Report # 3

Course Title: Software Engineering Spring 2019

Group Number: 19

Project Title: Rutgers Parking System

URL of Project Website:

https://github.com/gabriel-shen1/Parking-Project

Submission Date: April 14, 2019

Team Members:

Suva Shahria, Krithika Uthaman, Andrew Schneeloch, Josh LoGiudice, Gabriel Shen, Anthony Lau, Jahidul Islam, Yu Liu & Max Davatelis

Individual Contributions Breakdown

Report Section	Points	Suva Shahria	Jahidul Islam	Krithika Uthaman	Josh LoGiudice	Gabriel Shen	Yu Liu	Andrew Schneeloch	Anthony Lau	Max Davatelis
• Summary of Changes	5		100							
1. Customer Statement of Requirements	6	30	30	30			10			
2. Glossary of Terms	4		100							
3. System Requirements	6	25	25	25				25		
4. Functional Requirements Specification	30	15	20	15	10		15	10	15	
5. Effort Estimation	4									
6. Domain Analysis	25	12.5	12.5	12.5	12.5		12.5	12.5	12.5	12.5
7. Interaction Diagrams	30			50			50			
plus, use of *Design Patterns*	10	100								
8. Class Diagram and Interface Specification	10		100							
plus, OCL Contract Specifications	10		50			50				
9. System Architecture and System Design	15	14.29		14.29	14.29		14.29	14.29	14.29	14.29
10. Algorithms and Data Structures	4								100	
11. User Interface Design and Implementation	11							33.3	33.3	33.3
Design of Tests	12		100							

14. History of Work, Current Status, and Future Work	5			50		50				
15. References	-5									
PROJECT MANAGEMENT	13	70		30						
TOTAL:	**200**	32.2	53.425	34.5	8.27	2.5	25.4	13.43	17.43	8.93

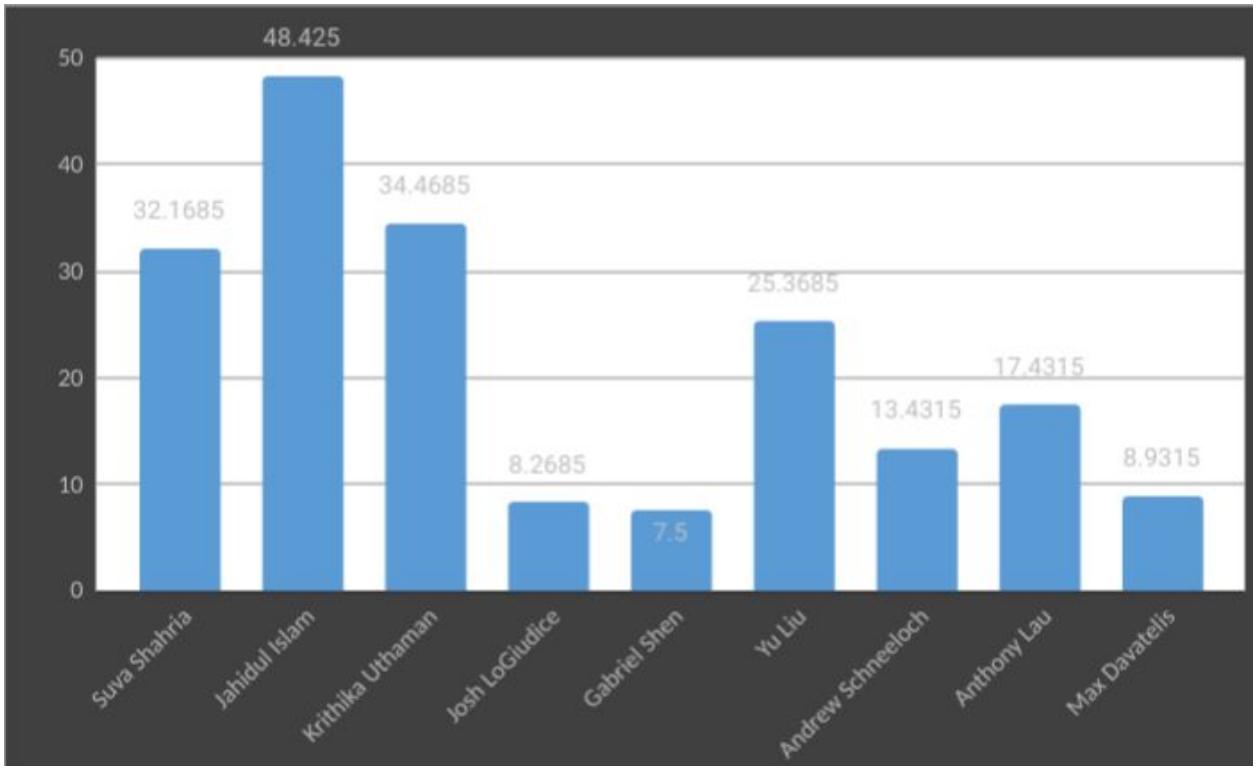


Table of Contents

Summary of Changes

A major change was the inclusion of the idea of statuses throughout the system which can be for example a student, or a guest. This status is chosen during registration and allows the system to identify restrictions for that user to eliminate user hassle. Another change is the description that the user will sign in and out of the parking lot by using their Rutgers ID on a scanner at the parking lot. Other then changing the language of the documentation into including this notion other notable changes include:

- REQ-4 changed into incorporating guest pass status structure
- REQ-11,12,13 removed with a new REQ--12 added: This was done as the other requirements did not meet the model of the current system.
- REQ-14 changed to consider statuses
- REQ-20 changed interactable map description to show general areas of lots
- Garage owners removed as a stakeholder as focus is on Rutgers parking
- Garage owners removed as an actor for aforementioned reason
- UC-14 removed as implementation of the user's schedule cannot be done at this time while considering parking by available space. Thus it was removed from all descriptions, traceability matrices and diagrams
- UserSchedule removed from attribute definitions because it was UC-14
- UserStatus added into attributes of Database to incorporate this new foundation of our design
- Use case weight changed from 115 to 110
- Actor complexity weight changed from 18 to 17
- Total use case points went from 133 to 127
- Class diagram updated to include new system of statuses and info regarding the class diagram that follows updated to match including traceability matrix
- Removed notification section form class diagram
- State Diagram added to display simplicity of design which will benefit user interactions
- Test cases updated for aforementioned changes

Section 1: Customer Statement of Requirements

With a continuously rising population density, large issues arise with simple accommodations such as parking. Diminishing open parking spaces often are left unresolved even with the addition of new parking garages with their currently inefficient allocation of reserved spots. Certain ventures have been made to automate the process through a website or app, but still leave room for improvement. Many parking spaces still remain with empty unused portion for varying amounts of time with the potential of supplying more consumers with available space. Creating new facilities without improving the currently existing only increases congestion in new areas. In order to fully take advantage of current parking lots and garages a new system is needed. With each parking spot being efficiently utilized, congestion will become highly limited.

In addition to maximizing parking usage, another issue affecting parking decision lies with confusing regulations. With a plethora of parking systems, some with their own specific rules, a customer's confusion leads to incorrect parking which then in turn affects other customers creating greater congestion. Pairing such confusion with an event which requires large parking accommodations leads to visible evidence regarding the faults of the current system. Overall, the new system must recognize available space immediately as they become available in order to ensure proper utilization.

A specific example of these issues with parking are found within the Rutgers University community. Rutgers students, faculty, and staff depend on the parking system to be reliable and efficient especially with a huge portion of the student body being commuters. Instead they are dealing with the confusing system currently in place which is unclear on which lots an individual is allowed to park in and when, which then makes it difficult for students to get to class on time. Currently a common occurrence is many vehicles frequently and inefficiently roaming around the lots looking for an ideal parking spot and sometimes following students back to their spots. Individuals currently show up to parking lots clueless to the actual availability and resort to the problematic pattern of searching for a spot. Many times the students receive parking tickets

despite the fact of parking in the correct lot or due to small technicalities in the parking guidelines.

The parking issues are then amplified when considering special events, such as football games. These events bring massive amounts of people and vehicles who aren't usually here. The inability to predict parking accommodations for such occurrences hinder the situation further leading to prolonged searches and confusion.

The old fashioned method of just creating more space does not tackle the core of the issue. Furthermore, in prioritizing creation of more parking spots it may reduce overall transportation as it neglects space to accommodate other modes of travel such as biking or even possible locations for public transport such as bus stations. By reducing options for other modes of transport, the congestion dilemma remains prevalent. An increase in traffic surrounding the new garage is also to be expected. A consumer focused improvement on existing garages is needed rather than an increase of capacity.

Proposed Solution

In beginning the approach to solve the given issue, first, data is needed from the customer. Specifically, their arrival time, time parked, as well as exit time. This will provide the app's database with required information needed to recommend parking areas with availability. The app utilization of login and identification info such as ID will be linked to their parking tendencies which will be protected via encryption. That data of when the parking spots are vacant or occupied will allow the app to properly dictate to parking seekers exactly where parking is available.

The solution to the parking issues at Rutgers is an app which all individuals of the Rutgers community can download. An app would be more suitable than a website since most of the users demographic are expected to own smartphones and it would be easier to use on the go.

Users will be able to create their own accounts, with a username and password, where they will enter their parking permissions, for example a user may hold a "Commuter Zone A" parking permit. The app will let them know which lots they can park in according to their

parking permissions, the occupancy of that lot and the day/time. There will also be an interactive map which will let the user know which spots are vacant and which are occupied.

As the user uses the app more and more often, the app will be able to more efficiently recommend certain spots. The system will also keep track of a user's favorite spots and log how long they usually stay there. The user can input where their classes take place so that the system can recommend spots accordingly.

Although the solution would work whether or not everyone who parks at Rutgers uses the app, the efficiency of the system would increase as more of the community uses it. The system will be able to track the usage of certain lots and track how busy they are over the course of the day as well as throughout the week. By comparing the actual occupancy of the lot to the average occupancy the system will be able to offer better parking suggestions.

For special cases, such as football games, the system will be notified that certain lots, like the ones around the stadium on Busch, are currently off limits and the system will respond accordingly. In order to accommodate the large influx of new parkers, certain areas will be reserved for visitors only as well in order to minimize negative effects for students.

Assumptions

In order for this solution to work, some assumptions need to be made. This includes that all users have smartphones and have access to the app. The second assumption is that there is a spot-occupancy recognition system in place that works correctly 100% of time. This system would keep track of whether spots are currently vacant or occupied. This system would keep track of whether spots are currently vacant or occupied. The choice of system will be a scanner capable of scanning a student or or faculty member's Rutgers ID.

Section 2: Glossary of Terms

- Account: Customer creates an account with a username and password along with their parking permissions which holds all the users information
- App: Mobile application that customers can log into and will help recommend them parking spots
- Database: Store user's account information and parking guidelines that feeds data to the app
- Encryption: The act of encrypting user information so that personal information can't get stolen
- Interactive Map: Maps of every parking lot/garage which shades occupied spots in red and vacant spots in green
- Occupied Spot: A spot which has a vehicle in it
- Parking Permissions: Where a user is permitted to park
- Registration: Creating an account on the app
- User: A person who uses the app and intends to find an ideal parking spot
- Vacant Spot: A spot which doesn't have a vehicle in it

Section 3: System Requirements

Enumerated Functional Requirements

Identifier	Priority	Requirement
REQ-1	5	The system will let users create, edit and delete their own accounts through the app. Account information includes username, password and the user's parking permissions
REQ-2	5	The system will have the current status (vacant, occupied) of every parking lots based on the number of people scanned in
REQ-3	4	The system will have an occupancy chart for every parking lot
REQ-4	5	The system will operate with the notion of statuses of passes for authorization i.e. guest pass, student pass, etc.
REQ-5	4	The system will let the user know which lots they are permitted to park in
REQ-6	2	The system will issue a warning if the lot they have chosen to park in has certain restrictions
REQ-7	3	The system will be aware of any special cases (football games, a lot is being closed off, etc.)
REQ-8	2	The system will keep know the occupancy patterns of lots throughout the day and throughout the week by monitoring scanned in IDs
REQ-9	3	The system will track various stats of lot usage and efficiency by utilizing data such as occupancy through ID scan in

Enumerated Nonfunctional Requirements

Identifier	Priority	Requirement
REQ-10	3	The app will track users and when they park in the lot and compare that to how many spots are in the lot this will allow the app to give a more accurate occupancy chart the more the app is being used
REQ-12	2	System will allow users to view history of parked locations
REQ-13	1	The stats collected by the system can be passed on to the Department of Transportation at Rutgers if they are interested in using it to improve their own system

On-Screen Appearance Requirements

Identifier	Priority	Requirement
REQ-14	5	The app should have an initial page where you sign in, with a link that says if you don't not have a account make one with a link to register for an account
REQ-15	5	The registration link will take you to a page where you will fill out your information including status i.e. student, guest, etc.
REQ-16	4	Once signed in the app will display a menu of campuses that you can park on with the ability to choose one
REQ-17	4	Once a campus is chosen another menu will be displayed with possible parking lot locations on that campus
REQ-18	3	The user will then be allowed to select on a lot and check the status of the lot's occupancy
REQ-19	2	The app will have a profile page where users can edit or delete their accounts
REQ-20	3	The app will have an interactive map for viewing lot areas
REQ-21	3	The app will have a chart showing how busy lots are expected to be throughout the day as well as a comparison of how busy the lot actually is currently

Section 4: Functional Requirement Specification

Stakeholders

- Parkers
    - Students/Professors: Need reliable information on parking availability to attend or teach classes on time
    - Other Users: Seek parking for any variety of reasons and would like to resolve parking issues efficiently
- Department of Transportation: Needs to monitor transportation services and information regarding which parking is used to provide an outlook of how many passengers will be using public transportation and from where. This will allow efficient public transport
- Event Planners: Organizing large events require ample parking in order to manage both attendance and road congestion. By supplying customers with more options, the negative effects can be reduced
- Drivers: Anyone driving near parking facilities can be met with unwanted traffic resulting from people searching for parking as they enter and leave garages. By providing the parker with exact places to park, road congestion will be reduced
- Services
    - Federal/State: Services, like the mail as well as school buses, need safe places to stop and are impacted by traffic and road congestion with people seeking to park or parking incorrectly on the streets lead to negative effects
    - Company: Companies, like Amazon, depend on delivering efficiently and traffic congestion in city residential areas due to parking searches hinder potential profit.
- Creators: The app development team

Actors and Goals

Actors	Roles	Goals
Parkers	Initiating	Find parking efficiently and know the exact destination to avoid hassle
Department of Transportation	Offstage Actor	Utilize info on parking use to efficiently place public transportation
Event Planners	Offstage Actor	Limit congestion during planned events by providing customers with parking solutions
Drivers	Supporting Actor	Alleviate traffic headaches near parking locations by minimizing the people wandering about
Services	Offstage Actor	Less traffic congestion will allow other services to function efficiently
Creators	Supporting Actor	Maintain database and update app
App	Initiating Actor	Used to reserve parking spots as well as indicate which are available. Exiting after parking spot usage is done also provides notification to the system that this spot is ready to be assigned to someone else.

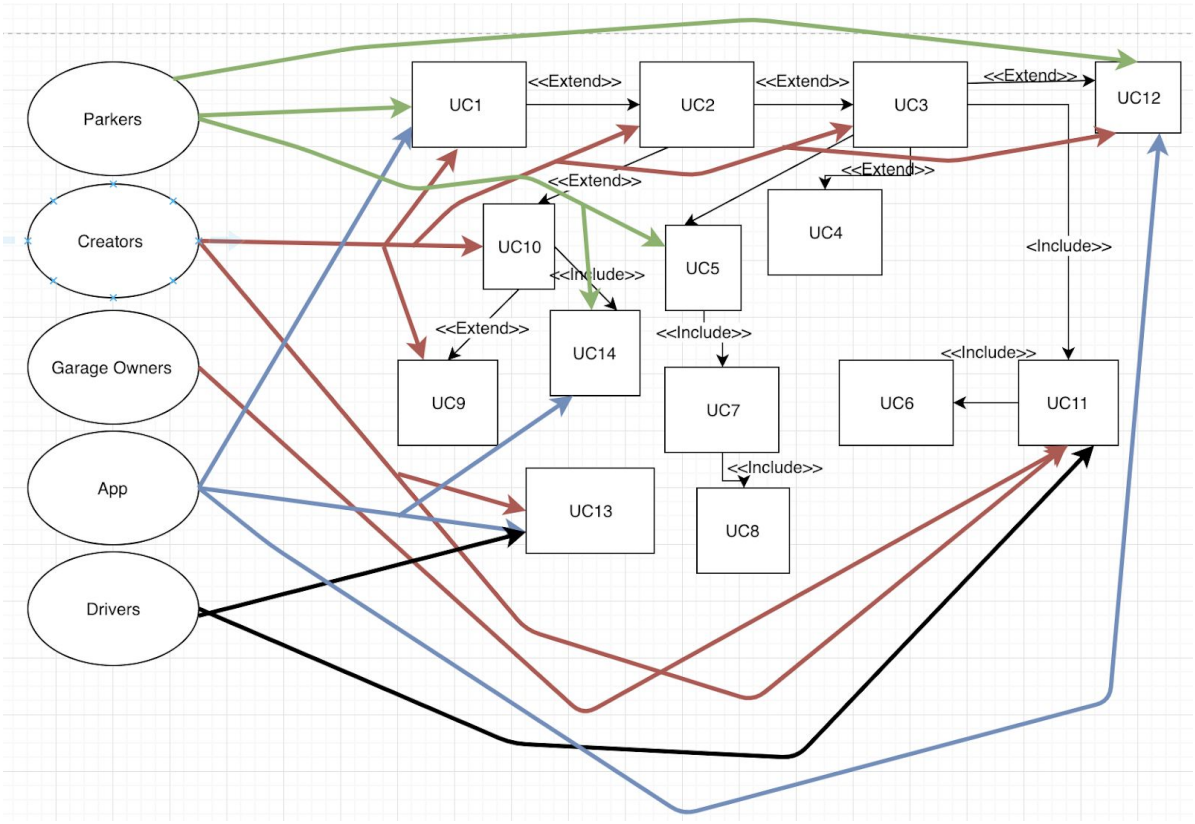
Use Cases

Casual Description

Use Case Identifier	Use Case Name	Description
UC-1	Register	The user will be able to register for a new account within the app
UC-2	Login	The user will be able to login to an existing account from the app
UC-3	Choose Campus	A list of campuses will be listed. The user can choose which campus they want to park on
UC-4	View Spots	A display of the parking lot will be displayed that show both vacant and occupied parking spots
UC-5	Pick Spot	The user can choose a parking spot that is available from the app
UC-6	Events	The system will notify the which lots are closed due to any events
UC-7	Arrival Time	The system will record the time the user arrives in their spot
UC-8	Exit Time	The system will record the time the user leaves
UC-9	Delete Account	The user can delete their account if they want
UC-10	Edit Account	The user can edit any account information such as

		changing their password
UC-11	Lot Information	The system will display all available parking lots for the user as well as status
UC-12	Interactive Map	The system will display an interactive map of the lots
UC-13	Patterns	The system will keep track of expected occupancy patterns of the lots over time and days of the week

Use Case Diagram



Traceability Matrix

Req.	**Priority**	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12	UC-13
REQ-1	5	X	X							X	X			
REQ-2	5				X							X	X	
REQ-3	4				X							X		X
REQ-4	2			X	X	X								
REQ-5	4		X											
REQ-6	2						X					X		
REQ-7	3						X							
REQ-8	2											X		X
REQ-9	3											X		X
REQ-10	3											X		X
REQ-11	2		X			X		X	X					
REQ-12	2					X								X
REQ-13	2		X											
REQ-14	1													X
Max PW														
Total PW														

Fully-Dressed Description

Use Case: UC-1 Register
Related Requirements:REQ-1, REQ-4, REQ-14, REQ-15
Initiating Actor: Customer
Actor’s Goal: To register an account on the App
Participating Actors: System, Customer
Preconditions: The system will ask the customer information to register.

Postconditions: The customer will have created an account that will be store in the database.

Flow of events for main success:

- > 1. The customer opens the app to register an account
- < 2. The system requests necessary information from the customer
- > 3. The customer enters in the necessary information
- < 4. The system registers the username into the database for the customer

Flow of Events for Extensions (Alternate Scenarios):

2A Username is already taken in the database

- < 1. System detects an error
- < 2. System notifies customer of error
- < 3. System returns user to the registration page

Use Case: UC-5 Pick Spot

Related Requirements:REQ-2, REQ-5, REQ-6,REQ-16,REQ-17, REQ-18

Initiating Actor: Customer

Actor's Goal: Customer can pick an open parking spot from the app

Participating Actors: Customer, System

Preconditions: The system stores information about the current status of all parking slots as open or closed.

Postconditions: The system will update the status of the chosen parking spot

Flow of events for main success:

- > 1. Customer will access the app to choose a parking spot
- < 2. System will display a visual representation of the parking spots. Each spot will indicate whether they are open or closed
- > 3. The customer will choose an open parking spot to take
- < 4. The system will update the status of the chosen parking spot
- < 5. The System will show a confirmation message

Flow of Events for Extensions (Alternate Scenarios):

2A User chooses a closed spot

- < 1. The system will give a warning message

3A An open spot is taken as the customer is browsing

- < 1. The system will update the visual representation of the parking spots to match the current state of the slots

Section 5: Effort Estimation Using Use Case Points

Use-Case Complexity	Number of Transactions	Use-Case Weight
Simple	≤3	5
Average	4 to 7	10
Complex	>7	15

Number of Simple use cases: 6
Number of average use cases: 5
Number of complex use cases: 2
Unadjusted uses case weight: 110

Actor Complexity	Example	Actor Weight
Simple	A System with defined API	1
Average	A System interacting through a Protocol	2
Complex	A User interacting through GUI	3

Number of Simple actors: 1
Number of Average actors: 2
Number of complex actors: 4
Unadjusted Actor weight: 17

Unadjusted Use case points = UUCW+UAW = 110+17 = 127

Section 6: Domain Analysis

Domain Model

Concept Definitions

UC-1 Register

Responsibility	Type	Concept
Coordinates actions of concepts associated with this use case and delegates the work to other concepts	D	Controller
The visible pages associated with the options to sign in or sign up	K	Interface Page
Prepare a database query with information on existing accounts and credentials	D	Database Connection
The system will make sure no data matches with existing credentials for login are made	D	Postprocessor
Initializes sign up process started by user	K	Sign Up Request
Add sign up information to database for new user	D	Archive
Notify user of successful account registration	D	Notifier

UC-5 Pick Spot

Responsibility	Type	Concept
Coordinates actions of concepts associated with this use case and delegates the work to other concepts		Controller
User specifies desire to search for available parking spot		Parking Spot Request
Database records of nearby available parking spots as well as occupancy charts will be requested		Database Connection
Shows the current context of available spots, and what actions can be taken for desire outcome		Interface Page
Retrieved data must sorted by available and unavailable parking spots		Postprocessor
Retrieved data converted to map display for user		Page Maker
Updates database based on selected parking spot and amount of time desired		Updater
Archives parking request under account history		Archiver
Notifies user of selected parking spot if valid		Notifier

Association Definitions

Concept Pairs	Association Description	Association name
Interface Page←→Controller	The user can choose to login in to an existing account, or create a new one.	Conveys request

Controller ←→Sign Up Request	The controller sends a request to create a new account, with information provided by the user.	Conveys request
Sign Up Request←→Archiver	The user enters their information for a new account.	generates
Archiver ←→ Database	The new account information received by the database is stored.	Provides data
Interface page ←→ Parking Spot Request	The user will select where he/she wants to park based on the lot number and the available parking spots.	Conveys request
Database←→Postprocessor	Convert Database Info Into User Display.	Provides data
Postprocessor←→Interface Page	Display Choices Based On User Request.	Provides data
Updater←→Database	Update Database Based on User Selection.	Provides data
Archiver←→Database	Archive requests to save user data in the database.	Request save
Notifier←→Interface Page	Notify User of Selections that are invalid.	Request notify
Page Maker ←→ Database	The database passes the retrieved information to the page maker to be displayed.	Provides data

Page Maker ←→ Interface Page	The page maker prepares the display for the interface page.	Prepares

Attribute Definitions

Concept	**Attributes**	**Attribute Description**
Interface Page	Register	Creates a new customer account by requesting the customer's information.
	Login	The App will ask the customer's user ID and password to login.
	ManageAccount	The customer is able to edit and update the user info or delete the current profile.
	SelectCampus	Customer chooses which campus to park in.
	SelectLot	Customer selects the parking spot.
	CancelLot	Customer is able to cancel the current parking lot so he can choose another.
Database	UserInfo	The App will contain user's info such as Scarletmail, NetID, RU Parking Permit, Driver's License, phone number, etc.
	UserID	Customer's username for the app
	UserPassword	Customer's password to login.

	RecordTime	Records the time to the database when the customer enters and exits the parking lot.
	DataAnalysis	The systems records the customer's parking patterns to predict and locate the best parking lot in the future.
	UserStatus	App takes into consideration user's status to determine which lots are permitted
Page Maker	DisplayMap	The system shows the map of the campuses and its associated parking lots.
	LotStatus	Shows which parking lots are currently open and which are closed.
	LotNumber	The system displays the selected parking lot number in which the customer will proceed to park.
Notifier	UserEntrance	Notifies the database when the customer has entered the parking lot.
	UserExit	Notifies the database when the customer leaves the parking lot.
	NotifyUser	The system sends helpful notifications to the customer such as: confirmation emails, how much time it is left, any special events, which campus has more open parking lots, etc.

Traceability Matrix

Use Cases	**Domain Concepts**							
	Interface Page	Database	Page Maker	Notifier	Archiver	Controller	Updater	Post Processor
UC-1	**X**	**X**		**X**	**X**	**X**		
UC-2	**X**	**X**				**X**		**X**
UC-3	**X**		**X**					
UC-4	**X**		**X**					**X**
UC-5	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
UC-6				**X**			**X**	
UC-7		**X**		**X**	**X**		**X**	
UC-8		**X**		**X**	**X**		**X**	
UC-9	**X**	**X**			**X**	**X**		
UC-10	**X**	**X**			**X**	**X**	**X**	
UC-11			**X**				**X**	**X**
UC-12			**X**			**X**	**X**	**X**
UC-13		**X**			**X**		**X**	

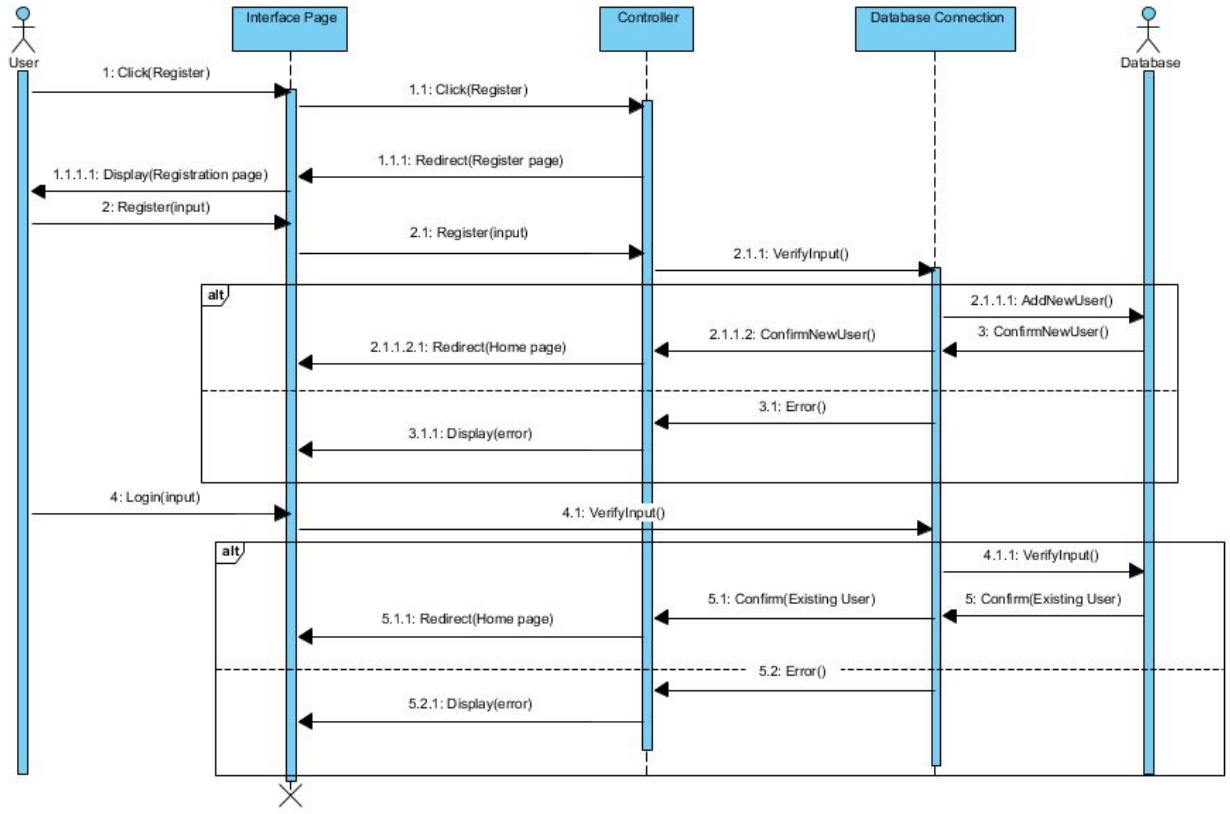
System Operation Contracts

Operation	Register
Preconditions	Customer has not registered into the system Customer will enter valid information
Postconditions	The account created by the customer is stored in the database

Operation	Pick Slot
Preconditions	Customer has not picked a slot
Postconditions	The slot picked by customer is remembered by the system

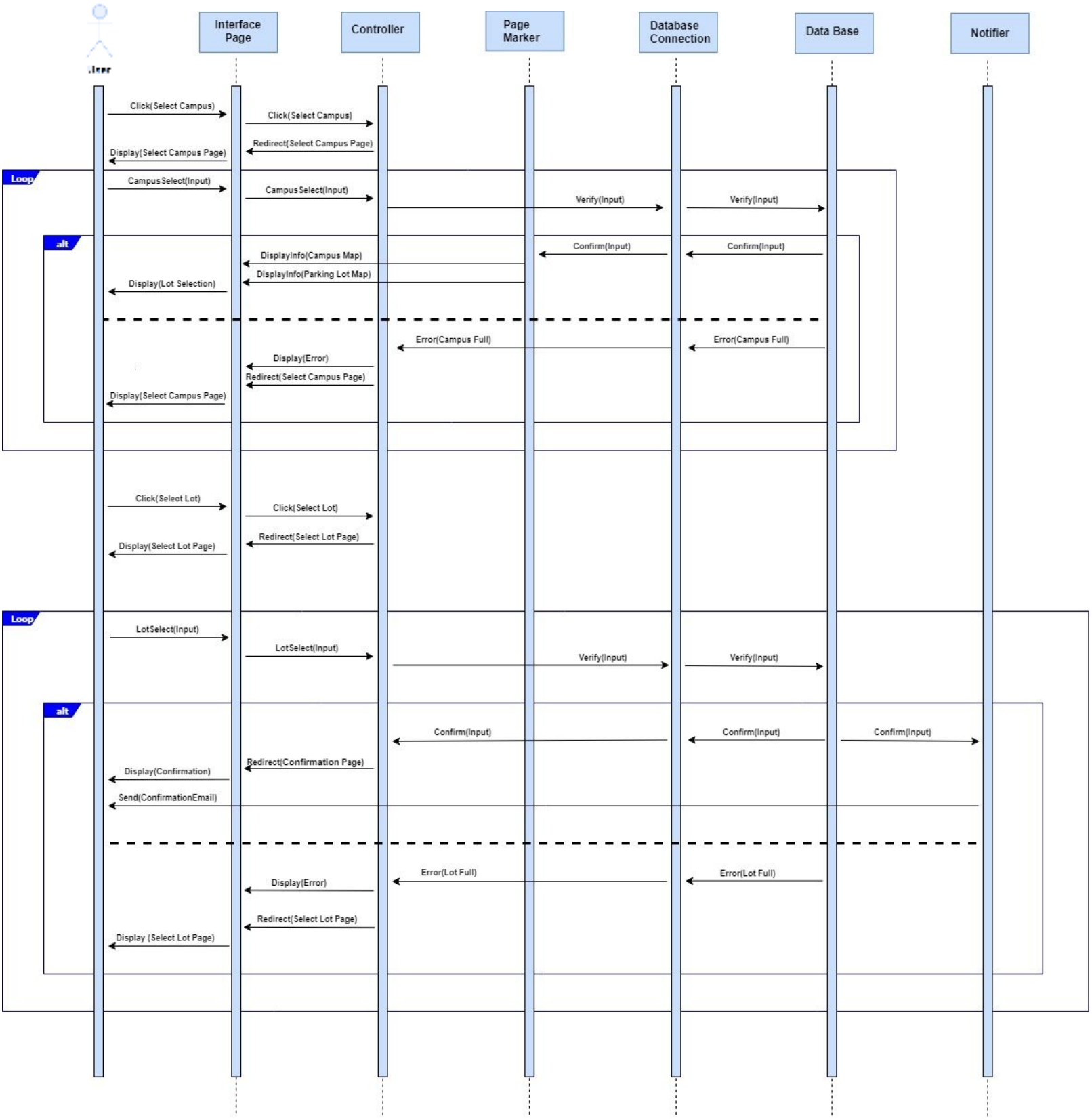
Section 7: Interaction Diagrams

UC - 1 Register + Login



Here much of the responsibility is assigned to the controller to communicate between the interface page and the database connection.

UC - 5 Pick Spots

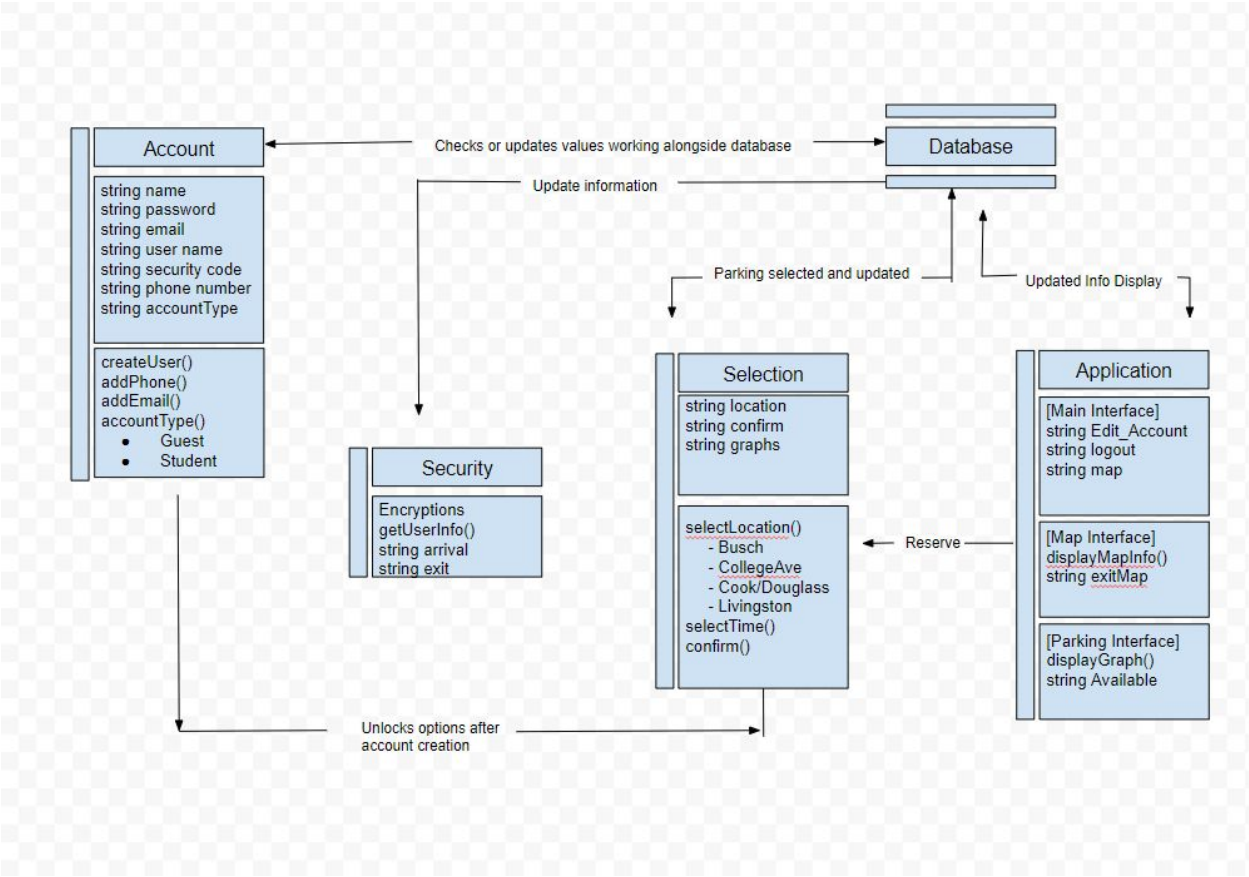


use of Design Patterns

After the user logs in successfully. The user will now attempt to select the preferred campus and parking lot. The Database has the most important responsibility of keep track the availability of the lots. The user can choose different campuses. Once a campus is chosen all the lots that the user can park in will be displayed. The user can than choose to see the location of the parking lot via google maps. They can also choose to see the predictive availability of the lot throughout the week. So after each selection input the database will verify both the campus and parking lots' availability. Then the data will be retrieved by the Page Maker in which its job is to convert the most updated information on the status of the lots into a map for user's convenience and display it for on the Interface Page. Meanwhile, the Controllers will direct the data to its corresponding locations.

Section 8 : Class Diagram and Interface Specification plus Design Pattern and OCL Contract Specifications

2.1 Class Diagram



2.2 Data Types and Operation Signatures

1) Account

(A) Attributes

string name : user's full name

string firstName : account user's first name

string lastName : account user's last name

string phone number : user's phone number

string password : user selects a password

string date_of_birth : user's date of birth

string security code : security code sent for confirmation to the account

string forgot_password : When password is forgotten

int userID : identification number assigned to the user

(B) Processes

createUser( ) : creates the new user with the inputted information and adds to the database

addPhone( ) : binds phone number to user account for notifications

addEmail( ) : used as an identifier as well as backup for sending notifications

2) Application

I. Main Interface

(A) Attributes

string Location : Can select the location for parking

string Account : Manage the user account

string Logout :To logout of app

(B) Processes

userInterface( ) : takes user input and moves throughout the screen

II. Map Interface

(A) Attributes

string Map : Displays the map of the area

(B) Processes

getLocationInfo( ) : Allows user can view the surrounding area in map form

III. Parking Interface

(A) Attributes

string diplayGraphs : Shows graphs of availability at specific times

string Available : Allows user to see what parking spots are currently available

(B) Processes

displayCurrentInfo( ) : Shows current info regarding availability

displayCurrentGraphs( ) : Info on current events that might affect parking

3) Selection

(A) Attributes

string location : Allows user to view available locations

string busch : Info for Busch campus

string college_ave : Info for College Ave campus

string cook_douglass : Info for Cook/Douglass campuses

string livingston : Info for Livingston campus

string graphs : Displays availability graphs

(B) Processes

selectLocation( ) : Sets the user selected location that they what to reserve at

confirmSelection( ) : Finalizes the reservation and prepares notifications

displayGraphs( ) : Shows the user availability graphs

5) Security

(A) Attributes

encryption : Encrypts data

string arrival : arrival time at lot recorded to update database

string exit : exit time of lot recorded to update database

(B) Processes

sendCode( ) : Sends user confirmation codes during important instances like creating an account or when password is forgotten.

getUserInfo( ) : Can display user info only when properly logged in

2.3 Traceability Matrix

	Concepts					
Classes	Database	Security	Main Interface	Map Interface	Parking Interface	Notification Controller
Registration	X	X	X			X
Login	X	X	X			
Parking Availability	X			X	X	
Selection	X		X			X
Location Info	X			X		
Parking updates	X					X
Area Map	X			X		
Database Connection	X	X	X	X	X	X

1) Database

(A) Responsibilities

- Holds customers' registration information as well as any other info added to the account such as where to send notifications
- Holds map and event data to display to the user and updates data with recent events
- Holds info in graph form regarding availability during the day

(B) Classes

- displayAvailableParking( ) : uses the database information to display the current available parking information to the user upon request

- displayAreaMap( ) : uses database information on the local area to generate a map of the area and display to the user parking lots nearby and other areas of interest
- registerUser( ) : allows user to register and account and add it to the database
- login( ) : when a user enters credentials, it is confirmed with the database information to identify the user and allow them to login

2) Main Interface

(A) Responsibilities

- Displays options to the user to navigate the application that includes: Edit Account, Make Selection, Display Map, and Logout.

(B) Classes

- changeAccount( ) : allows user to make account changes and updates database
- displayMap( ) : shows the map of the local area based on database information
- makeSelection( ) : allows user to make a parking selections
- checkHistory( ) : allows user to check their own current or previous registration
- logout( ) : logs out of the account

3) Map Interface

(A) Responsibilities

- Displays to the user the local map of selected location and shows parking lots in the area

(B) Classes

- getLocationInfo( ) : displays map of surrounding area for user benefit

4) Parking Interface

(A) Responsibilities

- Gives the user the options to properly select the location they wish to park at and then updates database on location availability

(B) Classes

- getAvailabilityInfo( ) : allows user to attain info on parking availability in locations such as Busch, Livingston, College Ave, and Cook/Douglass based on information that is available in the database
- makeSelection( ) : users can move directly to making reservation after seeing the available information on the map
- checkGraphs( ) : Check availability through a graph of different times of day

6) Security

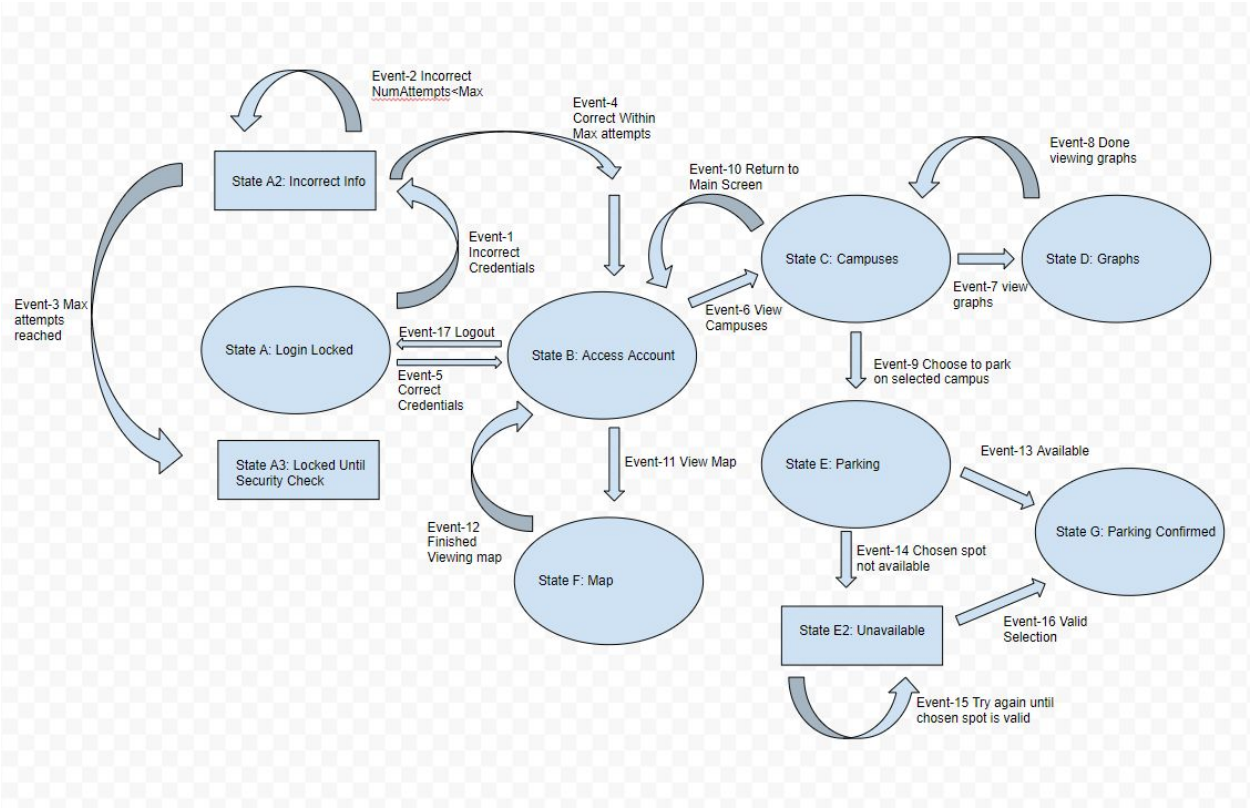
(A) Responsibilities

- Encrypts user data and uses authentication codes when needed

(B) Classes

- confirmationCode( ) : sends user code to confirm account for security reasons and confirms with database

State Diagram:



A state diagram was utilized to visualize the process interactions through the use of events. This allows for the consideration of user interactions further to ensure a simple easy to understand methodology is used to limit user error and lead to a satisfactory experience. By visualizing the connections we can see that the methodology used to arrive at the parking confirmed final desired location is streamlined and intuitive.With the use of the state diagram and test cases, user satisfaction can be ensured. Below are the specifications:

State A: Login Locked - Starting state of locked account

- Event 1: Incorrect Credentials: When inputted login info is not matching with those saved on database
- Event 5: Correct Credentials : When the correct login information is inputted

State A2: Incorrect Info - Incorrect input of credentials

- Event 2: Incorrect: Can reattempt if the number of attempts is less than max number of attempts
- Event 3: Max Attempts: The max number of attempts is reached and account is locked until user confirms security directly via emailed message or contacting support
- Event 4: Correct: The correct credentials were inputted within max number of attempts

State A3: Locked until Security Check

State B: Access Account - Account properly entered and unlocked

Event 6: View Campuses: Can move to screen to select from available campuses

Event 11: View Map: Can choose to move to map screen for surrounding area

Event 17: Logout: Can logout from signed in account

State C: Campuses - Selecting desired campus

Event 7: Graphs: Moves to graph screen of parking availability at different times of day in graph form

Event 9: Parking: Moves to parking screen of selected campus to start parking process

Event 10: Return: Return to main home screen the “Access Account State to view other available options

State D: Graphs - Viewing availability graph

Event 8: Complete Graph View: Return to campus selection screen once done with viewing graph for currently selected campus

State E: Parking - Selecting parking spot

Event 13: Parking selected at available lot with correct credentials selected during account creation.This means that the account selected whether guest or student etc., is an account that has permission for the particular parking lot. If that is true and space is available move to final state of parking confirmed.

Event 14: Not Available: Chosen spot is not available

State E2: Unavailable - Selected spot not available

Event 15: Retry: If selecting another unavailable spot try again

Event 16: Valid: Selected spot is available and parking is finally confirmed

State F: Map- Viewing map of surrounding area

Event 12: Done: Finished with looking at map so return to main screen for more options

Stage G: Parking Confirmed - Parking officially confirmed

Object Constraint Language Contracts

Important Contracts:

- Invariants:
    - context location inv selectLocation: select-> self.Busch | self.CollegeAve | self.CookDoug | self.Livingston
    - context create inv createUser: select -> forAll
- Pre-conditions:
    - context selection parkingSelection: select -> accountType.guest | accountType.student
    - context confirm Confirm : select -> selectLocation & selectTime

Section 9: System Architecture and System Design

Architectural Styles

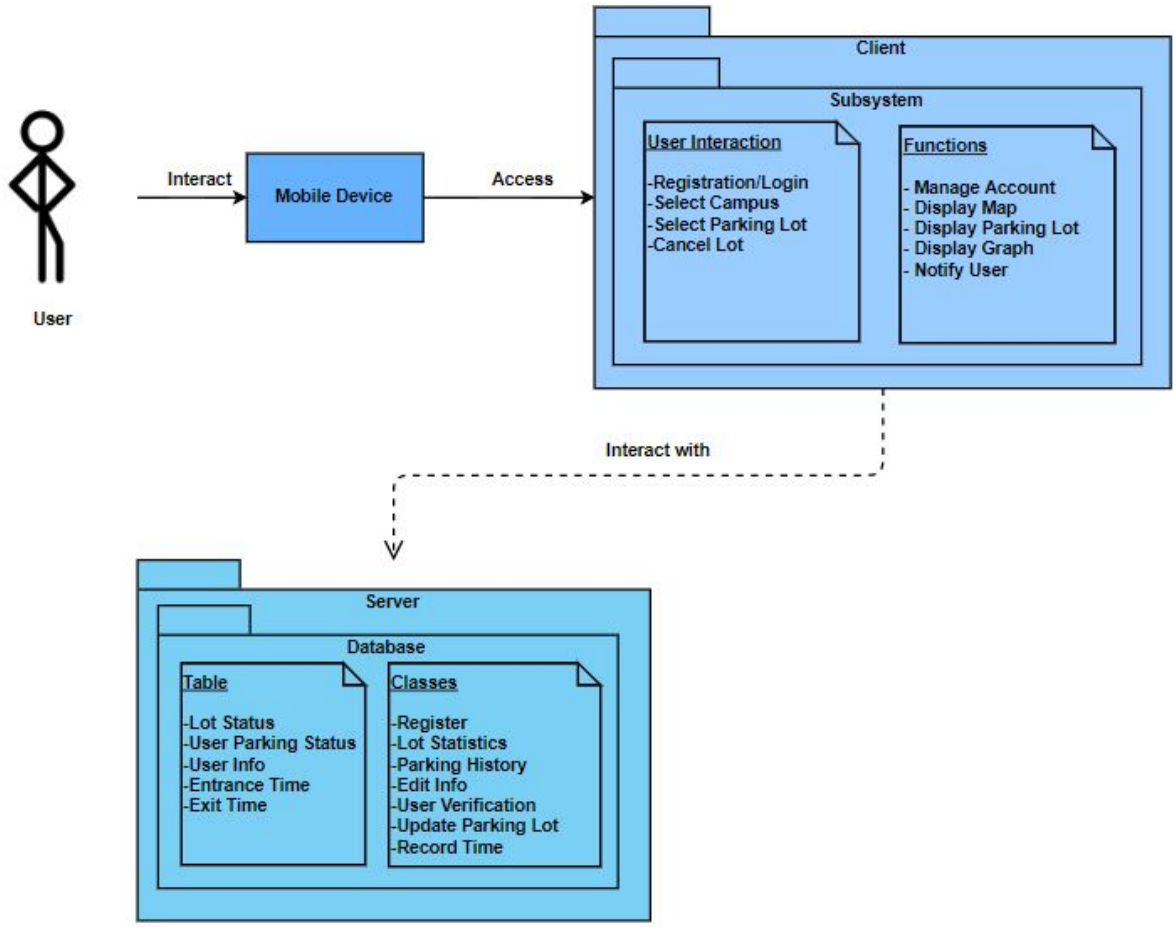
Overall, the architectural style our system follows is the layered architecture. There are three different layers: the view layer, application layer, and database layer. The top most layer is the view layer, which is what enables the user to access the application layer. The application layer then interacts with the database layer and finally, the database layer is where all the data regarding user information and parking lot information is stored.

The database and phone application will follow a client-server architectural style, because we need something to maintain and retrieve data, such as login information or parking lot information. The application will request a service from the server, and the server will access the information and display the requested information to the user

The event driven architecture is also implemented for displaying the status of each parking spot, as well as sending information directly to the user. When a spot in a parking lot is taken, the

status of that spot will go from vacant to occupied and vice versa. Additionally, if there are any lots that are full or closed due to events happening on campus, a message will be sent to the user.

Identifying Subsystems



The UML Package Diagram shows the interaction between the user and the client-server based application. The user will interacts with his/her own smartphone in order to request services.

Upon request, the client system will performs the user's requests by gathering information through the database from the server side. The database contains the information of the user's profile info, parking lot info, statistics, and necessary codes to update parking lot availability. On the client side, the application is able to perform tasks such as edit user's profile, select/cancel parking lot, and display interactive map from the subsystem..

Mapping Subsystems to Hardware

Web Servers:

The web server will contain the main database in which all of the user's information will be stored on the net. Its primary function is to store, process, and deliver specific information to the clients. Parking lot info and statistics will also be located here in which it will be sent to the client upon request. It will also maintain most of the codes for the application to run through clients' phones. Any new information will be updated immediately on the server so the client will be notified with the most accurate parking lot info.

Client Server:

The client will be using their mobile devices to utilize the parking app's feature. The interface will be provided for the user to request services from the web server. Obviously, Internet is required to access the server, and once connected, the client is able to set up accounts, upload class schedules, and search for parking lots. For example, the client will send requests such as parking lot info or registers in which the web server retrieves the info from the database and display it through the client server.

Persistent Data Storage

The system requires a relational database to store key data about the users and the lots for the system to run correctly.

Network Protocol

The system is going to use HTTP communication protocol because of the fact that it is a stateless protocol so the people using the app can make a request to get some data and the server will return that required data. We could use other protocols but this will be the easiest to use for an app.

Global Control Flow

Execution Orderness

Our service is mainly procedural. The order of actions the user will have to take will always be linear. For instance, the user will always need to verify log-in before making any other actions. And before choosing a lot, the user will need to select a campus. In between these steps, the application will make requests to the server to populate user interface with data selections.

Our service also employs some event driven systems. This is used mainly for notifications i.e. if a reserved lot becomes full before the user get there, a message from the server will trigger a notification event that requires the user to pick a new lot. Another case is when the user swipes into a lot, they will receive a notification confirming their vacancy as an event from the database.

Time Dependency

The service will not rely on any timings as requests for data are made at user input. However, time will be used by the app on entry of a lot and sent to the database as part of the entry log.

Hardware Requirements

Users need a working phone in order to use our app.

Garage will need a scanner to scan license plates as they come in.

We will need 10gb of hard disk space to host the database.

As we code and implement our design more hardware requirements may arise, in such a case we will add them.

Section 10: Algorithms and Data Structures

Algorithms

One of the main algorithms that we need is to keep track of the number of users in a lot at a given time. This will help predict how filled a lot is, so the user can determine whether or not they can park at the given lot, or if they need to park in a different lot. When the app user looks at the app, they should be able to see the current occupancy of the lot. When a driver decides to park in a lot, the will swipe to get in. The swipe will increment the occupancy of the lot. When the driver leaves, they will have to swipe out, and it will decrement the occupancy. The daily occupancy for each day at specified hours will be saved. With this data, a chart will be created that shows the expected occupancy on a given day. An average of the occupancy will be determined with data from the last three weeks. This will be calculated by adding up the data, and then dividing by three to be displayed to the user.

Data Structures

There are two different databases set up using the SQLite database that is integrated within Android Studio. The fist database is used to hold the student information such as :

- RUID: int
- Email: string
- Password: string
- First Name: string
- Last name: string
- Lot access: int
- Temp Lot: string (for additional parking privileges)
- Faculty: int (-1 for student, 1 for faculty)
- Day: int

Queries made here are used to verify if the login information of the user if valid or not, at which campus they have access to park on.

The other database is used to hold the information of each parking lot on each campus. It has information such as:

- ID: int
- Name: string
- Campus: int
- Faculty: int (1 for faculty lots, -1 for student lots)
- Curr: int (for the current number of cars in the lot)

It also has the occupancy for each week day for the time frames 10-2, 2-4, 4-6, 6-8 and 8-10. This is used to help predict the future occupancy for each lot for the user to see.

Section 11: User Interface Design and Implementation

We modified the initial login screen with a few basic design features to make it look more appealing and just to make it look nice. We first change the background to a rutgers page since our app will be dealing with rutgers affiliates. We also changes the font login page to match the background a little better. Android studio only has three fronts to work with so we were limited in that sense. The rest of the pages are very basic because we did not want to confuse people or have too much going on. I think that flow well and are organized in manner that makes its less confusing. We also used a google maps api to show the location of the individual lots. In this app we focused more on functionality then the aesthetic because we found that an app that word and does its job it's better then one that is cool but doesn't do much.

Section 12: Design of Tests

Test-Case Identifier:	TC-1
Use Case Tested:	UC-1
Pass/Fail Criteria:	The test case passes if the user can create an account with non-existing credentials
Input Data:	New user email, chosen password, confirm password, status

Test Procedure:	**Expected Result:**
Step 1: Select a unique email and password you wish to associate with newly created account and press "Register"	System recognizes unique email and password and allows account creation by confirming with database that email is unique
Step 2: Enter previously created account info into registration process for a new account	System matches email with email from existing account and thus returns "error email already in use"
Step 3: Enter unmatching passwords for "confirm password"	System responds with "error, passwords do not match"
Step 4: Enter only email, or password, or confirm password (only one)	System responds with "error, complete all required fields"
Step 5: Select the account status	System allows account type to process and updates database

Test-Case Identifier:	TC-2
Use Case Tested:	UC-2
Pass/Fail Criteria:	The test case passes if the user can login to the account with existing email and password with less than max number of allowed attempts.
Input Data:	Existing user email, existing password

Test Procedure:	Expected Result:
Step 1: Enter proper credentials on the very first attempt	System will successfully allow user to enter account
Step 2: Enter proper email and password of an existing account previously created within maximum allowed attempts	System will state incorrect information but allow user to attempt login again
Step 3: Incorrectly enter password for existing email exceeding or equaling maximum allowed attempts.	System will restrict access to account for a predetermined amount of time
Step 4: Enter either only the email or the password (only one)	System will respond with “error, complete all required fields”

Test-Case Identifier:	TC-3
Use Case Tested:	UC-3, UC-4, UC-5, and UC-11
Pass/Fail Criteria:	The test case passes if the user can select a particular campus, followed by being able to view and select an available lot on that Campus
Input Data:	Chosen campus, chosen lot

Test Procedure:	Expected Result:
Step 1: Select from one of the four Rutgers campuses then select an available parking lot that shows up as available on the info screen of the lot	System will successfully allow user to proceed with selected parking spot and display information that it is currently available
Step 2: Select one of the campuses then choose a lot that shows up as unavailable on the information screen	System will state in the info section that parking is currently unavailable in the selected spot. For example the College Ave parking deck becomes available to anyone after 5:00 pm.

Test-Case Identifier:	TC-4

Use Case Tested:	UC-5, UC-7, and UC-8
Pass/Fail Criteria:	The test case passes if the user can select a parking spot and set up the arrival and exit time for their parking
Input Data:	Chosen lot, arrival time, exit time

Test Procedure:	**Expected Result:**
Step 1: Select a parking spot based on rules and availability of the lot as well as the user's status	System will successfully allow user to proceed with selected parking spot
Step 2: Select a parking spot and an arrival and exit time that breaks a rule for the particular lot (Ex: Reserving College Ave Parking Deck before 5:00PM with a student status)	System will state that the selected times are invalid and direct user to info regarding the restrictions of their selected parking spot
Step 3: Select a parking spot that is not allowed and try that parking spot again	System will respond with "try again" to signify trying a different spot

Test-Case Identifier:	TC-5
Use Case Tested:	UC-11 Main information, UC-12
Pass/Fail Criteria:	The test case passes if the user can select map view and the user can view the map of the desired location and look around for greater understanding of the area.
Input Data:	Map view, map interactions (move, etc.)

Test Procedure:	**Expected Result:**
Step 1: Select map view on the app	System will successfully allow user to select map view opening up interactable map
Step 2: Select map view and click on the screen in random locations other that the go back option	System will do nothing as only the signified buttons will allow for something to happen
Step 3: Attempt to move around with registered inputs	System will respond accordingly ("move left" will move the map, etc.)

Step 4:Attempt to interact with the map with unregistered inputs	System will do nothing with unregistered inputs (clicking randomly at nothing will do nothing or shaking the screen will do nothing)

Test-Case Identifier:	TC-6
Use Case Tested:	UC-9, and UC-10
Pass/Fail Criteria:	The test case passes if the user can edit account information such as change password, change username, or even delete the account
Input Data:	Change email, change password, view account information
Test Procedure:	**Expected Result:**
Step 1: After having already logged in select the "view account information" option and choose change email	System will allow the user to change their email to a different email with verification of account currently as account has already been successfully logged on to; A notification of change is also sent out
Step 2: After already being logged in the user navigates to change password	The user can change their password and receive a notification of a password change
Step 3: The user attempts relogging in with updated email and password	The database already updated the user's info during change and so the user should be able to log in with their new credentials
Step 4: The user attempts logging in with their old credentials	The system notifies the user of incorrect email or password

Test-Case Identifier:	TC-7 (Integration Test)
Pass/Fail Criteria:	The test case passes if the user can login after registering and proceed to make a reserve or view information about selectable parking lots. The user must then be notified of important information like selected parking spot.
Methodology:	Integration will be done by combining previously separated coding by identifying the important factors of each code's variables and

	functions and proceeding to use that information to update the other existing code portions. For example, the functions defined for making a reserve will be noticed and used when developing the notifications portion of the code to ensure code unity.
Integration Types:	**Description:**
A) Interactive Map ←→ Database/UI	The Interactive Map will be integrated with the database to ensure proper map display while the UI functionalities must also be considered for interactions
B) Login ←→ Database ←→ Security	Login must be properly secured and also connected to the database for authorization through correct credentials
C) Register ←→ Database ←→ Security	Similar to login, registering must update the database regarding the new account and be properly secured
D) Notifications ←→ Database/Reservation	Notifications must be sent when database changes in account occur or a reservation is made
E) Reserve ←→ Database	When reserves are made database must track reserve and update info on user parking spot

Test-Case Identifier:	TC-7
Use Case Tested:	UC-13
Pass/Fail Criteria:	The test case passes if the user can view the graphs of the selected campus lot's availability at different times of day and then return.
Input Data:	Select graphs, exit graphs
Test Procedure:	**Expected Result:**
Step 1: After selecting a campus lot choose to view availability information	System will show updated availability information in graph form obtained from database

Step 2: Click anywhere on the graph at random locations that are not interactable	The system will do nothing and wait for a correct input to proceed to something different
Step 3: Choose the option to go back to campus lot view	The system will respond and take the user back to the previous screen

Section 13: History of Work, Current Status, and Future Work

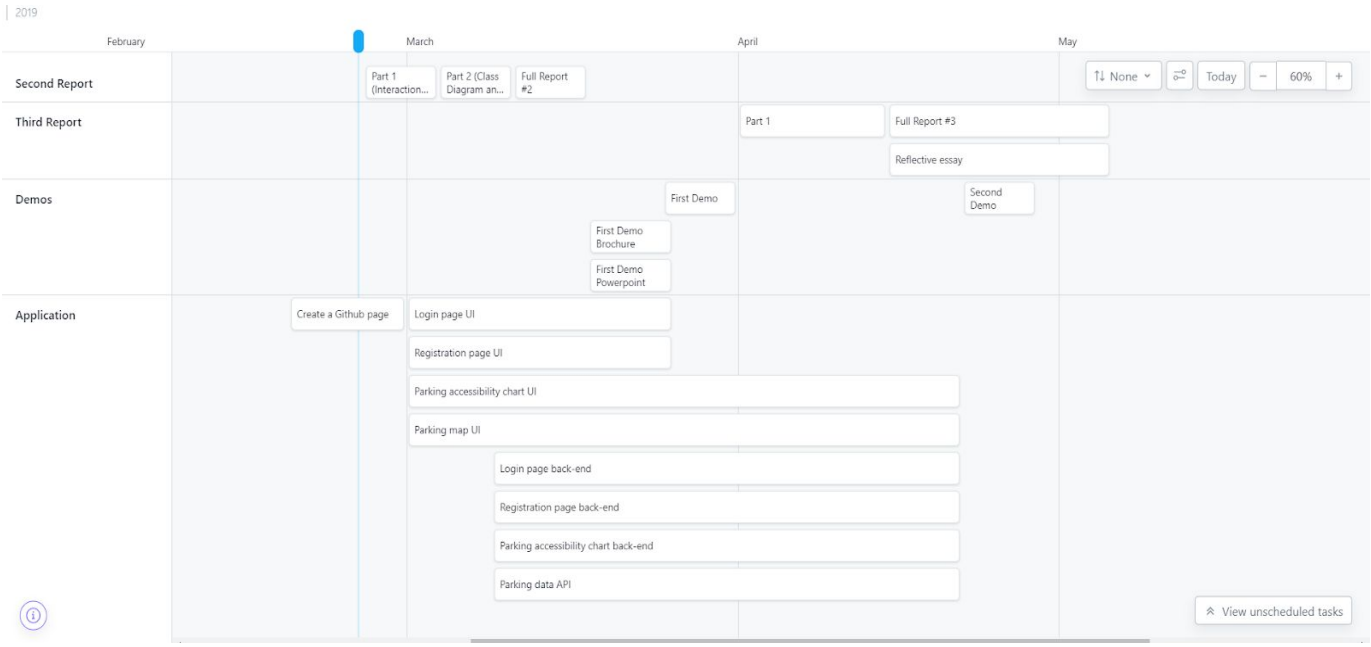
Plan of Work

Merging the Contributions from Individual Team Members:

To ensure that the final copy of the report is consistent we need to establish a set format for the various diagrams and chart. We must also agree on more minute details of the formatting like the use of indents and so on.

Project Coordination and Progress Report:

We are currently tackling the project by splitting the programming of the use cases, UI and backend development amongst the team members.



Breakdown of Responsibilities/Work Done

- App Development
    - Andrew
    - Anthony
    - Suva
- UI
    - Anthony
    - Max
    - Josh
- Tests
    - Jahidul
- Database
    - Krithika
    - Suva

Section 14: References

"An Overview of HTTP." *MDN Web Docs*, developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

Banerjee, and Associates. "An Overview of Common Parking Issues, Parking Management Options,and Creative Solutions." *Pipta*, pipta.org/wp-content/uploads/2014/04/Parking-Problems-and-Creative-Solutions.pdf

Tutorialspoint.com. "Estimation Techniques Use-Case Points." *Www.tutorialspoint.com*, Tutorials Point,

Marsic, Ivan. "Software Engineering Course Lecture Slides." *RU ENG ECE 14:332:452*, www.ece.rutgers.edu/~marsic/books/SE/instructor/slides/.