**Software Requirements Specification (SRS)  
For Smart Parking Management System**

# 1. Introduction

## 1.1 Purpose

This document specifies the requirements for the Smart Parking Management System, a web-based platform developed using PHP, MySQL, HTML, CSS, and JavaScript. The system allows users to search, book, and cancel parking slots in real time, with integrated secure payments and admin management.

## 1.2 Scope

The Smart Parking System improves urban parking efficiency by digitizing slot allocation and payments. It supports:  
- Real-time visualization of available, booked, and occupied slots  
- User registration, login, and dashboards  
- Online booking and cancellation with automated refunds  
- Secure payments (bKash, Nagad, Cards)  
- Booking history and receipt downloads  
- Admin panel for slot, user, and transaction management  
  
Target users: Drivers, Admins, Parking Managers.

## 1.3 Definitions

Slot Status: Available (Green), Booked (Yellow), Occupied (Red)  
Refund Policy: Full refund (>24h cancellation), partial refund (<24h), no refund (<1h)  
Role-based Access: Different privileges for admin, employee, warehouse manager, and consumers.

## 1.4 References

Payment Gateway APIs (bKash, Nagad, Stripe)  
GDPR & Local Data Privacy Policies  
PHP/MySQL Documentation

# 2. Overall Description

## 2.1 Product Features

- Authentication: Registration, login, role-based access  
- Slot Booking: Search by location, time, and vehicle type  
- Cancellation & Refunds: Automated policies for users/admins  
- Dashboard: Active bookings, history, cancellation, receipts  
- Admin Panel: Slot management, reports, booking overrides  
- Payment Processing: Multiple gateways with refund API integration

## 2.2 User Classes & Roles

Consumer/Driver: Book slots, cancel, view booking history  
Admin: Manage users, slots, and bookings; generate reports  
Employee: Assist in slot allocation and monitoring  
Warehouse Manager: Manage storage and parking logistics

## 2.3 Operating Environment

Frontend: HTML, CSS, JS (responsive design)  
Backend: PHP 8.x, MySQL 8.x  
Hosting: Apache/XAMPP/WAMP, scalable cloud servers  
Browsers: Chrome, Firefox, Edge

# 3. Specific Requirements

## 3.1 Functional Requirements

1. Authentication  
- Register with name, email, phone, role, password  
- Login with email/password validation  
- Password recovery via email/SMS  
  
2. Slot Search & Booking  
- Search slots by location, date, entry/exit time  
- Display slot status dynamically  
- Book slot with confirmation email/SMS  
  
3. Booking Cancellation  
- User cancellation via dashboard  
- Refund automation based on policy  
- Admin override with optional refund  
  
4. Booking History  
- View active/past bookings  
- Download receipts in PDF/CSV  
- Admin view: filter by user, slot, or date  
  
5. Payment Integration  
- Gateways: bKash, Nagad, Stripe/Cards  
- Secure refund to original method  
  
6. Admin Controls  
- Add/edit/delete parking slots  
- Manage users and roles  
- Generate reports and logs

## 3.2 Non-Functional Requirements

Security: End-to-end encryption (SSL/TLS), role-based access  
Performance: Booking/cancellation processed <3s  
Usability: WCAG 2.1 compliant; mobile-first UI  
Reliability: 99.9% uptime, auto database backups  
Scalability: Cloud-ready, supports 10k+ users concurrently

# 4. External Interfaces

## 4.1 User Interfaces

Customer Dashboard: Bookings, cancellations, refunds  
Admin Dashboard: Manage slots, bookings, users, reports

## 4.2 Software Interfaces

Payment gateways (bKash, Nagad, Stripe)  
Notification services (SMTP, SMS API)

## 4.3 Communication Protocols

HTTPS for secure data transfer  
REST APIs for payment/refund

# 5. Other Requirements

Legal: Refund/cancellation terms must be displayed  
Maintenance: Monthly updates, bug fixes, and patches  
Support: 24/7 helpdesk (email, chat, phone)

# 6. Appendices

Use Case Diagrams:  
- Customer: Search → Book → Cancel → Refund → History  
- Admin: Manage Slots → Manage Users → Reports  
  
ER Diagram (from project code): Users, Slots, Bookings, Payments tables with relationships  
  
Sequence Diagram:  
- Booking workflow & cancellation workflow

# Conclusion

This Software Requirements Specification for the Smart Parking System defines a clear framework for building a robust, scalable, and user-friendly web-based solution for parking management. The document ensures alignment among developers, stakeholders, and end-users while enabling effective design, implementation, and evaluation of the system. Through secure, real-time booking and cancellation flows, and a powerful admin backend, the solution aims to reduce congestion and improve urban mobility.

**1. Requirement Elicitation Document**

This section describes the methods and outcomes used to gather requirements for the Smart Parking System.

**Techniques Used:**

* **Interviews**: Conducted with drivers, parking staff, and administrators to identify common pain points and desired features.
* **Surveys**: Distributed to urban commuters to gauge interest and usability expectations.
* **Observation**: Analyzed manual parking operations in busy zones to find inefficiencies in space allocation and time tracking.
* **Competitive Analysis**: Studied existing smart parking apps and identified missing or flawed features such as refund mechanisms and slot availability accuracy.

**Requirements Discovered:**

* Live status updates of parking slots (Available, Booked, Occupied)
* Real-time booking with confirmation and receipt generation
* Flexible cancellation with automated refund calculation
* Payment options compatible with local services (bKash, Nagad)
* Admin access for slot management, reports, and booking overrides

**2. Requirement Analysis & Negotiation Document**

This section outlines how collected requirements were refined, prioritized, and aligned with technical and business feasibility.

**Requirement Grouping:**

* **Functional**: Booking, cancellation, dashboard access, admin control
* **Non-Functional**: Usability, performance, scalability, data protection

**Conflicts and Resolutions:**

* **Issue**: Customers wanted instant cancellations; admins required operational notice.
* **Resolution**: Refund policies were categorized by cancellation time (e.g., 100% if >24hrs, 50% if <24hrs, no refund <1hr).

**Prioritization Matrix:**

|  |  |
| --- | --- |
| **Priority** | **Requirement** |
| High | Real-time booking, secure payments |
| Medium | Refund automation, audit logs |
| Low | Localization, accessibility features |

**Feasibility Review:**

* **Technical**: Implemented using PHP and MySQL, which support dynamic databases and session control.
* **Operational**: Integrates with popular mobile payments, aligned with urban policies.
* **Financial**: No licensing costs due to open-source stack; minimal infrastructure needed initially.