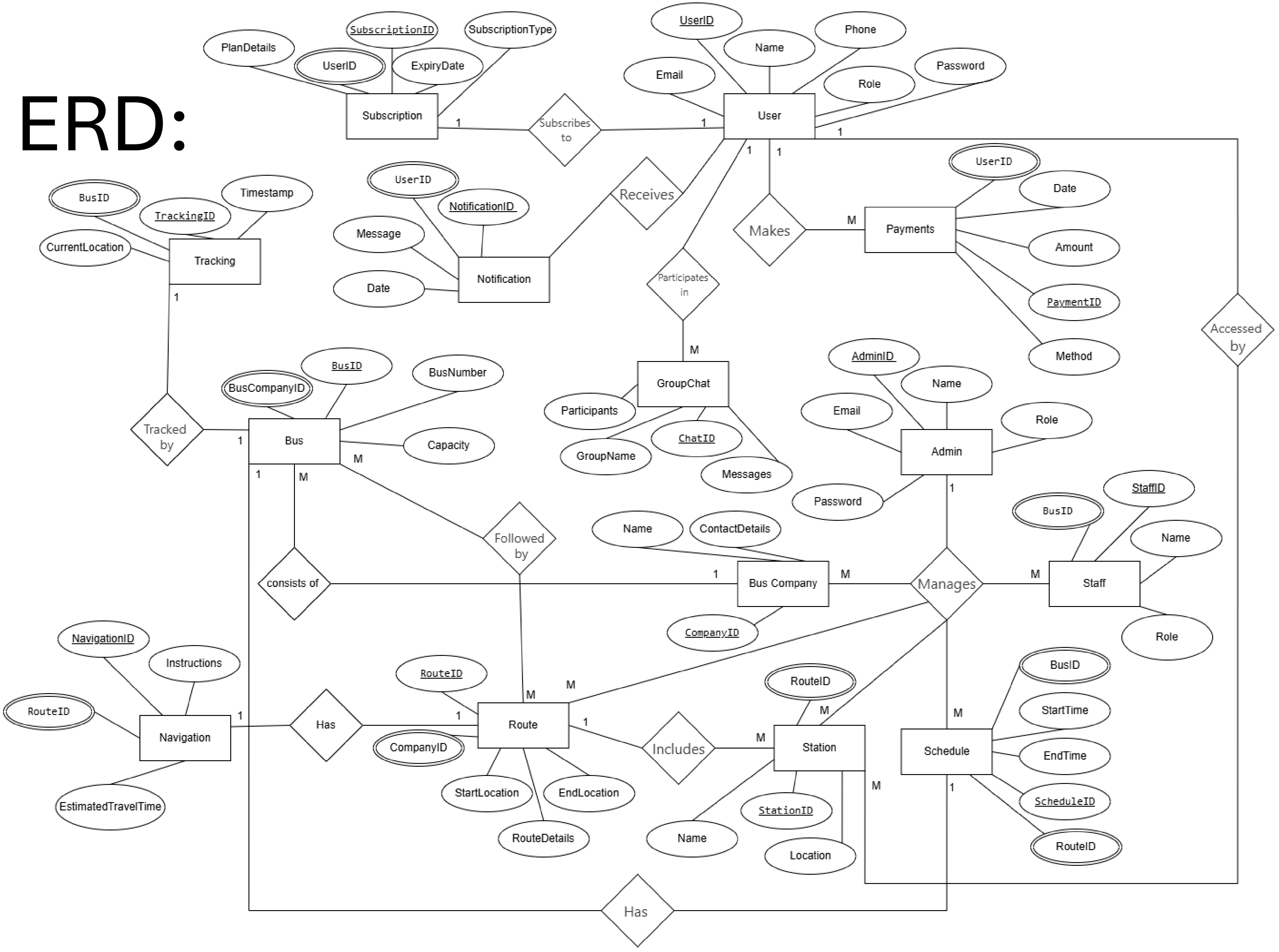
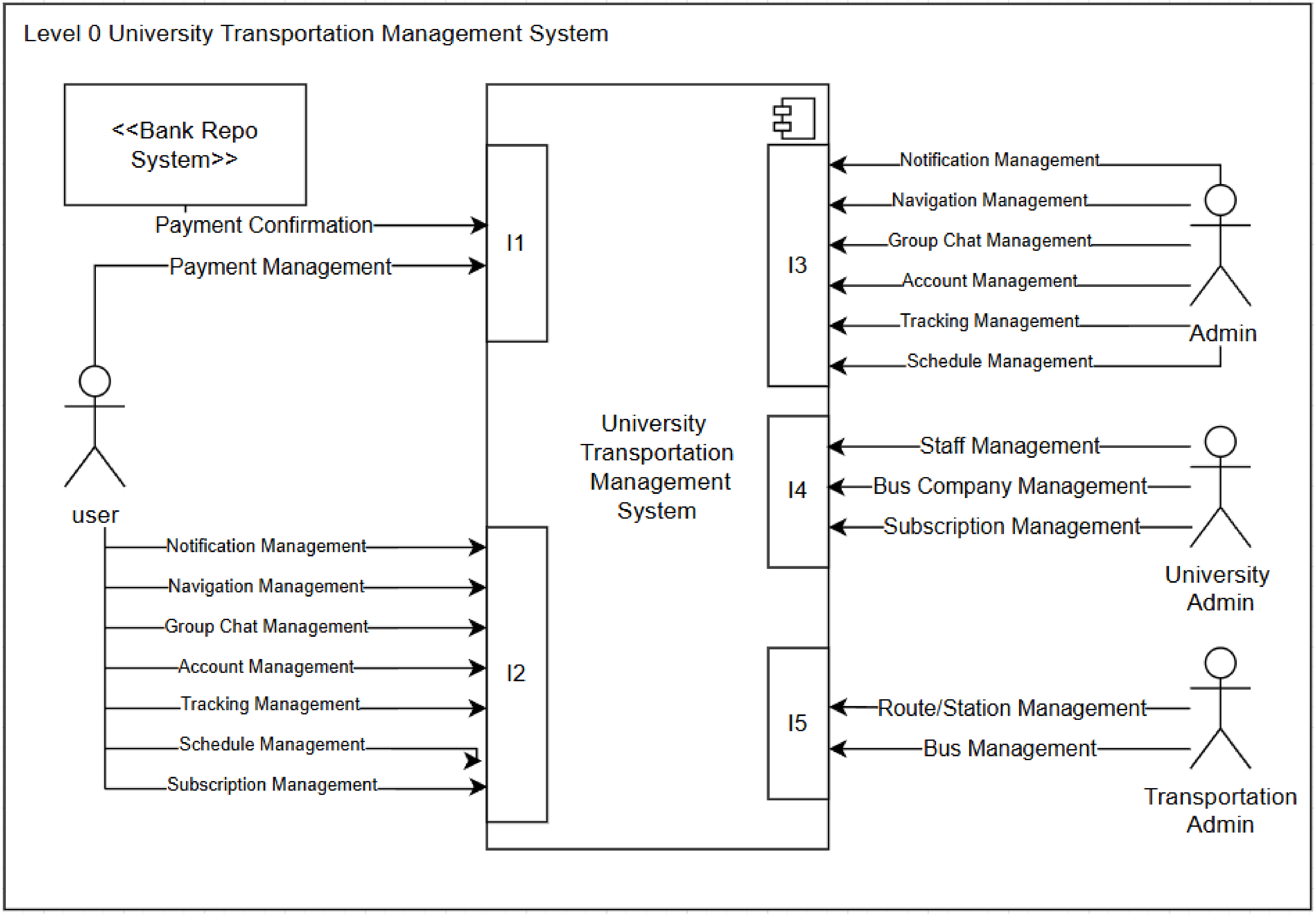
AIU. ALAMEIN INTERNATIONAL UNIVERSITY جامعة العلمين الدولية

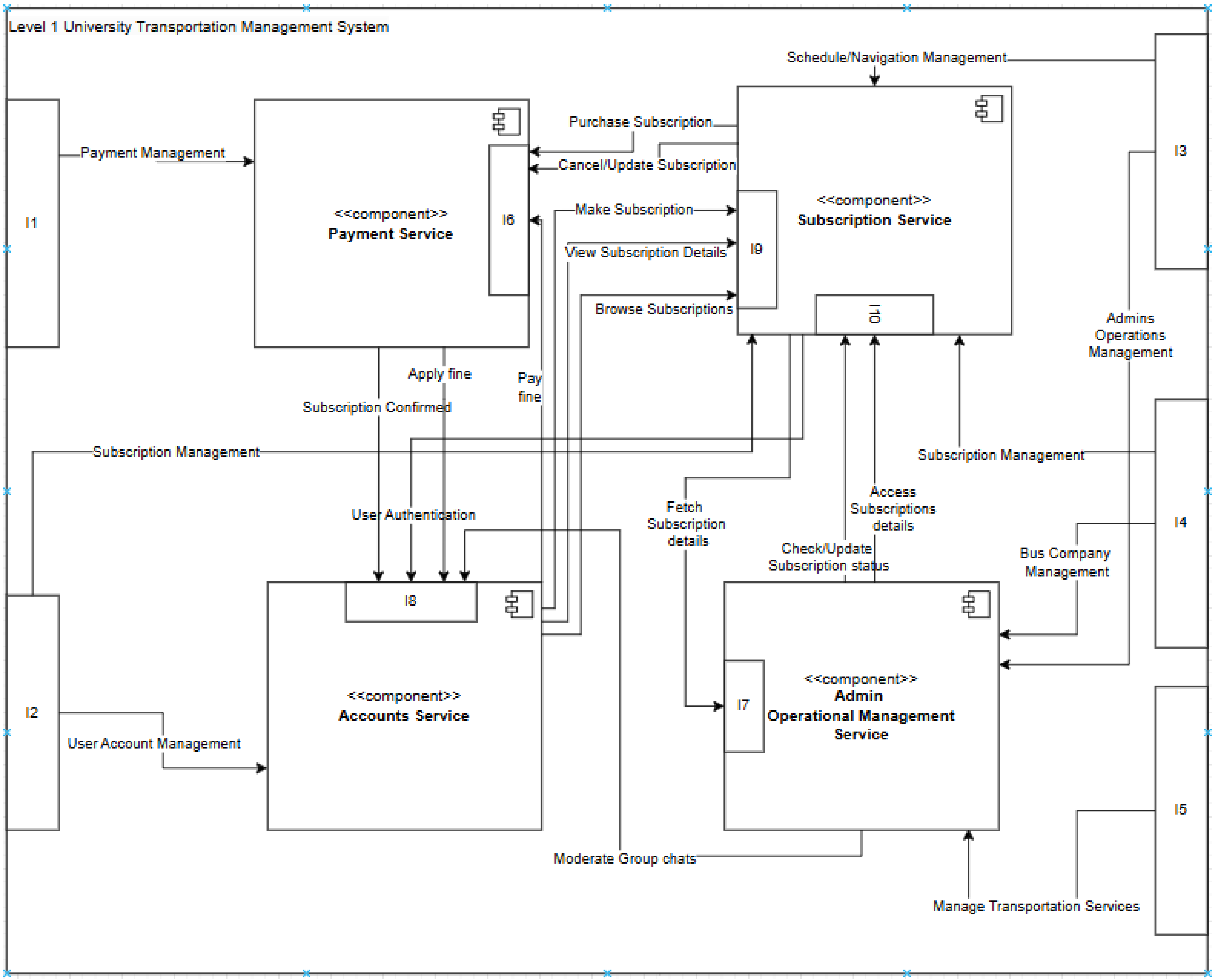
UniRoutes

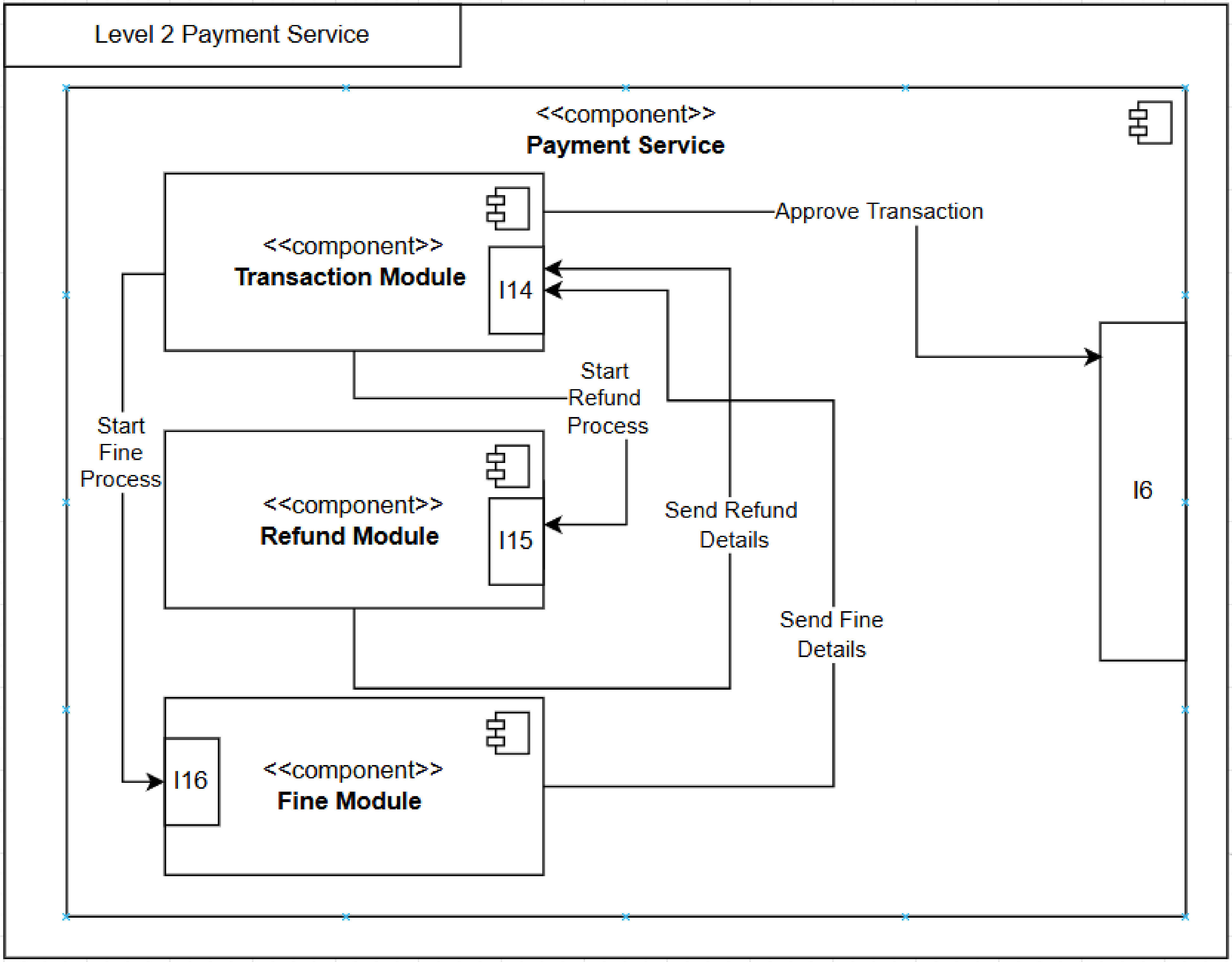
Name	ID
Abdallah Basem Project Manager	**22100848**
Ahmed Islam	**22101008**
Belal Fathy	**22101311**
Mazen Ahmed	**22100369**
Eyad Elnakib	**22100757**
Mahmoud Eid	**22100680**

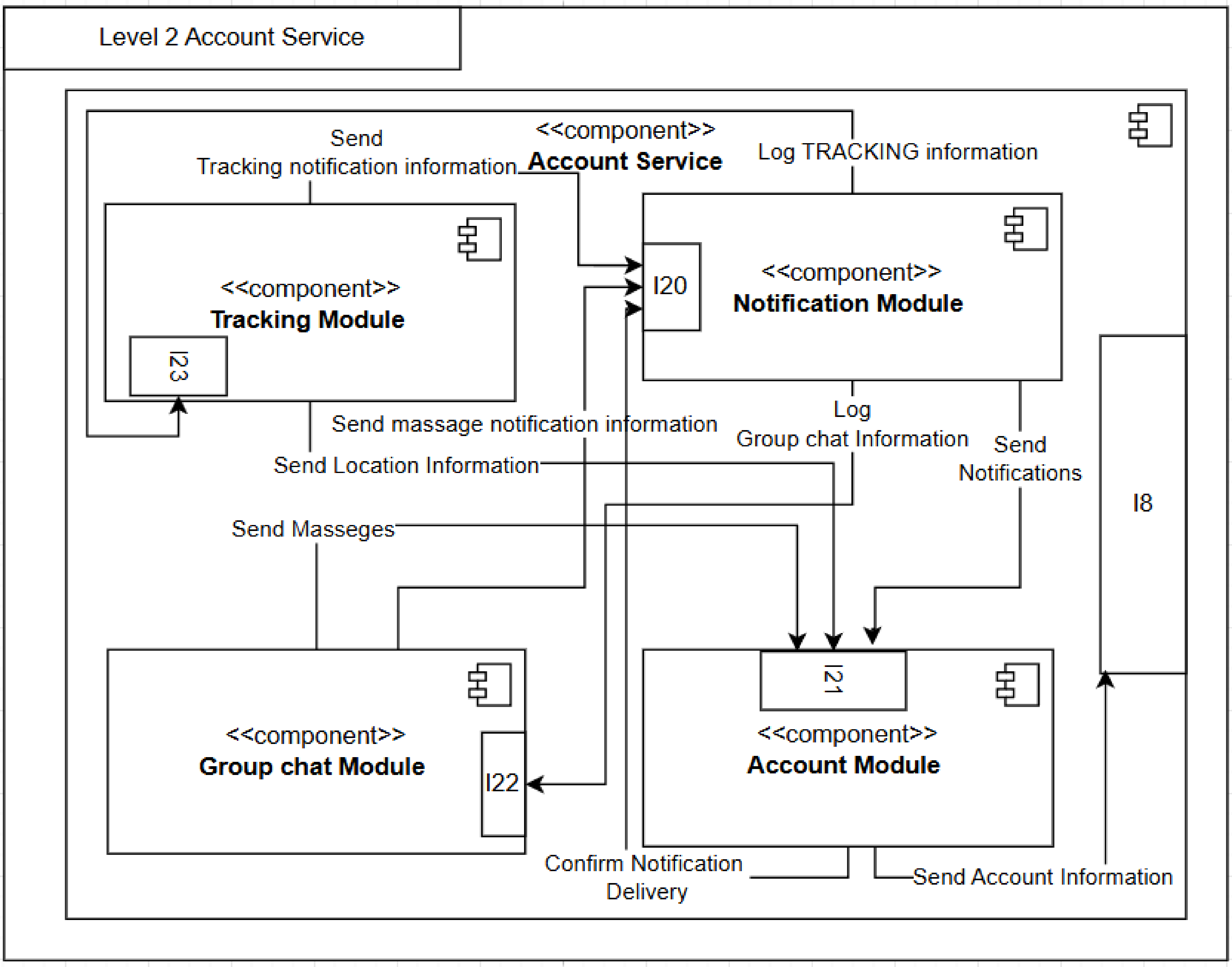
ERD:

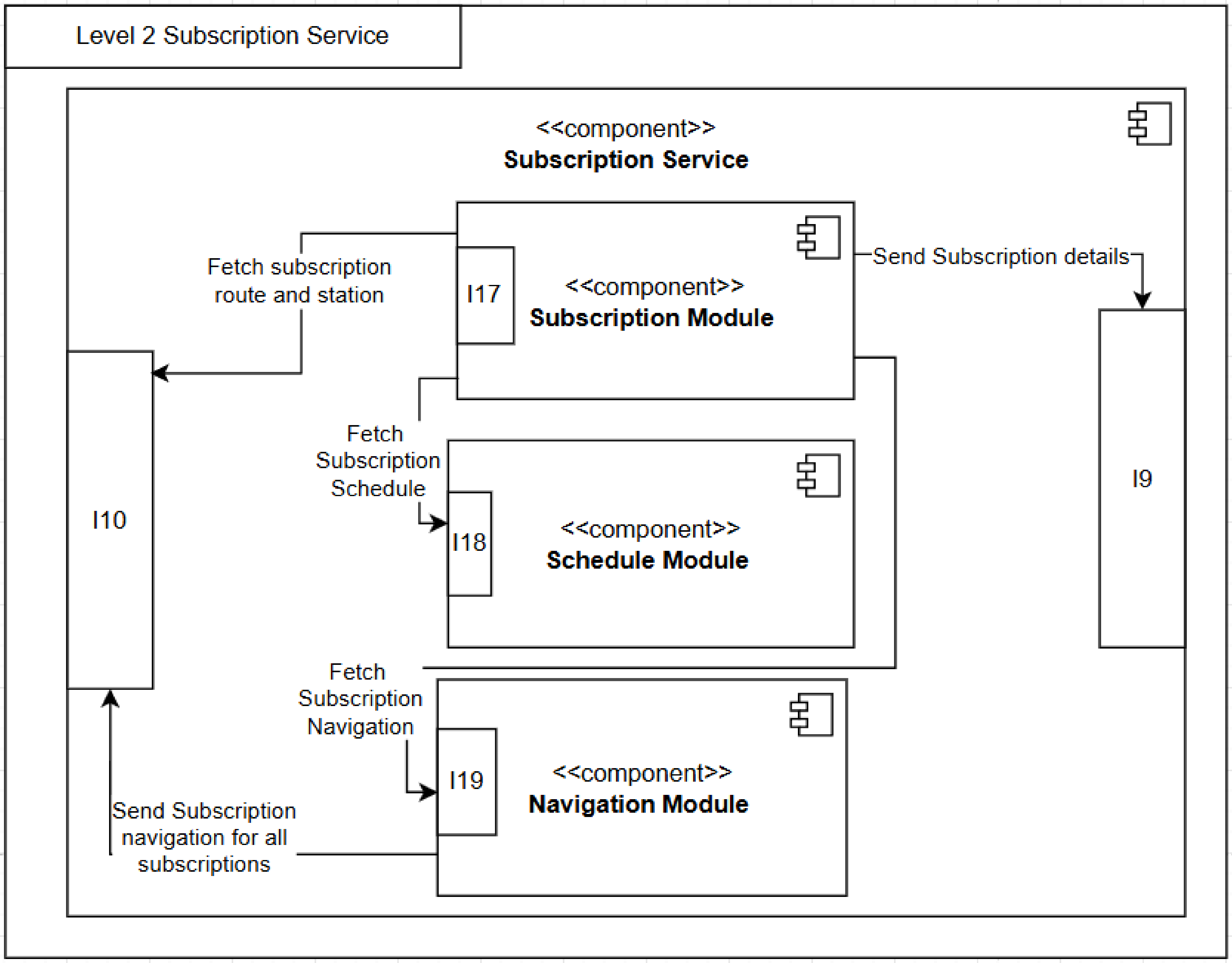

User
UserID
Name
Phone
Email
Role
Password
Subscription
SubscriptionID
SubscriptionType
PlanDetails
UserID
ExpiryDate
Subscribes to
Receives
Makes
Participates in
Accessed by
Payments
UserID
Date
Amount
PaymentID
Method
Notification
UserID
NotificationID
Message
Date
Tracking
BusID
TrackingID
Timestamp
CurrentLocation
Tracked by
Bus
BusID
BusNumber
BusCompanyID
Capacity
GroupChat
Participants
GroupName
ChatID
Messages
Admin
AdminID
Name
Email
Role
Password
Followed by
consists of
Bus Company
Name
ContactDetails
CompanyID
Manages
Staff
StaffID
BusID
Name
Role
Navigation
NavigationID
Instructions
RouteID
EstimatedTravelTime
Has
Route
RouteID
CompanyID
StartLocation
EndLocation
RouteDetails
Includes
Station
RouteID
StationID
Name
Location
Schedule
BusID
StartTime
EndTime
ScheduleID
RouteID
Has
1
M

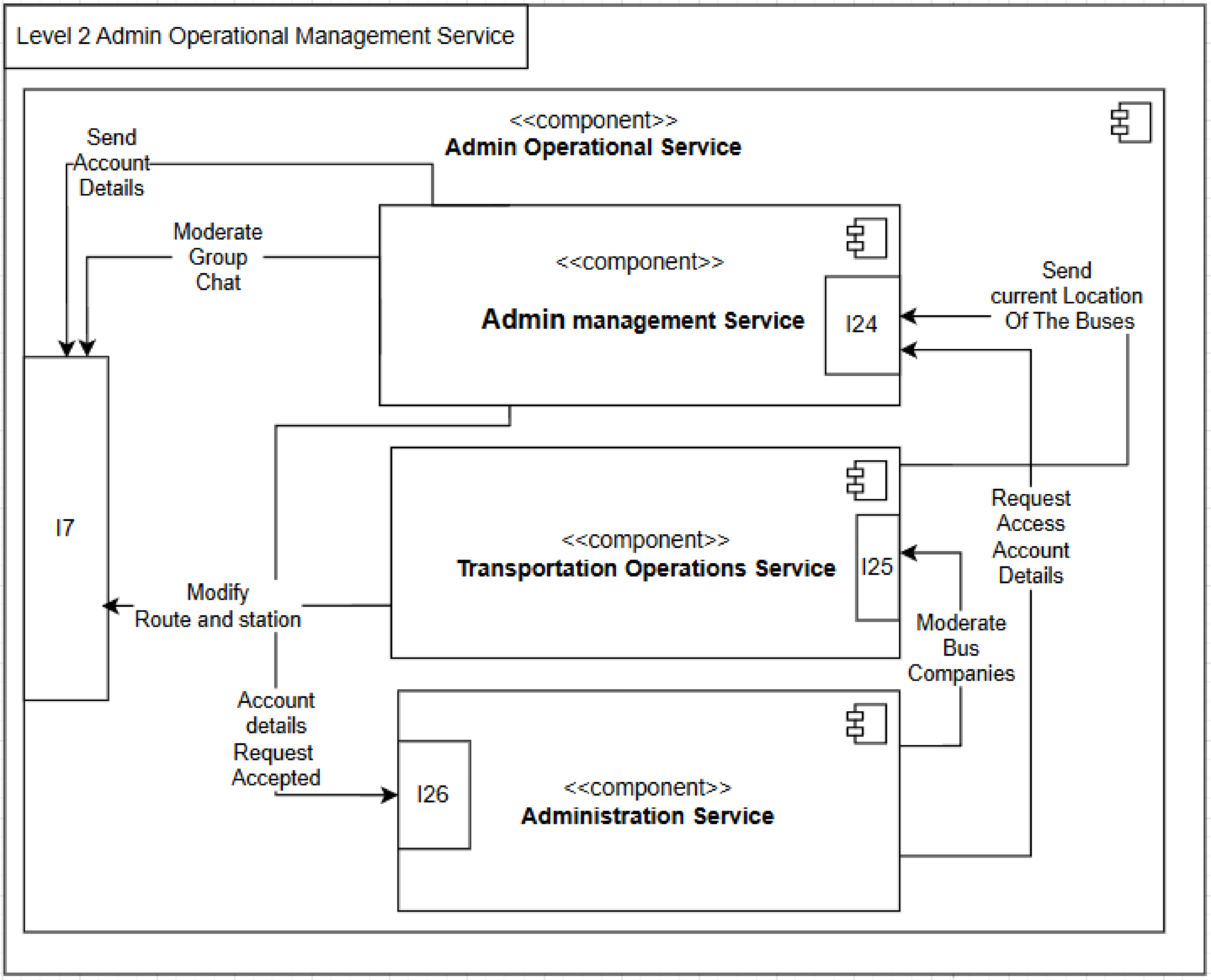

Level 0 University Transportation Management System
<<Bank Repo System>>
Payment Confirmation
Payment Management
I1
user
Notification Management
Navigation Management
Group Chat Management
Account Management
Tracking Management
Schedule Management
Subscription Management
I2
University Transportation Management System
I3
Notification Management
Navigation Management
Group Chat Management
Account Management
Tracking Management
Schedule Management
Admin
I4
Staff Management
Bus Company Management
Subscription Management
University Admin
I5
Route/Station Management
Bus Management
Transportation Admin

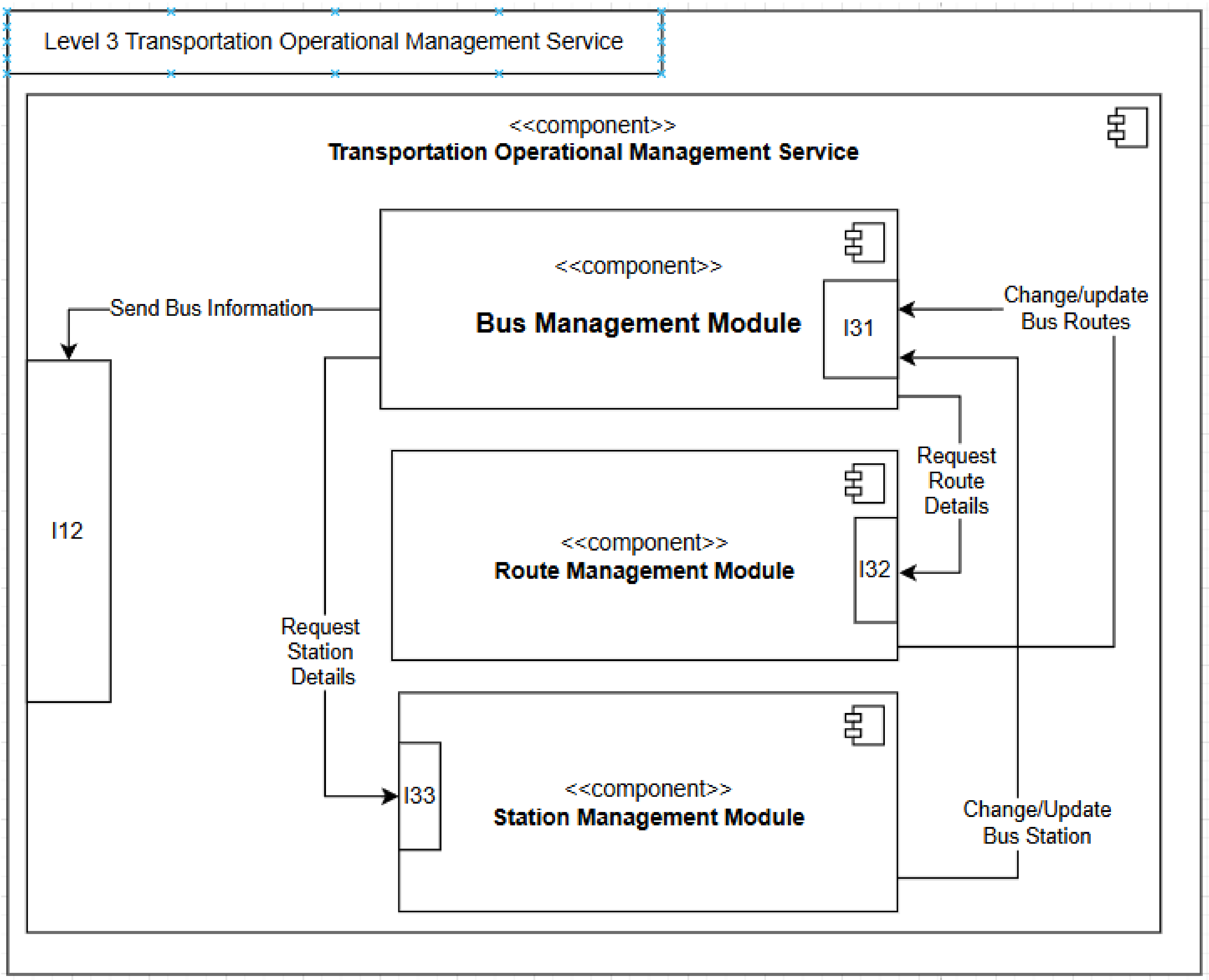

Level 1 University Transportation Management System
I1
Payment Management
<<component>>
Payment Service
I6
Purchase Subscription
Cancel/Update Subscription
Make Subscription
View Subscription Details
Browse Subscriptions
Schedule/Navigation Management
<<component>>
Subscription Service
I9
I10
I3
Admins Operations Management
Apply fine
Pay fine
Subscription Confirmed
Subscription Management
User Authentication
Fetch Subscription details
Access Subscriptions details
Check/Update Subscription status
Subscription Management
I4
Bus Company Management
I2
User Account Management
I8
<<component>>
Accounts Service
I7
<<component>>
Admin Operational Management Service
I5
Moderate Group chats
Manage Transportation Services

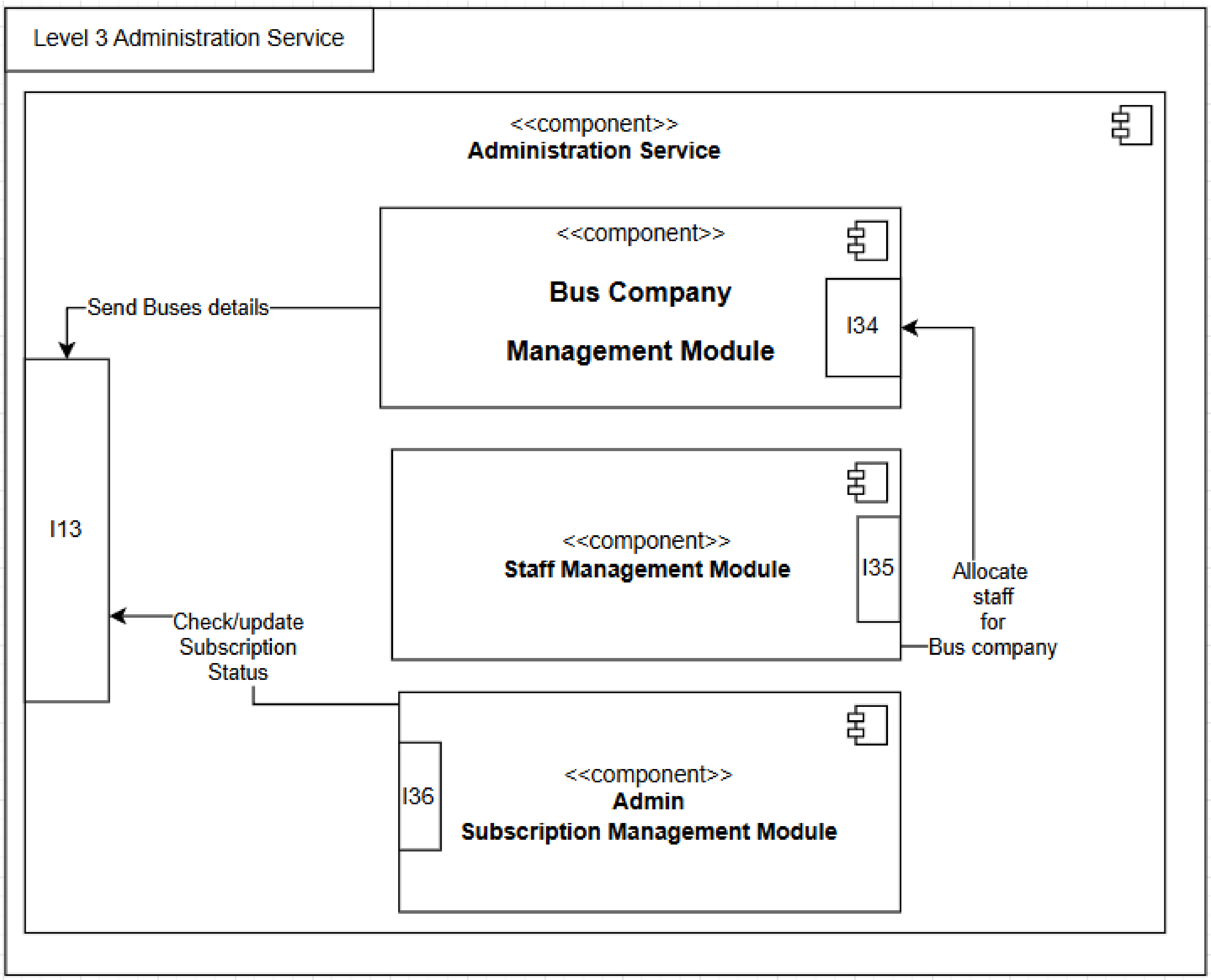

Level 2 Payment Service
<<component>>
Payment Service
<<component>>
Transaction Module
I14
Approve Transaction
I6
Start Fine Process
Start Refund Process
<<component>>
Refund Module
I15
Send Refund Details
Send Fine Details
I16
<<component>>
Fine Module

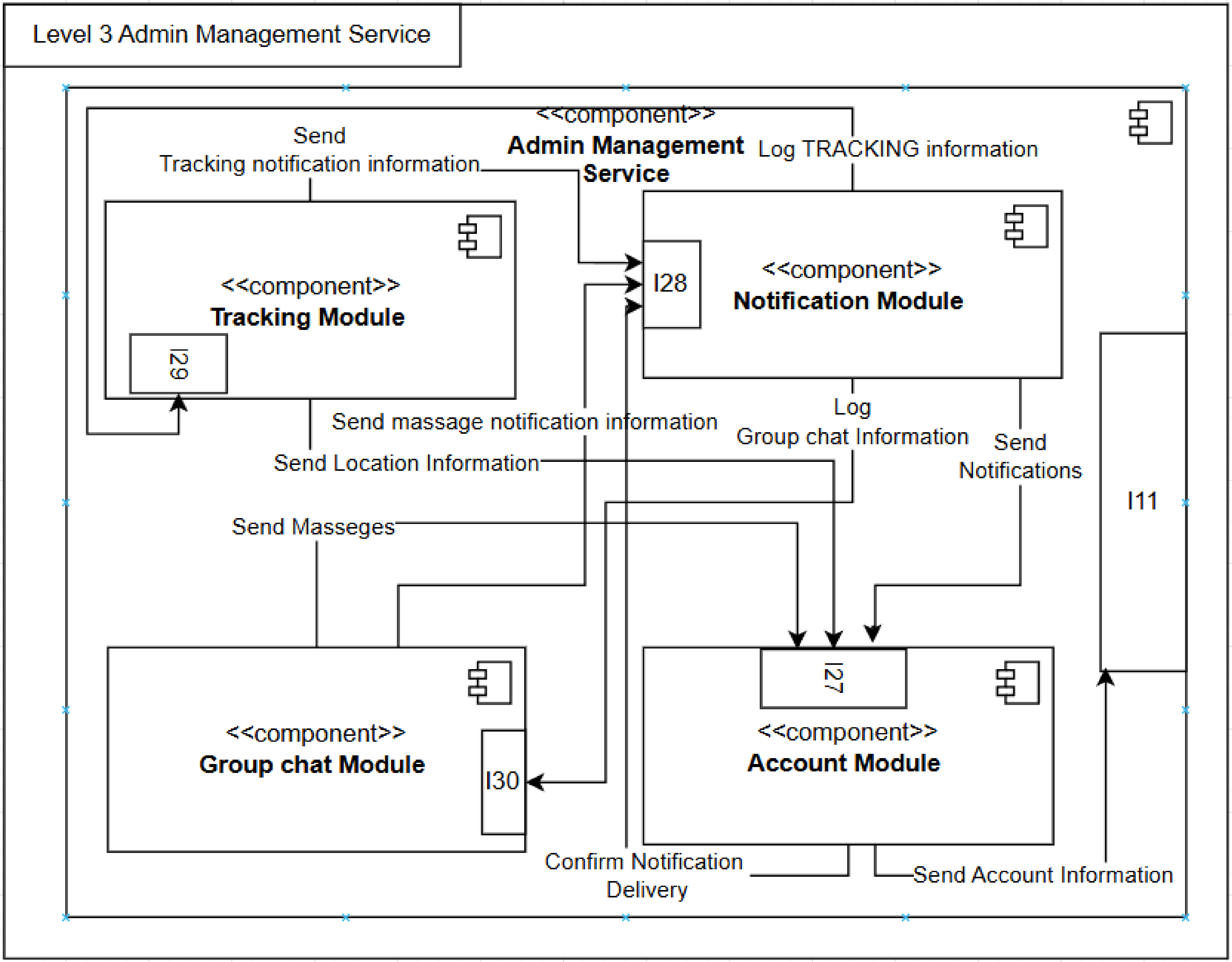

Level 2 Account Service
<<component>>
Account Service
Send
Tracking notification information
Log TRACKING information
<<component>>
Tracking Module
I23
I20
<<component>>
Notification Module
Send massage notification information
Log
Group chat Information
Send
Notifications
Send Location Information
I8
Send Masseges
<<component>>
Group chat Module
I22
I21
<<component>>
Account Module
Confirm Notification
Delivery
Send Account Information


Level 2 Subscription Service
<<component>>
Subscription Service
<<component>>
Subscription Module
I17
Fetch subscription route and station
Send Subscription details
I9
I10
Fetch Subscription Schedule
<<component>>
Schedule Module
I18
Fetch Subscription Navigation
<<component>>
Navigation Module
I19
Send Subscription navigation for all subscriptions


Level 2 Admin Operational Management Service
<<component>>
Admin Operational Service
Send Account Details
Moderate Group Chat
<<component>>
Admin management Service
I24
Send current Location Of The Buses
I7
<<component>>
Transportation Operations Service
I25
Request Access Account Details
Modify Route and station
Moderate Bus Companies
Account details Request Accepted
I26
<<component>>
Administration Service


Level 3 Transportation Operational Management Service
<<component>>
Transportation Operational Management Service
<<component>>
Bus Management Module
I31
Send Bus Information
I12
Change/update Bus Routes
Request Route Details
<<component>>
Route Management Module
I32
Request Station Details
I33
<<component>>
Station Management Module
Change/Update Bus Station


Level 3 Administration Service
<<component>>
Administration Service
<<component>>
Bus Company
Management Module
I34
Send Buses details
I13
<<component>>
Staff Management Module
I35
Allocate staff for Bus company
Check/update Subscription Status
I36
<<component>>
Admin
Subscription Management Module


Level 3 Admin Management Service
<<component>>
Admin Management Service
Send
Tracking notification information
Log TRACKING information
<<component>>
Tracking Module
I29
I28
<<component>>
Notification Module
Send massage notification information
Log
Group chat Information
Send
Notifications
Send Location Information
I11
Send Masseges
I27
<<component>>
Group chat Module
I30
<<component>>
Account Module
Confirm Notification
Delivery
Send Account Information

Level 0 interface description

I1 (Payment Interface):

- Manages payment processing and integration with the external banking system. It ensures payment confirmation and facilitates payment submissions for subscriptions and other services

I2 (Student Interaction Interface):

- Serves as the central hub for Students to access core functionalities such as subscription management, navigation management, group chat, tracking, account, schedule, and notification services.

I3 (Admin Interaction Interface):

- Allows administrative staff to manage core functionalities related to the transportation system. This includes notification management, navigation management, group chat oversight, account, tracking, and schedule management.

I4 (University Admin Interface):

- Provides university administrators with tools to manage staff, bus company partnerships, and subscriptions.

I5 (Transportation Admin Interface):

- Designed for transportation administrators to handle route and station management, as well as bus operations.

Level 1 interface Description

I6 (Payment Service Interface):

- Connects the Payment Service with other system components, facilitating user transactions for subscriptions, fines, and payment confirmations.

I7 (Admin Operational Management Interface):

- Provides administrative tools to manage operational aspects of subscriptions and transportation. This interface ensures admins can perform updates and monitor activities effectively. While also acting as the central interface for admins to manage their accounts also including notification management, navigation management, group chat oversight, account, tracking, and schedule management

I8 (Accounts Service Interface):

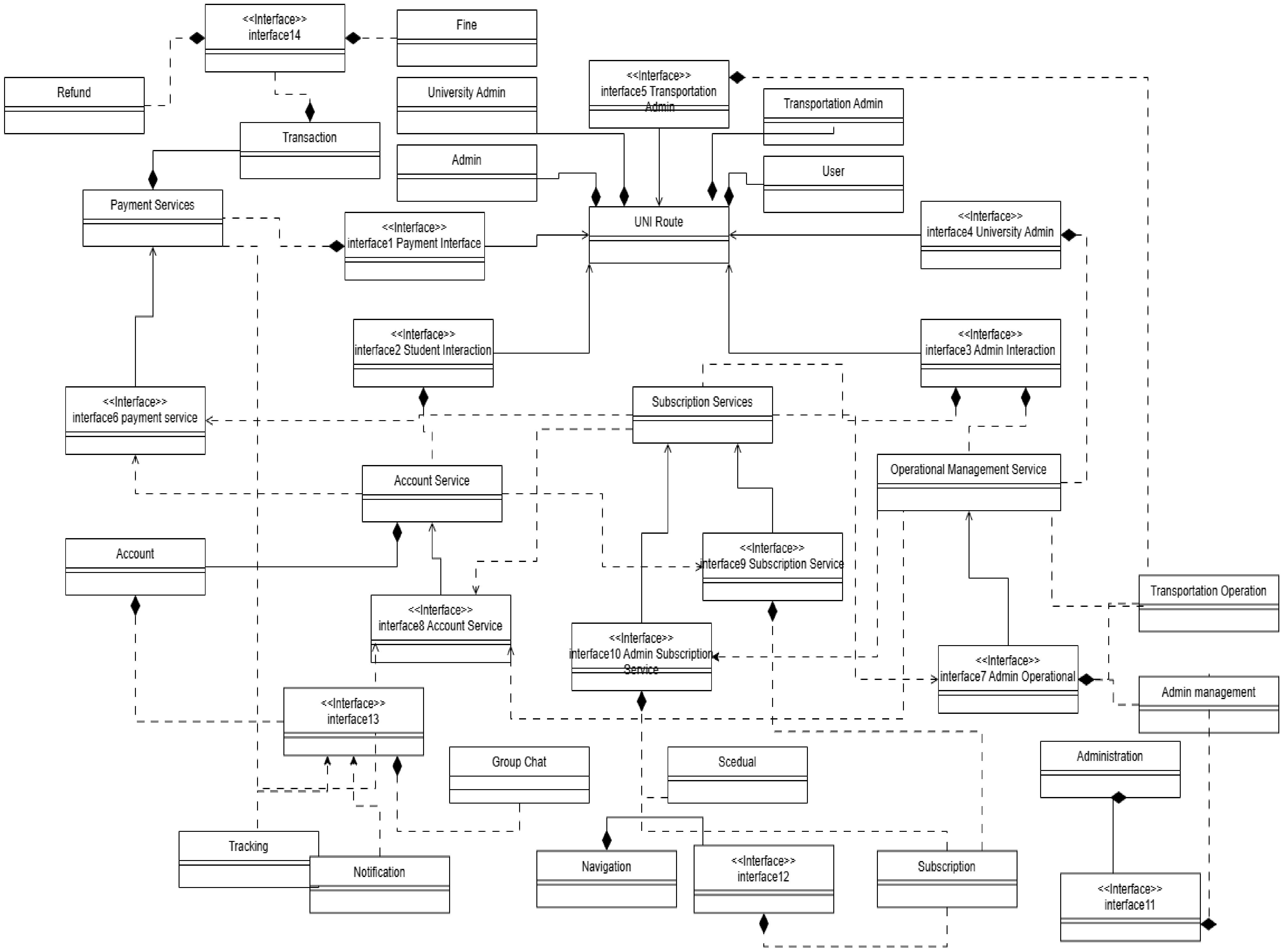
- Acts as the central interface for managing user accounts and authentication. It bridges other services like subscriptions and payments with user account information. Also enabling them to handle their group chats tracking and Notifications.

I9 (Subscription Service Interface):

- Focused on subscription-related actions such as browsing, purchasing, canceling, or updating subscriptions. This interface ensures seamless user access to subscription services.

I10 (Admin Subscription Service Interface):

- Provides administrators with full control and monitoring capabilities over all active subscriptions. This interface allows admins to manage, update, and oversee subscription details effectively.



<<Interface>> interface14
Fine
Refund
University Admin
<<Interface>> interface5 Transportation Admin
Transportation Admin
Transaction
Admin
User
Payment Services
<<Interface>> interface1 Payment Interface
UNI Route
<<Interface>> interface4 University Admin
<<Interface>> interface2 Student Interaction
<<Interface>> interface3 Admin Interaction
<<Interface>> interface6 payment service
Subscription Services
Operational Management Service
Account Service
Account
<<Interface>> interface9 Subscription Service
<<Interface>> interface8 Account Service
<<Interface>> interface10 Admin Subscription Service
Transportation Operation
<<Interface>> interface7 Admin Operational
Admin management
<<Interface>> interface13
Group Chat
Scedual
Administration
Tracking
Notification
Navigation
<<Interface>> interface12
Subscription
<<Interface>> interface11

This is the Mapping Module Matrix

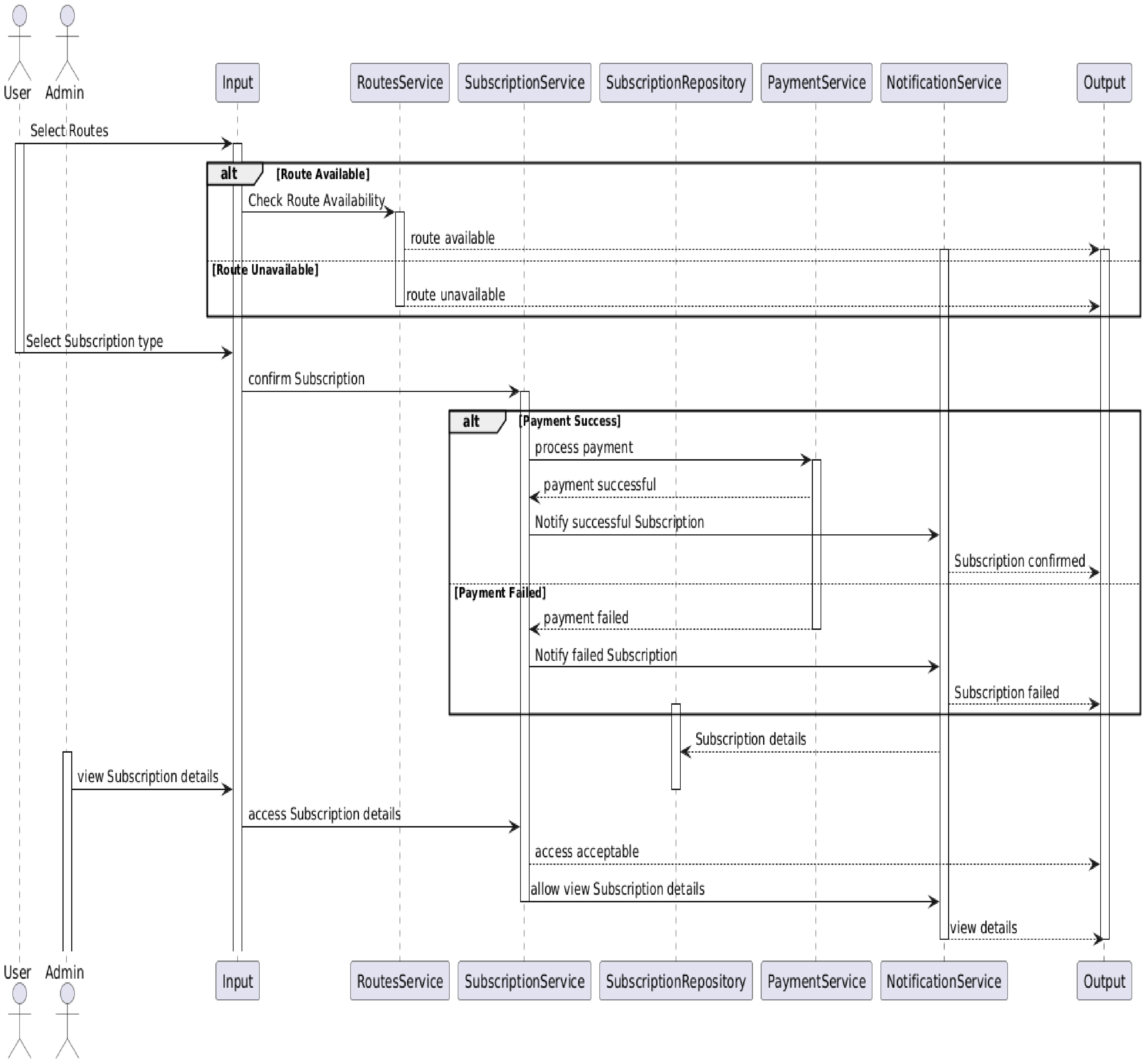
Columns are the use cases and the rows are the main classes

	Payment M.	Subscription M.	Buscompany M.	Staff M.	Schedule M.	Bus Mang.	Route/ Station M.	Navigation M.	Tracking Mang	Account Mang	Group Chat M.	Notification M
Admin					✓			✓	✓	✓	✓	✓
Uni Admin		✓	✓	✓				✓	✓	✓	✓	✓
Transport Admin						✓	✓	✓	✓	✓	✓	✓
User	✓	✓			✓			✓	✓	✓	✓	✓
Payment	✓	✓										✓
Account		✓			✓				✓		✓	✓
Tracking		✓				✓		✓	✓			✓
Notification	✓	✓			✓	✓		✓	✓		✓	✓
Group Chat		✓			✓			✓	✓	✓	✓	✓
Navigation		✓						✓	✓		✓	✓
Schedule		✓			✓			✓			✓	✓
Subscription		✓			✓		✓	✓			✓	✓
Operations		✓			✓	✓		✓	✓		✓	✓

Belal Fathy

Use case: Subscription Management

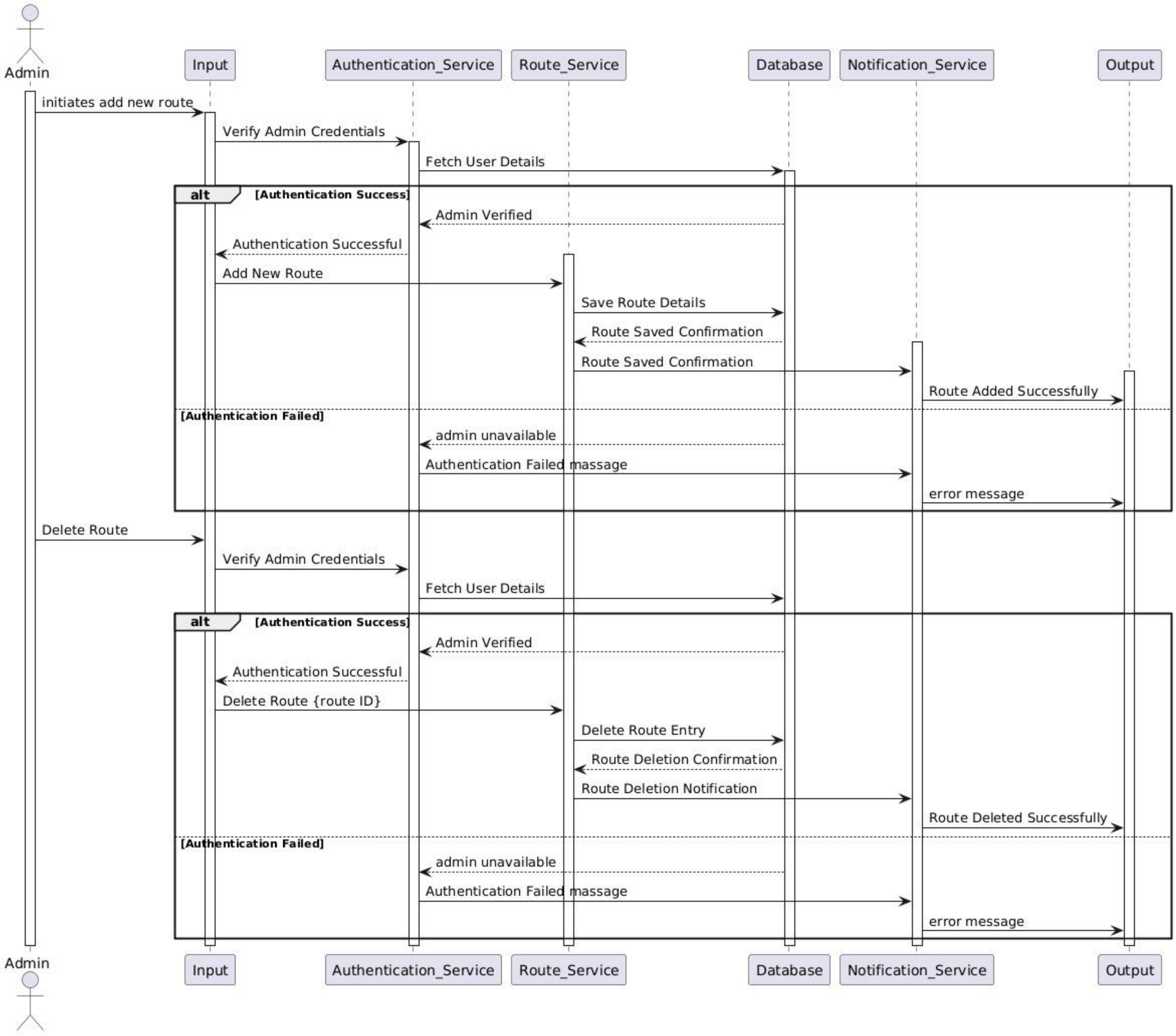
22101311



Mahmoud Eid

Use case: Route Management

22100680

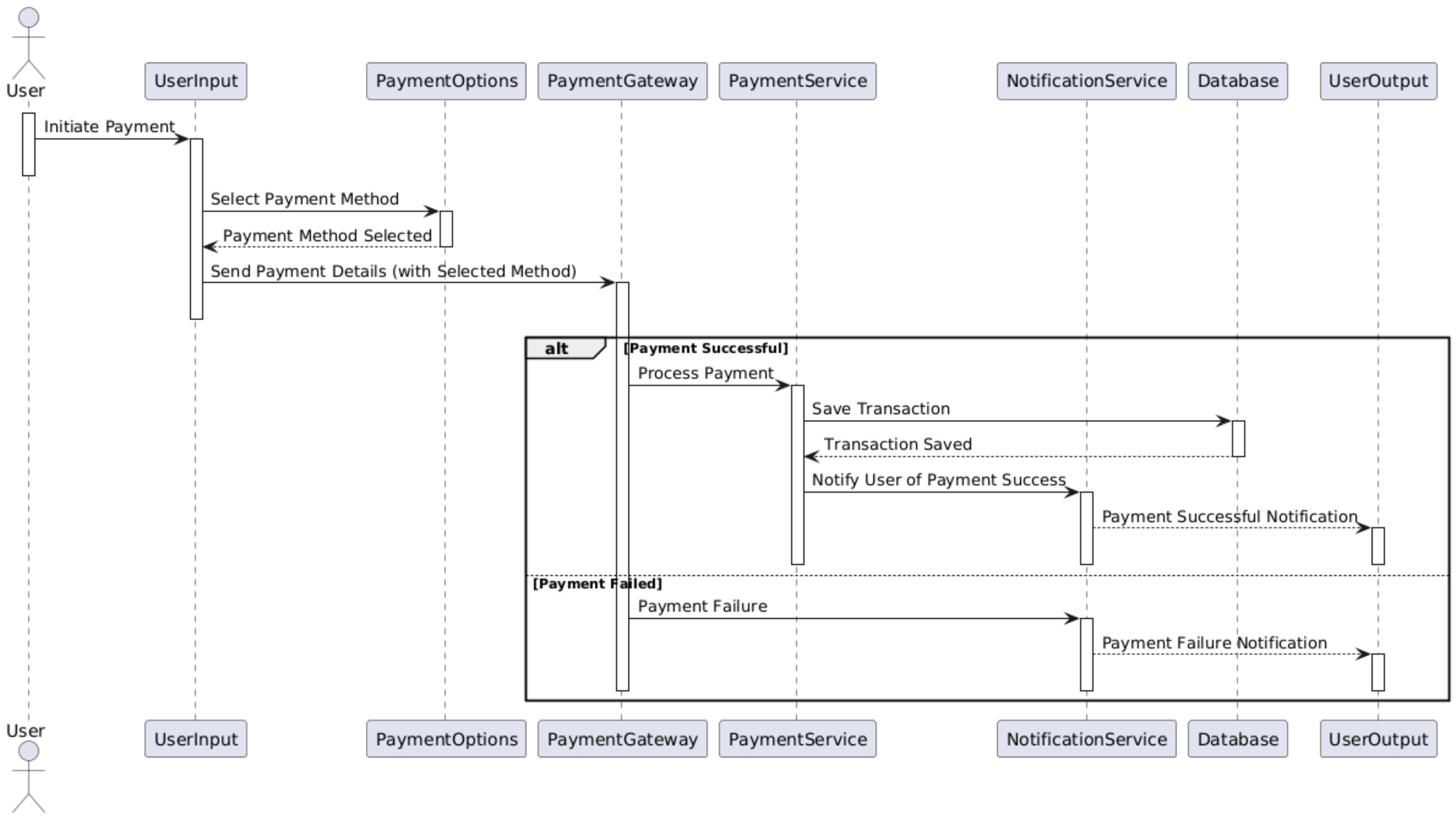


Ahmed Islam Farouk 22101008

Use Case : Payment Management

Use Case	Controller	Payment Controller	Domain	Data Table	Display Screens	Other UC
Payment	UserInput	PaymentGateway	User	Transaction Data	User Input Screen	Retry Payment/Alternative Options
	PaymentOptions	PaymentService			Payment Options Screen	
		NotificationService			Notification Screen	
					User Output Screen	

Step	Message Name	Owner Class Name
1	InitiatePayment()	UserInput
2	SelectPaymentMethod()	PaymentOptions
3	SendPaymentDetails()	PaymentGateway
4	ProcessPayment()	PaymentService
5	SaveTransaction()	Database
6	NotifyUserOfPaymentSuccess()	NotificationService
7	NotifyUserOfPaymentFailure()	NotificationService
8	DisplayNotification()	UserOutput
9	RetryPaymentOrAlternativeOption()	PaymentOptions


User
UserInput
PaymentOptions
PaymentGateway
PaymentService
NotificationService
Database
UserOutput
Initiate Payment
Select Payment Method
Payment Method Selected
Send Payment Details (with Selected Method)
alt
[Payment Successful]
Process Payment
Save Transaction
Transaction Saved
Notify User of Payment Success
Payment Successful Notification
[Payment Failed]
Payment Failure
Payment Failure Notification

Abdallah Basem Zain 22100848
Use case: Navigation Management

Domain	**Database Table**	**Display Screens or Report**
User	Accounts table	Customer App Screen
	Subscriptions table	Registration Screen
	Routes table	Map Screen

The system validates the user account and retrieves the list of subscriptions.

The system fetches and displays the relevant routes based on the user subscriptions.

The user can one see his location on the map If user is not logging on.

The user can review a map showing their subscribed routes.

The user can view detailed information about that route by clicking on a specific route.

Messages:

Get account cardinalities ()

Return cardinalities ()

get subscription details ()

return details ()

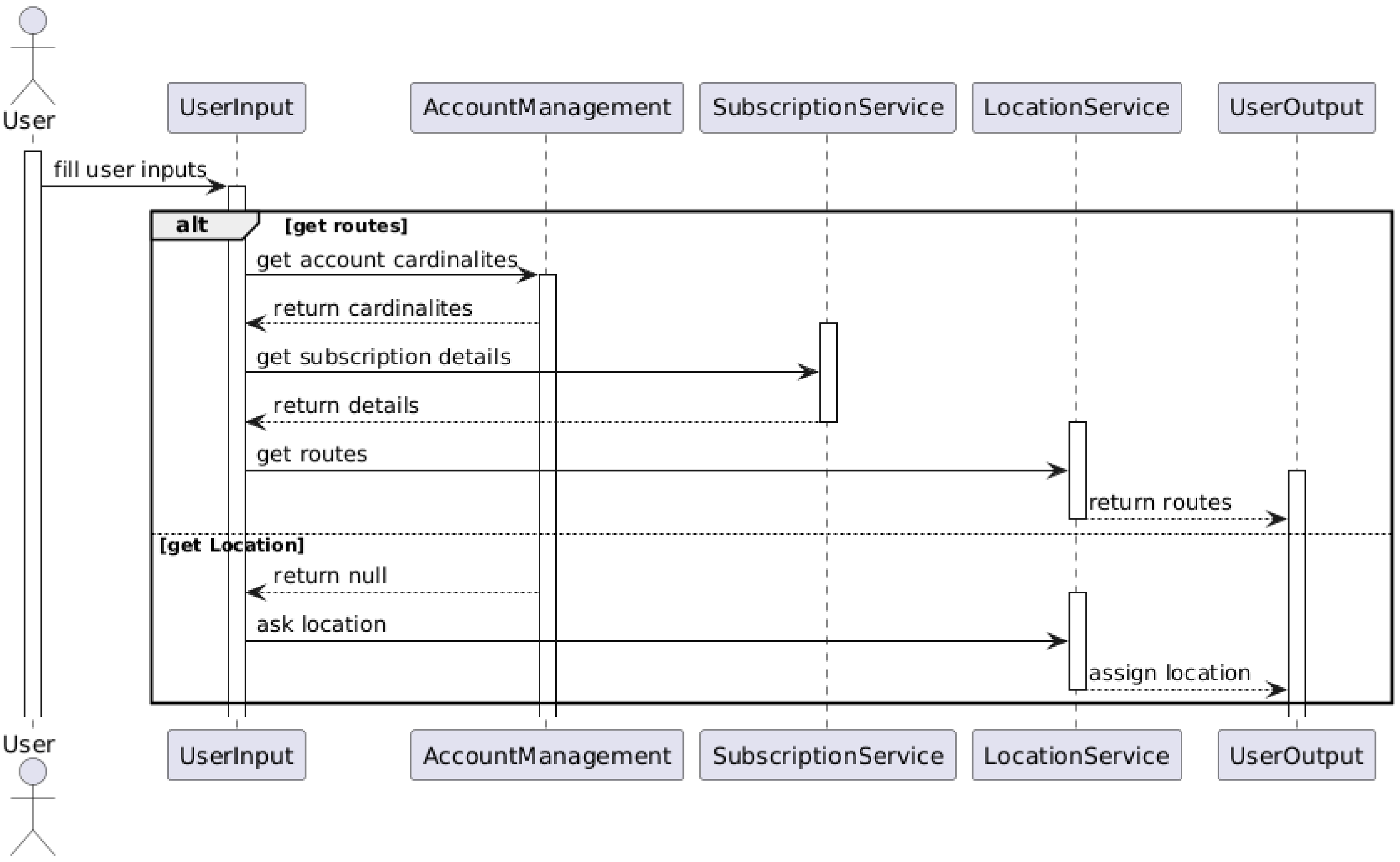
Get routes ()

Return routes ()

Return null ()

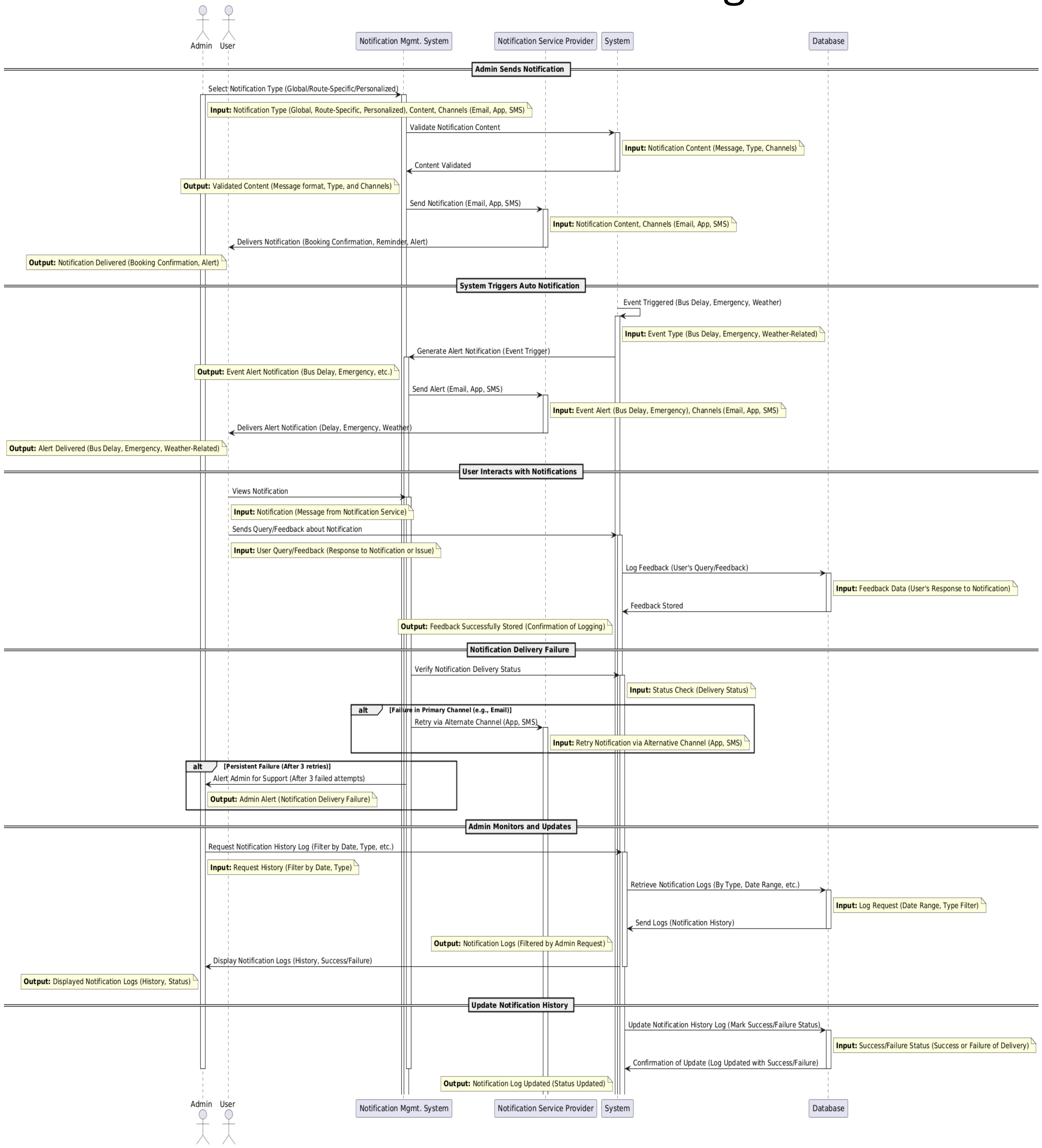
Ask location ()

Assign location ()


User
UserInput
AccountManagement
SubscriptionService
LocationService
UserOutput
fill user inputs
alt
[get routes]
get account cardinalites
return cardinalites
get subscription details
return details
get routes
return routes
[get Location]
return null
ask location
assign location
User
UserInput
AccountManagement
SubscriptionService
LocationService
UserOutput

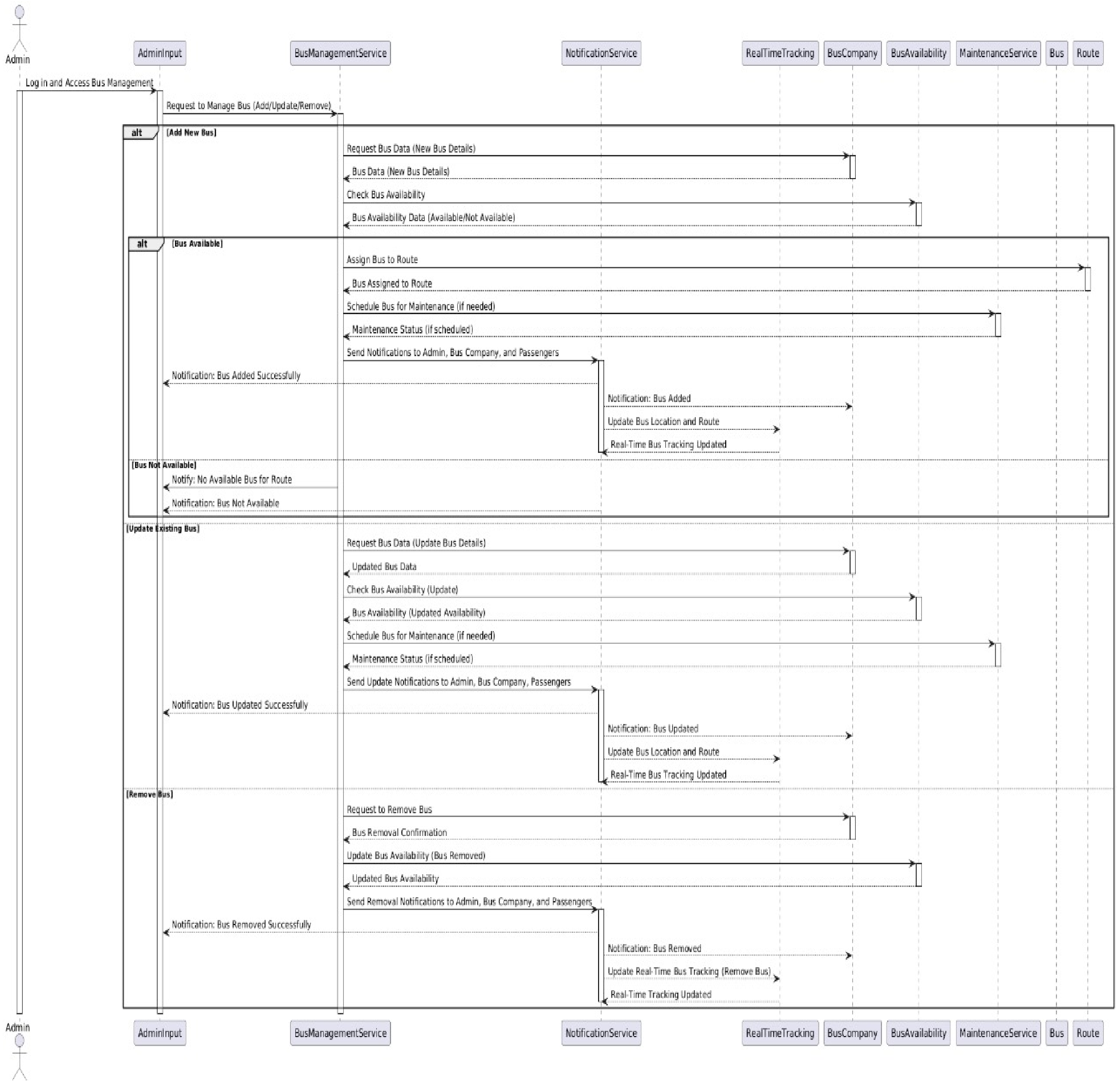
Eyad Metwally 22100757

Use Case: Notification Management


Admin
User
Notification Mgmt. System
Notification Service Provider
System
Database
Admin Sends Notification
Select Notification Type (Global/Route-Specific/Personalized)
Input: Notification Type (Global, Route-Specific, Personalized), Content, Channels (Email, App, SMS)
Validate Notification Content
Input: Notification Content (Message, Type, Channels)
Content Validated
Output: Validated Content (Message format, Type, and Channels)
Send Notification (Email, App, SMS)
Input: Notification Content, Channels (Email, App, SMS)
Delivers Notification (Booking Confirmation, Reminder, Alert)
Output: Notification Delivered (Booking Confirmation, Alert)
System Triggers Auto Notification
Event Triggered (Bus Delay, Emergency, Weather)
Input: Event Type (Bus Delay, Emergency, Weather-Related)
Generate Alert Notification (Event Trigger)
Output: Event Alert Notification (Bus Delay, Emergency, etc.)
Send Alert (Email, App, SMS)
Input: Event Alert (Bus Delay, Emergency), Channels (Email, App, SMS)
Delivers Alert Notification (Delay, Emergency, Weather)
Output: Alert Delivered (Bus Delay, Emergency, Weather-Related)
User Interacts with Notifications
Views Notification
Input: Notification (Message from Notification Service)
Sends Query/Feedback about Notification
Input: User Query/Feedback (Response to Notification or Issue)
Log Feedback (User's Query/Feedback)
Input: Feedback Data (User's Response to Notification)
Feedback Stored
Output: Feedback Successfully Stored (Confirmation of Logging)
Notification Delivery Failure
Verify Notification Delivery Status
Input: Status Check (Delivery Status)
alt [Failure in Primary Channel (e.g., Email)]
Retry via Alternate Channel (App, SMS)
Input: Retry Notification via Alternative Channel (App, SMS)
alt [Persistent Failure (After 3 retries)]
Alert Admin for Support (After 3 failed attempts)
Output: Admin Alert (Notification Delivery Failure)
Admin Monitors and Updates
Request Notification History Log (Filter by Date, Type, etc.)
Input: Request History (Filter by Date, Type)
Retrieve Notification Logs (By Type, Date Range, etc.)
Input: Log Request (Date Range, Type Filter)
Send Logs (Notification History)
Output: Notification Logs (Filtered by Admin Request)
Display Notification Logs (History, Success/Failure)
Output: Displayed Notification Logs (History, Status)
Update Notification History
Update Notification History Log (Mark Success/Failure Status)
Input: Success/Failure Status (Success or Failure of Delivery)
Confirmation of Update (Log Updated with Success/Failure)
Output: Notification Log Updated (Status Updated)
Admin
User
Notification Mgmt. System
Notification Service Provider
System
Database

Mazen Ahmed Samir 22100369

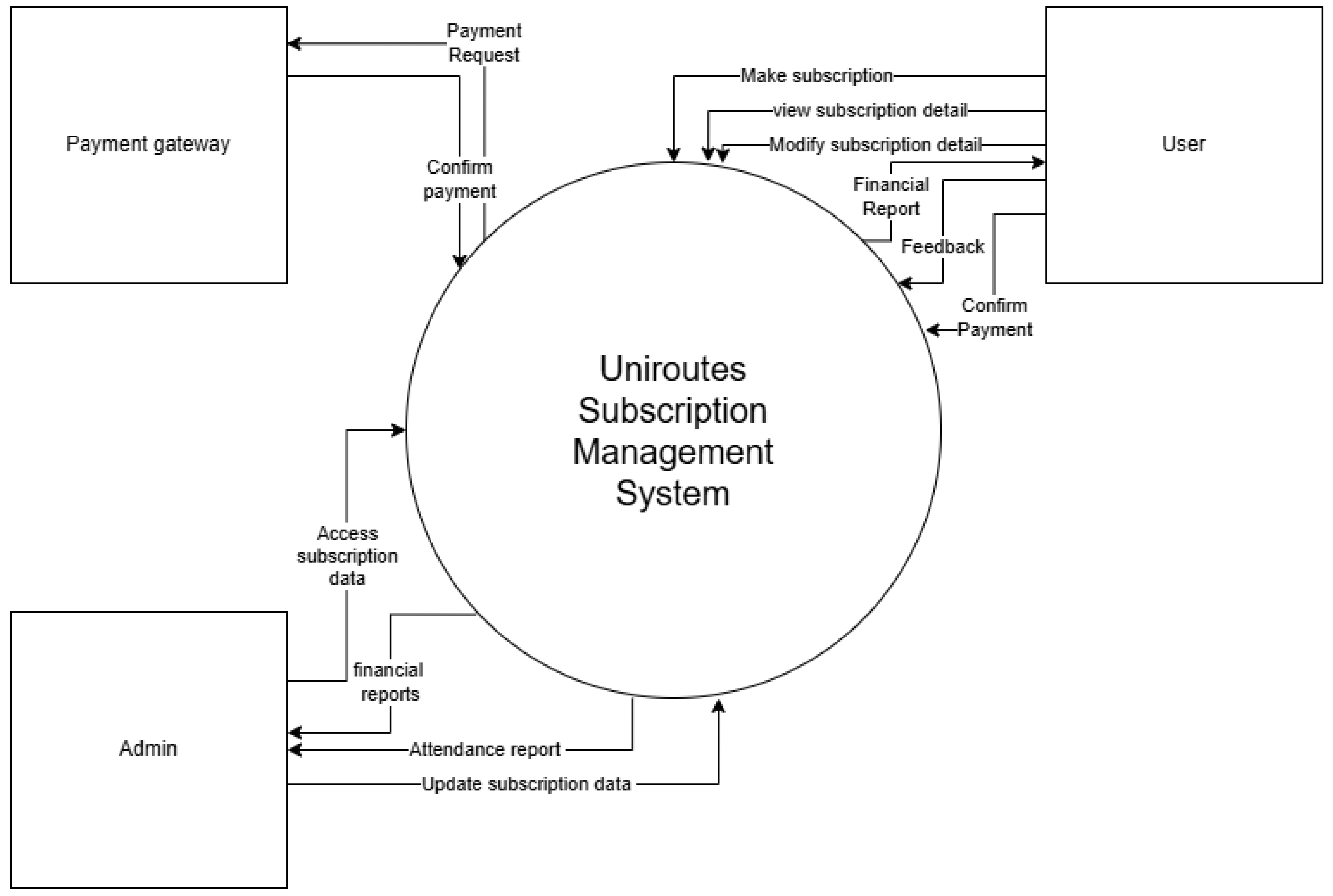
Use Case: Bus Management


Admin
AdminInput
BusManagementService
NotificationService
RealTimeTracking
BusCompany
BusAvailability
MaintenanceService
Bus
Route
Log in and Access Bus Management
Request to Manage Bus (Add/Update/Remove)
alt
[Add New Bus]
Request Bus Data (New Bus Details)
Bus Data (New Bus Details)
Check Bus Availability
Bus Availability Data (Available/Not Available)
alt
[Bus Available]
Assign Bus to Route
Bus Assigned to Route
Schedule Bus for Maintenance (if needed)
Maintenance Status (if scheduled)
Send Notifications to Admin, Bus Company, and Passengers
Notification: Bus Added Successfully
Notification: Bus Added
Update Bus Location and Route
Real-Time Bus Tracking Updated
[Bus Not Available]
Notify: No Available Bus for Route
Notification: Bus Not Available
[Update Existing Bus]
Request Bus Data (Update Bus Details)
Updated Bus Data
Check Bus Availability (Update)
Bus Availability (Updated Availability)
Schedule Bus for Maintenance (if needed)
Maintenance Status (if scheduled)
Send Update Notifications to Admin, Bus Company, Passengers
Notification: Bus Updated Successfully
Notification: Bus Updated
Update Bus Location and Route
Real-Time Bus Tracking Updated
[Remove Bus]
Request to Remove Bus
Bus Removal Confirmation
Update Bus Availability (Bus Removed)
Updated Bus Availability
Send Removal Notifications to Admin, Bus Company, and Passengers
Notification: Bus Removed Successfully
Notification: Bus Removed
Update Real-Time Bus Tracking (Remove Bus)
Real-Time Tracking Updated
Admin
AdminInput
BusManagementService
NotificationService
RealTimeTracking
BusCompany
BusAvailability
MaintenanceService
Bus
Route

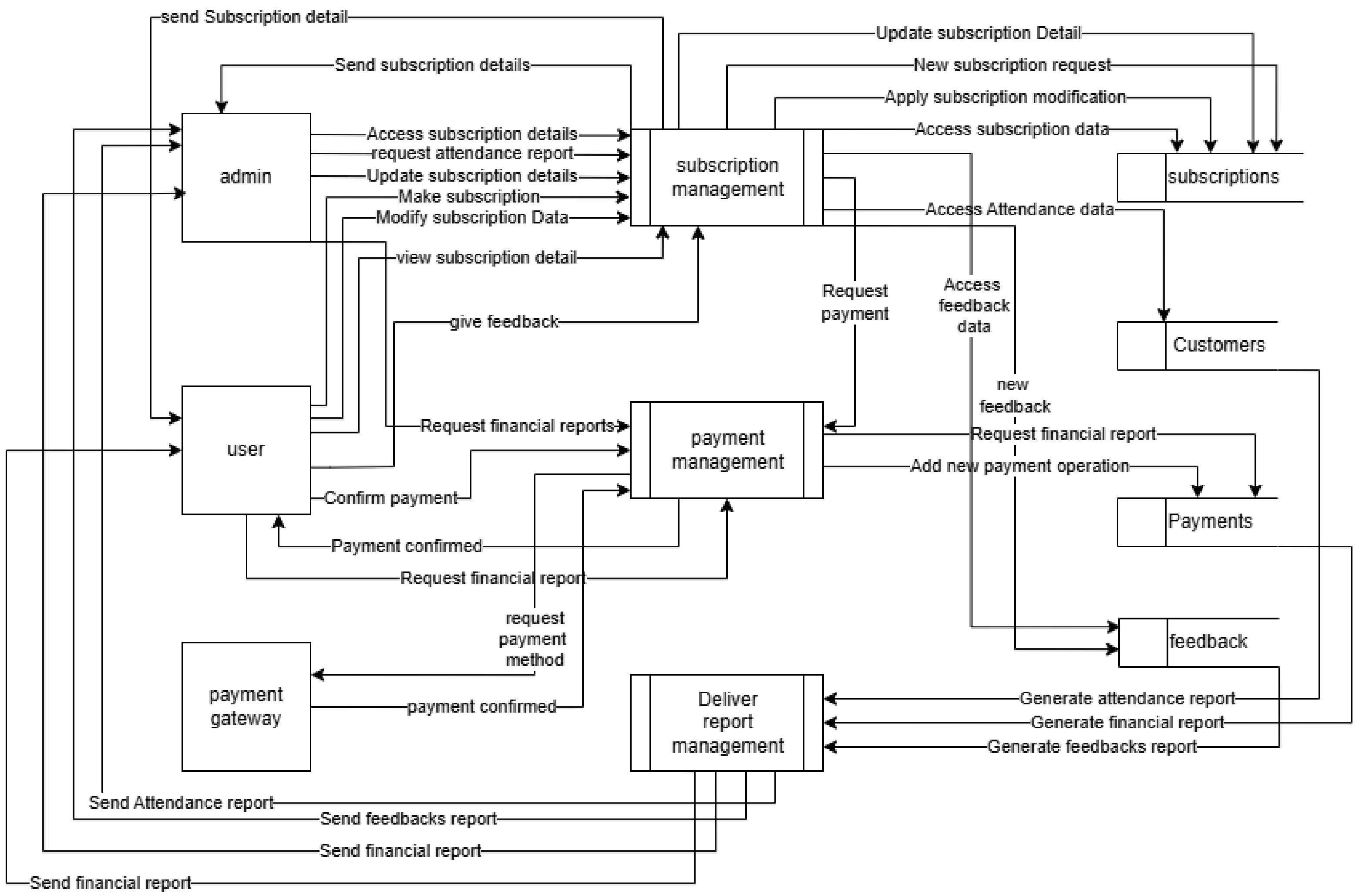
Belal Fathy 22101311

Use Case: Subscription Management

Level 0



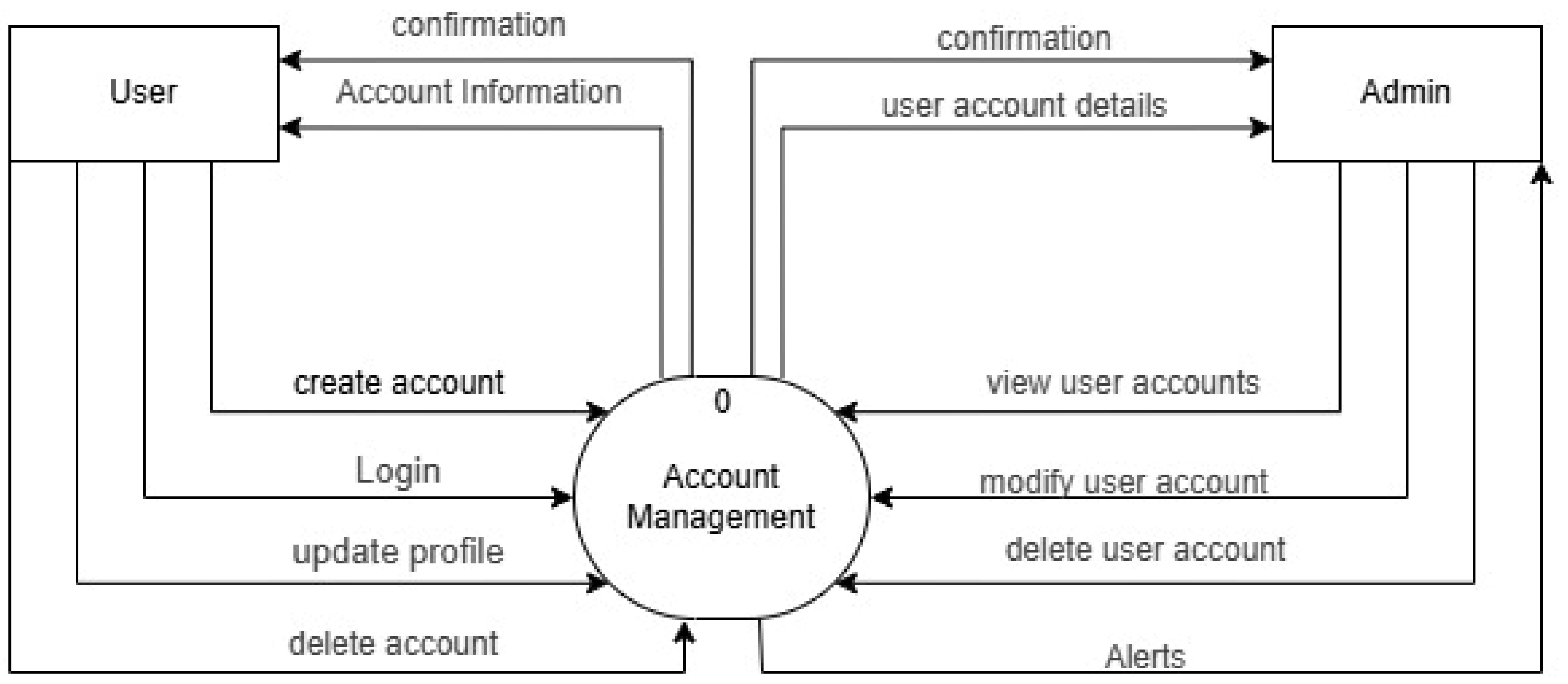
Level 1

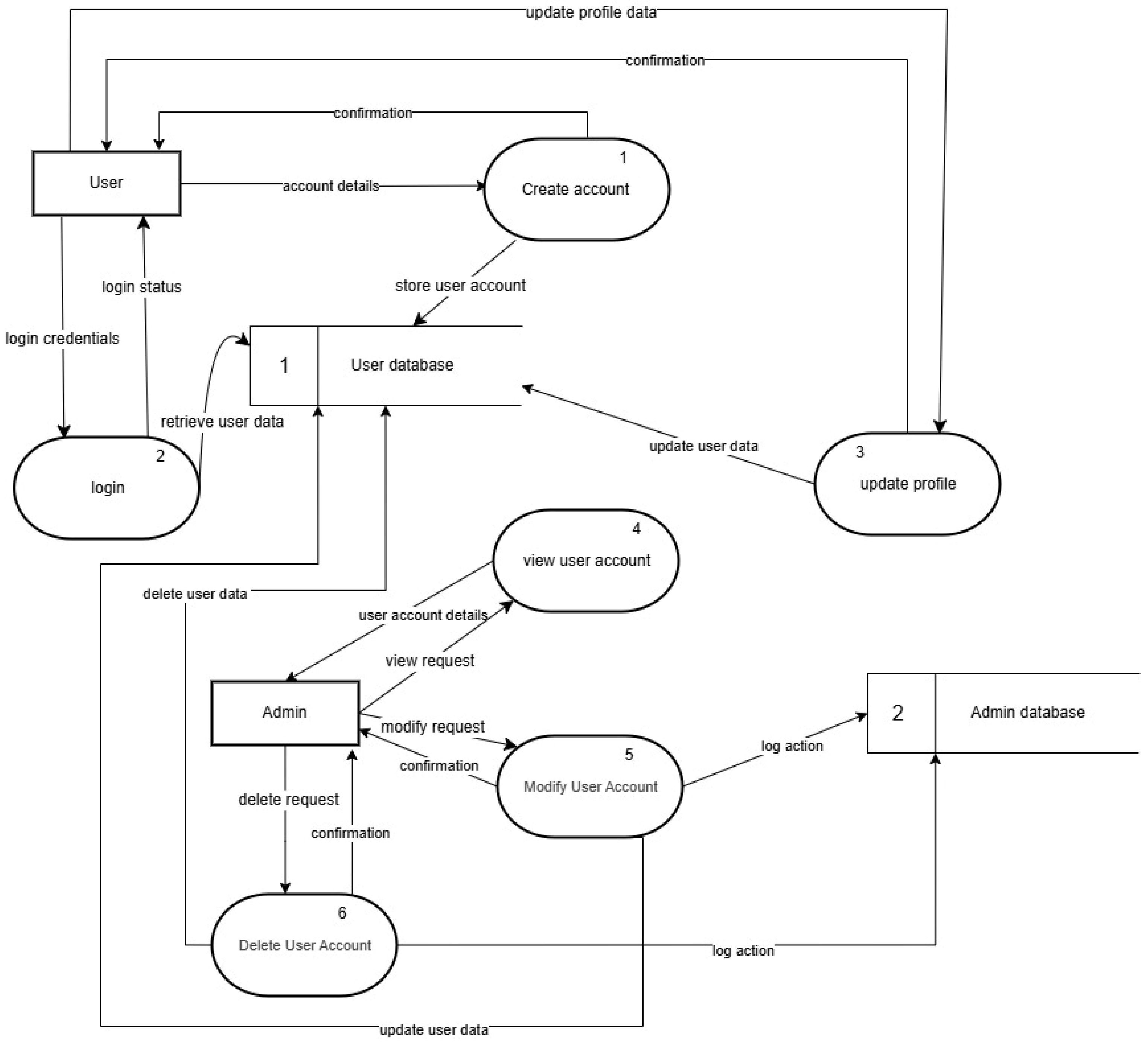

send Subscription detail
Send subscription details
Update subscription Detail
New subscription request
Apply subscription modification
Access subscription data
admin
Access subscription details
request attendance report
Update subscription details
Make subscription
Modify subscription Data
subscription management
subscriptions
Access Attendance data
view subscription detail
give feedback
Request payment
Access feedback data
Customers
new feedback
user
Request financial reports
payment management
Request financial report
Add new payment operation
Payments
Confirm payment
Payment confirmed
Request financial report
feedback
request payment method
payment gateway
payment confirmed
Deliver report management
Generate attendance report
Generate financial report
Generate feedbacks report
Send Attendance report
Send feedbacks report
Send financial report
Send financial report

Mahmoud Eid 22100680

Use Case: Account Management

DFD Account Management level 0

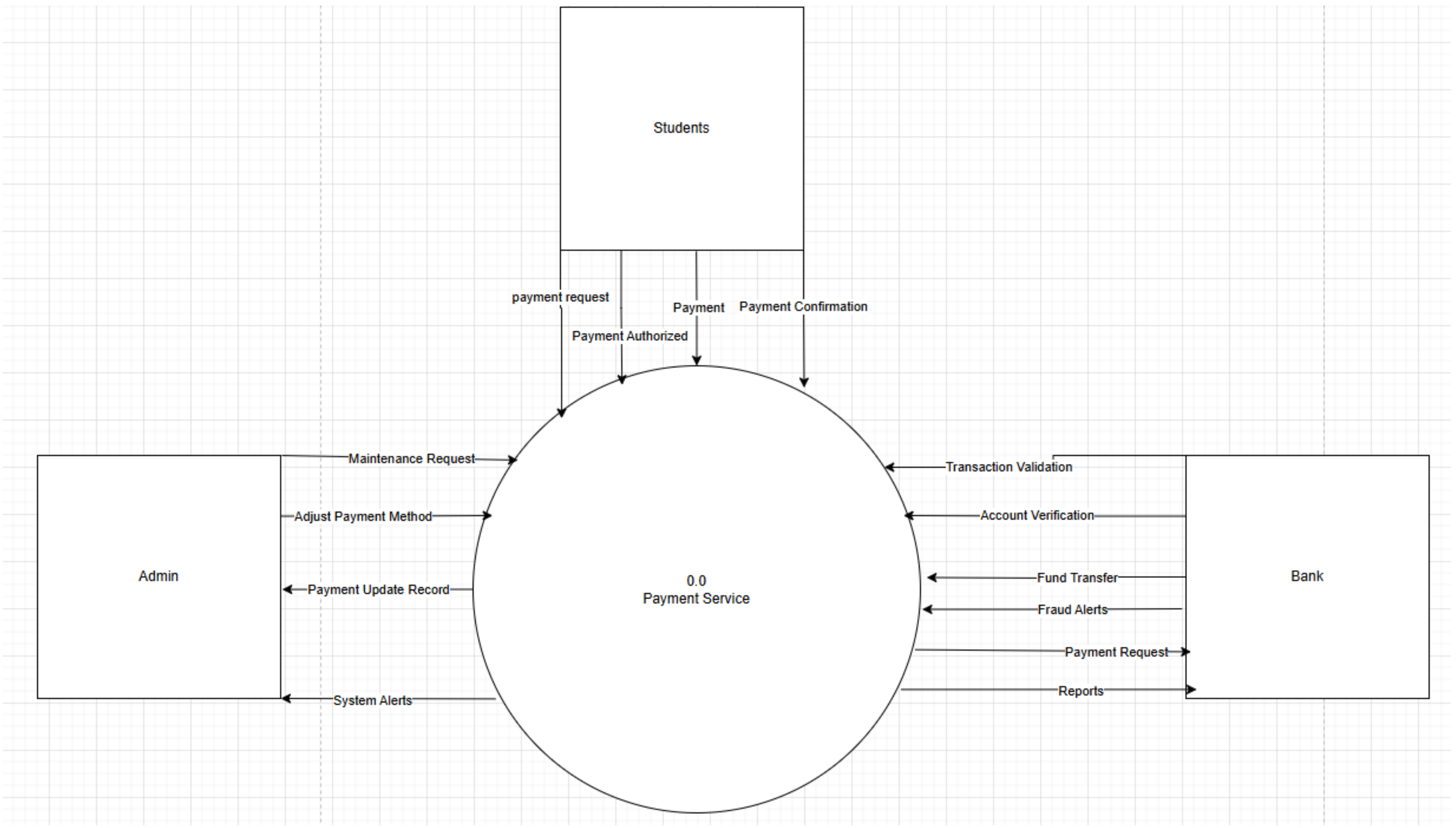



update profile data
confirmation
confirmation
User
account details
1
Create account
store user account
login status
login credentials
1
User database
retrieve user data
2
login
update user data
3
update profile
4
view user account
delete user data
user account details
view request
Admin
modify request
2
Admin database
confirmation
5
Modify User Account
log action
delete request
confirmation
6
Delete User Account
log action
update user data

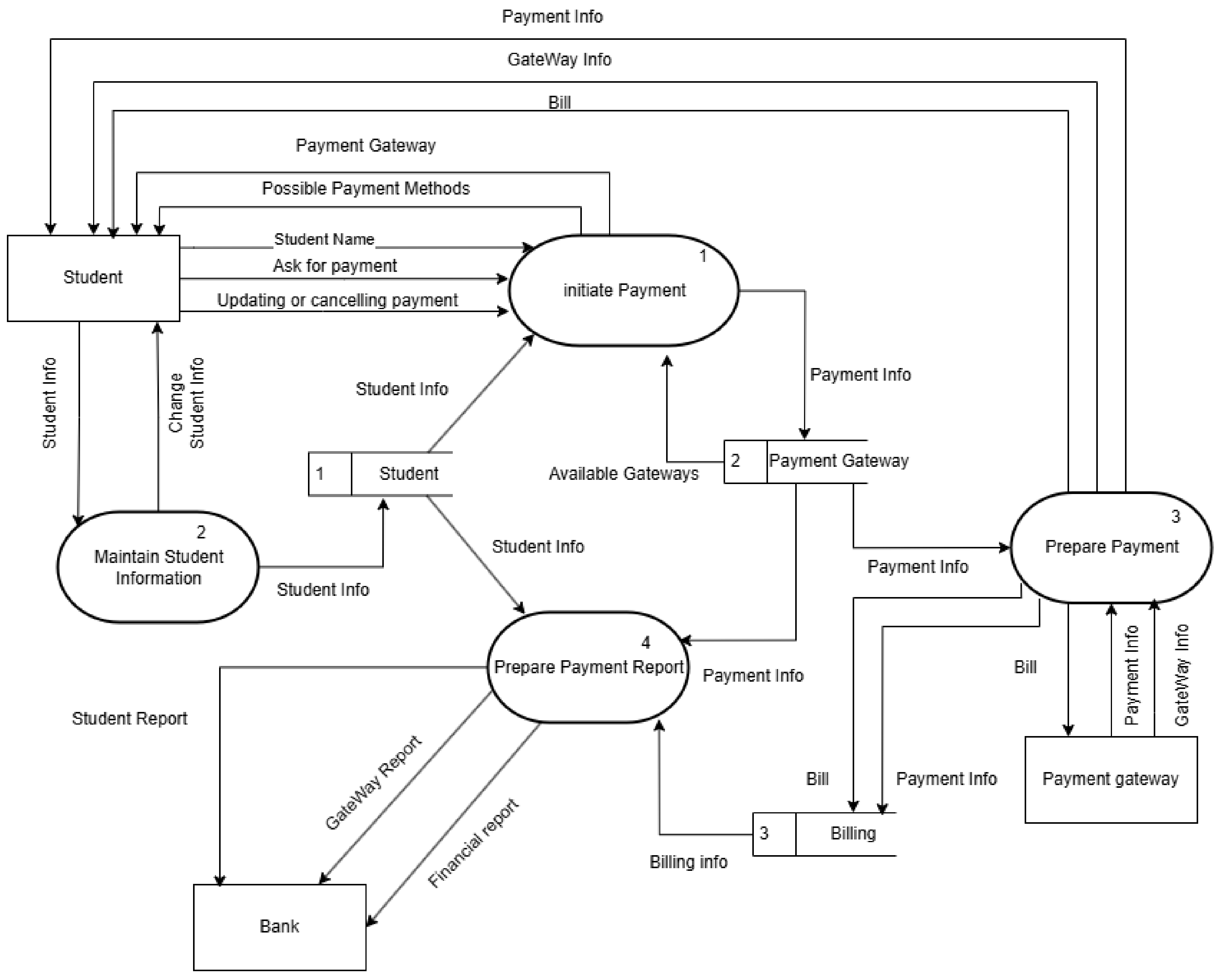
Ahmed Islam Farouk 22101008

Use Case: Payment Management

Level 0



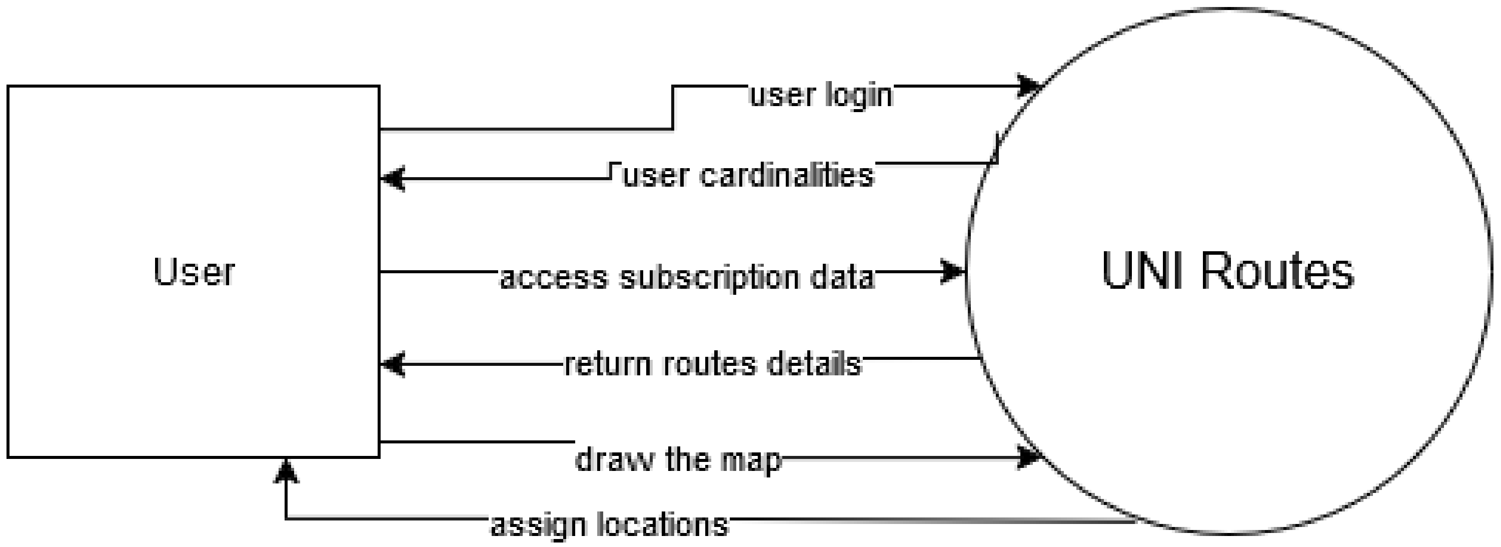
Level 1


Payment Info
GateWay Info
Bill
Payment Gateway
Possible Payment Methods
Student Name
Ask for payment
Updating or cancelling payment
Student
1
initiate Payment
Student Info
Change Student Info
Student Info
Payment Info
1
Student
Available Gateways
2
Payment Gateway
2
Maintain Student Information
Student Info
Student Info
3
Prepare Payment
Payment Info
4
Prepare Payment Report
Payment Info
Bill
Payment Info
GateWay Info
Student Report
GateWay Report
Financial report
Bill
Payment Info
Payment gateway
3
Billing
Billing info
Bank

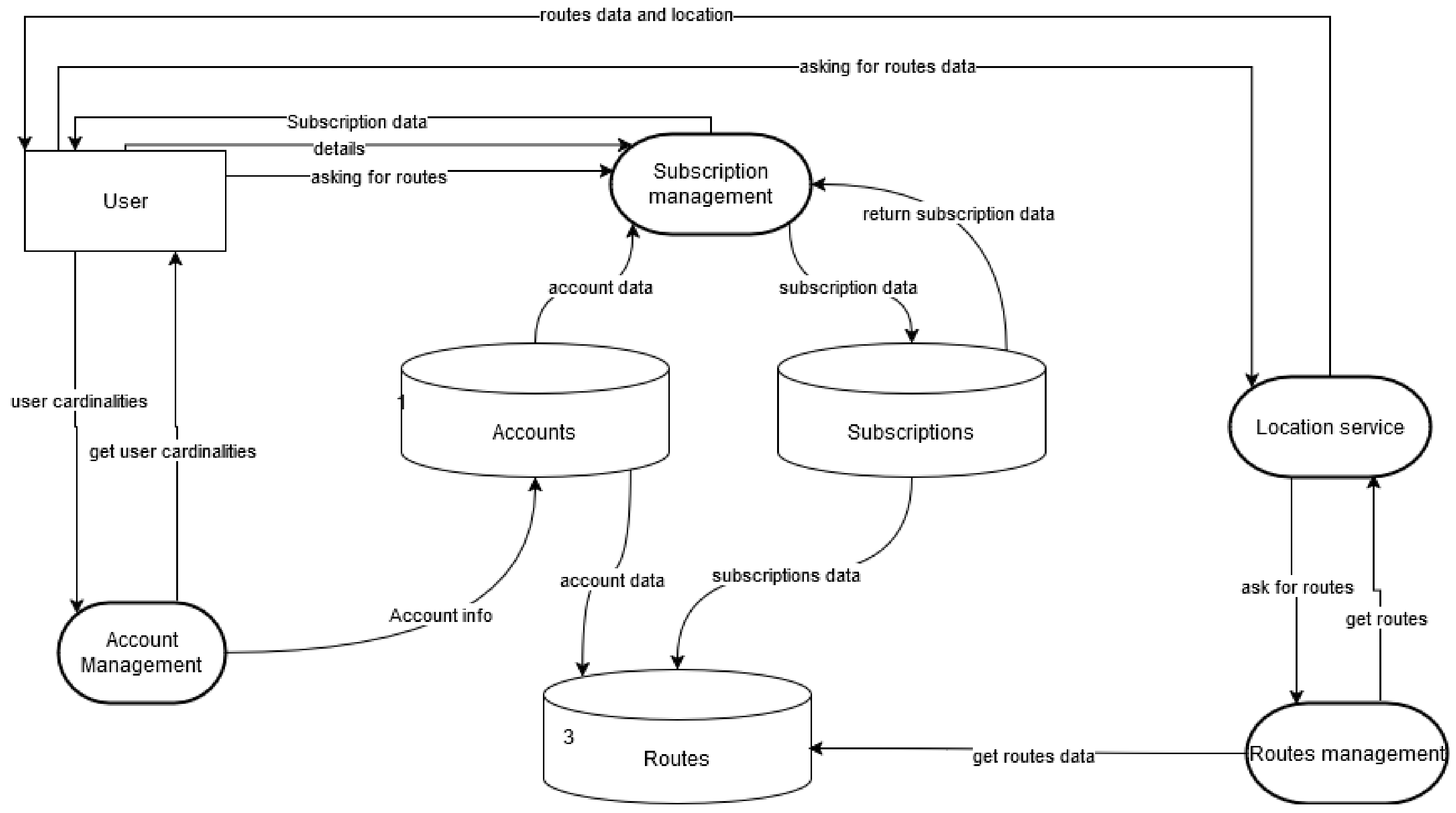
Abdallah Basem 22100848

Use Case: Navigation Management

Level 0

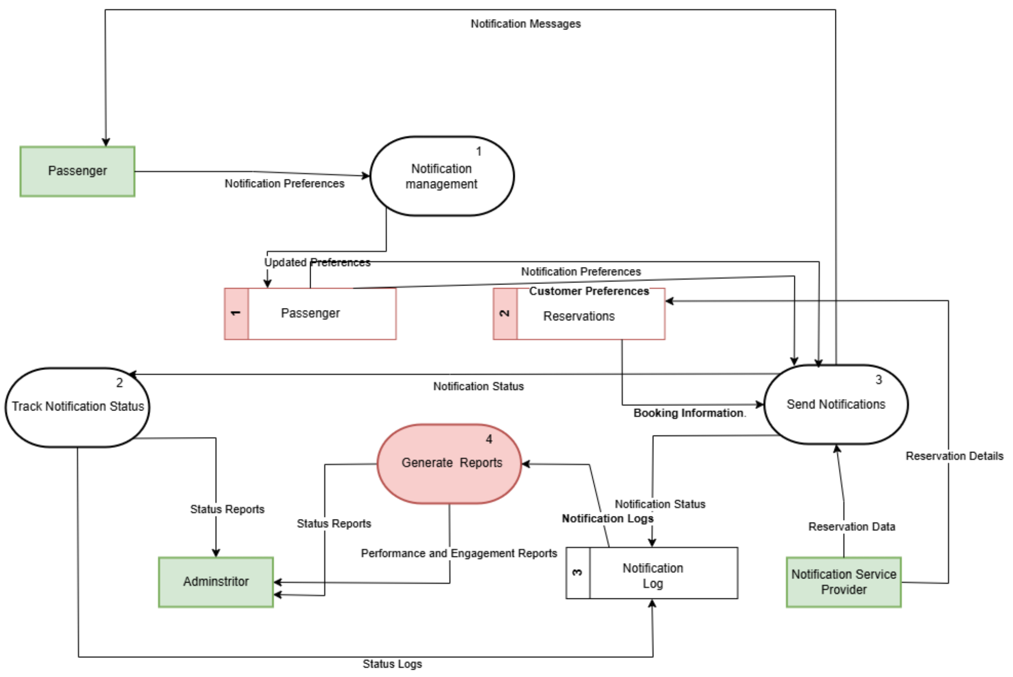


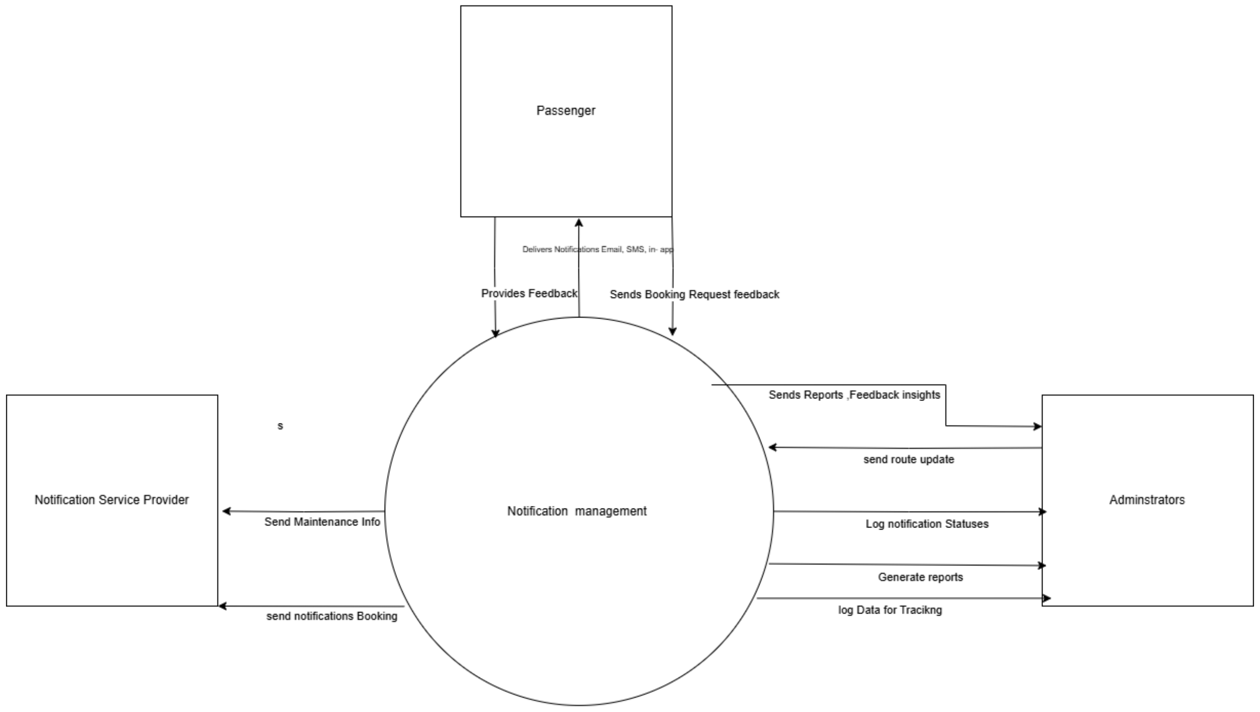
Level 1



Eyad Metwally 22100757

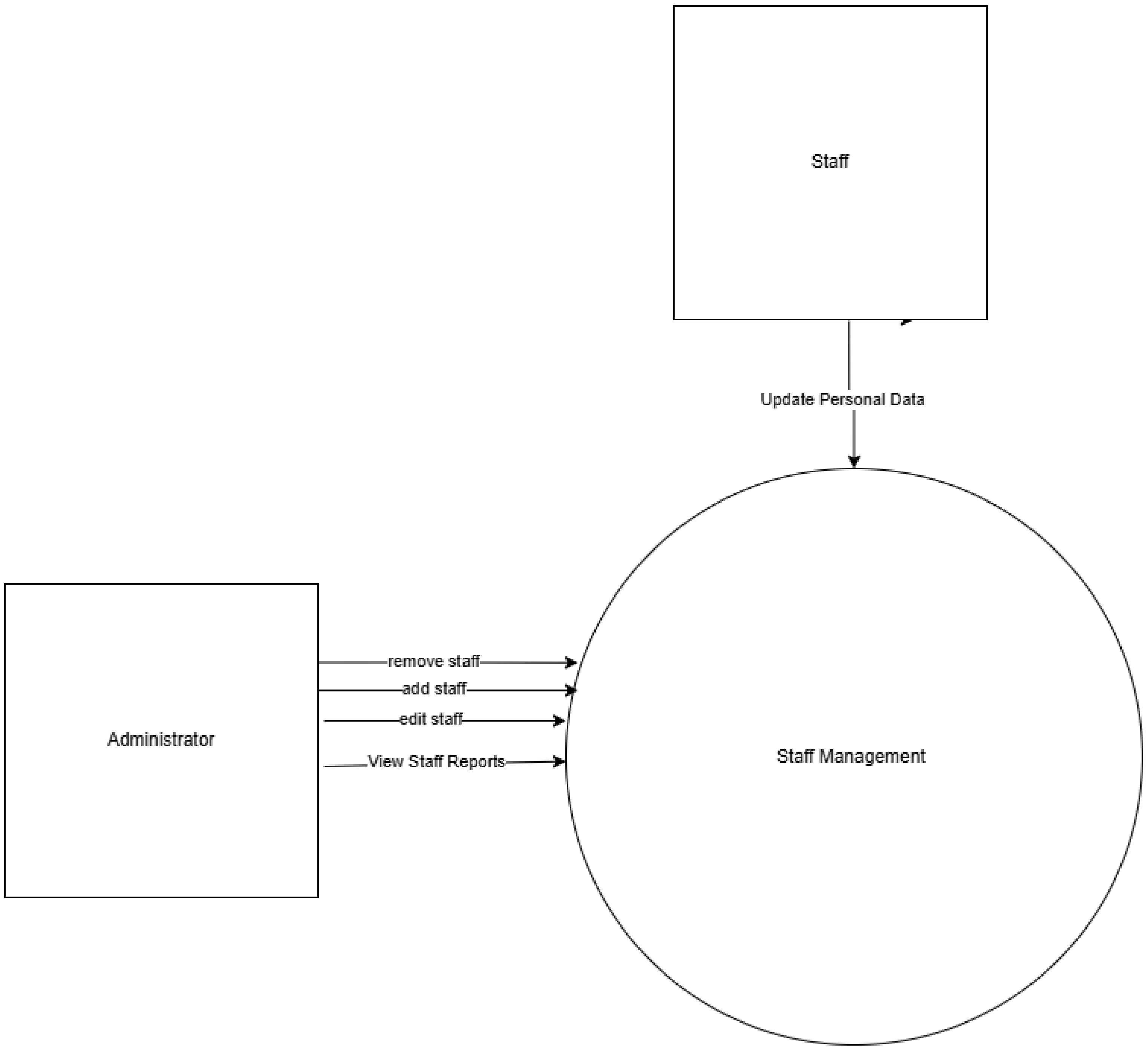
Use case: Notification

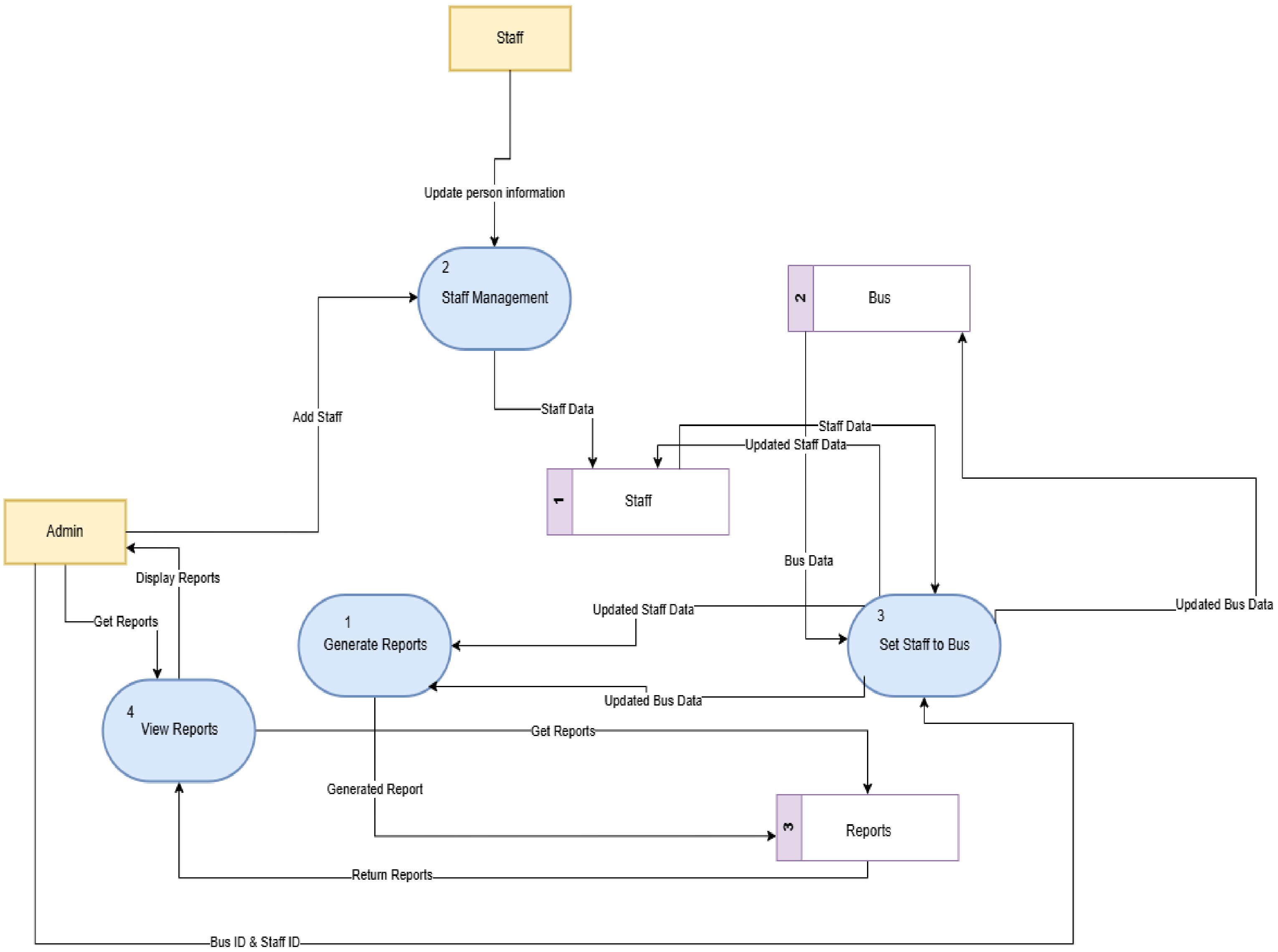



Passenger
Delivers Notifications Email, SMS, in- app
Provides Feedback
Sends Booking Request feedback
Sends Reports ,Feedback insights
s
Notification Service Provider
send route update
Notification management
Adminstrators
Send Maintenance Info
Log notification Statuses
Generate reports
send notifications Booking
log Data for Tracikng

Mazen Ahmed Samir 22100369

Use Case: Staff Management

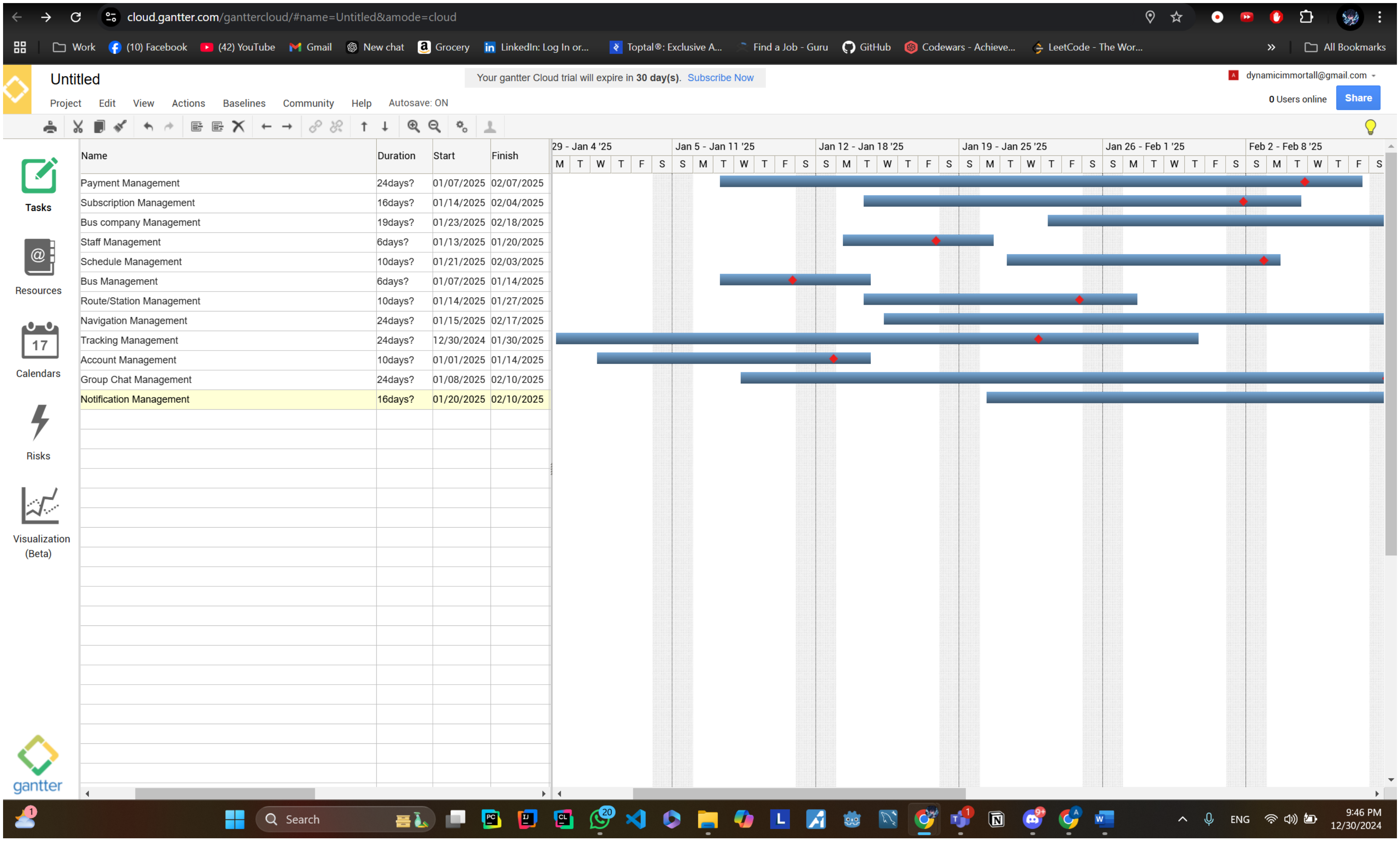



Staff
Update person information
2
Staff Management
Add Staff
Staff Data
2
Bus
Staff Data
Updated Staff Data
1
Staff
Admin
Display Reports
Bus Data
Get Reports
1
Generate Reports
Updated Staff Data
3
Set Staff to Bus
Updated Bus Data
4
View Reports
Updated Bus Data
Get Reports
Generated Report
3
Reports
Return Reports
Bus ID & Staff ID

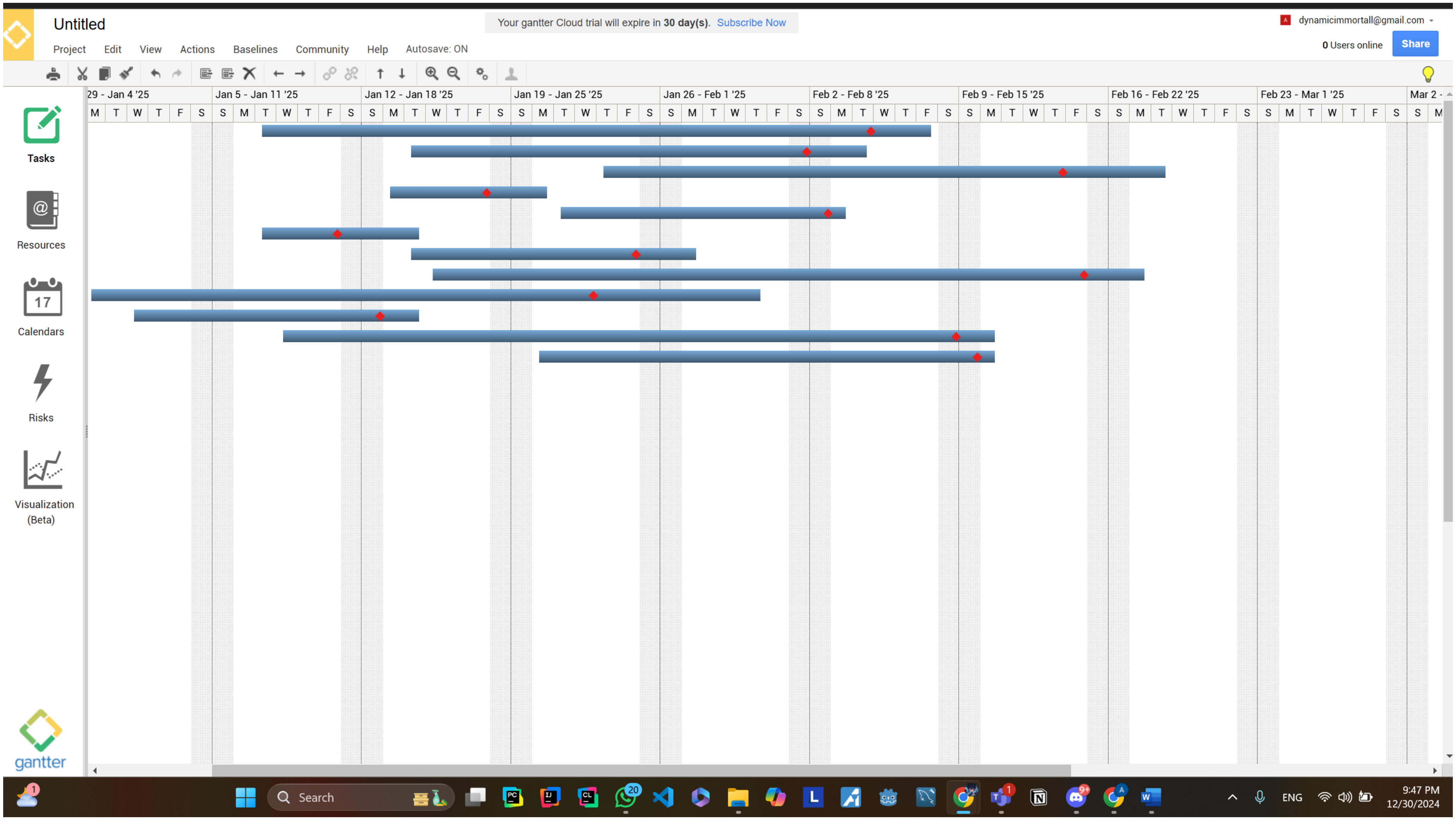
Use Case Estimation Each point is 2 days

Use Case	**Estimation(Fibonacci)**
Payment Management	12
Subscription Management	8
Bus company Management	3
Staff Management	3
Schedule Management	5
Bus Management	3
Route/Station Management	5
Navigation Management	12
Tracking Management	12
Account Management	5
Group Chat Management	12
Notification Management	8

Gantt chart


cloud.gantter.com/gantercloud/#name=Untitled&amode=cloud
Untitled
Your gantter Cloud trial will expire in 30 day(s). Subscribe Now
Project
Edit
View
Actions
Baselines
Community
Help
Autosave: ON
dynamicimmortall@gmail.com
0 Users online
Share
Tasks
Resources
Calendars
Risks
Visualization (Beta)
gantter

Name	Duration	Start	Finish
Payment Management	24days?	01/07/2025	02/07/2025
Subscription Management	16days?	01/14/2025	02/04/2025
Bus company Management	19days?	01/23/2025	02/18/2025
Staff Management	6days?	01/13/2025	01/20/2025
Schedule Management	10days?	01/21/2025	02/03/2025
Bus Management	6days?	01/07/2025	01/14/2025
Route/Station Management	10days?	01/14/2025	01/27/2025
Navigation Management	24days?	01/15/2025	02/17/2025
Tracking Management	24days?	12/30/2024	01/30/2025
Account Management	10days?	01/01/2025	01/14/2025
Group Chat Management	24days?	01/08/2025	02/10/2025
Notification Management	16days?	01/20/2025	02/10/2025


Untitled
Your gantter Cloud trial will expire in 30 day(s). Subscribe Now
dynamicimmortall@gmail.com
0 Users online
Share
Project
Edit
View
Actions
Baselines
Community
Help
Autosave: ON
Tasks
Resources
Calendars
Risks
Visualization (Beta)
29 - Jan 4 '25
Jan 5 - Jan 11 '25
Jan 12 - Jan 18 '25
Jan 19 - Jan 25 '25
Jan 26 - Feb 1 '25
Feb 2 - Feb 8 '25
Feb 9 - Feb 15 '25
Feb 16 - Feb 22 '25
Feb 23 - Mar 1 '25
Mar 2 -
gantter
Search
ENG
9:47 PM
12/30/2024

UniRoutes Test Plan

This test plan covers:

1. Unit Testing: Testing individual classes and methods.
2. Integration Testing: Testing interactions between subsystems.
3. System Testing: Testing the complete system functionality.
4. Acceptance Testing: Verifying that business requirements are met.

Modules to Test:

1. Bus Company Management

   Add, update, and delete bus companies.

2. Driver and Staff Management

   Add, remove, and update driver and supervisor profiles.

   Assign staff to routes and chat groups.

3. Bus Route and Stops Management

   Manage routes, stops, and schedules.

4. Subscription Management

   Reserve and manage bookings.

5. Notifications and Alerts

   Manage and send notifications for delays, cancellations, and emergencies.

6. Real-Time Tracking

   Display real-time bus locations.

7. Emergency and Safety Features

   Alert dispatchers with emergency notifications.

8. Feedback System

   Collect feedback from users post-travel.

9. Payment Management

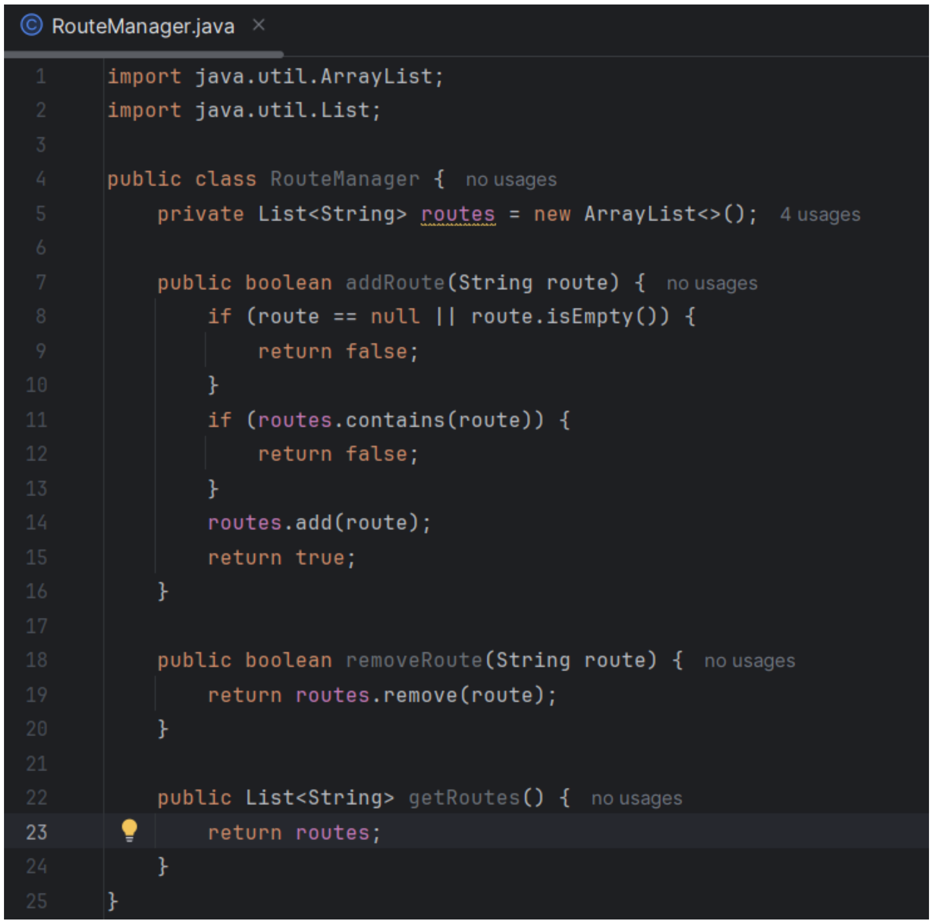
   Process payments and handle refunds.

Testing Strategies

1. Unit Testing

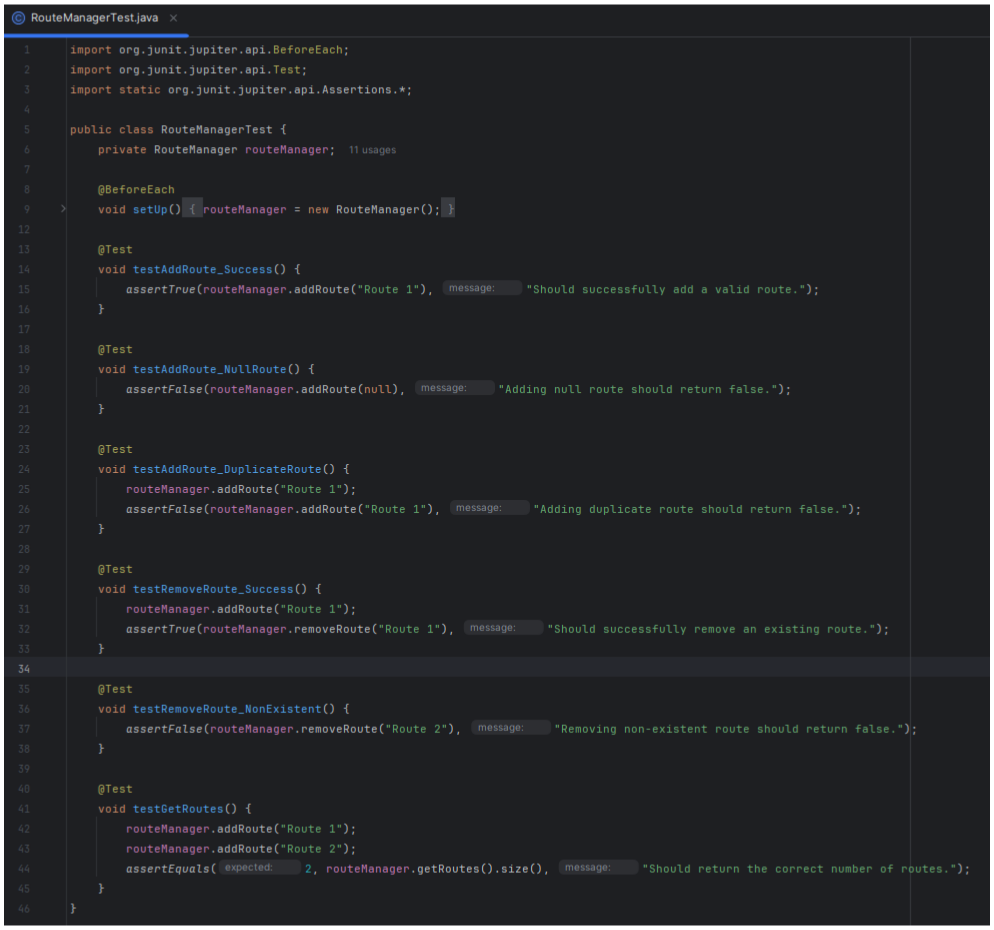
Tools: JUnit, TestNG.
Examples:

- Class: RouteManager
    - Method: addRoute()
        - Test Case 1: Provide valid route details; expect success.
        - Test Case 2: Provide a null object; expect failure.
    - Method: removeRoute()
        - Test Case 1: Remove an existing route; expect success.
        - Test Case 2: Attempt to remove a non-existent route; expect failure.
    - Code:
        - RouteManger:


RouteManager.java

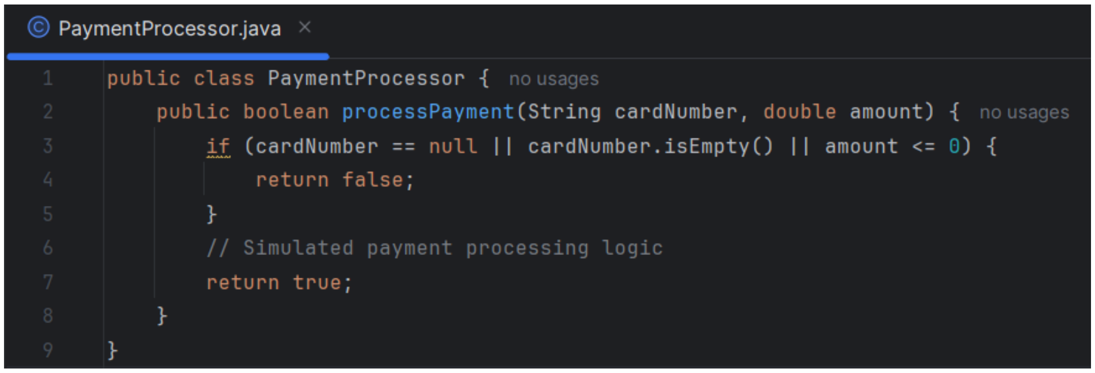
```java
import java.util.ArrayList;
import java.util.List;

public class RouteManager {  no usages
    private List<String> routes = new ArrayList<>();  4 usages

    public boolean addRoute(String route) {  no usages
        if (route == null || route.isEmpty()) {
            return false;
        }
        if (routes.contains(route)) {
            return false;
        }
        routes.add(route);
        return true;
    }

    public boolean removeRoute(String route) {  no usages
        return routes.remove(route);
    }

    public List<String> getRoutes() {  no usages
        return routes;
    }
}
```

- RouteManagerTest:



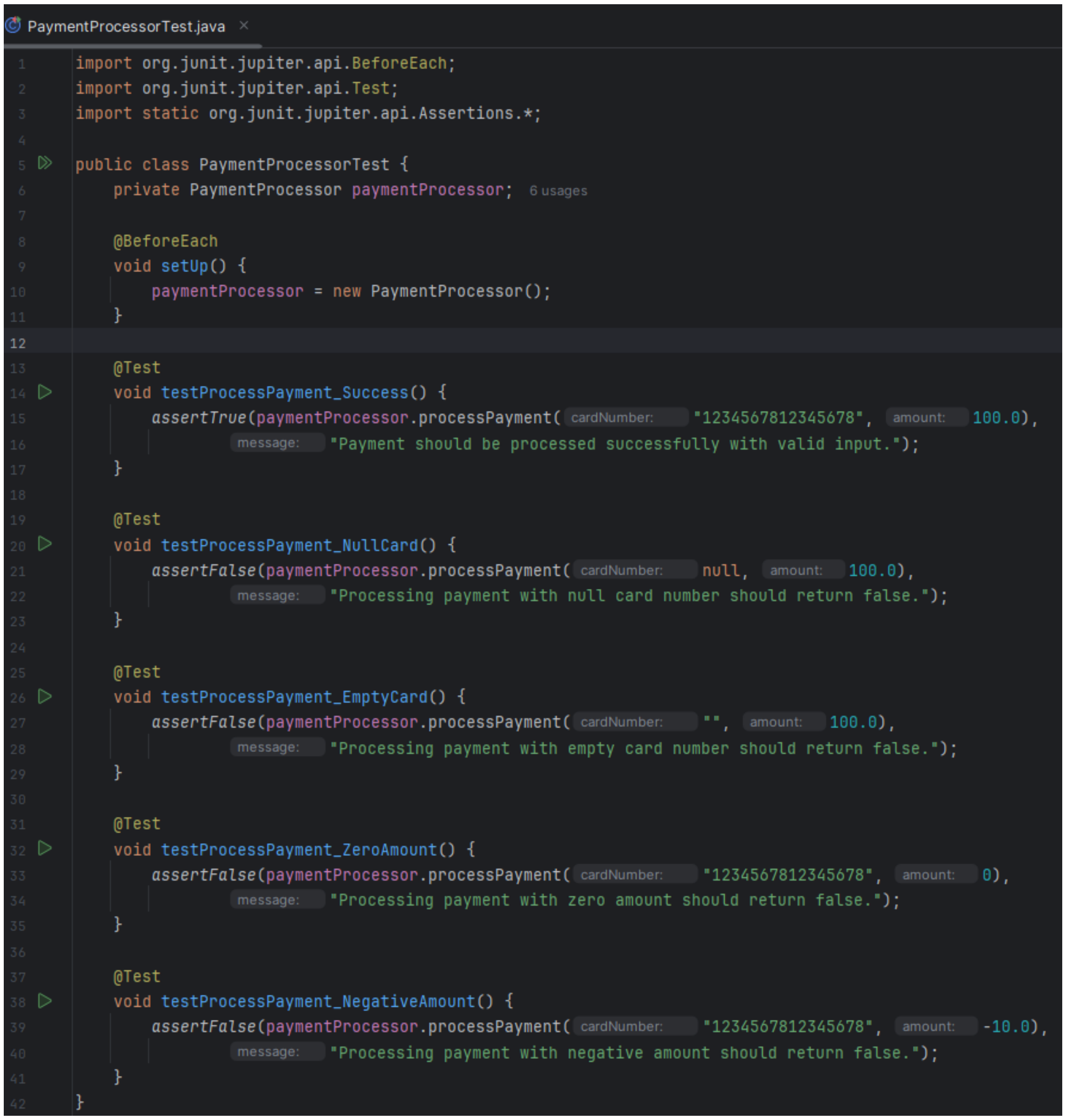
```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class RouteManagerTest {
    private RouteManager routeManager;  11 usages

    @BeforeEach
    void setUp() { routeManager = new RouteManager(); }

    @Test
    void testAddRoute_Success() {
        assertTrue(routeManager.addRoute("Route 1"),  message: "Should successfully add a valid route.");
    }

    @Test
    void testAddRoute_NullRoute() {
        assertFalse(routeManager.addRoute(null),  message: "Adding null route should return false.");
    }

    @Test
    void testAddRoute_DuplicateRoute() {
        routeManager.addRoute("Route 1");
        assertFalse(routeManager.addRoute("Route 1"),  message: "Adding duplicate route should return false.");
    }

    @Test
    void testRemoveRoute_Success() {
        routeManager.addRoute("Route 1");
        assertTrue(routeManager.removeRoute("Route 1"),  message: "Should successfully remove an existing route.");
    }

    @Test
    void testRemoveRoute_NonExistent() {
        assertFalse(routeManager.removeRoute("Route 2"),  message: "Removing non-existent route should return false.");
    }

    @Test
    void testGetRoutes() {
        routeManager.addRoute("Route 1");
        routeManager.addRoute("Route 2");
        assertEquals( expected: 2, routeManager.getRoutes().size(),  message: "Should return the correct number of routes.");
    }
}
```

- Class: PaymentProcessor
  - Method: processPayment()
    - Test Case 1: Process payment with valid details; expect success.
    - Test Case 2: Insufficient funds; expect failure and error message.
  - Code:
    - PaymentProcessor:



```
public class PaymentProcessor {  no usages
    public boolean processPayment(String cardNumber, double amount) {  no usages
        if (cardNumber == null || cardNumber.isEmpty() || amount <= 0) {
            return false;
        }
        // Simulated payment processing logic
        return true;
    }
}
```

- PaymentProcessorTest:



```
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class PaymentProcessorTest {
    private PaymentProcessor paymentProcessor;  6 usages

    @BeforeEach
    void setUp() {
        paymentProcessor = new PaymentProcessor();
    }

    @Test
    void testProcessPayment_Success() {
        assertTrue(paymentProcessor.processPayment( cardNumber: "1234567812345678",  amount: 100.0),
                 message: "Payment should be processed successfully with valid input.");
    }

    @Test
    void testProcessPayment_NullCard() {
        assertFalse(paymentProcessor.processPayment( cardNumber: null,  amount: 100.0),
                 message: "Processing payment with null card number should return false.");
    }

    @Test
    void testProcessPayment_EmptyCard() {
        assertFalse(paymentProcessor.processPayment( cardNumber: "",  amount: 100.0),
                 message: "Processing payment with empty card number should return false.");
    }

    @Test
    void testProcessPayment_ZeroAmount() {
        assertFalse(paymentProcessor.processPayment( cardNumber: "1234567812345678",  amount: 0),
                 message: "Processing payment with zero amount should return false.");
    }

    @Test
    void testProcessPayment_NegativeAmount() {
        assertFalse(paymentProcessor.processPayment( cardNumber: "1234567812345678",  amount: -10.0),
                 message: "Processing payment with negative amount should return false.");
    }
}
```

2. Integration Testing
Tools: Mockito, JUnit.

- Subsystems:
    1. GPS System and Notification System
    2. Booking System and Payment Gateway

3. System Testing

1. Scenario: Student booking workflow
2. Scenario: Emergency Alert

4. Acceptance Testing

Objective: Ensure the system meets business requirements.
Actors: Students, admins, and supervisors.

To verify that the system meets business requirements, we will conduct thorough acceptance testing based on predefined user stories and requirements. This involves simulating real-world scenarios, such as route management by admins and feedback submission by students, and validating the results against the expected outcomes. Key business processes like creating and updating routes, as well as collecting and analyzing user feedback, will be tested to ensure they align with organizational goals and provide value to end-users. Continuous feedback from stakeholders will guide iterative improvements and ensure compliance with business needs.

UniRoutes Test Plan

1. Bus Company Management

Unit Tests

Unit Test ID	**Method Name**	**Input Example**	**Expected Output**
U1	AddCompany	Name: "XYZ Buses"	Company "XYZ Buses" added successfully.
		Address: "123 Main St"	
U2	UpdateCompany	CompanyID: 101	Details updated for CompanyID 101.
		New Name: "ABC Transport"	
U3	DeleteCompany	CompanyID: 102	CompanyID 102 removed successfully.

Unit Test ID	Method Name	Input Example	Expected Output
U4	GetCompanies	None	List of all bus companies displayed.

2. Driver and Staff Management

Unit Tests

Unit Test ID	Method Name	Input Example	Expected Output
U5	AddDriver	Name: "John Doe"	Driver "John Doe" added successfully.
		LicenseID: "D12345"	
U6	UpdateDriver	DriverID: 201	DriverID 201 details updated successfully.
		New Phone: "9876543210"	
U7	DeleteDriver	DriverID: 202	DriverID 202 removed successfully.
U8	GetDrivers	None	List of all drivers displayed.
U9	AddSupervisor	Name: "Alice Smith"	Supervisor "Alice Smith" added successfully.
		Contact: "1234567890"	
U10	UpdateSupervisor	SupervisorID: 301	SupervisorID 301 details updated successfully.
		New Email: "alice@example.com"	
U11	DeleteSupervisor	SupervisorID: 302	SupervisorID 302 removed successfully.
U12	GetSupervisors	None	List of all supervisors displayed.

3. Bus Route and Stops Management

Unit Tests

Unit Test ID	**Method Name**	**Input Example**	**Expected Output**
U13	AddRoute	Name: "Route A"	Route "Route A" added successfully.
		Stops: ["Station 1", "Station 2"]	
U14	UpdateRoute	RouteID: 401	RouteID 401 updated with new stops.
		New Stops: ["Station 3"]	
U15	DeleteRoute	RouteID: 402	RouteID 402 removed successfully.
U16	GetRoutes	None	List of all routes displayed.
U17	AddStop	RouteID: 401, Stop: "Station 4"	Stop "Station 4" added to RouteID 401.
U18	UpdateStop	StopID: 501	StopID 501 details updated.
		New Name: "Main Plaza"	
U19	DeleteStop	StopID: 502	StopID 502 removed successfully.
U20	GetStops	RouteID: 401	List of stops for RouteID 401 displayed.

4. User Roles and Permissions

Unit Tests

Unit Test ID	**Method Name**	**Input Example**	**Expected Output**
U21	AssignRole	UserID: 601, Role: "Supervisor"	Role "Supervisor" assigned to UserID 601.
U22	UpdateRole	UserID: 602, Role: "Admin"	Role updated to "Admin" for UserID 602.
U23	RemoveRole	UserID: 603	Role removed for UserID 603.
U24	GetRoles	None	List of user roles displayed.

5. Advanced Subscription Management

Unit Tests

Unit Test ID	Method Name	Input Example	Expected Output
U25	ReserveSeat	UserID: 701, RouteID: 401, Seat: 10	Seat 10 on RouteID 401 reserved for UserID 701.
U26	CancelReservation	ReservationID: 801	ReservationID 801 cancelled successfully.
U27	GetReservations	UserID: 701	List of reservations for UserID 701 displayed.
U28	SendReminder	ReservationID: 801	Reminder sent for ReservationID 801.

6. Notifications and Alerts

Unit Tests

Unit Test ID	Method Name	Input Example	Expected Output
U29	SendGlobalAlert	Message: "System maintenance"	Alert sent to all users successfully.
U30	SendRouteAlert	RouteID: 401, Message: "Delay"	Alert sent to users on RouteID 401.
U31	SendBusAlert	BusID: 301, Message: "Breakdown"	Alert sent to users of BusID 301.
U32	GetAlerts	UserID: 701	List of alerts received by UserID 701.

7. Real-Time Bus Tracking

Unit Tests

Unit Test ID	Method Name	Input Example	Expected Output
U33	GetBusLocation	BusID: 301	Real-time location of BusID 301 displayed.
U34	GetRouteETAs	RouteID: 401	ETAs for RouteID 401 stops displayed.

8. Emergency and Safety Features

Unit Tests

Unit Test ID	**Method Name**	**Input Example**	**Expected Output**
U35	SendEmergencyAlert	UserID: 701, Location: "Station 1"	Emergency alert sent successfully.
U36	DeployAssistance	AlertID: 901	Assistance deployed to location in AlertID 901.

Execution Plan

1. Unit Tests

- Unit Test U1
- Unit Test U2
- Unit Test U3
- Unit Test U4
- Unit Test U5
- Unit Test U6
- Unit Test U7
- Unit Test U8
- Unit Test U9
- Unit Test U10
- Unit Test U11
- Unit Test U12
- Unit Test U13
- Unit Test U14
- Unit Test U15
- Unit Test U16
- Unit Test U17
- Unit Test U18
- Unit Test U19
- Unit Test U20

- Unit Test U21
- Unit Test U22
- Unit Test U23
- Unit Test U24
- Unit Test U25
- Unit Test U26
- Unit Test U27
- Unit Test U28
- Unit Test U29
- Unit Test U30
- Unit Test U31
- Unit Test U32
- Unit Test U33
- Unit Test U34
- Unit Test U35
- Unit Test U36

2. Integration Testing

- Integration Testing for U1, U6
- Integration Testing for U2, U3, U6, U1
- Integration Testing for U5, U6, U2, U1
- Integration Testing for U4, U6, U1

3. Regression Testing

- Regression Testing for U6
- Regression Testing for U1
- Regression Testing for U2
- Regression Testing for U3
- Regression Testing for U4
- Regression Testing for U5

- Regression Testing for U1, U6
- Regression Testing for U6, U2, U3, U1
- Regression Testing for U6, U5, U2, U1
- Regression Testing for U6, U4, U1

4. Resources

- Testers: 3 testers.
- Tools: Junit, Selenium.