Fit-A-Lot

Group 5

Report 2

Group Number 5:

Matthew Brazza	mattbrazza@gmail.com
Justin Cruz	jaa.cruz16@gmail.com
Sheldon Wong	sheldonwong@gmail.com
Parth Patel	psp022@gmail.com
Sean Wang	seanvwang@gmail.com

Instructor: Prof. Ivan Marsic

URLs:
https://sites.google.com/site/parkinggaragesp13/home
https://se1.engr.rutgers.edu/~13group5

Revision History:

Final	17th March 2013 via Sheldon Wong's Sakai Dropbox

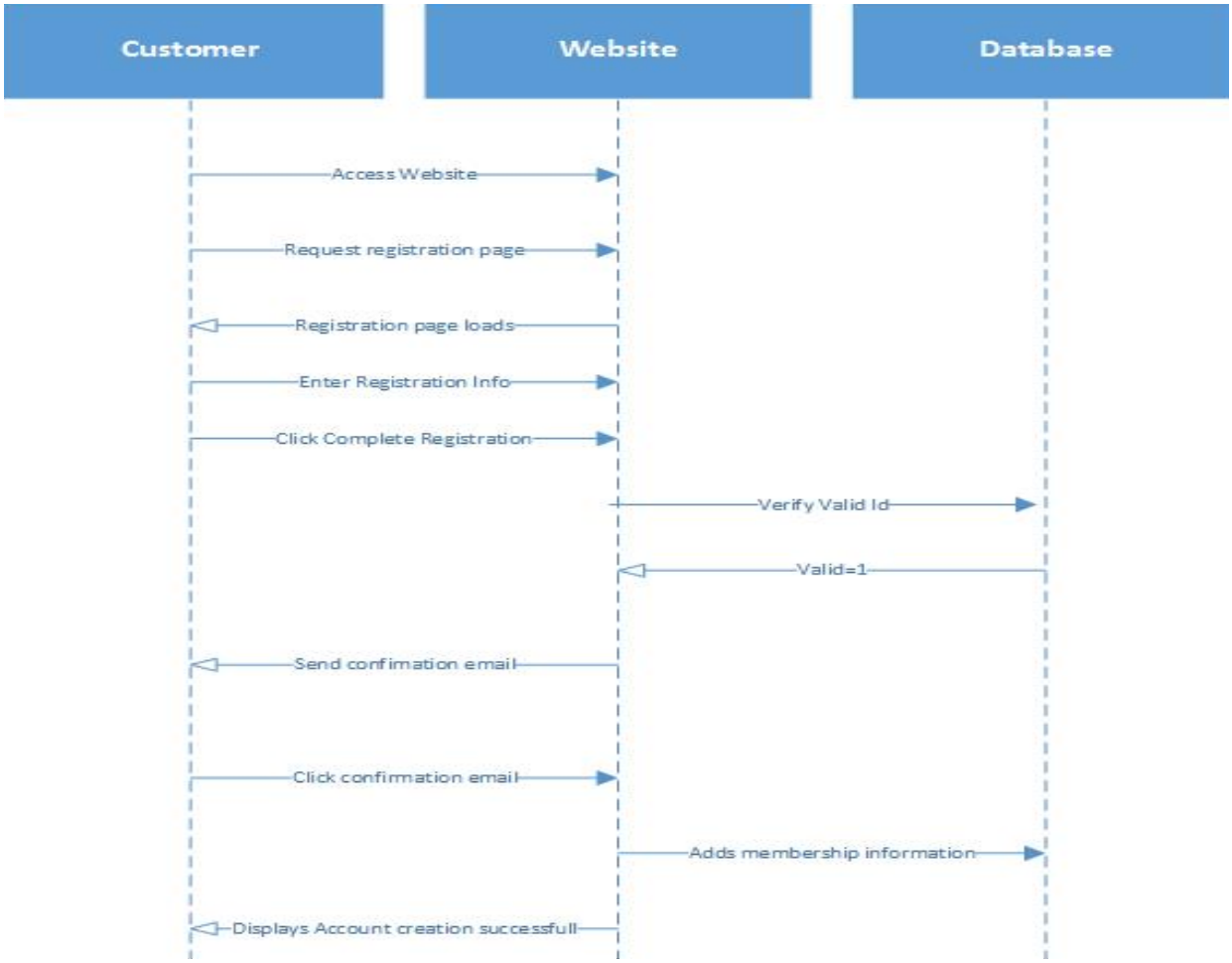
Distribution of Work

	Justin Cruz	Parth Patel	Matthew Brazza	Sean Wang	Sheldon Wong
Group Information			100%		
Distribution of Work			100%		
Table of Contents			100%		
Interaction Diagrams	33%	33%		33%	
Interaction Diagram Comments/Explanations					100%
Class Diagrams and Interaction Specifications		50%	50%		
Traceability Matrix					100%
Architectural Styles	50%			50%	
Identifying Subsystems	100%				
Mapping Subsystems to Hardware	100%				
Persistent Data Structures				100%	
Global Control Flow	100%				
Hardware Requirements	100%				
Algorithms				100%	
Data Structures			100%		
User Interface Design and Implementation		50%			50%
Design of Tests	100%				
Project Management and Plan of Work			10%	90%	

Table of Content

Interaction Diagrams

Online Registration



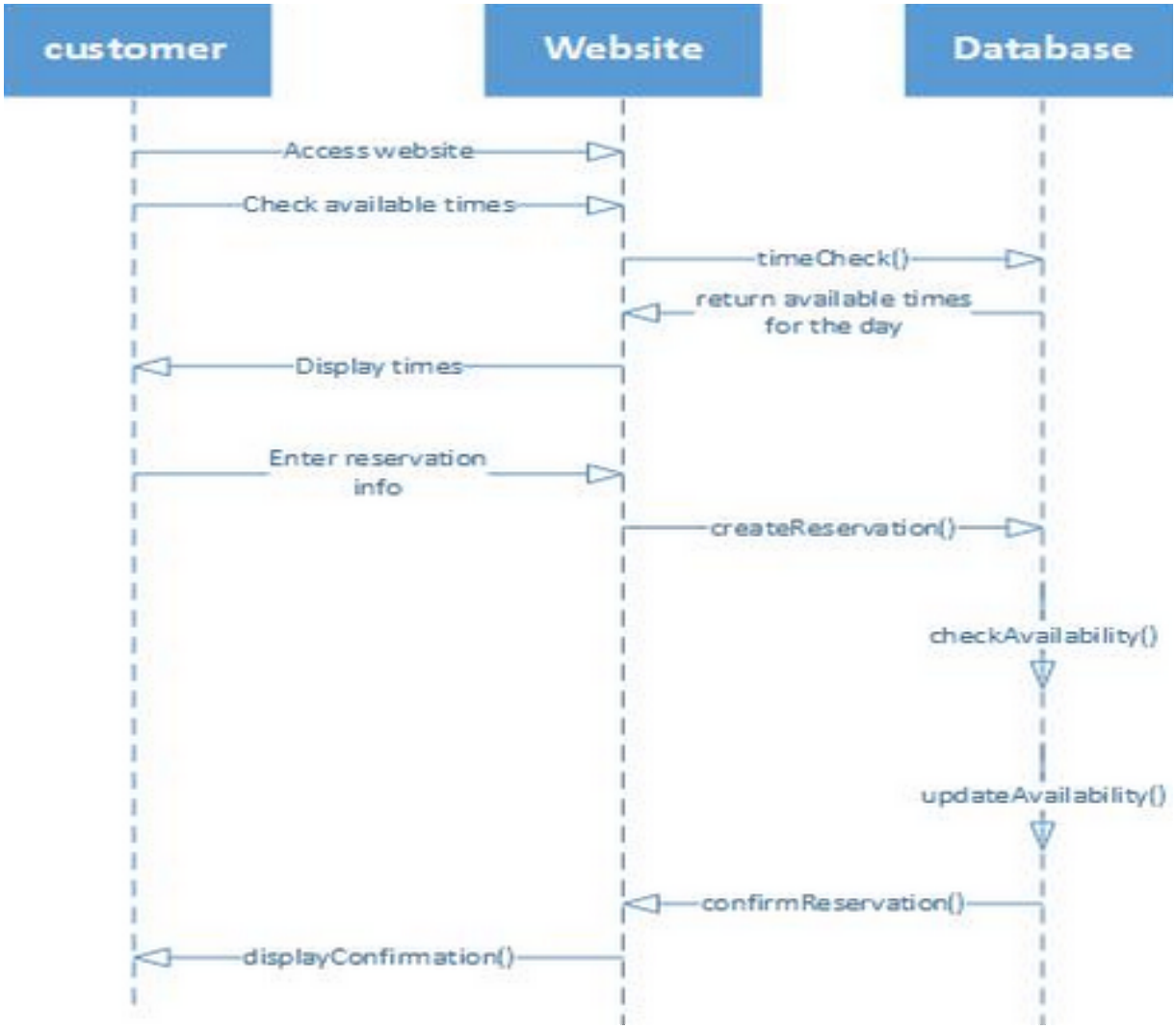
Customer ⇔ Website

- C → W: The customer access the website to enter the registration information and then interacts with the website one more time by entering in confirmation email.
- W → C: The website sends the confirmation email to the customer when all registration information is entered to verify the validity of the person and once verified, displays a "Creation Successful" to customer.

Website ⇔ Database

- W → D: The website will verify the validity of the entered user identification by checking it with database records and once confirmed, adds the new account to the database.
- D → W: Database sends the result of the identification check back to the website.

Online Reservation



Customer ⇔ Website

- C → W: The customer uses the website to check available times and input reservation information.
- W → C: The website displays the available times and confirms the reservation to the customer

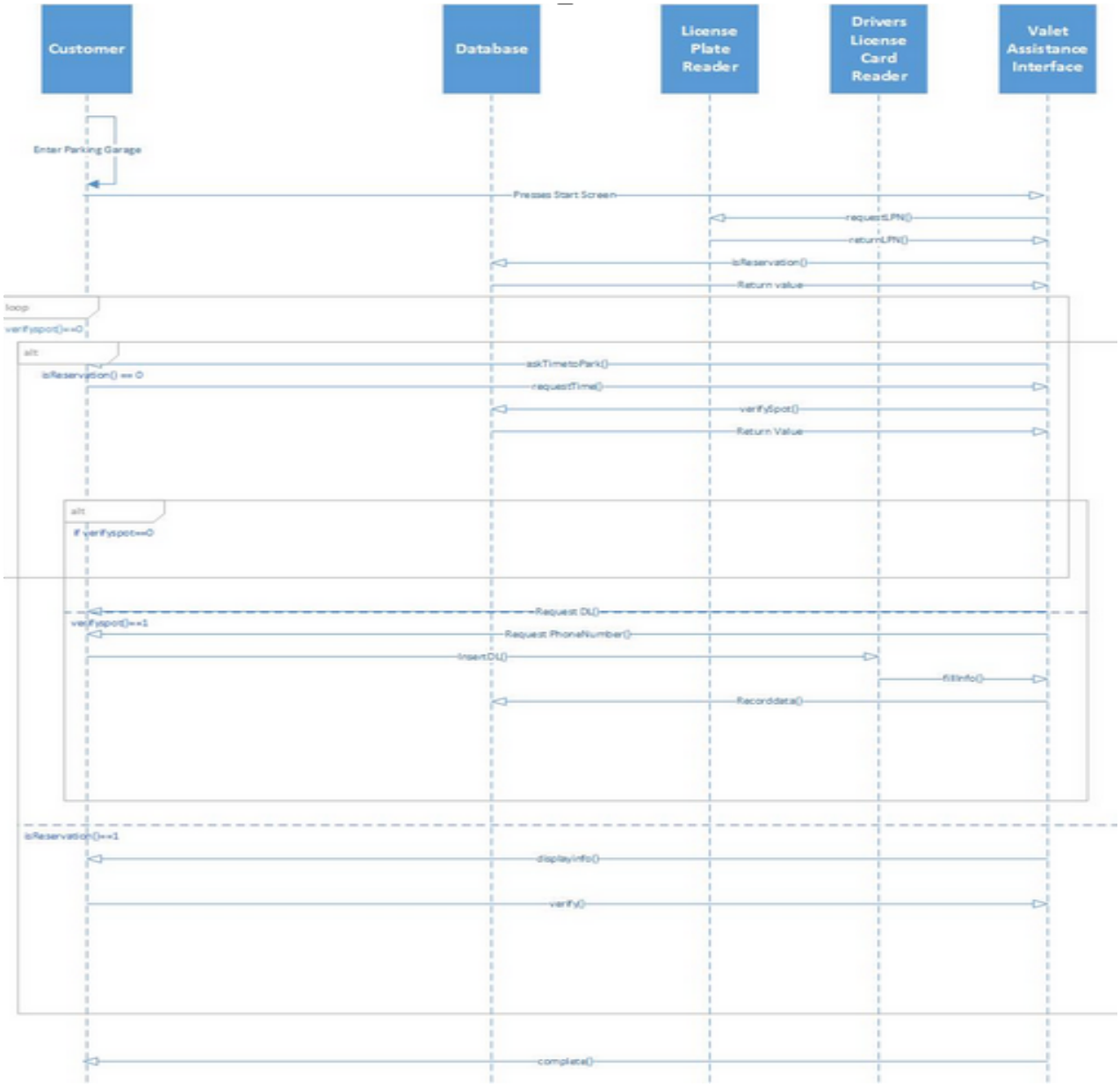
Website ⇔ Database

- W → D: The website requests from the database available parking times and stores newly created reservations.
- D → W: The database returns available times upon request from the website and confirms the reservation after it's been verified.

Database ⇔ Database

- D → D: The database upon being creating a new reservation, checks itself to see whether or not the spot is available and update the status of the parking spot.

Entering the Garage



Customer ⇔ Valet Assistant Interface

- C → VAI: The customer starts the registration verification process by pressing start and when they verify the information displayed by the interface when found..
- VAI → C: The valet assistance interface interacts with the customer in requesting the estimated time to park and phone number when they don't have a reservation and displaying the customers reservation information after they successfully make one. The interface will also complete the process once the customer verifies the information.

License Plate Reader ⇔ Valet Assistant Interface

- VAI → LPR: The valet assistance interface requests the license plate number from the reader when the car is at the checkpoint.
- LPR → VAI: The LPR sends the requested license plate number to the valet

assistance interface.

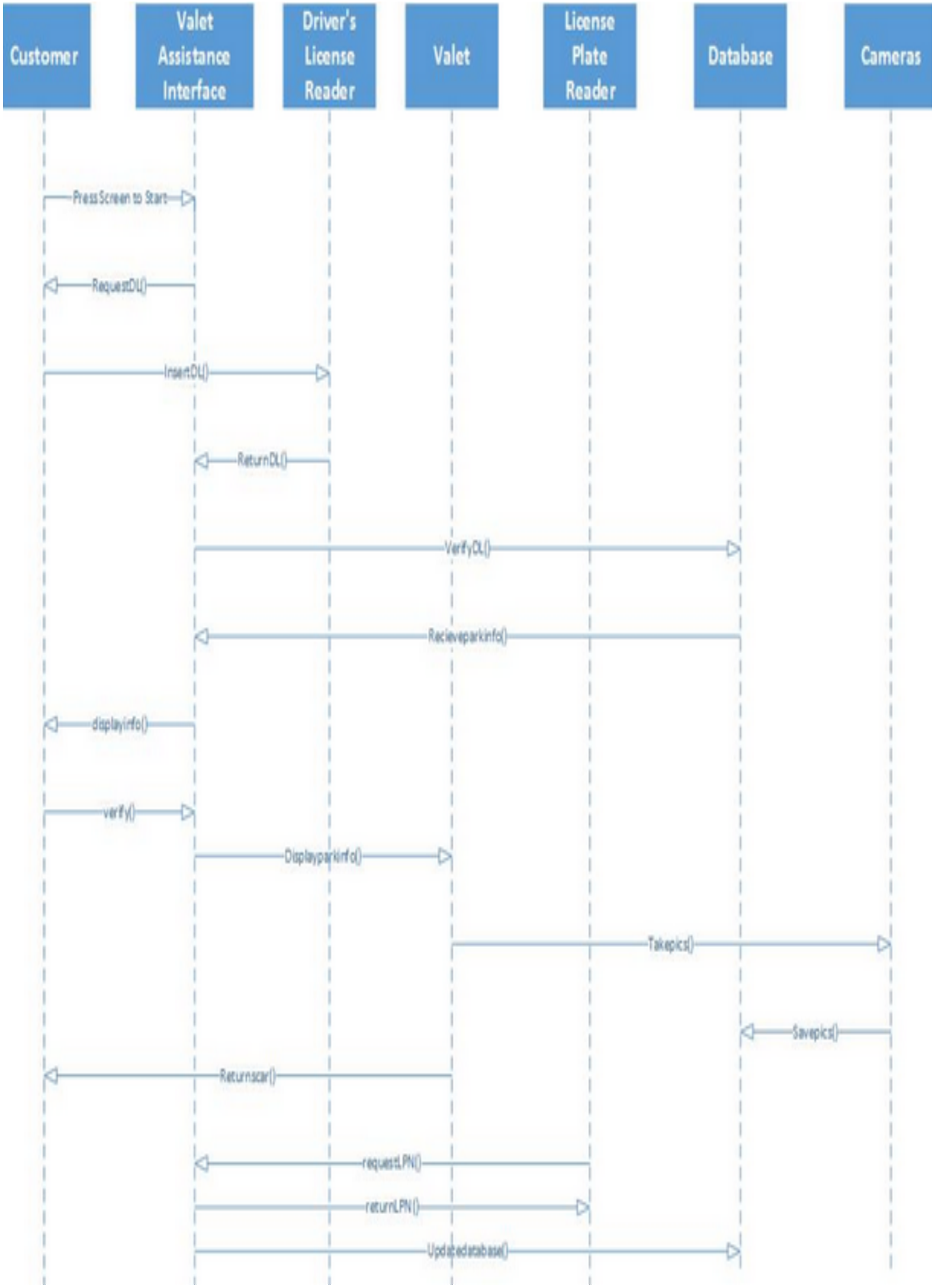
Database ⇔ Valet Assistant Interface

- D → VAI: When requested, the database returns to the VAI whether or not the customer has a reservation as well as whether or not the spot.
- VAI → D: The Valet Assistant Interface requests from the database, whether or not a customer has a reservation, a spot is available, and stores new updated information to the database on new reservations.

Drivers License Card Reader ⇔ Valet Assistant Interface

- DLCR → VAI: The Drivers License Card Reader takes the scanned information and fills in the customer information.

Exiting the Garage


Customer
Valet Assistance Interface
Driver's License Reader
Valet
License Plate Reader
Database
Cameras
PressScreen to Start
RequestDL()
InsertDL()
ReturnDL()
VerifyDL()
Recieveparkinfo()
displayinfo()
verify()
Displayparkinfo()
Takepics()
Savepics()
Returnscar()
requestLPN()
returnLPN()
Updatedatabase()

Customer ⇔ Valet

- C → V: The customer interacts with the valet by asking for their car back.
- V → C: The valet interacts with the customer by returning their car.

Valet ⇔ Valet Assistant Interface

- V → VAI: The Valet requests parking information from the Valet Assistance Interface.
- VAI → V: The Valet Assistance Interface returns the information on the requested parking space.

Database ⇔ Valet Assistant Interface

- D → VAI: The Database sends to the Valet Assistance Interface the customer's parking information

Valet ⇔ Camera

- V → C: The valet uses the cameras to take the pictures of the car.

Database ⇔ Camera

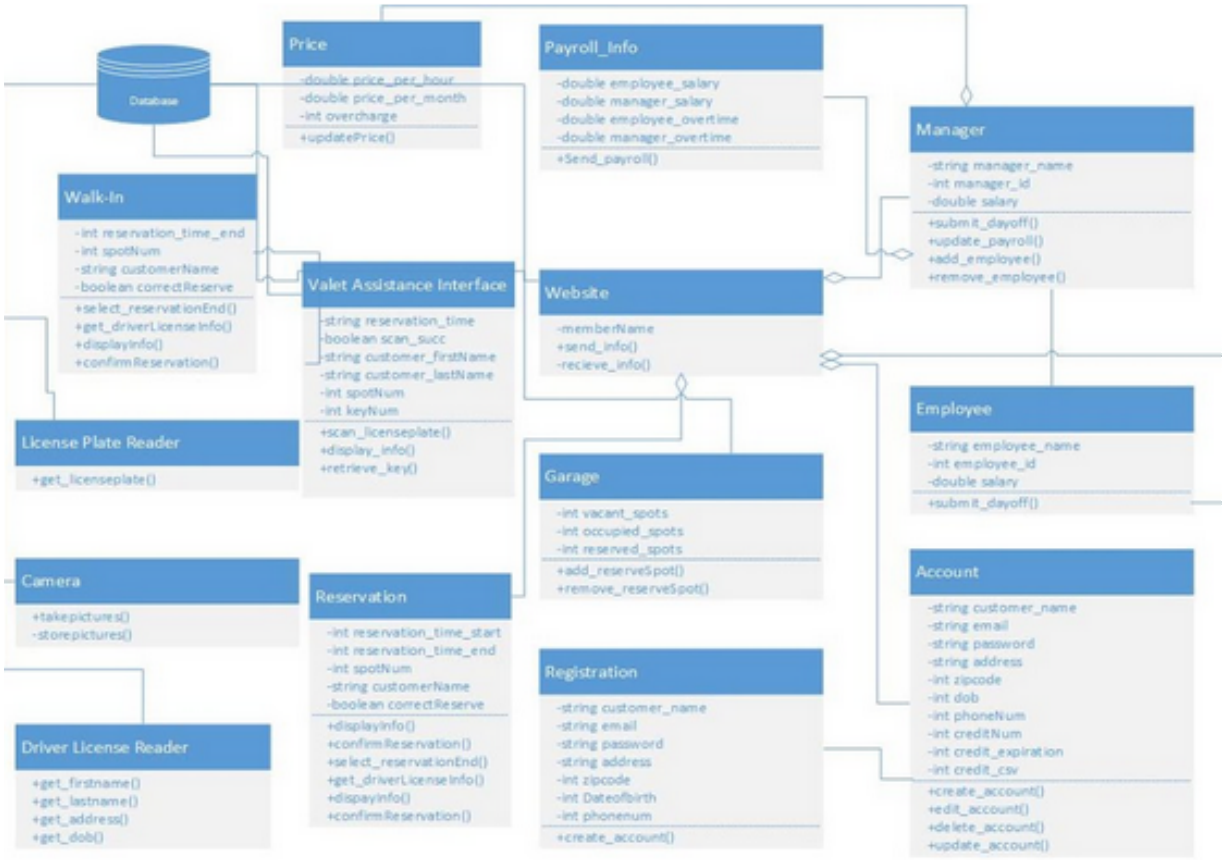
- C → D: The Camera sends the pictures taken to be stored in the Database.

License Plate Reader ⇔ Valet Assistance Interface

- LPR → VAI: The License Plate Reader sends the license plate number of the car to the Valet Assistance Interface so the valet can verify that it is the correct car.

Class Diagram and Interaction Specifications

Class Diagram - Figure 2.1.1



Data Types and Operational Signatures:

Classes that were displayed above in the diagrams are expanded below. The class has its attributes listed and their operations given with a brief explanation of what it does.

Valet Assistance Interface:

- Attributes:
  -string reservation_time;
  -boolean scan_succ;
  -string customer_firstName;
  -string customer_lastName;
  -int spotNum;
  -int keyNum;
- Operations:

+scan_licensePlate(): If License Plate scans correctly, display reservation information
+display_info(): Will display reservation information to Valet
+retrieve_key(): Valet will know what identification number is associated with car key

Reservation:

- Attributes:
  -int reservation_time_start;
  -int reservation_time_end;
  -int spotNum;
  -string customerName;
  -boolean correctReserve;
- Operations:
  +displayInfo(): Will display reservation information to the monitor
  +confirmReservation(): Customer will confirm reservation information
  +select_reservationEnd(): Customer will declare end time of reservation
  +get_driverLicenseInfo(): Customer will enter license to extract information
  +displayInfo(): Will display customer information to the monitor
  +confirmReservation(): Customer will confirm reservation information

Walk-in:

- Attributes:
  -int reservation_time_end;
  -int spotNum;
  -string customerName;
  -boolean correctReserve;
- Operations:
  +select_reservationEnd(): Walk-in will declare end time of reservation
  +get_driverLicenseInfo(): Walk-in will swipe license to extract information
  +displayInfo(): Will display customer information to the monitor
  +confirmReservation(): Customer will confirm reservation information

Registration:

- Attributes:
  -string customer_name;
  -string email;
  -string password;
  -string address;
  -int zipcode;
  -int dateOfBirth;
  -int phoneNum;
- Operations:

+create_account()

Account:

- Attributes:
  -string customer_name;
  -string email;
  -string password;
  -string address;
  -int zipcode;
  -int dateOfBirth;
  -int phoneNum;
  -int credit_num;
  -int credit_experation;
  -int credit_csv;
- Operations:
  +create_account()
  +edit_account()
  +delete_account()
  +update_creditInfo()

Price:

- Attributes:
  -double price_per_hour;
  -double price_per_month;
  -int overcharge;
- Operations:
  +updatePrice(): Manager has ability to update price of garage

Garage:

- Attributes:
  -int vacant_spots;
  -int occupied_spots;
  -int reserved_spots;
- Operations:
  +add_reserveSpot()
  +remove_reserveSpot()

Employee:

- Attributes:
  -string pemployee_name;
  -int emplyee_id;
  -double salary;
- Operations:

+submit_sickDay()

Manager:

- Attributes:
  -string manager_name;
  -int manager_id;
  -double salary;
- Operations:
  +submit_sickDay()
  +update_payroll()
  +add_employee()
  +remove_employee()

Payroll Information:

- Attributes:
  -double employee_salary;
  -double manager_salary;
  -double employee_overtime;
  -double manager_overtime;
- Operations
  +send_paycheck()

Traceability Matrix:

Domain Concepts are **BOLDED**
Responsibilities taken from Concept Definitions (Report 1)

Valet Assistance Interface
Responsibilities:
- Collect customer information (no reservation)
- Obtain Parking Lot and key storage number
Classes:
- scan_licensePlate(): -> collect customer information
- display_info():
- retrieve_key(): -> obtain key storage

License Plate Reader
Responsibilities:
- Check if customer has reservation
- Signal car has left
Classes:
- scan_licensePlate(): -> collect customer information and checks for reservation

Outside Parking Sign
Responsibilities:
- Shows how many parking spaces available
Classes:
- N/A

Website
Responsibilities
- Obtain customer information and make reservations
- Manage prices/fees for using the garage
- Manage employee information and statuses
- Analyze Garage statistics
Classes:
- create_account(): -> Obtain customer information
- edit_account(): -> Obtain customer information
- delete_account(): -> Remove customer information
- update_creditInfo(): -> Obtain customer information (pay)
- updatePrice(): -> Manage prices/frees for using garage
- displayInfo(): -> Called multiple times to display information to screen
- confirmReservation(): -> Customer will confirm reservation information
- select_reservationEnd(): -> Make reservations
- get_driverLicenseInfo(): -> Obtain customer information
- add_reserveSpot(): -> Make reservations
- remove_reserveSpot(): -> Delete reservation
- submit_sickDay(): -> Manage employee status
- update_payroll(): -> Manage employee information
- add_employee(): -> Manage employee information
- remove_employee(): -> Manage employee information

Database
Requirements:
- Notify manager of overstay
- Hold all customer data/parking information within a server (new req)
Classes:
- N/A

Valet
Requirements:
- Park car in lot
- Return car
- Use Valet Assistance Interface to verify reservation (new req)
Classes:

- scan_licensePlate():
- retrieve_key(): Assists in letting the valet know where the car is

Camera
Requirements:
- Obtain pictures of the condition of the car
Classes:
- N/A

Drivers License Reader
Requirements:
- Collect driver's license information
Classes:
- get_driverLicenseInfo(): Walk-in will swipe license to extract information

Explanation of Traceability Matrix:

The above domain concepts were the concepts taken from Report 1 Concept Definitions. Stated are the responsibilities for the above concepts and what the software classes should be accomplishing. Each class was grabbed from the above section "Data Types and Operation Figures", where the classes were created. The three concepts without software classes were not worked on in this part, although a server is going to be included under system in the future. Classes created within this lab reported were account for and added into the above line-by-line description of the evolution of each concept. Each software class introduced was paired with the responsibilities it was meant to satisfy. e.g "create_account" is meant to have the customer create an account upon which their information is saved within the database which covers the responsibility of obtaining customer information. There were some concepts with repeated classes because they satisfied multiple responsibilities among multiple classes or the concepts coincided with each other to a degree.

System Architecture and System Design

Architectural Styles:

The overall style of our system follows a component-based methodology. Our system will be comprised of multiple hardware and software system that will need to be able to function on their own and relay their information to the appropriate component. The aspects of component -based styles are that they are: reusable, replaceable, no context specific, extensible, encapsulated, and independent. The benefits of of component-based styles is that they have ease of deployment, ease of development, reduced cost, and are reusable.

Each of our components such as the license plate reader, license card reader, Valet Assistant Interface, and cameras are all components that are taken advantage of and will need to communicate with one another to create a working system. A lot of these components already exist in the real world which makes this system take advantage of component-based methodologies.

The database and website will follow a client/server architectural style. The reason for the client/server architecture is because we need something to maintain the data and retrieve information. The website will need to access the server which will then access information that needs to be displayed on the the user's web browser. This server will also be needed when the Valet Assistant Interface communicates with the database.

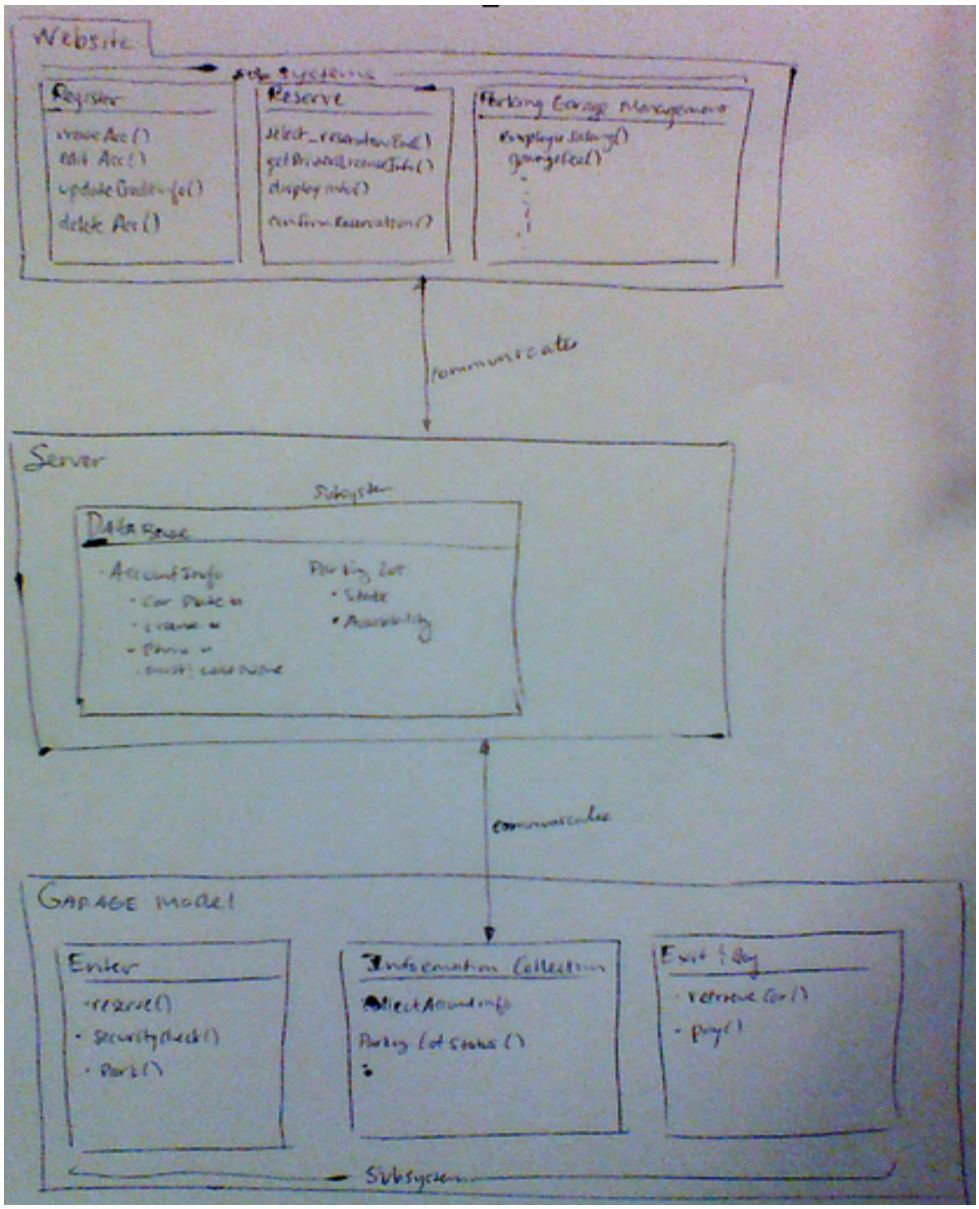
The database and website together will be of an event-driven architecture. The database and website are both highly dependent on either user input or the current status of the parking spots within the garage structure. It therefore requires that these two systems be event-driven. These two aspects will not be the only thing that is event driven. Because our garage components are all event driven we decided to use a pipe and filter model.

We are using a pipe and filter model to maintain this parking garage. The speed of our information collection will "pump" the car through the garage and "sink" the information into our database. The faster the processes the faster the user can park. We are trying to optimize the time the user waits by using the License Plate Reader to query the system if the driver has a reservation. If the user does have a reservation, the driver would be able to continue to the valet where the car is now in the companies's hands.

This pipe and filter model also describes how our event-driven model works. The car entering the garage and triggering the Assistant Interface, allows for the system to start collecting information to send the driver to the security check. The completion of the information collection, allows the flow to the speedy security check which is handled by the cameras and the valet's use of the interface. Then, this completion of the security check leads to the Parking of the Valet and Ticket for Driver to retrieve his car later in the day. The customer's return triggers the retrieval of the car which sends a Valet to obtain the car and trigger the Exit subsystem. This Exit subsystem causes the License Plate Reader to send the License Plate of the car leaving to the database. The database then

updates the status of the lot for future use. The car is returned to the customer as he pays the cashier. The filtering deals with the exceptions such as a car not wanting to park and signaling the car to exit or managing invalid inputs for information collection.

Identifying Subsystems:



The subsystems of our System Architectures comprises of the:

- For the Garage Model
  - Enter
  - Information Collection
  - Exit and Pay
- For the server
  - Database

- For the Website
    - Register
    - Reserve
    - Parking Garage Management

Our information collection subsystem is comprised of our License Plate Reader, Driver's License Reader, and the Cameras, database. This subsystem is responsible for the collection of important information needed to run securely run the parking garage.

The Enter subsystem is comprised of the Assistant Interface, Driver's License Reader, and the License Plate Reader, database. This subsystem will maintain that the entrance stage is complete to park the car, else leave the garage.

The Exit and Pay subsystem is comprised of updating the parking lot through the use of the License Plate Reader and the database and money management to validate the return of the car to the Driver.

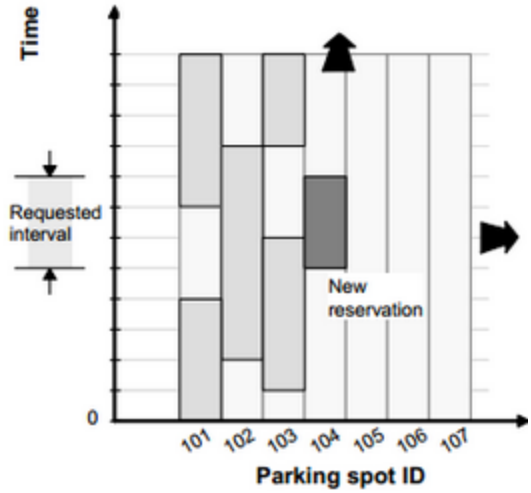
Mapping Subsystems to Hardware:

The system will have to be able to run on multiple devices. In particular, the system will have to be able to communicate with the Valet Assistant Interface. The Information Collection subsystem should be able to maintain the information that is provided by the external devices such as the License Plate Reader, Cameras, and the Drivers License Reader. It is responsible for making sure the information collected is valid and is properly stored into the database.

Persistent Data Storage:

The system will need to be able to preserve the states of the parkings spots within the parking structures for an indefinite amount of time. The preservation of these states will be stored in a flat file in a form that is a variation of a bitmap.

The file format will be of the following:



The file format will consist of a 2-dimensional matrix (contained in an array) that

contains the parking spot ID and the time interval that are reserved. The matrix will consist of integer digits from 0-3 for the four possible states of reservation the parking spots can be in: unoccupied, occupied, reserved unoccupied and reserved occupied.

Data storage will differ for other data that will be preserved. Employee information that is stored on the database will follow the Entry-attribute-value model, which can also be stored as a flat file. Information that could stored are as follows: Name, salary, job title, and shifts.

Global Control Flow:

Our system is event-driven and we will be able to have multiple requests happening in short periods of time.

Time dependency:

There is some time dependency after the cameras are taken. The camera will need a few seconds to automatically adjust to the lighting. This will be solved by the valet pressing the capture picture button as soon as he sees that the light is green signaling the camera is ready for a picture.

Hardware Requirements:

- Tablet 8GB(min): as the Valet Assistant Interface
- LED Display: for outside to show vacancy of the garage
- Cameras: 10MP for security check
- Driver's License Reader: to quickly obtain information with one swipe
- License Plate Reader: 10MB min to read the plate
- Database: 100GB at least to store all the information.
- Hard-drive: At least 50 GB of space. This disk will store only cached information.
- Server: to process the information valet interface and the license plate reader
- Internet connection: to connect the garage system with the garage. (At least 2Mbps)

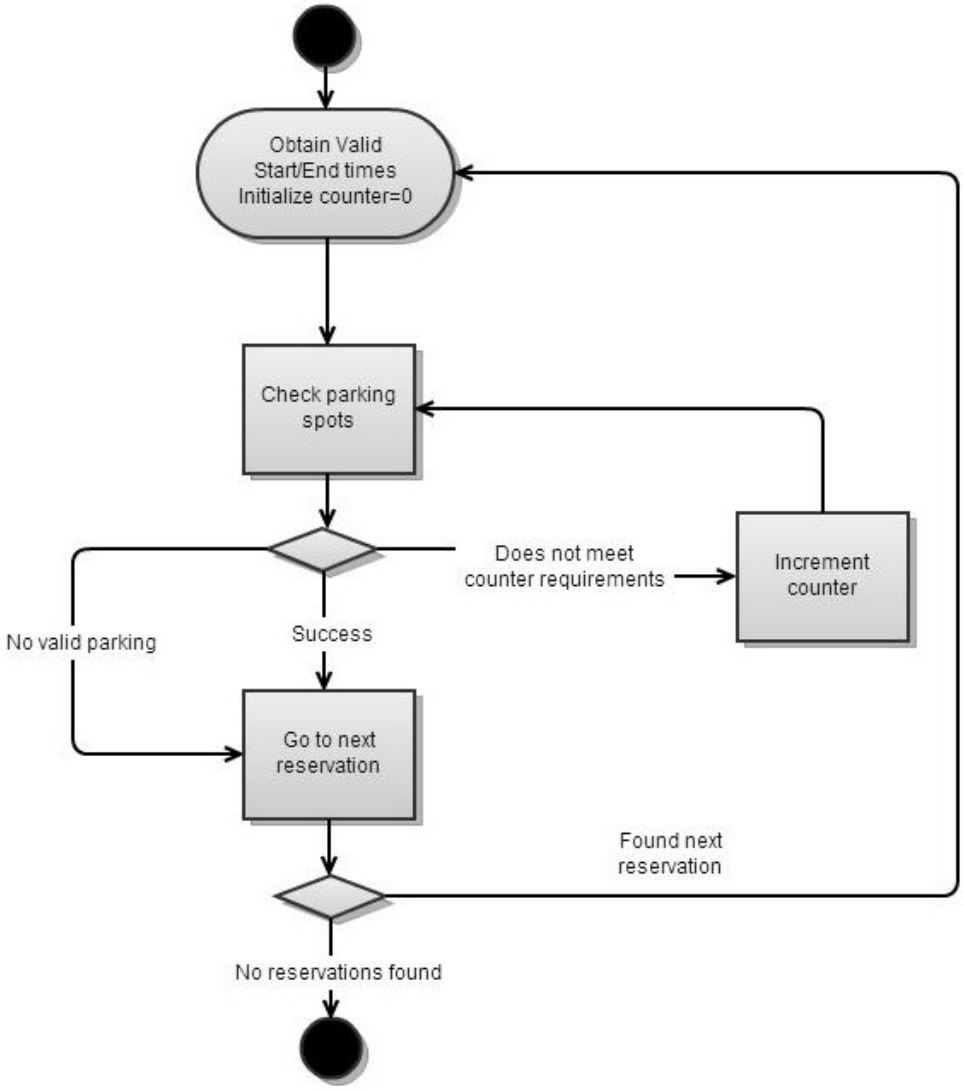
Algorithms and Data Structures

Algorithms:

The mathematical model for the simulation of arrivals and departures will be fairly simple to implement. The algorithm to describe the arrivals/departures of vehicles will closely follow the original equations defined in Report 1. Within a conditional while loop, this algorithm will continue to execute with a random number generator until the garage is full.

There will also be an algorithm that will sort the reservations within the garage. The data will be stored as integers within an array. The algorithm will note in the array matrix where the parking reservation begins and ends. These two integers are critical for determining where the spot can be shifted to. The algorithm will need to check each parking spot and check if there is no conflict between the beginning and end times with any previously placed reservations. Since the consolidation of these parking reservations is the number one goal, it is important to minimize the unused times between parking reservations. To take this into account, a counter which is initialized to zero will be used.

The counter is used to describe the maximum time units (here our time unit is described as 15 minute intervals) there can be between the end of the swapped reservation and a reservation in place or the beginning of the swapped reservation and end of a reservation in place. If the algorithm cannot find a spot to place the reservation, then the algorithm goes onto the next reservation to begin swapping. However, if there are available spots but does not meet the maximum time unit requirement, the counter will be incremented by one and the process repeats until the conditions are satisfied. It it is important to note that the counter cannot increment more than the number of time units there were in the original reservation spot.



Data Structures:

Our system does not use "complicated" structures such as trees, hash tables, or linked lists, but we do utilize arrays, MySQL tables, and bitmaps. The logic in this decision is that since our system has such a large portion in a database on a web server, we should avoid implementing things that are not inherently or implicitly supported, such as the aforementioned trees and linked list.

Since the web language is HTML and CSS, PHP, and MySQL, we decided to optimize the built in features with the MySQL table system, which is based in Structured Query Language (SQL), where queueing something is as simple as a stating something like:

```
$> SELECT * FROM my_table WHERE field1 = 'attr1' AND field2 > attr2
```

This would quickly search, within table <my_table>, for entries whose <field1> stores the string <attr1> and in which <field2> is greater than <attr2>. This is powerful for getting specific users within a certain time frame, or logging in a user that has matched their username and password.

The other prominent structure is a bitmap used to help keep track of reservation times within the garage. The bitmap is essence a two-dimensional array that can store information in its cells. It may not be the most efficient data structure in computer science, but it is one of the bests that is able to easily move from web applications to mobile applications and can operate in many languages.

User Interface Design and Implementation

Valet Assistance Interface:

(Screen 1) In report 1, the customer would be initially brought to the start screen that would display whether or not the lot/garage has vacancies with the longest interval at which they can park. To cut back on excess information, this screen will just ask whether or not the customer has a reservation with the option buttons of “Yes” and “No”.

(Screen 2) “Yes” If the customer selected yes, it will display the same information as stated in Report 1.

“No” If the customer selected “No”, it will display at the top of the screen whether or not the lot/garage has vacancies and also display the longest interval the customer can park. The rest of the screen will contain the same information as Report 1 in which the customer will enter the amount of hours and their phone number, but instead of them having the “Confirm” option, it will be “Next”.

(Subscreen 2) This screen will be added, increasing the user effort by 1 click, or more if they decide to press the “Back” button. This screen will display the information that the customer had just entered and will ask if it is correct. If it is correct, then the customer will hit the “Confirm” option. If it is incorrect, the customer will hit the “Back” option and correct any information that was inputted incorrectly in the previous window. In the end, this will save both sides the time it takes to correct any mishaps that may occur in the

future due to incorrectly inputted information.

(Screen 3) This screen will be the same as in Report 1. It was ask the customer to scan their driver's license for confirmation.

As in Report 1, there will be another assistant upon exiting and the customer will be asked to touch the screen to start.

(Screen 1) This screen will be the same as in Report 1 and have a "Start" option.

(Screen 2) This screen will ask the customer to scan their driver's license as in Report 1.

(Screen 3) This screen will be added and will display what car the system has for that customer and ask for confirmation with the "Yes" button or "No" button.

(Subscreen 3) "Yes" The screen will display "Thank You for Parking" as well as the cost of parking.

"No" The screen will display "Please contact an employee, sorry for the inconvenience"

For the Valet Assistance Interface (Entering), the initial screen is easier to follow now that it will just display "Do you have a reservation?" with the options Yes or No. A confirmation screen was added that will increase user effort by a minimum of 1 click, but in the long run it will save effort on both sides.

For the Valet Assistance Interface (Exiting), the program was elaborated on more. Two extra screens were added, increasing user effort by a minimum of two clicks. One is a confirmation page that will display whether or not the correct car is linked to their driver's license and is ready for retrieval. Then based the response, it will take them to the final screen, displaying the cost of parking, or whether or not a problem needs to be resolved.

Valet Application/Website Interface (For Employees):

1. The first page/screen will remain the same as explained in Report 1. It will ask the valet to input their ID and if valid, it will bring them to the next page. If it is incorrect, they will be prompted again until a correct ID is inputted.
2. The second page/screen will generally remain the same, displaying the reservation list and a "Customer Information" Link. There will now also be options "Back to Main Page" and "Log Out"
3. The third page/screen will be changed to make it easier for the valet. On this page, they will be given a text box in which they can enter a customer's name. If the customer exists within the database, the valet will be brought to the next page. If not, they will be prompted to enter the name again.
4. The fourth page/screen will generally remain the same. It will display the

customer's personal information (ex. Birthday, Phone #, Driver's License Plate, etc.). This page/screen will now also include "Back", "Back to Main Page", and "Log Out" function.

5. The final page/screen will display "You have successfully logged out".

For this application, an extra window was added for successful log out at the end, not really increasing user effort as it will just display it. User effort was decreased in page 3. In report 1, The customer information page required entering personal information, when now it will just ask for the name before displaying personal information.

Customer Website Interface:

In report 1 we did not have an precise user effort estimation for the website interface below is the user effort estimation for the following webpages.

i. To **create an account** the user will have to take the following steps.

1. **Navigation**:
   a. Open homepage
   b. Point mouse to the 'account' menu
   c. click on the 'register' button
   d. Complete data entry(As shown below)
   e. Press enter
2. **Data entry**
   a. Click on the User Name data field
   b. Fill in the user name needed
   c. Press the tab key to move to the next field 'Password'
   d. Fill in a password which satisfy the requirements stated.
   e. Press the tab key to move to the next field 'First Name'
   f. Fill in customers first name
   g. Press the tab key to move to the next field 'Last Name'
   h. Fill in customers 'last name'
   i. Press the tab key to move to the next field 'Street Address'
   j. Fill in customers 'Street Address'
   k. Press the tab key to move to the next field 'City'
   l. Fill in customers 'City'
   m. Press the tab key to move to the next field 'State'
   n. Fill in customers 'state'.
   o. Press the tab key to move to the next field 'Zip code'
   p. Fill in customers 'zip code'
   q. Press the tab key to fill in the security code
   r. Enter correct security code

ii. To **create a reservation** the user will have to take the following steps:

1. **Navigation**:
   a. Open homepage

b. Click on the reserve option
c. Complete data entry(As shown below)
d. Press enter

2. **Data Entry**
   a. Enter the customers users name
   b. Press tab and move to the next field 'password'
   c. Enter the correct password
   d. Press enter to go the reservation page.
   e. Enter Vehicle License Number
   f. Press tab to go to the next field (drop-down menu) 'Start Reservation Time'
   g. Select the time customer will be there
   h. Press tab to go to the next field (drop-down menu) 'End Reservation Time'
   i. Select the time customer will leave
   j. Confirm required time

iii. To **edit account information** the user will have to take the following steps:

1. **Navigation:**
   a. Open homepage
   b. Point mouse to 'account' option
   c. Select Account Info and choose edit Account
   d. Make Changes as needed.
   e. Click on save changes

iv. To **edit a reservation** the user will have to take the following steps.

1. **Navigation:**
   a. Open homepage
   b. Point mouse to 'reserve' option
   c. Select Edit Reservation
   d. Enter Reservation number
   e. Press enter
   f. Make Changes as needed.
   g. Click on save changes

iv. To **log out** of the respective account:

1. **Navigation:**
   a. Point mouse to 'account' option
   b. Click on the log-out option
   c. Click yes to confirm account log-out

Design of Tests

Each unit will be tested before integration with one another. The units that need to be tested are (These also cover the requirements mentioned in Report #1):

- Valet Assistant Interface
- Website
- Database
- Classes and Methods

Testing the Valet Assistant Interface:

Goal: The Valet Assistant Interface is responsible for interacting with the driver upon entrance of the garage and interacting with the Valet as he/she works, parking and retrieving cars.

Test Cases:

- Check if driver has reservation
  - This makes sure that the interface can communicate with the license plate reader and then checks the database
  - To check this we need to see if we are able to search for a reservation given a license plate number provided by the reader
- Check for valid input
  - Filters input to keep the system safe
  - To check this we will try to put invalid input such as letters for phone number and numbers for first and last name.
- Minimal strokes for driver (ease of use)
  - Use the valet assistant interface to see if modifications to the layout is needed.
  - To check this we will play with the interface and get input to see how to change the layout to make it more user friendly
- Valet Login
  - Allow the valet to use the Valet side of the system to do his/her job
  - To check this we will see if the valet can log in using a valid username and password
- Security Check
  - Obtain pictures from camera and send to the database
  - To check this we need to see if the pictures are stored into the database and able to obtain the pictures from the database to show the driver incase he wants to see them

Testing the Website:

Goal: The website is responsible for allowing the drivers to make online reservations and create an account to manage their reservations.

Test Cases:

- Create account
  - To check this we will see if we can create an account and verify it with an email address
- Create reservation
  - To check this we will see if a reservation is stored in the database and make sure that a driver cannot make a reservation in a time slot that is taken already
- Manager login
  - To check this we will see if the manager can log in using a unique username and password
- Manage employee information
  - To check this we will see if the website displays proper employee information and make sure that changes can be made. We will also see if we can delete and add more employees
- Manage garage fees and prices
  - To check this we will see if a change in the in managers log in will update the fees

Testing the Database:

Goal: The database is responsible for recording driver's information, vacancy of the parking garage, employee information, garage fees, security check pictures and reservations.

Test Cases:

- Record data
  - To check this we must check the database and see if information is being stored properly and if we are able to obtain the correct information

Testing Classes and Methods:

Goal: The classes and methods in our system are responsible for making the units work, these are the tools that the units use to accomplish their task.

Test Cases:

- Refer to: Data Types and Operational Signatures
  - These are the classes and methods we will be testing
  - To test these we will see this we will use a combination of print and if statements to see if the output is valid

Integration Testing Strategy:

When the units are complete we will slowly integrate a few units together at a time to ensure proper integration of the whole system. We plan to complete the database first and test if we can properly store information into it. While this is happening we will also make sure that our website it working properly such that we can create accounts and reserve and the proper information is being stored into the database. The valet assistant interface will be tricky but once the website and the database seem like they are working properly we will try to tackle the challenge of integrating the valet assistant interface.

Project Management and Plan of Work

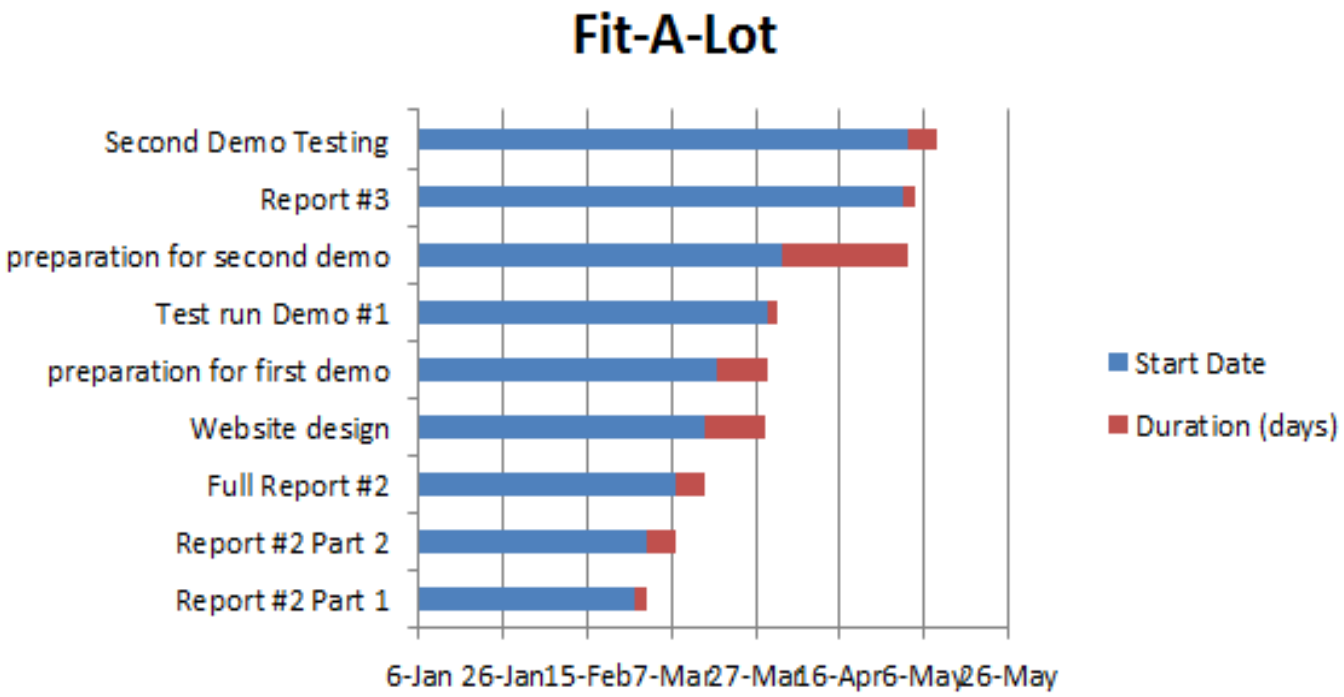
Merging the Contributions from Individual Team Members:

Matthew Brazza will work to compile Report 2's complete form; we hope to work together for the final report. We need to think about what the diagrams already look like in the parts we submitted, so when we combine them all, they will be more uniform and thus more professional. This also extends to how we create the reports, because it is surprisingly difficult to combine the reports then try to make sure they are uniform in style and such.

Project Coordination and Progress Report:

Currently we have a website up with the basic layout and functional aspects being worked on. The Valet Assistant Interface has a basic layout set but the functionalities will still need to be worked on heavily due to the loss of a member who was working with the Valet Assistant Interface.

Plan of Work:



Breakdown of Responsibilities:

Currently the website and server/database functionalities are being split amongst Matthew Brazza and Parth Patel; Matthew is focusing on log-in/registration process with HTML, PHP, and MySQL, and the use of cookies, while Parth is working a lot on general

website layout and functionality. Justin Cruz will continue to work on the Valet Assistant Interface, but we will make sure he is supported due to the loss of his partner, who dropped the class. Sean Wang and Sheldon Wong are working on the algorithm and sorting efficiency of the garage. They will also work on some classes such as Garage, while Justin works on Walk-in, and Parth and Matthew work on Account, Registration, Reservation, and other website related classes. As soon as the separate units are completed the groups who completed those parts will communicate with each other to do integrated testing.

References

---

http://en.wikipedia.org/wiki/Software_componentry

http://en.wikipedia.org/wiki/Event-driven_architecture

http://en.wikipedia.org/wiki/Entity-attribute-value_model