Technical Documentation

ZippyPark

Team 2

Samantha Moy
Atmika Ponnusamy
Samantha Cheng
Kylie Chow
Andrew Ko
Parth Patel
Shreya Patel
Nandita Shenoy
Piotr Zakrevski

Table of Contents

Website

The website utilizes Node application written in Node.js.

Server.js

This file starts a server for the Node application and creates an SQL database connection that is utilized in database queries in index.js.

Routes

In the routes folder there is one file, index.js, which receives HTTP requests from and returns data to the client. This file renders the files contained in the pages folder, and also contains RESTful routes to query the SQL database so the client and server can interact.

Views

In the views folder there are two folders, pages and partials. The pages folder contains the EJS code that outlines how the website's pages will look, and the partials folder contains any EJS code that can be used by multiple pages, making it unnecessary to rewrite code for each page. The pages folder includes the following completed files, customers.ejs, dashboard.ejs, reservations.ejx, and statistics.ejs. Customers.ejs is a layout for the customers page and it displays the customers in a list with the following information: first name, last name, barcode, license number, and points. There are also buttons to update the information or delete the information. Dashboard.ejs is a layout for the dashboard page. This displays tiles with quick garage information, including number of open parking posts, the number of reservations for the day, yesterday's revenue, number of customer accounts, and the number of upcoming reservations. The layout also includes the price adjustment and points adjustment modals. Reservations.ejs is a layout for the reservations page where the reservations are displayed in a list with the following information: date, start time, end time, barcode, assigned spot, charge, and rID. There are also buttons to update the information or delete the information. The update button will bring up a modal from edit-reservation.ejs. Statistics.ejs is the layout for the statistics page. This layout displays tabs for different types of charts that display various statistics that are labeled on each tab. Clicking on each tab will expand it to show the corresponding graph.

The partials folder contains EJS code that may be used in multiple webpages or otherwise allows files in the pages folder to appear cleaner and easier to use. It contains the following files: add-customers.ejs, add-reservation.ejs, delete-customer.ejs, delete-reservation.ejs, footer.ejs,

head.ejs, and sidebar.ejs, as well as a folder for graphs used in the statistics page. Add-customer.ejs is a modal for the customers page to add customers. It contains the text fields for the manages to edit and a button to submit the changes to the database. Add-reservation.ejs is a modal for the reservations page to add reservations. It contains text fields for the manager to edit and a button that submits the changes to the database. Delete-customer.ejs is a modal that displays prompts that ask for confirmation of deletion as well as a delete or cancel button. The cancel button closes the modal and the delete button will delete the customer from the database and close the modal. Delete-reservation.ejs is a modal that displays prompts that asks for confirmation of deletion as well as a delete or cancel button. The cancel button closes the modal and the delete button will delete the reservation from the database and then close the modal. Footer.ejs is used to display the footer (Software Engineering Team 2 Spring 2020). Head.ejs is used to link the necessary JS and CSS code. Sidebar.ejs is used to display the sidebar which has navigation to all pages. The files in the graphs folder contain the charts and tables that are then displayed in the accordion in the statistics page.

Android

The android application utilizes Java for the controller code and XML for the view code. The app is built for Andriod SDK version 28 and the minimum version supported is SDK version 24.

Controller Code

The controller code is found by following this path, App → src → main → java → com/example/user → zippypark. Within the code are the following functions.

CreateReservationActivity.java allows the customer to create a new reservation. Which gets saved to the remote database.

CurrentReservationsActivity.java allows the customer to view their currently valid reservations in a scrollable list with details such as date, start time, end time, and price of each reservation.

DeleteProfileActivity.java allows the customer to delete their profile.

DeleteReservationActivtiy.java allows the customer to cancel any selected upcoming reservation.

EditProfileActivity.java allows the customer to edit the details of their profile, including personal information, car information, or contact information.

EditReservationActivity.java allows the customer to edit the details of their reservation, including the date, start time, and end time of the reservation.

HomeMenuActvity.java allows the customer to choose an action after the customer logs in. The customer can view their profile, create a new reservation, view current reservations, or view reservation history.

MainActivity.java allows the customer to login with an existing username and password if they are an existing user, or it allows new customers to create an account.

ProfileActivity.java allows the customer to view the profile details, such as: name, billing info, vehicle info, rewards/points, and various other information.

ReservationHistoryActivity.java allows the customer to view their past reservations in a scrollable list, with details such as date, start time, end time, and the price of each reservation.

SignUpActivity.java allows the customer to create a new account, by prompting them to provide details such as: name, billing info, vehicle info, and various other personal information.

View Code

The other major code for the android application is the view code. Most of the view code is found by following this path, App → src → main → res → layout. Within the code are the following functions.

Activity_main.xml is the layout description for the login activity.

Barcode_dialogue.xml is the layout description for the barcode pop-up dialogue.

Create_reservation.xml is the layout description for the create reservation activity.

Current_reservation.xml is the layout description for the current reservations activity.

Home_menu.xml is the layout description for the home menu activity.

Profile.xml is the layout description for the profile activity.

Res_history.sml is the layout description for the reservation history activity.

Res_list_item.xml is the layout description for displaying a single reservation as an item within a list.

Sign_up.xml is the layout description for the sign up activity.

The other two functions in the view code are found using the following path, App → src → main → res → values. The functions are described as the following.

Colors.xml defines the colors used in the app.

Strings.xml defines the text strings used in the app.

Scanner

The scanner utilizes java for the scanner code and swing GUI for the applet. The executable jar file is built for Java Runtime Environment 8 or higher.

Scanner.java

This file contains all of the program's functional code and consists of 3 functions. The first function is initialize(), which is called by the program to set up the application GUI. The initialize function constructs two frames, the login frame, and the scanner frame. The login frame prompts the user to enter their netID and password and calls the second function, isValid(), to determine whether their credentials are correct. The scanner frame is displayed when the valid credentials are entered and allows barcodes to be inputted, running the third function, process(barcode), on each barcode inputted.

The function isValid(String netID, char[] password) checks whether the credentials are right by attempting to establish an ssh connection with the given credentials

The last function, process(String barcode), will take in a barcode and perform one of the following actions: enter a reservation, exit a reservation, create a walk-in, exit a walk-in, or send an error. If the barcode has an open reservation, the scanner will enter a reservation by simply assigning the user a parking spot. VIP and handicap statuses are checked to find the right parking spot. If the barcode is currently assigned a reservation parking spot, the scanner will exit the reservation by removing the entry in the reservation table and store the entry in the records table. Late reservation fees are calculated using base prices and multipliers in the database in the case the customer has overstayed their reservation. If there are no reservations existing, the scanner will create a walk-in order if there are currently available parking spots. Finally, if the barcode is currently assigned a walk-in parking spot, then it exits the user by removing the entry in the walk-in table and stores the entry in the records table. The walk-in prices are determined using the base price and multipliers defined in the database.

Jsch-0.1.55.jar

This is an external java library that enables the scanner to establish an ssh connection with the Rutgers ECS server.

Mysql-connector-java-8.0.16.jar

This is an external java library that enables the scanner to connect to the database once an ssh connection is established.

Scanner.jar

This is a runnable jar file that contains the fully functional Scanner application.