

Parking Reservation System

Yassine BLALI

December 7, 2023

Abstract

This document presents a comprehensive exploration of the parking reservation system, addressing its contextual background, problem-solving objectives, implemented solutions, encountered challenges, and supplementary details provided in the appendices.

Contents

1	Project Overview	3
1.1	Context of the Project	3
1.2	Problems Addressed	3
1.3	Solutions Implemented	3
1.4	Encountered Challenges	3
2	Project Heart	5
2.1	Methodology	5
2.1.1	Technology Stack Selection	5
2.1.2	Development and Implementation	5
2.1.3	Testing and Quality Assurance	5
2.2	Implementation Details	5
2.2.1	register.php file	5
2.2.2	login.php file	6
2.2.3	make reservation.html file	6
2.2.4	process reservation.php file	7
2.2.5	view reservation.php	8
2.2.6	cancel reservation.php	9
2.3	Database Structure	9
2.4	Results and Achievements	10
2.4.1	Successful System Development	10
2.4.2	User Engagement and Feedback Incorporation	10
2.4.3	Deep Understanding and Implementation of Logic into Web Interface	10
2.5	Future Enhancements	11
2.5.1	Integration of Dynamic Space Allocation Algorithms	11
2.5.2	Incorporation of Machine Learning for Predictive Analytics	11
3	Appendices	11
3.1	Code Samples	11
3.2	Supplementary Simulations	12

1 Project Overview

1.1 Context of the Project

Have you ever driven into a busy city like Casablanca, Paris, or London and struggled to find a parking spot? The frustration of circling blocks, only to discover no available parking, is a common experience for many drivers. In bustling urban areas, this challenge of finding parking spaces often leads to wasted time, increased stress, and traffic congestion. The frequent scenario of driving around without finding a spot not only disrupts plans but also impacts the overall driving experience, causing inconvenience and delays. This project aims to tackle this widespread issue by addressing the scarcity of parking spaces in cities, aiming to improve the overall experience for drivers by offering a solution to the persistent problem of parking availability in urban settings.

1.2 Problems Addressed

My project aims to resolve several persistent challenges associated with parking in urban areas. The primary issue revolves around the difficulty individuals face in finding parking spots amidst densely populated cities. This struggle not only leads to increased stress and frustration for drivers but also contributes to heightened traffic congestion. Additionally, the inefficient utilization of parking spaces exacerbates these issues. I seek to address these problems by simplifying the parking process, making it user-friendly for individuals to locate and reserve spots in advance. By optimizing the utilization of these spaces through a fair pricing structure and encouraging pre-booking, my project endeavors to reduce urban congestion, enhance user convenience, and ultimately improve the overall urban mobility experience.

1.3 Solutions Implemented

To address parking challenges, I've incorporated several solutions in my project. I designed an intuitive reservation system allowing users to easily find and reserve parking spaces via web interface for enhanced convenience. Additionally, a dynamic pricing model optimizes parking space utilization based on factors like duration, demand, and location, promoting efficient spot usage. The system includes a feature allowing users to view their reservations, providing transparency and easy access to booked spots. Alongside, a reservation canceling feature offers users flexibility in modifying their bookings. Algorithms ensure accurate bookings, timely spot availability, and a dependable reservation process. Encouraging pre-booking, the project aims to reduce traffic congestion, enhancing urban mobility and promoting smoother traffic flow within cities.

1.4 Encountered Challenges

A pivotal obstacle was selecting an appropriate programming language . The language had to effectively manage the database housing user and reservation

data while presenting a user-friendly interface for hassle-free reservations. At the project's outset, I relied solely on Python, utilizing text files for data storage and Python's 'tkinter' library to construct the system's user interface.

Simultaneously, crafting an intuitive reservation system tailored to diverse user preferences emerged as a significant deliberation, entailing a seamless booking experience. The intricate task of integrating a dynamic pricing model demanded thorough considerations of demand, duration, and peak days to strike a fair balance between optimal spot utilization and pricing. Ensuring the accuracy and reliability of system algorithms for bookings and cancellations added another layer of complexity, necessitating meticulous planning, continuous testing, and iterative enhancements to attain a robust and user-centric reservation system.

Furthermore, ensuring valid data entry and managing user conduct within the system posed distinct challenges. Implementing stringent validation protocols to ensure future-dated reservations and proper quitting times, coupled with addressing scenarios where users failed to vacate spots at designated times, necessitated the integration of robust validation mechanisms and automated alert systems. Overcoming these complexities entailed meticulous design and development, encompassing strict data validation measures and automated notifications to ensure adherence to reservation timelines.

Beyond technical complexities, managing multiple concurrent commitments outside the project emerged as a non-technical hurdle. Juggling diverse tasks and allocating dedicated time amidst other projects and assignments demanded effective time management strategies. Balancing these responsibilities required meticulous planning and structured schedules to maintain steady progress in developing the parking reservation system.

2 Project Heart

2.1 Methodology

2.1.1 Technology Stack Selection

- Primary Language - PHP: I have chosen PHP for its robustness and compatibility with web development, facilitating server-side scripting.
- Front-End Technologies: Employed HTML, CSS, and JavaScript for front-end development, creating an interactive and visually appealing user interface.
- Database Management - MySQL: Selected MySQL for its reliability and efficiency in handling complex queries and large datasets.
- Server and Client-Side Scripting: Used PHP for server-side logic and JavaScript for client-side functionalities, ensuring a dynamic and responsive system.

2.1.2 Development and Implementation

- User Registration and Authentication: Implemented secure user registration and login functionalities using PHP and MySQL for database interactions.
- Reservation Logic: Developed logic for searching, selecting, and reserving parking spots, handling real-time availability checks.
- Data Management: Established robust data handling mechanisms for storing and retrieving user data and reservation details in the MySQL database.

2.1.3 Testing and Quality Assurance

- Testing Strategies: Executed comprehensive testing, including unit, integration, and user acceptance tests, to ensure system reliability and performance.
- Iterative Development: Continuously refined the system based on test results, addressing bugs and enhancing functionalities.

2.2 Implementation Details

2.2.1 register.php file

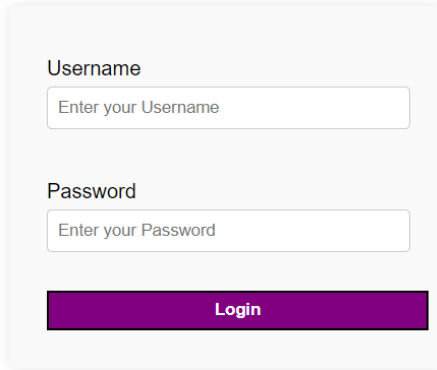
This PHP script serves as a user registration system. It verifies if a username is already in use within the database. If the username is unique, the script proceeds to securely register the new user, hashing their password for enhanced security. In cases where the chosen username is already taken, the system notifies the

user with an error message. After a successful registration, the script redirects the user to a login page. The HTML section of the code constructs the signup form interface. The inclusion of password hashing significantly improves the security aspect of the script.

2.2.2 login.php file

This PHP script is a login system that checks submitted usernames and passwords against a database. If a username exists, it verifies the password using password hashing. On successful login, it starts a session and redirects to the home page. If the login fails, it displays an error message. The HTML part creates a login form with alerts for login success or failure. The script enhances security through hashed password verification.

Login Page



The image shows a login form titled "Login Page". It contains two input fields: "Username" with a placeholder "Enter your Username" and "Password" with a placeholder "Enter your Password". Below these fields is a purple button labeled "Login". The form is enclosed in a light gray rounded rectangle.

Figure 1: Login Page

2.2.3 make reservation.html file

This HTML document features a reservation form that allows users to input spot number, entry and quitting dates and times. It uses JavaScript for client-side validation, ensuring:

1. The entry date is in the future.
2. The quitting date is after the entry date.
3. On the same day, the quitting time is not earlier than the entry time.

If the input data fails these checks, the script displays alert messages and prevents form submission. Successful validation leads to the form data being sent to process-reservation.php for further processing.

Make Reservation

Spot Number:

Entry Date:

mm / dd / yyyy

Entry Time:

-- : -- --

Quitting Date:

mm / dd / yyyy

Quitting Time:

-- : -- --

Reserve

Figure 2: Make-reservation page

2.2.4 process reservation.php file

In this file, I have implemented a dynamic pricing algorithm that takes into account multiple criteria such as duration, period of day, weekends, demand ...

getFreeParkingSpots function

Total number of parking spots, T , is set to 100. Threshold, θ , is defined as 20% of the total spots $\theta = T \times 0.20$.

A query counts the number of currently reserved spots. Let's denote the number of reserved spots as R .

The number of free spots, F , is calculated as $F = T - R$

The demand factor, D , initially set to 0, is used to adjust pricing based on parking spot availability.

Now if $0 < F \leq \theta$ then $D = 1 - \frac{F}{T}$

getHolidays function

This function retrieves public holiday data for a specified country and year from an external API. It constructs the request URL with

the country code and year, initializes a cURL session, sets necessary options, executes the session to fetch data, and then closes the session. The function decodes the received JSON response into a PHP array and returns it. This allows for easy access to holiday date and name from the API in the script

***SetPrice* function**

Let P be the total reservation fee, initialized to 0 and P_{hour} the normal price per hour.

Hourly Rate Calculation :

If $0 \leq \text{hour} \leq 12$ then $P += 2 \times P_{hour}$

If $12 < \text{hour} \leq 18$ then $P += 3 \times P_{hour}$

If $18 < \text{hour} < 24$ then $P += 4 \times P_{hour}$

Weekend Surcharge :

If the day is a weekend (Saturday or Sunday), $P *= 2$

Holiday Surcharge :

Holidays are fetched based on the country code and year using *getHolidays* function. If the entry date is a holiday, $P *= 2$

Demand Adjustment :

A demand rate D is calculated using *getFreeParkingSpots* function.

P is adjusted as $P = (1 + D) * P$

In summary, the function calculates the parking fee based on the time of day, whether it's a weekend, if it's a holiday, and the current parking spot demand. Each of these factors contributes to a dynamic pricing model.

The script attempts to insert the reservation details into the database. If successful, it redirects the user to a confirmation page;

2.2.5 view reservation.php

This file uses PHP to display a user's parking reservations. It initiates a session to access the user's username, connects to a MySQL database, and retrieves

Your Reservations				
Entry Date	Quitting Date	Spot Number	Reservation code	Price(MAD)
2023-12-28 23:03:00	2023-12-29 13:06:00	5		75
2023-12-30 01:04:00	2023-12-31 04:08:00	77		30

Home

Figure 3: View-reservation Page

reservation details like entry and quitting dates, spot number, price, and reservation code from the database. These details are dynamically displayed in an HTML table. If no reservations are found, a message is displayed.

2.2.6 cancel reservation.php

This PHP script allows users to cancel their parking reservations. A session is started to retrieve the user's username. The script then executes a SQL query to delete the reservation from the database where the username and spot number match. If the deletion is successful and affects the database, it sets a success flag; otherwise, it sets an invalid flag if no matching reservation is found. Messages are displayed based on the success or failure of the operation. A button is provided for navigation back to the home page.

2.3 Database Structure

I have structured the database schema for managing user data, reservation details, and parking spot availability as follows:

register table

id: primary key for this table.

username: Unique username for user login.

password: Hashed password for user authentication.

reservation table

id: Primary key, uniquely identifies each reservation.

username: Foreign key linking to the Users table.

spot_number: Parking spot number reserved.

entry_date: Date and time when the reservation starts.

quitting_date: Date and time when the reservation ends.

price: Total price for the reservation.

reservation_code: Unique code for the reservation.

id	username	entry_date	quitting_date	price	reservation_code	spot_number
1	yassine	2023-12-29 03:30:00	2023-12-29 07:30:00	20	RLN5XW	5
2	yassine	2023-12-29 03:30:00	2023-12-29 07:30:00	20	RKG8OU	45
3	Mouhi	2024-01-04 20:30:00	2024-01-05 00:30:00	40	XOA7VQ	47

Figure 4: Reservation Table

2.4 Results and Achievements

2.4.1 Successful System Development

- **Functional System:** Developed a robust parking reservation system that encompasses critical functionalities such as user registration, personalized spot selection and efficient reservation management. This comprehensive system addresses all essential aspects of parking reservation, from initial user entry to final transaction completion.
- **User-Centric Design:** Prioritized user experience by designing an intuitive and accessible interface, facilitating ease of use for diverse user groups. This design approach ensures that users can navigate the system effortlessly, make reservations quickly, and manage their bookings with ease, thereby enhancing overall user satisfaction.

2.4.2 User Engagement and Feedback Incorporation

- **User Testing and Feedback:** Conducted extensive beta testing phases with fictional users to gather insightful feedback. This user-centric approach enabled me to identify and address real-world usability challenges and preferences. By incorporating this valuable feedback, significant improvements were made in system functionality, user interface design, and overall user experience. This iterative process of testing and refinement ensures that the system is continually evolving to meet and exceed user expectations.
- **Proactive User Engagement:** Established a proactive channel for ongoing user engagement, encouraging continuous feedback and suggestions. This engagement strategy not only fosters a sense of community among users but also keeps the development team attuned to user needs and emerging trends, ensuring the system remains relevant and user-friendly.

2.4.3 Deep Understanding and Implementation of Logic into Web Interface

- **Mastered the Art of Logic-to-Web Interface Translation:** Achieved a profound understanding of converting complex logic into efficient and user-friendly web interfaces. This mastery involves not only the technical know-how of web development frameworks and languages but also a keen insight into user interaction patterns and experience design.

- **Seamless Integration of Backend Logic with Frontend Presentation:** Demonstrated exceptional skill in integrating sophisticated backend algorithms and data processing logic with intuitive frontend interfaces. This achievement highlights the ability to create web applications that are functionally robust.

2.5 Future Enhancements

2.5.1 Integration of Dynamic Space Allocation Algorithms

Sensor-Driven Spot Management: I plan to integrate advanced dynamic space allocation algorithms that are synchronized with sensors in the parking area. These sensors will detect when a car vacates its spot, updating the system in real-time. This enhancement will significantly optimize parking space utilization and provide real-time updates on spot availability.

Overstay Charges: Introduce a feature where users who do not vacate their parking spots within their reserved quitting time are automatically charged an additional fee. This system will calculate overstay charges based on the extra time the spot is occupied, ensuring fair usage and turnover of parking spaces.

2.5.2 Incorporation of Machine Learning for Predictive Analytics

Predictive Spot Allocation: Implement machine learning models to predict parking spot availability patterns. By analyzing historical data, such as peak times, average duration of stay, and frequency of visits, the system can intelligently suggest the most suitable parking spots to new users, enhancing their experience and streamlining the reservation process.

Personalized User Experience: Use machine learning algorithms to offer personalized recommendations to users based on their parking habits and preferences. The system can learn from individual user behavior and provide tailored suggestions, such as preferred spots, ideal reservation times, and even promotional offers.

3 Appendices

3.1 Code Samples

Here is the PHP script that I have implemented to enable user to check their reservations

```

<?php
session_start();
$servername = "localhost";
$user = "root";
$password = "Yassine@22@";
$dbname = "prs";

$conn = new mysqli($servername, $user, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$username = $_SESSION['username'];

// Fetch reservations from the database
$sql = "SELECT entry_date, quitting_date, spot_number, price, reservation_code FROM reservation WHERE username = '$username'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        echo "<tr><td>" . $row["entry_date"] . "</td><td>" . $row["quitting_date"] . "</td><td>" . $row["spot_number"] . "</td><td>" .
            $row["reservation_code"] . "</td><td>" . $row["price"] . "</td></tr>";
    }
} else {
    echo "<tr><td colspan='5'>No reservations found</td></tr>";
}
$conn->close();

```

Here is the function to generate all holidays for a country in a specific year, it uses Nager.Date API to do so.

```

function getHolidays($countryCode, $year) {
    $url = "https://date.nager.at/Api/v2/PublicHolidays/$year/$countryCode";

    // Initialize cURL session
    $ch = curl_init();

    // Set cURL options
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_URL, $url);

    // Execute cURL session and get the response
    $response = curl_exec($ch);
    curl_close($ch);

    // Decode JSON response
    return json_decode($response, true);
}

```

3.2 Supplementary Simulations

Invalid dates :

localhost says

Entry date should be in the future.

OK

Make Reservation

Spot Number:

Entry Date:

Entry Time:

Invalid cancellation credentials :

Sorry ! There is no such reservation under your username.

Go Back Home

Reservation overlap management :

A user "A" reservations list :

Your Reservations

Entry Date	Quitting Date	Spot Number
2024-01-04 20:30:00	2024-01-05 00:30:00	47
2023-12-30 19:47:00	2023-12-31 04:43:00	11

Now, when user "B" tries to reserve spot number 47, he get this alert :

Spot Already Reserved

Sorry, this spot has already been reserved. Please try reserving a different spot.

Available spots visualization :

