National University of Computer and Emerging Sciences

- FAST Computer Science Department

Fundaments of Software Engineering
CS-2004

Project

Submitted to:

Ma'am Maheen Arshad

Submitted by:

1. Musaab Imran (20i-1794)
2. Muhammad Ismail Ramzan (20i-1941)
3. Muhammad Usman Shahid (20i-1797)

Contents

1. Introduction:

The project is the implementation of the bus management that is done manually by Mr. Naveed and his team. The idea was to make an application where they would have to put in less effort and manage everything systematically. We immediately started our work to make this project. In manual management, there was money loss, as many students didn't pay their fees and the was a lot of printing of bus cards. These all reasons summed up to become the real driving force and motivation for us to make the application.
The goals that motivated us to make the application:

- Save money with digital cards.
- Save the customer from money loss by digital checks on money matters.
- Less hustle and fights about seats as all will be selected beforehand.
- Human ease by making a digital management system.

2. Scope:

The project would cover the whole management of the fast bus system, not only managing but will also be fulfilling the uses of the students, drivers, and the admin. While showing the path, route, and travel details to the students, it would also give the driver detailed information about the travelers, routes, and picking points. The admin would be able to manage the accounts of students and drivers and update the salary and fee status for both drivers and students respectively. He would be able to check the bus's condition and **ANY** requirements and needs except the offered functionalities would be out of this project's scope. Any change in the given deliverables or core functionalities would cause a change in the schedule.

3. Stakeholders:

a. Customer
b. User (admin, driver, and students)
c. Developers
d. Designers
e. Project Manager
f. IMU Solutions

4. High-Level goals of Stakeholders:

Students:

- Login

Login functionality is used by all of the stakeholders. Login is used for the authentication of the users. Students would use this every day when they want to log in to the system to book a ride. Check for new users to make their accounts and add the information to the database. If the user is already registered then allow him/her to enter.

- Register

Registration requires much information which would allow the users to log in and use the account. The student and driver would be able to register their account into the system. They would enter the required information and their account will be registered for further use the password should be strong. Confirmation code check which is sent in the mail.

- Book ride

Booking a ride is the functionality of the user/ student. Students would be given any information that would be required for travel. Without the boarding number, students mustn't be allowed to sit in the vehicle. A boarding list should be generated against the name of different students. They booked seats that shouldn't be shown or allowed to be rebooked. They should be locked.

- Travel details

The student can view all of the different information after booking the ride. They can view the list of information related to the travel like bus number, driver number, and the route. This functionality is required for students so, they can contact the driver in case of emergency.

- Check fee details

The students can check their fee status to see if their fee has been accepted or not. If not, they won't be able to book a ride. If they have paid the fee, they can book the ride.

Driver:

- Login

Login functionality is used by all of the stakeholders. Login is used for the authentication of the users. Students would use this every day when they want to log in to the system to book a ride. Check for new users to make their accounts and add the information to the database. If the user is already registered then allow him/her to enter.

- Register

Registration requires much information which would allow the users to log in and use the account. The student and driver would be able to register their account into the system. They would enter the required information and their account will be registered for further use the password should be strong. Confirmation code check which is sent in the mail.

- Travel details

Travel details are also important for the driver because the driver needs to have all of the information which is required for the travel. In case students are unable to reach the bus so, the drivers can contact the students.

- Check salary

The drivers would be able to check the status of their salary so that they can get it from the bank. This message will be updated by the admin so he can inform the drivers about their salary.

Admin:

- Login

Login functionality is used by all of the stakeholders. Login is used for the authentication of the users. Students would use this every day when they want to log in to the system to book a ride. Check for new users to make their accounts and add the information to the database. If the user is already registered then allow him/her to enter.

- Manage accounts

The driver can manage the accounts of students as he can add and delete the account of the students based on the circumstances. If the student has left the account will be deactivated Once the account is deleted the particular user can't access them afterward there is a new student, he/she would, and have a new account for the travel. Once the account is created, they should be allowed to change the passwords. This would be a type of default account.
The admin should also have the access of student and driver information for the record.

- Manage finance

The admin is able to update the fee and salary status of student and drivers respectively. If students have paid there fee they can book ride and if drivers salary has been updated he can get it from the bank.

- Manage bus

Managing the bus is also an important task for the admin as he has to check the quality of the buses. If one month is completed after the maintenance of the bus then it should be shown on the allotment list. There should be a list named "maintenance" which must include buses that need immediate maintenance. Daily reminders should be given unless the maintenance of the bus is done.

5. List of features:

Requirement #	Description
1	Login system for students, bus drivers, and admin.
2	Chalan status for students and salary confirmation for the bus drivers.
3	Seat booking and boarding number generation for students.
4	Travel information, details like bus and driver numbers, and time mentioned routes on maps.
5	Passenger information log, showing names, tokens, and picking points to the driver.
6	Students and drivers register who are traveling for the first time against their Gmail id.
7	Daily bus and route allotment by the admin.
8	Account deletion of students and drivers who have left.
9	And adding new students and drivers, also creating their accounts.
10	Maintaining bus data, also adding and removing the buses.
11	Log out of the account after the user has used stopped using the app.

6. Use case identification:

- Login
- Register
- Book ride
- Travel details
- Check fee details
- Manage student information
- Manage finance
- Check salary
- Manage accounts
- Manage bus

After applying the three tests:

1. **Boss Test**
2. **Size Test**
3. **Elementary Business Process (EBP) Test**

The use cases were shortlisted.

The top five useful use cases are now:

- Book ride
- Manage student information
- Manage finance
- Manage accounts
- Manage buses

7. High-Level Use Cases:

US01:	
Use case:	Login
Actors:	Student, driver, admin
Type:	Primary
Description:	The user(student/admin/driver) wants to log in to the app to book the ride, manage the app, or check the ride status. When he/she enters the right credentials, the system allows them to use the app and perform the activity.

US02:	
Use case:	Register
Actors:	Student, driver
Type:	Primary
Description:	The student and driver would be able to register their account into the system. They would enter the required information and their account will be registered for further use.

US03:	
Use case:	Book ride
Actors:	Student
Type:	Primary
Description:	The student can book the ride. He/she should enter the required information into the system and their ride would be booked. He/ she can also redefine the path (picking point).

US04:	
Use case:	Travel details.
Actors:	Student
Type:	Primary
Description:	When the student enters the booking information, all travel information, including maps, numbers, route, driver, and bus details, will be shown to the student. After this, he could have a confirmation about his/her ride.

US05:	
Use case:	Check fee details
Actors:	Student
Type:	Primary
Description:	The student can check the status of his/her fee. If the fee is paid, he/she can book the ride else, a 2-day ultimatum would be shown to that person, or else he would not be able to book the ride.

US06:	
Use case:	Manage student information
Actors:	Driver and admin
Type:	Primary
Description:	When the student enters the information to book the ride, the booking/traveling information is stored and displayed to the driver. This way the driver has the picking points and contact information of all the same route students.

US07:	
Use case:	Manage finance
Actors:	Admin
Type:	Primary
Description:	The admin should be able to manage all the financial status, this includes the student fee, driver's salary, and the revenue. This way he can check all the money matters.

US08:	
Use case:	Check salary
Actors:	Driver
Type:	Primary
Description:	The driver logs in to the app and checks the status of his salary and whether it has been transferred to the bank or not.

US09:	
Use case:	Manage accounts
Actors:	Admin
Type:	Primary
Description:	The admin manages the accounts of drivers and students. He can either add or delete the accounts based on addition or removal.

US010:	
Use case:	Manage buses
Actors:	Admin
Type:	Primary
Description:	The admin manages the buses by having a regular reminder message for the bus service. He can also allot drivers to a particular bus with a particular route.

8. Extended Use Case:

a. Book ride scenario

Use case section	Description
Use case name: (UC01)	Book ride.
Scope	The system is under design and its application (Booking ride functionality).
Level	User goal.
Primary actor	Student
Priority	High
Stakeholders and Interests	Student: wants easy and fast booking of the daily ride to the university, without any technical errors. Admin: wants to save all the information for managing the routes. Driver: wants to have data for picking points and contact information.
Preconditions	Students must be identified and authenticated.
Postconditions	• Ride is booked • Address is saved • Travel details are displayed
Extensions	At any time, the System fails: 1a. The student must contact the admin or the numbers provided in the HELP. 2a. The student must try to close the app and log in again.
Special requirement	• Text must be visible • UI should be simple and understandable • Travel details display within 5-10 seconds • Robust recovery when booking the ride is failing.

<table>
<tr><td>Main success scenario</td><td>
<table>
<tr><th>Actor Action:</th><th>System Responsibility:</th></tr>
<tr><td>1. The student enters the required information.</td><td></td></tr>
<tr><td></td><td>2. The system checks the validation of the information, especially the picking point.
3. If the input is valid the travel details are displayed to the student.</td></tr>
<tr><td>4. The student can see all of the travel information.</td><td></td></tr>
<tr><td></td><td>5. The system sends all of the information to the admin and driver.</td></tr>
</table>
</td></tr>
<tr><td>Technology and Data Variations List</td><td>• Location entered is correctly mapped. (Location check)
• Fee status checker. (Fee receipt checker).</td></tr>
<tr><td>Frequency of occurrence</td><td>• This use case is continuous.</td></tr>
<tr><td>Miscellaneous/ Open issues</td><td>• Can the students have information about other students?
• What if the student is unable to book the ride?
• How to inform in case of any delay in bus arrival?</td></tr>
</table>

b. Manage Finance scenario

Use case section	Description
Use case name: (UC02)	Manage finance.
Scope	The system is under design and its application (finance functionality).
Level	User goal.
Primary actor	Admin
Priority	High
Stakeholders and Interests	Student: wants to check whether his/her fee has been paid or not and is updated on the app. Admin: wants to save all the financial information of students, drivers, and savings. Driver: wants to know when his salary will be available in the bank.
Preconditions	Admin must be identified and authenticated.
Postconditions	• The fee deposit information is received by the admin. • All the money information is updated from the bank. • The fee status is updated. • The salary status is updated.
Extensions	At any time, the System fails: 1a. The admin must contact the developers or the numbers provided. 2a. The admin must try to close the app and log in again. At any time, the bank doesn't respond: 1a. The admin should contact the bank directly by the provided contact information. At any time, the student doesn't deposit dee: 1a: The admin should jam the account of the student.
Special requirement	• Text must be visible • UI should be simple and understandable • The bank updates should be within 5-10 mins. • Robust recovery when bank contact is failing.

<table>
<tr><td>Main success scenario</td><td>
<table>
<tr><th>Actor Action:</th><th>System Responsibility:</th></tr>
<tr><td>2. The admin checks with the bank the fee status of students.</td><td></td></tr>
<tr><td></td><td>2. The system responds to queries made by the admin.</td></tr>
<tr><td>3. The admin gives an ultimatum to those who haven’t submitted their fee.
4. Even after the ultimatum if they have submitted the fee, the admin jams their accounts.</td><td></td></tr>
<tr><td></td><td>5. The system either gives an ultimatum to that particular student’s account or freezes the booking functionality.</td></tr>
<tr><td>6. The admin tells the system to show the salary status of drivers.</td><td></td></tr>
<tr><td></td><td>7. The system displays the salary status message.</td></tr>
</table>
</td></tr>
<tr><td>Technology and Data Variations List</td><td>• Fee status checker. (Fee receipt checker).
• Remote bank checking facility.</td></tr>
<tr><td>Frequency of occurrence</td><td>• This use case is continuous.</td></tr>
<tr><td>Miscellaneous/ Open issues</td><td>• What will the admin do if he/she cannot contact the bank?
• What if the admin is not able to update the fee and salary status?</td></tr>
</table>

c. Manage Accounts scenario

<table>
<tr><th>Use case section</th><th>Description</th></tr>
<tr><td>Use case name: (UC03)</td><td>Manage accounts.</td></tr>
<tr><td>Scope</td><td>The system is under design and its application (account management functionality).</td></tr>
<tr><td>Level</td><td>User goal.</td></tr>
<tr><td>Primary actor</td><td>Admin</td></tr>
<tr><td>Priority</td><td>High</td></tr>
<tr><td>Stakeholders and Interests</td><td>Student: wants to register for the first time.
Admin: wants to hire or fire a driver, make changes to their accounts, and delete or add the accounts of students.
Driver: wants to register for the first time.</td></tr>
<tr><td>Preconditions</td><td>Admin must be identified and authenticated.</td></tr>
<tr><td>Postconditions</td><td>• The account is added.
• The account is updated.
• The account is deleted.</td></tr>
<tr><td>Main success scenario</td><td>
<table>
<tr><th>Actor Action:</th><th>System Responsibility:</th></tr>
<tr><td>1. The admin adds a person (driver or student) and makes a default account.</td><td>2. The system makes the account of that person.</td></tr>
<tr><td>3. The admin removes a person (driver or student) and deletes their account.</td><td>4. The system deletes the account of that person.</td></tr>
<tr><td>5. The admin wants to make changes to a person’s account.</td><td>6. The system changes the account of that person.</td></tr>
</table>
</td></tr>
</table>

Extensions	At any time, the System fails: 1a. The admin must contact the developer or the numbers provided in the HELP. 2a. The admin must try to close the app and re-login.
Special requirement	• Text must be visible • UI should be simple and understandable. • The admin can communicate with students/drivers. • Robust recovery when booking the ride is failing.
Frequency of occurrence	• This use case is continuous.
Miscellaneous/ Open issues	• What if the admin is not able to add/remove the account of the student/driver? • What if the admin is not able to make changes to the accounts of students/drivers?

d. Manage Bus scenario

Use case section	Description
Use case name: (UC04)	Manage buses.
Scope	The system is under design and its application (account management functionality).
Level	User goal.
Primary actor	Admin
Priority	High
Stakeholders and Interests	Admin: wants to update the service status of the buses and every route updating and bus allotment to the drivers. Driver: wants to know which bus and route he would be taking the next day.
Preconditions	Admin must be identified and authenticated.

<table>
<tr><td>Postconditions</td><td>

The drivers have been allotted to the respective busses.
The routes of the buses have been scheduled.
Th definite picking points are marked.

</td></tr>
<tr><td>Main success scenario</td><td>
<table>
<tr><th>Actor Action:</th><th>System Responsibility:</th></tr>
<tr><td>1. The admin ask the system for all the selected route.</td><td></td></tr>
<tr><td></td><td>2. The system shows all the defined routes.</td></tr>
<tr><td>3. The admin selects all the routes and allots the driver to a specific route.</td><td></td></tr>
<tr><td></td><td>4. The system allots all the buses to the specific driver.</td></tr>
<tr><td>5. The admin wants to check the bus service status.</td><td></td></tr>
<tr><td></td><td>6. The system shows the status of all the bus services.</td></tr>
</table>
</td></tr>
<tr><td>Extensions</td><td>At any time, the System fails:
1a. The admin must contact the developers or the numbers provided in the HELP.
2a. The admin must try to close the app and log in again.
3a. If the routes don’t get updated the admin should go with the previous schedule and then contact the developer’s team.</td></tr>
<tr><td>Special requirement</td><td>

Text must be visible
UI should be simple and understandable
Buses details display within 5-10 seconds
Robust recovery when booking the ride is failing.

</td></tr>
<tr><td>Technology and Data Variations List</td><td>

Location (picking points) are saved and validated.
Best communication channel possible to communicate the route details to students and drivers.

</td></tr>
<tr><td>Frequency of occurrence</td><td>

This use case is continuous.

</td></tr>
<tr><td>Miscellaneous/ Open issues</td><td>

Can the admin change the default routes and update them?
What will the admin do if the bus service status is not updated?

</td></tr>
</table>

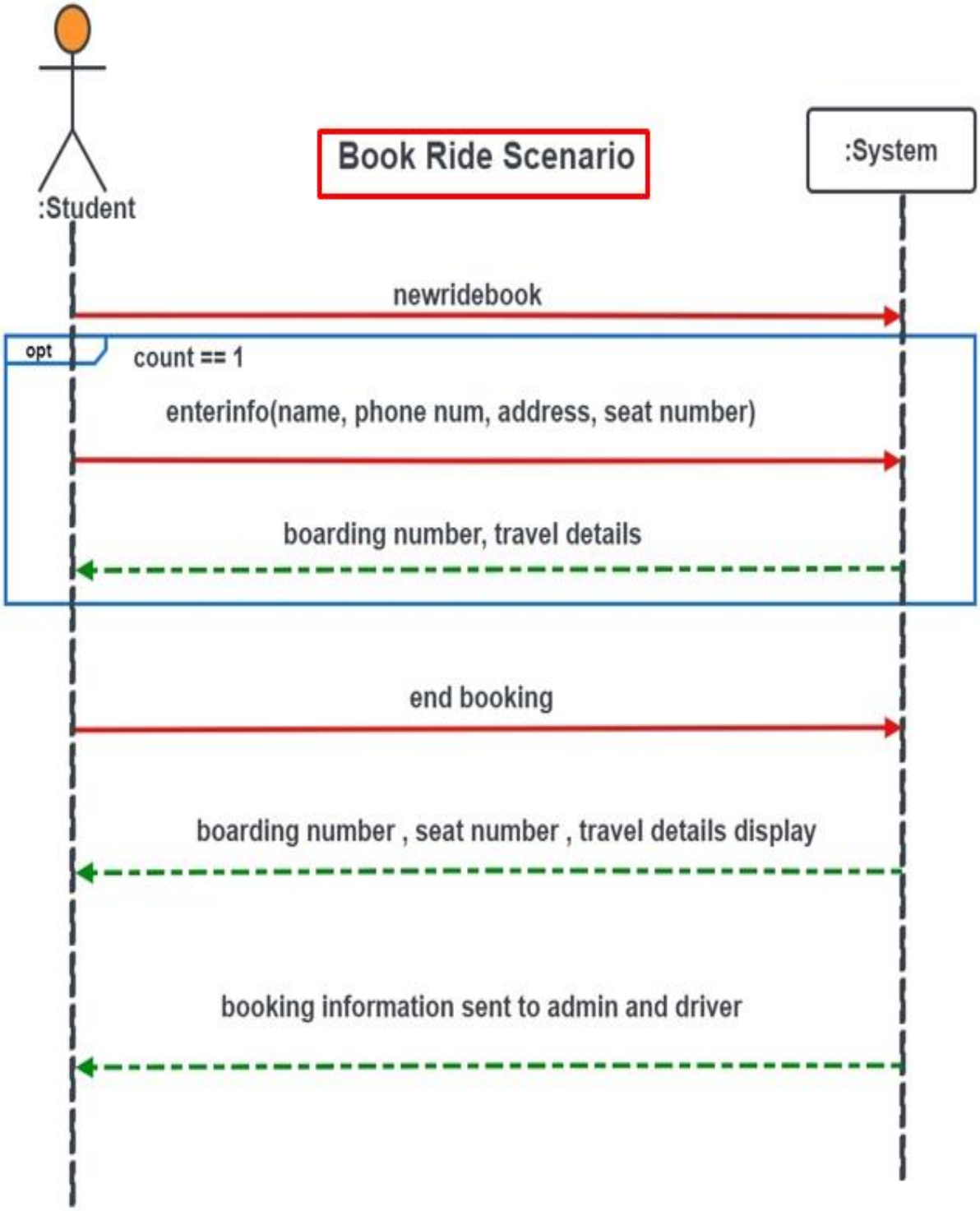
e. Travel Details scenario

Use case section	Description
Use case name: (UC05)	Travel details.
Scope	The system is under design and its application (account management functionality).
Level	Subfunction.
Primary actor	Admin, student, driver.
Priority	High
Stakeholders and Interests	Admin: wants to set the map for the route setup for the next day. Driver: wants to know which students are going to be traveling in his bus. He should have the boarding numbers, seat numbers, and contact information of all the travelers. Students: wants to see all the travel information for the next day's travels. He/she should be also able to see all of the available picking points along the route of the university.
Preconditions	Admin must be identified and authenticated.
Postconditions	• The drivers can see all of the information of the travelers. • The students can see all of the information about the next day's travel. • The admin can set the maps one night before the next day's travel.
Extensions	At any time, the System fails: 1a. The admin must contact the developers or the numbers provided in the HELP. 2a. The admin must try to close the app and log in again. 3a. If the routes don't get updated the admin should go with the previous schedule and then contact the developer's team. 4a. Check your internet connection.
Special requirement	• Text must be visible • UI should be simple and understandable • Buses details display within 5-10 seconds • Robust recovery when booking the ride is failing.

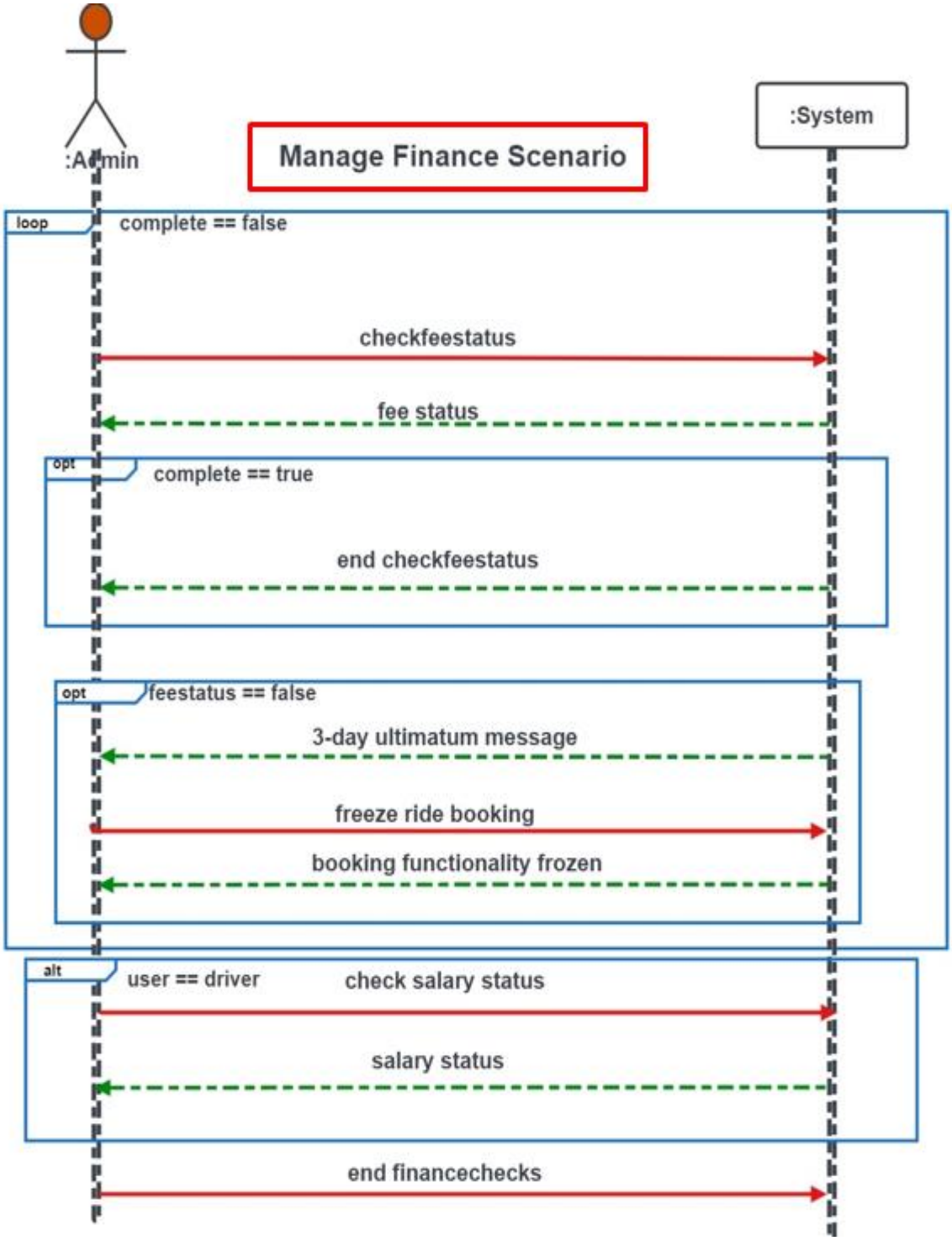
<table>
<tr><td>Main success scenario</td><td>
<table>
<tr><th>Actor Action:</th><th>System Responsibility:</th></tr>
<tr><td>1. The student enters all of the information regarding the trip.</td><td></td></tr>
<tr><td></td><td>2. The system shows all the defined routes.
3. The system sends all of the information to the driver.</td></tr>
<tr><td>4. The driver asks the system for all of the route details.</td><td></td></tr>
<tr><td></td><td>5. The system shows all of the information to the driver.
6. The system also send all of the information to the admin.</td></tr>
</table>
</td></tr>
<tr><td>Technology and Data Variations List</td><td>• Location (picking points) are saved and validated.
• Best communication channel possible to communicate the route details to students and drivers.</td></tr>
<tr><td>Frequency of occurrence</td><td>• This use case is continuous.</td></tr>
<tr><td>Miscellaneous/ Open issues</td><td>• Can the admin change the default routes and update them?
• What will the admin do if the bus service status is not updated?</td></tr>
</table>

9. System sequence diagrams:

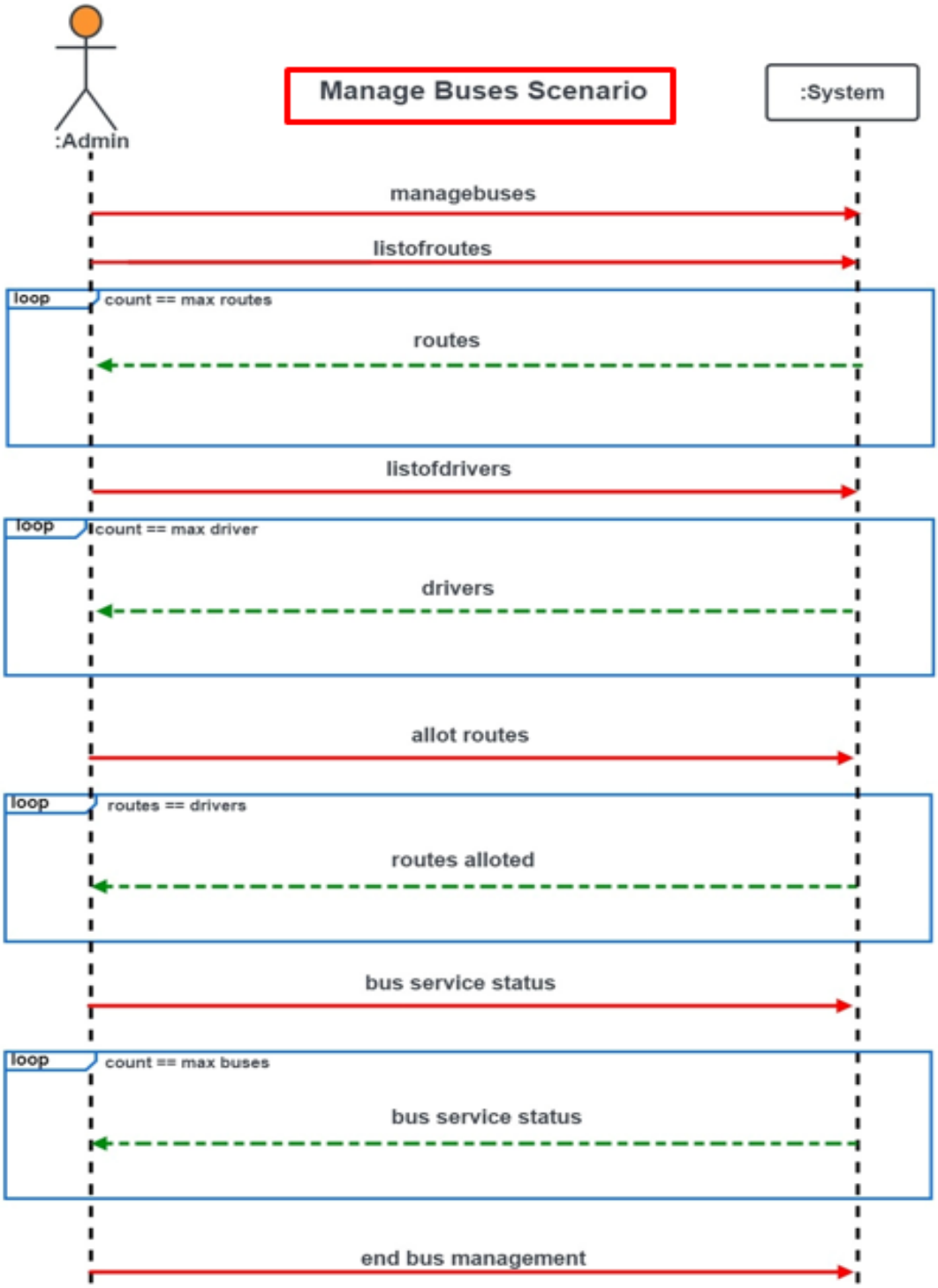
a. Book ride scenario



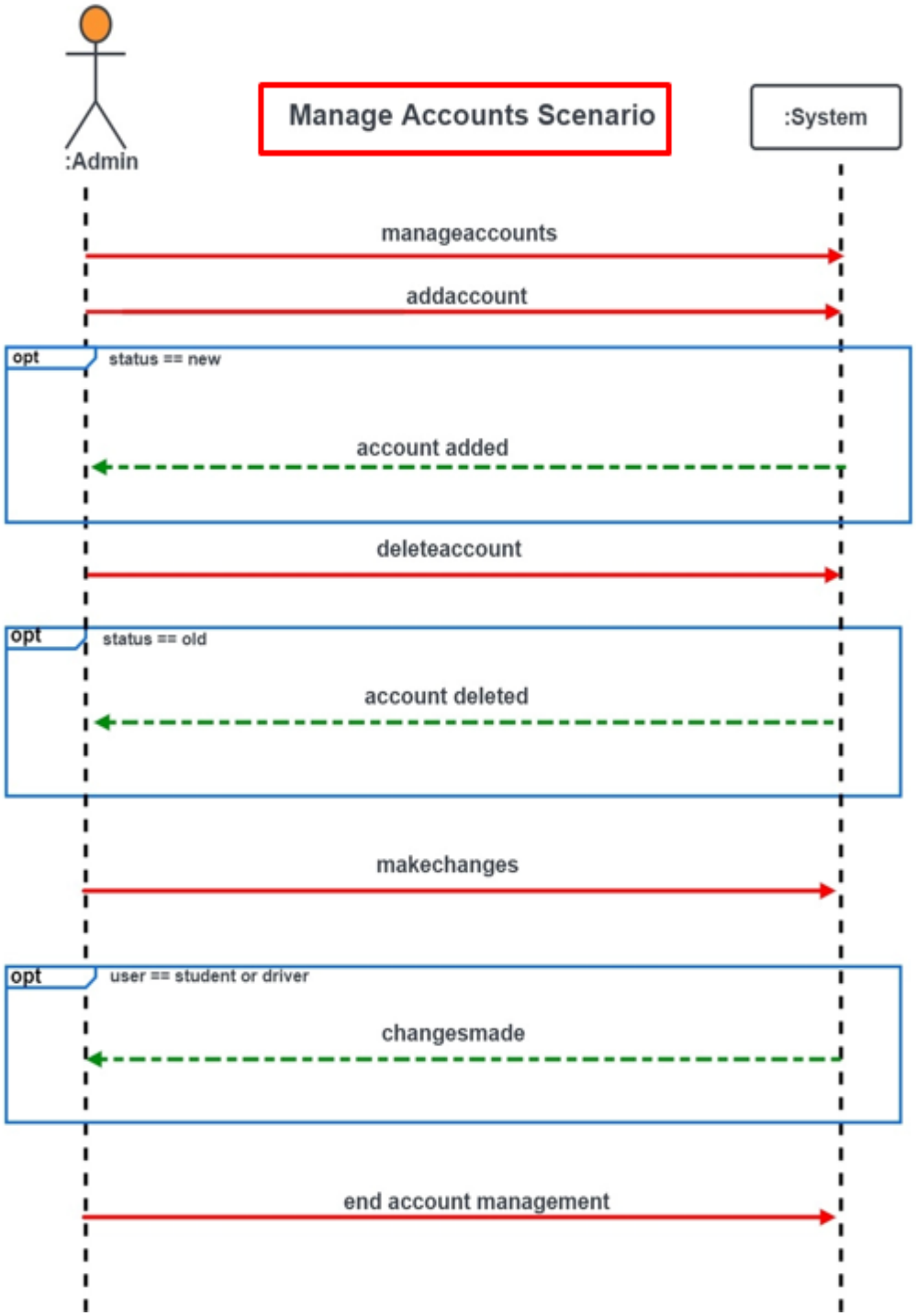
b. Manage Finance scenario



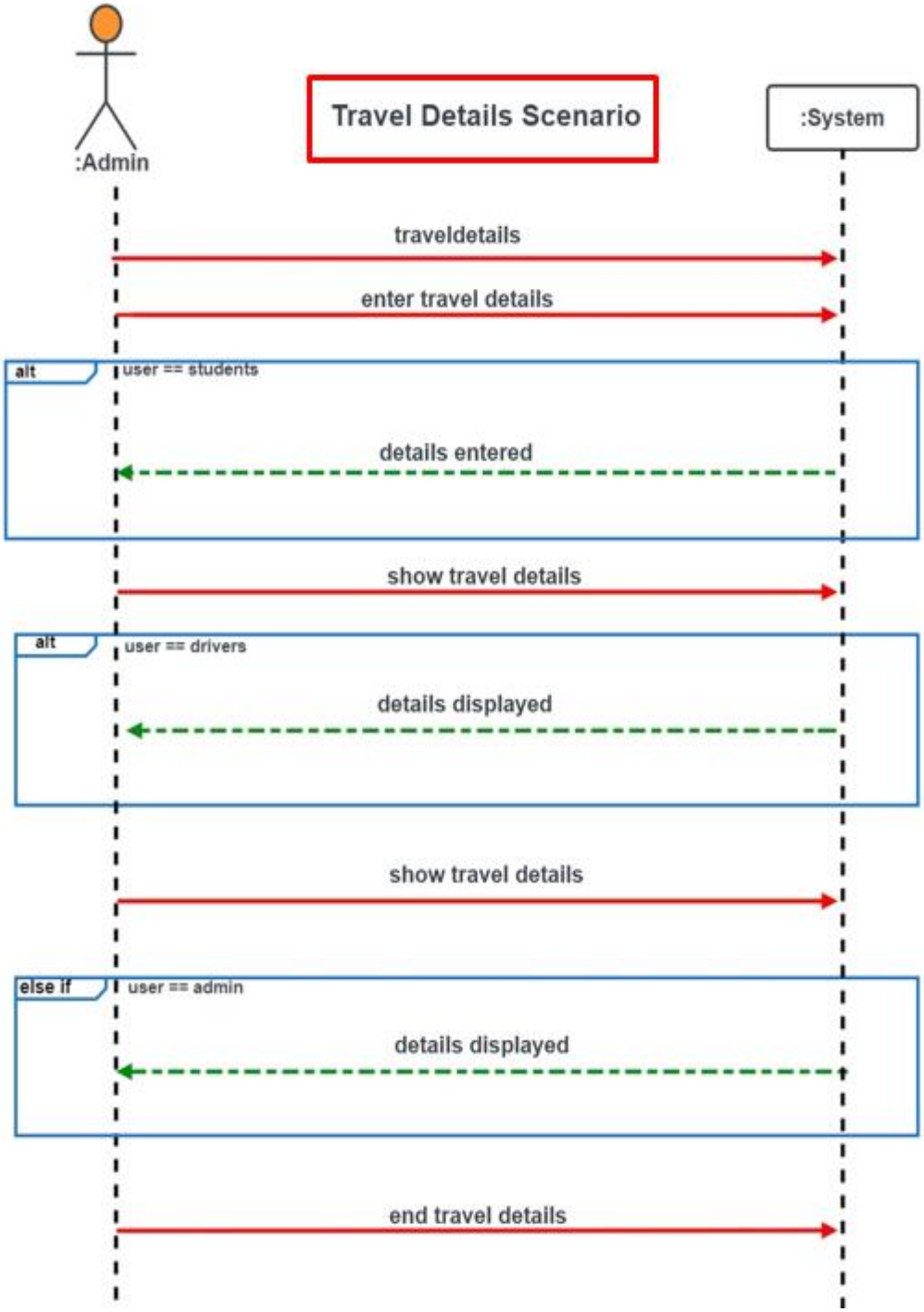
c. Manage Bus scenario



d. Manage Accounts scenario

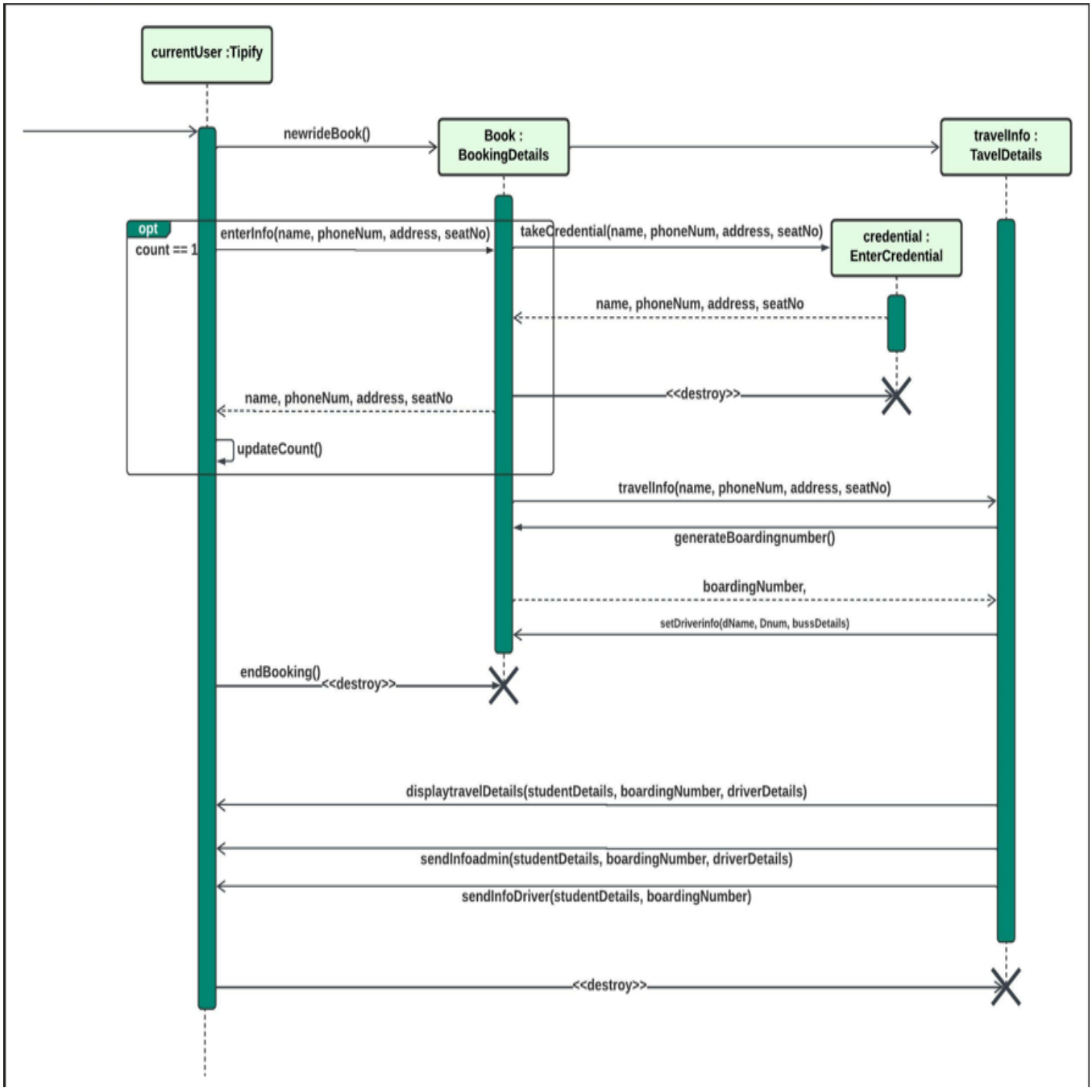


e. Travel Details scenario

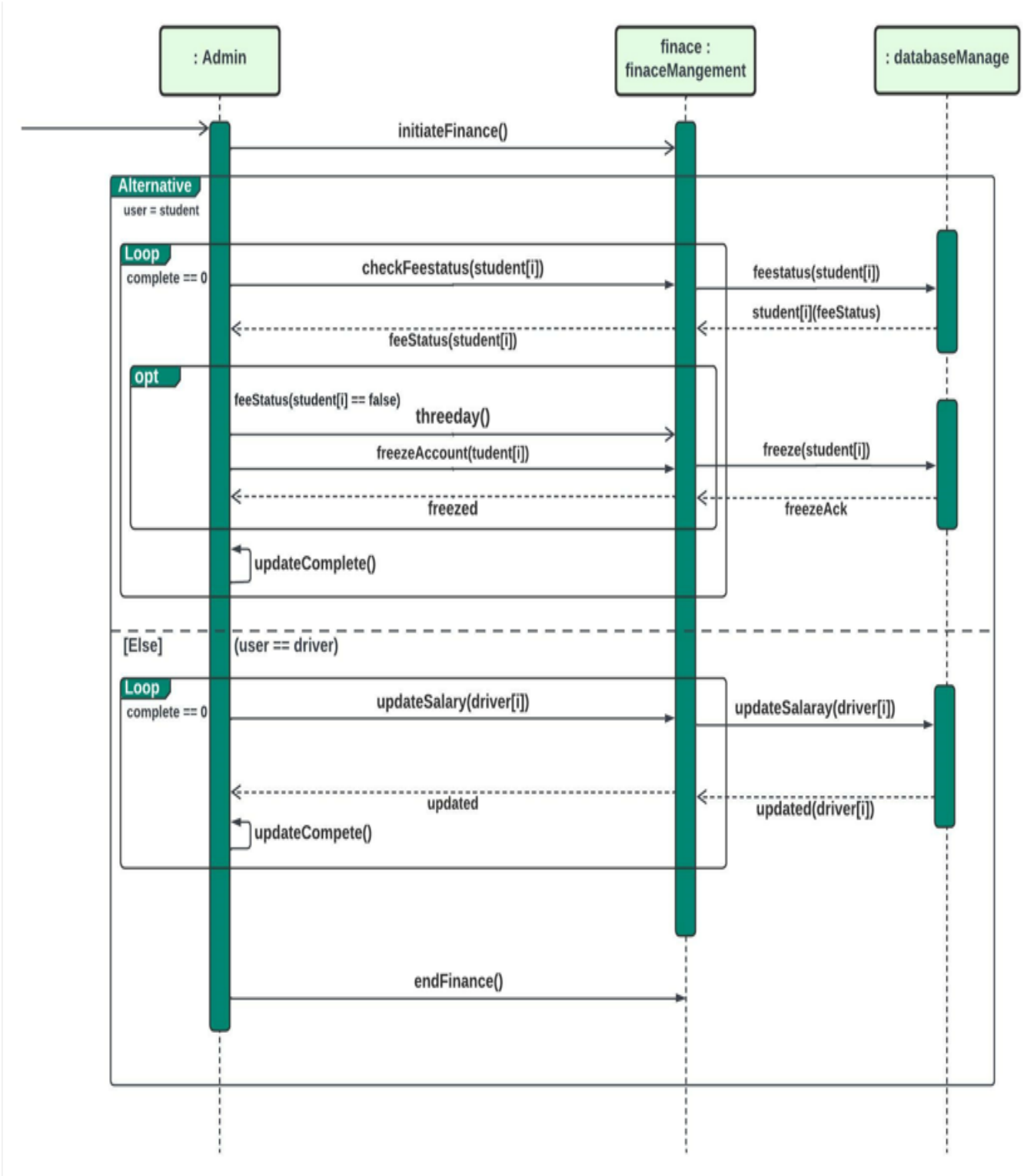


10. Sequence diagrams:

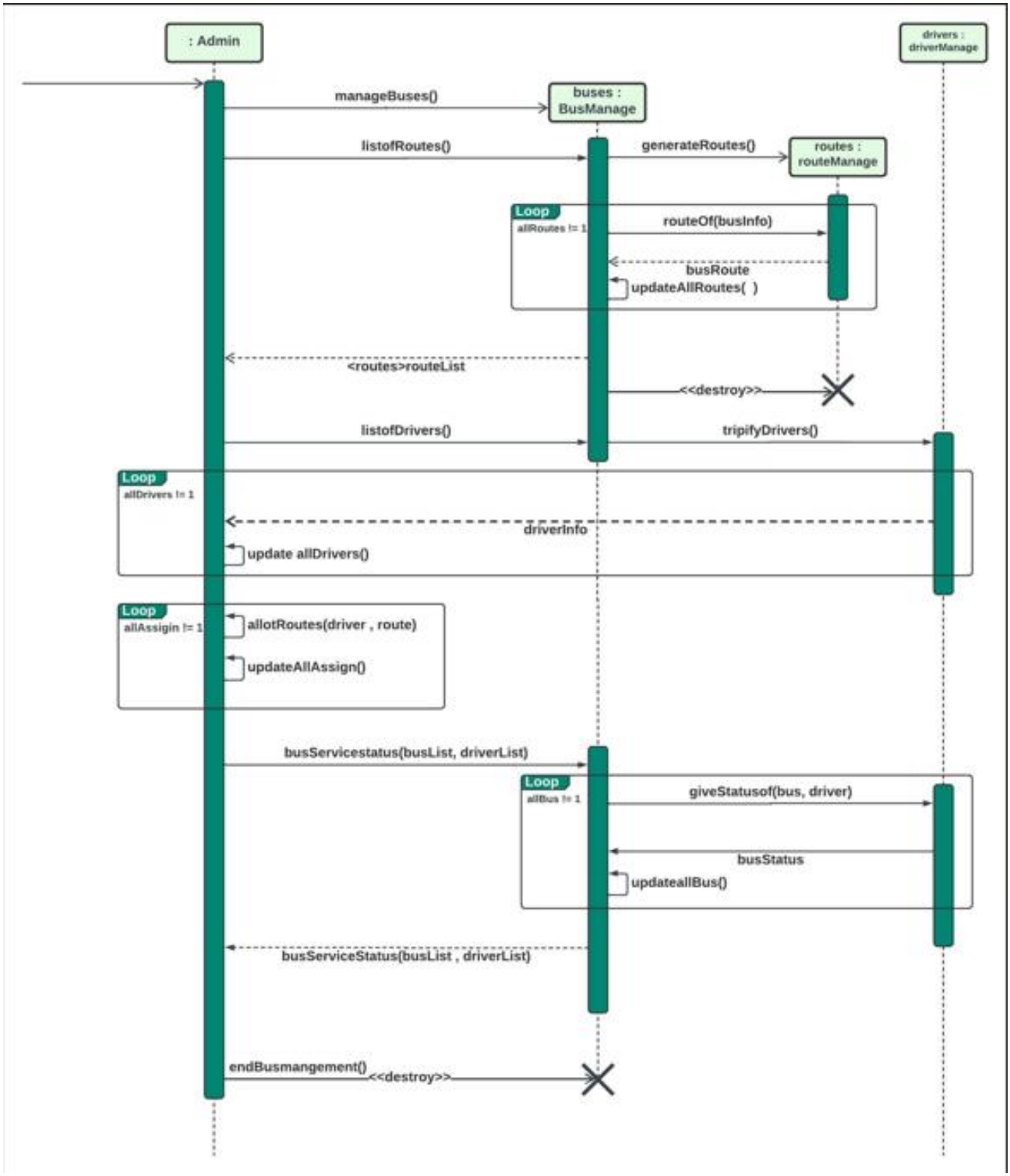
a. Book ride scenario



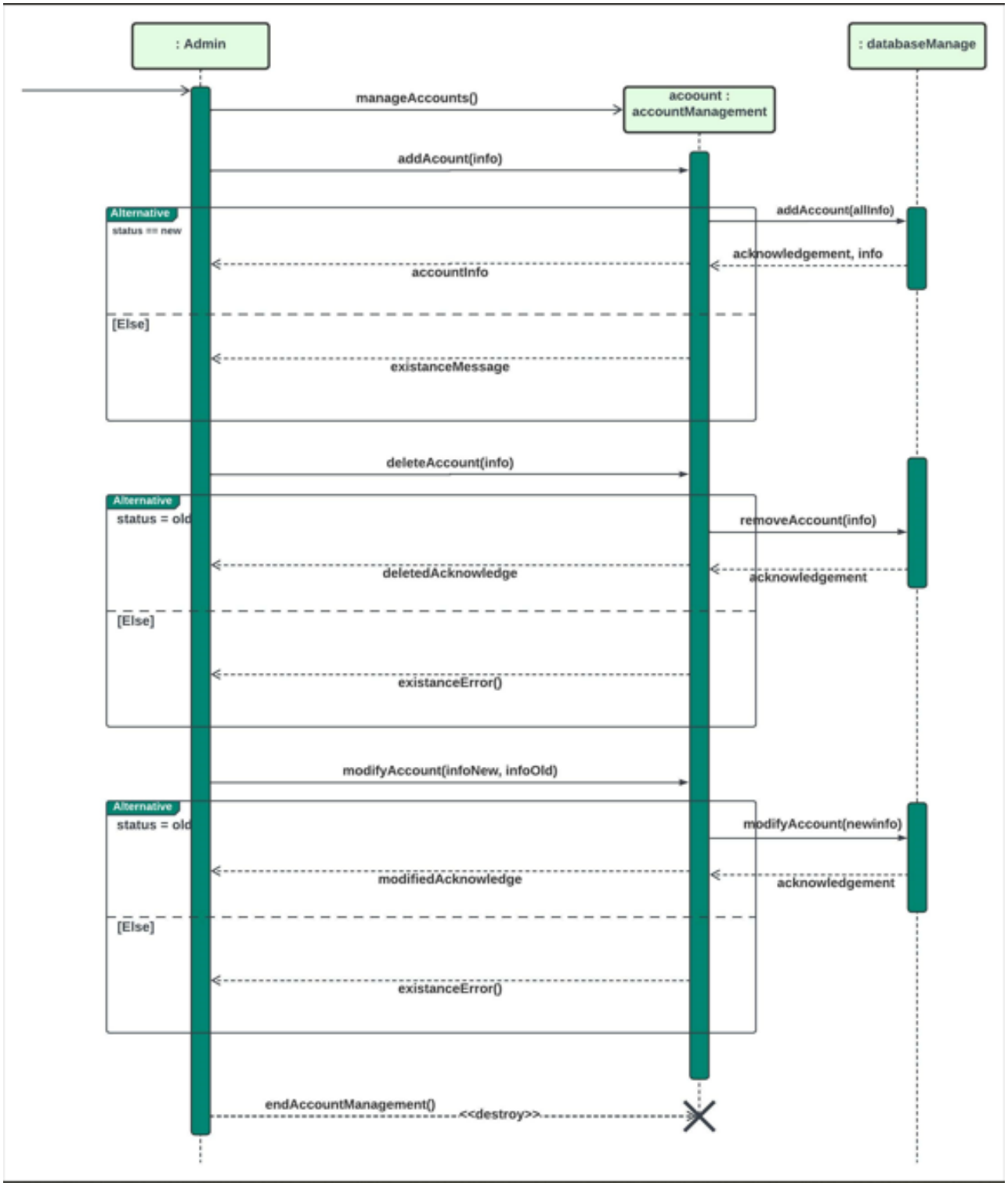
b. Manage Finance scenario



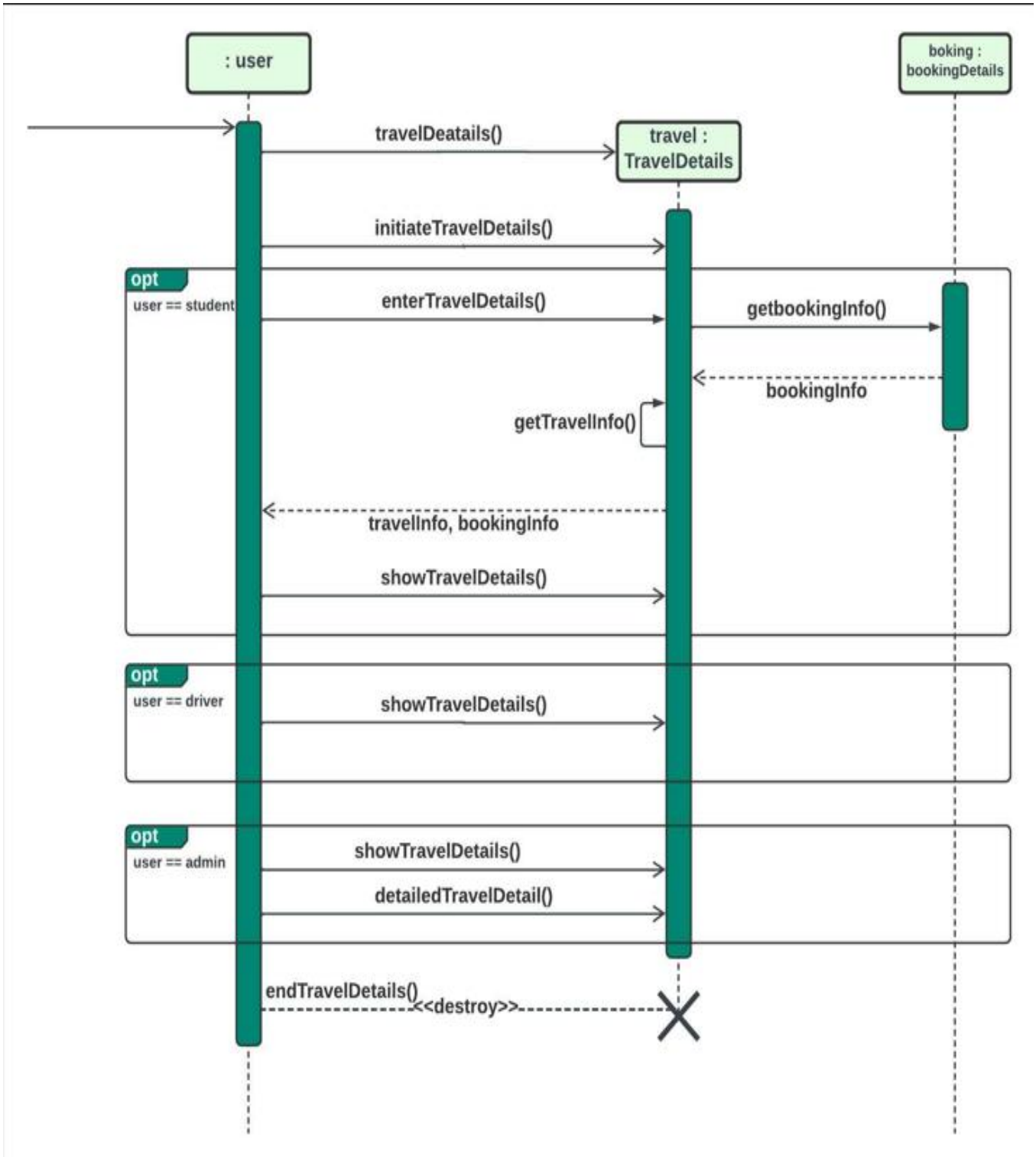
c. Manage Bus scenario



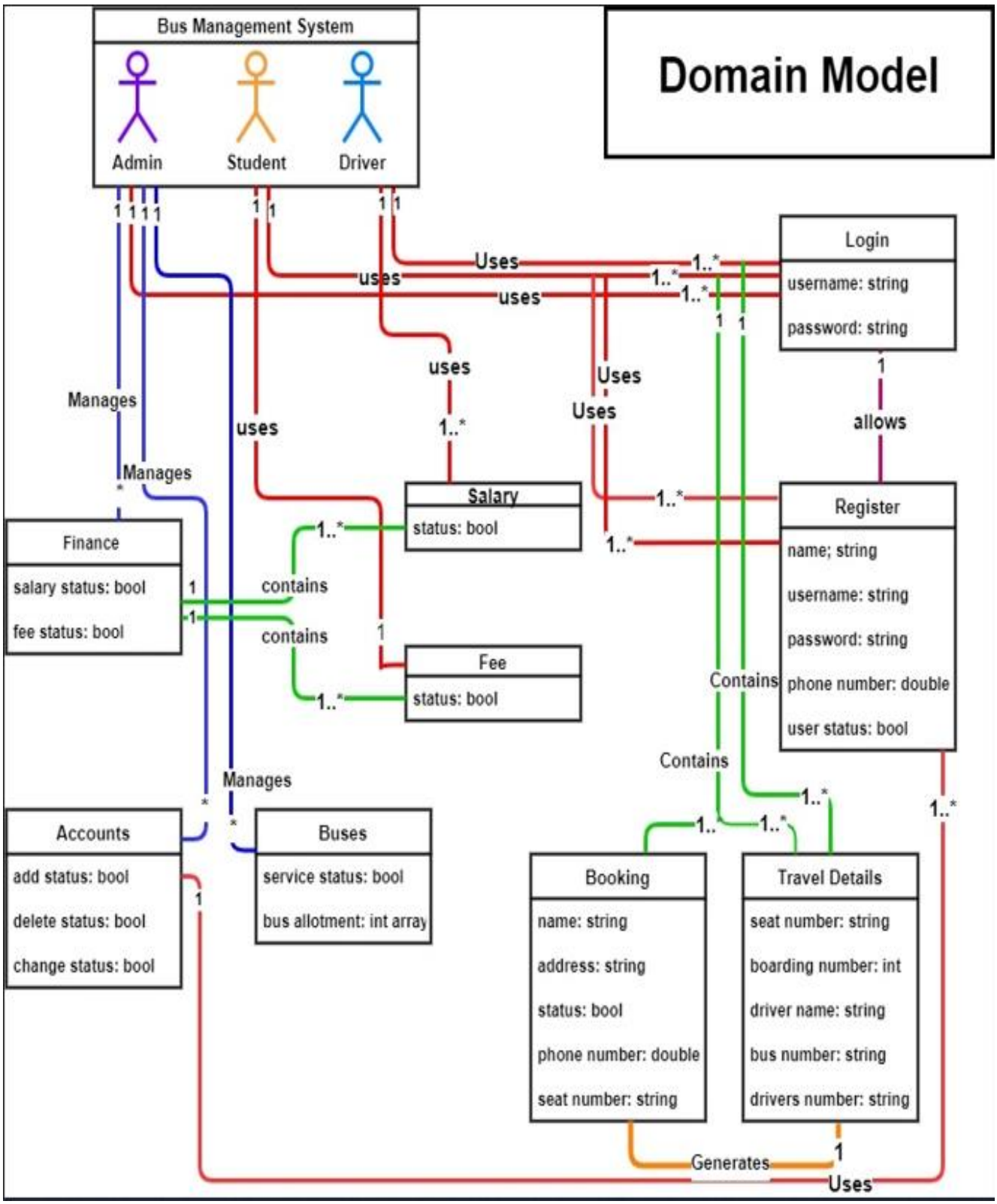
d. Manage Accounts scenario



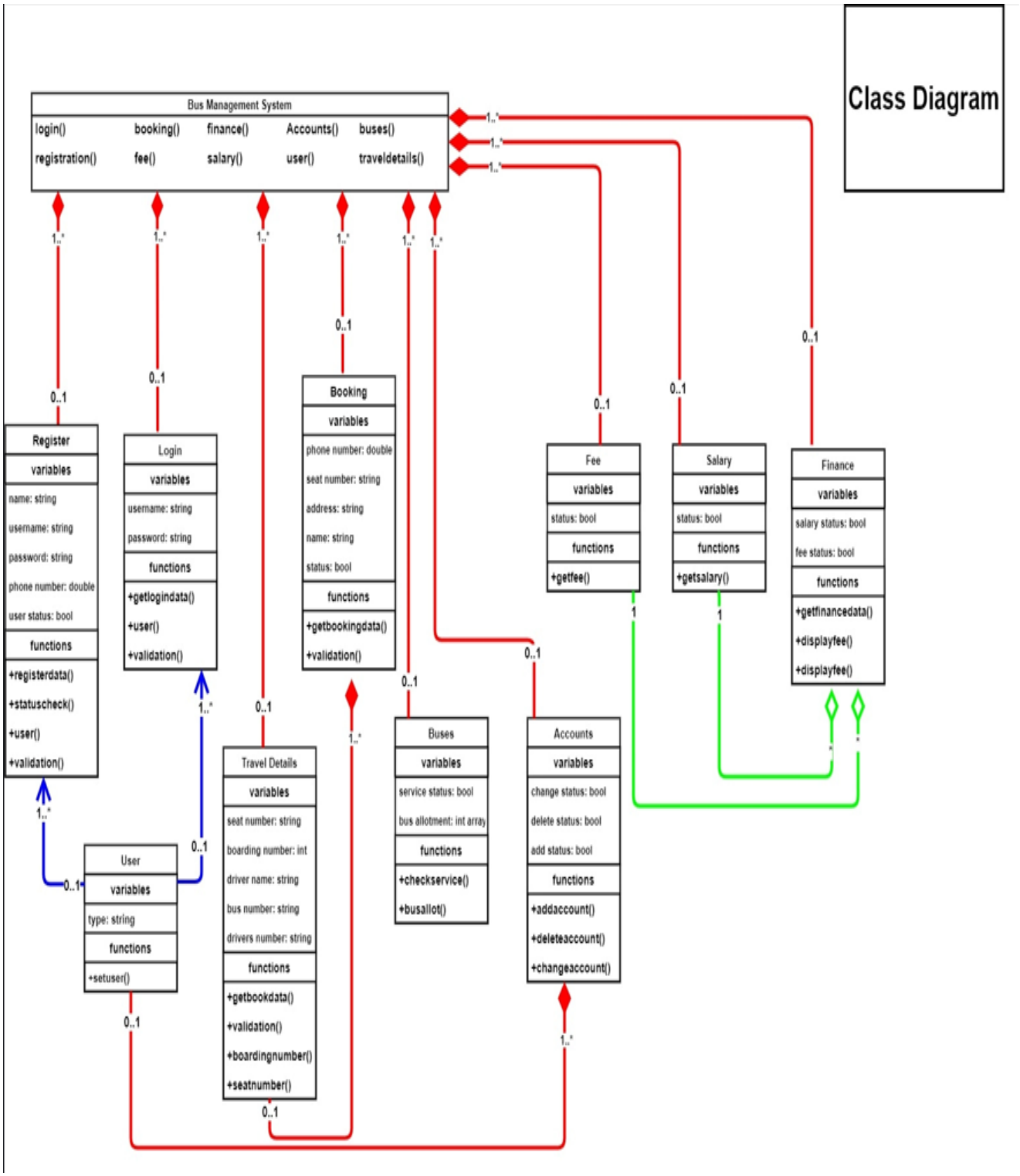
e. Travel Details scenario



11. Domain Model:



12. Class Diagram:

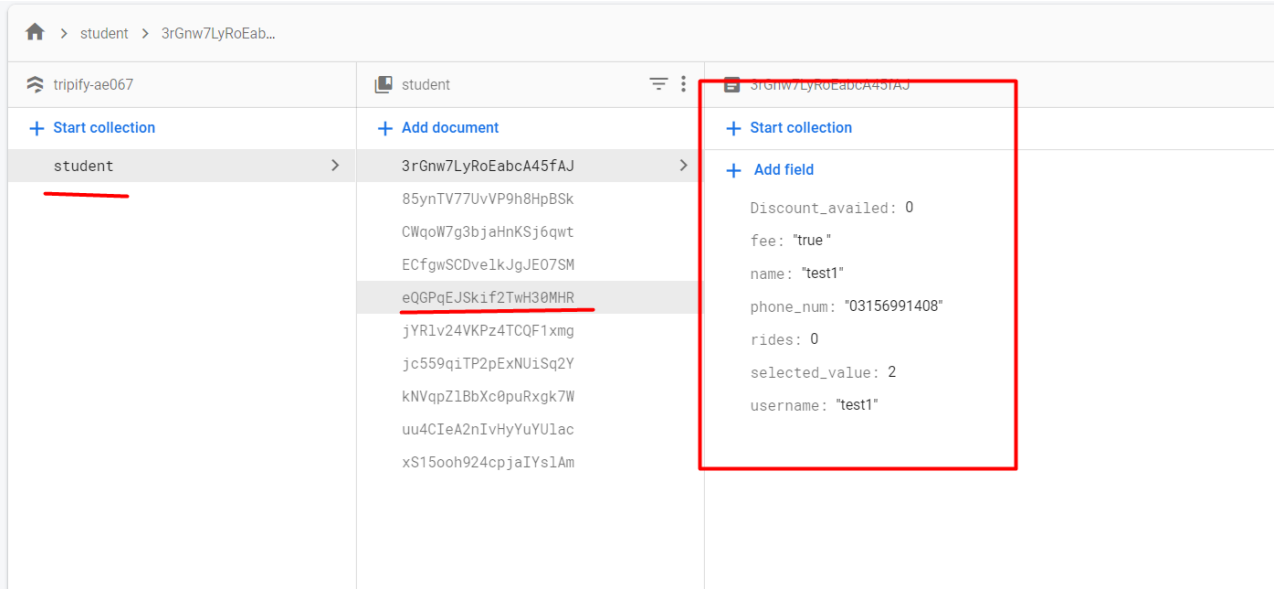


13. Database:

We used the Firebase database. Google makes it. They provide free Databases and authentication services for the developers to build the application. After the application is built then they charge money according to the database size and the users. Moreover, it is cloud-based. We do not need to download it locally to fetch and transmit information.

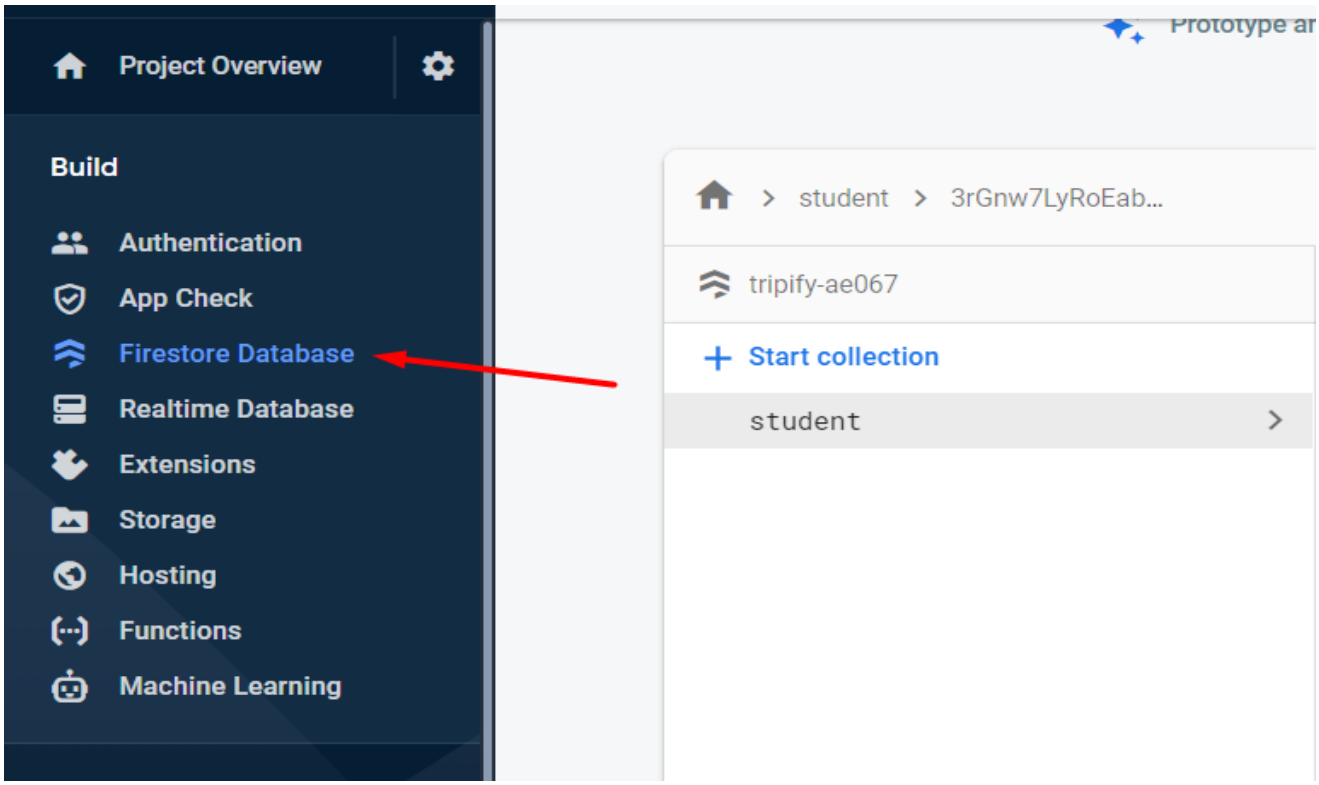
Non-Relational

Firebase is a non-relational database. It means it does not use rows and columns like SQL database. It has the concept of the documents and the collections. There can be multiple collections and documents. Each Collection contains multiple documents and Documents can further have the collection.



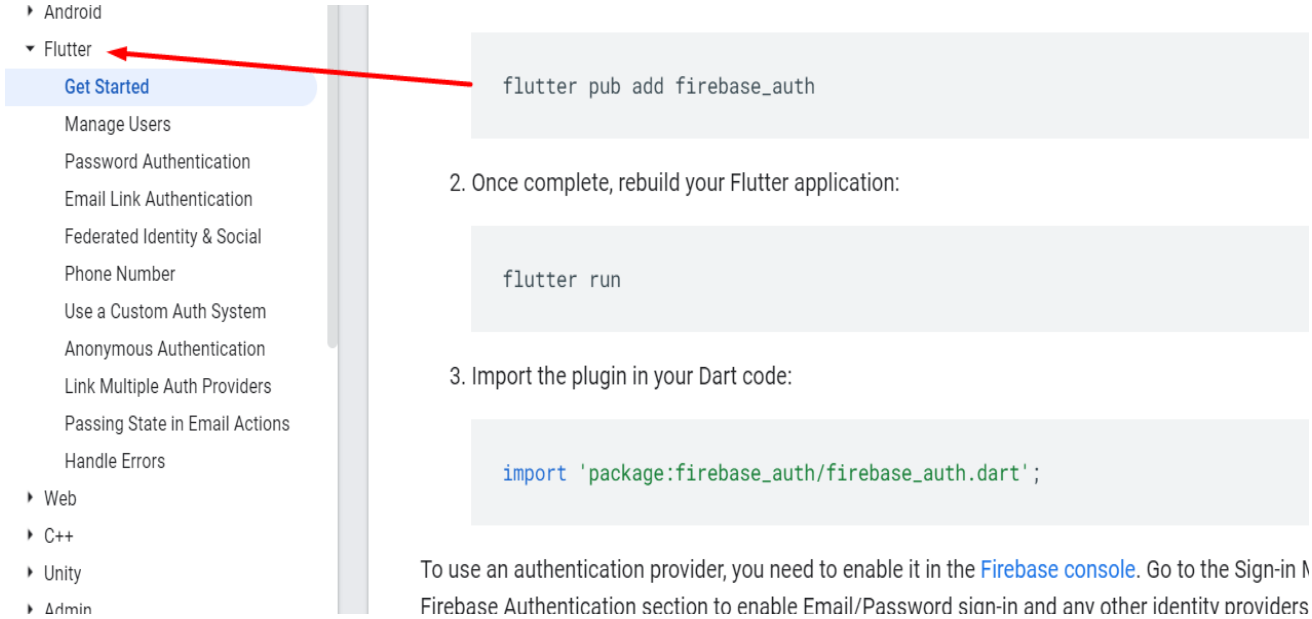
Realtime Database vs Fire Store Database:

The Realtime database is the Firebase old database while the Fire store Database is the new one. We used the Fire store for our project because it is specifically designed for the mobile apps developer & provides faster Queries and is scale able. Fire Store is more structural as compared to the Realtime Database. Other than that, Fetching the information from the Fire Store takes less time as compared to the Realtime Database. Lastly, the Fire Store provides built-in security while the real-time database does not provide built-in security. That is why we choose the Fire Store Database for our project.



Communication with the Database

Firebase provides its APIs to communicate with the database, Moreover, it has an authentication service that can be accessed and used by different APIs to Register and Login Any User with Proper Sanitization. Other than that, it has APIs for all the CRUD operations. It has the special concept of state management to know the current state of the user.



Links:

- https://app.diagrams.net/#G11mERNChEszvT75v6e_uVEVixzsHHw59U (Link of all the images/diagrams made in the assignment)