



**CSCI 5308**  
**Advance Topics in Software Development**

**FINAL PROJECT REPORT ON**

**Phoenix**

**GROUP -14**

Dharminsinh Pankajsinh Rathod – B00908277

Nirav Radadiya – B00909651

Shivam Dinesh Rank – B00897772

Sparsh Purwar – B00885683

Tejaswini Rallapalli – B00888478

**Client Team (Group-3)**

Smriti Mishra-B00904799

Shiva Shankar Pandillapalli-B00880049

Ferin Rakeshkumar Patel-B00891975

Chirag Jayeshkumar Pancholi-B00911403

Paras Patel-B00911202

**Professor – Dr. Tushar Sharma**

**TA – Harsh Samirbhai Bhatt**

## TABLE OF CONTENTS

1. Executive Summary.....	2
2. Background Research.....	2
3. Technologies Used.....	3
4. Build/ Deployment Instructions.....	3-5
5. Usage Scenarios.....	6
5.1 User Management.....	7-8
5.2 Slot Booking.....	8-10
5.3 User Slot addition.....	10
5.4 Payment.....	11-13
5.5 Map Integration.....	13-14
5.6 Feedback.....	15
5.7 Admin portal management.....	16
5.8 Email Confirmation.....	17
6. Scope.....	18
7. Further Improvements.....	18
8. GitLab URL .....	18
9. Heroku Link .....	18
10. References.....	19

## **1. EXECUTIVE SUMMARY**

The main idea of the project Phoenix is to develop a web application where users can have a hassle-free experience of booking slots for parking with one tap payment system. This application has two portals User portal and an Admin portal. Admin is responsible for adding and removing the parking spots in the portal. User will be able to search for the nearest parking spots and based on the availability they will be able to book the slots for the required time. After successfully booking the slot, user will be redirected to the payment page. A confirmation email will be sent to the user that a particular slot has been booked with the Transaction ID and the slot details. The main aim of this application is that users can grow with phoenix. They can add their private spots for parking and earn from it. This means that a user can be either an end-customer who is using this application and an admin who can add the slots for other users. Hence, a user will be able to book a public or a private spot based on his choice. This unique feature in the application makes more impact on the end users. The user-friendly registration, viewing the location on maps, finding more spots for parking and the secure payment system allows the user for guaranteed parking for the upcoming trips and save time on rush days.

## **2. BACKGROUND RESEARCH**

Before we directly jumping to the project, we first gave time in understanding what we need to achieve and how are we going to achieve. Our main focus was to build an efficient project satisfying the client requirements. For this, we referred some documentations available online on how this booking a parking spot feature can be done. We dedicated more time in collecting resources required for this project. After we got everything in our hand, we divided the work among ourselves that each person will contribute to the one feature and will collect all the required resources to complete that feature. For every feature that we did, we put some short-term goals and helped each other with gathering inputs for the task. We figured it out that a parking system can be implemented in several ways. The approach we opted for is using Java and Spring Boot framework. Spring Boot allows us to create standalone applications without fuss that runs on their own without relying on any external web servers. It also reduces the overall development time of the application. We used Thymeleaf, which is a java-based library that provides a good support for serving an XML and HTML in web applications. It will directly reference the model that we produce in the controller during template rendering. We have taken the inspiration from SB Admin2 and Bootstrap which has default templates for the front-end. We learned how the deployment of standalone applications can be done instead of using the localhost. We used Heroku which is a Container-based cloud platform for deploying our application. Heroku uses dynos for running apps which is nothing but a virtual system.

### 3. TECHNOLOGIES USED

**Back End:** Java, Spring Boot

**Front End:** Thymeleaf, Bootstrap, HTML, CSS, Java Script

**Database:** MySQL

### 4. BUILD AND DEPLOYMENT INSTRUCTIONS:

There are three stages:

```
stages:          # List of stages
- build
- quality
- deploy
```

- **Build:**

1. “mvn clean install” will check if the test cases and dependencies are passed or not.
2. Maven is taken as base for build stage.

```
build:           # This job runs in the build stage, which runs first.
image: maven:3-jdk-11
stage: build
script:
- cd /builds/courses/2022-winter/csci-5308/group14/PH
- mvn clean install
```

#### Commands

- cd /builds/courses/2022-winter/csci-5308/group14/PH
- mvn clean install

- **Quality:** We are taking Ubuntu as base image and with the help of Designite tool the code quality will be checked and smells will be analyzed. Below is the screenshot of code quality analysis that we did for our project using Designite.

```
Searching classpath folders ...
DesigniteJava Enterprise. Version 2.0.2
Copyright (C) 2022 Designite. All rights reserved.
Parsing the source code ...
Resolving symbols...
Computing metrics...
Detecting code smells...
Exporting analysis results...
wrapping up ...
--Analysis summary--
Total LOC analyzed: 1340      Number of packages: 14
Number of classes: 37      Number of methods: 243
--Total architecture smell instances detected--
Cyclic dependency: 2      God component: 0
Ambiguous interface: 0      Feature concentration: 0
Unstable dependency: 1      Scattered functionality: 0
Dense structure: 0
--Total design smell instances detected--
Imperative abstraction: 0      Multifaceted abstraction: 0
Unnecessary abstraction: 0      Unutilized abstraction: 14
Feature envy: 0      Deficient encapsulation: 2
Unexploited encapsulation: 0      Broken modularization: 0
Cyclically-dependent modularization: 0      Hub-like modularization: 0
Insufficient modularization: 3      Broken hierarchy: 0
Cyclic hierarchy: 0      Deep hierarchy: 0
Missing hierarchy: 0      Multipath hierarchy: 0
Rebellious hierarchy: 0      Wide hierarchy: 0
--Total testability smell instances detected--
Hard-wired dependency: 5      Global state: 2
Excessive dependency: 0      Law of Demeter violation: 0
--Total implementation smell instances detected--
Abstract function call from constructor: 0      Complex conditional: 0
Complex method: 0      Empty catch clause: 0
Long identifier: 0      Long method: 0
Long parameter list: 6      Long statement: 11
Magic number: 32      Missing default: 0
Done.
```

```

quality:      # This job runs in the build stage, which runs first.
image: ubuntu:18.04
stage: quality
script:
  - apt-get update
  - apt-get clean
  - apt install default-jre -y
  - apt install default-jdk -y
  - apt install maven -y
  - apt install wget -y
  - apt-get install curl -y
  - apt install git -y
  - cd /builds/courses/2022-winter/csci-5308/group14/PH
  - wget https://www.designite-tools.com/static/download/DJC/DesigniteJava.jar
  - java -jar DesigniteJava.jar -i ./ -o code-quality/ -f XML

```

## Commands

- apt-get update
- apt-get clean
- apt install default-jre -y
- apt install default-jdk -y
- apt install maven -y
- apt install wget -y
- apt-get install curl -y
- apt install git -y
- cd /builds/courses/2022-winter/csci-5308/group14/PH
- wget <https://www.designite-tools.com/static/download/DJC/DesigniteJava.jar>
- java -jar DesigniteJava.jar -i ./ -o code-quality/ -f XML

- **Deploy:** We are using Ubuntu as base image and with the help of Ruby we are deploying it to Heroku.

The screenshot shows the Heroku dashboard for the application 'asdc-group-14'. At the top, there's a navigation bar with 'Personal' and 'asdc-group-14'. Below this is a tab bar with 'Overview', 'Resources', 'Deploy', 'Metrics', 'Activity', 'Access', and 'Settings'. The 'Overview' tab is selected. Under 'Free Dynos', there's a 'Change Dyno Type' button. Below that, a command line is shown: `web java -Dserver.port=$PORT $JAVA_OPTS -jar target/phoenix-0.0.1-SNAPSHOT.jar`. To the right of the command line is a toggle switch and the text '\$0.00'. Below the command line, there's an 'Add-ons' section with a 'Find more add-ons' button. A search bar is present with the text 'Quickly add add-ons from Elements'. Below the search bar, a message states: 'There are no add-ons for this app. You can add add-ons to this app and they will show here. [Learn more](#)'. At the bottom, the 'Estimated Monthly Cost' is shown as '\$0.00'.

```

deploy:      # This job runs in the build stage, which runs first.
  image:      ubuntu:18.04
  stage:      deploy
  script:
    - apt-get update
    - apt-get clean
    - apt install default-jre -y
    - apt install default-jdk -y
    - apt install maven -y
    - apt install wget -y
    - apt-get install curl -y
    - apt install git -y
    - cd /builds/courses/2022-winter/csci-5308/group14/PH
    - mvn clean install
    - apt-get install -y ruby-dev
    - gem install faraday -v 1.10.0
    - apt-get install -y ruby-dev
    - gem install dpl
    - dpl --provider=heroku --app=asdc-group-14 --api-key="818f1242-336d-4db1-896d-94fff45e033f"

```

## Commands

- apt-get update
- apt-get clean
- apt install default-jre -y
- apt install default-jdk -y
- apt install maven -y
- apt install wget -y
- apt-get install curl -y
- apt install git -y
- cd /builds/courses/2022-winter/csci-5308/group14/PH
- mvn clean install
- apt-get install -y ruby-dev
- gem install faraday -v 1.10.0
- apt-get install -y ruby-dev
- gem install dpl
- dpl --provider=heroku --app=asdc-group-14 --api-key="818f1242-336d-4db1-896d-94fff45e033f"

## Command for deploying on Heroku:

web java -Dserver.port=\$PORT \$JAVA\_OPTS -jar target/phoenix-0.0.1-SNAPSHOT.jar

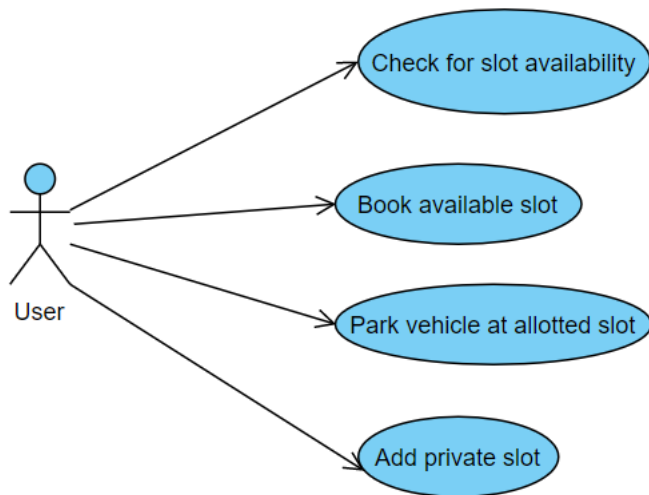
## Screenshot showing three stages running on Gitlab:

The screenshot shows the GitLab CI/CD interface for a pipeline named "Pipeline #154595" triggered 1 hour ago by Sparsh Purwar. The pipeline is marked as "passed". Below the pipeline status, there is a section titled "Update .gitlab-ci.yml file" which shows 3 jobs for the main branch in 45 minutes and 34 seconds (queued for 2 seconds). The jobs are listed as "latest" and "be261e46". Below this, there is a section titled "Pipeline" with tabs for "Needs", "Jobs", and "Tests". The "Jobs" tab is selected, showing three stages: "Build", "Quality", and "Deploy". Each stage has a green checkmark and a refresh icon, indicating that all jobs in the pipeline have passed successfully.

## 5. USE CASE SCENARIOS

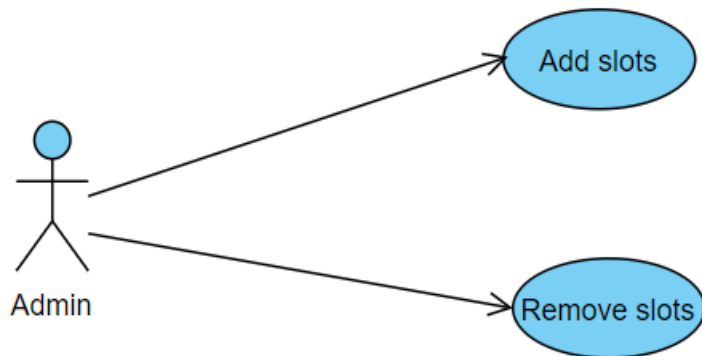
### High level Use case diagram from User perspective:

In the user management system, a user can check for the available slots, book the slot, park the vehicle at the slot and user can also add his/her private slots for parking.

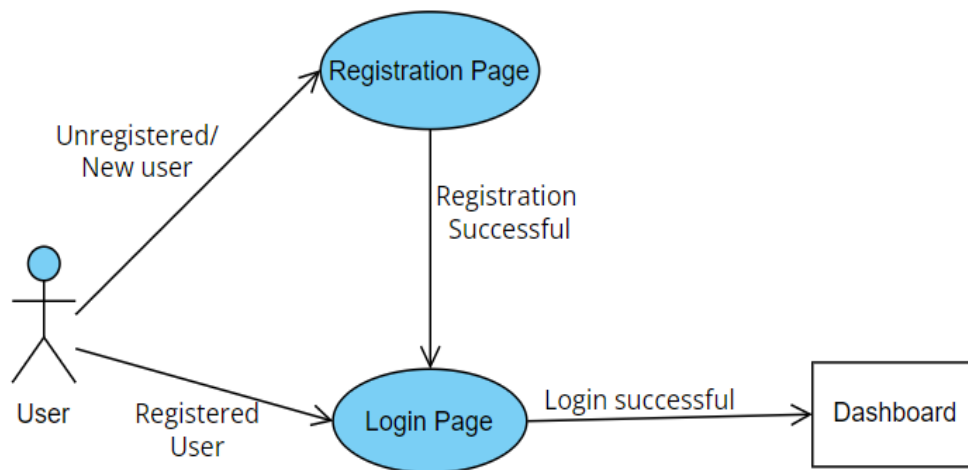


### High level Use case diagram from Admin perspective:

For the admin portal, Admin is responsible for adding and removing the slots from the portal which will be reflected to the users.



## 5.1 User Management



### a. User Registration:

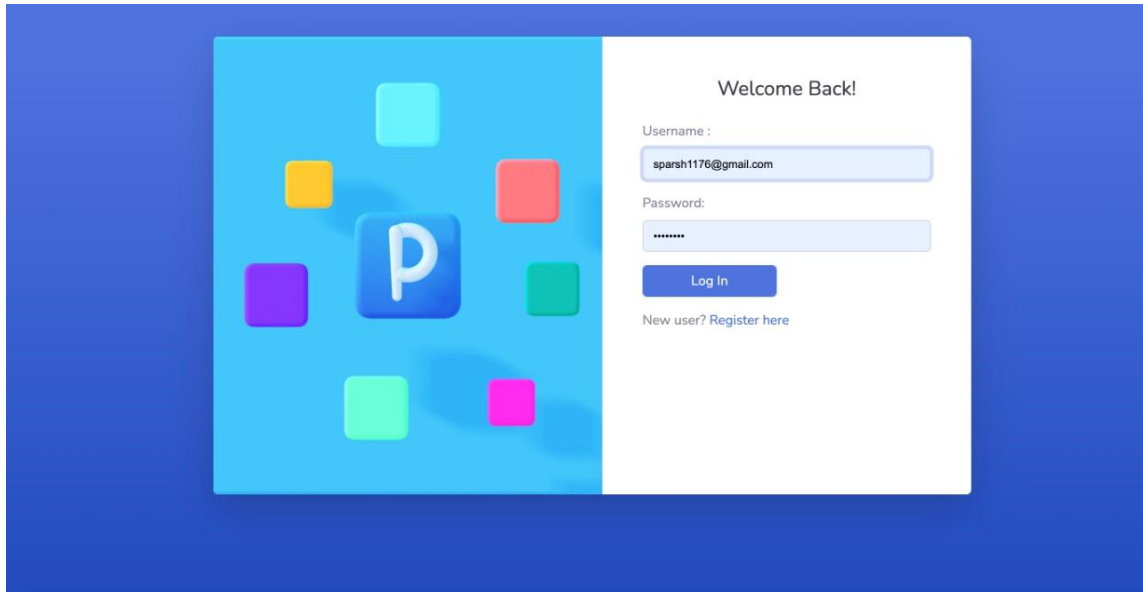
Registration to phoenix is free. So, to use this application, a new user has to first register by filling out the details like First name, Last name, Email ID, create password for the application, Street name, Unit number and city which are the inputs for registration. Once the user is successfully registered with Phoenix, he/she will be redirected to the login page where they can login using the registered credentials i.e., Email ID, Password.

The screenshot shows a web interface for creating an account. On the left is a blue sidebar with a large white 'P' logo and several colorful squares. The main content area is white and titled 'Create an Account!'. It contains the following form fields: 'First Name' and 'Last Name' (two input boxes), 'Email' (one input box), 'Password' (one input box), 'Street Name' (one input box), and 'Unit Number' and 'City' (two input boxes). Below these fields is a blue 'Register' button. At the bottom, there is a link that says 'Already registered? Login here'.

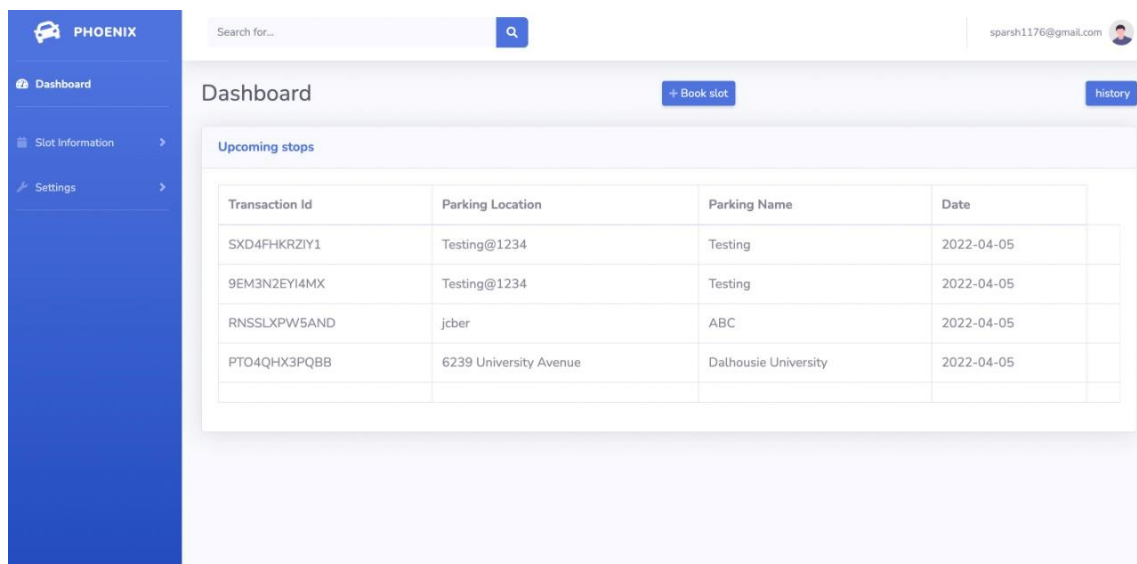
### b. User Login:

In the user login page, user has to enter email id and password which is the input for login. After successful login, user will see a Dashboard with his profile which has current trips i.e., current day slot bookings along with Slot information and Slot booking option. The output of user login is that a user should be able to successfully login and should see the dashboard.

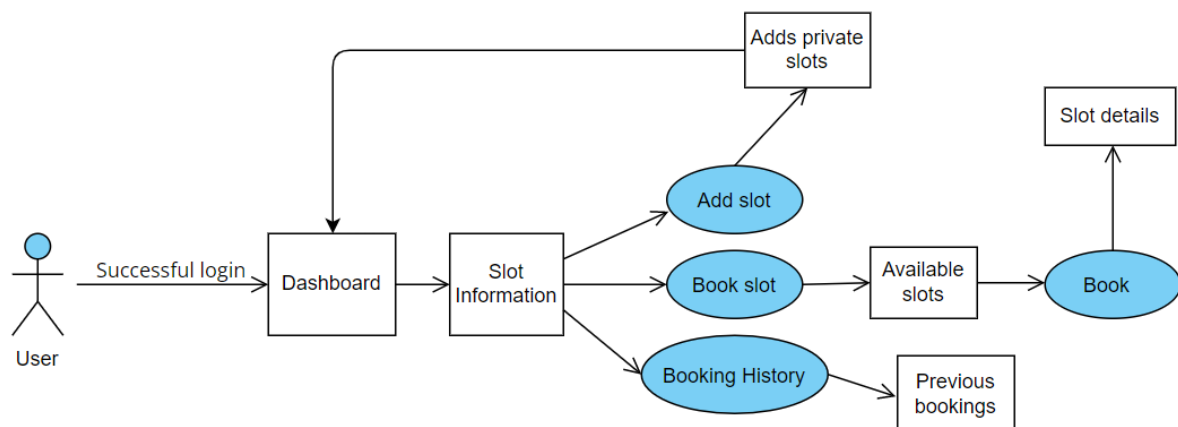




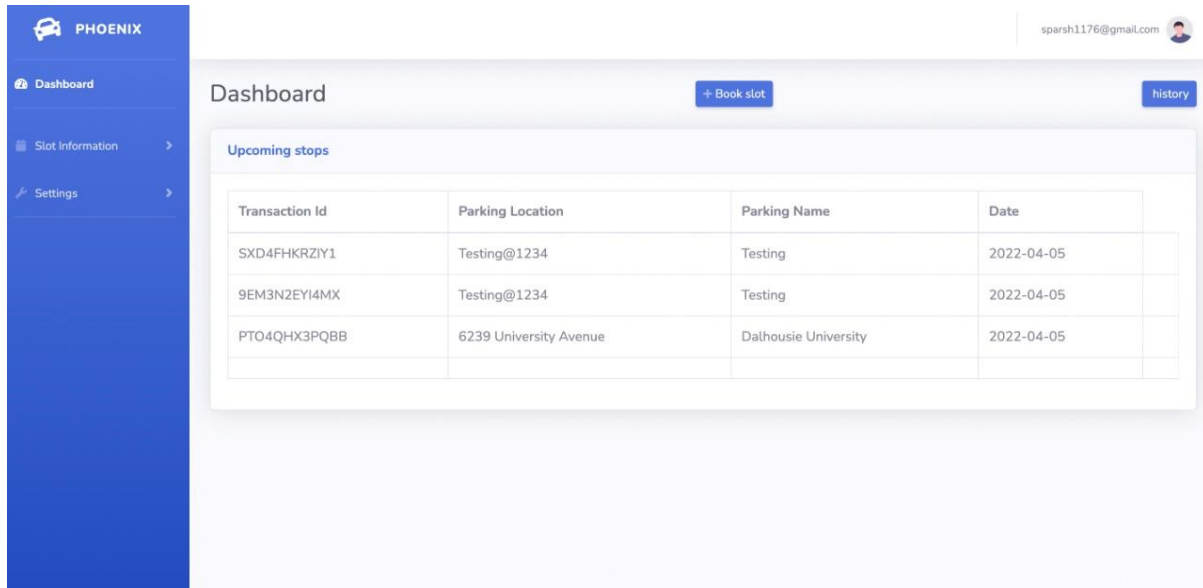
## Screenshot of the Dashboards:



## 5.2 Slot booking

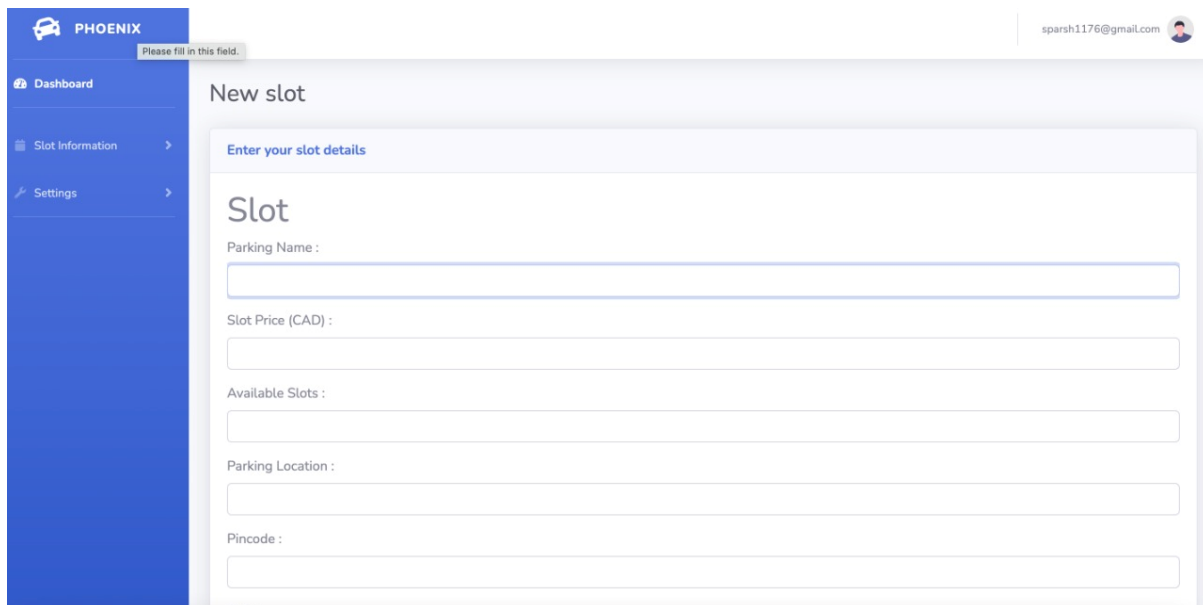


This is one of the main features of this application where user will be able to book a parking spot, add his/her private slot for other users, see the booking history. By adding the slot for users, a user can earn from it. After a user adds the slots for other users, it will reflect to other users in the dashboard. In the booking history, user will be able to see all the previous bookings he made along with the details like booking date, Transaction ID, Parking location and Parking Name. For booking a new slot user has to enter all the details like slot date, email, time for which they need the Parking name, Postal code, location and price. If a user wants to select a slot from the available slots, they can choose a spot of their choice and proceed for the booking. The output of this module is that a user should be able to book a parking spot and should see the booked parking details.



The screenshot shows the Phoenix application's Dashboard. On the left is a blue sidebar with the Phoenix logo and navigation links: Dashboard, Slot Information, and Settings. The main content area is titled 'Dashboard' and includes a '+ Book slot' button and a 'history' button. Below these is a section titled 'Upcoming stops' containing a table with the following data:

Transaction Id	Parking Location	Parking Name	Date
SXD4FHKRZIY1	Testing@1234	Testing	2022-04-05
9EM3N2EYI4MX	Testing@1234	Testing	2022-04-05
PTO4QHXPQBB	6239 University Avenue	Dalhousie University	2022-04-05



The screenshot shows the Phoenix application's 'New slot' form. The sidebar is identical to the previous screenshot. The main content area is titled 'New slot' and includes a 'Please fill in this field.' message. Below this is a section titled 'Enter your slot details' with the following form fields:

- Slot
- Parking Name :
- Slot Price (CAD) :
- Available Slots :
- Parking Location :
- Pincode :

Search for...

spارش1176@gmail.com

Following Slots are available near your location

+ Book New slot

Slots which are available Near by

Parking Name	Parking Location	Available Slots	Price(\$)	Book Slot
Bayers Park	Bayers Road	20	10	<a href="#">Book Slot</a>
Bayers Park	Bayers Road	43	50	<a href="#">Book Slot</a>
Testing	Testing@1234	30	28	<a href="#">Book Slot</a>
ABC	jcber	23	12	<a href="#">Book Slot</a>
Dalhousie University	6239 University Avenue	39	8	<a href="#">Book Slot</a>
Dharmin	1333 South Park Street	9	7	<a href="#">Book Slot</a>

Search for...

spارش1176@gmail.com

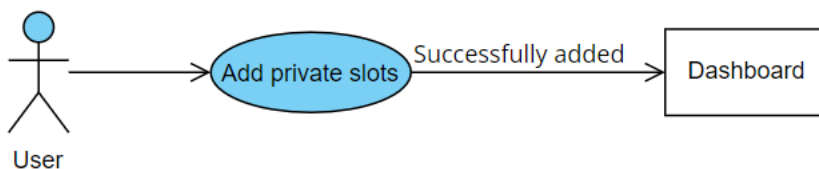
Following Slots are available near your location

+ Book New slot

Slots which are available Near by

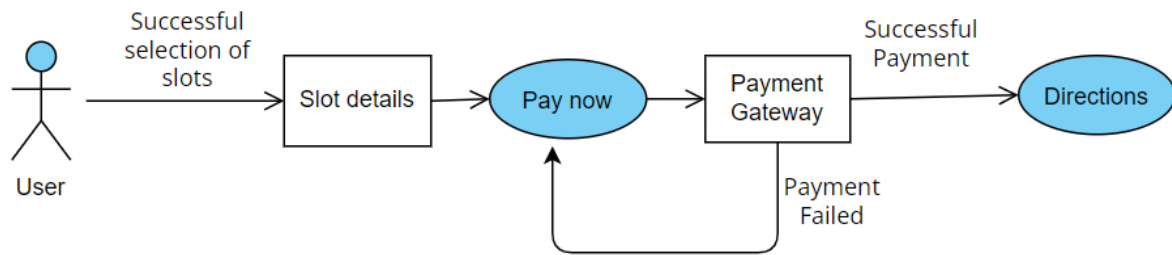
Parking Name	Parking Location	Available Slots	Price(\$)	Book Slot
Bayers Park	Bayers Road	20	10	<a href="#">Book Slot</a>
Bayers Park	Bayers Road	41	50	<a href="#">Book Slot</a>
Testing	Testing@1234	29	28	<a href="#">Book Slot</a>
Dalhousie University	6239 University Avenue	39	8	<a href="#">Book Slot</a>
Dharmin	1333 South Park Street	9	7	<a href="#">Book Slot</a>

### 5.3 User slot addition




This is the unique feature of the application where a user can behave as an admin as well as the customer who will be able to book the parking spot. It is kind of role-based scenario in which if a user has any private spot, he/she can rent that spot and earn from it. So, at first a user has to add his parking details same as how admin does, and this addition will be visible to the users. By doing this a user not only earns from it which will eventually reduce admins efforts in some cases but also, gives more slot options to other users.

## 5.4 Payment



After successfully selecting a parking spot, user has to see the slot details and he should be able to proceed to the payment. We have implemented a secure payment gateway where a user will be able to pay using PayPal for the booked slot based on the price mentioned. After the payment is successful, user will be redirected to the location page where they should be able to see the slot address with an option to view the address on Google maps.




sparsh1176@gmail.com

### Confirm the below Slots

Parking Name	Parking Location	Available Slots	Price(\$)
Testing	Testing@1234	30	28

Pay Now



### Pay with PayPal

Enter your email or mobile number to get started.

Next

or

Pay with a credit or Visa Debit card

[Cancel and return to Test Store](#)

[English](#) | [Français](#)


Hi, John!

Ship to

John Doe  
1 Maire-Victorin, Toronto ON M5A 1E1  
[Change](#)

Pay with


☒

 PayPal balance


Backup: VISA \*\*\*\*7493

☐ Make this my preferred way to pay

☐

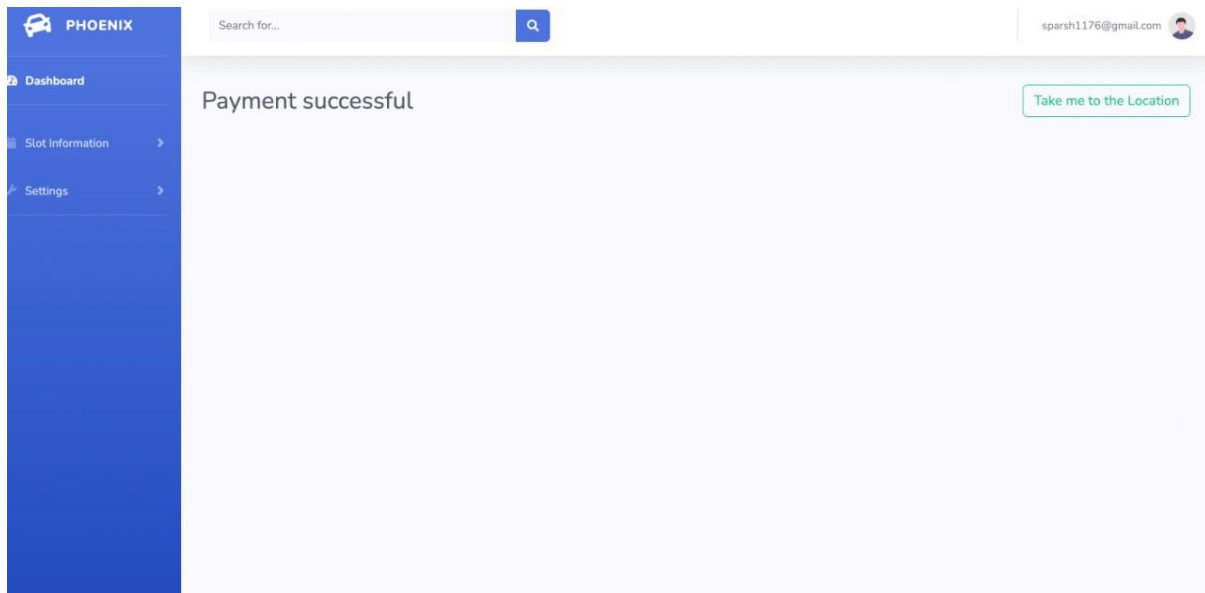
 RBC Royal Bank  
Checking \*\*\*\*6470

☐

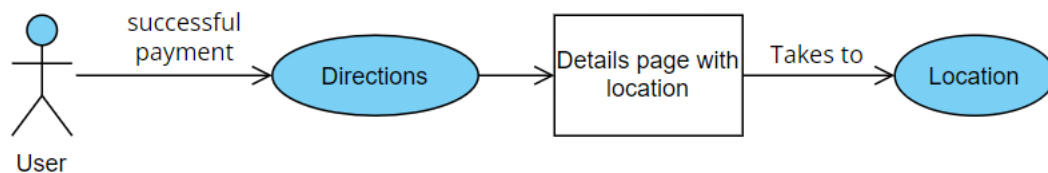
 Visa  
\*\*\*\*7493

[+ Add debit or credit card](#)

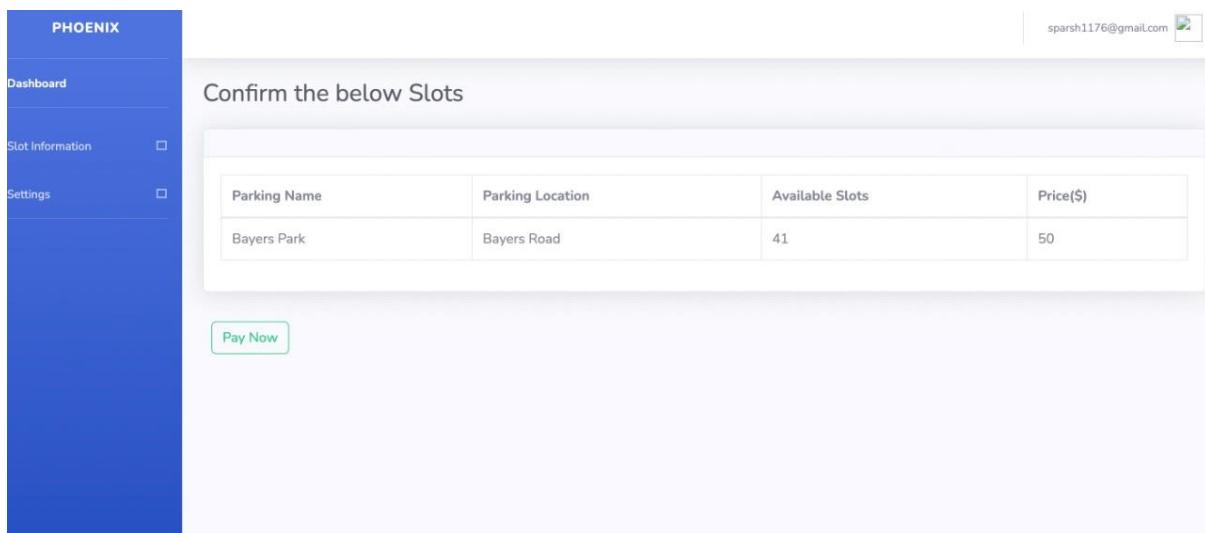
Continue

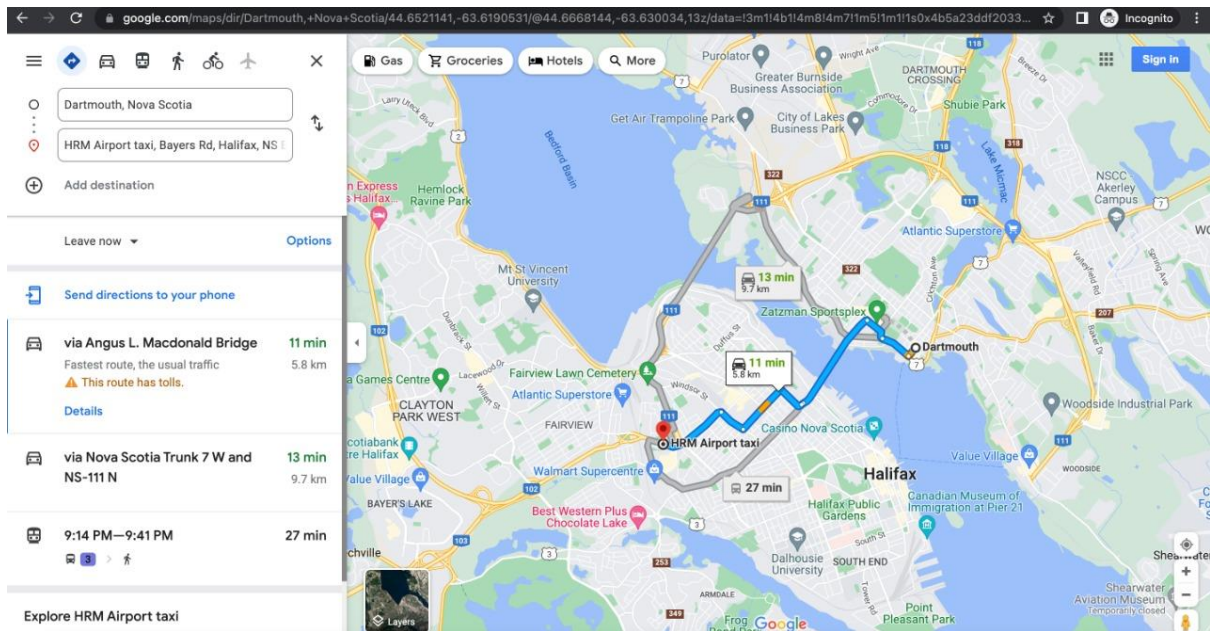


## 5.5 Map Integration



We have integrated a map in our application where a user after successful payment can check the location on Google maps. We have first fetched latitude and longitude from google and this data is passed to the google





## 5.6 Feedback

Users can provide feedback on the whole experience of Phoenix by calling admin. This feature can be found when a user goes to the booking history and in the provide feedback section where call will be directed to the admin of previous bookings.

Dashboard
Slot Information
Settings

Search for...

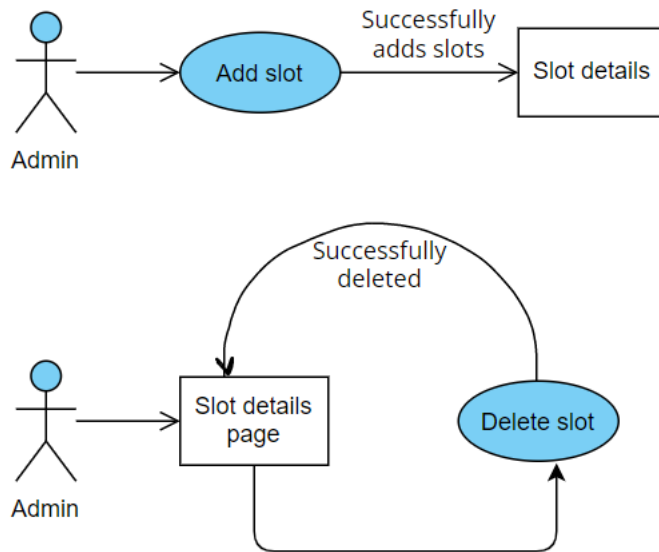
Dashboard

+ Book New slot

Existing Booking

Transaction Id	Parking Location	Parking Name	Date	Provide Feedback
6FFGTW0U720Q	Bayers Road	Bayers Park	2022-04-11	Call us at 123-456-7890 <a href="#">Call Admin</a>
MXJI5KI23ETE	Bayers Road	Bayers Park	2022-04-11	Call us at 123-456-7890 <a href="#">Call Admin</a>
57WONCIGHTGQ	Bayers Road	Bayers Park	2022-04-20	Call us at 123-456-7890 <a href="#">Call Admin</a>
4C2V2VZVAREU	Bayers Road	Bayers Park	2022-04-20	Call us at 123-456-7890 <a href="#">Call Admin</a>
XNGXI6Z9O2IX	Bayers Road	Bayers Park	2022-04-20	Call us at 123-456-7890 <a href="#">Call Admin</a>
SXD4FHKRZIY1	Testing@1234	Testing	2022-04-05	Call us at 123-456-7890 <a href="#">Call Admin</a>
9EM3N2EYI4MX	Testing@1234	Testing	2022-04-05	Call us at 123-456-7890 <a href="#">Call Admin</a>

## 5.7 Admin portal management



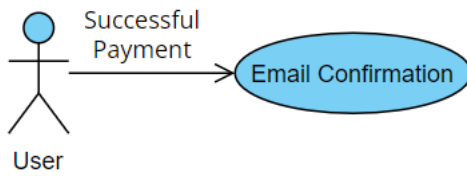
In the admin portal, admin has the authority to add the slots and remove the slots. Here admins can be a parking spot owner or a user who will be providing the spots and earn through it. First admin has to login using his profile and he gets all the access for the application.

The screenshot shows the Phoenix Admin Portal Dashboard. The left sidebar contains navigation links for Dashboard, Phoenix information, and Settings. The main content area displays a table titled 'Total Available slots' with columns for Slot Name, Slot location, Total Available Slots, Total Booked Slots, Slots Price, and Actions. The table lists six slots, each with a 'Delete' button in the Actions column.

Slot Name	Slot location	Total Available Slots	Total Booked Slots	Slots Price	Actions
Bayers Park	Bayers Road	20	0	10	<a href="#">Delete</a>
Bayers Park	Bayers Road	43	7	50	<a href="#">Delete</a>
Testing	Testing@1234	30	0	28	<a href="#">Delete</a>
ABC	jcber	23	0	12	<a href="#">Delete</a>
Dalhousie University	6239 University Avenue	39	1	8	<a href="#">Delete</a>
Dharmin	1333 South Park Street	9	1	7	<a href="#">Delete</a>

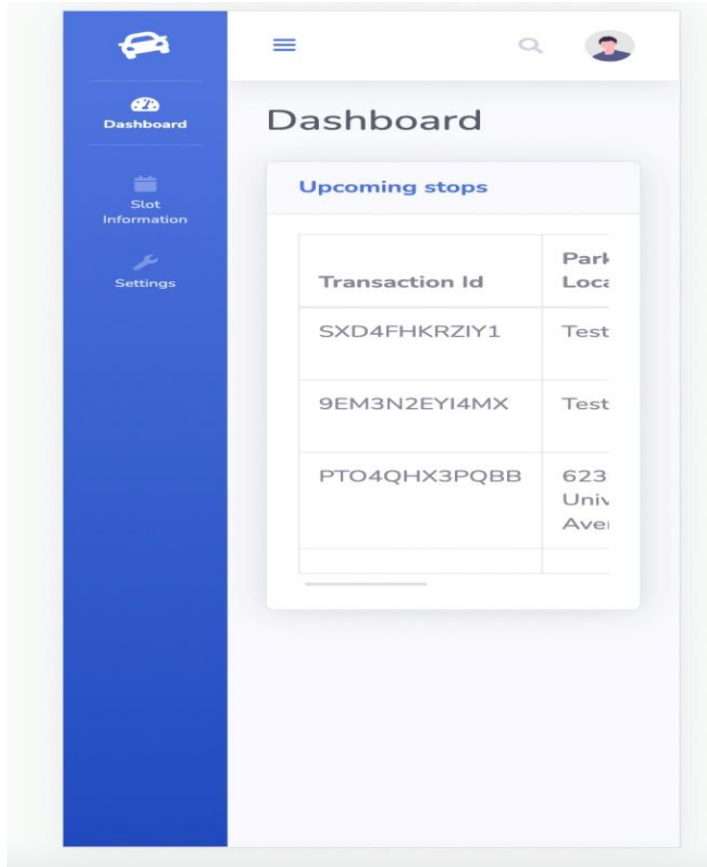


## 5.8 Confirmation Email



This is the extra feature we implemented in the project where the user gets a confirmation email after successfully booking the parking spot. Confirmation email contains Transaction ID, and slot details.

Below is the mobile view of the Dashboard



## **6. SCOPE OF THE PROJECT**

This project can be used by every vehicle owner including car, bike or even a bicycle and reserve a spot beforehand when it is a peak or on a holiday. The end users are the vehicle owners who will be benefitted and the parking spot providers who can be public property owners such as Halifax shopping center owners. Its user-friendly interface and secure payment system makes the application create more impact on the people. User will experience a hassle-free experience by booking a parking spot for the upcoming trips which will eventually reduce the time to get a parking spot without ending up returning home sometimes without a parking spot. Multiple users can use this application and book the slots for the parking who will eventually earn by adding private slots for parking. With map feature, a user can search for the nearest parking spots which reduces the tedious process of checking at the time of parking.

## **7. FUTURE IMPROVEMENTS**

For the future improvements of this project,

1. This application can be built on Android and iOS platforms.
2. Various other payment options can be included which makes the payment gateway more feasible giving more options to the user.
3. Using IOT sensors in the project will automatically detect the available parking slots.

**8. Gitlab URL :** <https://git.cs.dal.ca/courses/2022-winter/csci-5308/group14/-/tree/master>

**9. Heroku Link:** <https://asdc-group-14.herokuapp.com/>

## 10. REFERENCES

- [1] "Serving web content with Spring MVC," Spring.io. [Online]. Available: <https://spring.io/guides/gs/serving-web-content/>. [Accessed: 06-Apr-2022].
- [2] "SB admin 2," Start Bootstrap. [Online]. Available: <https://startbootstrap.com/theme/sb-admin-2>. [Accessed: 06-Apr-2022].
- [3] "Free Use Case Diagram tool," Visual-paradigm.com. [Online]. Available: <https://online.visual-paradigm.com/diagrams/solutions/free-use-case-diagram-tool/>. [Accessed: 06-Apr-2022].
- [4] "Creating a random string with A-Z and 0-9 in Java," Stack Overflow. [Online]. Available: <https://stackoverflow.com/questions/20536566/creating-a-random-string-with-a-z-and-0-9-in-java>. [Accessed: 06-Apr-2022].
- [5] "JUnit – about," Junit.org. [Online]. Available: <https://junit.org/junit4/>. [Accessed: 06-Apr-2022].
- [6] "Mockito framework site," Mockito.org. [Online]. Available: <https://site.mockito.org/>. [Accessed: 06-Apr-2022].
- [7] "PayPal developer," Paypal.com. [Online]. Available: <https://developer.paypal.com/home>. [Accessed: 06-Apr-2022].