CSC 131
Software Requirement Specification

SRS

SACRAMENTO
STATE

JFELT Team
CSC 131 Section 02
Tuesday / Thursday
Instructor: Ahmed Salem, Ph.D.
Spring 2016

Table of Contents

1 - Introduction

1.1 - Purpose of this document

The purpose of this document is to give a detailed description of the Safe Rides Android application. It will explain the purpose, features of the system, what it will accomplish, the interfaces presented, and how it will respond to outside usages. This document is intended for use by the volunteers working at Safe Ride and for any future programmers modifying the application on Safe Ride's behalf.

1.2 - Scope of the Development Project

The Safe Rides Android App is a program that allows students and faculty of California State University, Sacramento (CSUS) to request a ride home from their current location during the program's hours of operation, free of charge. The Android app will be available to download from Google Play with instructions on how the process works both in the app and on the Safe Rides website.

A coordinator will be assigned during the hours of operation to control the requests that come in. They will be the sole holder of personal information at any given time in order to protect the privacy of its users. The coordinator will receive the request and assign a pick-up team to facilitate the pickup.

Since the app needs to communicate with the Server in order to receive the requests, an internet connection is required either through a data plan or WIFI access. Furthermore, the location services will have to be enabled in order for the GPS to provide the current location of the client if needed.

2 - General Description

2.1 - Glossary

Android - The operating system for the mobile app.
Client - Student/Faculty who uses the app.
Client Destination - The GPS coordinate of where the Client is requesting a ride. Although it should be their final destination for the night, there will be no way to determine if it is in fact the Client's residence.
Client Location - The GPS coordinates of the Client as tracked by the Driver on the map.
Coordinator - An ASI (Associated Students, Inc.) employee who is the sole person responsible for the Safe Rides program on that particular night.

Driver - A volunteer with the Safe Rides program who will be the one to drive and pick up the client and drive them home.
Driver Location - The GPS coordinates of the Driver as tracked by the Client on the map.
Navigator - A volunteer with the Safe Rides program who rides along with the driver, operates the app, and gives the driver directions to the Client and to their destination. The Driver/Navigator team will be referred to simply as Driver.
Client- A CSUS student or faculty member that uses the mobile app to request a ride home.
Ride - A one-per-night trip from the client’s current location to their home.
Smartphone - The Android phone of choice for the clients and the drivers to request rides and receive ride information respectively.
Server - The part of the system that the communicates between the different entities to provide information from the System.
System - The main part of the app where the information will be stored and distributed from.

2.2 - User Characteristics

Student/Faculty/Staff

- **Expectations**: To be picked up and dropped off safely when they are not in a good state to drive.
- **Requirements**: Having access to a smartphone that has or can download the app and go on the internet.
- **Necessary Experience**: Basic knowledge of how to use mobile app with GUI interface in a smartphone.

Driver

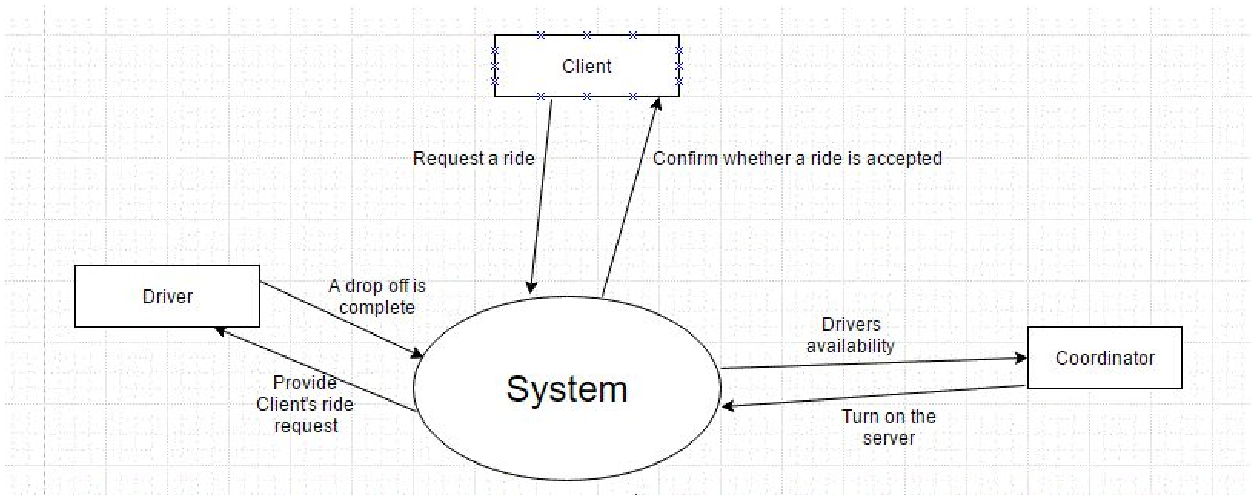
- **Expectations**: Receive ride information from the app whenever he is assigned to a client without the need to be in the room with the coordinator.
- **Requirements:** Have the app already installed on his smartphone and be registered on the server.
- **Necessary Experience**: Basic knowledge of how to use mobile apps and how to manage some of its features.

Coordinator

- **Expectations:** Being able to start and close the server. While the server is open, need to be able to receive ride requests and send the request to a designated driver or ask more information from the client.
- **Requirements:** To have received a basic tutorial on how to use the desktop application on the computer.

- **Necessary Experience**: Basic knowledge on how to log in a computer, run applications and use basic GUI components.

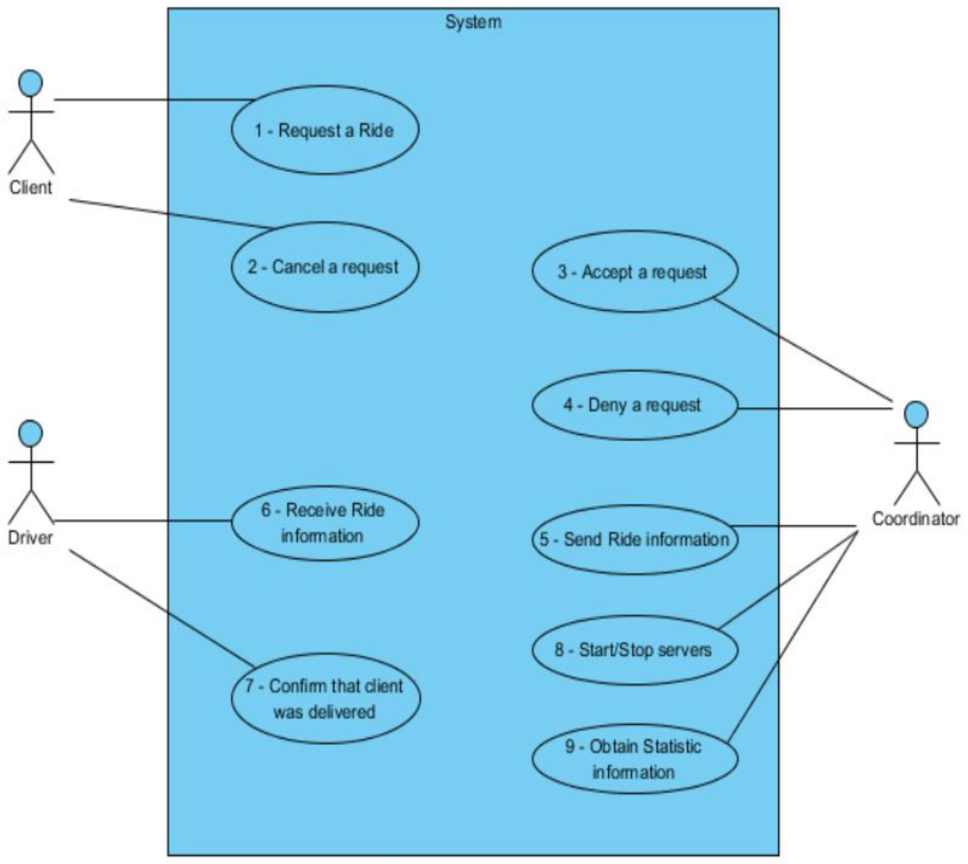
2.3 - Product Perspectives



Description
Client can submit a request for a ride and in return the system will confirm whether the ride request is a success or not
Driver gets client's ride information from the system and later returns confirmation whether the drop off is success
The system will let the coordinator know which driver is available and the coordinator will have to turn on the server to begin the service

3 - Object Oriented Analysis (OOA) - UML

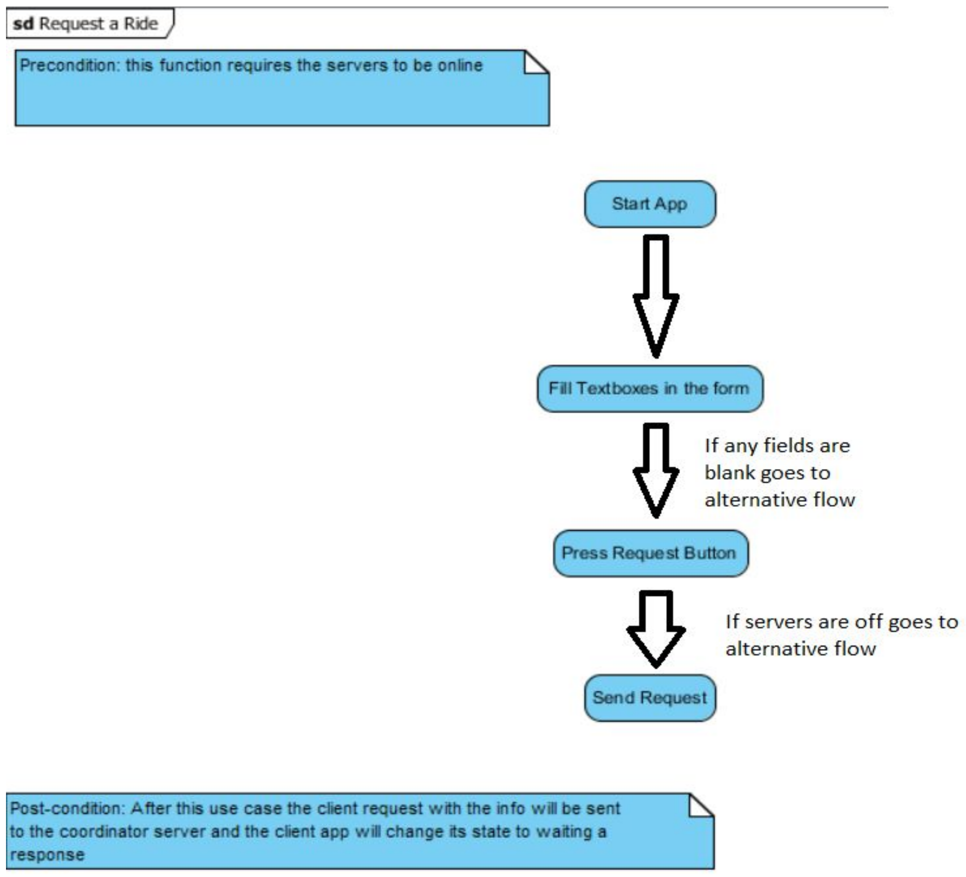
3.1 - Use Case Diagram



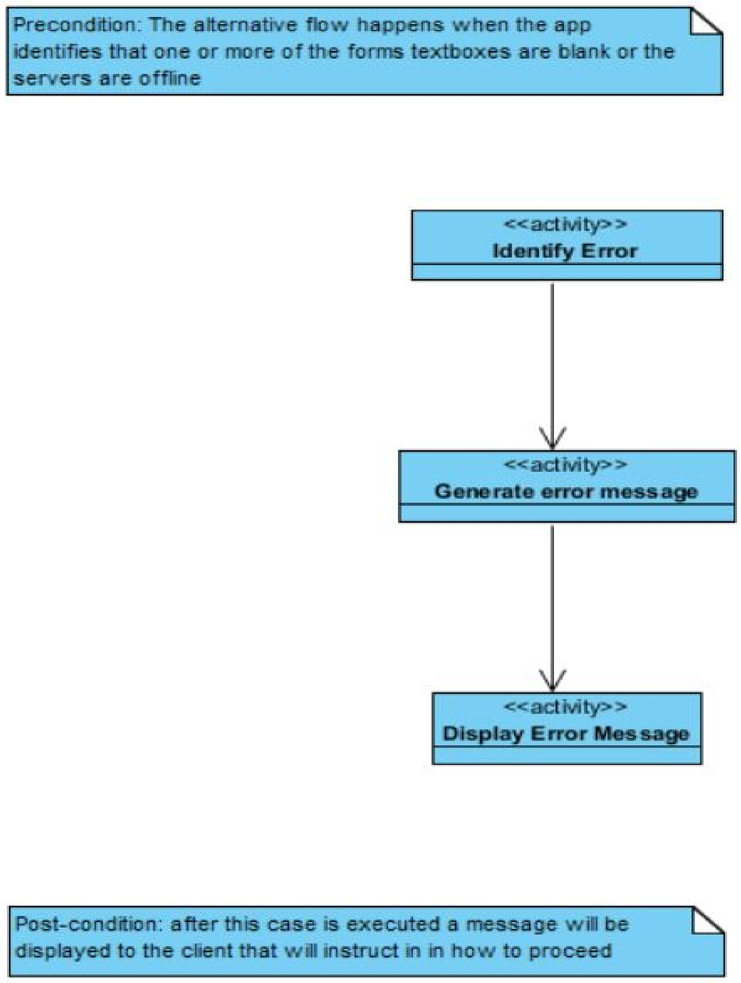
3.2 - Description of the Use Cases

Use Case 1 - Request a Ride

Name: Request a Ride
Number: UC1
Author: Felipe Izepe
Actors: Client
Overview: This uses the form in the app to send a ride request to the server with all the information needed
References: This requirement uses the case captures- FR1, FR5, FR7, FR8, FR16.
Related use cases: UC2, UC3, UC4, UC5, UC6
Typical Flow:

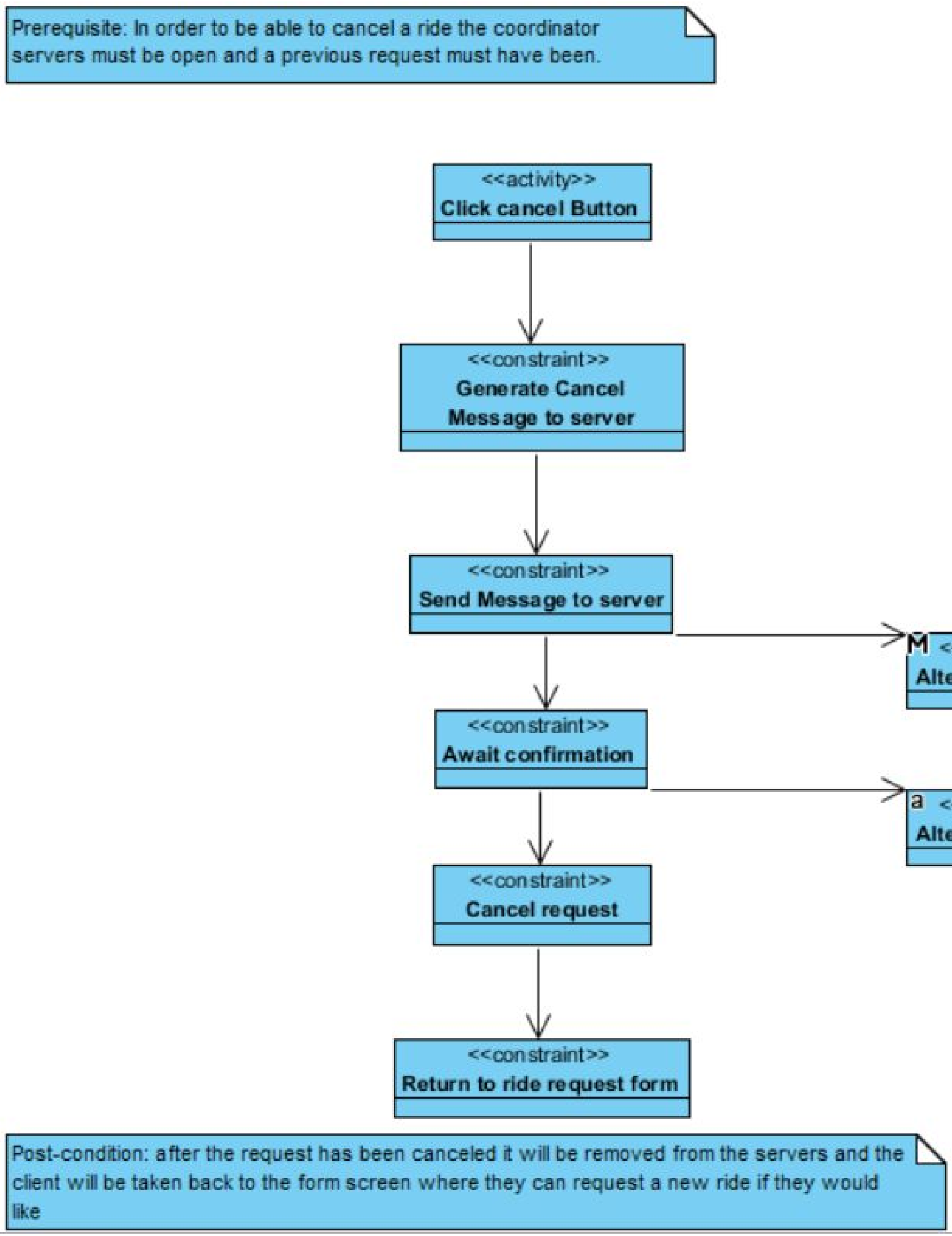

sd Request a Ride
Precondition: this function requires the servers to be online
Start App
Fill Textboxes in the form
If any fields are blank goes to alternative flow
Press Request Button
If servers are off goes to alternative flow
Send Request
Post-condition: After this use case the client request with the info will be sent to the coordinator server and the client app will change its state to waiting a response

Alternative Flow:

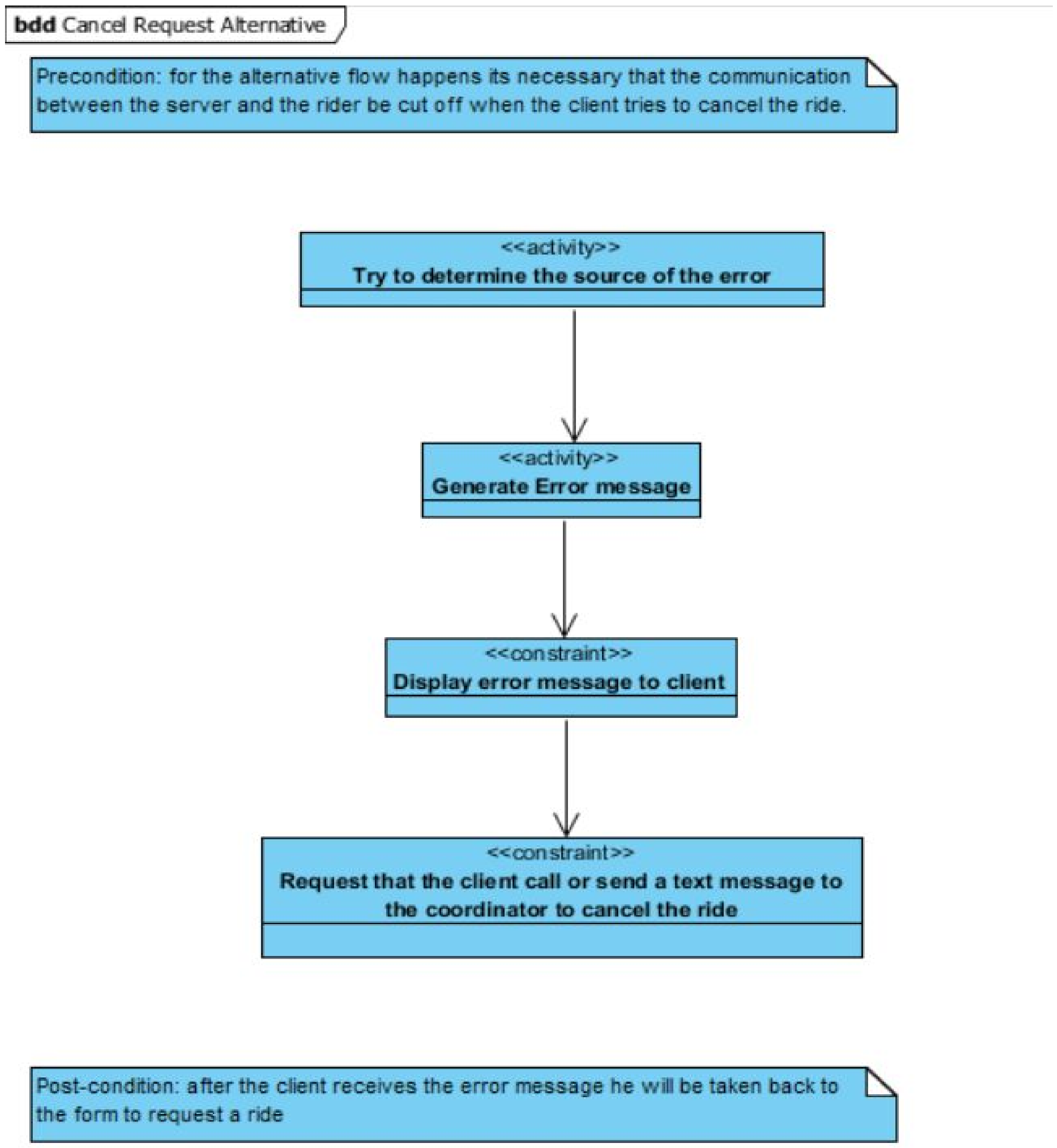

Precondition: The alternative flow happens when the app identifies that one or more of the forms textboxes are blank or the servers are offline
<<activity>> Identify Error
<<activity>> Generate error message
<<activity>> Display Error Message
Post-condition: after this case is executed a message will be displayed to the client that will instruct in in how to proceed

Use Case 2 - Cancel a Request

Name: Cancel a Request
Number: UC2
Author: Felipe Izepe
Actors: Client
Overview: This use case enables the client to cancel a request for a ride after it has been sent
References: This requirement uses the case captures- FR1, FR7.
Related use cases: UC1
Typical Flow:


Prerequisite: In order to be able to cancel a ride the coordinator servers must be open and a previous request must have been.
<<activity>>
Click cancel Button
<<constraint>>
Generate Cancel Message to server
<<constraint>>
Send Message to server
<<activity>>
Alternative Flow
<<constraint>>
Await confirmation
<<activity>>
Alternative Flow
<<constraint>>
Cancel request
<<constraint>>
Return to ride request form
Post-condition: after the request has been canceled it will be removed from the servers and the client will be taken back to the form screen where they can request a new ride if they would like

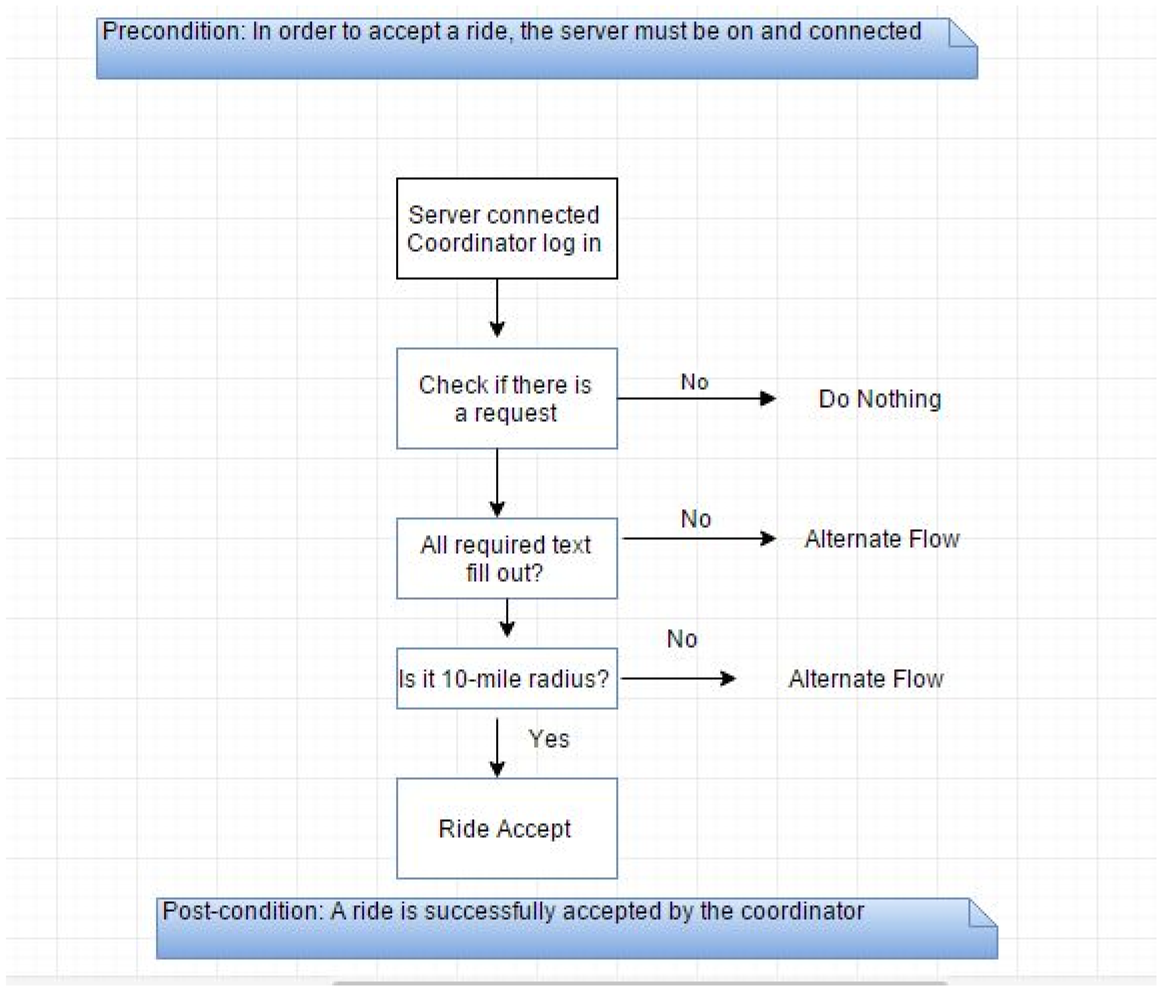
Alternative Flow:


bdd Cancel Request Alternative
Precondition: for the alternative flow happens its necessary that the communication between the server and the rider be cut off when the client tries to cancel the ride.
<<activity>>
Try to determine the source of the error
<<activity>>
Generate Error message
<<constraint>>
Display error message to client
<<constraint>>
Request that the client call or send a text message to the coordinator to cancel the ride
Post-condition: after the client receives the error message he will be taken back to the form to request a ride

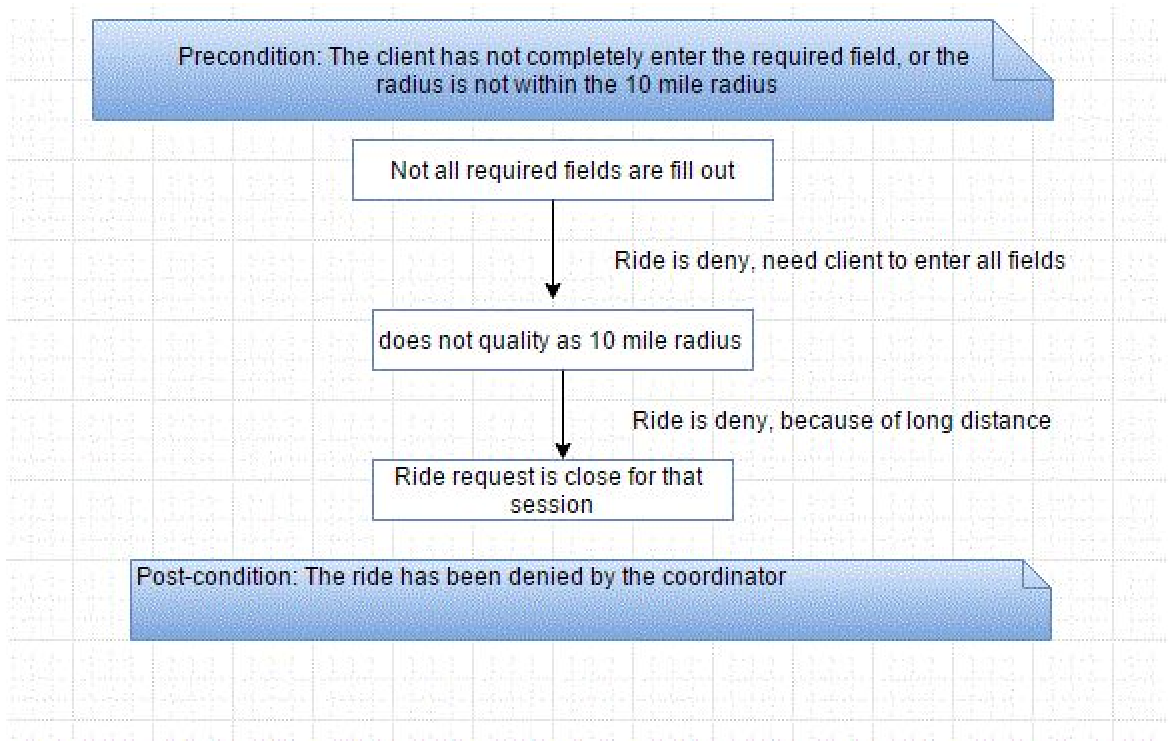
Use Case 3 - Accept a Request

Name: Accept a Request
Number: UC3
Author: Luong Phung
Actors: Coordinator

Overview: This use case enables the coordinator to accept a ride request from the client
References: This requirement uses the case captures - FR 5, FR 4
Related use cases: UC1
Typical Flow:


Precondition: In order to accept a ride, the server must be on and connected
Server connected Coordinator log in
Check if there is a request
No
Do Nothing
All required text fill out?
No
Alternate Flow
Is it 10-mile radius?
No
Alternate Flow
Yes
Ride Accept
Post-condition: A ride is successfully accepted by the coordinator

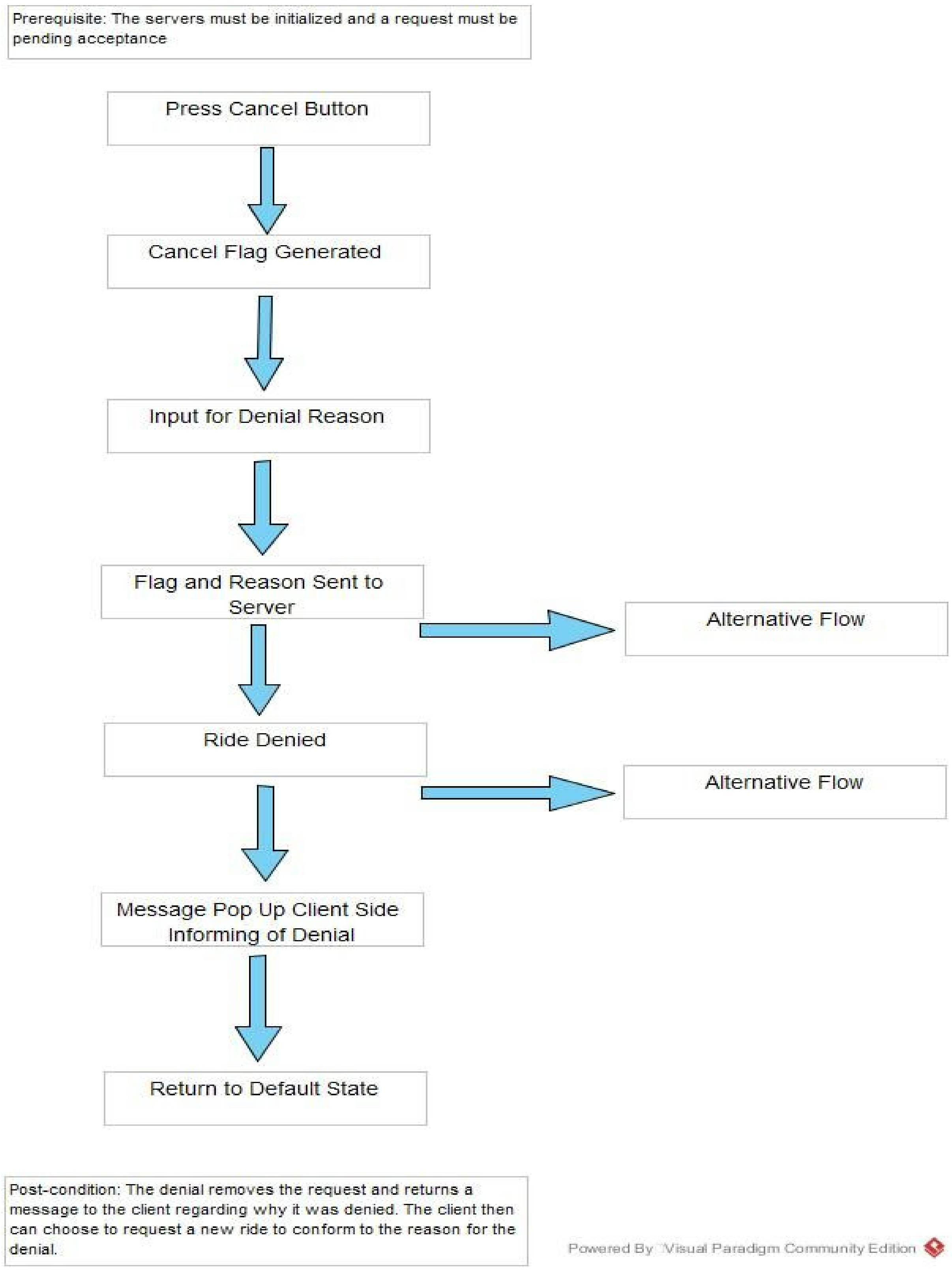
Alternate Flow:


Precondition: The client has not completely enter the required field, or the radius is not within the 10 mile radius
Not all required fields are fill out
Ride is deny, need client to enter all fields
does not quality as 10 mile radius
Ride is deny, because of long distance
Ride request is close for that session
Post-condition: The ride has been denied by the coordinator

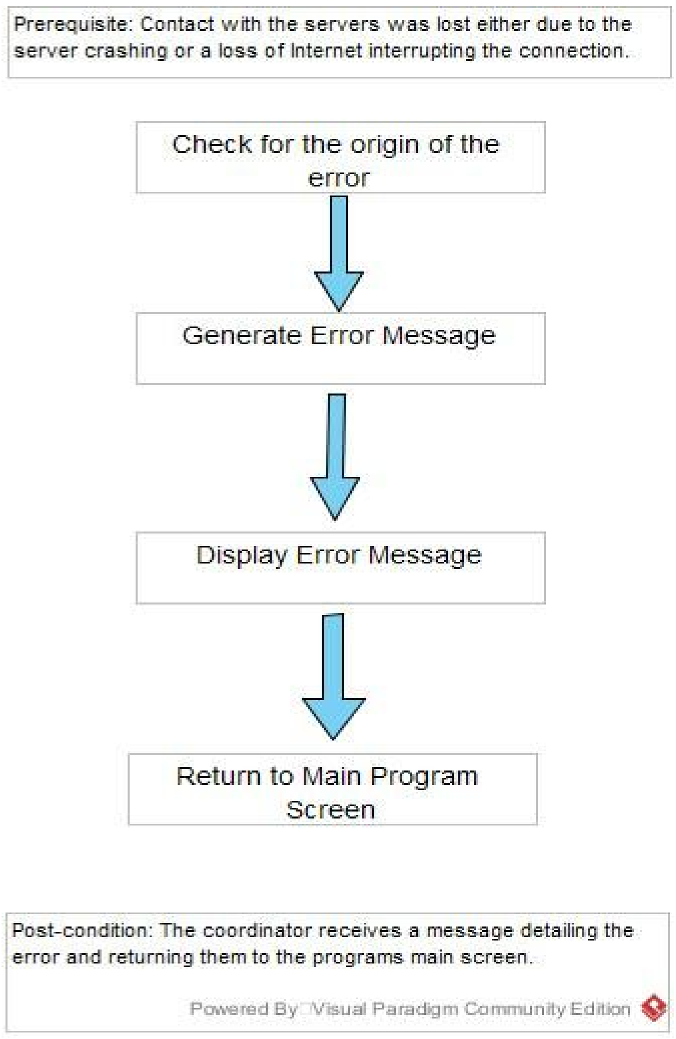
Use Case 4 - Deny a Request

Name: Deny a Request
Number: UC4
Author: Edgar Marroquin
Actors: Coordinator
Overview: This use case enables the coordinator to deny a ride from the client
References: This requirement uses the case captures- FR1, FR 7
Related use cases: UC1

Typical Flow:


Prerequisite: The servers must be initialized and a request must be pending acceptance
Press Cancel Button
Cancel Flag Generated
Input for Denial Reason
Flag and Reason Sent to Server
Alternative Flow
Ride Denied
Alternative Flow
Message Pop Up Client Side Informing of Denial
Return to Default State
Post-condition: The denial removes the request and returns a message to the client regarding why it was denied. The client then can choose to request a new ride to conform to the reason for the denial.
Powered By Visual Paradigm Community Edition

Alternate Flow:

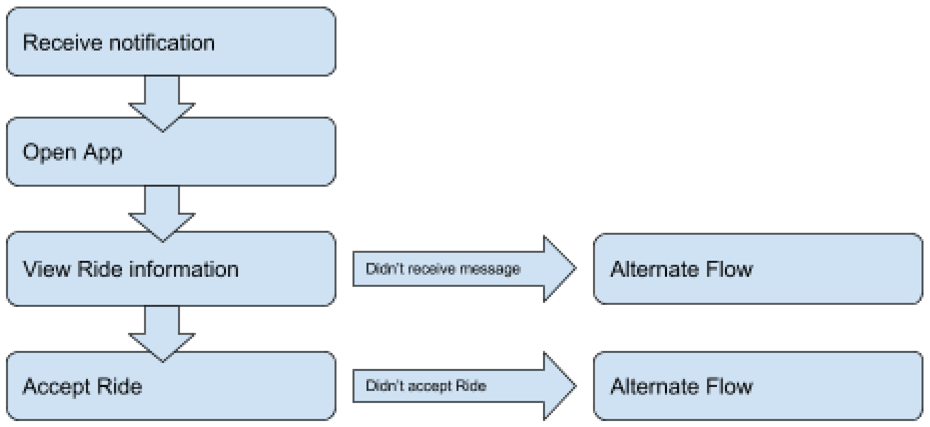

Prerequisite: Contact with the servers was lost either due to the server crashing or a loss of Internet interrupting the connection.
Check for the origin of the error
Generate Error Message
Display Error Message
Return to Main Program Screen
Post-condition: The coordinator receives a message detailing the error and returning them to the programs main screen.
Powered By Visual Paradigm Community Edition

Use Case 6 - Receive Ride Information

Name: Receive Ride Information
Number: UC6
Author: Juan Ruiz
Actors: Driver
Overview: This use case allows the driver to receive and accept client information so they can perform the pickup.
References: This requirement uses the case captures- FR1, FR3, FR10, FR11, FR16
Related use cases: UC5

Typical Flow:

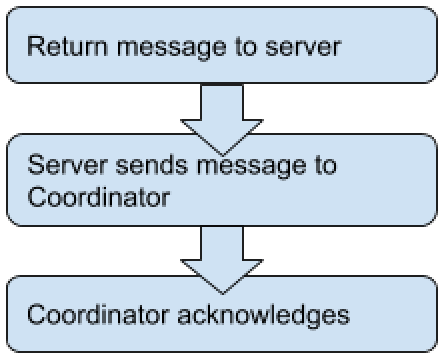
Precondition: This function requires that the server be online and that there should be a Ride pending. It is also required that there be drivers registered and available to receive Ride information.



Post-Condition: After the Ride information has been sent, the next driver in the queue will be on a wait status to receive the next Ride information.

Alternate Flow:

Precondition: This alternate flow requires that the server be on and that a Ride message was not received or that the driver did not accept the Ride.

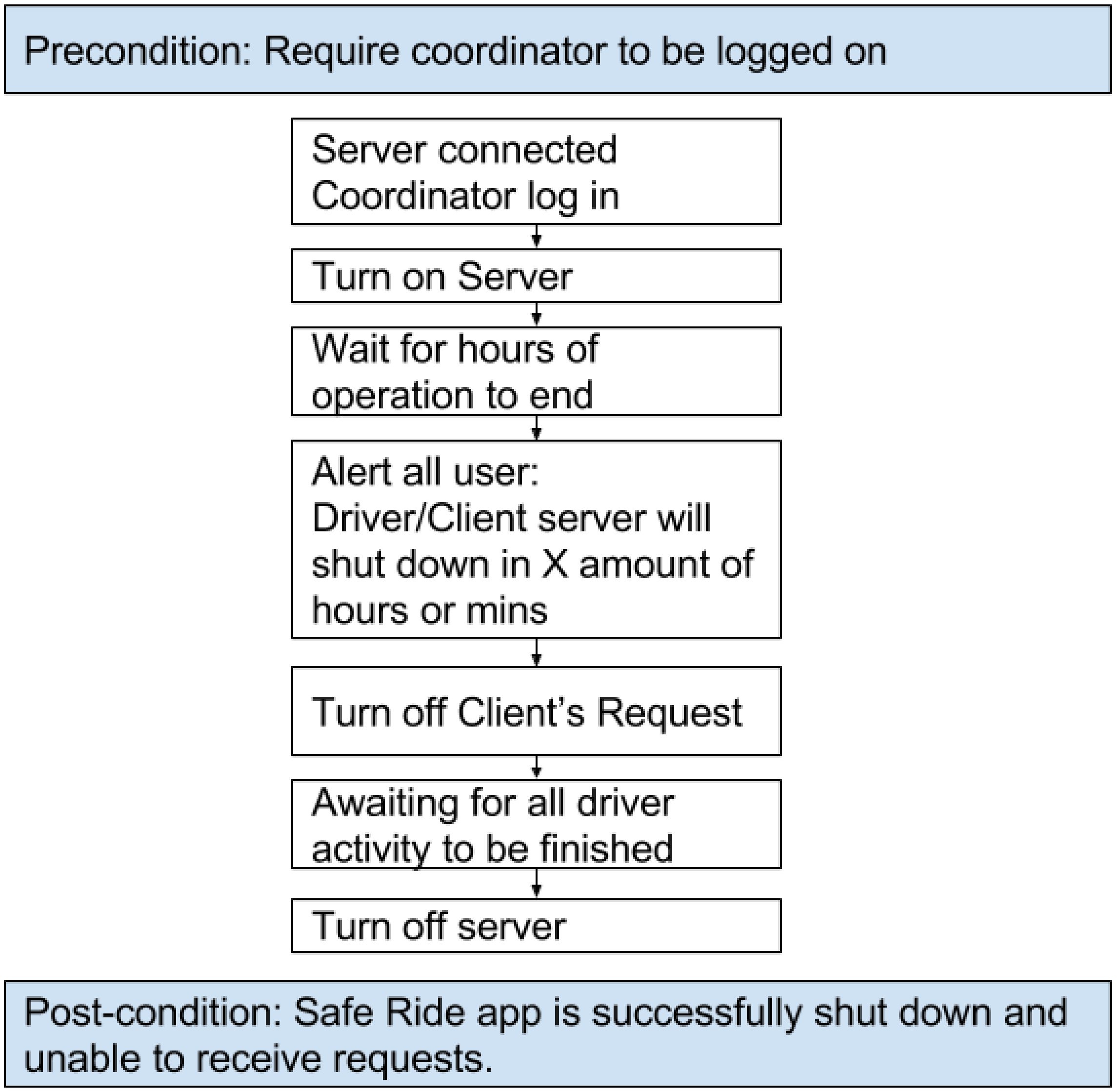


Post-Condition: Once the Coordinator acknowledges that the Ride still needs a Driver, they will be able to assign to a new Driver.

Use Case 8 - Start/Stop Server

Name: Start/Stop Server
Number: UC8
Author: Tony Liang
Actors: Coordinators
Overview: This use case allows the Coordinator to turn the server on and off.

References: This requirement uses the case captures- FR2.
Related use cases: N/A
Typical Flow:


Precondition: Require coordinator to be logged on
Server connected Coordinator log in
Turn on Server
Wait for hours of operation to end
Alert all user: Driver/Client server will shut down in X amount of hours or mins
Turn off Client's Request
Awaiting for all driver activity to be finished
Turn off server
Post-condition: Safe Ride app is successfully shut down and unable to receive requests.

Alternate flow: N/A

4 - System Functional Requirements

1. Communication Client-coordinator-Driver
2. App stop options
3. Driver registration
4. Client eligibility to service check
5. 10 mile radius check
6. Driver picture

7. API GUI design
8. Client need for multiple cars
9. Multiple Clients request
10. GPS tracking of Client location
11. GPS tracking of Driver location for client
12. Dealing with close Clients
13. Data tracking for statistics
14. Check student ID with university server
15. Cover Driver Milage
16. Have a Client photo for the Driver
17. Rating System

5 - General Constraints and Assumptions

This app will only work on android operating system and internet access is required. Some features need to have access to the phone's GPS location.

6 - User View of Product

Students

Upon opening SafeRide app for the first time, there is an option that allows them to switch to Student or Driver activity. If they pick Student then they are asked to input their personal info such as: name, ID, phone number, and Home Address and it may be saved on storage for an easier use. Other than that, Student are to enter information regarding their eligibility to qualify for the rides. Following that will be questions about their current information such as how many people are getting picked up, how many drop-off locations, their current location at and notes to the driver.
There will always be 3 buttons appearing below the screen: Share Location, Personal Info, and Confirmation.

***Share Location**:* students are to either their current location from the question where they at to or have an option that will get their coordinates shown to them that will be submitted later
***Personal Info**:* Enables the student to save info for later use.
Save button will save all of the new info.
Back button will go back to the main Student screen.
***Confirmation**:* bring them to a screen that show all the information that is about to be submitted to the server. There will also be the buttons Submit and Back.
Submit button will send all the information displayed on the screen to the server.
Back button will go back to the main Student screen.

Upon Submitting- SafeRide app will enter a Waiting State that will alert the user through notifications if the request is being granted or denied,when a driver is assigned and when driver is close to arriving at the pick up location. Both the denial of a request or the driver arriving will change this current state.

Driver/Navigator

Driver Login

Upon switching to Driver activity, the driver must enter their username and password provided by the Coordinator/ server.

Upon entering data for login user and password, the data is then sent to the server to check its validity.If there is a failure to authenticate the login and password the driver will be returned to the login screen where he can re-enter the information.

Upon successful login, the application will change to the main driver activity.

Driver Information Page

Upon entering Driver Main page for the first time the driver will be asked to enter his personal info and that data will be saved for later use. Info includes : name, license plate, picture, picture of the car, if a navigator is present, etc . After that they are directed to the Main Driver activity, If this is not their first time then they are directed Main Driver activity

Main Activity

The app will go to waiting state to receive information from the server. With 3 Buttons:
Finish Night, Change Info, and Map.

Finish Night**:** reports that they are done with tonight volunteering and will not take any more rides

Change Info**:** switches to personal info activity to edit their information.

Map**:** switches over to Map activity that will display driver's location in overhead view.

Confirmation Activity

Upon receiving information from the server, the driver will be alerted with a notification. It will then automatically switch to Confirmation Activity. Driver and the navigator will either choose to accept or refuse the ride. With 4 buttons: Accept, Refuse, P. Info, Map.

Accept**:** switches over to Client's View Activity

Refuse**:** brings the state back to Main activity

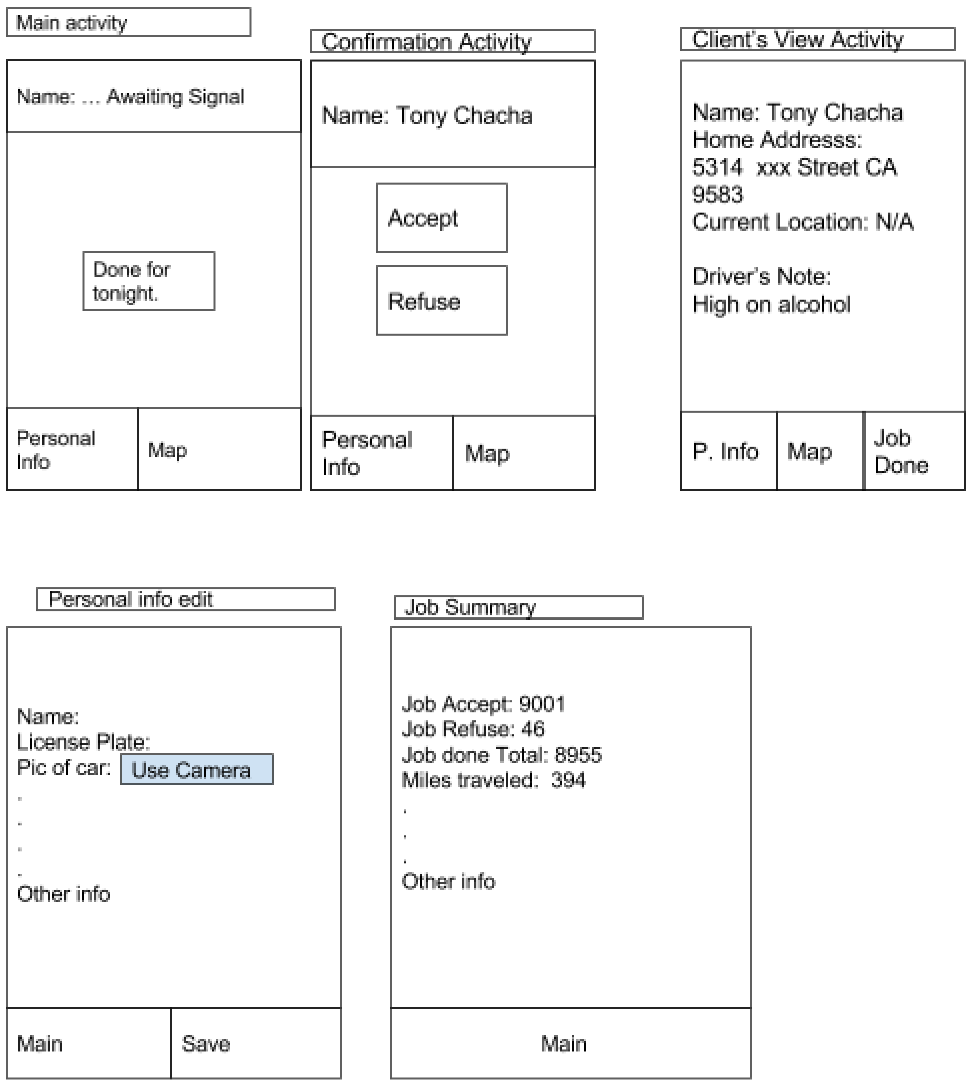
Personal info**:** switches over to Personal info activity

Map**:** tracks driver current location and shows the way to the pick up location.

Client's View Activity

In this activity, driver will see all the information that the client input to the server with 3 buttons: Personal Info, Map, and Client Delivered.

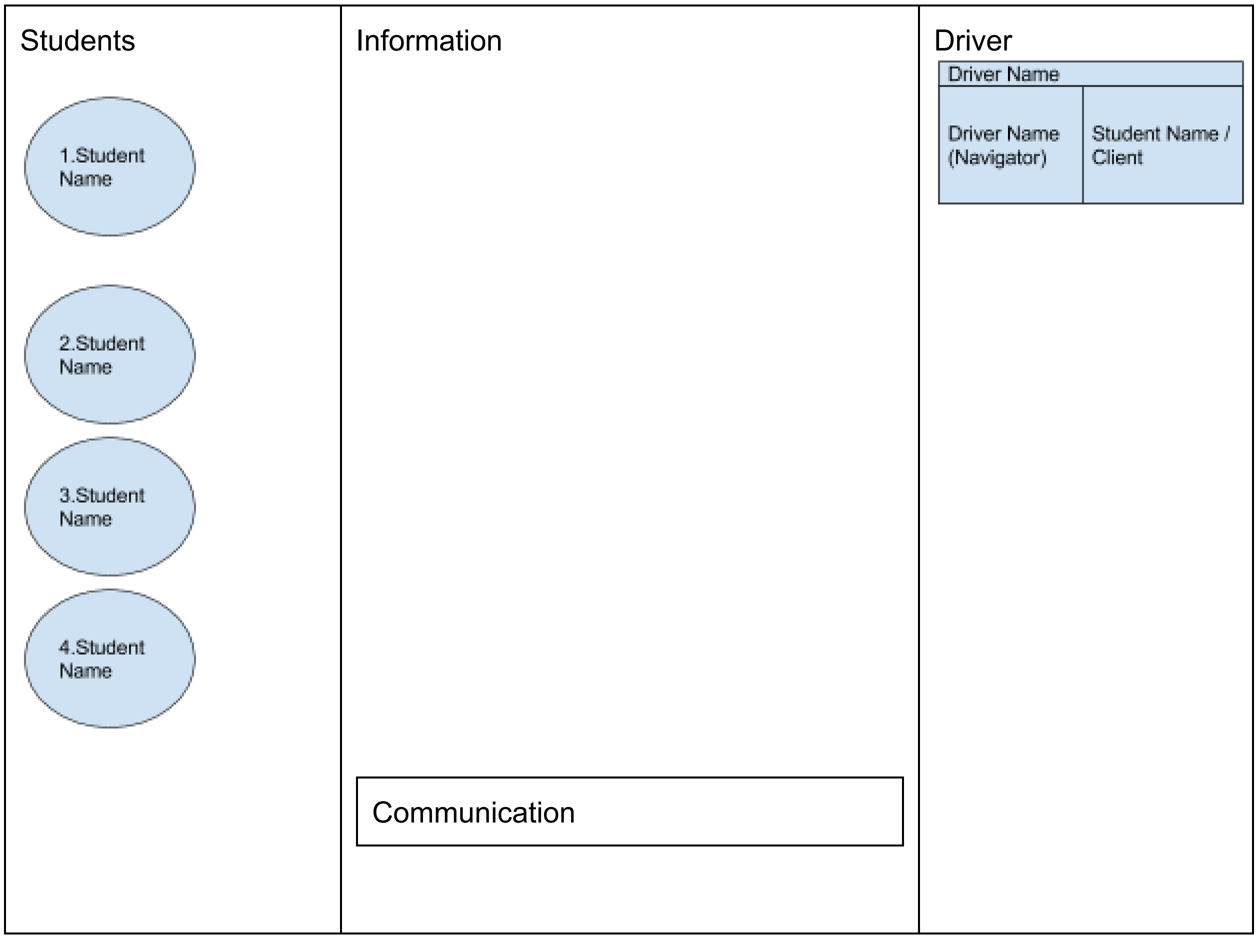
***Personal Info*:** shows clients basic information to the driver.
***Map*:** same operation as in Map at Confirmation Activity.
***Client Delivered*:** informs the coordinator that the client has been delivered and the milage covered. Furthermore, this will switch to Job done summary activity which summarize their entire night status.



Above all the rough sketches of driver's Interface.

Server

Upon executing the desktop application, the coordinator will be taken to the main screen.
Main Screen rough Sketches:



On the left side are displayed objects that represent the students that are created upon receiving the request from the app.Upon appearing there will be a alert in the student object to notifying the coordinator. To which the coordinator will have to click on the student object and manually grant or deny that person request base on the information received. In case of denial, the object will be terminated.
On the middle of the screen is information about the student or the driver currently selected.
On the right side, display all the drivers that are registered for that night. In order to assign a driver for a ride the coordinator must click and drag the student object on top of the drivers. If the driver denies the student assigned the coordinator will be notified and can choose to call the driver or chat with him in the communication box by clicking on his icon on the screen.

There will be button for settings and also button for turning on or off for the server

<u>7 - Specific Requirements</u>

7.1 - Interface Requirements

7.1.1 - User Interface

It is required for the client to have access to an android smartphone, whether from his or from another person, so that he can download and utilize the app. The smartphone needs to have the basic java libraries installed so that it can function properly.

7.1.2 - Hardware Interface

The smartphone needs to be able to connect to a internet provider, either wifi access or an internet provided by a company, to enable the app to send the information for the request to the administrator.

7.1.3 - Communications Interface

The communication between the student/faculty and the coordinator will be an underlying one. The client submits a request and receives a confirmation on whether or not the ride was accepted and also the approximate time the driver's arrival. There will be no direct communication between the two other than a phone call if necessary.

The drivers and the coordinator will have have a similar relationship where the driver will be able to signal that they have completed the trip. The driver will have to submit certain information before the trip completion message can be sent (i.e. odometer reading, current location, additional comments).

8 - Description of Functional Requirements

8.1 - Template

Functional Requirement 1: Communication Client-coordinator-Driver

- **Description:** The purpose of this requirement is to create the communication between the client, coordinator and the driver that will be the basis of the whole application.
- **Input:** Form on the client-side app with the information required for the ride. The information will be sent through an online connection in text form and shall contain the student ID, address of pick-up and drop offs, number of people and distance from the university. This input will be send to the coordinator.
- **Processing:** The coordinator receives the the information from the client and checks if it is acceptable, after he will send the info to the driver.
- **Output:** The driver will receive the ride information and will proceed to attend the client.

Functional Requirement 2: App stop options

- **Description:** Enables to coordinator to stop the app and disable the service and lets the client know the service is offline.
- **Input:** Manual input from the coordinator through a disconnect service button in the coordinator desktop.
- **Processing:** The program will deny all connection attempts and disconnect all user that may still be connected.
- **Output:** The client that tries to connect to the coordinator server will be unable to and will receive a service offline screen on his app.

Functional Requirement 3: Driver registration

- **Description:** Enables to coordinator to register drivers in the desktop application who will later be receiving ride assignments.
- **Input:** The driver or coordinator will fill an application form with the name, phone number and a photo of the driver and navigator if there is one and the car information, this information is text based with exception of the photo that will be treated as an .jpeg image.
- **Processing:** The application will receive this info and store it in file, the application will try to connect to the navigator to signal that the information is correct.
- **Output:**.The driver and navigator will be added to the system and will be able to receive ride requests from the coordinator.

Functional Requirement 4: Client eligibility to service check

- **Description:** Allows the information provided by the client to be checked for accuracy and completeness.
- **Input:** Information is entered by the client on the client side mobile app. The information includes: Name, Student ID, Pick up address, Drop off address, phone number.
- **Processing:** The program will flag any client requests so that the coordinator is able to make a decision whether to accept or decline.
- **Output:** Requests will be sent to the coordinator marked as either: incorrect information but proper phone number, completely incorrect information.

Functional Requirement 5: 10 mile radius check

- **Description:** The addresses provided are checked for accuracy and whether they fall within the predetermined 10 mile radius.
- **Input:** The addresses are processed to ensure accuracy and whether they fall within the limits of the program.

- **Processing:** The program will flag any requests outside of the 10 mile radius so that the coordinator can make the decision whether to accept or decline.
- **Output:** The client request is sent to the coordinator marked with either one or two of the addresses provided being outside the radius.

Functional Requirement 6: Driver Picture

- **Description:** Allows the driver to upload a current picture of the vehicle that will be used to pick up clients so that they are more easily recognized.
- **Input:** Manual input by the drivers while registering for the night.
- **Processing:** The photo will be stored with the driver information to be used upon receipt of a trip.
- **Output:** The client that is approved for pickup will receive the drivers picture with the approval message.

Functional Requirement 7: API GUI design

- **Description:** uses different Java API classes to design a user friendly interface for client and driver so they can interact with the system
- **Input:** Use Java API classes such as Graphics, JComponent, or Swing Components
- **Processing:** The components will be built by the smartphone and shown to the client
- **Output:** Visual output in the phone screen for the client

Functional Requirement 8: Clients need for multiple cars

- **Description:** In case a client has more than 3 people with him that also need a ride there would be a need to send multiple drivers
- **Input:** Number of people that also need a ride
- **Processing:** The request description will be analysed and multiple driver will be selected
- **Output:** The client will have multiple drivers to pick them up

Functional Requirement 9: Multiple clients request

- **Description:** allows more than one clients to request a ride from SafeRide program at the same time without conflict
- **Input:** each client still has to enter basic information to request a ride
- **Processing:** The server will process each request and add the clients to a queue
- **Output:** Queue with all the clients that is able to be accessed by the coordinator

Functional Requirement 10: GPS Tracking of Client Location

- **Description:** Allows the client to make use of a GPS device in their smartphone to record and utilize their current location for use in filling out a pick up request.
- **Input:** The latitude and longitude of the client as specified by their current GPS location

- **Processing:** The GPS coordinates taken from this action will be automatically inputted to the ride request form that will be sent to the coordinator
- **Output:** The latitude and longitude will show up in text form in the cell designated for the pick up address.

Functional Requirement 11: GPS Tracking of Driver Location for Client

- **Description:** Allows the client to make use of a GPS device in the navigator's smartphone to record and utilize their current location for the client to see.
- **Input:** The latitude and longitude of the navigator as specified by their current GPS location
- **Processing:** The GPS coordinates taken from this action will be sent to the client and be viewable in a map application to see a visual representation of where their pick up is currently at the time.
- **Output:** The latitude and longitude will show up in a map application of the client.

Functional Requirement 12: Dealing with close clients

- **Description:** In the event of a busy night, allow a driver who has finished a dropoff to immediately receive a pick up request from the coordinator for a client in close proximity to the driver's current location.
- **Input:** The driver's current GPS location and the client's current GPS location.
- **Processing:** The location of the driver after a drop off is utilized along with the possible client's location to determine the distance the two are from each other (i.e. ~2 miles) so that the coordinator can decide to immediately give the driver the client without returning to base.
- **Output:** A marker of both the drivers and client's location with a numerical distance the two are from each other.

Functional Requirement 13: Data tracking for statistics

- **Description:** Allows for data to be collected and saved at the end of each night for use by the administration of Safe Ride to determine how busy the night was, the popularity in usage of the application, and distance travelled by drivers for gas compensation.
- **Input:** A collection of each driver's data for the night including: Clients picked up, a counter for what method used to request a ride, total miles travelled, a general start and end location for all trips, and how many hours volunteered for the night.
- **Processing:** Each collected form of data will be added on to the previous as the night goes on until the service shuts down for the night. The data will then be sent to be made into a report for the night.
- **Output:** A list of each driver's number of pickups, total miles travelled, and how many hours they volunteered for. At the end a tally will be shown for the total number of pickups for the night, a counter for the total number of requests that came from the

application, a call in, or a text in, the combined total of miles driven that night, and general locations for all pickup and dropoff locations.

Functional Requirement 14: Check student ID with university server

- **Description:** allows the client to sign in to the android app so they can use features such as request for ride
- **Input:** The client will enter their Sac State ID number to login the android app
- **Processing:** The university server will validate the client's entered password against the actual client ID store by the university
- **Outcome:** will confirm whether the client successful verify their ID number in order to use the android app

Functional Requirement 15: Cover Driver Milage

- **Description:** keep track of driver's milage from the minutes they are working to the end of their shift to reimburse them for the milage
- **Input:** Manual input based on the reading of their car's milage from the beginning to the end of their shift
- **Processing:** Based on the driver's distance from the client and drop off point the milage will be added throughout the night
- **Output:** The driver will be reimburse by a certain amount of dollar for the mileage

Functional Requirement 16: Have a Client photo for the Driver

- **Description:** The Client photo is used to assist the driver in identifying the Client.
- **Input:** The Client submits a photo of themselves when requesting a Ride.
- **Processing:** The photo will be stored alongside the other Client information to be used when needed.
- **Output:** The Driver will receive the photo of the Client when they receive the Ride information.

Functional Requirement 17: Rating System

- **Description:** let the client write a review or feedback about the night they receive the service
- **Input:** allows client to give feedback to the service by means of writing within the next couple of days by submitting the feedback
- **Processing:** The system can send out message remind client to submit feedback if they want and their feedback will be display on the android app for new client
- **Output:** A rate for each driver that the coordinator will be able to view

8.2 - Data Dictionary

Item #	Name	Type	Size (bytes)	Occurrence	Description
1	CFName	string	15	Client, Server, Driver	Client’s First name
2	CLName	string	15	Client, Server, Driver	Client’s Last name
3	CPNumber	int	9	Client, Server, Driver	Client’s phone number
4.	CLLat	float	9	Client, Server, Driver	Client’s Latitude Location
5	CLLong	float	10	Client, Server, Driver	Client’s Longitude Location
6.	CSID	int	9	Client,Server, Driver	Client’s Student ID
7.	CAPeople	int	1	Client, Server, Driver	Amount of people for the ride
8.	CNTDriver	string	400	Client, Server, Driver	Client notes to driver
9.	CCLocation	String	200	Client, Server, Driver	Clients pick up Address
9.	CDrop	String	200	Client, Server, Driver	Clients drop off Address
10.	DLPNum	String	15	Client, Server, Driver	Driver Liscense Plate number
11.	DLU	string	20	Driver, Server	Driver Login User Name info
12.	DLP	string	20	Driver, Server	Driver Login Password info

9 - Non-Functional Requirements

User Interface should be clean and easy to use for the client, but not required to be perfect as long as features are grouped in the right place. Security means a lot for this project since we have to keep confidential information of client’s name and other data in a safe environment and use appropriately. Performance and reliability are not too much of a certain as a starter with only 30 clients per night of usage for this project

10 - Special Remarks and Comments

None

11 - References or Resources Used

- Google Drive
- Google Docs
- Visual Paradigm
- Android Studio
- SRS Deliverables #2 Handout
- Class Website: http://csc-se.wix.com/csc131/

12 - Team Member’s Roles and Approval

12.1 - Team Member’s Roles in Completing the SRS

Felipe Izepe: sections 2.2, 7.1.1, 7.1.2, FR1, FR2, FR3, 8.1, UC1, UC2, Overall review

Tony Liang: section 6, FR14, FR 15, FR 17 , 8.2, UC8

Edgar Marroquin: sections 1.1, FR 10-13, UC4, 11

Luong Phung: sections 5, 9, 2.3, FR 7, FR 8, FR 9, FR 16, UC3, Overall review

Juan Ruiz: sections 1.2, 2.1, 7.1.3, FR 4-6, UC6, 12.1, 12.2

12.2 - Team Member’s Signatures and Date Signed

Felipe Izepe / Date	Tony Liang / Date
Edgar Marroquin / Date	Luong Phung / Date
Juan Ruiz / Date	