Photonest

Object Design

1.0

<15.05.2020>

<Feyzullah Berkay Danış

Özay Ezerceli

Nursena Karakulah

Dilara Ünbay>

Prepared for

SE302 Software Engineering



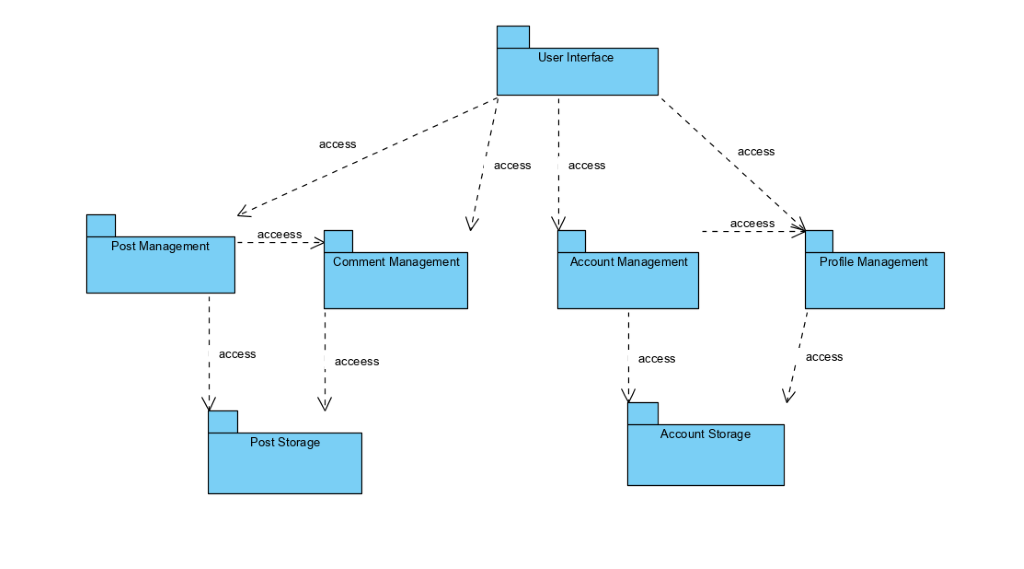
**OBJECT DESIGN DOCUMENT**

1. **Introduction**

Photonest was programmed for people to share pictures online with other people. It is a social sharing application where users can upload and share posts and view posts uploaded by other users, a post would need to include an image and also users can add location information and hashTags to the Post. Users can also see others users posts, make comments on them or like their posts using like function. Users can follow each other which this would inform the users when the other user shares a new post. Each users will have a profile in the system which they can edit their profile information or delete their profile.

Objective of this system is to provide an application that satisfies user needs and requests, and provide every function we have mentioned in the RAD document. The application needs to be simple and enjoyable for users but still effective, therefore user interface of the system has been simplified as much as possible. GUI is made of page, activity and fragment views and functions. The application will not be usable for unregistered users, after creating an account and logging in to the system, registered users will be able to use the application at full functionality. Account information will be kept in firebase realtime database in private, the application has security measures to ensure that the privacy constraints are satisfied according to universal privacy policy.

The subsystem decomposition from which the systems and objects in this document were developed is summarized in the figure below:



**1.1.Object Design Trade-offs**

During the object design of our project we had make several design trade-offs. One of them was deciding between writing the code ourselves or outsource it. We have written almost all the code by ourselves, we used firebase methods during authentication, change password and forgot password functions.

Another trade-off decision we made was between memory space and response time of the application, since we wanted to make this application enjoyable by users we decided to choose response time over memory space because in a mobile application like Photonest a slow response time could make user dislike the app.

We had to decide between delivery time and functionality and we decided to give delivery time more priority. Since we had a certain delivery date, we decided to add as much functionality to the system as we can and deliver it on time.

**1.2.Interface Documentation Guidelines**

In order to create better communication between developers during object design we developed some guidelines for interface documentation. We used singular nouns or phrases for class names that included activity or fragment at the end of the name to indicate weather the class is an activity or fragment and every word in the name started with a capital letter. All the objects created were derived from the class diagram and use cases in the RAD. When an exception is caught, the error is displayed to the user.

**1.3.Definitions, Acronyms, and Abbreviations**

RAD : Requirement Analysis Document

Post : An object that contains an image, hashTags, location which users can view.

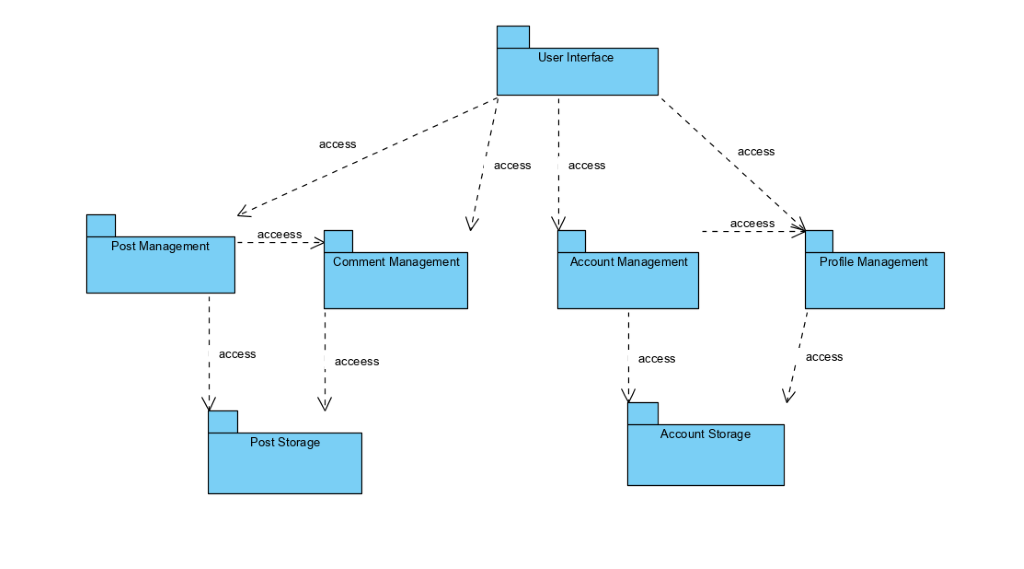
Location : Information of place where a Post was created at.

GUI : Graphical User Interface.

**1.4.References**

* Instagram

1. **Packages**



* 1. **User Interface Package Definition**

**2.1.1 File organization of code for User Interface Package**

This package contains the following files,

res/

,which this file includes the files,

* + anim/ : This file contains the xml files for animations needed in package.
  + drawable/ : This file contains the xml/png/jpeg files for icons and pictures needed in the package.
  + layout/ : This file contains xml files for templates of pages needed in the package.
  + menu/ : This file contains xml files of menu designs’ needed in the package.
  + mipmap/ : This file contains design files of launcher icon.
  + values/ : This file contains references to reach values of multiple used string, color, styles.

**2.1.2 Overview of User Interface package**

This package is responsible of showing reactions received from controller subsystem after the user performed a function. Any data in the model subsystem will be displayed to the user through this package.

**2.1.3 Dependencies with other packages**

User interface package contains layout files that are independent from other packages, this package will display reactions received from other packages.

**2.1.4 Usage of User Interface package**

User interface package will contain the entire GUI and it will provide design layout of every page, activity and fragment of the application that user interacts with. Also it will provide images and icons that were used in application which makes the application visually interesting, menus that users use to navigate through the activities and packages.

* 1. **Profile Management Package**

**2.2.1 File Organization of code for Profile Management Package**

This package will include the following files:

* + com.302.photonest/

This file contains activity and fragment classes that has the functions provided by the Profile management package in them.

**2.2.2 Overview of Profile Management Package**

The Profile Management Package is responsible with handling functions that are related with profile information. It retrieves all user information from firebase database.

**2.2.3 Dependencies with other Packages**

The Profile Management Package is related with Account Management and User Interface. When a user needs to be reached from firebase database user information is first received from Account Management. When a user wants to perform a function provided by this package user interacts with the function through the User Interface Package.

**2.2.4 Usage of the Package**

Profile Management Package is used for user profile functions such as edit profile, change profile photo, delete profile photo. This package is also used to retrieve user data from fireabase and display it to user.

* 1. **Account Management Package**

**2.3.1 File Organization of code for Account Management Package**

This package will contain the following files:

* + com.302.photonest/

This file includes activity and fragment classes that contain code for functions provided by this package.

**2.3.2 Overview of Account Management Package**

Account management handles authentication and other account related functions such as register, login, forgot password, change password logout. It retrieves user’s account information from Account Storage Package.

**2.3.3 Dependencies with other Packages**

Account Management package is dependent on the User Interface Package and Account Storage Package. Users interact with the functions through User Interface Package and Account Management Package retrieves user account information from Account Storage.

**2.3.4 Usage of the Package**

Account Management Package handles account related functions, it retrieves user input from User Interface package and updates data of Account Storage Package and perform appropriate action according to user input.

* 1. **Post Management Package**

**2.4.1 File Organization of code for Post Management Package**

Post Management Package contains the following files,

* + com.302.photonest/

This file contains activity and fragment classes that contains functions provided by this package.

**2.4.2 Overview of Post Management Package**

Post Management Package handles functions related to posts and applies changes to post data on firebase realtime database.

**2.4.3 Dependencies with other packages**

Post Management package is dependent on User Interface Package and Post Storage Package. It retrieves user input from User Interface to perform appropriate function and accesses post data from Post Storage Package.

**2.4.4 Usage of the Package**

Post Management Package performs functions of creating, deleting and editting posts. It receives the required post data coming from user through the User Interface package and updates the post data on Post Storage Package.

* 1. **Comment Management Package**

**2.5.1 File Organization of code for Comment Management Package**

Comment Management Package contains the following files,

* + com.302.photonest/

This file contains activity and fragment classes that contains functions provided by this package.

**2.5.2 Overview of Comment Management Package**

Comment Management Package handles comment related functions and accesses to comment data on Post Storage Package.

**2.5.3 Dependencies with other Packages**

Comment Management Package dependent on Post Storage, User Interface and Post Management. Comment Management Package receives comment data from Post Storage Package, it receives user input from User Interface Package before applying appropriate function, while using functions provided by the package comments reaches to post data through Post Management Package.

**2.5.4 Usage of the Package**

Comment Management Package handles comment related functions such as adding or deleting comment.

* 1. **Post Storage Package**

**2.6.1 File Organization of code for Post Storage Package**

Files related to this package does not exist in application they are held on firebase realtime database.

**2.6.2 Overview of Post Storage Package**

Post Storage Package holds data about posts and comments that are added by users. Application acceses this package for post and data.

**2.6.3 Dependencies with other packages**

Post Storage Package is dependent on Comment Management Package and Post Management Package. Post Storage Package receives input from Comment Management before updating comment data and receives input from Post Management before updating post data.

**2.6.4 Usage of the Package**

This package holds all the information about posts and comments, other packages reaches to Post Storage Package for post and comment data.

* 1. **Account Storage Package**

**2.7.1 File Organization of code for Account Storage Package**

Files related to this package does not exist in application they are held on firebase realtime database.

**2.7.2 Overview of Account Storage Package**

Account Storage Package holds data about user accounts and profiles that are created by users such as username, email, password-are kept as encrypted-, full name.

**2.7.3 Dependencies with other Packages**

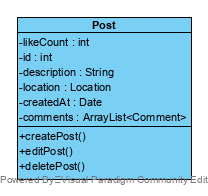
Account Storage package is dependent on Profile Management and Account Management packages. This package receives input from Profile Management Package before updating profile data of user, it receives input from Account Management Package before updating account data.

**2.7.4 Usage of the Package**

Account Storage is used for holding profile and account related data. When a data about profile or account needs to be reached other packages access this Package.

1. **Class Interfaces**
   1. **Post Class**

The Post class allowes users to create new posts , edit existing posts and change their information or delete existing posts. Post class is as follows :



**Class Name : Post**

**Attributes :**

* + *-Likecount : int*

Likecount is value of type int and a private attribute. It shows the number of likes the Post object has.

* + -id : int

Id is an integer type value, it is an identification number, it is unique for every post object. It is a private attribute. It distinguishes every post object created from each other.

* + -description : String

Description is a string type value and a private attribute. It holds the hashTags the post has.

* + -location : Location

location is Location type value and a private attribute. It holds the location the post was created at.

* + -createdAt : Date

createdAt holds a Date type value and it is a private attribute. It holds the date that the post created at.

* + -comments : ArrayList<Comment>

Comments is an ArrayList for comments and it is a private attribute. It holds the list of comments that were made for the post.

**Operations :**

* + +createPost() :

CreatePost() is a void type of function, it is a public function. When this function is invoked, a new post will be created and added to the firebase database.

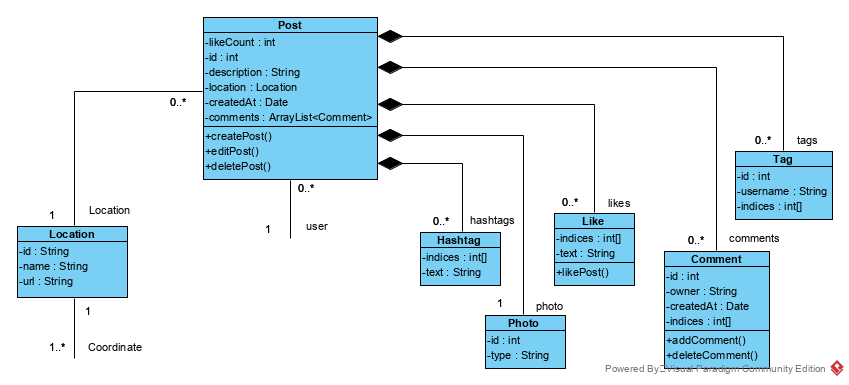
* + +editPost() :

EditPost() is a void type of function that does not return any value. It is a public function. When this function is invoked the information of the choosen post such as it’s description and location is changed.

* + +deletePost() :

DeletePost() is a void type of function that does not return any value. It is a public function. When this function is invoked, the choosen post will be deleted from the firebase database based on the given post id.

**Dependencies with other classes and packages:**



The Post class interacts with Location class, Hashtag class, Photo class, Like class, Comment class, Tag class and Post Storage. When a user creates a post if the user wants to add a hashTag to the post user creates a new Hashtag or uses an existing one, therefore Post interacts with the Hashtag class.While a user is creating a post, if the user wants he/she can add location to the post, therefore Post class interacts with the Location class. While a user is creating a post, user needs to pick a photo from his/her device for post image, therefore Post class interacts with Photo class. When a user likes a post, a like object is created with reference to the post, therefore Post class interacts with the Like class. When a user makes a comment to a post, a comment objects is created with reference to the post, therefore Post class interacts with the Comment class. All the information about the posts are kept in Post Storage package, so for a user to do a Post related function such as creating, editting or deleting, Post class needs to interact with the Post Storage package.

**Exceptions Raised and Exception Handling:**

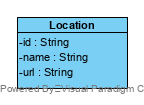
When unexpected(not due to user error) failures occures during creating or deleting post process, a message is displayed on the screen informing user of the failure.

While creating a post, location and caption data can be empty, this was handled using NullPointer exception handling.

While adding location if current location is null, a message is displayed informing the user that since current location is null, a default location will be used.

**3.2 Location Class**

Location class is reached during post creation. If a user wants to add location information to the post he/she created, a Location object is created for the procedure. Location class is as it seem below.



**Class Name : Location**

**Attributes:**

* + -id : int

A private attribute of String type. This attribute holds the ID number of the Location object, used for distinguishing Location objects and it is unique for every Location object.

* + -name : String

A private attribute of String type. This attribute holds the name of the Location.

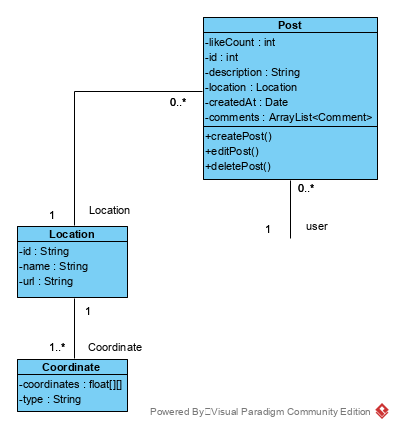
* + -url : String

A private attribute of String type. This attribute holds the URL of the post image that Location will be added to.

**Operations:**

This class does not have any operations, since it’s only use is to be added to Post object.

**Dependencies with other classes:**



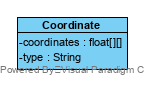
The Location class is in interaction with Post class and Coordinate class. Location objects are created during post creation if user wants to add location to the post. Therefore Location object needs a Post object for creation. The Location class interacts with Coordinate class. While the location object is being created, it retrieves coordinate information from a Coordinate object.

**Exceptions Raised and Exception Handling:**

This class does not raise any exceptions because it does not have any functions that provide service inside the class.

**3.3 Coordinate Class**

Coordinate class is used when a Location object is created for getting the coordinates of location. Coordinate class is given below.



**Class Name : Coordinate**

**Attributes :**

* + -coordinates : float[][]

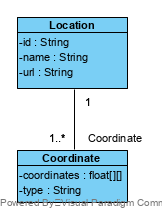
It is a private attribute of float array type. It holds the coordination values of the object.

* + -type : String

**Operations:**

This class does not contain any functions inside.

**Dependencies with other classes:**



Coordinate class interacts with Location class, while a Location object is being created, coordinate information is gathered using a Coordinate class, therefore a Location class needed before creating a Coordinate class.

**Exceptions Raised and Exception Handling:**

There is no exception handling in this class since it does not provide any functions.