Guestly – Hotel Management System: Minimum Viable Product (MVP) Description

Author: Nikolozi Gagua

Technology Stack: Django MySQL, HTML/CSS/JavaScript, Bootstrap

# 1. Project Overview

Guestly is a web-based hotel management system designed to demonstrate my proficiency in full-stack development using Django, SQLite, and Bootstrap. The MVP focuses on delivering core functionalities for hotel operations, emphasizing clean code, intuitive design, and role-based workflows. This application is not intended for production deployment but serves as a portfolio piece to highlight my ability to build a functional, user-friendly system from scratch.

# 2. MVP Objectives

- Demonstrate expertise in Django’s backend development, including models, views, and authentication.  
- Showcase front-end skills with responsive, Bootstrap-based interfaces.  
- Implement role-based access control to highlight security practices.  
- Provide user-friendly experience for key hotel management tasks.  
- Maintain well-documented, modular code to reflect best practices.

.

# 3. MVP Features

## 3.1 User Management

- User registration and login with role-based authentication (Guest, Receptionist, Housekeeping, Manager).  
- CustomUser model extending Django’s authentication system to include roles.  
- Role-specific dashboards redirecting users to relevant interfaces.

## 3.2 Room Booking System

- Guests can view available rooms and create bookings with check-in/check-out dates.  
- Receptionists can confirm bookings, check guests in, and check them out.  
- Room status updates dynamically (available, reserved, occupied).

## 3.3 Housekeeping Workflow

- Housekeeping staff can view assigned rooms and mark them as cleaned.  
- Room status updates to “available” upon cleaning completion.  
- Simple task list interface for housekeeping dashboard.

## 3.4 Basic Reporting

- Managers can view a summary of total bookings and revenue for a selected period.  
- Data displayed in a clean table format (no advanced charts in MVP).

## 3.5 Service Requests

- Guests can submit basic service requests (e.g., extra towels, room cleaning).  
- Requests are logged and visible to housekeeping for action.

# 4. Technical Implementation

## 4.1 Backend

- Framework: Django for rapid development.  
- Database: MySQL(primary) Sqlite3(alternative)  
- Authentication: Django’s built-in auth system with custom decorators for role-based access.  
- Code Structure: Follows Django’s Model-Template-View (MTV) pattern for clarity.

## 4.2 Frontend

- Templates: Django Templates with Bootstrap 5 for responsive, professional styling.  
- JavaScript: Minimal use for dynamic form validation and user interactions.  
- Design: Clean, intuitive layouts with role-specific dashboards.

# 5. Development Scope and Constraints

To keep the MVP focused on demonstrating coding skills:  
- Features are limited to core functionalities (no advanced reporting or room service ordering).  
- SQLite is used instead of MySQL to simplify setup for reviewers.  
- No deployment or scalability considerations, as the goal is to showcase code quality.  
- Testing is manual, with sample test cases to verify functionality (e.g., booking a room updates status).

# 6. Deliverables

- Source Code: Hosted on GitHub (github.com/NikoloziGagua/guestly\_hotel) with a README for setup instructions.  
- Documentation: This description and inline code comments explaining key logic.  
- Demo: Localhost-run application showcasing dashboards and workflows for all roles.

# 7. Evaluation Criteria

Reviewers can assess the MVP based on:  
- Code Quality: Clean, modular, and well-commented code following Django best practices.  
- Functionality: All features (booking, housekeeping, reporting) work as described.  
- User Experience: Intuitive, responsive interfaces for all user roles.  
- Security: Proper implementation of role-based access control.  
- Documentation: Clear setup instructions and explanations of key components.

# 8. Next Steps for Review

To explore Guestly:  
- Clone the GitHub repository and follow the README to run locally.  
- Test the application by creating users for each role and navigating their dashboards.  
- Review the code for structure, comments, and adherence to Django conventions.  
- Provide feedback on functionality, design, or areas for improvement.