

**Smart Parking System**

**Software Engineering Project**

Presented by:

1-Mahmoud Taher: 19100361

2- Amr Eldesouky: 221001421

3-Abd El-Rahman Walid Maamoun: 211000096

4-Fady Zarif Deep: 211002035

5- Ahmed Amin: 221000043

**Under the supervision of:**

Dr. Ahmed Fathy El Nokrashy

Eng. Rahma Mohamed

**Contents**

[1. Document Information](file:///C:\Users\omaro\OneDrive\Desktop\UNI\SPRING_24'\CSCI_313\Project\References\hotel\Hotel%20mangment%20systemsoftware%20last%20edtion.docx#_Toc1252611119)

[1.2. Scope of the Project](file:///C:\Users\omaro\OneDrive\Desktop\UNI\SPRING_24'\CSCI_313\Project\References\hotel\Hotel%20mangment%20systemsoftware%20last%20edtion.docx#_Toc799101122)

[1.3. Technologies Used](file:///C:\Users\omaro\OneDrive\Desktop\UNI\SPRING_24'\CSCI_313\Project\References\hotel\Hotel%20mangment%20systemsoftware%20last%20edtion.docx#_Toc338204360)

[1.4. Intended Audience](file:///C:\Users\omaro\OneDrive\Desktop\UNI\SPRING_24'\CSCI_313\Project\References\hotel\Hotel%20mangment%20systemsoftware%20last%20edtion.docx#_Toc1025786163)

[1.5. Overview of the Document](file:///C:\Users\omaro\OneDrive\Desktop\UNI\SPRING_24'\CSCI_313\Project\References\hotel\Hotel%20mangment%20systemsoftware%20last%20edtion.docx#_Toc199484491)

**1. Document Information**

**1.1. Purpose of the Document**

This Software Requirements Specification (SRS) document outlines the functionalities, features, and technical details of Smart E-Parking. Its purpose is to:

* Clearly define the project's goals and objectives, outlining desired outcomes such as enhanced parking efficiency, improved user experience, optimized space utilization, and increased security.
* Specify the software's functionalities and features, encompassing user management, real-time parking space availability monitoring, automated payment processing, and advanced analytics.
* Provide a detailed roadmap to guide the development team, delineating key phases including planning, design, development, testing, deployment, and maintenance.

This SRS document serves as a foundational reference, ensuring that all stakeholders share a common vision and expectations for the successful implementation of the Smart E-Parking System.

**1.2. Scope of the Project**

The E-parking system is a web-based application designed to simplify parking management and improve drivers' user experience. It provides features such as real-time parking slot availability, online booking, and secure payment options. By leveraging technology, this system aims to reduce the time spent searching for parking, minimize traffic congestion, and streamline the administration of parking spaces. The project is focused on creating a user-friendly, efficient, and scalable solution for modern parking challenges.

**1.3. Overview of the Document**

The subsequent sections of this document detail the system’s functionalities, design, and implementation. It covers the overall description, functional and non-functional requirements, system architecture, user roles, and test cases**.**

**2. Overall Description**

**2.1. General Overview**

Our E-Parking system aims to provide flexible and straightforward solutions for users. Through the system, we integrate locating and mapping solutions through TomTom or Google Maps APIs to provide a seamless and familiar user experience. Additionally, we ensure that the UI/UX is intuitive and user-friendly to make the service accessible to all.

The core functionality of our E-Parking system is to enable IoT-driven parking solutions for day-to-day needs.

**2.1.1. User Roles and Functions**

**The system categorizes users into three main roles:**

1. **Drivers:**
   * Access vacant spots.
   * Utilize the reservation system.
   * Make payments and receive real-time updates.
2. **Parking Attendants:**
   * Monitor expired meters and parking violations.
   * Access live congestion data for better management**.**
3. **Admins:**
   * Oversee system performance.
   * Handle user-submitted reports.
   * Ensure seamless service delivery.

**2.1.2. Constraints**

* API Downtime: Dependencies on external APIs like mapping and payment services may impact functionality.
* Peak Hour Load: High traffic during rush hours may stress system resources, potentially leading to slower responses.

**3. Functional Requirements**

1. **Parking Spot Reservation:**
   * Users can reserve a parking spot in advance to ensure availability upon arrival**.**
2. **Real-Time Parking Availability:**
   * Users can view live parking availability across different areas or lots.
3. **Payment Integration:**
   * Secure payment options are available to finalize reservations seamlessly.
4. **User Authentication:**
   * Secure login methods ensure the privacy of user information and transactions.
5. **Notifications and Alerts:**

Users receive reminders about reservations, changes, cancellations, or spot unavailability.

1. **Reporting:**
   * Users access insights about parking history, expenses, and reservation frequency.

**4. Non-Functional Requirements**

1. **Performance:**
   * The system must respond quickly to parking searches, reservations, and payments.
2. **Security:**
   * Ensure data safety with encryption and compliance with regulations like GDPR.
3. **Usability:**
   * The interface should be straightforward, requiring minimal effort to navigate.
4. **Reliability:**
   * Minimize downtimes to maintain user trust and dependability.
5. **Scalability:**
   * The architecture should support increasing users and parking spaces efficiently.
6. **Maintainability:**
   * Provide organized code and documentation for easy updates and quick fixes.

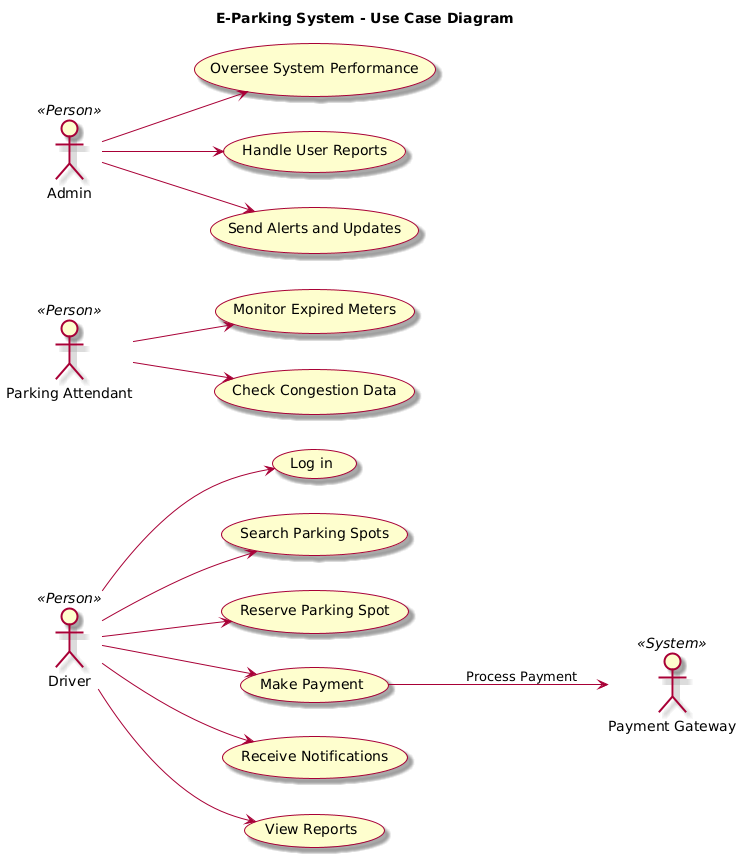
**5. System Design**

**5.1. Architecture Overview**

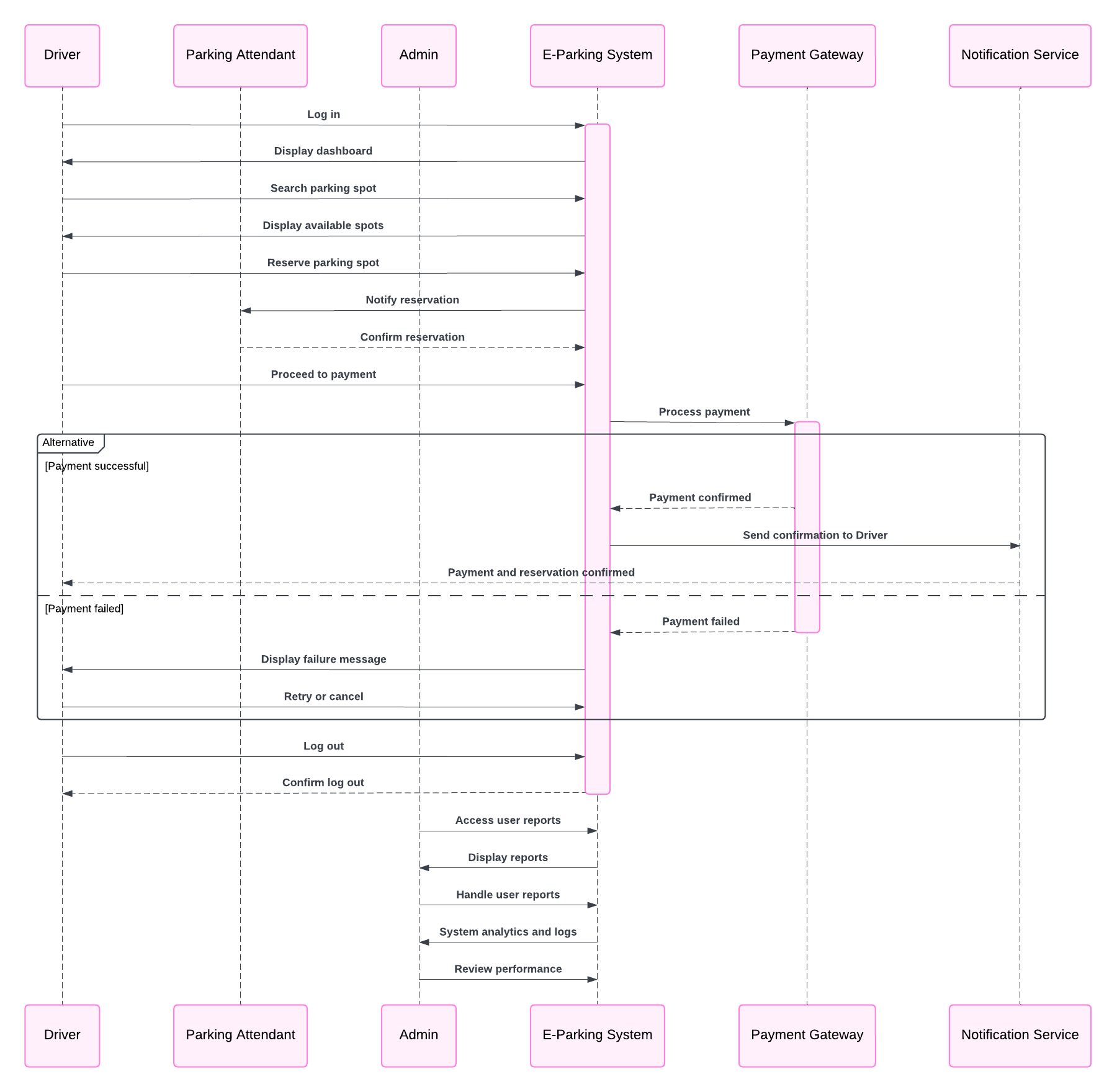
**The Smart E-Parking system follows a client-server architecture with the following components:**

* **Frontend:** A web-based user interface built with HTML, CSS, and JavaScript.
* **Backend**: A Java-based server handling business logic and communication**.**
* **Database:** MySQL database for storing user data, reservations, and reports.
* **Third-Party Services**: Integration with APIs for mapping (Google Maps) and payment gateways.

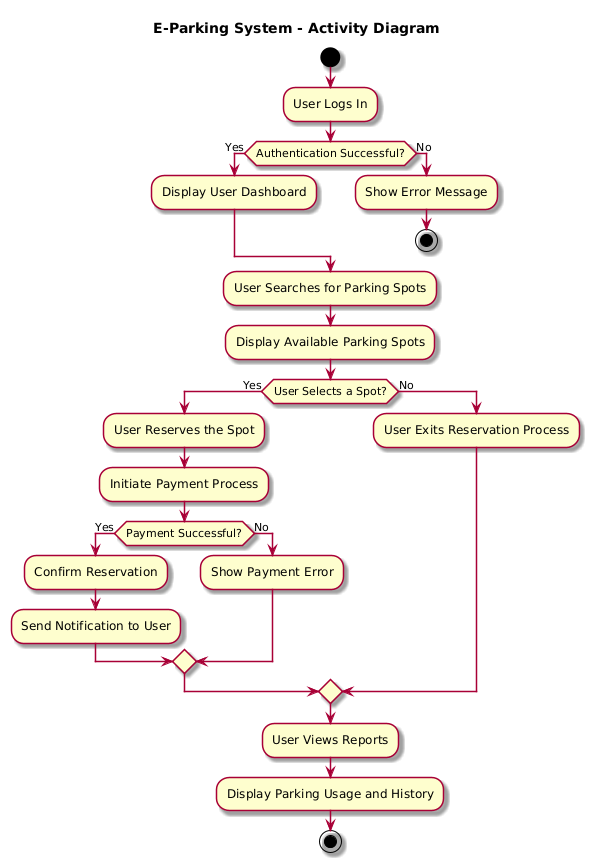
**5.2. Use Case Diagram**

****

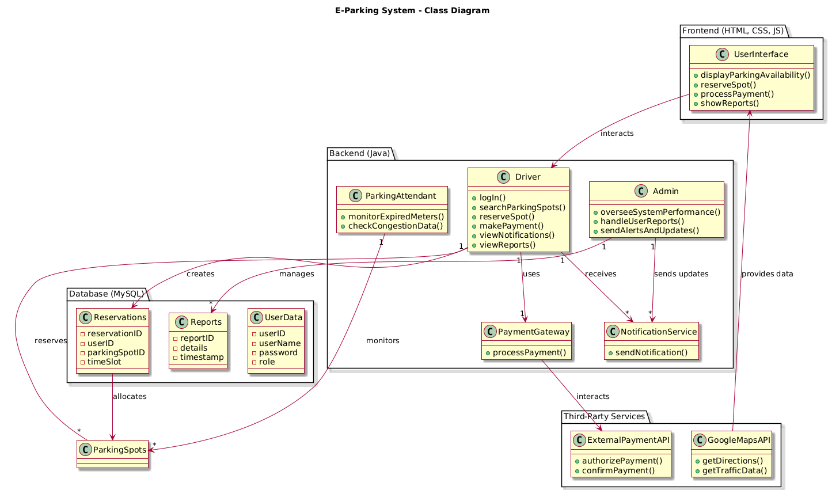
**5.3. Sequence Diagrams**

****

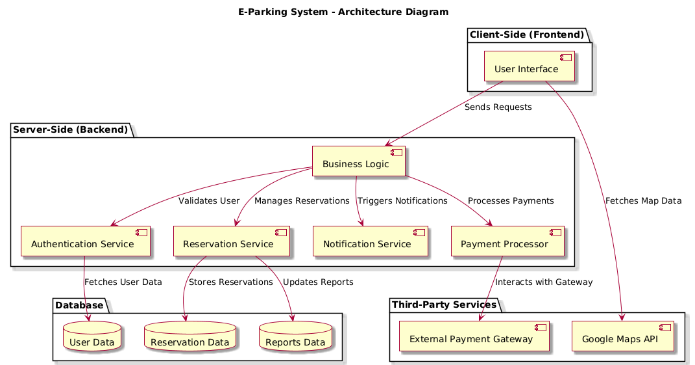
**5.4 Activity Diagram**

****

**5.5 Class Diagram**

****

**5.6 Architecture Diagram**

****

**6. Test Cases**

**6.1. Test Case 1: User Login**

* Description: Verify that users can log in with valid credentials.
* Steps:
  1. Open the login page.
  2. Enter a valid username and password.
  3. Click the "Login" button.
* Expected Result: The user is redirected to the dashboard**.**

**6.2. Test Case 2: Parking Spot Reservation**

* Description: Ensure users can reserve a parking spot successfully.
* Steps:
  1. Log in to the system.
  2. Search for available spots.
  3. Select a spot and proceed to payment.
  4. Confirm reservation.
* Expected Result: Reservation is completed, and the user receives confirmation.

**7. Deployment Plan**

1. Development Environment Setup:
   * Tools: Visual Studio Code, MySQL Workbench, Java SDK.
   * APIs: Google Maps, Payment Gateway.
2. Testing:
   * Conduct unit testing, integration testing, and performance testing.
3. Deployment:
   * Host the system on a cloud platform (e.g., AWS, Azure).
4. Maintenance:
   * Regularupdates to improve performance and add features**.**

**Dashboard:**

A screenshot of a computer

Description automatically generated

**Log-In:**

A screenshot of a computer

Description automatically generated

.

**Sign-up:**

A screenshot of a computer

Description automatically generated

**Parking Slots:**

A screenshot of a computer

Description automatically generated

**Back-end:**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Dataset:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated