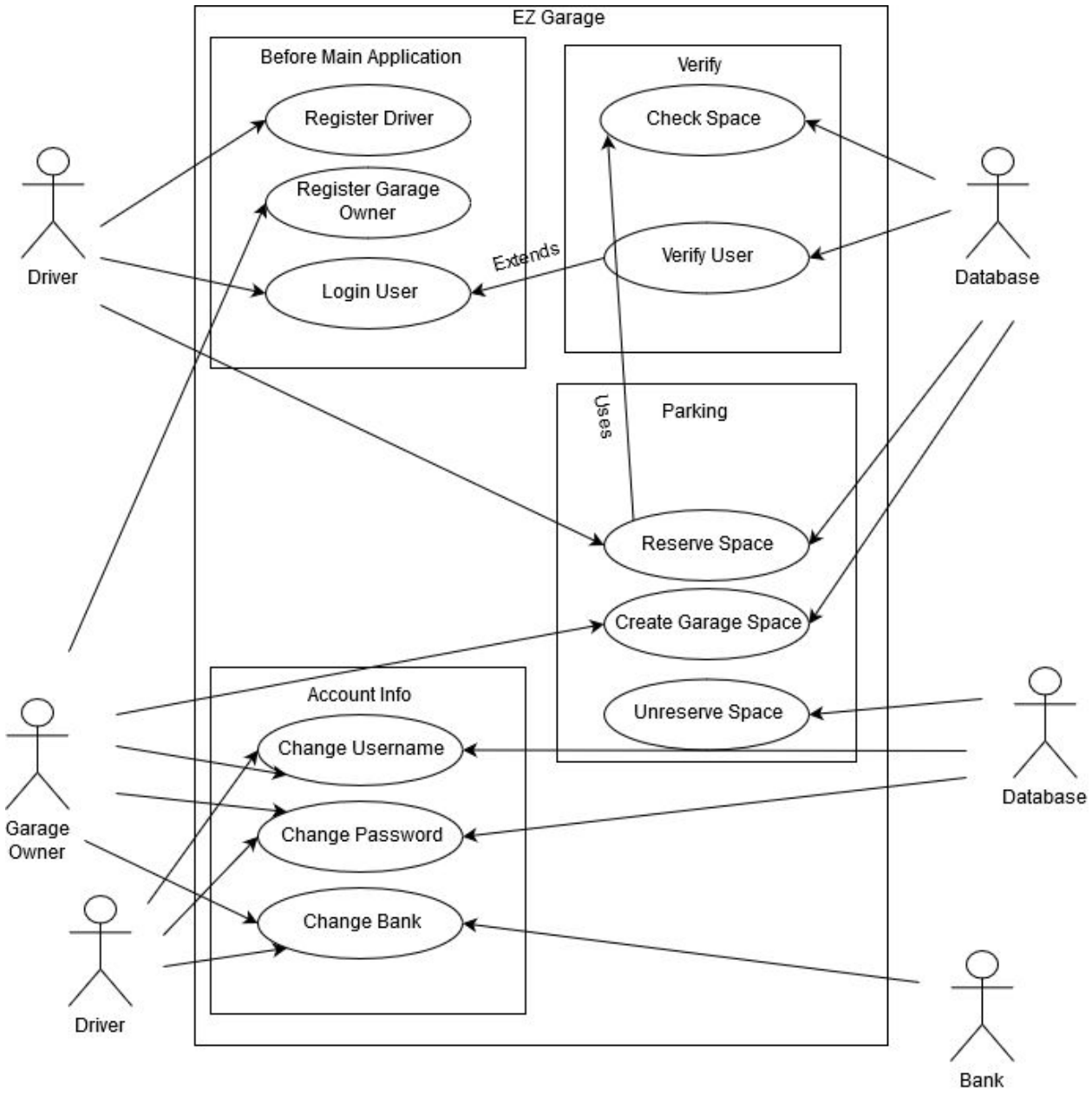
CS 1530 - OOA and OOD

Group 4
Justin Kramer
Mike Scales
Matt Choi
Kyle Thorpe
Jared Frank

OOA:

1. System Overview

Our use case diagram is based off the functions defined in our DFD(data flow diagram). We define each actor which will be utilizing our system, which in this case will be drivers, garage owners, banks, and our database that provides data. These actors interact with the processes to produce results in our program.

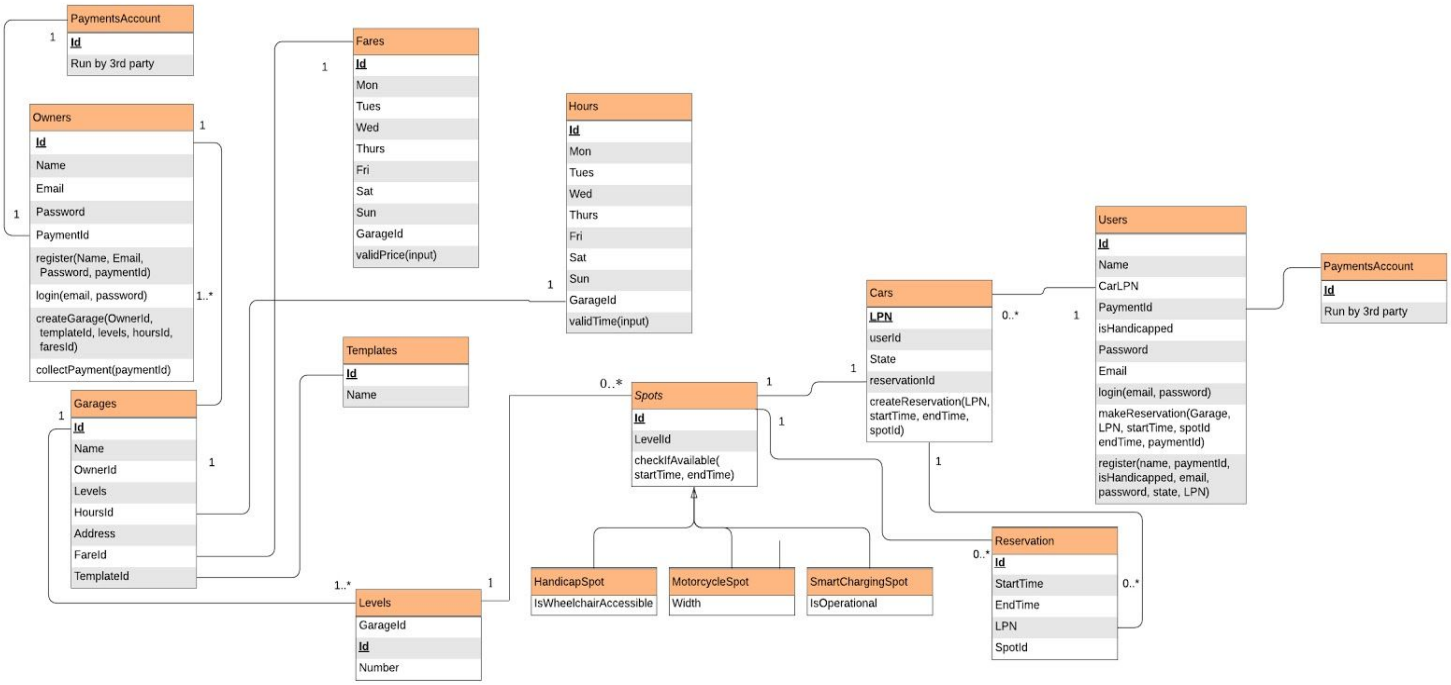


2. The Class Model

2.1. The Classes

- Owners(Id, Name)
- Fares(Id, Mon, Tues, Wed, Thurs, Fri, Sat, Sun, GarageId)
- Hours(Id, Mon, Tues, Wed, Thurs, Fri, Sat, Sun, GarageId)
- Garages(Id, Name, OwnerId, Levels, HoursId, Address, FareId, TemplateId)
- Templates(Id, Name)
- Levels(GarageId, Id, Number, ProjectId)
- Spots (Id, isMotorSpot, isHandicapped, isSmartCharger, LPN)
- Cars(LPN, userId, State, SpotId)
- Users(Id, Name, CarLPN, PaymentId, isHandicapped, Password, Email)

2.2. The Class Diagram



3. The Dynamic Model

3.1. The Scenarios

3.1.1 The Client Side Login Scenario

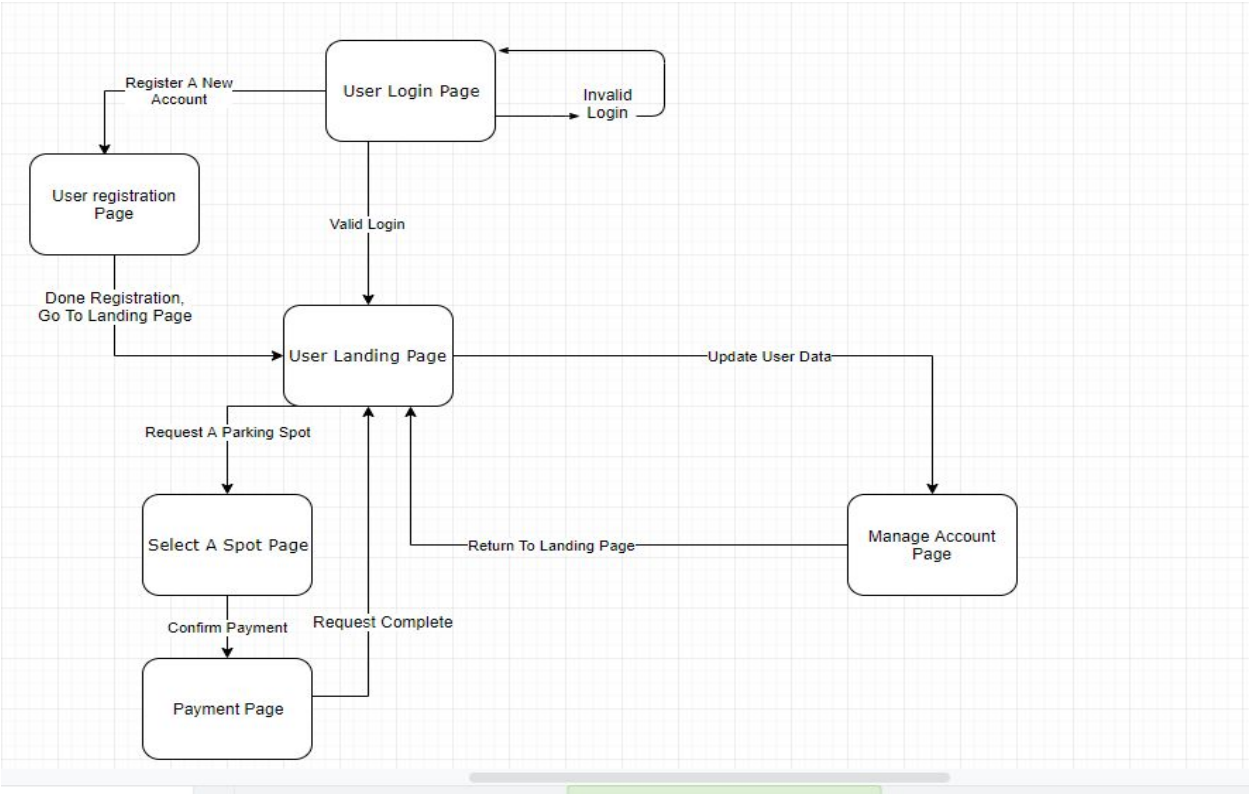
- The User Logs in
- The User Login Is Invalid
- The User Requests a Parking Spot
- The User Requests to Manage Account
- The User Registers an account
- The User Pays

3.1.2 The Owner Side Login Scenario

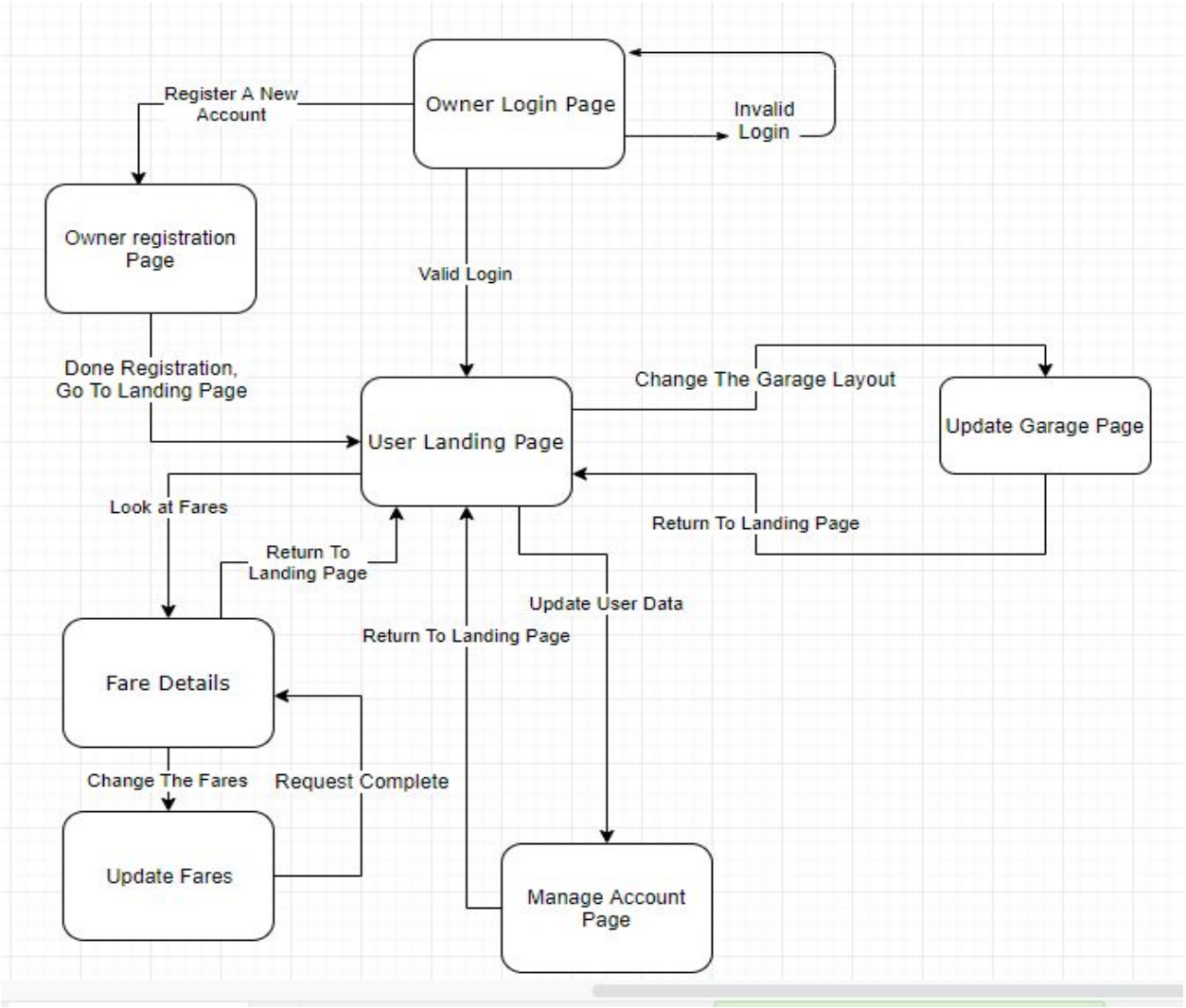
- The Owner Logs into their account
- The Owner Login Is Invalid
- The Owner Registers an account
- The Owner updates parking fares
- The Owner Updates the Garage Data
- The Owner Manages Their Account

3.2. The State Diagrams

3.1.1 The Client Side Login Diagram

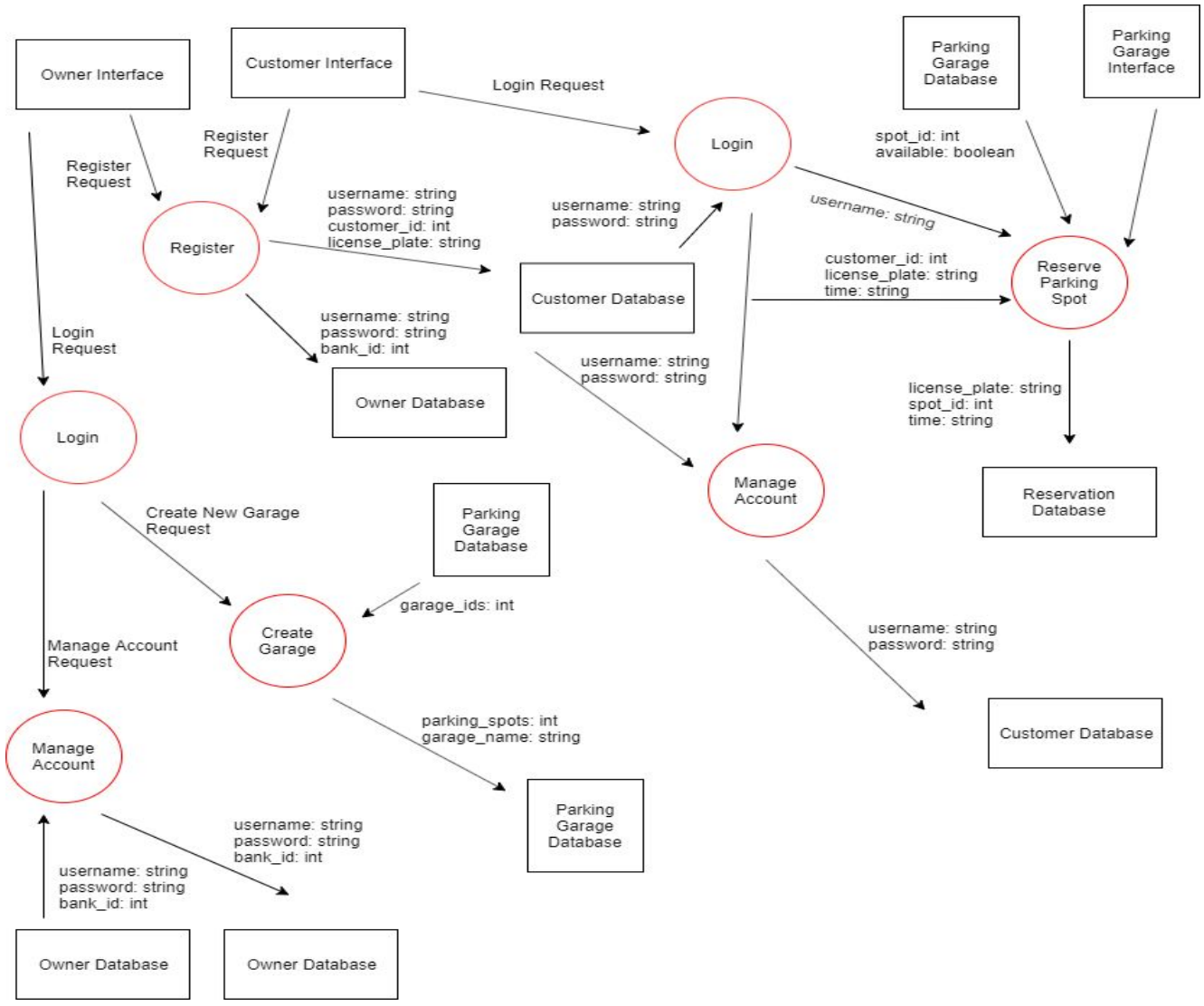


3.1.2 The Owner Side Login Diagram



4. The Functional Model

Our functional model takes the layout from our DFD and expands it into something that flows from process to process. In addition, our functional model displays how customers and owners interface with our application to move through steps of processes. This shows a procedural order to our application.



OOD:

1.

   Module Name: confirm_invoice_paid
   Module Type: Method
   Return Type: Boolean
   Input arguments: Invoice information from 3rd party software
   Output arguments: Boolean value of whether or not the
   Error messages: None
   Files accessed: None
   Files changed: None
   Modules called: Payment Processor
   Narrative: This method will confirm to the customer that the invoice has been paid

2.

   Module Name: confirm_spot
   Module Type: Method
   Return Type: Boolean
   Input arguments: Spot Selected
   Output arguments: Boolean value of if the spot is currently open to be reserved
   Error messages:
   Files accessed: Database entries that contain spot reservation data
   Files changed: Parking Garage Database
   Modules called: Reference
   Narrative: This module will take the input of what spot the user wants the reserve, check the corresponding spot in the database, and then return if the spot is open or not

3.

   Module Name: customer_confirm_login
   Module Type: Method

Return Type: Boolean
Input arguments: email address, password
Output arguments: boolean value of if
Error messages: User information is invalid
Files accessed: Database files related to email address and password
Files changed: None
Modules called: User Database
Narrative: This module takes in the email address and password input from the user's form and then compares it to the database entries to see if it is valid

4.

Module Name: customer_login
Module Type: method
Return Type: Void
Input arguments: User's email address and password
Output arguments: None
Error messages: None
Files accessed: Database entries involving user email and password
Files changed: None
Modules called: User Database
Narrative: This module will move the user forward in the login process after their login has already been confirmed within the customer_confirm_login module

5.

Module Name: customer_payment_info
Module Type: Method
Return Type: Void
Input arguments: User's credit card information as well address
Output arguments: None
Error messages: Error could not send payment information to payment processor
Files accessed: None

Files changed: None
Modules called: Collect Payment
Narrative: This module will send the user's payment information to the third party payment system that we will be using in order to charge and invoice users

6.

Module Name: customer_register
Module Type: Method
Return Type: Void
Input arguments: Customer name, username, password, and email
Output arguments: None
Error messages: Error registering Customer
Files accessed: None
Files changed: User Database Entries
Modules called: User Database
Narrative: This module will take a customers information to begin the registration process for creating a new account within the system

7.

Module Name: customer_registration_check
Module Type: Method
Return Type: Boolean
Input arguments: Customer name, email address, and password
Output arguments: Boolean value relating to whether or not the customer was successfully registered for a new account
Error messages: Error could not register new customer account
Files accessed: User Database
Files changed: None
Modules called: Register
Narrative: This module will confirm that the user has successfully been registered into the user database

8.

Module Name: garage_processor_add

Module Type: Method
Return Type: None
Input arguments: Garage information
Output arguments: None
Error messages: Error could not add garage to list
Files accessed: Garage Database
Files changed: Garage Database
Modules called: Garage Processor
Narrative: This module will send the garage information to the garage processor module to then add it to the garage database

9.

Module Name: garage_processor_check
Module Type: Method
Return Type: Boolean
Input arguments: Garage Information
Output arguments: Boolean value of whether or not the garage was added to the database
Error messages: Error could not add garage to list
Files accessed: Garage Database
Files changed: Garage Database
Modules called: Garage Processor
Narrative: This module will check to see if the garage database has successfully been updated

10.

Module Name: invalid_spot
Module Type: method
Return Type: boolean
Input arguments: Parking space information
Output arguments: Boolean value if the parking space is invalid
Error messages: None
Files accessed: Database of valid/invalid spaces
Files changed: Database of valid/invalid spaces

Modules called: Reservation processor
Narrative: Method us used to check that a space is invalid during reservation

11.

Module Name: make_owner_payment
Module Type: method
Return Type: boolean
Input arguments: Payment amount, payment information
Output arguments: Boolean value if payment was successful
Error messages: None
Files accessed: None
Files changed: None
Modules called: Payment Processor
Narrative: Send the owner the payment amount

12.

Module Name: order_parking_spot
Module Type: method
Return Type: void
Input arguments: parking space information
Output arguments: None
Error messages: If spot was successfully ordered
Files accessed: Garage database
Files changed: Garage database
Modules called: Reservation Processor
Narrative: Customer can order a parking space

13.

Module Name: owner_bank_info
Module Type: method
Return Type: boolean
Input arguments: bank info
Output arguments: Boolean if info is valid

Error messages: If the information is not valid
Files accessed: User Database
Files changed: None
Modules called: Payment Processor
Narrative: Sends bank info to payment processor

14.

Module Name: owner_confirm_login
Module Type: method
Return Type: void
Input arguments: Email, password
Output arguments: None
Error messages: If the login information was invalid
Files accessed: User Database
Files changed: None
Modules called: User Database
Narrative: Method that allows an owner to attempt a log in. If unsuccessful, the user will be able to attempt again

15.

Module Name: owner_garage_creation
Module Type: method
Return Type: boolean
Input arguments: garage information (floors, spaces, etc.)
Output arguments: None
Error messages: If the processor is unable to create the garage
Files accessed: None
Files changed: None
Modules called: Garage Processor
Narrative: Method allowing owner to

16.

Module Name: owner_login
Module Type: method

Return Type: boolean
Input arguments: Email and password
Output arguments: Boolean indicating successful/failed login
Error messages: Message indicating no contact with
Files accessed: User Database
Files changed: None
Modules called: None
Narrative: Method used to log in to the system

17.

Module Name: owner_register
Module Type: method
Return Type: boolean
Input arguments: Email, password, password confirm, license plate information
Output arguments: boolean
Error messages: Message indicating user already exists, message for non-matching confirmation of password
Files accessed: User Database
Files changed: User Database
Modules called: owner_registration_check
Narrative: Used to register a new owner in the database

18.

Module Name: owner_registration_check
Module Type: method
Return Type: boolean
Input arguments: Email, password, password confirm, license plate information
Output arguments: boolean indicating validity
Error messages: None
Files accessed: User DB
Files changed: None
Modules called: None

Narrative: Used to check whether a registration was valid, returns a boolean to indicate

19.

Module Name: receive_invoice_amount
Module Type: method
Return Type: Double
Input arguments: None
Output arguments: Double in the amount of the payment
Error messages: None
Files accessed: Invoice DB
Files changed: None
Modules called: Payment Processor
Narrative: Method that receives the payment amount from the payment processor. Allows a user to pay the correct amount

20.

Module Name: send_spot_confirmation
Module Type: method
Return Type: boolean
Input arguments: parking space information
Output arguments: Boolean indicating whether spot is available
Error messages: None
Files accessed: Garage DB
Files changed: Garage DB
Modules called: Garage DB
Narrative: Method that confirms a spot is available for reservation