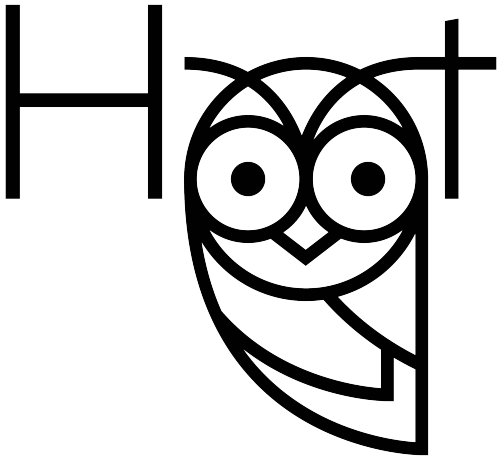
TEAM ALESTA

CSE 343 Software Engineering

Final Report



DOĞUKAN TAŞTAN

EMİRCAN DEMİREL

EMRE YILMAZ

MUHAMMET SEFA CAHYİR

MUSTAFA MERT

TARIK EMİR KALDIRIM

Contens

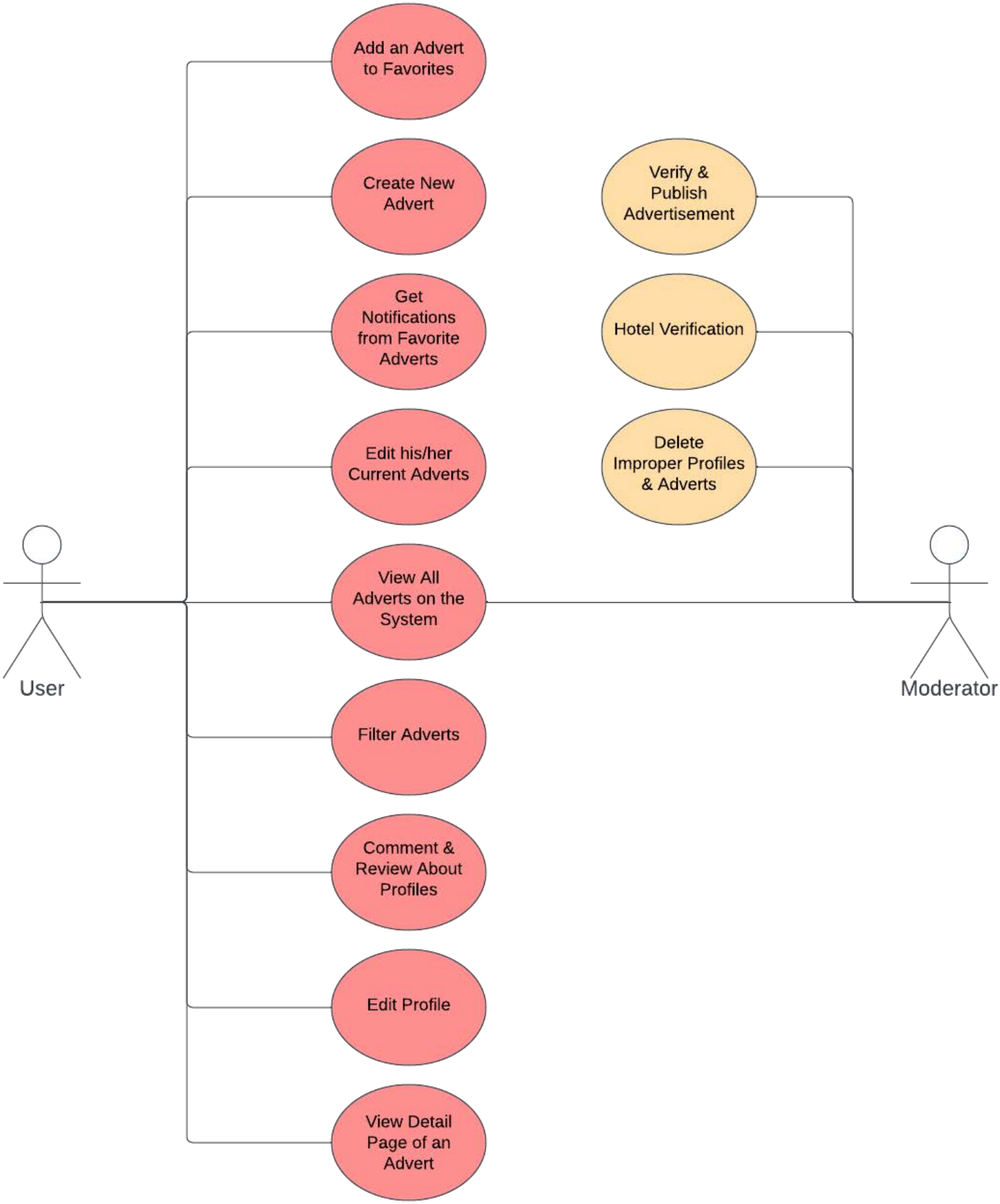
1.UML Diagrams

1.1 Context Model

By using firebase storage service, we both store photos and make it easier to access them by converting them into urls.

With the help of firebase authentication service, we increase the security of the application by hashing user passwords, as well as services such as sending emails for account confirmation, resetting passwords. This service also offers us an interface through which we can comfortably manage users.

1.2 UUse-case diagrams and their detailed tables



System	*Moderation system*
Use Case	*Hotel verification*
Actors	*Moderator*
Data	*is user a hotel => if user is a hotel it will be true*
Stimulus	*There can be two type of user normal people who loves animals and want to keep them or a place that keep your animal as a job*
Response	*User set as hotel*
Comments	-

System	*Moderation system*
Use Case	*Verify & publish Advert*
Actors	*Moderator*
Data	*is advert okay to publish => if an advert okay to publish data will be true*
Stimulus	*There can be automated or fake adverts because of publishment depends on a control*
Response	*Advert published*
Comments	-

System	*Moderation system*
Use Case	*Delete improper profiles*
Actors	*Moderator*
Data	*user profile => if a profile improper moderator can delete it*
Stimulus	*if a profile improper moderator can delete it*
Response	*Improper user deleted*
Comments	-

System	*User actions system*
Use Case	*Add advert to favorites*
Actors	*User*
Data	*advert id => mapping with advert id for user*
Stimulus	*If a user likes and advert and wants to keep it for future*
Response	*Advert added to favorites*
Comments	-

System	*User actions system*
Use Case	*Edit profile*
Actors	*User*
Data	*user's data => user's data updated*
Stimulus	*If user wants to edit any information about him/herself*
Response	*Profile updated*
Comments	-

System	*User actions system*
Use Case	*Get notifications from favorite adverts*
Actors	*User*
Data	*user's favorite adverts => after changes on the adverts user will be notificated*
Stimulus	*If advert has an update user will be notificated about that*
Response	*Favorite advert update notification*
Comments	-

System	*User actions system*
Use Case	*User comments about profile*
Actors	*User*
Data	*user comments => when a user comments about a profile added to profile*
Stimulus	*User wants to share experience with the user to others to help*
Response	*Comment sent*
Comments	-

System	*Advert management system*
Use Case	*Upload photo for new advert*
Actors	*User*
Data	*image => image added to related advert*
Stimulus	*If user wants to add photo to their advert*
Response	*Photo uploaded*

Comments	-

System	*Advert management system*
Use Case	*Add new advert*
Actors	*User*
Data	*advert data => saved to database*
Stimulus	*If user wants to share new advert*
Response	*Advert created*
Comments	-

System	*Advert management system*
Use Case	*Add/edit advert features*
Actors	*User*
Data	*advert's data => advert's data added or updated*
Stimulus	*If user wants to add new info or update one*
Response	*info added/uptaded*
Comments	-

System	*Advert searching system*
Use Case	*View last adverts*
Actors	*User*
Data	*Recent adverts => showed to user*
Stimulus	*If user wants to see latest adverts*
Response	*Latest adverts will be shown*
Comments	-

System	*Advert searching system*
Use Case	*Filter Adverts by Location*
Actors	*User*
Data	*Adverts => filtered according to location*
Stimulus	*If user wants to see the adverts filtered by location*

Response	*Adverts filtered by location will be shown*
Comments	-

System	*Advert searching system*
Use Case	*Filter Adverts by Date*
Actors	*User*
Data	*Adverts => filtered according to date*
Stimulus	*If user wants to see the adverts filtered by date*
Response	*Adverts filtered by date will be shown*
Comments	-

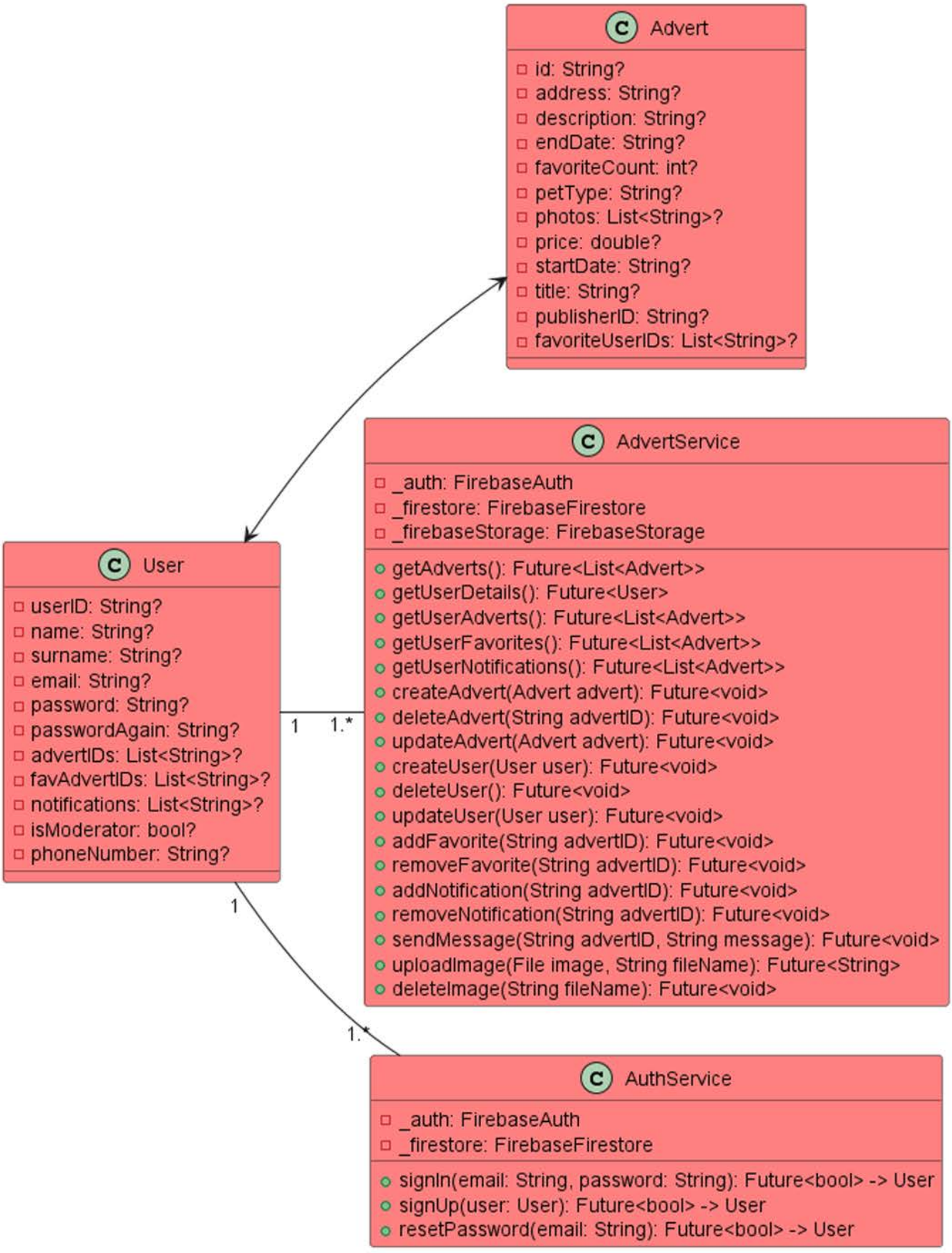
System	*Advert searching system*
Use Case	*Filter Adverts by Price*
Actors	*User*
Data	*Adverts => filtered according to price*
Stimulus	*If user wants to see the adverts filtered by price*
Response	*Adverts filtered by price will be shown*
Comments	-

System	*Advert searching system*
Use Case	*Filter Adverts by Pet Type*
Actors	*User*
Data	*Adverts => filtered according to pet type*
Stimulus	*If user wants to see the adverts filtered by pet type*
Response	*Adverts filtered by pet type will be shown*
Comments	-

System	*Advert searching system*

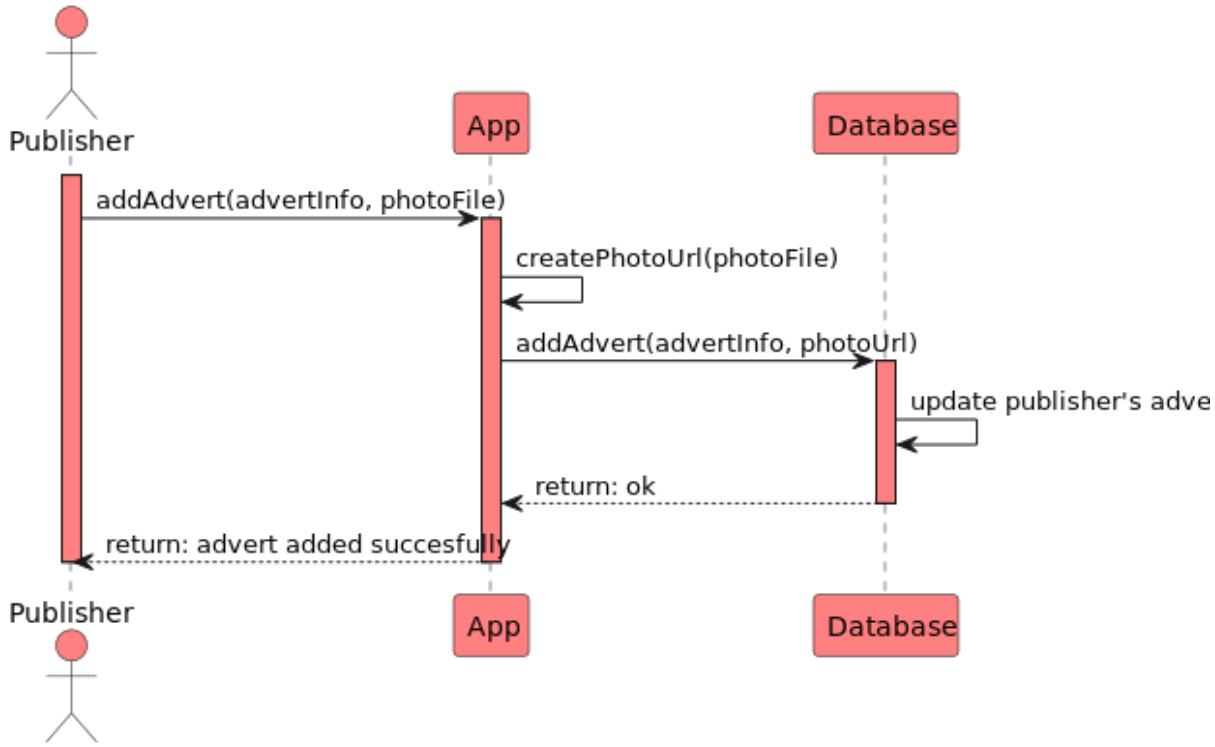
Use Case	*View advert details*
Actors	*User*
Data	*advert's detailed data => showed to user*
Stimulus	*If user wants to see details*
Response	*Detailed advert will be shown*
Comments	*Adverts normally shown with main information for better user experience*

1.3 Class Diagrams

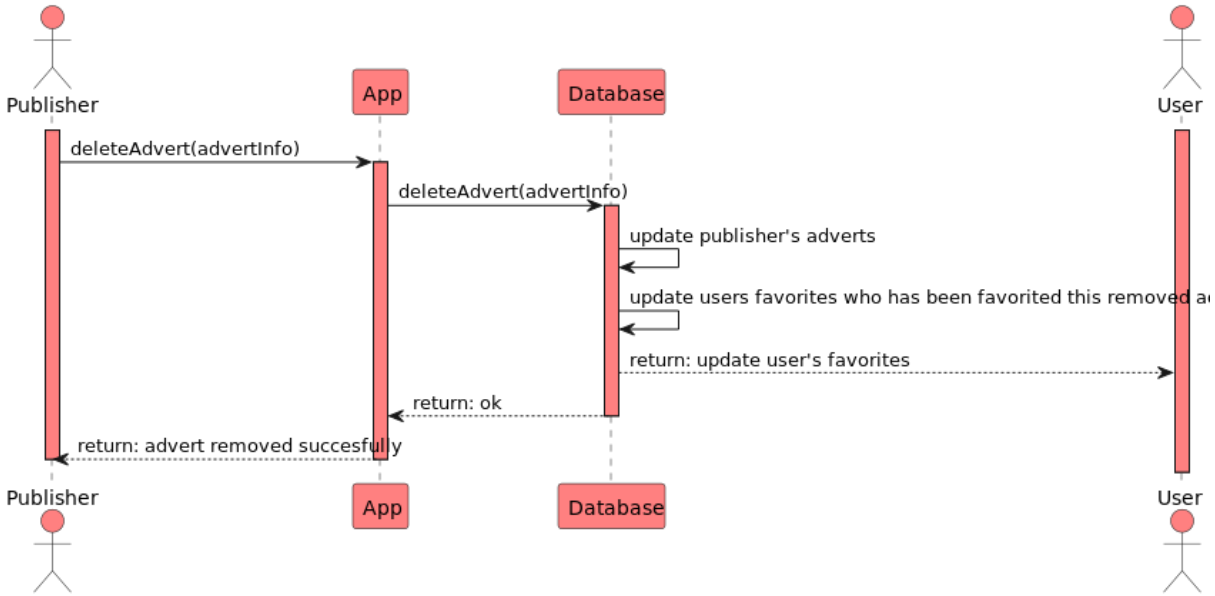


1.4 Sequence Diagrams

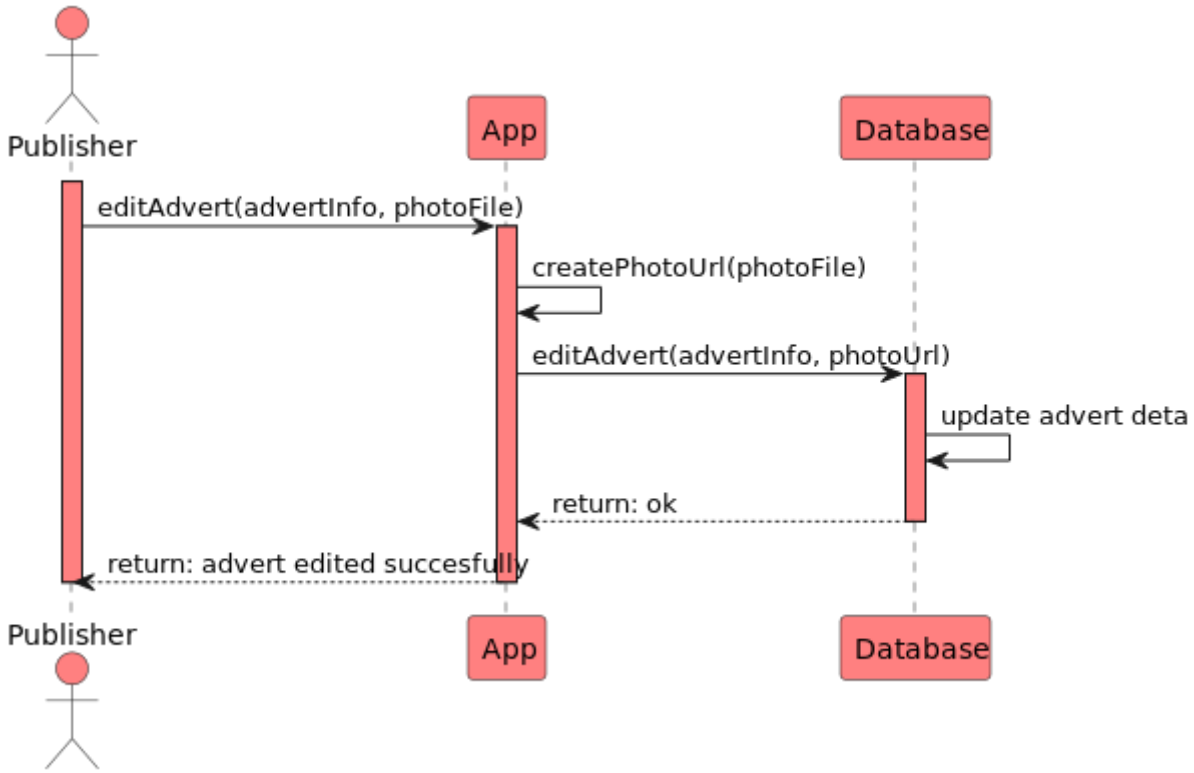
1.4.1 Add Advert - Sequence Diagram:



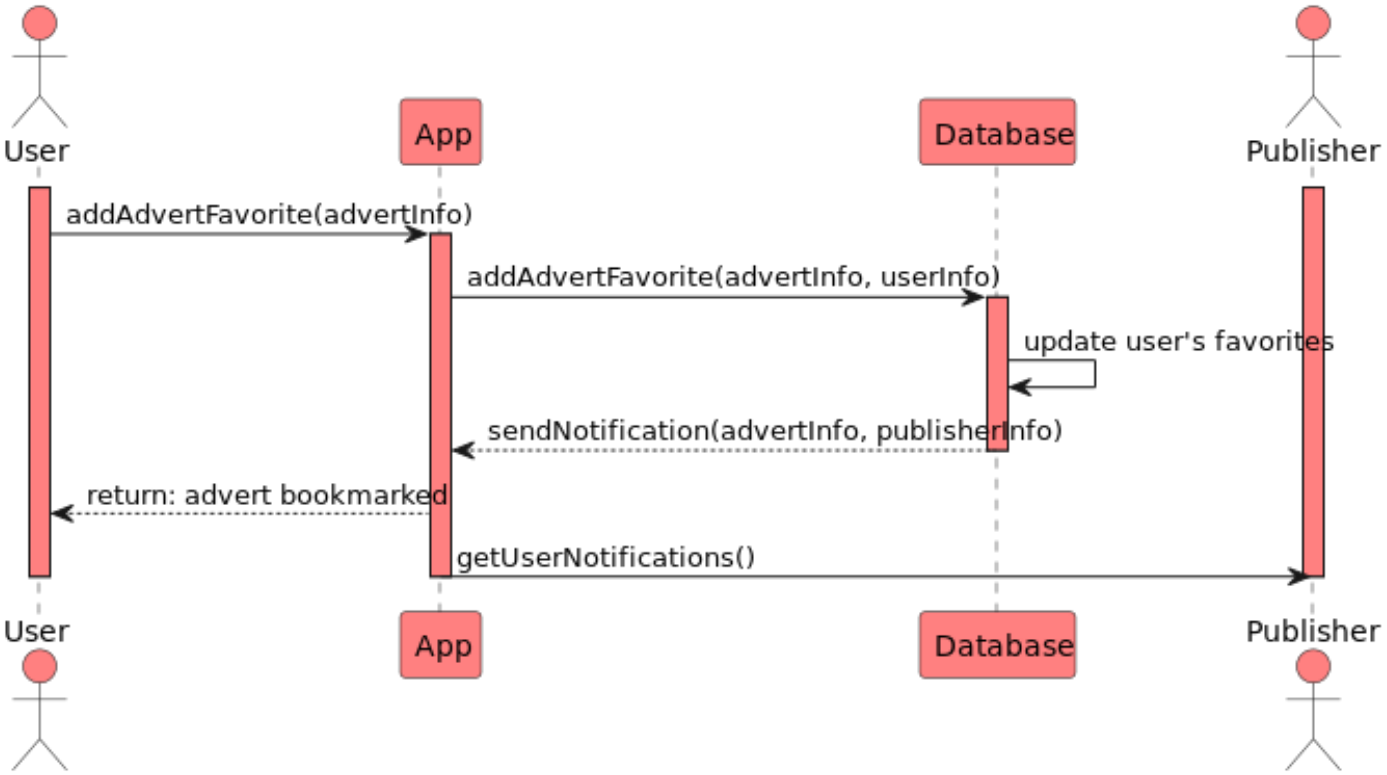
1.4.2 Remove Advert - Sequence Diagram



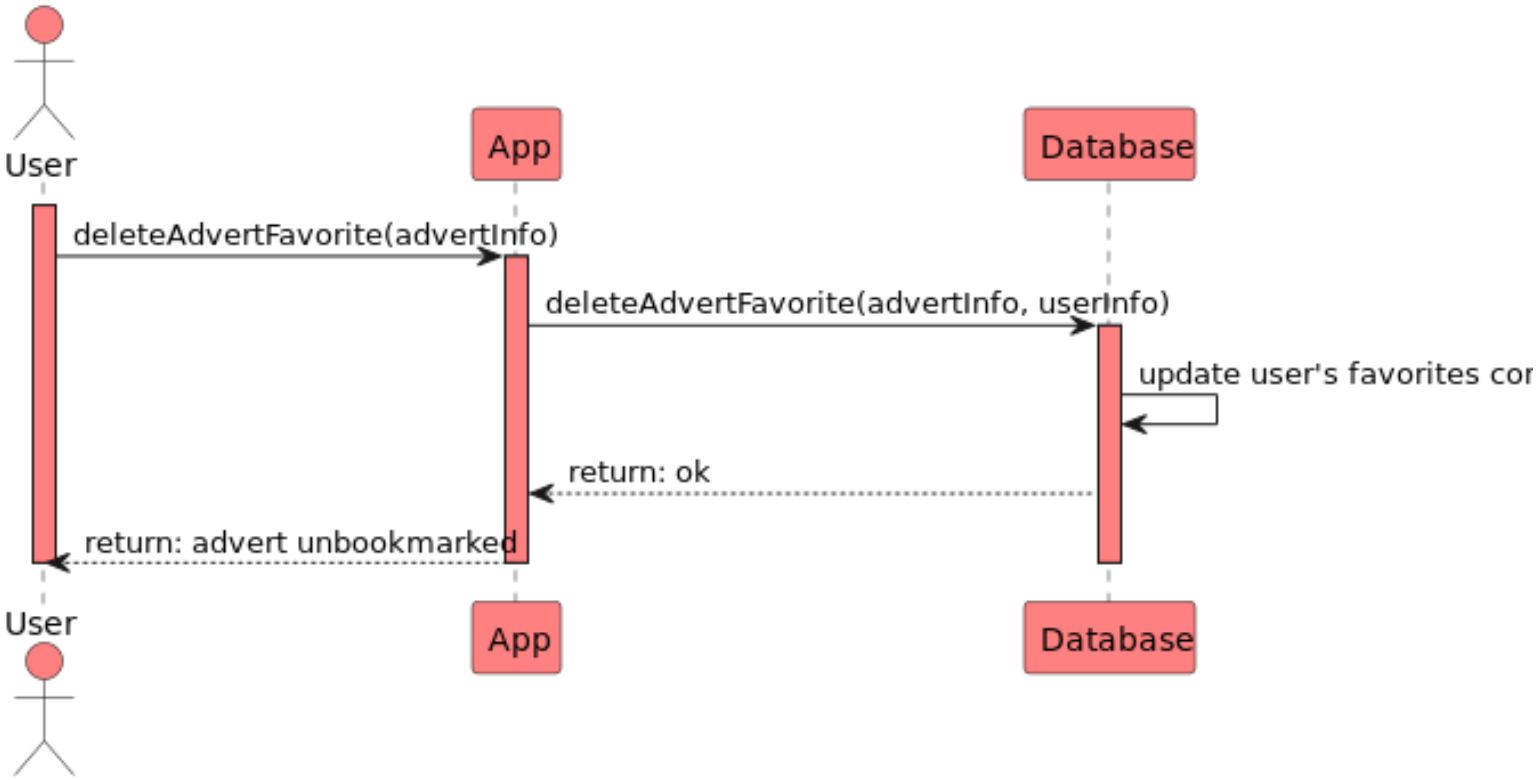
1.4.3 Edit Advert – Sequence Diagram



1.4.4 Favourite an Advert – Sequence Diagram

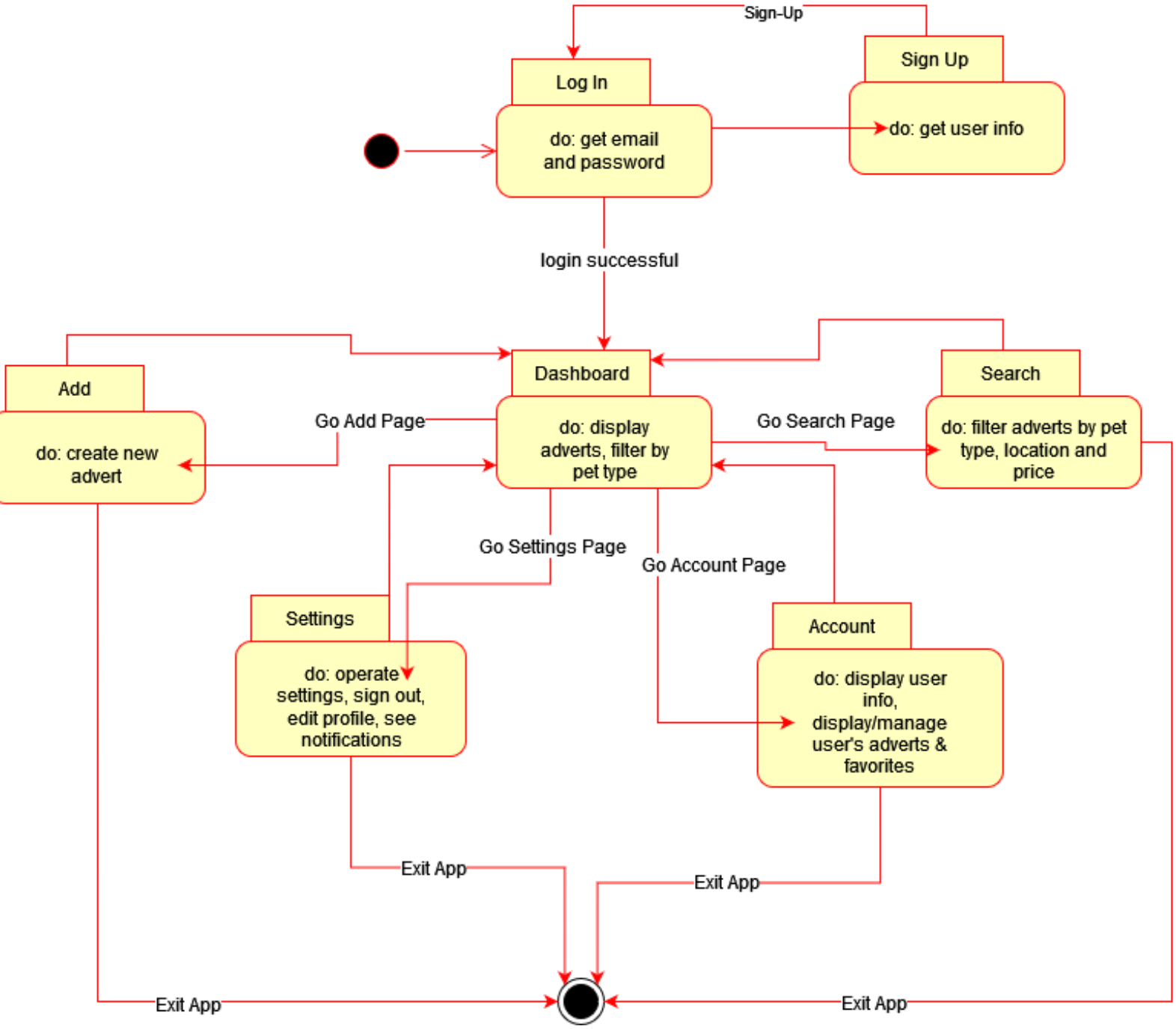


1.4.5 Unfavourite an Advert – Sequence Diagram

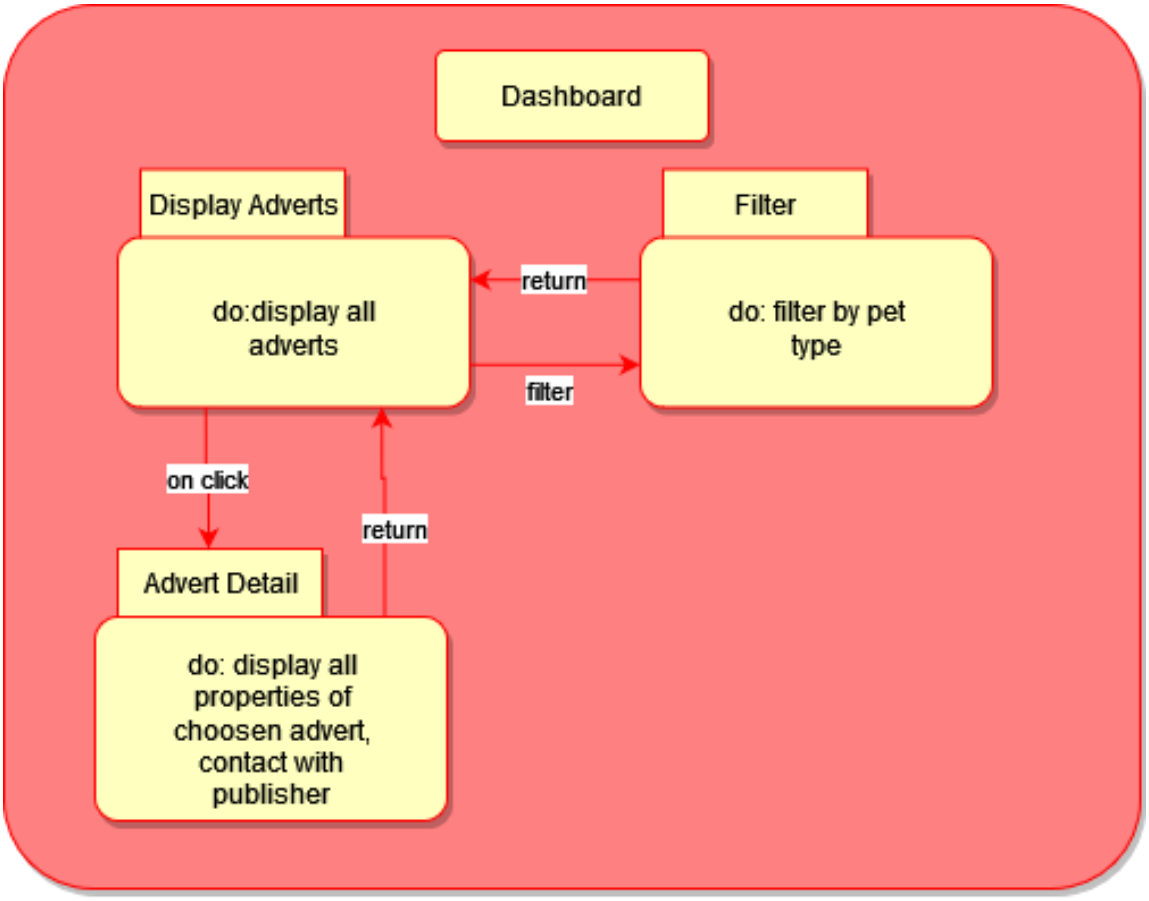


1.5 State Diagrams

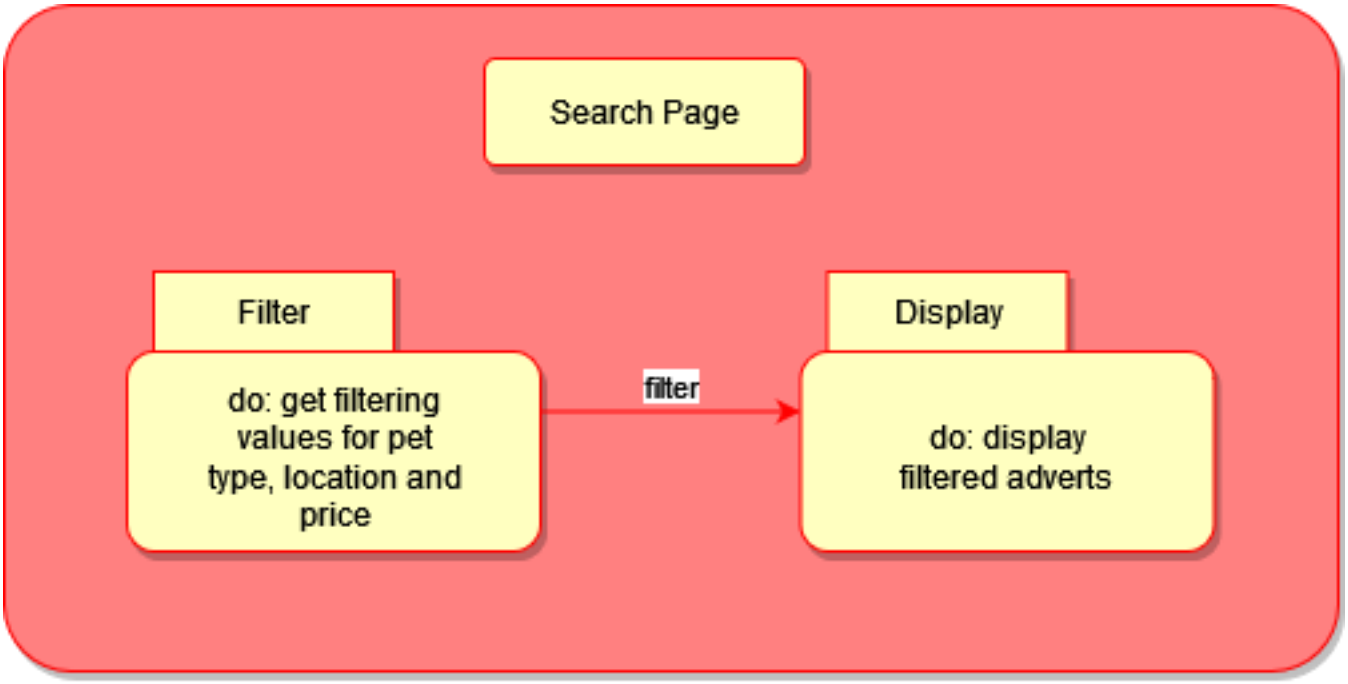
1.5.1 System State Diagram:



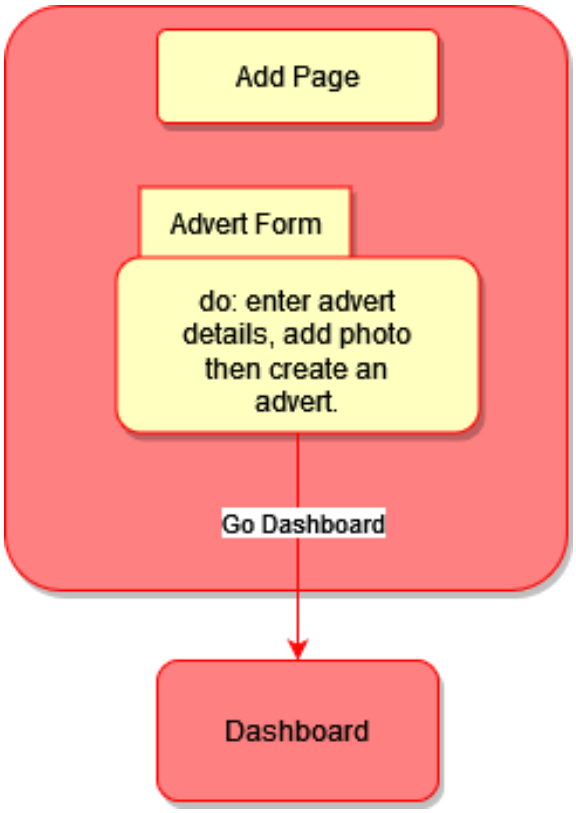
1.5.2 Dashboard State Diagram:



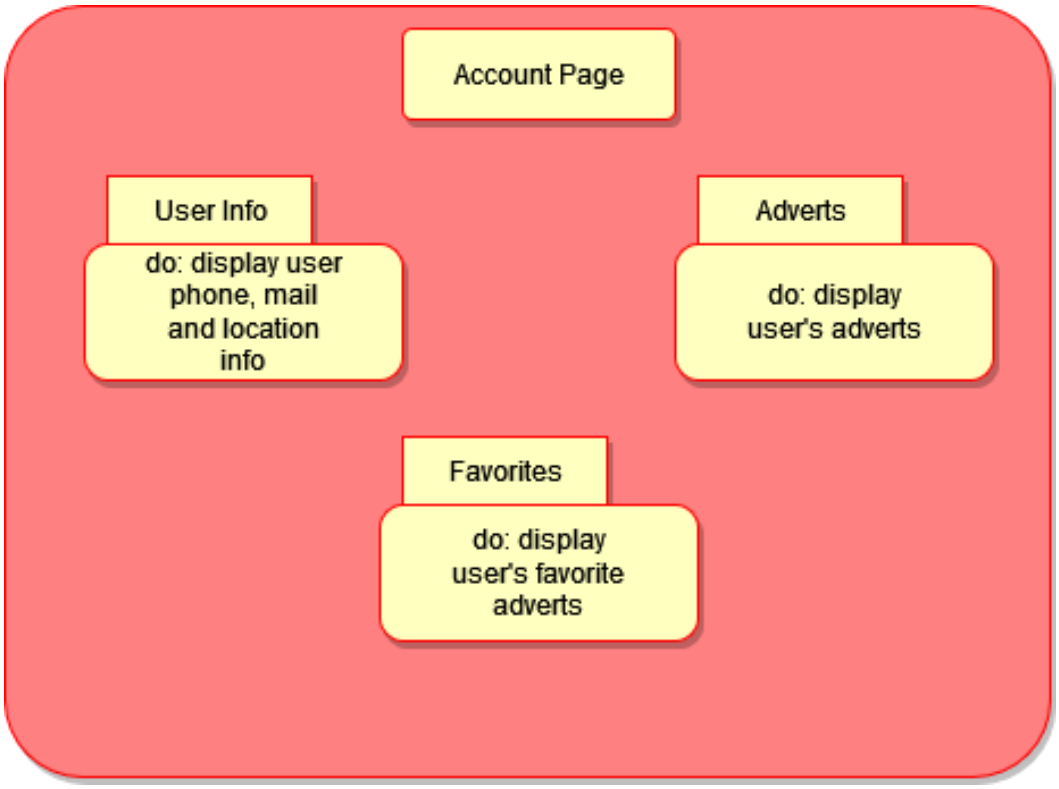
1.5.3 Search Page State Diagram:



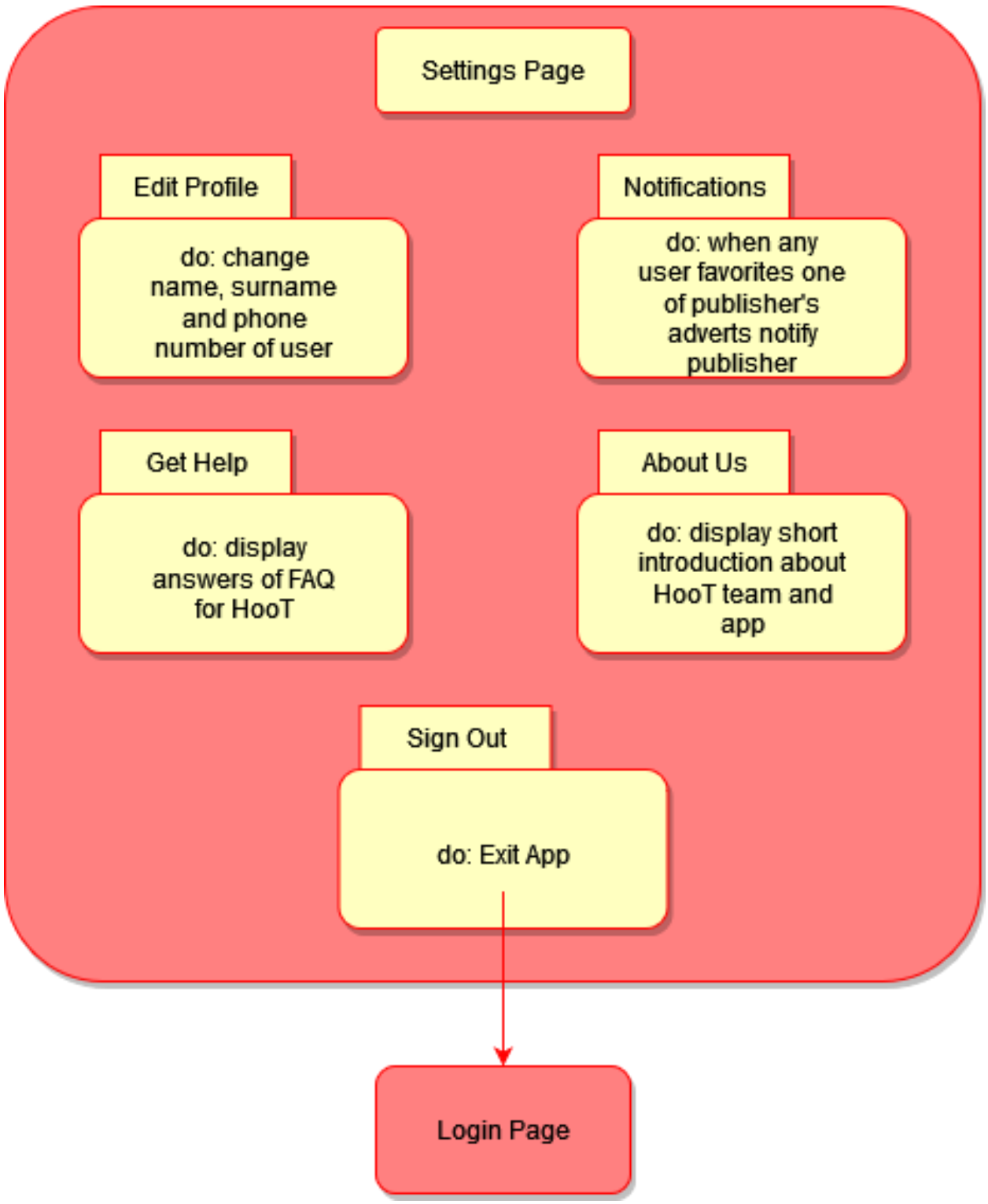
1.5.4 Add Page State Diagram:



1.5.5 Account Page State Diagram:



1.5.6 Settings Page State Diagram:



2. Simple System Architecture

Title: Hoot App (Pet Care Advertisement System)

Purpose: The purpose of the Hoot App is to provide a platform for pet owners to find and connect with suitable pet sitters in their area.

Stakeholders: Pet owners: People who own pets and are looking for a pet sitter to take care of their pets while they are away.

Pet sitters: People who offer pet sitting services to pet owners.

Admin: The administrator of the app, who is responsible for managing the system and ensuring its smooth operation.

System Components:

Mobile app: The main interface for pet owners and pet sitters to use the system.

The app will be available for download on the App Store and Google Play.

Database: A database to store information about pet owners, pet sitters, and their respective profiles and ratings.

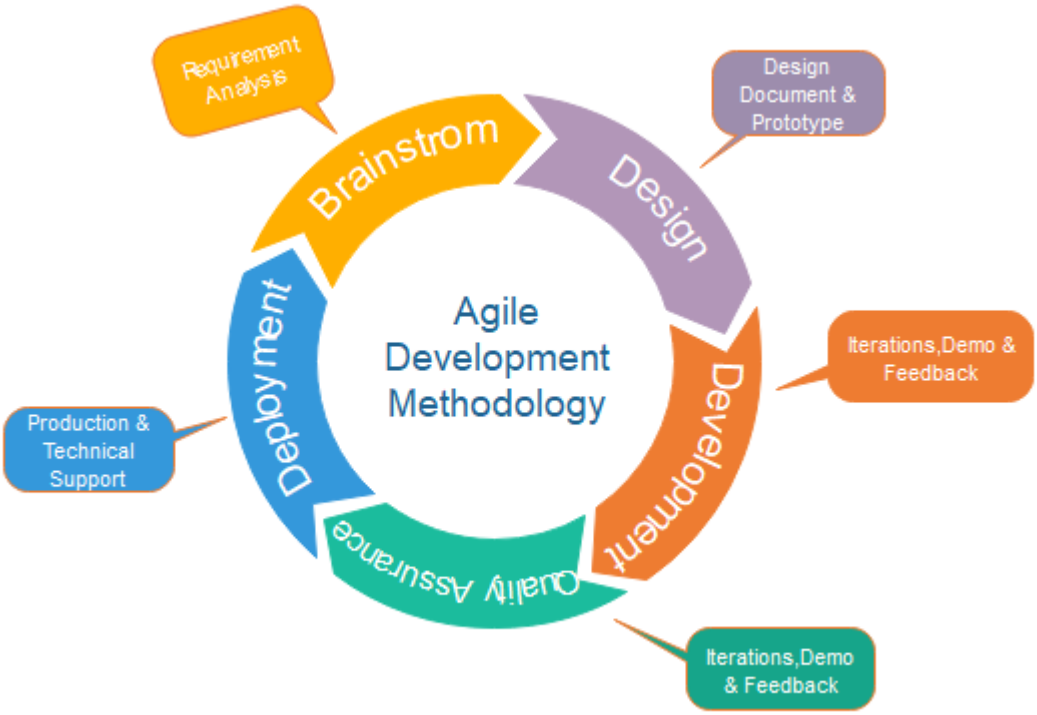
Functions:

Pet owners can create a profile, search for pet sitters in their area, and contact them through the app.

Pet sitters can create a profile, advertise their services, and provide information to pet owners through the app.

The app will match pet owners and pet sitters based on their preferences.

The payment processing system will facilitate the exchange of money between pet owners and pet sitters in their preference app doesn't provide a service for it yet.



In this project, we employed an agile development approach, which is a iterative and incremental method of developing software that emphasizes flexibility, collaboration, and rapid delivery. This approach allowed us to quickly adapt to changing requirements and prioritize the most important features for our users. By working in short, iterative cycles, we were able to deliver functional software early and often, which enabled us to gather valuable feedback and make iterative improvements throughout the project. The agile development approach also facilitated close collaboration between our team members, which enabled us to make efficient use of our resources and effectively address any challenges that arose. Overall, the agile development approach proved to be a highly effective way to develop software in a dynamic and fast-paced environment.

Use Case View

The audience for this document includes all stakeholders of the system, including end-users. It covers the significant functionality of the system and describes the actors and use cases. The purpose of this view is to understand the needs of the user and to further elaborate on the flows and constraints at the design level. The document should focus on understandability and usability. Related artifacts may include the use-case model and use-case documents.

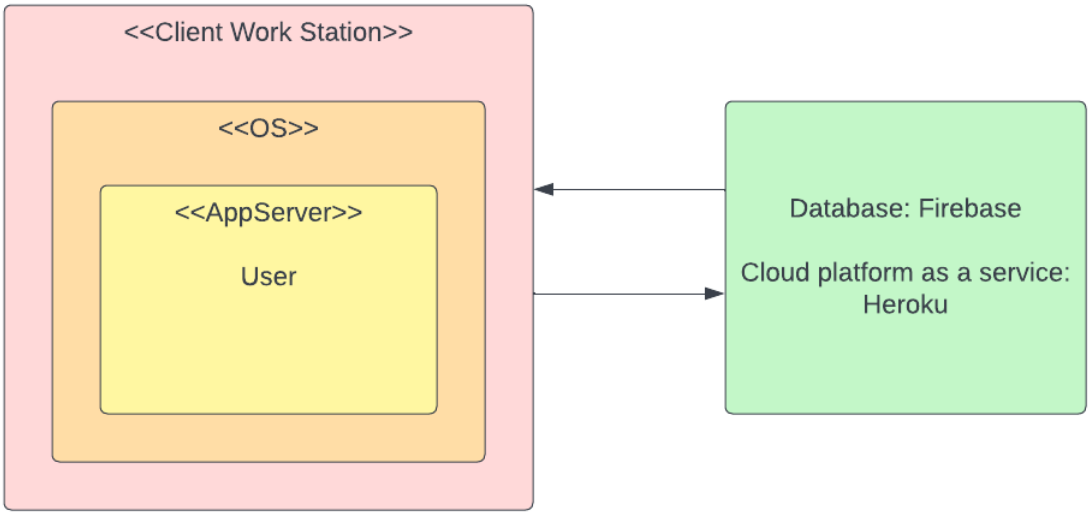
Click on this and you can jump into Use case diagrams.

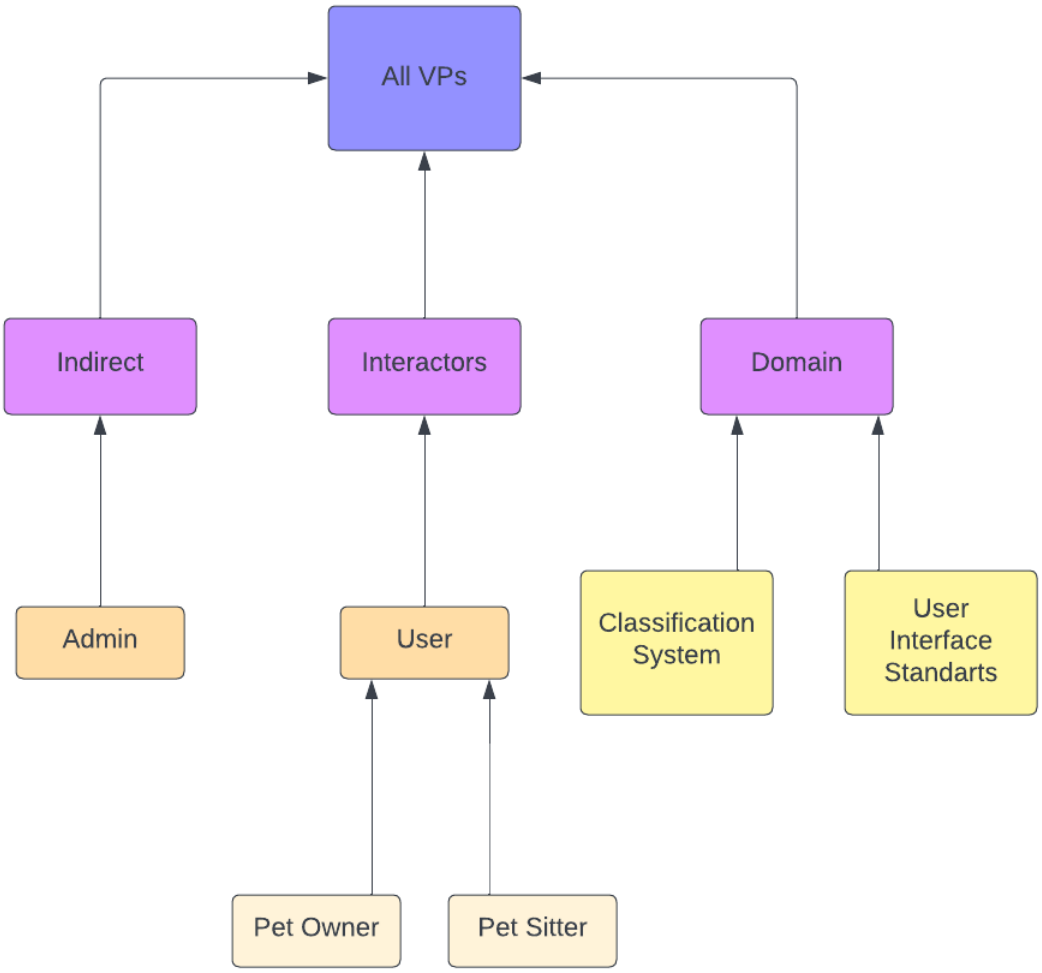
Logical View

The Hoot app is built on a front-end and real-time database architecture, which enables users to interact with the app and make changes to the database in real-time. The front-end of the app is responsible for the user interface and the presentation of data to the user. It is built using a combination Flutter and JavaScript, and is designed to be intuitive and responsive. The real-time database is a cloud-based system that stores and manages the data for the app. It is designed to be highly scalable and able to handle a large volume of data and concurrent users. Changes made to the database through the front-end are immediately reflected in the app, providing a seamless and responsive user experience. Overall, the front-end and real-time database architecture of the Hoot app enables users to easily and efficiently interact with and manage the data within the app.

Deployment View

The audience for this document includes Admins of the system. The system architecture document outlines the hardware and deployment structures for the software, including known and anticipated scenarios. It helps implementers optimize the systeam for its intended use.

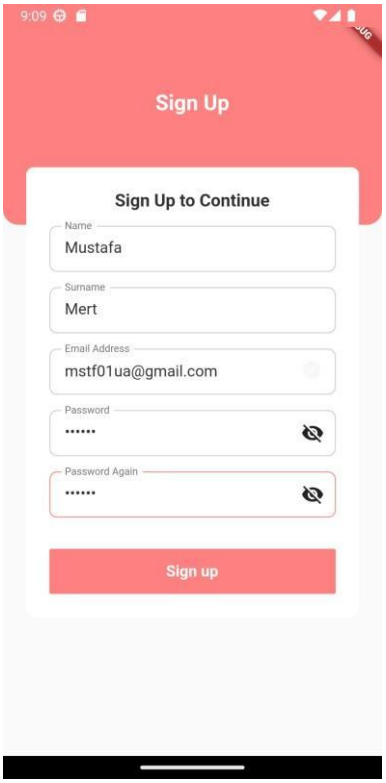


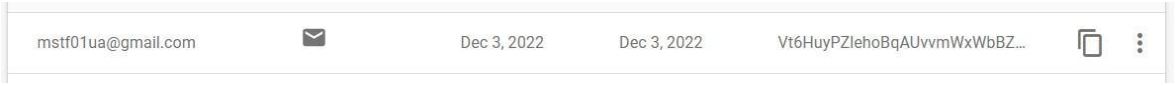


Overall, Pet owners and pet sitters can create profiles on the system with their consent. Pet owners can view the profiles of pet sitters and mark the ones they like as favorites. The system handles the favorite system and sends notifications to the pet sitters when they are marked as favorites by a pet owner. The system works with a database to store and manage this information, and the admin can edit both the pet care advertisement system and the database, as well as view the database.

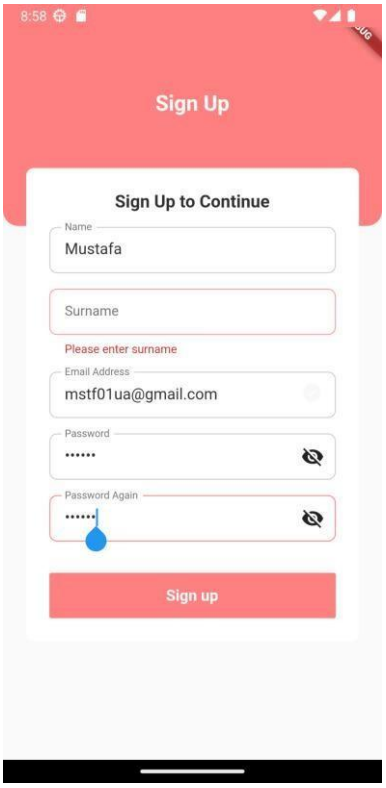
3. Test Reports

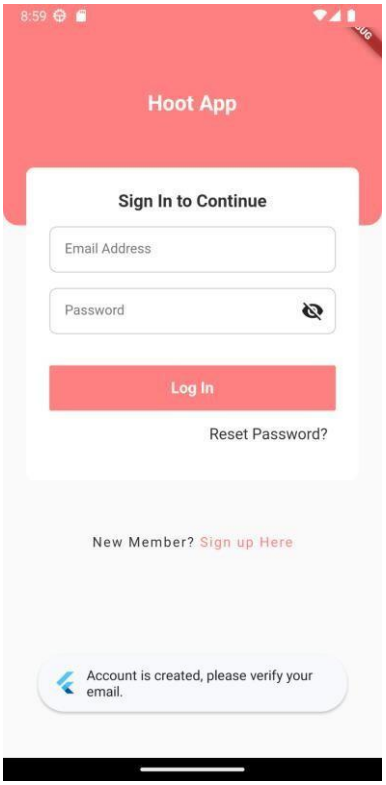
Test Case (Signing Up Successfully)	
Description: An unregistered user should be able to successfully sign up.	**Test Steps:** 1. Click ‘Sign Up' text 2. Enter valid email address 3. Enter at least 6 character password 4. Enter name 5. Enter surname 6. Click sign-up button

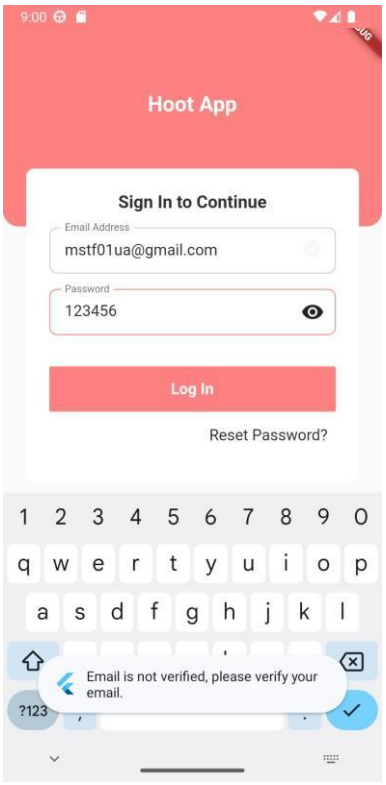


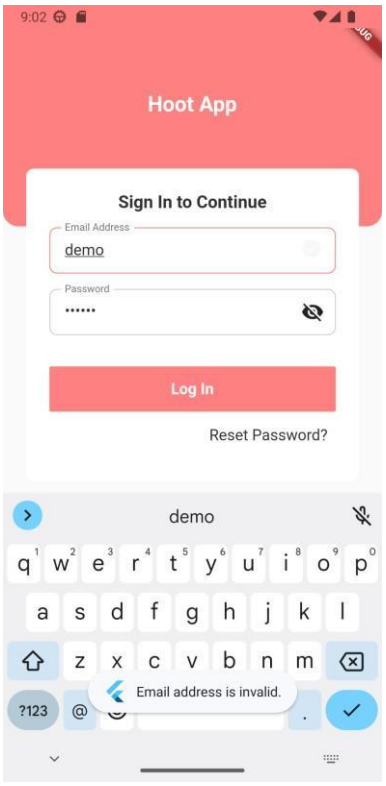


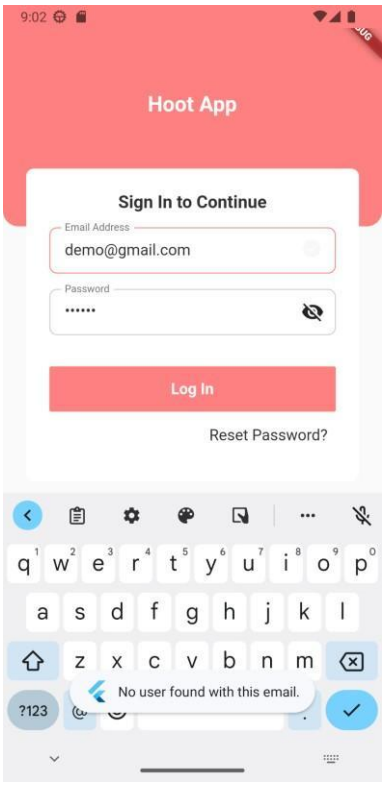
Result: *Sign up successfully*

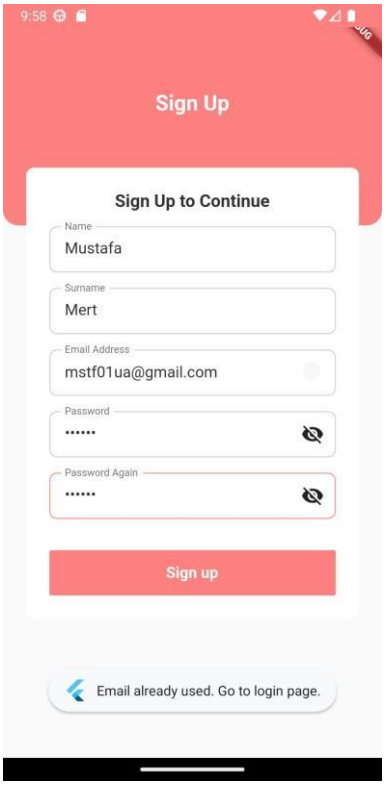
Test Case (Signing Up with Missing Information)	
Description: An unregistered user should be able to unsuccessful sign up.	**Test Steps:** 1. Click 'Register Now' text 2. Enter valid email address 3. Enter at least 6 character password 4. Enter name 5. Enter empty surname 6. Click sign-up button
	
Result: *Sign up unsuccessfully*	

Test Case (Login Successfully)	
Description: An unverified user should not be able to successfully sign up.	**Test Steps:** 1. Click 'Register Now' text 2. Enter valid email address 3. Enter at least 6 character password 4. Enter name 5. Enter surname 6. Click Login button
	
Result: *Unsuccessful Login*	

Test Case (Login Unsuccessful)	
Description: An user with unverified email should not successfully login to the app	**Test Steps:** 1. Enter unverified email address 2. Enter at least 6 character password 3. Click the Login button
	
Result: *Unsuccessful Login*	

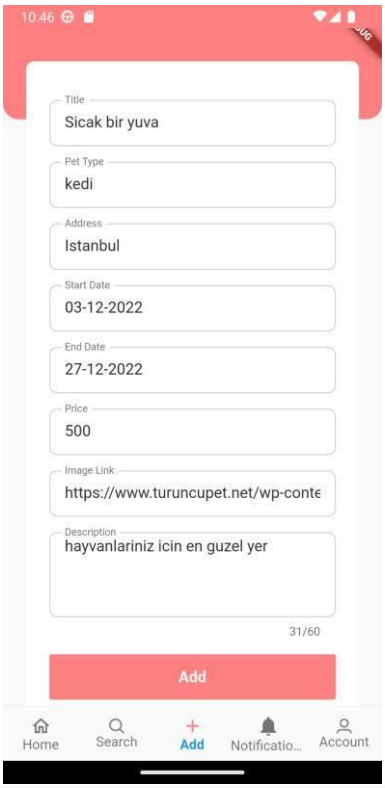
Test Case (Login Unsuccessful)	
Description: Unregistered user should not be able to login.	**Test Steps:** 1. Enter invalid email address 2. Enter match password 3. Click login button
	
Result: *Login unsuccessfully*	

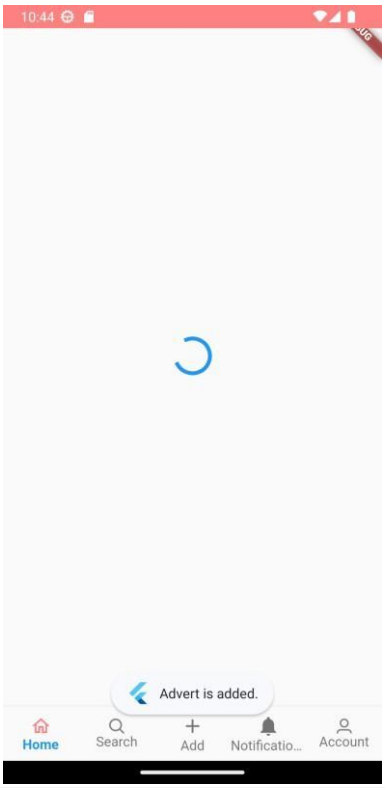
Test Case (Login Successfully)	
Description: An registered user should be able to successfully login	**Test Steps:** 1. Enter valid email address 2. Enter match password 3. Click login button
	
Result: *Login successfully*	

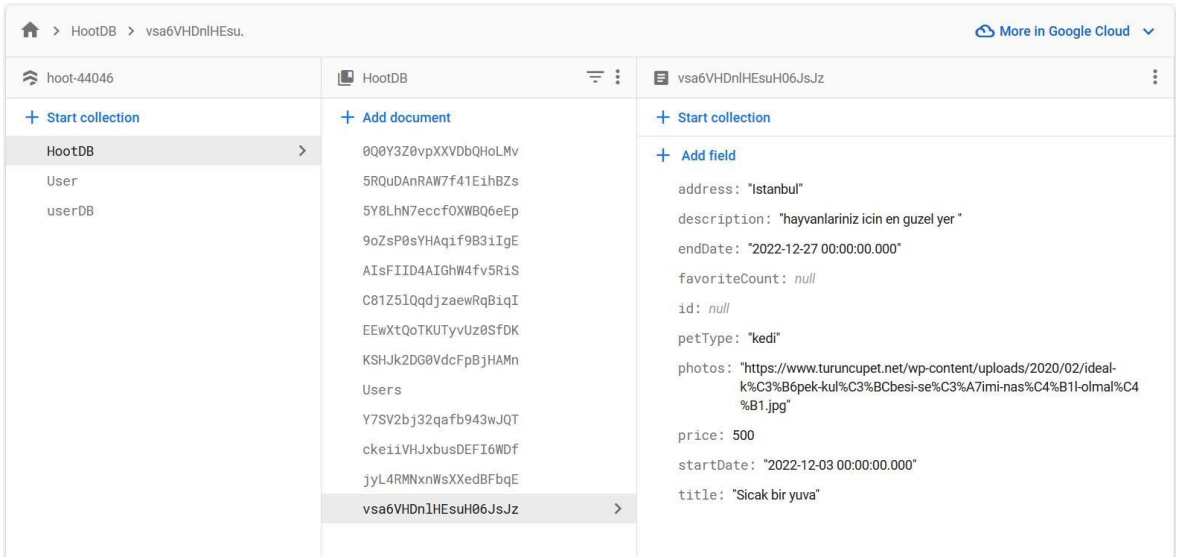
Test Case (Signing Up Unsuccessfully)	
Description: An unregistered user should be able to unsuccessfully sign up	**Test Steps:** 1. Click ‘Sign Up' text 2. Enter valid email address 3. Enter less than 6 character password 4. Enter name 5. Enter surname 6. Click sign-up button
	
Result: *Sign up unsuccessfully*	

Test Case (Adding Advert Successful)

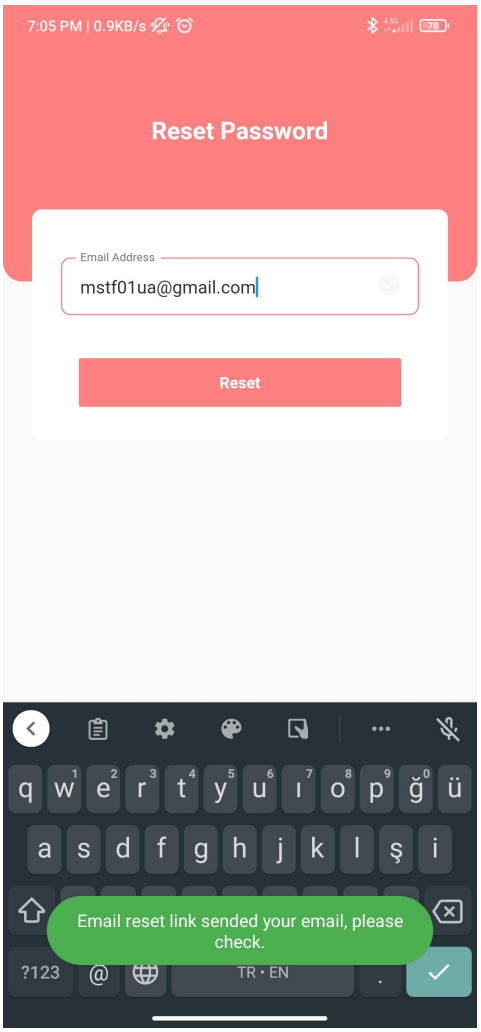
Description: A registered user can add an advert.	**Test Steps:** Login successfully Click '+' button in button bar Choose and enter all properties correctly Click Add





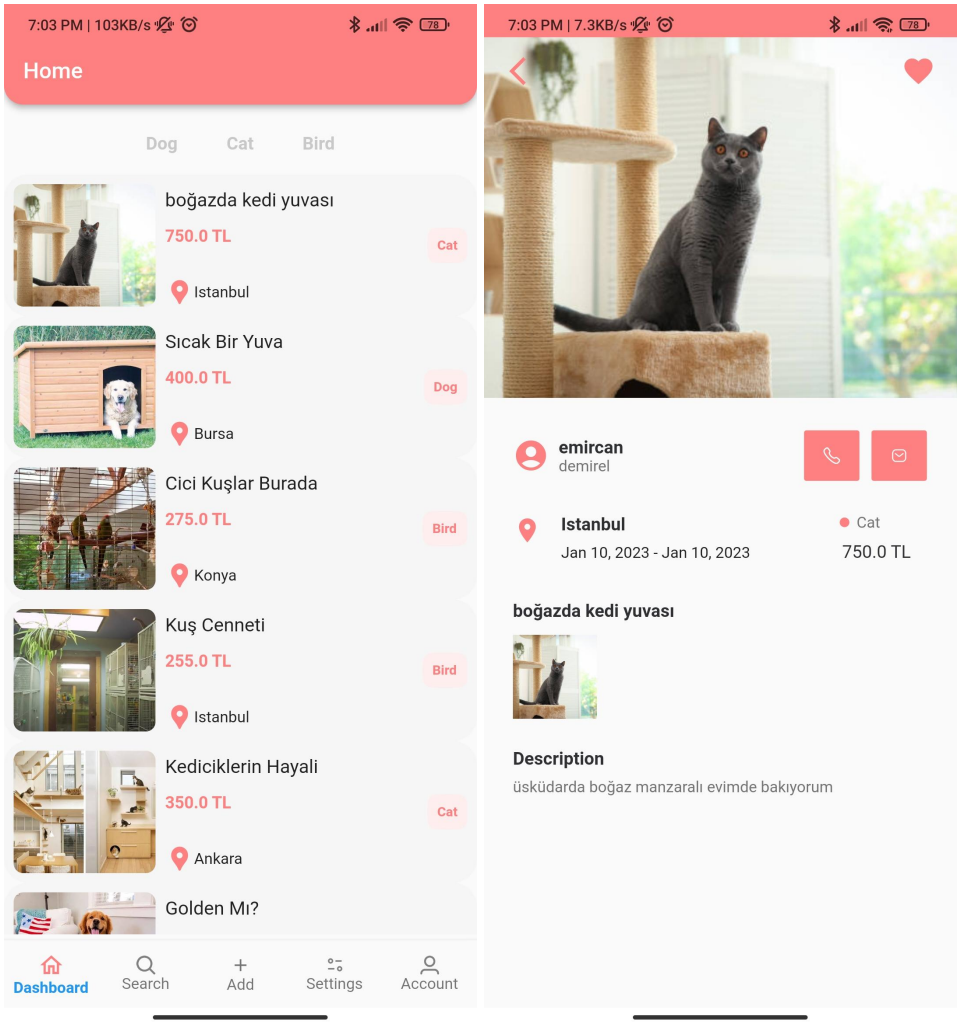


Result: *Advert added*

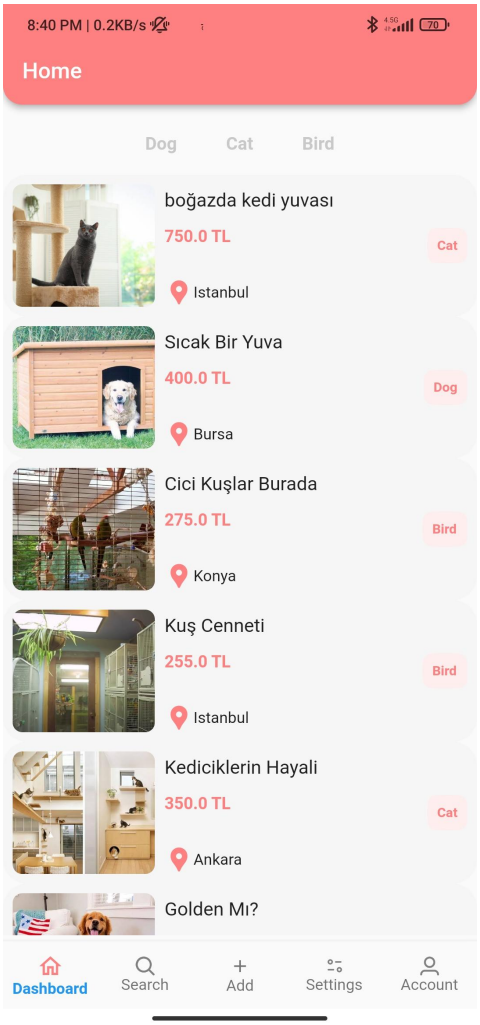
Test Case (Password Reset Successfully)	
Description:Regenerates the previously registered user password.	**Test Steps:** 1. The application opens 2. Press reset my password 3. Write the e-mail address 4. The password is reset with the incoming connection.
	
Result: Password regenerated	

Test Case (Showing Advert Detail)

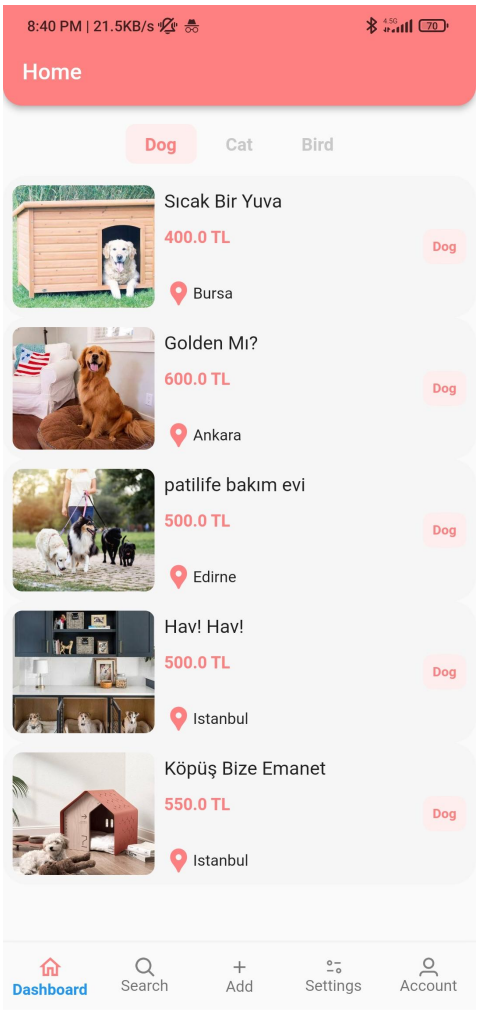
Description: Viewing the details by clicking on the advertisement selected by the logged in user.	**Test Steps:** 1. Login to the application. 2. An ad is selected from the homepage

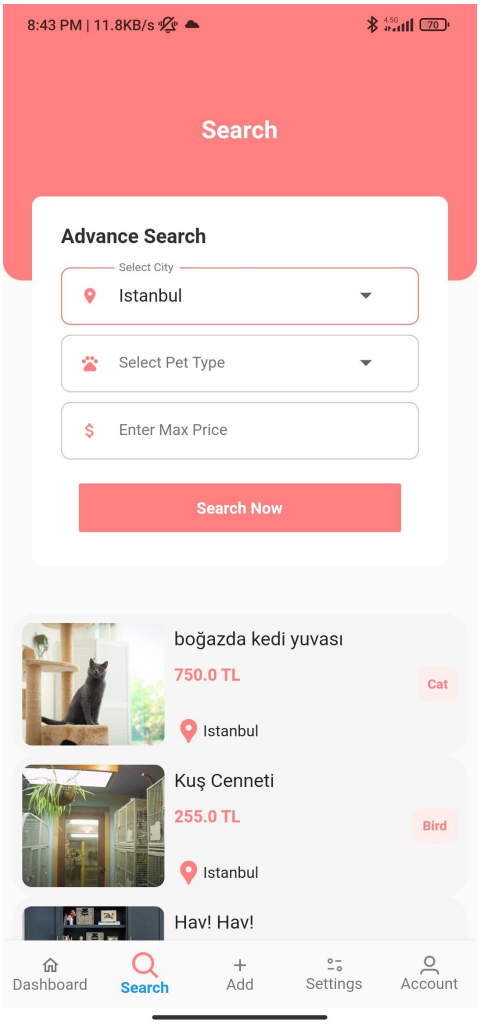


Result: App details viewed

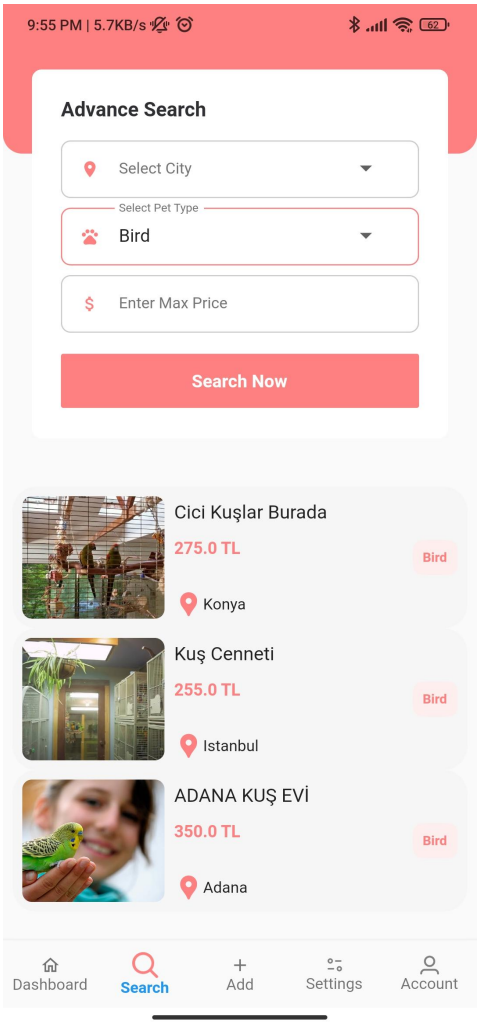
Test Case (Viewing All Adverts)	
Description: All adverts are displayed	**Test Steps:** 1. Login to the application. 2. The homepage is displayed.
	
Result: All ads have been viewed.	

Test Case (Filtering Adverts)

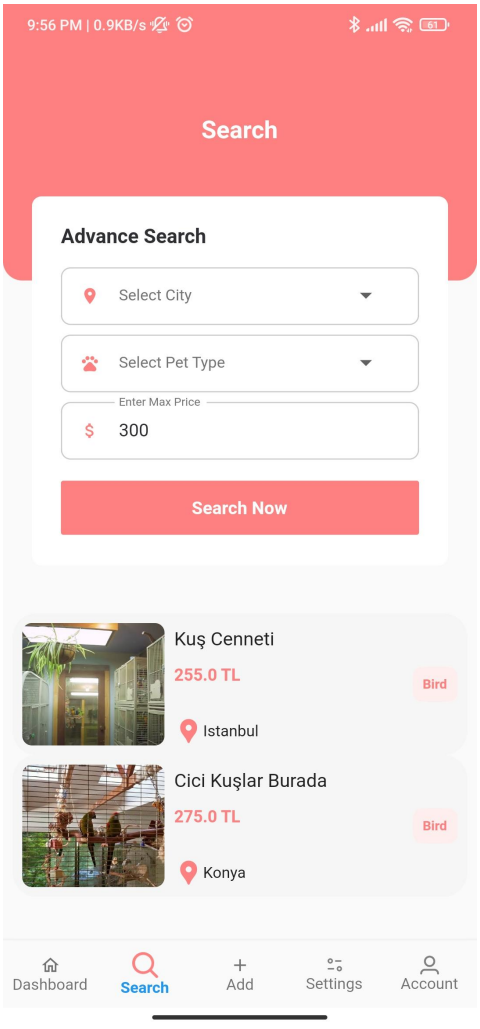
Description: Ads are filtered according to the selected animal type.	**Test Steps:** 1. Login to the application. 2. Select one of the animal species on the homepage.
	
Result: Ads have been filtered.	

Test Case (Searching by City)	
Description: Advertisements are filtered according to city information.	**Test Steps:** 1. Login to the application. 2. Go to the Search section 3. Select the city from the City section. 4. Press the Search button.
	
Result: Ads are filtered by city information.	

Test Case (Searching by Pet Type)	
Description: Advertisements are filtered according to pet type information.	**Test Steps:** 1. Login to the application. 2. Go to the Search section 3.The desired type is selected from the Type section. 4. Press the Search button.



Result: Ads are filtered by pet type information.

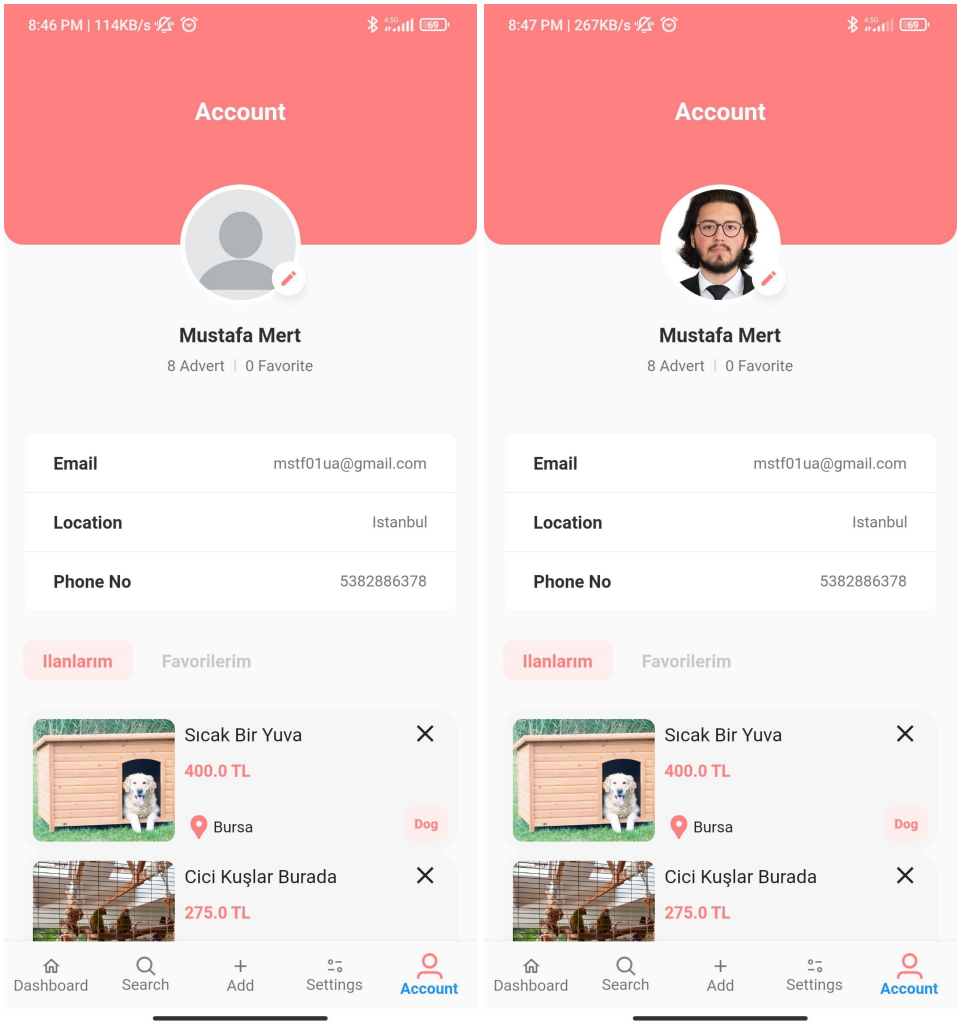
Test Case (Searching by Max Price)	
Description: Advertisements are filtered according to max price information.	**Test Steps:** 1. Login to the application. 2. Go to the Search section 3. The desired value is entered in the max price section. 4. Press the Search button.
	
Result: Ads are filtered by max price information.	

Test Case (Changing Profile Photo)

Description: The user changes the profile photo.

Test Steps:

1. Login to the application.
2. Go to the Account section.
3. Click on the pencil icon and select the new photo.



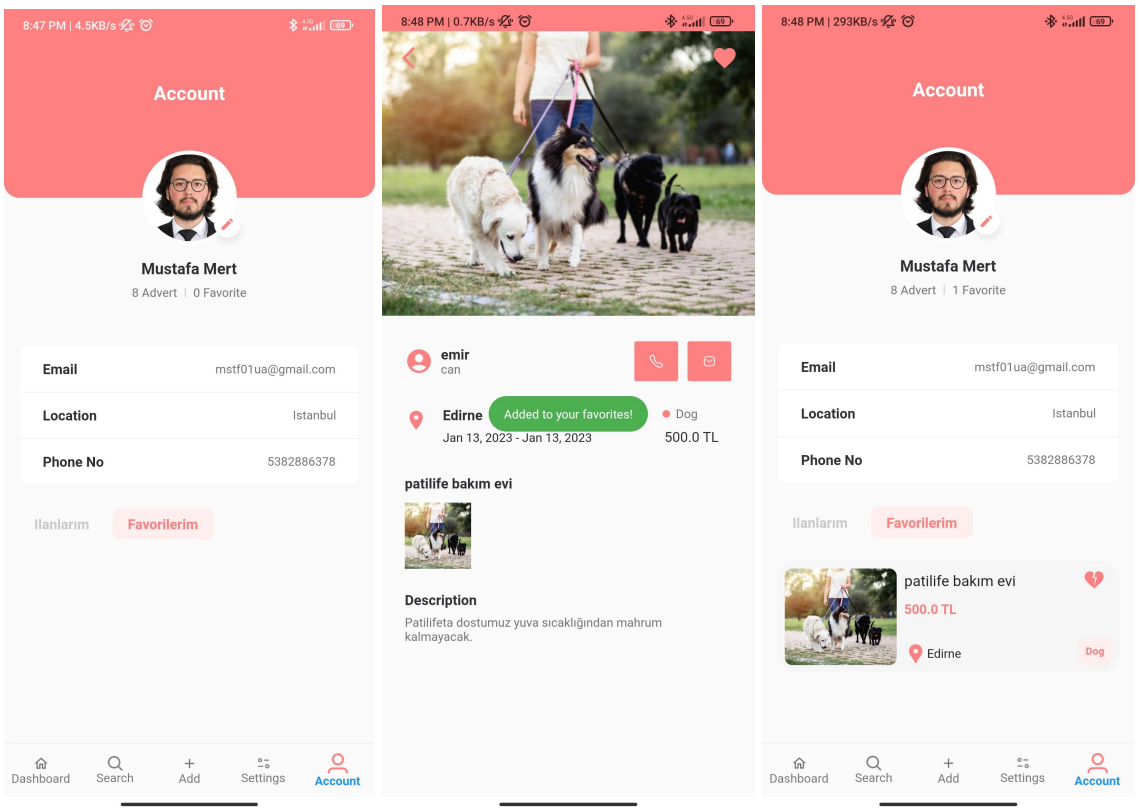
Result: User profile photo has been changed.

Test Case (Add the advert to favorites)

Description: Ads added to favorites are saved in the user's favorites tab.

Test Steps:

1. Login to the application.
2. Go to the details of the advertisement by selecting an advertisement.
3. It is added to favorites by pressing the heart icon.
4. It is displayed in the favorites section of the Account section.



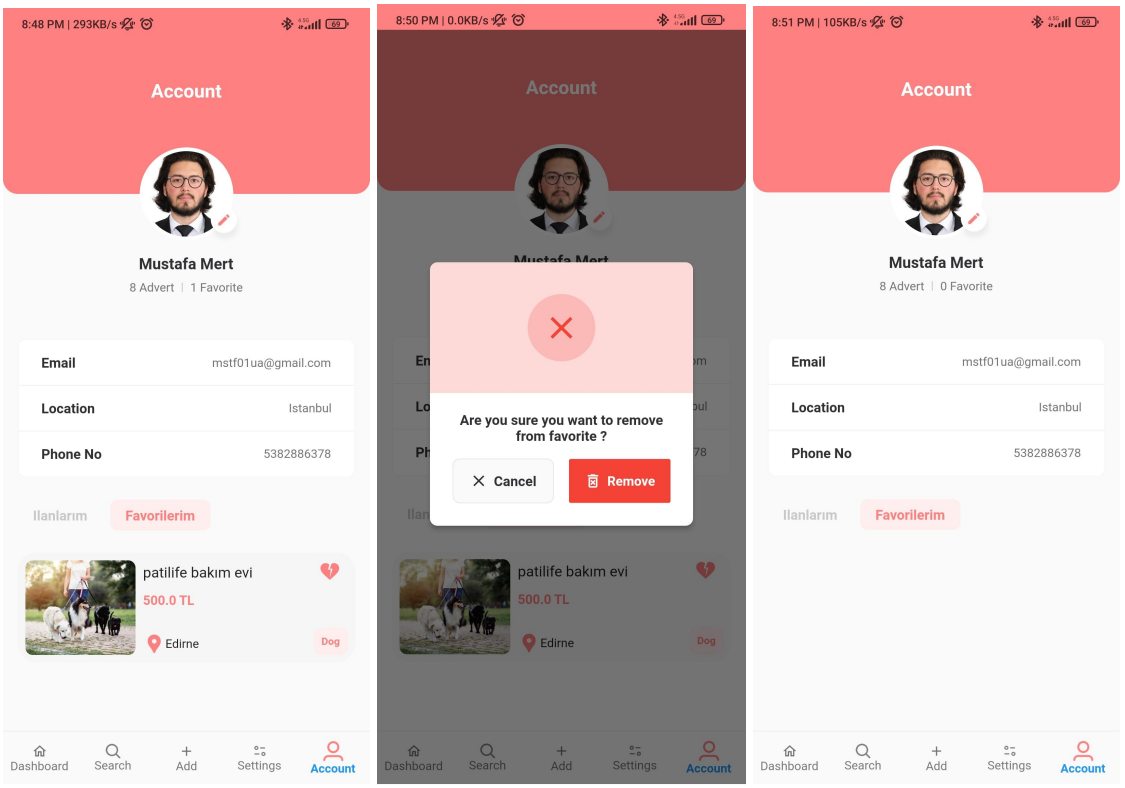
Result: The ad has been added to favorites and viewed on the profile.

Test Case (Delete the advert from favorites)

Description: Ads in favorites are removed from favorites.

Test Steps:

1. Login to the application
2. Go to the favorites section in the Account tab.
3. Click on the broken heart icon to remove the ad from favorites.



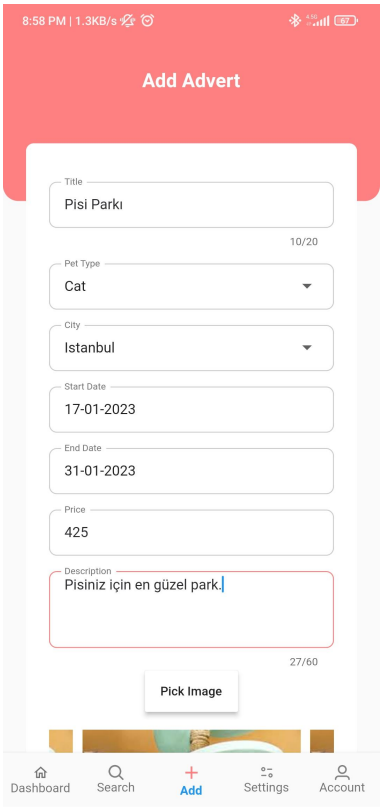
Result: Ad removed from favorites.

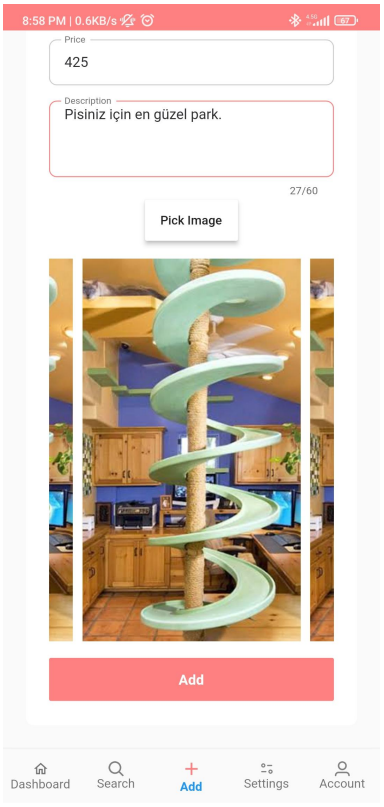
Test Case (Create new advert)

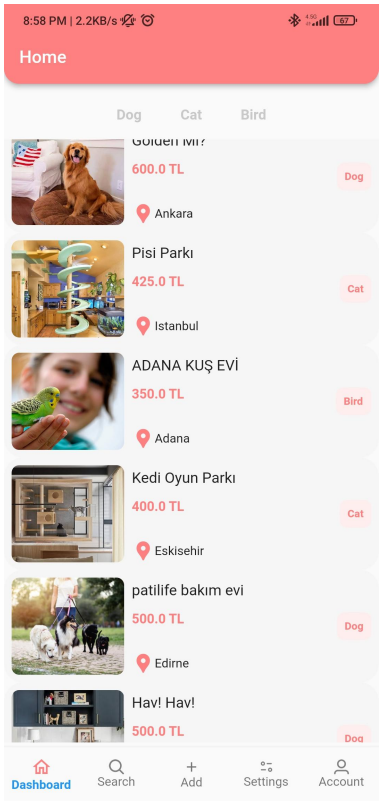
Description:A new advert is created.

Test Steps:

1. Login to the application.
The
2.Add section opens.
3. Required information is entered.
4. Click the Add button.



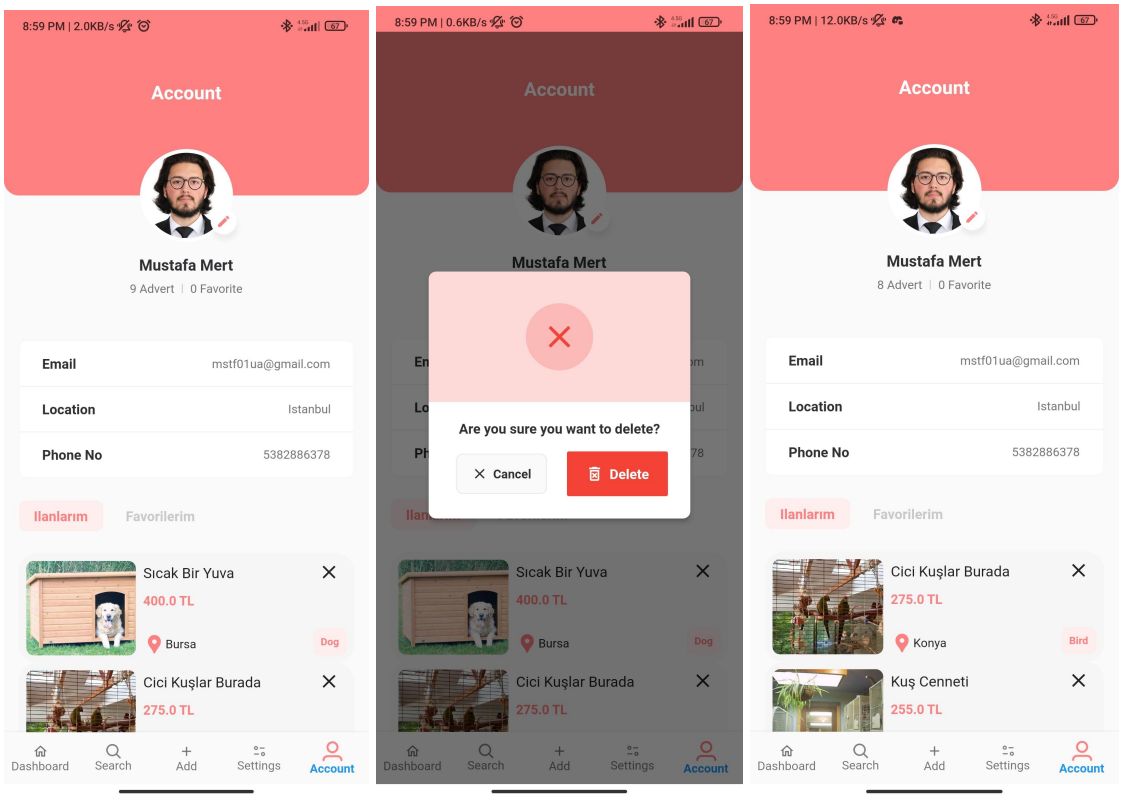




Result: A new ad has been posted.

Test Case (Delete advert)

Description: The advert is removed.	**Test Steps:** 1. Login to the application. 2. Go to the classifieds section in the Account section. 3. Click on the X icon of the advertisement that you want to remove.



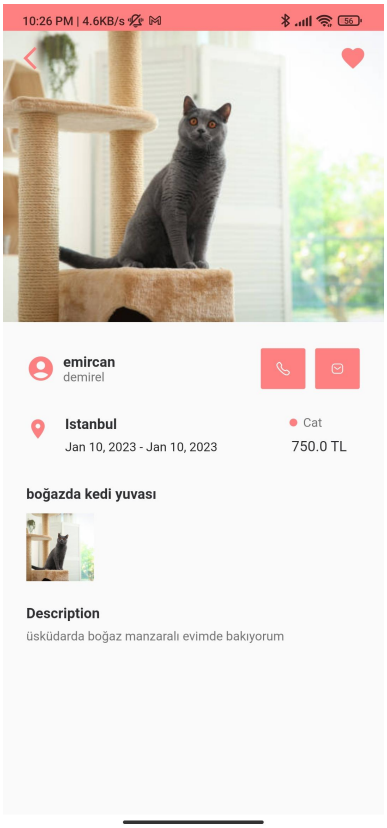
Result: Advert has been removed successfully.

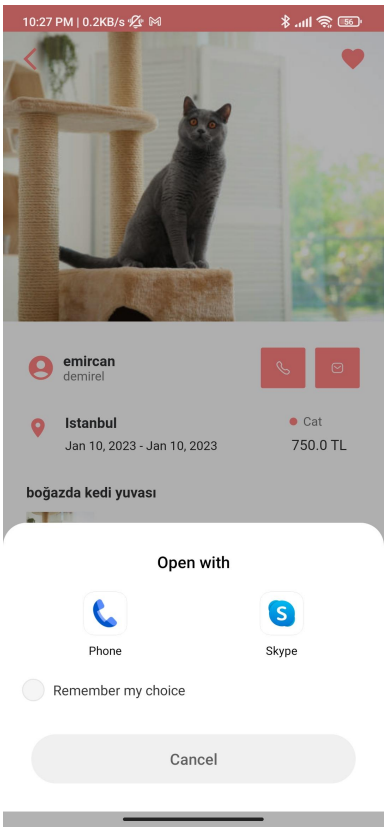
Test Case (Calling the selected advertiser)

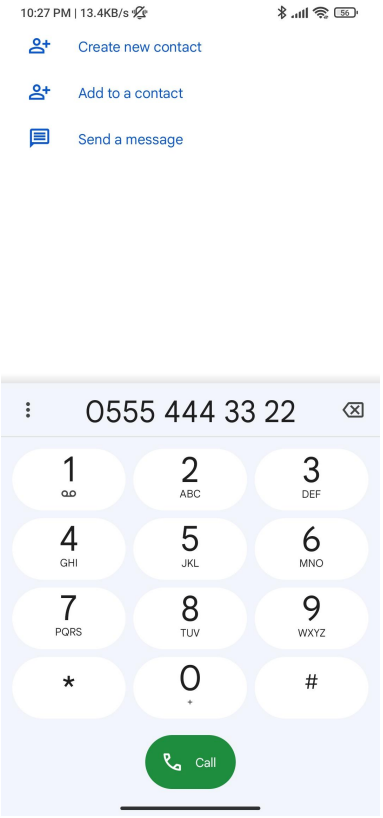
Description: The selected ad owner is called with his phone number.

Test Steps:

1. Login to the application.
2. Announcement is selected.
3. Click on the phone icon.
4. The output number is dialed.



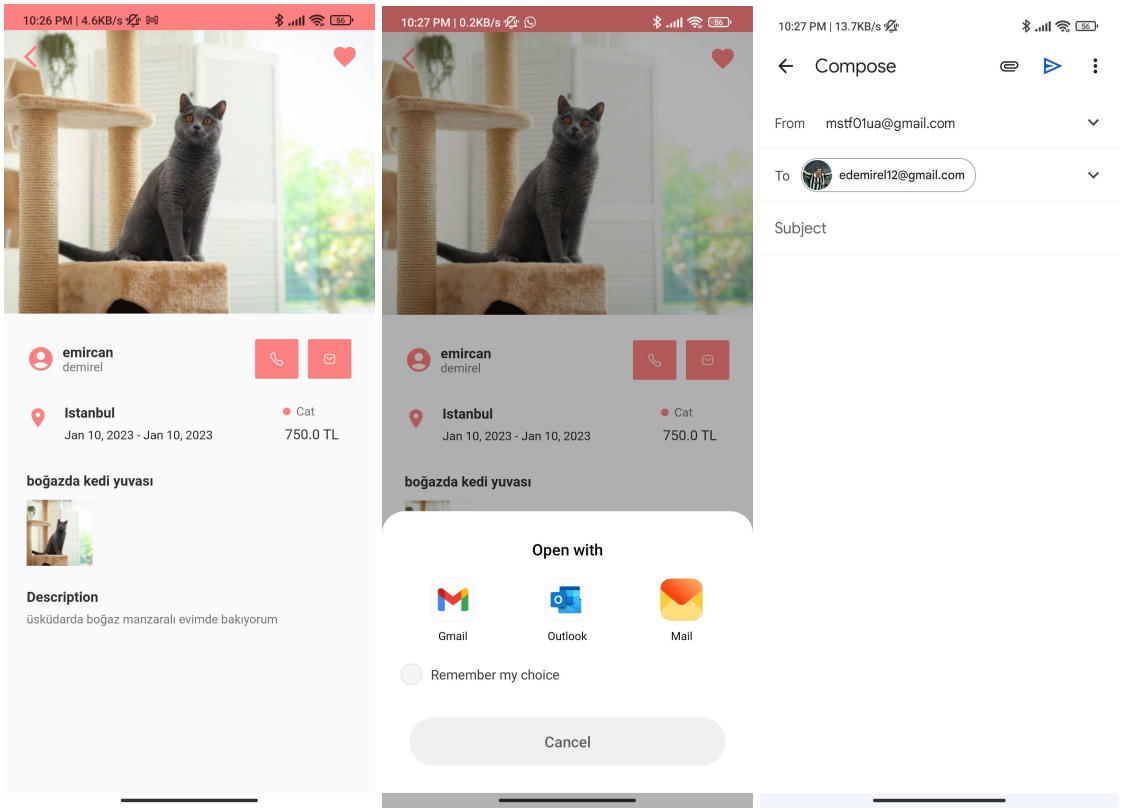




Result: Advertiser's number was called.

Test Case (Send e-mail the selected advertiser)

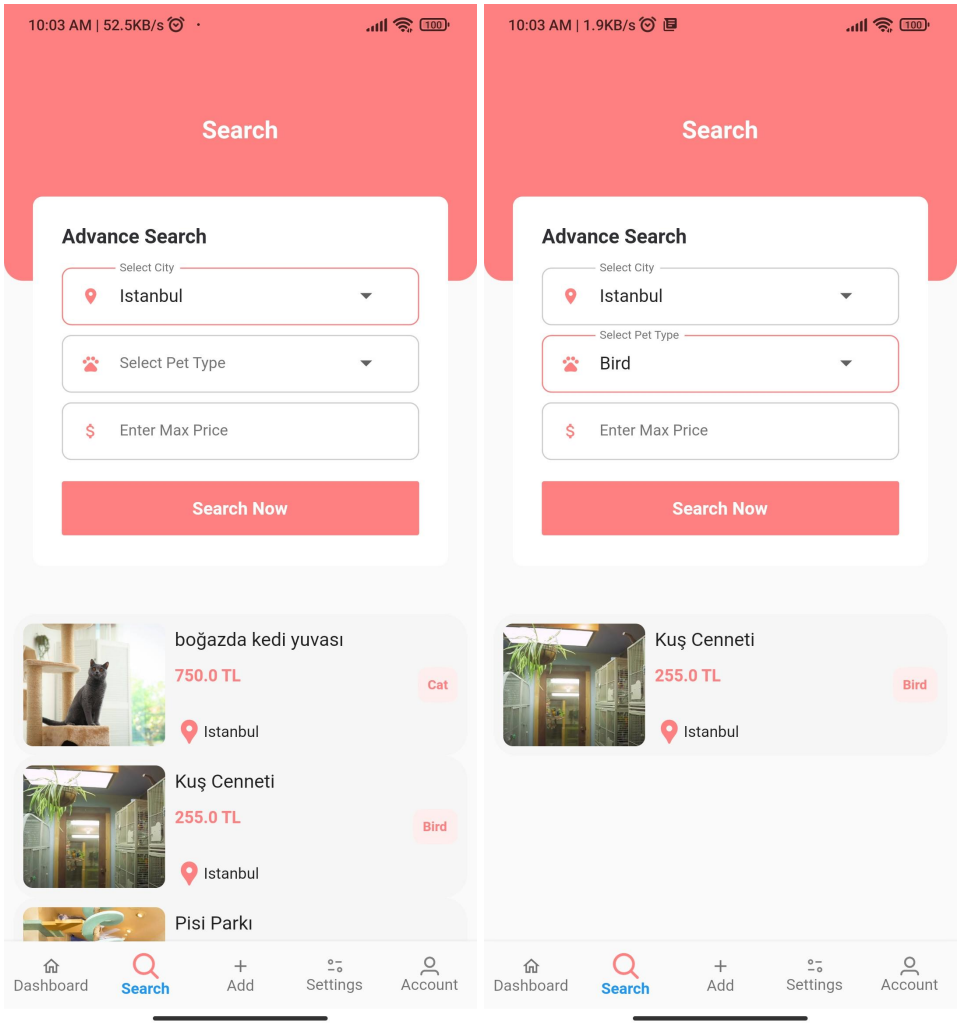
Description: A message is sent to the advertiser via e-mail.	**Test Steps:** 1. Login to the application. 2. Announcement is selected. 3. Click on the letter icon. 4.A message will be sent to the resulting e-mail address.



Result: Email successfully sent to selected advertiser.

Test Case (Searching by Multiple Filtering)

Description: Using multiple filters, searches are performed based on more than one option.	**Test Steps:** 1. Login to the application. 2. Go to the Search section. 3. Options are selected 4. Press the Search button



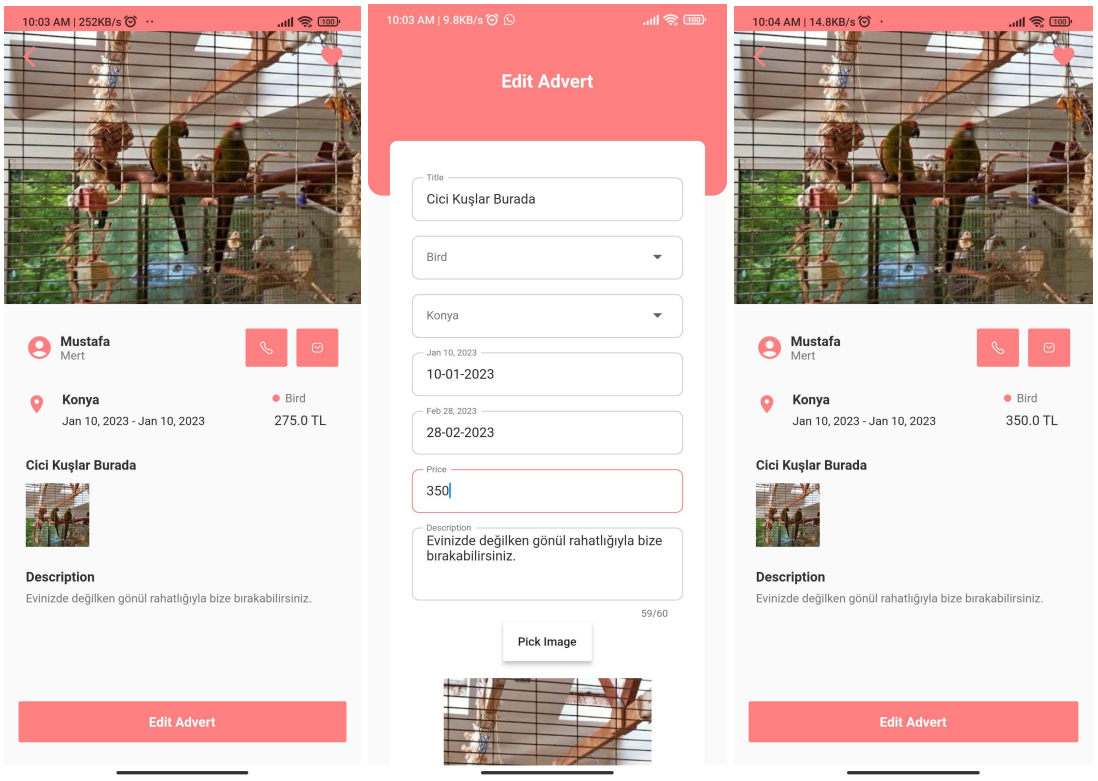
Result: Search by filtering successful

Test Case (Editing Advert Information)

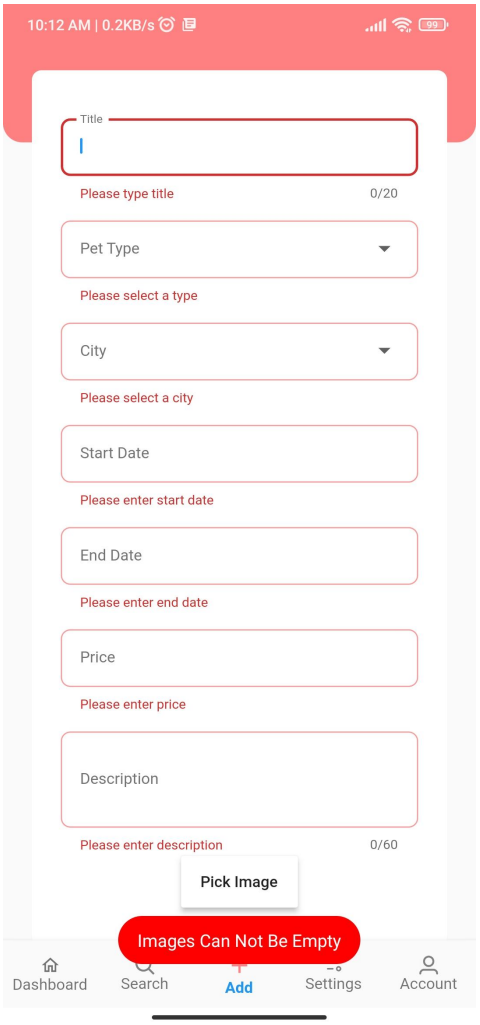
Description: Editing an existing posting

Test Steps:

1. Login to the application.
2. Go to the My Ads section in the Account section.
3. Select the ad and click on the edit button.
4. New information is entered and saved.



Result: The advert has been successfully edited.

Test Case (Adding Advert Unsuccessful)	
Description: Adding an ad with incomplete information should fail.	**Test Steps:** 1. Login to the application 2. Go to Add section. 3. The information is left blank and the add button is pressed.
	
Result: Posting with missing information failed.	

4. Accessible to the potential users

In our research, we have seen that many social media users want the service we offer because of their needs. Here are some examples:

We could not publish it in the markets because we did not have enough budget for developer accounts. Currently we only share .apk file for Android OS.

Download on the App Store

GET IT ON Google Play

Click and download now !

5.Required Links

https://github.com/Software-Engineering-Hoot/hootApp

https://drive.google.com/drive/folders/15gaRPJvStfeSM9_X7DrGd4_g5WazkatI?usp=sharing