Helwan University – Faculty of Computers & Artificial Intelligence

Module: CS251 Software Engineering 1 – Spring “Semester 2” 2023-2024

Homestay & Cultural Exchange System

Name:

- **Fatima Amr**
  20220332
- **Nada Ashraf**
  20220532
- **Mariam Abdelnabi**
  20220466
- **Sarah Ahmed**
  20220194
- **Miar Wafa**
  20220520
- **Shahd Bahaa**
  20220235

PART 1: Overview & Software Requirements Specification

I. Introduction:

a) Purpose.

Homestay programs are programs that provide tourists with the chance to stay with a local family or individual when visiting a foreign nation or even inside their own nation. Through this immersion experience, travelers can fully acclimate to the habits, traditions, and way of life of the host nation. Staying with a host family enables students to gain a distinctive perspective on the local community, language, and cuisine, which enhances their understanding of and appreciation for the host country's culture. Depending on the traveler's interests and the host's availability, homestays can range in length from a few days to many months or even years.

Homestay programs are popular among travelers for several reasons:

Cultural Immersion: Homestays offer a deeper cultural immersion compared to traditional accommodation options like hotels or hostels. Staying with a local host family provides travelers with firsthand exposure to local customs, traditions, language, and cuisine.

Authentic Experience: Travelers get the chance to experience daily life in the destination country from a local perspective. They can participate in family activities, attend local events, and explore off-the-beaten-path attractions that might not be accessible to tourists.

Language Practice: For travelers interested in language learning, homestays provide an excellent opportunity to practice speaking the local language in a natural setting. Living with native speakers can significantly enhance language skills and fluency.

Personal Connections: Homestays foster personal connections between travelers and their hosts, often leading to long-lasting friendships and cultural exchange. Hosts may offer valuable insights, guidance, and support to help travelers navigate their new environment.

Affordability: In many cases, homestay accommodation is more affordable than hotels or other lodging options, making it an attractive choice for budget-conscious travelers. Additionally, some hosts may provide meals or other amenities as part of the accommodation package, further enhancing the value for travelers.

Homestay programs are facilitated by various websites and platforms that connect travelers with host families or individuals. These platforms typically allow travelers to browse available homestay listings, view host profiles and reviews, communicate with potential hosts, and make bookings securely, which is what we aim to do in this project.

Before booking a homestay, travelers should carefully review the host's profile, read guest reviews, and communicate with the host to ensure that the accommodation meets their preferences and expectations. Additionally, travelers should familiarize themselves with any house rules, cancellation policies, and payment terms specified by the homestay platform. Overall, homestay programs offer travelers a unique opportunity to connect with local communities, gain cultural insights, and create memorable travel experiences.

b) Project Scope

Our Project aims to provide travelers with a platform to connect with local hosts for short-term accommodation experiences. The prototype will focus on creating a basic system that allows users to browse homestay listings, view host profiles, and make bookings securely.

Objectives and Deliverables:

- ➢ Develop a user-friendly web-based platform for browsing and booking homestay accommodations.
- ➢ Implement a registration system for users and hosts to create profiles and manage bookings.
- ➢ Design a database schema to store user and host information, homestay listings, and booking details.

- Create a booking functionality that allows users to make secure reservations for homestay accommodations.
- Provide administrative capabilities for managing user accounts, host profiles, and booking transactions.
- Allow users to express their experiences during their journeys through articles accompanying them with photos taken during these journeys.
- Rating and reviewing hosts and the homestays they tried before giving the other users an overview about the homestay/host.

Inclusions:

- User registration and login functionality.
- Host profile creation and listing management.
- Homestay listing browsing and search functionality.
- Booking and reservation management.

Exclusions:

- Advanced features such as reviews and ratings.
- Integration with external payment gateways beyond the prototype stage.

Dependencies:

- Availability of development team members and resources.
- Integration with third-party services or APIs for certain functionalities.

c) Glossary and Abbreviations (for any technical or non-technical terms.)

1. Homestays: Accommodation provided by local hosts in exchange for volunteer work or cultural exchange.
2. Cultural Exchange: Interaction between individuals from different cultural backgrounds to share customs, traditions, and experiences.
3. Volunteers/Travelers (VT): Individuals seeking homestays and cultural exchange opportunities.

4. Hosts: Individuals providing accommodation and requesting assistance from volunteers/travelers.
5. Lodging and Food Exchange (LF Exchange): The arrangement where volunteers contribute time in exchange for accommodation and meals provided by hosts.
6. Language Learners: Individuals looking to improve foreign language skills through immersion.
7. Specialist/Niche Help (SH): Specific skills or assistance requested by hosts, such as gardening, animal care, cooking, etc.
8. Membership Fee: Fee charged to travelers for access to the platform's services.
9. Duration of Exchange: The length of time that volunteers stay with hosts, ranging from days to over a year.
10. Cultural Immersion: Engaging deeply in the local culture, customs, and way of life.
11. Regional Recommendations: Suggestions provided by hosts about places of interest and activities in their area.
12. Volunteer Help: Assistance provided by volunteers to hosts in various tasks and projects.
13. Homestay Registration System: The system used by the platform for hosts to register their homestay opportunities.
14. Local Customs: Traditions and practices specific to a particular region or community.
15. Cultural Orientation: Guidance provided to volunteers/travelers to help them adapt to the customs and norms of the host country.
16. Traveler Requirements: Criteria or expectations set by travelers regarding their accommodation and exchange experience.
17. Profile Highlights: Notable aspects or skills highlighted in a volunteer's online profile.
18. Cultural Tolerance: Acceptance and respect for cultural differences and diversity.
19. Community Connections: Building relationships and networks within local communities through the exchange program.
20. Farming Information Request: Specific details or requirements related to farming tasks provided by hosts.
21. Animal Care Details: Information about caring for animals provided by hosts.
22. Gardening Care Preferences: Preferences and instructions related to gardening tasks given by hosts.

23. Cultural Competency Training: Education and training programs designed to enhance individuals' ability to work and interact effectively in multicultural environments.
24. Interpersonal Skills: The ability to effectively interact and communicate with others, including active listening, empathy, and conflict resolution.
25. Cultural Sensitization: The process of raising awareness and educating individuals about cultural differences and similarities.
26. Cultural Enrichment: The enhancement of one's understanding and appreciation of different cultures through exposure and experience.
27. Cultural Diversity: The existence of a variety of cultural groups within a society or community.
28. Language Barrier: Difficulty in communication due to differences in language spoken by volunteers and hosts.
29. Cross-Cultural Communication: Communication between individuals from different cultural backgrounds.
30. Cultural Exchange Activities: Planned events or experiences designed to facilitate cultural exchange between volunteers and hosts.
31. Homestay Etiquette: Expected behaviors and manners when staying with a host family.
32. Cultural Adaptation: The process of adjusting to and becoming comfortable with a new cultural environment.
33. HSCE: Homestays and Cultural Exchange
34. WW: Worldwide
35. CT: Cultural tolerance
36. ET: Exchange duration types (short-term, long-term, etc.)

d) List of the System Stakeholders.

- **Users:**

  Travelers: Students or classmates who will use the homestay program to find short-term accommodation.

  Hosts: Other students or classmates offering their homes for accommodation.

- **Development Team:** Classmates working together to create the homestay program prototype. Roles may include coding, designing, managing the project, and testing the software.
- **Admin:** Supervising the project and providing guidance and feedback to the student development team.

e) References.

- Homestay.com. (n.d.). Homestay accommodation worldwide for short and long term stays. Retrieved from https://www.homestay.com/
- Booking.com. (n.d.). Booking.com: The largest selection of hotels, homes, and vacation rentals. Retrieved from https://www.booking.com/
- Project Management Institute. (2017). A guide to the project management body of knowledge (PMBOK guide) (6th ed.). Project Management Institute.
- Smith, J. (2020). The rise of homestay accommodations in the tourism industry. Hospitality Management Quarterly, 15(3), 207-220.
- Johnson, R. (2019). Exploring the cultural exchange benefits of homestay experiences. Journal of Tourism Research, 25(2), 123-135.
- Brown, A. (2018). The impact of online platforms on the homestay accommodation sector. Journal of Hospitality and Tourism Technology, 12(4), 321-335. doi:10.1108/JHTT-03-2018-0034

II. Functional Requirements:

a) User Requirements Specification.

User Registration:

Users should be able to register as either hosts or travelers.

Registration should require basic personal information such as name, email, and password.

User Profiles:

Hosts should be able to create a profile detailing information about themselves, the type of help they require, the accommodation they offer, and the type of person they expect.

Travelers should be able to create a profile including personal details and any specific skills they possess.

Search and Contact:

Travelers should be able to search for hosts based on location, type of help required, and availability.

Travelers should be able to contact hosts through the platform to discuss potential exchanges.

Exchange Management:

Users should be able to manage their exchanges, including confirming arrangements, updating availability, and leaving feedback after the exchange.

Membership and Payment:

The platform should offer a membership option for travelers, allowing them to connect with hosts.

Travelers may be required to pay a yearly membership fee to access the platform.

b) System Requirements Specification:

Platform Infrastructure:

The system should be hosted on a reliable and scalable web hosting service to ensure availability and performance.

The platform should be developed using web technologies such as HTML, CSS, JavaScript, PHP, and a database system like MySQL.

User Authentication and Security:

User registration and authentication should be implemented securely to protect user data.

The system should use encryption for sensitive data transmission, such as passwords.

Database Management:

The system should utilize a database management system (e.g., MySQL) to store user profiles, exchange details, and other relevant data.

The database schema should be designed to efficiently store and retrieve user and exchange information.

Search and Matching Algorithm:

The system should implement a search and matching algorithm to match travelers with suitable hosts based on location, availability, and preferences.

The algorithm should prioritize matches based on compatibility and user preferences.

User Interface and Experience:

The platform should have an intuitive and user-friendly interface for both hosts and travelers to navigate and interact with.

The user interface should be responsive and accessible across different devices and screen sizes.

c) Requirements' Priorities:

Using the MoSCoW Scheme:

Must Have:

User registration and authentication

User profiles for hosts and travelers

Search and contact functionality.

Exchange management features

Secure payment processing for traveler membership

Should Have:

Advanced search and matching algorithm

Feedback and rating system for users

Responsive and user-friendly interface

Could Have:

Integration with social media platforms for user authentication and sharing.

Additional features for organizing group exchanges or events.

Won't Have (this release):

Mobile application development (unless deemed necessary in future iterations)

Integration with external services (e.g., travel booking platforms)

III. Non-Functional Requirements:

a) The General Types/Categories of Non-Functional Requirements (that you will follow. you may select from the categories presented in the lecture.)

- Performance: Concerned with system responsiveness, throughput, and resource utilization.
- Security: Focuses on protecting data, ensuring confidentiality, integrity, and availability.
- Usability: Relates to the ease of use and user experience of the system.
- Reliability: Deals with the system's ability to perform consistently and reliably over time.
- Scalability: Addresses the system's capability to handle increasing workload or user base.

- Availability: Ensures the system is accessible and operational when needed.
- Maintainability: Refers to the ease with which the system can be maintained, updated, and extended.
- Compatibility: Ensures the system functions correctly with different devices, browsers, and operating systems.

b) Non-Functional Requirements Specification (including the category/type of each.)

Performance:

The system should respond to user interactions within 2 seconds under normal load conditions.

Database queries should execute within 500 milliseconds on average.

The platform should support concurrent access by at least 1000 users without significant performance degradation.

Security:

User passwords should be stored securely using industry-standard encryption algorithms (e.g., bcrypt).

All user inputs should be validated and sanitized to prevent SQL injection and cross-site scripting (XSS) attacks.

Access to sensitive user data should be restricted based on user roles and permissions.

Usability:

The user interface should be intuitive and easy to navigate, with clear labeling and minimal clutter.

Error messages should be informative and guide users on how to resolve issues.

The platform should provide assistance and guidance for first-time users through tooltips or walkthroughs.

Reliability:

The system should have a uptime of at least 99.9% excluding scheduled maintenance.

Data backups should be performed regularly to prevent data loss in case of system failure.

Scalability:

The system architecture should be designed to scale horizontally to accommodate increased user demand.

Load balancing mechanisms should be in place to distribute traffic evenly across multiple servers.

Availability:

The platform should be available 24/7, with scheduled maintenance windows communicated to users in advance.

Redundant systems and failover mechanisms should be implemented to minimize downtime.

Maintainability:

The codebase should be well-documented and follow coding standards to facilitate ease of maintenance.

Modular design principles should be followed to allow for easy updates and modifications without impacting other components.

Compatibility:

The platform should be compatible with the latest versions of popular web browsers (Chrome, Firefox, Safari, Edge).

Responsive design techniques should be employed to ensure proper display on various screen sizes and devices.

c) The fit criteria for every Non-Functional Requirement (Testable Non-Functional Requirements.)

- Performance: Measure system response times using tools like Apache JMeter or Google Chrome DevTools.
- Security: Conduct penetration testing and code reviews to identify and address security vulnerabilities.
- Usability: Perform usability testing with representative users to evaluate ease of use and navigation.
- Reliability: Monitor system uptime and track incidents to ensure reliability targets are met.
- Scalability: Conduct load testing to assess system performance under increasing user loads.
- Availability: Track system downtime and measure uptime against availability targets.
- Maintainability: Evaluate code maintainability metrics such as code complexity and test coverage.
- Compatibility: Test the platform on different browsers, devices, and operating systems to ensure compatibility.

d) How would the above-mentioned Non-Functional Requirements affect the System's overall Architecture?

Non-functional requirements influence architectural decisions such as the choice of technology stack, database design, and deployment strategy.

For example, scalability requirements may lead to the adoption of a microservices architecture to enable independent scaling of different components.

Security requirements may dictate the implementation of encryption protocols, access control mechanisms, and regular security audits within the architecture.

Availability requirements may necessitate the use of redundant servers, failover mechanisms, and distributed caching to ensure continuous operation.

Overall, the architecture should be designed and optimized to meet the non-functional requirements while providing a robust and reliable platform for users.

e) Requirements' Priorities:

Using the MoSCoW Scheme:

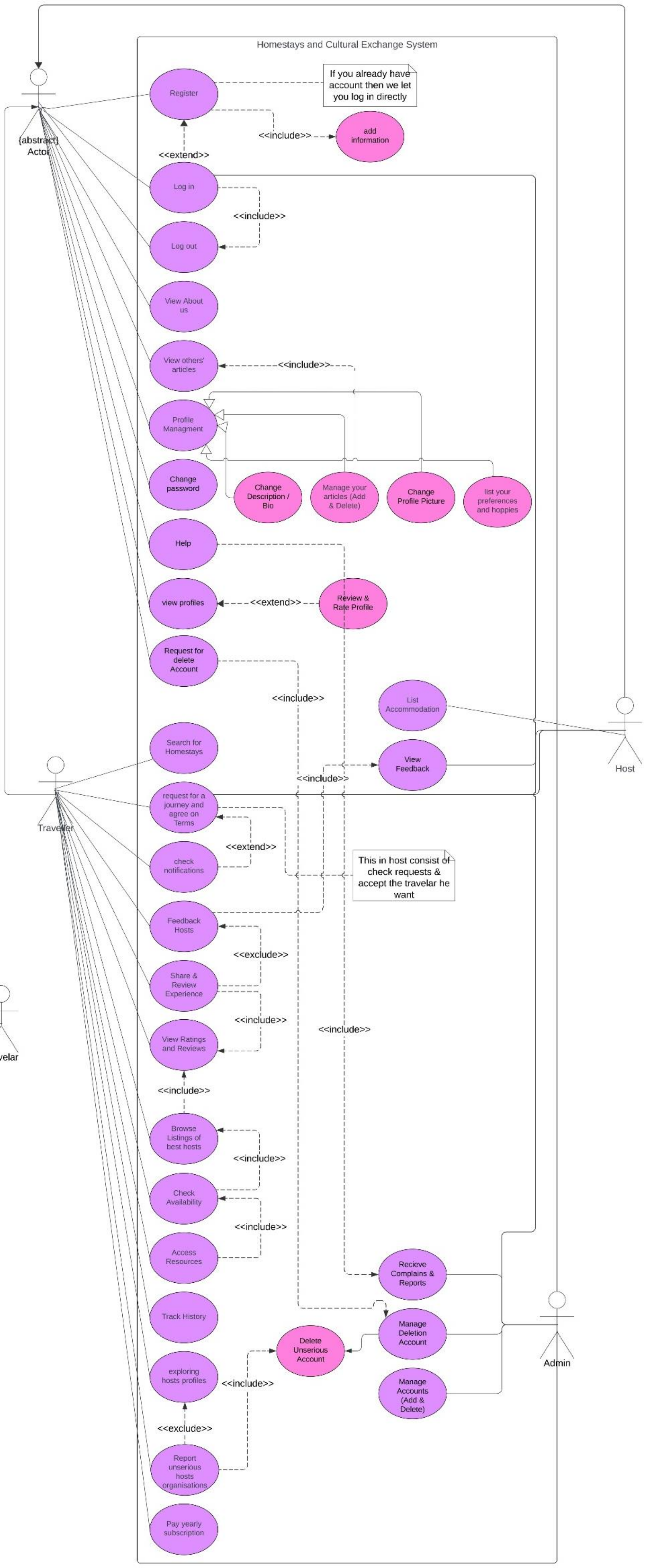
Must Have:

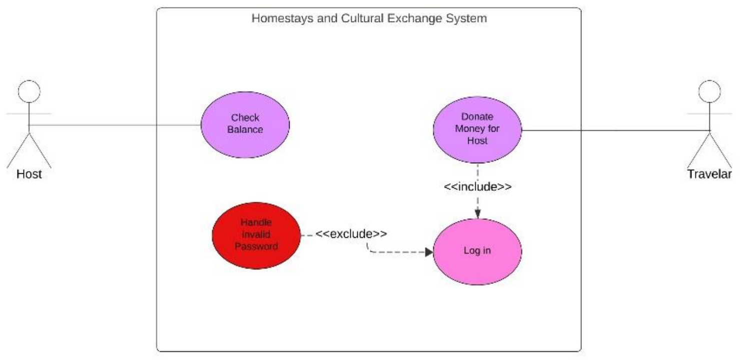
- Performance
- Security
- Usability
- Reliability
- Scalability
- Availability
- Should Have:
- Maintainability
- Compatibility

PART 2: System Design & Models

VIII. Functional Diagrams:

a) Use-Case Diagram(s) including all the Use Cases for the system





b) Detailed Use-Cases Description

id	Name	Actor	Goals	Preconditions	Postconditions	Main Success Scenario	Alternate Scenario	Exceptions
1	Register	Host traveler	allow users to create an account	1. The user must navigate to registration page 2. the user must not already have an account	1.the user will have unique account. 2.the user will be able to log in to the system. 3.a confirmation email may be sent to the user	1.the user navigate to registration page. 2.the user filled out the required registration form. 3.the system validates the information	-	If the information is invalid the system prompts the user to handle it
2	Log in	Host traveler	Allow registered users to access their account	1.the user must have a registered account. 2.the user must navigate to log in page	the user will be granted access to their account	1.the user enters their registered email and password. 2.the system validates the credentials against the information stored in database. 3.the system redirects the user to their account homepage	If the user wants to log in using a social media account, the system will need to redirect him to social media platform	1.if the user enters incorrect credentials the system displays an error message 2.after multiple failed logs in attempts the system should lock the user account
3	Search for homestays	traveler	Search for homestays accommodations based on preferences	1.the user must have access to “search for homestays” feature. 2.the user must have specified search criteria	1.the user finds homestays accommodations that match their search criteria. 2.the user can view details of the homestay listing. 3.the user can save or book a homestay for their desired dates	1.the user enters their search criteria. 2.the system retrieves a list of accommodations that match user search criteria. 3.the user selects a specific homestay listing to view more details	if the user chooses to filter homestay options based on specific criteria such as location, price range, amenities, or host preferences. The system would allow the user to input these filters to narrow down their search results	1.If the user's search criteria are too specific or there are no homestays available that match their filters, the system should display a message indicating that no results were found 2.If there are only a limited number of homestays available that match the user's search criteria, the system may display a message indicating the limited availability and suggest considering

								alternative dates or locations for their stay.
4	5C6h7e8ck availability	traveler	1.Allow the user to check the availability of the host 2.provide travelers information on the host availability	1.traveler must have selected a specific host 2.host must have provided their availability status on the platform	1.the system displays the availability status of the selected host. 2.traveler can request or book if the host is available 3.hosr receives a notification of the traveler request	1.traveler navigates to the page for selecting a host 2.the system retrieves the availability status of the host then displays it to the traveler. 3. if the host is available the traveler can request for booking 4.host receives a notification of booking and can accept it or decline it	if the user wants to check the availability of a specific homestay for their desired dates. The system would allow the user to input their check-in and check-out dates, as well as the number of guests, to see if the homestay is available during that time.	1. If the homestay is fully booked for the user's desired dates, the system should display a message indicating that there is no availability 2. If the user inputs invalid or incorrect dates, such as a past date or an invalid date format, the system should display an error message prompting the user to correct the input
5	Request for a journey	traveler	Request a journey to a specific destination	1.The traveler must have access to the "Request for a Journey" feature. 2.The traveler must have specified their current location and destination in their account settings.	1.The journey request is submitted successfully. 2.The traveler receives confirmation of the journey request	1.The traveler enters their desired destination 2.The traveler confirms the journey request. 3.The traveler reviews the available options and The system confirms the journey request	if the user wants to request a customized journey experience that includes multiple homestays, cultural activities, and sightseeing tours. The user would be able to input their preferences The system would then match the user's preferences with available homestays, activities, and tours to create a customized journey package.	1.If the user requests a journey to a destination where there are no available homestays or cultural activities listed in the system, the system should display a message indicating that the destination is not currently supported 2.If the user's requested preferences for the journey are too specific or complex to be accommodated by the system, the system should display a message indicating that it may not be possible to create a customized

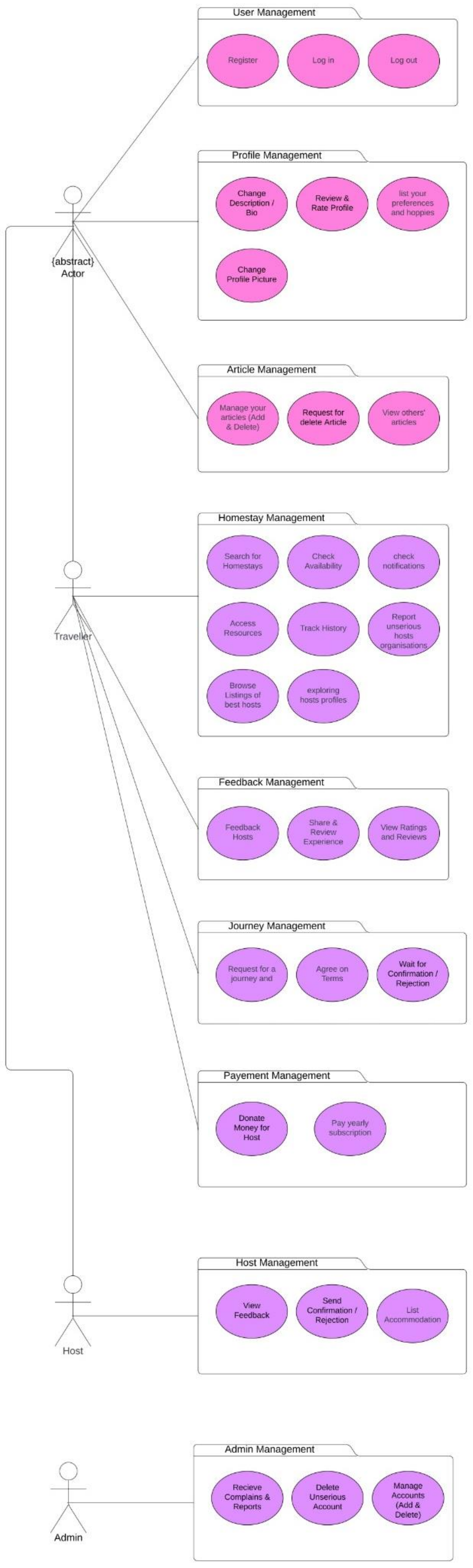
								6journey matching all of the user's preferences
6	Check notification	traveler	To enable users to view notifications related to their homestay or cultural exchange activities.	There should be notifications available for the user to check.	1.User has viewed their notifications. 2.System updates notification status	1.User navigates to the notifications section. 2.User sees a list of notifications. 3.User marks notifications as read or takes relevant actions.	Users receive additional options to manage or respond to notifications.	1.System experiences downtime, making notifications inaccessible. 2.Notifications are incorrectly displayed or missing.
7	Access resources	traveler	Enable hosts to share resources and information with travelers once availability is confirmed.	1.Traveler has requested a booking with a host and the host has confirmed availability. 2. Traveler has agreed to receive and access resources from the host.	1.Traveler can access and view the resources shared by the host 2 Communication between the host and traveler is facilitated through the exchange of information.	1.Host prepares relevant resources to share with travelers 2.Host initiates the "Access Resources" function in the system and selects the traveler to share the resources with. 3.Traveler views and accesses the resources shared by the host	1.User's access level may restrict certain resources. 2.They may be prompted to contact an administrator for assistance	If a resource is temporarily unavailable or has been removed, the user should receive a notification or message explaining the situation
8	Request for communica-tion	traveler	Enable travelers to initiate communication with hosts to ask questions, seek clarification, or discuss details related to their upcoming stay.	1.Traveler has received access to resources shared by the host for their upcoming stay. 2.Traveler has a specific question, request, or need for further information that they want to communicate to the host.	1.Traveler successfully sends a communication request to the host through the system 2.Host receives the communication request and can respond accordingly.	1.Traveler initiates the "Request for Communication" function in the system and selects the host as the recipient of the communication request. 2.Host responds to the communication request, providing answers, clarification, or additional information as needed. 3.Communication between the host and traveler is established	User specifies preferred communication method If the preferred method is not available, the system prompts the user to choose an alternative method.	if the user being contacted does not respond within a reasonable timeframe, the system may send reminders
9	Feedback hosts	traveler	To enable users to provide feedback on their experience with hosts	1.User must have completed a stay with a host. 2.User must be logged into their account	1.Feedback is submitted and recorded in the system. 2.Host's rating and reputation may be updated based on the feedback provided.	1.User navigates to the feedback section 2.User selects the host they wish to provide feedback for.	1.User provides positive ratings and comments about their experience with the host. 2.User provides negative ratings	-

						03.User fills out th10e feedback form, including ratings and comments.	and comments, indicating areas where the host could improve.	
10	Share and view experience	traveler	Enable users to share their experiences and allow others to view these experiences.	User must have participated in a homestay or cultural exchange activity.	1.Experience is shared and visible to other users. 2. Users can view shared experiences for inspiration or information.	1.User writes a description of their experience, optionally including photos or videos. 2.Experience is published and becomes visible to other users. 3.Another user views the shared experience for inspiration or information.	1.User includes photos or videos to enhance their shared experience. 2.Other users are inspired by the shared experience to participate in similar activities	User shares an experience containing inappropriate
11	Browse listing of best hosts	traveler	Allow users to browse a curated list of the best hosts in the homestay and cultural exchange program.	The system must have a curated list of best hosts based on criteria such as ratings, reviews, and host performance.	1.User can view the listing of best hosts. 2. User may choose to contact or book a stay with one of the listed hosts.	1.User views the curated list of best hosts, including details such as host name, location, ratings, and reviews. 2.User selects a host from the list to view more information. 3. User may contact or book a stay with the selected host directly from the listing.	1. System provides personalized recommendations based on user preferences and past interactions with hosts. 2.System highlights a subset of hosts as "featured" based on promotional campaigns or partnerships.	1.If there are not enough hosts meeting the criteria for inclusion in the "best hosts" list, the system may display a message indicating
12	Track history	traveler	Allow users to track their history of homestays and cultural exchange activities within the system.	User must have participated in at least one homestay or cultural exchange activity.	1.User can view their history of past homestays and cultural exchange activities. 2. System records and updates the user's history as new activities are completed.	1.User views a chronological list of their past homestays and cultural exchange activities 2.User may choose to view additional details or provide feedback for specific activities.	1.User has the option to filter their history based on criteria 2.User can export their history data for record-keeping or sharing purposes.	If there are gaps or missing entries in the user's history, the system should prompt the user to verify or provide additional information to complete their history.
13	Exploring hosts profiles	traveler	provide users (guests) with detailed information about potential hosts to help them make informed decisions about	1.The user must have initiated a search for accommodation. 2.The host whose profile is being explored must be registered in the system and have	1.he user can either choose to contact the host for further inquiries or proceed with booking the accommodation. 2.The system records the user's interaction with the host's profile for analytics purposes	1.User initiates a search for accommodation then selects a host from the search results. 2.User views detailed information about the host	1.the system displays a message indicating missing information and prompts the user to contact support for assistance. 2.The user reads negative reviews	If a host deletes their profile or discontinues their participation in the system, the profile becomes inaccessible to users, and the system removes

			accommodation during their stay.	available accommodation.		4.User decides whether to contact the host or proceed with booking.	about the host or accommodation, prompting them to look for another option.	it from search results.
14	Report unserious hosts organization	traveler	empower users to report hosts or organizations that do not uphold the standards or seriousness expected in providing accommodation and cultural exchange experiences.	User must have encountered a host or organization that they believe is unserious or does not meet the expected standards.	1.The reported host or organization is flagged for review by system administrators. 2.Depending on the severity of the report, actions may be taken such as warning the host, suspending their account, or removing them from the platform.	1.User encounters a host or organization that they believe is unserious or does not meet the expected standards. 2.User navigates to the appropriate section of the system to report the host or organization. 3.User provides details about their experience and reasons for reporting then submits the report.	The reported host or organization is reviewed by system administrators, but no action is deemed necessary based on the information provided. The user is informed of the outcome of the review.	The system experiences downtime or errors, preventing users from submitting reports. In such cases, users may be encouraged to try again later or contact support for assistance.
15	Pay yearly subscription	traveler	enable users to subscribe to a yearly plan, granting them access to premium features and benefits within the platform for a period of one year.	1.User must have selected the yearly subscription option from the available plans 2. User must have a valid payment method registered in the system.	1.User's subscription status is updated to reflect the yearly plan. 2.User gains access to premium features and benefits associated with the subscription. 3.Payment is processed successfully, and a confirmation is sent to the user.	1.User selects the yearly subscription option then confirms the subscription and proceeds to payment. 2.User enters payment details and completes the transaction.	1.The user's payment method is declined or encounters an error during processing. In this case, the system prompts the user to verify their payment details or try an alternative payment method. 2.The user decides to cancel the subscription after payment but before the subscription period begins. The system refunds the user according to the cancellation policy.	1.If the user already has an active yearly subscription, the system notifies the user and may offer options such as renewing the subscription or managing the existing plan. 2.The system experiences downtime or errors during the subscription process, preventing users from completing the transaction. In such cases, users may be advised to try again later or contact support for assistance.
16	List accommod-ation	host	enable hosts to list their accommodations on the platform, making them available for guests to discover and	1.Host must have completed the registration process and provided necessary information about the accommodation.	1.Accommodation listing is successfully added to the platform. 2.Accommodation becomes searchable and bookable for guests.	1.Host fills out the necessary details about the accommodation 2.Host submits the listing for review 3.System verifies the information	The system rejects the listing due to incomplete or inaccurate information provided by the host. In this case, the system	If the accommodation listing contains prohibited content or violates the platform's policies (e.g.,

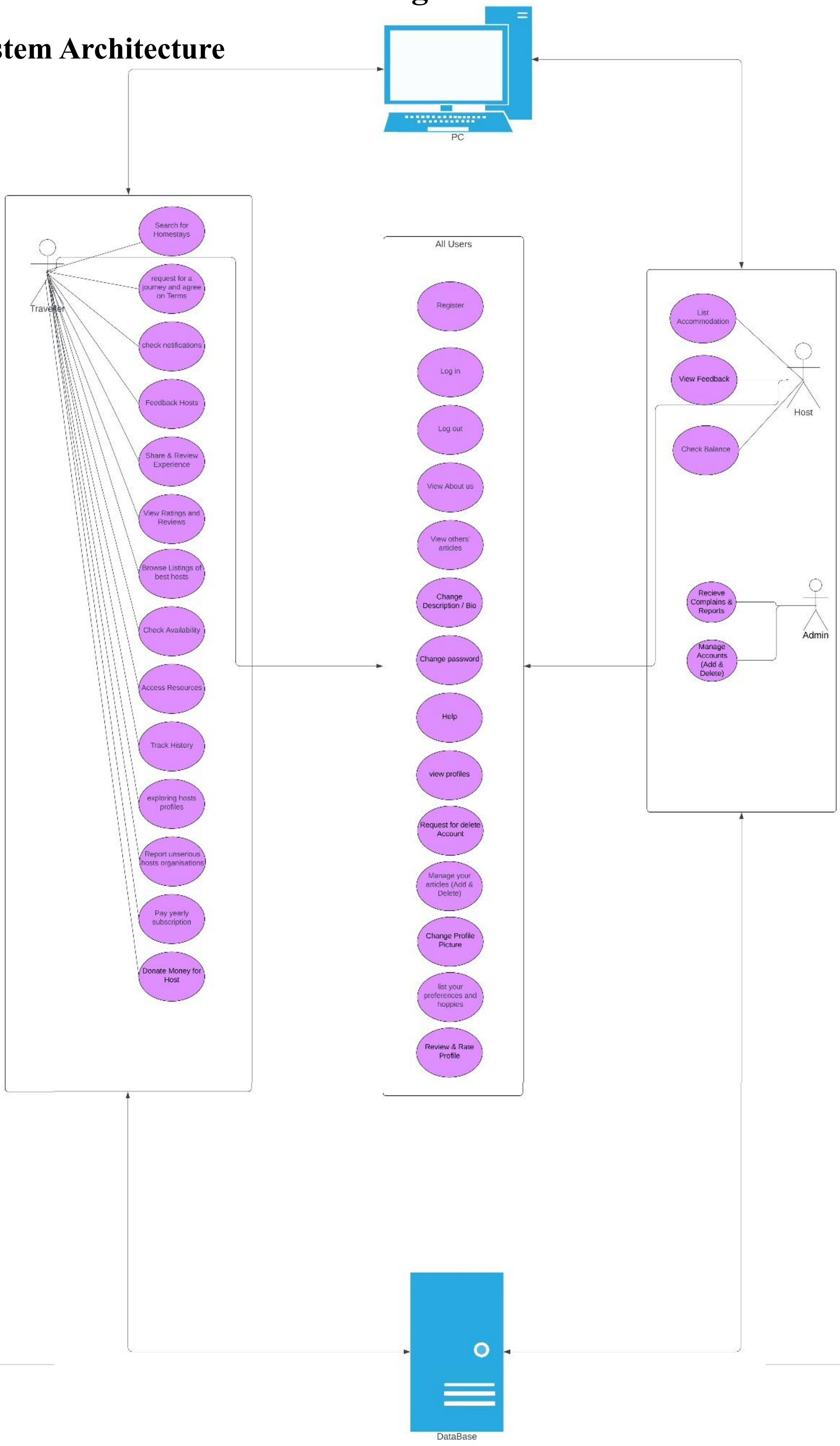
			book for their stays.	2. Host must agree to the terms and conditions of listing accommodations on the platform.	3.Host receives notifications about booking requests and other relevant activity related to their listing.	provided and approves the listing then 5. Accommodation becomes searchable and bookable by guests.	notifies the host of the reasons for rejection and may provide guidance on correcting the issues.	discriminatory language, illegal activities), the system rejects the listing and notifies the host of the violation.

c) Package Diagram grouping relevant Use Cases into Packages.



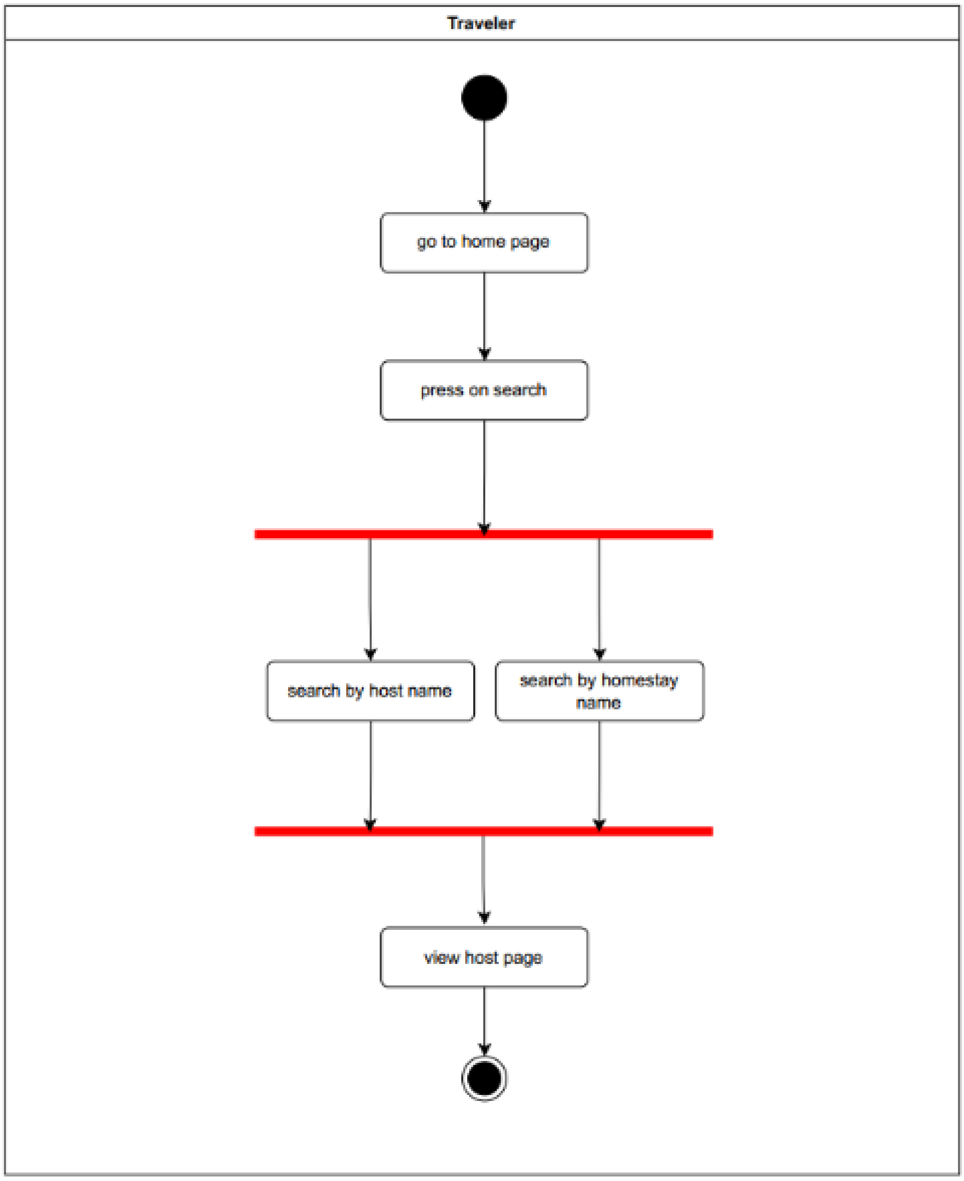
IX. Structural & Behavioral Diagrams:

a) System Architecture

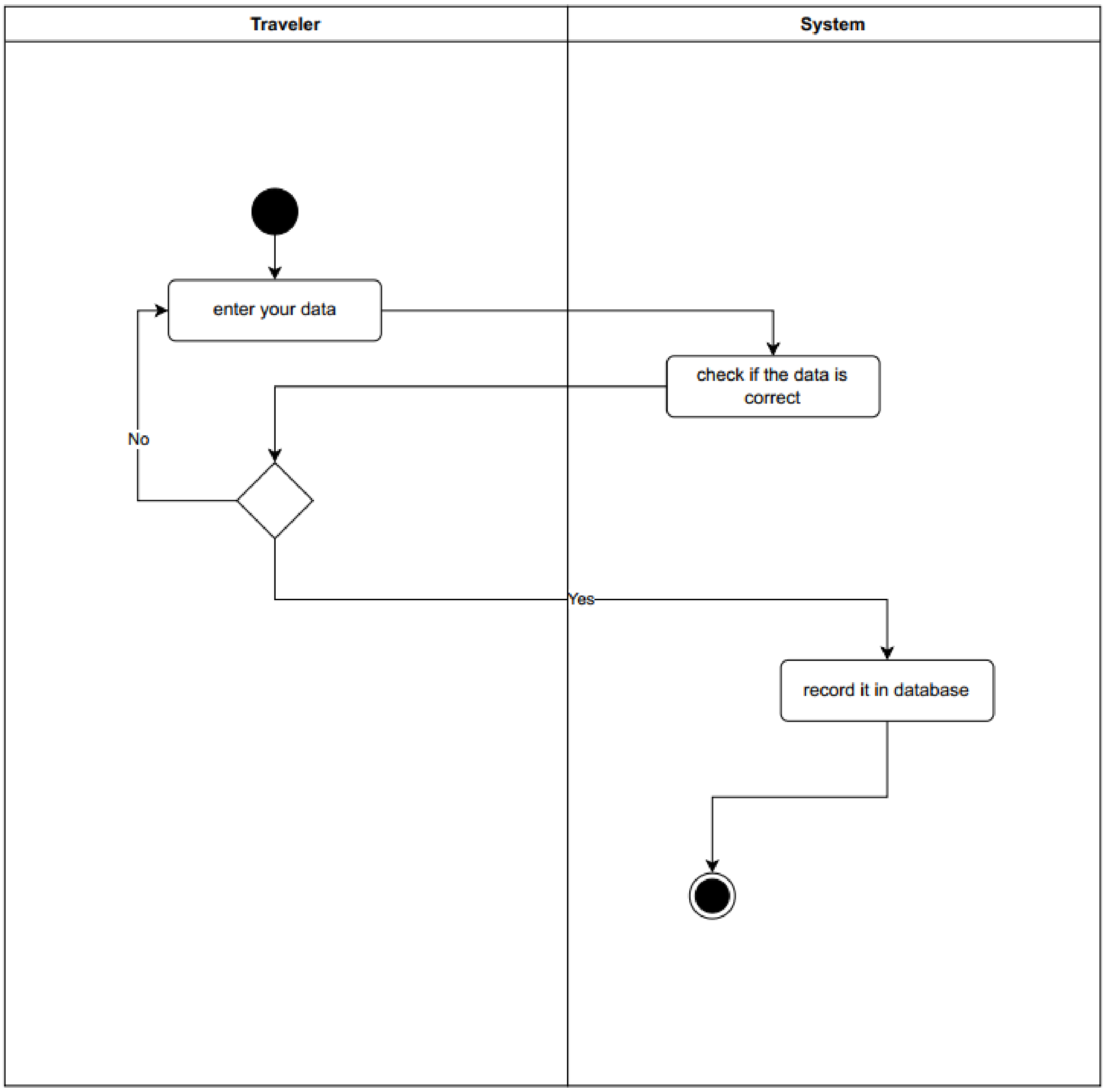


b) Activity Diagrams (for every major business process in the system, at least 6 diagrams.)

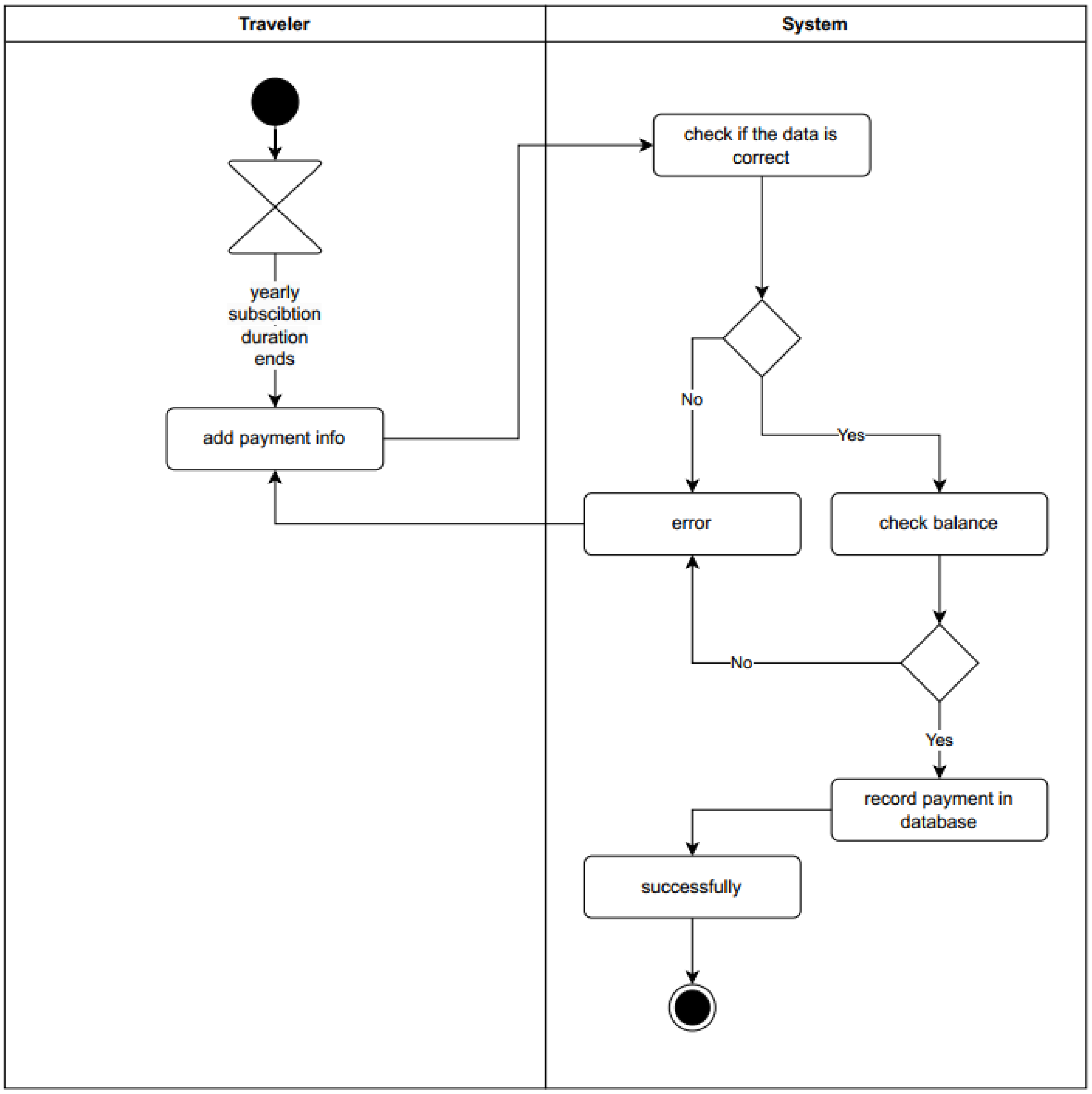
Search for Homestays



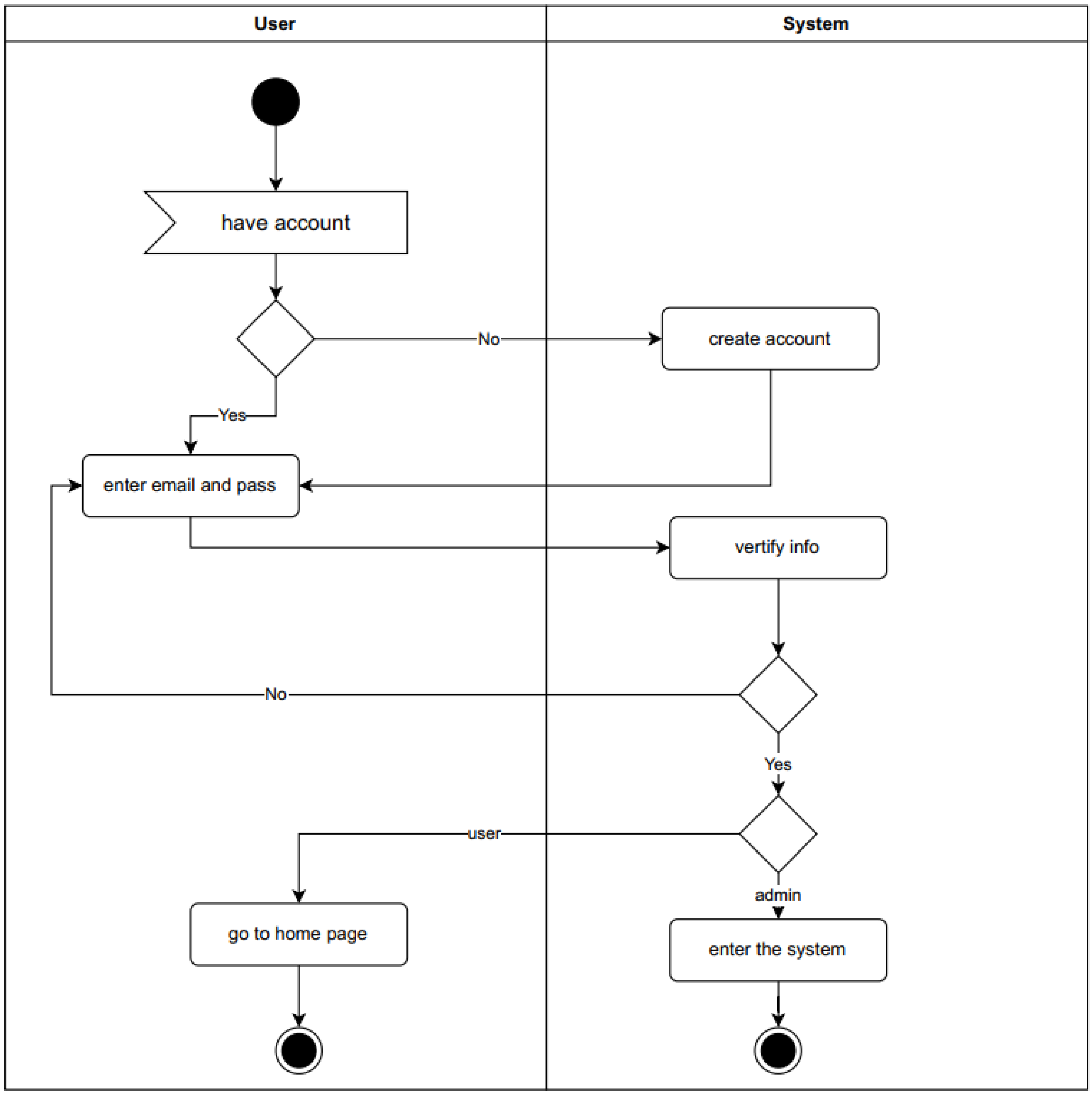
Register


Traveler
System
enter your data
check if the data is correct
No
Yes
record it in database

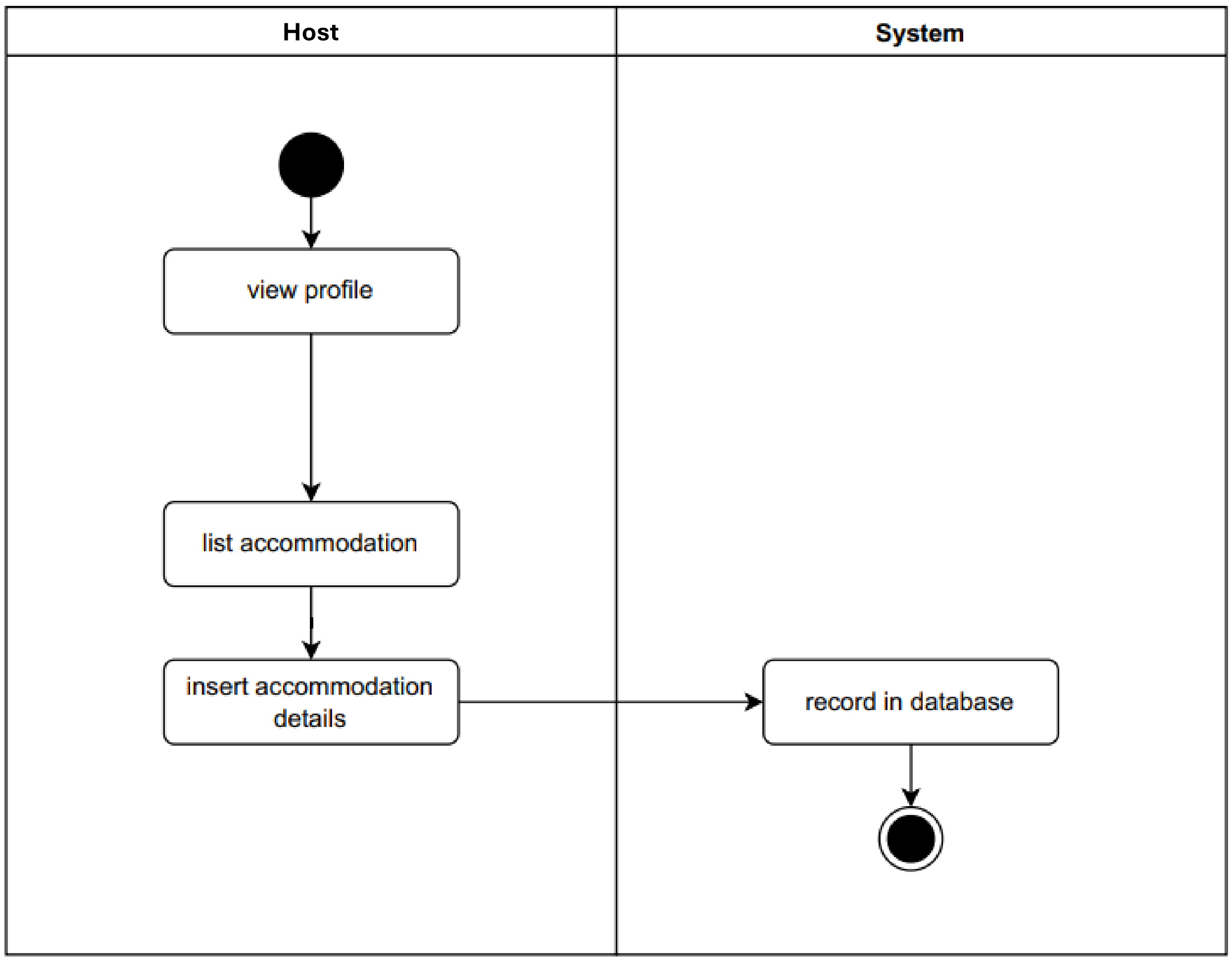
Payment for Yearly Subscribtion


Traveler
System
yearly subscibtion duration ends
add payment info
check if the data is correct
No
Yes
error
check balance
No
Yes
record payment in database
successfully

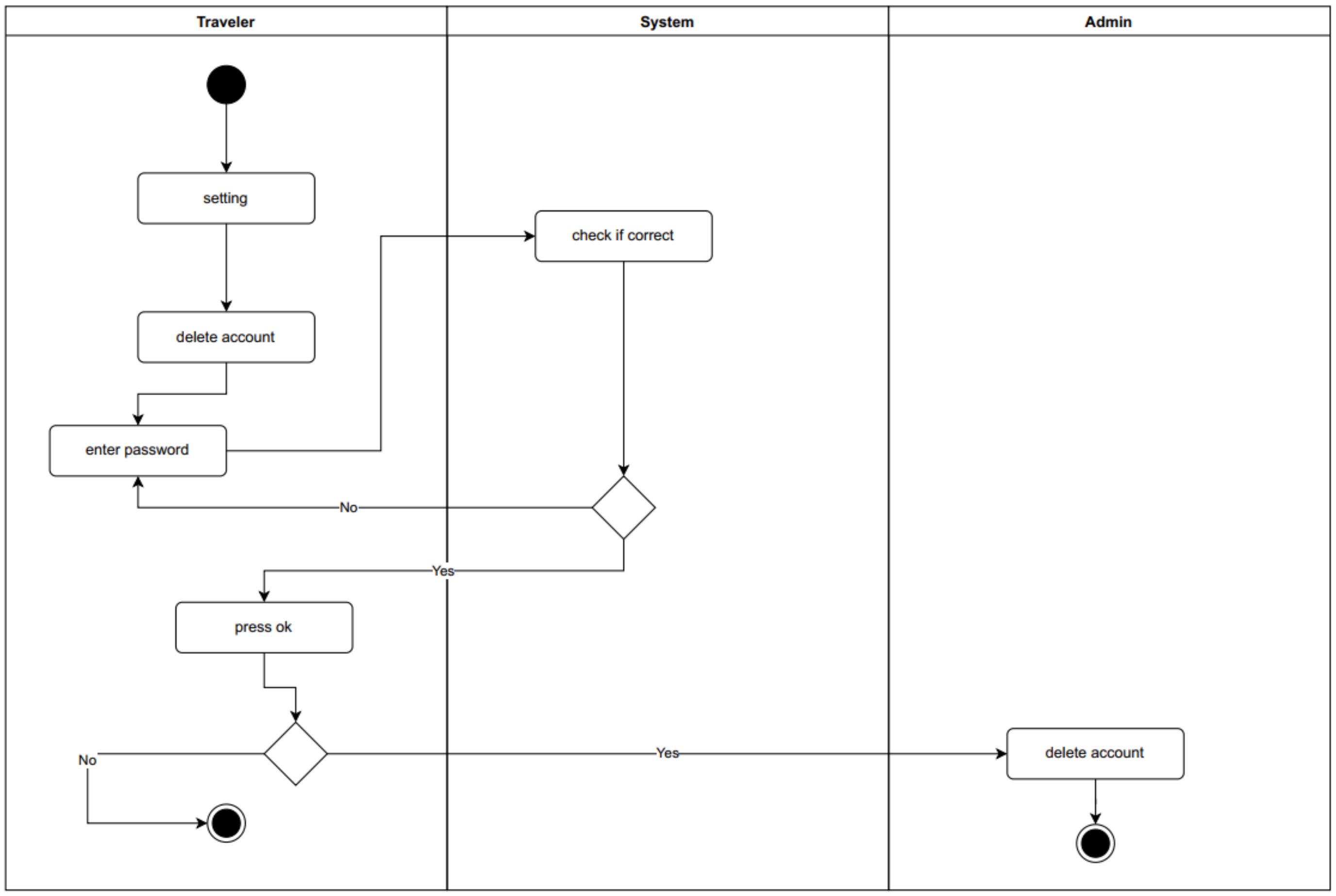
Login

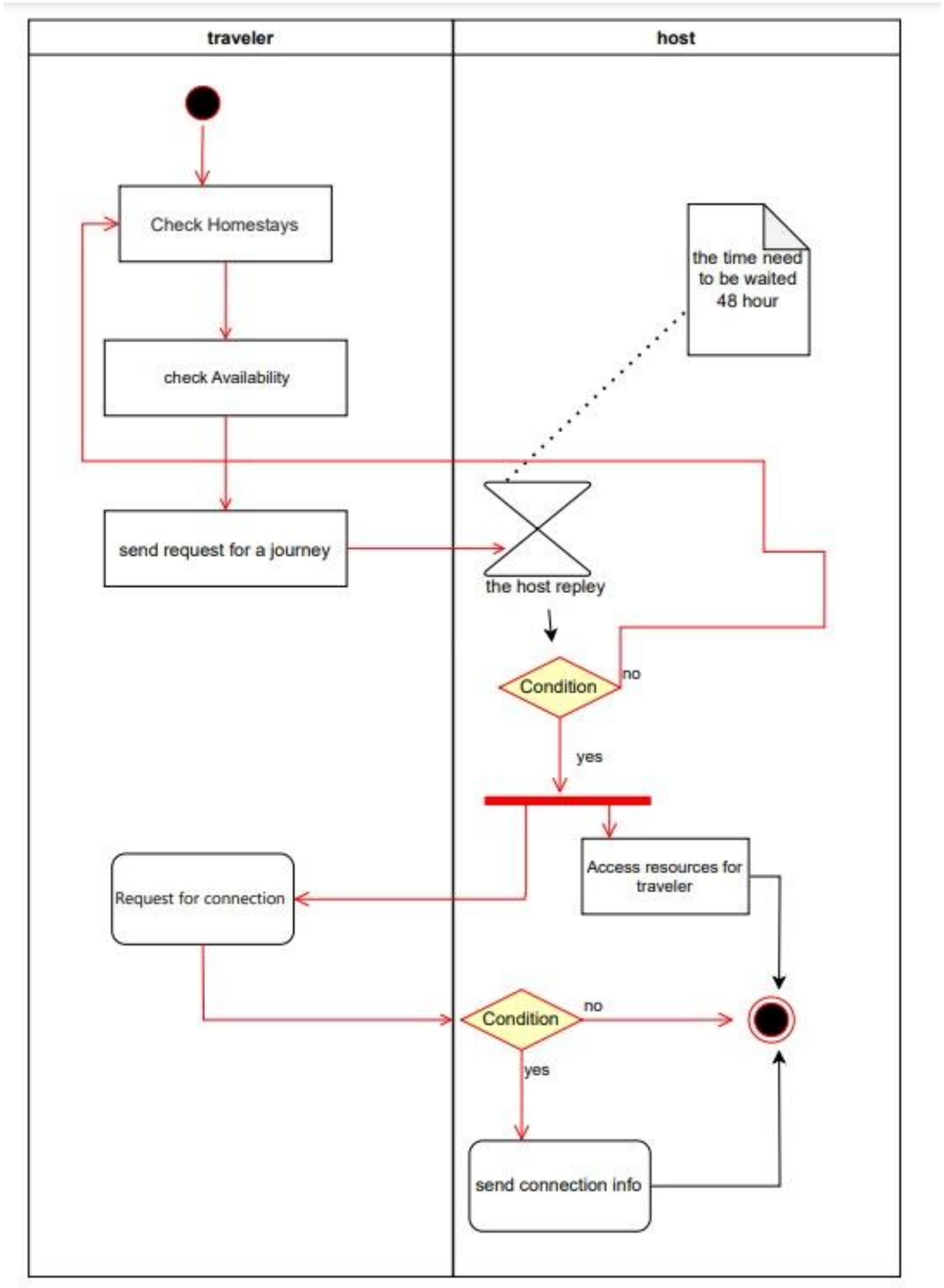

User
System
have account
No
create account
Yes
enter email and pass
vertify info
No
Yes
user
admin
go to home page
enter the system

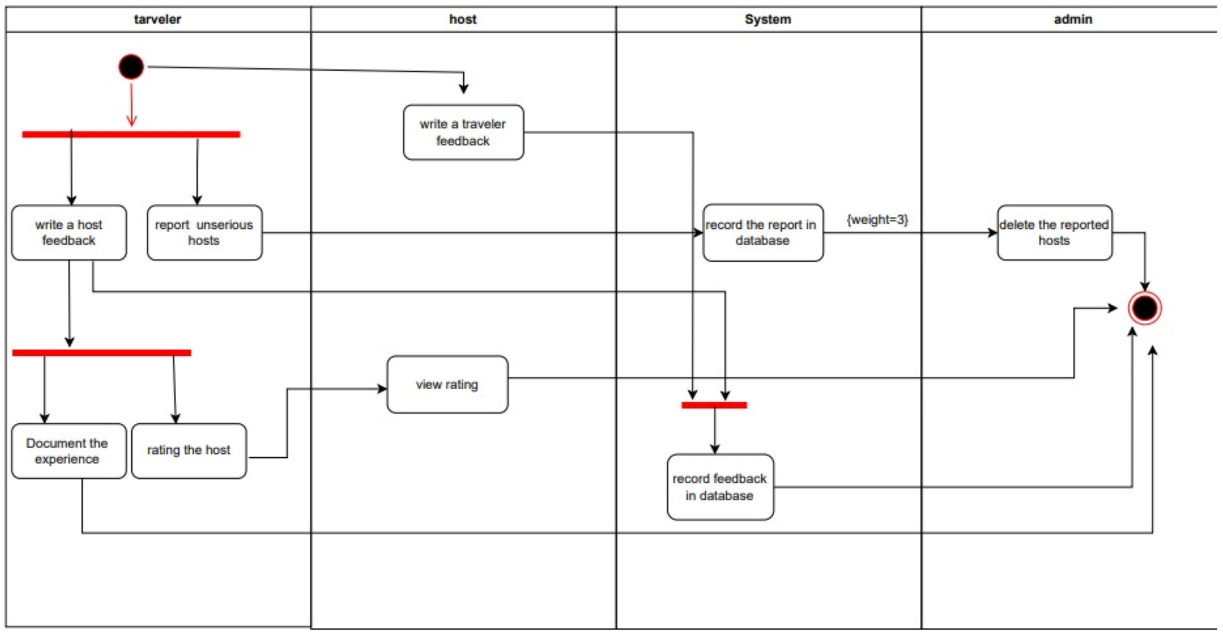
List Accomodation

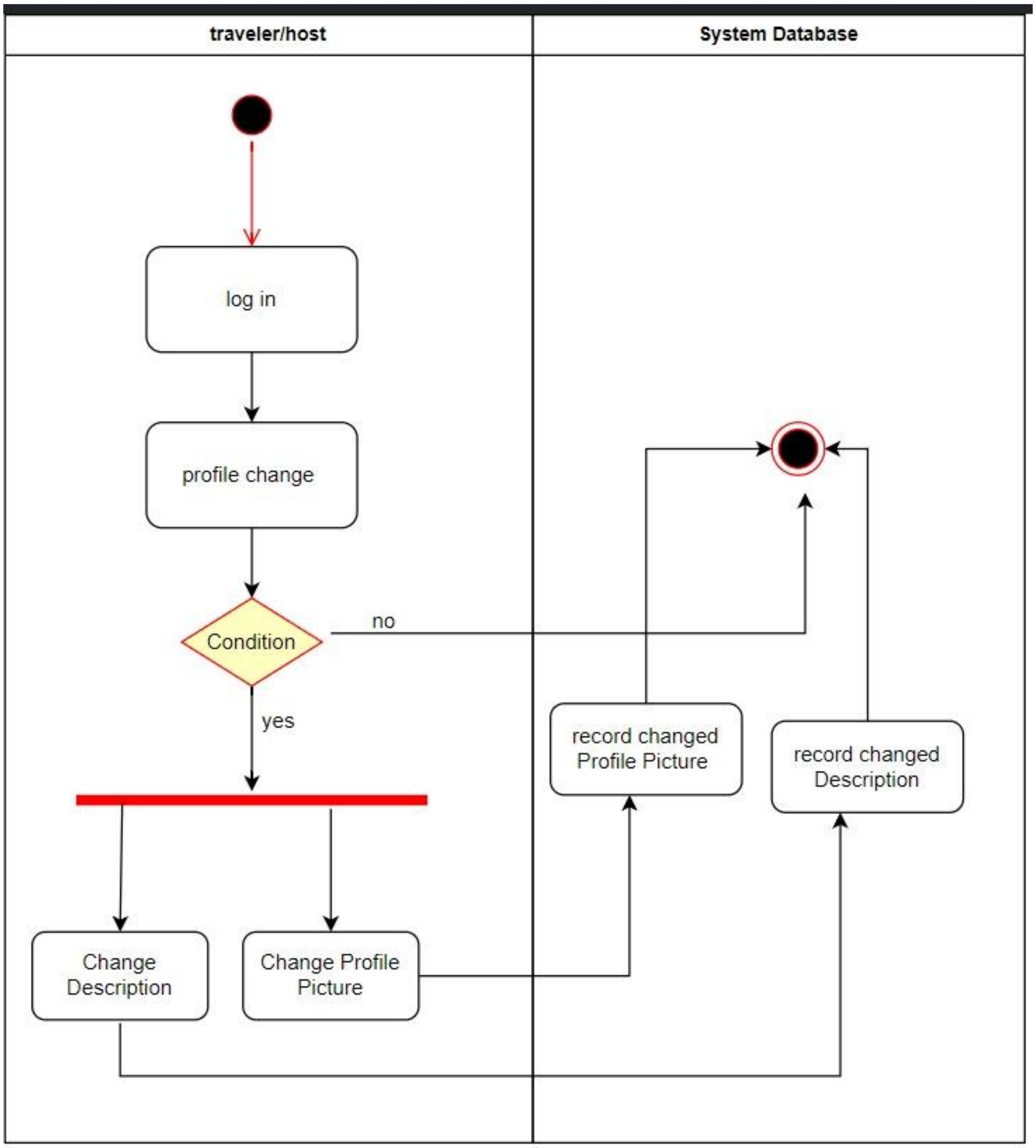


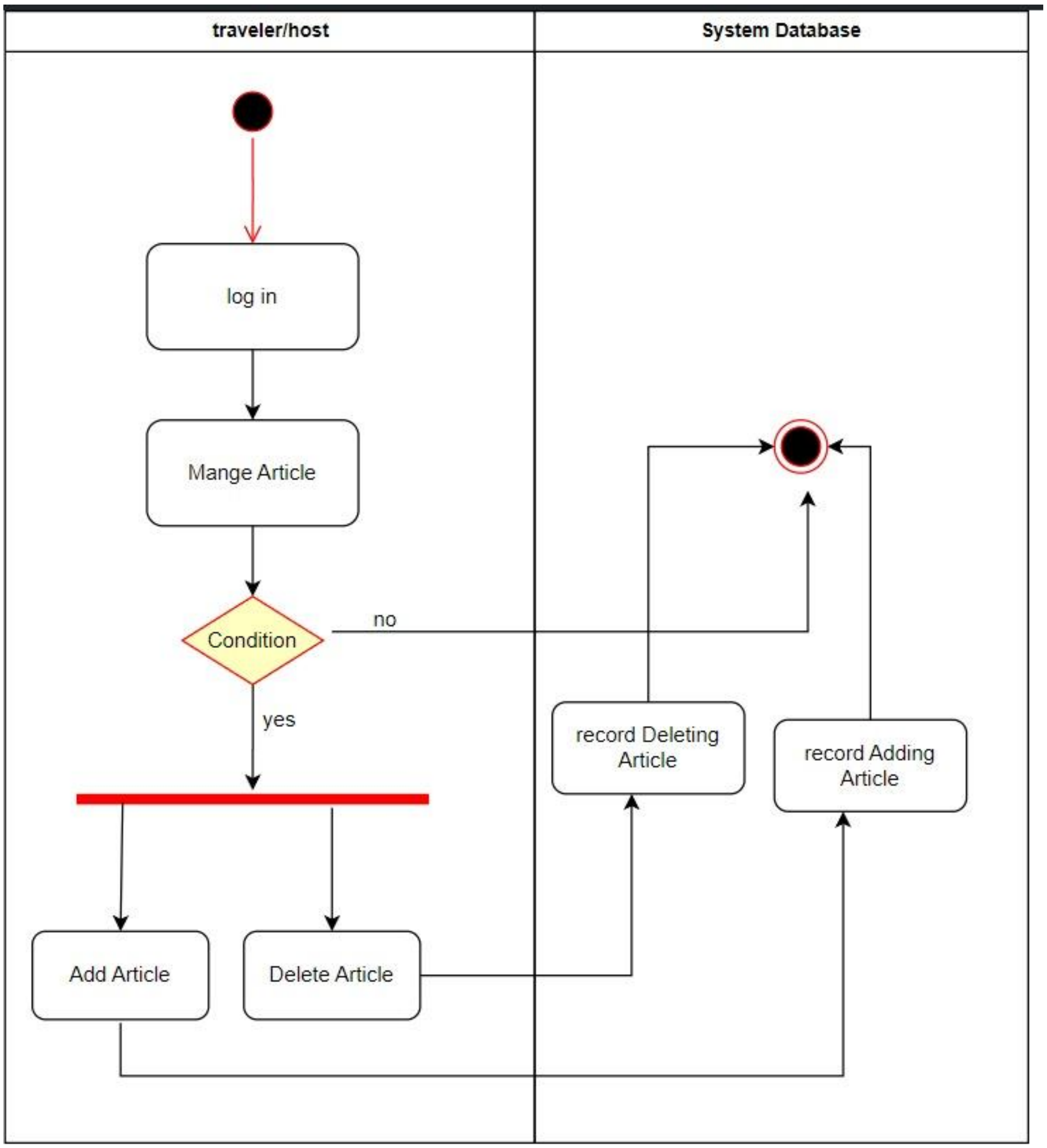
Delete Account



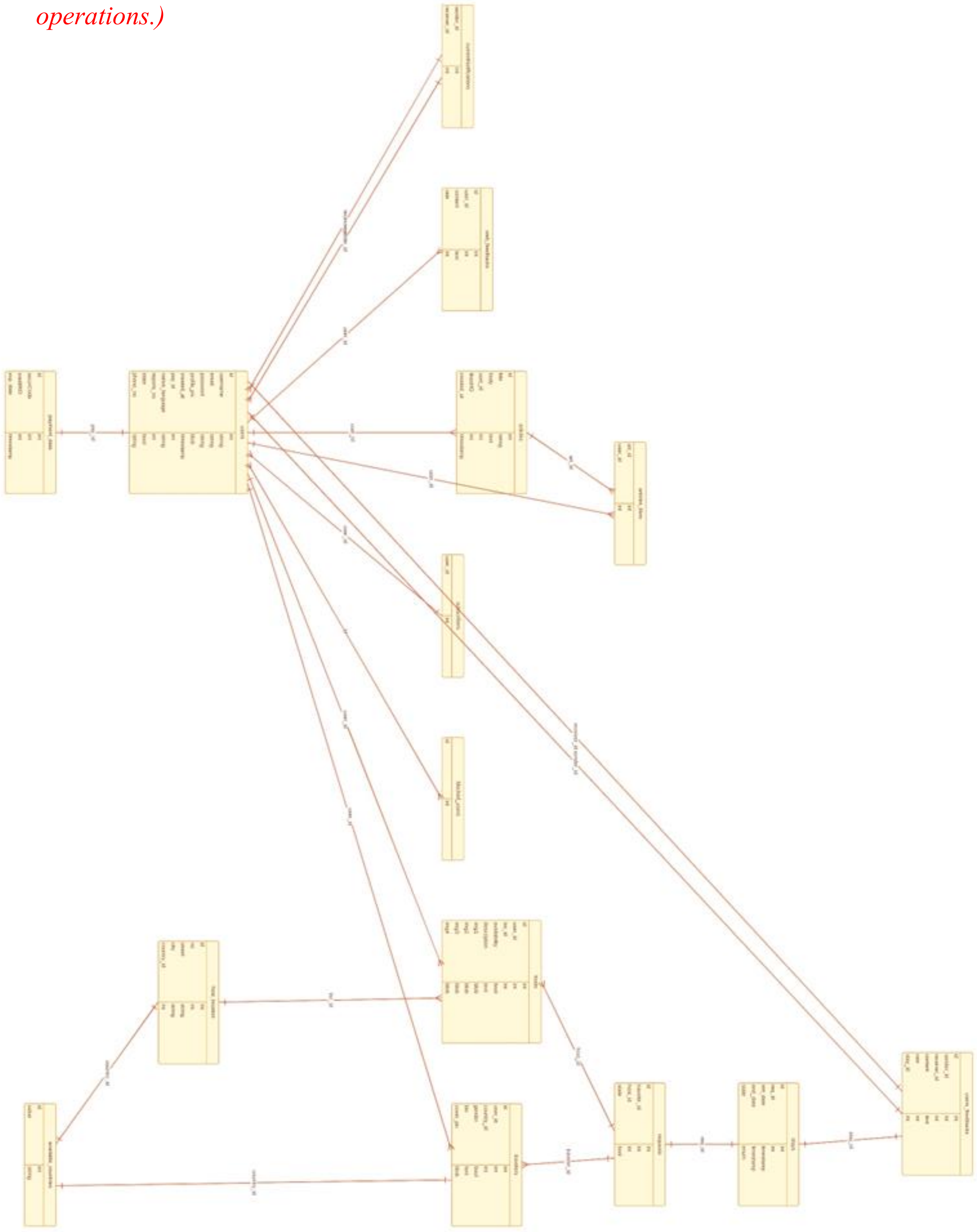

traveler
host
Check Homestays
check Availability
send request for a journey
the time need to be waited 48 hour
the host repley
Condition
no
yes
Request for connection
Access resources for traveler
Condition
no
yes
send connection info


tarveler
host
System
admin
write a traveler feedback
write a host feedback
report unserious hosts
record the report in database
{weight=3}
delete the reported hosts
view rating
Document the experience
rating the host
record feedback in database

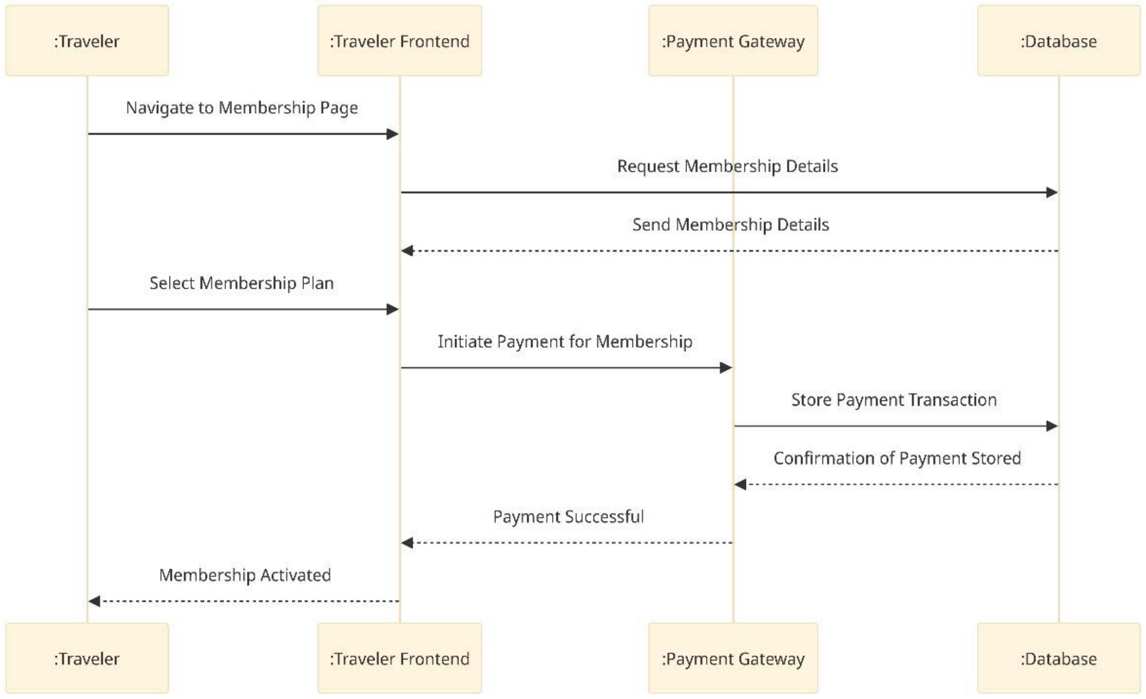

traveler/host
System Database
log in
profile change
Condition
no
yes
Change Description
Change Profile Picture
record changed Profile Picture
record changed Description

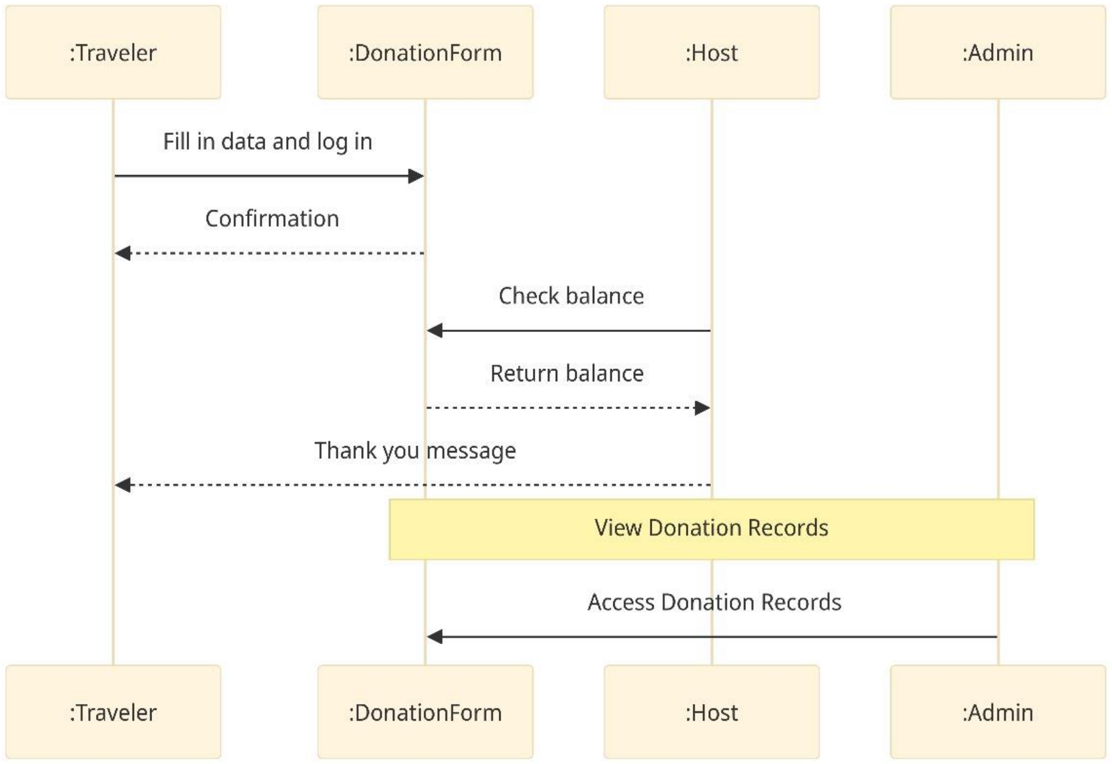

traveler/host
System Database
log in
Mange Article
Condition
no
yes
Add Article
Delete Article
record Deleting Article
record Adding Article

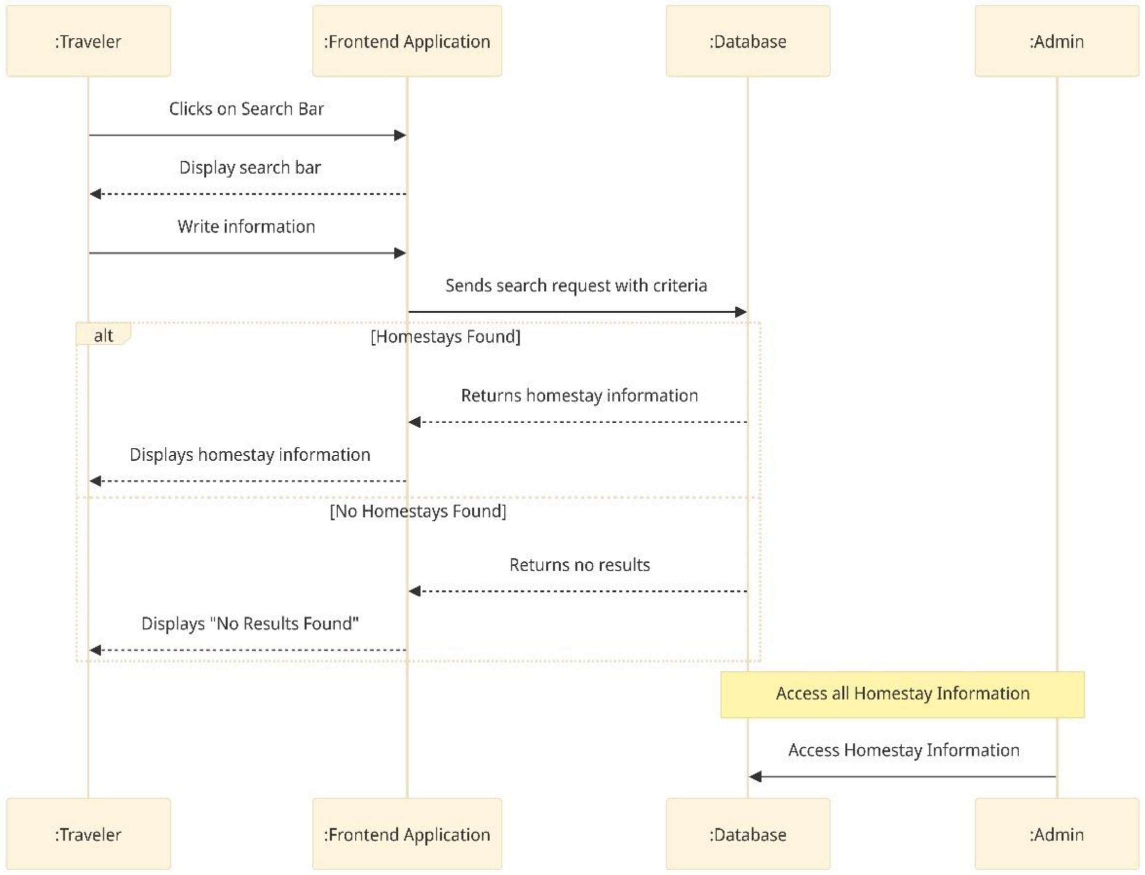
d) Class Diagram 1: An initial version based on the requirements and UseCase/Activity diagrams *(including classes, initial attributes, and basic operations.)*

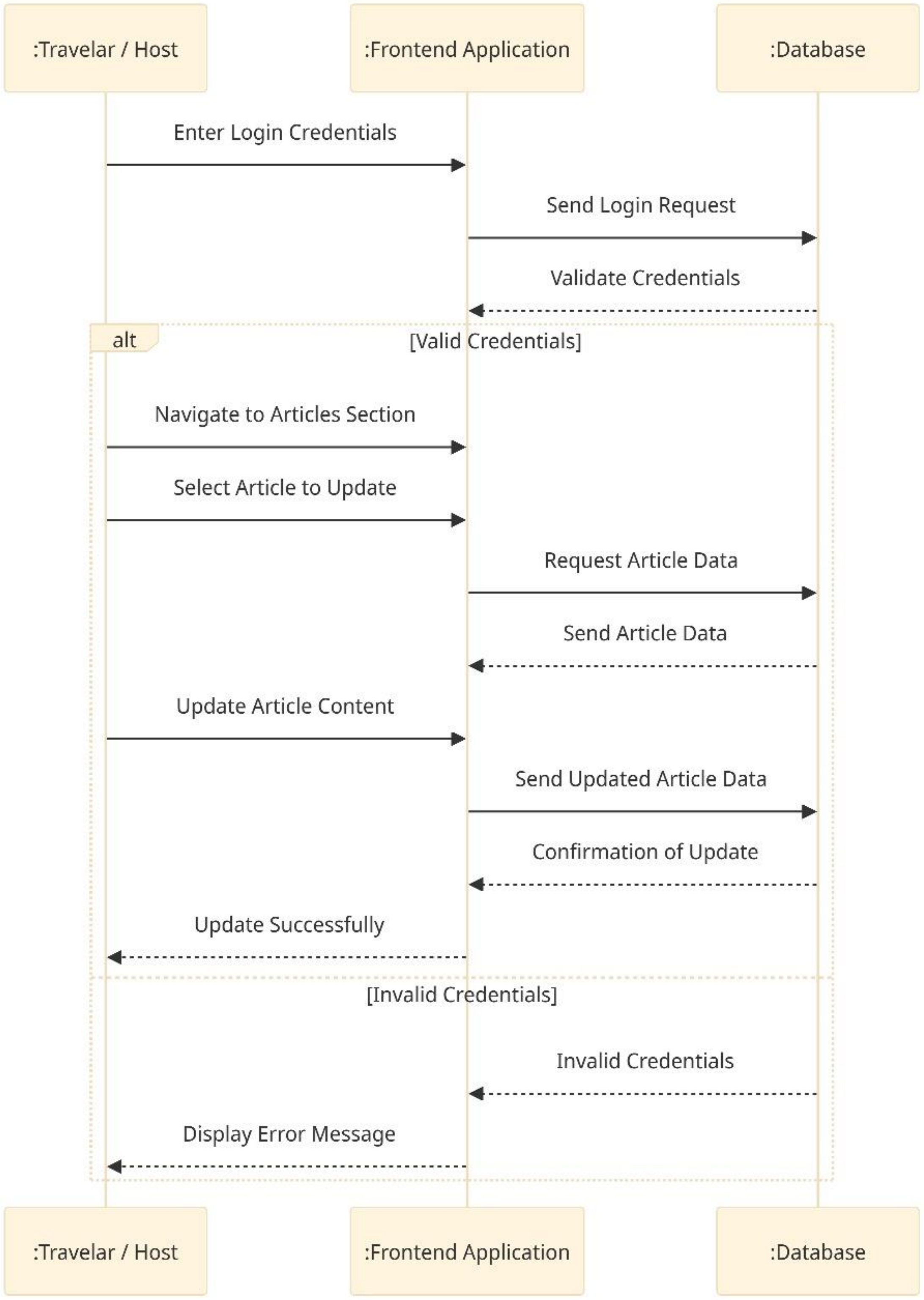


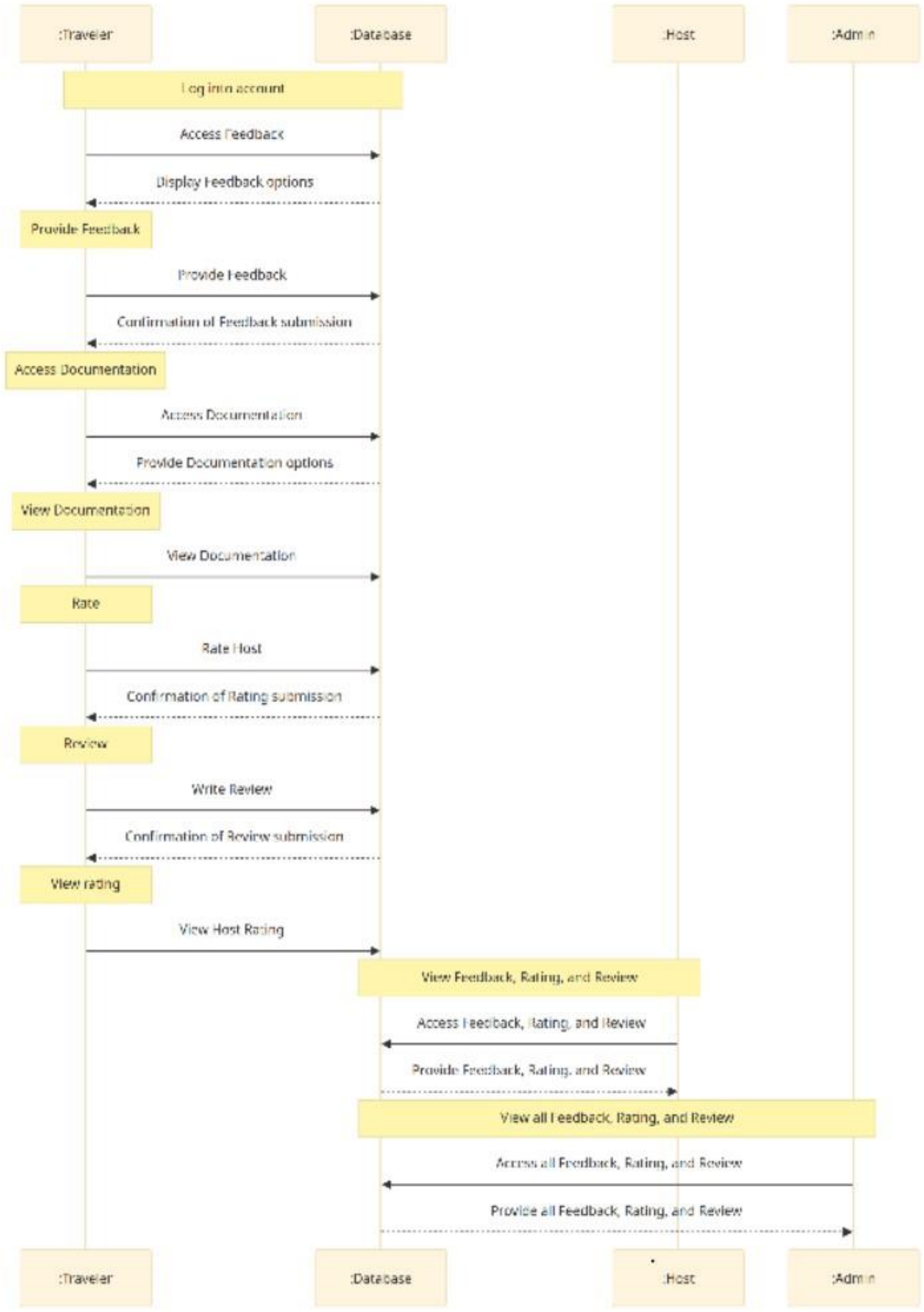
e) Sequence Diagram(s) (for every Use Case.)

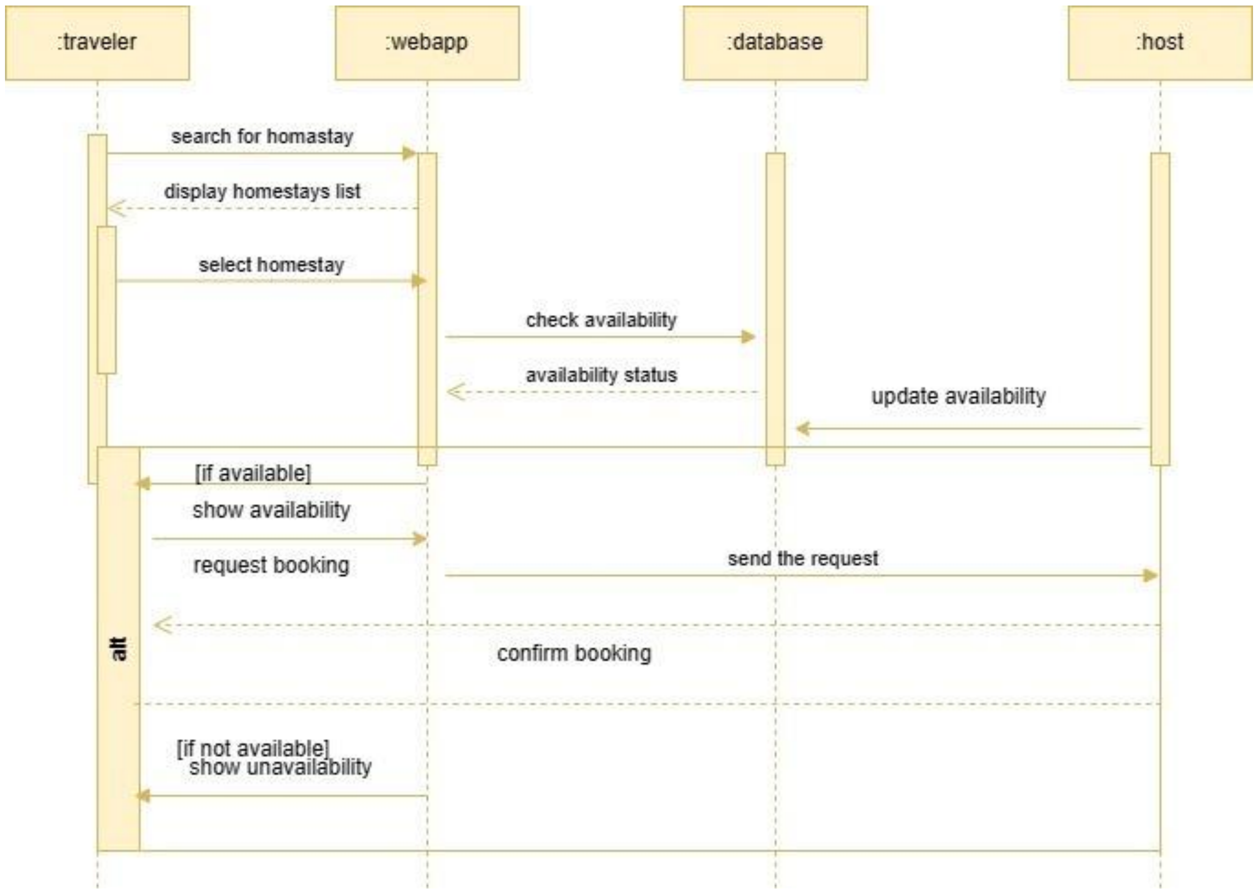


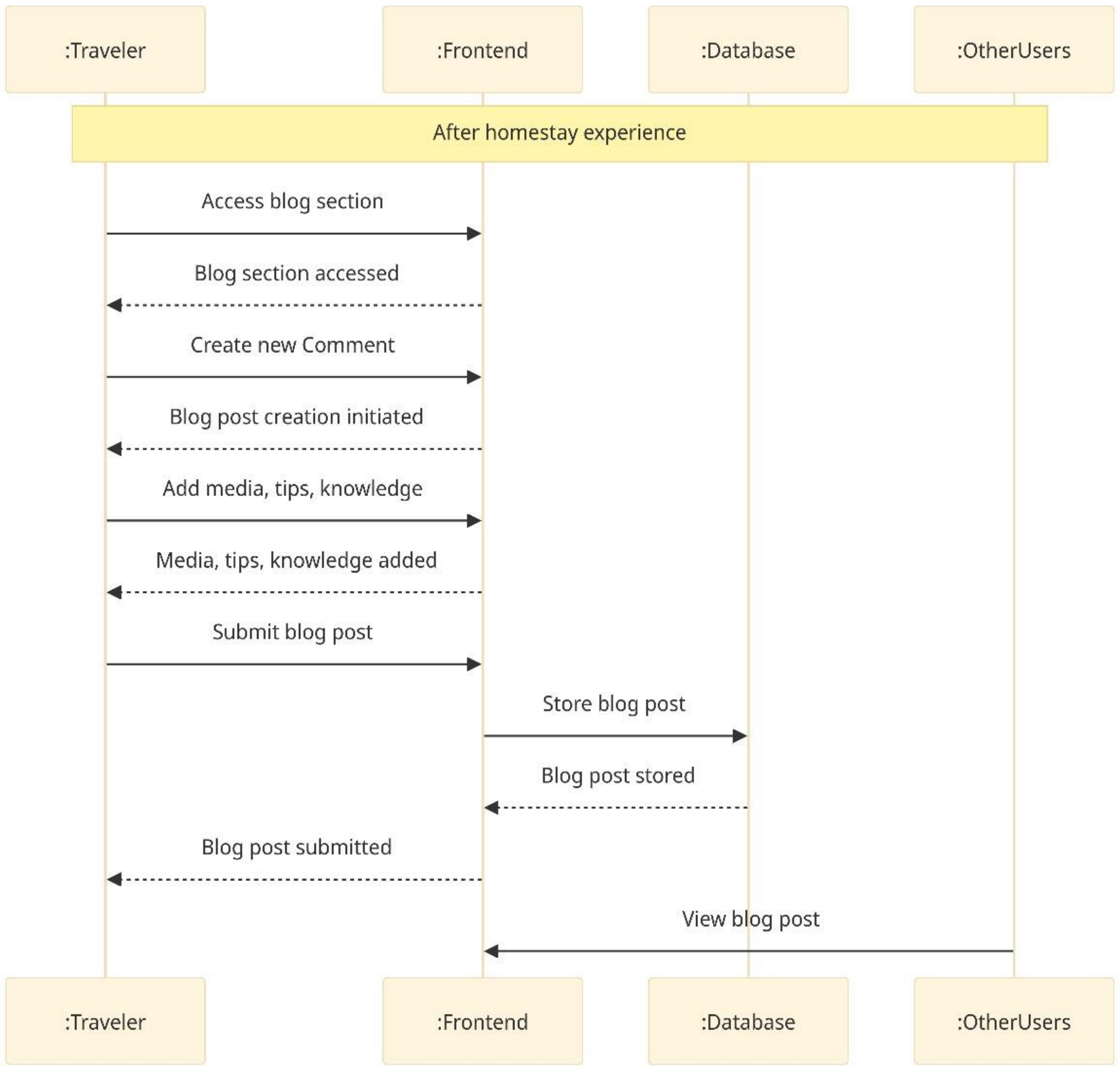


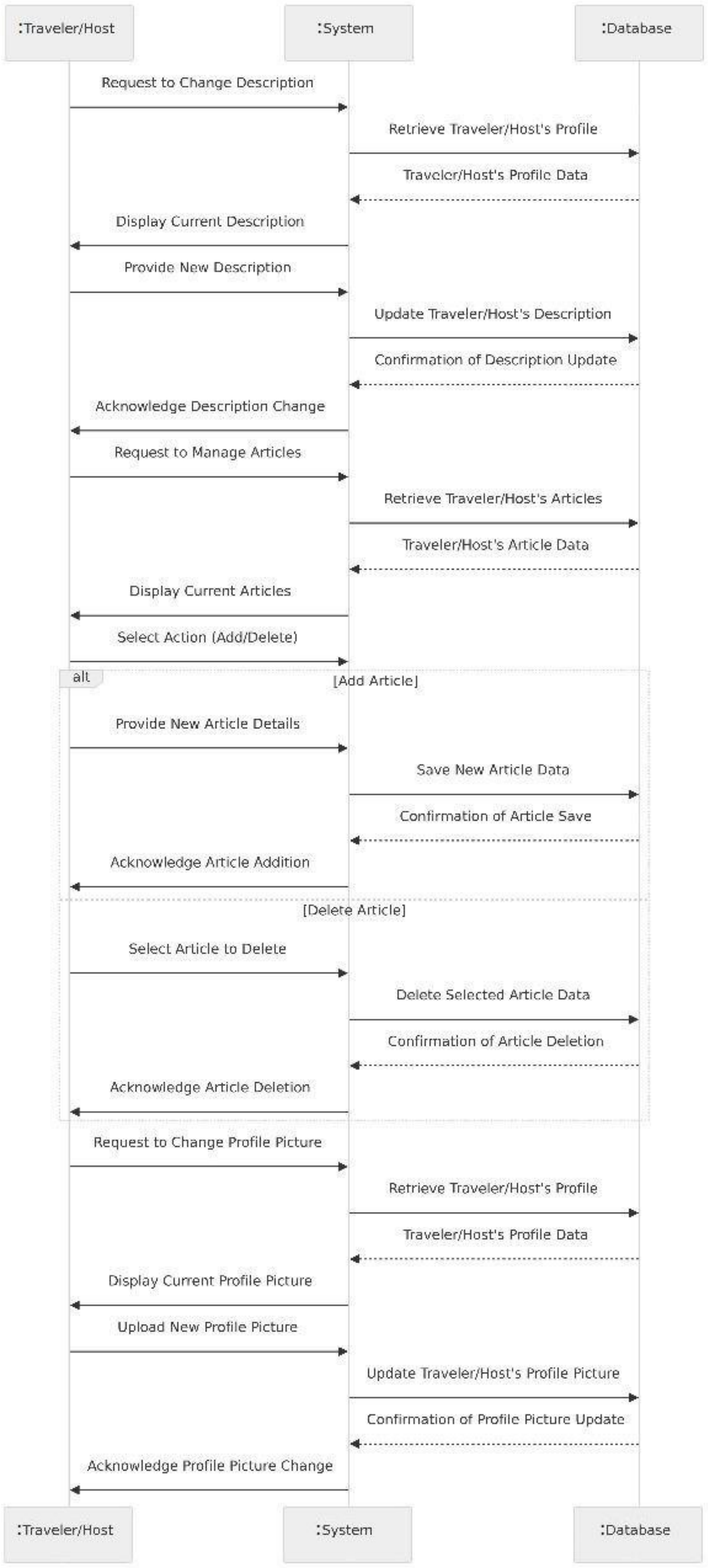

:Traveler
:Frontend Application
:Database
:Admin
Clicks on Search Bar
Display search bar
Write information
Sends search request with criteria
alt
[Homestays Found]
Returns homestay information
Displays homestay information
[No Homestays Found]
Returns no results
Displays "No Results Found"
Access all Homestay Information
Access Homestay Information
:Traveler
:Frontend Application
:Database
:Admin

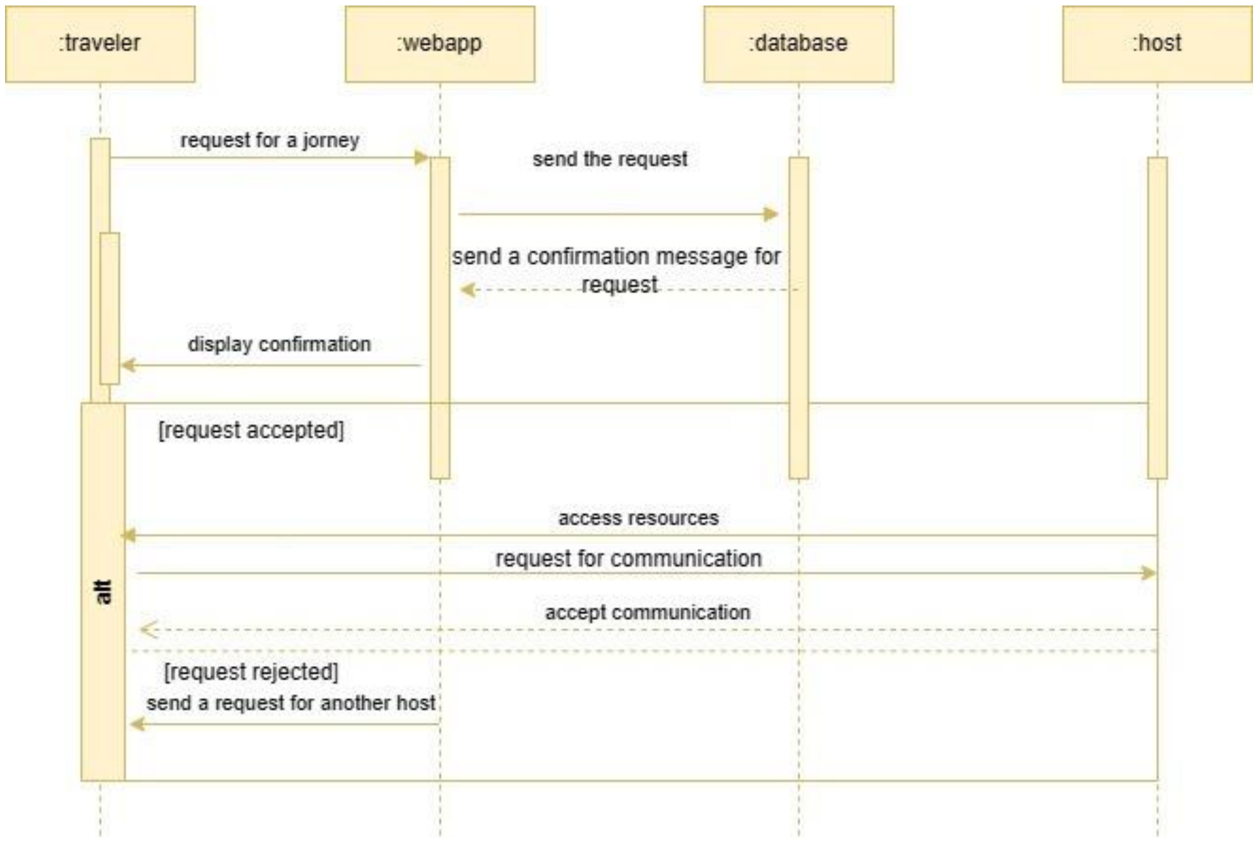

:Travelar / Host
:Frontend Application
:Database
Enter Login Credentials
Send Login Request
Validate Credentials
alt
[Valid Credentials]
Navigate to Articles Section
Select Article to Update
Request Article Data
Send Article Data
Update Article Content
Send Updated Article Data
Confirmation of Update
Update Successfully
[Invalid Credentials]
Invalid Credentials
Display Error Message
:Travelar / Host
:Frontend Application
:Database

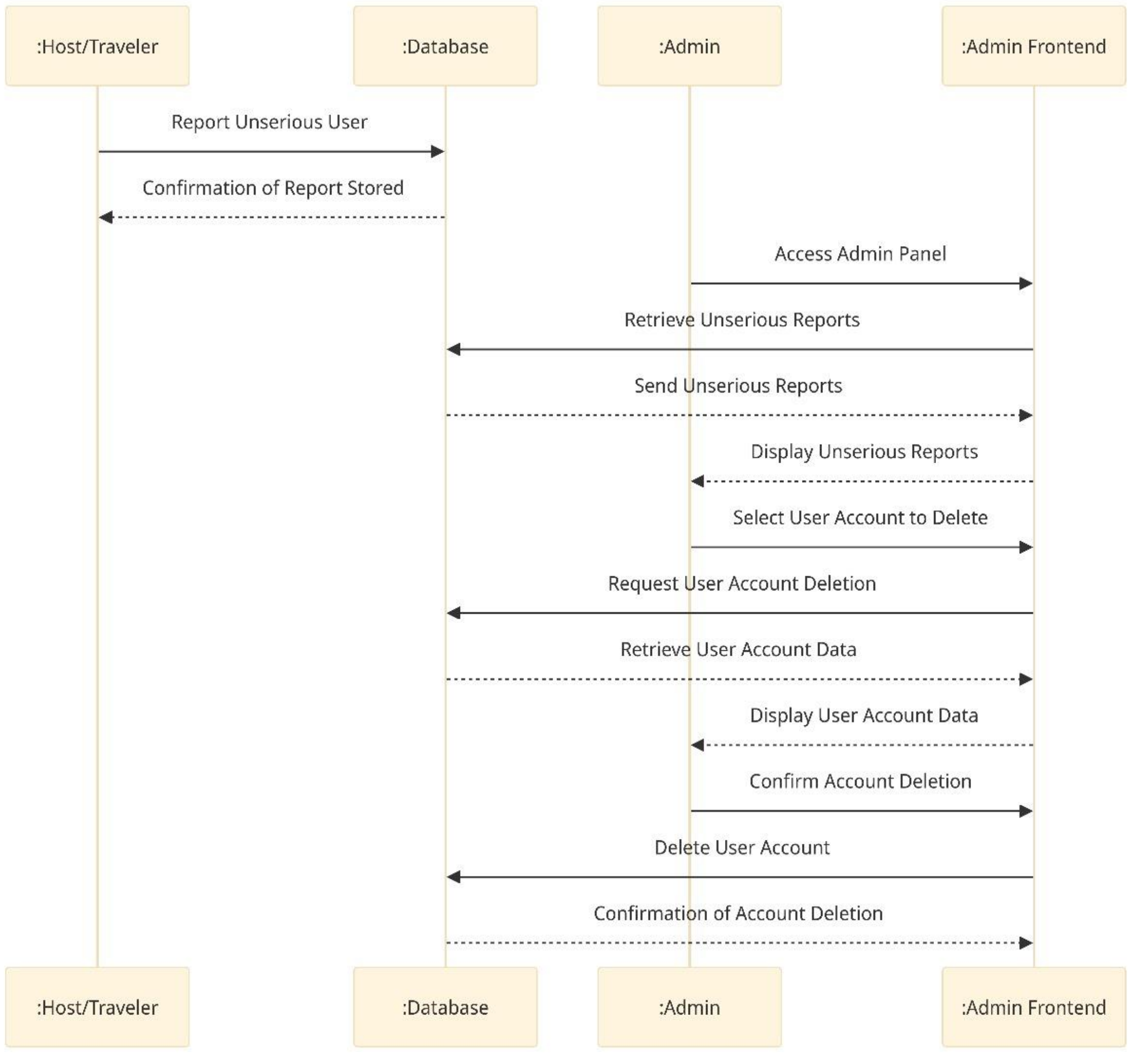

:Traveler
:Database
:Host
:Admin
Log into account
Access Feedback
Display Feedback options
Provide Feedback
Provide Feedback
Confirmation of Feedback submission
Access Documentation
Access Documentation
Provide Documentation options
View Documentation
View Documentation
Rate
Rate Host
Confirmation of Rating submission
Review
Write Review
Confirmation of Review submission
View rating
View Host Rating
View Feedback, Rating, and Review
Access Feedback, Rating, and Review
Provide Feedback, Rating, and Review
View all Feedback, Rating, and Review
Access all Feedback, Rating, and Review
Provide all Feedback, Rating, and Review
:Traveler
:Database
:Host
:Admin


:traveler
:webapp
:database
:host
search for homastay
display homestays list
select homestay
check availability
availability status
update availability
[if available]
show availability
request booking
send the request
confirm booking
alt
[if not available]
show unavailability

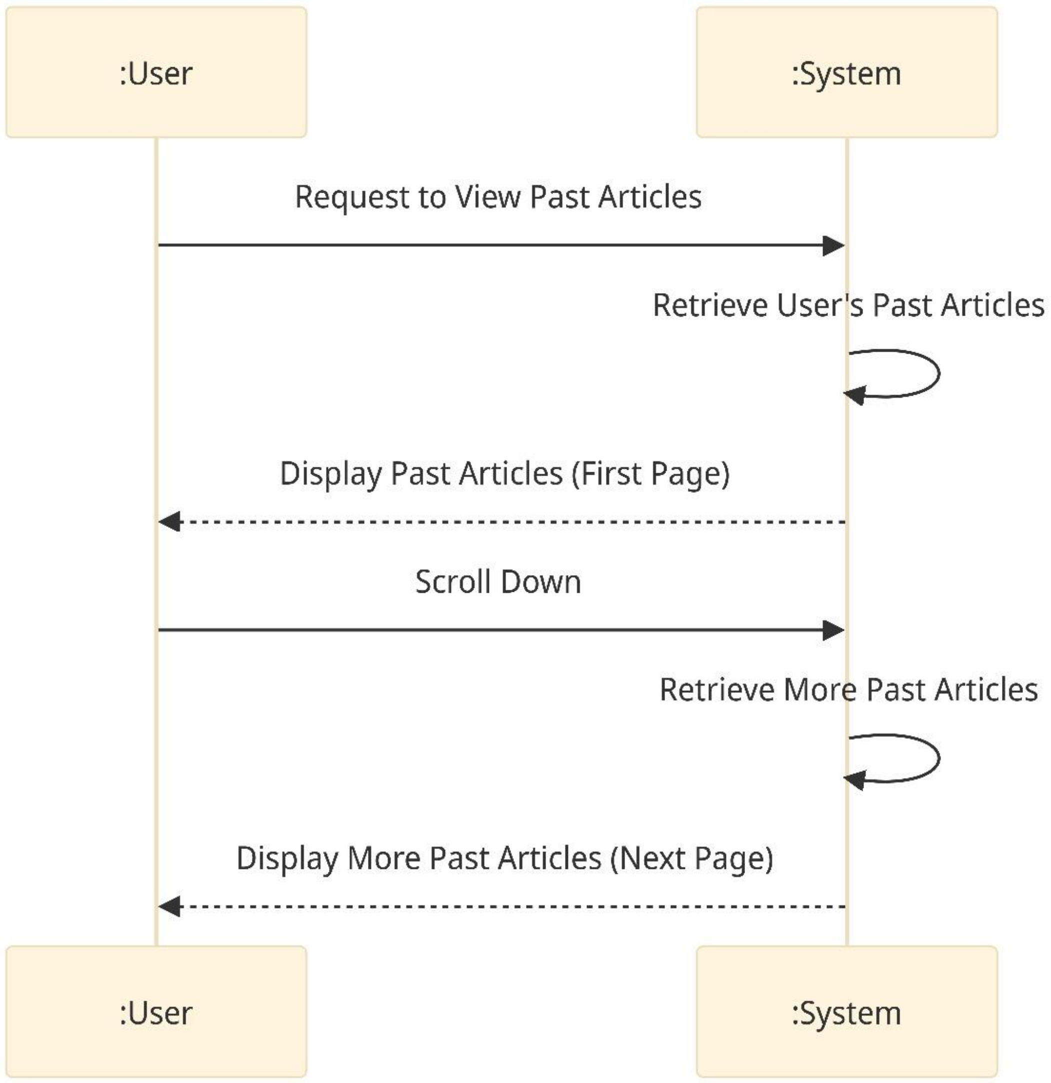

:Traveler
:Frontend
:Database
:OtherUsers
After homestay experience
Access blog section
Blog section accessed
Create new Comment
Blog post creation initiated
Add media, tips, knowledge
Media, tips, knowledge added
Submit blog post
Store blog post
Blog post stored
Blog post submitted
View blog post
:Traveler
:Frontend
:Database
:OtherUsers

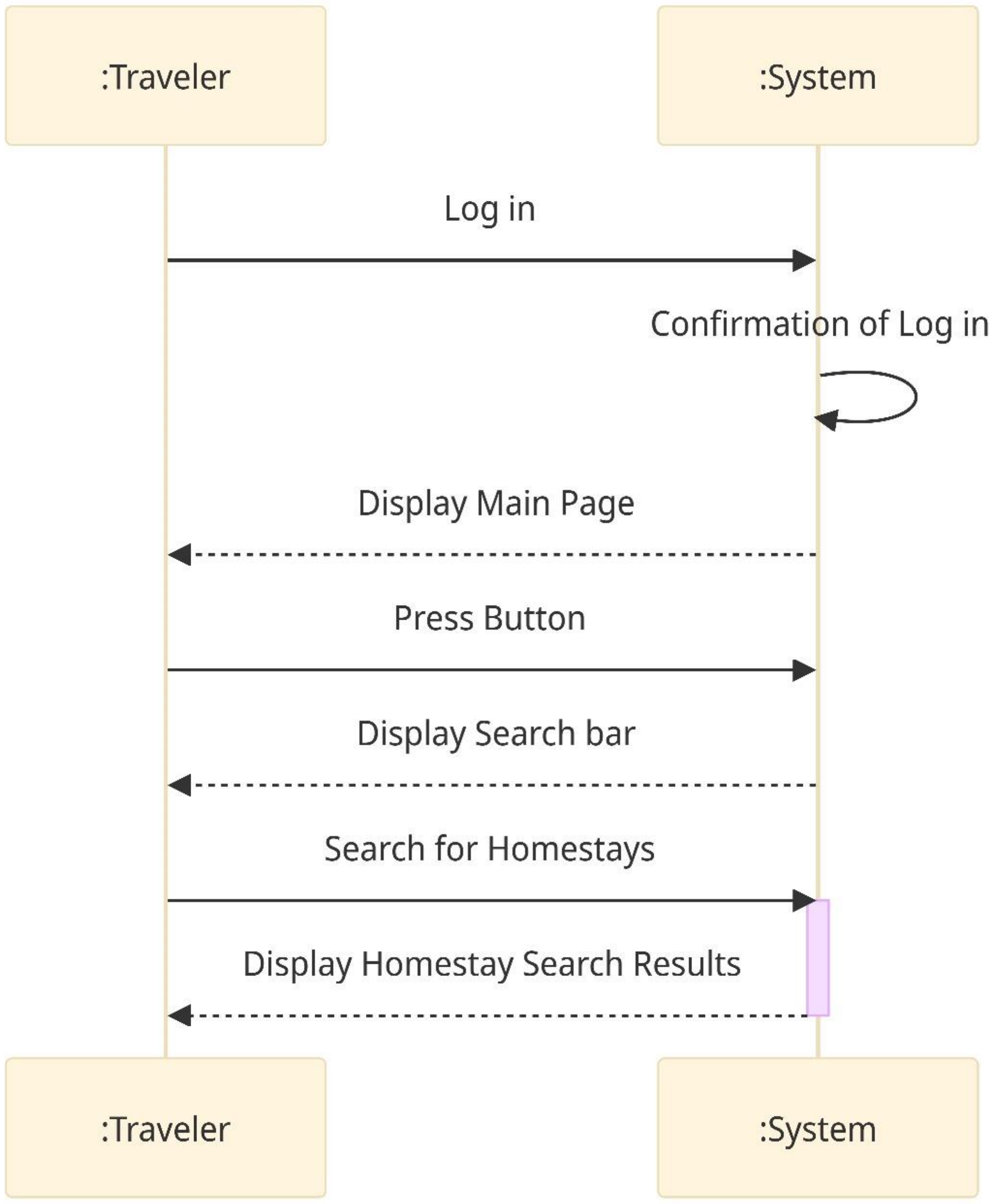

:Traveler/Host
:System
:Database
Request to Change Description
Retrieve Traveler/Host's Profile
Traveler/Host's Profile Data
Display Current Description
Provide New Description
Update Traveler/Host's Description
Confirmation of Description Update
Acknowledge Description Change
Request to Manage Articles
Retrieve Traveler/Host's Articles
Traveler/Host's Article Data
Display Current Articles
Select Action (Add/Delete)
alt
[Add Article]
Provide New Article Details
Save New Article Data
Confirmation of Article Save
Acknowledge Article Addition
[Delete Article]
Select Article to Delete
Delete Selected Article Data
Confirmation of Article Deletion
Acknowledge Article Deletion
Request to Change Profile Picture
Retrieve Traveler/Host's Profile
Traveler/Host's Profile Data
Display Current Profile Picture
Upload New Profile Picture
Update Traveler/Host's Profile Picture
Confirmation of Profile Picture Update
Acknowledge Profile Picture Change
:Traveler/Host
:System
:Database

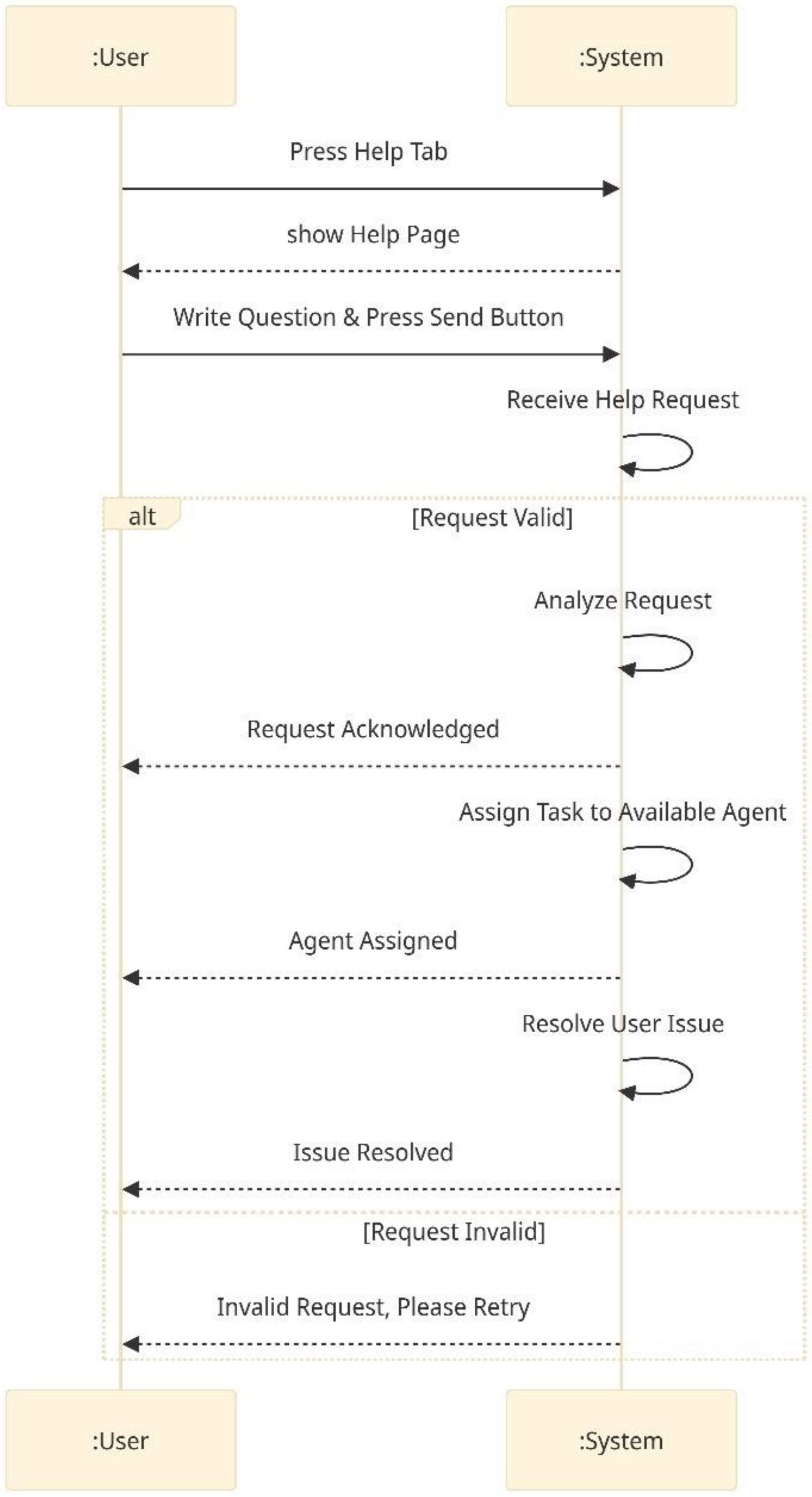

:traveler
:webapp
:database
:host
request for a jorney
send the request
send a confirmation message for request
display confirmation
alt
[request accepted]
access resources
request for communication
accept communication
[request rejected]
send a request for another host

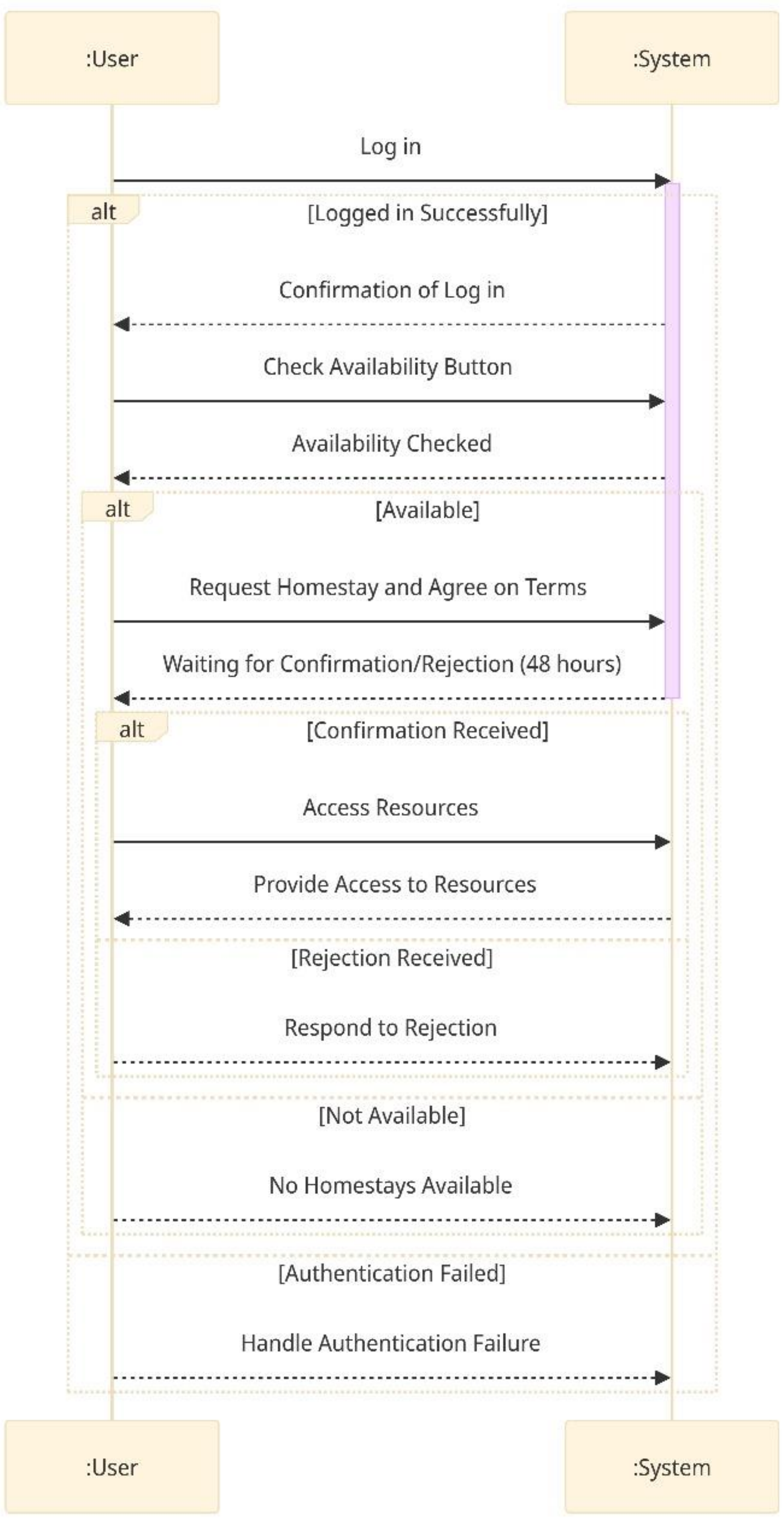

:Host/Traveler
:Database
:Admin
:Admin Frontend
Report Unserious User
Confirmation of Report Stored
Access Admin Panel
Retrieve Unserious Reports
Send Unserious Reports
Display Unserious Reports
Select User Account to Delete
Request User Account Deletion
Retrieve User Account Data
Display User Account Data
Confirm Account Deletion
Delete User Account
Confirmation of Account Deletion
:Host/Traveler
:Database
:Admin
:Admin Frontend

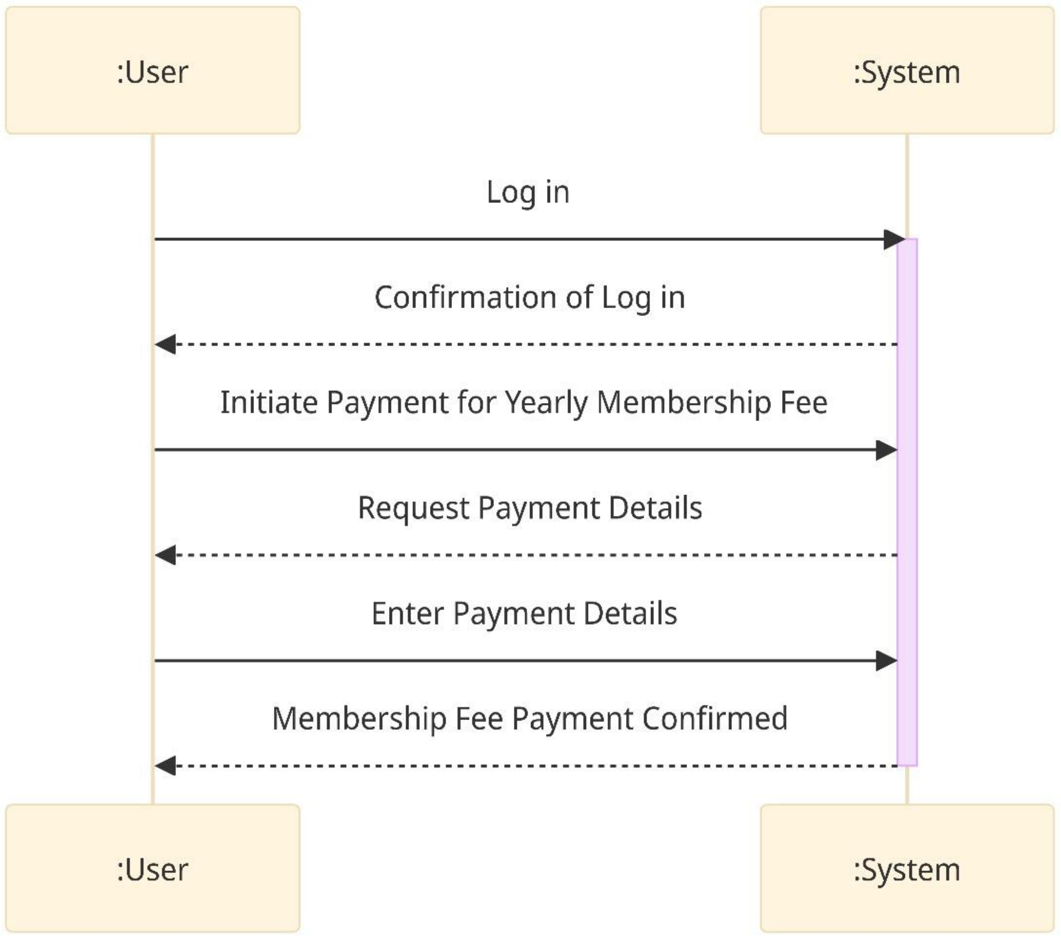
f) System Sequence Diagrams (SSDs) (at least 6 diagrams for every major business process in the system).

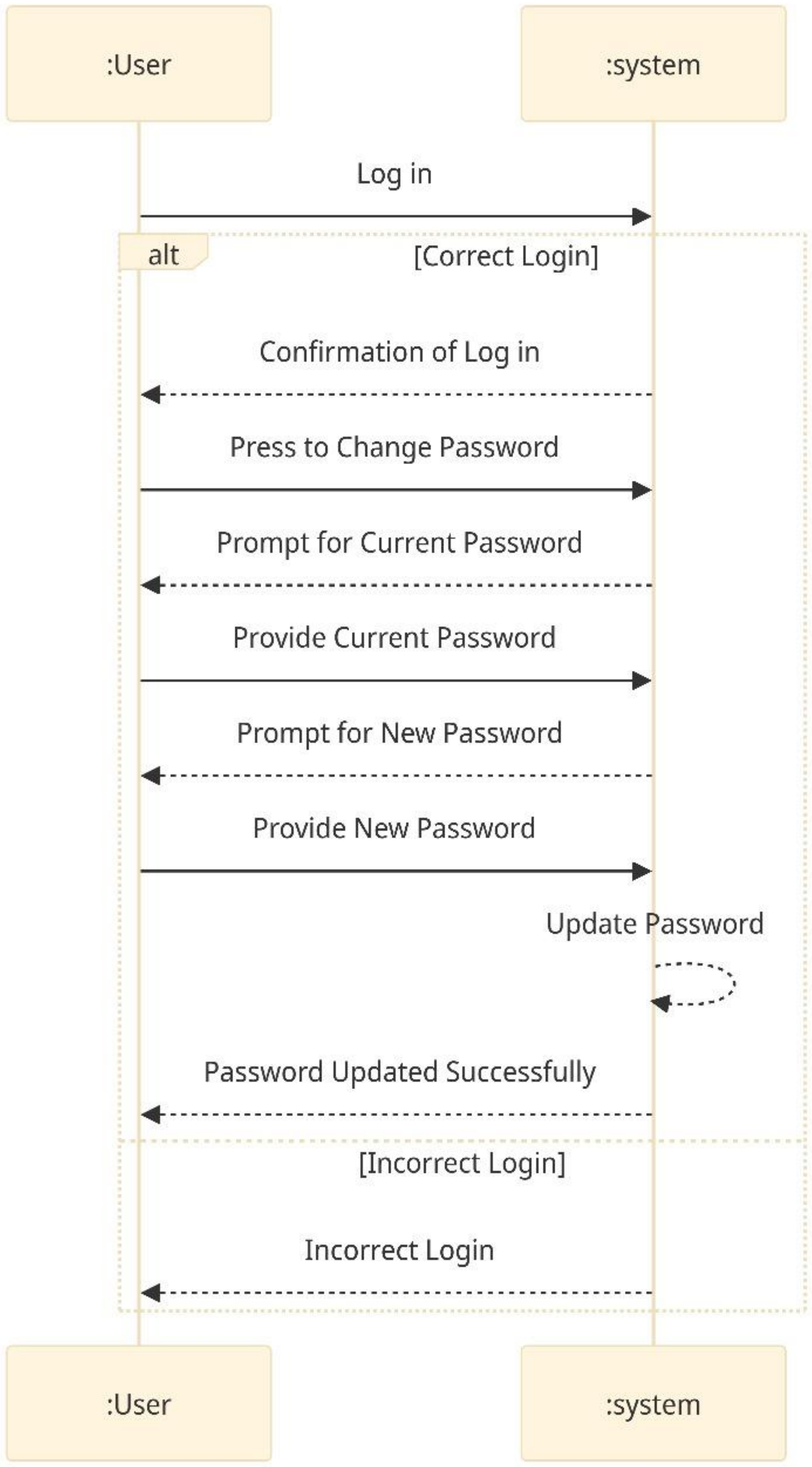

:User
:System
Request to View Past Articles
Retrieve User's Past Articles
Display Past Articles (First Page)
Scroll Down
Retrieve More Past Articles
Display More Past Articles (Next Page)
:User
:System


:Traveler
:System
Log in
Confirmation of Log in
Display Main Page
Press Button
Display Search bar
Search for Homestays
Display Homestay Search Results
:Traveler
:System

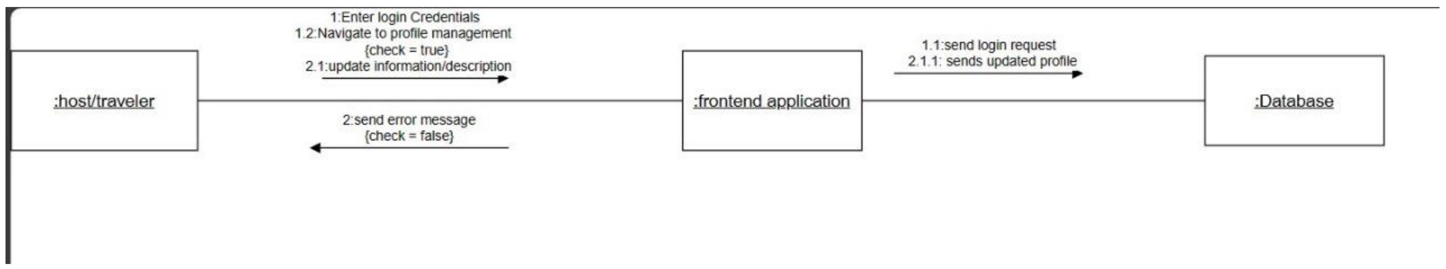

:User
:System
Press Help Tab
show Help Page
Write Question & Press Send Button
Receive Help Request
alt
[Request Valid]
Analyze Request
Request Acknowledged
Assign Task to Available Agent
Agent Assigned
Resolve User Issue
Issue Resolved
[Request Invalid]
Invalid Request, Please Retry
:User
:System

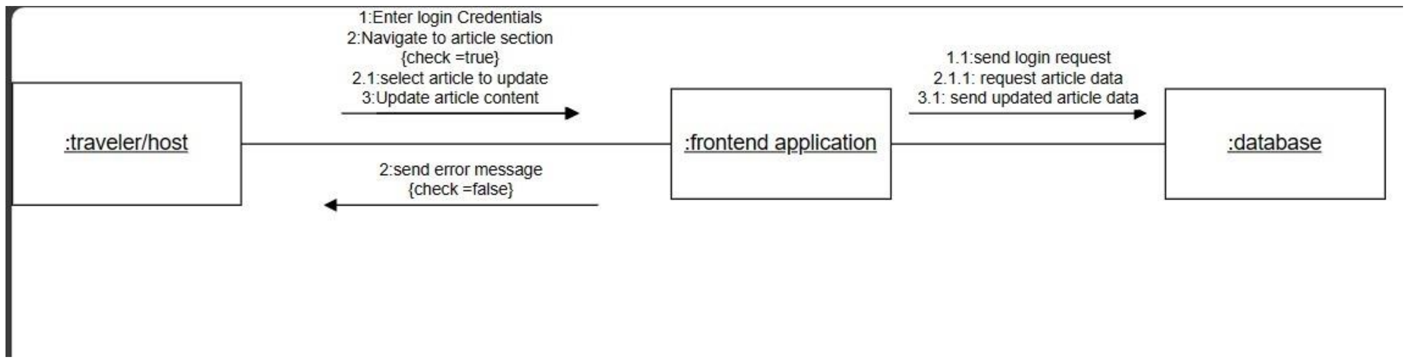

:User
:System
Log in
alt
[Logged in Successfully]
Confirmation of Log in
Check Availability Button
Availability Checked
alt
[Available]
Request Homestay and Agree on Terms
Waiting for Confirmation/Rejection (48 hours)
alt
[Confirmation Received]
Access Resources
Provide Access to Resources
[Rejection Received]
Respond to Rejection
[Not Available]
No Homestays Available
[Authentication Failed]
Handle Authentication Failure
:User
:System

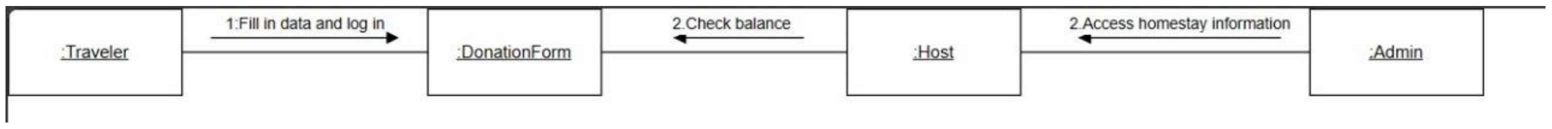

:User
:System
Log in
Confirmation of Log in
Initiate Payment for Yearly Membership Fee
Request Payment Details
Enter Payment Details
Membership Fee Payment Confirmed
:User
:System

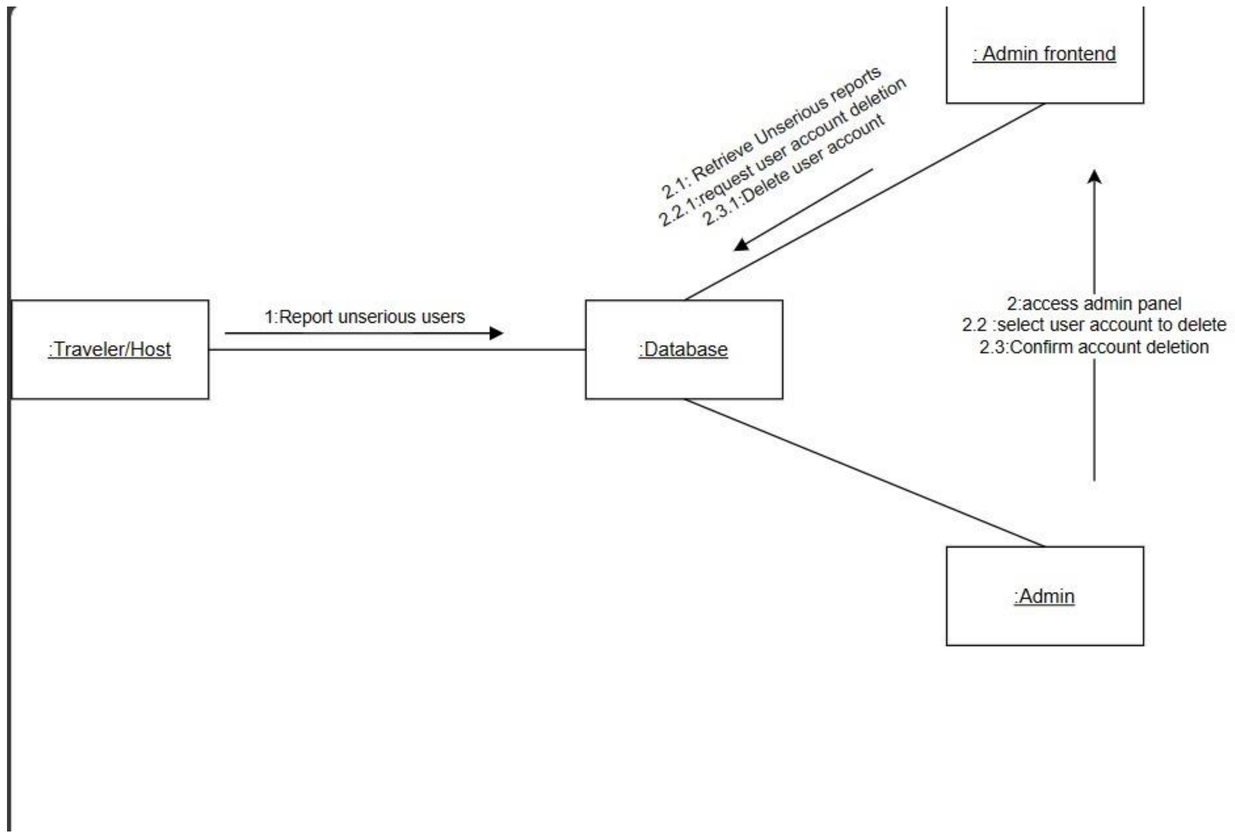

:User
:system
Log in
alt
[Correct Login]
Confirmation of Log in
Press to Change Password
Prompt for Current Password
Provide Current Password
Prompt for New Password
Provide New Password
Update Password
Password Updated Successfully
[Incorrect Login]
Incorrect Login
:User
:system

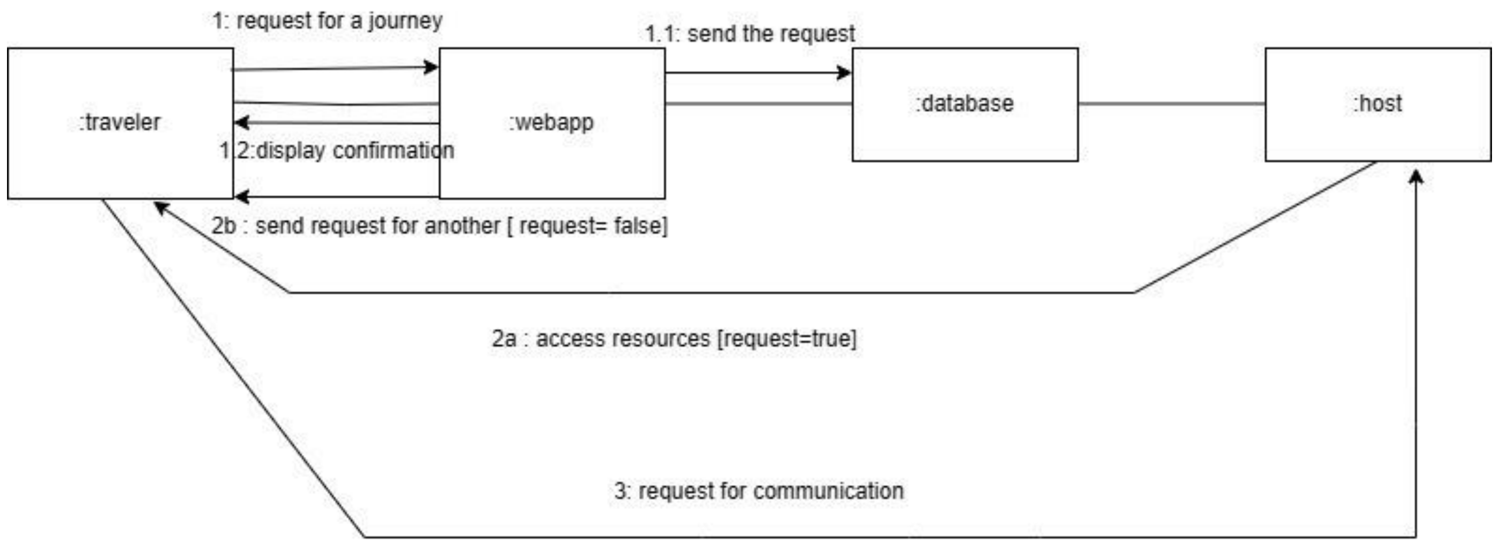
g) Collaboration/Communication Diagram(s) (including all the messages mentioned in the sequence diagrams.)

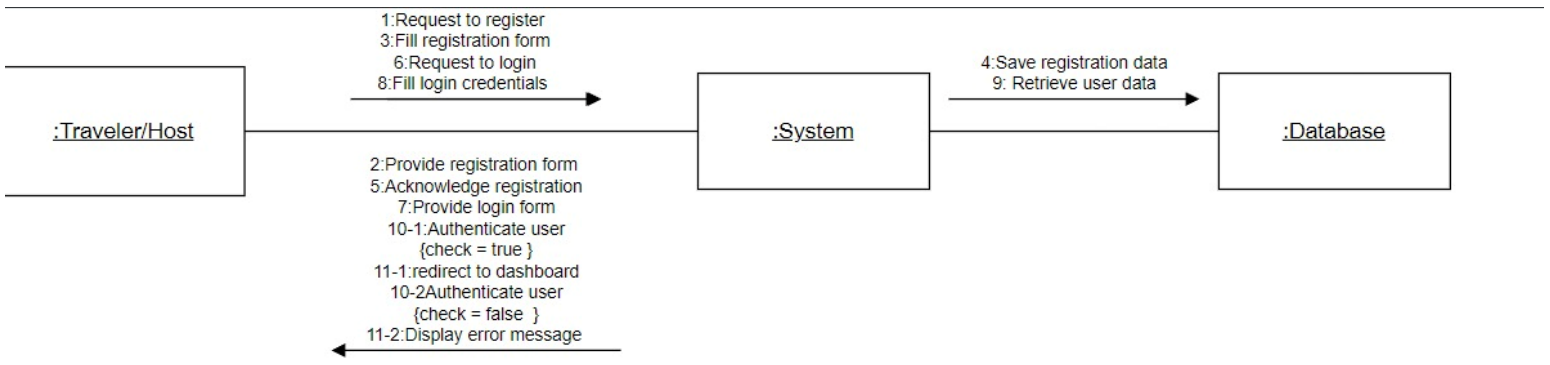

1:Enter login Credentials
1.2:Navigate to profile management
{check = true}
2.1:update information/description
:host/traveler
2:send error message
{check = false}
:frontend application
1.1:send login request
2.1.1: sends updated profile
:Database

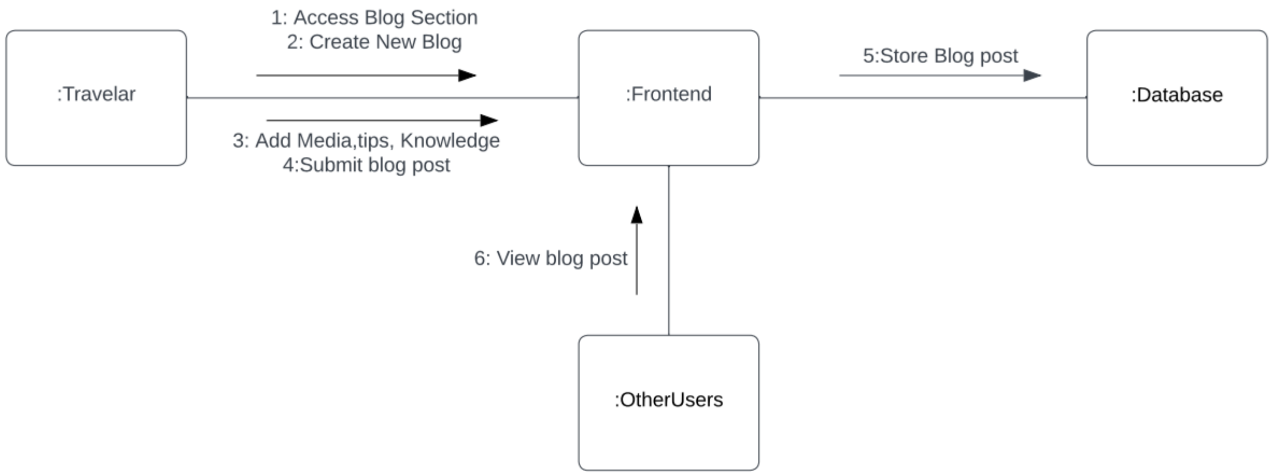

1:Enter login Credentials
2:Navigate to article section
{check =true}
2.1:select article to update
3:Update article content
:traveler/host
2:send error message
{check =false}
:frontend application
1.1:send login request
2.1.1: request article data
3.1: send updated article data
:database

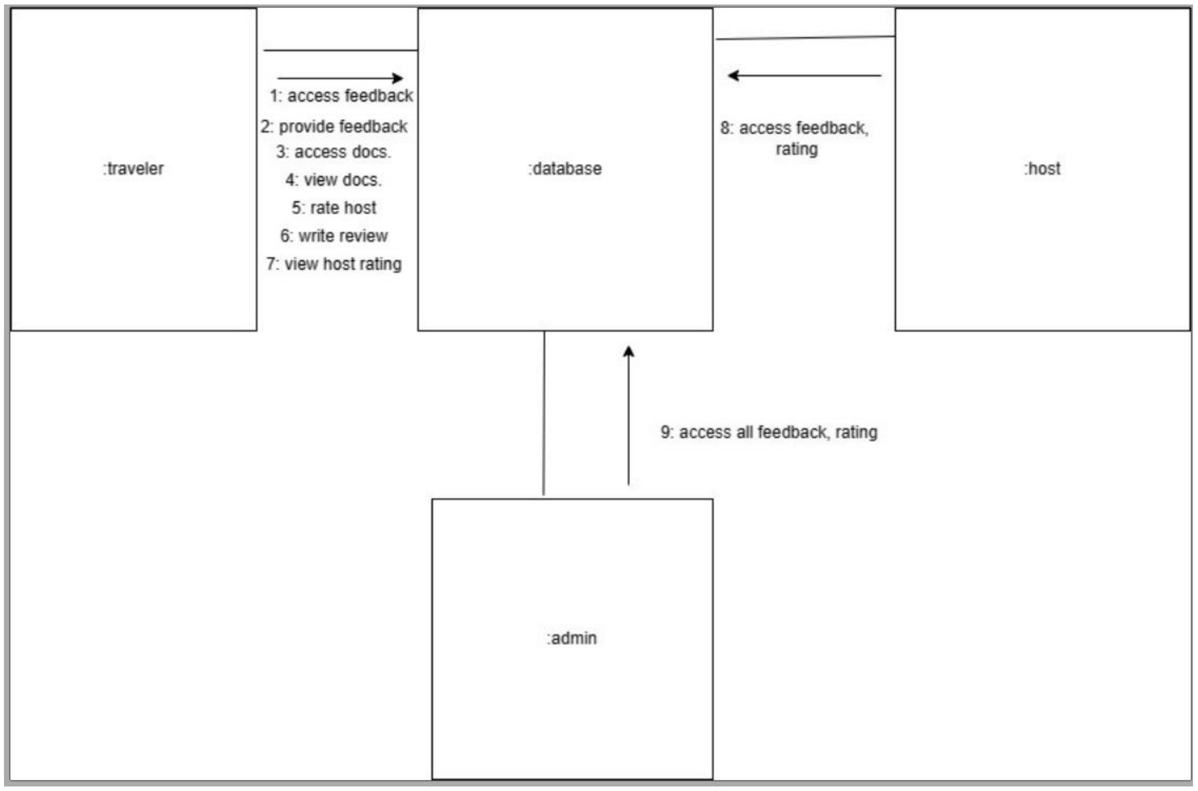


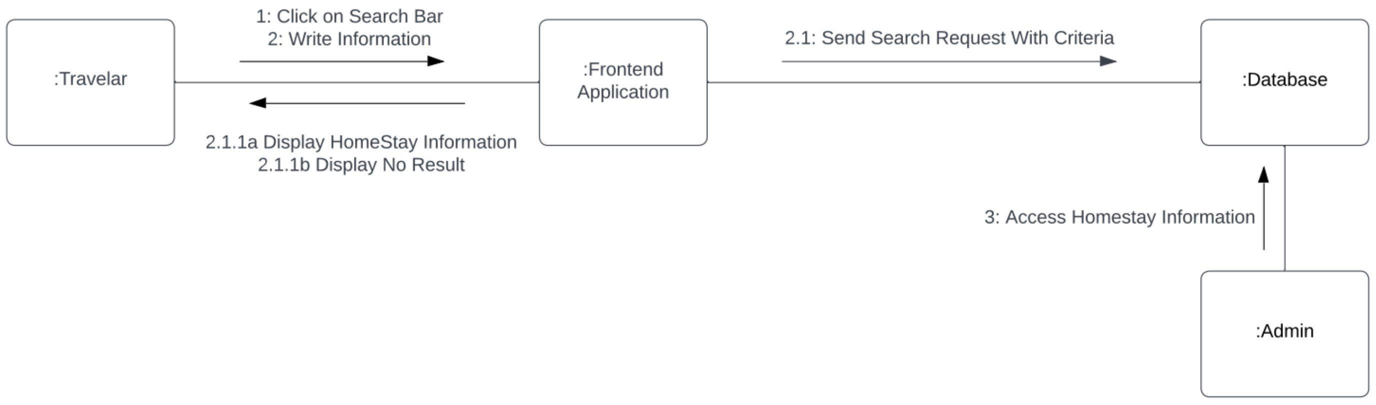

: Admin frontend
2.1: Retrieve Unserious reports
2.2.1:request user account deletion
2.3.1:Delete user account
:Traveler/Host
1:Report unserious users
:Database
2:access admin panel
2.2 :select user account to delete
2.3:Confirm account deletion
:Admin

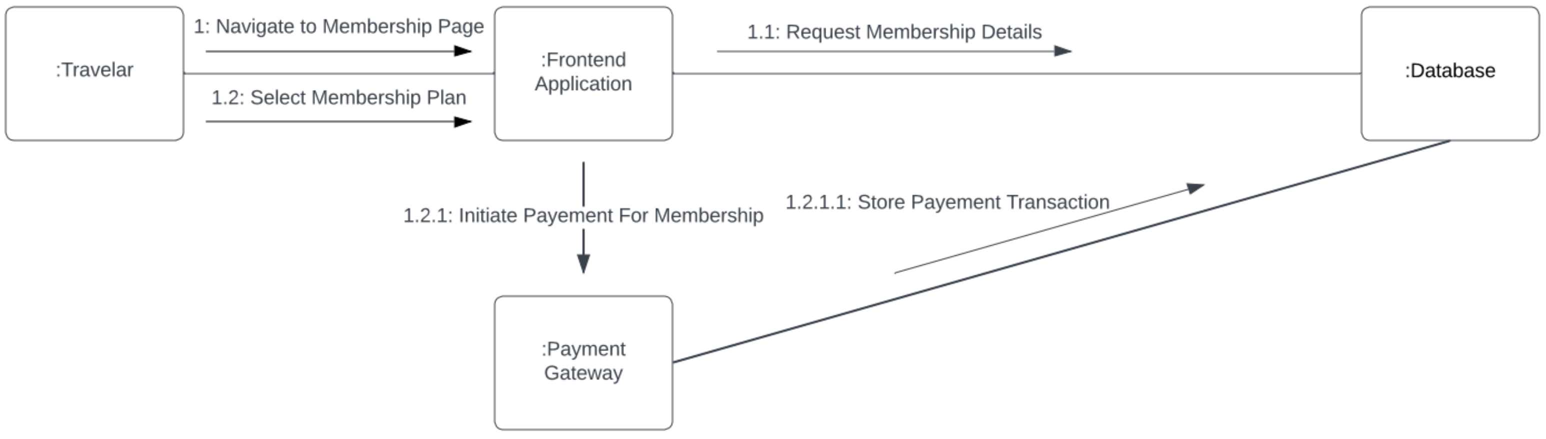

1: request for a journey
1.1: send the request
:traveler
:webapp
:database
:host
1.2:display confirmation
2b : send request for another [request= false]
2a : access resources [request=true]
3: request for communication

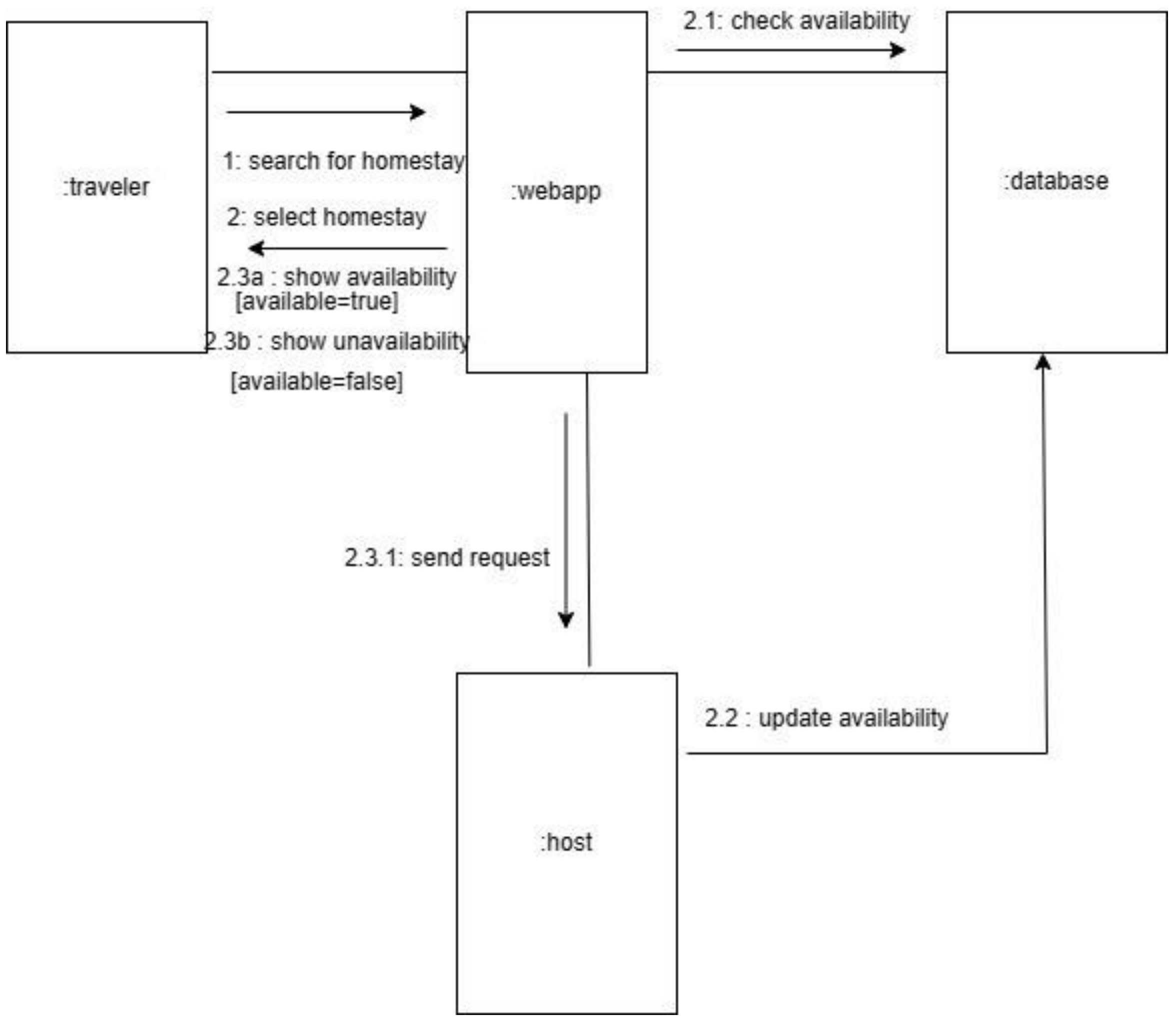

1:Request to register
3:Fill registration form
6:Request to login
8:Fill login credentials
4:Save registration data
9: Retrieve user data
:Traveler/Host
:System
:Database
2:Provide registration form
5:Acknowledge registration
7:Provide login form
10-1:Authenticate user
{check = true }
11-1:redirect to dashboard
10-2Authenticate user
{check = false }
11-2:Display error message


:Travelar
1: Access Blog Section
2: Create New Blog
3: Add Media,tips, Knowledge
4:Submit blog post
:Frontend
5:Store Blog post
:Database
6: View blog post
:OtherUsers


:traveler
1: access feedback
2: provide feedback
3: access docs.
4: view docs.
5: rate host
6: write review
7: view host rating
:database
8: access feedback, rating
:host
9: access all feedback, rating
:admin


:Travelar
1: Click on Search Bar
2: Write Information
:Frontend Application
2.1: Send Search Request With Criteria
:Database
2.1.1a Display HomeStay Information
2.1.1b Display No Result
3: Access Homestay Information
:Admin


:Travelar
1: Navigate to Membership Page
1.2: Select Membership Plan
:Frontend Application
1.1: Request Membership Details
:Database
1.2.1: Initiate Payement For Membership
1.2.1.1: Store Payement Transaction
:Payment Gateway


:traveler
:webapp
:database
:host
1: search for homestay
2: select homestay
2.1: check availability
2.2 : update availability
2.3a : show availability [available=true]
2.3b : show unavailability [available=false]
2.3.1: send request

h) Which strategy (or strategies) did you use to implement the use cases: One Central Class, Actor Class, or Use-Case class? Please explain your choice and the potential advantages/disadvantages of your design.

The strategy we use to implement the use case diagram is **Actor Class.**

We chose this use case strategy because each actor in the system is represented by a separate class that contains the use case logic related to the actor and that was the most suitable strategy for our system.

The actor class strategy is a design pattern used in OOP to model actors in a system. in this pattern actors communicate with each other by sending messages, which are processed asynchronously.

Advantages:

1. Improves modularity and encapsulation by separating use case logic based on actors.

2. Easier to understand and maintain as each actor class focuses on a specific set of use cases.

3. Actors can be easily distributed across multiple threads, processes, or even machines, allowing for efficient use of resources and improved scalability.

4. Actors can perform tasks concurrently without the need for explicit synchronization mechanisms such as locks or semaphores, simplifying the development of concurrent systems.

5. Errors that occur within an actor are isolated from that actor, preventing them from affecting other parts of the system. This makes it easier to build fault-tolerant applications.

6. 4. Actors encapsulate their own state and behavior, making it easier to reason about and maintain complex systems. This promotes a modular design that can be easily extended or modified.

7. Communication between actors is done through message passing, which promotes loose coupling and allows for flexible interactions between components.

Disadvantages:

1. Can lead to a proliferation of classes if there are many actors in the system.

2. Coordination between actor classes may be complex and require additional communication mechanisms.

3. Implementing actors can introduce some overhead due to the need to manage actor lifecycles, message queues, and scheduling. This can impact performance in some cases.

4. Designing systems using the actor model can be more complex than traditional approaches, especially for developers who are not familiar with the concept of actors and message passing.

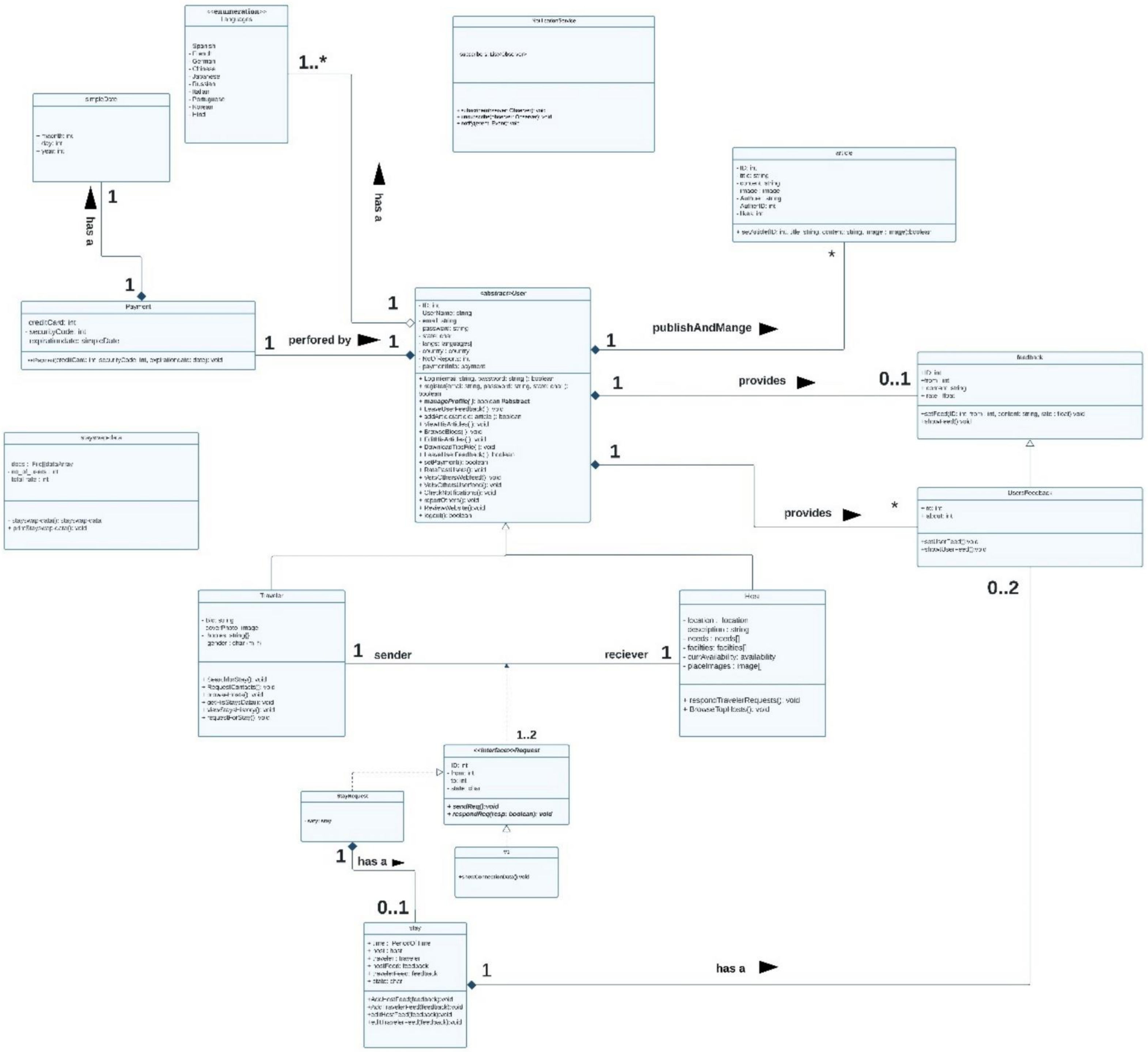
5. Debugging actor-based systems can be challenging, as errors may be distributed across multiple actors, and it is difficult to trace back to their source.

6. Asynchronous message passing can introduce latency in communication between actors, especially in distributed systems where messages need to be sent over a network.

7. Developers may need to learn new concepts and programming paradigms when working with the actor model, which can require additional time and effort.

i) Class Diagram 2: An intermediate version based on the interaction diagrams (including all classes, attributes, and operations mentioned in the interaction diagrams.)

VERSION 2



j) Three Design Patterns Applied (including the description for each design pattern, what problem it solves, and how it affects your system's design.)

In our system we have **4 design patterns**, and they are:

1: Singleton Pattern:

Context: is a design pattern that ensures a class has only one instance and provides a global point of access to that instance.

Problems: The main problem that the Singleton pattern aims to solve is to restrict the instantiation of a class to only one object, ensuring that all access to that object is through a single point.

Forces: 1. Ensure only one instance of a class exists.

2. Provide a global point of access to that instance.

3. Control access to a shared resource.

4. Avoid multiple instances causing conflicts or inconsistencies.

Solutions: 1. ensures that there is a single point of access to the instance of a class. In our system, for functionalities like managing user profiles, tracking history, or accessing resources, having a single instance can provide global access to these features across different parts of the system.

2. we can avoid creating multiple instances of classes that manage shared resources. For example, managing user sessions, handling notifications, or accessing travel guides materials can be efficiently handled by a Singleton instance to prevent resource conflicts.

3. helps maintain a consistent state across the system by ensuring that all components interact with the same instance of a class. This can be beneficial for functionalities like managing user preferences, reviewing hosts, or handling requests and responses consistently.

2: Observer Pattern:

Context: is a behavioral design pattern that establishes a one-to-many dependency between objects, so that when one object (the subject) changes its state, all dependent objects (observers) are notified and updated automatically.

Problems: The main problem that the Observer pattern aims to solve is the need for a flexible and decoupled way to notify multiple objects about changes in the state of a subject object.

Forces: 1. Allow multiple observers to be notified of state changes.

2. Maintain consistency between subject and observer objects.

3. Improve scalability and flexibility of the system.

Solutions: 1. We can establish a communication mechanism where users (observers) can subscribe to receive notifications or updates from specific components (subjects) in the system. This can be useful for notifying travelers and hosts about new requests, updates on listings, reviews, or any other relevant information.

2. we can enable real-time updates and notifications for users when there are changes or events that they need to be informed about. For example, hosts can receive notifications when a traveler requests to stay at their accommodation, or travelers can be notified when hosts respond to their requests.

3. supports a scalable and extensible notification system where new types of notifications or events can be easily added without affecting existing components. This flexibility is beneficial in a dynamic system like yours, where new features or functionalities may be introduced over time.

3: Model – View – Controller (MVC):

Context: is a widely used architectural design pattern in software development, it separates an application into three interconnected components: the Model (data and business logic), the View (presentation layer), and the Controller (handles user input and updates the model).

Problems: Without proper separation of concerns, changes in one component can affect others, leading to tightly coupled code that is difficult to maintain and extend.

Testing becomes challenging when components are tightly coupled, making it hard to isolate and test individual parts of the application.

forces: The need to separate data, presentation, and user interaction logic to improve maintainability, readability, and reusability of code.

Facilitating the ability to scale and evolve an application by making it easier to add new features, update existing ones, and adapt to changing requirements.

Solutions: 1. The MVC pattern helps in separating the application into three interconnected components - Model, View, and Controller. This separation ensures that each component has a specific role and responsibility, making the codebase more organized and maintainable. In your system, the Model represents the data (homestay listings, user profiles, reviews), the View represents the user interface (search results, profile pages), and the Controller handles the business logic (search filters, request handling).

2. By following the MVC pattern, you can create modular components that can be reused across different parts of the system. For example, the controller logic for handling user requests or updating user profiles can be reused in multiple views without duplicating code. This promotes code reusability and reduces development effort.

3. MVC separates the user interface (View) from the business logic (Model) and user input handling (Controller). This separation ensures that changes to the user interface do not affect the underlying data or application behavior. In your system, this separation can help in managing the various user interactions such as searching for homestays, requesting stays, reviewing hosts, and managing profiles.

4: Abstraction-Occurrence:

Context: a domain model you find a set of related objects “occurrences”; the members of such a set share common information but also differ from each other in important ways.

Problems: Without abstraction, developers may end up duplicating code to handle similar occurrences or patterns throughout the system, leading to redundancy and increased maintenance overhead.

Managing complexity in large systems without abstraction can hinder scalability and make it challenging to add new features or adapt to changing requirements.

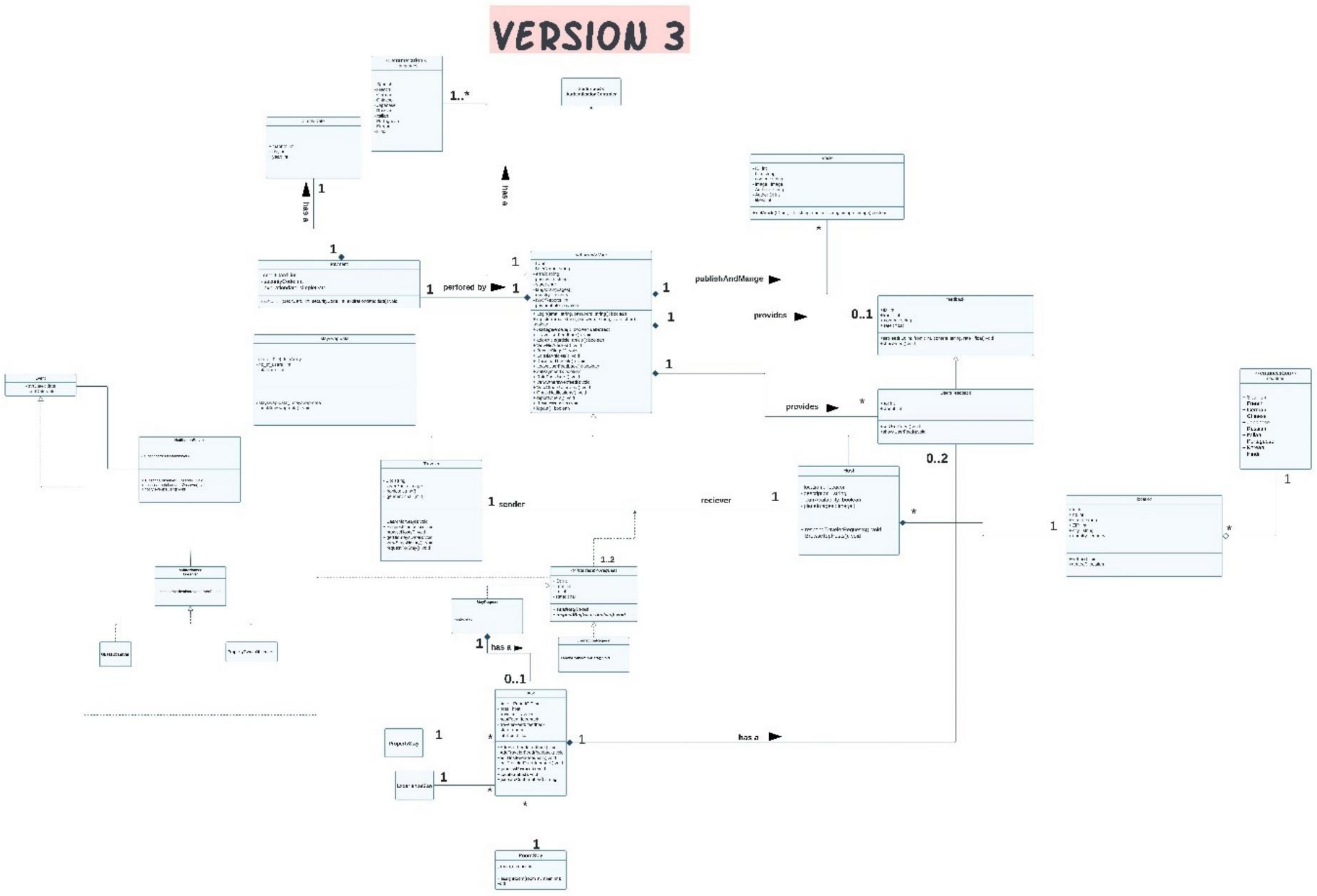
Forces: he needs to identify common patterns or occurrences in a system and create abstractions to reuse them across different parts of the application.

Encouraging a modular design that separates concerns and isolates components based on their functionality, promoting easier development, testing, and maintenance.

Solutions: 1. Consistency: By abstracting common functionalities, like profile management or notifications, into a shared pattern, it ensures that these features work similarly for both travelers and hosts, providing a consistent user experience.

2. Enhanced User Experience: By standardizing interactions and workflows through shared patterns, the abstraction-occurrence pattern enhances the overall user experience. Users can expect consistent behavior and familiar interfaces regardless of their role within the platform, leading to higher user satisfaction and engagement.

k) Class Diagram 3: The final version, after applying the design pattern(s) and any other modifications (the Class Diagram should include Associations, Self-associations (Recursive Associations), Multiplicities, Roles, Inheritance relationships, Polymorphism, Aggregation(s), and/or Composition(s), Qualified association(s), Association classes, and Interface class/es.)



l) [1 Bonus Mark] Define a design smell, highlight a design smell in your design, and suggest how it can be avoided.

A design smell refers to a characteristic or property of a software design that indicates a potential problem or inefficiency. One common design smell is "Feature Envy." Feature Envy occurs when a method in one class uses or manipulates more attributes or methods of another class than its own.

Highlighting a Design Smell in the ERD:

In the provided ERD, a potential design smell could be "Feature Envy" between the "articles" and "users" entities. The "articles" entity contains attributes such as "user_id" (foreign key referencing the "users" table) and "likesNO." Methods related to user-specific operations, such as retrieving user details or managing user authentication, might be implemented in the "articles" class, indicating Feature Envy.

Avoiding Feature Envy:

To avoid Feature Envy and improve the design, we can adhere to the principle of encapsulation and ensure that each class is responsible for its own behavior and data manipulation. Here are some strategies to avoid Feature Envy:

1. Encapsulate Behavior: Move methods related to user-specific operations to the "users" class instead of the "articles" class. For example, methods for retrieving user details, managing user authentication, or handling user preferences should belong to the "users" class.

2. Use Dependency Injection: Instead of directly accessing user-related functionality from the "articles" class, inject the "users" class or its dependencies into the "articles"

class as collaborators. This way, the "articles" class can delegate user-specific tasks to the "users" class without having Feature Envy.

3. Design Clean Interfaces: Define clear and concise interfaces for each class, specifying the interactions and responsibilities. By designing clean interfaces, it becomes easier to identify Feature Envy and refactor the design accordingly.

4. Apply SOLID Principles: Follow SOLID principles, such as the Single Responsibility Principle (SRP) and the Dependency Inversion Principle (DIP), to design classes that are focused, cohesive, and loosely coupled. This helps prevent Feature Envy and promotes maintainability and flexibility in the design.

By addressing Feature Envy and applying good design practices, we can improve the cohesion and maintainability of the software design.

m) [1 Bonus Mark] Read about Class Structuring Criteria and classify all the classes in your class diagram into one of the different categories of application classes. The general categories are an Entity Class, a Boundary Class, a Control Class, or an Application Logic Class. Some of those categories are further categorized.

Class Structuring Criteria refers to principles or guidelines used to classify classes based on their responsibilities and roles within a software application. Here's how we can classify the classes in the provided class diagram into different categories:

1. Entity Class:

- users: Represents a domain entity that stores information about users of the homestay web app.
- articles: Represents a domain entity that stores information about articles posted on the platform.
- blocked_users: Represents a domain entity that stores information about blocked users.
- contacts: Represents a domain entity that stores information about contacts between users.
- hosts: Represents a domain entity that stores information about hosts offering homestay accommodations.
- host_location: Represents a domain entity that stores information about the location of hosts' accommodations.
- payment_data: Represents a domain entity that stores payment-related data.
- requests: Represents a domain entity that stores information about requests made by travelers to hosts for stays.
- stays: Represents a domain entity that stores information about stays booked by travelers.

- subscribers: Represents a domain entity that stores information about subscribers to the platform.

- travelers: Represents a domain entity that stores information about travelers using the platform.

- users_feedbacks: Represents a domain entity that stores feedback provided by users.

- web_feedbacks: Represents a domain entity that stores feedback provided by users about the web platform itself.

2. Boundary Class:

- Presentation: Although not explicitly mentioned in the provided class diagram, classes related to the user interface and presentation layer would fall under this category.

3. Control Class:

- Application: Represents classes related to application logic, such as controllers and service classes. This category may include classes responsible for handling user requests, coordinating interactions between entities, and enforcing business rules.

4. Application Logic Class:

- Persistence: Represents classes related to data access and persistence, such as repositories and database entities. These classes are responsible for storing and retrieving data from the database.

Based on this classification, here's how the classes in the class diagram are categorized:

- Entity Classes: users, articles, blocked_users, contacts, hosts, host_location, payment_data, requests, stays, subscribers, travelers, users_feedbacks, web_feedbacks.
- Control Class: Application.
- Application Logic Class: Persistence.
- Boundary Class: Presentation (not explicitly shown in the provided class diagram).

It's important to note that the specific classification of classes may vary depending on the context of the application and its requirements.

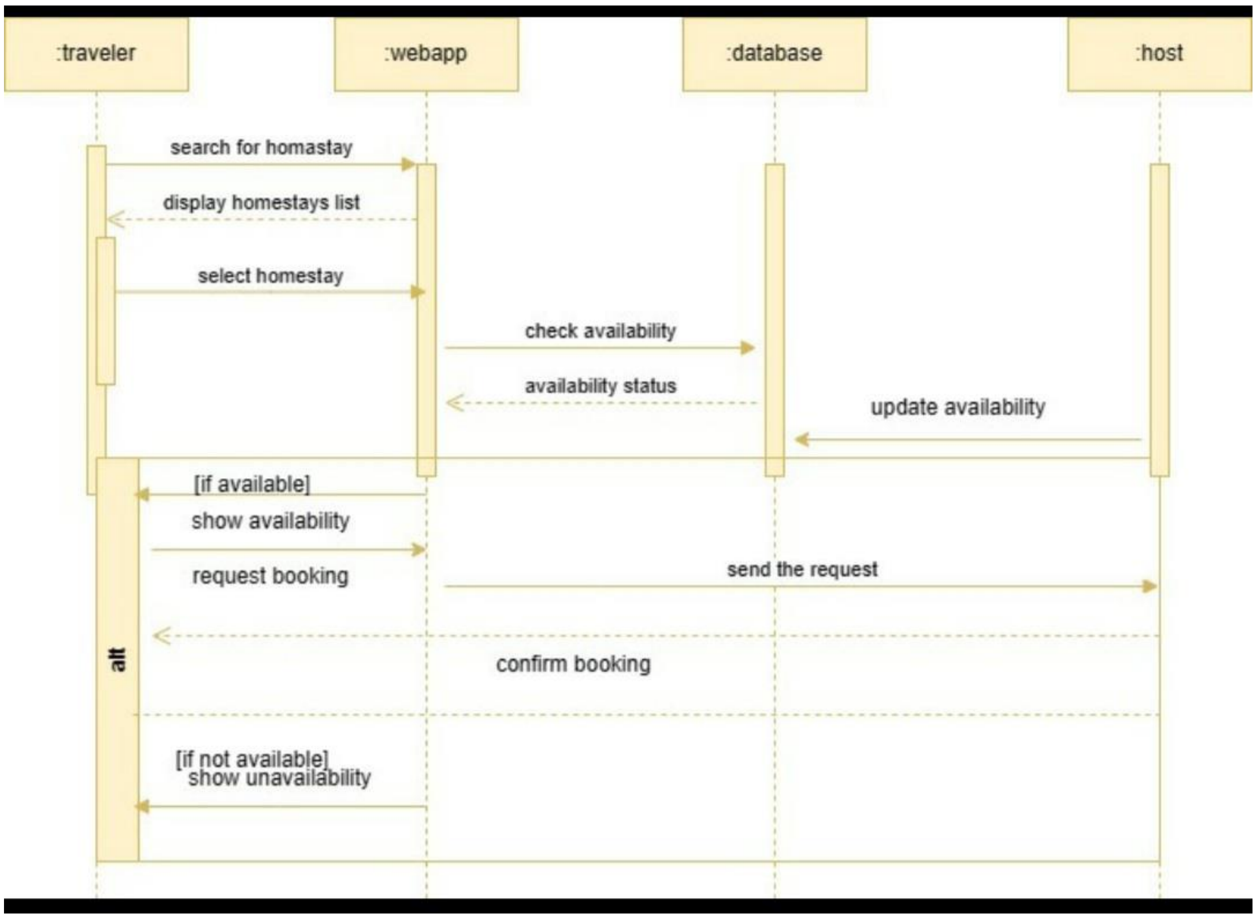
n) Did you rely on Forks or Cascades in your interaction diagrams? Give an example of your choice and mention its advantage(s)/disadvantage(s).

In our interaction diagram we depend on cascade as once a specific action is terminated the next action begins to happen (the flow of actions and scenarios is sequential)

Some examples show that we use cascade:

1. in the scenario of check availability first the traveler search for a specific host and the app displays a list of hosts to him then he selects one depending on his interests and preferences

This scenario is finished … after that the scenario of checking if the host chosen by traveler begins by checking if the host is available or not and if he is available the



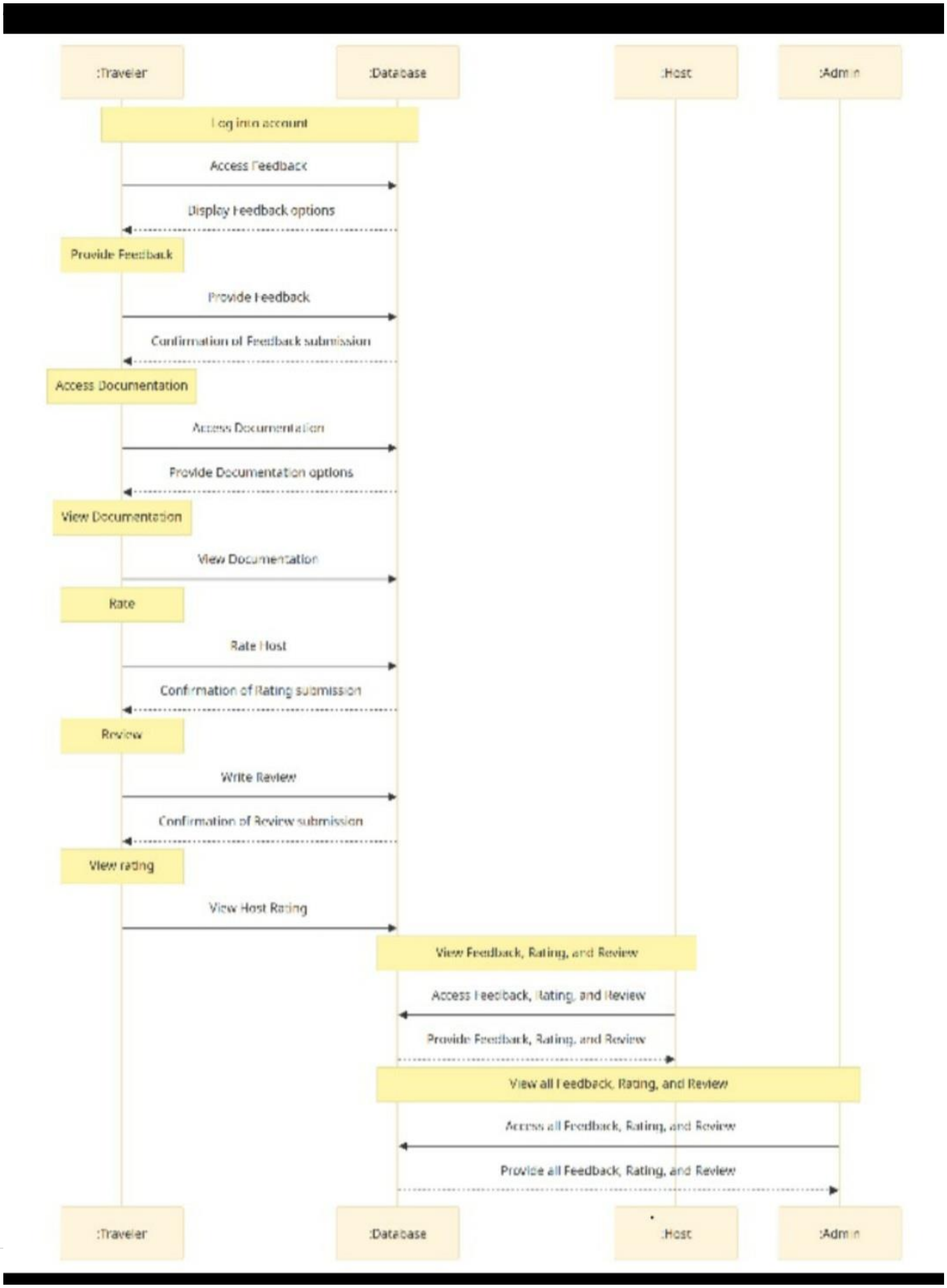
app will display a message show the availability and if not it will display a message that that host is not available

2.in the scenario of feedback and review the traveler access to feedback field and database displays options of feedback then the traveler provides feedback related to his stay

This scenario is finished … After that the scenario of docs. Begins by accessing docs. The traveler and database provide options of docs. To him then the traveler views the docs.

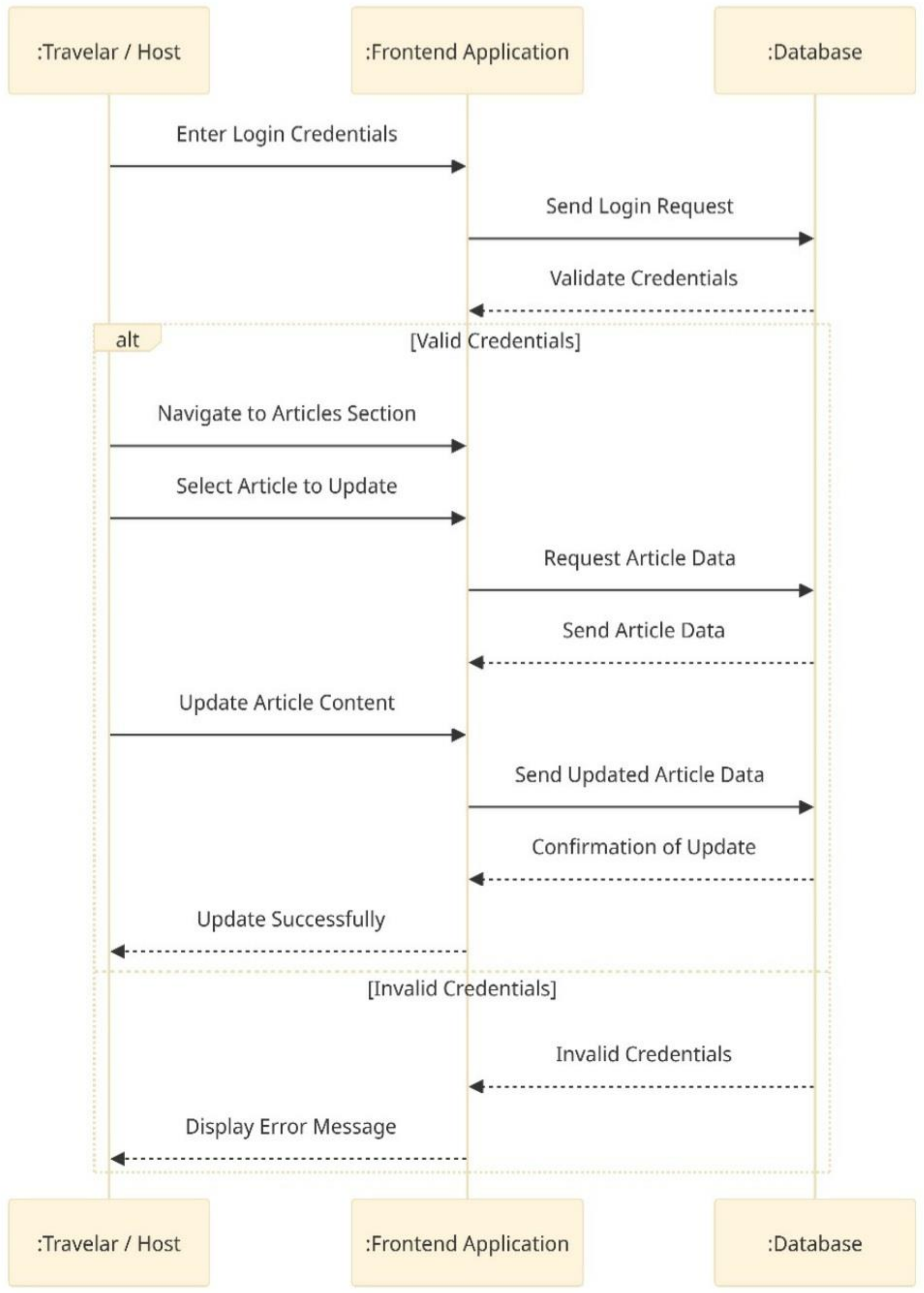
This scenario is finished … after that the traveler rates and write review about the host then he will view the rating of the host.

3. in the scenario of show articles first the user (traveler/host) enters the credentials such as: username and password then the request will be sent to be recorded in database then it will be validated by database


:Traveler
:Database
:Host
:Admin
Log into account
Access Feedback
Display Feedback options
Provide Feedback
Provide Feedback
Confirmation of Feedback submission
Access Documentation
Access Documentation
Provide Documentation options
View Documentation
View Documentation
Rate
Rate Host
Confirmation of Rating submission
Review
Write Review
Confirmation of Review submission
View rating
View Host Rating
View Feedback, Rating, and Review
Access Feedback, Rating, and Review
Provide Feedback, Rating, and Review
View all Feedback, Rating, and Review
Access all Feedback, Rating, and Review
Provide all Feedback, Rating, and Review
:Traveler
:Database
:Host
:Admin

selects article to update then the app will send a request to database informing it that the user wants to update an article then the database will send a the data of articles to be updated and the updated will be done successfully

[redacted]



[redacted]

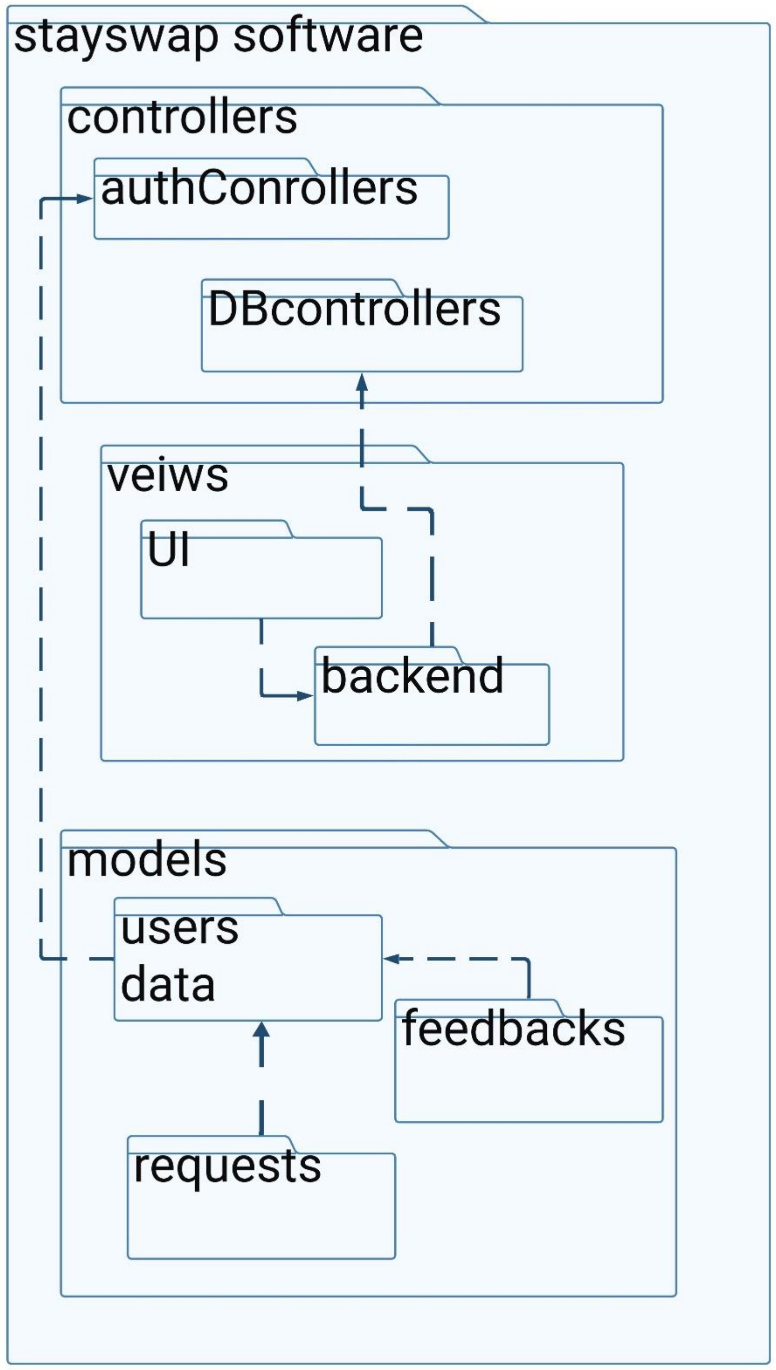
The Advantages:

1. Clear visualization of interactions: The cascade approach in interaction diagrams helps in visually representing the sequence of interactions between objects or components in a clear and structured manner.
2. Easy to follow: The linear flow of interactions in a cascade diagram makes it easier for stakeholders to understand the order of message exchanges and communication between different elements.
3. Helps in identifying dependencies: By showing interactions in a sequential manner, cascade diagrams can help in identifying dependencies between different components or objects.

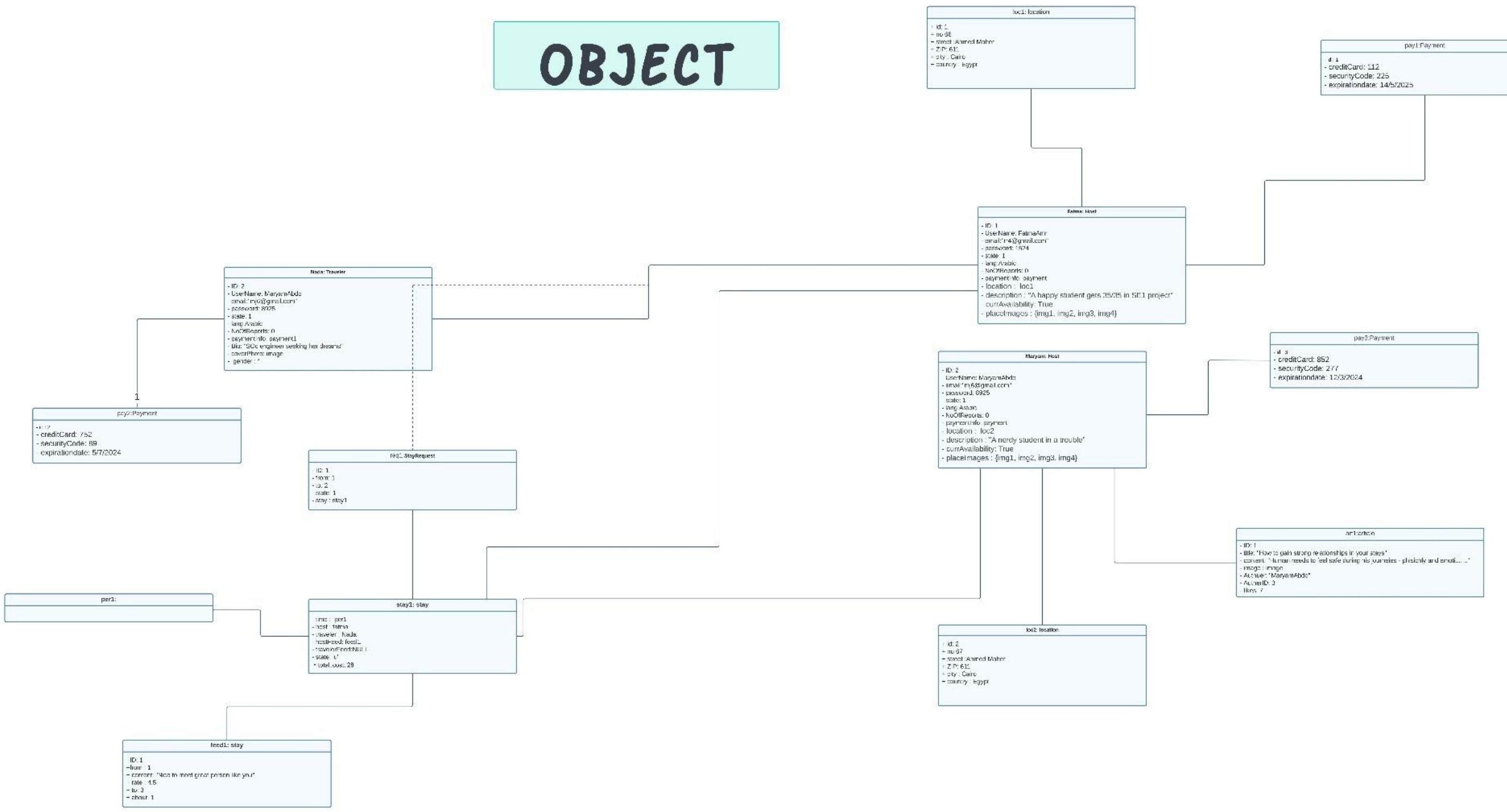
The Disadvantages:

1. Limited flexibility: The cascade approach may not be suitable for representing complex or dynamic interactions that involve multiple parallel or asynchronous processes.
2. Difficulty in representing real-time systems: For real-time systems or scenarios where timing is critical, the linear nature of cascade diagrams may not accurately represent the timing constraints.
3. limited scalability: As the number of interactions increases, cascade diagrams can become cluttered and difficult to read, making it challenging to represent large and complex systems.

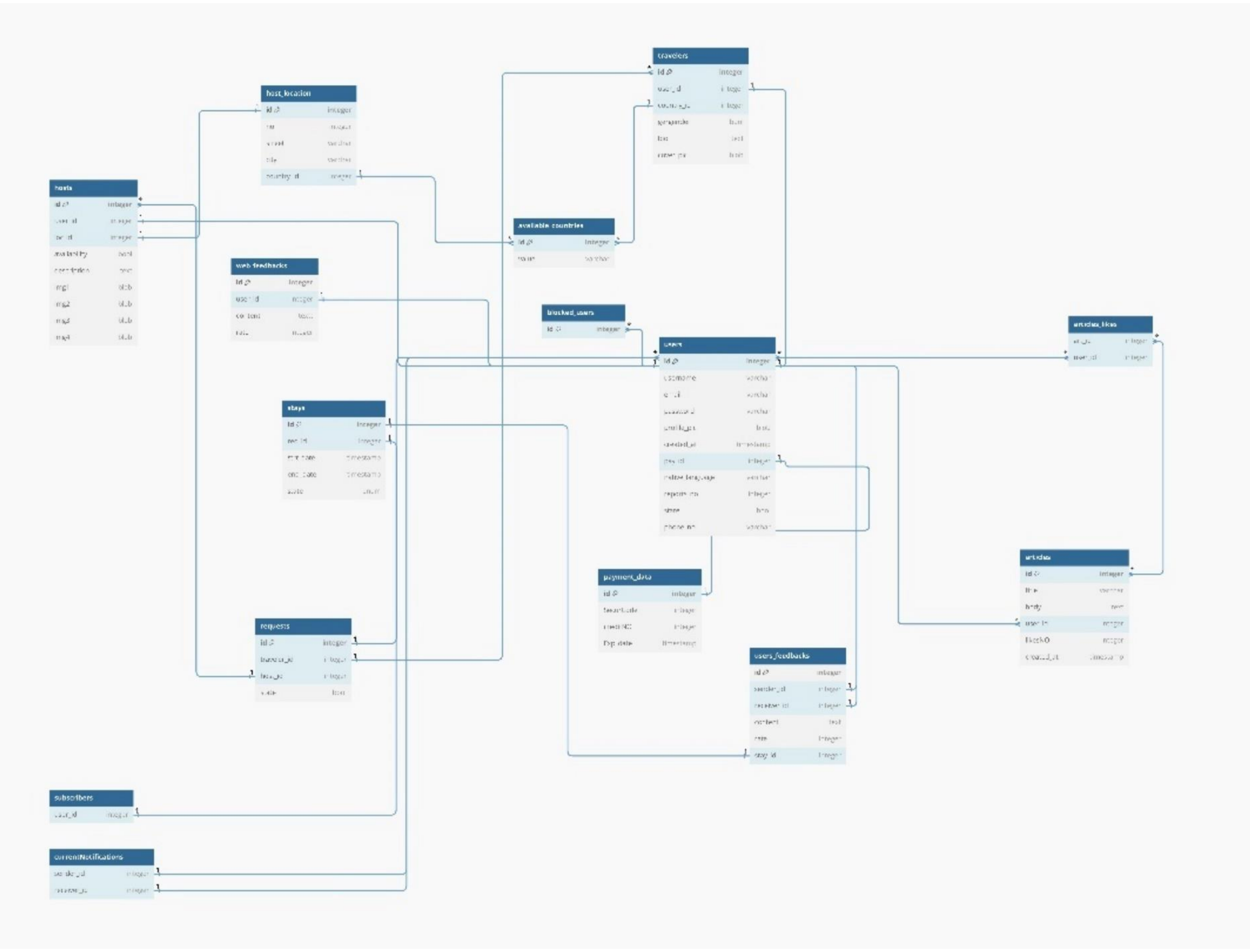
o) Package Diagram grouping relevant Classes into Packages.



p) Object Diagrams (including object diagrams that illustrate the preconditions and the postconditions for every major function.)



q) Database Specification (ERD, Tables.)



PART 4: Complexity & Testing

XII. Are there pairs of Software Quality Factors that are not independent in your system? Give an example.

Software Quality Factors in Homestays Platform Development

In the meticulous development process of our homestay’s platform, we conscientiously evaluate various software quality factors to ensure an unparalleled user experience and robust functionality. Among these factors, we recognize pairs that operate independently within our system:

1. Reliability & Scalability: Reliability denotes the consistency and error-free performance of the system, while scalability pertains to its ability to handle increased workload or data volume. These factors are considered independent as a system that can exhibit reliability without inherent scalability, and vice versa. For instance, a highly reliable system may lack provisions for scaling up to accommodate sudden surges in user activity, while a scalable system may encounter variations in reliability based on its implementation.

2. Usability & Maintainability: Usability refers to the ease with which users can interact with the system, while maintainability concerns the simplicity of maintaining and updating the system's codebase. These factors are deemed independent as a system can possess high usability without being inherently maintainable, and vice versa. For example, a system may boast a user-friendly interface but possess a complex and poorly documented codebase, presenting challenges in maintenance. Conversely, a system with a well-structured codebase may require users to navigate through intricate menus, impacting usability.

3. Performance Efficiency & Testability: Performance efficiency assesses how effectively a system utilizes resources, while testability focuses on the ease of testing the system for correctness and reliability. These factors are regarded as independent as a system that can demonstrate high performance efficiency without being inherently testable, and vice versa. For instance, a system may efficiently utilize resources but lack adequate logging and debugging mechanisms, complicating testing, and issue diagnosis. Conversely, a system may exhibit high testability but inefficiency in resource utilization due to suboptimal optimization.

By recognizing and addressing the independence between these pairs of software quality factors, we adopt a comprehensive approach to software development. This approach ensures that each aspect - be it reliability and scalability, usability and maintainability, or performance efficiency and testability - receives focused attention, resulting in a homestays platform that excels in all facets of quality and user experience.

4) Design & Implementation Constraints.

5) System Evolution:

a) Anticipated changes.

b) How should any anticipated changes in the future (due to hardware evolution,

changing user needs, and so on...) affect the system design?

6) What are the requirements discovery approaches that you’ll rely on? (Give detailed examples.)

7) What requirements validation techniques will you employ/use? (Give detailed examples.)

Constraints:

- Limited development timeframe and resources.
- Budget constraints for software development and hosting.
- Technical limitations of the chosen development framework and tools.
- Compliance with data privacy regulations and security standards.

Assumptions:

- Users and hosts will provide accurate and reliable information during registration and profile creation.
- The prototype will be tested with a small group of users for initial feedback and validation.
- Basic functionalities, such as user authentication and booking management, will be prioritized over advanced features for the prototype.