Parking Garage Automation: Reserve Your Spot!

Software Engineering – 14:332:452

Group #3:
Bartosz Agas, Christoper Tran, Marvin Germar, Michael Genderen, Justin Levatino, Tarun Katikaneni

URL(s):
www.reserve-your-spot.com (Actual Website)
www.sites.google.com/site/ece452parkinggarage (Project Tracking Site)

Submission Date:
May 3, 2012

Effort Breakdown

All team members contributed equally

Table of Contents

Summary of Changes

Revisions to Report One

- Built upon the description of the system mentioned at the beginning of the report
- *Glossary* – Clarified certain items mentioned
- *Stakeholders* – Updated to identify the proper people interested in this system
- *Customer Statement of Requirements* – Provided additional requirements to meet the business policies and cases required to uphold the system
- *Domain Model* – Modified the domain model to account for all the objects communicating within the system and explained the attributes and concepts used in the highlighted use cases

Revision to Report Two

- Clarified section numbers, enumerated references, and numbered figures
- *Interaction Diagrams* – Added design principles to each diagram
- *Class Diagrams* – Added additional descriptions to the class diagrams.
- *Traceability Matrix* – Provided an up to date traceability matrix to account for the addition of several new user requirements
- *Identifying Subsystems* – Provided a more in depth description to help better understand the client/server relationship established for the system
- *Hardware Requirements* – Modified the descriptions to be more specific
- *User Interface* – Displayed all the basic screen shots of the main page of the completed web site
- *Design of Tests* – Added specific test cases and state diagrams

Customer Statement of Requirements

Goals

The objective is to design a sophisticated system which will seek to maximize occupancy and profit while allowing the customer quick and easy access to his vehicle.

Problem Statement

At the moment, the garage is not equipped with any computerized system. Additionally, the current system involves the employees to walk around inspecting the occupancy of parking spots. Due to the lack of a computerized system, congestion inside the garage is rampant. The current system is not well designed; during peak times the garage could have free spots but would have no way of checking instantly, this would discourage customers from wanting to park in the garage thereby robbing the garage of additional income. The management has taken note of the situation and has requested the designing and implementation of software that would increase their efficiency, thereby increasing their profit.

Proposed Solution

In order to fix the mentioned problems, the management has requested us to design a system that would make their garage more efficient thereby increasing its occupancy and profit.

The new system will include a website or mobile application that will allow customer to place online reservations. The reservation would include date, time and duration of stay. Each customer will be required to register on the website; at registration time, the customer is not compelled to enter in a license plate number for their vehicle, this is allowed so that customers are not tied down to one vehicle. The system allows for customers to be able to park even with rented or borrowed vehicles, it does this by creating a temporary association to the new license plate to the customer. This was done so that the garage is accessible by more customers, which would bring in more income to the garage.

The garage is also being remodeled such that the parking decks above ground level will be accessible only by an elevator that will lift vehicles to different decks. One of the major problems of the garage before the implementation of the new software was rampant congestion caused by drivers searching for parking spots. To alleviate this problem, the management has devised one-way entering and exiting systems. As we have already discussed, the elevator is the only way for vehicles entering the garage to get to their parking spaces; but in addition to that, the management has also constructed a ramp that connects all the floors to the ground floor. The customers will use this ramp to exit the parking garage; as there is no two-way traffic on the ramp, there is a very little chance of accidents and such there is no way congestion can form while customers enter and leave the garage.

The proposed system will not depend on employees to check if spots are available; instead, the garage relies on camera based license plate recognition software to track vehicles as they enter and exit the garage. Additionally, the garage also employs sensors on the parking spots to recognize which spots have been taken and which are open. This capability reduces the chance of mistakes by the employees there by making the garage more efficient in assigning parking spots to customers. Each time a vehicle enters the garage, the tracking software quickly takes reads the license plate and refers to the database.

If the software cannot recall the necessary information or if the license plate recognition software is not able to read the license plate, the elevator will not function and the software would prompt the customer to manually input their membership number at the terminal next to the vehicle elevator for it to proceed. The system is designed to account all possibilities, such that if the vehicle does not have a front license plate, the software would alert the customer to do the same thing. As we have mentioned earlier, if the software cannot recall a certain license plate then the customer can register that license plate to his account as well.

If a registered customer forgets to make a reservation and decides to use the garage then he may be allowed to take a walk-in parking spot without a registration if there are any available spots. These types of customers are known as walk-in customers. If the software recognizes the vehicle registration number but cannot find an existing reservation to the customer who owns the vehicle, then the customer will have to specify the expected duration and time of departure using the terminal at the vehicle elevator. If the vehicle registration number is not recognized then the software will prompt the customer to type in their membership number and their estimated parking duration.

In order to restrict people from making reservations they cannot meet, the system has broken down reservations into two groups, confirmed and guaranteed. A confirmed reservation is when a registered used places a reservation, but does not have a credit card on file. A guaranteed reservation is when a registered customer has done the same, but has a credit card on file and uses it when placing their reservation. These two types of reservations differ when the customer shows up late to the garage for their reservation.

If a customer with a confirmed reservation fails to show up after reserving a spot, the spot will be held reserved for a 30-minute grace period, during which the customer can park on his reserved spot and be billed for the full reserved period. If the customer does not show up to claim his spot during the grace period then the parking spot will be marked unreserved. With a guaranteed reservation, the customer can arrive to their spot anytime during the requested interval and will be charged to their card for that interval. As registered customers they will be charged a registered price per hour during the time of their interval. Any customer who stays past their requested reservation time will be charged a registered price for the reserved time and an unregistered price the amount of time gone over. If any customer arrives and their spot is still occupied by a previous customer who overstayed then the garage will direct him to another parking spot, however if there are no vacant parking spots, then the customer is given a rain check.

With so many machines working at the same time, there can be cases where things can go wrong. One of the most important machines in the garage is the camera based license-plate recognition software. Without the software, the elevator would not function and the garage would not know if a customer has exited. The importance of the cameras are so great that there are several fail-safe's built into the system; firstly the camera is designed to work under all conditions and it is able to read the license plates despite its condition. Also, if the system does not recognize a license plate, the software prompts the customer to enter their member number; if the software doesn't recognize that as well, then it does not allow the customer to enter the garage.

One of the main advantages of this new system was that employees no longer had to manually check parking spots to see if they were occupied; to assist them, floor sensors were installed in all the parking spots. These sensors are triggered when a vehicle parks in the parking spot, at this time, the sensor automatically alerts the garage's system that the spot has been occupied. The other function these sensors have is that they also alert the system whenever the vehicle leaves the parking spot and the system marks that spot as vacant. To prevent malfunction, the sensor was designed such that it doesn't trigger unless a car occupies the spot or leaves the spot.

A major problem in the old garage was that customers generally spent a long time looking for parking spots which resulted in congestion within the garage. The new system is designed such that if the customer’s spot is still occupied by a previous customer then he or she will be assigned a new parking spot. To help customers, the vehicle elevator was designed such that it will always lift the vehicle to the appropriate deck and never stop at a wrong deck; this is quite important because, it will prevent customers from searching for their spot. It will also reduce the chance that a customer will accidentally park in another spot instead of the one that was assigned to them.

The website will be used to allow customers to place online reservations as well as provide the parking garage staff with basic customer information and statistics. The goal is to design a friendly user interface in order to allow the customer to use the website on a computer and possibly a mobile device. Any mobile device app will be presented in our demo’s using a phone emulator on our computers. It will have a fundamental structure so anyone using the site can do so while on the go or multitasking. The home page will take the user to a login page where unregistered users can easily create an account with basic information. As changes are made, the database will be adjusted through this website as customers will enter their account information, register their vehicles, and place reservations.

Glossary of Terms

App – A mobile application where customers can access the system to view their account or make a reservation.

Camera – A device for recording or reading visual images in the form of photographs and video, used to read license plates and send information to garage system.

Cancelled Reservation – Occurs when a customer cancels their reservation before the reservation period.

Confirmed Reservation – A reservation placed by a registered user where the user chooses not to attach a credit card to it

Customer – A person who wishes to use the garage's services.

Database – Entity that stores all the system's information.

Elevator – A platform used to raise vehicle to different floors.

Elevator Terminal – A console or screen inside the elevator where the customer can enter in necessary information.

Grace Period – An amount of time for a late customer, with a confirmed reservation, to claim his spot before the reservation is removed.

Guaranteed Reservation – A reservation placed by a registered user that has a credit card attached it

License-Plate Recognition Software – A camera based system that reads the license plates of vehicles and checks the information against the database.

Member Number (Customer ID) – A unique number that is given to customers who have registered on the site.

No-Show – A customer who does not show up for their reservation.

Overbooking – Accept more reservation for parking spots than there is room for.

Overstay – When a customer doesn't leave the garage at the end of his reservation period.

Rain Check – A ticket given to a registered customer who has reserved a parking spot but, there are no vacancies.

Registered Customer – A customer who has registered an account on the garage's website prior to showing up to the garage.

Reservation – The act of reserving a parking spot via the system's web site.

Reservation Confirmation Number (Reservation ID)– A number that is given to the customer as confirmation of their requested reservation.

Sensors – A device that is placed on the floor of every parking spot that detects if that spot is occupied or vacant.

Under Stay – The act of leaving a parking spot before the reservation period is over.

Vehicle – A thing that is used to transport people (I.E: car).

Walk-In – When a customer requests an immediate parking spot without prior reservation.

Website – An interface that the customer can use to register, and reserve parking spots.

System Requirements

Enumerated Functional Requirements

Identifier	Priority	Requirement
REQ-1: Plate_to_Read	5	The system shall read the plate
REQ-2: Car_to_Customer	5	The system shall identify the customer based on the license plate
REQ-3: Spot_Sensor	3	The system should detect which spots are vacant/occupied
REQ-4: Vacancy_Display	2	The system should display vacant spots
REQ-5: Elevator_Display	2	The system should display information to customer on the elevator
REQ-6: Elevator_Bad_Read	4	The system shall notify the user on the elevator that their plate was not in the system allowing a confirmation number input
REQ-7: Internet_Site	5	The system shall be linked to the internet
REQ-8: App	1	The system should be linked with an app
REQ-9: Elevator	5	The system should take the car to the correct floor level where the parking spot is reserved
REQ-10: Rain_Check	4	The system shall give rain checks to customers who have been overbooked
REQ-11: Exit_Gate	2	The system should have a gate at the exit
REQ-12: Register_Account	5	Each customer is required to register an account to place reservation in the system
REQ-13: Late_NoShow	4	If customers make a reservation, but do not show up, they will have a 30 minute grace period. Customers will be billed for their initial reserved time, the grace period, and the new reserved time if they extend their reservation
REQ-14: Pay_Overstay	4	Customers who stays pass their reserved period will pay an additional fee
REQ-15: Pay_Bill	3	Customers will be billed electronically once a month and are expected to pay
REQ-16: One_Way	5	The parking garage has a one-way entering and exiting system. An elevator lifts the car to the proper parking level and a ramp is used to leave the parking garage
REQ-17: Recognition	5	If the system cannot read the license plate number customers manually input their membership number in order to enter the parking garage.
REQ-18: Redirect	5	If a customer arrives at the reserved spot and it is occupied, the customer will be redirected to a vacant parking spot. If there are no spots available, a rain check will be given to the customer

Enumerated Nonfunctional Requirements

Identifier	Priority	Requirement
REQ-19: Sensors	5	The parking garage will have fully functional floor sensors and cameras that detect when a spot is vacant and when a customer is entering or exiting a garage, respectively. They will check if a customer has parked properly.
REQ-20: Simple_Design	4	Web pages will have a simple design to enhance user experience. Each page will have a similar template to prevent confusion.
REQ-21: Fail-Safe	5	The system is designed with fail-safes to decrease the chance of failures.
REQ-22: Last_Minute	4	The system will accept last minute parking request depending on availability and existing reservations.
REQ_23: Database	5	The database is stored on site to decrease the chance of data loss.
REQ-24: Ground_Spot	3	Customers will be assigned ground level parking as opposed to having them choose their own.

FURPS Table

FURPS (Priority Five)	
Functionality	• Features floor sensors that detect when a parking spot is occupied or vacant. • Includes a camera-based system that detects when a vehicle is entering or exiting the garage. • Contains a terminal-based system that allows the customer to enter in member information.
Usability	• There is consistency as all the web pages follow the same template. • To accomplish simplicity, each page has a navigation bar to access the individual pages • Some pages even include instructions and/or guidelines
Reliability	• System is designed with fail-safes to decrease the chance of failure. • System uses sensors and cameras to check if customers have parked properly. • Additional parking spots will be available if customer's spot is occupied by another vehicle. • Database is stored on site to decrease the chance of data loss.
Performance	• In order to be efficient, parking spots for the ground level will be assigned as customers enter the lot as opposed to having them choose their own. ○ The same method is applied to reservations that are placed online. • Throughput is increased as a result of the online reservation process. ○ By eliminating this step from the procedure done at the garage, it decreases the change of traffic congestion due payment transactions.
Supportability	• This design is very adaptable as far as the garage size is concerned. • If there are ever future plans to expand the garage additional floors or any other renovation, the adjustment would be as simple as making an update in the database table. • Maintainability is also fairly straightforward as there isn't too much customer information to deal with.

Functional Requirements Specification

Stakeholders

- Parking Garage Owners
- Camera and Sensor Companies
- Scanning Device Programmers
- Business Enterprises
- Architects
- Engineers

A parking garage system can be effective and lucrative if run efficiently, but it requires work and interest to create such a system. Consider an overpopulated area such as a city or theme park where parking spaces are limited. A parking garage system provides a resolution when demands for a parking spot are high. However, the parking garage system needs workers for sustainability and maintenance. To name a few, these include programmers, architects, construction agencies, and engineers. The stakeholders mentioned above are a few people out of many who are interested in implementing a functional parking garage system.

Actors and Goals

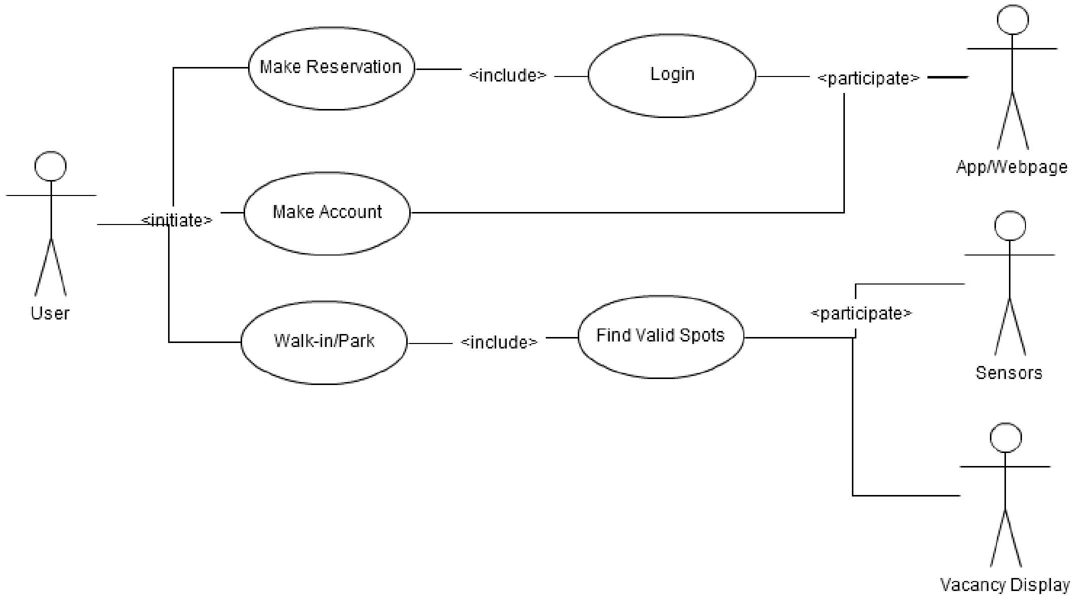
Actors	Goals
Car	To be shown to the correct level to park
Management	To set the parking prices and have maximum efficiency
Worker	To set up the reservations or to know that an invalid customer is trying to access the garage
Customer	To park at their parking spot
Cameras	To identify the car to the customer
Sensors	To find out if there are vacant spots in the lot
Vacancy Display	To show what spots are vacant or occupied
Database	To store all of the customer information
Internet Site	To allow customers to make their reservations
Phone	To allow the customer to use the app
App	To allow the user to make a reservation on the go
Servers	To allow the internet to run
Elevator	To allow the customer to access the correct level
Elevator Keypad	To allow customers to input reservation numbers if they're not identified by their license plate
Elevator Display	To update an identified user with information about their parking or to notify a non-identified user that they were not identified and to enter their reservation number

Casual Description

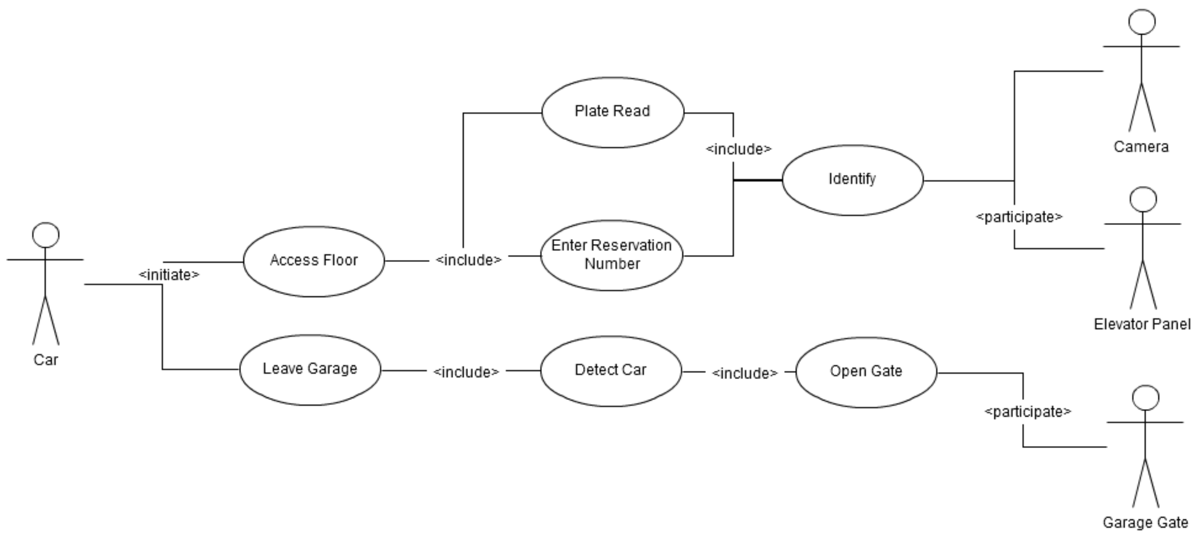
Use Case	Name	Description	Requirement
UC-1	Leave the Garage	To exit the garage via the exit gate	Exit_Gate
UC-2	Look for Vacant Spots	To see if there are vacant spots to park in (walk in)	Spot_Sensor, Vacancy_Display
UC-3	Access floor	To access the correct parking floor via the plate read or a reservation number entry	Elevator, Plate_Read, Car_To_Customer, Elevator Display
UC-4	Make Reservation	To make a reservation to park	Internet_Site, App
UC-5	Walk In/Park	To get parking without a reservation	Spot_Sensor
UC-6	Get a Rain Check	To get a rain check if overbooking happens	Rain_Check
UC-7	Change Information	To change your contact information	Internet_Site
UC-8	Make Contract	To make a contract for guaranteed parking	Internet_Site
UC-9	Set Price	To set the price for parking in the garage	Internet_Site

Use Case Diagrams

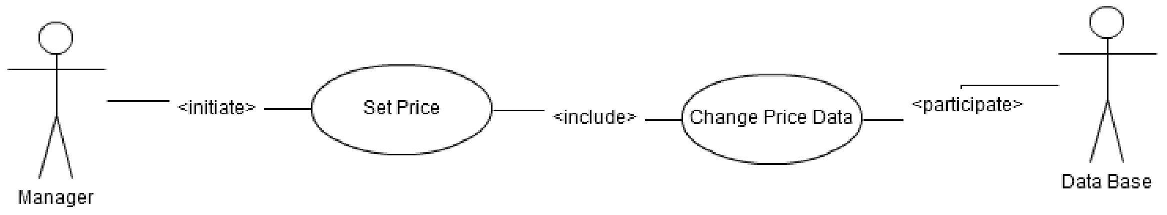
Customer (Figure 1)



Car (Figure 2)



Manager (Figure 3)



Full-Dressed Descriptions

Use Case 4

Use Case (UC-4)	Reservation
Requirements	Internet site; App
Initiating Actors	Any customers; Workers
Actor's Goal	To reserve a parking spot
Participating Actors	Internet site; Phone; Servers
Precondition	The customer logged into the internet site; The user registered their information
Post condition	The user logs out
Main Success Scenario $\longrightarrow$ $\longleftarrow$ $\longrightarrow$ $\longleftarrow$	**1.** The user chooses whether they want a long term parking situation or just for a day **2a**. If customer chose one day they are asked to put in from what time they will be there **2b**. If customer chose long-term the system asks what floor the customer wants **3a**. Customer puts in times **3b**.Customer chooses level **4a**.If spots available output information, otherwise outputs no spot available **4b**.if spots available on that level for long term, output information otherwise ask for another level

Use Case 5

Use Case (UC-5)	Walk In
Requirements	Spot_Sensor; Vacancy_Display
Initiating Actors	Customers
Actor’s Goal	To get a parking spot without a reservation
Participating Actors	Display system; Sensors; Data-base
Precondition	The system is ready to receive the customers input
Post conditions	The system marks the spot the customer has taken (if any) The system gets ready for another customer’s input
Main Success Scenario $\longrightarrow$ $\longleftarrow$ $\longrightarrow$ $\longleftarrow$	**1**. The customer puts in the times that they want to hold a spot **2a**. The system displays that a spot is open during the requested reservation time and requests a credit card number **2b**.The system displays that no spot is open and then gets ready to receive more input **3a**. The customer puts in credit card information **3b**. The customer cancels **4a**. The system confirms that the information entered is correct and then adds the customer information to the database **4b**. The system is ready for another customer

Use Case 6

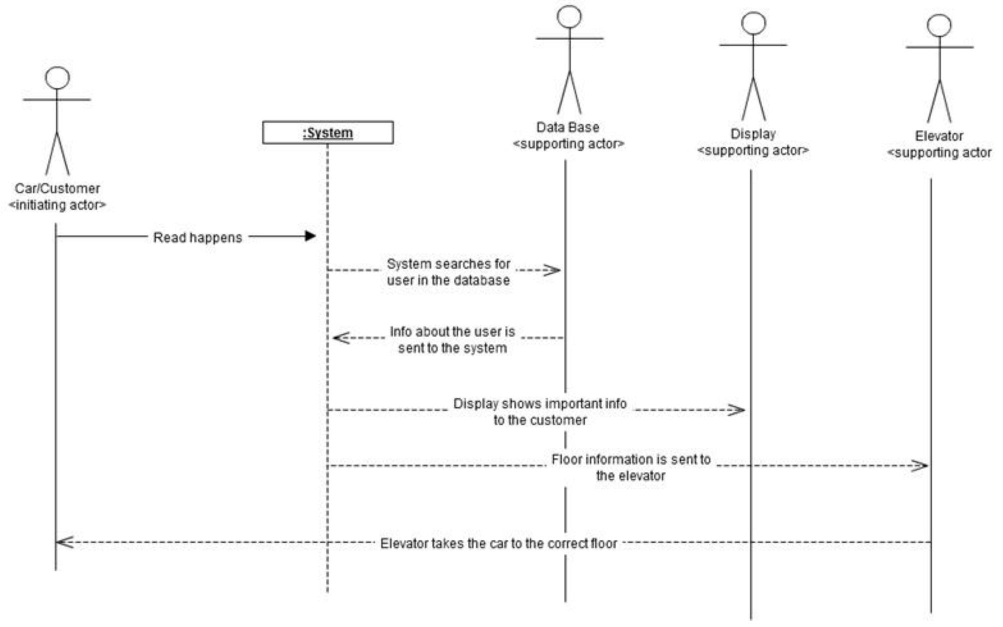
Use Case (UC-6)	Access_floor
Related Requirements	Plate_Read, Car_To_Customer, Elevator, Elevator_Bad_Read
Initiating Actors	Any of: Car, Customers
Actor’s Goal	To access the correct parking level floor
Participating Actors	Elevator, Elevator panel, Database
Precondition	The customer has tried to be identified by the system
Post conditions	The Elevator returned to ground floor and ready for another customer
Main Success Scenario $\longrightarrow$ $\longleftarrow$	Car enters elevator as Identify gives parking level info to the system The system takes the customer to designated floor
Alternate Scenarios $\longrightarrow$ $\longleftarrow$ $\longrightarrow$ $\longleftarrow$ $\longrightarrow$ $\longleftarrow$	1. Car enters elevator as Identify returns bad read to the system 2. Elevator display prompts user for reservation input 3. a) Customer puts in correct reservation number b) Customer puts in incorrect reservation number 4. a) Customer is taken to the correct level b) Wrong input counter goes up 5. a) Wrong input counter != max b) Wrong input counter === max 6. a) Loops back to reservation prompt b) Elevator display asks customer to leave, security is notified

Traceability Matrix

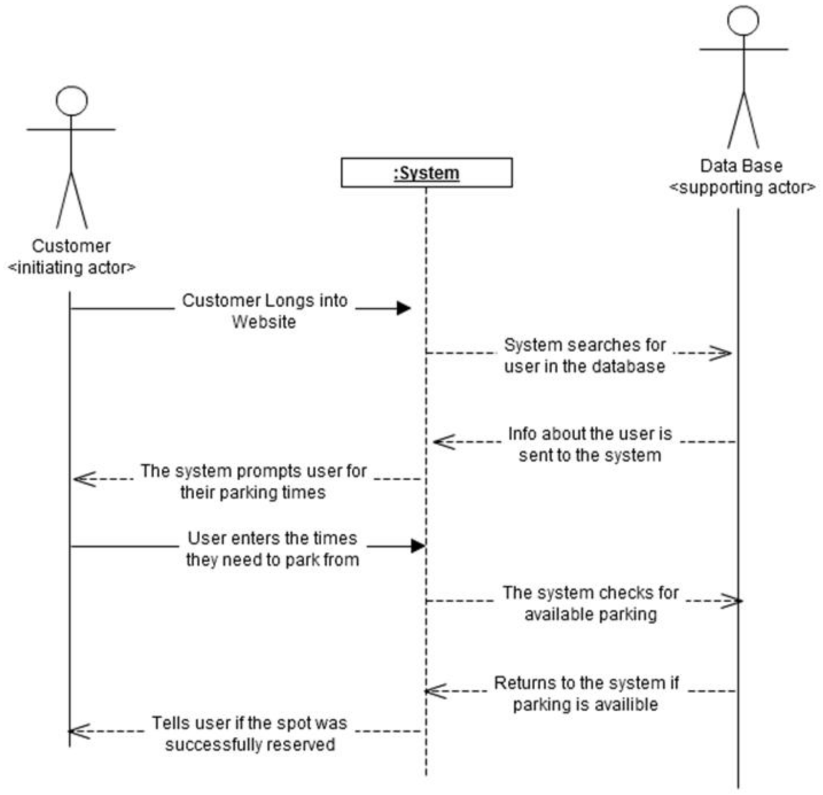
	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9
REQ-1: Plate_to_Read			X						
REQ-2: Car_to_Customer			X						
REQ-3: Spot_Sensor		X			X				
REQ-4: Vacancy_Display		X			X				
REQ-5: Elevator_Display			X						
REQ-6: Elevator_Bad_Read			X						
REQ-7: Internet_Site				X			X	X	X
REQ-8: App				X			X	X	
REQ-9: Elevator			X						
REQ-10: Rain_Check						X			
REQ-11: Exit_Gate	X								
REQ-12: Register_Account							X	X	
REQ-13: Late_NoShow	X	X		X				X	
REQ-14: Pay_Overstay	X								
REQ-15: Pay_Bill							X	X	
REQ-16: One_Way	X								
REQ-17: Recognition				X					
REQ-18: Redirect		X				X		X	
REQ-19: Sensors		X							
REQ-20: Simple_Design							X	X	X
REQ-21: Fail_Safe		X	X						
REQ-22: Last_Minute		X		X					
REQ-23: Database				X			X		
REQ-24: Ground_Spot		X							
Total PW	**15**	**24**	**23**	**19**	**5**	**9**	**18**	**25**	**8**

Sequence Diagrams

Access_Floor System Sequence Diagram (Figure 4)



Reservation System Sequence Diagram (Figure 5)



Effort Estimation using Use Case Points

Unadjusted Actor Weight (UAW)

Car: 1
Management: 1
Worker: 1
Customer: 2
Cameras: 1
Sensors:1
Vacancy Display: 2
Database:3
Internet Site:2
Phone:1
App:1
Servers:3
Elevator:1
Elevator keypad:1
Elevator display:1

UAW = 22

Unadjusted Use Case Weight

Leave the Garage:5
Look for Vacant Spots:10
Access floor:15
Make Reservation:15
Walk in/Park:10
Get a Rain Check:5
Make Contract:10
Set Price:5

UUCW=75

Unadjusted Use Case Points

UCCP= 97

Technical Complexity Factors

T1:3*2 = 6
T2:1*1 = 1
T3:1*1 = 1
T4:1*1= 1
T5:1*1=1
T6:2*.5 = 1
T7:2*.5 = 1
T8:1*2=2
T9:2*1 = 2
T10:3*1=1
T11:1*1=1
T12:1*1=1
T13:0*1=0

Technical Factor Total =19
TCF =.6+.01*19 = .79

Environment Complexity Factors

E1:5 *1.5 = 7.5
E2:3*.5 =1.5
E3:3*1 =3
E4:.5*.5= 2.5
E5:5*1=5
E6:3*2 = 6
E7:3 * -1 = -3
E8:3*-1 = -3

Environment Factor Total= 19.5

ECF =1.4*(-.03*19.5) =.815

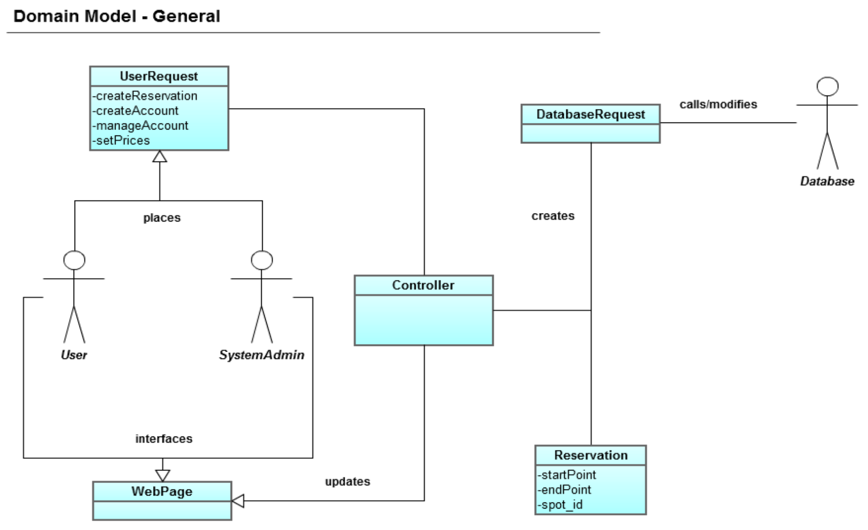
Use Case Points

UCP=97*.79*.815 = 62

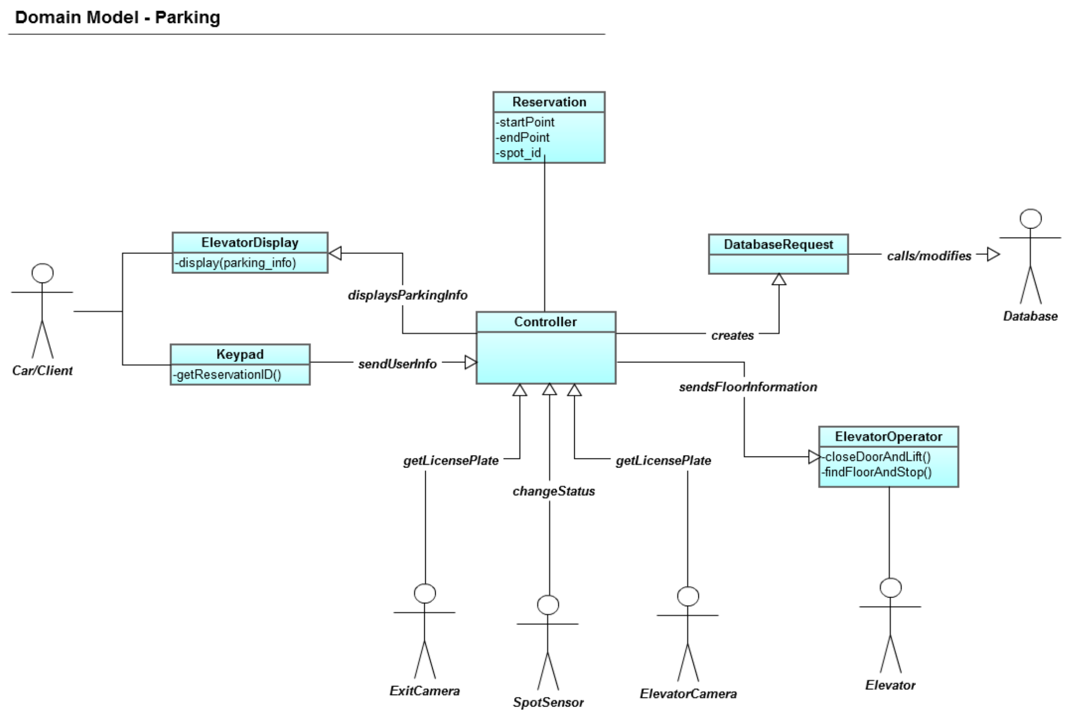
Domain Analysis

Domain Models

General Model (Figure 6)



Parking Model (Figure 7)



UC-3: Access Floor

Concept Definitions

Responsibility Description	Type	Concept Name
Container that stores all client information (including license plate, name and credit card)	K	DataBase
Takes in plate numbers and accesses Database to find which client it is associated with	D	Controller
Read the plate number	D	PlateReader
Displays useful information regarding system status to the user	D	ElevatorDisplay
Controls the elevator position	D	ElevatorController

Association Definitions

Concept Pair Name	Associated Description	Association Name
Plate_reader ⟷ Controller	Plate_reader passes the license plate number to Controller	Provides data
DataBase ⟷ Controller	Controller accesses the database and links plate to the right client	Provides data
Controller ⟷ ElevatorDisplay	Controller passes the reservation confirmation and floor number to ElevatorDisplay	Provides data
Controller ⟷ ElevatorController	Controller passes the right floor number to ElevatorController	Conveys request

Attribute Definitions

Concept	Attributes	Attribute Description
Database	client identity	Used to determine right reservation
	client license plate	Used to link a car to the right client
	client credit card	Used to pay for parking
Controller	client search	Used to search the database for right client
	license plate search	Used to search database for license plate associated with client
	GetFloor	Used to get what floor the elevator is on
ElevatorDisplay	displayRsvInfo	Used to display the spot, start time and end time of a given reservation to the user
ElevatorController	liftToFloor	Used to select right floor to go to

UC-5 Walkin/Park

Concept Definitions

Responsibility Description	Type	Concept Name
Container that stores all client information (including license plate, name and credit card)	K	Database
Checks for open spots	D	SpotController
Checks database to see if client is registered with parking garage	D	Controller
Inform user if there is a spot available or not	D	VacancyDisplay

Association Definitions

Concept Pair Name	Associated Description	Association Name
Database ⟷ Controller	Controller accesses the database and links plate to the right client	Provides Data
Controller ⟷ Vacancy Display	If no match the Controller requests to prompt user to register	Conveys Requests
SpotController ⟷ Vacancy Display	SpotController uses VacancyDisplay to show results of its actions	Provides Data

Attribute Definitions

Concept	Attributes	Attribute Description
Database	client identity	Used to determine right reservation
	client license plate	Used to link a car to the right client
	client credit card	Used to pay for parking
Controller	client search	Used to search the database for right client
	license plate search	Used to search database for license plate associated with client
Vacancy Display	user prompt	Used to prompt user to enter walking information
SpotController	spot sensors	Uses spot sensors to check for empty spots

UC-4 Make Reservation

Concept Definitions

Responsibility Description	Type	Concept Name
Container that stores all client information (including license plate, name and credit card)	K	Database
Field in which user is prompted to login	D	Login
Field in which user is prompted to register	D	Register
Knows where there is an empty spot	K	Spot checker
Logs reservation into database	D	Logger

Association Definitions

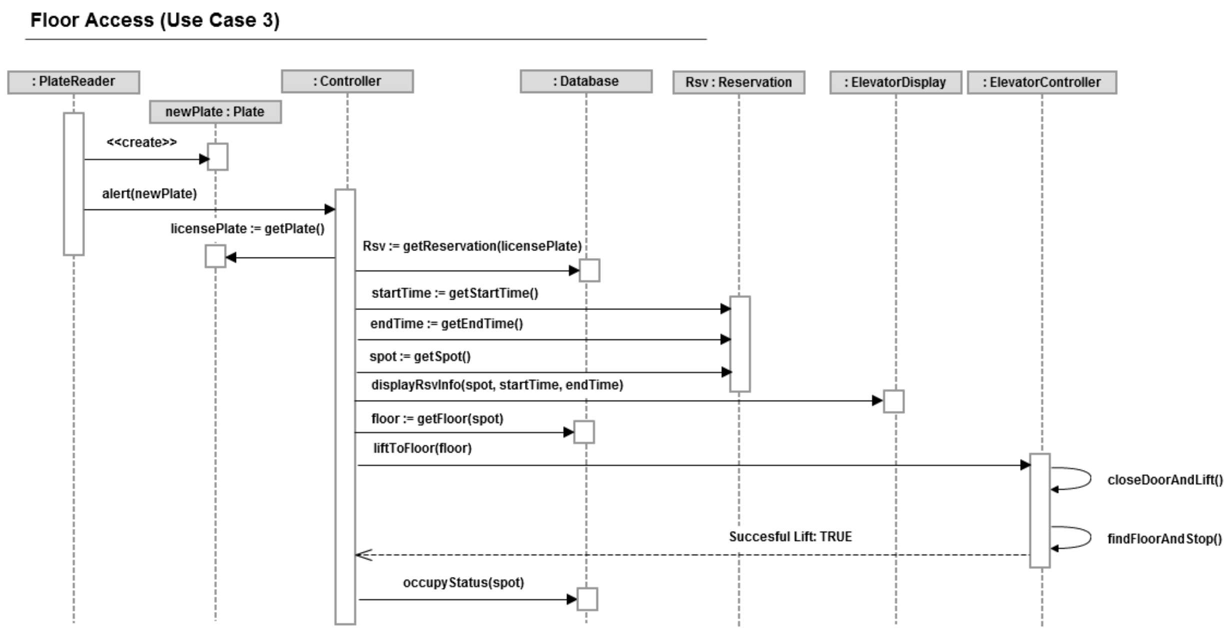
Concept Pair Name	Associated Description	Association Name
Login ⟷ Database	Login uses database to find right client	Conveys Requests
Register ⟷ Database	Register sends the client info to the database	Provides Data
Logger ⟷ Database	Logger accepts the reservation sends to database	Conveys Requests

Attribute Definitions

Concept	Attributes	Attribute Description
Database	client identity	Used to determine right reservation
	client license plate	Used to link a car to the right client
	client credit card	Used to pay for parking
Register	Field Checker	Used to make sure all the required fields are filled in
	Data passer	Used to record client in database
Login	Field Checker	Used to make sure all the required fields are filled in
	Client Search	Used to search database for entered client
	Password Checker	Used to check for the correct password
Logger	Data passer	Used to record the reservation into database
	Field Checker	Used to make sure all required fields are filled in

Interaction Diagrams

Floor Access – Use Case 3 (Figure 8)



In general, this interaction diagram has four main tasks. These responsibilities include reading the incoming license plate (R1), pulling a reservation (R2), displaying the information (R3), and taking the elevator to the proper floor (R4). The first responsibility would be assigned to the controller, via the high cohesion principle, as it is primarily responsible for delegating tasks and does so here by pulling the vehicle information. For R2, the Expert Doer design principle favors assigning R2 to the Controller as it knows the query parameters, the license plate. The process of displaying the information (R3) is also carried by the controller via the high cohesion principle of delegating the necessary tasks. Lastly, R4 is assigned to the elevator controller to lift the elevator to the proper floor using the high cohesion design

PlateReader

This object is a camera located in the elevator and it is responsible for extracting the information off the actual vehicle. It does so by creating a new plate in the system. The camera shows high cohesion since it is a physical device that handles one task, scanning the license plate off cars that enter the elevator.

Controller

The controller is the most essential portion of this interaction as it interacts with all other objects within it. When a vehicle enters the elevator, the controller will get the license plate number extracted by the camera and query the database for the cars reservation. Once it finds the reservation is gets the start point, end point, and spot location the user requested when they made the reservation. It then displays all this information on the screen in order to remind the user of their time limits and location within the garage. It then proceeds to lift the elevator to the proper level and marking that spot as occupied in the database.

Database

The database will be responsible for maintaining the information about the customer and the reservation. It meets the requirements for the expert doer principle because it holds the pertinent information pertaining to a reservation, which it therefore distributes to any object requesting it.

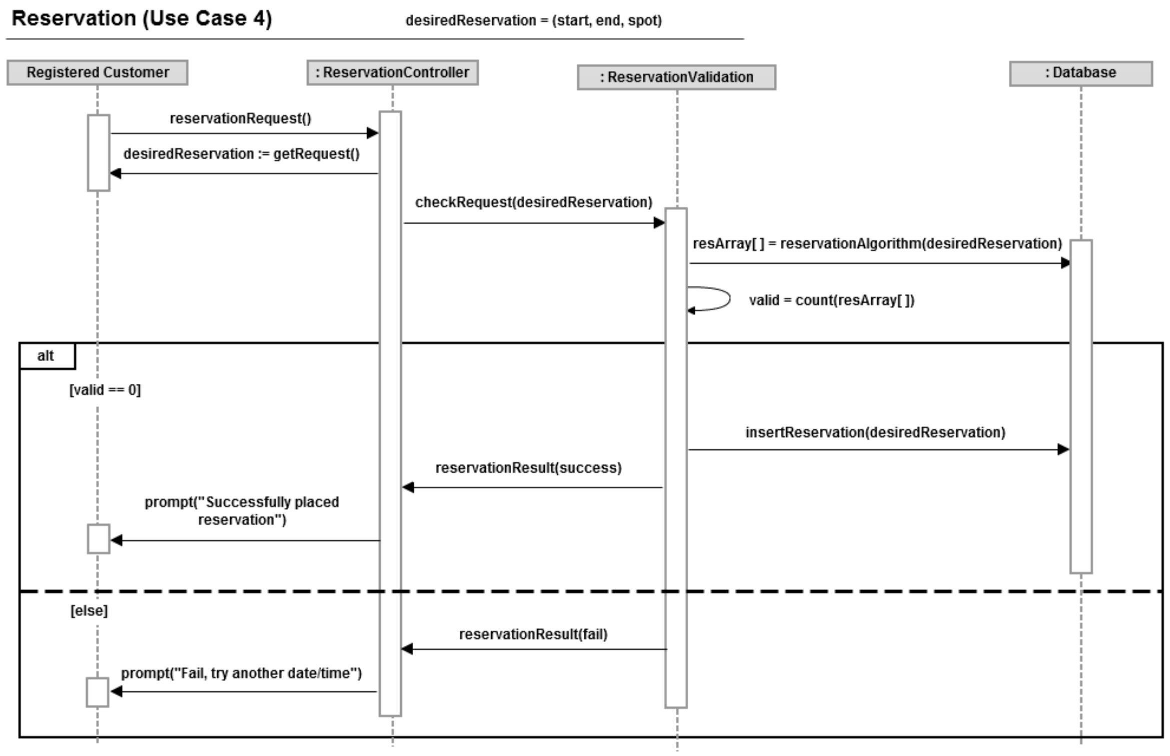
Reservation

A reservation object is created when the controller can successfully pull the reservation from the database. It is used to extract the reservation information in order to display the information to the user as well as send the elevator to the proper floor. Just like the database, once a reservation object is created, it holds the information regarding a reservation, therefore demonstrating the expert doer principle.

ElevatorDisplay and ElevatorController

The elevator display is an LCD panel located in the elevator that will display important information to the user regarding their reservation. The elevator controller is responsible for taking the elevator to the proper level as instructed by the main controller. Both these objects meet the specifications for low coupling because they don't take on too many tasks at once. Each object simply talks to the next as shown in the diagram above.

Reservation – Use Case 4 (Figure 9)


Reservation (Use Case 4)
desiredReservation = (start, end, spot)
Registered Customer
: ReservationController
: ReservationValidation
: Database
reservationRequest()
desiredReservation := getRequest()
checkRequest(desiredReservation)
resArray[ ] = reservationAlgorithm(desiredReservation)
valid = count(resArray[ ])
alt
[valid == 0]
insertReservation(desiredReservation)
reservationResult(success)
prompt("Successfully placed reservation")
[else]
reservationResult(fail)
prompt("Fail, try another date/time")

For this use case, the responsibilities include getting the reservation request from the user (R1) and checking to see if it is valid (R2). The first task is carried out by the Controller via the high cohesion principle, allowing this to be the center point in assigning tasks. R2 is assigned to ReservationValidation as the Expert Doer principle calls to assign responsibilities that contain search query parameters to the object that possesses that information.

***Note:** When a customer places their reservation, they are asked to enter a start point (time and date), end point (time and date), and a spot within the garage. This is represented as one entry, for simplicity, in the above diagram as *desiredReservation*.

Registered Customer

The registered customer will be the one initiating this interaction by placing a reservation request. After the request is processed, the customer will be informed whether that reservation is valid or not.

ReservationController

When alerted that a request for a reservation has been put in, it will prompt the user for the reservation information. Once this data is received, it will forward the request to see if there are any conflicts with existing reservations. Upon receiving the result, the controller will return to the customer whether this transaction is valid or whether the user should attempt another reservation.

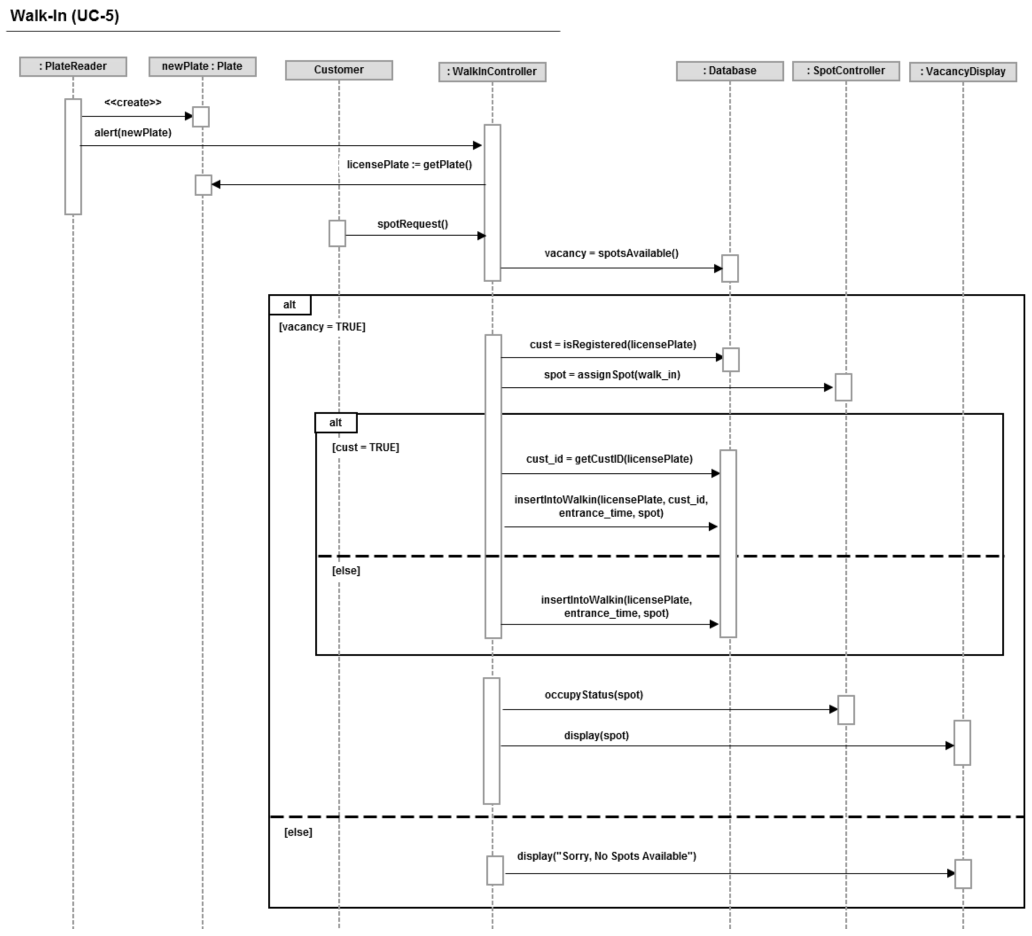
ReservationValidation

This will serve as both a connection to the database, responsible for passing messages to and from the database, and validating a reservation request by running it through an algorithm to validate it.

Database

The database will be responsible for maintaining the information about the customer and the reservation. It means the requirements for the high cohesion principle because it takes in a variety of inputs and then processes the data to return the information.

Walk-In – Use Case 5 (Figure 10)



Walk-In (UC-5)
: PlateReader
newPlate : Plate
Customer
: WalkInController
: Database
: SpotController
: VacancyDisplay
<<create>>
alert(newPlate)
licensePlate := getPlate()
spotRequest()
vacancy = spotsAvailable()
alt
[vacancy = TRUE]
cust = isRegistered(licensePlate)
spot = assignSpot(walk_in)
alt
[cust = TRUE]
cust_id = getCustID(licensePlate)
insertIntoWalkin(licensePlate, cust_id, entrance_time, spot)
[else]
insertIntoWalkin(licensePlate, entrance_time, spot)
occupyStatus(spot)
display(spot)
[else]
display("Sorry, No Spots Available")

This last interaction diagram involves checking to see if spots are available (R1), entering that information into the database (R2), and displaying information to the driver (R3). All these four tasks are carried out by the controller as it has to interact with both the driver and the database to process this entire request. The design principles here involve both high cohesion to carry out tasks and expert doer to make all the necessary database queries.

Customer

The customer is the initiator in this transaction and responsible for entering the reservation duration and billing information accurately. If there are no spots available at the time the customer arrives to the parking garage, the interaction will terminate and inform the customer accordingly.

WalkInController

The controller is responsible for interacting with the customer via the vacancy display and the system to validate all requests. It follows the expert doer principle as it knows who and what should perform each task.

Database

The database will be responsible for maintaining the information about the customer and the reservation. It means the requirements for the high cohesion principle because it takes in a variety of inputs and then processes the data to return the information.

SpotController

This will be responsible for assigning the customer with the first available spot located in the ground level parking. If the interaction reaches this point, it has been confirmed that there are spots available and this will pull the first available spot and assign it to this customer.

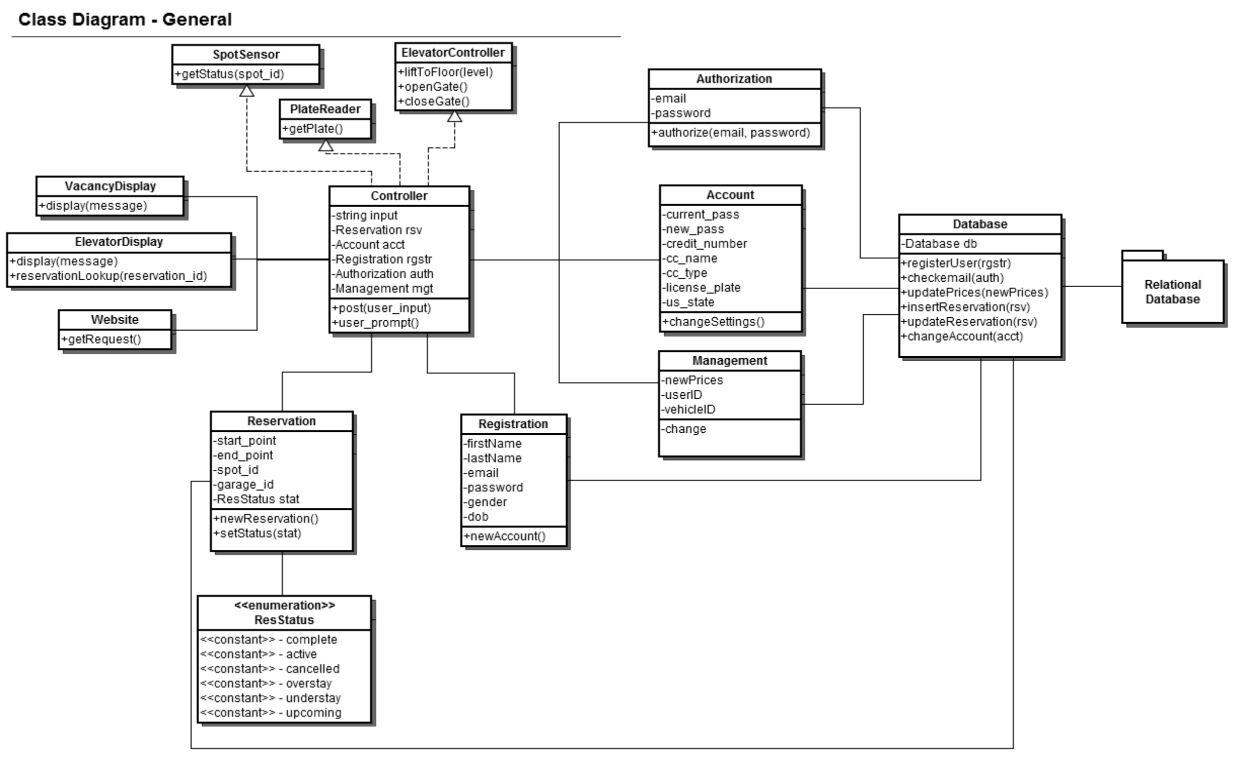
VacancyDisplay

This will be solely responsible for engagin the user to find out how long the user would like to stay as well as how to charge them for the parking. All users interact with this if they want to park on the ground level as a 'walk-in customer.'

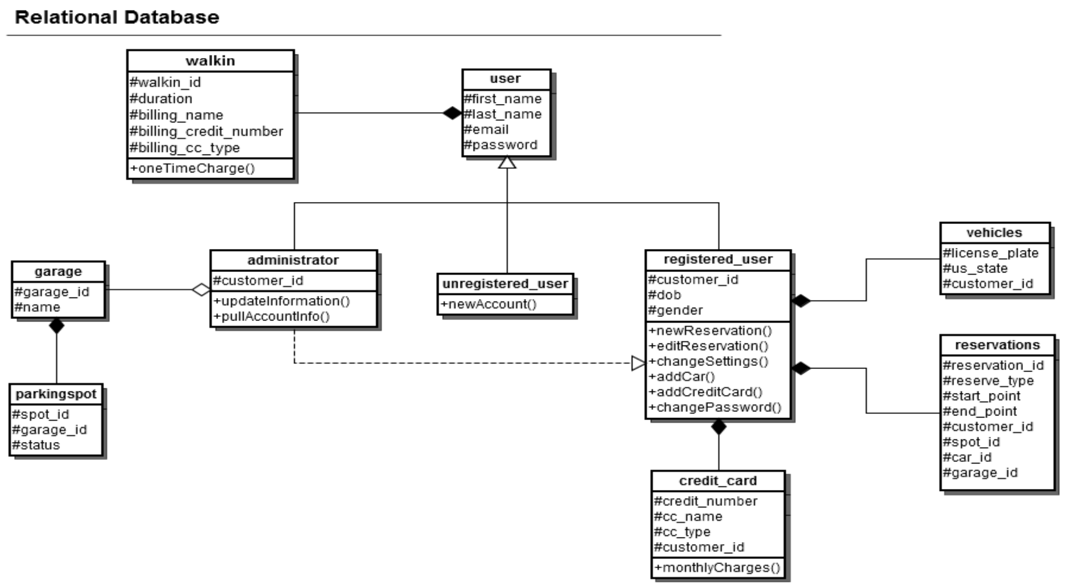
Class Diagram and Interface Specification

Class Diagram

General (Figure 11)



Database (Figure 12)



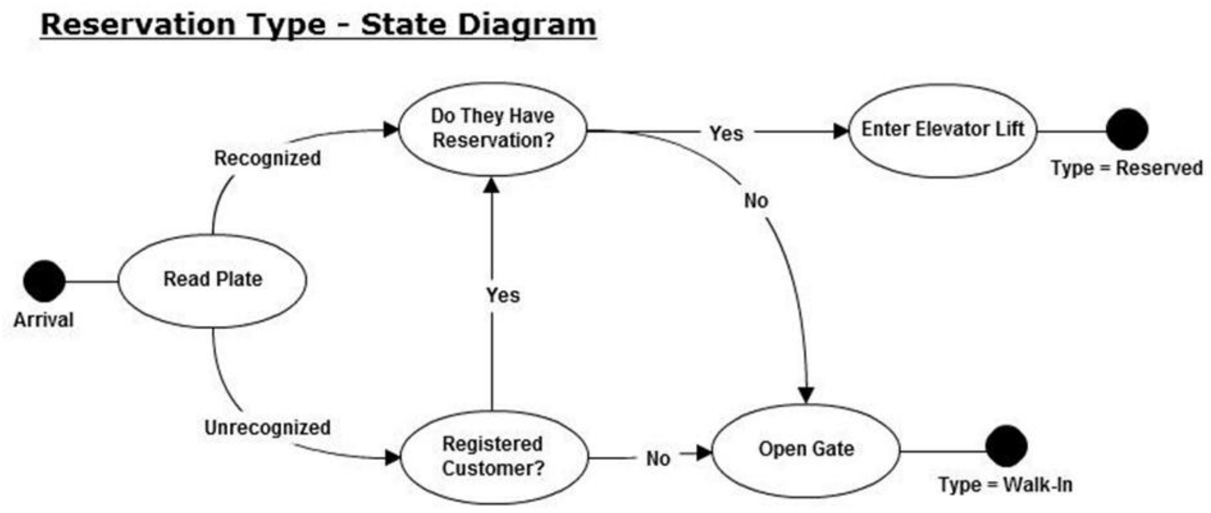
Beginning with the General Class Diagram, this figure goes on to demonstrate how the system collaborates with the different classes and hardware, located throughout the garage. The equipment the user interacts with, such as the displays, spot sensors, and elevator, all have their specified functions to send to the controller in order to place the proper requests. The controller then takes these functions and sends them over to the proper class to carry them out. Obviously most functions require input parameters that most of these items are responsible for acquiring, but of course this varies on different situations and they are explained in full detail in the individual interactions diagrams.

All the information is stored in the database and can be viewed in the second diagram. The strategy design pattern (Reference #13) is being implemented here as there are three different types of users and they all have different privileges. These different types are administrator, unregistered, and registered users. The strategy design pattern calls for selecting algorithms at runtime in order to prompt the user with the proper action/information. Each table has the necessary fields to upkeep the system and they are called based on the type of user requesting/sending information. This pattern is best demonstrated when a customer arrives at the garage. The following table demonstrates all the possibilities regarding reservation type that a customer can be labeled for upon their arrival to the garage:

Registered?	**Recognized?**	**Reserved?**	**Type**
No	No	No	Walk-In
No	No	Yes	Not Possible
No	Yes	No	Not Possible
No	Yes	Yes	Not Possible
Yes	No	No	Walk-In
Yes	No	Yes	Reserved
Yes	Yes	No	Walk-In
Yes	Yes	Yes	Reserved

The following state machine helps explain how the system distinguishes between reservations types upon the customer's arrival to the garage:

Figure 13



In order to ensure the authenticity of a user, an authentication proxy design pattern is used to confirm the user is accessing the proper information. The administrator is assigned to the system manually and given a user name and password for their account. Registered and unregistered users are the customers themselves who can go about getting a parking spot differently, as seen above. Authenticity of these users is essential and the table above builds off of the state machine, above, to show the credentials the system considers when identifying users.

Data Types and Operation Signatures

All the classes above are broken down a little better below. Every class contains the attributes associated with it as well as the operations they carry out.

customer:

- Attributes:
  -int customer_id
  -string first_name
  -string last_name
  -string email
  -string password
  -date dob
  -enum gender('Male', 'Female')
- Operation: (all operations attach customer to any created objects)
  +addCar() : creates a car and inserts into database
  +addCreditCard() : creates a credit card and inserts into database
  +newReservation() : creates a reservation and inserts into database

parkingspot:

- Attributes:
  -int spot_id
  -int garage_id
  -enum status('vacant', occupied', 'reserved')
- Operation:
  +updateStatus()

garage:

- Attributes:
  -int garage_id
  -string name
- Operation:
  +createGarage( )
  +insertSpot( )

creditcard:

- Attributes:
  -string credit_number
  -int customer_id
  -string cc_name
  -string cc_type
- Operation:
  +createCreditCard( )

car:

- Attributes:
  -car_id
  -license_plate
  -us_state
  -customer_id
- Operation:
  +createCar( )
  +removeCar( )

reservations:

- Attributes:
  -int reservation_i
  -time start_time
  -time end_time
  -date reserve_date
  -int customer_id
  -int car_id
  -int garage_id
  -int spot_id
  -enum reserve_type('active', 'canceled', 'overstay', 'understay', 'completed', 'upcoming')
- Operation:
  +createReservation( ) : creates a reservation record
  +cancelReservation( ) : voids the reservation placed
  +editReservation( ) : for upcoming reservations only, changes can be made

Traceability Matrix

All the classes seen above were derived from the traceability matrix, seen earlier on page 17. It is here that all the domain concepts were merged with the use cases in order to see how the system will function. Once this was set up, the classes began to form starting with user and from what point of view he/she would see the system. The views were dependent on the user being an administrator, unregistered, or registered user. All the essential equipment used throughout the garage also got involved as these pieces of equipment pulled essential information from the customer and/or garage in order to efficiently maintain the web service.

The remainder of the classes were focused around the database as it does hold the key pieces to upkeep all this information. These classes included creating credit cards, vehicles, and the garages along with their spots as well. They each have their own respective operations within them to insert, remove, and upkeep accurate data. This is reflected upon the traceability matrix as the database is involved with practically all the use cases created.

Object Constraint Language (OCL) Contracts

Authorization
Invariants - A user needs access to an account.
Pre-Conditions - The user acquires the proper log on credentials to access their account.
Post-Conditions - The user can make changes as well as place reservations.

Account
Invariants - Must be a valid registered user.
Pre-Conditions - The user must meet the restrictions placed on certain changes made to an account (such as two credit cards per customer).
Post-Conditions - Any valid transactions are recorded and then displayed on the account page.

Reservation
Invariants - Must be a valid registered user and place a valid registration request.
Pre-Conditions - The garage must be able to allocate a spot for the requested time interval.
Post-Conditions - Garage vacancy decreases.

Registration
Invariants - Can be any unregistered user
Pre-Conditions - The user can provide the basic information needed to create an account.
Post-Conditions - The new user's basic information is entered into the database and that user is re-directed to their account page.

User
Invariants - Must be a registered user with a confirmed email account.
Pre-Conditions - Needs to be registered with basic user information.
Post-Conditions - Once entered into the system, the user can further manage their account by entering vehicles, credit cards, and other registered user privileges.

Garage
Invariants - Contains spots available for parking.
Pre-Conditions - Spots are available.
Post-Conditions - As parking spots begin to fill, the vacant space decreases till it hits zero.

Vehicle
Invariants - Must be a valid vehicle (contain a valid license plate as well as a licensed driver)
Pre-Conditions - The vehicle has not been registered by another user.
Post-Conditions - The vehicle is entered into the database, under the respective user, and used labeled with a unique name, to that user.

System Architecture and System Design

Architectural Styles

An architectural style provides a framework for a system which includes software components, its properties, and the relationships among them. The most useful style for the parking garage automation is the event-driven architecture. This software architecture pattern promotes the production, detection, consumption of, and reaction to events. One specific event is reserving a parking spot. It is the system's focus, and thus is the reason why event-driven architecture is most suitable.

When a parking spot is reserved, it causes software components to change and others to react. The table below illustrates the system's cause and effect to an event.

Event	**Before**	**After**
Reserve	none	reserved
Cancel	reserved	cancel
Extend	reserve, extend	reserve, extend
Overstay	parked	overstay
Understay	parked	understay
Missed	reserved	missed
Completed	reserved, overstay, understay, missed	completed

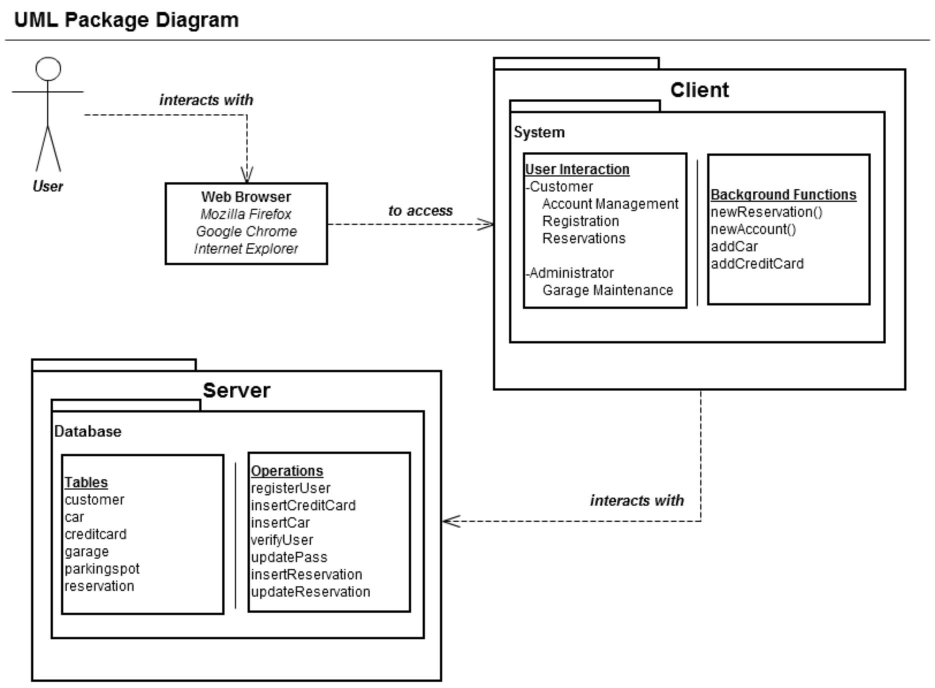
Event-driven architecture is geared towards unpredictable and asynchronous environments. This is common when a customer interacts with the system. By using event-driven architecture, the parking garage automation sustains a stable and responsive system.

Identifying Subsystems

As this website will be an online service, it will require a client to interact with the user and a server to maintain a record of all the interactions and requests. This is accomplished by the user accessing the web client through a web browser using the http protocol. The client is comprised of HTML, CSS, and PHP and will interact with the server within those limits. User interaction will take place on the client side of the system and make function calls to the server in the background. Once in the server, the client will update the tables stored through the operations location within the database class.

This subsystem combines the classes seen in the class diagram (Figures 11 and 12) in order to design a subsystem that remains hidden from the rest of the model. These two systems comprise of the server, which is essentially the database containing all the information, and the client, which would be any aspect of the system the user interacts with. The diagram below demonstrates this by showing the user interacting with the web browser to access the client subsystem. This system grants the user certain privileges like modifying account settings, placing/editing reservations, and just getting general information on the garage itself. All these tasks are carried out by the client in the background through its interaction with the server that actually stores it. This architectural style resembles that above by encompassing multiple classes to create the client and all the information tables to create the server, or database.

Figure 14



Mapping Subsystems to Hardware

The model of the System calls for it to be ran on more than one computer. These computers can be broken into two categories; Web servers and client computers.

Web Servers

The web server will be used as the main database in which all of the sensors will store the data in. Majority of the source code will be run within the web servers. All client account information will be stored within the web servers.

In addition the web server will have a container that stores empty as well as occupied parking spots. The web server will keep track of all of the reservations and contracts. It will store a history of all of the reservations and overstays so the system can better predict occupancy.

The web server will store all of the prices and fees and issue them accordingly. It will have the ability to contact the client via email to send reminders; such as a reminder that a reservation is coming up and a reminder that a payment is due.

Client Computers

The client computers will access the web servers via a browser website that is running on html code. The client computers include the elevator display, vacancy display and client's personal computers.

The client's personal computers will access the web servers to set up an account, make a payment, make a reservation, create a contract and edit account information. There is an administrator mode that can be accessed when administrator accounts login. From administrator mode the user will be able to set prices and policies as well as manage delinquent accounts.

The elevator display and vacancy display will access the web servers to retrieve system status. This includes account information as well as vacant parking spots. It will access the web servers in order to display reservation information to the client.

Persistent Data Storage

MySQL was chosen as the database to maintain all the information acquired by the system. Seven tables, customer, creditcard, car, garage, parkingspot, walkin and reservation, will be used to hold all the relevant information pertaining to the user and their requests. The tables are seen below along will their respective attributes and detail.

DATABASE TABLES				
Field	**Type**	**Null**	**Extra**	**Links**
customer				
customer_id	int	no	auto increment	
first_name	varchar(45)	no		
last_name	varchar(45)	no		
email	varchar(45)	no	unique	
password	varchar(45)	no		
dob	date	no		
gender	enum(‘Male’, ‘Female’)	no		
status	enum(‘Active’, ‘Inactive’)	no		
creditcard				
credit_id	int	no		
credit_number	varchar(45)	no		
customer_id	int	no		customer->customer_id
cc_name	varchar(45)	no		
cc_type	enum(‘Visa’, ‘Mastercard’, ‘American Express’)	no		
name_given	varchar(45)	no		
car				
car_id	int	no	auto increment	
car_name	varchar(45)	no		
license_plate	varchar(45)	no		
us_state	varchar(3)	no		
customer_id	int	no		customer->customer_id
garage				
garage_id	int	no		
name	varchar(45)	no		

parkingspot				
spot_id	int	no		
status	enum('vacant', occupied', 'reserved')	no	default: 'Vacant'	
garage_id	int	no		garage->garage_id
walkin				
walkin_id	int	no	Auto increment	
license_plate	varchar(45)	no		
us_state	varchar(3)	no		
customer_id	int	no		customer->customer_id
spot_id	int	no		parkingspot->spot_id
garage_id	int	no		garage->garage_id
entrance	datetime	no		
departure	datetime			
reservation				
reservation_id	int	no	auto increment	
reserve_type	enum('active', 'canceled', 'overstay', 'understay', 'missed', 'completed', 'upcoming')	no		
start_point	datetime	no		
end_end	datetime	no		
spot_id	int	no		parkingspot->spot_id
customer_id	int	no		customer->customer_id
credit_id	int			creditcard->car_id
car_id	int	no		car->car_id
garage_id	int	no		garage->garage_id
arrive	datetime			
depart	datetime			
priceplan				
priceplan_id	int	no	auto increment	
garage_id	int	no		garage->garage_id
registered	decimal	no		
unregistered	decimal	no		
raincheck				
check_id	int	no	auto increment	
duration	decimal	no		
customer_id	int	no		customer->customer_id

Network Protocol

Our service will be offered as a website and it will be hosted off of a single server/machine, therefore, there will not be a need to utilize any other communication protocol other than HTTP. The system is still being built but it will be built in such a way that it can be adapted quickly; in the future, if the need for security arises, the system can be modified to accept other communication protocols. Also, as per the problem statement, the system is designed such that it communicates with our database, which stores all client names and information.

Global Control Flow

Execution Orderness:

Our service is two-fold; it is event-driven in that customers are required to enter in information regarding themselves and the vehicle that they will be parking in the garage. It is procedure-driven in that as customers approach the garage, the camera based license-plate recognition software checks each and every license plate for reservations; the floor sensors that act to verify if a parking spot is occupied or vacant is also procedure-driven.

Time Dependency:

Our system does employ timers; they serve an important role in that they measure the duration a customer has utilized the parking garage service. They are also used to measure when a customer has checked in to the garage and they also serve to inform the management of duration of the over-stay, if the customer over-stay's his reservation.

Concurrency:

No.

Hardware Requirements

Hard-drive: A hard-drive is needed for storing customer information. Assuming that customer information per customer costs about a Megabyte of memory and there are about 200 parking spots which would create a maximum of 71,200 customers a year (200 * 356), then the hard-drive should be at a minimum of 100,000 MB per year.

Elevator Display: The display in the elevator should be a minimum of 800x600 resolution since it is a little smaller and closer to the customer.

Vacancy Display: The display in the lobby should be bigger so more people can see it at once. This should be a minimum of 1280x720 resolution.

Network Bandwidth: Since the memory cost of the customer was estimated at 1MB, a bandwidth of 50 KB per second should be the minimum bandwidth. This would mean that a customer would have to wait at the most 20 seconds for their memory to transfer.

Elevator Keypad: The elevator keypad should consist of only number buttons, an enter button, a backspace button, and a cancel button since it is only meant to put in reservation numbers.

Walk-In/Park Computer: A basic computer is required here as the only tasks it will need to do it carry out simple data base calls. This doesn't require too much memory and the calls are instant.

Sensor: A light sensor or a ultrasonic sensor would both suffice to sense the vacancy of each parking spot while a pressure sensor is needed for the elevator.

Servers: There should be a minimum of one server to handle the garages reservations.

Cameras: The cameras do not have to be too expensive as long as they can differentiate between license plates within a 100 meter distance.

Android Phone: For the customers that choose to have the app, they will need an Android based smart phone.

Computer: To access the website, a customer will need a computer to create a reservation.

Central Computer: To run the whole garage in parallel, the garage will need a computer solely to run all of the garage programs.

Algorithms and Data Structures

Algorithms

Scheduling a Reservation

The complex algorithm used to check for possible reservations is more accurately described as a database query. Like an algorithm, the query follows a protocol by communicating with the database in a specific manner. The protocol that is followed consists of steps such as retrieving, inputting, and verifying user inputs with the database. When a user places a reservation via the website, they select when it will begin, when it will end, and which spot they would like to park in. Once this selection is made and submitted, the database query is put together to check to see if this reservation can be made.

This is accomplished by establishing a couple important ground rules with the database itself. For starters, the start and end points for a reservation will be stored into the reservation table with a DATETIME type. These entries are stored in the following format:

'YYYY-MM-DD HH:MM:SS'

This simplifies the process because the values in these fields can be compared directly via MySQL and don't need to go through any concatenation. The system begins by pulling all the reservations made for a particular spot, called *requested_spot*, via the following query:

*SELECT * FROM reservation WHERE spot_id=request_spot*

With this query, we pull all the reservations in our database related to that one spot. It will serve the first half of our nested, complex query. The next step is to compare the start and end points themselves. For demonstration purposes, let's call the start point *newStart* and the end point *newEnd*. Again, these two points are of the datetime format. All time comparisons fall into one of five total categories:

1. The best case scenario, when there is no reservation overlap and the requested reservation can be made.

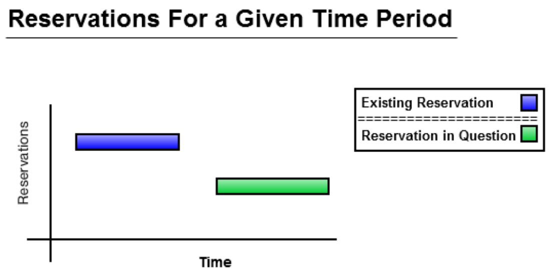


Figure 15

2. This next case occurs when a user wishes to place a reservation that starts before an existing reservation and ends after it. This overlap is an issue and the system will inform the user that it is not possible.

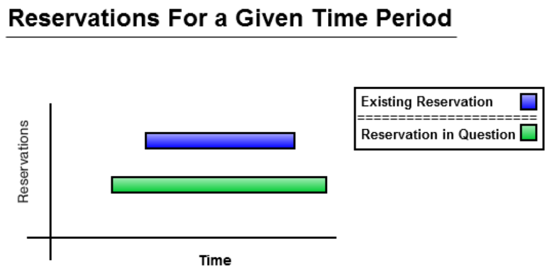


Figure 16

3. The reverse of the previous scenario arises when a user wishes to place a reservation that falls right in the middle of an existing reservation. Just like the previous case, this is an overlap and the system will inform the user it is not possible.

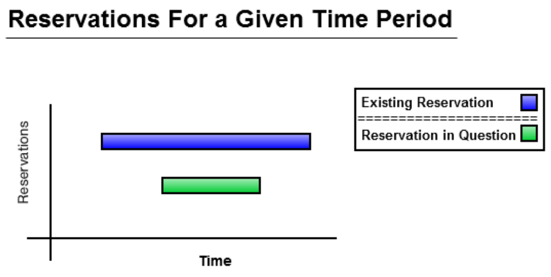


Figure 17

4. Another instance of an overlap will occur when a user attempts to place a reservation that starts right before an existing reservation is about to end. This is overlap is not acceptable and the system will inform the user.

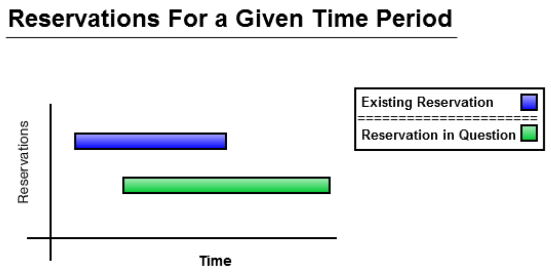


Figure 18

5. Lastly, when a user wishes to place a reservation that will start right before an existing reservation is set to start. Once again, the system will inform the user this is not acceptable.

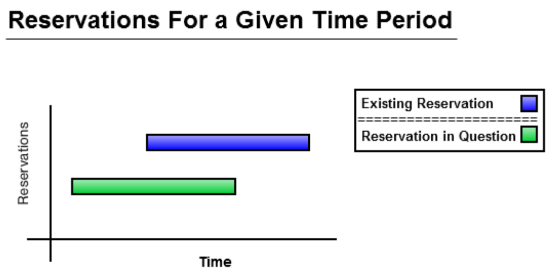


Figure 19

The following query is a generic example that will pull any of the four possible overlap cases:

mysql> *select * from reservation R where (R.start >= newStart and R.end <=newEnd) or (R.start <= newStart and R.end >= newEnd) or (R.start >= newStart and R.start <= newEnd) or (R.end >= newStart and R.end <= newEnd);*

Once all this is put together and the final query is made, if there are any results, there is an overlap and the requested reservation cannot be made.

Data Structures

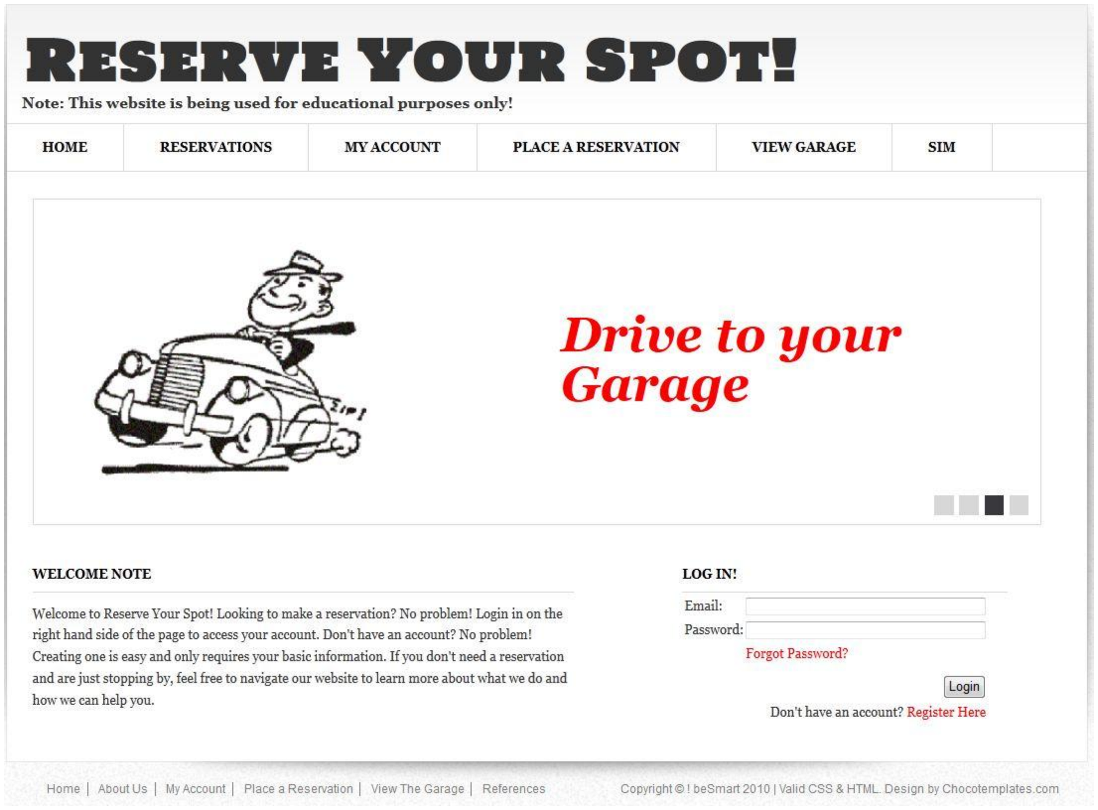
As mentioned earlier, all the information the system obtains will be stored and maintained in a database. MySQL will be used to make the necessary queries and modifications, as well as perform the algorithms described later in this section. It was chosen as it is the world's most used relational database management system.

As some database queries will pull multiple entries at once, an array data structure is used to store all these search query results in one place. It's really the only data structure that was implemented due to its simplicity and efficiency. Future changes to this system will include a map of the garage at the current time to give the user a better understanding of what is available. As this portion hasn't been implemented yet, any additional data structures can't be guaranteed just yet.

User Interface Design and Implementation

Home Page

Figure 20


RESERVE YOUR SPOT!
Note: This website is being used for educational purposes only!
HOME
RESERVATIONS
MY ACCOUNT
PLACE A RESERVATION
VIEW GARAGE
SIM
Drive to your Garage
LOG IN!
Email:
Password:
Forgot Password?
Login
Don't have an account? Register Here
Home | About Us | My Account | Place a Reservation | View The Garage | References
Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

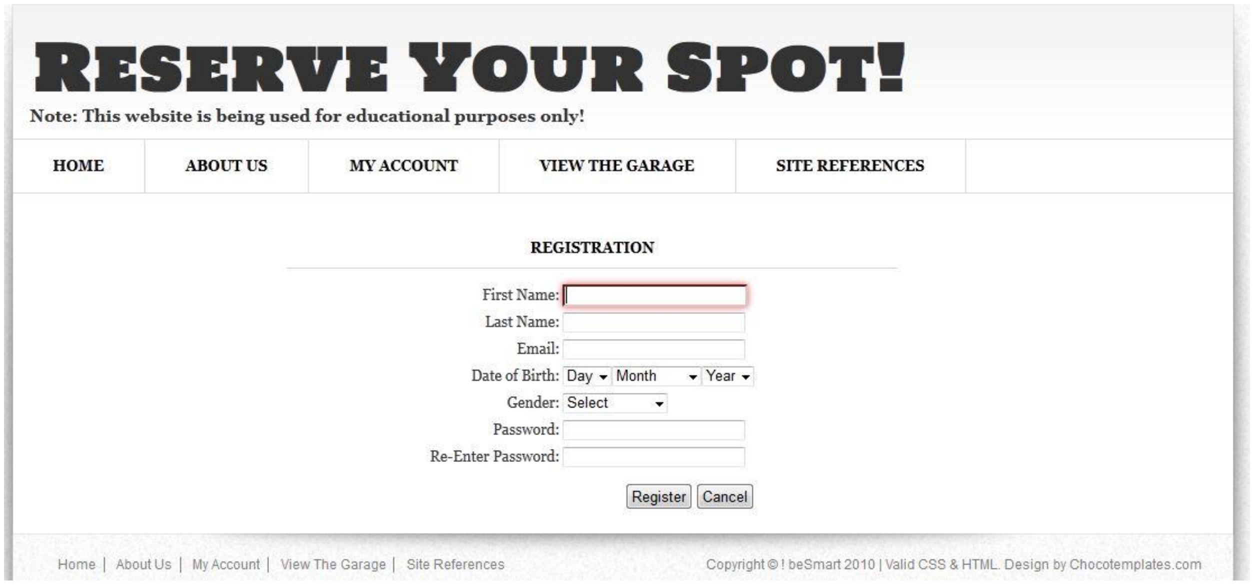
WELCOME NOTE

Welcome to Reserve Your Spot! Looking to make a reservation? No problem! Login in on the right hand side of the page to access your account. Don't have an account? No problem! Creating one is easy and only requires your basic information. If you don't need a reservation and are just stopping by, feel free to navigate our website to learn more about what we do and how we can help you.

This is the first page users see when the go to the website. Can't really do much on this page other than log in and make requests to other pages. In addition to this, this page provides a brief welcome note to provide the user a very brief idea of what they can do here.

Registration

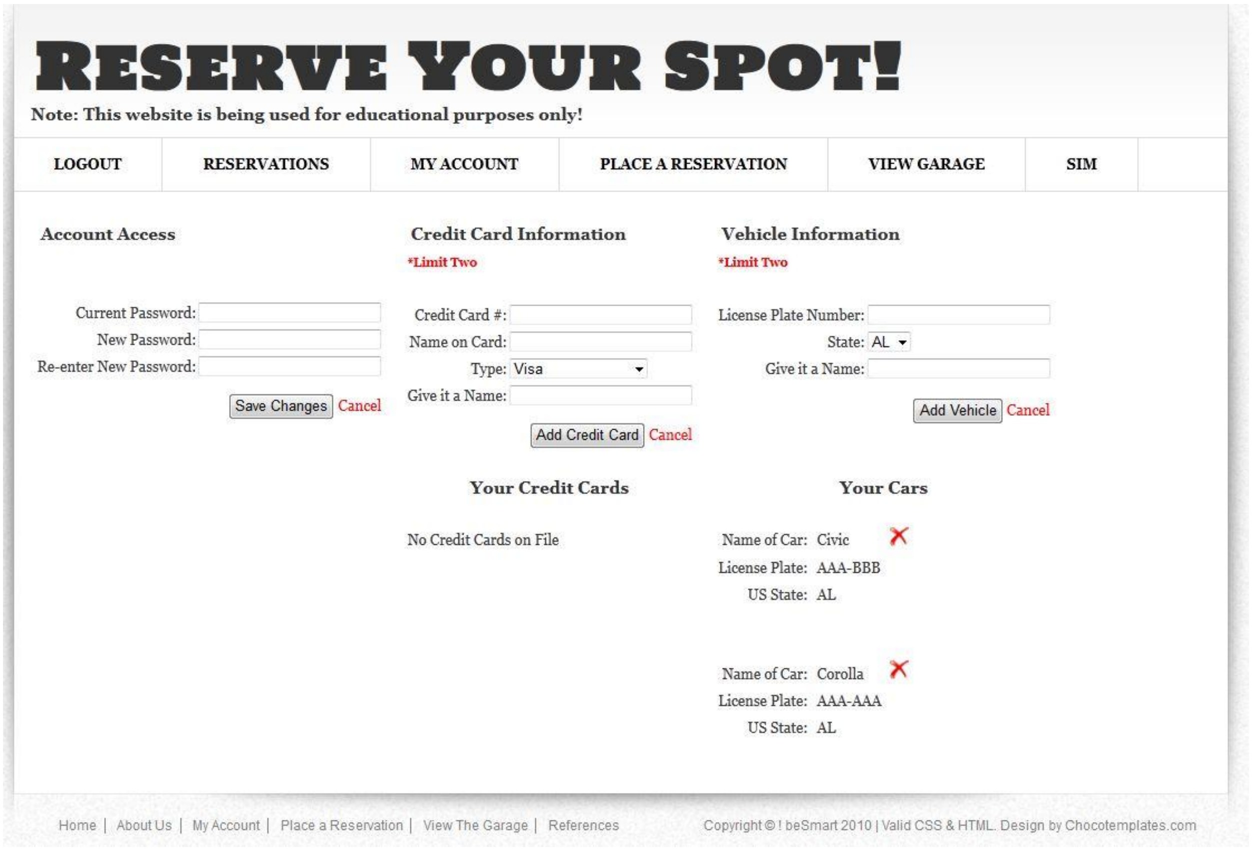
Figure 21


RESERVE YOUR SPOT!
Note: This website is being used for educational purposes only!
HOME
ABOUT US
MY ACCOUNT
VIEW THE GARAGE
SITE REFERENCES
REGISTRATION
First Name:
Last Name:
Email:
Date of Birth: Day Month Year
Gender: Select
Password:
Re-Enter Password:
Register
Cancel
Home | About Us | My Account | View The Garage | Site References
Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

In order to keep the registration process quick and painless, there isn't much information needed to register. Upon registering, a confirmation email will be sent to the email registered on this page. All fields must be entered and the email address has to be unique in order for the account to successfully get created. All passwords stored are encrypted prior to being entered in the database.

My Account

Figure 22



RESERVE YOUR SPOT!
Note: This website is being used for educational purposes only!
LOGOUT
RESERVATIONS
MY ACCOUNT
PLACE A RESERVATION
VIEW GARAGE
SIM
Save Changes
Cancel
Add Credit Card
Cancel
Add Vehicle
Cancel
Home | About Us | My Account | Place a Reservation | View The Garage | References
Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

Account Access

Current Password:
New Password:
Re-enter New Password:

Credit Card Information

*Limit Two

Credit Card #:
Name on Card:
Type: Visa
Give it a Name:

Vehicle Information

*Limit Two

License Plate Number:
State: AL
Give it a Name:

Your Credit Cards

No Credit Cards on File

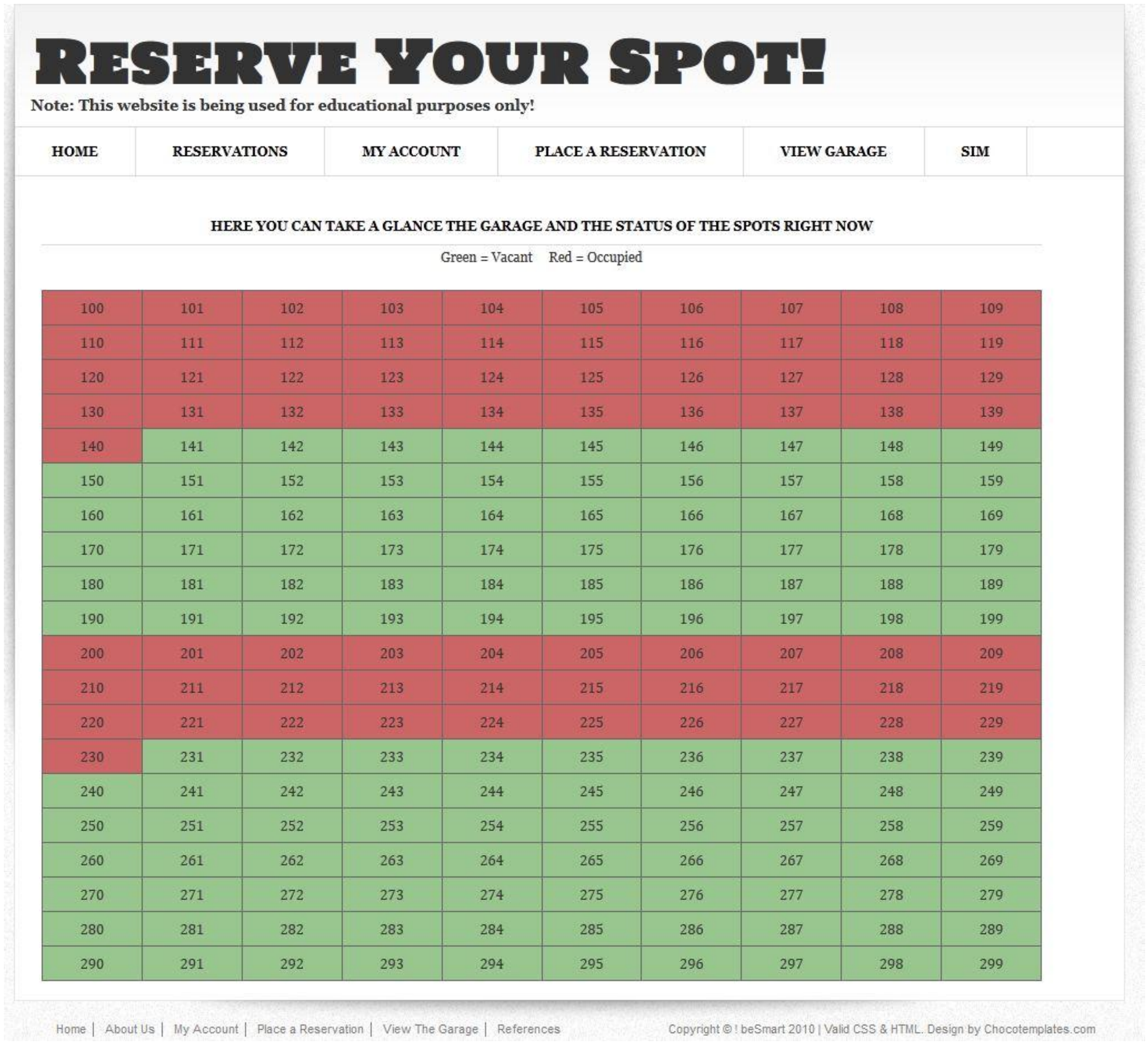
Your Cars

Name of Car:	Civic
License Plate:	AAA-BBB
US State:	AL
Name of Car:	Corolla
License Plate:	AAA-AAA
US State:	AL

After logging in or registering a new account, the user is re-directed towards their account page. On this page the user has the option to change their password, insert a credit card, and insert a vehicle. Once a credit card or vehicle is entered, they will be displayed below with some of the information pertaining to that one item. This is done to ensure privacy and not display information like credit card numbers. As the user is limited to two credit cards and two vehicles, they have the option of deleting one of the existing ones from the database.

View Garage

Figure 23



RESERVE YOUR SPOT!

Note: This website is being used for educational purposes only!

HERE YOU CAN TAKE A GLANCE THE GARAGE AND THE STATUS OF THE SPOTS RIGHT NOW

🟩 Green = Vacant 🟥 Red = Occupied

🟥 100	🟥 101	🟥 102	🟥 103	🟥 104	🟥 105	🟥 106	🟥 107	🟥 108	🟥 109
🟥 110	🟥 111	🟥 112	🟥 113	🟥 114	🟥 115	🟥 116	🟥 117	🟥 118	🟥 119
🟥 120	🟥 121	🟥 122	🟥 123	🟥 124	🟥 125	🟥 126	🟥 127	🟥 128	🟥 129
🟥 130	🟥 131	🟥 132	🟥 133	🟥 134	🟥 135	🟥 136	🟥 137	🟥 138	🟥 139
🟥 140	🟩 141	🟩 142	🟩 143	🟩 144	🟩 145	🟩 146	🟩 147	🟩 148	🟩 149
🟩 150	🟩 151	🟩 152	🟩 153	🟩 154	🟩 155	🟩 156	🟩 157	🟩 158	🟩 159
🟩 160	🟩 161	🟩 162	🟩 163	🟩 164	🟩 165	🟩 166	🟩 167	🟩 168	🟩 169
🟩 170	🟩 171	🟩 172	🟩 173	🟩 174	🟩 175	🟩 176	🟩 177	🟩 178	🟩 179
🟩 180	🟩 181	🟩 182	🟩 183	🟩 184	🟩 185	🟩 186	🟩 187	🟩 188	🟩 189
🟩 190	🟩 191	🟩 192	🟩 193	🟩 194	🟩 195	🟩 196	🟩 197	🟩 198	🟩 199
🟥 200	🟥 201	🟥 202	🟥 203	🟥 204	🟥 205	🟥 206	🟥 207	🟥 208	🟥 209
🟥 210	🟥 211	🟥 212	🟥 213	🟥 214	🟥 215	🟥 216	🟥 217	🟥 218	🟥 219
🟥 220	🟥 221	🟥 222	🟥 223	🟥 224	🟥 225	🟥 226	🟥 227	🟥 228	🟥 229
🟥 230	🟩 231	🟩 232	🟩 233	🟩 234	🟩 235	🟩 236	🟩 237	🟩 238	🟩 239
🟩 240	🟩 241	🟩 242	🟩 243	🟩 244	🟩 245	🟩 246	🟩 247	🟩 248	🟩 249
🟩 250	🟩 251	🟩 252	🟩 253	🟩 254	🟩 255	🟩 256	🟩 257	🟩 258	🟩 259
🟩 260	🟩 261	🟩 262	🟩 263	🟩 264	🟩 265	🟩 266	🟩 267	🟩 268	🟩 269
🟩 270	🟩 271	🟩 272	🟩 273	🟩 274	🟩 275	🟩 276	🟩 277	🟩 278	🟩 279
🟩 280	🟩 281	🟩 282	🟩 283	🟩 284	🟩 285	🟩 286	🟩 287	🟩 288	🟩 289
🟩 290	🟩 291	🟩 292	🟩 293	🟩 294	🟩 295	🟩 296	🟩 297	🟩 298	🟩 299

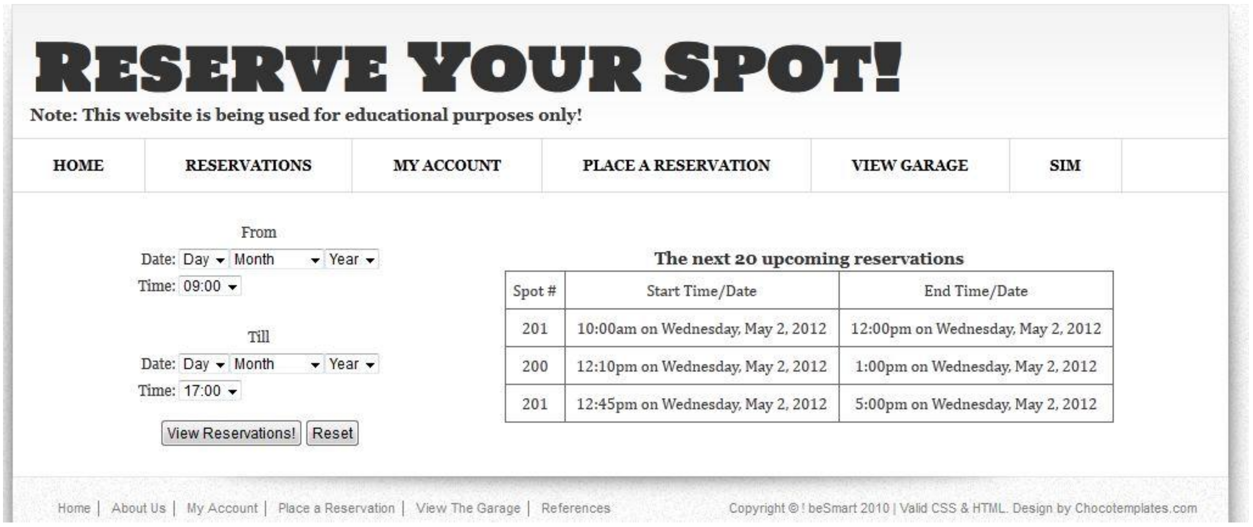
Home | About Us | My Account | Place a Reservation | View The Garage | References

Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

On this page the user can take a look at the current status of the garage. Spots starting with a one, 1xx, denote walk in parking and spots starting with a two or higher, denote reservation parking. All spots colored in red are currently occupied and all spots colored in green denote a vacant spot.

View All Reservations

Figure 24


RESERVE YOUR SPOT!
Note: This website is being used for educational purposes only!
HOME
RESERVATIONS
MY ACCOUNT
PLACE A RESERVATION
VIEW GARAGE
SIM
From
Date: Day Month Year
Time: 09:00
Till
Date: Day Month Year
Time: 17:00
View Reservations!
Reset
Home | About Us | My Account | Place a Reservation | View The Garage | References
Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

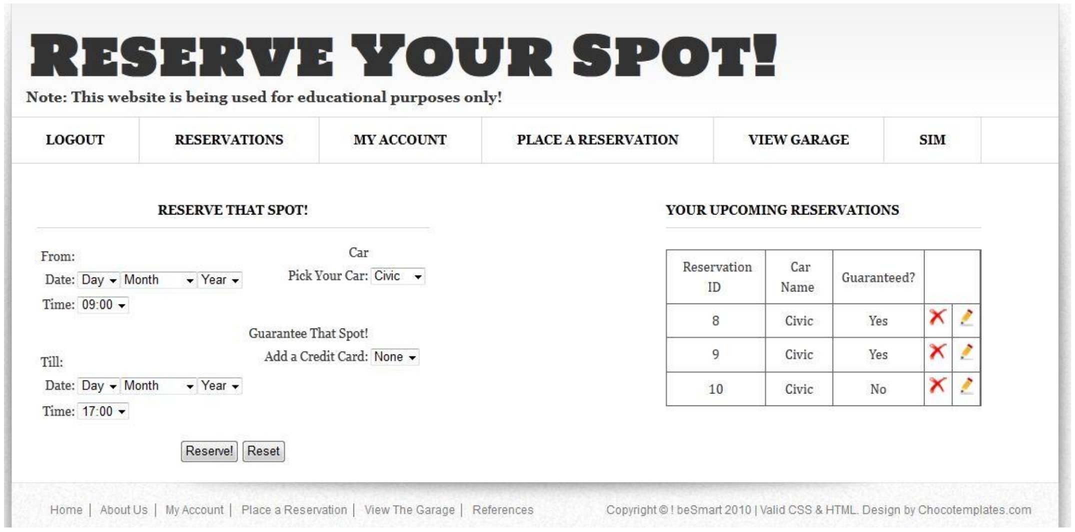
The next 20 upcoming reservations

Spot #	Start Time/Date	End Time/Date
201	10:00am on Wednesday, May 2, 2012	12:00pm on Wednesday, May 2, 2012
200	12:10pm on Wednesday, May 2, 2012	1:00pm on Wednesday, May 2, 2012
201	12:45pm on Wednesday, May 2, 2012	5:00pm on Wednesday, May 2, 2012

If the user is experiencing issues reserving a spot or is curious as to what type of reservations people have previously made, they can access this page. On this page the user can pick a time frame and view previous and/or future reservations. To ensure privacy, only the spot number, start point, and end point are displayed. If no time frame is specified, the page will display the next twenty upcoming reservations, if any.

Reservations

Figure 25



RESERVE YOUR SPOT!

Note: This website is being used for educational purposes only!

LOGOUT | RESERVATIONS | MY ACCOUNT | PLACE A RESERVATION | VIEW GARAGE | SIM

RESERVE THAT SPOT!

From:
Date: Day Month Year
Time: 09:00

Car
Pick Your Car: Civic

Guarantee That Spot!
Add a Credit Card: None

Till:
Date: Day Month Year
Time: 17:00

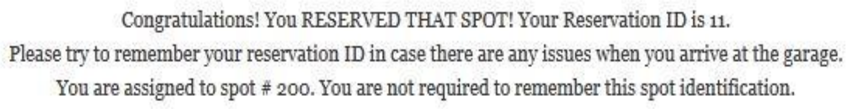
Reserve! Reset

YOUR UPCOMING RESERVATIONS

Reservation ID	Car Name	Guaranteed?		
8	Civic	Yes	✗	✎
9	Civic	Yes	✗	✎
10	Civic	No	✗	✎

Home | About Us | My Account | Place a Reservation | View The Garage | References   Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

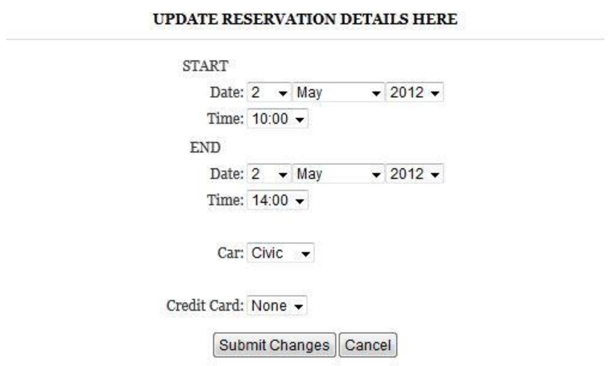
On this page the user can perform multiple tasks. On the left hand side of the page the user can place a reservation by entering a start point, end point, vehicle, and, optionally, a credit card. Upon successfully choosing a timeframe the system agrees with, the reservation confirmation number and spot will be displayed on the bottom. A valid reservation consists with one that passes the reservation algorithm (described in the previous section), contain at least one vehicle, have a start date that comes before the end date, yet after the current time, and be carried out by an 'active' user who has validated his email address.



Congratulations! You RESERVED THAT SPOT! Your Reservation ID is 11.
Please try to remember your reservation ID in case there are any issues when you arrive at the garage.
You are assigned to spot # 200. You are not required to remember this spot identification.

The right hand side allows the user to edit/delete existing reservations. If a user selects the 'red x' button, the reservation will be removed from the system. If the user selects the 'pencil' button, the user will be re-directed to a screen with all the values. All the changed entries go through the same procedure a regular reservation would go through and therefore need to be valid.

Figure 26



UPDATE RESERVATION DETAILS HERE

START
Date: 2 May 2012
Time: 10:00
END
Date: 2 May 2012
Time: 14:00

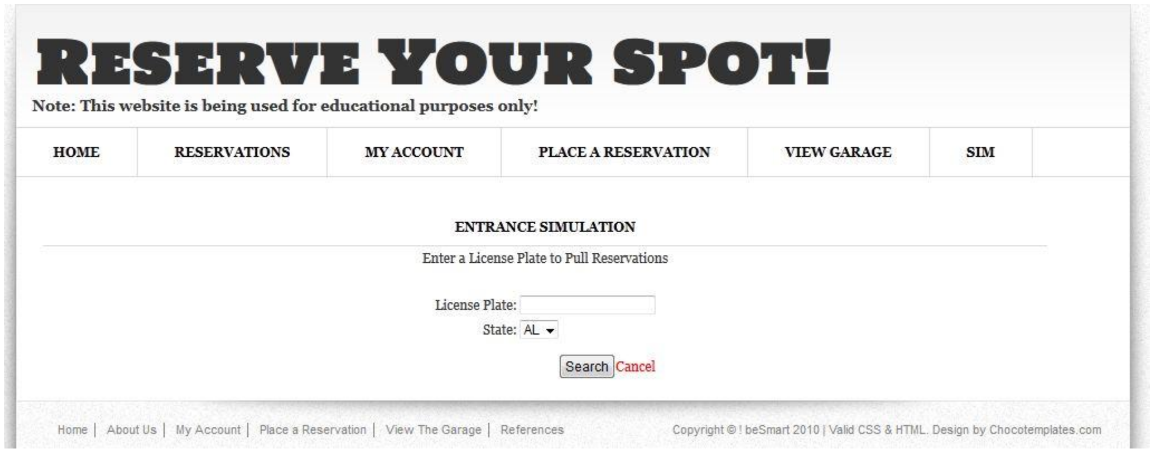
Car: Civic

Credit Card: None

Submit Changes Cancel

Entrance Camera

Figure 27



Note: This website is being used for educational purposes only!

ENTRANCE SIMULATION

Enter a License Plate to Pull Reservations

License Plate:

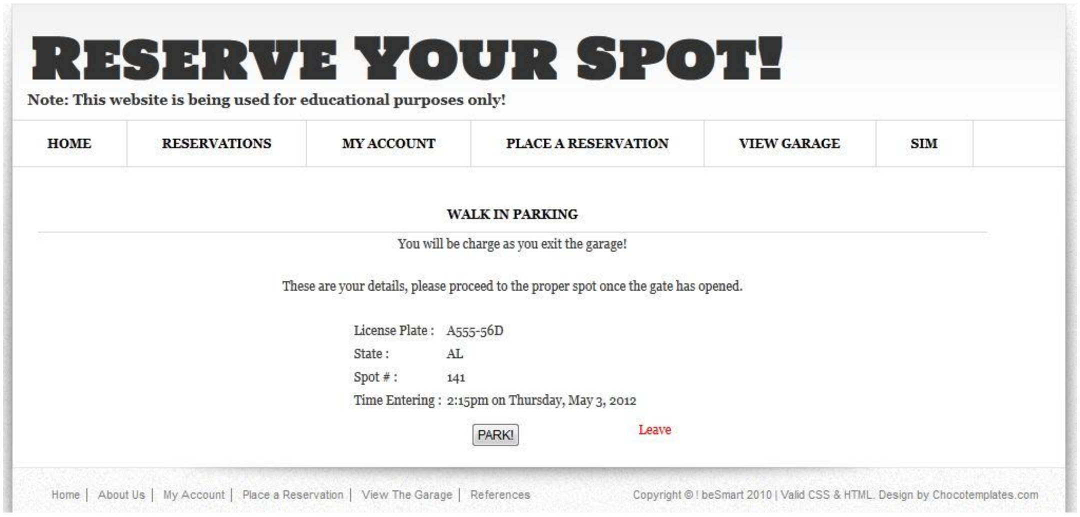
State: AL

Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

On this hidden page, the user can simulate a vehicle entering the garage. This is carried out by entering a license plate and the state associated with it to demonstrate how the system would go about reacting to input given by the license plate reader (camera). To provide a quick demo of the two most common tasks, walk-ins and reserved customers, the following screen shots are the step by step demo.

If a user enters a plate that is unknown to the system, the system will take it to be a walk in customer and proceed accordingly.

Figure 28



Note: This website is being used for educational purposes only!

WALK IN PARKING

You will be charge as you exit the garage!

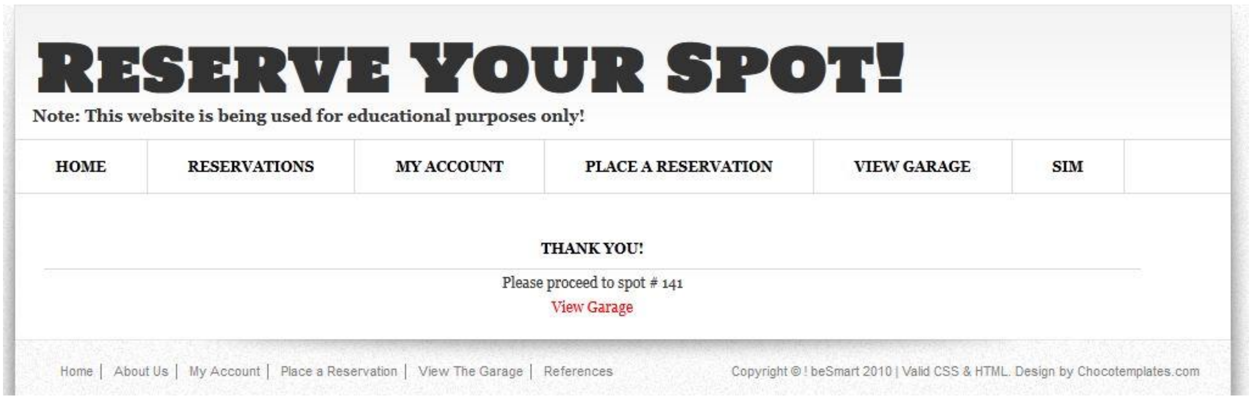
These are your details, please proceed to the proper spot once the gate has opened.

License Plate : A555-56D
State : AL
Spot # : 141
Time Entering : 2:15pm on Thursday, May 3, 2012

Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

Once the plate has been entered, a spot is assigned to that car along with its timestamp and asks if the user wishes to continue to their spot or leave. Leave will not have any affect, but selecting park will set that spot to vacant and initialize the counter.

Figure 29


RESERVE YOUR SPOT!
Note: This website is being used for educational purposes only!
HOME
RESERVATIONS
MY ACCOUNT
PLACE A RESERVATION
VIEW GARAGE
SIM

THANK YOU!

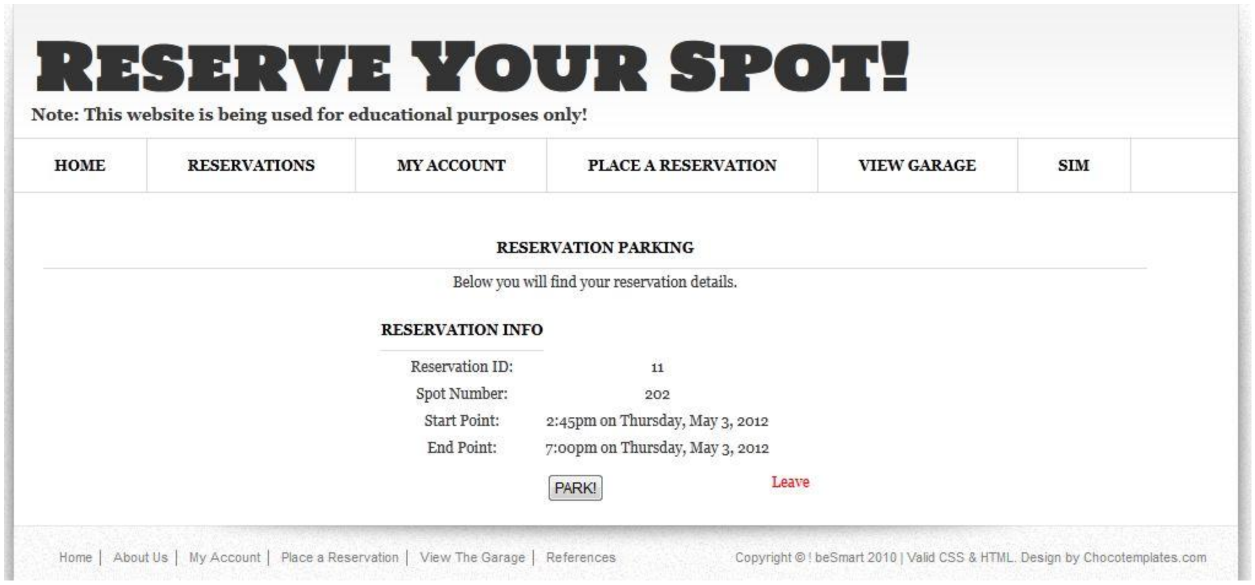
Please proceed to spot # 141

View Garage

Home | About Us | My Account | Place a Reservation | View The Garage | References
Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

When a user selects a license plate that is related to an upcoming reservation, the following screen will show. Note this is very similar to the walk in page as information is displayed and the options to park or leave are given. The only difference here is the fact that the displayed information pertains to what the user requested as well as a spot on the upper floor of the deck.

Figure 30


RESERVE YOUR SPOT!
Note: This website is being used for educational purposes only!
HOME
RESERVATIONS
MY ACCOUNT
PLACE A RESERVATION
VIEW GARAGE
SIM

RESERVATION PARKING

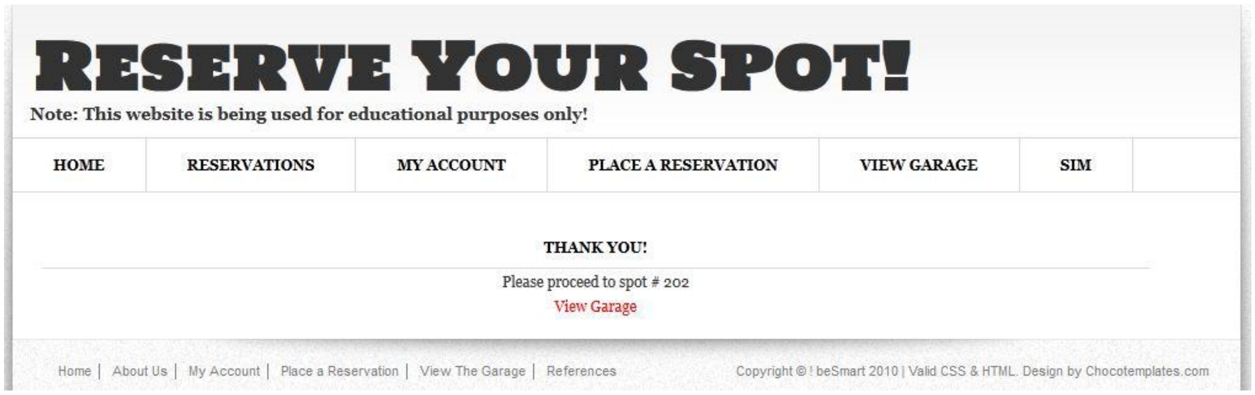
Below you will find your reservation details.

RESERVATION INFO

Reservation ID:	11
Spot Number:	202
Start Point:	2:45pm on Thursday, May 3, 2012
End Point:	7:00pm on Thursday, May 3, 2012

PARK!
Leave
Home | About Us | My Account | Place a Reservation | View The Garage | References
Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

Figure 31


RESERVE YOUR SPOT!

Note: This website is being used for educational purposes only!

HOME
RESERVATIONS
MY ACCOUNT
PLACE A RESERVATION
VIEW GARAGE
SIM

THANK YOU!

Please proceed to spot # 202

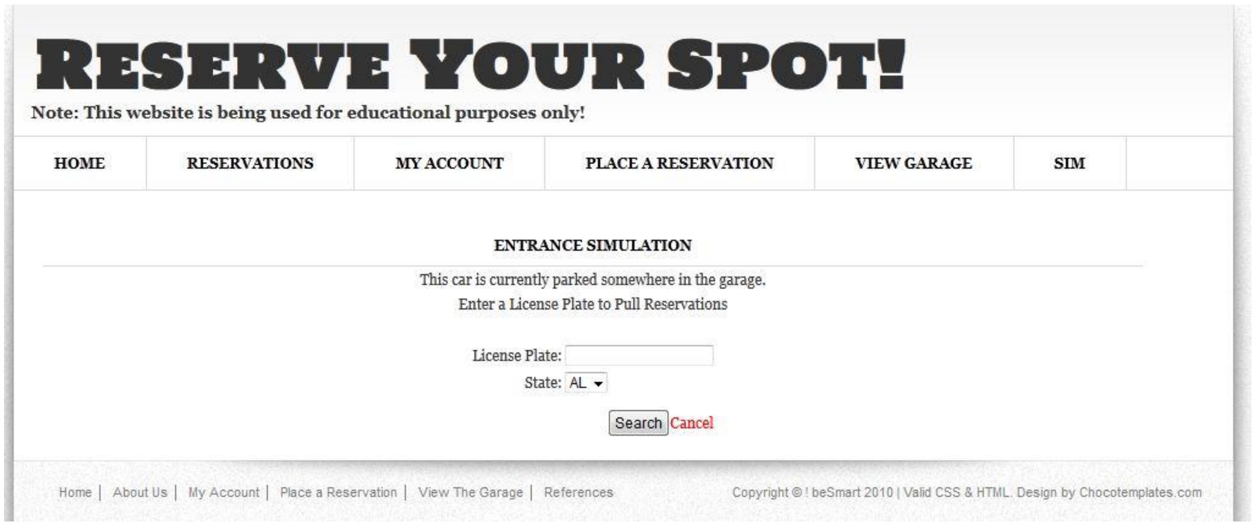
View Garage

Home | About Us | My Account | Place a Reservation | View The Garage | References

Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

Lastly, if a user tries to park a car that has already entered the garage, but has not yet left, the site will re-direct the user to the same page and inform them of the situation.

Figure 32


RESERVE YOUR SPOT!

Note: This website is being used for educational purposes only!

HOME
RESERVATIONS
MY ACCOUNT
PLACE A RESERVATION
VIEW GARAGE
SIM

ENTRANCE SIMULATION

This car is currently parked somewhere in the garage.

Enter a License Plate to Pull Reservations

License Plate:

State: AL

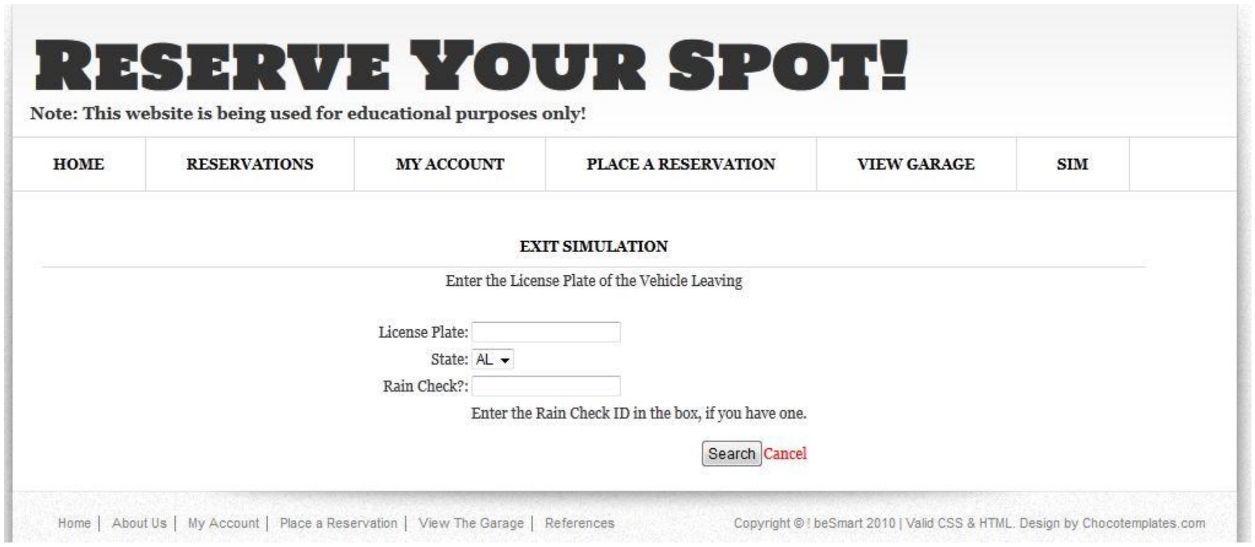
Search
Cancel
Home | About Us | My Account | Place a Reservation | View The Garage | References

Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

Exit Camera

Similar to the entrance camera, the exit camera will simulate how the system will react to vehicles leaving the garage via the exit ramp. The license plate information will be needed upon departure and a raincheck ID, if any.

Figure 33


RESERVE YOUR SPOT!
Note: This website is being used for educational purposes only!
HOME
RESERVATIONS
MY ACCOUNT
PLACE A RESERVATION
VIEW GARAGE
SIM

EXIT SIMULATION

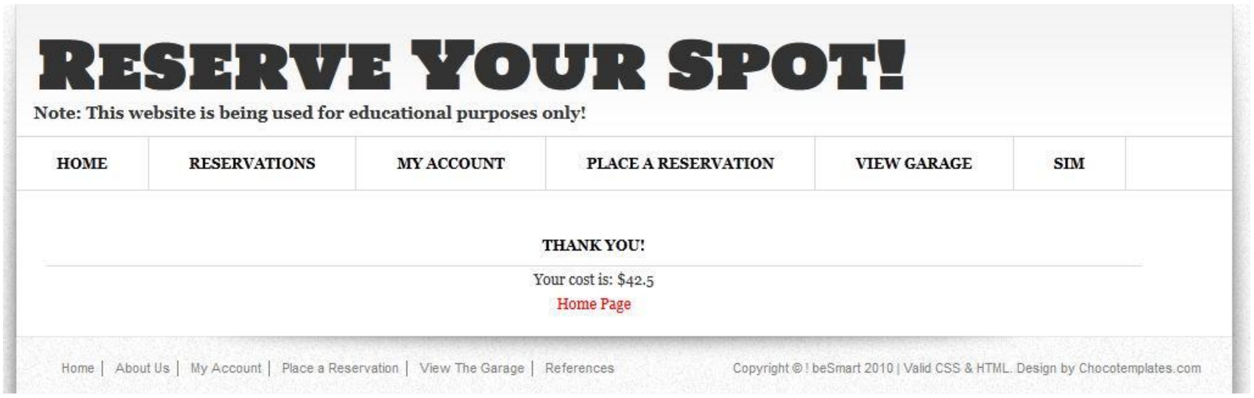
Enter the License Plate of the Vehicle Leaving

License Plate:
State: AL
Rain Check?:
Enter the Rain Check ID in the box, if you have one.

Search
Cancel
Home | About Us | My Account | Place a Reservation | View The Garage | References
Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

Once this criterion is entered, the garage will display the amount charged for the parking, change the spot status, and set the timestamp for the departure.

Figure 34


RESERVE YOUR SPOT!
Note: This website is being used for educational purposes only!
HOME
RESERVATIONS
MY ACCOUNT
PLACE A RESERVATION
VIEW GARAGE
SIM

THANK YOU!

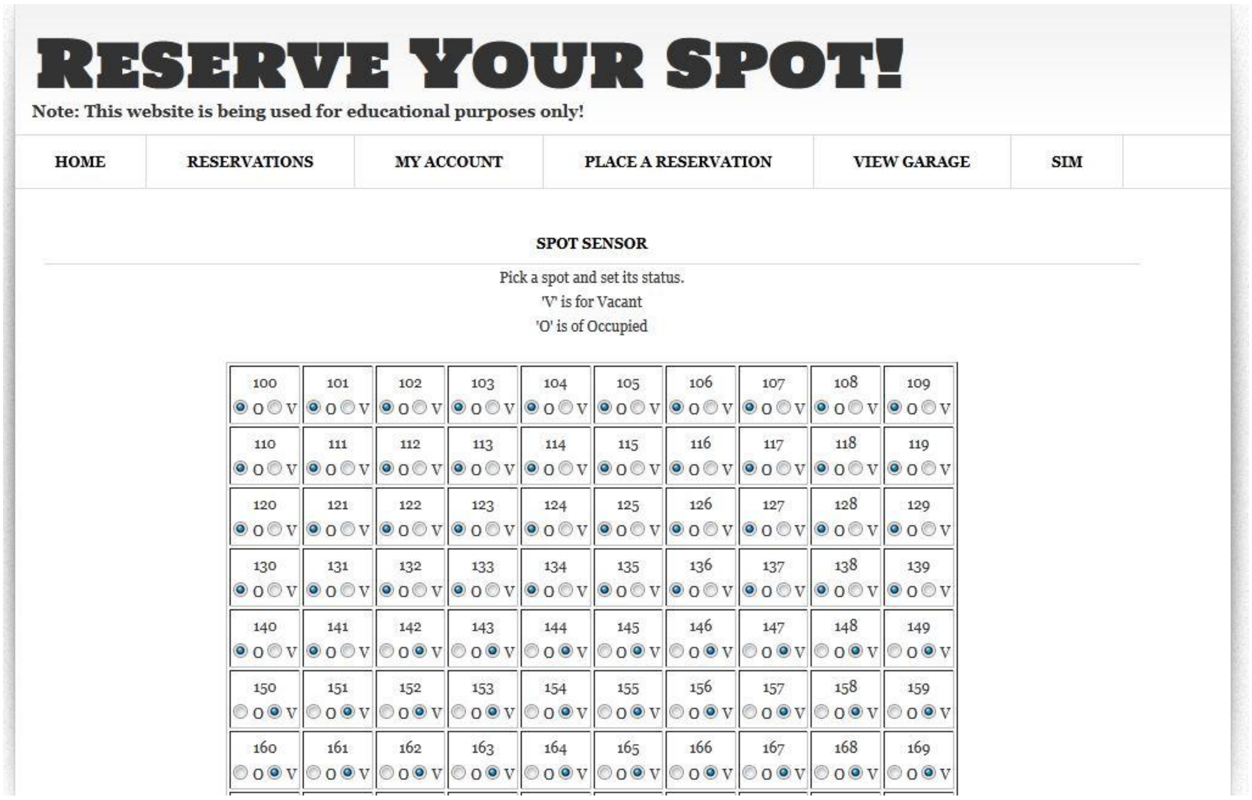
Your cost is: $42.5

Home Page
Home | About Us | My Account | Place a Reservation | View The Garage | References
Copyright © ! beSmart 2010 | Valid CSS & HTML. Design by Chocotemplates.com

Spot Sensor

This was just an additional page used to simulate what the spot controller would be held responsible for. Upon arriving to this page, each spot is filled with the appropriate radio button, based on whether or not the spot is vacant or occupied. This page was implanted as sort of a manual override in case the exit camera chose to fail in some sort of fashion. An employee could proceed to manually set the spot status as vehicles exit the garage, based on their reservation and/or vehicle information.

Figure 35



Note: This website is being used for educational purposes only!

SPOT SENSOR

Pick a spot and set its status.
'V' is for Vacant
'O' is of Occupied

100 [✓ O] V	101 [✓ O] V	102 [✓ O] V	103 [✓ O] V	104 [✓ O] V	105 [✓ O] V	106 [✓ O] V	107 [✓ O] V	108 [✓ O] V	109 [✓ O] V
110 [✓ O] V	111 [✓ O] V	112 [✓ O] V	113 [✓ O] V	114 [✓ O] V	115 [✓ O] V	116 [✓ O] V	117 [✓ O] V	118 [✓ O] V	119 [✓ O] V
120 [✓ O] V	121 [✓ O] V	122 [✓ O] V	123 [✓ O] V	124 [✓ O] V	125 [✓ O] V	126 [✓ O] V	127 [✓ O] V	128 [✓ O] V	129 [✓ O] V
130 [✓ O] V	131 [✓ O] V	132 [✓ O] V	133 [✓ O] V	134 [✓ O] V	135 [✓ O] V	136 [✓ O] V	137 [✓ O] V	138 [✓ O] V	139 [✓ O] V
140 [✓ O] V	141 [✓ O] V	142 O [✓ V]	143 O [✓ V]	144 O [✓ V]	145 O [✓ V]	146 O [✓ V]	147 O [✓ V]	148 O [✓ V]	149 O [✓ V]
150 O [✓ V]	151 O [✓ V]	152 O [✓ V]	153 O [✓ V]	154 O [✓ V]	155 O [✓ V]	156 O [✓ V]	157 O [✓ V]	158 O [✓ V]	159 O [✓ V]
160 O [✓ V]	161 O [✓ V]	162 O [✓ V]	163 O [✓ V]	164 O [✓ V]	165 O [✓ V]	166 O [✓ V]	167 O [✓ V]	168 O [✓ V]	169 O [✓ V]

Design of Tests

Integration Testing

In order to keep things simple, a specific type of integration testing, known as the big bang integration approach (Reference #10), will be used to carry out the testing for this system. The goal of the big bang method is to take all the individual unit test cases and combine them to form a complete or major part of the system. One type of big bang integration, known as Usage Model Testing, runs testing by carrying out user-like workloads in user-like integrated environments. The individual components are proved through this method by proving through the environment itself. The main focus will be to not avoid any smaller cases that may arise through the individual components. This integration approach was chosen due to its simplicity and efficiency. Its simplicity stems from its capability to cover a significant amount of individual unit cases.

Acceptance Test Cases

Acceptance test cases are events that the customer wishes to specify in order to ensure they handled properly. Since the customers using the website for any reservation requests and account changes, this is where most of the test cases will stem from. In order to carry out the integration testing mentioned above, acceptance test cases must be generated in order to understand what the user-like workloads will be. Below are the acceptance test cases designed for the website that account for the majority of the functions that can be carried out on the site. These test cases show the required input, pass/fail criteria, and the various scenarios that can occur.

Test Cases

Test-Case Identifier: TC-1, Logging In **Pass/Fail Criteria:** The test passes if the user enters an email password combination that is contained in the database. **Input Data:** Email Address, Password	
Test Procedure:	**Expected Result:**
Step 1. Submit blank fields	Website displays “Please fill out this field.” and does not process the request
Step 2. Submit an unknown email address, password combination	Website re-directs the user to the home page and displays “Incorrect email or password.”
Step 3. Submit a valid email address, password combination	Website re-directs the user to their respective ‘My Account’ page
Test-Case Identifier: TC-2, Vehicle Update **Pass/Fail Criteria:** The test passes if the user puts in a request that does not conflict with another existing database entry and stays within the account limits. **Input Data:** Car Name, License Plate, US State where vehicle is registered	
Test Procedure:	**Expected Result:**
Step 1. Submit a request without having all the necessary fields filled	Website displays “Please fill out this field.” and does not process the request
Step 2. Submit a change that conflicts with another database entry	Website re-directs the user to their account page and displays “Vehicle already registered.”
Step 3. Add a third vehicle	Website re-directs the user to their account page and displays “Too many registered vehicle.”
Step 4. Submit a valid vehicle change that does not conflict with another database entry and stays within the account limits	Website re-directs the user to their account page and displays the change(s).

Test-Case Identifier:	TC-3, Placing/Editing a reservation
Pass/Fail Criteria:	The test passes if the user places a request with the start date coming before the end date and if it passes the scheduling algorithm
Input Data:	Start Point, End Point, Car, Credit Card (Optional)

Test Procedure:	**Expected Result:**
Step 1. Submit a request without filling all the required fields	Website displays “Please fill out this field.” and does not process the request
Step 2. Submit a request where the start date comes AFTER the end date	Website re-directs the user to the reservation page and displays “Please request a reservation whose end date is BEFORE the start date.”
Step 3. Submit a request that does not pass the scheduling algorithm (no space available for that time)	Website re-directs the user to the reservation page and displays “There are no spots available for that requested time frame, please try another request or check to see if any walk-in spots are available.”
Step 4. Submit a valid request that passes the scheduling algorithm	Website re-directs the user to the reservations page and displays the reservation ID along with the spot they were assigned

Test-Case Identifier:	TC-4, Registering
Use Case Tested:	
Pass/Fail Criteria:	The test passes if the user attempts to register an account with a unique email address
Input Data:	Email Address, Name, Password, Date of Birth

Test Procedure:	**Expected Result:**
Step 1. Submit a request without filling all the required fields	Website displays “Please fill out this field.” and does not process the request
Step 2. Submit the request with an email address that already exists in the databas	Website re-directs the user to the registration page and displays “The email is already in use.”
Step 3. Submit a request where the two password fields do not match	Website re-directs the user to the registration page and displays “The passwords do not match.”
Step 4. Submit a unique email address, password fields that match, and the other requested information	The system creates a session and re-directs the user to the account page

Unit Testing

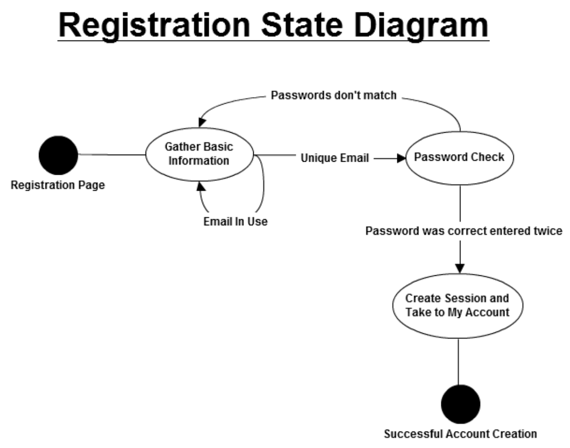
The test cases for this system will be based of the generic class diagram, non-PHP specific, mentioned earlier. The five main classes in this system are the registration, reservation, management, authorization, and account classes. Unit testing will involve going through each of these classes and their modules, asserting them with different parameters.

The breakdown of the testing that will be performed on the individual classes is seen below. Each class has a brief description and some classes even contain a state diagram to demonstrate how requests are validated.

Registration

- Registering a new user with information that matches data of another existing user in the system. All users are uniquely identified by their email and this will be the essential factor in testing this.

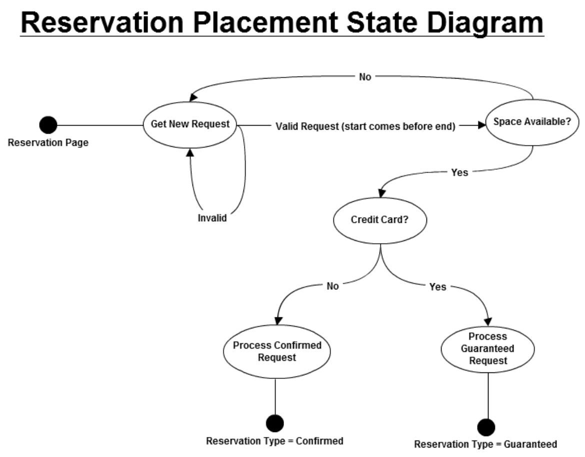
Figure 36


Registration State Diagram
Passwords don't match
Gather Basic Information
Unique Email
Password Check
Registration Page
Email In Use
Password was correct entered twice
Create Session and Take to My Account
Successful Account Creation

Reservation

- The registration process can lead to a variety of situations and will therefore include testing try to expose any leaks. This will involve inserting any reservations that conflict with existing reservations, including the parking spot ID, start points, and end points. In addition to this, the design will also test that confirmed and guaranteed reservations do not conflict as they both pertain to the same spots, but are looked at with two different statuses. Lastly, any revisions made to upcoming reservations will also be tested to find conflicts as they arise.

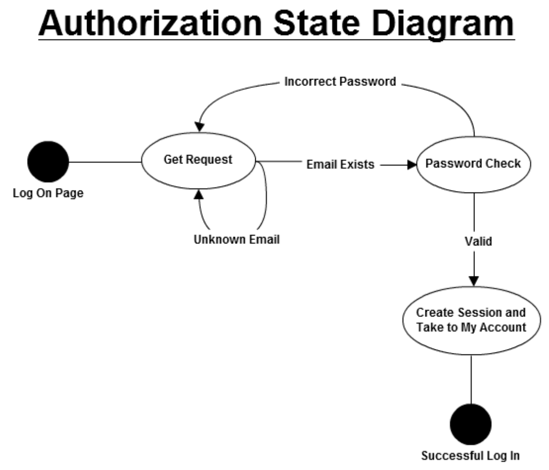
Figure 37


Reservation Placement State Diagram
Reservation Page
Get New Request
Valid Request (start comes before end)
Space Available?
No
Invalid
Yes
Credit Card?
No
Yes
Process Confirmed Request
Process Guaranteed Request
Reservation Type = Confirmed
Reservation Type = Guaranteed

Authorization

- Authorizing is an a crucial key that grants certain users, certain privileges and testing this class is key to making sure order is maintained in the system. As users attempt to identify themselves by either the Vacancy Display or through the website, their log on credentials will be passed through this class. Testing will involve asserting the modules with log on credentials that both match up and differentiate themselves from the ones in the database.

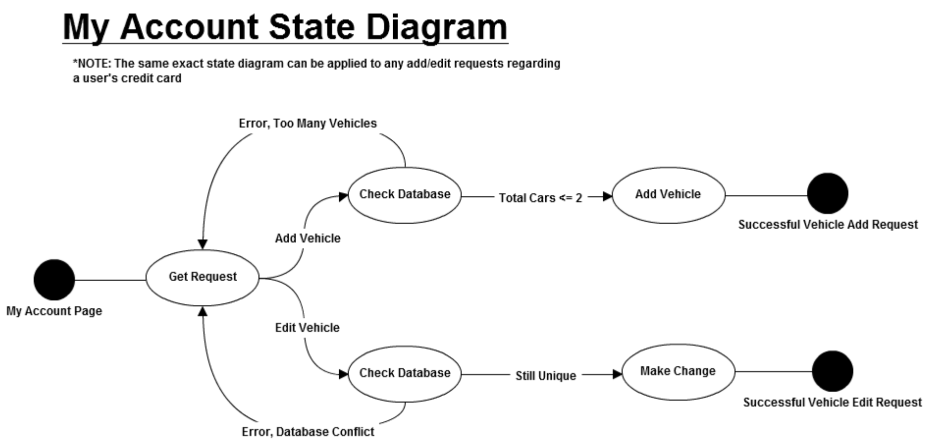
Figure 38



Account

- The account class is responsible for maintaining the information regarding the user and their account. Testing in this class will involve asserting a validation process that will take any requested changes or additions and look them over before they are carried out.

Figure 39



Management

- Just like the account class, management is the same principle except it involves pricing and regulation for the entire system. Therefore it will assert the modules in this class with parameters to validate all changes.

The goal for the design of these tests is to cover the interactions a user has with the system covered from end to end. As the user can interact with the system in a variety of ways through a few channels, it is important to have coverage to account for all the scenarios, no matter what the situation.

History of Work, Current Status, and Future Work

History of Work

Although there were multiple steps involved in finalizing this project, the milestones are listed in the table below along with their completion dates. These were the big picture objectives that were pre-assigned a date for submission and revised as needed, based on the feedback given.

Milestone	Completion Date
Report One	February 17, 2012
Report Two	March 9, 2012
Demo One	March 30, 2012
Web Design	April 20, 2012
AI Implementation	April 23, 2012
Testing	April 25, 2012
Demo Two	May 2, 2012
Report Three	May 3, 2012

The milestones regarding all three reports and the two demoes were all met with ample time to review the work and revised for future submissions. The other milestones were not exactly lined up with the Plan of Work sections mentioned in previous reports as there were newer items being added to the web page that were either not originally proposed and/or updated based on the feedback given. Nonetheless, all the work for the project was done so there was time for a group review where each member would proofread the sections of other team members to assure we were all on the same page and that everything lined up correctly.

Current Status

As of today, a user can access the web page and create their personal account. As a registered user, they are given the privileges of registering a credit card(s) and vehicle(s) to be able to place a reservation for an upcoming date. If at any time a registered user would like to make changes to these features, they can do so on their respective editing pages as long as they uphold the account and system limitations. All users can go to the web page and take a look at the current garage state as well as the reservations in the system. These functions ensure the privacy of other users by not displaying any sensitive information, but rather just the start and end points specific to a reservation. Once all the required information is in the database, a customer can go to the garage and park in a spot, based on their reservation type, if space is available.

In addition to all this, we are in the beginning phases of developing an android application to allow customers to interact with the system via their phone. Although very basic at this point, the app does create a log in screen and will take the user to the website. Not fully functional as of yet, but further work will allow being able to log into existing accounts and place reservations so users can do it on the fly. With all these available features, the current implementation allows customers to work with the system in a convenient fashion. This is accomplished by getting the minimum amount of information from the user that will both get their request through and protect their privacy.

Lastly, the aspect of artificial intelligence can be used to test this system. This is accomplished by setting up a script to run on a certain time schedule. The scheduler used in this project is called cron (Reference #12). Cron is a time-based scheduler in Unix operating system that enables used to schedule commands to run repeatedly at certain dates/times. This can be used to show heavy traffic on the system through having reservations placed in large quantities at once. As this system is being hosted on GoDaddy, the cron jobs were set up on the server, which runs Linux. It has to be enabled by logging into that server.

Future Work

Future work for this project will definitely be to try and expand the phone application to allow customers to use it with the same capabilities as the web site. This process would involve completing the application for android phones and then expanding over to the iOS platform. This is the first step in future work because as of late 2011, over 75% of smart phone users have either android or iOS phones (Reference #11). This is a big share of the wireless market as there are a significant number of smart phone users in the United States and that numbers continues to grow.

As the site is designed to be able to handle multiple garages, even though the current implementation focuses on one, future work may involve establishing contracts with large companies to meet the parking needs of their employees. There are many large companies, such as Johnson and Johnson, who hold locations in urban areas where parking isn't easily available. These companies then turn to acquiring their own garage where their employees can safely park without having to go through the hassle of the crowded city environment. These contracts would be focused around one, very large customer, with a large number of vehicles. The current system places limitations on these types of transactions and future design may allow large scale businesses to establish custom contracts.

Conclusion

As with most software engineering endeavors, this project is challenging. There are many factors that went into planning the approach to the parking garage automation. Many discussions debated over which class structure might perform better or make more sense for future system modification. The compromise is the system that is outlined in this report.

Given a short time period, a lot is accomplished. The website is fully functional and able to run on mobile browsers. The database system works and can recognize a rain check situation when the garage is full. The system is capable to store credit cards while employing a form of security so that the credit card numbers cannot be easily stolen. The system is also able to intelligently assign parking spots to the users.

However, every single software project can be improved. There is room for improvement with the current system. With more time an enhanced the GUI would be implemented. This update includes minimizing keystrokes to accomplish a given task. Also, sharper graphics and animations would be employed to give the user a feeling that he/she is using a premium service. This improves on the customer satisfaction and experience.

Furthermore, another major improvement to accomplish is to fully finish the app. After creating the app's design and its features there is an unexpected problem connecting the app to the database. The app is built on the android platform so the coding is JAVA based. The servers use mostly http and php, but unfortunately the JAVA code is not translated to the proper programming language in time.

Overall, the technical challenges faced are solved with an efficient software program. With proper project management, the system requirements and use cases used developed a functional parking garage system. Despite some shortcomings, mostly due to time constraint, the project helps understand the software engineering approach and its significance. Through the software engineering application, the parking garage automation is implemented successfully.

Breakdown of Responsibilities

- Bartosz Agas
    - Worked on Report 1, Web Site Design, Database Structuring, Report 2, Coding and Implementation, Simulation, Report 3 (Design of Tests)
- Christopher Tran
    - Worked on Report 1, Web Site Design, Database Structuring, Report 2, Coding and Implementation, Simulation, Report 3 (Interaction Diagrams)
- Marvin Germar
    - Worked on Report 1, Web Site Design, Database Structuring, Report 2, Coding and Implementation, Simulation, Report 3 (Class Diagram Specficantion)
- Michael Van Genderen
    - Worked on Report 1, Web Site Design, Report 2, Coding and Implementation, Simulation, Report 3 (System Architecture and System Design), Testing
- Justin Levatino
    - Worked on Report 1, Web Site Design, Report 2, Coding and Implementation, Simulation, Report 3 (Effort Estimation), Testing
- Tarun Katikaneni
    - Worked on Report 1, Web Site Design, Report 2, Coding and Implementation, Simulation, Report 3 (Functional Requirements), Testing

References

1. Marsic, Ivan. *Software Engineering*. 2012.
2. Bruegge, Bernd, and Allen H. Dutoit. *Object-oriented Software Engineering: Using UML, Patterns, and Java*. Boston: Prentice Hall, 2010
3. Bardi, James A. *Hotel Front Office Management*. Hoboken, NJ: John Wiley & Sons Inc., 2007.
4. Duckett, Jon. *Beginning HTML, XHTML, CSS, and JavaScript*. Hoboken, NJ: Wiley, 2010.
5. Miles, Russ, and Kim Hamilton. *Learning UML 2.0*. Sebastopol, CA: O'Reilly, 2006.
6. Nixon, Robin. *Learning PHP, MySQL, and JavaScript*. Beijing: O'Reilly, 2009.
7. Ramakrishnan, Raghu, and Johannes Gehrke. *Database Management Systems*. Boston: McGraw-Hill, 2003.
8. Website Template - *ChocoTemplates - The Sweetest CSS Templates WorldWide*. Web. 1 Feb. 2012. <http://chocotemplates.com/>.
9. "Event-Driven Architecture," Wikipedia, <http://en.wikipedia.org/wiki/Event-driven_architecture>
10. "Integration Testing." *Software Testing Fundamentals*. Web. <http://softwaretestingfundamentals.com/integration-testing/>.
11. Mayer, Chris. "Android Rules the Yard, but IOS Developers Still Rake in the Cash." *Jaxenter.com*. 23 Nov. 2011. Web. <http://jaxenter.com/android-rules-the-yard-but-ios-developers-still-rake-in-the-cash-39531.html>.
12. "Newbie: Intro to Cron." *UNIXGEEKS.ORG*. Web. <http://www.unixgeeks.org/security/newbie/unix/cron-1.html>.
13. "Strategy Design Pattern." *Design Patterns and Refactoring*. Web. <http://sourcemaking.com/design_patterns/strategy>.