Software Development Life Cycle (SDLC) Document

Project: Guestly – Hotel Management System

Author: Nikolozi Gagua

Technology Stack: Django 4.x, MySQL/SQLite, HTML/CSS/JavaScript, Bootstrap

# 1. Planning Phase

## 1.1 Requirements Gathering

***Key Requirements:***  
- Role-specific dashboards for Guests, Receptionists, Housekeeping, and Managers.  
- Efficient workflows for room booking, check-in, and check-out processes.  
- Real-time tracking for service requests, cleaning, maintenance, and food orders.  
- financial reporting to monitor revenue and staff payments.

## 1.2 Feasibility Study

- Technical Feasibility: Django is chosen for its robust framework, development capabilities, and built-in authentication system, I chose this to ensure secure and scalable solution.  
- Operational Feasibility: The role-based workflows align as much as possible with standard hotel operations, making potential future adoption straightforward.  
- Economic Feasibility: Using open-source tools like Django and MySQL minimizes costs while meeting my needs.

# 2. Analysis Phase

## 2.1 Functional Requirements

|  |
| --- |
| Functional Requirements Table |
| |  |  | | --- | --- | | Module | Functions | | User Management | Assign roles, handle login/logout, redirect to role-specific dashboards. | | Booking System | Check room availability, create or cancel bookings. | | Housekeeping | Update cleaning statuses, track task completion. | | Room Service | Place food orders, update order status (pending to delivered). | | Reporting | Calculate revenue, generate staff payment summaries. | |

## 2.2 Non-Functional Requirements

- Performance: Dashboards and key pages load in under 2 seconds to ensure a smooth user experience.  
- Security: Role-based access control (RBAC) is implemented using Django decorators to restrict unauthorized access.  
- Usability: The interface, built with Bootstrap, is responsive and intuitive across devices.

# 3. Design Phase

## 3.1 System Architecture

- Frontend: Django Templates paired with Bootstrap for a responsive and visually consistent user interface.  
- Backend: Django ORM for efficient database interactions, integrated with

- Authentication: Django’s built-in authentication system, extended with my CustomUser model to support role-based access.

## 3.2 Database Schema

## Database Schema Table

|  |  |
| --- | --- |
| Table | Purpose |
|  |  |
| CustomUser | Stores user details and roles (guest, staff, manager). |
| Room | Manages room types, availability, and occupancy status. |
| Booking | Tracks check-in/check-out dates and links guests to rooms. |
| ServiceRequest | Logs cleaning and maintenance requests with status updates. |
| FoodOrder | Records food orders and their delivery progress. |

## 3.3 Wireframes

Wireframes will be created to visualize the user interface for key dashboards:  
- Guest Dashboard: Features a simple booking form, service request buttons, and a summary of active bookings.  
- Receptionist Dashboard: Displays pending check-ins/check-outs and room availability.  
- Housekeeping Dashboard: Lists assigned cleaning tasks with status update options.  
- Manager Dashboard: Includes financial charts, staff management tools, and user activity logs.

# 4. Development Phase

## 4.1 Implementation

Development is structured into three iterative sprints to ensure steady progress:  
- Iteration 1 : Build core user authentication and the room booking system.  
- Iteration 2 : Implement receptionist workflows (check-in/check-out) and housekeeping features.  
- Iteration 3 : Develop financial reporting tools and room service functionality.

# 5. Testing Phase

## 5.1 Testing Tools

- Unit Tests: Django’s built-in testing framework to validate backend logic.  
- Manual Testing: Browser-based testing to verify frontend responsiveness and usability.

# 6 SDLC Process Overview

The project adopts a hybrid Waterfall-Agile methodology to balance structure with flexibility:  
- Phases: Planning, Analysis, Design, Development, Testing,.  
- Iterations: Feedback loops after each development sprint allow for continuous refinement.

# Appendices

- GitHub Repository: github.com/NikoloziGagua/guestly\_hotel