logo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**CNG 495**

**FALL - 2023**

**TERM PROJECT PROPOSAL**

**Team Members:** Bartu Can PALAMUT, Doğukan AKDAĞ, Nurberat GÖKMEN

**ID Numbers:** 2386225, 2452928, 2453231

**Table Of Contents**

[**1. Introduction** 4](#_Toc152458686)

[**1.1**  **Project Purpose** 4](#_Toc152458687)

[**1.2**  **Cloud Delivery Model** 4](#_Toc152458688)

[**1.3**  **Cloud Provider** 4](#_Toc152458689)

[**1.4**  **Project Contribution** 4](#_Toc152458690)

[**1.5**  **Project Development Environment** 6](#_Toc152458691)

[**1.6**  **Sequence Diagrams** 6](#_Toc152458692)

[**1.7**  **Data Flow Diagram** 8](#_Toc152458693)

[**2. Progress** 8](#_Toc152458694)

[**2.1**  **Data Flow Diagram** 8](#_Toc152458695)

[**3. References** 9](#_Toc152458696)

**Table Of Figures**

[Figure 1 Sequence Diagram for Registration Process 5](#_Toc149932525)

[Figure 2 Sequence Diagram for Send Photo Process 6](#_Toc149932526)

[Figure 3 Sequence Diagram for Open Photo Process 6](#_Toc149932527)

[Figure 4 Data flow diagram of PhotoChat 7](#_Toc149932528)

# **1. Introduction**

## **1.1 Project Purpose**

Our goal is to develop a dynamic application designed for seamless photo sharing. In the application we developed, users will be able to register, and send them friend requests. If the user to whom the user sent a friend request accepts the request, a private chat will be opened between each other. In this chat, users will be able to send photos to each other. To enhance privacy, received photos can be viewed only once. We've implemented a database to store photos in base64 format until they are accessed, ensuring that the photos captured are both conveniently accessible and securely preserved. Also, in this database, we store user informations.

## **1.2 Cloud Delivery Model**

We are planning to use Software as a Service (SaaS) for our Snapchat clone project, and the chosen provider will be Google Cloud. We've selected SaaS because we are planning to develop a java application as a back-end server. After we develop this server, we make an artifact for this java application. We will deploy this artifact to Google Cloud App Engine. Google Cloud App Engine minimizes our involvement with infrastructure management, allowing us to concentrate exclusively on application development. This choice aligns with our project's specific focus on sending and receiving photos. On the other hand, for choosing this cloud delivery model, we are planning to develop a flutter application as a client side of this project for users. If we update an existing service in this client application installed on the users' phone, we save the users from maintenance trouble by updating it in the cloud where we installed the back-end, without the need for the users to update the application.

## **1.3 Cloud Provider**

We are planning to use Google Cloud as a provider. We use two services of Google Cloud which are App Engine and Cloud SQL. We are going to deploy our back-end application to App Engine. Also we connect our application to Cloud SQL for database operations.

## **1.4 Project Contribution**

We are planning for everyone to contribute an equal amount to the different parts of the project, so everyone will get an equal amount of experience from the different parts of the project and improve themselves. Also, as you can see below, our milestones and tasks for each user are listed.

|  |  |  |
| --- | --- | --- |
| **ID** | **DESCRIPTION OF TASK** | **ASSIGNED PERSON** |
| 1 | Preparation user authentication services. The user can register and login to the system. When a user logs in to the system, JWT token will be provided to user. With this token, users can use the application.  • User entity should be implemented.  • Services and interfaces that are written on Spring Security Documentation[¹](#_3.__References) should be implemented.  • Login and Register endpoints should be implemented. | BARTU CAN PALAMUT |
|  |
|  |
|  |
| 2 | Preparation “Friendship” services.   • Users send a friend request to any user. Service should validate username that entered from user. If this username validate, “FriendRequest” should be created.   • UserRequest entity should have sender and receiver. If the second user accept that request, this user should be added to sender user’s friend list and sender user should be added to this user’s friend list.  • UserRequest entity should be deleted after completing process to reduce unnecessary data. |  |  |
|  |
|  |
|  |
|  |
|  |
| 3 | Preparation “Image” services.  • Users should be able to send photos to any user they are friends with.  • Service that takes image, converts it to base64 format, then saves to the database needed.  • If a user opens that image, this image should be deleted. So, there is a service that deleting image needed. |  |  |
|  |
|  |
|  |
| 4 | Preparation onboarding screens with flutter.  •Welcome page, login and register page of the application should be prepared. |  |  |
|  |
| 5 | Preparation camera screen with flutter.  • “Flutter Camera Plugin” [²](#_3.__References) package should be implemented.  • A camera screen should be implemented. On that screen the user can open front end back cameras. Also, the user can take a photo and send this photo to the user. |  |  |
|  |
|  |
|  |
| 6 | Preparation chat screen with flutter.  • If the user has a received image that s/he have not opened before, it should be listed on this page.  • Users can open that image. |  |  |
|  |
|  |
| 7 | Cloud configurations. Google Cloud App Engine should be configured for this app. | ALL TEAM MEMBERS |  |
|  |
| 8 | Deploying. This application should be deployed to this cloud. | ALL TEAM MEMBERS |  |
|  |

## **1.5 Project Development Environment**

We are planning to develop the front-end side of the project with Flutter framework which is based on Dart programming language. We are going to deploy our application on both iOS and Android. We are planning to develop the back-end side of the project with Spring boot framework which is based on Java programming language.

## **1.6 Sequence Diagrams**

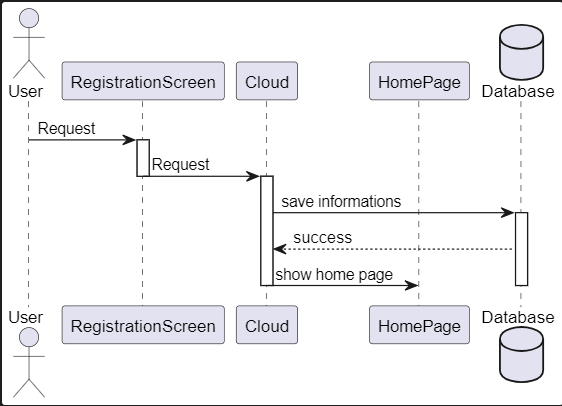


Figure 1 Sequence Diagram for Registration Process

Figure 1 shows that sequence diagram for the registration process of our project.

A diagram of a cloud

Description automatically generated

Figure 2 Sequence Diagram for Send Photo Process

Figure 2 shows that sequence diagram for sending photo process of our project.

A screenshot of a computer screen

Description automatically generated

Figure 3 Sequence Diagram for Open Photo Process

Figure 3 shows that sequence diagram for opening photo process of our project.

## **1.7 Data Flow Diagram**

A diagram of a user

Description automatically generated

Figure 4 Data flow diagram of PhotoChat

Figure 4 shows that context level data flow diagram of our project.

# **2. Progress**

## **2.1 Spring Boot Initializer**

Beginning of development, we set up basic spring boot configurations on Spring Initializr. This tool is published by Spring Framework’s team. This tool basically set up necessary spring dependencies, also we can add some dependencies.

A screenshot of a computer

Description automatically generated

Figure 5 Spring Initializr

As you can see from Figure X, we add some dependencies for this project. Spring Data JPA will help us to SQL operations. Spring Web will help us to create web services. Lombok will help to create method, such as getters, setters, and constructors. Because we are planning to use Google Cloud SQL, we need to use MySQL, so we added MySQL driver to this application. Then we generate basic project file from this page.

## **2.2 Project Structure**

A screenshot of a computer

Description automatically generated

Figure 6 Project Packages

We created some packages in our project to keep it tidy. The “domain” package will contain entities that actually represents tables on database. The “controller” package will contain http requests. The “dto” package, data transfer object, package will contain basic classes. The classes in this package will actually contain objects that carry data. The “service” package will contain service class/eses. We make our services on this/these class/eses. The “exception” class will contain sample exception class for our project. We are planning to create our back-end server as transactional structured. It means that, if any error that don’t want by us happen, we want to roll back the database easily. Spring boot has some annotations. One of them, we can say to framework "Roll back when this exception throws". We are planning to roll back operation when our exception throws. The “repository” package will contain repository interfaces that help us to make database operations. Last package, “security” will contain security configs for authentication processes.

## **2.2 User Entity and Services**

A screen shot of a computer program

Description automatically generated

As you can see from Figure X, on our user entity, there are “id”, “username”, “password” and “friends” attributes. We encrypt user’s password MD5 algorithm before saving to database for security. Also, as you can see above, there is an attribute called “friends”. This attribute holds a list of users whose friend of this user.

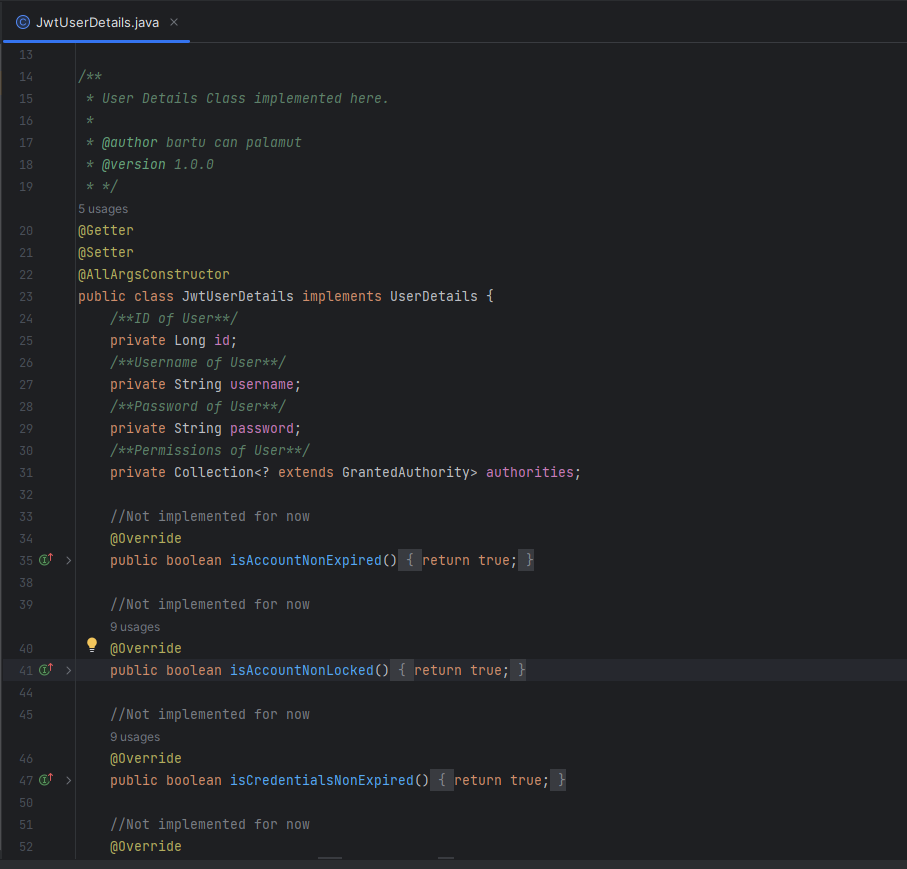
A screen shot of a computer program

Description automatically generated

After that we create simple repository for this class, and we create a very simple SQL command that takes user from the database by id.

## **2.3 Spring Security**

According to the Spring Security Documentation, we need to prepare some implementations for our project for creating security rules.



First of all, we implement “UserDetails” interface. This interface comes from Spring Security dependency. In this interface, there are some methods, but we don’t want to use these methods. We just want to make simple security chain because on our system, there is no user-blocking, role based authentication. We create “JwtUserDetails” class and we add some attributes on this class, such as id of the user, username of the user, password of the user and permission of the user. We are not planning to role-based authentication but based on documentation, we have to implement “getAuthorities” method, so that we add this attribute here too. We are set default role to everyone in the future.

# **3. References**

1. <https://docs.spring.io/spring-security/reference/index.html>
2. <https://pub.dev/documentation/camera/latest/>