**Research Report: Smart Parking System**

--- By Taksham Sharma

Jims Scector-3

**Smart Parking System (SPS) Report**

Table of Contents

|  |  |  |
| --- | --- | --- |
| **SNO.** | **Topic** | **Page No.** |
| 1 | Introduction | 3 |
| 2 | System Overview | 4 |
| 3 | System Features | 5 |
| 4 | System Requirements | 6 |
| 5 | Design and Implementation | 7 |
| 6 | Code | 8-13 |
| 7 | Testing and validation | 14-15 |
| 8 | Security and Backup | 16 |
| 9 | Testing and Deployment | 17 |
| 10 | Maintenance and Support | 18 |
| 11 | Future Enhancements | 19 |
| 12 | Conclusion | 20 |

1. **Introduction**

**Purpose of the Smart Parking System**: The Smart Parking System (SPS) is designed to efficiently manage and automate parking operations. The system aims to provide real-time updates on parking space availability, a user-friendly interface for parking reservations, and a billing system to calculate parking fees based on the duration of parking.

**Scope of the System**: The SPS will handle:

* Real-time tracking of parking space occupancy.
* User interface for reserving and freeing parking slots.
* A billing module for calculating parking fees.
* Administrative tools for managing parking spaces and generating reports.

1. **System Overview**

**System Architecture**: The system is based on a standalone architecture where the application runs on individual machines. The system interacts with the users through a console-based interface and manages the parking data locally.

**Functional Components**:

* **Parking Space Detection**: Tracks and updates the availability of parking spaces.
* **User Interface**: Provides a console-based interaction for users to view and manage parking reservations.
* **Billing System**: Calculates parking fees based on the time of occupancy.
* **Admin Panel**: Allows administrators to manage parking spaces and view system statistics.

1. **System Features**

**Parking Space Detection**:

* Tracks the occupancy of parking spaces using sensors or simulated data.
* Updates the availability status of parking slots in real-time.

**User Interface**:

* Console-based interface allowing users to view available parking spots.
* Options for reserving and freeing parking slots.

**Billing System**:

* Calculates fees based on the duration of parking.
* Generates bills when a parking slot is freed.

**Admin Panel**:

* Displays parking statistics such as the number of occupied and available slots.
* Allows configuration of parking space settings.

1. **System Requirements**

**Hardware Requirements**:

* A computer with a modern CPU.
* Sufficient memory to handle multiple parking slots and concurrent users.

**Software Requirements**:

* A C++ development environment (e.g., Dev C++).
* Operating system capable of running C++ applications (Windows/Linux).

1. **Design and Implementation**

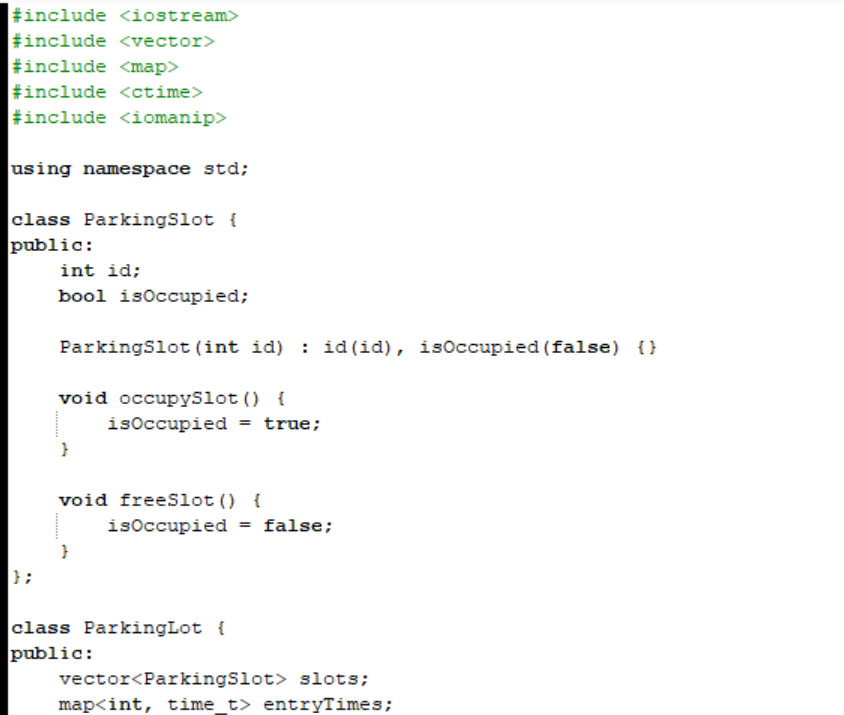
**Database Design**:

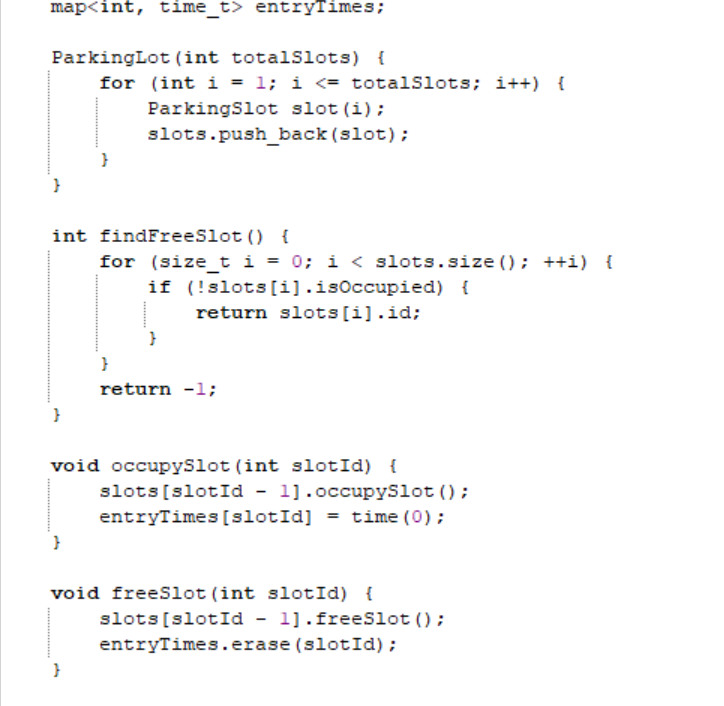
* The system uses in-memory data structures such as vectors and maps to manage parking slots and entry times.
* The parking slot occupancy and billing information are tracked and stored within the application.

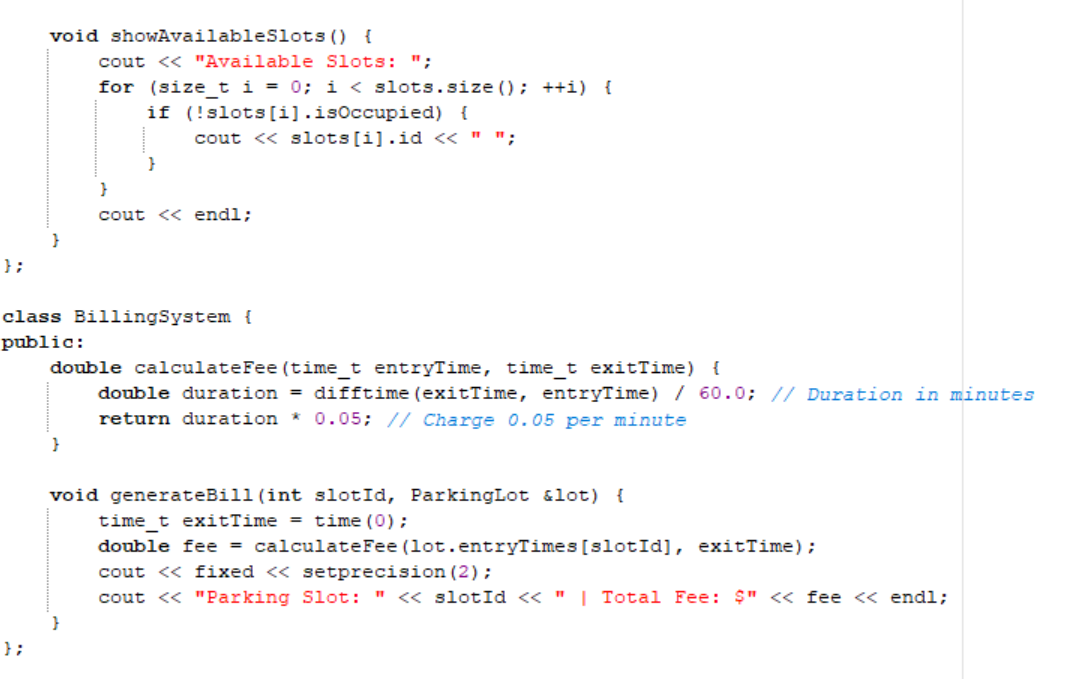
**User Interface Design**:

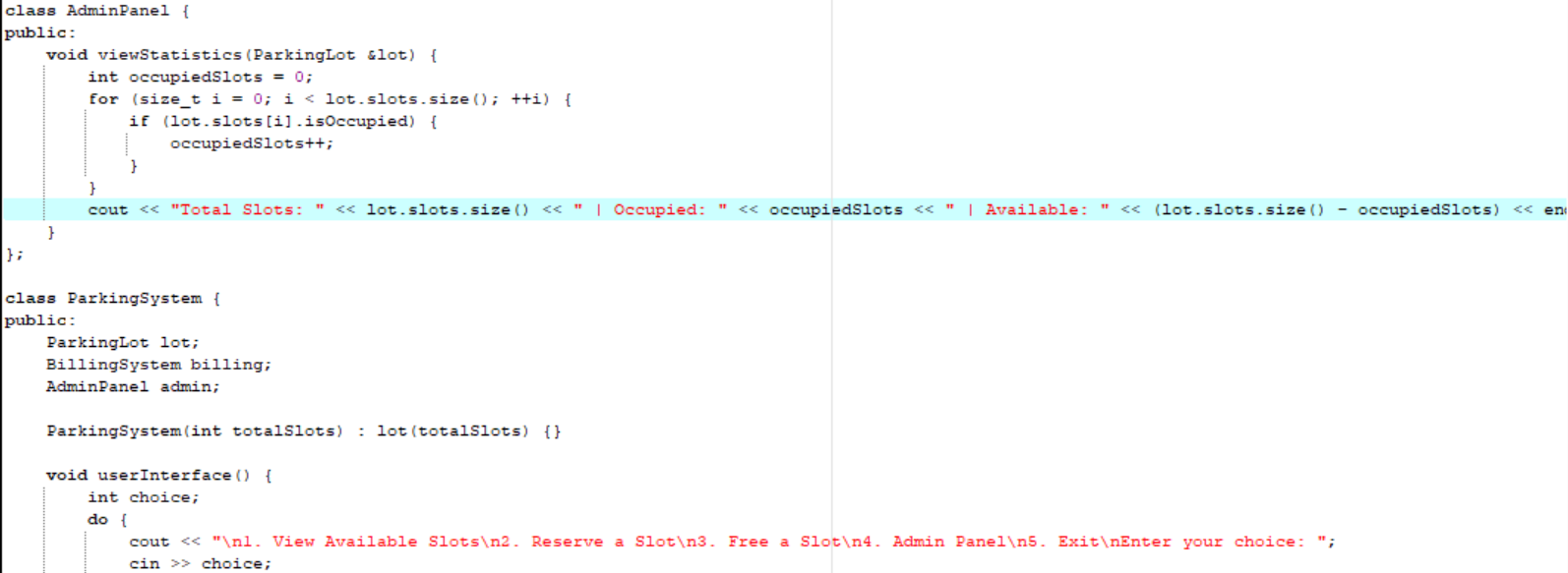
* The user interface is designed as a simple, text-based console application.
* Users can navigate through options to view, reserve, or free parking slots.

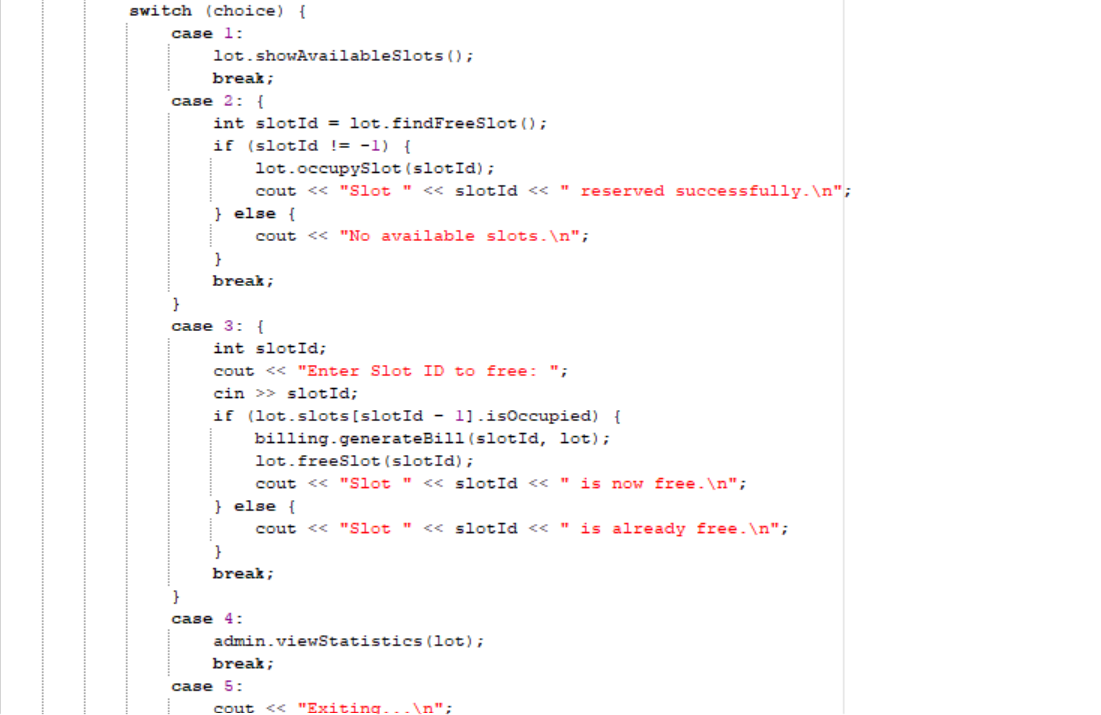
1. **Code**

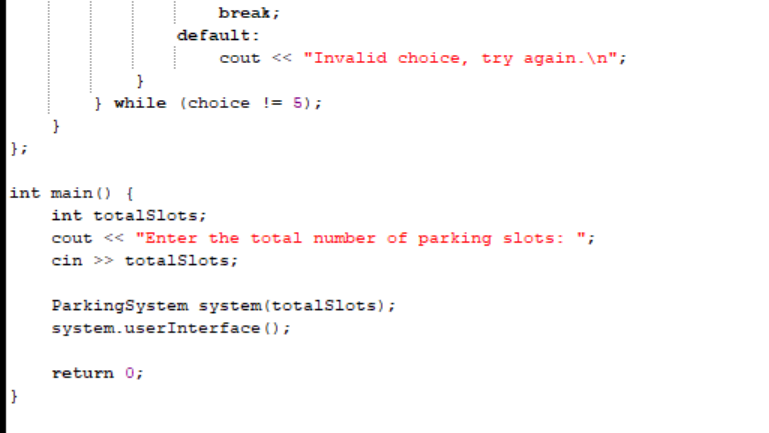
****

****

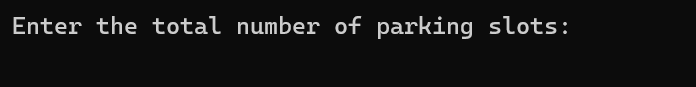




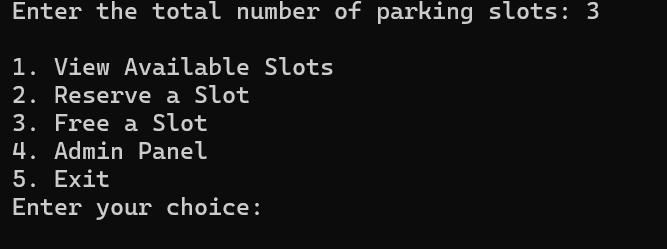




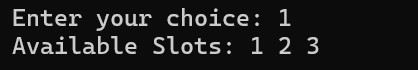
1. **Testing and Validation**



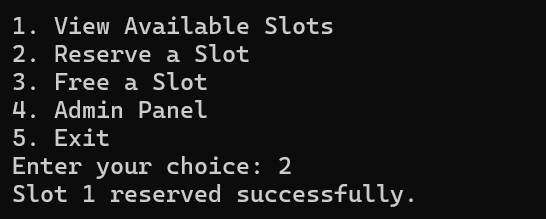
**Fig-1: Number of parking slots**

****

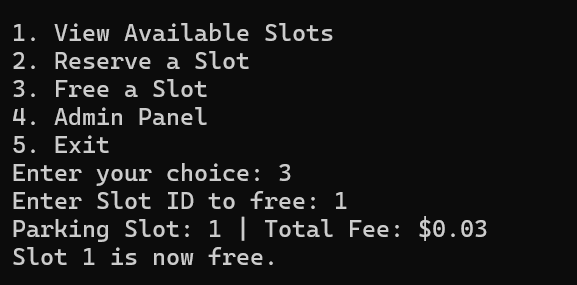
**Fig-2: Menu**

****

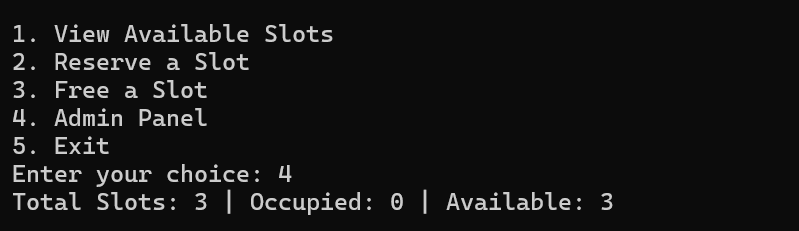
**Fig-3: Available Slots**

****

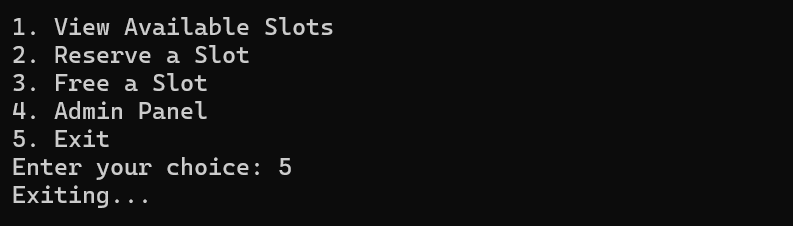
**Fig-4: Reserve a slot**

****

**Fig-5: Free a slot**

****

**Fig-6: Admin Panel**

****

**Fig-7: Exit the program**

1. **Security and Backup**

**Security Measures**:

* Access to the admin panel is restricted to prevent unauthorized changes.
* Proper validation of user inputs to avoid potential errors or misuse.

**Backup and Recovery Plans**:

* Periodic backups of in-memory data to files (if extended in future versions).
* Recovery mechanisms to restore the state of the system in case of failures.

**9. Testing and Deployment**

**Testing Strategies**:

* **Unit Testing**: Individual components were tested in isolation.
* **Integration Testing**: Combined tests to ensure all parts of the system interact correctly.

**Deployment Plan**:

* The system is deployed as a console application that can be run on any machine with a C++ runtime environment.
* Initial deployment in a test environment followed by gradual rollout to users.

**10.Maintenance and Support**

**Maintenance Plans**:

* Regular updates to improve system performance and add new features.
* Monitoring for bugs and issues reported by users.

**Support Structure**:

* Documentation provided for installation and use.
* A helpdesk system for addressing user queries and issues.

**11.Future Enhancements**

* Integration with IoT sensors for real-time parking space detection.
* Development of a graphical user interface (GUI) to enhance user experience.
* Implementation of a mobile application for remote parking slot management.

**12.Conclusion**

The Smart Parking System is designed to streamline parking operations, enhance user experience, and provide efficient management of parking spaces. The system's modular design allows for easy maintenance and future enhancements, ensuring it remains a valuable tool for modern parking management.