



Department of Computer Science

This project has been satisfactorily demonstrated and is of suitable form.

This project report is acceptable in partial completion of the requirements for the Master of Science degree in Computer Science.

Electronic Parking Rental System

Project Title

Padsala Chirag Gokul

Student Name

Dr. Mikhail I. Gofman

Advisor's Name

Advisor's signature

Date

Dr. Kenneth Kung

Reviewer's name

Reviewer's signature

Date

Electronic Parking Rental System



Project Report

By:

Padsala Chirag Gokul

CWID: 805613197

Spring 2018

CPSC 597

Project Advisor: Dr. Mikhail I. Gofman

Department of Computer Science

California State University, Fullerton

Abstract

The issue of parking availability is increasing day by day. Research by Adam Millard-ball, Rachel Weinberger and Robert Hampshire(SF Park, March 2016) found that cruising for parking in a 15-block business district in Los Angeles has been estimated to produce 3,600 miles of excess travel each day-equivalent to two round trips to the Moon each year. In addition to wasting people's time it also wastes 180 gallons of fuel producing more than 3500 pounds of worth of carbon emissions which pollute the environment.

This project aims to help address the above problem by increasing the parking availability. Specifically, it proposes a web-based application that allows people to rent their unused parking spots to others for a fee set by the renter. This business model is mutually beneficial to both the renter and the rentee and we believe it be economically sustainable. This report gives an overview of the proposed system, gives the system design and implementation, and discusses the tests done to ensure that the system meets all key requirements.

Keywords: Parking, Renting, Cruising, Pollution, User Interface.

Table of Contents

Introduction.....	1
Preliminaries.....	1
Description of Problem.....	1
Project Objective.....	2
Development Environment.....	3
Operational Environment.....	3
Requirement Description.....	4
Functional Requirements.....	4
Non- Functional Requirements.....	5
Software Interface Requirements.....	6
Design Description.....	7
System Architecture.....	7
Three-tier Architecture.....	7
Use Case.....	8
Use Case Diagram.....	9
Activity Diagram.....	10
User Interface Description.....	13
Implementation.....	16
Technical Approach and Process.....	16
Class Diagram.....	28
Data Design.....	29
Test and Integration.....	30
Functionality Testing.....	31
Usability Testing.....	31
Interface Testing.....	32

Compatibility Testing.....	32
Performance Testing.....	32
Security Testing.....	33
Installation Instruction.....	33
Software Required.....	33
Installation Steps.....	34
Executing Project.....	34
Operating Instruction.....	34
Recommendation for Enhancement.....	35
Future Scope.....	35
Bibliography.....	36
References.....	36

List of Figures

Figure 1: 3-Tier Client-Server System Architecture.....	7
Figure 2: System Use Case Diagram.....	9
Figure 3: Activity Diagram- Register User.....	10
Figure 4: Activity Diagram- Rent Spot.....	11
Figure 5: Activity Diagram- Book a Spot.....	12
Figure 6: System Activity Diagram.....	13
Figure 7: Desktop Application Interface.....	14
Figure 8: Tablet Interface, Mobile Interface.....	15
Figure 9: User already Registered Page.....	16
Figure 10: Register Successful Page.....	17
Figure 11: Login Error Page.....	18
Figure 12: Dashboard Page.....	19
Figure 13: Edit User Profile Page.....	19

Figure 14: Duplicate Parking Error Page.....	20
Figure 15: New Parking Added Page.....	21
Figure 16: List of Spots Added and Rented Page.....	22
Figure 17: Change Parking Status Page.....	22
Figure 18: Re-renting Parking Spot Page.....	23
Figure 19: Reserve a Spot Page.....	23
Figure 20: Review and Payment Page.....	24
Figure 21: Reservation Confirmation Email.....	24
Figure 22: Spot Rented Email.....	25
Figure 23: Check Existing Reservations Page.....	25
Figure 24: Admin Login Page.....	26
Figure 25: Admin Edit User Page.....	27
Figure 26: Admin Reports Page.....	27
Figure 27: Class Diagram.....	29
Figure 28: Entity Relationship Diagram.....	30

Introduction

This project focuses on the problem of parking availability. Difficulties in finding places to park results in driver frustration, wasted time, wasted fuel, and increases the carbon footprint. This project proposes a solution that allows drivers to rent their unused parking spots for a fee set by the renter. This results in a sustainable business model that is mutually beneficial to the renters and the rentees and helps address the above problems.

The project implements the renting system as a web-based app that allows renters to set the fee and make the renting opportunities known to the rentees. The rentees can use the app to search for available spots, rent the spots, and pay the fee to the renter. The application was designed for desktop, laptop, mobile phone, and tablet platforms.

Preliminaries

Description of the Problem

Parking related problems have been gradually rising recently. It's becoming more and more difficult to find a parking spot nearby the desired location, especially in the metropolitan areas. The parking availability problem is exacerbated by increasing number of drivers(John Laporte, 2016). Booking a parking spot before your journey or long trip can save you a lot of time, gas and lowers the air pollution otherwise caused by it. On the other hand, having an extra parking spots or permits does no good to the owner until now, it's just treated as extra piece of land being added to your property value. Those extra parking spots can be used for renting thus creating more parking spots for others and extra income for the owner without doing anything.

Finding a parking can be very stressful task at times, especially in California. Finding a parking spot in desired time is nothing but the result of luck, planning or some tricky maneuvers. Drivers often burn gas and waste time while cruising to find a parking spot and yet can't find a spot at the desired location. As analysis lessons from SFPark which is a system for managing the availability of on-street parking, states that cruising for parking in a 15-block business district in Los Angeles has been estimated to produce 3,600 miles of excess travel each day-equivalent to two round trips to the Moon each year. It's not just waste of time but 180 gallons of fuel getting used for no good reason and producing more than 3500 pounds of carbon dioxide when burned, which in turn is doing nothing but polluting the environment.

Project Objectives

The main objective of this project is to completely eliminate the stress related to parking, by adding more parking options which can be used by anyone using this web application. Also you can book the spot in advance and pay using the web application itself. The user can book any parking space which is available and pay for it in advance. It would eliminate the unnecessary cruising, at the same time decreasing the pollution which could have caused otherwise. It will also benefit the user to generate extra income by just renting their private parking when unused using this web application. Below are more objectives and features for this project.

Purpose of this application

- Stop drivers cruising for parking
- Provide efficient way to organize traffic related problems
- Ease of finding parking spot
- Control pollution
- Provide secured parking
- Make parking smart and easy
- Provide extra income for private parking spots
- Increase parking spots than ever before
- Reduce the driver stress associated with finding parking

Real-time application features

- Search nearby car parking spots
- Lookup for available empty spots
- Book parking spots in advance
- Pay for reservation
- Email notification about reservation
- Rent private parking area for parking when unused

- Private parking added by user/owner can be made available/unavailable anytime with a single click
- Simple web form for third party user to add their parking spots

Other Objectives

- Easy to use User Interface
- Fast rendering on user's input
- Loosely coupled system
- Mobile/Tablet responsive User Interface

Development Environment

Software/System Requirements

Operating System: Windows 7 or above.

Client-Side Programming: HTML, CSS, Bootstrap, Javascript/JQuery, AJAX.

Server-Side Programming: ASP.NET, ADO.NET, C#.

Database: MS-SQL Server 2008 or above.

Development IDE: Microsoft Visual Studio 2012 or above.

Hardware Requirements

Processor: Intel Dual Core or higher, AMD A7 or higher.

RAM: 4 GB or more.

Disk Space: 8 GB or higher

Additional Device: Trackpad/Mouse and Keyboard or similar equivalent device.

Operational Environment

Software Requirements

Web Browser: Mozilla Firefox, Google Chrome, Microsoft Edge or similar

Hardware Requirements

Processor: 1.3GHz or higher

RAM: 2 GB or more

Disk Space: 1 GB or higher

Additional Device: Trackpad/Mouse and Keyboard or similar equivalent device.

Requirements Description

Functional Requirements

Renter Use-case

- The system must be able to register a user.
- The system must provide feedback for the new registered users.
- The system must populate City, State and Country using Google Places API.
- The system must populate State, City and Country based on Zip in less than half second.
- The system must guide new registered user to Login.
- The system must allow registered user to Login.
- The system must generate errors for unauthorized users
- The system must generate validation error while user submit Register/Login form.
- The system must redirect user to dashboard after log in.
- The system must allow user to view/edit their profile information.
- The system must add parking spot for registered user after filling form.
- The system must check for duplicate parking spots.
- The system must generate error for duplicate parking spot.
- The system must allow user to reserve a parking spot for future dates.
- The system must allow user to search parking spots by city.
- The system must allow user to search parking spots nearby them.
- The system must display all the spots added.
- The system must allow user to re-rent parking spots added previously.

- The system must allow user to pay for the spot online.
- The system must validate card information correctly.
- The system must display all the reservations made by user.
- The system must notify user by email for successful reservations
- The system must allow user to log out.

Rentee Use-case

- The system must allow user to Register.
- The system must allow user to Login.
- The system must generate validation error for web forms.
- The system must allow user to add parking spot.
- The system must populate City and State using Google Places API.
- The system must generate error for duplicate parking spot.
- The system must allow user to re-rent parking spot.
- The system must allow user to set a price.
- The system must allow user to select date and time to rent spot.
- The system must not allow user to select past dates.
- The system must not allow user to select close time before open time.
- The system must allow owner to make spot unavailable.
- The system must check pre-reservations before making spot unavailable.
- The system must notify owner if their spot is reserved by someone.
- The system must display renter's user information who rented the spot.
- The system must allow user to log out.

Non-Functional Requirements

List of Non-Functional Requirements

- User interface must be easy to operate by new users.
- System must handle operational error efficiently.
- User must be allowed to navigate easily among all the features.
- User details must be populated in less than couple seconds.

- User requests must be served appropriately with low latency throughout the application.
- List of parking spots must be provided to user within couple second of button click.
- Parking spots should be made unavailable instantly on a button click.
- Email must be sent within 5 minutes of user's successful reservation.
- Owner must be notified via email in less than 5 minutes after his/her spot is reserved.
- Parking list must be updated instantly when searched by city.
- Parking list must be populated instantly when searching near-by user's location.
- Modified parking spot details must be updated for all users within 2 seconds.
- The system should store minimal data on user's system.
- The system should be easy accessible via mobile, tablet, laptop and PC.
- The system must be accessible to user all time.
- The system should not remain down other than deployment.
- All the user and parking related data must be stored into the database in couple seconds.
- System should be able to scale vertically without any problem.
- System should be able to serve at least 25 requests per seconds without scaling application horizontally.
- System should serve parallel users request without affecting overall performance.

Software Interface Requirements

List of Software Interfaces

- AJAX must establish connection and request data from Google API within a second.
- AJAX must request geolocation data from the backend within a second.
- Back end system must be able to communicate with the database system within 20ms of a request.

- Front end interface must comply with the back-end binding request in page load event.

Design Description

System Architecture

High Level System Architecture:

The below diagram represents the 3-tier architecture of the web application. A three-tier architecture is a client-server architecture which represent presentation logic which consist of the front-end user interface components which is connected to business layer, which contains all the business logics which links the data layer and front-end components. The Data layer is a part of back end data retrieving logic which includes the SQL queries, stored procedures which helps the application to retrieve the data from the database which is MS SQL Server for this project. A clear idea can be seen in the image below.

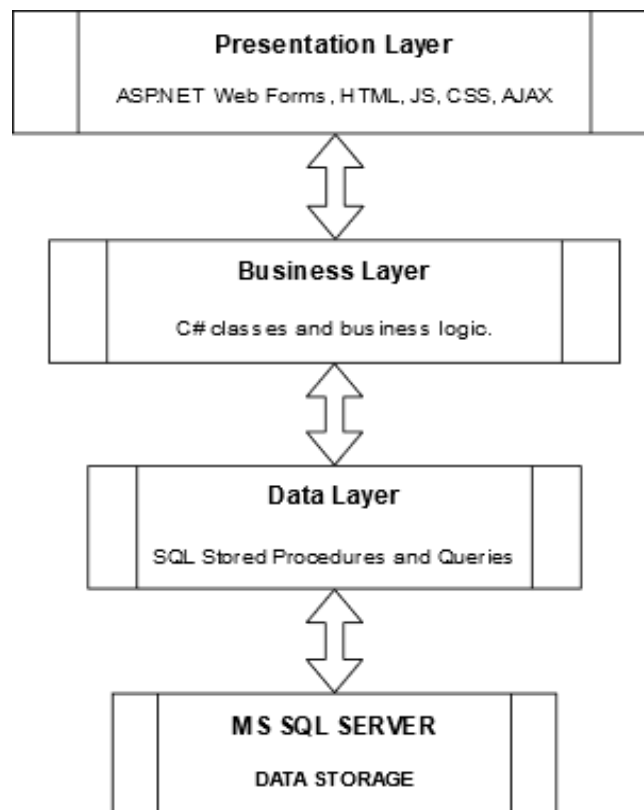


Figure 1: 3-Tier Client-Server System Architecture

Use Case

Use case is a list of events, maybe success and failure scenarios which defines the actor and system to support a goal. It describes the functional requirements. Use case must be initiated by an actor and must end with an actor. A simple use case example scenario would be: If a user request to update the parking spot information, the software should allow the user/owner and should also update it into the system and notify the user. Below is the Use Case list for this application.

Actor 1- User:

- Register account
- Login account
- View dashboard
- View/Edit profile
- Search available parking spots
- Search near-by parking spots
- Book parking spot
- Make payment for reservation
- Receive booking confirmation
- View existing reservations
- Log out account

Actor 2- Owner:

- Register account
- Login account
- Add parking spot details
- Rent parking spot
- Change availability
- Re-rent added parking spot
- Add multiple parking spots
- Receive confirmation for rented spots
- Change price

Actor 3- Admin:

- Login Account

- Edit User Data
- View Reports

Use Case Diagram

Use case diagrams are a part of Unified Modeling Diagrams generally referred to as UML. Use case diagram generally represents the way any type of user interacts with the use case of our system. It can be understood as a behavior diagram which is used to understand the set of actions which are referred as use cases that actors interacting with the system can perform. In the below diagram we can see that there are three types of actors/users- Normal User, Owner of a parking spot and Admin operating the system. All three actors can be seen interacting with the respective use case in the below diagram. The below Use Case Diagram explains the overall behavioral working of our system.

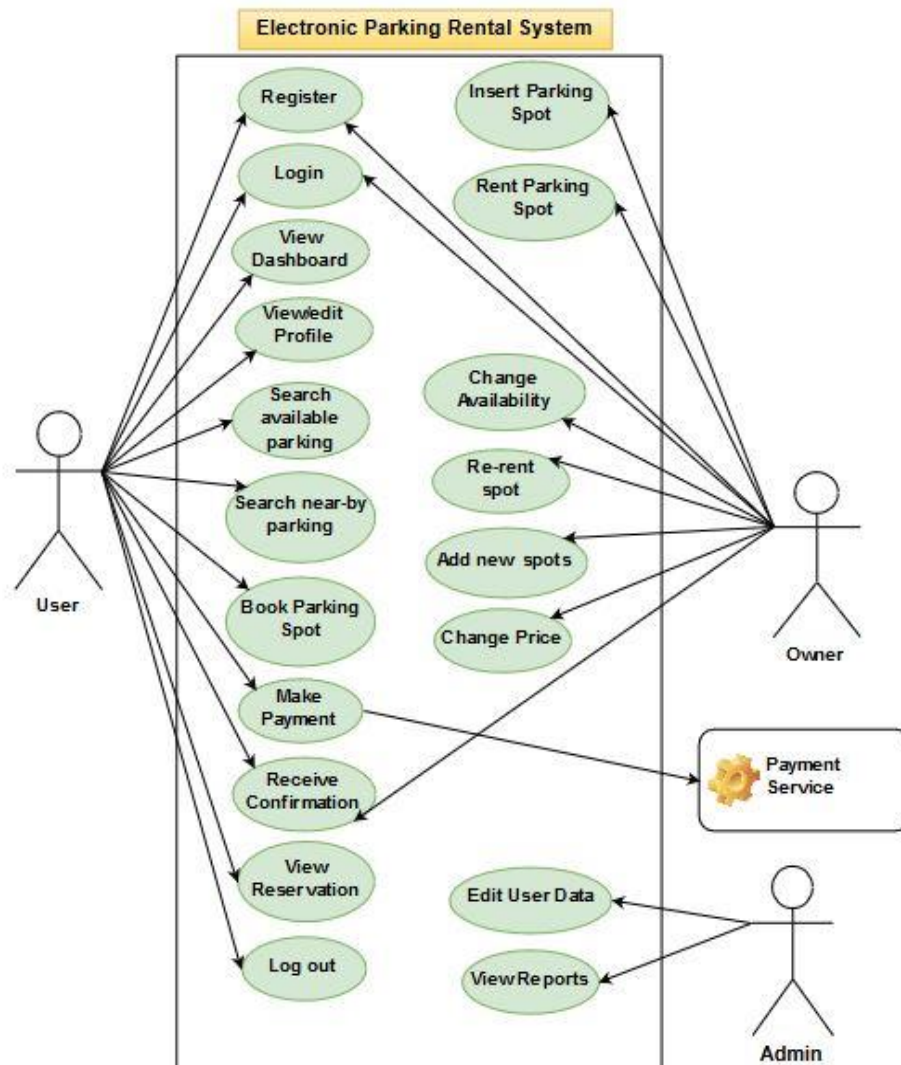


Figure 2: System Use Case Diagram

Activity Diagrams

Activity diagrams are also a part of Unified Modelling Languages (UML). Activity diagrams generally represent the flow of activities which are a part of systems. Activity diagrams can help understand multiple activities in detail with the flow of actions to achieve certain outcome. We will see multiple activity diagrams below which will help us understand all the activities taking place in our system. Below is the activity diagram which represents the new user registration activity.

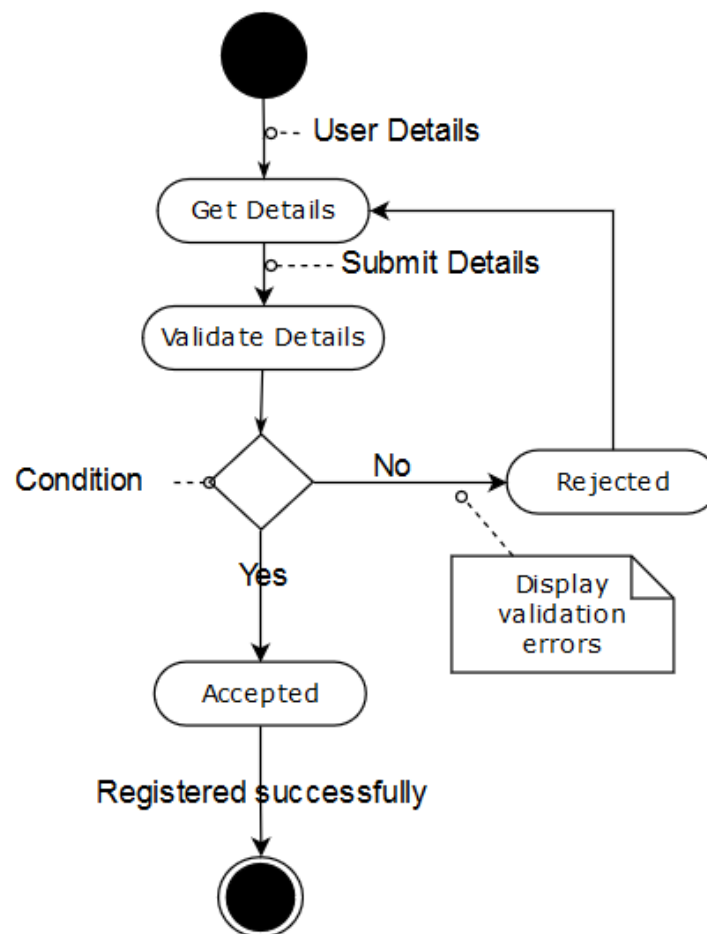


Figure 3: Activity Diagram- Register User

The similar flow fits when any user tries to login after successful registration. The user will fill in the username and password and submit the credentials which follows by some

business logic which checks if user is registered already, if so it then matches the username and password and redirect user to the dashboard, after which user can use all the features provided by the system. Below activity diagram represent the user adding new parking spot to rent.

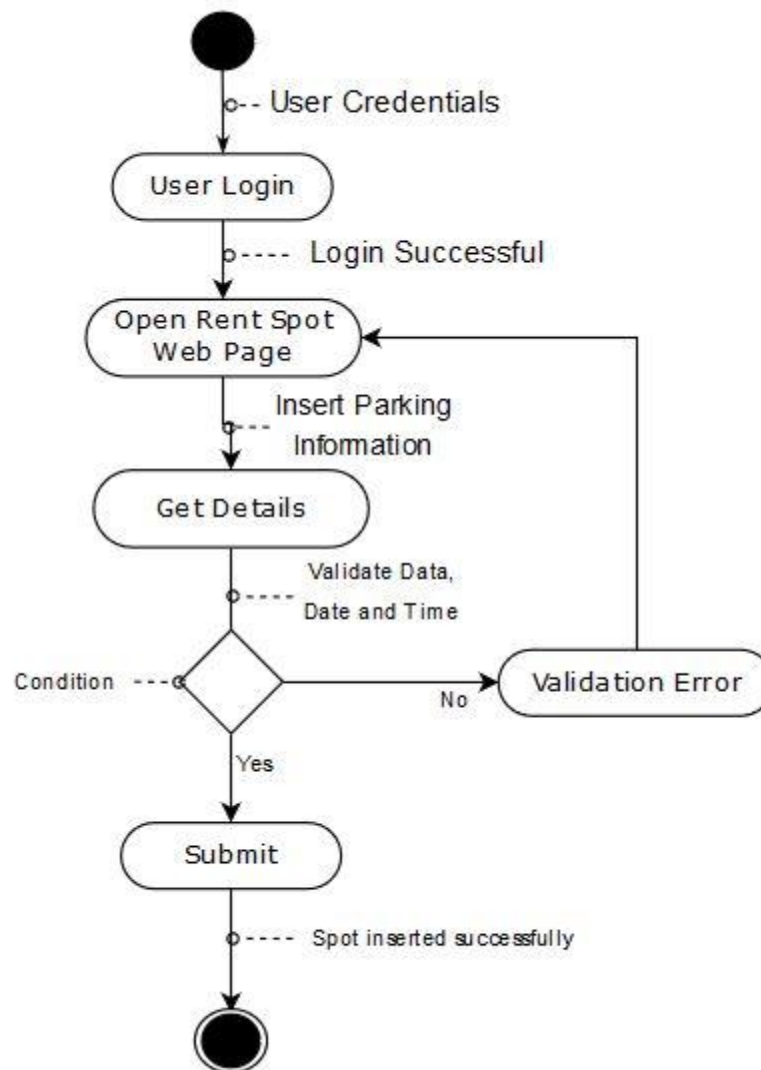


Figure 4: Activity Diagram- Rent Spot

The above diagram represents the flow of user adding new spot, which can be rented by other users. User navigates to Add Spot Web page of the application and fill a form where user is asked for address, zip, available date, start time and close time of the parking available to another user. User can set price for their parking which later is booked by another user after paying for the spot. Below activity diagram represent the user paying and booking a parking spot.

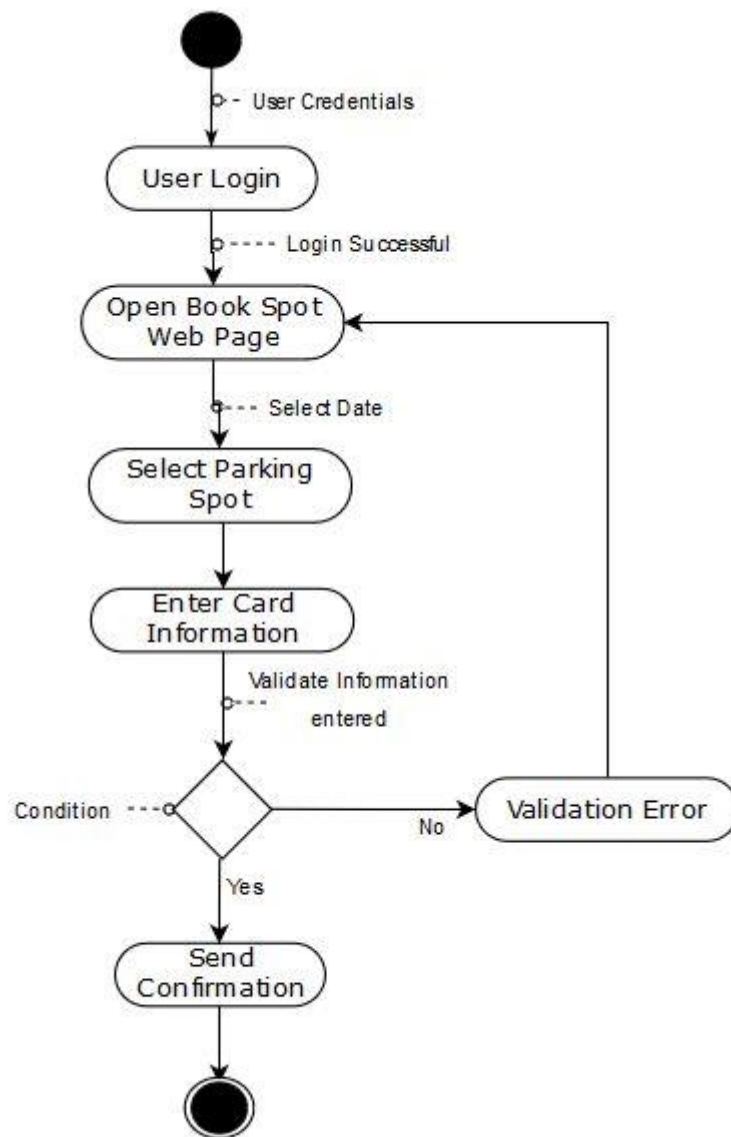


Figure 5: Activity Diagram- Book a Spot

The above diagram represents the flow of trying to book a parking spot, by paying for the price set by renter. User navigates to Book Spot Web page of the application and fill a form where user is asked for address, zip, available date, start time and close time of the parking available to another user. User can set price for their parking which later is booked by another user after paying for the spot. Similarly, the below activity diagram represents the working of the whole system.

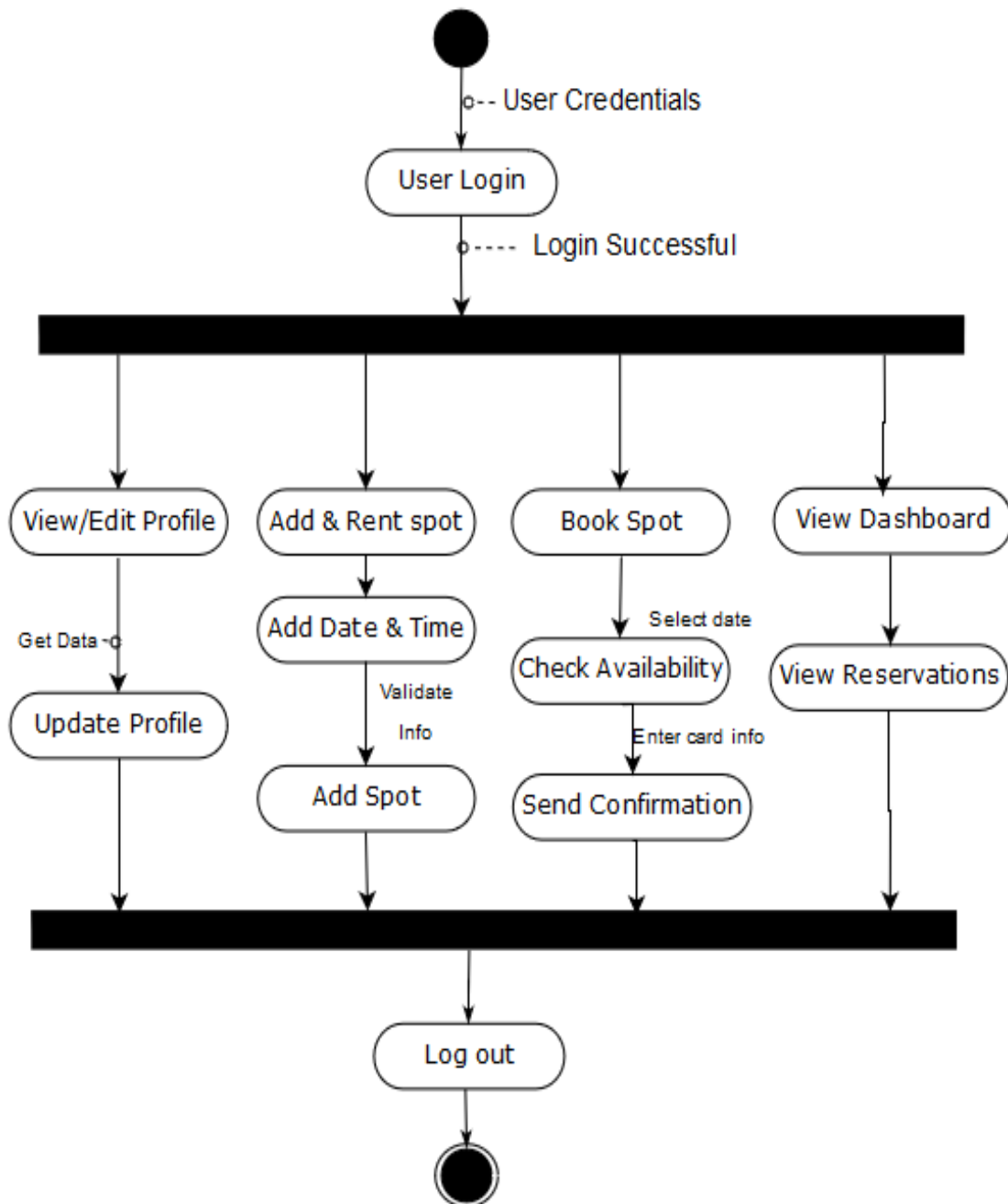


Figure 6: System Activity Diagram

User Interface Description

User interface is the medium which is used by user to control an application. User interface also known as UI or Interface is developed to make it easier for users to communicate with our service, system or application. Now-a-days mostly all software application have a user interface.

For this application we have created an aesthetic and simple user interface for users to interact with our application. Developing a web application, we used HTML as a markup language, which are rendered by the browser. We use Cascading Style Sheets (CSS) to add style to the elements of HTML example, to style fonts, add colors, images to background, provide margin and spacing to HTML elements. We use JavaScript to control client-side scripting, example checking values of HTML elements, adding, removing or appending values to existing values of the elements. We used bootstrap framework to make web page responsive to any screen size. Bootstrap framework is widely used framework which provides CSS classes and JavaScript functions to make a website responsive.

The user interface of Electronic Parking Rental System is developed using HTML CSS, JavaScript, and Bootstrap framework, which helps to create an aesthetic user interface and experience to users.

Below three images are taken on three different size of devices, the web page components adjust according to the size of the device on which it is viewed. Thus, providing best user experience and responsive interface. User can access the application on any type of device but will still feel the same interface irrespective of size or type of device they are using. The first image below represents a mobile sized device, the second image represent tablet sized device and third one is a laptop or desktop size device.

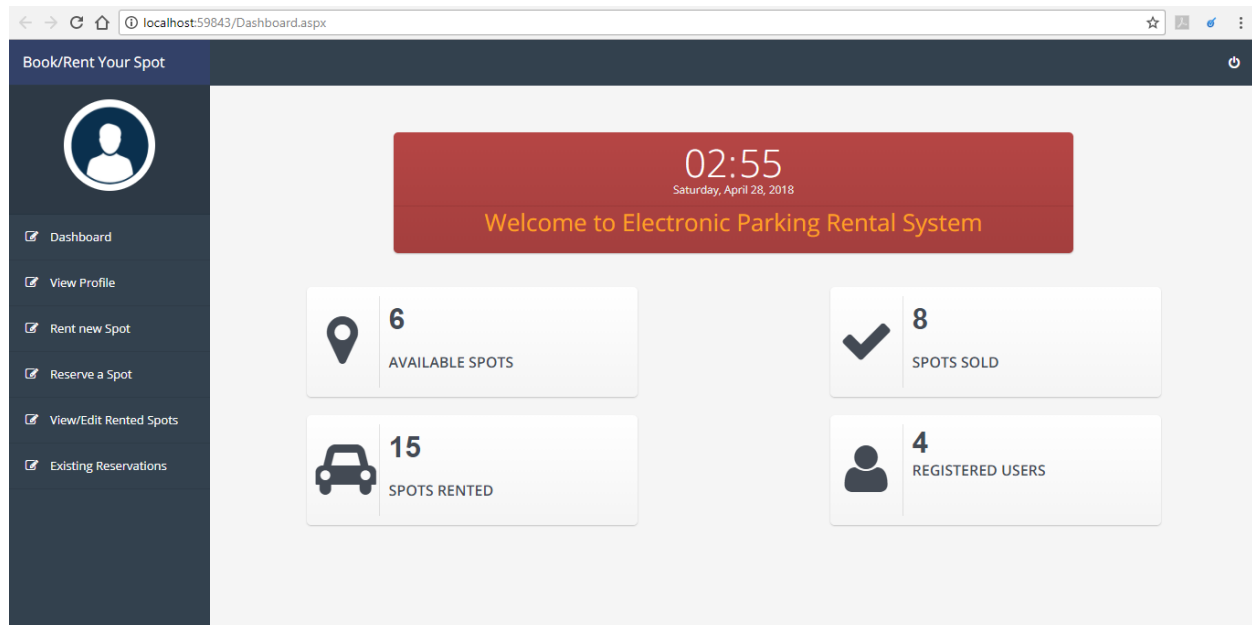


Figure 7: Desktop Application Interface

Electronic Parking Rental System

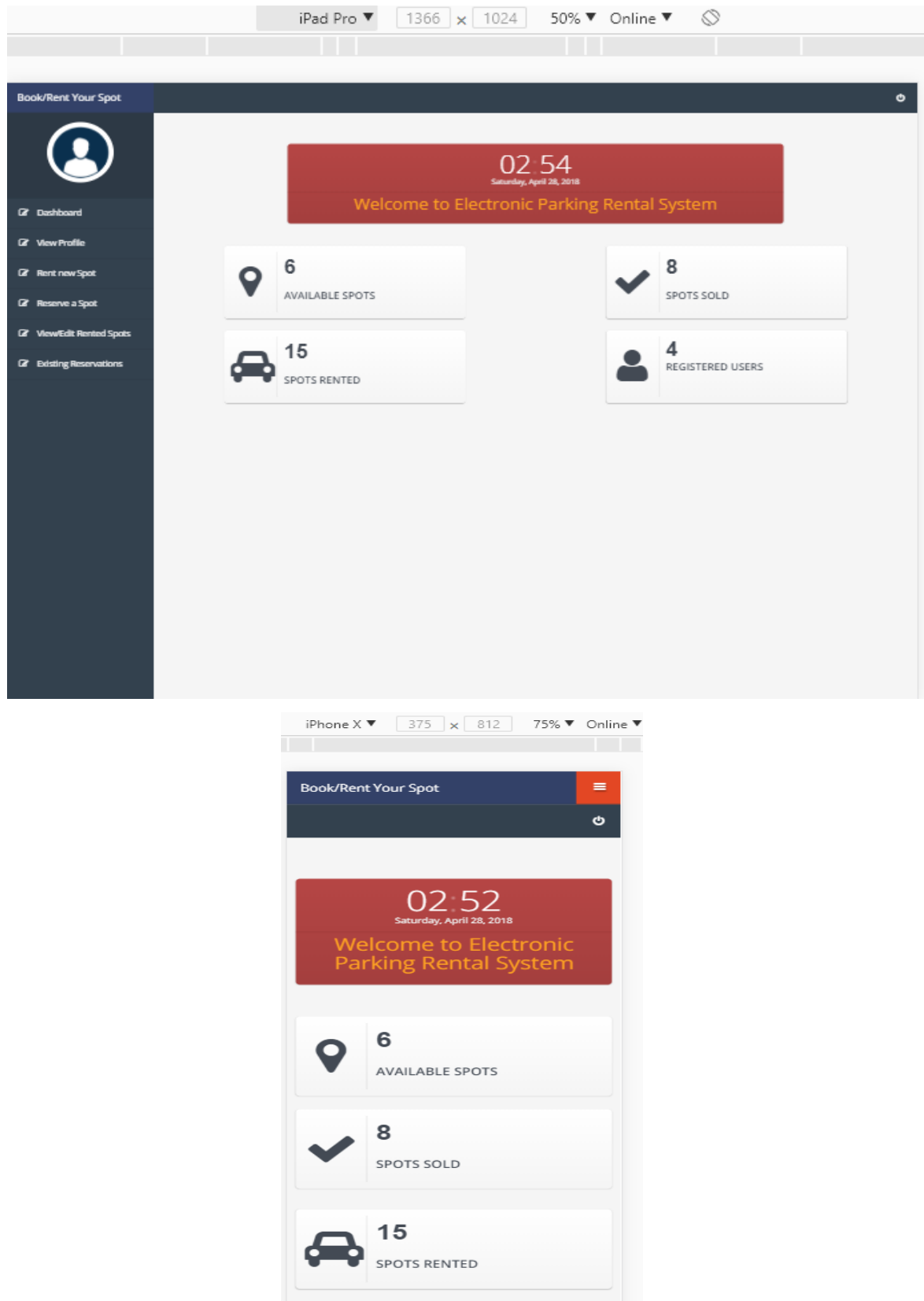


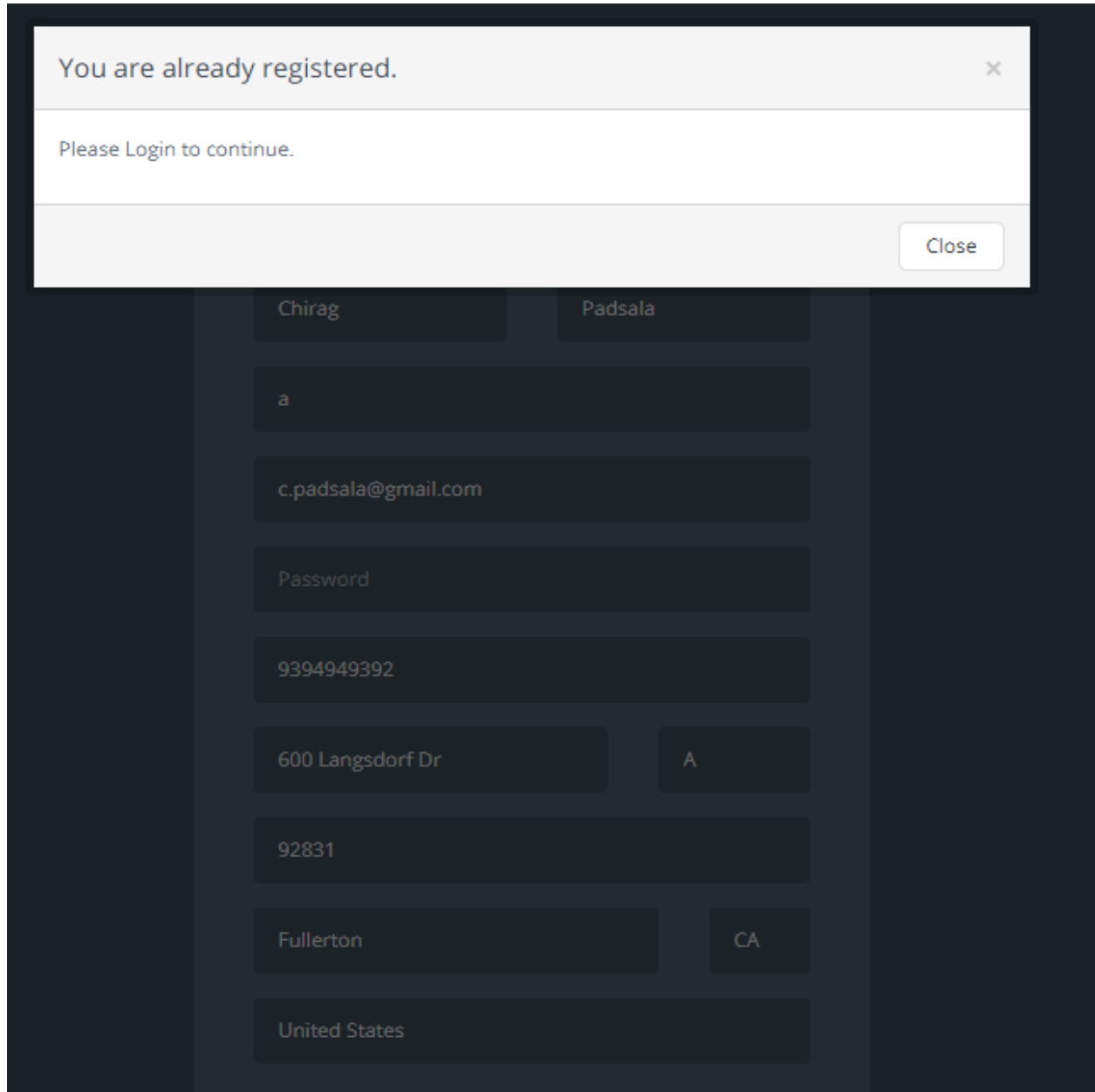
Figure 8: Tablet Interface(above) Mobile Interface (below)

Implementation

Technical Approach and Process

Register

To use this application user first needs to register themselves. User will be prompted with appropriate message. If the user is already register it will notify the user that he/she is already registered and will ask them to login. This can be seen in the image below.



The image shows a registration form for an 'Electronic Parking Rental System'. A modal message box is overlaid on top of the form, stating 'You are already registered.' with a close button (X) in the top right corner. Below the message, it says 'Please Login to continue.' and a 'Close' button is in the bottom right corner. The registration form in the background contains the following fields:

- First Name: Chirag
- Last Name: Padsala
- First Initial: a
- Email: c.padsala@gmail.com
- Password: Password
- Phone Number: 9394949392
- Address Line 1: 600 Langsdorf Dr
- Address Line 2: A
- Zip Code: 92831
- City: Fullerton
- State: CA
- Country: United States

Figure 9: User already Registered Page

If the user is a new user it will notify with a message that user is registered successfully and can now login to use the application. If the user enters invalid credentials the application will display the error message, performed while validating the form fields. Example: The validation expression for password asks them to enter password between 8 to 16 characters, including at least one uppercase, one lowercase and one number. If user enters password which fails the validation expression, the application will notify the user to enter password which matches the validation expression mentioned. Once the user is registered successfully, he/she will be displayed with the message below.

The image shows a web application interface. At the top, a light gray notification box with a close button (X) displays the message "Register Succesful." and "You are all set. Please Login to continue." Below this, a registration form is visible with the following fields and values:

Field	Value
First Name	Chirag
Last Name	Padsala
Username	abcc
Email	c.padsala@gmail.com
Password	
Phone Number	6586586585
Address Line 1	600 Langsdorf Dr
Address Line 2	A
City	92831
State	Fullerton
Zip	CA
Country	United States

Figure 10: Register Successful Page

Login

Once the user is registered successfully, he/she can login with the username and password they entered while registering for the application. While logging in, if the username and password for a user doesn't matches it will ask user to try again and if the user is not registered it will ask user to register first and then login. Which can be seen in the image below.

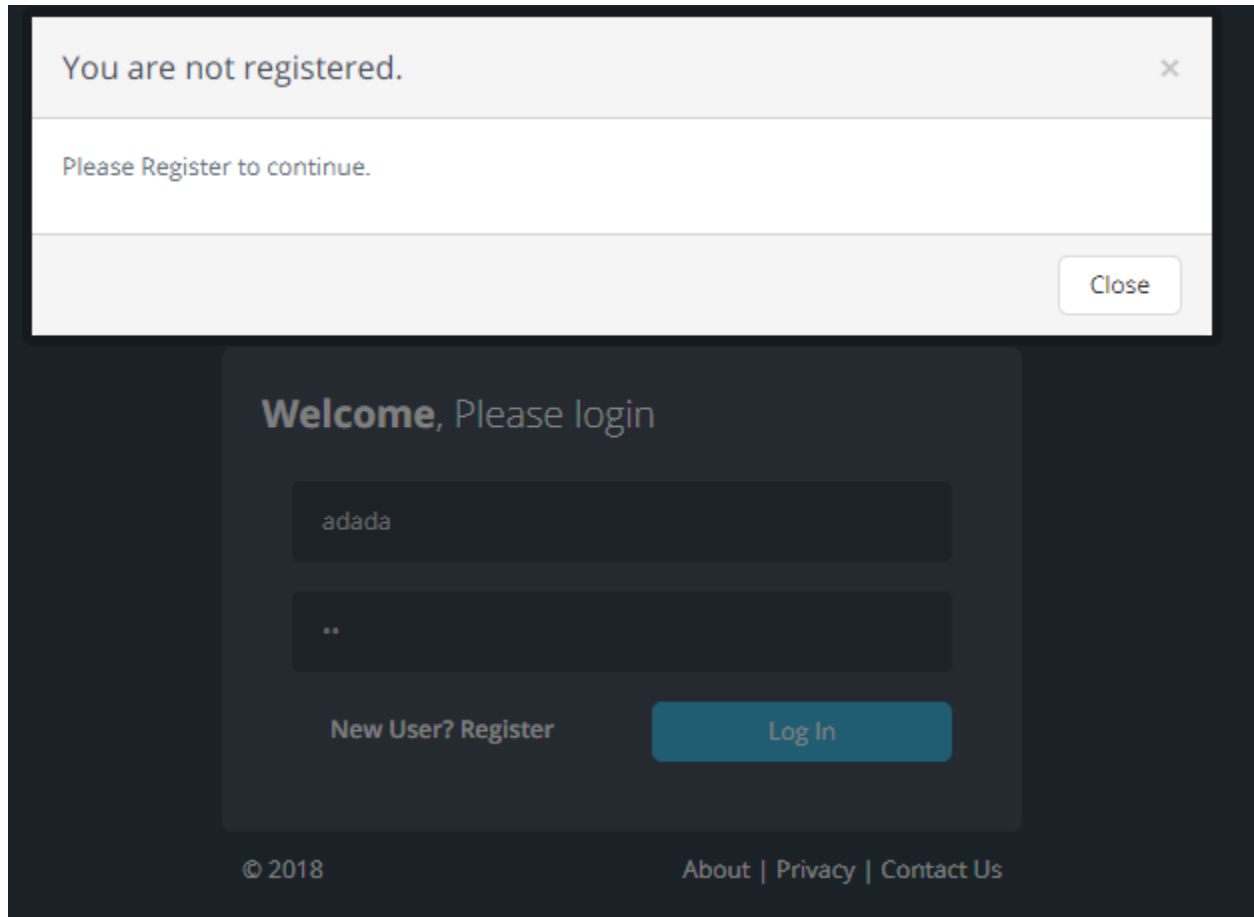


Figure 11: Login Error Page

Once the user is logged in using correct credentials he/she will be redirected to Dashboard page of the application.

Dashboard

The dashboard page displays a quick overview off the application which can be seen in the image below. It consists of multiple tiles which gives an overview about the application. Example: Welcome to the application, Numbers of registered user for this system, Number of

parking rented and booked by the user and similar. The Dashboard view can be seen in the image below.

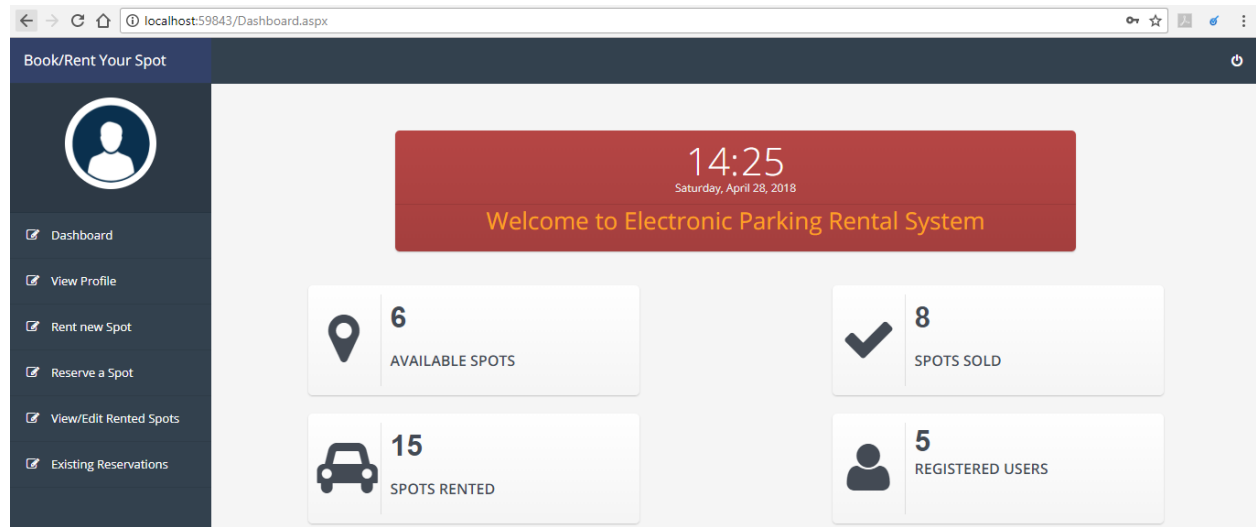


Figure 12: Dashboard Page

The user can use the side bar to navigate among the pages of the application. The side bar can be seen in the image above. It consists of 6 different pages Dashboard, View Profile, Rent new Spot, Reserve a Spot, View/Edit Rented Spot, Existing Reservation.

View Profile

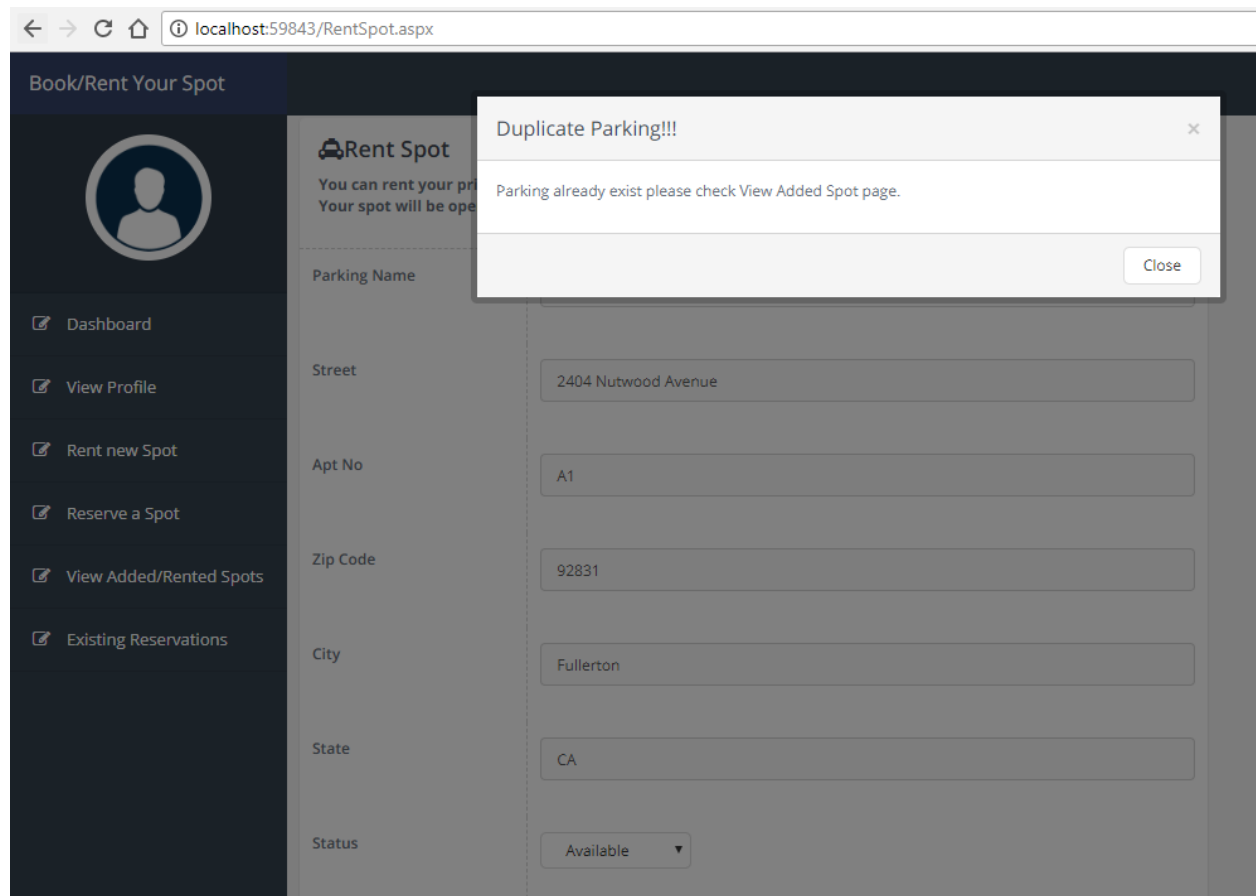
The View profile tab in the side bar redirects the user to their profile page, where user can view and edit their profile information and change password. The view for the same can be seen below.

The screenshot shows the "View Profile" page. The sidebar is the same as in the dashboard. The main content area is titled "Chirag Padsala" and features a profile picture placeholder. Below the name, there are input fields for #ID (29), Login (a), and E-mail (c.padsala@gmail.com), with a "Change password" button. To the right, there are two sections: "Profile" and "Additional Info". The "Profile" section has input fields for First Name (Chirag), Last Name (Padsala), and Phone (6573639812), with an "Update" button. The "Additional Info" section has input fields for Street Name & Apt No (600 Langsdorf Dr, Apt: F2), City (Fullerton), State (CA), and Country & Zip (United States - 92831).

Figure 13: Edit User Profile Page

Rent new Spot

The rent new spot page allows user to rent their parking spot to anyone using this application. While adding a spot to rent user can select the status of the spot to be available or unavailable. If unavailable is selected then the parking details are added to the database and later the user can rent the spot by visiting View Added/Rented Spots page. This web form performs multiple validation. First it checks for the duplicate parking entries, if found in the database it displays message indicating the spot already exist and asks them to re-rent the previously added spot using View Added/Rented Spots page.



The screenshot shows a web browser at `localhost:59843/RentSpot.aspx`. The page has a dark sidebar with a user profile icon and a menu with the following items: Dashboard, View Profile, Rent new Spot, Reserve a Spot, View Added/Rented Spots, and Existing Reservations. The main content area is titled "Rent Spot" and contains a form with the following fields: Parking Name, Street (2404 Nutwood Avenue), Apt No (A1), Zip Code (92831), City (Fullerton), State (CA), and Status (Available). A modal dialog box is displayed over the form with the title "Duplicate Parking!!!" and the message "Parking already exist please check View Added Spot page." A "Close" button is located at the bottom right of the dialog.

Figure 14: Duplicate Parking Error Page

If the spot is added successfully the system displays message to the use, which can be seen in the image below. In the future if the user wants to rent their spot again, they can just re-rent it by providing valid date and open-close time, the process for the same is also explained below in the next section.

The screenshot displays the 'Rent Spot' page in a web application. The browser's address bar shows 'localhost:59843/RentSpot.aspx'. The page has a dark sidebar on the left with a user profile icon and navigation links: Dashboard, View Profile, Rent new Spot, Reserve a Spot, View/Edit Rented Spots, and Existing Reservations. The main content area is titled 'Rent Spot' and includes a sub-header 'You can rent your parking spot. Your spot will be open for 1 hour.' Below this is a form for adding a new parking spot. The form fields are: Parking Name (empty), Street (2404 Nutwood Avenue), Apt No (A1), Zip Code (92831), City (Fullerton), State (CA), and Status (Available). A success message overlay is present, stating 'Parking Added & Rent Details Updated!' and 'Parking added. FYI- Your Id is: 41.' with a 'Close' button.

Figure 15: New Parking Added Page

View Added/Rented Spots

Once the user has added or rented spot, the user can visit this page if he/she wants to re-rent the parking spot previously added, or if the user wants to change the status or price of the rented spot. User can change the parking spot from available to unavailable only if the spot is not reserved by someone already. Similarly, the user can only change the price of the spot only if the spot is still available and not reserved by someone using the application. Below image shows all the parking spots rented in the List of spots rented Grid, and List of all the spots which can be re-rented whenever a user wants to in the List of your spots Grid.

Electronic Parking Rental System

The screenshot shows a web application interface for an Electronic Parking Rental System. The browser address bar displays `localhost:59843/ViewRentList.aspx`. The page has a dark sidebar on the left with the title "Book/Rent Your Spot" and a user profile icon. Below the icon are several menu items: "Dashboard", "View Profile", "Rent new Spot", "Reserve a Spot", "View Added/Rented Spots", and "Existing Reservations". The main content area is titled "List of spots rented" and contains a table with the following data:

Rent Id	Parking Name	Street Name & Apt No	City & State	Date Available	Status	Open Time	Close Time	Change Status
2025	F2 spot	800 N State Colg Blvd F2 spot	Fullerton CA	04/05/2018	Available	08:00	11:00	
2026	F2 spot 2	2404 Nutwood Ave F2 spot 2	Fullerton CA	05/05/2018	Available	08:00	12:00	
2027	F2 spot	800 N State Colg Blvd F2 spot	Fullerton CA	07/05/2018	Available	09:00	18:00	

Below this table is another section titled "List of your spots" with a table containing the following data:

Parking Id	Parking Name	Street Name & Apt No	City & State	Rent
24	F2 spot	800 N State Colg Blvd F2 spot	Fullerton CA	
25	F2 spot 2	2404 Nutwood Ave F2 spot 2	Fullerton CA	

Figure 16: List of Spots Added and Rented Page

If the user wishes to make a rented spot unavailable he/she can click on the edit button and make the spot unavailable and thus it won't be visible to the user searching for parking. But if the spot is already reserved by another user then the user will be displayed with a message Parking spot is already booked by a customer. Status cannot be changed. If the parking is not reserved user will be displayed with message indicating information updated successfully. Below image shows the screen where the user is changing status of parking spot which is already rented from available to unavailable.

The screenshot shows the "Update Spot Details" page in the same web application. The browser address bar displays `localhost:59843/ViewRentList.aspx?mode=edit&rentid=2025`. The sidebar is identical to Figure 16. The main content area is titled "Update Spot Details" and contains a form with the following fields:

- Rent Id: 2025
- Status: A dropdown menu with "Available" selected and "Unavailable" highlighted in blue.
- Date Available: A text input field.
- Price: 10

Below the form is an "Update" button. At the bottom of the page is a table titled "List of spots rented" with the same data as in Figure 16.

Figure 17: Change Parking Status Page

If the user wants to rent a parking spot which is added previously for some particular data and time. He/she can click the Rent Button in the List of your spots Grid and fill in the valid Date, Open-Close Time, Price and click Rent. The parking spot will then be opened for any user for the respective date and time. The user re-renting the parking spot can be seen in the image below.

Book/Rent Your Spot

Rent Spot

Spot Id: 24

Status: Available

Date Available: 2018-05-15

Open Time: 08:00 am

Close Time: 05:00 pm

Price: 10

Rent

Figure 18: Re-renting Parking Spot Page

Book a Spot

If some user using this application wishes to look for a parking spot in particular city he/she is travelling to he can navigate to Reserve a Spot page and select a date he/she is looking parking for and search the result via city name. Example of a user searching for parking on 05/04/2018 in Fullerton city can be seen in the image below.

Book/Rent Your Spot

Book a Spot

See all the available parking spots for Selected Day: 05-04-2018

May 2018						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Fullerton Search Near Me

Parking Details	From	Till	Price	Book & Pay
Street: 2404 Nutwood Avenue Apt No: A1 City: Fullerton State: CA Zip: 92831	08:00	17:00	\$ 10	Book

Figure 19: Reserve a Spot Page

The user can see if there are any parking spot for that city on the selected date. If so the user can simply click Book button and he/she will be redirected to payment page. Where the application will provide the user with parking Spot Owner's necessary information and ask them to fill in the card details and reserve a spot by clicking Pay button, which can be seen in the image below.

Figure 20: Review and Payment Page

Once the user clicks Pay button, the user will be notified via email that their parking spot is successfully reserved. Also, the owner will receive an email stating that their parking spot was rented by XYZ user and user's contact info within the email. The snap shots of emails can be seen below.

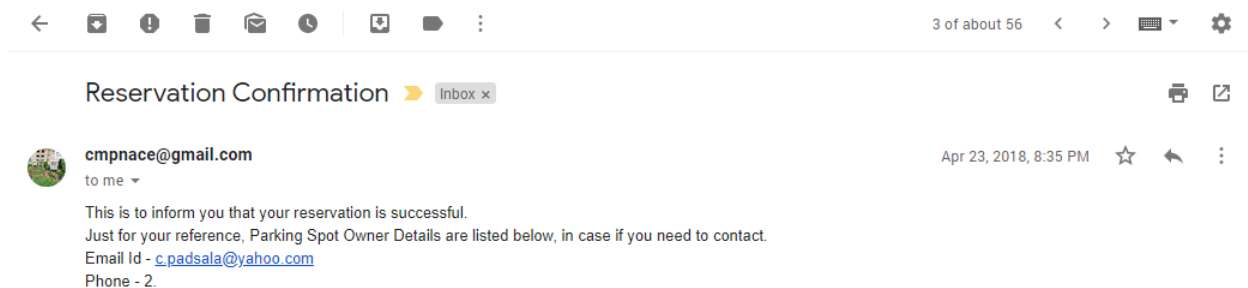


Figure 21: Reservation Confirmation Email

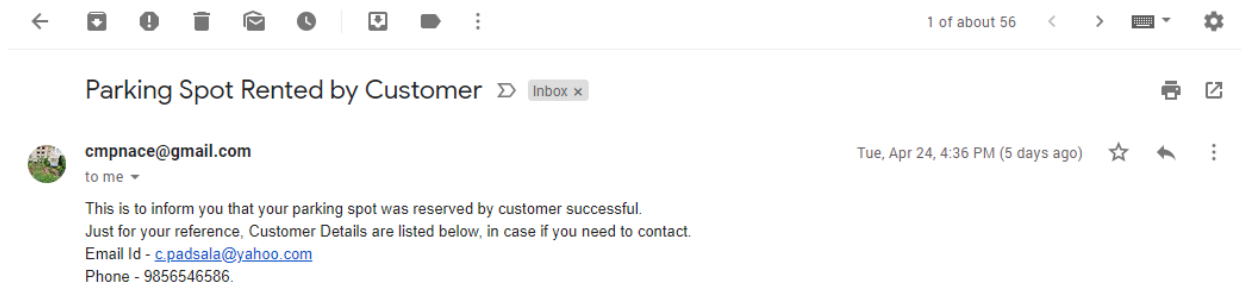


Figure 22: Spot Rented Email

Existing Reservations

Once the user has reserved a parking spot by paying for it. User can check their reservation in Existing Reservations web page. The system will display details for all the spots rented by that user. It can be seen in the image below.

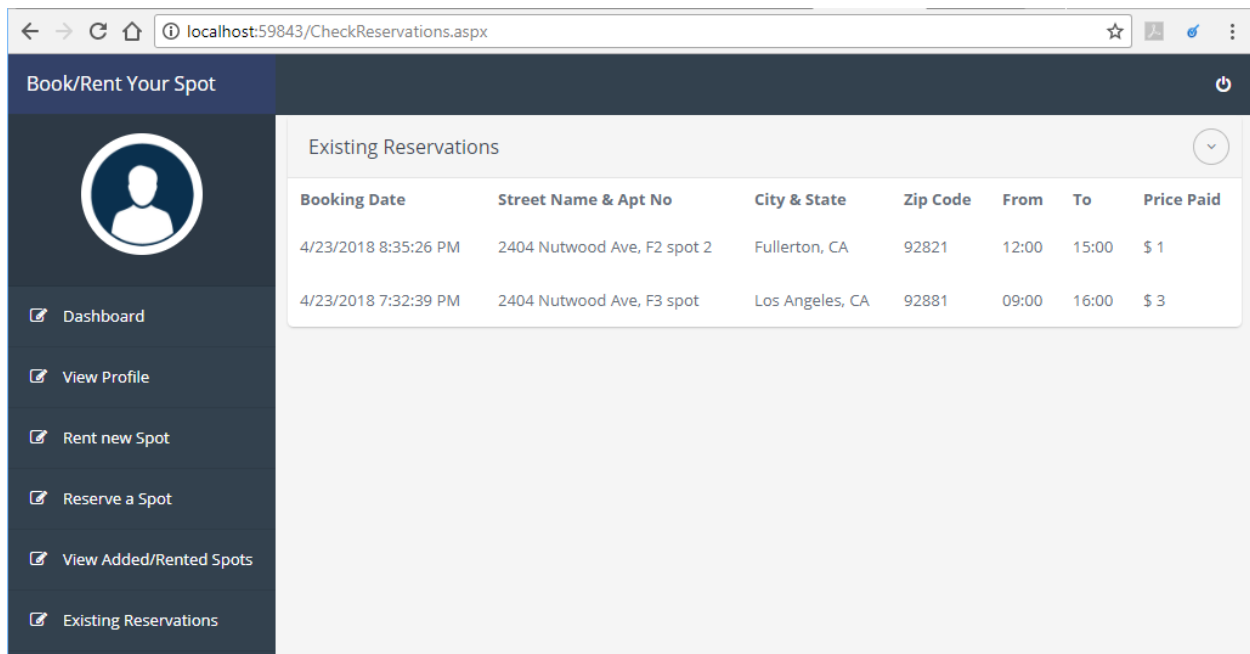


Figure 23: Check Existing Reservations Page

Admin Login

The admin can login using admin's credential, admin can login by navigation to URL /AdminLogin.aspx. The admin can fetch the booking reports for a booking date. The admin can also edit and update user information. Admin login page can be seen below.

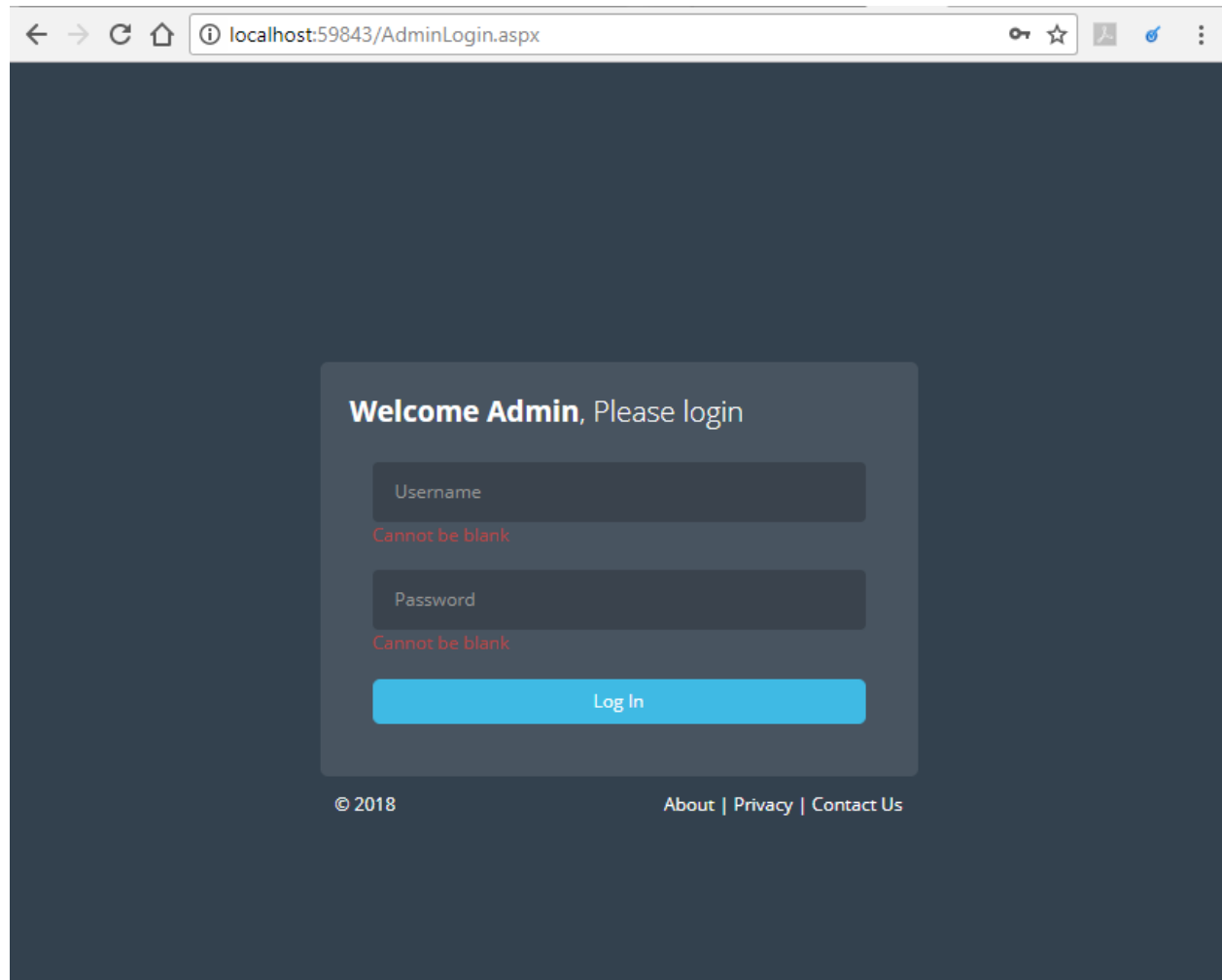


Figure 24: Admin Login Page

Edit User Info

Only admin have the access to edit user info and this page, the admin have to enter a username, a grid is populated with the list of user with that user name which is always going to be one as username is unique key. The admin updating the user information of a user can be seen in the image below.

Electronic Parking Rental System

localhost:59843/AdminEditUser.aspx?custid=29

Edit User Info

First Name:

Last Name:

Street Name:

City:

State:

Country:

List of Users

Username:

Customer Id	Username	Name	Edit User
29	Chirag	Chirag Padsala	<input type="button" value="Edit"/>

Figure 25: Admin Edit User Page

Fetch Report

This page is admin access page only. Admin can use this page to fetch all the reservation made on a particular date. Admin can select a Date and click Search, upon which all the parking spot booked on that day will be displayed. A search result for a specific day by an admin can be seen below.

localhost:59843/AdminReport.aspx

Book/Rent Your Spot

Date:

Device History Reports

Cust Id	Customer Name	Owner Id	Owner Name	Booked Street Name	Apt No	City	State	Zip Code	Booking Date
31	Barry Allen	30	Dwayne Johnson	800 N State Colg Blvd	F2 spot	Fullerton	CA	92821	4/23/2018
29	Chirag Padsala	30	Dwayne Johnson	2404 Nutwood Ave	F2 spot 2	Fullerton	CA	92821	4/23/2018
29	Chirag Padsala	31	Barry Allen	2404 Nutwood Ave	F3 spot	Los Angeles	CA	92881	4/23/2018
30	Dwayne Johnson	31	Barry Allen	600 Langsdorf Dr	F3 spot 2	Fullerton	CA	92881	4/23/2018
30	Dwayne Johnson	29	Chirag Padsala	600 Langsdorf Dr	F1 spot 4	Fullerton	CA	92831	4/23/2018
30	Dwayne Johnson	29	Chirag Padsala	800 N State Colg Blvd	F1 spot 4	Fullerton	CA	92831	4/23/2018
30	Dwayne Johnson	29	Chirag Padsala	600 Langsdorf Dr	212	Fullerton	CA	92832	4/23/2018

Figure 26: Admin Reports Page

Class Diagram

Class diagram can be considered as a major building block of application. Class diagrams help to understand the relationship between multiple classes, their interfaces, their methods and operations. It gives an overview of the relationship among each other. Class diagrams are generally helpful when building application based on object-oriented modeling. Below is the class diagram of this application, we can see there are three different types of class each representing the major methods which help build the application. The relationship amongst them can be seen below. It is generally read as one Customer can have multiple Reservations. One Parking Space can be rented multiple times, one customer can rent multiple parking spots. Each class has corresponding methods, for class Parking Space we can see methods which check for duplicate parking entries using `CheckDuplicate()` method. Inserts new parking spots if no duplicates are found using `InsertParking()` method. Let customer re-rent previously added parking spots using `ReRentParking()` method. Allow customers to change price and status of the spots using `ChangePrice()` and `ChangeStatus()` methods. Reservation class contains method which fetches all the reservation to be displayed to respective customer using `FetchReservations()` method. Customer class contains methods which registers new user by inserting their information using `InsertCustomer()` method. Also if a customer updates their profile information `UpdateCustomerInfo()` method is used to change the user info within the database. The below diagram represents the general overview, relationship among multiple classes, some methods and functions of this application using a class diagram.

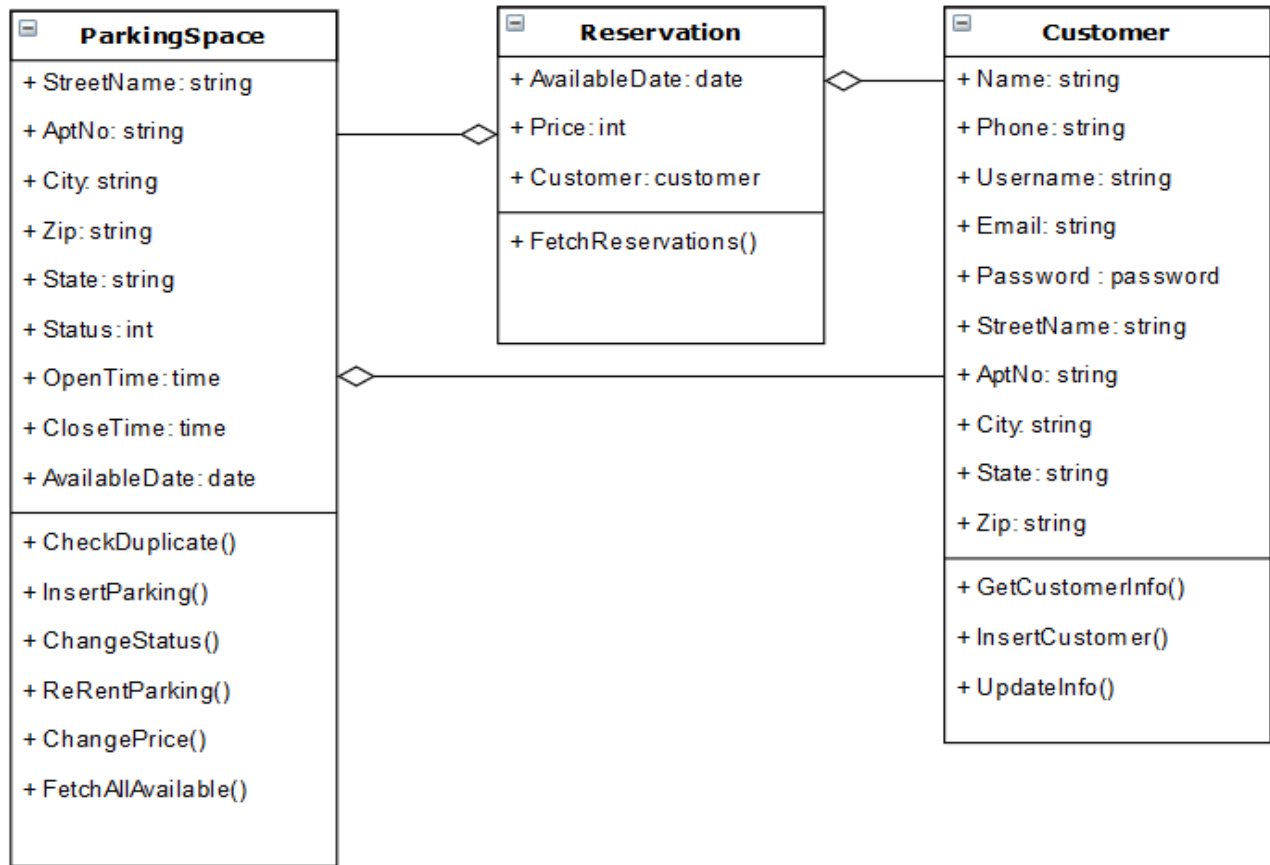


Figure 27: Class Diagram

Data Design

Database designing is the most crucial part of any application. All the data coming and going through requests needs to be stored in a single place which is accessible easily when needed. This is where database designing comes into picture. Data can be store in a structured and unstructured database; structured data are mostly relational data which are stored in multiple table with primary key and foreign key relationship. While unstructured data are stored without any relation in a file type format. Our system uses relational database management system, in which data is stored in multiple tables, which can then be retrieved by firing queries on the database. Below image shows the entity relationship diagram of ERPS database used to store data related to this application.

Parentetical method of ERPS Database:

1. Customer (**CustomerId**, Username, Email, Password, FirstName, LastName, StreetName, AptNo, City, State, Country, ZipCode, Phone, Type)
2. ParkingSpace (**ParkingId**, *CustomerId*, ParkingName, StreetName, AptNo, State, City, ZipCode)
3. ParkingHours (**RentId**, *CustomerId*, StatusId, *ParkingId*, DateAvalable, *TimeId*, DateAdded, OpenTime, CloseTime, Price, BookingDate, *BookingCustId*, Booked)
4. Time (**TimeId**, Time24Hr, TimeAMPM)
5. Status (**StatusId**, StatusType, Desc)

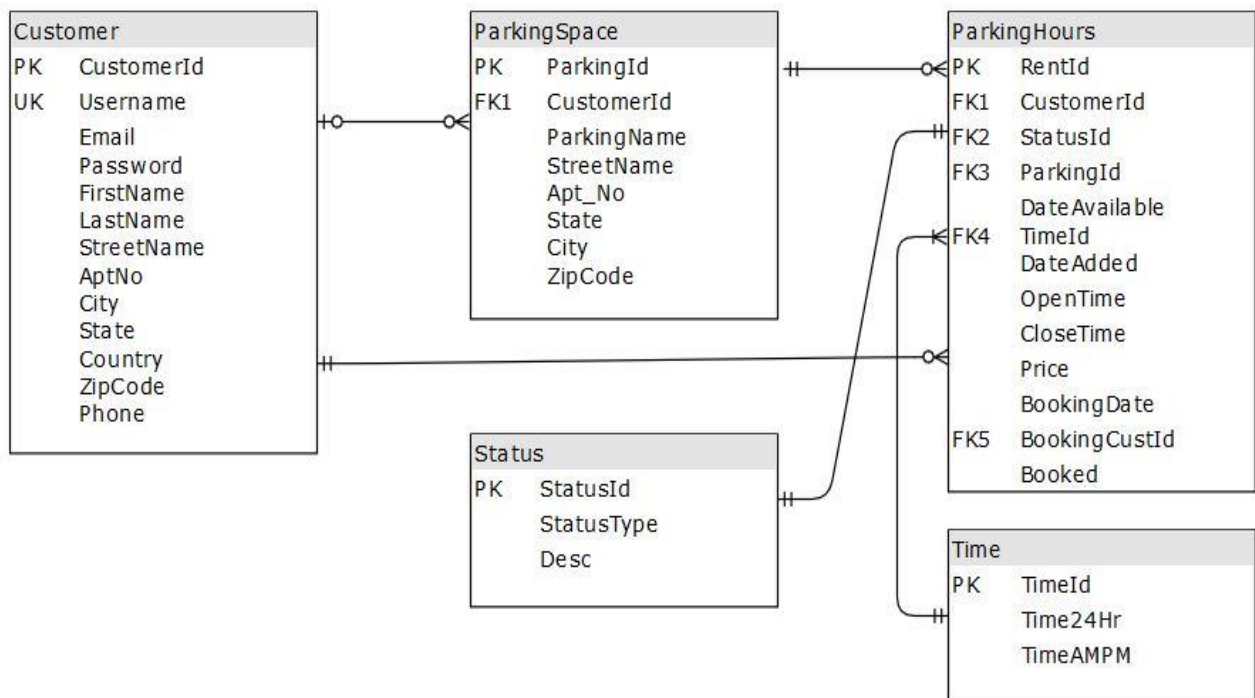


Figure 28: Entity Relationship Diagram

Test and Integration

Testing is one of the important side of building an application. There are various types of testing which were performed while developing the application and after the application was up and running. Below are the various types of testing that the application went through and explanation about each of it.

Functionality Testing

Functionality testing refers to the testing of all the interfaces, components links used in the web application. Example, testing database connection, testing web forms for getting or updating user information.

Test Performed	Outcome
Tested if database connection is established every time it is called.	Passed
Testing of redirection of all the internal and external links.	Passed
Testing of links redirecting to the same page.	Passed
Validation testing of all the web form in the application.	Passed
Tested the modification of web form based on the inputs and dropdown selection.	Passed
Testing if error message is generated for wrong input to fields in web form.	Passed
Testing of smtp connection to send email confirmation.	Passed
Testing of database integrity after either of the CRUD operation is performed.	Passed
Testing of Search by city feature to display accurate result.	Passed
Testing the near-by feature to fetch correct result.	Passed

Usability Testing

Usability testing generally refers to the process through which characteristics of systems are measured based on user interaction with the application.

Test Performed	Outcome
Tested the messages generated for the user guide them through the application.	Passed
Tested the navigation of links from one page to another.	Passed
Tested the application as per user perspective.	Passed

Tested the accessibility of the side menu bar from all the web pages except for Register and Login	Passed
Testing all the predefined Bootstrap classes and self-generated CSS classes to match the styles and colors of the application.	Passed
Testing if error message is generated for wrong input to fields in web form.	Passed
Tested the placement of web forms and containers of text for each page.	Passed

Interface Testing

The interface testing generally refers to the testing of communication and compatibility of different systems with each other. Testing how do different system interface works and talk with each other.

Test Performed	Outcome
Tested all the exceptions raised while back end server communicates with the database while executing a of query or stored procedure.	Passed
Tested the data binding of the front-end interface, components and their respective values, through back end system while the page loads.	Passed
Tested the connection established from C# to SQL Database using the connection string.	Passed
Tested the behavior and redirection of the application by restarting the server multiple times.	Passed

Compatibility Testing

Compatibility testing is one of the basic testing which should be given higher importance. It specifies the compatibility of the application with every interface it communicates with.

Test Performed	Outcome
Tested all the operating environments compatibility.	Passed
Tested the operating system compatibility with the application.	Passed

Tested the browser compatibility by running the application on multiple browsers.	Passed
Tested the CSS class compatibility to work with multiple top used browser.	Passed
Tested the application responsiveness by running the application on different size of devices including mobile, tablet, laptop and desktop.	Passed
Tested the AJAX and JavaScript compatibility with multiple top used browsers.	Passed

Performance Testing

Performance testing is the most crucial part of testing. Performance testing is must irrespective of the type of application we are developing. Performance testing generally checks for the accuracy and responsiveness of the application, how fast it is for multiple users to use is simultaneously. Load testing and stress testing are the key of performance testing.

Test Performed	Outcome
Tested the application to service 25 request per seconds without scaling the application.	Passed
Tested the system by running it on different browser parallelly and registering 4 users at the same time.	Passed
Reviewed the CPU and Memory usage by the application inside the Visual Studio. Decreased the usage by decreasing the requests to the server-side programs.	Passed
Tested the application by accessing the same page on multiple browser parallelly.	Passed

Security Testing

Security testing as the name suggests refers to the security of the application and data to any unauthorized users. It checks for any unauthorized user trying to access the resource of the application without authentication.

Test Performed	Outcome
Tested to log in the application by using invalid credentials.	Passed

Tested the application by entering the URL manually to access a particular page without logging in.	Passed
Tested the error messages while registering new users and duplicate users.	Passed
Tested the application by logging in the application without registering.	Passed
Tried to access the application by providing the URL with invalid query parameters.	Passed
Tested the application by trying to access each page to check if the errors are handled efficiently by the try-catch method.	Passed
Tested the application to check if it handles the SQL injection problem efficiently.	Passed

Installation Instruction

Software Required

- Windows OS
- Microsoft Visual Studio
- MS-SQL Server
- Web Browser

Installation Steps

- Install Microsoft Visual Studio 2012 or higher
- Install MS-SQL Server 2008 or higher
- Install Google Chrome Web Browser or Similar

Executing Project

- Open EPRS.sln file from EPRS Project Folder. It will open in Microsoft Visual Studio.
- Run ERPS.sql file. This is a script file for creatind a Databse with name EPRS. It will open in MS-SQL Server.
- Navigate to the project opened in Microsoft Visual Studio. Run the project in the installed browser.
- Go forward, create account and login to experience the application.

Operating Instruction

Basic Instructions to use the Application

- After the project starts running on a web browser, user will be redirected to Login Page.
- If the user isn't registered the Login page will have a link indicating user to Register.
- Once the user is registered and logged into the application, the user can rent a new spot by navigating to the Rent a Spot page using the side navigation menu bar and filling the web form and upon successful request the user spot will be rented to other users using the application.
- To reserve a parking spot user can navigate to Reserve a Spot Page, by navigating to the Reserve a Spot page using the side navigation menu bar, user will then select a date and book the spot which is convenient to him/her based on the location. User can also search for parking spot based on the city and near-by location for the selected date.
- Once the user has reserved the parking spots, user will get a confirmation via email. User can also check their reservation by navigating to Existing Reservation Page.
- Once the parking spot is rent by user, the information is stored in the database, the user can then re-rent the same parking spots as many time for different dates and time. To do so user can navigate to the View Rented Spots Page using the side bar navigation.
- Once the user has completed their task, he/she can log out of the application.

Recommendations for Enhancements

Future Scope

- The application can be hosted on a private server to make it accessible to actual users around the globe to generate revenue.
- The application can be enhanced by adding feature which let user to rent their parking spots overnight without renting it multiple times.
- The application can be enhanced by allowing user to reserve a parking spot for a particular time in the range of the spot rented by Owner. This can help multiple user to rent the same parking spots as per its availability.
- The application code can be scaled horizontally because of the coding standards and framework followed, if new features are to be added.

- The user experience can be enhanced by providing on screen “How to use application tutorials” for first time users.
- The application can be enhanced by allowing user to rent and reserve the parking spots on hourly basis.
- The parking spots reserved can be canceled by the same user to receive a refund.
- The application can be made available on Android, iOS and similar platform.

Bibliography

References

Web Testing Complete Guide, January 2018.

<http://www.softwaretestinghelp.com/web-application-testing/>

Draw.io

<https://www.draw.io/>

Entity Relationship Diagrams, Lucid Chart 2018

<https://www.lucidchart.com/pages/ER-diagram-symbols-and-meaning>

Multitier architecture, Wikipedia 2018

https://en.wikipedia.org/wiki/Multitier_architecture

Software design description, Wikipedia 2018

https://en.wikipedia.org/wiki/Software_design_description

Software Architecture, Wikipedia 2018

https://en.wikipedia.org/wiki/Software_architecture

User Interface Design, Wikipedia 2018

https://en.wikipedia.org/wiki/User_interface_design

UML-Activity Diagram, Tutorials point 2018

https://www.tutorialspoint.com/uml/uml_activity_diagram.htm

UML-Use Case Diagram, Tutorials point 2018

https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm

Relational Database Design, Nanyang Technological University

https://www.ntu.edu.sg/home/ehchua/programming/sql/Relational_Database_Design.html

Class Diagram, IBM 2010

<https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/>

Parking Problems, Parking Reform.

<http://www.parkingreform.org/parking-problem.html>

Parking Management, Todd Litman.

<https://www.planetizen.com/node/19149>

Functional vs Nonfunctional requirements, ReQtest April 2015

<https://reqtest.com/requirements-blog/understanding-the-difference-between-functional-and-non-functional-requirements/>