Emma Brunell, Michael Doran, Andrew Smith

Project Description

The Parking Buddy App will provide information to students at SUNY Oswego about available commuter parking spots, namely in the parking lots by Shineman and Sheldon Hall. Parking for commuter students can be very difficult and time consuming, as there is limited parking. Students tend to spend large amounts of time circling the parking lots looking for an open space. This causes many students to be late to class. It can sometimes be difficult to predict when there will be the most number of open parking spots as well.

The idea is to create an app for Android -- possibly for IOS eventually as well -- that will allow students to plan ahead for their parking. The app will allow students to check in and out of parking spots. The parking lot information will be provided by an overhead view of the lot, similar to Google Maps, so that the layout is accurate. A red icon in a parking spot will indicate that it is currently being used, and a green icon will indicate that the spot is available to be parked in. Users will have to find their parking spot on the map and check in and out of it. They will also be able to provide information on how long they plan on staying in the spot. Other users will be able to see about how long it will be until various parking spots will become open.

Additionally, the app will collect information on times that students are checking in and out of parking spots. There will be a section in the app dedicated to providing intel to users about popular times for parking throughout the day, in the form of a graph.

System Requirements

Identifier	Priority	Requirement
REQ1	5	The information on parking spots maintained by the system, should be available to all instances of the app
REQ2	2	The system allows users to mark a spot as “obstructed” if unavailable for a reason other than the spot being taken (snow, construction, etc.).
REQ3	5	The app has a GUI that provides an overhead view of the Shineman and Sheldon parking lots.
REQ4	5	The app allows users to “check out” parking spots and be able to mark them as “available” or “taken”.
REQ5	2	The system allows the user to set a timer for how long they plan on being parked, after which the spot becomes available again.
REQ6	3	The system takes analytics for how long a spot is occupied and when it is most often open. These analytics will be available for viewing in a separate part of the app.

User Stories

Identifier	User Story	Size
ST-1	I want other users to be able to view any other user's parking updates in real time.	9
ST-2	I want to be able to provide information to other users about potential problems with parking in certain spots.	2
ST-3	I want an accurate map of the parking lot to be used to view parking locations.	10
ST-4	I want to be able to check my car in and out of a parking spot as needed.	4
ST-5	I want to be able to provide information to other users about how long I am planning on staying parked in a spot.	6
ST-6	I want to be able to view common times that a parking lot is most heavily used.	5

Use Cases

Use Case UC-1: Update
Related Requirements: REQ1, REQ2, REQ3, REQ4, REQ5
Initiating Actor: Student Commuter
Actor's Goal: To view the most current version of the parking lot's situation.
Participating Actors: Map, other commuters
Preconditions: The program is functioning properly with the server to take real-time updates.
Postconditions: The map refreshes with any new changes added by users.
Flow of Events for Main Success Scenario:

1. Student Commuter opens app and selects wanted parking lot
2. Student Commuter selects spot and chooses to enter, leave, or declare an obstruction.
3. This newly changed spot displays to all others using the app on this page.

Use Case UC-2: Mark Obstructed
Related Requirements: REQ1, REQ2, REQ3
Initiating Actor: Student Commuter
Actor's Goal: To let other Student Commuters know if a specific parking spot is unable to be used for a reason besides another car parking there.
Participating Actors: Map, other commuters
Preconditions: The program is functioning properly with the server to take real-time updates; the parking spot is non-null; the parking spot has not already been marked filled.
Postconditions: The map displays to other users a symbol that means that the parking spot is unable to be used.
Flow of Events for Main Success Scenario:

1. Student Commuter opens app and selects wanted parking lot
2. Student Commuter notices that a particular spot is unable to be parked in , due to being covered up by a snowbank, construction, or a similar reason.
3. Student Commuter selects parking spot and chooses it to be obstructed.
4. The parking spot displays to all other users that it is obstructed.

Use Case UC-3: Scrollable Map
Related Requirements: REQ1, REQ3
Initiating Actor: Map
Actor's Goal: To display an accurate layout of the parking lot selected by a user.
Participating Actors: Student Commuters
Preconditions: The GUI is in place and users are able to navigate through all of the states; the program is functioning properly with the server to take real-time updates.
Postconditions: The map displays to users any changes that occur; the map is able to be moved around on the screen according to where the user wants to park; the user is able to interact with the map.
Flow of Events for Main Success Scenario:

1. The map displays to the users and is not warped by differing resolutions.

Use Case UC-4: Check In and Out
Related Requirements: REQ1, REQ2, REQ3, REQ4
Initiating Actor: Student Commuter
Actor's Goal: To check themselves in or out of a selected parking spot so other users may plan for parking accordingly.
Participating Actors: Map, other commuters
Preconditions: The parking spot is non-null; the parking spot is unobstructed
Postconditions: The parking spot displays as either green (spot is empty) or red (spot has been filled).
Flow of Events for Main Success Scenario:

1. Student Commuter is entering or leaving a parking spot.
2. Student Commuter selects appropriate option to check in or out.
3. The map accurately displays to Student Commuter and other users this new update in real-time.

Use Case UC-5: Timer
Related Requirements: REQ1, REQ3, REQ4, REQ5
Initiating Actor: Student Commuter
Actor's Goal: To display an estimate of the amount of time they plan on being parked to other users so they may plan for parking accordingly.
Participating Actors: Timer, Map, other commuters
Preconditions: The user has checked themselves into a parking spot; the map is updating in real-time.
Postconditions: The timer is displayed to other users with a countdown of how much longer the spot will be filled for.
Flow of Events for Main Success Scenario:

1. Student checks themselves into a parking spot successfully.
2. Student selects an estimate for how long they will likely remain in that spot.
3. The timer displays to other users in real time a countdown.
4. The student checks out of the parking spot by the end of the timer countdown and the spot is marked empty. The timer also disappears with this action.

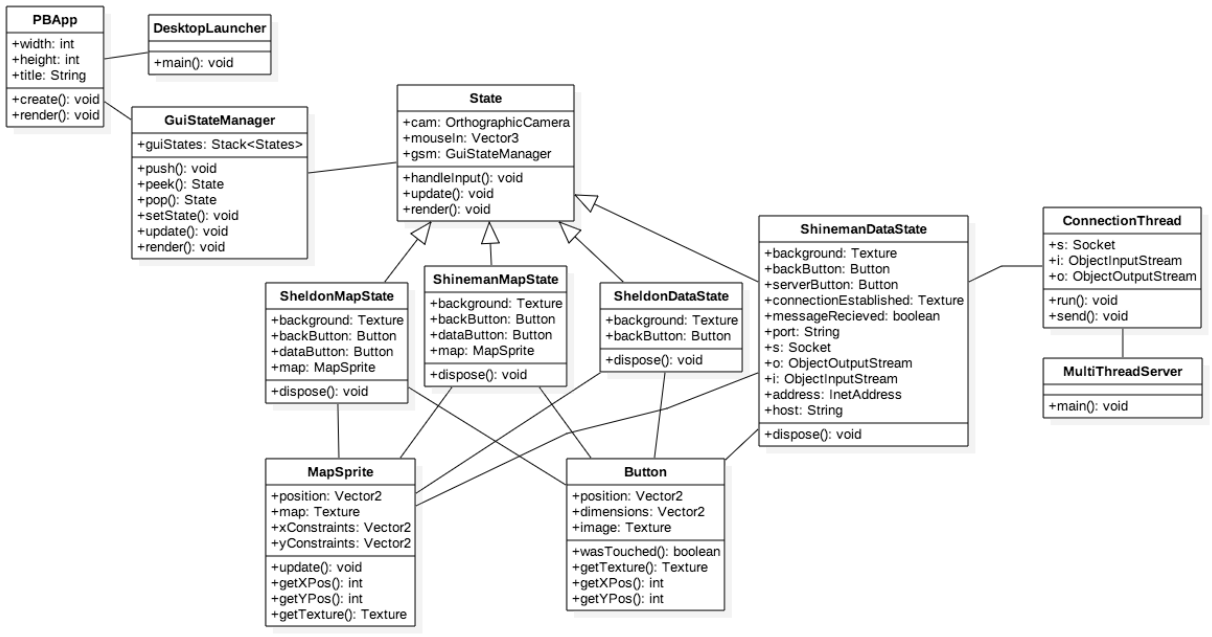
Use Case UC-6: Analytics
Related Requirements: REQ6
Initiating Actor: ParkingBuddy
Actor's Goal: To provide data on popular times parking lots are most used to other users.
Participating Actors: Student Commuters, Map
Preconditions: The maps have been interacted with to gather data on the usage of parking lots.
Postconditions: The data analytics page displays to users a graph showing when a parking lot is likely to be the most busy. This graph updates based on map interaction by student commuters.
Flow of Events for Main Success Scenario:

1. Student Commuters check in and out of parking spots accurately.
2. The analytics page saves the information gathered in UC-4 and displays it visually.

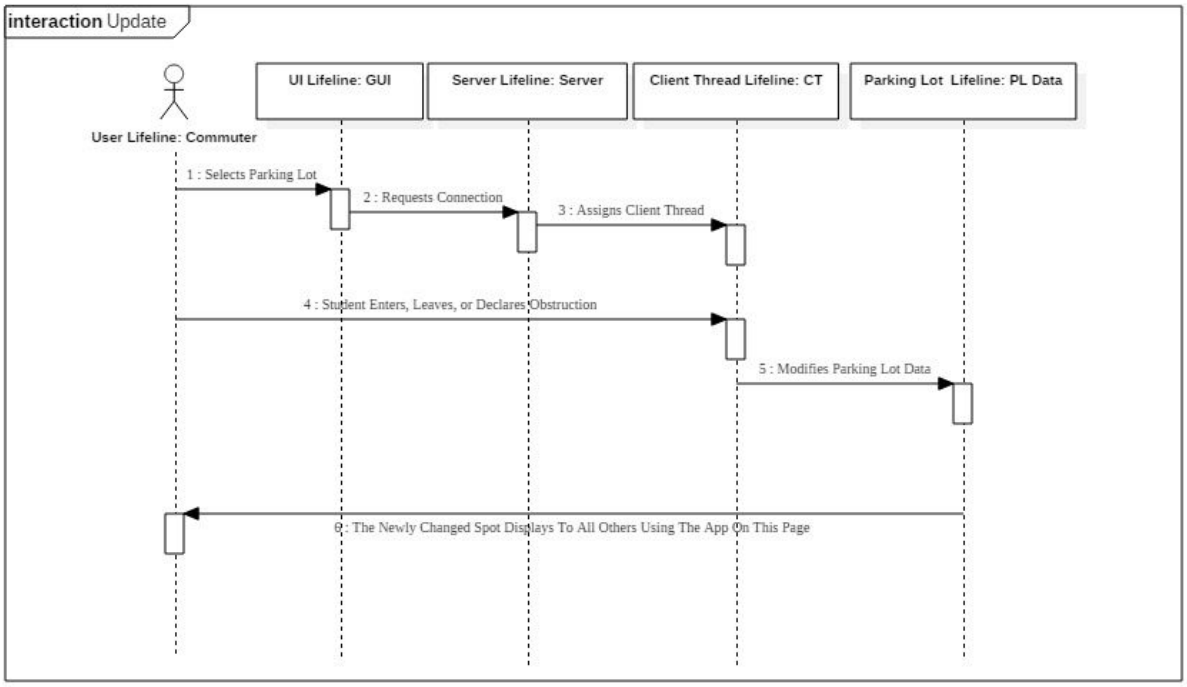
Traceability Matrix

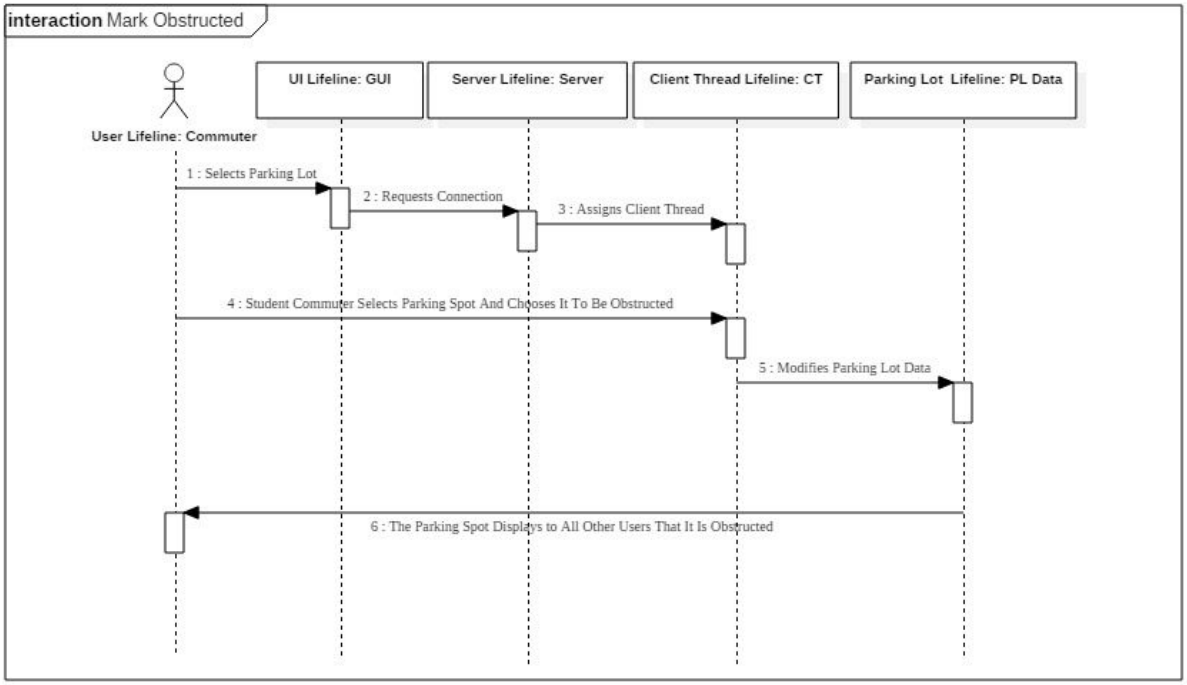
Req't	**PW**	UC1	UC2	UC3	UC4	UC5	UC6
REQ1	5	X	X	X	X	X	
REQ2	2	X	X		X		
REQ3	5	X	X	X	X	X	
REQ4	5	X			X	X	
REQ5	2	X				X	
REQ6	3						X
Max PW		5	5	5	5	5	3
Total PW		18	12	10	17	17	3

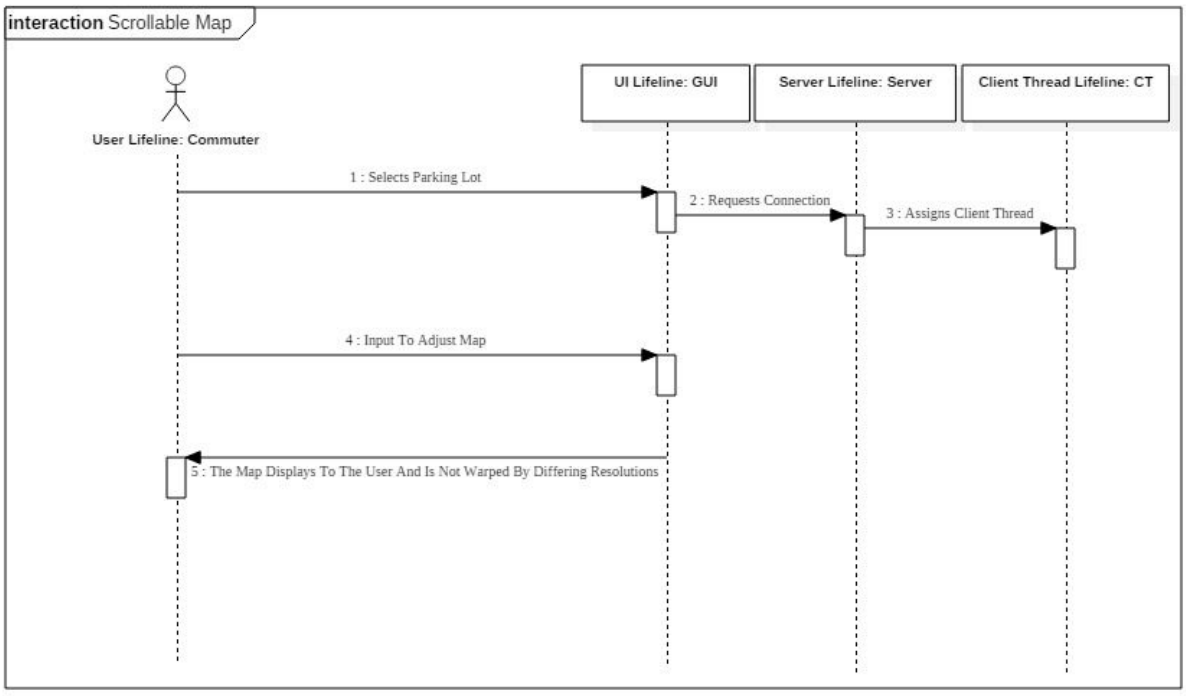
UML Classes Diagram

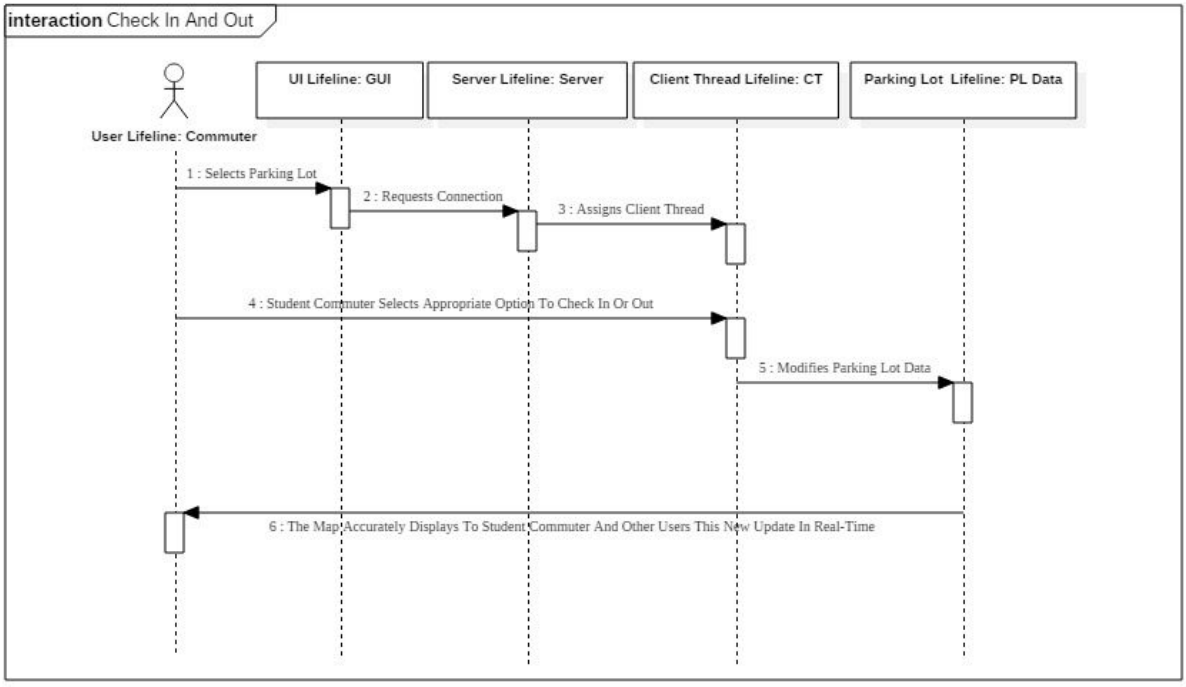


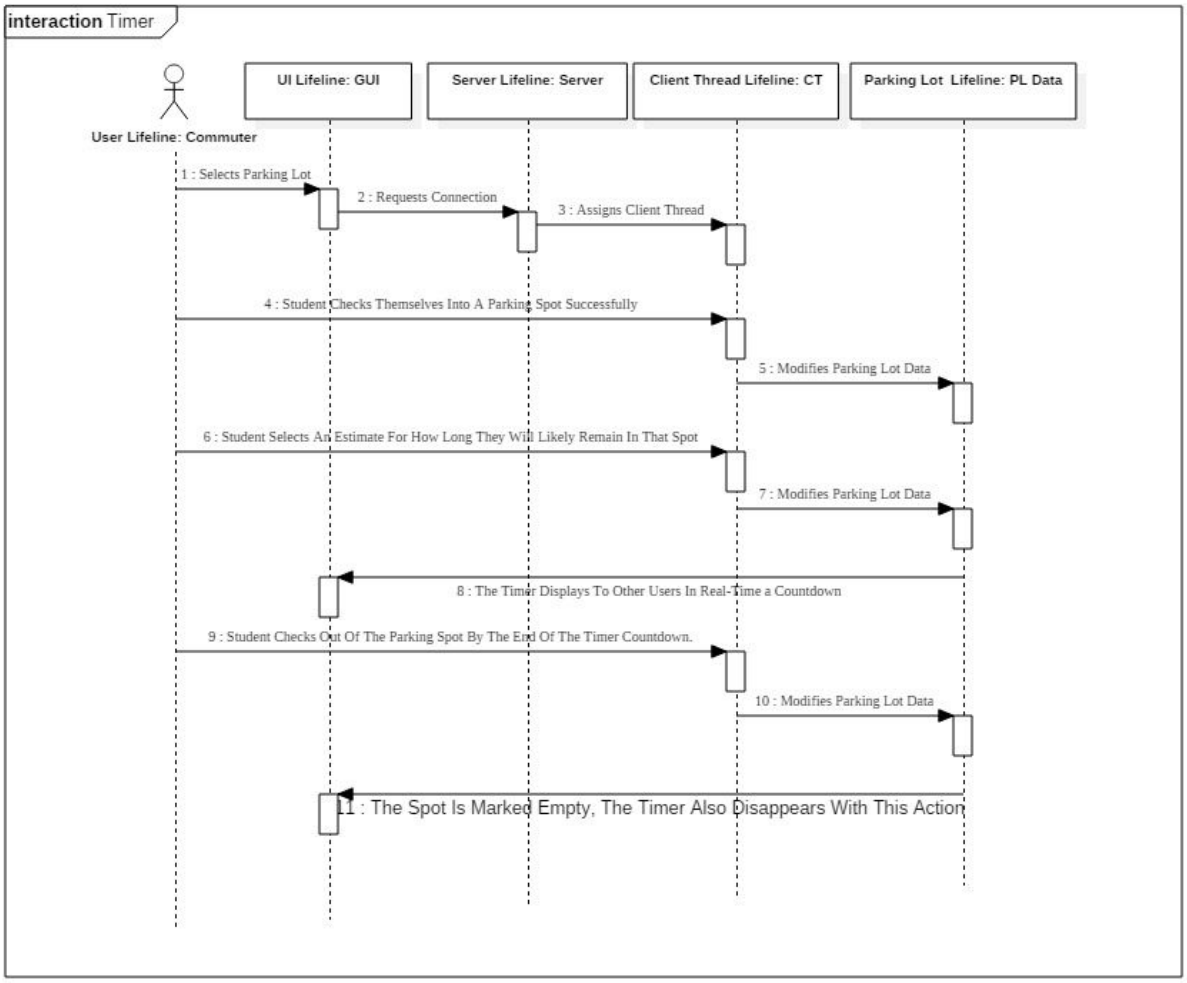
UML Sequence Diagrams

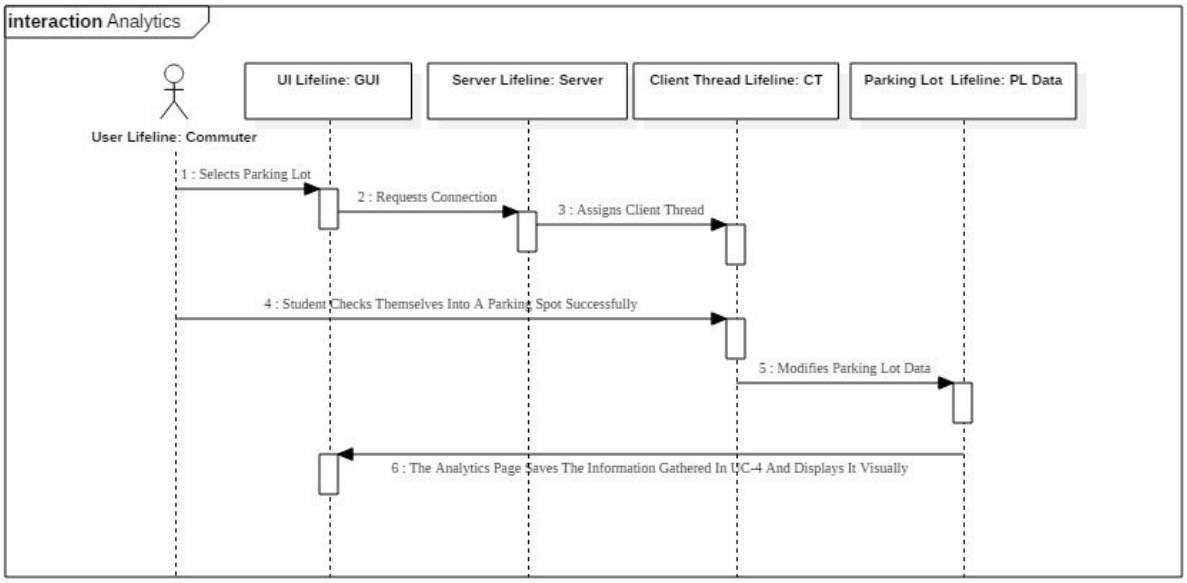





interaction Scrollable Map
User Lifeline: Commuter
UI Lifeline: GUI
Server Lifeline: Server
Client Thread Lifeline: CT
1 : Selects Parking Lot
2 : Requests Connection
3 : Assigns Client Thread
4 : Input To Adjust Map
5 : The Map Displays To The User And Is Not Warped By Differing Resolutions


interaction Check In And Out
User Lifeline: Commuter
UI Lifeline: GUI
Server Lifeline: Server
Client Thread Lifeline: CT
Parking Lot Lifeline: PL Data
1 : Selects Parking Lot
2 : Requests Connection
3 : Assigns Client Thread
4 : Student Commuter Selects Appropriate Option To Check In Or Out
5 : Modifies Parking Lot Data
6 : The Map Accurately Displays To Student Commuter And Other Users This New Update In Real-Time


interaction Timer
UI Lifeline: GUI
Server Lifeline: Server
Client Thread Lifeline: CT
Parking Lot Lifeline: PL Data
User Lifeline: Commuter
1 : Selects Parking Lot
2 : Requests Connection
3 : Assigns Client Thread
4 : Student Checks Themselves Into A Parking Spot Successfully
5 : Modifies Parking Lot Data
6 : Student Selects An Estimate For How Long They Will Likely Remain In That Spot
7 : Modifies Parking Lot Data
8 : The Timer Displays To Other Users In Real-Time a Countdown
9 : Student Checks Out Of The Parking Spot By The End Of The Timer Countdown.
10 : Modifies Parking Lot Data
11 : The Spot Is Marked Empty, The Timer Also Disappears With This Action


interaction Analytics
UI Lifeline: GUI
Server Lifeline: Server
Client Thread Lifeline: CT
Parking Lot Lifeline: PL Data
User Lifeline: Commuter
1 : Selects Parking Lot
2 : Requests Connection
3 : Assigns Client Thread
4 : Student Checks Themselves Into A Parking Spot Successfully
5 : Modifies Parking Lot Data
6 : The Analytics Page Saves The Information Gathered In UC-4 And Displays It Visually