Contributors: Luke Miller, Harshil Patel, Yufeng Liu, Tommy Walters Brandon Dunlap, Xiang Xing

To run the software, one must import all provided SQL files into a WAMP server. (Windows, Apache, MySQL, PHP). The files provided under HTML,PHP,CSS folder will needs to be added to the “www” folder of the WAMP server. Import the provided eclipse archive into eclipse and start WAMP. You can now access the website and databases. Running the Java program in eclipse will automate the website. (Verify that “Automate Website” at the header of simulation.java is set to true.

All File Names and their Brief Descriptions of Content

1. Reservation\_store.php (Harshil Patel)

This file contains the part of the reservations system that processes regular reservations. It includes html content to display a form to the user and php content for connecting to the reservations database and storing reservations.

2. a. recurring.html b. Recurring.php (Harshil Patel)

These files contain parts of the reservations system that process recurring/subscribing reservations. The first file includes html content to display a form to the user and the second file processes this information in accordance with our reservations database using php scripts.

3. Businesshtml.php (Harshil Patel)

This file contains the section of the reservations system that lets business account users make reservations via a form. PHP scripts are also built in to the file to once again communicate with the reservations database.

4. Cancellation.php (Harshil Patel)

This last section of the reservations system lets users cancel their reservation by inputting their confirmation number and again it includes a section of php code that manages the user form-input with the reservations database.

NOTE: Files 1 - 4 were updated to include added security so the system only lets a user make a reservation with his/her, unique information only and of course only after successfully logging in with the correct credentials. The updated versions are in the folder titled “Updated Reservation Scripts” and their corresponding updated SQL database is in the folder titled “Updated Reservation SQL.” Please use these folders for unit testing of this reservations subsystem, a readme file is included in the scripts folder stated earlier on login info. Also, Files 2a and 2b were combined into one php file titled “recurring.php” for ease of accessibility.

5. Simulation.java (Luke Miller)

This is the parent/main class for the software simulation of Park-A-lot. In addition to providing unit test funtionality, simulation.java is the primary software loop that invokes all of class calls. Example processes in Simulation.java include initializing users, vehicles and generate new reservations.

6. Supervisory.java (Luke Miller)

This is the child process of Simulation.java, run in another thread to check for arrivals/departures. Supervisory.java uses a staticly defined clock tick in simulation.java to determine cycle time between checks. When the reservation arrival time or departure time has been exceeded, supervisory.java uses randomized varaibles to determine if a customer arrives or departs late.

7. User.java, BusinessUser.java Vehicle.java, Reservation.java, Policy.java, PaymentMethod.java(Luke Miller)

These object classes represent components within the entire Park-A-Lot system. For each run of the simulation, these objects are instantiated and have their fields populated to values specific to each individual. The User object represents the users of the system. Each user has a name and other personal identifying information. The reservation object is used to identify an active reservation within the system and contains information for who owns the reservation/when it will occur. Each instance of reservation object can be moved around between Reserved, Parked, Late or Overstay ArrayLists depending upon their statuses. The policy object represents a garage policy that applies during a particular time. All of these objects are initialized in Simulation.java but actively check and referenced in Supervisory.java.

8. AccessWebite.java, UDPSend.java, CommandRunner.java (Luke Miller)

These classes are custom interfaces designed to integrate the simulation with other code including the website, arrival and departure. AccessWebsite.java uses a library called WebSpec to automate any URL much like javascript. This class is used to populate the Park-A-Lot databse using objects generated through the simulation. UDPSend.java is used to send UDP packets over the network to a particular IP address. This is currently used to interface the Park-A-Lot departure code with the simulation. CommandRunner.java is used to run Windows system commands from within java. This class has been depreciated and is no longer in use.

9. Garage.java (Luke Miller)

Garage.java serves as a container for the enitre simulation. This static object contains many data structures which house many of the objects mentioned above. Although Garage.java does not contain many methods that get invoked during runtime, it serves at an important location to house multi-threaded protected information between running threads.

10. homepage.html/login.php (Yufeng Liu)

These two files completes the functions which help user to login. The algorithm is using the email address the user typed in to find the matched password in the databass. Then if the matched password is as same as the one the user typed, it will allow user to login. Otherwise the system will deny the access.

11.BusinessRegisteration.html/PersonalRegisteration.html/businessregistration.php/businessregistration.php(Yufeng Liu)

These files completes the registration function. The algorithm is inserting the data the user typed in into the database. Since i uses the email address as the primary key of the database table logininfo, company and user. It won’t allow two same username exist in the same time.

12.businessprofile.php/profile.php/error.php(Yufeng Liu)

These files is mainly about user interfaces. It helps user to access other utility pages. In addition, if a user try to access this without login, the access will be denied and directed to error.php

13.linkCompany.html/link.php(Yufeng Liu)

Users have to use a link password to link to a business account. The algorithm is similar as the login.php. Checking if the information the user type in matches with the data in the database.

14.PaymentMethod.php/VehicleInfo.php(Yufeng Liu)

Although these two pages complete the distinct functions, the algorithm inside them is very similar.

The website will check if there is any payments/cars link to this account(using the username variable under the vehicleinfo and payment datatable). If there isn’t any payments and vehicles have been founded, it will show “You don’t have cars/payments yet.” When user links to a business account and business authorize him/her, the user is able to see extra cars information from business account and use that to make the reservation. Otherwise it will only print out the user’s own vehicle information. The deletion function is quite straightforward, it just delete the selected information from the database. If user default a payment/car, it will show the sign “(default)” beside the information. And this information will be automatically used when they make the reservation.

15. ProfileEditting.php/ResetPassword.php (Yufeng Liu)

There two functions just help user to change and update their exist data in the database.

16.rates.php(Tommy Walters)

This file brings the manager to a separate page for rates. The page will pull from the rates database and display the contents to the page. There are then a few buttons that allow the manager to either go back to the homepage, add new rates, or delete current rates.

17.rateadd.php(Tommy Walters)

This file brings the manager to a separate page from rates, which allows the manager to add new rates based on a form. The form asks for specific details and then the manager clicks submit and the info is sent to the database.

18.ratedelete.php(Tommy Walters)

This file brings the manager to a separate page from rates, which allows the manager to delete current rates based on a form. The form asks for specific details and then the manager clicks submit and the current rate is deleted from the database.

19.Policy\_Decision.php(Tommy Walters)

This file brings the manager to a separate page for all of the garage policies. The page will pull from the policies database and display the contents to the page. There are then a few buttons that allow the manager to either go back to the homepage, add a new policy, or delete a current policy.

20.policyadd.php(Tommy Walters)

This file brings the manager to a separate page from Policy\_Decision, which allows the manager to add new policies based on a form. The form asks for specific details and then the manager clicks submit and the info is sent to the database.

21.policydelete.php(Tommy Walters)

This file brings the manager to a separate page from Policy\_Decision, which allows the manager to delete a current policy based on a form. The form asks for specific details and then the manager clicks submit and the current policy is deleted from the database.

22.specials.php(Tommy Walters)

This file brings the manager to a separate page for specials. The page will pull from the specials database and display the contents to the page. There are then a few buttons that allow the manager to either go back to the homepage, add new specials, or delete current specials.

23.specialsadd.php(Tommy Walters)

This file brings the manager to a separate page from specials, which allows the manager to add new specials based on a form. The form asks for specific details and then the manager clicks submit and the info is sent to the database.

24.specialsdelete.php(Tommy Walters)

This file brings the manager to a separate page from specials, which allows the manager to delete current specials based on a form. The form asks for specific details and then the manager clicks submit and the current special is deleted from the database.

25.hours.php(Tommy Walters)

This file brings the manager to a separate page for hours of operation. The page will pull from the hours database and display the contents to the page. There are then a few buttons that allow the manager to either go back to the homepage, add new hours, or delete current hours.

26.hoursadd.php(Tommy Walters)

This file brings the manager to a separate page from hours, which allows the manager to add new hours based on a form. The form asks for specific details and then the manager clicks submit and the info is sent to the database.

27.hoursdelete.php(Tommy Walters)

This file brings the manager to a separate page from hours, which allows the manager to delete current hours based on a form. The form asks for specific details and then the manager clicks submit and the current hours of operation are deleted from the database.

28. User\_list.php(Yufeng Liu)

Business accounts could see the personal account which link to them. If business accounts deny the user, that user will not be able to use the business vehicle. And that user will be put into a database table called deny\_list which allow business\_account to check whether user has been denied.

29. Main.java (Brandon Dunlap)  
This is the main file of the application. It builds the application initial stage and collects and verifies the user’s input. Upon pressing the “Submit” button, Main calls makeChart to produce the next stage containing the output for the user’s request.   
30. makeChart.java (Brandon Dunlap)  
Produces the chart requested by the user. If the user selected a month it will display the chart containing the number of reservations for that month. If the user doesn’t choose a month, only a year, then makeChart will produce a chart showing the number of reservations that occurred each month of that year.   
31. Reservation.java (Brandon Dunlap)  
The class file defining reservations including; the reservation ID number, the date the reservation was made for, and the time it was made.   
32. reservationDataBase.java (Brandon Dunlap)  
This file generates a faux database of reservation for testing purposes. Every time the application is executed, this file will randomly generate a large number of reservations over a 10 year period.   
33. Date.java(Brandon Dunlap)  
This class file defines the date information for reservations.  
34. Time.java (Brandon Dunlap)  
This class file defines the date information for reservations.

35 connect.cpp(Xiang Xing)

This file is the main program of the departure system. It will launch the database, calculate the fee and send the confirmation email to the customer

36. sender.cpp(Xiang Xing)

A fake simulator that I created for test use, It gives out license plate number as udp packages.