Software Design Description

CapGrab
Caption Sharing/Forum Mobile Application

Table of Contents

6.1. Introduction

This document provides the architecture and detailed design for the software for the CapGrab project. CapGrab is an iOS application that allows users to get crowdsourced captions for photos that they are intending on posting to social media.

6.1.1 System Objectives

The objective of this application is to provide users with a crowdsourced caption generating app. Users will post photos for other users to add captions to, allowing the poster to pick their favorite caption when they post the photo to their social media of choice. Users will additionally vote on others captions giving them a caption score.

6.1.2 Hardware, Software, and Human Interfaces

6.1.2.1 Human Interface: Phone Screen In order to navigate through the interface of the application, the user will tap on the phone screen. This will allow for the user to initiate certain events and explore the available pages.

6.1.2.2 Human Interface: Machine To access the application, the user will need an iOS device, such as an iPhone or iPad.

6.1.2.3 Software Interface: Graphical User Interface The GUI is made up of a tab bar for navigation, multiple types of buttons for specific actions, collection views to hold user images, and a search bar to find other CapGrab users.

6.1.2.4 Software Interface: Database The database holds all the data besides user images for the application. This data includes user information, photo storage paths, and caption information.

6.1.2.5 Software Interface: Storage The storage system holds all of the photos uploaded to CapGrab. The image storage paths are then saved in the database in order for the photos to be accessed for the interface.

6.1.2.6 Hardware Interface: Wireless/Wired Networking In order for the application to access the database and storage, the phone needs to have connection to the internet.

6.2 Architectural Design

The design of the CapGrab application is made up of three pieces: the mobile frontend, the Firebase database, and Firebase storage. The mobile frontend displays all of the pages of the application. The Firebase database both sends and receives information to and from the frontend. Finally, the Firebase storage holds all the user images uploaded to the application.

6.2.1 Major Software Components

6.2.1.1 Mobile User Interface - The mobile frontend is made up of seven pages: Login/Sign Up Page, Account Page, Search Page, Upload Page, Notification Page, Forum Page, Individual Photo Page. First, the Login/Sign Up Page will handle account creation or login. It is shown when the user logs out of their account and when a user opens the application and they either are logged out or do not have an account. The Account Page shows the user's account information and photos. Next, the Search Page offers a search bar, which the user can use to search for other user accounts. The Upload Page

6.2.1.2 Database Our database stores all forum activity such as posts, replies, and upvotes, as well as user account information, such as user ID, (paths to) photos, email, username, etc. Queries can be used to access the information within the database and apply it to the user interface, as well as save information, gained from the user interface, into the database. The database serves as our bridge between our user interface and our storage.

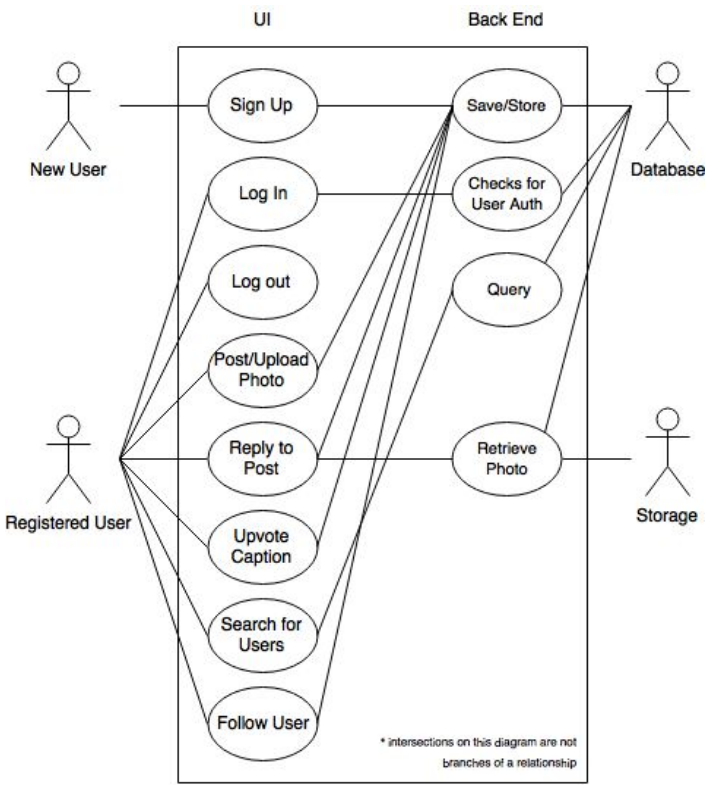
6.2.1.3 Storage Our storage stores all of our user-generated content, specifically photos. The database stores paths (urls) to specific photos, which points to and accesses our storage to retrieve that specific photo.

6.2.2 Major Software Interactions

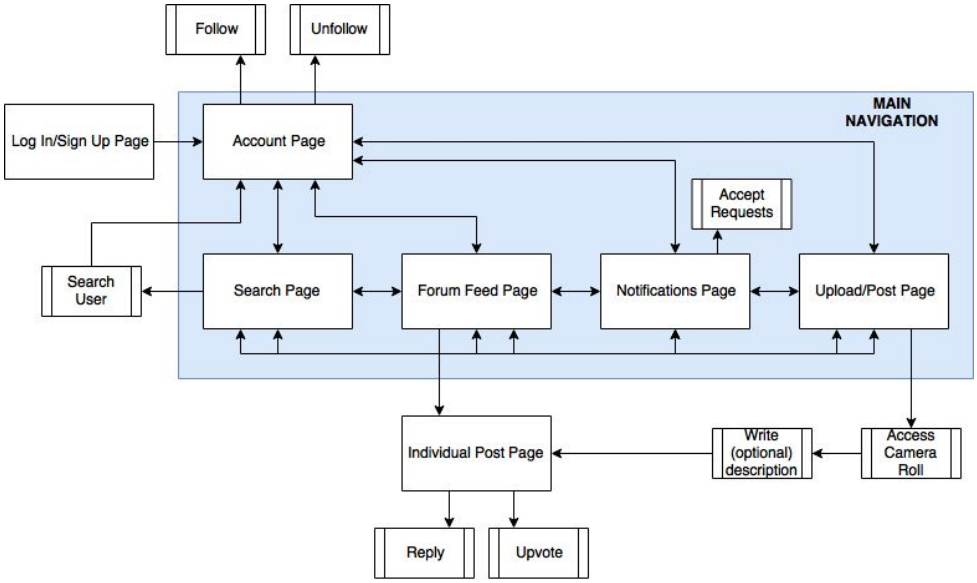
Our entire user interface is laid out in Swift and built through XCode. Swift allows us to communicate to our database, which communicates with our storage, both of which are hosted through Firebase, which removes the need for a server. The database acts as our bridge between the user interface and storage, as the storage stores all of our user-generated content (photos), which are fetched through paths (URLs) generated by Firebase in our database.

6.2.3 Architectural Design Diagrams

Use Case Diagram



State/Action Diagram



Class Diagram

6.3 CSC and CSU Descriptions

Mobile Front End CSC

Our user interface is our client-side component for our CapGrab application. Login/Sign Up Page, Account Page, Search Page, Upload Page, Notification Page, Forum Feed Page, Post-Specific Page.

- Login/Sign Up Page CSU - This unit handles user authentication and account creation
- Account Page CSU - This unit allows users to have an overview of their own or another user's account. It is comprised of a photo grid of all their photos that they have requested, their profile picture, their follower and following numbers, and their CapGrab score
- Search Page CSU - This unit allows a user to search for other users in the CapGrab community
- Upload Page CSU - This unit allows the user to upload a photo from their camera roll, and compose a post requesting for captions on that photo, and publishes it to the CapGrab forum feed
- Notification Page CSU - This unit is holds a list of notifications regarding any account activity that has occurred since the user's last session (closing of the app)
- Forum Feed Page CSU - This unit is a scrollable feed consisting of posts from users that have requested caption suggestions for their photos. It is comprised of thumbnails, on which users can tap and be brought to the post-specific page to submit a caption suggestion or upvote previously suggested ones.
- Post-Specific Page CSU - This unit is a forum thread, specific to a post/photo that a user is requesting a caption for. Other users will be able to submit their caption suggestions as well as upvote any captions that have already been suggested.

Database CSC

Our database stores all forum activity and user account information. The database serves as our bridge between our user interface and our storage.

- Querying CSU - This unit runs queries on stored data to allow for retrieval
- Indexing CSU - This unit indexes data that is stored
- Saving CSU - This unit allows for data to be stored

Storage CSC

Our storage stores all of our user-generated content, specifically photos. The database stores paths (urls) to specific photos, which points to and accesses our storage to retrieve that specific photo.

- Photo CSU - User-generated and uploaded images; paths to this unit are stored in the database in the form of uniform resource locators (URLs)

6.3.1 Detailed Class Descriptions

6.3.1.1 Database

- Fields:
  - Users: defines the users nodes, which are created by the users.
  - Photos: defines the photos nodes, which are paths to user images in Firebase storage.
- Methods:
  - setValue(): creates or updates new nodes in the database.

6.3.1.2 Storage

- Fields:
  - Profile Pictures: holds all the user profile pictures
  - Account Images: holds the images uploaded by each user.
- Methods:
  - putData(): uploads images to storage.

6.3.1.4 ViewController (User Account Page):

- Methods:
  - viewDidLoad(): loads the design of the view controller and receives data from the database
  - userNameVisibility(): adjusts the design of the page based off the value of the segment controller.
  - selectProfilePicture(): opens the photo library when the “select profile picture” is tapped.
  - loginOrSignUpAction(): depending on the value of the segment controller, either creates an account for the user or logs them in.

6.3.1.5 UserAccountViewController:

- Methods:

- viewDidLoad(): loads the design of the view controller and receives data from the database.

6.3.1.6 UserImageViewController:

- Fields:
  - userImage: holds a single user image for the collection view of the user account page.

6.3.1.7 UserSearchViewController:

- Methods:
  - viewDidLoad(): loads the design of the view controller.
  - tableView(): sets up the table view, which holds the returned usernames.
  - searchBarSearchButtonClicked(): takes the value entered in the search bar and queries the database for the usernames that match the value.
  - prepare(): sets the values of the next view controllers that will be segued to.

6.3.1.8 ReturnedUserTableViewCell:

- Fields:
  - userID: holds the userID for each table cell in the UserSearchViewController.

6.3.1.9 SearchedUserAccountViewController:

- Methods:
  - viewDidLoad(): loads the design of the view controller and calls the reloadPage function.
  - userRelationAction(): adjusts the user account information in the database when the follow button is tapped.
  - collectionView(): sets up the collection view that holds the user images.
  - backToSearch(): brings the user back the UserSearchViewController.
  - reloadPage(): queries the database to get information to set up the page.

6.3.1.10 SearchedUserImagesCollectionViewCell:

- Fields:
  - userImage: olds a single user image for the collection view of the SearchedUserAccountViewController.

6.3.1.11 NotificationViewController:

- Methods:
  - viewDidLoad(): loads the design of the view controller and receives data from the database.
  - captionOrFollow(): reloads the table view based off the value of the segment controller.
  - tableView(): sets up the table view, which holds the notifications.

6.3.1.12 NotificationTableView:

- Methods:

- notificationAction(): updates the user account following, follower, and follow requests data in the database when the button in the table view cell in tapped.

6.3.1.13 ForumViewController:

- Methods:
  - viewDidLoad(): loads the design of the view controller and receives data from the database.

6.3.2 Detailed Interface Descriptions

6.3.2.1 Login/Sign Up CSU The user either creates an account by setting their email, password, username, and profile picture, or logs in by entering their email and password.

6.3.2.2 Account CSU The user can view their photos, amount of followers, amount of users they are following, and profile picture.

6.3.2.3 Search CSU The user can search for a particular user by entering their username in the search bar. The search then presents a list of users in the table.

6.3.2.4 Upload CSU The user can select a photo from their phone's photo library, which can then be uploaded to their CapGrab account.

6.3.2.5 Notification CSU The user is presented with a table of notifications, including follow requests and caption information. The user is also able to accept follow requests and navigate to other user accounts.

6.3.2.6 Forum CSU The user can scroll through multiple public user photos and give caption ideas as well as upvote or downvote other captions.

6.3.2.7 Individual Photo CSU The user can observe the image, as well as give caption ideas and/or upvote or downvote other captions.

6.3.3 Detailed Data Structure Descriptions:

6.3.3.1 Login/Sign Up CSU This CSU creates multiple arrays to hold specific user information, such as followers, following, follow requests, notifications, and photo paths. All of the arrays are empty upon account creation.

6.3.3.2 Account CSU: This CSU makes makes use of the user's followers, following, and photo path arrays from the database to populate the page. The followers and following arrays are used to present the number of followers and users that the current user is following. Furthermore, image paths from the photo path array are used to download the user's photos from Firebase storage.

6.3.3.3 Search CSU: Makes use of the username data within the database.

6.3.3.4 Upload CSU: This CSU updates the user's photo path array by appending the new photo path to it. The database is then updated with the new array.

6.3.3.5 Notification CSU: This CSU makes use of the user's notifications, follow requests, and followers arrays from the database in order to populate the notification

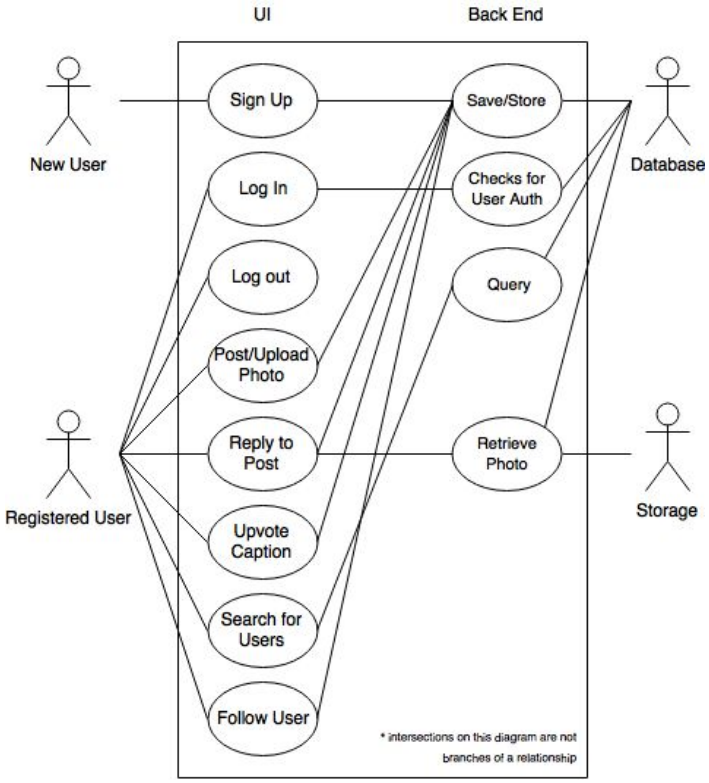
table. When certain user actions occur, the arrays are adjusted and sent to the database.

6.3.3.6 Forum CSU: This CSU makes use of the public photo paths array which is held in the database. The paths in this array are used to populate the table view with the images from Firebase storage.

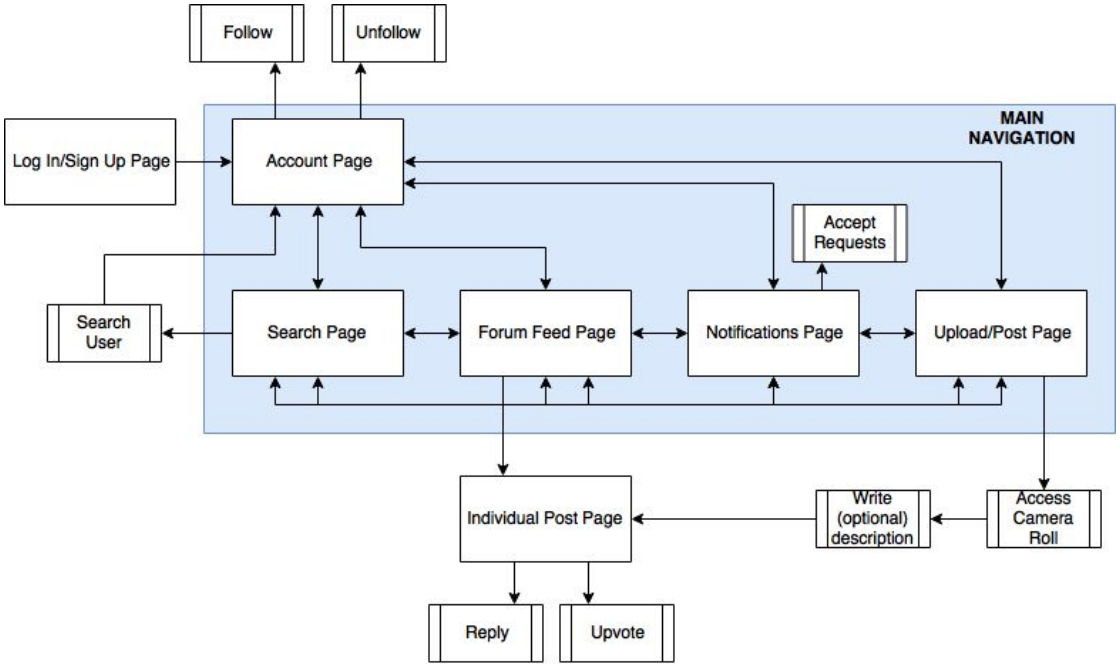
6.3.3.7 Individual Photo CSU: This CSU makes use of the caption objects, which are located in the database. Each object holds a particular caption, as well as information about that caption, associated with the presented photo on the individual photo page. When the user adds a caption to the photo, another caption object is created and saved to the database under that particular photo.

6.3.4 Detailed Design Diagrams

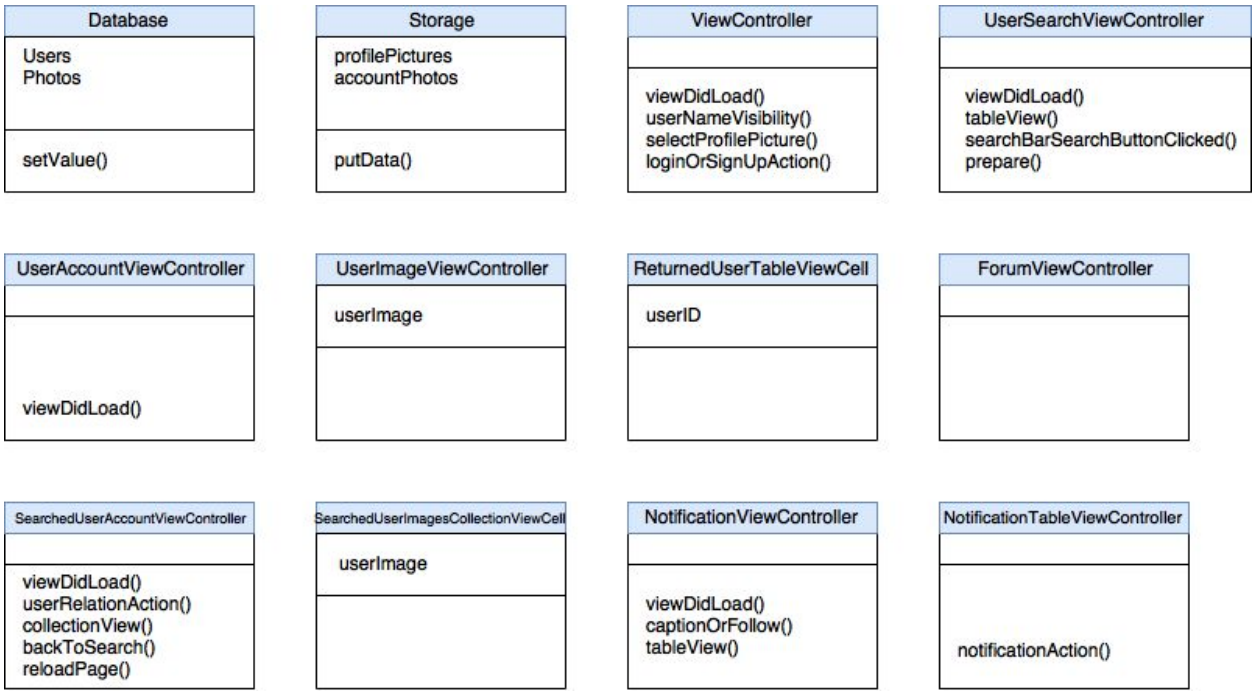
Use Case Diagram



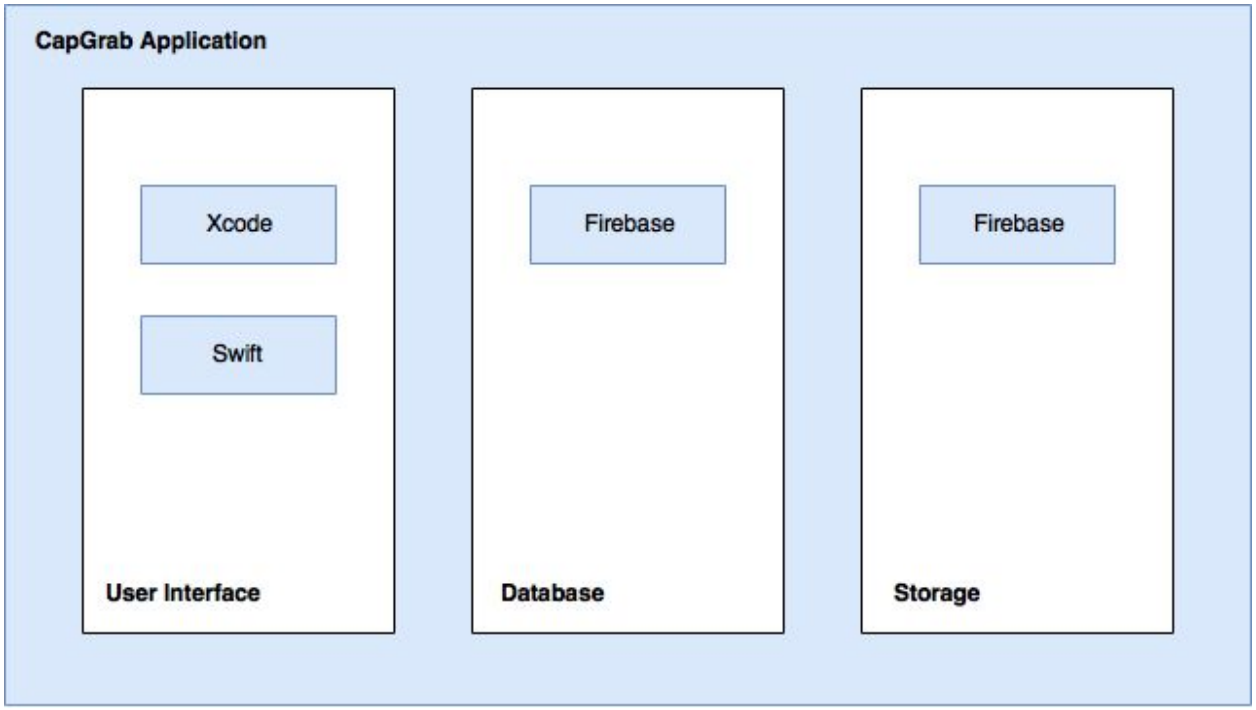
State/Action Diagram



Class Diagram

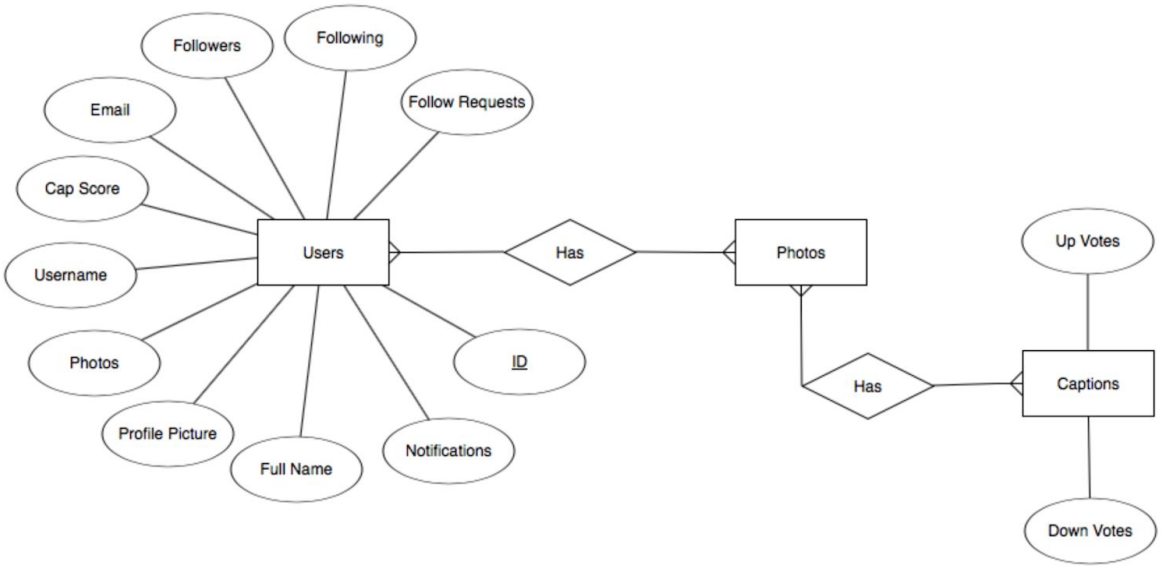


Package Diagram



6.4 Database Design and Description

6.4.1 Database Design ER Diagram:



6.4.2 Database Access:

CapGrab is using Google Firebase's realtime database to store the data for the application. The setup is extremely convenient, and connection to the database is done through simply creating a reference object to the database in the code. The reference object has a set of functions that can then be used to query the database.

The database can be accessed by the users through the frontend. The user has the ability to adjust most of the data in the database that pertains to them, such as their username, photos, followers, and more. They can also access other user's data by going to other user accounts and following them. This being said, a user cannot change another user's data in the database.

6.4.3 Database Security:

Google Firebase offers an abundance of database security measures, such as password hashing and user ID creation. Firebase also allows for the database manager to create a set of read, write, and validate rules for different levels of the database. Furthermore, only the database manager, who is a member of the CapGrab team, can access and manipulate all of the data in the database through the database console. CapGrab users will only be able to update their own data through the Frontend of the application. Due to all of this, no further database security needs to be implemented within the code by the CapGrab team.