# Report 2
# Detailed Design Document



## 1. Student


| | |
|---|---|
| Name | Minh Nguyen Quang Le |
| Email | minh.le4@ucalgary.ca |
| UCID | 30295926 |


## 2. System Name

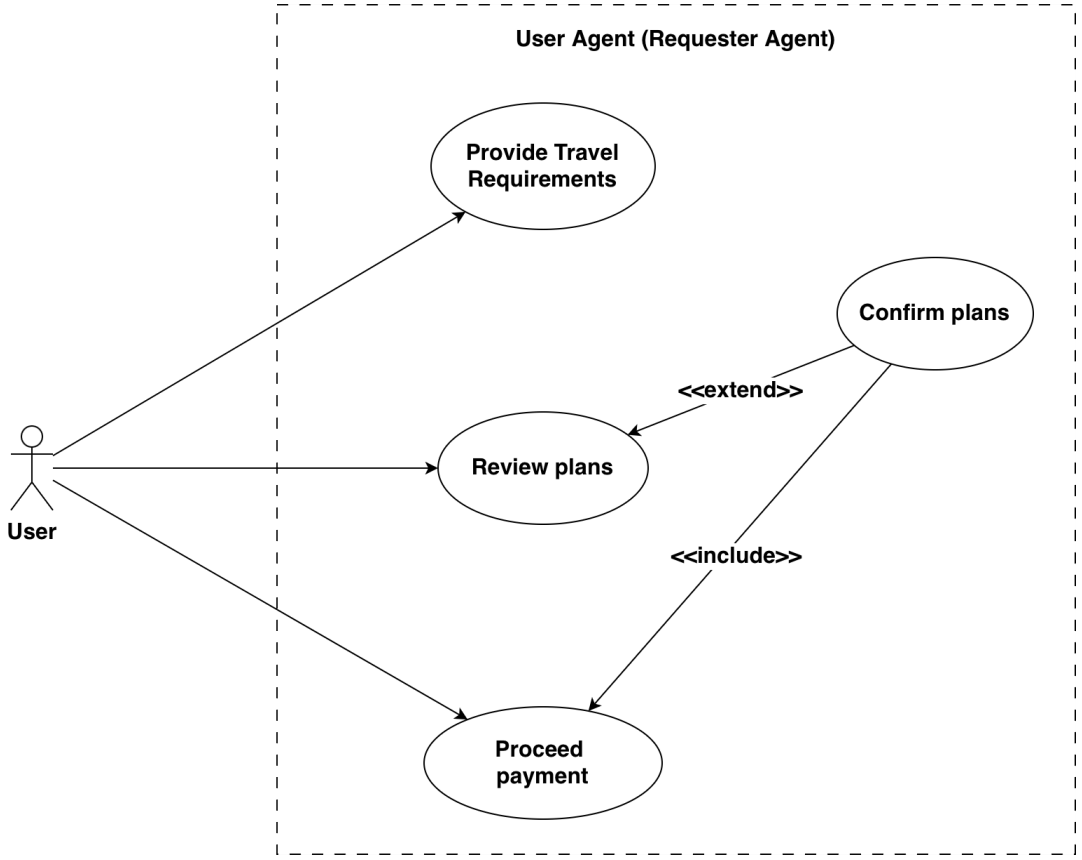
Simple Travel Planner (using GAIA methodology)

## 3. Detailed Design

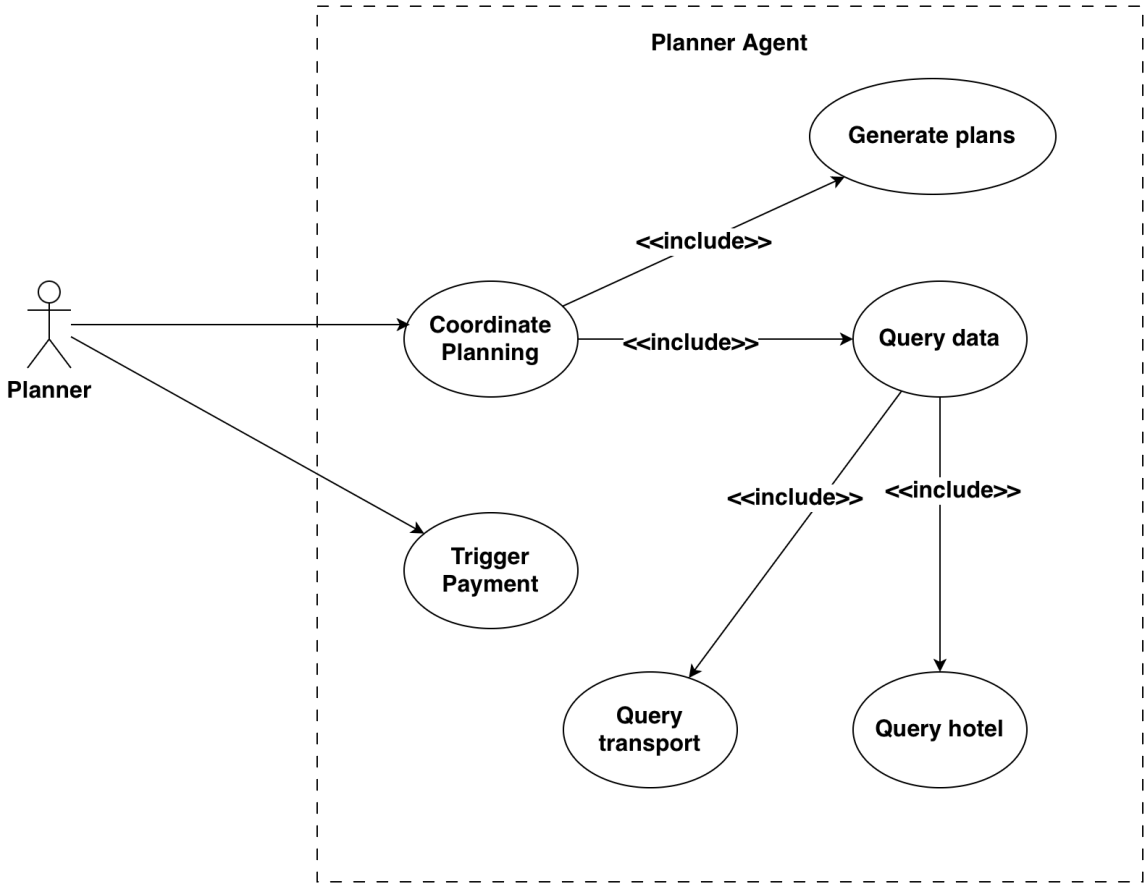
## 3.1 Use Case Diagram Of Agents

The use case diagram illustrates the main interactions between user and multiple agents in the multi-agent system. It defines the my system's functional boundaries and identifies how 5 agents collaborate to fulfill user goals in my system.

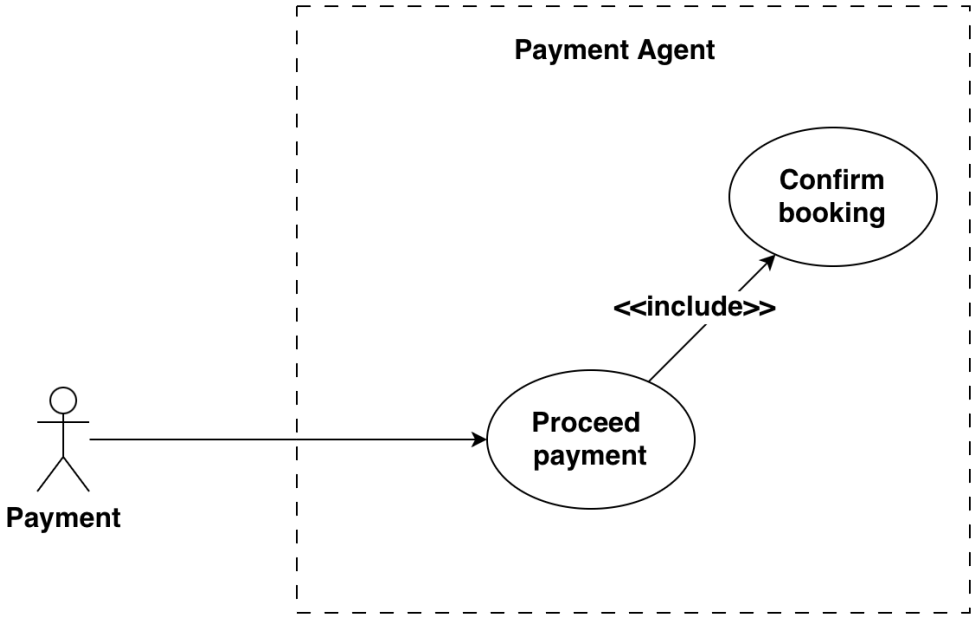
### 3.1.1 User Agent (Requester Agent)



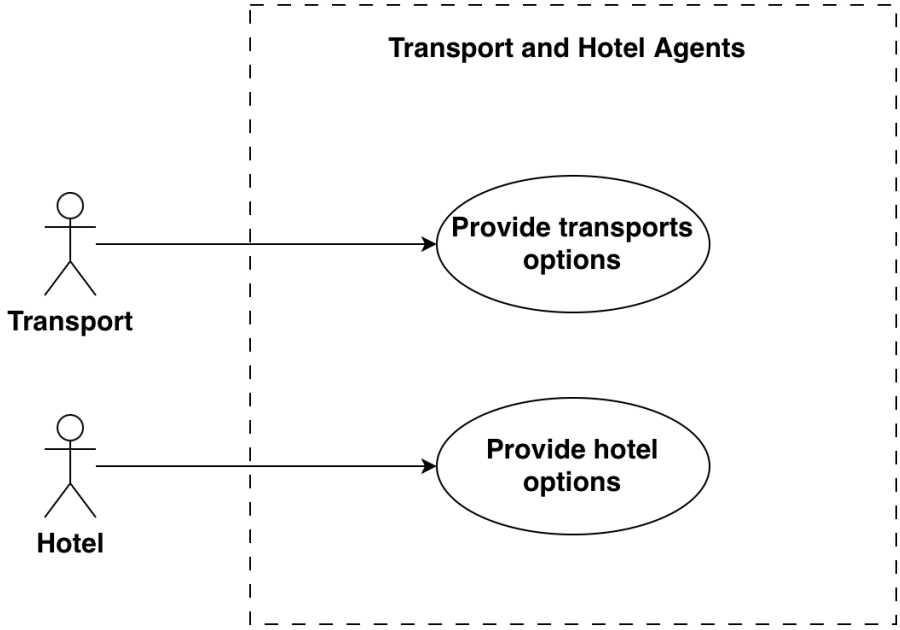

### 3.1.2 Planner Agent



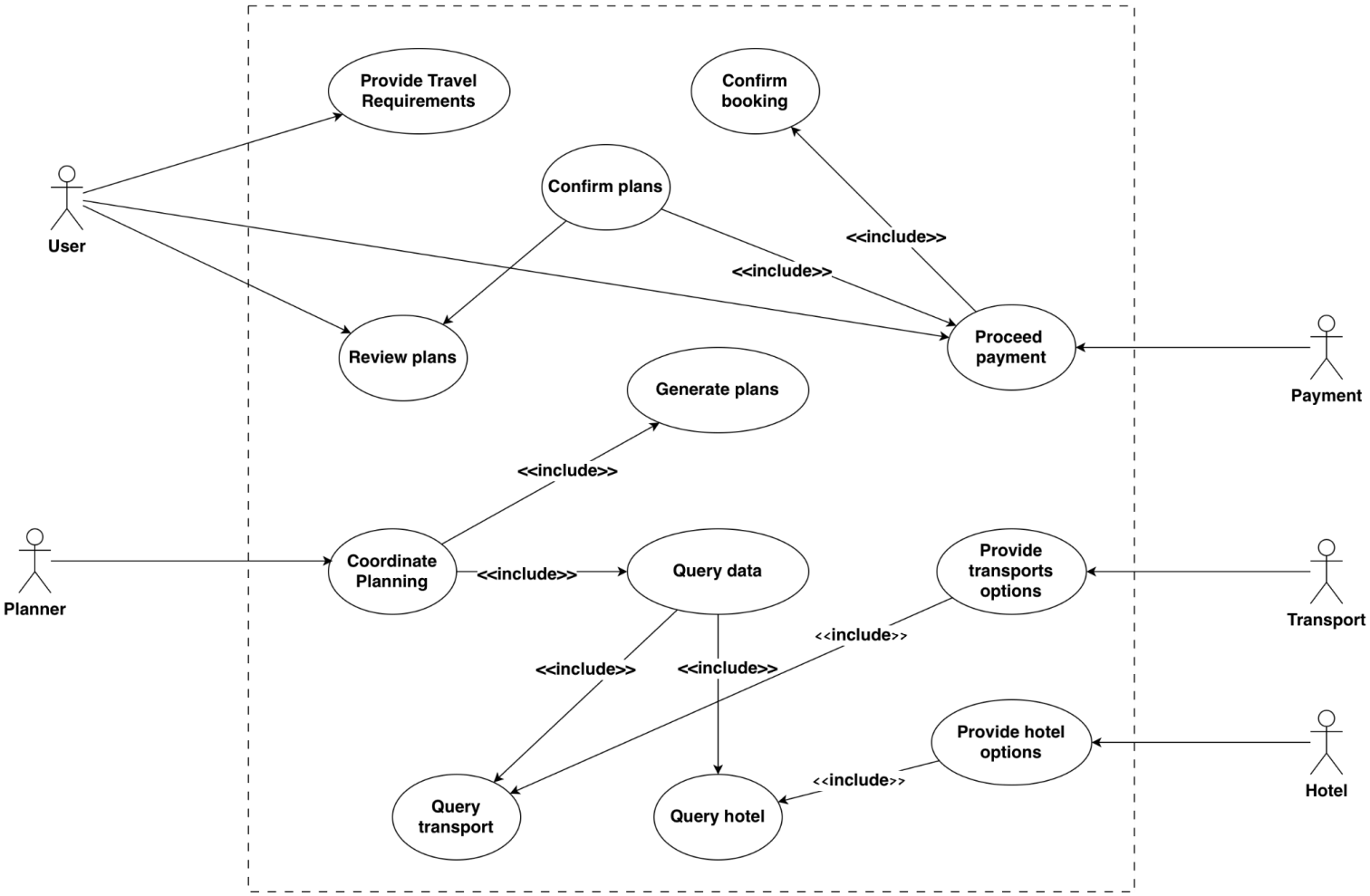

### 3.1.3 Payment Agent




### 3.1.4 Hotel and Transport Agents




### 3.1.5 Use case diagram for whole system



## 3.2 Use Case Definitions for Participating Agents

### Use Case: Provide Travel Requirements

| | |
|---|---|
| Actor | user, User Agent |
| Preconditions | The User Agent is initialized and has a connection to the Planner Agent. The human user has access to an interface (console or GUI) for input |
| Postconditions | User travel preferences (destination, dates, budget) are captured and transmitted to the Planner Agent for processing |
| Main Flow | 1. The user enters travel details (destination, dates, budget) via the User Agent interface<br><br>2. The User Agent validates the inputs<br><br>3. The User Agent sends a request message to the Planner Agent<br><br>4. The Planner Agent acknowledges receipt, initiating the planning process |
| Alternative Flow | If the user provides partial inputs, the User Agent prompts for completion before sending |
| Exceptions | Invalid inputs (past dates) - the User Agent displays an error and requests re-entry |

### Use Case: Coordinate Planning

| | |
|---|---|
| Actor | Planner Agent (primary), Transport Agent, Hotel Agent |
| Preconditions | The Planner Agent has received valid travel requirements from the User Agent. Transport and Hotel Agents are available with predefined datasets |
| Postconditions | Available options are queried, combined, evaluated, and the top 3 plans are generated and proposed to the User Agent |
| Main Flow | 1. The Planner Agent receives and parses the travel requirements<br><br>2. The Planner Agent initiates parallel queries (includes "Query Transport" and "Query Hotel" use cases) by sending query messages to the Transport Agent and Hotel Agent with destination and dates<br><br>3. The Planner Agent receives responses with options lists<br><br>4. The Planner Agent executes the "Generate Plans" use case: combines transport and hotel options and calculates total costs |

| | |
|---|---|
| | 5. The Planner Agent sends top 3 plans to the User Agent |
| Alternative Flow | If fewer than 3 viable plans are found, propose all available ones (or none if zero) |
| Exceptions | Timeout on provider responses (treat as no options and send failure to User Agent)<br>Invalid options - discard and log |

**Use Case: Query Transport**

| | |
|---|---|
| Actor | Planner Agent (initiator), Transport Agent |
| Preconditions | The Transport Agent is running with a simulated transport database (predefined list of bus, train, flight options). A valid query from the Planner Agent is received |
| Postconditions | Matching transport options are returned to the Planner Agent (includes "Provide Transports Options" use case) |
| Main Flow | 1. The Planner Agent sends a query message with destination and dates<br><br>2. The Transport Agent searches its database for options matching the criteria<br><br>3. The Transport Agent invokes "Provide Transports Options"<br><br>4. The Transport Agent sends an message back to the Planner Agent with the list (empty if no matches) |
| Alternative Flow | None |
| Exceptions | Invalid query parameters (unknown destination) - send failure message |

**Use Case: Query Hotel**

| | |
|---|---|
| Actor | Planner Agent (initiator), Hotel Agent |
| Preconditions | The Hotel Agent is running with a simulated hotel database (predefined list of hotel options). A valid query from the Planner Agent is received |
| Postconditions | Matching hotel options are returned to the Planner Agent (includes "Provide Hotel Options" use case) |

| | |
|---|---|
| Main Flow | 1. The Planner Agent sends a query message with destination and dates<br><br>2. The Hotel Agent searches its database for available hotels<br><br>3. The Hotel Agent invokes "Provide Hotel Options"<br><br>4. The Hotel Agent sends amessage back to the Planner Agent with the list (empty if no matches) |
| Alternative Flow | None |
| Exceptions | Invalid query parameters - send failure message |

**Use Case: Provide Transports Options**

| | |
|---|---|
| Actor | Transport Agent |
| Preconditions | A valid transport query has been received and processed (included in "Query Transport") |
| Postconditions | A list of transport options is prepared and sent. |
| Main Flow | 1. Filter database entries by destination and dates<br><br>2. Create TransportOption objects for matches<br><br>3. Package and send |
| Alternative Flow | If no options, send empty list |
| Exceptions | Database access error (simulated) - return empty lisT |

**Use Case: Provide Hotel Options**

| | |
|---|---|
| Actor | Hotel Agent |
| Preconditions | A valid hotel query has been received and processed (included in "Query Hotel") |
| Postconditions | A list of hotel options is prepared and sent |
| Main Flow | 1. Filter database entries by destination and dates |

| | |
|---|---|
| | 2. Create HotelOption objects for matches<br><br>3. Package and send |
| Alternative Flow | If no options, send empty list |
| Exceptions | Database access error (simulated) - return empty list. |

**Use Case: Generate Plans**

| | |
|---|---|
| Actor | Planner Agent |
| Preconditions | Transport and hotel options have been received (included in "Coordinate Planning") |
| Postconditions | Top 3 plans are created as internal data structures |
| Main Flow | 1. Iterate through combinations of transport and hotel options<br><br>2. Calculate total cost for each (transport cost + hotel cost * number of nights)<br><br>3. Filter combinations where total cost ≤ budget<br><br>4. Sort remaining by ascending cost and select top 3 (or all if fewer)<br><br>5. Create TravelPlan objects for proposal |
| Alternative Flow | If no valid combinations, prepare rejection notification |
| Exceptions | Calculation error ( date mismatch for nights) - skip invalid pairs |

**Use Case: Review Plans**

| | |
|---|---|
| Actor | Human User (primary), User Agent |
| Preconditions | The User Agent has received proposed plans from the Planner Agent |
| Postconditions | The user reviews the plans and decides to confirm or reject (extends to "Confirm Plans" if accepted) |
| Main Flow | 1. The User Agent receives the a message and displays the top 3 plans (details, costs) to the user |

| | |
|---|---|
| | 2. User reviews the options<br><br>3. If accepted, extend to "Confirm Plans"<br><br>4. if rejected, send reject message to Planner Agent |
| Alternative Flow | User requests more details on a plan - User Agent displays expanded info if available |
| Exceptions | No plans received - display "No plans found" and end session |

**Use Case: Confirm Plans**

| | |
|---|---|
| Actor | User Agent |
| Preconditions | The user has selected a plan during "Review Plans" |
| Postconditions | The selected plan is confirmed, triggering booking |
| Main Flow | 1. The Human User selects one plan<br><br>2. The User Agent sends accept message to the Planner Agent with the selected plan<br><br>3. The Planner Agent processes the acceptance and invokes "Confirm Booking" |
| Alternative Flow | None |
| Exceptions | User cancels selection – revert to rejection |

**Use Case: Confirm Booking**

| | |
|---|---|
| Actor | Payment Agent |
| Preconditions | A plan has been confirmed by the user. The Payment Agent is available |
| Postconditions | Booking is simulated, payment processed, and confirmation sent to the user |
| Main Flow | 1. The Planner Agent sends a request to the Payment Agent with plan details<br><br>2. The Payment Agent invokes "Proceed Payment" use case<br><br>3. If successful, the Payment Agent sends booking confirmation to the Planner Agent |

| | |
|---|---|
| | 4. The Planner Agent forwards the confirmation to the User Agent for display |
| Alternative Flow | If payment fails (simulated), send failure and notify user to retry or cancel |
| Exceptions | Invalid plan details - Payment Agent sends refuse |

**Use Case: Proceed Payment**

| | |
|---|---|
| Actor | Payment Agent |
| Preconditions | A valid booking trigger is received (included in "Confirm Booking") |
| Postconditions | Payment is simulated and confirmation generated |
| Main Flow | 1. Validate plan details<br>2. Simulate payment process<br>3. Generate Booking Confirmation object<br>3. Send to Planner Agent |
| Alternative Flow | None |
| Exceptions | Payment failure - return failure |

## 3.3 Class Diagram

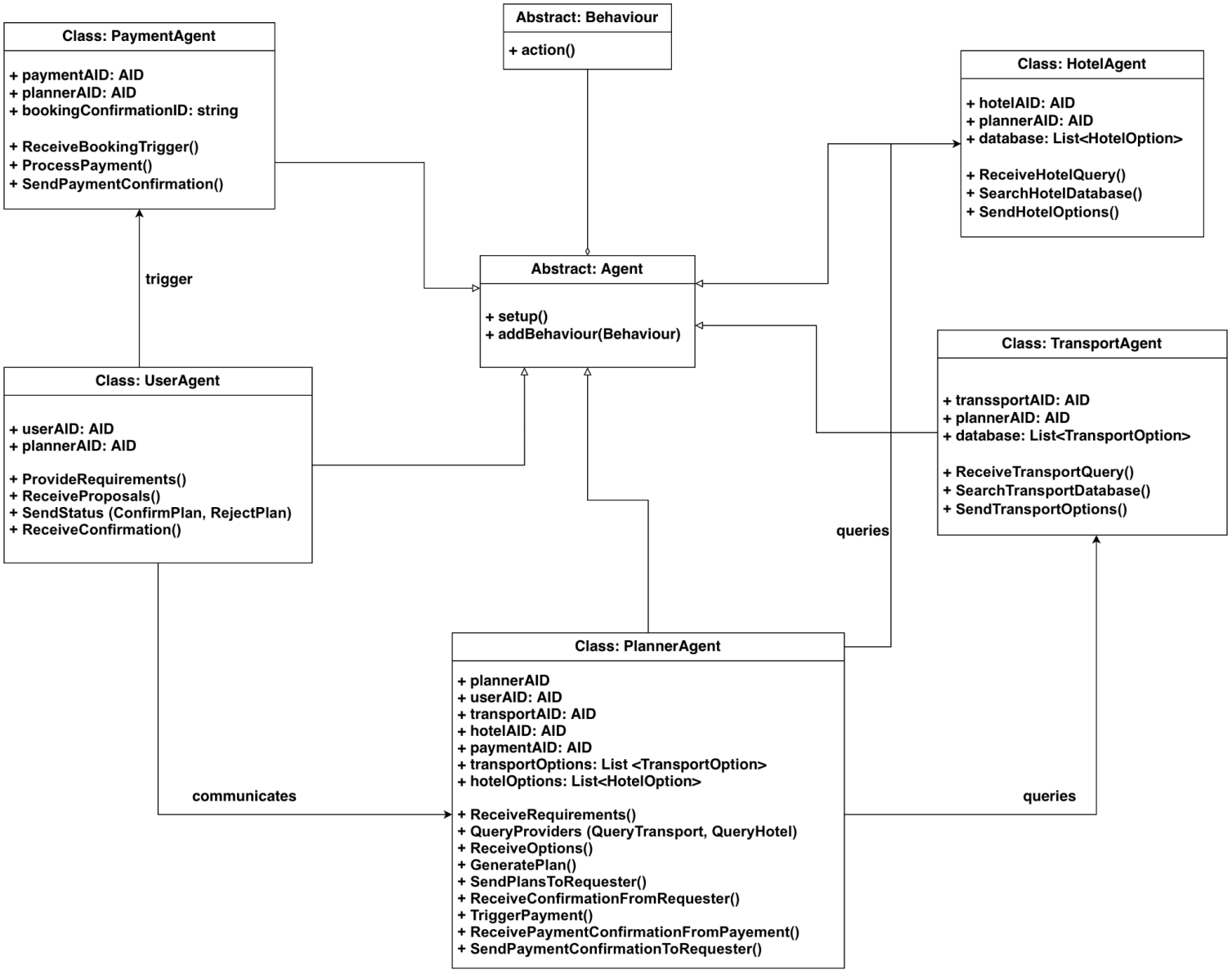
The class diagram gives a clear picture of how the my system is built using JADE. It shows the main agents like User Agent, Planner Agent, Transport Agent, Hotel Agent, and Payment Agent, each based on the roles from the GAIA design.

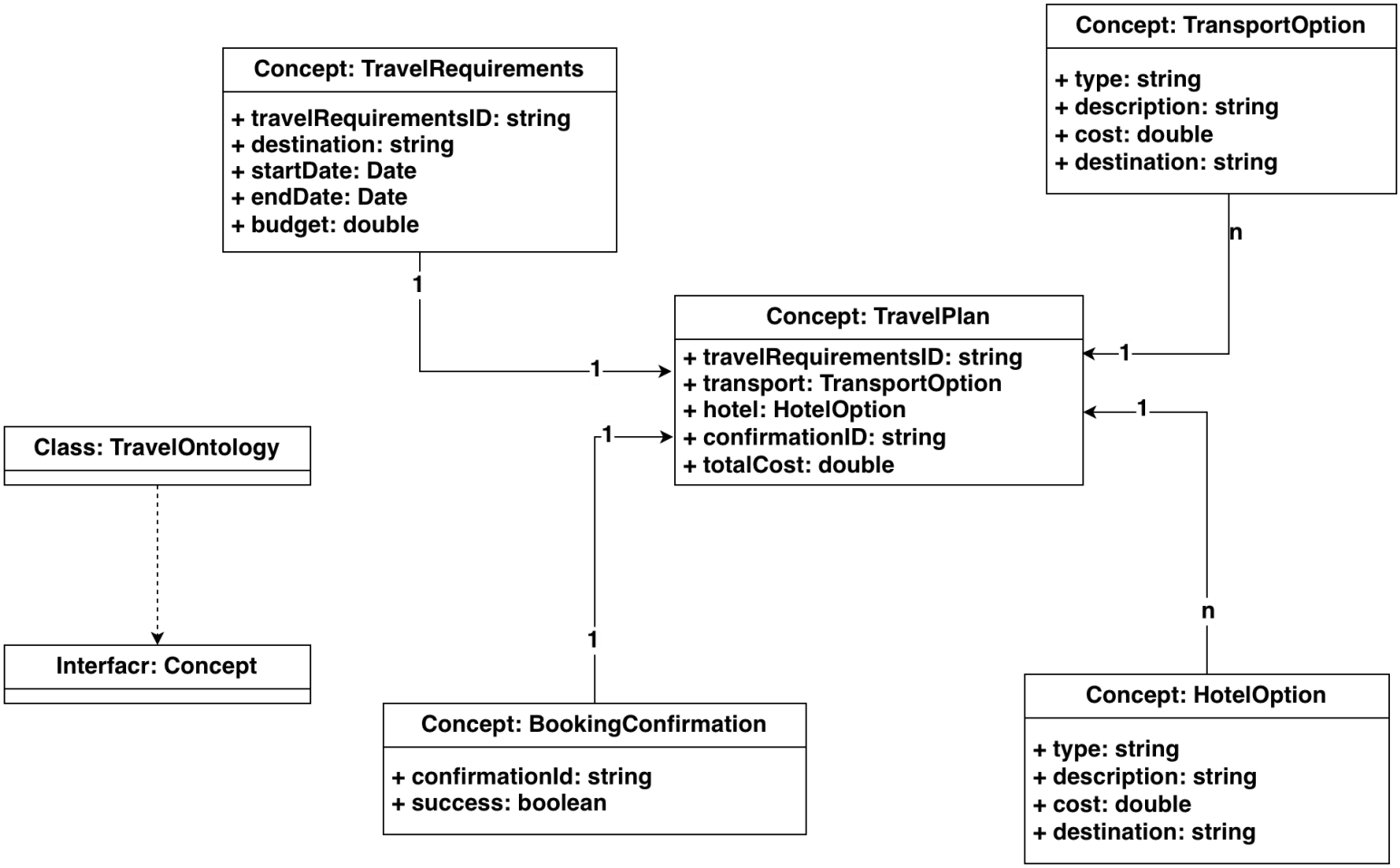
These agents inherit from **jade.core.Agent** and they use behaviors like ProvideRequirements or HandleQuery to handle tasks such as sending messages or processing data. Data classes such as TravelRequirements and TransportOption are set up as simple concepts for messages, with attributes like destination, cost, and dates. This setup keeps things modular, making it easy to update or fix parts of the code later.

Overall, the diagram connects everything with lines showing relationships, like how the Planner Agent talks to the others for queries and bookings. Simulated databases are just lists in the Transport and Hotel Agents to hold fake data for testing.

It ties back to the system's goals by ensuring agents work together smoothly, supporting reliability and easy maintenance. This acts as a guide for writing the actual Java code helping turn the design into a working multi-agent system.

My class diagram includes 2 parts.



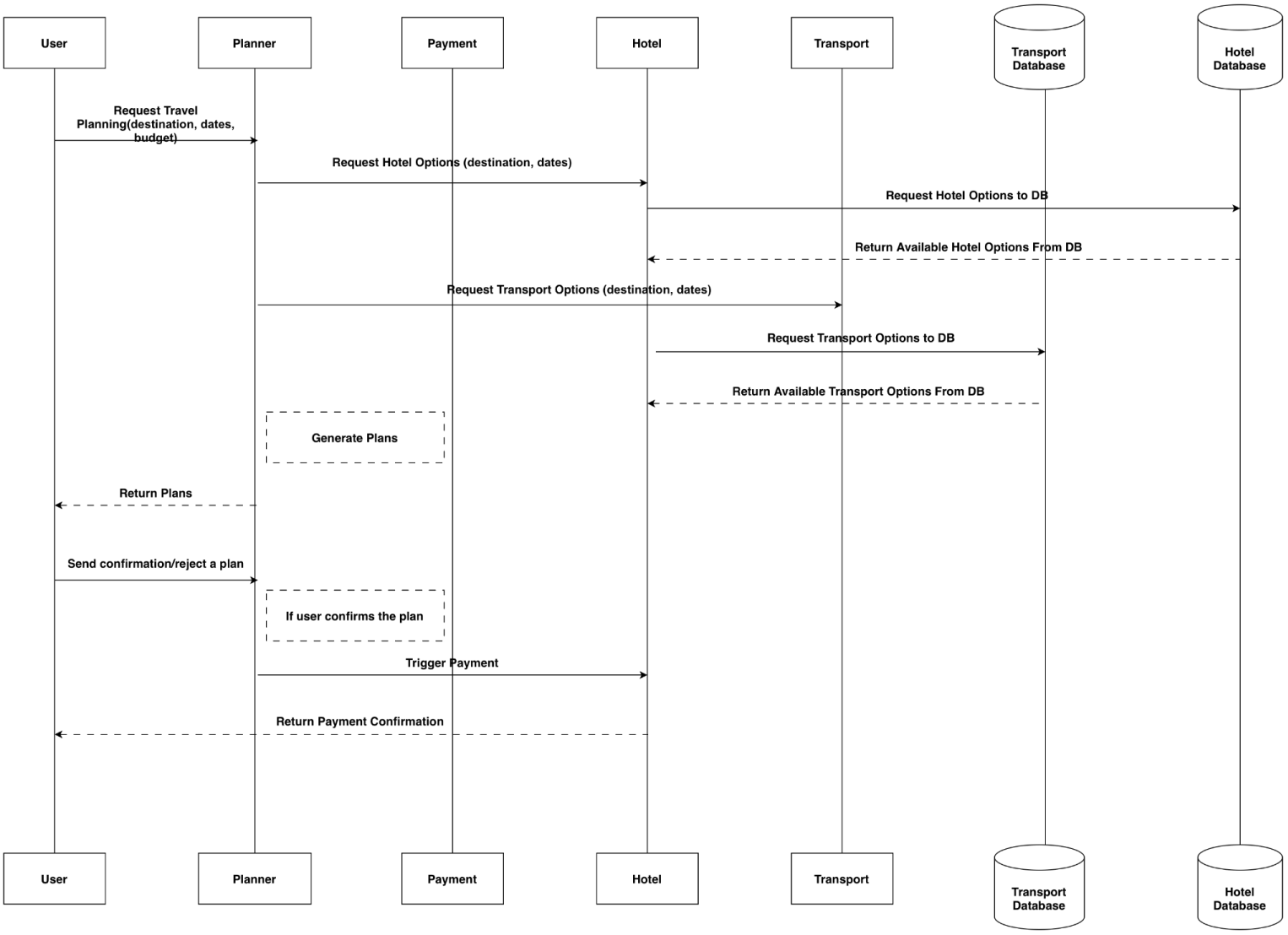



## 3.4 Message Sequence Chart

### 3.4.1 Agent Communication

The inter-agent communication is based on the FIPA standards using the ACL (agent communication language). In FIPA, we have the privilege to transport and encode the inter-agent messages among various remote platforms

### 3.4.2 Interaction Chart

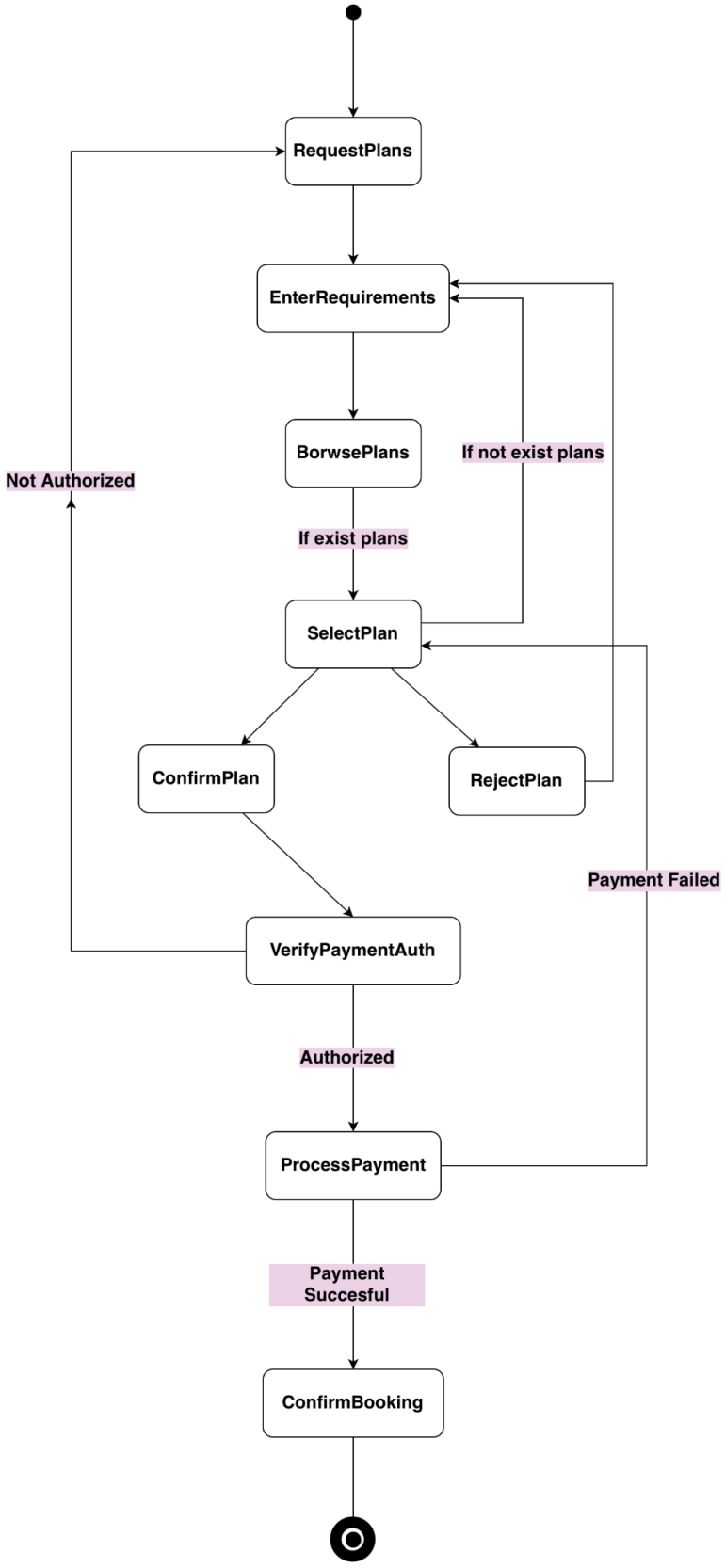



The User Agent initiates the communication by requesting a travel plan, providing input parameters such as destination, travel dates, and budget. Upon receiving this request, the Planner Agent sends separate queries to the Transport Agent and the Hotel Agent to obtain data from the Transportation and Hotel databases.

The Transport Agent and Hotel Agent each retrieve the relevant data from their respective databases and return the results to the Planner Agent. After receiving both responses, the Planner Agent processes and evaluates the data to generate the top three optimized travel plans based on the user's preferences and constraints. These plans are then sent to the User Agent for review.

Once the User Agent confirms the selected plan, the Planner Agent triggers a signal to the Payment Agent to initiate the payment process. After the transaction is successfully completed, the Payment Agent sends a confirmation message to the Planner Agent, which in turn forwards the confirmation to the User Agent to finalize the process.

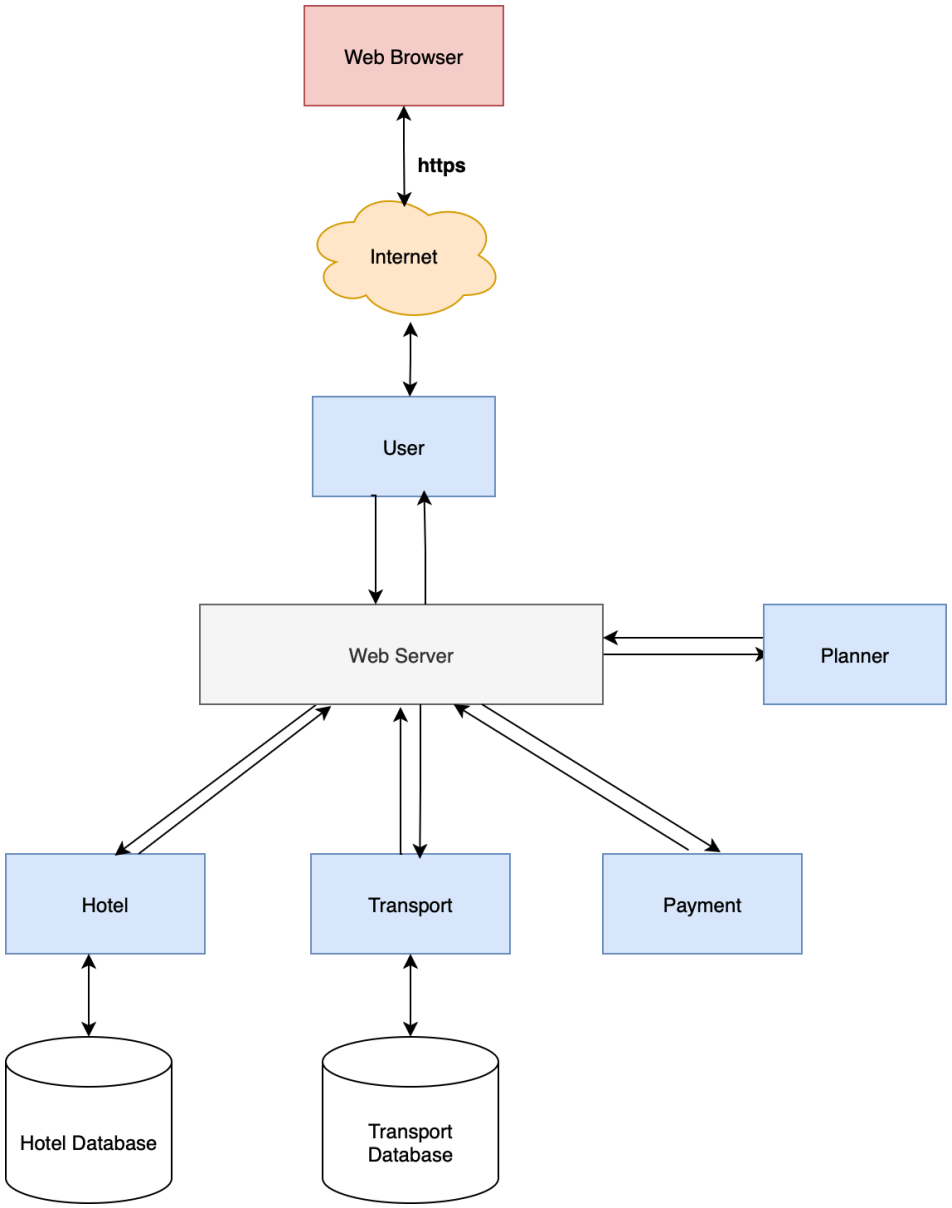
## 3.5 Activity Diagram



## 3.6 Overall System (Already mentioned in first report)

The MAS communicates with the web browser over the internet using the HTTPS protocol. The web browser allows users to enter travel details, which are sent to the web server.

The web server forwards these requests to the JADE platform, where agents such as the User, Planner, Transport, Hotel, and Payment Agents coordinate to process data and generate optimized travel plans. The final results are returned through the web server to the web browser for user display.



# 3.7 Data specification

## 3.7.1 E-R Diagram