours of resolution