MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

School bus sharing for near-by students

Group 4 - IS	
Group members	Nguyễn Việt Hùng – SE62279 (Leader) Thái Hiếu Trung – SE62768 Tống Văn Giang– SE62256 Ngô Thế Vinh – SE61840
Supervisor	Nguyễn Anh Khoa
Ext. Supervisor	N/A
Project Code	SCB

– Ho Chi Minh City, 7 January, 2019 –

TRƯỜNG ĐẠI HỌC FPT

CAPSTONE PROJECT REGISTER

Class: Duration time: from …./…/…. To …/… /…..

(*) Profession: <Software Engineer> Specialty: <ES> ☐ <IS> ☐ <JS> ☐

(*) Kinds of person make registers: Lecturer ☐ Students ☐

1. Register information for supervisor (if have)

	Full name	Phone	E-Mail	Title
Supervisor 1	Nguyen Anh Khoa		Khoa.nguyen@saigontechnology.com	Mr.

2. Register information for students (if have)

	Full name	Student code	Phone	E-mail	Role in Group
Student 1					
Student 2					
Student 3					
Student 4					

3. Register content of Capstone Project

(*) 3.1. Capstone Project name:

English: School bus sharing for near-by students

Vietnamese: Ứng dụng đưa đón học sinh ở gần nhà nhau

Abbreviation: **Schoolbus**

- **Context**:

- Many families have car today and they take their kids to school then pick up back to home every day. Many kids live near by each other often go to the same school.

- Building the system provides following services:

- Allow bus driver (a parent and also service supplier) register offer service with information: available time, available sheets, starting location, destination, children age range.

- Allow parents to register buy service with information: quantity of children, pickup location, pick-up time, drop time, specific days that parents want the bus driver pick-up children
- The system should auto map supply and demand details to produce:
    - Journey details for bus driver
    - Journey details for parents
    - Should only limit within 1km radius to make sure quality of service

(*) 3.2. Main proposal content (including result and product)

a) Theory and practice (document):

- Student should apply the software development process and the UML.
- Software artifacts include User Requirement, Software Requirement Specification,
- Architecture Design, Detail Design, System Implementation and Testing Document,
- Installation Guide, sources code, and deployable software packages.
- 3 tiers should be applied.
- Server side technique:
    - Database design, OOA, OOD, OOP, MVC, Java or .Net technology, …
- Client side technique:
    - HTML5, CSS, JavaScript, jQuery, Ajax, Android, Swift, ...
- Communication technique:
    - Exchange information and transfer data in effective in networks, communicating protocol between mobile devices...
- Research
    - Mobile development; Android, iOS, Hybrid frameworks (React Native, Flutter)
    - Algorithms about finding matching between supply and demand details.

b) Program:

- Web application: for site admin
    - Mange master data
    - Track status and progress of all suppliers and buyers
- API for mobile applications
- Mobile Applications
    - One for service supplier (bus driver)
    - One for buyers

c) Other products:

- All of management functions of the system must be implemented to support the operating system in best.
- Papers.

4. Other comment (propose all relative thing if have)

N/A

………………., date …./…../…………

Supervisor (If have)	**On behalf of Registers**
(Sign and full name)	*(Sign and full name)*

Table of contents

A. Introduction

1. Project Information

- Project name: **School bus sharing for near-by students**
- Project Code: **SCB**
- Product Type: **Mobile Application**
- Start Date: **January 7th, 2019**
- End Date: **May 6th, 2019**

2. Introduction

Nowadays, life of married couples is very busy, especially when they have young children. In the morning, the wife, or the husband has to take their children to school because kids cannot go to school by themselves. It would be really inconvenient when parents get their kids to school, then go to work, because not every parent has enough time in the morning, due to rush hours, or some other personal reasons.

In this document, we introduce a solution for parents: a mobile application called School Bus. School Bus is a powerful tool to help parents not worry about shuttling their kids to school every day anymore. Kids would be picked up to school and taken back home on time and safely. This document also describes our working process in 4 months including our perspectives on the system, component designs and detailed core workflow.

3. Current Situation

The busy life does not allow parents to take care of their kids properly. One issue of these is that fathers and mothers have to think of many ways to shuttle their kids to school daily regardless of difficult situations such as traffic jam, inflexible working time, etc.

Many temporary approaches have been given to solve the problem, parents can take the kids to school very early (which wastes time of whole families), or rent taxi as well as Grab to take their responsibilities (which could pose a lot of risks from dealing with a strange driver every day without any permanent contract).

4. Problem Definition

Following are the disadvantages of the current situation:

- Parents are very busy that they could not take their kids to school in the morning and pick up their kids in the afternoon.
- Parents’ time for work and kids’ time for school are not compatible.
- The ways to kids’ schools and way to parents’ workplace are opposite.
- Hard to find a reliable rental motorbikers in neighborhood or Grab, GoViet, etc.

Above problems may have negative impact on both parents and kids' working days. Kids could be late for school, parents could be late for work because of traffic jam or some unexpected situations caused by unreliable rental motorbikers.

5. Proposed Solution

Our proposed solution is to build a system named "School Bus" to resolve the current problems. With this application, parents wouldn't be worried anymore about taking their kids in the morning or taking them home after school. Another parent with a car would register to be a driver. This driver would take nearby kids to their school and pick them up after class following an arranged schedule based on the service that their parents register before.

School Bus system includes one administration web-based application along with two mobile applications, one for service providers known as drivers, and one for service consumers known as customers, with the functions as follow.

5.1 Feature functions

- Mobile application for drivers
  - Service registration: a parent could register to be a driver. In order to do that, he or she must provide basic information and meet service requirements, including children's information and schedule.
  - Considering the contract: the driver would be notified when he/she is chosen by a customer. Then, they could choose to accept or reject the contract agreement.
  - Journey's planning: everyday, the driver would receive the detailed journey and notifications for picking up customer's kids, taking them to school then taking them back after class.
- Mobile application for customers
  - Requirement registration: parents could register a requirement to use service, he or she must provide basic information and service requirements, including children's information and schedule.
  - Choose the appropriate driver: customers are able to choose a suitable driver from a provided list of available options. Then, there will be notification when their contract agreements are approved by the driver.
  - Journey's notification: everyday, customers would receive their driver's detailed journey and notifications about the time when the driver comes in the morning, comes back again in the afternoon, and when the driver arrives at school.

5.2 Advantages and disadvantages

The advantages and disadvantages of proposed solution:

- Advantages
  - Increased reliability of drivers because they're parents, too and their profiles are explicit.

    - Drivers have an explicit visual route shown on the map for every day journey.
    - The route is calculated to be shortest and most appropriate for all pick-up locations and the drivers.
    - Parents could track the position of their drivers.
    - Parents could find the most suitable drivers for their children.
    - Parents could always receive notifications about their drivers and their kids' location.
- Disadvantages
    - System only supports limited radius between pick-up locations to ensure the quality of service.
    - System only supports contracts in which kids are from same school.
    - System only allows drivers to register one school for one service.

6. Functional Requirements

Functional requirements of the system are listed as following:

6.1 Administrator

- Administrator could manage all user's accounts.
- Administrator could verify driver's registrations.
- Administrator could configure all global specifications for the whole system.
- Administrator could review statistics and make throughout reports of whole system's activities by date.

6.2 System

- System could push notifications to user.
- System could find matching customers and drivers.
- System could calculate money.
- System could process and find the most appropriate route for driver in each journey.

6.3 Driver (Service Provider)

- Driver could register account.
- Driver could login.
- Driver could manage profile.
- Driver could register services.
- Driver could manage his/her services.
- Driver could view notifications.
- Driver could accept or reject contract agreement requests.
- Driver could view his/her contract details.
- Driver could view daily journey's detail and its route explicitly and visually on a map.
- Driver could have bill information after each trip.
- Driver could mark if the kids are picked up or not.

- Driver could call the customer.
- Driver could cancel some trips.
- Driver could cancel contracts.

6.4 Customer (Service Consumer)

- Customer could register account.
- Customer could login.
- Customer could manage profile.
- Customer could register requirements.
- Customer could manage his/her requirements.
- Customer could view notifications.
- Customer could view appropriate drivers then decide to choose or not.
- Customer could view his/her contract details.
- Customer could view the journey's route of their drivers, explicitly and visually on a map.
- Customer could have bill information after each trip.
- Customer could call the driver.
- Customer could cancel some trips.
- Customer could cancel contracts.
- Customer could extend their contracts.

7. Roles & Responsibility

No	Full Name	Role	Position	Contact
1	Nguyễn Anh Khoa	Project Manager	Supervisor	khoa.nguyen@saigontechnology.com
2	Nguyễn Việt Hùng	Developer	Leader	nvhungkt1997@gmail.com
3	Thái Hiếu Trung	Developer	Member	trungthaihieu93@gmail.com
4	Tống Văn Giang	Developer	Member	giangtvse62256@fpt.edu.vn
5	Ngô Thế Vinh	Developer	Member	Vinhntse61840@fpt.edu.vn

B. Software Project Management Plan

1. Problem Organization

1.1 Software Process Model

This project is developed using Scrum framework – a part of the Agile software development for the following reasons:

- The software requirements are not clear enough for us in the beginning, therefore, we have to adapt to new knowledges, new technical challenges and changes in requirements. Scrum is suitable for us to handle these challenges.
- There is no hierarchy in team, which allows everyone to feel free to share and cooperate better.
- The project is complicated and we have many plans in the future, so working in iterative sprints enables us to process everything step by step.

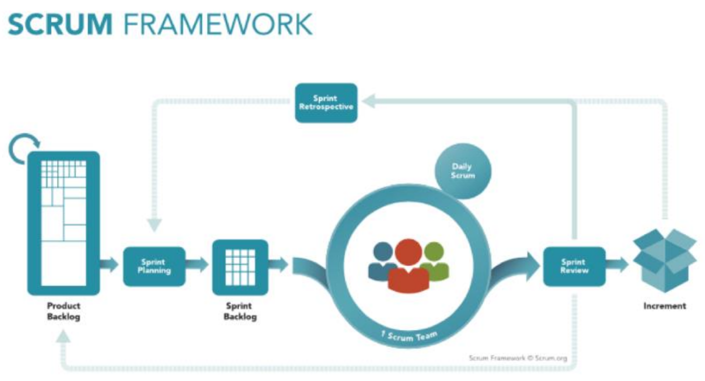


Figure 1 - Scrum Framework

1.2 Roles and responsibilities

No	Full name	Role in Group	Responsibilities
1	Nguyễn Anh Khoa	Product Owner	- Specify user requirement - Control the development process - Give out technique and business analysis support
2	Nguyễn Việt Hùng	Scrum Master	- Managing process - Designing database - Clarifying requirements - Prepare documents - GUI Design - Create test plan - Coding - Testing

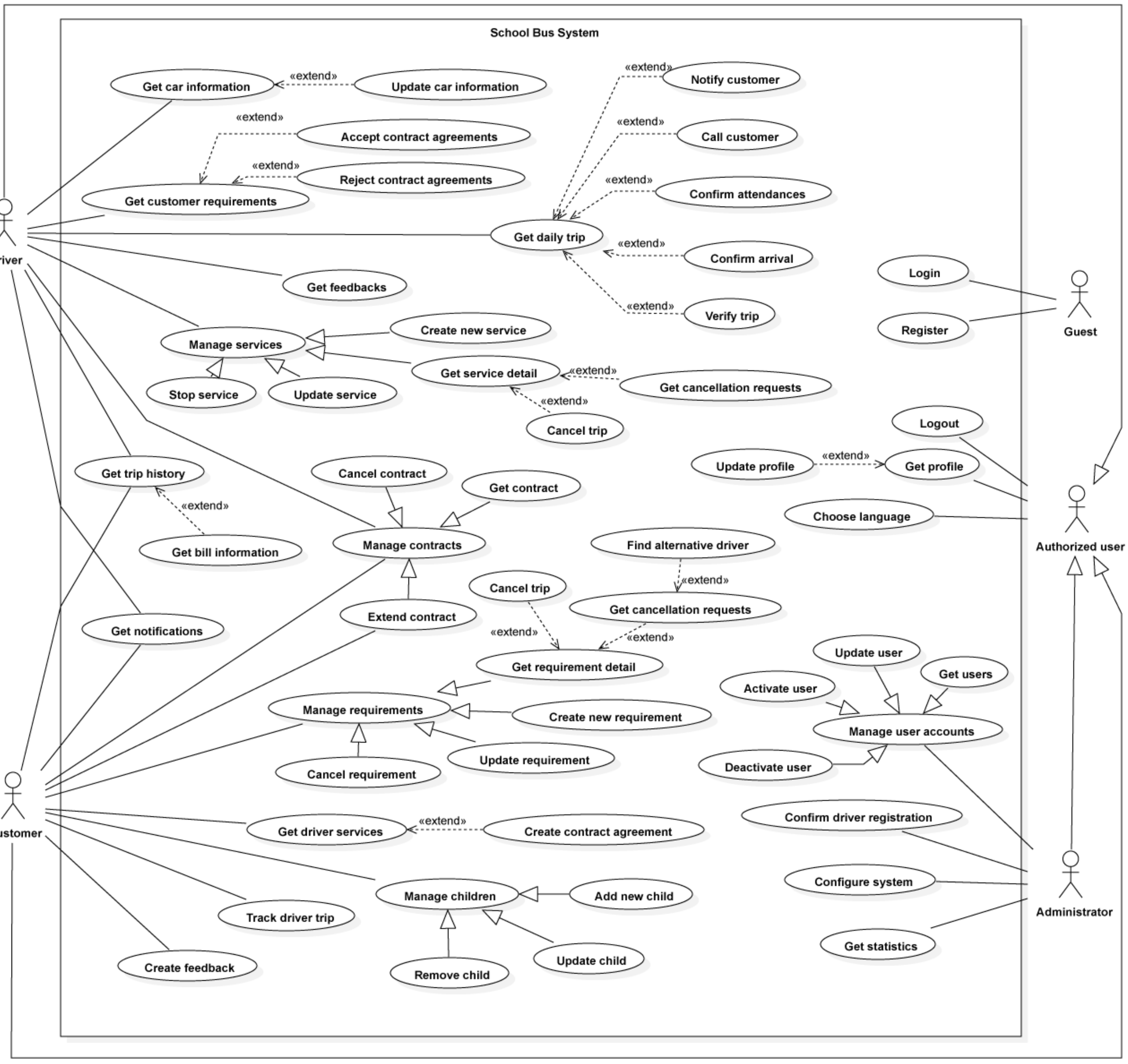
			- Arrange Meeting - Risk Management
3	Thái Hiếu Trung	Scrum team member	- Designing database - Clarifying requirements - Prepare documents - GUI Design - Create test plan - Coding - Testing
4	Ngô Thế Vinh	Scrum team member	- Designing database - Clarifying requirements - Prepare documents - GUI Design - Create test plan - Coding - Testing
5	Tống Văn Giang	Scrum team member	- Designing database - Clarifying requirements - Prepare documents - GUI Design - Create test plan - Coding - Testing

Table - Roles and responsibilities

C. Software Requirement Specification

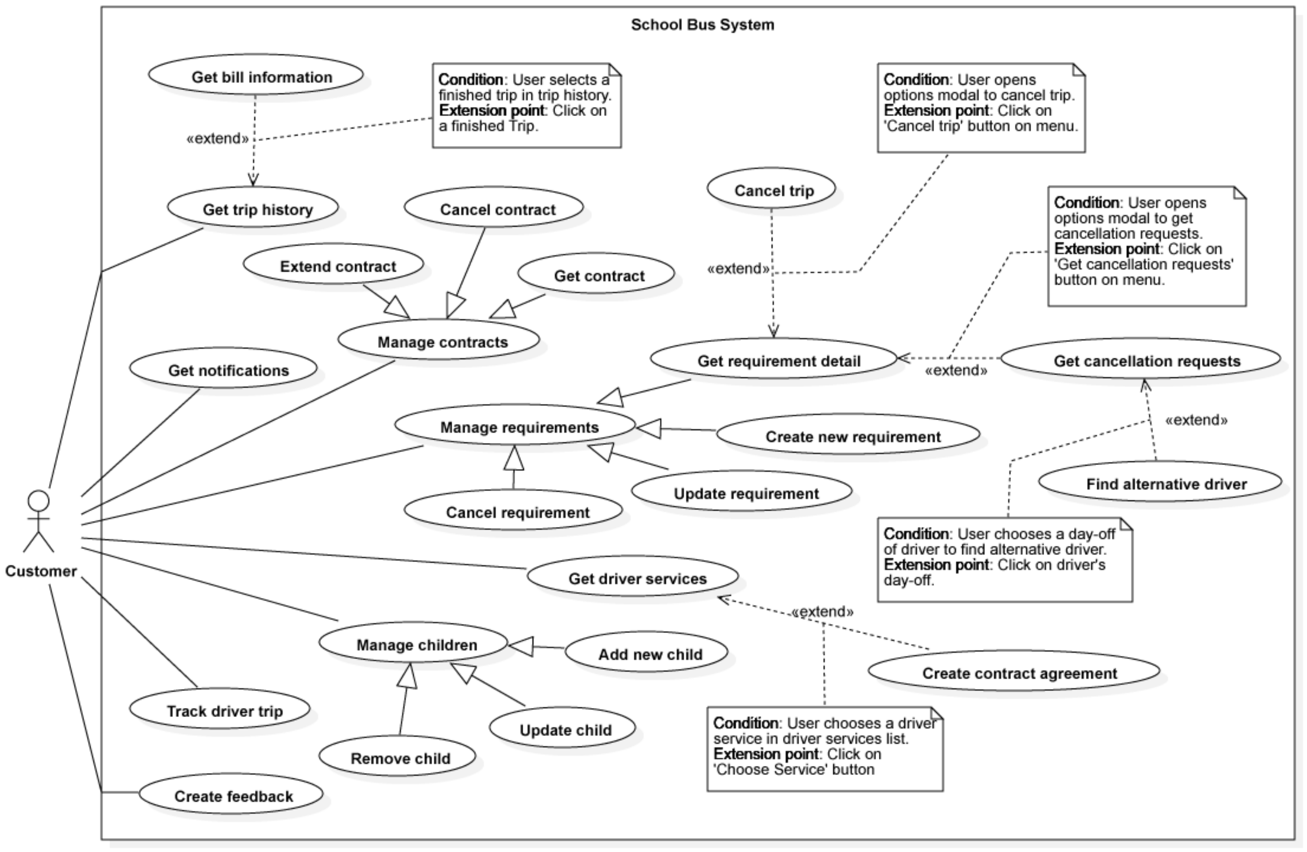
1. System Requirement Specification

1.1 System Overview Use Case



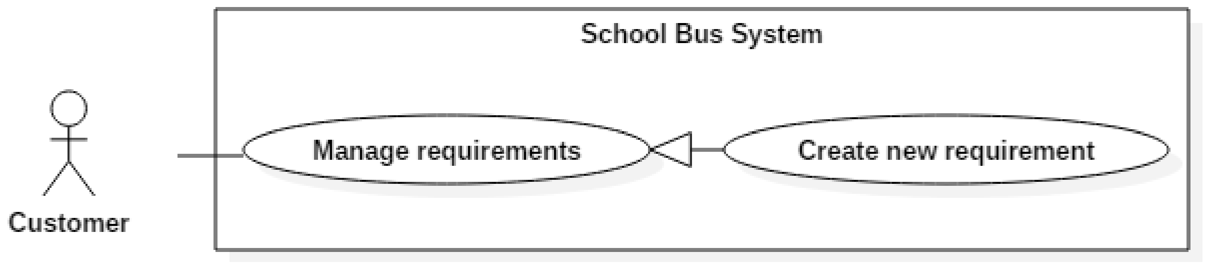
1.2 List of use case

1.2.1 <Customer> Overview Use Case


School Bus System
Get bill information
Condition: User selects a finished trip in trip history. Extension point: Click on a finished Trip.
«extend»
Get trip history
Cancel contract
Extend contract
Get contract
Manage contracts
Condition: User opens options modal to cancel trip. Extension point: Click on 'Cancel trip' button on menu.
Cancel trip
«extend»
Condition: User opens options modal to get cancellation requests. Extension point: Click on 'Get cancellation requests' button on menu.
Get requirement detail
«extend»
Get cancellation requests
«extend»
Find alternative driver
Condition: User chooses a day-off of driver to find alternative driver. Extension point: Click on driver's day-off.
Get notifications
Manage requirements
Create new requirement
Update requirement
Cancel requirement
Customer
Get driver services
«extend»
Manage children
Add new child
Create contract agreement
Update child
Remove child
Track driver trip
Create feedback
Condition: User chooses a driver service in driver services list. Extension point: Click on 'Choose Service' button

Customer Overview Use Case

1.2.1.1 <Customer> Create new requirement



USE CASE – SCB_UC_11			
Use Case No.	SCB_UC_11	**Use Case Version**	1.0
Use Case Name	Create new requirement		
Author	TrungTH		
Date	13/2/2019	**Priority**	Normal
Actor: - Customer **Summary:** - Allow customers to create transporting service requirements. **Goal:** - Customer could create transporting service requirements.			

Triggers:

- Customer sends creating transporting service requirements command.

Preconditions:

- Customer must login first.

Post conditions:

- Success: Customer create new transporting service requirements successfully.
- Fail: System shows error messages.

Main Success Scenario:

Step	Actor Action	System Response
1	Customer goes to creating transporting service requirements screen	Creating requirements screen with following fields: - Days of week (multiple weekday picker) - Pick up address (place suggestion) - Pick up location (place suggestion) - Pick up time (time picker) - Arrival time (time picker) - Return time (time picker) - Start date (date picker) - End date (date picker)
2	Customer inputs transporting service requirements	
3	Customer sends creating transporting service requirements command	Customer register new transporting service requirements [Alternate 1] [Alternate 2] [Exception 1]

Alternative Scenario:

No	Actor Action	System Response
1	Customer leaves fields blank	System shows appropriate validating message. Ex:” You must choose pick up address", ...
2	Datetime logic constraints	Date-Time should be valid. Ex: “Pickup time must before arrival time, arrival time must before return time, start date must before end date.”
3	Requirement logic constraints	New requirement must not have duplicate properties with old ones, like: “Days of week, children, timeline, …”

Exceptions:

No	Actor Action	System Response
1		System shows message the "Please check your connection!" when the internet is lost.

Relationships: [Login Use Case]

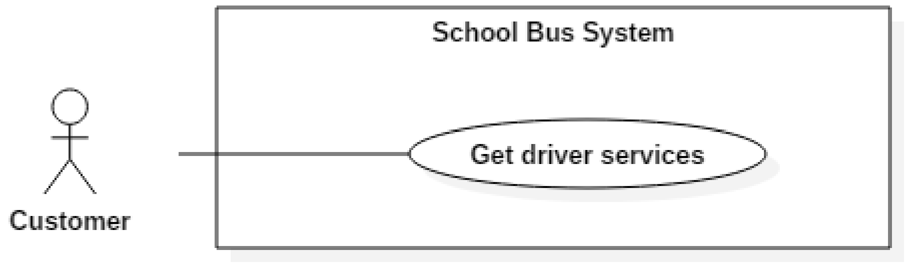
Business Rules:

- Address coordinates must be provided.

- Pickup address should be suggested as customer's address, school address should be suggested as children's school.
- Unit price and estimated total price must be shown.

<Customer> Create new requirement Use Case Specification

1.2.1.2 <Customer> Get Driver Services



USE CASE – SCB_UC_18			
Use Case No.	SCB_UC_18	**Use Case Version**	1.0
Use Case Name	Get driver services		
Author	TrungTH		
Date	13/2/2019	**Priority**	Normal

Actor:
- Customer

Summary:
- Allow customers to get driver's transporting service.

Goal:
- Customer could get driver's transporting service.

Triggers:
- Customer sends getting driver's transporting service command.

Preconditions:
- Customer must login first.
- Customer must create transporting service requirement.

Post conditions:
- Success: Customer gets driver's transporting service successfully.
- Fail: System shows error messages.

Main Success Scenario:

Step	Actor Action	System Response
1	Customer goes to getting driver services screen	Getting driver services screen with following information: - Driver's name - Driver's avatar - Driver's phone - Car's info: Plate number, Brand, Model, Color - Driver's average score based on previous feedbacks [Alternate 1] [Exception 1]

Alternative Scenario:

No	Actor Action	System Response

1	Driver service is not found	Display some announcement, likes “Driver service is not found yet”

Exceptions:

No	Actor Action	System Response
1		System shows message the "Please check your connection!" when the internet is lost.

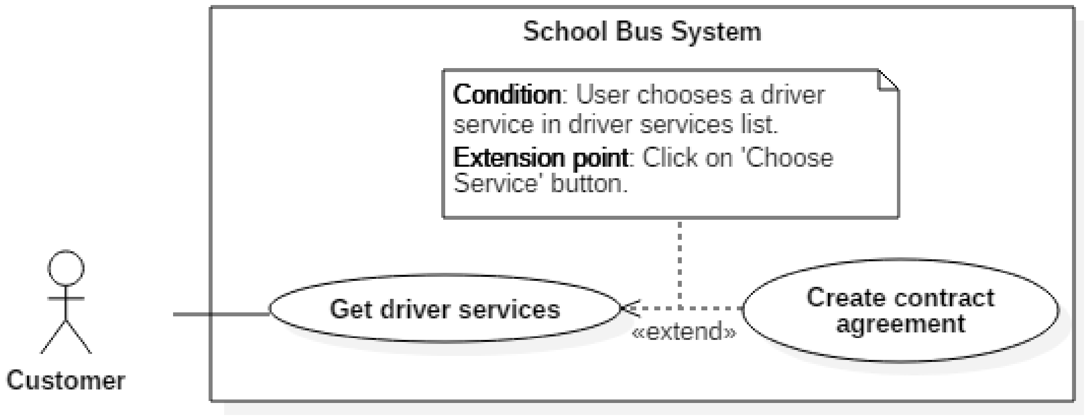
Relationships: [Login Use Case] [Create Requirement Use Case]

Business Rules:

- Appropriate driver services are matched based on matching percentage calculated with following conditions: in-range start point and same destination, days in week, timeline, start and end date, available capacity.

<Customer> Get Driver Services Use Case Specification

1.2.1.3 <Customer> Create Contract Agreement


School Bus System
Condition: User chooses a driver service in driver services list.
Extension point: Click on 'Choose Service' button.
Get driver services
«extend»
Create contract agreement
Customer

USE CASE – SCB_UC_19			
Use Case No.	SCB_UC_19	**Use Case Version**	1.0
Use Case Name	Create Contract Agreement		
Author	TrungTH		
Date	13/2/2019	**Priority**	Normal

Actor:

- Customer

Summary:

- Allow customers to create contract agreement with driver.

Goal:

- Customer could create contract agreement with driver.

Triggers:

- Customer sends creating contract agreement with driver command.

Preconditions:

- Customer must login first.
- Customer must create transporting service requirement.
- Customer must be in viewing driver services screen.

Post conditions:

- Success: Customer create contract agreement with driver successfully.
- Fail: System shows error messages.

Main Success Scenario:

Step	Actor Action	System Response

1	Customer goes to getting driver services screen	Getting driver services screen with following information: - Driver's name - Driver's avatar - Driver's phone - Car's info: Plate number, Brand, Model, Color
2	Customer chooses driver	Alert box for confirming contract with driver
3	Customer confirms creating contract agreement	Customer creates contract agreement successfully. [Alternate 1] [Exception 1]

Alternative Scenario:

No	Actor Action	System Response
1	Customer choose "No"	Continue to the getting driver services screen

Exceptions:

No	Actor Action	System Response
1		System shows message the "Please check your connection!" when the internet is lost.

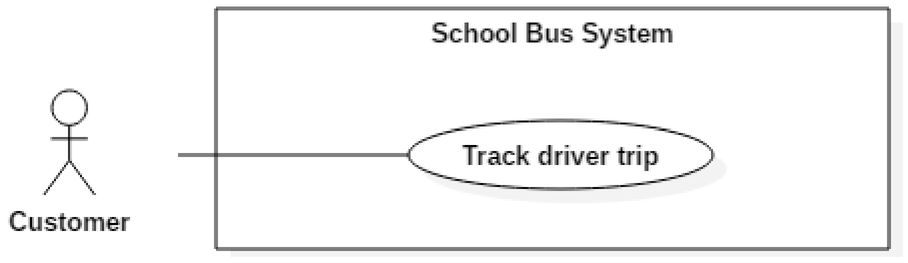
Relationships: [Login Use Case] [Create Requirement Use Case] [Get Driver Service Use Case]

Business Rules:

- After contract agreement is created, we consider that available capacity in driver's provided service is occupied by children in requirement.
- After creating contract agreement, customer would be redirected to viewing transporting service requirements screen.
- System should push notification to driver to let them know about the contract.

<Customer> Create Contract Agreement Use Case Specification

1.2.1.4 <Customer> Track Driver Trip



USE CASE – SCB_UC_21			
Use Case No.	SCB_UC_21	**Use Case Version**	1.0
Use Case Name	Track Driver Trip		
Author	TrungTH		
Date	13/2/2019	**Priority**	Normal

Actor:

- Customer

Summary:

- Allow customers to track driver's current trip.

Goal:

- Customer could track driver's current trip.

Triggers:

- Customer sends tracking driver's current trip command.

Preconditions:

- Customer must login first.
- Customer must be in viewing trip history screen/getting requirement detail screen
- There is a contract.
- The trip must be in "On going" status.

Post conditions:

- Success: Customer tracks driver's current trip successfully.
- Fail: System shows error messages.

Main Success Scenario:

Step	Actor Action	System Response
1	Customer goes to history of all trips screen or notification screen.	
2	Customer goes to tracking driver's current trip screen	Integrated map with following information: - Driver's current position by marker - Driver's current route [Exception 1]

Alternative Scenario:

No	Actor Action	System Response

Exceptions:

No	Actor Action	System Response
1		System shows message the "Please check your connection!" when the internet is lost.

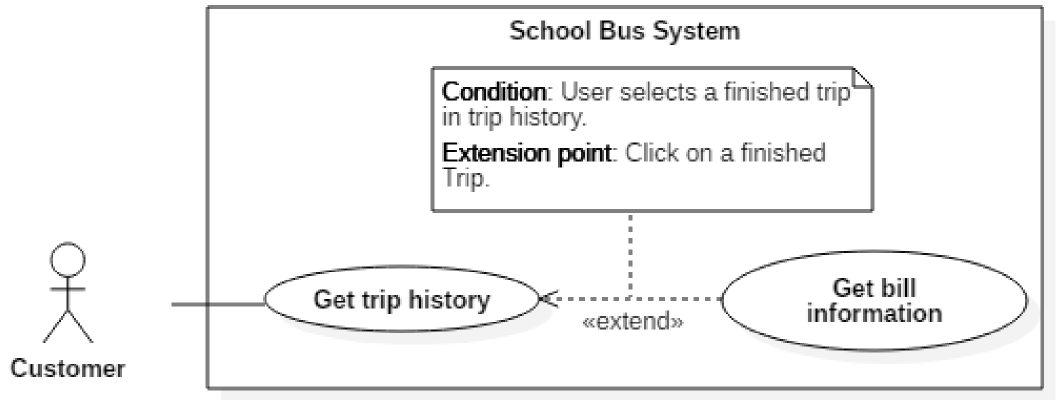
Relationships: [Login Use Case] [Get Trip History Use Case] [Get Requirement Detail Use Case]

Business Rules:

- The location and route of driver's trip must be received repeatedly and automatically.

<Customer> Track Driver Trip Use Case Specification

1.2.1.5 <Customer> Get Bill Information


School Bus System
Condition: User selects a finished trip in trip history.
Extension point: Click on a finished Trip.
Customer
Get trip history
«extend»
Get bill information

USE CASE – SCB_UC_22			
Use Case No.	SCB_UC_22	**Use Case Version**	1.0
Use Case Name	Get Bill Information		
Author	TrungTH		
Date	13/2/2019	**Priority**	Normal

Actor:
- Customer

Summary:
- Allow customers to get trip’s bill information.

Goal:
- Customer could get trip’s bill information.

Triggers:
- Customer sends getting trip’s bill information command.
- Or after finishing trip.

Preconditions:
- Customer must login first.
- Customer must be in viewing trip history screen screen
- There is a contract.
- The trip must be in “Finished” status.

Post conditions:
- Success: Customer gets trip’s bill information successfully.
- Fail: System shows error messages.

Main Success Scenario:

Step	Actor Action	System Response
1	Customer goes to getting history of all trips screen (if not in tracking driver screen)	Getting history of all trips screen with following information: - Trip’s date - Trip’s status - Driver’s avatar - Driver’s name - Children’s avatar Children’s name
2	Customer goes to getting trip’s bill information screen	Getting trip’s bill information screen with following information: - Children’s information - Trip’s detail information: pick-up/drop-off addresses, timeline - Total charge fee. And a Confirm Button [Alternative 1] [Exception 1]

Alternative Scenario:

No	Actor Action	System Response
1	Customer confirms bill	Back to the viewing history of all trips screen

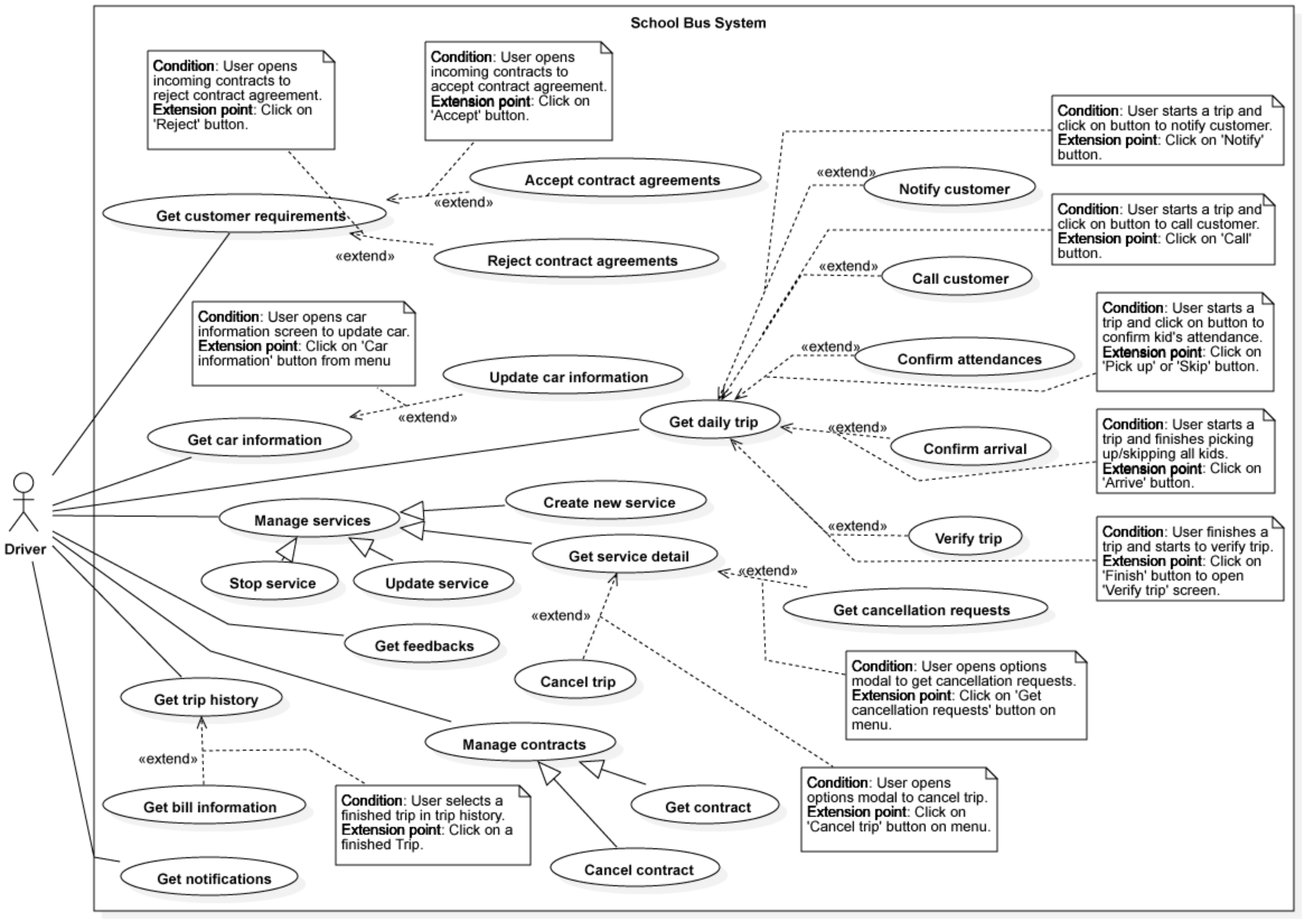
Exceptions:

No	Actor Action	System Response
1		System shows message the "Please check your connection!" when the internet is lost.

Relationships: [Login Use Case] [Get Trip History Use Case] **Business Rules:** - The order detail must have list of kids and the corresponding price. - Total price for the trip must be calculated.

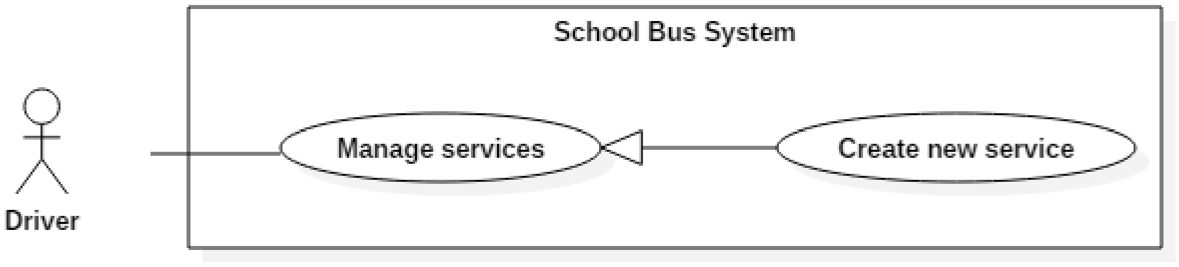
<Customer> Get Bill Information Use Case Specification

1.2.2 <Driver> Overview Use Case


School Bus System
Condition: User opens incoming contracts to reject contract agreement. Extension point: Click on 'Reject' button.
Condition: User opens incoming contracts to accept contract agreement. Extension point: Click on 'Accept' button.
Condition: User starts a trip and click on button to notify customer. Extension point: Click on 'Notify' button.
Accept contract agreements
«extend»
Notify customer
Get customer requirements
«extend»
Condition: User starts a trip and click on button to call customer. Extension point: Click on 'Call' button.
Reject contract agreements
«extend»
Call customer
Condition: User opens car information screen to update car. Extension point: Click on 'Car information' button from menu
Condition: User starts a trip and click on button to confirm kid's attendance. Extension point: Click on 'Pick up' or 'Skip' button.
«extend»
Confirm attendances
Update car information
«extend»
Get daily trip
«extend»
Get car information
Confirm arrival
Condition: User starts a trip and finishes picking up/skipping all kids. Extension point: Click on 'Arrive' button.
Driver
Manage services
Create new service
Get service detail
«extend»
Verify trip
Condition: User finishes a trip and starts to verify trip. Extension point: Click on 'Finish' button to open 'Verify trip' screen.
Stop service
Update service
«extend»
Get cancellation requests
«extend»
Get feedbacks
Cancel trip
Condition: User opens options modal to get cancellation requests. Extension point: Click on 'Get cancellation requests' button on menu.
Get trip history
«extend»
Manage contracts
Condition: User opens options modal to cancel trip. Extension point: Click on 'Cancel trip' button on menu.
Get bill information
Condition: User selects a finished trip in trip history. Extension point: Click on a finished Trip.
Get contract
Cancel contract
Get notifications

Driver Overview Use Case

1.2.2.1 <Driver> Create new service



USE CASE – SCB_UC_29			
Use Case No.	SCB_UC_29	**Use Case Version**	1.0
Use Case Name	Create new service		
Author	Giangtv		

Date	17/02/2019	**Priority**	High

Actor:
- Driver

Summary:
- This use case allows driver to create new shuttling service.

Goal:
- Create new shuttling service.

Triggers:
- Driver touches 'Create' button on 'Service Registration' screen.

Preconditions:
- Driver must login in.

Post conditions:
- Success: Show "Service Detail" screen with created service information.
- Fail: Show error message.

Main Success Scenario:

Step	Actor Action	System Response
1	Driver opens 'Service Registration' screen.	Application shows 'Service Registration' screen with following properties: - Days of week - Start Time - Arrival Time - Return Time - Start Address - School - Class - Available Capacity
2	Driver inputs necessary information.	
3	Driver touches 'Create' button.	Show success notification. [Alternate 1] [Alternate 2] [Exception 1]

Alternative Scenario:

No	Actor Action	System Response
1	Some fields are blank	All fields must not be blank
2	Time constraints are not right	Return time must be greater than arrival time, arrival time must be greater than start time

Exceptions:

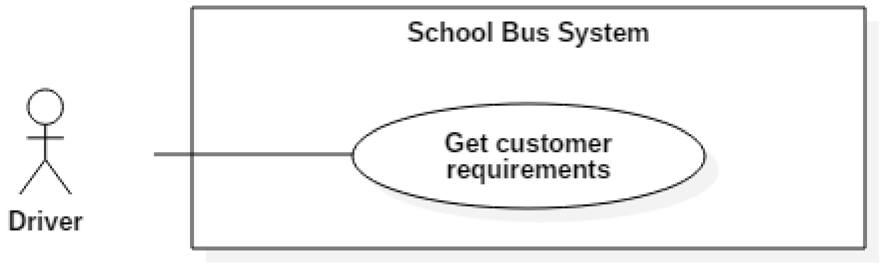
No	Actor Action	System Response
1	No internet connection	Show error message with error code.

Relationships: [Login Use Case]

Business rules:
- Registered time must not overlap any others existing service.
- Address coordinates must be provided.
- Start address should be suggested as driver's address

<Driver> Create new service Specification

1.2.2.2 <Driver> Get customer requirements



USE CASE – SCB_UC_36			
Use Case No.	SCB_UC_36	**Use Case Version**	1.0
Use Case Name	Get customer requirements		
Author	Giangtv		
Date	17/02/2019	**Priority**	High

Actor:

- Driver

Summary:

- View who are registered to use driver service and give driver UI option to reject or accept that requirement.

Goal:

- View registered requirements.

Triggers:

- Driver opens 'Service Detail' screen and choose Incoming Contracts.

Preconditions:

- Driver must be logged in.

Post conditions:

- Success: Show Pending Contract Agreements.
- Fail: Show error message.

Main Success Scenario:

Step	Actor Action	System Response
1	Driver touches "Service Detail" entry in "Services" screen	Show detail of service with following properties: - Start Address - School - Days of week - Going trip – Returning trip - Contracts: active, pending, history
2	Driver touches Incoming tab inside service detail	Show list of pending contract agreements for driver to response, with following properties: - Customer name - Customer avatar - Children List - Going trip – Returning trip - Two buttons for accepting or rejecting [Exception 1] [Alternate 1]

Alternative Scenario:

No	Actor Action	System Response

1	No pending requirements	Show no pending requirements

Exceptions:

No	Actor Action	System Response
1	Internet connection error.	Show error message with error code.

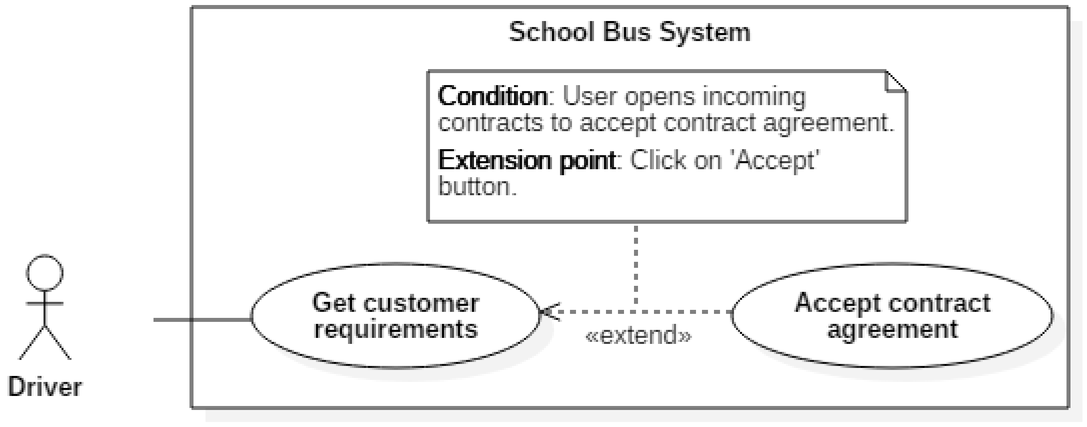
Relationships: [Login Use Case] [Choose Service Use Case] [Extend Contract Use Case]

Business rules:

- The requirements must be in some pending contracts waiting for driver's confirmation

<Driver> Get customer requirements Specification

1.2.2.3 <Driver> Accept contract agreement


School Bus System
Condition: User opens incoming contracts to accept contract agreement.
Extension point: Click on 'Accept' button.
Get customer requirements
«extend»
Accept contract agreement
Driver

USE CASE – SCB_UC_37			
Use Case No.	SCB_UC_37	**Use Case Version**	1.0
Use Case Name	Accept contract agreement		
Author	Giangtv		
Date	17/02/2019	**Priority**	High

Actor:

- Driver

Summary:

- Accept an agreement in the request list.

Goal:

- Accept agreement and create contract.

Triggers:

- Driver touches 'Accept' button in contract agreement.

Preconditions:

- Driver must be logged in.

Post conditions:

- Success: Show success message.
- Fail: Show error message.

Main Success Scenario:

Step	Actor Action	System Response
1	Driver opens list of contract agreements	Show list of pending contract agreements for driver to response, with following properties: - Customer name - Customer avatar - Children List

		- Going trip – Returning trip - Two buttons for accepting or rejecting
2	Driver touches 'Accept' button	Update data and display message. [Exception 1]

Alternative Scenario:

Exceptions:

No	Actor Action	System Response
1	Internet connection error.	Show error message with error code.

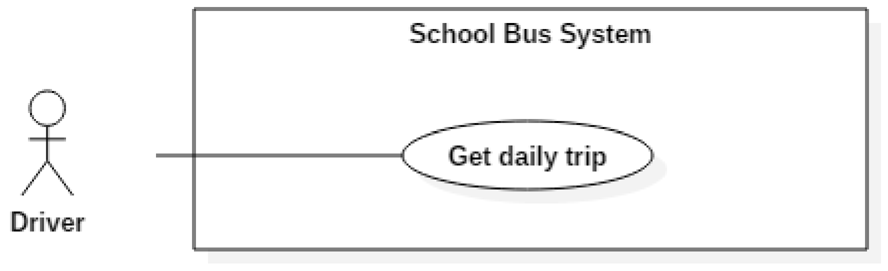
Relationships: [Login Use Case] [Choose Service Use Case] [Extend Contract Use Case]

Business rules:

- Driver capacity is checked before processing Accept.
- If there are any conflicts between the days of week of driver service and customer requirement, the missing days of week in customer will not be removed out of contract.
- Notification is sent to customer when driver accepted.

<Driver> Accept contract agreement Specification

1.2.2.4 <Driver> Get daily trip



USE CASE – SCB_UC_41			
Use Case No.	SCB_UC_41	**Use Case Version**	1.0
Use Case Name	Get daily trip		
Author	Giangtv		
Date	17/02/2019	**Priority**	High

Actor:

- Driver

Summary:

- Get Trip detail for upcoming trip.

Goal:

- Get Trip detail.

Triggers:

- Driver touches 'Start Trip" button.

Preconditions:

- Driver must be logged in.
- Selected trip must have suitable time.

Post conditions:

- Success: Open "Driving" screen.
- Fail: Show error message.

Main Success Scenario:

Step	Actor Action	System Response
1	Driver touches ''Start Trip" button	Open "Driving" screen with following properties:

		- Map (with direction) - Picking-up List: Customer name, avatar, pickup address, children - Buttons: Notify, pickup, skip, call, arrive at school [Exception 1] [Alternate 1]

Alternative Scenario:

Step	Actor Action	System Response
1	Trip has no suitable time	Show message

Exceptions:

No	Actor Action	System Response
1	Internet connection error.	Show error message with error code.

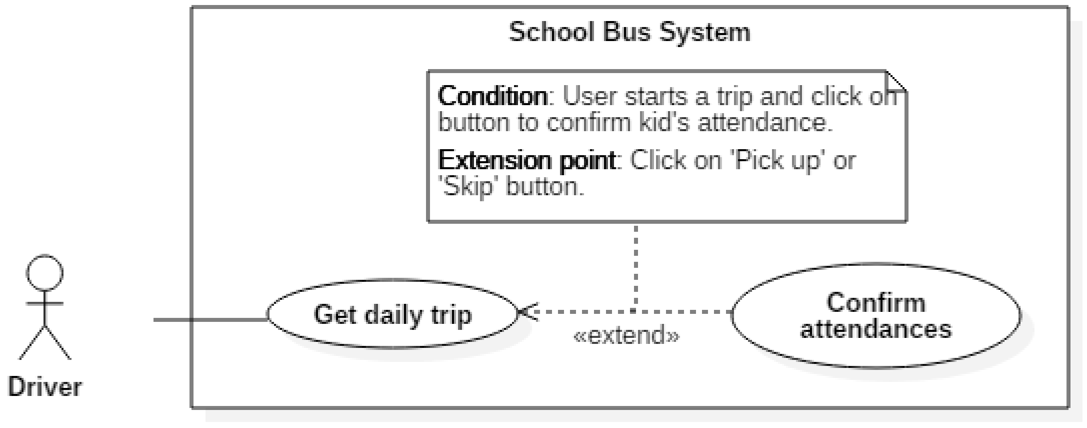
Relationships: [Login Use Case] [Get Trip History Use Case]

Business rules:

- Create new trip for driver if not existed.
- Return current trip if the trip is existed.
- Only get contract that have the matching day of week.
- Absent children are not shown in trip.
- Driver's location and route must be continuously sent to server after getting trip.

<Driver> Get daily trip Specification

1.2.2.5 <Driver> Confirm attendances


School Bus System
Condition: User starts a trip and click on button to confirm kid's attendance.
Extension point: Click on 'Pick up' or 'Skip' button.
Get daily trip
«extend»
Confirm attendances
Driver

USE CASE – SCB_UC_44			
Use Case No.	SCB_UC_44	**Use Case Version**	1.0
Use Case Name	Confirm attendances		
Author	Giangtv		
Date	17/02/2019	**Priority**	High

Actor:

- Driver

Summary:

- Update the status of that kid inside the daily trip. Notify customer to let them know.

Goal:

- Record the data for the daily trip.

Triggers:

- Driver touch 'Picked up' or 'Skip' button.

Preconditions:

- Driver must be logged in.
- Driver must be in a Driving trip.

Post conditions:

- Success: none.
- Fail: Show error message.

Main Success Scenario:

Step	Actor Action	System Response
1	Driver touches 'Start Trip' button	Show 'Driving' screen.
2	Driver touches 'check-mark' or 'dismiss' icon in map or picking up list.	[Exception 1]

Alternative Scenario:

Exceptions:

No	Actor Action	System Response
1	Internet connection error.	Show error message with error code.

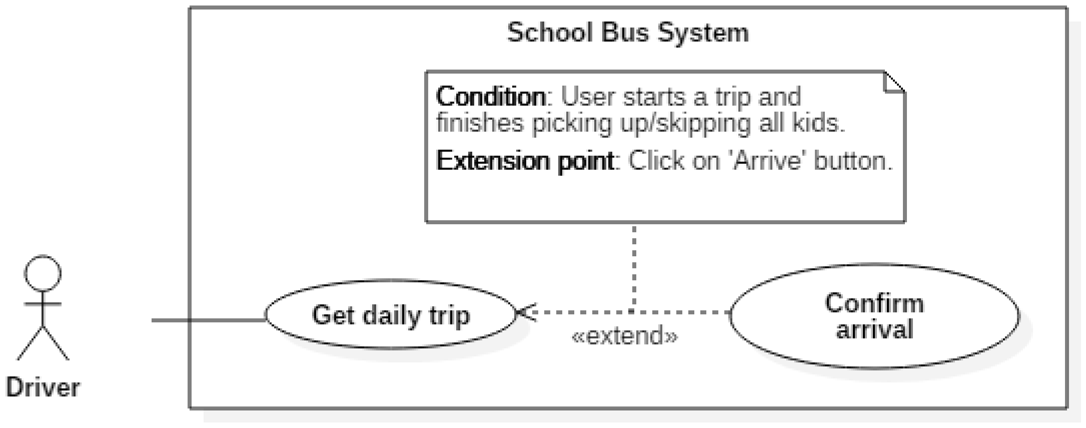
Relationships: [Login Use Case] [Get Daily Trip]

Business rules:

- Driver can only confirm attendances for kids who are not absent that day.
- Notification is sent to customer when driver picked-up or skip his/her child.

<Driver> Confirm attendances Specification

1.2.2.6 <Driver> Confirm arrival


School Bus System
Condition: User starts a trip and finishes picking up/skipping all kids.
Extension point: Click on 'Arrive' button.
Get daily trip
«extend»
Confirm arrival
Driver

USE CASE – SCB_UC_45			
Use Case No.	SCB_UC_45	**Use Case Version**	1.0
Use Case Name	Confirm arrival		
Author	Giangtv		
Date	17/02/2019	**Priority**	High

Actor:

- Driver

Summary:

- Update trip status and notify customers.

Goal:

- Record the data for the daily trip.

Triggers:

- Driver touch 'Arrive' button.

Preconditions:

- Driver must be logged in.
- Driver must be in a Driving trip.

Post conditions:

- Success: Open "Trip Completion" screen.
- Fail: Show error message.

Main Success Scenario:

Step	Actor Action	System Response
1	Driver touches 'Arrive' button	Open "Trip Completion" screen on success. [Exception 1]

Alternative Scenario:

Exceptions:

No	Actor Action	System Response
1	Internet connection error.	Show error message with error code.

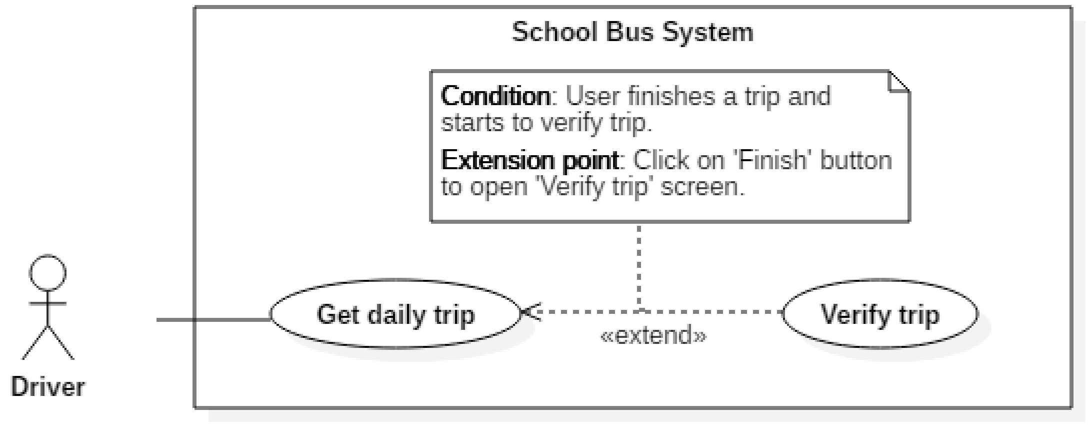
Relationships: [Login Use Case] [Get Daily Trip]

Business rules:

- Send notification to customer with notification information according to trip information.
- Customer whose children are all skipped or absent will not receive the notification.

<Driver> Confirm arrival Specification

1.2.2.7 <Driver> Verify trip


School Bus System
Condition: User finishes a trip and starts to verify trip.
Extension point: Click on 'Finish' button to open 'Verify trip' screen.
Get daily trip
«extend»
Verify trip
Driver

USE CASE – SCB_UC_46			
Use Case No.	SCB_UC_46	**Use Case Version**	1.0
Use Case Name	Verify Trip		
Author	Giangtv		
Date	17/02/2019	**Priority**	High

Actor:

- Driver

Summary:

- Driver has to take a picture of the children and message to announce parents after completing trip.

Goal:

- Make sure the driver has driven the kids to school successfully.

Triggers:

- Driver touches 'Submit' button.

Preconditions:

- Driver must be logged in.
- Driver must arrive after Driving trip.

Post conditions:

- Success: Open "Bill" screen.
- Fail: Show error message.

Main Success Scenario:

Step	Actor Action	System Response
1	Driver goes to the 'Verification' screen	'Verification' screen with following properties: - Image for evidence - Content
2	Driver touches "Send" button	Open bill screen [Exception 1] [Alternate 1]

Alternative Scenario:

No	Actor Action	System Response
1	No empty fields	All fields must not be blank

Exceptions:

No	Actor Action	System Response
1	Internet connection error.	Show error message with error code.

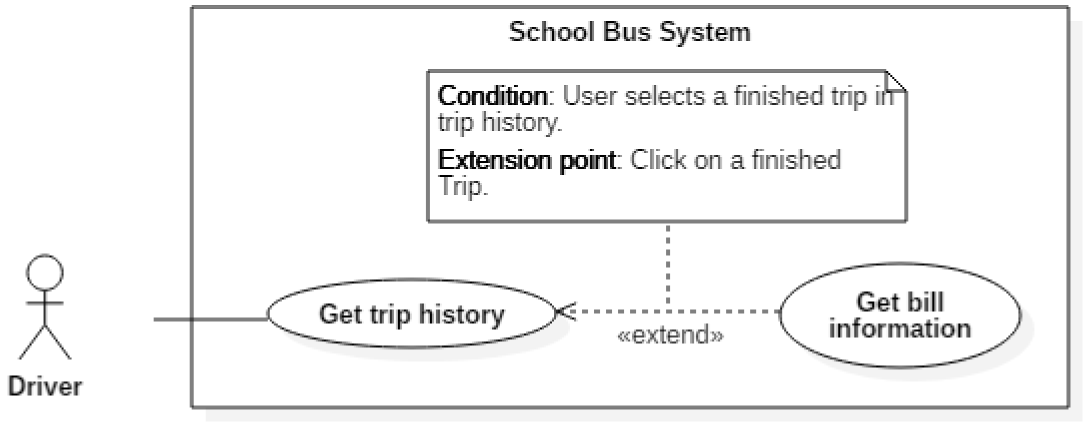
Relationships: [Login Use Case] [Get Daily Trip]

Business rules:

- Confirm image can only be taken using camera. No any others method accepted.
- Customer whose children are all skipped or absent will not receive the notification.

<Driver> Verify trip Specification

1.2.2.8 <Driver> Get bill information


School Bus System
Condition: User selects a finished trip in trip history.
Extension point: Click on a finished Trip.
Get trip history
«extend»
Get bill information
Driver

USE CASE – SCB_UC_48			
Use Case No.	SCB_UC_48	**Use Case Version**	1.0
Use Case Name	Get bill information		
Author	Giangtv		
Date	17/02/2019	**Priority**	High

Actor:

- Driver

Summary:

- This use case allows driver to view bill of a finished trip.

Goal:

- View the bill of a specific trip.

Triggers:

- Driver touches Trip record on 'History screen' or after completing trip.

Preconditions:

- Driver must be logged in.
- Trip must be finished.

Post conditions:

- Success: Show bill information
- Fail: Show error message.

Main Success Scenario:

Step	Actor Action	System Response
1	Driver opens 'History' screen.	Show history screen with following properties: - School - Day - Time - Status
2	Driver touches a trip in the list	Show bill information with following properties: - Day - Start Time - Trip id - Driver name - Driver avatar - Driver phone - Car's plate number - Children - Pickup address - Pickup time - Arrival time - Unit Price - Total Price [Exception 1]

Exceptions:

No	Actor Action	System Response
1	API cannot connect to server.	Show error message with error code.

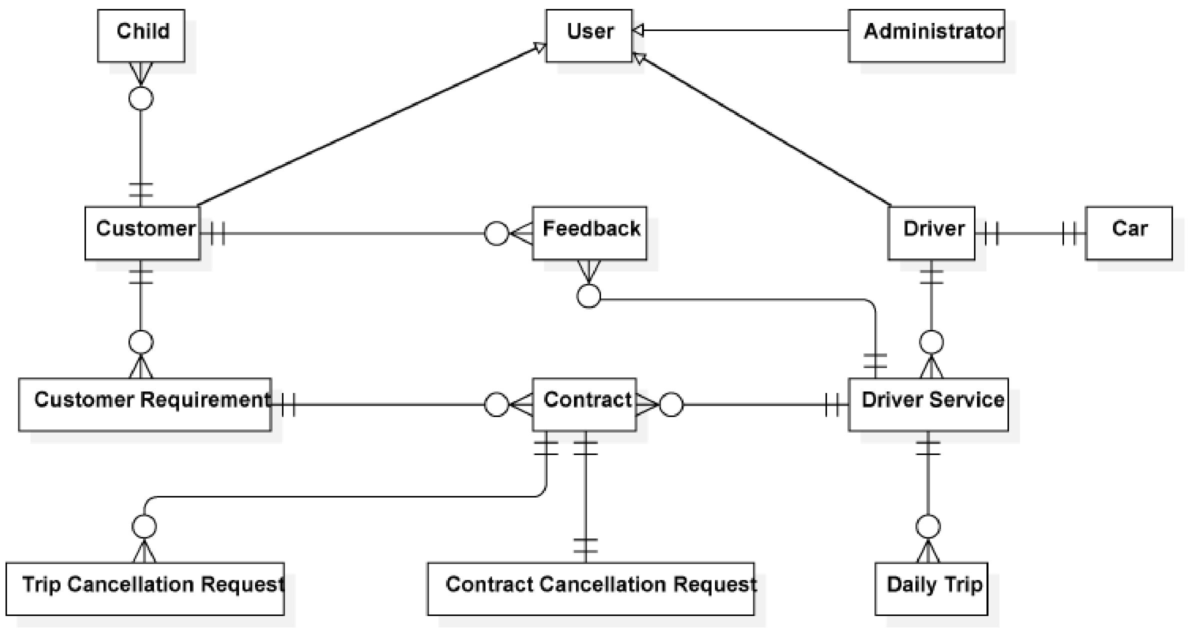
Relationships: [Login Use Case] [Get Trip History Use Case]

Business rules:

- The order detail must have list of kids and the corresponding price.
- Total price for the trip must be calculated.

<Driver> Get bill information Specification

2. Conceptual Diagram



Data Dictionary

Entity Data dictionary: describe all content of all entities	
Entity Name	**Description**
User	Abstract entity describes a user in system
Customer	Contain the customer information
Driver	Contain the driver information
Administrator	Contain the administrator information
Child	Contain the child information
Car	Contain the car information
Customer Requirement	Contain the customer requirement information
Driver Service	Contain the driver service information
Contract	Refer the contract between driver service and customer requirement. Contain the contract information.
Feedback	Contain the feedback information of customer for a driver service.
Daily trip	Contain the daily trip information.
Trip cancellation request	Contain the trip cancellation request for a day-off.
Contract cancellation request	Contain the contract cancellation request.

D. Software Design Description

1. System Architectural Design

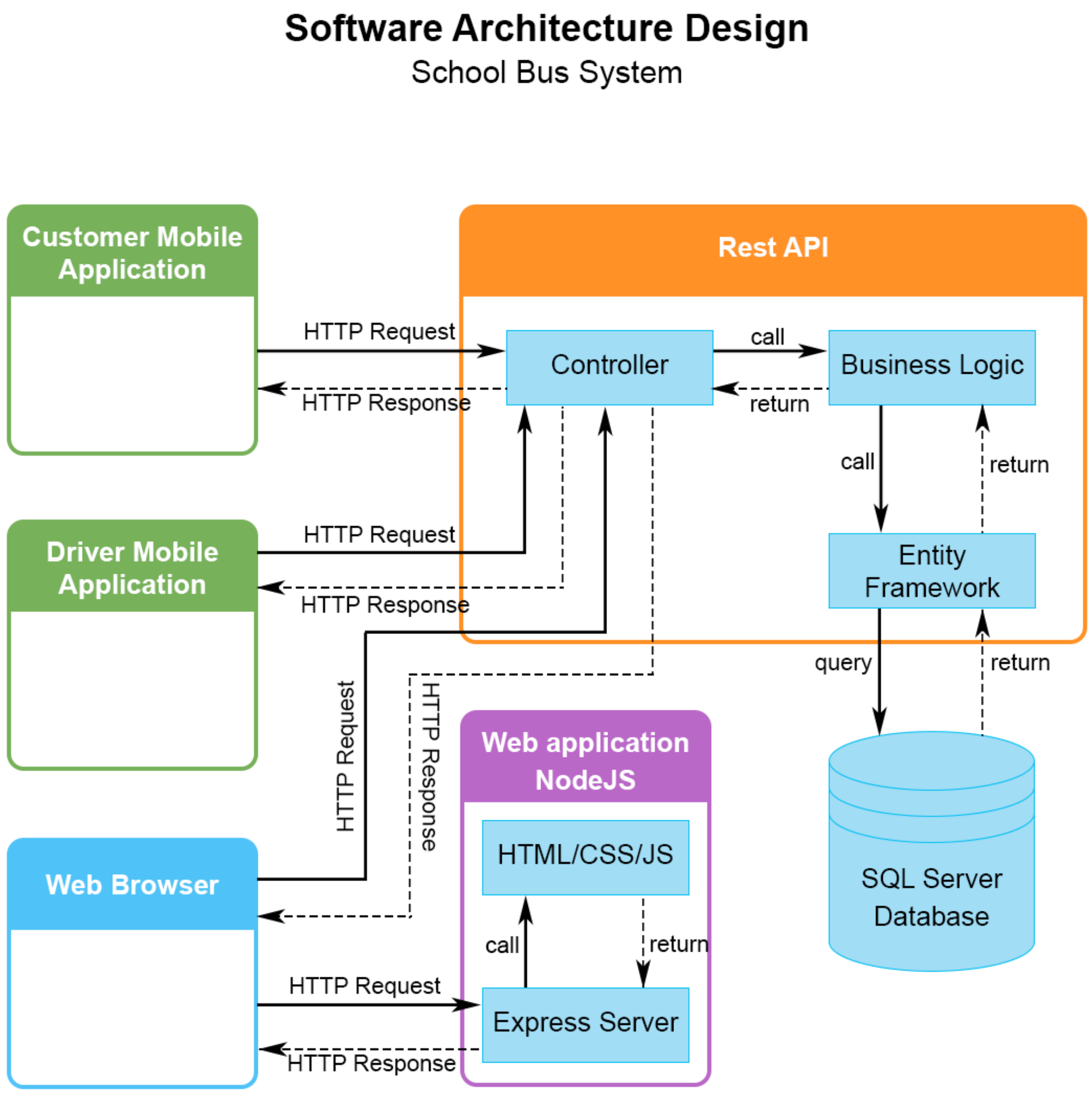


Figure - Software Architecture Design

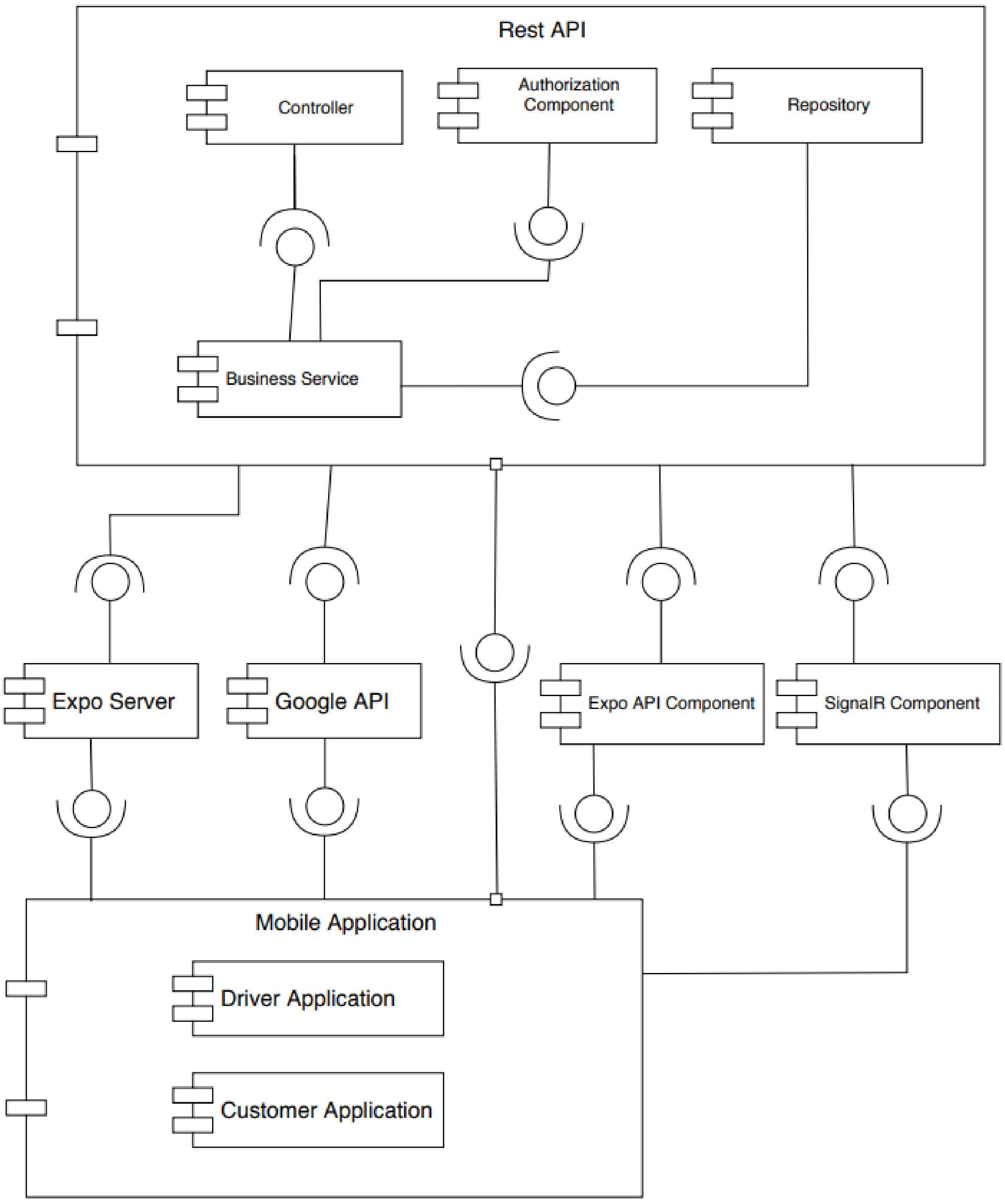
Description

Rest API: Because Mobile application and Web application are separated, web service is needed as a center to get and update data in database. Web service makes it easier to change business logic. Controller – Business Logic – Entity Framework is one of the most common structures used in modern systems.

Mobile application: Both driver and customer application are built with React-native and Expo framework. We choose React-native because it supports to build application in both Android and iOS. The component-based structure in React is the most common one used in user interface development, since it arranges everything in components, which are highly extendable, maintainable and reusable. Expo framework provides great SDK to develop mobile application, helps to build application easier and faster.

Web application: This is the web administration application developed with a stack combined by ReactJS, Webpack and some other related libraries. React gives many advantages for developing user interface as mentioned. Webpack is a JavaScript bundler, combining, minimizing, converting ES6 to pure JavaScript. Express Server is used to dispatch web application to user browsers.

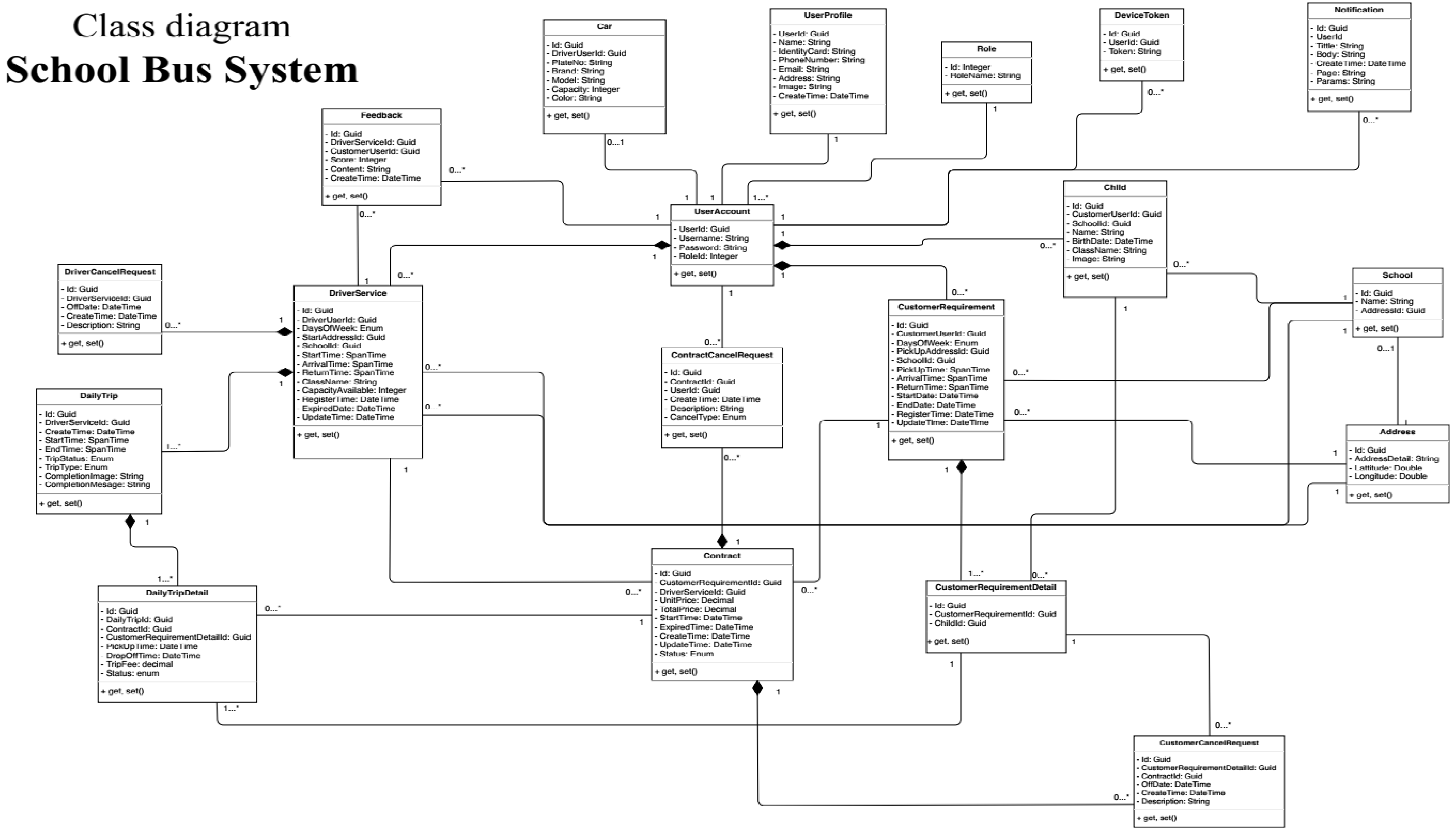
2. Component Diagram



COMPONENT DICTIONARY: DESCRIBES COMPONENTS	
Rest API	
Controller	Handle requests and responses; accept input and convert it to commands for model and view.
Authorization Component	Check user role before API called.
Business Service	Handle business process, transform data.
Repository	Entity framework to connect to database.
COMPONENT DICTIONARY: DESCRIBES COMPONENTS	
Expo Server	Third party component that sends notification.
Google API	Handle map direction, place, geocoding and distance function.
Expo API component	Send notification to expo server.
SignalR component	Real time socket for location tracking function.
Mobile Application	
Driver Application	Android/IOS application
Customer Application	Android/IOS application

3. Detailed Description

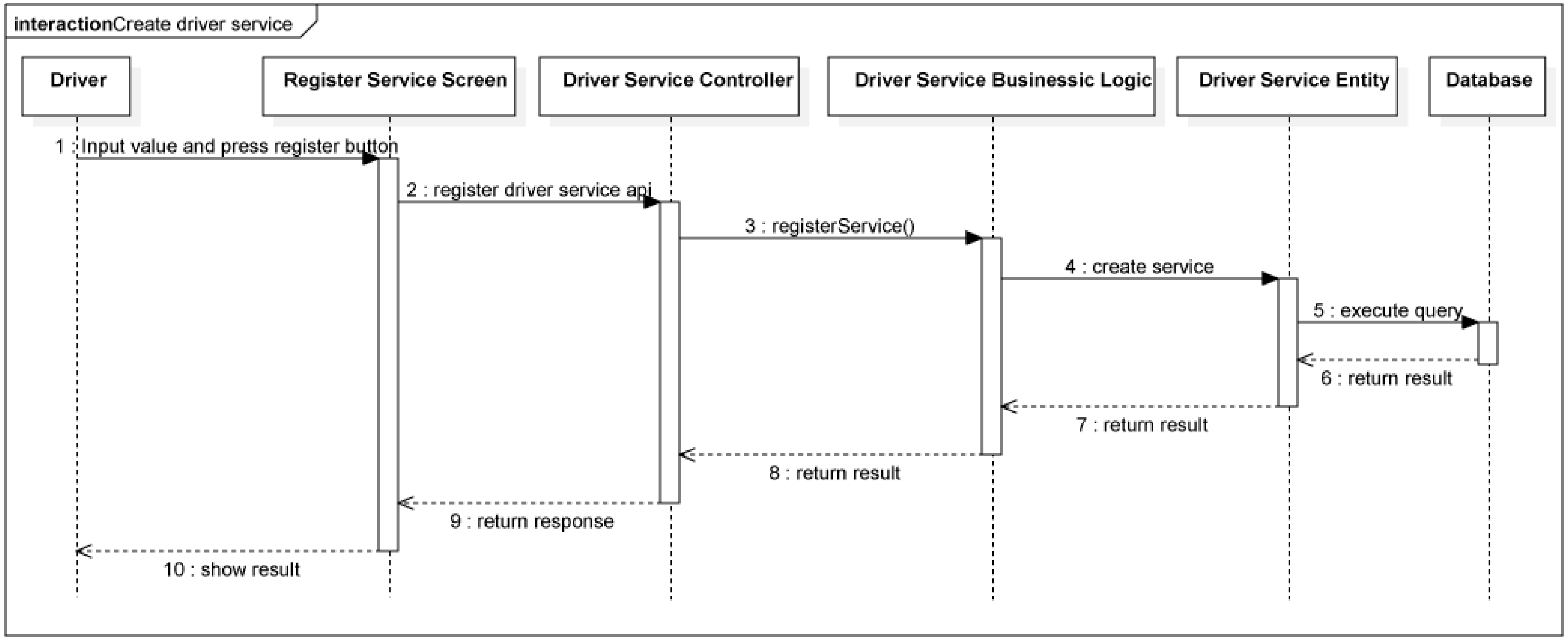
3.1 Class Diagram



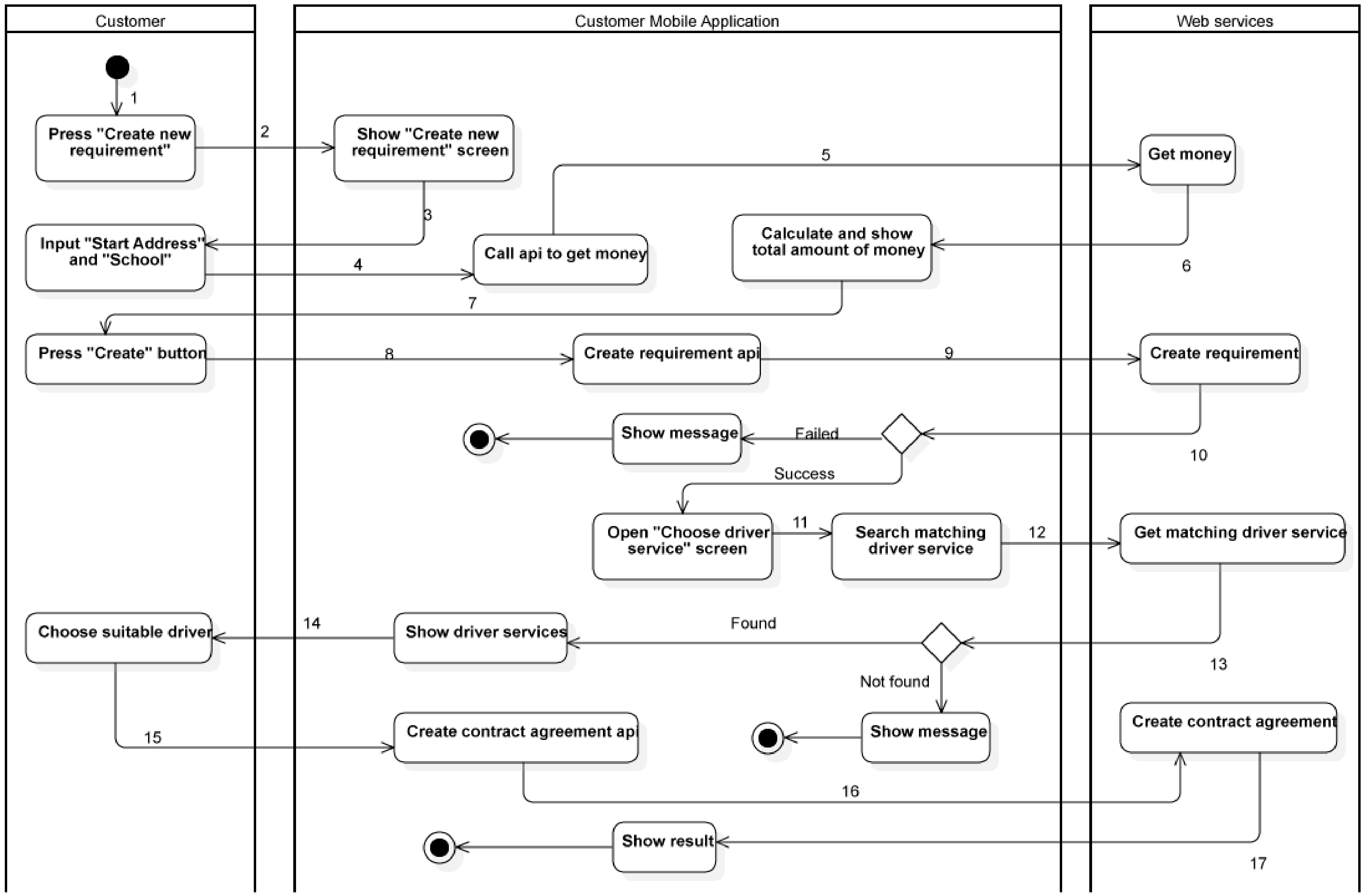
CLASS DICTIONARY: DECRIBE CLASS		
Class Name	**Mapping column with Conceptual Diagram**	**Description**
UserAccount	N/A	Not exist in conceptual diagram. Account information that contain login data.
UserProfile	User	Detail user information linked with UserAccount.
Role	N/A	Not exist in conceptual diagram. It's used to contain role information.
Car	Car	Information about Car linked to Driver account.
Child	Child	Children of customer who will directly use Driver Servicce.
DeviceToken	N/A	Not exist in conceptual diagram. Device identity linked to UserAccount. It's used to send notification.
Notification	N/A	Notification of user. User can read old notification out of OS notification.
Feedback	Feedback	Feedback of customer for driver services.
DriverService	DriverService	SchoolBus services that driver registered to our system.
DriverCancelRequest	Trip cancellation request	Contain driver request to off a trip.
CustomerRequirement	Customer requirement	Customer's requirements for a SchoolBus service. It's used to matching with DeriverService.
CustomerRequirement-Detail	N/A	Not exist in conceptual diagram. It contains children who linked to CustomerRequirement.
CustomerCancelRequest	Trip cancellation request	Customer's cancel request for child in a day.
Contract	Contract	It contains information of a contract between DriverService and CustomerRequirement.
ContractCancelRequest	Contract cancellation request	Requests of customer or driver to cancel current acctive contract.
DailyTrip	Daily trip	Trip information of a day in the contract.
DailyTripDetail	N/A	Not exist in conceptual diagram. Information for a child in a DailyTrip. It mostly contains status of child which changed after each stage of a trip.
School	N/A	Not exist in conceptual diagram. School information stored for reusing.
Address	N/A	Not exist in conceptual diagram. Contain address detail used for map, location function.

3.2 Interaction Diagram

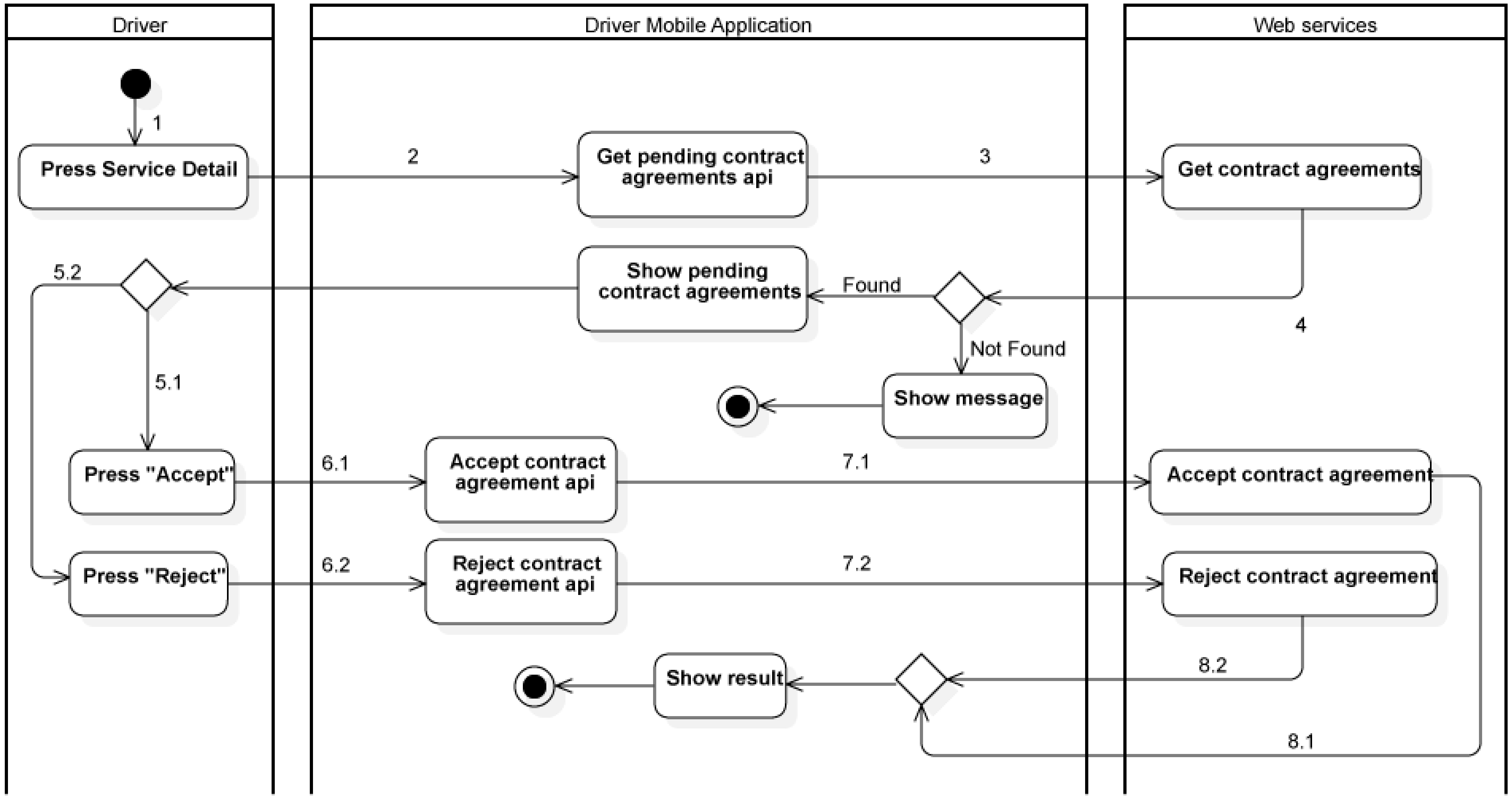
3.2.1 <Driver> Create driver service



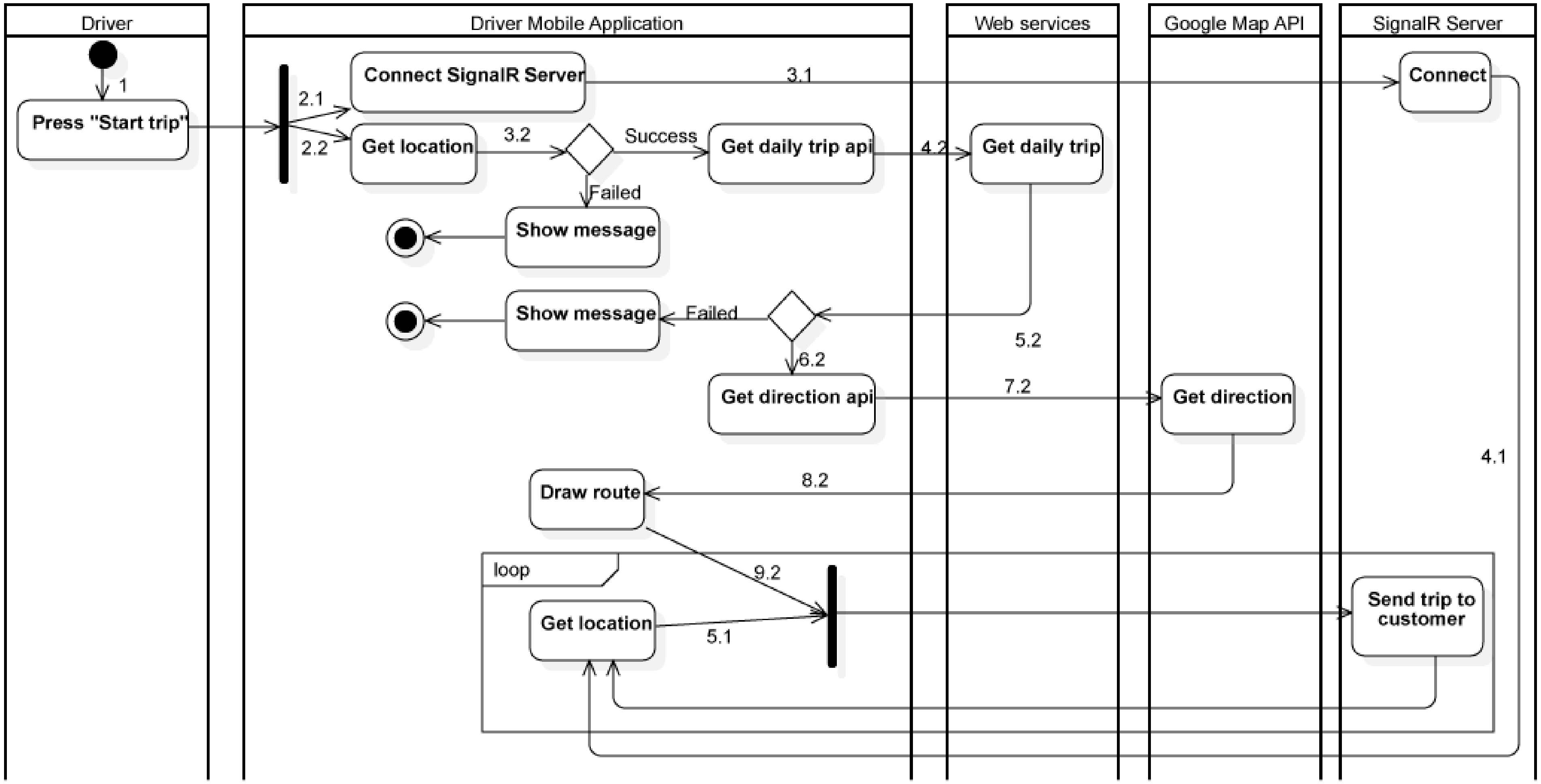
3.2.2 <Customer> Create customer requirement


Customer
Customer Mobile Application
Web services
1
Press "Create new requirement"
2
Show "Create new requirement" screen
3
Input "Start Address" and "School"
4
Call api to get money
5
Get money
6
Calculate and show total amount of money
7
Press "Create" button
8
Create requirement api
9
Create requirement
10
Failed
Show message
Success
Open "Choose driver service" screen
11
Search matching driver service
12
Get matching driver service
13
Found
Show driver services
14
Choose suitable driver
Not found
Show message
15
Create contract agreement api
16
Create contract agreement
17
Show result

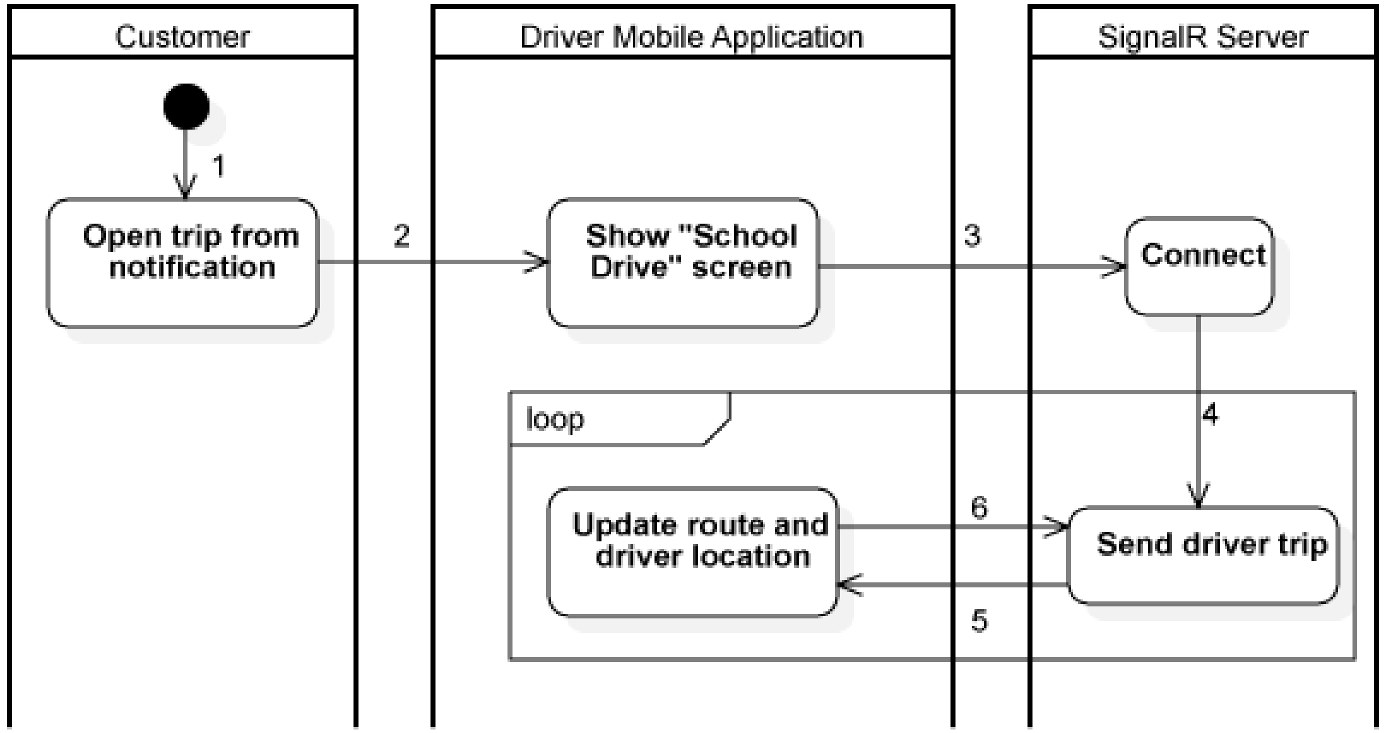
3.2.3 <Driver> Respond customer request


Driver
Driver Mobile Application
Web services
1
Press Service Detail
2
Get pending contract agreements api
3
Get contract agreements
4
Found
Show pending contract agreements
Not Found
Show message
5.2
5.1
Press "Accept"
6.1
Accept contract agreement api
7.1
Accept contract agreement
Press "Reject"
6.2
Reject contract agreement api
7.2
Reject contract agreement
8.2
Show result
8.1

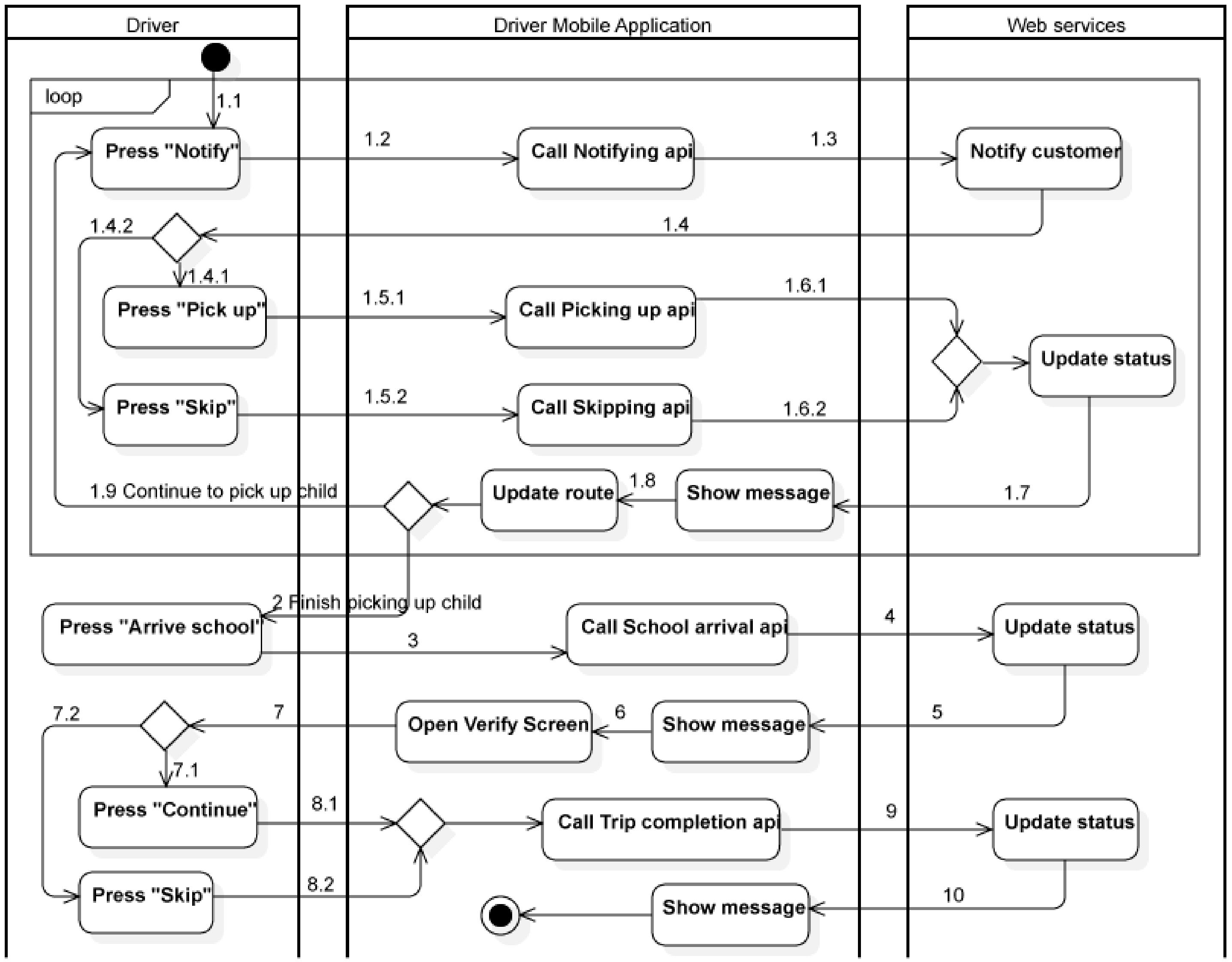
3.2.4 <Driver> Get daily trip



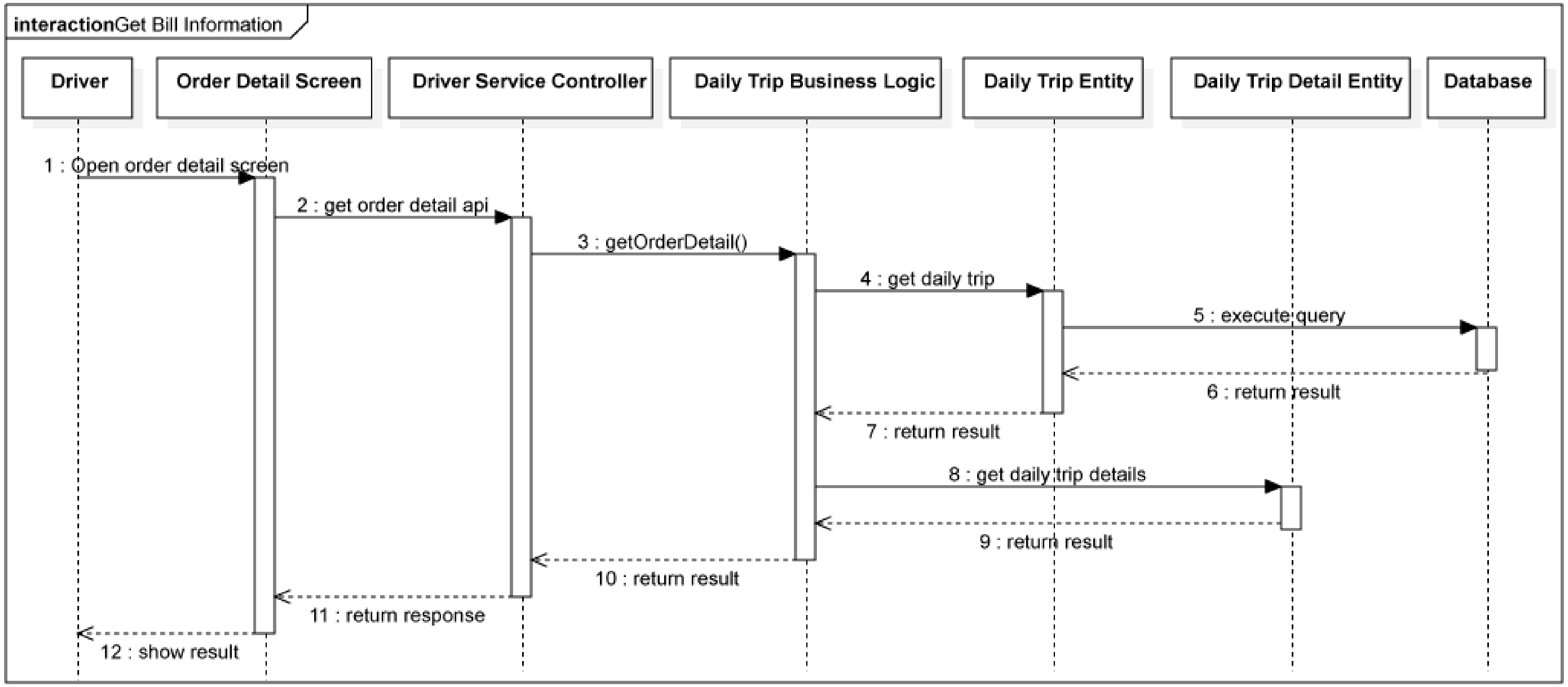
3.2.5 <Customer> Track driver


Customer
Driver Mobile Application
SignalR Server
1
Open trip from notification
2
Show "School Drive" screen
3
Connect
4
loop
Update route and driver location
6
Send driver trip
5

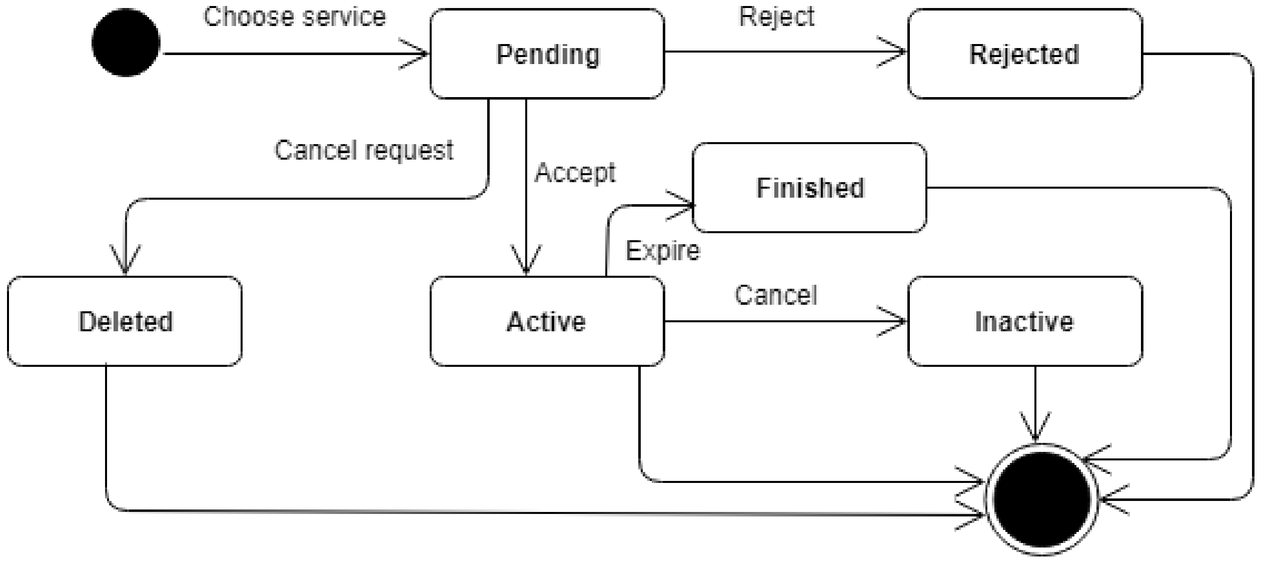
3.2.6 <Driver> Complete trip



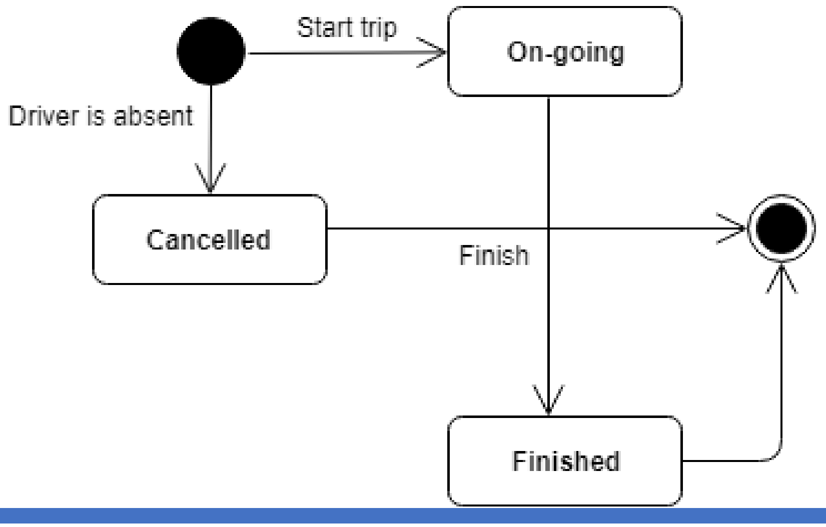
3.2.7 <Driver> Get Bill Information



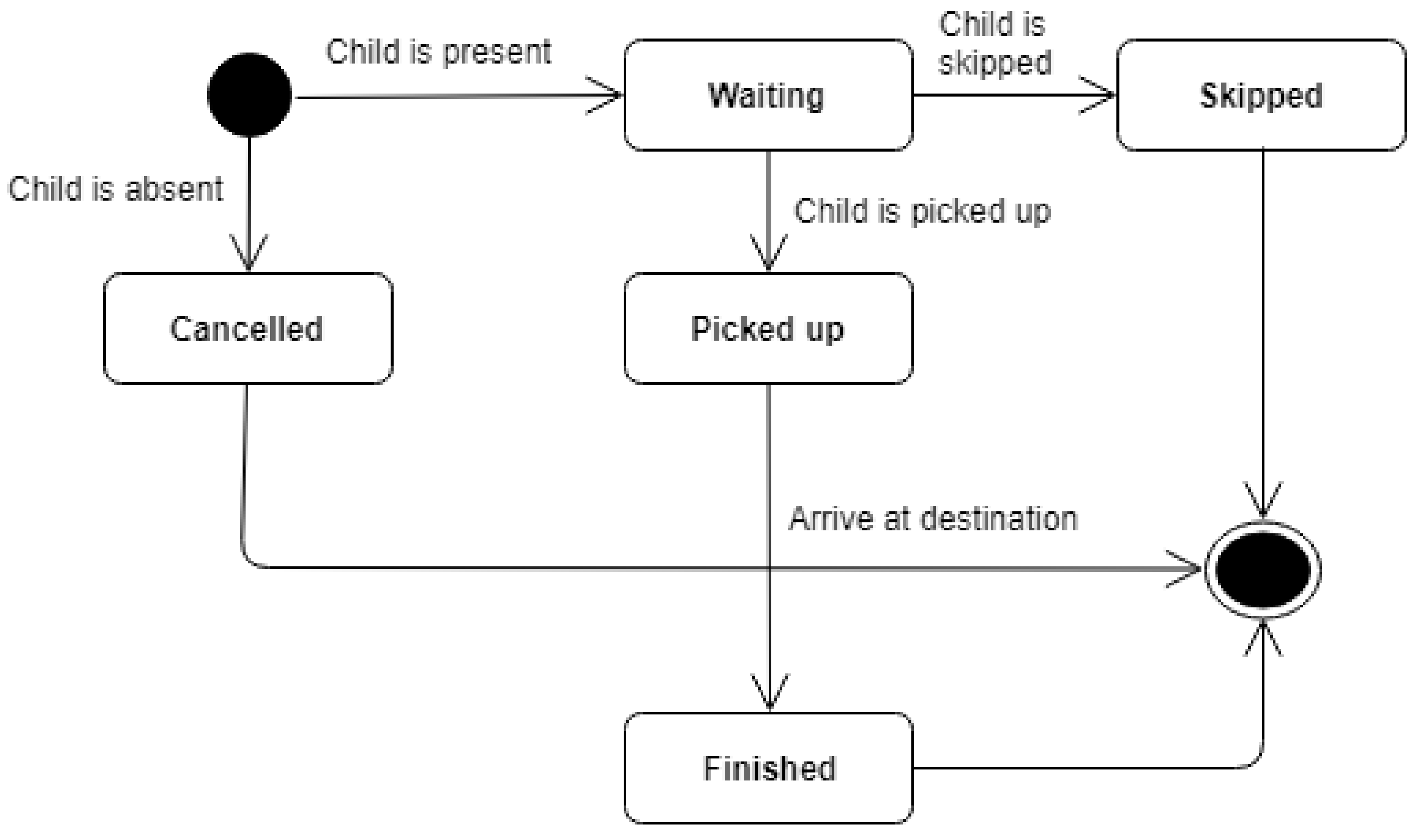
3.2.8 <Contract> State machine diagram



3.2.9 <Daily Trip> State machine diagram

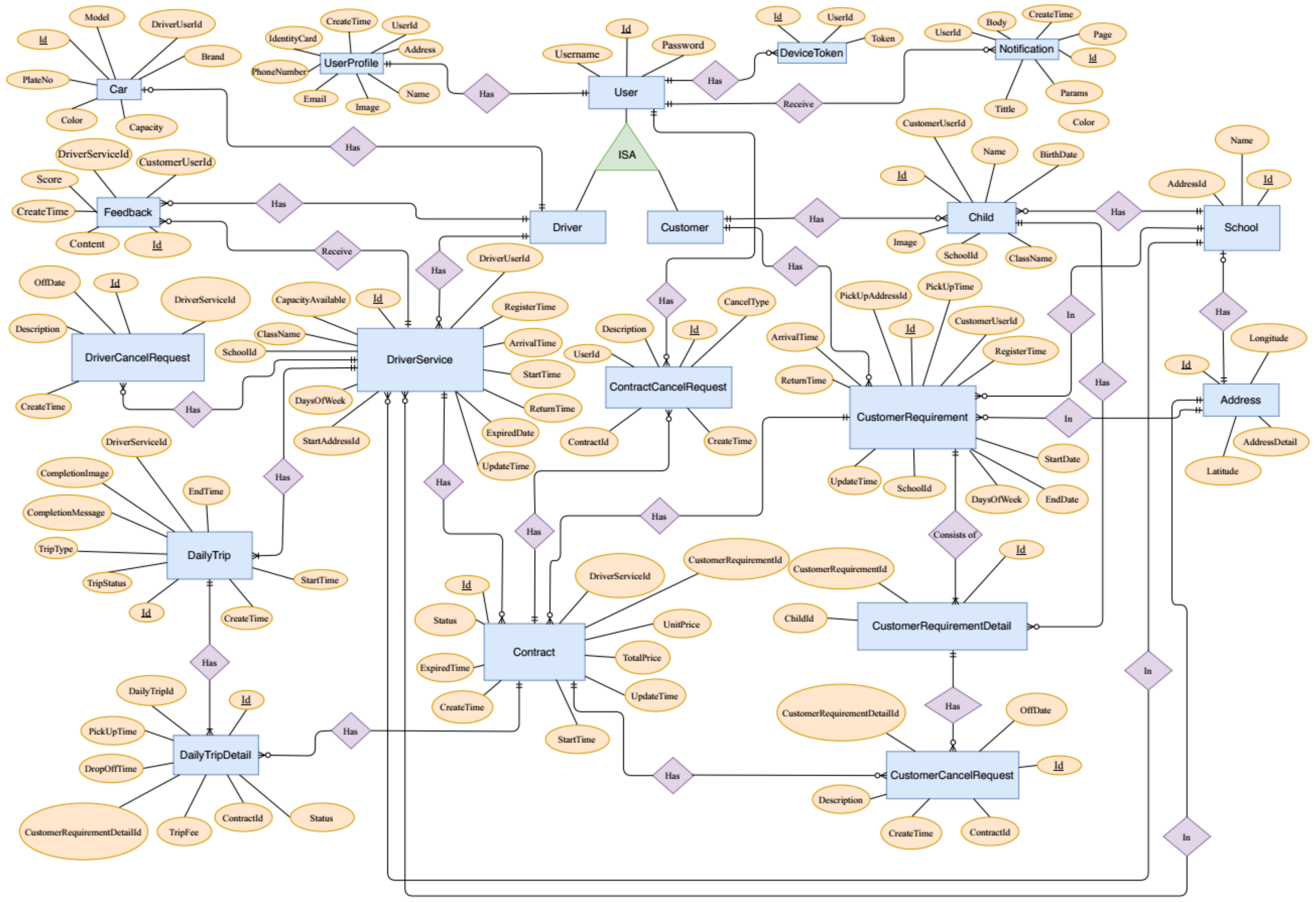


3.2.10 <Daily Trip - Child> State machine diagram



4. Database Design

4.1 Entity Relational Diagram



Entity Data Dictionary: describe content of all entities	
Entity name	Description
User	Contains the login's information.
User profile	Contain user's information.
DeviceToken	Contain user's devices token.
Car	Contain car's information of driver.
Feedback	Contain customer feedback for driver's service.
Notification	Contain notification of user.
Child	Contain child's information of customer.
DriverService	Contain Driver Service's information.
Contract	Contain Contract's information. Contract links CustomerRequirement with DriverService.
DriverCancelRequest	Contain cancel request of driver.
DailyTrip	Contain information for a trip.
DailyTripDetail	Contain daily trip information for a child.
ContractCancelrequest	Contain contract cancel request for driver and customer.
CustomerRequirement	Contain customer requirement for matching Driver Service.
CustomerRequirementDetail	Contain inforamtions for each child in a CustomerRequirement.
CustomerCancelRequest	Contain cancel request for a trip of customer.
School	Contain school information.
Address	Contain detail information for School and other Address.

Entity Name	Attributes	Description	Domain	Null
CustomerRequirement	<u>CustomerRequirementID</u>	ID of requirement	Guid	No
	UserID	ID of owner	Guid	No
	DaysOfWeek	String contain all day of requirement	Nvarchar(max)	No
	PickUpAddressID	ID of address	Guid	No
	SchoolID	ID of school	Guid	No
	PickUpTime	Time to pick children	DateTime	No
	ArrivalTime	Time required to go to school	DateTime	No
	ReturnTime	Time to pick up child after school	DateTime	No
	StartDate	The day customer wants to be served	DateTime	No
	EndDate	Day when customer want to stop the service	DateTime	No
	RegisterTime	Time when the Requirement is registered	DateTime	No
	UpdateTime	Latest time when the Requirement is updated	DateTime	No
	Status	Status of requirement	Bit	No
Contract	<u>ContractID</u>	Id of contract	Guid	No
	CustomerRequirementID	ID of customer requirement	Guid	No
	DriverServiceID	ID of driver service	Guid	No
	UnitPrice	Fee for a child	Decimal(18,2)	Yes
	TotalPrice	Total fee of the contract	Decimal(18,2)	Yes
	StartTime	Time when the contract is activated	DateTime	Yes
	ExpiredTime	Time when the contract is expired	DateTime	Yes
	CreateTime	Time when the contract is created	DateTime	No
	UpdateTime	Lasted time when the contract is updated	DateTime	No
	Status	Status of the contract	nvarchar(max)	No
DailyTrip	<u>DailyTripID</u>	ID of the trip	Guid	no
	DriverServiceID	ID of the service to be served	Guid	No
	StartTime	Time when the trip is started	DateTime	No
	EndTime	Time when the trip is finished	DateTime	Yes
	CreateTime	Time when the trip is created	DateTime	No
	Status	Status of the trip	Nvarchar(max)	No
	Type	Trip' type	Nvarchar(max)	No
	CompletionImage	Link to confirm image	Nvarchar(max)	Yes
	CompletionMessage	Completion message	Nvarchar(max)	Yes
UserProfile	<u>UserID</u>	Id of profile	Guid	No

	IdentityCard	Identity card number	Nvarchar(max)	No
	Name	User's name	Nvarchar(max)	No
	PhoneNumber	User's phone number	Nvarchar(max)	No
	Email	User's email	Nvarchar(max)	yes
	Address	User's address	Nvarchar(max)	Yes
	Image	Link to user's avatar	Nvarchar(max)	Yes
	CreateDate	Time when user account was created	DateTime	No
	IsActive	This account is activated or not	Bit	No
DailyTripDetail	<u>DailyTripDetailID</u>	Id of the trip detail	Guid	No
	DailyTripID	ID of the original trip	Guid	No
	ContractID	Id of the original contract	Guid	No
	CustomerRequirementDetailID	Id of the requirement detail	Guid	No
	PickUpTime	Time when driver picked up the child in the trip	DateTime	Yes
	DropOffTime	Time when driver dropped the child	DateTime	Yes
	TripFee	Fee of the trip	Decimal(18,2)	No
	status	Status of the trip	Nvarchar(max)	No
Children	<u>ChildID</u>	Id the the child	Guid	No
	UserID	Parent ID	Guid	No
	Name	Name of the child	Nvarchar(max)	No
	BirthDate	Date of birth of the child	DateTime	No
	SchoolID	Id of child's school	Guid	No
	ClassName	Nvarchar(max)	Class of the child	No
	Image	Link to child image	Nvarchar(max)	No
	IsActive	Is this child inactive or not	bit	No
Car	<u>CarID</u>	Id of the car	Guid	No
	PlateNo	Plate number of the car	Nvarchar(max)	No
	UserID	ID of the owner	Guid	No
	Brand	Car's brand	Nvarchar(max)	No
	Model	Car's model	Nvarchar(max)	No
	Capacity	Car's empty slot	int	No
	Color	Car's color	Nvarchar(max)	No
	IsActive	Is this car inactive or not	Bit	No
Notification	<u>NotificationID</u>	ID o the notification	Guid	No
	UserID	Id of the owner	Guid	No
	Title	Notification's title	Nvarchar(max)	No
	Body	Notification's body	Nvarchar(max)	No
	Time	Time when notificatin was pushed	DateTime	No
	Page	Todo action	Nvarchar(max)	Yes
	Params	Params for the action	Nvarchar(max)	Yes
CustomerCancelRequest	<u>CustomerCancelRequestID</u>	Id of the request	Guid	No

	CustomerRequirementDetailID	ID of the detail that need to be cancelled	Guid	No
	ContractID	Id of the contract	Guid	No
	OffDate	Day to be cancelled	DateTime	No
	CreateTime	Time when request was created	DateTime	No
	Description	Addition information for the request	Nvarchar(max)	No
ContractCancelRequest	ContractCancelRequestID	ID of the request	Guid	No
	ContractID	Id of the request to be cancelled	Guid	No
	UserID	Id of the request's owner	Guid	No
	CreateTime	Time when the request was created	DateTime	No
	Description	Cancel message	Nvarchar(max)	No
	CancelType	Type of the cancel	Nvarchar(max)	No
Feedback	FeedbackID	Id of the feedback	Guid	No
	DriverServiceID	Id of the service that the feedback is belong to	Guid	No
	UserID	Feedback owner	Guid	No
	Score	Score	Int	No
	Content	Feedback in message	Nvarchar(max)	Yes
	CreateTime	Time when the feedback was created	DateTime	No
DriverCancelRequest	DriverCancelRequestID	Id of the request	Guid	No
	DriverServiceID	Id of the service	Guid	No
	OffDay	Day to be cancelled	DateTime	No
	CreateTime	Time when the request was created	DateTime	No
	Description	Cancel message	Nvarchar(max)	No
Address	AddressId	Id of the address	Guid	No
	AddressDetail	Name of the address	Nvarchar(max)	No
	Latitude	Latitude	Float	No
	Longitude	Longitude	float	No
UserAccount	*Username*	User nick name	Nvarchar(128)	No
	UserID	Id of the account	Guid	No
	Password	User password	Nvarchar(max)	No
	RoleID	User role	int	No
CustomerRequirementDetail	CustomerRequirementDetailID	Id of the detail	Guid	No
	CustomerRequirementID	ID of the original requirement	Guid	No
	ChildID	Id of the child linked to this detail	Guid	No
DeviceToken	DeviceTokenID	ID of the token	Guid	No
	UserID	Id of the owner	Guid	No
	Token	Token string	Nvarchar(max)	No
School	SchoolID	Id of the school	Guid	No
	Name	School name	Nvarchar(max)	No

	AddressID	ID of the detail linked to the school	Guid	No
Role	<u>RoleID</u>	Id of the role	int	No
	RoleName	Role name	Nvarchar(max)	no

5. Algorithms

5.1 Find matching Driver Service and Customer Requirement

To find the appropriate driver's services for customers, we suggest the following matching algorithm which goes through 4 following steps:

- Firstly, the school from driver's service and customer's requirement must be matched; otherwise, there's no matching between driver's service and customer's requirement.
- Secondly, the distance from customer's requirement for pick-up address must be within configured radius of driver's service's starting address; or else, there's no matching between driver's service and customer's requirement.
- Thirdly, the pick-up time of customer's requirement must be before or after a configured moment from driver's service's start time; or else, there's no matching between driver's service and customer's requirement.
- Finally, there's a matching rate between the days in week from customer's requirement and the days in week from driver's service. If the registered days are matched, a matching rate is calculated by the available days of the driver divided by the total days in the requirement of customer, considering that the available day is checked by comparing the available seats with the number of children required by customer in that day. If this rate is greater than or equal to a configured rate, the customer requirement would match the driver's service; or else, there's no matching between driver's service and customer's requirement.

Complexity: (2n)^2

Pseudocode:

```
Input the requirement
Call services


Initialize counter to zero
Initialize 'matching_school' services
While counter is less than length of services
If requirement's school is service's school
        Add service to 'matching_school' services
Add one to counter
Set counter to zero
```

```
Initialize 'matching_distance' services
Initialize configured distance
While counter is less than length of 'matching_school' services
If requirement's picking up address is not far from service's start address over configured distance
    Add service to 'matching_distance' services

Add one to counter
Set counter to zero

Initialize 'matching_time' services
Initialize configured time
While counter is less than length of 'matching_distance' services
If requirement's picking up time is before or after service's start time for within a configured time
    Add service to 'matching_time' services
Add one to counter
Set counter to zero

Initialize 'matching_days_of_week_and_capacity' services
Initialize configured rate
Initialize service-percentage to zero
Initialize adding-percentage to 100 percent over numbers of requirement's days in week
While counter is less than length of 'matching_time' services
If requirement's day in week is the same as service's day in week
        If service's available capacity is greater than or equal to requirement's number of children
        Add adding-percentage to service-percentage
```

Add one to counter

Set counter to zero

Return 'matching_days_of_week_and_capacity' services

5.2 Find shortest routes between multiple stops (based on Traveling Salesman)

Situation: In our system, the driver has to go to multiple places to pick up kids before going to school. Therefore, the system has a responsibility to provide suitable routes for driver to help him/her to travel easier.

Input: The starting point, the end point and all of the stop points during the trip, each point must have latitude and longitude.

Output: A sequence of points, from starting point, to every of stop points and to end point.

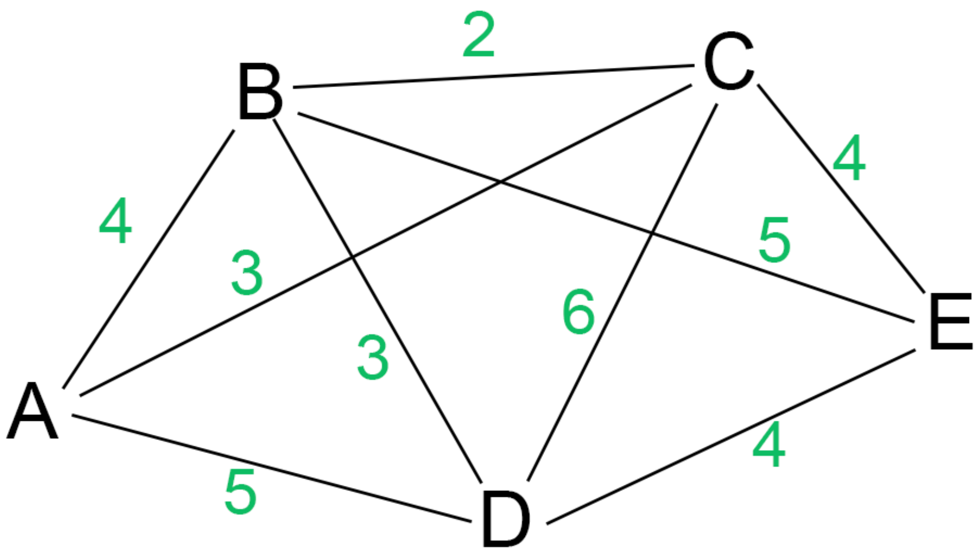
Implementation:

- ***Customized Dijkstra:***

We use Dijkstra algorithm to find the shortest way from starting point to end point. However, one limitation of this algorithm is that it only supports the shortest one without visiting every required point. We enhance this simply by adding a step to check if the current route has visited all of stop points yet.

To get started, the distances between all pairs of points must be calculated (except the distance between starting point and end point directly). These distances can be easy to get since the latitude and longitude are provided.

For example, we have a route from point A to point E, which has to visit point B, C, D. The below figure shows the distances among them.



We make a queue to contain some possible route. At first, the queue only has one element:

[A]

Then, take the first element of the queue, and find the next stop. For example, in this case, from A, we have 3 possible routes that is AB – 4, AC – 3 and AD – 5. Push all of them to queue after sorting:

[AC – 3, AB – 4, AD – 5]

Repeat previous step until done. However, in each route, we do not need to travel to visited points and we do not need to travel to stop point as well if all of stop points are not visited.

To be specific, in this case, from route AC – 3, we can generate ACB – 5 and ACD – 9, the queue now is:

[AB – 4, AD – 5, ACB – 5, ACD – 9]

Similar to that, the next queue is:

[AD – 5, ACB- 5, ABC – 6, ABD – 7, ACD – 9]

Then:

[ACB- 5, ABC – 6, ABD – 7, ADB – 8, ACD – 9, ADC – 11]

[ABC – 6, ABD – 7, ADB – 8, ACBD – 8, ACD – 9, ADC – 11]

[ABD – 7, ADB – 8, ACBD – 8, ACD – 9, ADC – 11, ABCD – 12]

[ADB – 8, ACBD – 8, ACD – 9, ADC – 11, ABCD – 12, ABDC – 13]

[ACBD – 8, ACD – 9, ADBC – 10, ADC – 11, ABCD – 12, ABDC – 13]

From here, we can find one of the shortest routes from ACBD – 8, that is ACBDE – 12. However, we need to checkout if any route is shorter than 12.

[ADBC – 10, ADC – 11, ACDB – 12, ABCD – 12, ABDC – 13]

We found another route, that is ADBCE – 14 from ADBC – 10. However, it is greater than ACBDE – 12.

[ACDB – 12, ABCD – 12, ABDC – 13, ADCB – 13]

Now we realize that all of elements in queue is not less than 12. So, the shortest route is ACBDE – 12.

With this algorithm, we definitely find the shortest route to solve the problem. However, this is the complete graph, the complexity for this solution is really costly. In the worst case, all of the available routes must be checked out to find the best one. Though, if we have n stops, we must travel n! possible routes, then, this algorithm shouldn't be used in this case.

- ***Greedy algorithm:***

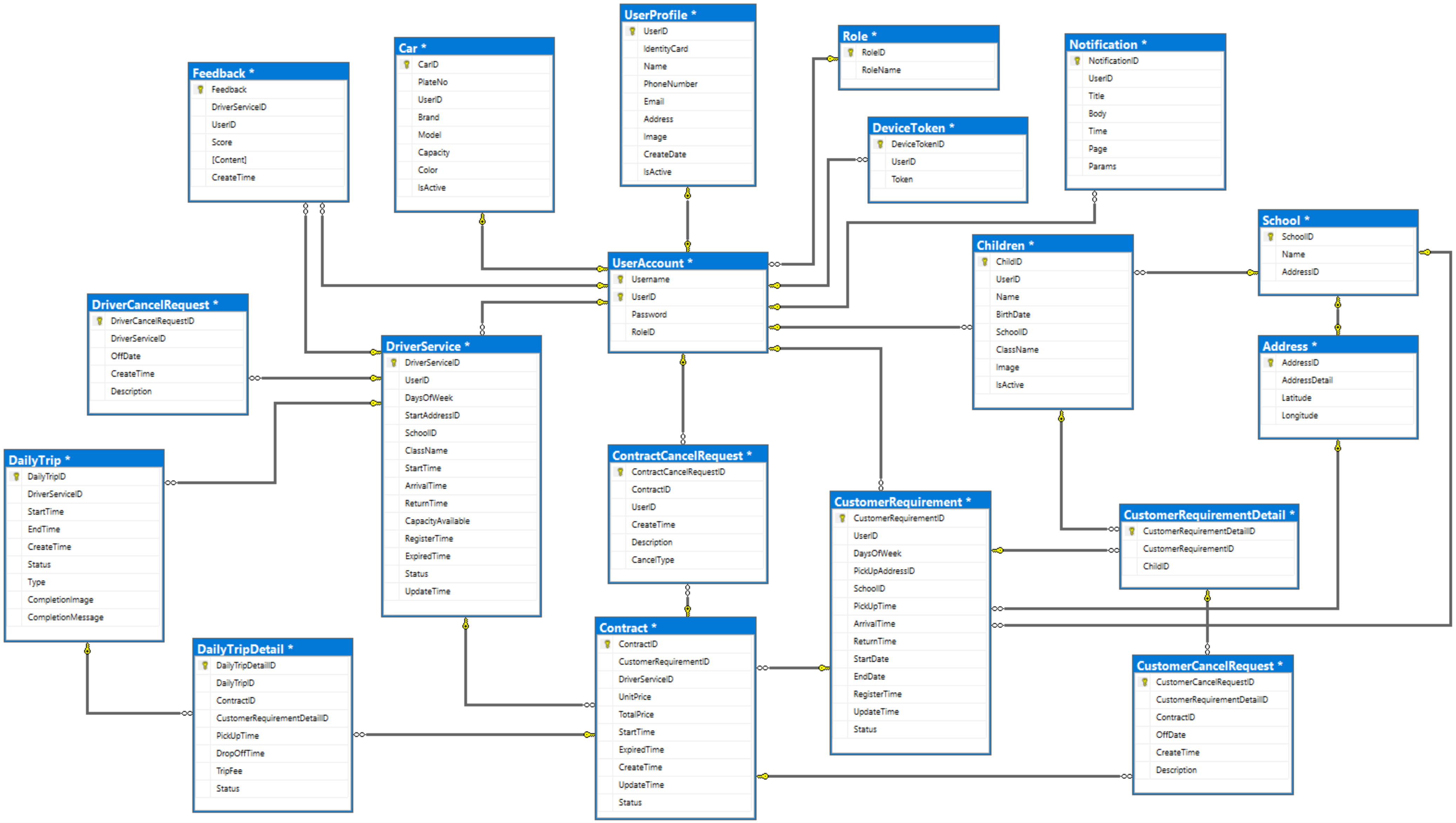
This algorithm can be stated simply as: Finding the nearest point from previous point and push to the list. Of course, the end point shouldn't be chosen if all of stop points are not visited.

This algorithm is fast enough, though it would not be the best route in many cases. However, the output is still usable.

In conclusion, we find two possible algorithms to apply in our problem. The first one has the best result, but it is slow in performance. On the contrary, the second one is fast but not effective. In real practice, we will apply the first one for the case there are no greater than 7 stops, and the second one for the rest. We choose 7 because 7! is 5040, so in the worst case, 5000 is acceptable. Moreover, our system expects drivers as people owning private cars, and the maximum slots for these cars are 7.

E. System Implementation & Test

1. Data Relationship Diagram



Entity Data Dictionary: Describe content of all tables		
No.	**Table Name**	**Description**
1	UserAccount	Contain account information.
2	UserProfile	Contain user profile.
3	Car	Contain car information of driver.
4	Role	Role for users.
5	DeviceToken	Device unique token for pushing notifications.
6	Notification	Contain user's notifications.
7	Children	Contain children information.
8	School	Contain school information.
9	Address	Contain address information.
10	Feedback	Contain user's feedbacks for driver.
11	CustomerRequirement	Contain customer requirement for a service.
12	CustomerRequirementDetail	Contain customer requirement detail.
12	DriverService	Contain driver service information.
13	Contract	Contain contract made by driver service and customer requirement.
14	DailyTrip	Contain daily trip of driver service.
15	DailyTripDetail	Contain daily trip detail information.
17	ContractCancelRequest	Contain requests for canceling a contract.
18	CustomerCancelRequest	Contain customer requests for canceling a daily trip in contract.
19	DriverCancelRequest	Contain driver requests for canceling a daily trip in contract.