



BILKENT UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Senior Design Project

Petrium

Final Report

6 May 2022

Supervisor:

Prof. Dr. Özcan Öztürk

Jury Members:

Erhan Dolak & Tagmac Topal

Project Members:

Osman Buğra Aydın – 21704100

Oğuzhan Angın – 21501910

Mehmet Alperen Yalçın – 21502273

Ertuğrul Aktaş – 21802801

Muhammed Doğançan Yılmazoğlu – 21801804

Contents

1. Introduction	4
2. Requirement Details	5
2.1 Functional Requirements	5
2.1 Nonfunctional Requirements	6
2.1.1 Usability	6
2.1.2 Supportability	7
2.1.3 Efficiency	7
3.1.4 Scalability	7
3.1.5 Reliability	7
3.1.6 Security	7
3. Final Architecture and Design Details	8
3.1 Overview	8
3.2 System Models	9
3.2.1 Use Case Model	9
3.2.2 Scenarios	9
3.2.3 Object and Class Model	15
3.2.4 Dynamic Models	16
3.3 Subsystem Decomposition	20
3.4 Hardware/Software Mapping	21
3.5 Persistent Data Management	21
3.6 Access Control and Security	22
3.7 Global Software Control	22
3.8 Boundary Conditions	23
3.8.1 Initialization	23
3.8.2 Termination	23
3.8.3 Failure	23
4. Development/Implementation Details	24
4.1 Front-End	24
4.2 Back-End	25
4.2 Persistent Data Management	26
5. Testing Details	26
5.1 Version Checking	26

5.2	API Testing.....	27
6.	Maintenance Plan and Details.....	27
6.1	Back-End Server Maintenance.....	27
6.2	Persistent Data Management Maintenance.....	28
6.3	Front-End Maintenance and Optimization.....	28
7.	Other Project Elements	28
7.1	Consideration of Various Factors in Engineering Design	28
7.2	Ethics and Professional Responsibilities	31
7.2.1	Ethical Responsibilities	31
7.2.1	Professional Responsibilities	31
7.3	Judgements and Impacts to Various Contexts	33
7.4	Teamwork Details	34
7.4.1	Contributing and functioning effectively on the team	34
7.4.2	Helping creating a collaborative and inclusive environment.....	34
7.4.3	Taking lead role and sharing leadership on the team	35
7.4.4	Meeting objectives	35
7.5	New Knowledge Acquired and Applied.....	36
8.	Conclusion and Future Work	37
8.1	Conclusion.....	37
8.2	Future Work	37
9.	User Manual	39
9.1	Sign in and Sign up.....	39
9.2	Home Page.....	41
9.3	User Profile	42
9.4	Host Application.....	43
9.5	Hire a Host	45
9.6	Contracts	47
9.7	Messaging.....	48
10.	Glossary.....	51
11.	References.....	52

1. Introduction

Petriam is a mobile application to serve the people with pets while they are unable to take care of their pets during their time outside their homes. The project aims to favor both sides of its target group which are hosts and clients. Using Petriam, clients who are pet owners could list the possible temporary homes for their pets while hosts who may or may not be pet owners can earn money by taking care of the pets of their clients.

Petriam will be focusing on the specific service of pet accommodations for the time being to accurately deliver a solution to the problem. In the application, a client could reserve a date for their pet, get in contact with the host, ask their questions and rate them at the end of their stay. On the other hand, hosts could accept or reject a reservation depending on the client and describe their services to the clients. In simpler words, Petriam will be a free marketplace for people who are looking for hosts and people who want to earn money by taking care of pets. Surely, the host registration process will have extra verification steps than the user registration process in order to create a trusting environment in the application. What makes Petriam an investment worthy application is that Petriam is only the middleman in this process. While correctly solving the issue, Petriam will not own any pets or pet homes to minimize expenditures. The main difference between Petriam and other pet care applications is that Petriam is the only mobile application that solely addresses this issue.

Moreover, by not relying on pet hotels, less waste is created by using Petriam since Petriam favors the principle of sharing rather than buying new equipment for pets. Pet hotels are expensive to maintain in terms of resources such as electricity, water and gas which are used in much higher amounts compared to a regular house [1]. Thanks to Petriam, leaving pets during travels will be cheaper, environmentally safer and economically sustainable.

2. Requirement Details

2.1 Functional Requirements

Requirements Related to the Pet Owner Users

- Pet owner users need to create an account and register to the system with a username or email and a password.
- Pet owner users need to sign in to the system with their account to use the application.
- To be able to hire a pet host, a pet owner-user needs to define at least one pet to their account.
- Pet owner users should be able to define multiple pets to their accounts.
- Pet owner users should accept the permission request related to GPS location.
- Pet owner users should be able to other host users around them with a map view that has pins on it as long as they accepted the GPS location permission.
- Pet owner users also should be able to host users that are close to them as a list of hosts with only necessary information such as rating or price.
- Pet owner users should be able to filter the list of hosts according to parameters such as city, price, rating, accepted pet type, and verification.

Requirements Related to the Pet Host Users

- Every type of user should be able to become a pet host user by registering as a pet host in the system.
- To become a pet host user, a user should be registered and signed up to the system.
- To become a pet host user, a user does not need to define any pet to their account.

- To become a pet host user, a user should accept the permission request related to GPS location.
- A user needs to provide additional information to their accounts such as social security number, phone number, price, and criminal record.
- A user needs to be verified in or to be able to become a pet host user.
- A user should be visible in the host list view and the map view after they registered as a pet host user.

General Requirements

- Any type of user should not be able to view any type of users' sensitive information such as phone number or social security number.
- Messages between pet owner users and pet host users need to be stored in a database system to constitute evidence for illegal or unwanted behavior in the future.
- Payment for the hosting should be completed among the host and owner in real life.
- Petriam should be able to locate the user and display their surroundings by using map API.

2.1 Nonfunctional Requirements

2.1.1 Usability

- The user interface of the system should be user-friendly, understandable and simple
- Users should have no confusion about how the system works
- Users should be able to easily register and sign in to the system

2.1.2 Supportability

- The system should be able to work on both IOS and Android operating systems.
- The system should be able to support the use of any APIs such as Google Maps.

2.1.3 Efficiency

- The Delay between user requests and the system providing them should not exceed 5 seconds
- Log in process should not last more than 5 seconds.

3.1.4 Scalability

- The number of pet owners and pet users registered to the system should not affect the overall performance and quality of the system. The system should be to support large numbers of users.
- The system should be to handle multiple host arrangements and messaging processes at the same time without any issues or conflicts.

3.1.5 Reliability

- The system should not crash ever during its runtime if possible. The number of crashes must be kept at a minimum and the time interval between these crashes should be long enough
- The system should be to roll-back its operations in case of failure or crash of the system or servers.

3.1.6 Security

- The system should store the user information in a database system securely by taking necessary precautions such as encryption.

- Any third-party person or organization should not be able to view any user's private and sensitive information.
- Pet owner users should be able to view a pet host's profile page and get in contact with the pet host by using the messaging functionality.
- Pet owner users should not be able to view sensitive information such as social security numbers or phone numbers related to any type of user whether pet host or pet owner user.
- Pet owner users should be able to rate a pet host user after both parties completed their hosting process.
- Any pet owner should not be visible in the host list view or map view as long as they are not also a pet host user.

3. Final Architecture and Design Details

3.1 Overview

The following sections discuss the internal workings of the system as in subsystem decomposition, hardware/software mapping, persistent data management, access control and security, global software control and boundary conditions. The aim of Petriam is to create a fast, safe and efficient environment for pet owners that the discussed elements are designed according to the mentioned purposed of the system.

3.2 System Models

3.2.1 Use Case Model

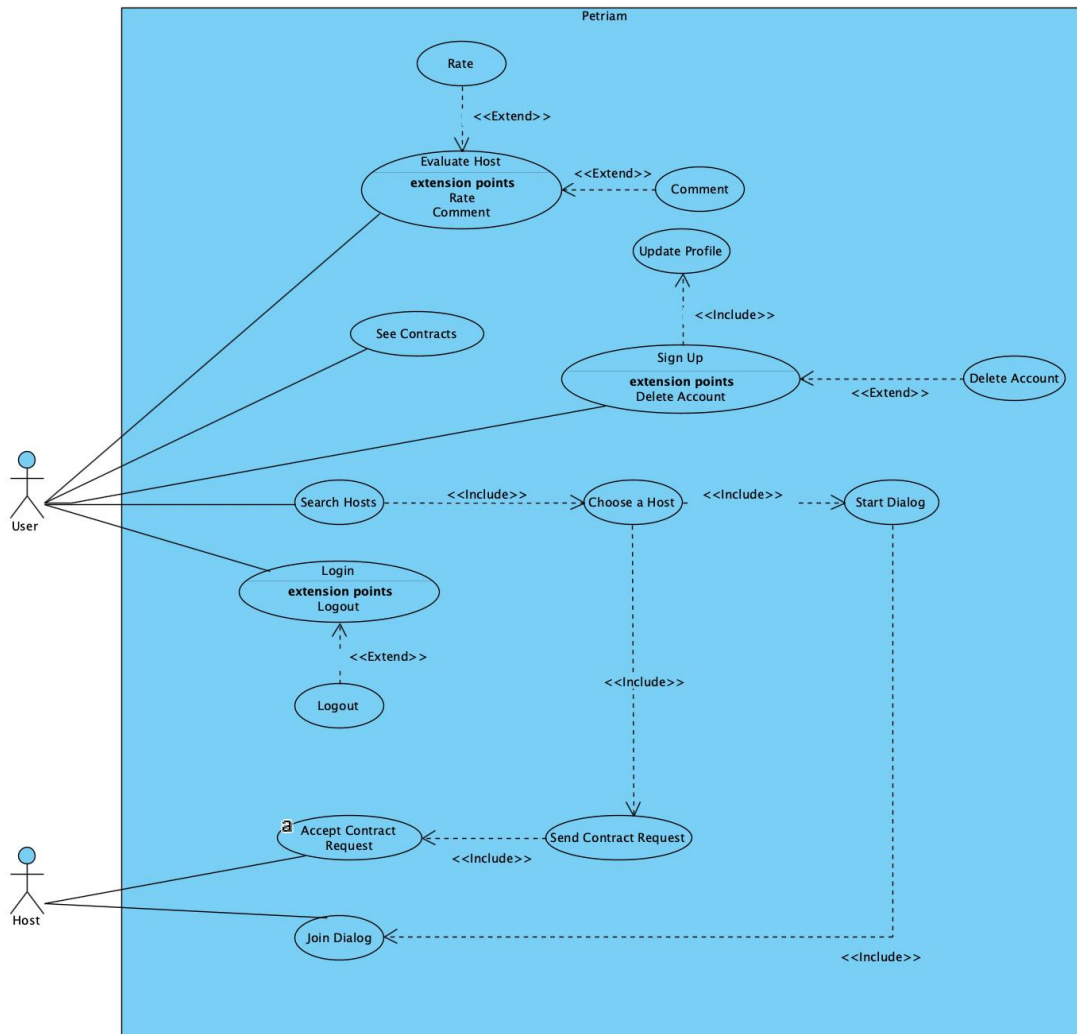


Figure 1: Use Case Model

3.2.2 Scenarios

Scenario 1 - Login

Actors: Pet Owner or Host

Entry Conditions:

- Users open the application.
- Users log out from their accounts.

Exit Conditions:

- Users are directed to the home page if their credentials are correct
- Users can close the application.
- Users can switch to the registration page.

Flow of Events:

1. Users enter their usernames and passwords.
2. Users click the “Sign In” button.
3. If the credentials are correct, the user is directed to the home page. Otherwise, there will be a sign that indicates credentials are incorrect.

Scenario 2 - Registration

Actors: Pet Owner or Host

Entry Conditions:

- Users on the “Sign In” page can click the “Sign up to Petriam” button and will be directed to the registration page.

Exit Conditions:

- Users fill the form with their credentials and will be directed to the home page.

Flow of Events:

1. Users provide necessary information for registration and click the “Sign Up” button.
2. If there are no existing users with given credentials or given information like username should not contain special characters, users will be registered, signed in and directed to the home page. If there is something wrong with the information, incorrect parts will be indicated.

Scenario 3 - Become a Host

Actors: Pet Owners

Entry Conditions:

- Users are signed in and click the hotel icon in the navigation bar.

Exit Conditions:

- Users provide the necessary information and submit the information.

Flow of Events:

1. Users sign in to their accounts.
2. Users click the hotel icon to become a host.
3. Users provide crucial information like identification numbers.
4. Users submit the form.
5. If provided information is valid, users can become a host. If it is not valid, users have to repeat the process from step 1.

Scenario 4 - Search a Host with Filters

Actors: Pet Owner or Host

Entry Conditions:

- Users need to be signed in and click on the search input box.

Exit Conditions:

- Users can select one of the listed hosts.
- Users click one of the sections in the navigation bar.

Flow of Events:

1. Users sign in to their accounts.
2. On the home page, users can click the search button on the upper side of the screen by writing a name.

3. Users can use advanced parameters like location, pet type, and more.
4. If users find an appropriate host, users can select the host.

Scenario 5 - Find a Host from Map

Actors: Pet Owner or Host

Entry Conditions:

- Users need to be logged in to their accounts.
- A logged-in user can click the home icon in the navigation bar.

Exit Conditions:

- Users can click one of the pinpoints which represents a host in the map.

Flow of Events:

1. Users need to be logged in.
2. Users can pinch the screen to zoom in or zoom out to find a host.
3. If a host is found, users can click on the pinpoint and select the host.

Scenario 6 - View Ratings

Actors: Pet Owner or Host

Entry Conditions:

- Users need to be logged in to their account and select a user in the way explained above. There will be an overall rating and a button to the direct reviews page.

Exit Conditions:

- Users can click the back icon in the right upper corner of the screen.

Flow of Events:

1. Users sign in.
2. Users select a host in two ways explained before.

3. After selecting the host, there will be a page with the information of the selected host. Users can click to view the ratings button.
4. All ratings will be listed with comments and extra information.

Scenario 7 - Send Contract

Actors: Pet Owner or Host

Entry Conditions:

- Users sign in and select a host and click the “Hire This Host” button.

Exit Conditions:

- The contract which contains information such as host, pet owner, pet and dates will be sent to the host.

Flow of Events:

1. Users log in to the system.
2. Users find and select the appropriate host according to their needs.
3. Users select the beginning and ending dates.
4. Users click the “Hire This Host” button and the contract will be sent.

Scenario 8 - Take a Pet as a Host

Actors: Host

Entry Conditions:

- Users need to be both signed in and registered to the system as a host. Then, a host can accept the contract.

Exit Conditions:

- Hosts can accept or reject the contract sent by the pet owners.

Flow of Events:

1. Users sign in to the application.

2. Users need to be registered as a host.
3. Hosts can check the contracts section in the navigation bar.
4. If there is a contract, the host can accept or reject the contract. If the host accepted the contract, contact information will be visible to both parties such as the host and pet owner.
5. A host can arrange a meeting with the pet owner and take the pet as a host.

Scenario 9 - Sending Messages

Actors: Pet Owner or Host

Entry Conditions:

- Pet owners find a host and send a message to the host.
- Users can open their inboxes and send messages to existing conversations.

Exit Conditions:

- Messages are sent successfully.

Flow of Events:

1. Users are logged in to their accounts.
2. Users can open the inbox and send messages to the people from who they have already sent or received messages. Otherwise, pet owners can find a host with the two methods explained.
3. After selecting the host, pet owners can send messages to the hosts for further clarifications.

Scenario 10 - Create a Review

Actors: Pet Owner

Entry Conditions:

- The Pet owner needs to be logged in to the system and a contract belonging to the pet owner needs to be successfully completed which means a pet is given to the host and is taken back from the host for the pet owner to create a review.

Exit Conditions:

- A review is successfully published.

Flow of Events:

1. Users need to be logged in.
2. Pet owners find an appropriate host.
3. Pet owners may send messages for further clarifications or directly send the contract to the host.
4. The host accepts the contract.
5. The pet is given to the host.
6. The host takes care of the pet between determined dates and gives back the pet to its owner.
7. The contract should be approved by two parties to be completed.
8. After the status of the contract is completed, the pet owner can create a review.

3.2.3 Object and Class Model

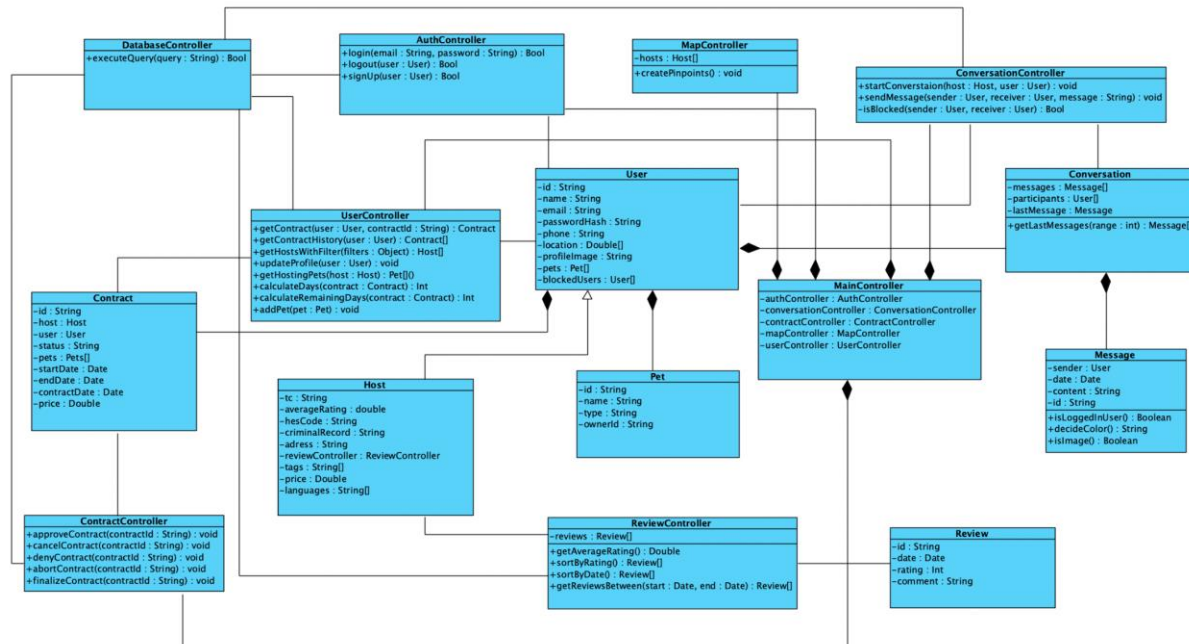


Figure 2: Object and Class Model

3.2.4 Dynamic Models

3.2.4.1 Activity Diagram

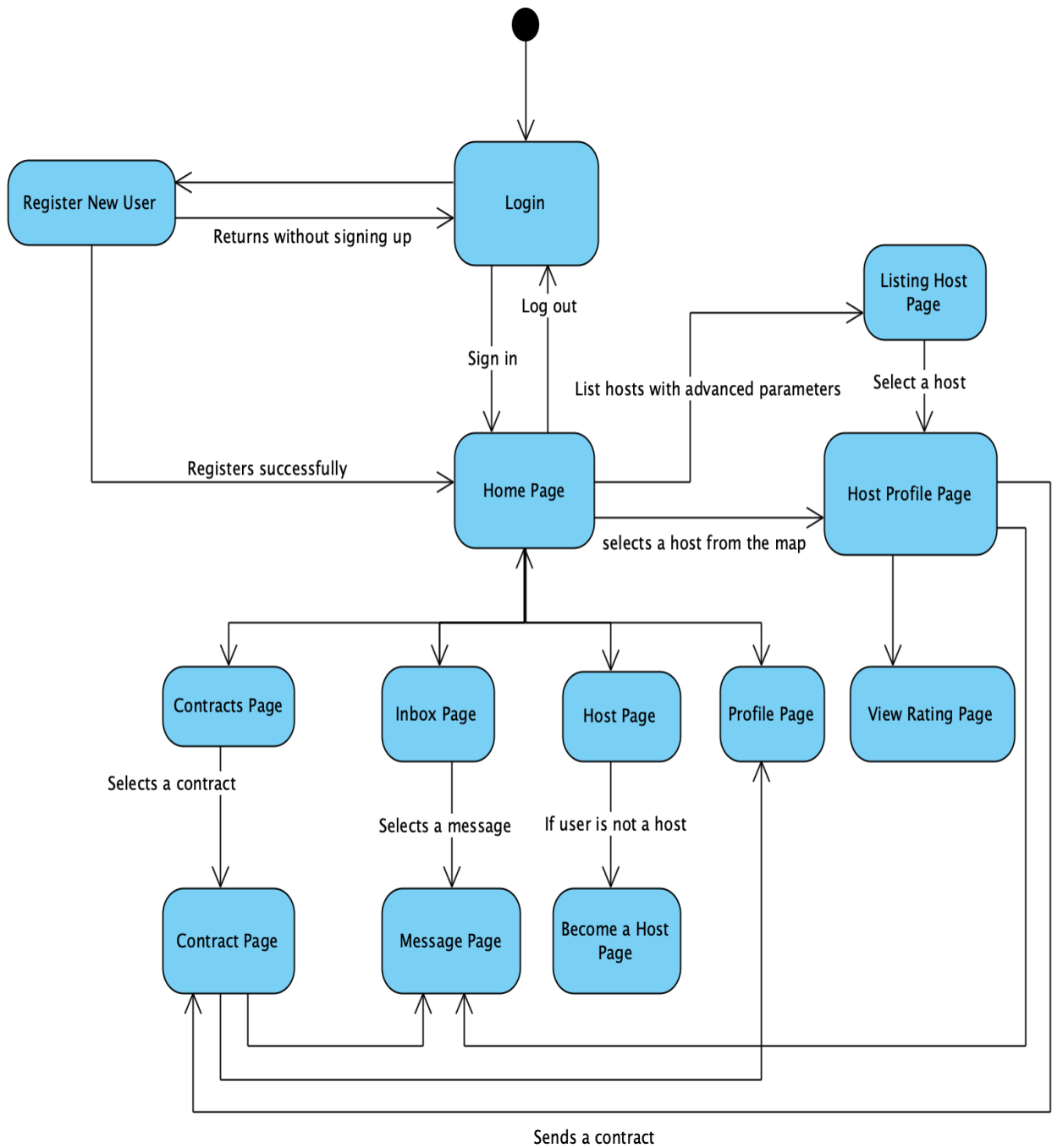


Figure 3: Activity Diagram

3.2.4.2 State Diagrams

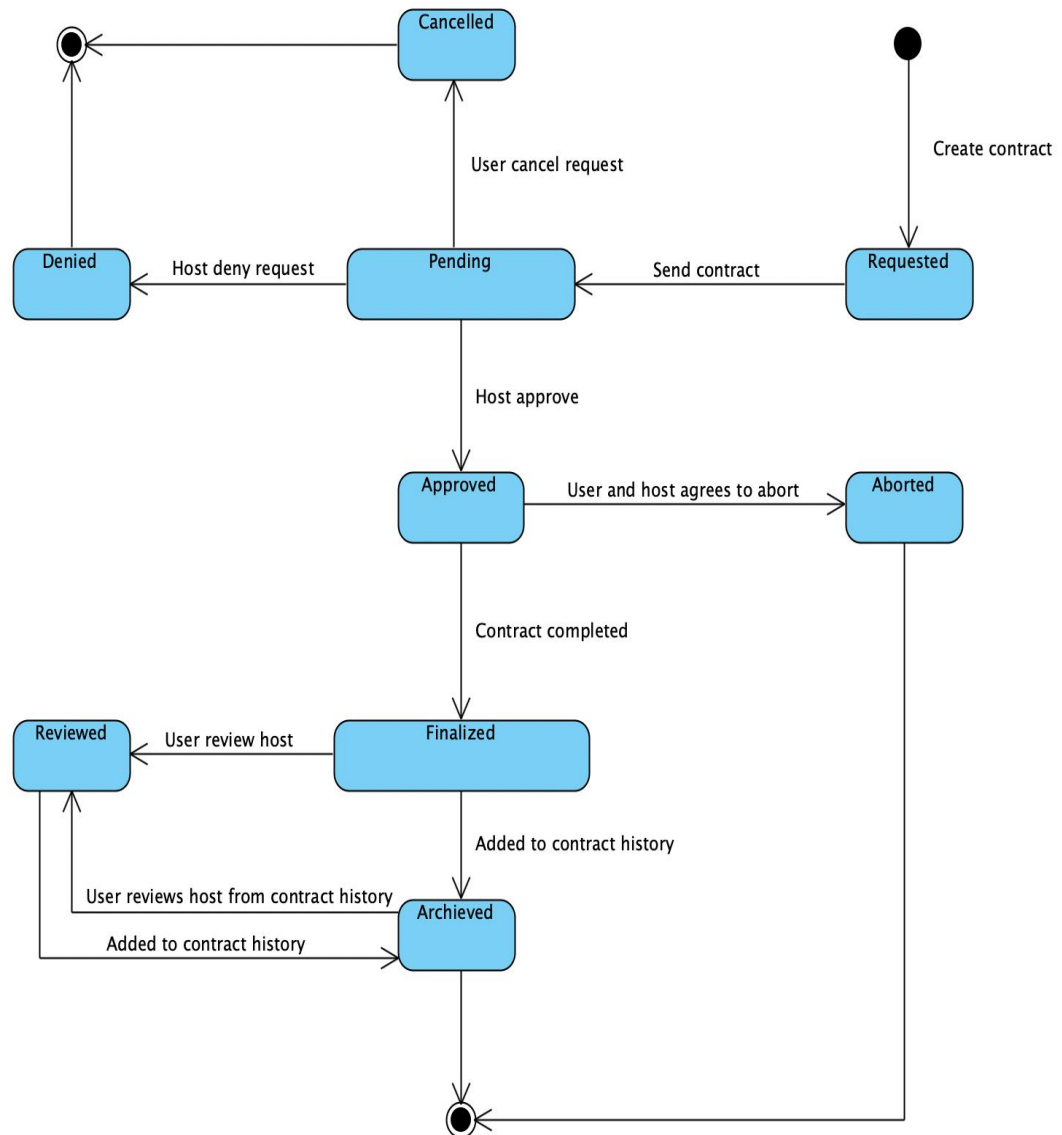


Figure 4: Contract State Diagram

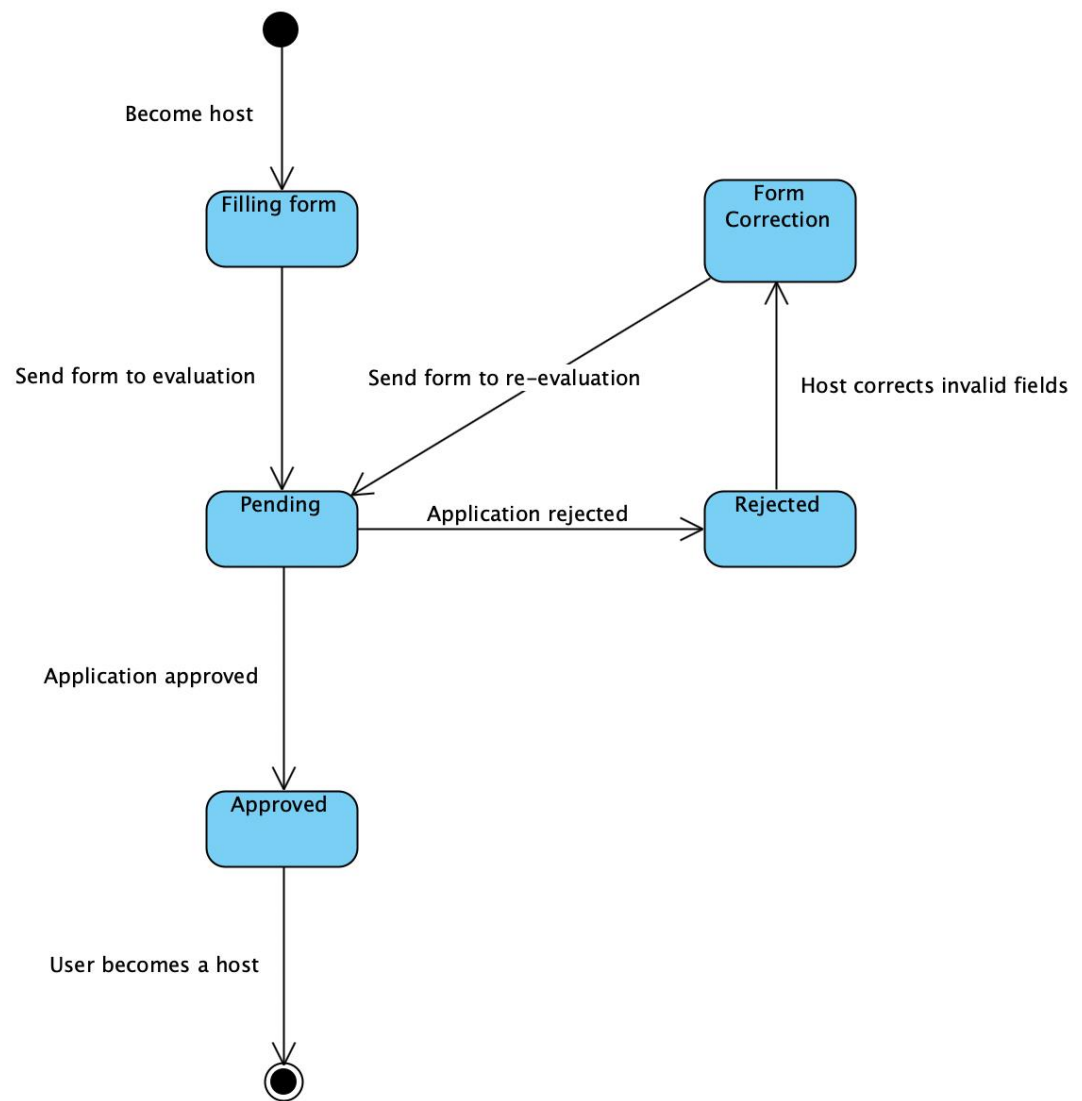


Figure 5: Become a Host Diagram

3.2.4.3 Sequence Diagrams

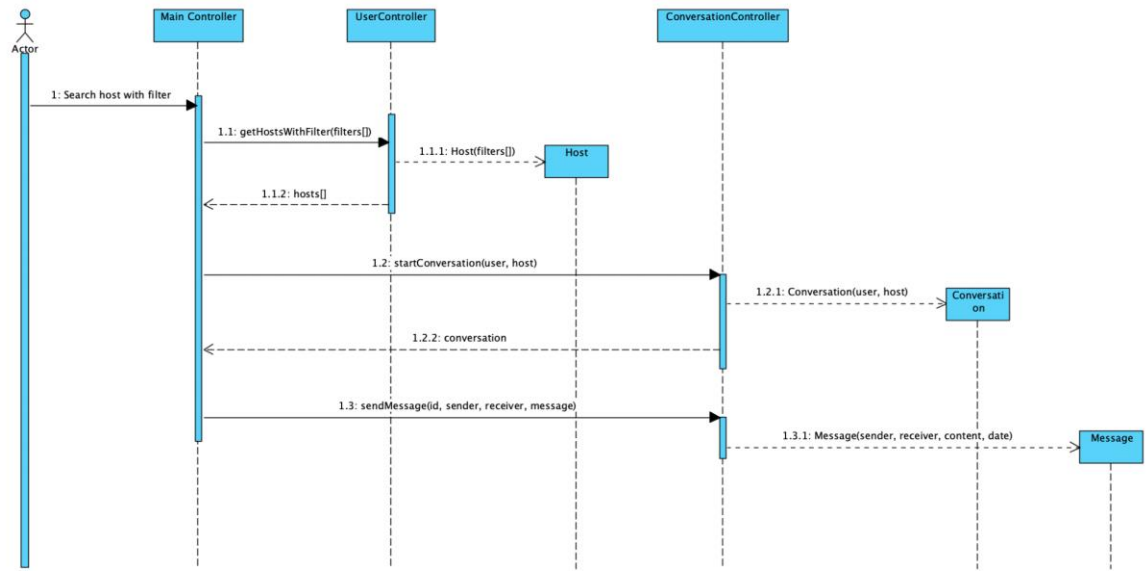


Figure 6: Start Conversation Sequence Diagram

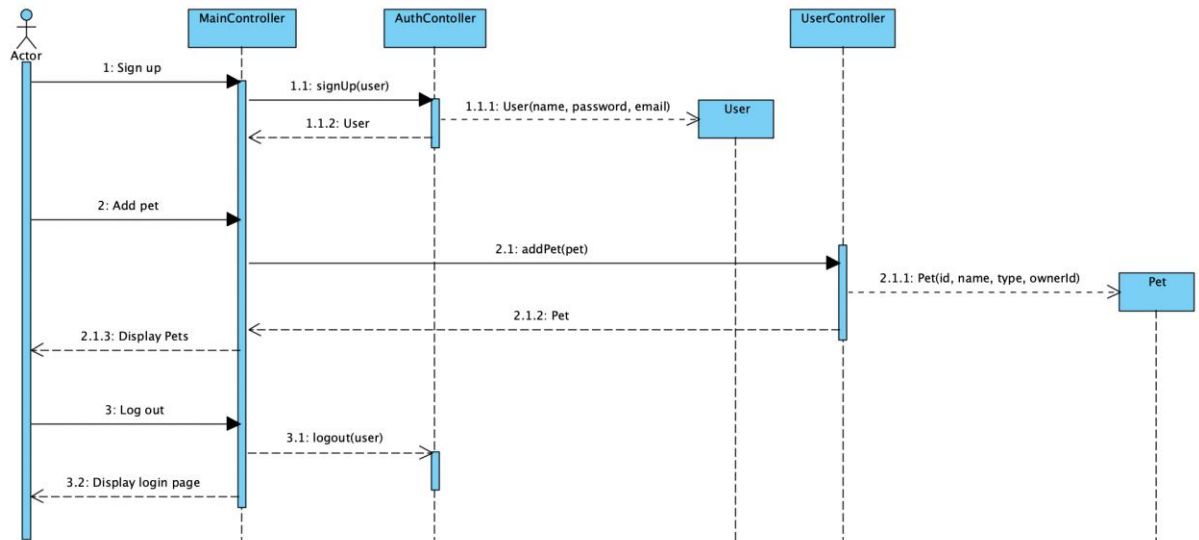


Figure 7: Sign up, Add Pet and Logout Sequence Diagram

3.3 Subsystem Decomposition

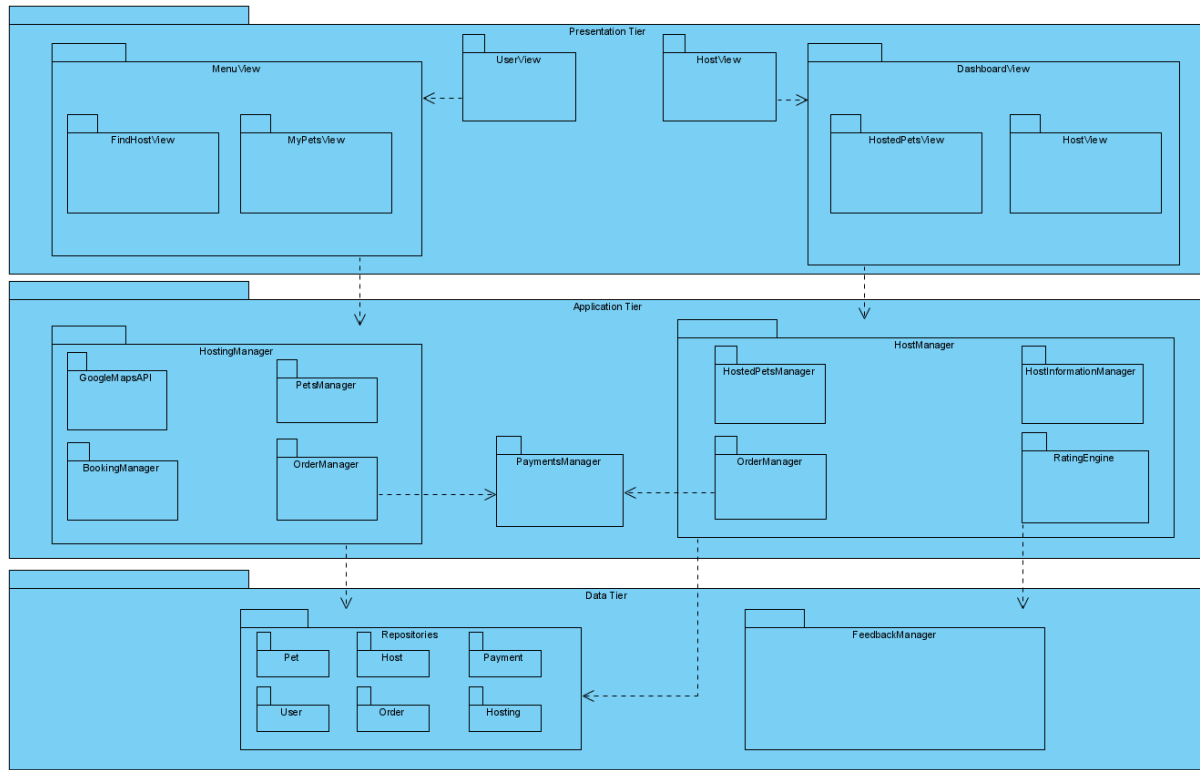


Figure 8: Subsystem Decomposition of Petriam

The architectural style of the system is 3-tier architectural style [2] that consists of three different layers: Presentation, Application and Data. The choice of 3-tier architectural style was made due to the scalability created by three different layers. As each layer has its own infrastructure, the development teams can work on their own simultaneously without running into problems.

The presentation tier represents the user interface of the application where the users can either see the application as a “user” or as a “host”. Depending on their view, they can access to different functionalities of the system. The Application tier on the other hand is where the logical operations of the system are handled. For example, whenever a user wants to register their pet on the system, an add operation is made on PetManager that can be viewed on MyPetsView. The third and the last layer which is the Data Tier is responsible for the actions that concern the

database of the system. The previously mentioned example registers a pet to the system but it must also be registered to the database. The Data Tier is responsible for such record keeping. In a way, the Application Tier is the bridge between the Data Tier and the Presentation Tier.

3.4 Hardware/Software Mapping

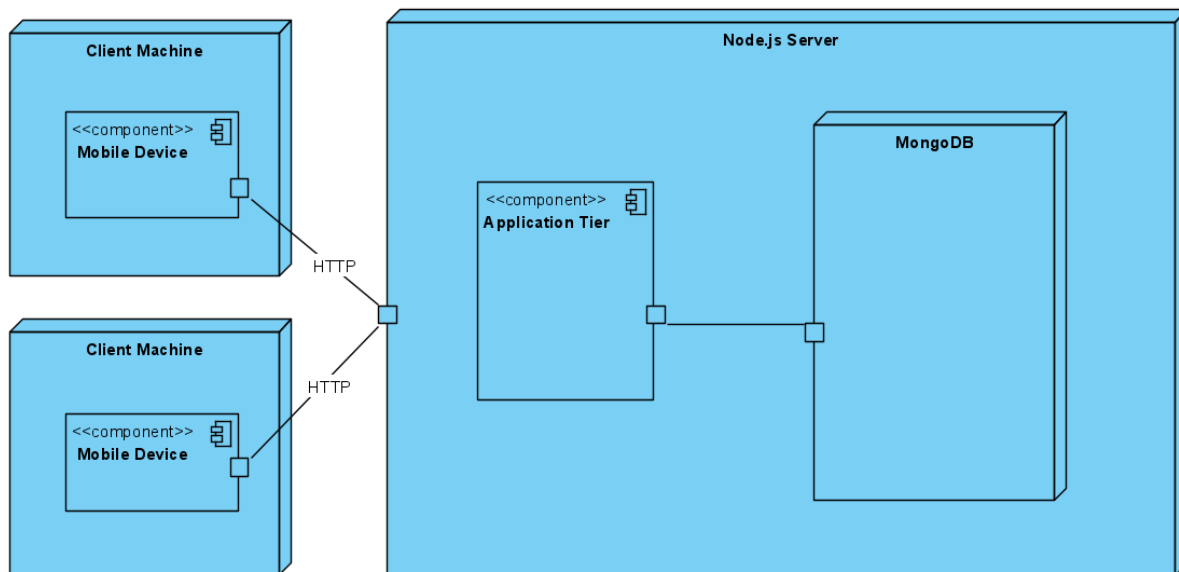


Figure 9: Component Diagram

Although Petriam is an application for mobile devices, the logic tier of the system will be run on a node.js server which is compatible with both mobile applications and web applications. Use of node.js allows the system the scalability to be expanded into the web one day. Each of the client machines connect to the node.js server through HTTP connection and the request of the clients are handled inside the server with the use of the MongoDB. After a request is finished, the node.js server responds to the client machines through the same HTTP connection. In the meantime, any changes in the system are stored in the Mongo database.

3.5 Persistent Data Management

For users, Petriam will need to store information such as profile picture, name, email, contact number, address as well as pet information. For hosts, data such as home pictures, logistics,

description of the services will be required to be stored on the database. The data collected by the application must be managed properly in order to give the users a positive experience of the platform. What is meant by properly managing data is that the reviews of the hosts must be carefully processed so that users are matched with hosts with higher ratings but also hosts who are trying to better their scores should be rewarded so that a host with an initial bad review is not outcasted by the application. The data will be stored on the Mongo Database and will be processed by the logic layer of the Node.js server.

3.6 Access Control and Security

The system allows two types of users: Users and hosts. However, a host can also benefit from the user side of the application. Currently, the system manages the authentication and authorization process through JWT which hashes the login information and passes it to the system. The JWT is later broken down by the internal authentication service using the secret hash values. For later improvements, the idea of setting up an external authentication server is in research phase. To better security, Google Authentication will also be added to the system for users who are not willing to sign up for new accounts.

3.7 Global Software Control

Petrium's software control is managed by a centralized event-driven system where the requests are handled by the respective managers of the events. After the client successfully logs in to the system, if the client is a user, then he/she can update their information by adding new pets to the system. In this case, PetsManager will add the pets to the system by registering them to the system. Later this user can list the available hosts by using the FindHostView. The FindHostView will request the list of the available hosts from the HostingManager and will place them on the map using GoogleMapsAPI. If the user wants to make a booking, then the

OrderManager handles the request as well as the PaymentsManager if the transaction gets successful. These events are simultaneously registered to the database in the meantime.

3.8 Boundary Conditions

3.8.1 Initialization

To initialize the application, the users must have downloaded the application from the respective application store (Google Play / AppStore) and have a stable internet connection. After the initialization, the users must login or sign up to the application. When the authentication is complete, the users can benefit from the hosting services of the system but must have a continuous and stable internet connection during their use of the application since everything on the system runs in real time and must be synchronized with the database.

3.8.2 Termination

The users can terminate the process of Petriam by simply logging out of the system. Note that if the user logs out of the system, he/she must log in again the next time the application is initialized. However, if the application is terminated by simply terminating the application using the mobile phone interface, their accounts will not be logged out automatically.

3.8.3 Failure

The system will have failures whenever the user has an unstable internet connection. Any changes made during lost connection will not be saved to the system, hence all of the information will be lost. If the system detects that the user has no internet connection, the user will be warned and will not be permitted to perform any actions while internet connection is not available. The user will also be notified when there is an internal server problem regarding the database that will not be able to perform any actions during system failures.

4. Development/Implementation Details

4.1 Front-End

The front-end development of Petriam was done using React Native [3], Typescript [4] and React Redux [5] that enabled us to publish our application on both iOS and Android platforms. For React Native, Expo framework [6] was utilized to fasten the application building phase by using the built-in functionalities of Expo. The reason behind choosing React Native was to be able to work on one environment while being able to publish the application cross platforms. One drawback of our choices on the front-end was that React Redux was an unfamiliar concept for our front-end developers that we had to do spent time on experimenting the functionalities of React Redux.

The front-end of Petriam is responsible for the presentation tier of our application where the user can interact and send requests to our back-end server. To be able to properly send requests, the Rest APIs built with the front-end were designed so that each of the required parameters for the back-end were carefully collected, wrapped and sent to the back-end server using Axios. One noticeable advantage of development of the front-end was that the developers were able to see the changes they were making simultaneously without requiring to reboot the system. This has allowed the front-end team to both work and discuss about the choices they were making.

Implementation of the front-end started with brainstorming sessions where the developers met up and used drawings of the application to determine the best UI design for the actual application. The two important factors for the developers were simplistic and to-the-point design. This is why Petriam has a focus on the map functionality that immediately draws the attention of the user as soon as they log in. After the drawing boards, the developers started to design the initial pages which were pages such as login, register and home. After finishing implementing the map on the home page as well as the simple pages such as user profiles, it was time to start integrating the functionalities of the back-end to the front-end. At that point, the two teams started working

together to handle user requests from front-end to back-end. The front-end implementation was completed when the UI of the application was able to send requests, edit data on the data tier and see the changes made from the back-end server.

4.2 Back-End

The back-end development of Petriam was done using Node.js [7] which is an open-source server environment that uses JavaScript on the server. The reason why we chose Node.js is that it has extensive scalability and good performance for the server. Specifically, we have used the Express.js [8] framework for our project because of the minimalistic and simplistic nature of Express.js. This choice was helpful for back-end development since the back-end developers were not fully experienced with Node.js and was easy for them to grasp and build the application. The back-end of Petriam utilizes MongoDB [9] which is a NoSQL database that is later discussed in the Persistent Data Management section.

The back-end system consists of the Application and Data Tier of the subsystem decomposition which was discussed in the previous sections. The application tier of the back-end system is responsible for handling the requests sent by the front-end of the application. The requests could include information about the users, hosts, pets or contracts that require processing before sending a response to the user. While handling requests, the back-end also makes use of the data tier which is the database of the system that contains the data models.

The implementation of the back-end started with initialization of the back-end server with the required ports. After starting the server on the localhost, we have decided that as the team, we have to work on servers rather than localhosts to be able to work simultaneously. For that reason, we have created our database on MongoDB atlas which is a hosting service for MongoDB and used Heroku [10] for our Node.js Express framework to run on. After setting up the servers, any

changes made on our machines were directly visible to each and every member of our team. This has increased our work efficiency since the team members worked remotely.

The data models were the first actual implementations of Petriam where we have defined our entities. The definition of entities was used as a guideline for the developers to provide proper endpoints to define as routes for each specific user types. When each of the endpoints for the models were completed, the back-end team started working synchronously with the front-end team to integrate each and every functionality of the back-end system. This has caused lots of refactoring and editing the back-end code of Petriam so that each request sent by users were correctly processed.

4.2 Persistent Data Management

The persistent data management of the application was done using Mongo Database. It was crucial of the project to keep the user data safe and in order since any slight conflict in the database could cause hosts to lose money and users to leave unsatisfied. It was our duty to keep the data collected from the users in order as it was sensitive data concerning private information such as national identification numbers, telephone numbers or addresses. Petriam has a collection for users, hosts, pets, contracts, host applications, admins, messages and reviews. Each of the listed collections have their unique ids and specific details concerning the data models.

5. Testing Details

5.1 Version Checking

GitHub [11] was used all throughout the project for back-end and front-end in separate repositories. This has allowed the developers to work on different branches and test their functionalities before actually putting their implementations to the main branches. The developers

also used code review on GitHub to constantly compare and review each other's code to solve bug issues and implementation problems.

5.2 API Testing

The endpoints written in the back-end server were prepared in the development stage and simultaneously tested while implementing them. To test the API endpoints, we have used Postman to ensure that we were getting the correct responses from the endpoints. To be able to fasten this testing process, we have created collections in Postman and worked in the same workspace to see each other's requests. We have made use of variables in Postman for different user types that allowed us to switch between requests in an efficient manner without having to prepare specific requests for each user types each time.

6. Maintenance Plan and Details

6.1 Back-End Server Maintenance

Petriam makes use of Heroku and MongoDB Atlas servers to run the requests. For small user scales, the current plan for the servers is to be renewed each year. However, in case of an expansion, the servers must be upgraded accordingly to withstand increasing number of users. In a scenario where the maintenance for the servers is omitting with increasing number of users, the requests which are sent to the back-end could be handled in slower time that would violate the non-functional requirements of the project. To be able to have faster response time, the servers must be optimized to the current user needs.

It is important to note that Node.js and React Native frameworks get regular updates that also require for the servers to be updated as well. If the servers are kept on the same versions, the servers will be vulnerable to security breaches in the future as they will be omitted from any future patches.

6.2 Persistent Data Management Maintenance

The use of MongoDB in Petriam has boosted the efficiency of developers so far since MongoDB is a NoSQL database that allows developers to interfere directly without requiring any SQL queries but the downside of NoSQL databases is that for larger user bases, they are comparably slower and less reliable than traditional SQL databases. To ensure that the persistent data management stays up-to-date and fast, the data tier of the project must be maintained regularly and be upgraded to a SQL database, if possible, in a scenario where the number of users increase exponentially.

6.3 Front-End Maintenance and Optimization

The front-end of Petriam requires constant feedback and upgrades as it is responsible for the presentation layer. Even if the builds for React are up to date, the visual aspect of the application can always be better considering that the market of mobile applications are highly competitive. To ensure that the front-end of Petriam is maintained and optimized regularly, monthly meetings could be arranged to check user data and feedback to further improve the feedback. Any slight problem in the front-end of the application might seem harmless to the functionality of the system but could be harmful to the user experience that could cause loss of users.

7. Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

Public Health

The human interaction in Petriam is limited to dropping off pets and getting them back from their hosts. The remaining functionalities of the application can be performed on the app which does not pose any dangers to public health. However, considering the very small window of human interaction in Petriam, there are possible threats to public health. The first and main concern is the situation created by the pandemic. A host or a user may cause the spread of the

virus if they continue to use the application when they need to be quarantined. Petriam will solve this problem by requiring mandatory HES code [12] from all of its users whether they are hosts or regular users. The collected HES codes will be used to check the user's Covid-19 risk status that will decide if a user is eligible to be a part of the services given in Petriam. There exists one other risk involving pets and humans. The guest pet or the pets belonging to the hosts may contain diseases that could cause the spread of various diseases. In Petriam, the health of the users and the pets are prioritized to ensure a healthy environment for everyone. In order to prevent any diseases among the pets, we have come up with two solutions: Mandatory submission of vaccination cards of the pets or a signed document from a veterinary that confirms the pet's availability to travel.

Public Safety

Since most of the application's services are available online, the only possible public safety concern is the human interaction which is described above. Possible public safety scenarios might occur when a user with a criminal background or mental disorders becomes a host. Although indication of such a scenario is less likely, Petriam will be requiring criminal and health reports from its users to become hosts. Users with any sort of criminal records or violent mental disorders will not be accepted to become hosts to lower the chances of having unpleasant situations. The rating system in the app will provide a competitive environment for the hosts that will promote good service among the hosts. Any host with unpleasant behavior can easily be rated or even reported if the behavior of the host was not acceptable. If the reports are found to be accurate and risk posing, the hosts will no longer be allowed to use the app to ensure public safety.

Public Welfare

Petriam will be a free to use application for everyone. The users will only be charged when they want to use the services of a host. Although Petriam does not decide on the host prices,

since it is an open market, the prices are expected to be reasonable due to the competition between the hosts. Global Factors The application will be developed according to the global set of standards imposed to every application on Google Play Store and Apple App Store. The nature of the application requires personal information such as name, address, telephone number to be collected and we will be abiding to General Data Protection Regulations which are standardized by the European Union to ensure data protection. Petriam will not share or use the data of its users under any circumstances.

Cultural Factors

The cultural factors have nearly no effect on Petriam as the main focus is on animals. Considering the fact that culture is a trait specific to people and not to animals, Petriam is open to use for people from every possible national and cultural background.

Social Factors

The social interactions in the application are limited by the interaction between the users and the hosts. In order to provide an equal social environment to all users, the application will come in English and Turkish language options. The social factors could be analyzed from the pets' point of view where certain pets might require social interaction to prevent any negative psychological effects on the pets. This requirement could be indicated by the pet owners while they are negotiating with the hosts.

Factor	Importance	Possible Negative Scenario
Public Health	High	Effect on health due to spread of diseases
Public Safety	High	Physical and psychological threat to users due to lacking background check
Public Welfare	Moderate	High host prices
Global	Moderate	Leak of personal data
Cultural	Low	Threat to highly specific cultural backgrounds
Social	Moderate	Indirect phycological threat to pets

Table 1: Summary of Various Factors and Their Status

7.2 Ethics and Professional Responsibilities

7.2.1 Ethical Responsibilities

Due to nature of Petriam, as the developers, user privacy and safety is the most important feature of the project. Unlike assets, pets are linked to their owners through an emotional connection. For this reason, Petriam must ensure that the applications of the hosts are examined thoroughly without any exceptions to prevent any predictable negative outcome. This is the most crucial ethical concern since failure to provide this responsibility would be highly unethical and would permanently damage the reliability of the application.

The second most important ethical concern is the preservation of personal data. The users will be sharing their home addresses, images, telephone numbers that are highly sensitive. The data should be kept in a hashed state in the database and never be shared. To prevent a scenario where the data is stolen from a third party, highly trusted APIs should be implemented in the system rather than using outdated security technologies. The developers will abide Code of Ethics and General Data Protection Regulation to ensure this responsibility.

7.2.1 Professional Responsibilities

The professional responsibilities of the project may be divided into two parts as internal and external professional responsibilities. The internal professional responsibilities concern the responsibilities that are related with the internal working environment of the team while the external professional responsibilities are concerned with the project code itself.

In terms of the internal professional responsibilities, the team must work in a responsible and coherent way to ensure a healthy work environment. The work appointed to the members should not be mandatory but they should voluntarily be based on their interest. The team mates must participate equally and must be present to all of the meetings unless there is a valid excuse.

The internal professional responsibilities are determined when the team was first formed and the work done before has been done according to the given rules.

The external professional responsibilities of the project include subjects such as plagiarism and documentation. The reports and implementation of the project must not be taken from any other third-party source and in parts where there is a case of inspiration, the sources must be credited. On the other hand, team mates should post their works on their repositories and keep the source code secure in order to prevent any data breaches. Each member of the team is responsible for securing their own work whether it is on the cloud or their personal computers.

7.3 Judgements and Impacts to Various Contexts

The following table demonstrates the judgements and impacts of Petriam on contexts such as global, economic, environment and societal. The impact levels on the table are rated out of 5 as 1 being the lowest and 5 being the highest impact on the context.

	Impact	Impact Level
Global Context	Petriam boosts the concept of middleman business model where services given by individuals are collected under one environment.	3
Environment	The carbon print of Petriam is in an omittable level since the only factors are server maintenances.	1
Societal	Petriam creates jobs for people and aims for a self-sustaining society where the need for pet hotels is lessened.	4
Economic	The jobs created from the hosting services are new sources of passive income for hosts and cheaper choice for users who are looking for pet hosting services.	5

Table 2: Judgements and Impacts to Various Contexts

7.4 Teamwork Details

7.4.1 Contributing and functioning effectively on the team

In order to ensure that everyone is contributing and functioning effectively on the team, we have made a detailed division of work and enabled communication in every possible channel to make sure the efficiency of the team was maximized. The mentioned “division of work” and “communication channels” will be talked in detail in the following sections.

7.4.2 Helping creating a collaborative and inclusive environment

Apart from the unofficial meetings conducted at school, we conduct official weekly meetings online to check each other's work and plan ahead. To provide the communication between the team members, we benefit from a variety of online tools which are:

- Zoom & Discord for Visual and Verbal Meetings
- WhatsApp for unofficial Meetings and Scheduling for Planned Meetings
- Slack for Planning
- Google Docs for Collaborated Written Work
- GitHub for Collaborated Coding Work

The listed communication channels made sure that everyone was on top of the subject rather than being left out. By allowing everyone to check in and being a part of the project, we have made sure a collaborative and inclusive environment.

7.4.3 Taking lead role and sharing leadership on the team

In order to promote efficient team work, we have appointed sub-teams with their own leaders to ensure the quality of the work done in each of the sections. The appointment of the leaders was based on past experience and proficiency in each of the respected fields.

- Front-End Development: Osman Buğra Aydın (L), Doğancan Yılmazoğlu
- Back-End Development: Alperen Yalçın (L), Oğuzhan Angın, Ertuğrul Aktaş
- Database Integration: Ertuğrul Aktaş (L), Alperen Yalçın
- Documentation and Reports: Ertuğrul Aktaş (L) and everyone
- Architectural Design: Osman Buğra Aydın (L), Alperen Yalçın, Oğuzhan Angın
- Visual Design: Doğancan Yılmazoğlu

The above-mentioned teams have their own respected leaders that make sure everyone who is willing to lead in the team gets the opportunity to lead in their provided sections.

7.4.4 Meeting objectives

This section focuses on the actual work done compared to the planned work at the beginning of project specification. The reasons for the cancelled or modified work are also discussed in this section.

At the beginning of the project, it was planned that a payment method should also be implemented to the application. However, after doing research on the topic, we have realized that in order to create revenue from the application, as the group members, we had to apply to be a legitimate company. Even though a payment method would be crucial to the system, focusing on the paperwork to legitimately become a company and applying for payment services would only divert us from the actual purpose of the project which is to create a platform for pet owners. The

paperwork would only slow the implementation down and prevent us from actually learning and applying our knowledge to the project.

Other than the payment methods, Petriam has the promised functionalities just as before with small adjustments made to better run from a software development point of view. An example for that would be the entity model specifics which were slightly different in the analysis report.

Lastly, after the removal of COVID-19 restrictions, the support for HES codes were removed from the project.

7.5 New Knowledge Acquired and Applied

The journey of Petriam has taught each and every member of the project unforgettable lessons about both project management and development skills. We have listed the main branches of the new knowledge we have acquired below and discussed how we applied them to our project.

- **Back-End Development:** Both of the development teams had no experience with Node.js before the project that was seemingly challenging for us. Although as the back-end team we have struggled in the beginning, the simple and scalable characteristic of Node.js has enabled us to use it efficiently in a short period of time. We have learned how to deploy and run a server on Node.js with functioning endpoints and data models and used this knowledge in the project.
- **Front-End Development:** The front-end team had previous experience with React Native however, React Redux was also an area that no one had full experience with. By having regular meetings and comparing progresses, we have taught each other what we have learnt on our own and applied it to the project. Especially, during the integration phase of front and back end of the project, both of the teams have experienced how the two ends of the project interact and work on their own.
- **Online Project and Communication Tools:** From the beginning of the project, the members were aware of the fact that asynchronous communication channels such as

WhatsApp were insufficient to track our progress in the project. For that reason, we have been using tools such as Trello to put detailed explanation and status of our current work. We have also used code reviews to check and understand each other's work and arranged a meeting when required further explanation.

- **Leadership:** In each phase of the project, we had different people lead the project on their own way. As the leaders appointed sub-leaders to tasks, we have learnt how a small scaled team works and understood our responsibilities to each other. This way, each member of the project experienced the mentality of being in charge of a project that can not be accomplished by themselves alone and used their experience on their work.

8. Conclusion and Future Work

8.1 Conclusion

The implementation of Petriam has been successfully completed with a few exceptions described above as it was planned to be implemented at the beginning of the year. The platform aims to create a market place for pet owners and pet hosts where they can interact and build an organic community. The current status of the system is a prototype that requires much more testing with actual users. The application can be started as a pilot project in specific local areas to get customer feedback and then Petriam can be scaled to countrywide after the improvements made from the feedback. Lastly, the project does not have the legal right to create revenue since Petriam is not recognized as a company regulated by the government that keeps the project as a prototype until further legal actions.

8.2 Future Work

The most crucial future work of the project would be to implement the payment functionality in Petriam since the platform aims to create revenue for its hosts. After the payment issue is

handled, the application can be opened to public and serve its users. To be able to scale Petriam for larger audiences, the persistent data management of the application could be updated and new functionalities such as pet sitting, brooming or care could be added to have more options for the users and hosts. For admin users, a panel on web could be developed to better process host data and a web application could be implemented for PC users to attract new possible users.

9. User Manual

This section discusses how to use the application with its given functionalities. The section omits the manual for admin users that only focuses on regular user and host user types.

9.1 Sign in and Sign up

The users can either create a default user account or directly sign in using their already existing email and password combinations. In order to become hosts, the users must first log in and apply to be a host inside the application.

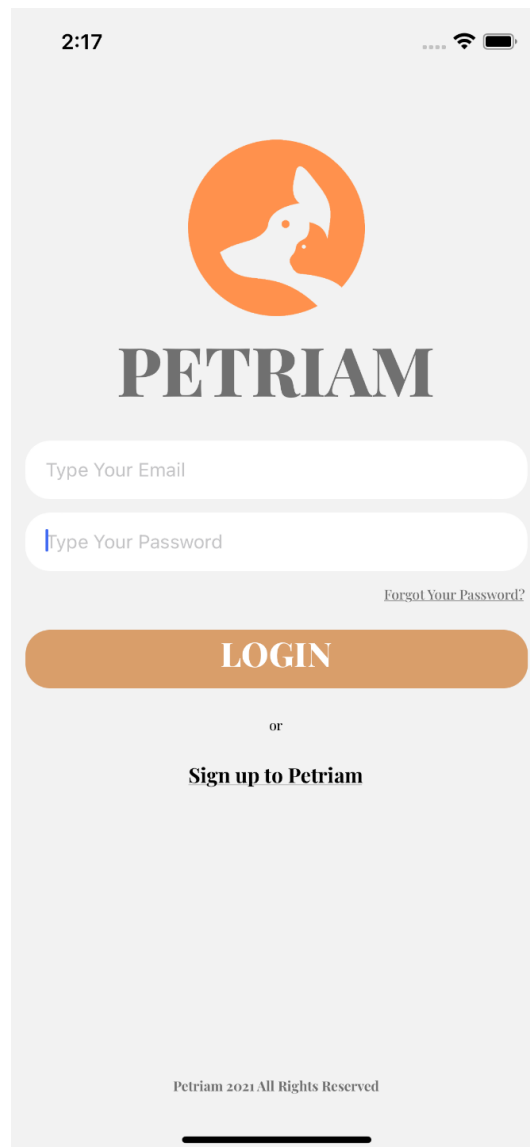


Figure 10: Login

Figure 11: Sign Up

Figure 11: Sign Up

9.2 Home Page

In the home page, the user is greeted with the map that contains the local hosts in his/her area. The tool bar below navigates to contracts, messages, host application and user profile.

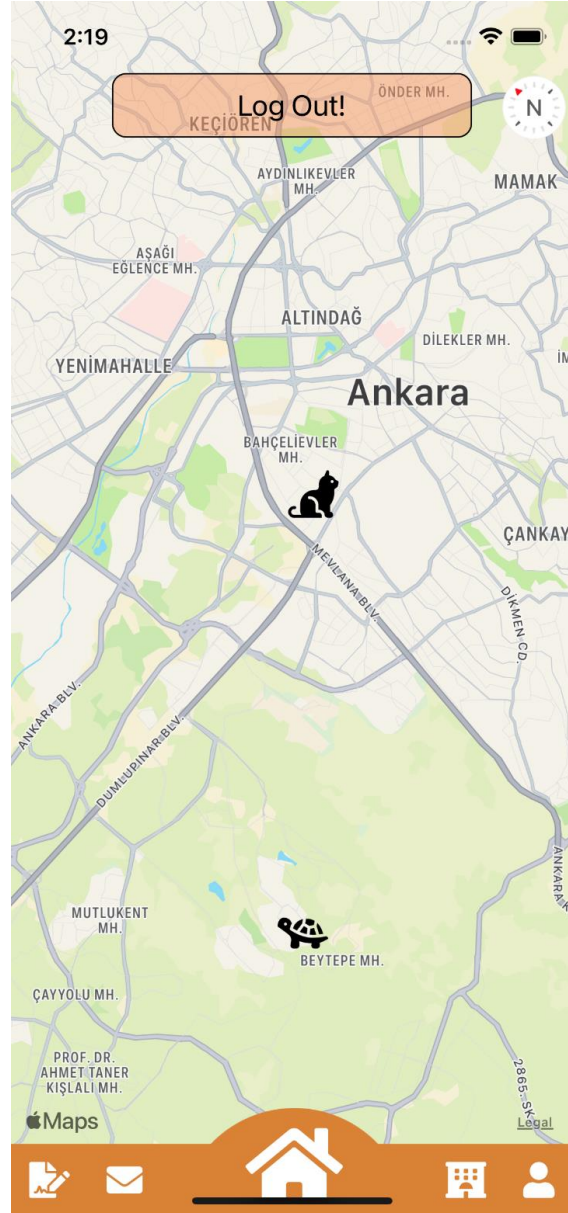


Figure 12: Home

9.3 User Profile

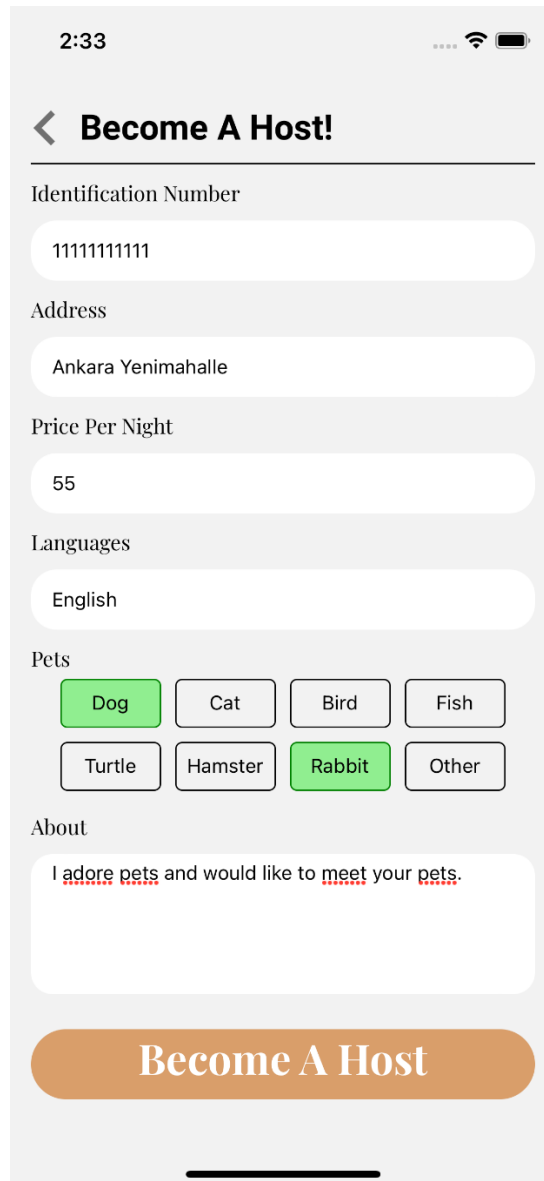
In the user profile page, the user can see his/her information on the application which is name, about me, address, pets and verification status.



Figure 13: User Profile

9.4 Host Application

In host application page, the user can apply to be a host by giving his credentials. However, the user must wait for an approval by from an admin. After the application is made, the user must wait for a response from an admin to be able to reapply again in case of a rejection.



The screenshot shows a mobile application interface for becoming a host. At the top, the status bar displays the time 2:33, signal strength, Wi-Fi, and battery icons. The app header features a back arrow and the title 'Become A Host!'. The form consists of several sections: 'Identification Number' with a text input containing '1111111111'; 'Address' with a text input containing 'Ankara Yenimahalle'; 'Price Per Night' with a text input containing '55'; 'Languages' with a text input containing 'English'; 'Pets' with a grid of buttons for 'Dog', 'Cat', 'Bird', 'Fish', 'Turtle', 'Hamster', 'Rabbit', and 'Other', where 'Dog' and 'Rabbit' are highlighted in green; and 'About' with a text area containing the sentence 'I adore pets and would like to meet your pets.' with red dotted lines indicating a character limit. At the bottom is a large orange button labeled 'Become A Host'.

2:33

< Become A Host!

Identification Number

1111111111

Address

Ankara Yenimahalle

Price Per Night

55

Languages

English

Pets

Dog Cat Bird Fish

Turtle Hamster Rabbit Other

About

I adore pets and would like to meet your pets.

Become A Host

Figure 14: Host Application

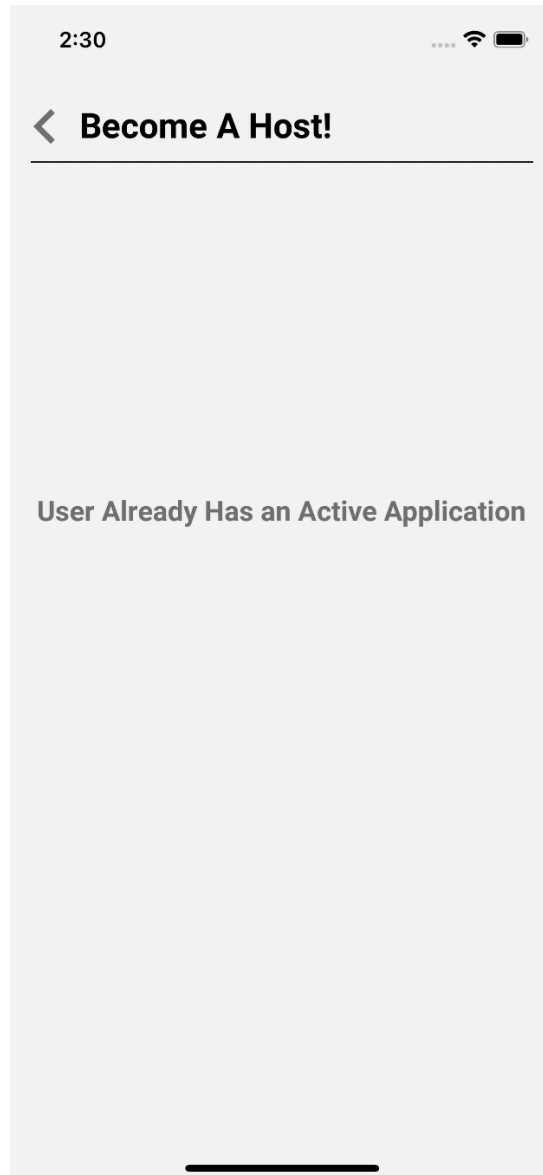



Figure 15: Already Existing Host Application

9.5 Hire a Host

Users can select a host from the map and hire the hosts by entering the parameters. After the user hires the host, the host should also approve the hiring on his contracts page. After both ends agree, the contract begins and is stored in the contracts section.

2:28

<



Ertuğrul Yalçın ✓

Dog Owner Cat Owner

*Verified with TC and Address

About Me

Hello, I am your friendly neighborhood host. I love cats and fishes. Please call me.

Address

Kazakistan Caddesi No: 138

Price
55 TL
Per Day

★★★★★
4.5/5
rating

Choose a Start Date
06/05/2022

Choose a Finish Date
06/05/2022

Your Pets

hagi (cat) karabaş (turtle) pamuk (cat)

Hire Host




Figure 16: Host Hiring

2:29



Ertuğrul Yalçın ✓

Dog Owner Cat Owner

**Verified with TC and Address*

About Me

Hello, I am your friendly neighborhood host. I love cats and fishes. Please call me.

Address

Kazakistan Caddesi No: 138

Price

55 TL

Per Day



4.5/5

rating

Choose a Start Date

06/05/2022

Choose a Finish Date

24/05/2022

Your Pets

hagi (cat)

karabaş (turtle)

pamuk (cat)

Successfully Hired!



Figure 17: Successful Host Hiring

9.6 Contracts

Hosts can accept the contracts sent to them on this page and the users can also get updated on their current host hiring processes. Any contracts that expire are flagged as “Passed” to indicate that the contract is no longer valid. If the contracts get accepted, they are flagged as “Accepted” and if the hosting process has started, it is flagged as “On-going”.

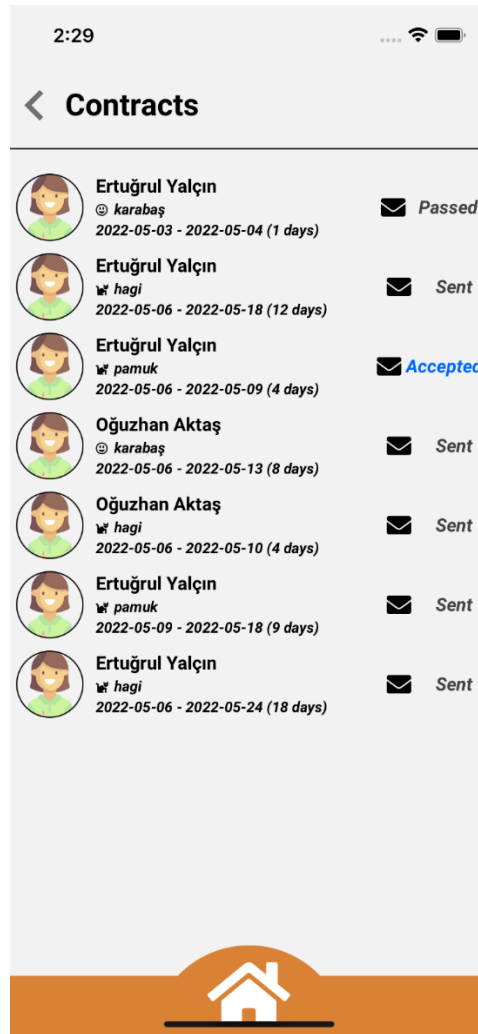


Figure 18: Contracts

9.7 Messaging

Users can see their messages to other hosts on this page where they can send messages or checkout their profiles.

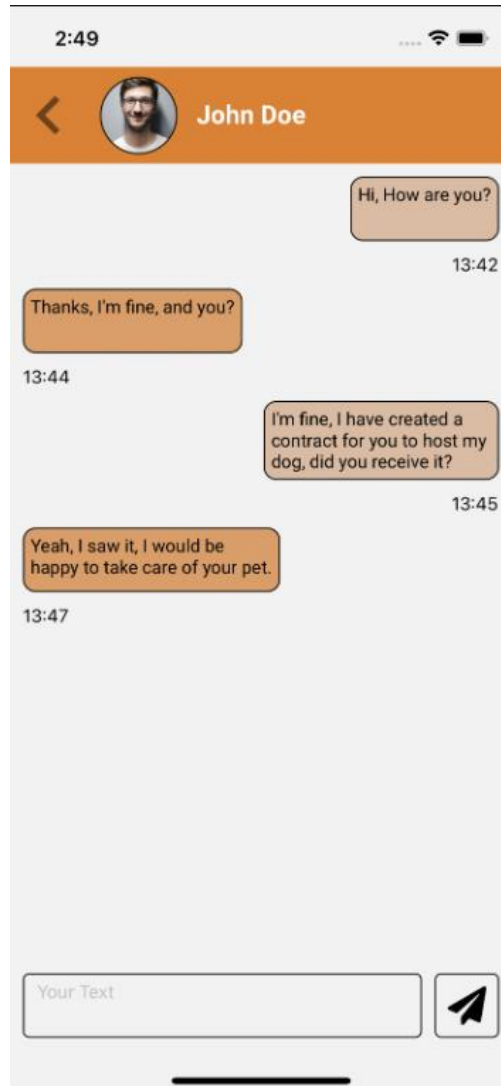


Figure 19: Messaging in between users

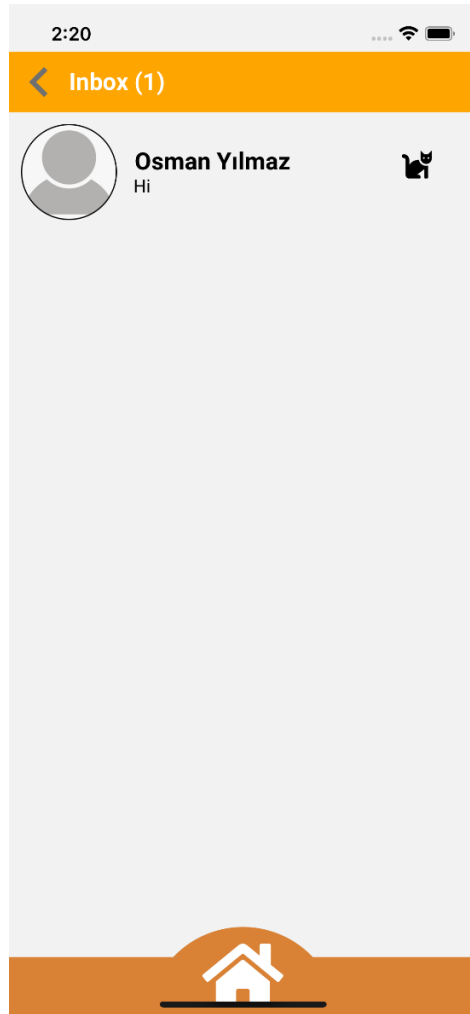


Figure 20: An Inbox with one message

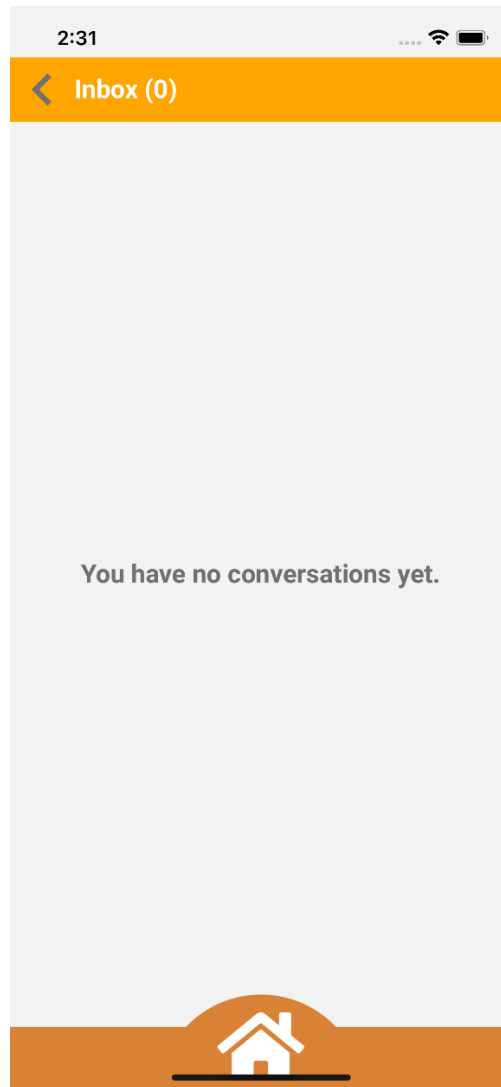


Figure 9: Empty Inbox

10. Glossary

- **Node.js:** An open-source server environment which is free and capable of dynamic page content.
- **React Native:** An open-source UI framework that can be used for Android, iOS, Web and Windows that was developed by Meta company.
- **Heroku:** A cloud platform that is used for building and monitoring servers.
- **MongoDB:** An open-source NoSQL database that stores the data in JSON format.
- **GitHub:** A version control system for developers where the users can store their implementations.

11. References

- [1] W. Coy, "How Much Does Dog Boarding Cost?," rover, [Online]. Available: <https://www.rover.com/blog/how-much-does-dog-boarding-cost/>.
- [2] IBM Cloud Education, "Three-Tier Architecture," IBM, [Online]. Available: <https://www.ibm.com/cloud/learn/three-tier-architecture>.
- [3] "React," Meta, [Online]. Available: <https://reactjs.org/>.
- [4] "What is TypeScript?," TypeScript, [Online]. Available: <https://www.typescriptlang.org/>.
- [5] D. Abramov, "React Redux," [Online]. Available: <https://react-redux.js.org/>.
- [6] "Expo Docs," [Online]. Available: <https://docs.expo.dev/>.
- [7] "About Node.js," [Online]. Available: <https://nodejs.org/en/about/>.
- [8] OpenJS Foundation, "Home," [Online]. Available: <https://expressjs.com/>.
- [9] MongoDB, Inc., "Our Mission," [Online]. Available: <https://www.mongodb.com/company>.
- [10] salesforce, "What is Heroku?," [Online]. Available: <https://www.heroku.com/what>.
- [11] GitHub, "GitHub Code Review," [Online]. Available: <https://github.com/features/code-review>.
- [12] "HES Kodu Nedir?," Ministry of Health, [Online]. Available: <https://hayatevesigar.saglik.gov.tr/hes.html>.