MILESTONE REPORT 2

COLUMBUS

GROUP 7

Table of Contents

Executive Summary 2

Summary of Individual Works 2

Progress According to Requirements 2

API Endpoints 2

Implementation of Core Functionality 2

User Experience\User Interface 2

Standards 2

1. Executive Summary

Contents:

We are Bogazici University Computer Engineering students. This report mainly contains our group work in CMPE 451 so far. You can see the evolution of our project in this report. It contains status of our project, progress of the requirements, individual efforts of team members, implementation of core functionality, screenshots of user interface

Project Status:

In this milestone, we focused on the main frame of application and user interface. We tried to create all components that we will use in the future. We thought that once we created pages and components properly, we can make changes on them easily. So initially we created authorization mechanisms and page navigation mechanisms of our app. We created post and map components. We implemented the guest role. We added a profile page and follow mechanism. We implemented a share post mechanism.

There will be some changes in our components and page designs in order to be compatible with our requirements. However we created the structure of components, pages and backend integration. We will implement searching and exploring mechanisms. We focused on finishing this application and meeting our requirements in the final milestone, so we are moving towards this goal.

Infrastructure:

What we have done:

- Created a S3 bucket for storing user images and story multimedia.

What we will do until Final Milestone :

- The S3 bucket API key is exposed. The API key will be added to GitHub secrets.

Frontend:

What we have done:

- Integrated Google Maps API for our application
- Design the authorization mechanism with the following functionalities:
    - Home page for guest role
    - Sign in navigation to the login page
    - User information and jwt token stored in local storage
- Added Create Story page with following functionalities. User can:
    - Add a topic to story
    - Add a story text to story
    - Add tag(s)
    - Add either a virtual location or three real locations. The real locations must have a value on the map
    - Add story start and end date to

- Created profile page with the following functionalities. User can:
    - View her profile information: username, email. birthday, name, last name, location and biography.
    - Edit her profile information with birthday, name, last name, location and biography.
    - View her post stories.
    - View her liked stories
    - also view other user profiles with all the functionalities above but can not edit other users profile
- Story component redesigned. Now user can:
    - like and comment posts
    - can expand stories to view the full post
    - view stories date and location data
    - view multimedias of story
- Tried to write unit tests for React.
    - Had problems with the React structural design to mock external libraries

What we will do until Milestone 3:

- The creation and end date for the stories are too specific. We will improve it by adding different types of dates which are more flexible.
- The explore page will be added with the following functionalities. User can:
    - For real locations see the stories on the map and select the marker to view the stories in there.
    - For real locations user can search an area in map to view the stories in that area
    - For virtual location user can filter by the location names
    - For all types of locations date filtering will be present
    - For all types of location tag filtering will be present
- Story component changes:
    - Profile images will be in the avatars of the stories
    - Date of the story component will be human readable
    - Comment of comment will be implemented
    - Number of likes and comments will be visible
- Search functionalities will be added to search posts and users.
- Notification system
- A testing mechanism will be added

Mobile:

What we have done:

- Refactor general application, register and login pages
- Run in the android emulator
- Add service structure for request to api
- Add authentication structure:
    - Add AsyncStorage to application for keep some common informations

    - Add authentication context which is AuthContext
    - Add React Query for request to take advantage of its benefits
    - Refactor navigations according to logged in user data
- Add Story Card Component:
    - Design Component in general
    - Add Image Carousel to show images
    - Add User Information (Avatar + username +link to User Profile)
    - Add Location Information field (+link to Location Page)
    - Add Story Time
    - Add Posting Time (relative to user)
    - Add Like and Comment
    - Add Tags field
    - Add like functionality
- Add icon libraries to application:
    - Add Vector Icons
    - Add configs for icons to two of the client which are ios and android
    - Refactor bottom tab bars
    - Add icons to bottom tab bar
    - Add default pages after login
- Remove redundant files from application
    - Remove redundant files
    - Refactor folder structure of the application
    - Update the document of application which is the ReadMe file
- Try to add unit test structure for mobile app, however we won't add to unit tests
    - Add mock async storage
    - Add mock navigation base
    - Add mock test setup
- Create Detailed Story Page
    - Create Page to show one specific Story
    - Add navigation from Story Card to page
    - Create Map to show one or more location at the same time
    - Add the same functionalities with Story Card
- Create Location Page
    - Create Page navigated from Story Card location field
- Create Profile Page
    - Add Page UI with mock data
    - Add Edit Profile page with mock data
    - Add upload image library which is React Native Image Picker
- Create Story Page
    - Add create story page UI
- Add Comment feature
    - Add Comment Component Designs
    - Add Comment Action Sheet

    - Fetch Comment from API
    - Comment functionality with API

.

- Create Story Page
    - Add create story page UI
- Add API connection for Profile page
    - Add API connection for profile page
    - Get user information from Auth Context
- Add API connection for Edit Profile Page
    - Get user information from profile page with react navigation params
    - Add api connection for edit profile page
- Add API connection for Create Story Page
    - Add api connection for create story page

What we will do until Milestone 3:

- Refactor create story page and edit profile page:
    - Add image upload to aws
    - Add google maps library for react native (implemented in Story Page)
    - Choose location from google maps
    - Choose free date inputs
    - Add validation of informations
- Search functionalities will be added to search posts and users. (also by tag)
- A testing mechanism will be added
- Add view followers and followings users
- Add follow functionality

Backend:

What we have done:

- We decided to use Django as a framework and postgresql as database management system
- We started the project from scratch.
- We dockerized our project.
- Authentication Mechanism:
    - We implemented endpoint for login
    - We implemented endpoint for register.
    - We implemented an email verification mechanism.
    - We implemented change password functionality.
- Guest Options:
    - We implemented home page for guests which shows all posts chronologically.
- User Options:
    - We implemented post creation API.
    - We implemented post edit API.

- We implemented post delete API.
- We implemented comment creation API.
- We implemented comment edit API.
- We implemented comment delete API.
- We implemented get comments of a story API.
- We implemented the follow API.
- We implemented user likes API, which returns the liked stories by the user.
- We implemented the home page API for users, which returns the stories of followings.
- We implemented like story API, which both likes and dislikes a story.
- We implemented profile page API, which shows all the stories belonging to the user.
- We implemented set profile API, which allows changing the profile information.

What we will do until Milestone 3:

- Block and unblock users API will be added.
- Profile deletion API will be added.
- Pin comments API will be added.
- Report user API will be added.
- Public and private profile API will be added.
- Search by username API will be added.
- Search by location name API will be added.
- Search by geographical location API will be added.
- Search by tags API will be added.
- Search by time API will be added.
- Recommendations by user location API will be added.
- Admin register API will be added.
- Admin login API will be added.
- Admin change password API will be added.
- Admin email confirmation API will be added.
- Show reported users API will be added.
- Show reported posts API will be added.
- Show reported comments API will be added.
- Show reported tags API will be added.
- Adding user to black list API will be added.
- Removing user from black list API will be added.
- Putting posts to spam folder API will be added.
- Putting comments to spam folder API will be added.
- Putting tags to spam folder API will be added.
- Notification base will be written.
- Post like notification API will be added.
- Post comment notification API will be added.
- Comment reply notification API will be added.

- New follower notification API will be added.
- Following request API will be added.
- Following request notification API will be added.
- Following request accept/reject API will be added.
- Tagging system will be supported with https://wikidata.org./

2. Summary of Individual Works

Member	Work
Abdulkadir Elmacı	- Added Backend API service for calling backend functions - Created a page for sharing posts - Created a S3 bucket for multimedia upload - Created a service structure in backend for uploading images to S3 - Added maps api integration to share post page - Fixed a bug where authorization header was not set after authenticating - Fixed a bug where user profile redirections was not working - Fixed a bug where user can edit other users profile information - Fixed a bug where follow button was not updating even after a user follows another user - Fixed an alignment bug where the follow button was skewed - Writing executive summary for milestone #1 and #2 w/ Erencan
Erencan Uysal	- Recreated register dialog screen - Taking meeting notes for Meeting #18 (30.11.2021) - Designed post component - Created location list dialog for post component - Changed field names across the app to be compatible with backend - Implementing authentication mechanism - Changed navigations to fix the logic of page order - Change the design of header for guest role

	- Changed the post component for the guest role - Research about typescript and google maps api - Implemented a google maps object - Created a dialog screen that returns latitude and longitude from map object - Integration between backend and post component - Implemented like functionality - Implementing make comment and see comments functionalities - Wrote exclusive summary for milestone 2 with Abdulkadir - Wrote implementation of core functionality for milestone 2 report - Wrote integration test with cypress - Attended weekly meetings and frontend team meetings - Reviewed pull requests of frontend team

Hamza Akyıldız	- Research about django models, rest-APIs, swagger, authentication and django serializers
	- I added swagger extension to implement backend environment detection for local-live-dev
	- the backend
	- I refactor old APIs to work with swagger
	- Django migrations bug detection and fixing
	- Django Model design and creation
	- I added authentication to the swagger
	- Bug fixing for backend deployment problems
	- API implementation for:
	  - Get profile page
	  - Set profile page
	  - Like/unlike post
	  - Follow/unfollow user
	  - Get user shared posts
	  - Get user liked posts
	  - Get users who like a story
	- Unittest implementation for:
	  - Get profile page
	  - Set profile page
	  - Get user shared posts
	  - Get user liked posts
	- Api refactoring according to frontend and mobile feedbacks

	- Documentation:
	  - API Summary for group milestone report
	  - Personal efforts
	  - Individual Milestone report
	- Meetings:
	  - Weekly team meetings
	  - backend team meetings

Merve Rabia Barın	- Added Infinite Scrolling mechanism to Home page
	- Redesigned the Header component:
	  - Removed Home page button
	  - Added text to navigation bar buttons
	  - Put a redirect to Columbus logo (it opens home page)
	- Added user's own Profile Page and other user's Profile page UI for web
	- Connected get profile info, shared/liked posts APIs with Profile Page
	- Added viewing followings/followers UI for web
	- Connected get followings, followers APIs with it.
	- Added Edit Profile Information UI for web
	- Connected set profile info API with Edit Profile Information
	- Organized the code of Header component
	  - Put styles in different file
	  - Added router for Profile page
	- Made Infinite Scrolling a component
	- Connected get home page posts for a user and guest API with Home page
	- Added links and screenshots of Web UI of Columbus app to Milestone report 2

Onur Can Avcı	- Refactor general mobile app - Add an icon libraries and Refactor Bottom tab bars and pages - Authentication Structure for mobile app - Review open pull requests for frontend and mobile - Analyze the tasks on mobile app - Research Activity Stream - Add Profile page and Edit Profile Page UI for mobile app - Hot/Fix Set token after login for mobile app - Create Story Page UI for mobile

	app - Api Connection in Profile Page for mobile - Api connection in Edit Profile Page for mobile - Api Connection in Create Story Page for mobile - Refactor general mobile app for api connection - Write 2nd Individual Report - Write Progress Based on Teamwork for Mobile App
Ramazan Bulut	- Research on React Native - Create Story Component - Create generic User Information Header - Create generic Posting Time Component - Creating Tasks for Mobile - Create generic Tag Component - Create Comment Component - Create Like Component - Create Comment Action Sheet - Create Detailed Story Component - Create Map Component - Create Home Page - Create Location Page navigated from Story Component - API integration for Home Page - Edit Profile Page to show other's profile (not editable - Write 2nd Milestone Individual Report - Review Pull Requests - Api Integration for Comment - Api Integration for Like - Add Google Maps dependency and integration for Detailed Story Page
Umut Kocasarı	- Research about backend implementation, Django, best practices and details of Django - Research about activity stream, its usage and practices especially in Django - Helped to implement the activity stream, solved potential problems and tested it - Implemented backend environment detection for live and test - Added token authentication for session management

	- Added logout functionality and expired users' tokens - Solved database migration errors by changing both migrations and docker-compose files - Implemented the search of users' posts functionality for their profile page - Implemented post creation feature with all its attributes - Added tags input to the post creation API - Implemented the is liked feature for posts returned for a specific logged in user - Added location information for posts returned after search - Implemented the functionality of tags and url photos returning with post search - Implemented the home page story functionality by showing the stories of followings - Implemented the guest page story functionality by showing all the stories chronologically - Added comment creation functionality - Added comment update functionality - Added the functionality of showing comments of a story - Took meeting notes for week 07.12.2021 - Attended weekly team and backend meetings - Wrote swagger documentation for each API written by me - Wrote unit tests for Django models and their behaviors - Wrote unit tests for post deletion - Wrote unit tests for post edit - Wrote unit tests for logout - Wrote unit tests for home page - Wrote the progress according to requirements part of the report - Helped writing for the what we have done for the backend
Yağız Efe Şabanoğlu	- Research about activity stream, its usage and practices especially in Django.

	- Research about django models, rest-APIs, swagger, authentication and django serializers
	- Unittest implementation for:
	  - Update Comment
	  - Delete Comment
	  - Create Comment
	  - Register
	  - Login
	  - Change Password
	  - Confirm Email
	  - Get Comment
	  - Activate Production Token
	  - Activate Development Token
	  - Like Post
	  - Post Create
	- All Django Models design and creation
	- I reformatting old APIs I wrote to new formats.
	-
	- API implementation for:
	  - Get Home Page
	  - Activity Stream
	
	- Some Bugs fixing for backend deployment problems
	- Api refactoring according to frontend and mobile feedbacks.
	- Standardization for group milestone report

3. Progress According to Requirements

X : Completed
X : In progress
X : Not Started

1. Functional Requirements

1.1 User Requirements

1.1.1 Profile

1.1.1.1 Users shall be able to block other users.
1.1.1.2 Users shall be able to view any of their personal information (email, username and name, location, birthday if provided).
1.1.1.3 Users shall be able to edit any of their personal information (email, username and name, location, birthday if provided).
1.1.1.4 Users shall be able to see users that they follow and are followed by.
1.1.1.5 Users shall be able to delete their accounts.

1.1.2 Posting

1.1.2.1 Users shall be able to create posts with a story.
1.1.2.2 Users shall be able to add tags, photo, video and voice to their posts.
1.1.2.3 Users shall be able to add a location with its name or by specifying the location on the map interface.
1.1.2.4 Users shall be able to add multiple locations related to their story.
1.1.2.5 Users shall be able to specify time in their posts to declare exact time or time interval with a maximum of 10 years.
1.1.2.6 Users shall be able to edit their own posts.
1.1.2.7 Users shall be able to delete their own posts.
1.1.2.8 Users shall be able to like the post of their followings.
1.1.2.9 Users shall be able to comment on the post of their followings.
1.1.2.10 Users shall be able to view users who liked or commented to their posts.
1.1.2.11 Users shall be able to delete comments made to their own posts.
1.1.2.12 Users shall be able to reply to comments made by other users with comments.
1.1.2.13 Users should be able to save the posts of their followings.
1.1.2.14 Users should be able to pin comments made to their own posts to the top.

1.1.3 User to User Interaction

1.1.3.1 Users shall be able to view the posts of their following accounts.
1.1.3.2 Users shall be able to report other users posts, if the user has suspicion about its accuracy.
1.1.3.3 Users shall be able to follow other users.
1.1.3.4 Users shall be able to unfollow other users.
1.1.3.5 Users shall be able to view posts of other users that they follow or have public profile.
1.1.3.6 Users shall be able to view the followers and followings of other users that they follow or have public profile.

1.1.4 Searching

1.1.4.1 Users shall be able search other users by their username.
1.1.4.2 Users shall be able search locations by their name.
1.1.4.3 Users shall be able to specify the searched location on the map.
1.1.4.4 Users shall be able to do semantic search by using tags.
1.1.4.5 Users shall be able to filter the search results according to time.

1.1.5 Recommendation

1.1.5.1 Users shall be able to view recommended locations based on their system activity and current location.

1.2 Guest Requirements

1.2.1 Guests shall be able to create a new account by choosing a password (email confirmation) and providing email, username (name, location, birthday optionally).
1.2.2 Guests shall be able to login to the system by entering username and password if they already have an account.
1.2.3 Guests shall be able to only search and view locations.

1.3 Admin Requirements

1.3.1 Admins shall be able to login to the system by entering username and password.
1.3.2 Admins shall be able to review any content that is reported by users.
1.3.3 Admins shall be able to filter reported contents for the types of posts, comments,tags and accounts.
1.3.4 Admins shall be able to add users to Black List.
1.3.5 Admins shall be able to remove the users from Black List.
1.3.6 Admins shall be able to add the content of post, comment, and tags into the spam folder.
1.3.7 Admins shall be able to use any search functions that are used by users.

1.4 System Requirements

1.4.1 Sign Up

1.4.1.1 System shall require e-mail confirmation after sign-up.
1.4.1.2 System shall not allow multiple registration with the same e-mail.

1.4.2 Login

1.4.2.1 System shall require a password and a unique username to login.
1.4.2.2 The password shall be at least 8 characters.
1.4.2.3 The password shall consist of alphanumeric characters.
1.4.2.4 The username shall consist of alphanumeric characters.

1.4.3 Recommendation

1.4.3.1 The system should recommend posts and locations to users according to their activity on the system and their current locations.

1.4.4 Notifications

1.4.4.1 The system shall support a notification mechanism.
1.4.4.2 The system shall notify the user if the following events occur via notification box.
1.4.4.2.1 Any activity that happened on a user's post
1.4.4.2.1.1 Post is liked
1.4.4.2.1.2 Post is commented
1.4.4.2.1.2 A comment on the post is replied
1.4.4.2.2 Any activity that happened on a user profile
1.4.4.2.2.1 The user has a new follower
1.4.4.2.2.2 The private user gets following request
1.4.4.2.2.3 The user following request gets accepted

2. Nonfunctional Requirements

2.1 Portability

2.1.1 The system shall have an Android application and a web page.

2.2 Accessibility

2.2.1 Language of the Android application and we page shall be English.

2.3 Performance

2.3.1 The system response should be in 3 second.

2.3.2 The system should handle 500 requests per second.
2.3.3 The system's Android application size should be less than 200 MB.

2.4 Reliability

2.4.1 The system shall go down for maintenance once a month.
2.4.2 The maintenance shall not take more than 30 minutes.
2.4.3 Rest of time, downtime should be less than %1 for failures.

2.5 Security

2.5.1 The passwords shall be kept encrypted in database.
2.5.2 The web application shall only be accessed by HTTPS protocol.

2.6 Legal and Ethical Issues

2.6.1 Any data collected from users shall be used according to KVKK and GDPR.
2.6.2 New users shall accept "Terms of Use" and "Privacy Policy".

2.7 Implementation

2.7.1 The system shall use W3C standards and W3C Activity Streams Protocol for subscriptions and notifications.
2.7.2 The tagging system should be supported with Wikidata.org.

2.8 Robustness

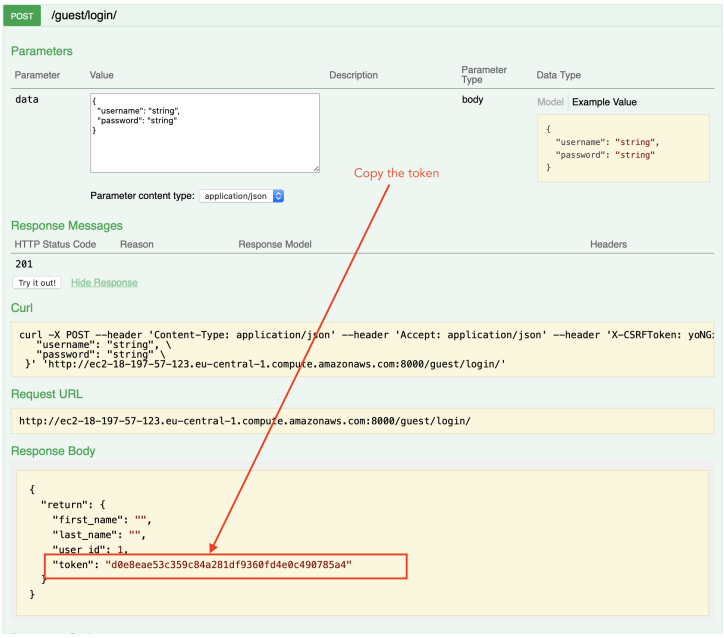
2.8.1 System should restart the application in 45 minutes after failure.
2.8.2 Server admins should restart application manually in big failures.
2.8.3 Restart after big failures should take up to 4-5 hours.
2.8.4 Less than %2 of events/features should cause system failure.

4. API Endpoints

API Documentation:

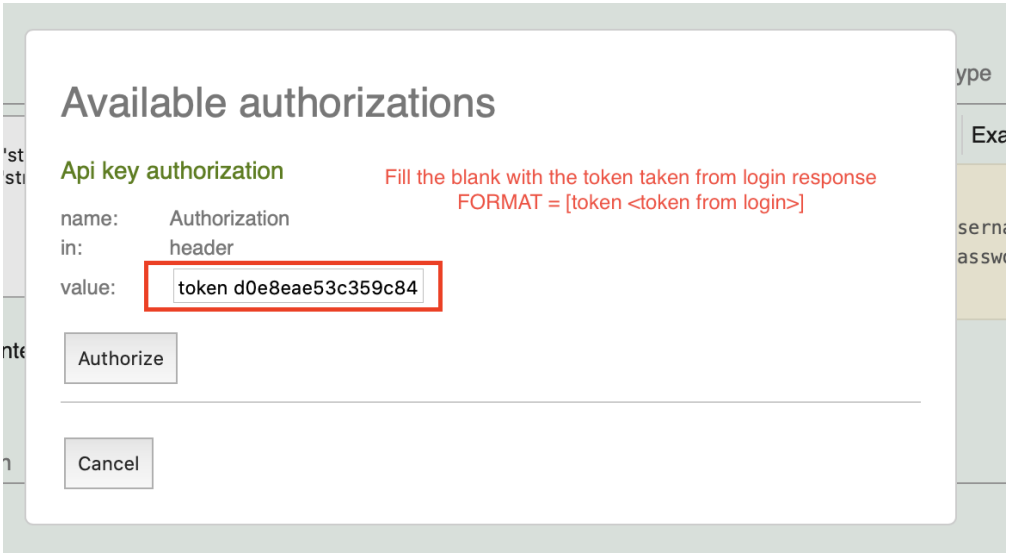
URL : SWAGGER

Most of the APIs need to have authorized users to operate. Here is how to authorize your api calls from the swagger. Go to the url highlighted above. Login as a user from the login endpoints. Copy the token value of the response body of the login API.



After copying the token value, click the authorize button on the up left corner of the page

Insert the token value to the blank described as value with prefix “token”.



After inserting the token with token prefix, click the Authorize button. And you will have full access to the user APIs.

API URLs:

Logout: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/logout/
Shared Post by a user: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/profile_post/
Create Post: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/create_post/
Edit Post: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/edit_post/
Delete Postt: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/delete_post/
Get Profile: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/get_profile/<str:user_id>/
Set Profile: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/set_profile/
Follow: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/follow/
Like Post: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/like/
Home Page: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/home_page/
Get Likes: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/get_likes/<str:story_id>/
Comment Create: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/comment_create/
Comment Update: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/comment_update/
Comment Delete: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/comment_delete/
Comments of a Post: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/get_comments/
Get user Likes: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/get_user_likes/
Register: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/guest/register/
Login: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/guest/login/
Change Password: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/guest/change_password/
Guest Home Page: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/guest/home_page/

API Examples:

Get User Profile Information:

Endpoint Url: http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/get_profile/<user_id:int>/

Curl Command:
```
curl -X GET --header 'Accept: application/json' --header 'X-CSRFToken: ah1wxdr83XjVqc1HIM4bOMoaXo66C6XBZQ3Ob89uONPls3S4diVsGbciGoNhg9Ek' --header 'Authorization: token a09413d52bef7ec44eed4b77a1417d2966017cba' 'http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/get_profile/2/'
```

Expected Response:

```
{
  "response": {
    "first_name": "Abdulkadir",
    "last_name": "Elmacı",
    "birthday": "2000-10-30",
    "location": {
      "location": "Mars",
      "latitude": 1,
      "longitude": 1,
      "type": "Real"
    },
    "photo_url": "",
    "username": "Kadir",
    "email": "abdulkadirelmaci@gmail.com",
    "followers": [
      {
        "user_id": 1,
        "user_id__username": "hamza"
      },
      {
        "user_id": 3,
        "user_id__username": "Erencan"
```

```
      },
      {
        "user_id": 8,
        "user_id__username": "gollum123"
      },
      {
        "user_id": 9,
        "user_id__username": "UmutKocasari"
      }
    ],
    "followings": [
      {
        "follow": 1,
        "follow__username": "hamza"
      },
      {
        "follow": 3,
        "follow__username": "Erencan"
      },
      {
        "follow": 4,
        "follow__username": "ringcarrier86"
      },
      {
        "follow": 5,
        "follow__username": "mervebrn"
      },
      {
        "follow": 6,
        "follow__username": "Gandalf"
      },
      {
        "follow": 7,
        "follow__username": "bounrabia"
      }
    ],
    "biography": "I'm a CMPE Studenta",
    "user_id": 2
  }
}
```


Get User Profile Information:

Endpoint Url: `http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/create_post/`

Curl Command: `curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'X-CSRFToken: ah1wxdr83XjVqc1HIM4bOMoaXo66C6XBZQ3Ob89uONPls3S4diVsGbciGoNhg9Ek' --header 'Authorization: token a09413d52bef7ec44eed4b77a1417d2966017cba' -d '{   "title": "string",   "text": "string", "multimedia": "string",   "username": "string",   "time_start": "2021-05-05",   "time_end": "2021-05-06",   "location": [{"location": "changed_asd", "latitude": 1901, "longitude": 120, "type": "Real"}   ],   "tags": ["string"   ] }' 'http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/create_post/'`

Expected Response: `{`


```
  "return": 14
}, different every time since it return stroy id created
```


Like A Post:

Endpoint Url: `http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/like/`

Curl Command: `curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'X-CSRFToken: ah1wxdr83XjVqc1HIM4bOMoaXo66C6XBZQ3Ob89uONPls3S4diVsGbciGoNhg9Ek' --header 'Authorization: token a09413d52bef7ec44eed4b77a1417d2966017cba' -d '{"story_id": 1,"user_id": 1}' 'http://ec2-35-158-103-6.eu-central-1.compute.amazonaws.com:8000/user/like/'`

Expected Response: `{`


```
  "response": {
    "user_id": 1,
    "story_id": 1,
    "isLiked": false
  }
}, if it is like it unlikes it and vice versa
```

Further detail about APIs can be found under swagger extension specified in the URL above

5. Implementation of Core Functionality

Mapping locations is a core functionality for our project. We have implemented two types of locations in our application. Virtual locations represent the applications in the virtual environment. A user can post stories with locations like zoom, teams, instagram, facebook or other frequently used social media applications. On the other hand real locations have geolocations.

We used Google Maps API and implemented a map object and a dialog screen which takes the latitude and longitude information from map object and send to the post sharing function. In this way, all posts with real locations has geolocation information too. We will use this information in searching and exploring mechanism.

We will implement a searching mechanism with using map object. A user will select an area from map and click a search button. We will retrieve all stories in this area with marker clustering feature. The user can click this marker to view the location names and stories under that cluster. Also the user will be able to filter out the posts using date filters to view the stories about the locations from the map.

We will implement a searching mechanism to search the posts and users by text. This text will search for story topics and story texts. Also by selecting a tag(s) the user can search the posts with that tag.

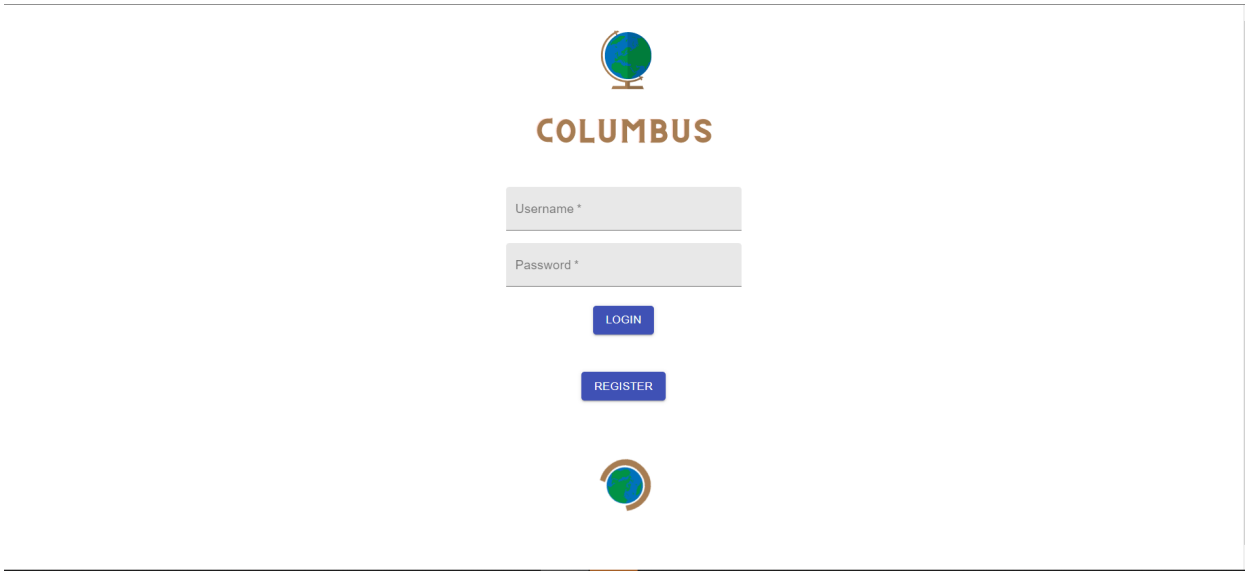
6. User Experience\User Interface

1. Web Interfaces

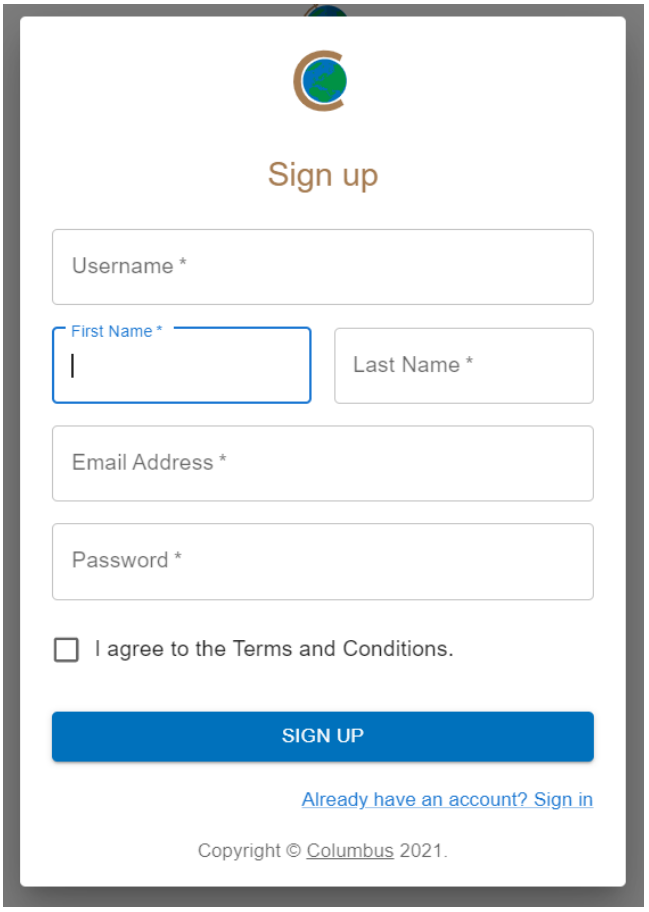
- Login Page

  Link :
  https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/pages/Login

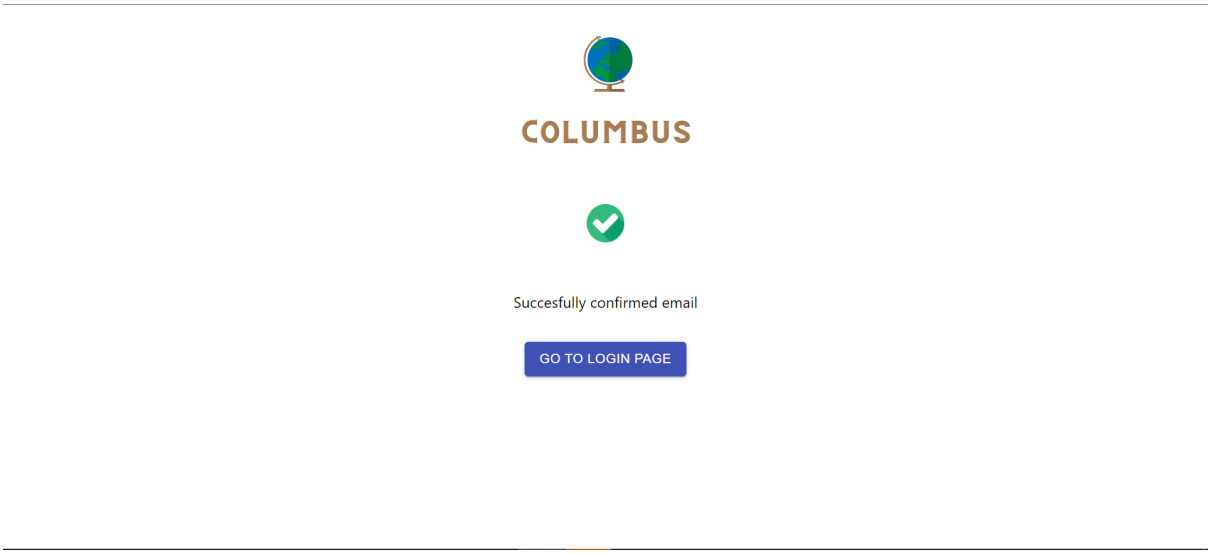
  Screenshot :



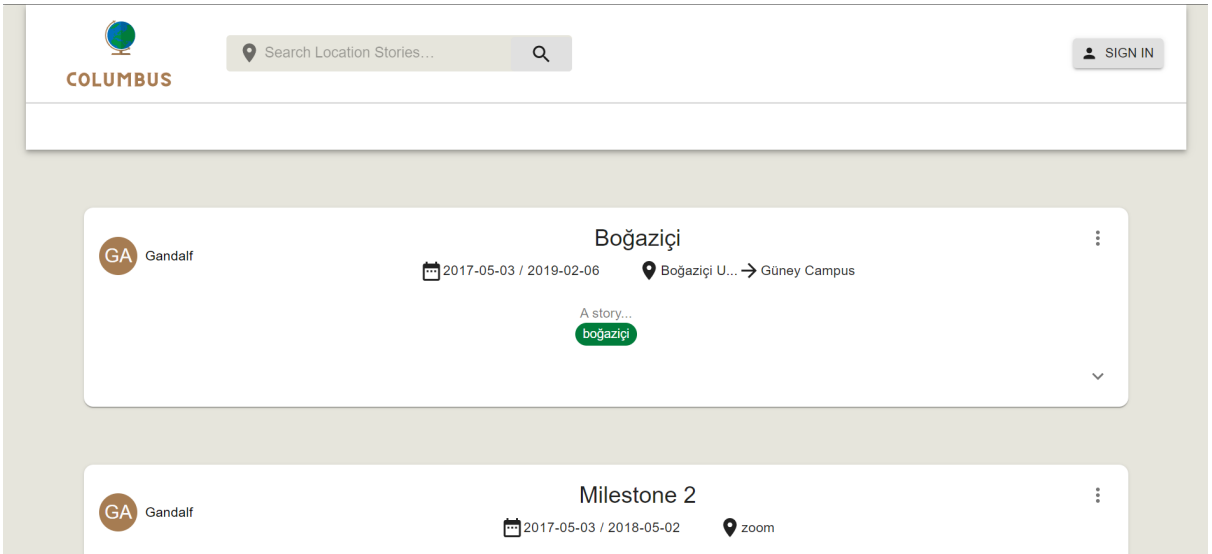
- Register Dialog
  Link :
  https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/pages/Register
  Screenshot :



- Email Confirmation Page
  Link :
  https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/pages/EmailConfirmationPage
  Screenshot :

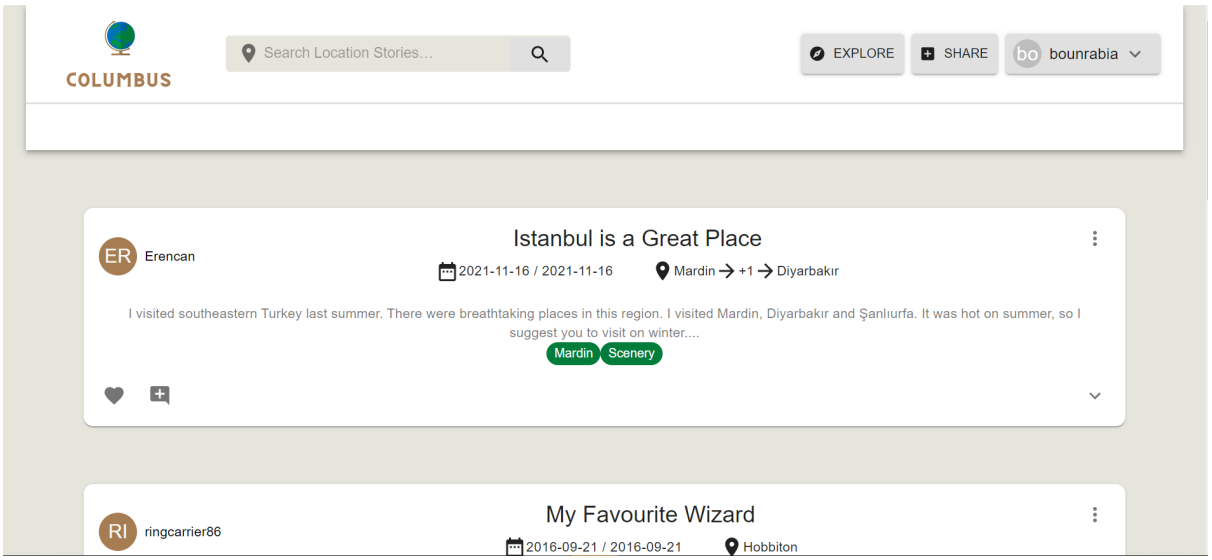


- Guest Home Page
  Link :
  https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/pages/Home
  Screenshot :



- User Home Page
- **Link :**
  https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/pages/Home

Screenshot :

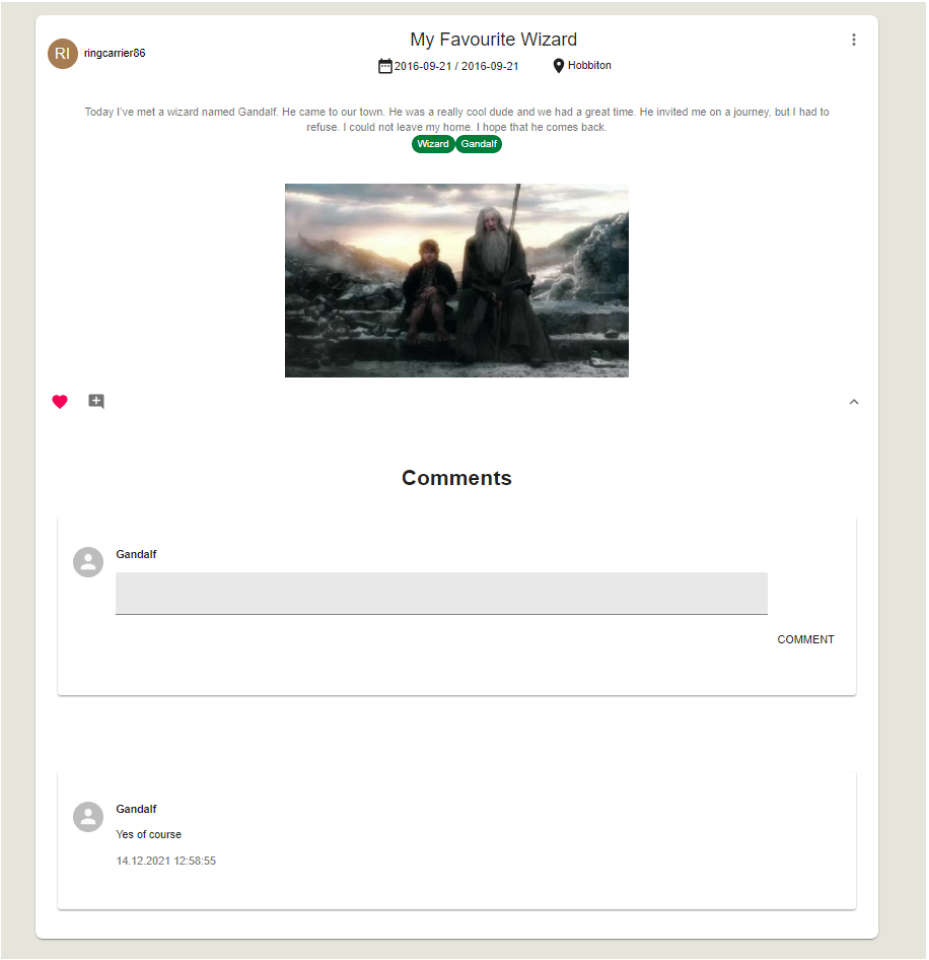


- Post component with comment and like feature

  Link :

  https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/components/Post

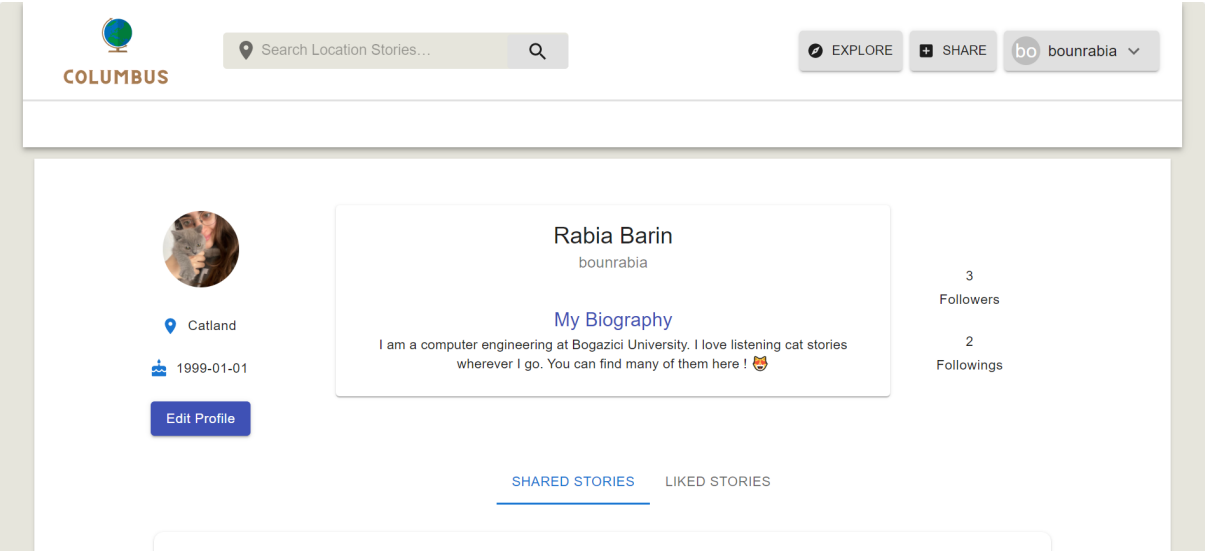
  Screenshot :



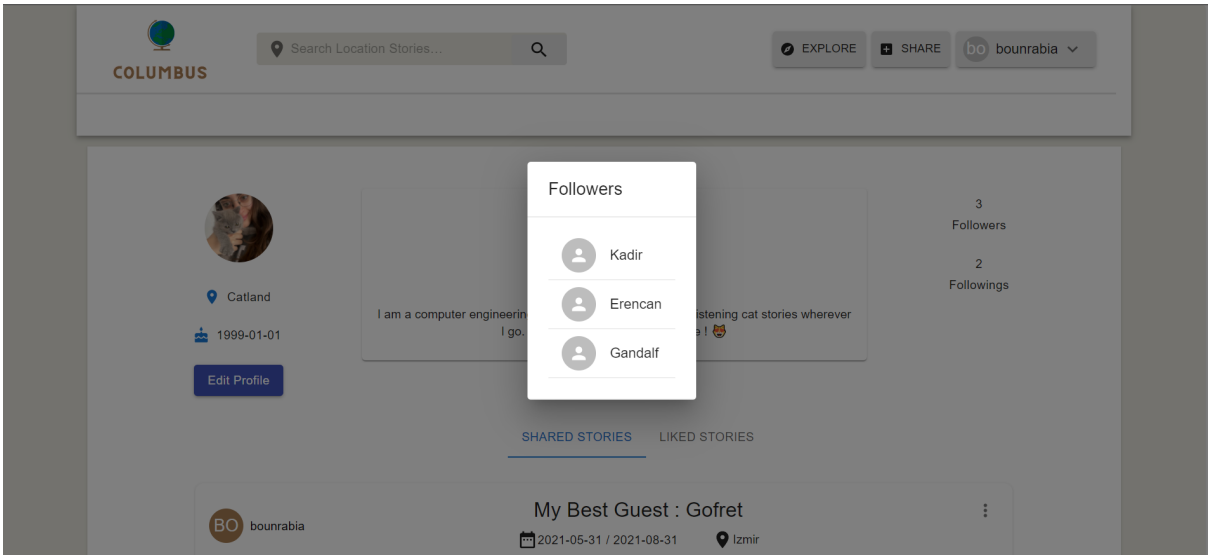
- User's own Profile Page

  Link :
  https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/pages/Profile

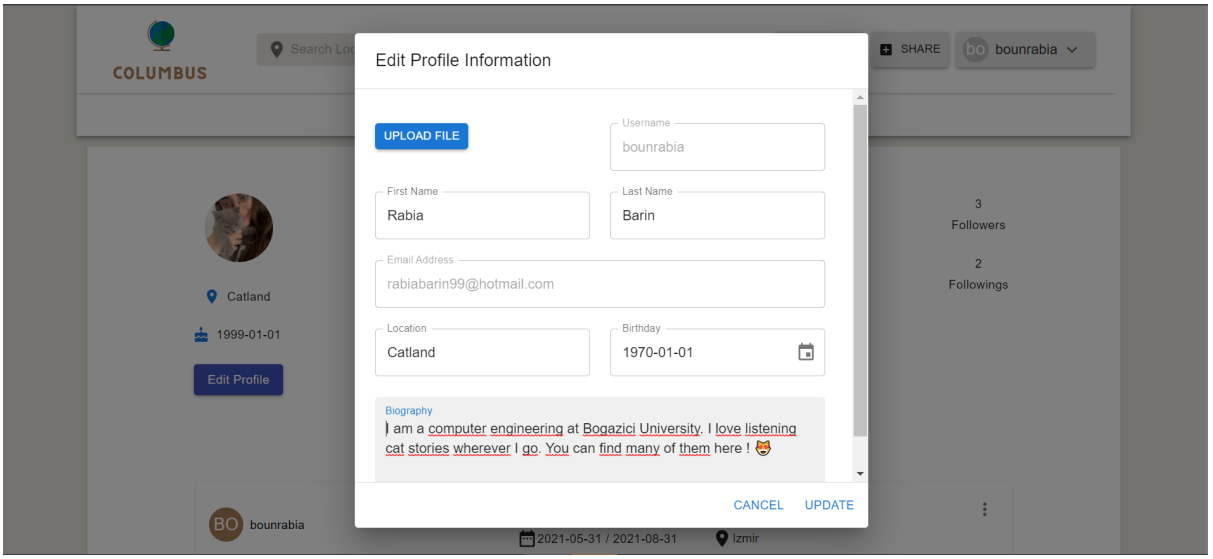
  Screenshot :



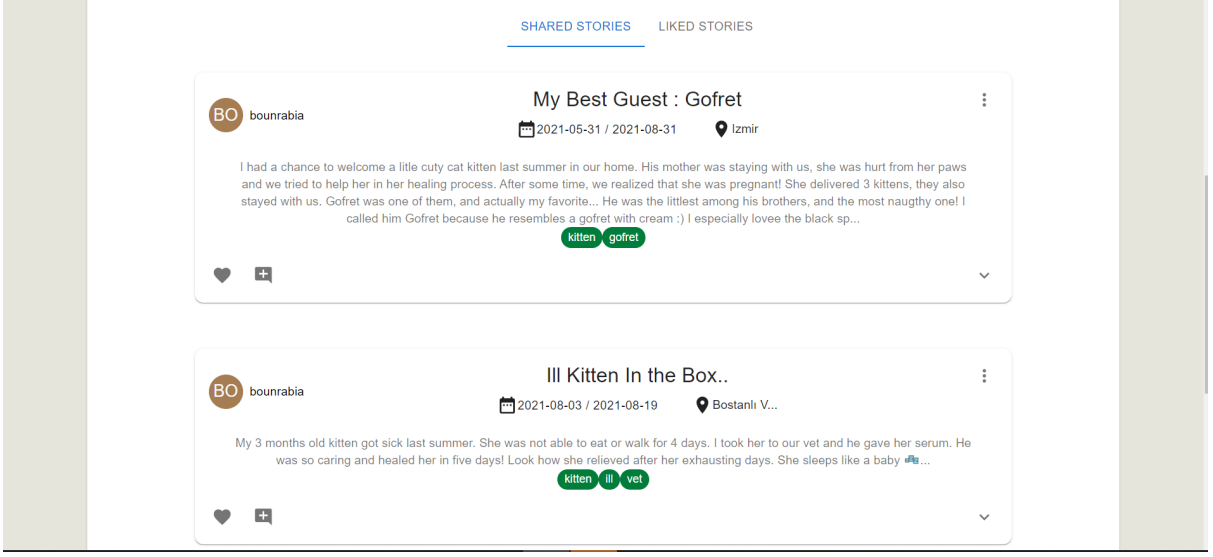
Follower/Following Dialog Screenshot:



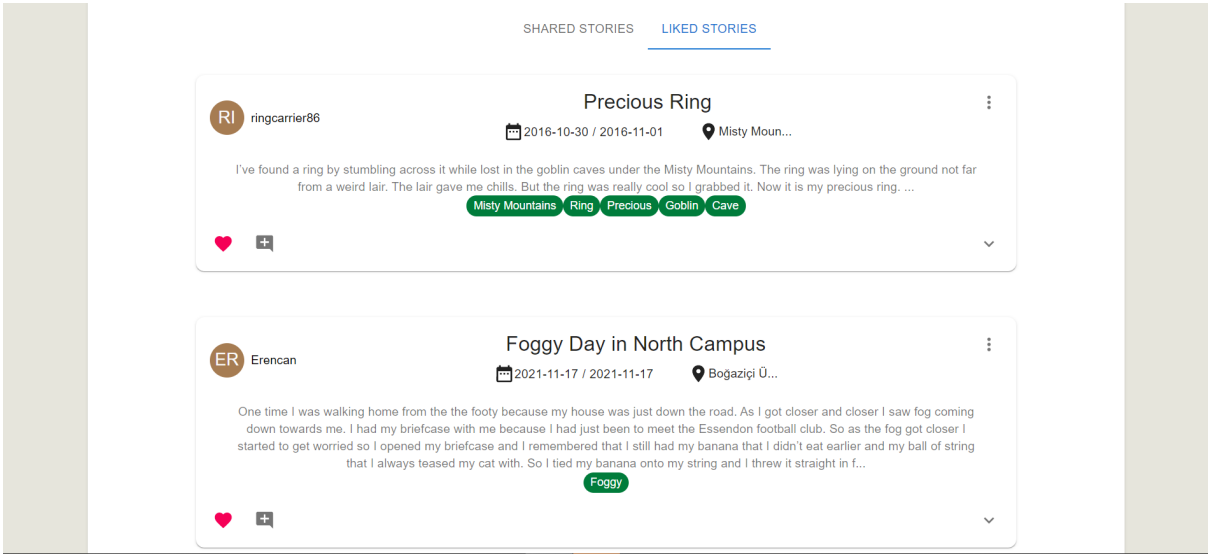
Edit Profile Dialog Screenshot:



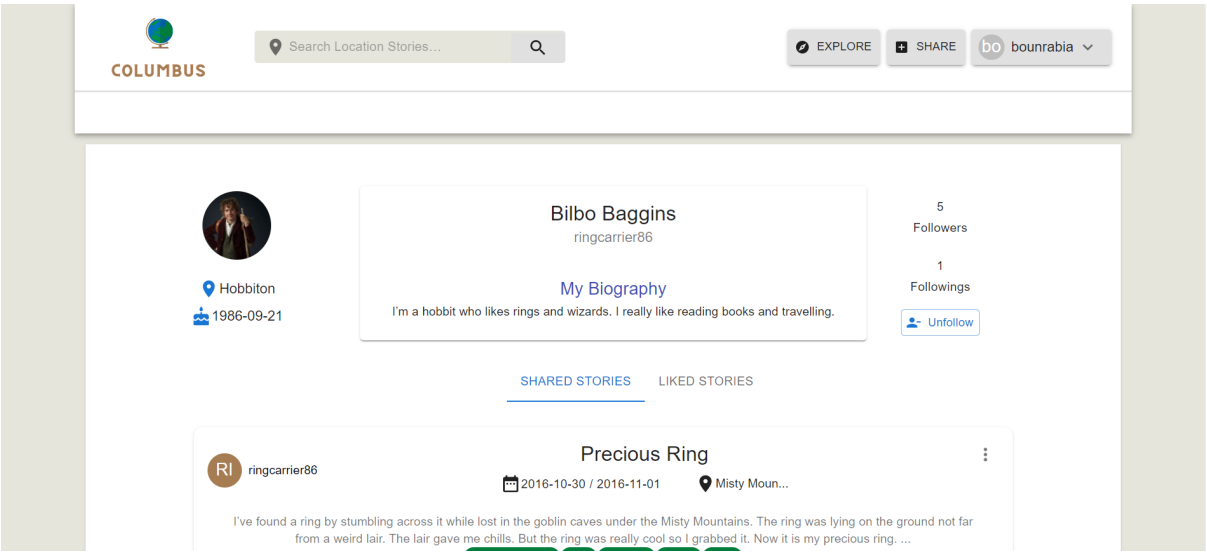
Shared Stories:



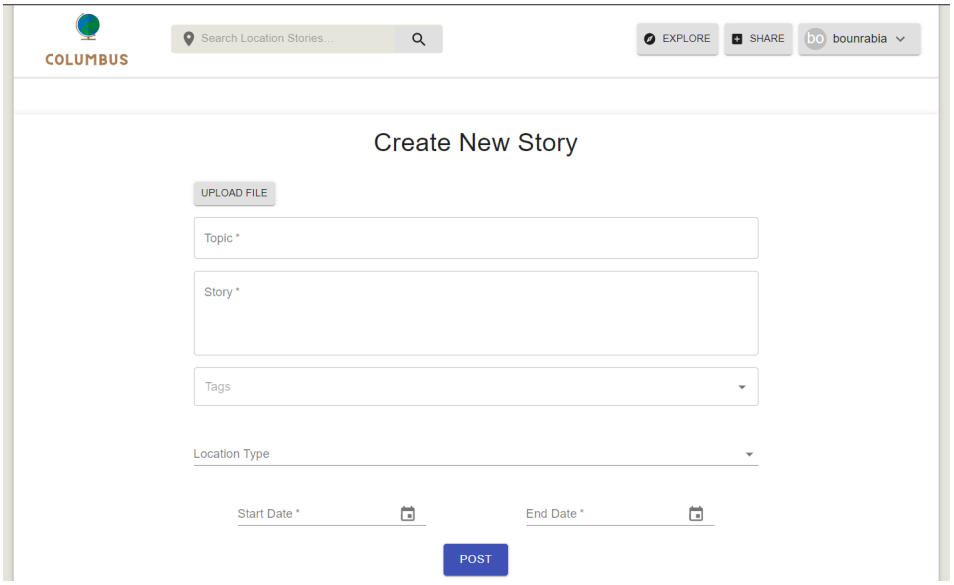
Liked Stories:



- User viewing other Profiles
  Link :
  https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/pages/Profile
  Screenshot :



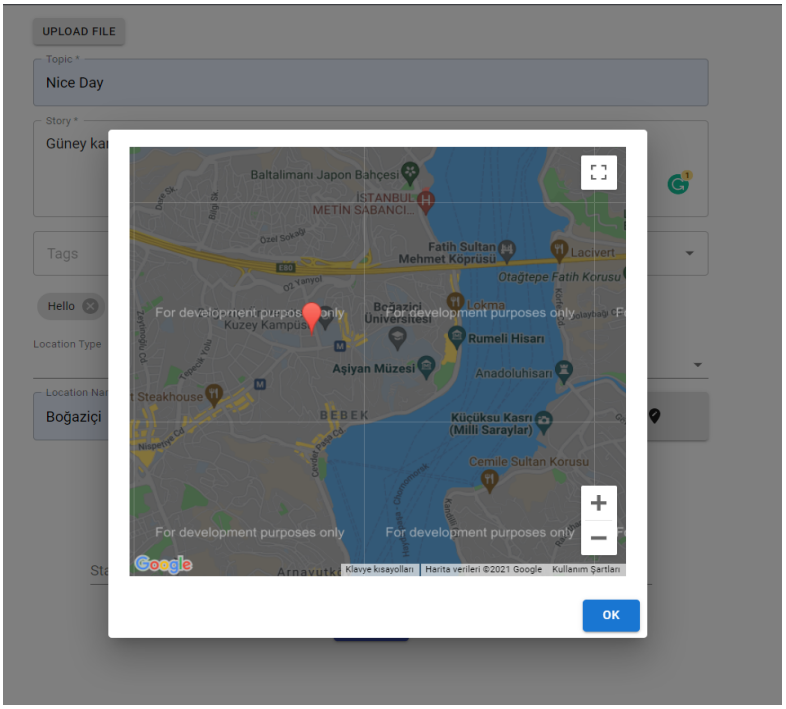
- Create Story Page
  Link :
  https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/pages/CreatePostPage
  Screenshot :



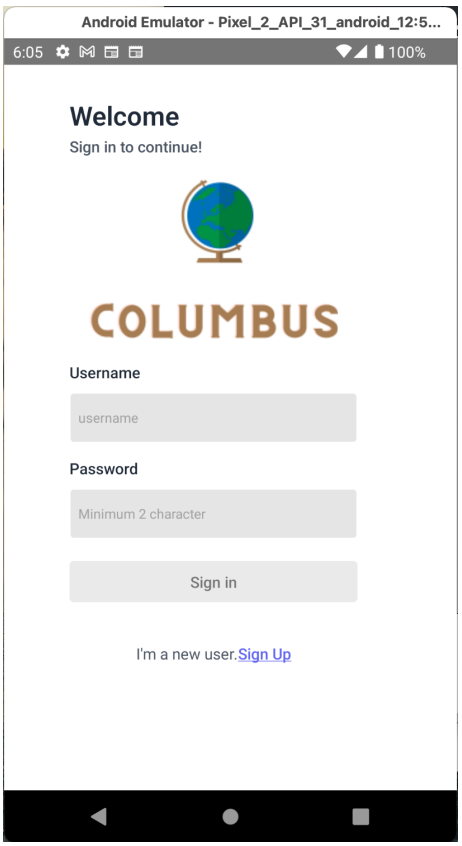
- Map dialog in create story

Link :
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/components/GoogleMaps
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/frontend/src/components/Dialogs/MapsDialog

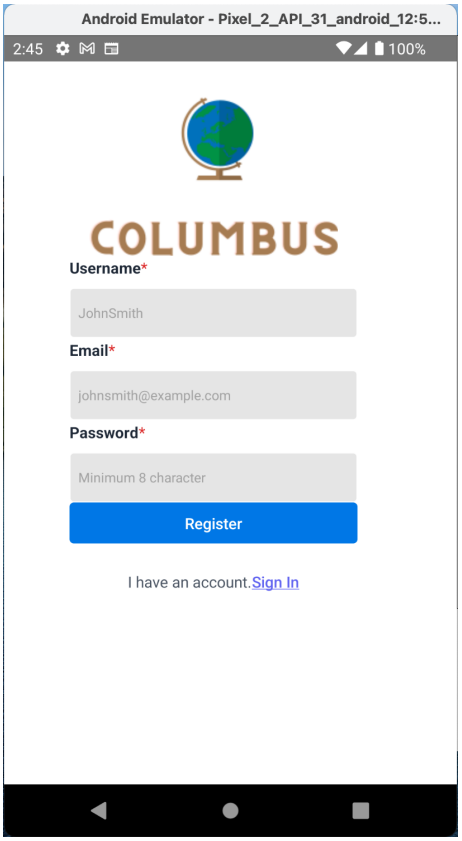
Screenshot :



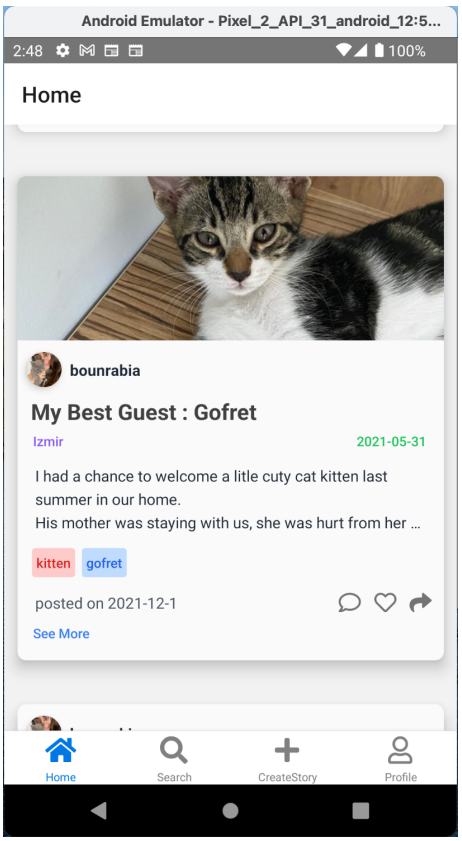
2. Mobile Application Interfaces

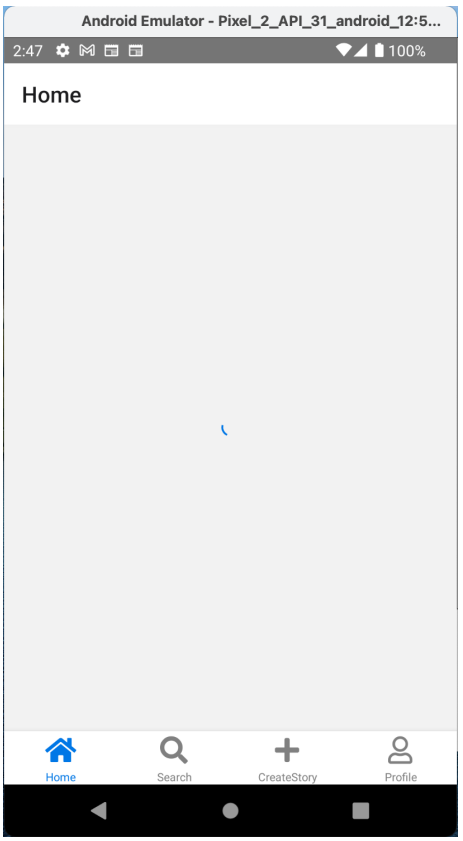


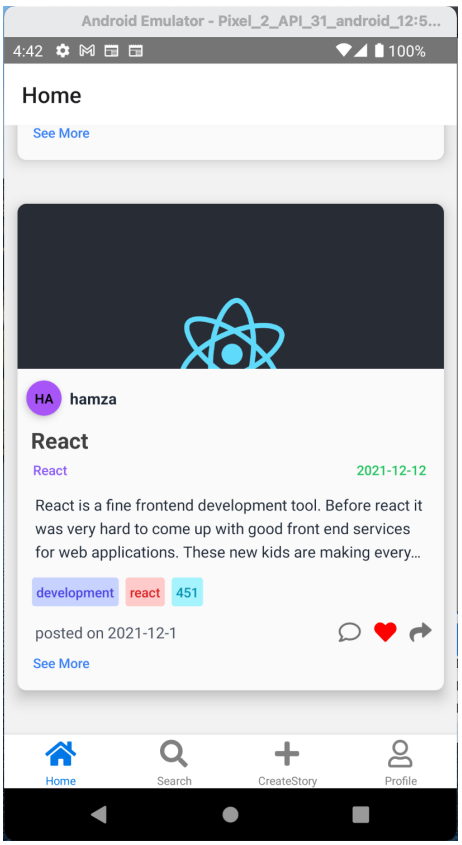
Login Page:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/views/Login



Register Page:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/views/Register



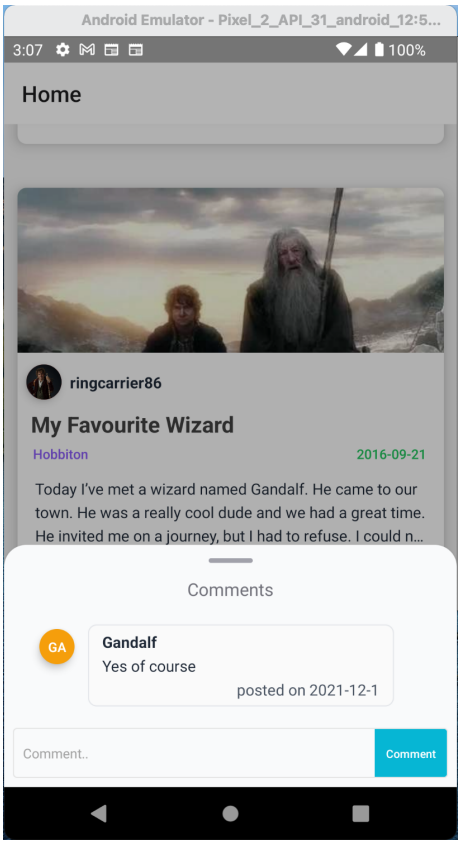




Home Page:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/views/Home

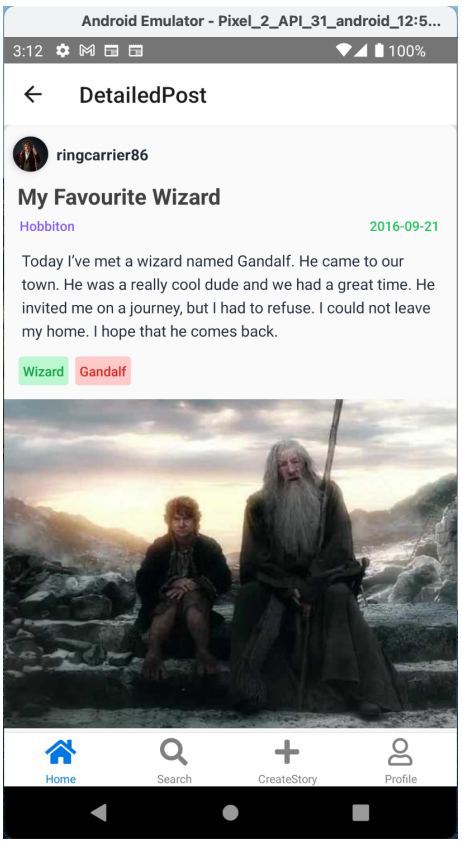
Postcard:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/components/PostCard

Spinner:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/components/PageSpinner

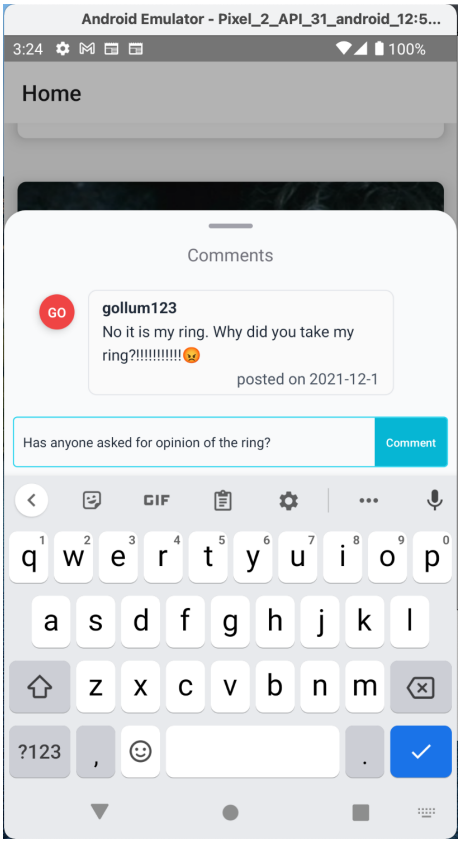


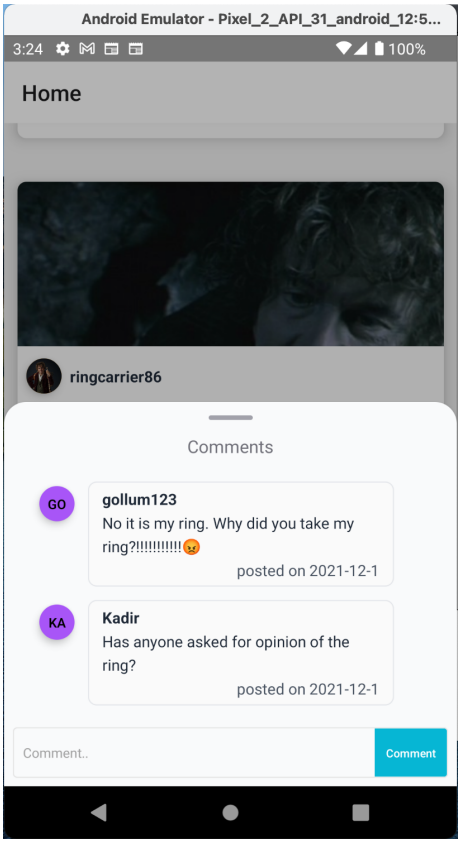
Comment:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/components/Comment

Comment Action Sheet:
https://github.com/bounswe/2021SpringGroup7/blob/master/Columbus/mobile/Columbus/src/components/PostCard/SubComponents/CommentSheet.js


Android Emulator - Pixel_2_API_31_android_12:5...
3:12
100%
DetailedPost
ringcarrier86
My Favourite Wizard
Hobbiton
2016-09-21
Today I've met a wizard named Gandalf. He came to our town. He was a really cool dude and we had a great time. He invited me on a journey, but I had to refuse. I could not leave my home. I hope that he comes back.
Wizard
Gandalf
Home
Search
CreateStory
Profile

Android Emulator - Pixel_2_API_31_android_12:5...
3:13
100%
DetailedPost
My Favourite Wizard
Hobbiton
2016-09-21
Today I've met a wizard named Gandalf. He came to our town. He was a really cool dude and we had a great time. He invited me on a journey, but I had to refuse. I could not leave my home. I hope that he comes back.
Wizard
Gandalf
posted on 2021-12-1
Home
Search
CreateStory
Profile


Android Emulator - Pixel_2_API_31_android_12:5...
3:24
100%
Home
Comments
GO
gollum123
No it is my ring. Why did you take my ring?!!!!!!!!!!
posted on 2021-12-1
Has anyone asked for opinion of the ring?
Comment

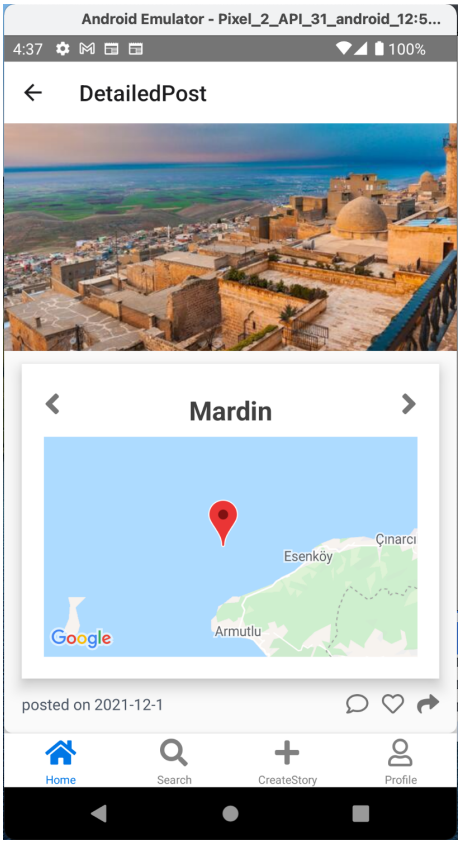

Android Emulator - Pixel_2_API_31_android_12:5...
3:24
100%
Home
ringcarrier86
Comments
GO
gollum123
No it is my ring. Why did you take my ring?!!!!!!!!!!
posted on 2021-12-1
KA
Kadir
Has anyone asked for opinion of the ring?
posted on 2021-12-1
Comment..
Comment

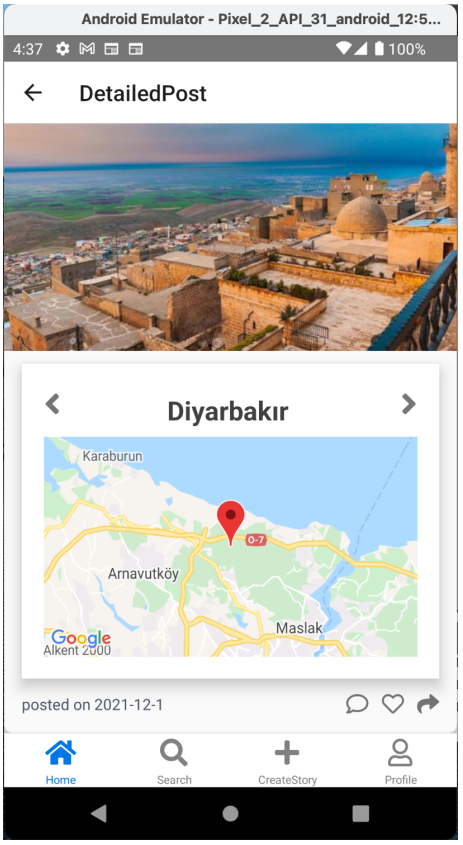
Page:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/views/DetailedPost

Components:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/components/PostCard/SubComponents

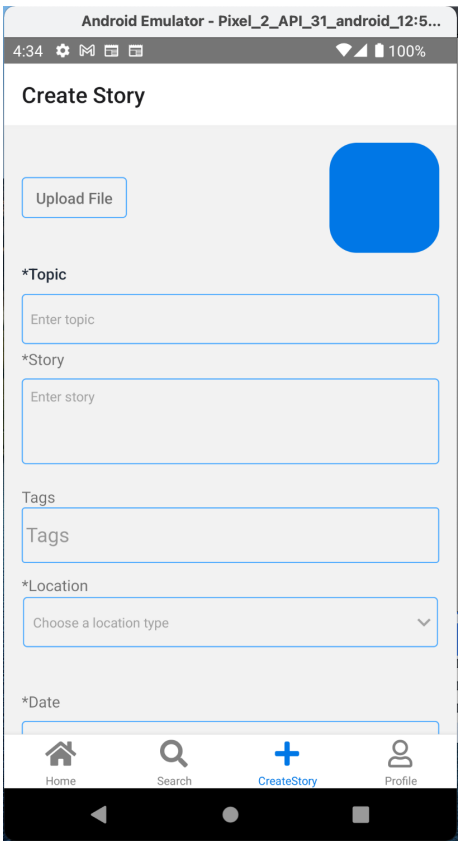
Comment:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/components/Comment

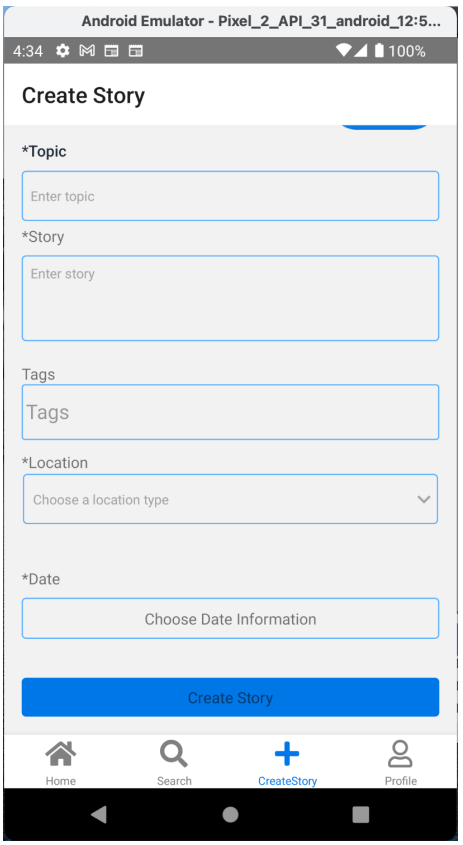
Comment Action Sheet:
https://github.com/bounswe/2021SpringGroup7/blob/master/Columbus/mobile/Columbus/src/components/PostCard/SubComponents/CommentSheet.js

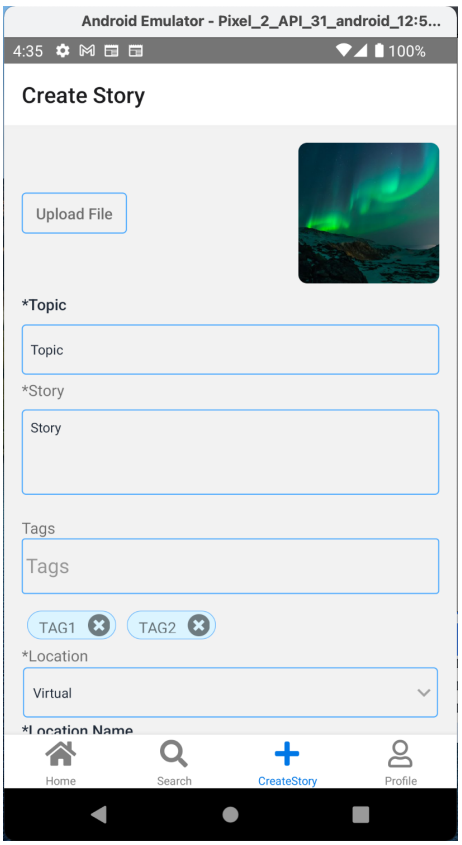


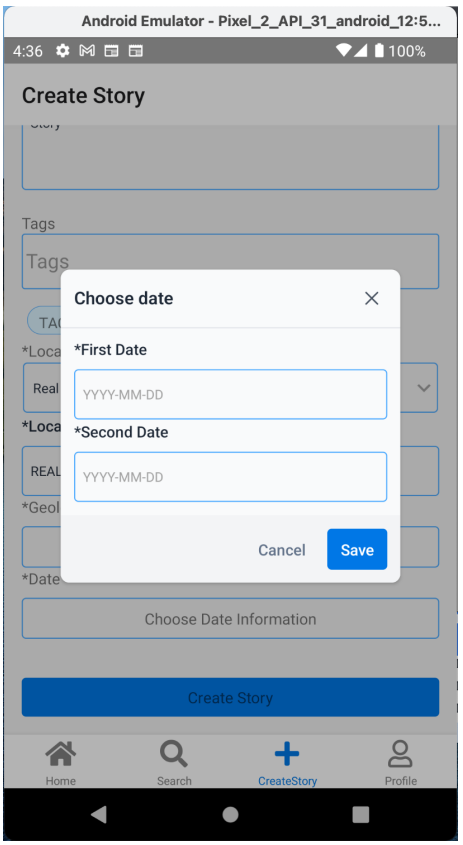


Map View:
https://github.com/bounswe/2021SpringGroup7/blob/master/Columbus/mobile/Columbus/src/views/DetailedPost/DetailedPost.js


Android Emulator - Pixel_2_API_31_android_12:5...
4:34
100%
Create Story
Upload File
*Topic
Enter topic
*Story
Enter story
Tags
Tags
*Location
Choose a location type
*Date
Home
Search
CreateStory
Profile

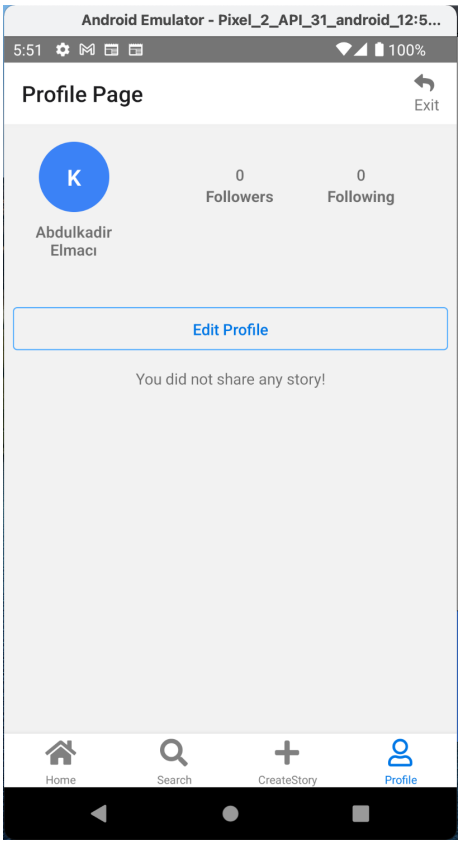

Android Emulator - Pixel_2_API_31_android_12:5...
4:34
100%
Create Story
*Topic
Enter topic
*Story
Enter story
Tags
Tags
*Location
Choose a location type
*Date
Choose Date Information
Create Story
Home
Search
CreateStory
Profile

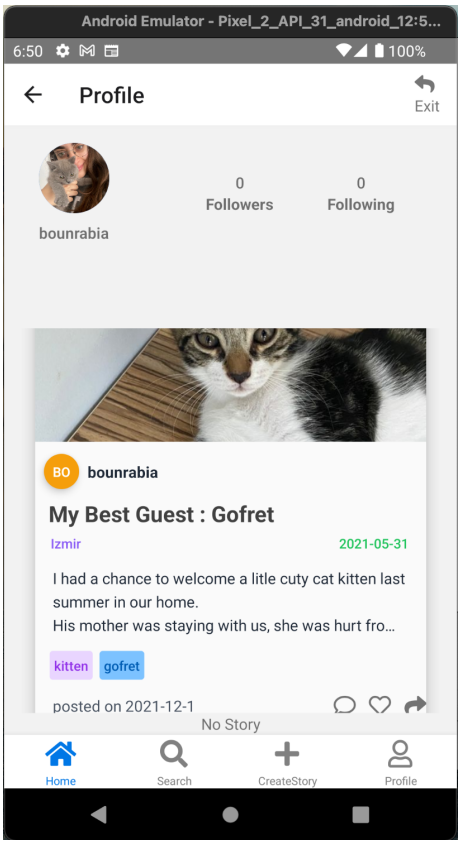

Android Emulator - Pixel_2_API_31_android_12:5...
4:35
100%
Create Story
Upload File
*Topic
Topic
*Story
Story
Tags
Tags
TAG1
TAG2
*Location
Virtual
*Location Name
Home
Search
CreateStory
Profile


Android Emulator - Pixel_2_API_31_android_12:5...
4:36
100%
Create Story
Tags
Tags
Choose date
*First Date
YYYY-MM-DD
*Second Date
YYYY-MM-DD
Cancel
Save
Choose Date Information
Create Story
Home
Search
CreateStory
Profile

Create Story Page:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/views/CreatePost

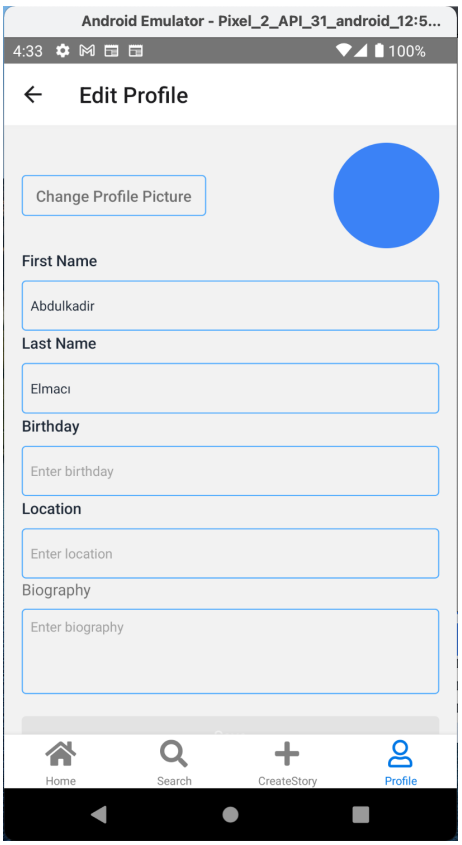
Dialog:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/components/CustomModal

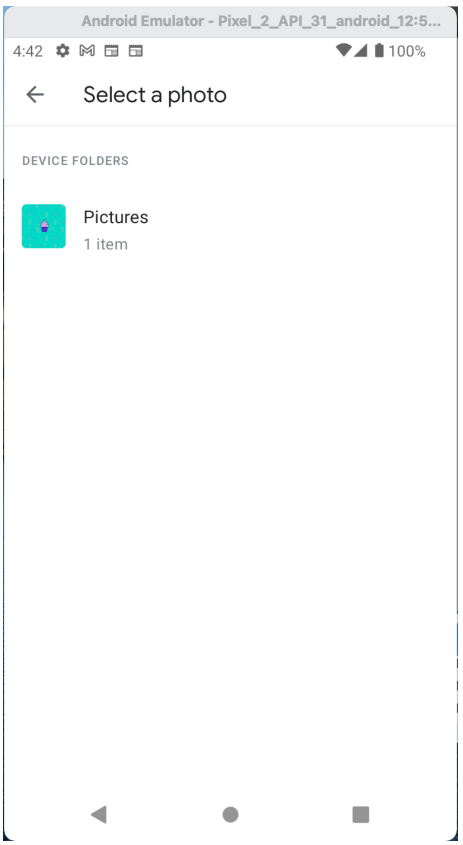




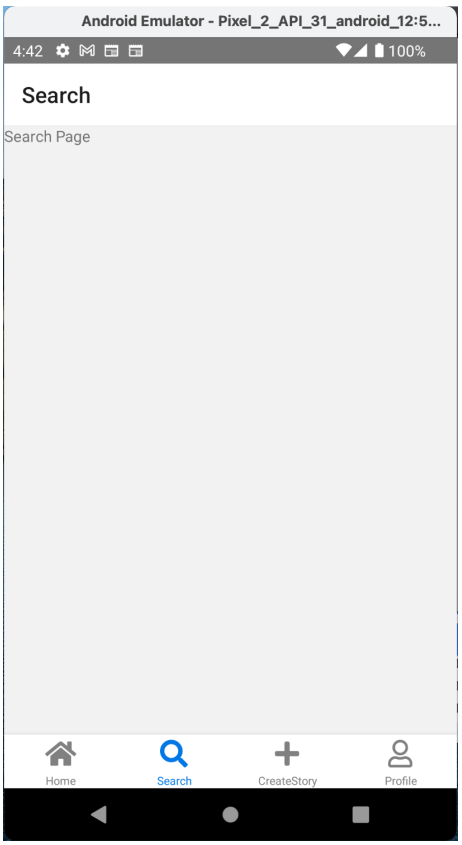
Profile Page:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/views/Profile

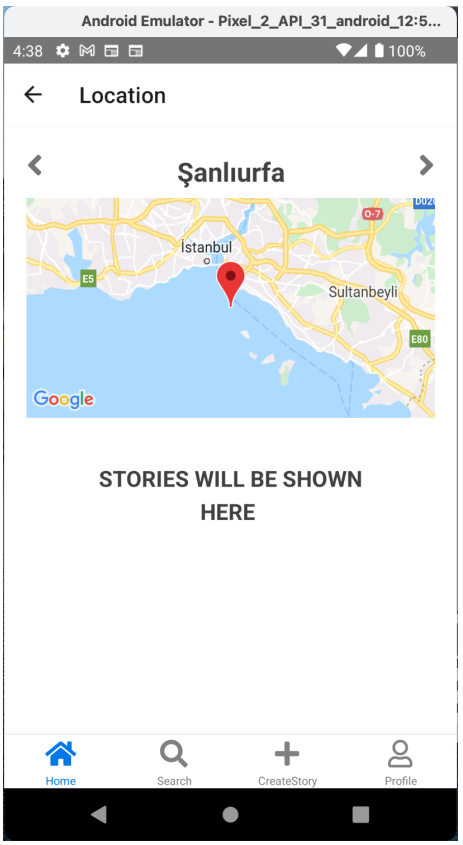
Components:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/views/Profile/components





Edit Profile Page:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/views/Profile/views/EditProfile





Location Page:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/views/Location

Search Page:
https://github.com/bounswe/2021SpringGroup7/tree/master/Columbus/mobile/Columbus/src/views/Search

7. Standards

Endpoint: /activitystream
Method: POST
Authorization: No Need.

Activity Stream is keeping track of activities done on an application/site. Many contributions can be made to the user, customer and company by keeping/displaying or examining this record.

Currently, Activity Streams in Columbus include user actions that cause **changes to the database** and **logout**.

Currently, the Activity Stream infrastructure is ready and operational. The API takes the **limit** you specify how many activities you want to see and/or the **offset** parameters that take the id of the **last ActivityStream** you want to see. In this way, you can see as many

activities as you want and you can get the activities **up to the last activity** you saw. If desired, we can expand the scope of the activity or easily filter the activities by users and dates.

Typical Response:

```
{
            "@context": Context of the activity stream,
            "summary": General summary of the response object,
            "type": Type of the returned object based on scheme,
            "total_items": Total number of activities returned,
            "orderedItems":   List of activities :
                         [ {
                        "@context": Context of the activity stream,
                        "summary": General summary of the activity,
                        "id": Activity Stream Model Id,
                        "type": Type of the activity
                        "actor": People doing the activity,
                        "object": Object of the activity,
                        "target": Thing affected from activity
                        }]
}
```

Activities:

_setprofile: {activity.actor.username} set profile informations.

_updatepost: {activity.actor.username} updated {activity.story.id}

_createpost: {activity.actor.username} created post

_logout: {activity.actor.username} logged out

_unlike: {activity.actor.username} uliked {activity.story.id}

_like: {activity.actor.username} liked {activity.story.id}

_unfollow: {activity.actor.username} unfollowed {activity.target.username}

_follow: {activity.actor.username} followed {activity.target.username}

_comment_update: {activity.actor.username} updated comment : {activity.comment.id}

_comment_create: {activity.actor.username} added comment to story : {activity.story.id}

Successful Response


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "summary": "Activity stream",
  "type": "OrderedCollection",
  "total_items": 10,
  "orderedItems": [
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "summary": "string set profile informations.",
      "id": 13,
      "type": "SetProfile",
      "actor": {
        "type": "https://schema.org/Person",
        "@id": "string"
      }
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",+
      "summary": "string set profile informations.",
      "id": 12,
      "type": "SetProfile",
      "actor": {
        "type": "https://schema.org/Person",
        "@id": "string"
      }
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "summary": "string updated 2 ",
      "id": 11,
      "type": "UpdatePost",
      "actor": {
        "type": "https://schema.org/Person",
        "@id": "string"
      },
      "object": {
        "type": "https://schema.org/ShortStory",
        "@id": 2
      }
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "summary": "string created post",
      "id": 9,
      "type": "CreatePost",
      "actor": {
        "type": "https://schema.org/Person",
        "@id": "string"
      },
      "object": {
        "type": "https://schema.org/ShortStory",
        "@id": 5
      }
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "summary": "string updated comment : 2  ",
      "id": 7,
      "type": "CommentUpdate",
      "actor": {
        "type": "https://schema.org/Person",
        "@id": "string"
      },
      "object": {
        "type": "https://schema.org/Comment",
        "@id": 2
      }
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "summary": "string added comment to story : 1 ",
      "id": 6,
      "type": "CommentCreate",
      "actor": {
        "type": "https://schema.org/Person",
        "@id": "string"
```

```
      },
      "object": {
        "type": "https://schema.org/ShortStory",
        "@id": 1
      }
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "summary": "string added comment to story : 1 ",
      "id": 4,
      "type": "CommentCreate",
      "actor": {
        "type": "https://schema.org/Person",
        "@id": "string"
      },
      "object": {
        "type": "https://schema.org/ShortStory",
        "@id": 1
      }
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "summary": "string created post",
      "id": 3,
      "type": "CreatePost",
      "actor": {
        "type": "https://schema.org/Person",
        "@id": "string"
      },
      "object": {
        "type": "https://schema.org/ShortStory",
        "@id": 3
      }
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "summary": "string created post",
      "id": 2,
      "type": "CreatePost",
      "actor": {
        "type": "https://schema.org/Person",
        "@id": "string"
      },
      "object": {
        "type": "https://schema.org/ShortStory",
        "@id": 2
      }
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "summary": "string created post",
      "id": 1,
      "type": "CreatePost",
      "actor": {
        "type": "https://schema.org/Person",
        "@id": "string"
      },
      "object": {
        "type": "https://schema.org/ShortStory",
        "@id": 1
      }
    }
  ]
}
```