



AKDENİZ UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
SENIOR PROJECT FINAL REPORT



Students' Information:

Doğukan ŞENER / Gizem Nur MURATOĞLU / Hakkı Can AKUT
20160808048 / 20170808028 / 20170808010

SUPERVISOR: Asst. Prof. Dr. Alper ÖZCAN

APPROVAL OF GRADUATION SENIOR PROJECT

The B.S. Graduation Thesis titled “Hope Nest” prepared by Doğukan ŞENER, Hakkı Can AKUT and Gizem Nur Muratoğlu has been unanimously approved as a Graduation Senior Project at Akdeniz University, Faculty of Engineering, Computer Engineering Department.

JURY MEMBERS

!!!Title Name SURNAME!!! (Supervisor)	Date	Signature
!!!Title Name SURNAME!!!	Date	Signature
!!!Title Name SURNAME!!!	Date	Signature
!!!Title Name SURNAME!!!	Date	Signature

ABSTRACT

“Hope Nest”

Gizem Nur MURATOĞLU / Hakkı Can Akut / Doğukan ŞENER

SUPERVISOR: Asst. Prof. Dr. Alper ÖZCAN

May / June 2022 – 30 Page

Hope Nest is a mobile-based pet adoption platform. The main purpose of the project is to provide a communication network for these communities, which have become widespread and dispersed on social media, and to accelerate this process. The reason why the application is mobile-based is that many animal adoption platforms exist in social media and many users access these networks from their phones.

When we examined the current applications, we noticed some things in the platforms, applications UI and user control, and we decided to complete these deficiencies and set out on this job voluntarily.

The purpose and general content of the application is to be a voice for these creatures who have no voice without making any material and moral profit. In addition, in this social platform, we have opened a blog, an activity area, in order to provide users with the opportunity to socialize both for themselves and the creatures they have adopted, and in this way, we aimed to transform this application into a more active social networking environment for users.

KEYWORDS: Animal adoption, Mobile Application, Social Media, Social responsibility Project, Community management

FOREWORD

First of all, Asst. Assoc. Dr. We are honored to extend our love and respect to our teacher. Mentoring at every stage in this process and who we are and who support us with evaluations by Özcan Alper.

As our team, we express our happiness and honor in giving each other support and making efforts under such an important issue during this process, and we thank each other.

We are honored to express our gratitude to our parents who instilled these spiritual feelings in us, gave us this spirituality to be sensitive to the environment, nature and the world, brought us to this day and never stopped supporting us. We wouldn't be here today if it wasn't for their efforts.

TABLE OF CONTENTS

ABSTRACT.....	i
FOREWORD.....	ii
TABLE OF CONTENTS.....	iii
SYMBOLS AND ABBREVIATIONS.....	v
LIST OF FIGURES.....	vi
LIST OF TABLES.....	vi
1. INTRODUCTION.....	1
2. LITERATURE REVIEW.....	3
2.1 CHALANGES.....	1
2.1.1 User-Friendly Design.....	1
2.1.2 Ease of Use.....	2
2.1.3 Application Performance.....	2
2.1.4 Data Management and Security.....	2
2.1.5 Compatibility with Different OS Versions.....	2
2.1.6 Choosing Development Technology.....	3
2.1.7 Choosing the Right Software Architecture Pattern.....	3
3 PROJECT ARCHITECTURE AND DESIGN CONSTRAINTS.....	4
3.1 PROJECT ARCHITECTURE.....	4
3.1.1 Software Architecture.....	6
3.1.2 Functional Requirements.....	6
3.1.3 Non-Functional System Requirements.....	8
3.2 DESIGN CONSTRAINTS.....	9
4. ORIGINALITY AND PREVIOUS WORKS.....	13
5. MATERIAL AND METHOD.....	7
5.1. Frontend.....	14
5.1.1. Authentication and User Service.....	14
5.1.2. Advert Service.....	15
5.1.3. Post-Blog Service.....	15
5.1.4. Messaging Service.....	16
5.1.5. Report Service.....	17
5.1.6. Notification Service	18

5.2. Backend.....	18
5.2.1. Firebase Authentication.....	18
5.2.2. Firestore Database.....	19
5.2.3. Cloud Storage.....	19
5.2.4. Cloud Messaging.....	19
6. CONCLUSIONS.....	21
7. REFERENCES.....	22
CURRICULUM VITAE	

SYMBOLS AND ABBREVIATIONS

UI: User Interface

OS: Operation System

MVVM: Model–View–View-Model

UML: Unified Modeling Language

FR: Functional Requirement

API: Application Programming Interface

DESC: Description

RAT: A Tool for the Formal Analysis of Requirements

DEP: Dependency

ID: Identity

App: Aplication

URL: Uniform Resource Locator

Admin: Administrator

FCM: Firebase Cloud Messaging

LIST OF FIGURES

Img 1- UML Class Diagram
Img 2- UML Use Case Diagram
Img 3- MVVM Structure
Img 4- UI Phase 1 Wirefrmae
Img 5- Login and Home Page
Img 6- Blog and Messages Page
Img 7- User Messages and Profile Page
Img 9 FCM send notification API

LIST OF TABLES

Table 1- Attributes of AppUser Model
Table 2- Attributes of Advert Model
Table 3 Attributes of Post Model
Table 4- Attributes of Chatroom
Table 5- Attributes of Messages in Chatroom
Table 6 Attributes of Report Model

1. INTRODUCTION

Today, especially in Central Asian communities, responsibility projects related to nature protection and animal rights are carried out by certain groups and associations. When we examine the social media platforms, we see that there are thousands of people who have given their hearts to this business and they are actively using these platforms. Unfortunately, although they do their best, the group members on the platforms have to follow each group separately or become a member, since the number of groups on the platforms is high. This causes the process to slow down or increase the lack of communication between groups.

HopeNest is a platform that has a user-friendly interface and aims to gather many users under one roof. Users can create their advertisements by using very little of their personal data on this platform, and besides, they can become members of a more social community by organizing events thanks to the blog feature.

The application is very simple to install and use, and the simplicity of the interface design makes it easy for users to use it from there. They can scan the region they want to search for advertisements. Thanks to the content of the advertisements, the users can get information about the friends we want to be a home and communicate directly through the application.

Of course, the application also has a report system for users who do not use the application according to its purpose, and users can report such profiles or messages directly and send the report description to our admins over the application, in this way, we take care not to host users who violate the community rules in the application. Thanks to the innovations we have added to the application, we aim to provide a more disciplined and reliable environment for users. It may be the first indication for them that we follow a user-friendly policy. As we always say, we aimed to develop this project on a voluntary basis and there is no profit motive in its content. The existence of the community order will make the application available to everyone and will enable us to appeal to a larger audience.

HopeNest is an idea that can be the common point of all animal lovers without any prerequisites. We think that the quality of the communication network and community will also increase thanks to the events organized by the users. In fact, we think that it will provide a new beginning of life for many people and an environment where they can break their prejudices towards our living friends. Because the popularity of such an application will arouse a sense of curiosity among people, and thanks to this, people will understand the importance of experiencing the feeling of taking care of an animal and the necessity of taking care of these souls.

If we explain in detail why such an application or platform is needed as a result of the research we have done independently of the application. First of all, as we mentioned at the beginning, there is a multi-group dispersed system and social network. In addition, users are not under the control or guarantee of the platform or group owners, and their user profiles are not fully traded unless they are moderated by the actual platform administrators. Besides, we solved this problem with the reporting system with admin control for the application.

Let's go back to another issue that we talked about before. The platforms or applications that users use are mostly based on ownership, and most of the groups are only open to sharing by admins, so the social network cannot move forward actively and can only be used for postings. In our creation, thanks to this area, our lively friends and owners will also be able to socialize thanks to the events organized, and the communication of the community will become more difficult. As it is always said, there is strength in unity, and

everyone's dedication to this work will ensure the growth of community and increase social awareness at the same time.

Returning to the application interface and control, it is ready for many future plans thanks to the interface and backend system we designed. We have a reporting area open to the report bot system, which we did not include in its production, but may be added in the future. Thanks to the software language and format we use, it provides us flexibility in many aspects. In addition to this flexibility, data is stored in a very secure way.

2. LITERATURE SURVEY

HopeNest is not actually a project inspired from anywhere, it is a completely fictional idea. When we searched the current application stores, we saw many similar applications, but when we created this idea, the application we were inspired by was a user advertisement-based sales platform. This platform, which we have made more useful for our users with some changes in the user interface, has a more portable interface and additional features compared to its counterparts. In this way, we think that we can keep users on this platform and make a difference.

2.1 Challenges

2.1.1 User-Friendly Design

Due to the theme of the design, it is important that it creates a feeling of love in you. Since our subject and theme are creatures in need of help, we need to create a softer and warmer theme instead of an aggressive theme and keep the user in the application in this way. After choosing colors based on pre-made logos and design examples, we made the colors a little lighter and paid attention to highlighting the visuals of the application contents.

2.1.2 Ease of Use

The simplicity of use of the application is an important factor for the age range, and it enables users to be more active and faster without any difficulty in the application. The benefit of this is that if a user performs a transaction without difficulty, he remains more connected to that platform. For example, it is easier to share from the Instagram application that is already on the phone. On the contrary, people make these posts on their blog pages before. Now, in addition to Instagram, we offer a platform where they can follow this whole process. With the button positions, navigation bar and return system in the design, the user will be able to use it very comfortably and will be very comfortable in sharing and researching.

2.1.3 Application Performance

Application performance is a two-sided factor. While we are happy with the user experience, it is important that the system works fast so that the processes carried out on the background can proceed without any damage and problems. By addressing all these, it is the first priority for the application to be able to address all systems on the platform to be prepared and at the same time, this process can proceed smoothly.

2.1.4 Data Management and Security

Data management is very important in terms of application performance and data security. The area you will use as my system, the api etc. Factors you will use for this plays a big factor. The system we will use here is highly supported by the environment in which we developed the application, and the area we use as a data store has been made very secure and it continues to work on it. The harmony between these two systems facilitates data management within the application and increases performance.

2.1.5 Compatibility with Different OS Versions

Today, many mobile development environments support more than one OS. Flutter is one of them, but it is a highly preferred and supported development environment in this field. Of course, as in everyfield, there are deficiencies or areas where flutter falls behind. However, for the application we will develop, it can provide an environment that will run smoothly on most operating systems. That's why we decided to develop the project in flutter environment.

2.1.6 Choosing Development Technology

Firebase is our web service. It is a Google-backend application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment. It use stream so we can pull datas instant. It also provides that if we have no internet connection, pulled datas are stored in cache so we can stil see them. A/B testing, 10k authentication per month, stored data until 1GiB total and real-time database can be used for free by Google Firebase.

We have also use flutter as language. It is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase

2.1.7 Choosing the Right Software Architecture Pattern

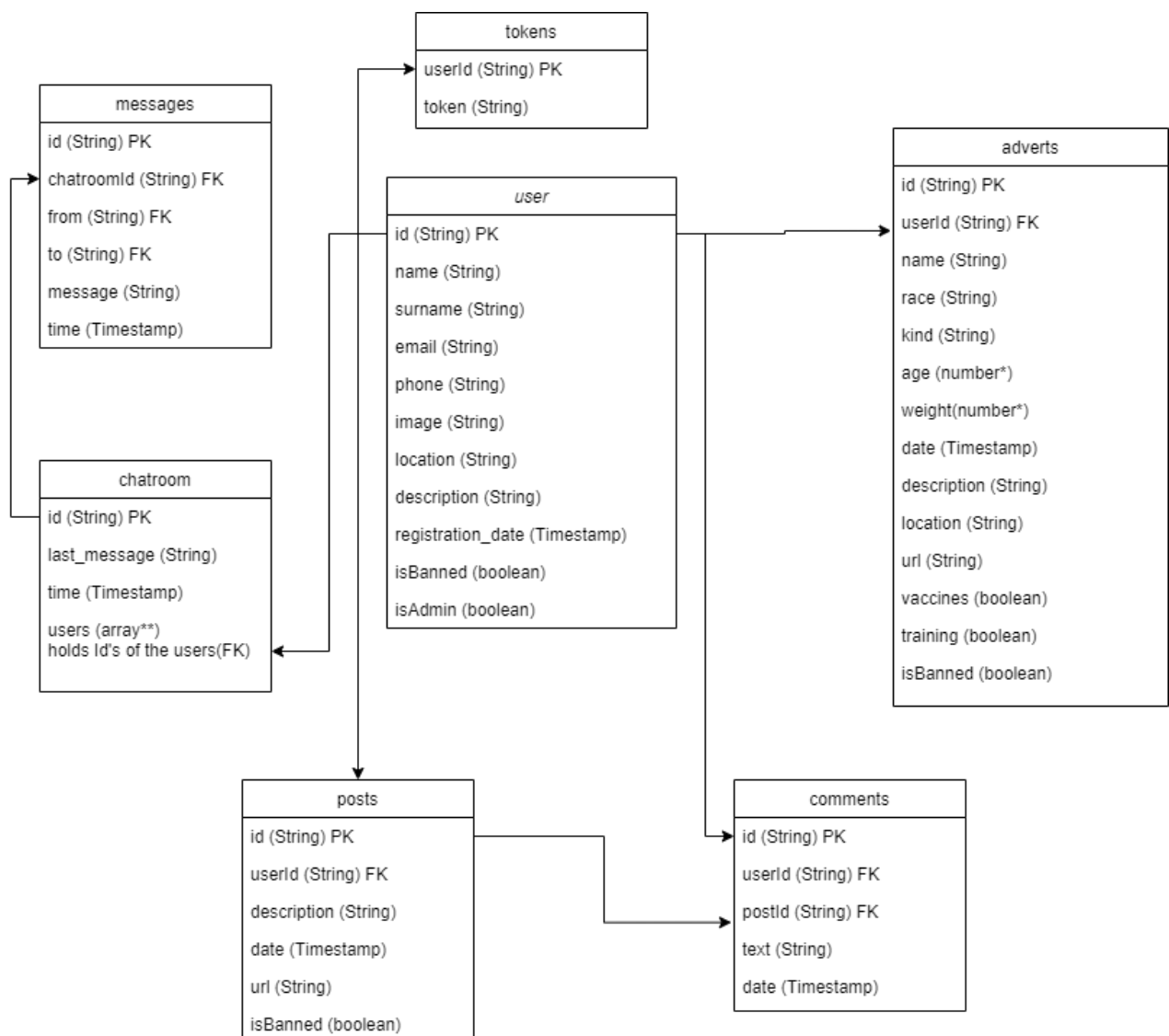
We use MVVM with repository architecture. This architecture provides a proper state management. Also, If we want to change our web service or write our backend, it can implemented on app easily.

3 PROJECT ARCHITECTURE AND DESIGN CONSTRAINTS

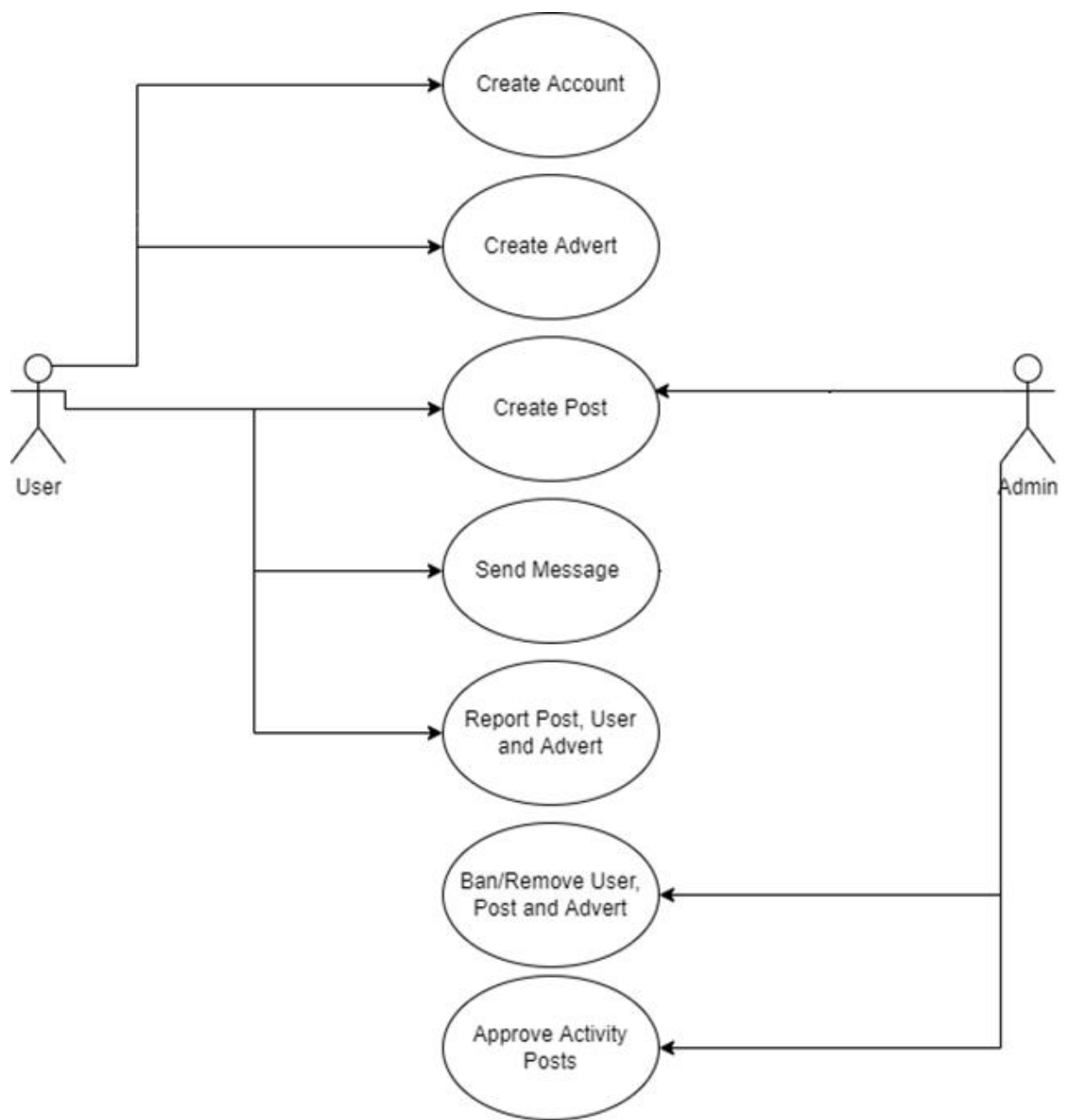
If we explain in detail what we talked about about the research we did before starting the project and the precautions we took regarding the challenges we faced. Instead of examining the project within our subject, we tried to pattern it with a general whole operation and derivatives. Then, after we put this layout on our subject and content and ensured integrity, we started the project architecture and design.

3.1 PROJECT ARCHITECTURE

In the first stage, after determining the necessary model for the project architecture, we created the necessary tables regarding the content table and data relationship, and this is how we took the first steps of the software phase. After deciding on the suitability of MVVM for the model we chose, we continued to create and maintain the project architecture in this way.



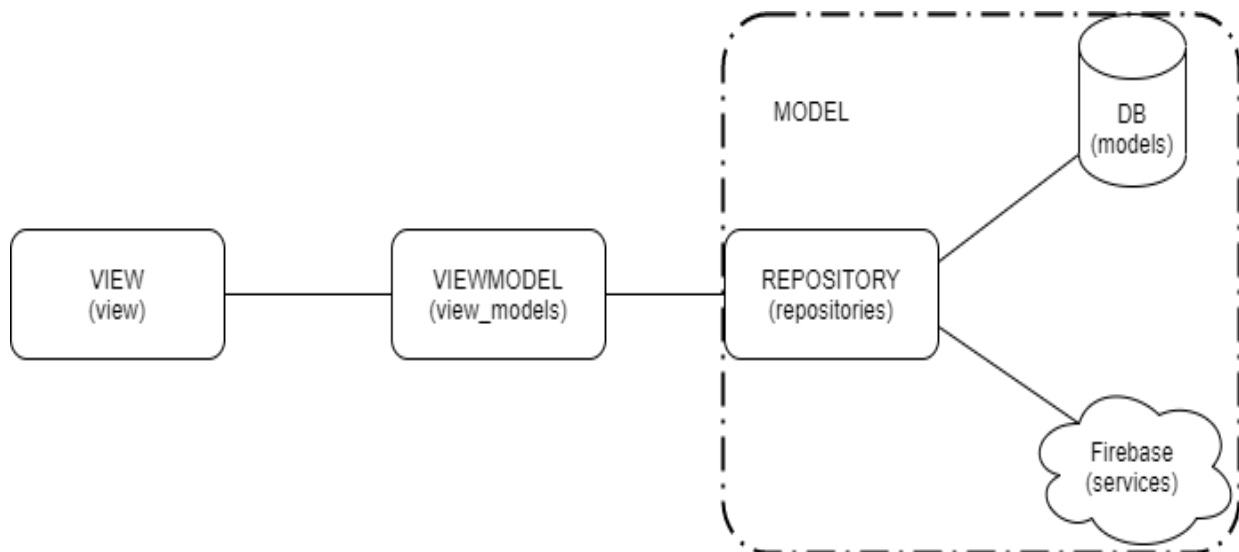
Img 1 UML Class Diagram



Img 1 UML Use Case Diagram

3.1.1 Software Architecture

Our architecture is MVVM with repository. We use MVVM because it reduces complexity of the code, since we separate view, state management and our methods. The reason for adding the repository is easiness when we change our web service.



Img 2 MVVM Structure

View: Manages user interfaces, if AppState (application state) is BUSY shows circular progress indicator, If not then shows screens(Login page, Home page etc.)

View Model: Manages AppState, if there is a call for back-end methods, (such as authentication methods etc.) it changes AppState to 'BUSY' state, when back-end method returns a response then changes AppState to 'IDLE'.

Repository: Controls which service will be used, with this if we decide to change web service or add new one it would be much more easier to implement.

Service and Model: Does back-end requests and returns needed data.

3.1.2 Functional Requirements

ID: FR1

TITLE: Download mobile application

DESC: A user should be able to download the mobile application through either an application store or similar service on the mobile phone. The application should be free to download.

RAT: In order for a user to download the mobile application.

DEP: None

ID: FR3

TITLE: User registration- Mobile application

DESC: After the user download the app, they can register from sign in screen which is implemented as a button on login page. Button can directly move to sign in page.

RAT: In order for a user to register on the mobile application.

DEP: FR1

ID: FR4

TITLE: User (Admin) log-in - Mobile application

DESC: Given that a user has registered, then the user should be able to log in to the mobile application. The log-in information will be stored on the phone and in the future the user should be logged in automatically.

RAT: In order for a user to register on the mobile application.

DEP: FR1, FR3

ID: FR5

TITLE: Mobile application- Search

DESC: User can search for advert or post on content pages also can use groups for specify the type of adverts or posts.

RAT: In order for a user to search for a adver or a post.

DEP: FR4

ID: FR6

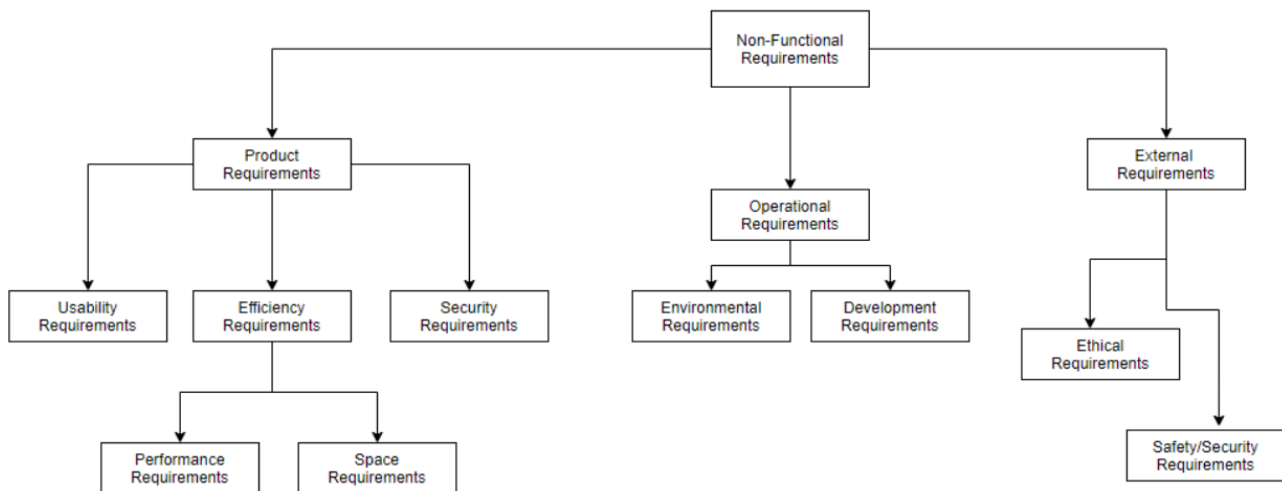
TITLE: Mobile application - Profile page

DESC: On the mobile application, a user should have a profile page. On the profile page a user can edit his/her information, which includes the password, e-mail address and phone number.

RAT: In order for a user to have a profile page on the mobile application.

DEP: FR1

3.1.3 Non-Functional System Requirements



Img 4 Requirements Scheme

Usability Requirements

You can easily select product and add your basket which is shown before UI diagrams. Also you can see information about your order from another page on the app.

Space Requirements

You must download the app your mobile device. So you have give a space on your storage of your mobile phone to use the services.

Performance Requirements

You need internet connection for the app. It must be upper bound of the standart to reload the map actively. Also you can use the quality on the product search page.

Security Requirements

We have user login use system and every user have own user information. Your order is related directly you so our system take your order and give a non-busy driver to make that deliver as much us quick. Also all IDs are encrypted on the system so no one can reach or create process on your ID.

Ethical-Safety/Securit Requirements

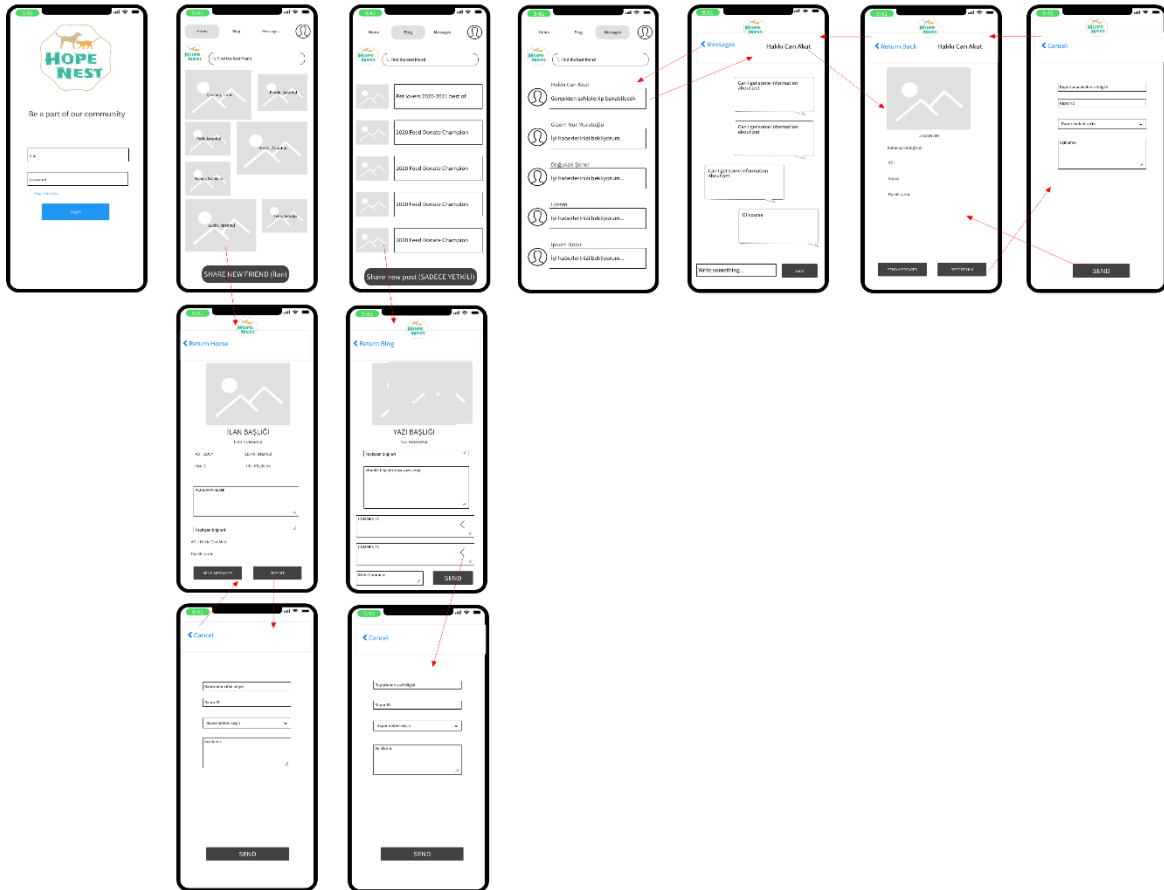
We use your location during the delivery state after complete the order. The system can not use your location anymore. So, deliver can not reach you location on the map anymore. We beyond the rules split the system with user informations.

3.2 DESIGN CONSTRAINTS

At the beginning of the project, the first thing we did was to create a mockup for the general design in order to have an idea in our minds during the initial design phase. The benefit of this for us was to ensure that the original template, which will appear in the UI part before the model and design elements decided for the actual design, starts to come to life in the software part. The outline of the user interface plays a big role in the emergence of the basic lines of the application.

We started with the drawing of the moment components and general structure of the design. At this stage, positions and relationships are more important to us than colors and shapes. We have worked on basic elements such as what will be on a page, which elements and structures will be used (button, box, navigation, img), and content areas and redirects on the pages. We continued this work online and we did it on mockflow.

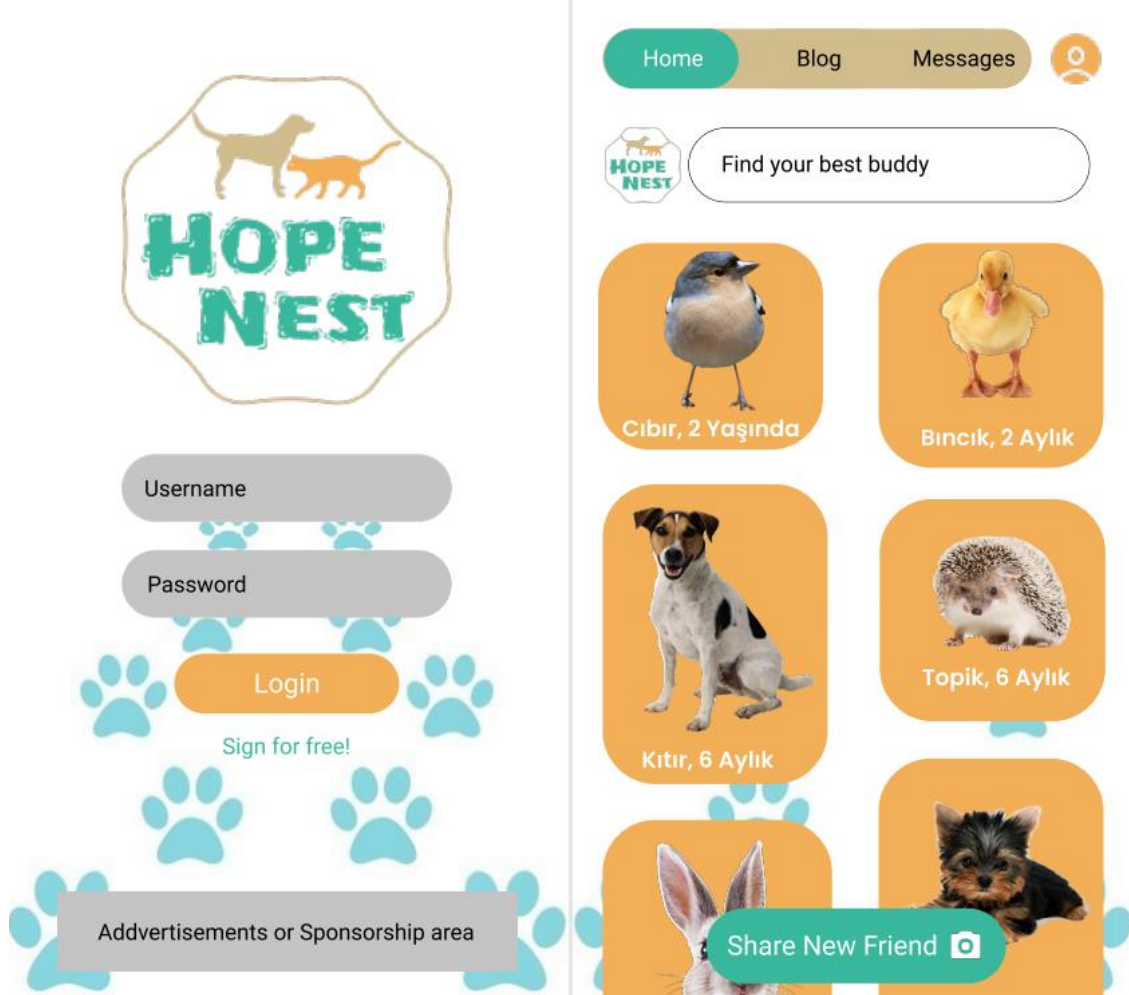
We did not need reference applications much during the study. We have prepared a suitable ground according to the structure we have in mind and what we may need, and we have prepared a suitable environment for the physical internal structure of the application.



Img 3 UI Phase 1 WireFrame

When the actual design of the application was started, we decided to go for a more red-yellow tone in terms of color and theme, which was decided when the previous applications were examined, because when we thought of this as a theme, it would provide a softer tone and eye harmony, and it was also a tone that did not tire the eyes. Then, if we go

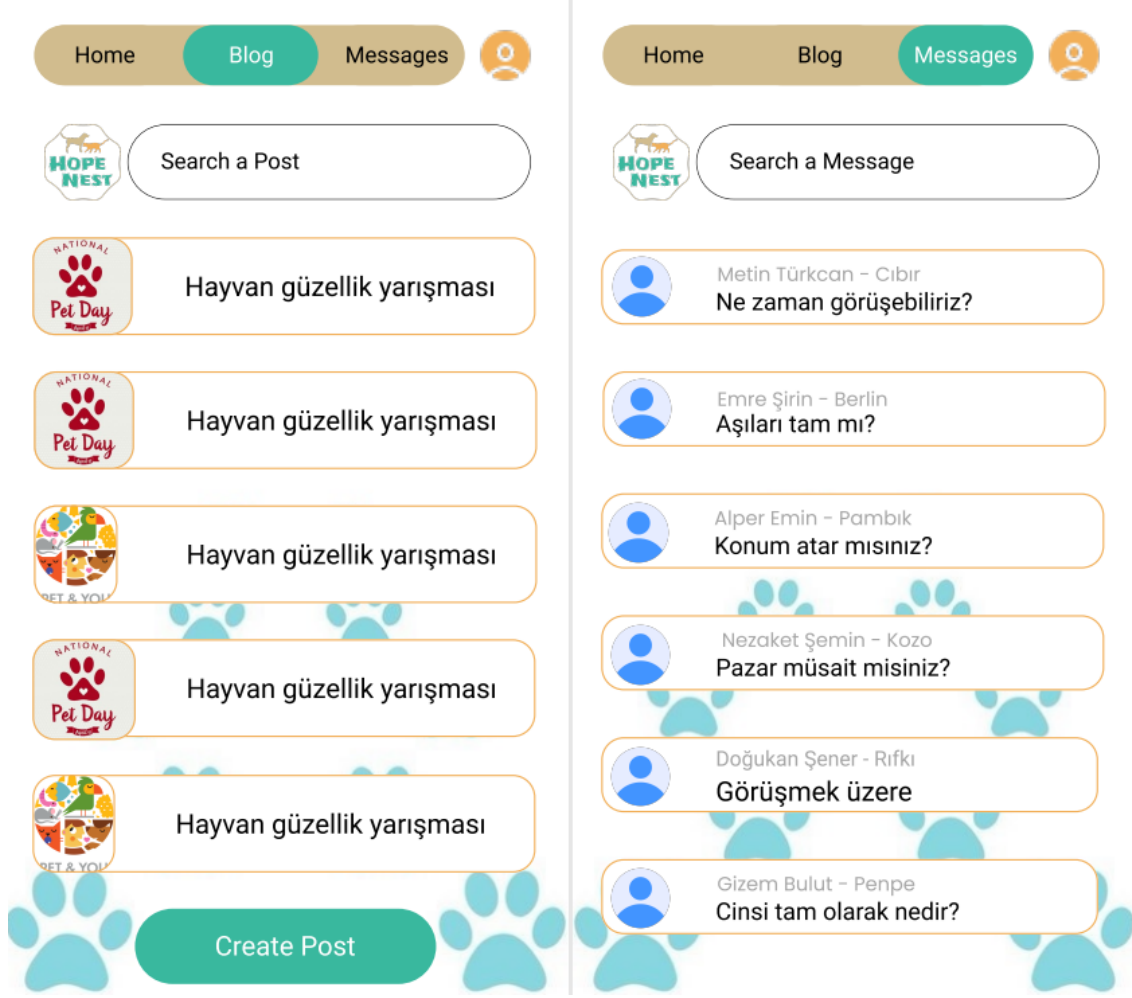
to another tone range that contrasts with scala to provide emphasis for the application, we achieved a green tone between yellow and blue and provided a harmonious tone for our theme, so that the user would not mix objects in the interface, but also would not be disturbed by any situation when looking at the screen thanks to its softness.



Img 4 Login and Home Page

The homepage welcomes us with the logo and the user login form. Below this, there is an article in a color tone that attracts the attention of the user, directing non-members to the registration page. When we examine the main screen, there is the navigation bar, user profile button and search field, which was mentioned before, and active ads are visible under it.

If we examine another element that attracts attention, we have increased the awareness of the user thanks to the color we have given to the posting button.

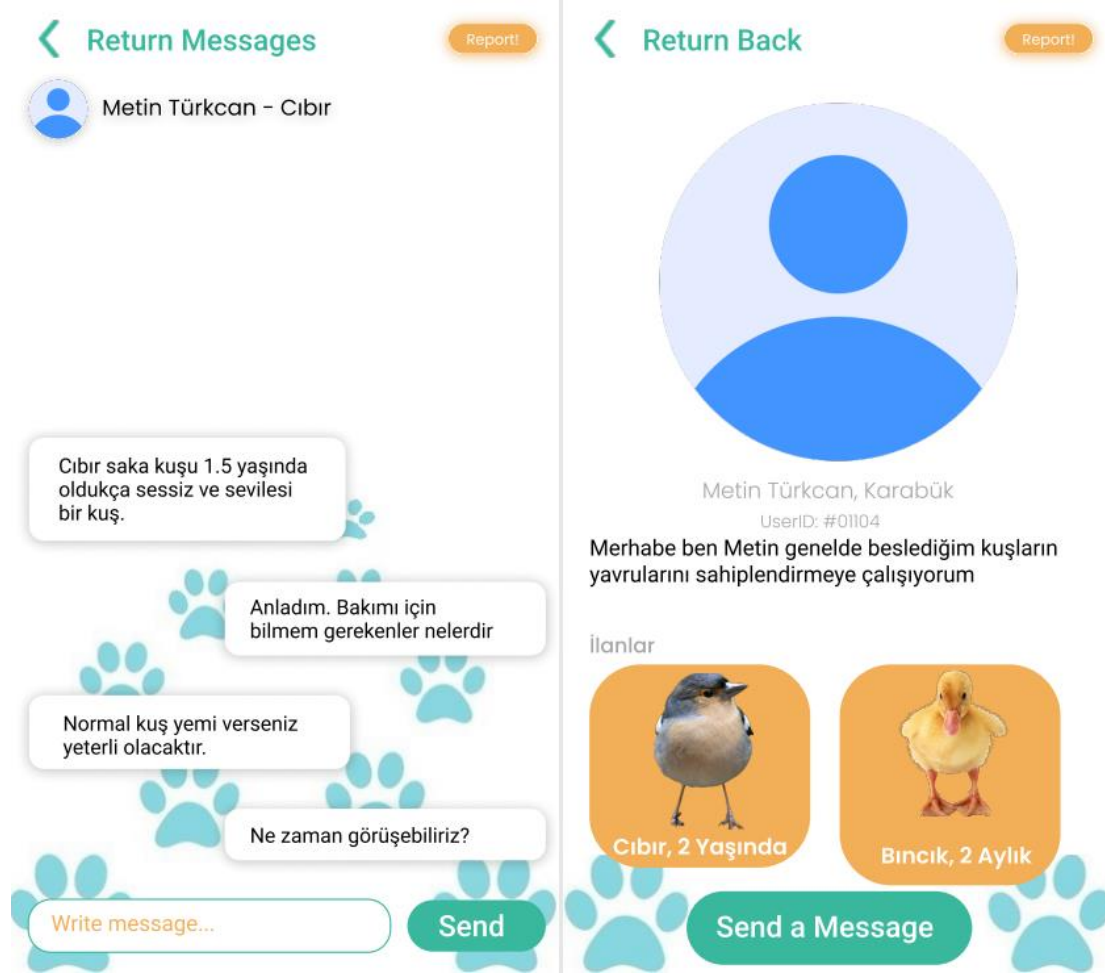


Img 7 Blog and Messages Page

You can see the postings added on the blog page from the current order to the past. The featured image and ad name for the displayed ad.

On the page that appears when you click on the ad, you can access the contents and information about the ad content, and you can observe the comments added by the users under the ad. If you want to add a new ad, as you can see, the content is as simple as the information requested, of course, the added ad will first be approved by the admin side. After approval, it will be added to the list in a visible way to you and other users.

On the other side, there is the side where our messages that welcome us are kept. Here you can see the messages of the advertisers that we have previously viewed and posted. These contents cannot be viewed by a third party, including the admin. If you report the message or a posting, it will be forwarded to the admin with the reported content. In this way, it is not possible for a third party to directly intervene and observe the communication between users.



Img 8 User Messages and Profile

As we mentioned before, there are two different places to report for the situation you want to be examined from the user message or profile, of course, you can easily give the reason for the report and the details of what you want to be examined by writing a description in the form created for you while the user is reporting. In this way, you will contribute to the progress and acceleration of the process and contribute to the order of the platform.

Of course, another feature that will attract our attention on this screen is that we can also observe other advertisements of the user you reach from an advertisement, in this way, if you contacted this user due to the proximity of the subject or for other reasons, you can easily access other advertisements belonging to the user that you can evaluate.

Another thing we will focus on is that the navigation bar disappears when you enter an inner page, because these pages are detail fields independent of the main area, instead of displaying the modules used to save space on this page, we provide guidance to the page you want to reach the content of, and when you want to return, it provided us with most of your applications. By using the button, we bring a performance-enhancing solution and when you return, this page will no longer be working.

4. ORIGINALITY AND PREVIOUS WORKS

In our research on the subject, we observed some situations that are contrary to personal data in the applications and platforms we examined, and as we try to keep the security of individuals at the forefront, we took care to keep the communication line completely within the application and we took care to keep user data confidential. In addition, due to the data we use, we did not include non-application communication lines within the application, and of course, we limited the opportunities offered by other applications among users. Some of the applications we reviewed allowed interpersonal interaction by using phone or e-mail information directly in the interface and forwarding it to different platforms, and we avoided this. As long as users kept their notifications turned on, they would be able to be aware of all the interaction. That way, they wouldn't have any losses.

In the interface review, some applications had a more aesthetic structure compared to our interface, due to the quality of the design, but what we highlighted was to increase performance and facilitate user use by avoiding multi-layered structures thanks to the navigation menu we used on the main screen. However, with content positioning, we aimed for users to see the necessary information and easily see the contents of advertisements and events.

When we returned to the idea base of our project, we started this way by producing this idea due to the load on social media, and when we saw similar applications, we thought that there was still a deficiency in this regard or that the reasons why the applications made so far were not successful were due to the deficits. We aimed to aim for continuity on the subject. This was not the first project we did as a group, and we worked on the deficiencies in other examples of our previous works, closing these gaps and adding more, and this was one of them.

5. MATERIAL AND METHOD

5.1. Frontend

In the beginning of the project development, we decided to develop a mobile application in our client side. One of the most important thing is which language will we use in application. Then, we chose Flutter and Dart as our application language.

In the chosing process, we firstly decide to use cross-platform languages because this way we can develop iOS and Android application with same code. Then we had three choice which are Xamarin, React Native and Flutter. We picked Flutter among them because, it has better performance than other two^[1]. Also it has Firebase support.

5.1.1. Authentication and User Service

Flutter has support for Firebase Authentication Service. There is a package named `firebase_auth`^[2] (version ^3.2.0) which provides Firebase Authentication API. We used sign in with email and password, sign up with email and password and, get current user features of this package.

Name	Data Type	Description
uid	String	ID of the user
name	String	Name of the user
surname	String	Surname of the user
phone	String	Phone number of the user
email	String	Email of the user
image	String	Image URL of the user
location	String	Location of the user
description	String	About text of the user
registrationDate	Timestamp	Registration Date of the user
isAdmin	bool	Is user an admin
isBanned	bool	Is user banned

Table 1 Attributes of AppUser Model

When we send a request via these API's, the web service will return Firebase User which mentioned in section 5.2.1. Authentication Service. We created a model named AppUser, with the attributes of shown in Table 1. We are created new model because, Firebase User only holds ID and email of the user. So, when a new user sign up, we add the other data of the user to users collection in Firestore Database. Also, when we get a Firebase User, we get the id of the user. With using id information, we get the other data from Firestore Database.

When the App starts it checks that is there any current AppUser, if there is an AppUser it directs user to home page otherwise, directs to the sign in page. These two pages shown in Img 6.

5.1.2. Advert Service

Flutter has support for Firebase Firestore Database. There is a package named `cloud_firestore`[\[3\]](#) (version ^3.1.0) which provides Cloud Firestore API. It basically helps to send API request to the Firestore.


Flutter also has support for Firebase Cloud Storage. There is a package named `firebase_storage`[\[4\]](#) (version ^10.2.8) which provides Firebase Cloud Storage API. With these API's we can add files to the storage which is images in our case.

With using this `cloud_firestore` package we set and get Advert data from adverts collection in Firestore Database. When we send a get request with the help of the package, we are creating a stream pipeline between database and application which mentioned in section 5.2. Backend. When we send a set request, we firstly add the selected image file to the Cloud Storage, this request will return the URL of the image. We are adding this URL to the object and set a new advert in database.

Name	Data Type	Description
id	String	ID of the advert
uid	String	ID of the owner user of the advert
date	Timestamp	Creation date of the advert
url	String	Image URL of the advert
description	String	Description of the advert
name	String	Name of the animal
age	int	Age of the animal
kind	String	Kind of the animal
race	String	Race of the animal
weight	double	Weight of the animal
vaccines	bool	Is animal vaccinated
training	bool	Is animal trained
location	String	Location of the advert owner
isBanned	bool	Is advert suspended

Table 2 Attributes of Advert Model

We are holding advert data in the Advert model which we created. This model has attributes which is shown in Table 2.

The current AppUser can create new advert. There is a floating action button in for creating a new advert which shown in . After clicking this button, a dialog will pop on the screen. With adding necessary information to this dialog user can create new advert.

5.1.3. Post-Blog Service

In blog page, we show user to the posts of the other users. Also current AppUser can send a new post in this page. For these features we used `cloud_firestore` and `firebase_storage` packages which mentioned in section 5.1.2. Advert Service.

We can set and get posts. When we send a get request with the help of the `cloud_firestore` package, we are creating a stream pipeline between database and application which mentioned in section 5.2. Backend. When we send a set request, we firstly add the selected image file to the Cloud Storage, this request will return the URL of the image. We are adding this URL to the object and set a new post in database.

Name	Data Type	Description
id	String	ID of the post
uid	String	ID of the owner user of the post
date	Timestamp	Creation date of the post
url	String	Image URL of the post
description	String	Description of the post
title	String	Name of the post
isBanned	bool	Is post suspended

Table 3 Attributes of Post Model

We are holding post data in the Post model which we created. This model has attributes which is shown in Table 3.

The current AppUser can create new post. There is a floating action button in for creating a new post which shown in Img 7. After clicking this button, a dialog will pop on the screen. With adding necessary information to this dialog user can create new post.

5.1.4. Messaging Service

In message page, users can send message to other users. Users follow their message box and histor in this page. Current user can start new communication with different user who has not a messages history. For these features we used streamBuilder to listen messages of users momentarily . So users do not have to refresh the message page.

We can send and see messages that is about us. When we send a request to send messages to other users, these are added in cloud firestore. And then, we pulled the messages using stream provider.

Name	Data Type	Description
id	String	ID of the message
last_message	String	last message received
time	Timestamp	Time of last message received
users	String	Users in contact
messages	collection	Messages history

Table 4 Attributes of Chatroom

Name	Data Type	Description
id	String	ID of the message
from	String	Sender user
time	Timestamp	Time to send
to	String	Users who is received message
message	String	Message

Table 5 Attributes of Messages in Chatroom

Chatroom is our message boxes which is in message page. Each boxes include messages that is contact. These messages are kept in messages collection which is in chatroom collection.

5.1.5. Report Service

This system can be abused so we decided to add report system. The report process is like that;

1. User opens a profile or a post or an advert.
2. User reports a user or a post or an advert.
3. This report added to the report collection in Firestore database.
4. Admins' reports page will be updated.
5. Admin will decide suspend or not suspend.

Name	Data Type	Description
id	String	ID of the report
uid	String	ID of the owner user of the report
reportType	ReportType(enum)	Shows is it user or post or advert
reportedId	String	ID of corresponding reportType
description	String	Description of the report
title	String	Title of the report
conclusion	String	Conclusion of the report
date	Timestamp	Creation date of the report
isDone	bool	Is report concluded
reportUser	AppUser(model)	AppUser which reported, can be null
reportAdvert	Advert(model)	Advert which reported, can be null
reportPost	Post(model)	Post which reported, can be null

Table 6 Attributes of Report Model

We are holding post data in the Report model which we created. This model has attributes which is shown in Table 6.

We used `cloud_firestore` package's API's which is mentioned in section 5.1.2. Advert Service. When user reports, we are sending a set request to report collection to add a new report. In the admin panel, admins views the reports with a get request which creates a stream pipeline. After an admin decides to suspend or not suspend. We send a set request to change the `isDone` attribute of corresponding report. If the decision is suspend then, we also change the `isBanned` attribute of corresponding user or post or advert to the true which means it is suspended.

We add report buttons on top right corner in profile, post and advert pages. When user clicks this, a dialog will be pop on the screen. With adding necessary information user can report a profile or a post or an advert.

We created an admin panel which shows the list of reports. Admin can click one of reports and will view report information in report page. In this page admin can suspend the corresponding user or post or advert.

5.1.6. Notification Service

Flutter has support for Firebase Cloud Messaging. There is a package named `firebase_messaging`[\[5\]](#) (version ^11.4.0) which provides Firebase Cloud Messaging API. We also used `flutter_local_notifications`[\[6\]](#) (version ^9.5.3+1) for displaying local notifications.

When a user enters the home page, we will create a stream pipeline with FCM for updating the token of the device. When the token of the device refreshes by FCM, we will update corresponding data in token collection in Firestore database with using `cloud_firestore` package which is mentioned in section 5.1.2. Advert Service. So, we hold every users token on our database.

For sending a notification, we created an API request function. In this function, we are sending request to FCM server. In the headers we are sending the our server key to FCM server, so it will known that which App that notification request come from. In the body part we are sending token of the reciever user, title and the body of the notification.

When user sends a message to another user, if the send message is successful, we will take token from Firestore database and send notification with calling API request which we mentioned above.

5.2. Backend

We used Firebase as our web service because it provides free Authentication Service, free Data Storage up to 1GB, free Cloud Storage up to 5GB, free Cloud Messaging, free A/B testing and Analytic tools.

Also it provides stream service. Stream service means when there is a get request with stream, it will create a pipeline between the server and the client device and, it will send the data via this pipeline. When there is a change in the database, if this stream pipeline is active the device will recieve the updated version of the data, so the app will updates it's view with new data. This is important because in messaging part, the message will send to other user almost instant.

5.2.1. Firebase Authentication

We are using Firebase Authentication Service for our authentication process. This service provides us end to end identification solution. We are using this service because, it's much more easier and safer to implement this structure. We are using email and password for authentication. This service also can provide two factor auth with phone auth and oauth protocol with Google, Facebook, GitHub login and more.

We are using three main feature of this service. These are get current user, sign in with email and password and sign up with email and password. When we call these requests, it will return a Firebase User. This User object only has ID and email attributes, since we sign in with email and password. We are holding the data of user in firestore database which we mentioned in section 5.1.1. Authentication and User Service.

5.2.2. Firestore Database

There are two different Firebase databases which are Firestore database and Realtime database. We chose Firestore because, Realtime database does not have stream API support.

Firestore database is a NoSQL database, we are working with collections and documents. We have seven collections and one sub-collection, these are:

- Users: holds documents of the users
- Posts: holds documents of the posts
- Adverts: holds documents of the adverts
- Comments: holds documents of the advert's comments
- Reports: holds documents of the reports
- Tokens: holds documents of the users' FCM tokens
- Chatroom: holds documents of the chatrooms
 - Messages: This is a sub-collection of the chatroom, every document of chatroom has this sub-collection, it holds messages of the corresponding chatroom.

The data inside of each document are shown in Img 1.

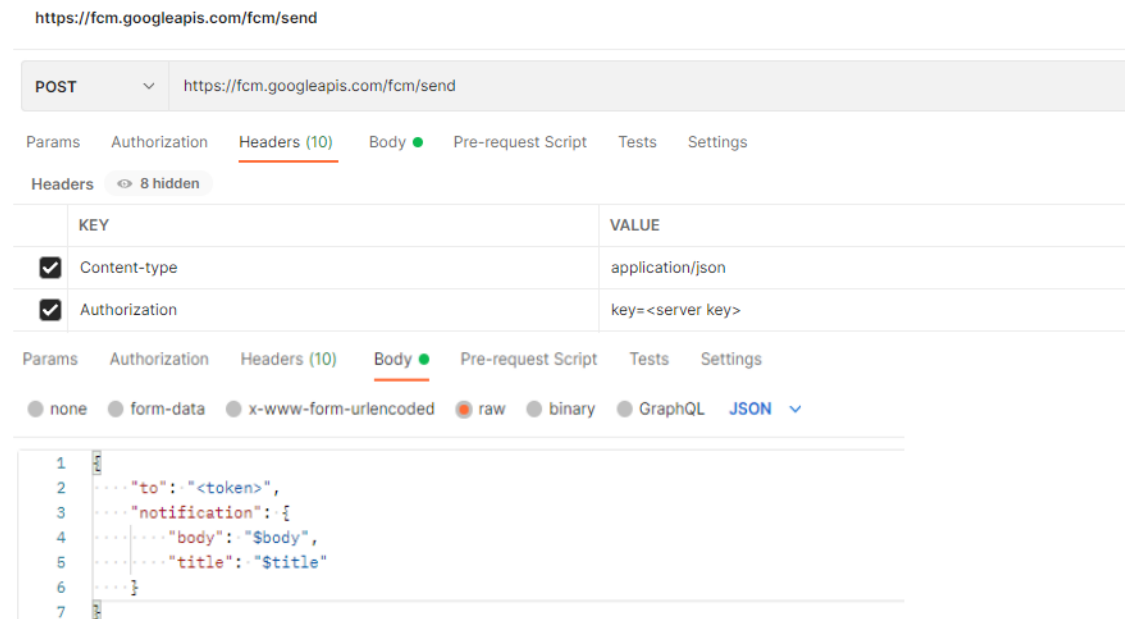
5.2.3. Cloud Storage

Since we want users to create new adverts and posts, we need to have a storage to store these adverts' and posts' images. So, we decide to use Firebase Cloud Storage. If the user is authorized which means the user that goes through authentication process, the user can add a file to the cloud storage.

When a user sends a request for adding a file to the cloud storage, if it is successful, the return has information of the URL of the file. The process explained in section 5.1.2. Advert Service and section 5.1.3. Post-Blog Service.

5.2.4. Cloud Messaging

In our app, users can message each other. So, we decided to create a notification system that sends notification to user when other user sends a message. For creating system, we used FCM.



Img 9 FCM send notification API

FCM has an API for sending notifications which is shown in *Img 9*. Since we send request the FCM server, we need to add our specific server key to headers for letting FCM server to understand which app sends this notification request. Then, we add token of the receiver user, body and title of the notification. The process of sending notification explained in section 5.1.6. Notification Service.

6. CONCLUSIONS

When we examine the project, we can observe that we are trying to follow the rules in terms of design and remain user-friendly. It is obvious when we have to avoid some content and features that can be added as well. We think that the biggest feature that can be added in terms of developability is the report evaluation bot, of course, the bot aims to set the priority order instead of working directly, just like in our e-mails. The control of the content is on most platforms, it's a bit of user responsibility, if the limitations are too much, we can remove users from the application, so we tried to be flexible enough. In addition to the application content, another important system we considered was the authorization and user materiality feature, which may be related to the reliability of the activity and content owners. Based on the results obtained by the user as a result of the evaluations, users can reach the level of user control.

When we examine it technically and in terms of content, we are aware that the project does not have a direct contribution to the literature. We aimed to progress based on ideas and benefits. Today, projects related to this issue are carried out in many countries and people's sensitivity on this issue has increased recently in Turkey. Our contribution in this cycle is to build a large community and increase communication and interaction among this community.

As a result of the research we have done, we tried to go over the shortcomings, of course, one of the most important things for an application is that that application is well advertised, while we were dealing with the subject, we looked at what we could do at the first stage, what kind of deficiencies we could remove and what advantages we could provide. As a result, we are now satisfied with the work we have done, the design can be improved. Content can also be expanded and added in many features. Like every entrepreneur, we tried to give the best to the user in the first stage and of course we did not give up on security.

7. REFERENCES

- [1]: <https://onedio.com/haber/bir-pati-de-sen-tut-hayvanlara-yardim-eden-14-sosyal-medya-hesabi-691937> [accessed on September 2021]
- [2]: <https://www.haytap.org/> [accessed on September 2021]
- [3]: <https://www.figma.com/> [accessed on September 2021]
- [4]: <https://inveritasoft.com/blog/flutter-vs-react-native-vs-native-deep-performance-comparison> [accessed on October 2021]
- [5]: https://pub.dev/packages/firebase_auth [accessed on November 2021]
- [6]: https://pub.dev/packages/cloud_firestore [accessed on November 2021]
- [7]: https://pub.dev/packages/firebase_storage [accessed on February 2022]
- [8]: https://pub.dev/packages/firebase_messaging [accessed on May 2022]
- [9]: https://pub.dev/packages/flutter_local_notifications [accessed on May 2022]

CURRICULUM VITAE

Doğukan ŞENER was born in İzmir on June 28, 1998. He completed his primary and middle school Zehra Hoca Hanım Primary and Middle School. He went to Cem Bakioğlu Anadolu Highschool. In 2016, he started to Akdeniz University Computer Sciences Engineering. Today He Works at Pen Digital Agency as Front-End developer since 2021 September.

Hakkı Can AKUT was born in Bursa on January 8, 1999. He completed his primary and middle school at Karamehmet Primary School. He went to the Özlüce Anadolu Highschool. In 2017, he started Akdeniz University Computer Sciences Engineering.

Gizem Nur MURATOĞLU was born in İstanbul