Final Group Report

CMPE451- Fall'21

Prepared by:

- Bekir Tatar
- Emin Deniz
- Eren Altunoğlu
- Fikri Cem Yılmaz
- Hazar Çakır
- Kürşat Bakış
- Meltem Arslan
- Mehmet Yasin Şeremet
- Selman Berk Özkurt

Group number: 11

Group name: Renaissance

Sub-groups:

Backend:

- Eren Altunoğlu
- Mehmet Yasin Şeremet
- Selman Berk Özkurt

Frontend:

- Bekir Tatar
- Emin Deniz
- Meltem Arslan

Android:

- Fikri Cem Yılmaz
- Hazar Çakır
- Kürşat Bakış

Tag-id: v0.9.2

Date: 12.01.2021

1. Executive Summary

1.1 Status of Deliverables

- Backend and Web are deployed.
- Mobile apk is prepared.
- Final report is prepared.

1.2 Status of all Requirements

1. Functional Requirements

1.1. Accounts

- **1.1.1.** User *shall* be able to register by providing a valid and verified e-mail address, username, and a strong password. -> Completed.
- **1.1.2.** User *shall* be able to login by providing verified e-mail address. -> Completed.
- **1.1.3.** The user *shall* be able to delete their accounts. -> Not completed.
- **1.1.4.** The user *shall* be able to reset its password via providing its e-mail address and via passing a challenge-response test such as "captcha" . -> Not completed.
    - **1.1.4.1.** In this case, System *should* send a mail for resetting password. In order for resetting to happen, user should initiate resetting process from the e-mail that is specifically sent from the system. -> Not completed.
- **1.1.5.** System *shall* include admin privileged user whose privileges are as follows: -> Not completed.
    - **1.1.5.1** An admin privileged user *should* be ban or permanently delete users -> Not completed.
    - **1.1.5.2** An admin privileged user *should* be remove comments, materials and resources. -> Not completed.
    - **1.1.5.3** An admin privileged user *should* be able to give admin privileges to a normal user while declaring it as another admin. -> Not completed.
- **1.1.6.** Every user *shall* have a unique and non-alterable username called user identifier which will *shall* be determined at the creation of the account. -> Completed.
    - **1.1.6.1.** System *shall* not allow new accounts to use an user identifier associated with a deleted account, or that is not associated with any account anymore for any reason. -> Completed.

1.2. Self-Development

- **1.2.1.** User *shall* create a resource which includes a description, image and materials. The details of resource is explained in the glossary. -> Completed.
- **1.2.2.** The System *shall* reliably record any progress data supplied by the users grouped by materials and topics. -> Not completed.
  - **1.2.2.1.** The System *should* display at least one central statistic of this information grouped by materials and topics. -> Not completed.
  - **1.2.2.2.** The System *should* display progress data associated by a user to the users associated with that user. -> Not completed.
- **1.2.3.** The System *should* provide the user with a graphical overview of their past effort. There *should* be a page to show this information to the user. -> Not completed.
- **1.2.4.** The System *shall* determine and inform its users about topics they could be potentially interested. -> Completed.
  - **1.2.4.1.** The System Interface *shall* provide a page that the users can intentionally navigate to and see this information that may also contain other functionality. -> Completed.
- **1.2.5.** The System *shall* supply its users with recommendation of resources. -> Completed.
  - **1.2.5.1.** The System Interface *shall* provide a page that the users can intentionally navigate to and see this information that may also contain other functionality. -> Completed.
- **1.2.6.** The System *should* offer a means of reliably proving realized effort and any accomplishment of a user to any third party authorized by that user. -> Not completed.
- **1.2.7.** The users *shall* be able to reliably record and edit some objectives they determine as a set of material-deadline pairs. -> Not completed.
- **1.2.8.** The users *should* be able to record topics they want to improve on, or break/develop a habit. -> Completed.
- **1.2.9.** The users *shall* be able to work on their determined objectives as a group. -> Not completed.
  - **1.2.9.2.** Users *should* be able to create groups under collective resources that can be only visible to users that are invited to this group. -> Not completed.

1.3. Social Interaction

- **1.3.1.** The users *should* have profile pages, and be able to modify their profiles. -> Completed.
- **1.3.2.** The users *should* be able to share their accomplishments and preferred topics via other applications. -> Not completed.

- **1.3.3.** The users *should* be able to share their comments on specific topics or resources. -> Completed. (“Comments” part was changed as “Resources” part)
    - **1.3.3.1.** Any comment *shall* display user identifier of the user that posted that comment -> Not completed.
        - **1.3.3.1.1.** If a user account is deleted, user identifier associated with a comment *shall* become "deleted user". -> Not completed.
    - **1.3.3.2.** A comment *should* be able to have a comment list associated with it, which *should* have nested aligning with respect to the owning comment. -> Not completed.
    - **1.3.3.3.** Any user *shall* have the option to delete any comment posted by that user. -> Not completed.
        - **1.3.3.3.1.** If a comment is deleted by the posting user and there exists a nonempty comment list associated with it, then the Interface *shall* display the deleted parent comment as the message "deleted comment". -> Not completed.
    - **1.3.3.4.** A comment *should* have a ballot associated with it which shows rate of the comment. Detailed ballot description is done at the glossary. -> Not completed.
- **1.3.4.** The users *should* be able to search for other users via topics that is associated with them or via username. -> Completed.
- **1.3.5.** Users *shall* be able to mark any resource, topic or milestone as "in progress", "following" or "done". -> Completed.
    - **1.3.5.1.** The system interface *shall* provide a means of marking and unmarking. -> Completed.
    - **1.3.5.2.** The system interface *shall* display number of users marking each object. -> Not completed.
    - **1.3.5.3.** System *shall* denote an association with the user and marked object as any of the above -> Completed.
- **1.3.7.** System *should* include two ballots denoting "effort" and "rating" associated with every resource. -> Completed.

1.4. User Experience

- **1.4.1.** The System interface *should* give its users notification messages. -> Completed. (Activity Stream)
    - **1.4.1.1.** If implemented, those notifications *shall* be given for scheduled resource milestones reminding the deadlines. -> Not completed.
- **1.4.2.** The users *should* be able to customize the way the notifications are sent. -> Completed.

- **1.4.3.** The users *should* be able to search data assigned to topics using strings. -> Completed.
- **1.4.4.** The users *should* be able to access other users' profiles. -> Completed.
    - **1.4.4.1.** The System Interface *should* display user's associations with topics, resources and milestones in the profile. -> Completed.

2. Non-Functional Requirements

2.1. Availability

- **2.1.1.** An Android Application and a Web Application *shall* be implemented as instances of the Interface. -> Completed.
    - **2.1.1.1.** Android operating systems of version 10 and above *shall* be supported. -> Completed.
- **2.1.2.** The Interface *shall not* require any user to give text commands for any of its functionalities. It *shall* be possible to give all commands through the interface using a graphical interface. -> Completed.
    - *2.2.2.1.* Those commands *should* be either the standard input types provided by HTML 5 standard or those with equivalent functionality. -> Completed.
- **2.1.3.** The System *shall* express topics using semantic representation as defined in Wikidata. -> Completed.
    - **2.1.3.1.** Topics *shall* be indexed using both the integer semantic ID as indexed in Wikidata in the Infrastructure. -> Completed.
    - **2.1.3.2.** The Interface *should* represent the topics as strings in the preferred language of the operating system of the user and a representation in whatever language is available if it is not available. -> Not completed.
        - **2.1.3.2.1.** If the representation of the user's language is not found and there are multiple languages for which a representation is available, English *shall* be prioritized. -> Not completed.
- **2.1.4.** The Web Application *should* only use standard HTML5 and Javacscript code that is compatible with Chromium and Firefox. -> Completed.

2.2. Performance

- **2.2.1.** The Infrastructure *shall* be able to support at least 100 user actions per second. -> Not completed.
- **2.2.2.** The Infrastructure *shall* be able to support at least 10000 accounts. -> Completed.
- **2.2.3.** The Infrastructure *shall* have a response time of less than 1.0 second excluding network delay. -> Not completed.

- **2.2.4.** The Infrastructure *shall* have an uptime of at least 99%. -> Completed.

2.3. Privacy

- **2.3.1.** The System *shall* be compliant with GDPR. -> Not completed.
  - **2.3.1.1.** Users *shall* be given explicit and concise explanation of how their data may be used. -> Not completed.
  - **2.3.1.2.** The interface *shall* give the users a means of approving that they understood how their data may be used at the time of account creation. -> Not completed.
  - **2.3.1.3.** If any change of how data may be used, all users *shall* be given a notice and prohibited from using the System until they approve it. -> Not completed.

- **2.3.2.** Users *shall* have authority over visibility by third entities of data related to their accounts. -> Not completed.
  - **2.3.2.1.** The interface *should* provide a simple page in which the users can select this. -> Not completed.

2.4. Security

- **2.4.1.** Data that shall be reliably recorded in the Infrastructure *should* be recorded using a means of redundancy, such as using RAID. -> Completed.

- **2.4.2.** Application layer protocol used by The System *shall* be restricted to HTTPS. -> Not completed.

- **2.4.3** Passwords *shall* be hashed before being stored and verified using a NIST-approved cryptographically secure hash function using a salt that depends on the exact user and a system-wide nonce acquired from a reliable entropy source. -> Completed.

1.3 The changes your team has planned/implemented to improve the development process based on previous milestone reflections and how they impacted your process.

In the previous milestone, since we have completed the part until the presentation without -nearly- any problem, we continued our project as planned as a group. But it created some problems while satisfying the expectations of our customer. So, we prepared our roadmap of the final milestone and prioritized the missing parts according to the feedback given during our previous milestone presentation and the answers for our questions.

In the previous milestone, we had two main problems: One of them was the “effort” term and the other one was about the path structure and its functionality.

The “effort” term was used for showing the difficulty level of a path, and the level was calculated according to the values that users gave for that path. However, during our previous presentation we had a hard time describing what exactly it was used to denote because the word was found

ambiguous. After the presentation, we replaced the “effort” term with “difficulty” term and the problem was solved.

According to the definition of our project, the personal development platform (Renaissance) should be an interactive space and users can develop themselves taking the advantage of collective knowledge. However, in the previous milestone, we only had a comment section under each path provided interaction (other than path creation), and the customer did not find it sufficient. Also, the milestone structure was not liked. So, we changed the structure of the path. Tasks were added which can be related to milestone and the checkboxes moved under tasks’ part. Since the “comment” idea was not liked, we have converted the “comment” part into “resources” part. In this way, users can share the resources they wanted related to any milestone or task of that path. Any related resources of a task or milestone can be visible only by clicking that task or milestone.

At the end, we saved time to improve our core functionalities which are path structure and activity stream by giving more importance on them.

1.4 Final Project Evaluations: Reflections related to your Final Milestone Demo including lessons learned.

Evaluation of some factors affecting final version of the project:

- Renaissance: The app was satisfying most of the functionalities which are activity stream, path operations, follow/unfollow tag/path mechanisms, tag component (using wiki data), search of users/tags/paths, tag/path suggestion mechanisms. The main missing point can be not designing the path structure to meet the expectations from the customer. It was difficult to fix this problem by trying to stick to the requirements. A more radical change was needed, and even at the beginning of this semester was too late to make this change because finalizing project outline and renewing the diagrams/project plan etc. until the implementation part was hard enough to complete.

- Scenarios & presentations: The general presentation structure: 5 minutes for android, 5 minutes for web and the last 5 minutes for the Q&A session. We have created our scenarios to show the main functionalities in a story and we have rehearsed the presentations before the actual presentation. We were satisfied with the scenarios and presentations, except that we had difficulty in managing the milestone 2 presentation well and that some points were not explained clearly.

- Deployment: The Renaissance was deployed before the first milestone, so we had no problem with that.

- Bug: We have not encountered any problems while presenting the Renaissance related to any bug in the code.

- Latency: Especially at the milestone 2 presentation, reaching some pages took time longer than expected, so we have opened those pages in other tabs to prevent delays. However, the app still does not meet the expected performance.

- Meeting the deadlines: As a group, we generally had no problems with meeting deadlines. We made releases, submissions, and presentations on time, even though there were sometimes missing features.
- Group meetings: During the semester, the attendance to the meetings was almost complete and the meeting notes were kept as planned and uploaded to the GitHub Wiki.

General evaluation:

Final Milestone Demo went as expected as a group and all group members were active in the decision making process. The main functionality of the project should have been developed and should have been in the plan. For this, more careful arrangements should be made on the requirements and more active communication with the customer should have been made.

1.5 Final Release Notes

What's Changed

Frontend

- About and licenses page added.
- Frontend unit tests for util functions are implemented.
- Frontend unit tests are implemented.
- Activity-Stream page implemented.
- Path page updated for resource.
- Search box bugfix.
- Profile page update for (un)follow button.
- List of follower/followings added to Frontend profile page.
- Path page updated for task.
- Home page updated to use actual data.
- System manual for Frontend added.
- .yml files updated.

Backend

- Resouce model created.
- `add-resource` endpoint implemented.
- Backend unit tests are implemented.
- `search-path` endpoint updated.
- `related-topic` endpoint bugfix.
- A new model Task created.
- JSON-LD templates created for Activity-Stream.

- Activity-Stream feature implemented.
- `get-finished-paths` endpoint implemented.
- `finish-path` endpoint updated.
- Task features(`add-task` , `finish-task`) added under path.
- `get-profile` endpoint updated.
- `search-topic` endpoint bugfix.
- `popular` , `for-you` , and `new` endpoints implemented for home page.

Android

- Path page updated for resource.
- Android unit tests are implemented.
- Android navigation updated.
- Implementation of User&Tag Container.
- Search page implemented.
- Topic page implemented.
- Activity-Stream page implemented.
- Android UX update for add/remove topic on path creation.
- Android UX update for task/milestone designs.
- Path page updated for task.
- Home page updated to use actual data.

2. Summary of work

Company	Contact
Kürşat	• Wrote unit tests • Made UX improvements • Implemented Resources feature • Added realistic data to the DB • Connected home page to backend • Implemented user profile refresh button • Fixed miscellaneous bugs on the app • Reviewed Android Interfaces part of this report.
Hazar	• Documentation and assessment of the previous milestone • The “Search Path, Profile and Topic” feature is implemented in mobile. • "Create Path” feature is revised and updated as updates on the path design in mobil. • UX improvements are done in various fields of mobile design.

	• Unit tests are written • The user manual for the mobile part is created as final report deliverable. • Mobile part of the presentation is done by me and Fikri. • Presentation scenario were created with Hazar and Fikri Cem.
Fikri	• Activity Stream is implemented • Task feature is implemented • Path page implemented according to task feature • Finish path feature is implemented. • UI responsiveness improved. • UX improvements has been made. • Topic Container click navigation feature is implemented. • Topic path feature is implemented. • Backend connections are made. • Unit test are implemented
Emin	• Complete user page, Users could be un/followed on their profiles. User can list their followers and followings. The un/follow button on the profile page should be colored. • Homepage mock data is removed and connected to backend • Github action bug is fixed. • Searchbox results are wrong due to HTTP response order, that is fixed. • Writing utils and their tests • About and license page
Meltem	• Renaissance was transferred from parcel to create-react-app. • 'Path' structure was changed. (e.g., tasks were added) • 'Resources' part was added inside the path component. • Unit tests were written but deleted after fail. • Meeting notes 20.1 were uploaded. • Scenario for the presentation was prepared. • Web part of the final milestone was presented. • System manual part (for web) was written with the front-end group. • Paths were favorited and enrolled with different users to make the app look like a real app. • The "Status of deliverables", "Status of all requirements", "Changes description", "Final project evaluations", "Web interfaces" parts of the final milestone report were prepared.
Bekir	• Activity Stream page is implemented and connected to the backend.

	• Made research about Redux unit tests • UI/ UX improvements related to the path page • Dockerized the application • Made Contribution to the CRA migration • Meeting notes 22.1 were uploaded. • System manual part (for web) was written with the front-end group.
Selman	• Most of the Activity Streams related implementation • Part of Wikidata integration • Some API definitions • Global and personalized Activity Streams endpoints • Finish path endpoint • Rate path & rate path's effort endpoints • Finish path endpoint • Favorite/Unfavortie path endpoints • Wordcloud generation endpoints for users and paths • Postman testing of many endpoints • Unit tests of Authentication Django app • • Pull request reviews and conflict resolutions for some backend features.
Eren	• Edit path endpoint • My paths endpoint • My topics endpoint • Get path endpoint • Search topics endpoint • Get paths of related topics endpoint • Enroll path endpoint • Get followed/enrolled paths endpoints response update • Follow user endpoint • Review a part of backend implementations • Create issues for some missing endpoints • Deployment of the latest stable code
Yasin	• Planning on milestone 5 features • Research on Activity Stream - modeling • Task modeling • Updating 'create-path' Endpoint • Updating 'search-path' Endpoint

	• Finish-Unfinish Task Endpoints • Get Finished Paths Endpoint • Unit Tests • Tests on Postman • Contributing API documentation

3. The work that has been completed related to the features necessary to realize this scenario using our application

Since our app is competible with Wikidata Semantic Tags, Monica should search (Android/Web) for related tags such as healty-living,fermented-foods, diet etc. After navigating topic page (Android/Web) Monica can see the related paths for topics here and navigate through paths (Android/Web) to see which one of them suitable for Monica. To track her progress Monica can complete the tasks and see her status according to milestones. For socializing, there is not any message or chat section. However, we still value the communties' effort on others' development. So, resource part is implemented. Monica can access it from resources section on path page and provide link for other users in system or see through already provided links to improve herself during that path.

4. W3C Activity Streams 2.0 standard and Wikidata

Wikidata Integration

Our definition *topic* as in our requirements specification is associated with a Wikidata entity. In our databases, topics are always stored as an integer denoting their Wikidata entity ID. Whenever a topic name must be returned from the backend, the most up-to-date representation of that entity is fetched using Wikidata API.

Further integration is provided in the `GET topic/search-topic/<search_text>/` endpoint, which gives related topics with their descriptions and user-specific favorited information. This functionality is used in the search interface and to recommend linkable topics to users providing content.

Activity Streams 2.0

Logs of user activity are kept in the database in a custom format that can be efficiently converted into an Activity Streams 2.0-compliant JSON-LD object. There are two backend endpoints that can be used to retrieve these logs:

- Global Activity Streams: `user/global-activity-streams/`
- User-Specific Activity Streams: `user/activity-streams/`

Upon an HTTP `POST` request with an empty body, both endpoints respond with a list of Activity Streams 2.0-compliant JSON-LD objects. The former endpoint gives activity streams logs in the whole community while the latter only gives those logs that the logged in user might be interested in.

User-specific endpoint (`user/activity-streams/`) can only be used by logged in users (with an approporiate jwt token in the `auth` header). Community-wide endpoint (`user/global-activity-streams/`) does not require such authentication.

Personalization

The way user specific activity streams endpoint determines which logs a user might be interested in is determined by the following:

- The actor of the activity streams log is a user followed by the requesting user, OR
- The object of the activity streams log is a path followed by the requesting user, OR
- The object of the activity streams log is a topic (Wikidata entity) favorited by the requesting user

Field Definitions

- Summary: A human-readable sentence describing the event.
- Type: "Create", "Join" or "Follow" depending on the actual event executed. The values are limited by the valid type fields defined in the standard.
- Actor: The user in the system that performed the action. Its type is always "Person".
- Object: Whatever entity is acted on the system.
- Published: ISO-formatted datetime string in UTC time.

Example Query

The server is currently deployed with the IPv4 address `35.209.23.51` . As described above, the global activity streams endpoint is accessible without logging in. An example query in Postman and its output is provided below. Also, simply executing `curl -X POST http://35.209.23.51/user/global-activity-streams/` works.

POST 35.209.23.51/user/global-activity-streams/ Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (11) Test Results Status: 200 OK Time: 1943 ms Size: 28.46 KB Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "@context": "https://www.w3.org/ns/activitystreams",
        "summary": "eren2 favorited topic Norway.",
        "type": "Follow",
        "actor": {
            "type": "Person",
            "name": "eren2"
        },
        "object": {
            "id": 20,
            "type": "Document",
            "name": "Norway"
        },
        "published": "2022-01-03T12:15:50.884948"
    },
```


In order to get a personalized activity streams output, a jwt token should be acquired from the endpoint `authentication/login/` throug a `POST` request with a body of JSON-encoded fields "username" and "password" with string values, for example: `{"username":"robertdown","password":"asd1234"}` . The endpoint gives a jwt token as a string. That string can be included in the auth header in the subsequent request `POST` ed to the `user/activity-streams/` otherwise same as specified above for `user/global-activity-streams/`

5. API

API Endpoint Definitions

Link (http://35.209.23.51/)

Authentication (`authentication/`)

Sign up (`authentication/signup/`) (`POST`)

Request Data Fields

- `username` : Username for the user
- `email` : E-mail address to be used later
- `password` : Password to be used for authentication
- `firstname` : Given name of the user

- `lastname` : Surname of the user
- `bio` (optional): A description to be displayed in the user's profile page
- `photo` (optional): BASE64-encoded image file to be used as the profile picture of the user

Response Data Fields

- `detail` : A string equal to `USER_CREATED` denoting success of the operation.

Authorization Level: None

Example

Request:

Response:

```
{
    "detail":"USER_CREATED"
}
```

Login (`authentication/login/`) (`POST`)

Request Data Fields

- `username` : Username of the user.
- `password` : Password of the user.

Response

A JWT token (as a string, not JSON) is returned.

Authorization Level: None

Example

Request:

```
{
    "username": "Eren",
    "password": "asd1234"
}
```

Response:


```
29tIiwiaXNBZG1pbiI6ZmFsc2UsImV4cCI6MTYzNjgzMTgwOX0.KWv65CJdQ5pp5SsyMCPGabx3TakKKmpBH9PJyZqHd6k
```


Refresh Token (`authentication/refresh-token/`) (`POST`)

Request Data Fields

- ``:

Response Data Fields

- ``:

Authorization Level: *Logged in User*

JWT in `auth` section of the header required. See "Login" for details.

Example

Request:

Response:

Forgot Password (`authentication/forgot-password/`) (`POST`)

Request Data Fields

- `username` : Username of the user.

Response Data Fields

- `detail` : Response message.

Authorization Level: *None*

Example

Request:

```
{
    "username": "Eren"
}
```

Response:

```
{
    "detail": "EMAIL_SENT"
}
```

User (`user/`)

Edit User (`user/edit-user/`) (`POST`)

Request Data Fields

- `firstname` (optional): First name of the user to be displayed.
- `lastname` (optional): Surname of the user to be displayed.
- `isDeleted` (optional): Boolean. Equivalent to using "Delete" endpoint.
- `bio` (optional): Text to be displayed in the user's profile page.
- `photo` (optional): BASE64-encoded image file. Is set as the profile picture of the user.

Response Data Fields

- `username` : Username of the user.
- `email` : Email of the user.
- `firstname` : First name of the user.
- `lastname` : Surname of the user.
- `isAdmin` : Boolean. Admin status of the user.
- `isBanned` : Boolean. Ban status of the user.
- `createdAt` : Create date of the user profile.
- `updatedAt` : Update date of the user profile.
- `lastLogin` : Last login date of the user profile.
- `finishedResourceCount` : Count of finished resources by the user.
- `isVerified` : Boolean. Profile verification status.

- `bio` : Text to be displayed in the user's profile page.
- `photo` : BASE64-encoded image file of the profile picture.

Authorization Level: *Logged in User*

JWT in `auth` section of the header required. See "Login" and "Refresh Token" for details.

Users can access the endpoint for only their `username` s in the JWT token.

Example

Request:

```
{
    "lastname": "Seremet",
    "bio": "I'm a computer engineering student."
}
```

Response:

```
{
    "username": "YasinDeneme1",
    "email": "yasindeneme1@gmail.com",
    "firstname": "yasin",
    "lastname": "Seremet",
    "isAdmin": false,
    "isBanned": false,
    "createdAt": 1636189688,
    "updatedAt": 1636189688,
    "lastLogin": 1636833653,
    "finishedResourceCount": 0,
    "isVerified": false,
    "bio": "I'm a computer engineering student.",
    "rememberMe": true
}
```

Search User (`user/search-user/<search_text>`) (`GET`)

Request Data Fields

None

Last portion of the URI is the query text.

Response Data Fields

Response data is a JSON list of `username` fields.

Authorization Level: *None*

Example

Request:

Response:

Ban User (`user/ban-user/`) (`POST`)

Request Data Fields

- `username` : Username of the user.

Response Data Fields

- `detail` : Response message.

Authorization Level: *Logged in Admin*

JWT in `auth` section of the header required. See "Login" and "Refresh Token" for details.

Example

Request:

```
{
    "username": "Eren"
}
```

Response:

```
{
        "detail": "User with username: Eren banned successfully"
}
```

Get Profile (`user/get-profile/<username>`) (`GET`)

Request Data Fields

None `username` is the last field in the URI

Response Data Fields

- `username` : Username of the user.
- `email` : Email of the user.
- `firstname` : First name of the user.
- `lastname` : Surname of the user.
- `isAdmin` : Boolean. Admin status of the user.
- `isBanned` : Boolean. Ban status of the user.
- `createdAt` : Create date of the user profile.
- `finishedResourceCount` : Count of finished resources by the user.
- `isVerified` : Boolean. Profile verification status.
- `bio` : Text to be displayed in the user's profile page.

Authorization Level: *None*

Example

Request:

Response:

```
{
    "username": "Eren",
    "email": "erenaltunoglu@gmail.com",
    "firstname": "Eren",
    "lastname": "Altunoglu",
    "isAdmin": false,
    "isBanned": false,
    "createdAt": 1636469244,
    "finishedResourceCount": 7,
    "isVerified": false,
    "bio": "You know who I am"
}
```

Change Password (`user/change-password/`) (`POST`)

Request Data Fields

- `password` : the new password specified by user

Response Data Fields

- `username` : Username of the user.
- `email` : Email of the user.
- `firstname` : First name of the user.
- `lastname` : Surname of the user.
- `isAdmin` : Boolean. Admin status of the user.
- `isBanned` : Boolean. Ban status of the user.
- `createdAt` : Create date of the user profile.
- `updatedAt` : Update date of the user profile.
- `lastLogin` : Last login date of the user profile.
- `finishedResourceCount` : Count of finished resources by the user.
- `isVerified` : Boolean. Profile verification status.
- `bio` : Text to be displayed in the user's profile page.

Authorization Level:

JWT in `auth` section of the header required. See "Login" for details.

Example

Request:

```
{
    "password": "123456"
}
```

Response:

```
{
    "username": "Eren",
    "email": "erenaltunoglu@gmail.com",
    "firstname": "Eren",
    "lastname": "Altunoglu",
    "isAdmin": false,
    "isBanned": false,
    "createdAt": 1636469244,
```

```
    "updatedAt": 1636469244,
    "lastLogin": 1636827704,
    "finishedResourceCount": 0,
    "isVerified": false,
    "bio": "You know who I am",
    "rememberMe": true
}
```


Get Follower and Followed Users(`user/get-follow/`) (`GET`)

Request Data Fields

- `username` : Username for the user

Response Data Fields

- `followers` : A list of usernames
- `followed` : A list of usernames

Authorization Level: *Logged in User*

Follow User(`user/follow-user/`) (`POST`)

Request Data Fields

- `username` : Username for the user

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation. `ALREADY_FOLLOWED` on already existing relation.

Authorization Level: *Logged in User*

Unfollow User(`user/unfollow-user/`) (`POST`)

Request Data Fields

- `username` : Username for the user

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation. `NOT_FOLLOWED` on already existing relation.

Authorization Level: *Logged in User*

User Wordcloud(`user/wordcloud/`) (`POST`)

Request Data Fields

- `width` : (*optional*) integer width in pixels of the image requested
- `height` : (*optional*) integer height in pixels of the image requested
- `user` : string of the username to generate the wordcloud about

Response

A string that contains a BASE64-encoded PNG file.

Authorization Level: *Logged in User*

Global Activity Streams(`user/global-activity-streams/`) (`POST`)

Request Data Fields

- no data requested

Response

- `streams` : List of activity streams

Authorization Level: *Logged in User*

Activity Streams(`user/activity-streams/`) (`POST`)

Request Data Fields

- `username` : Username of the user

Response

- `streams` : List of activity streams

Authorization Level: *Logged in User*

Admin (`admin/`)

This interface is automatically generated by Django. It is not related to the admin role as defined in the design documents.

Path (`path/`)

Create Path (`path/create-path/`) (`POST`)

Request Data Fields

- `title` : Title of the path
- `description` : Description of the path
- `topics` : Related topics of the path
- `creator_username` : Username of the path creator
- `creator_email` : Email of the path creator
- `created_at` : The time the path is created
- `photo` : Photo of the path
- `milestones` : List of milestones related to the path
- `is_banned` : Boolean. The condition indicates the path whether it is banned or not.
- `is_deleted` : Boolean. The condition indicates the path whether it is deleted or not.

Response Data Fields

- `pathID` : ID of the created path.

Authorization Level: *Logged in User*

Edit Path(`path/edit-path/`) (`POST`)

Request Data Fields

- `path_id` : ID of the path
- `title` : Title of the path
- `description` : Description of the path
- `topics` : Related topics of the path
- `photo` : Photo of the path
- `milestones` : List of milestones related to the path

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation

Authorization Level: *Logged in User*

Finish Milestone (`path/finish-milestone/`) (`POST`)

Request Data Fields

- `username` : Username of the user
- `milestone_id` : ID of the milestone

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation

Authorization Level: *Logged in User*

Unfinish Milestone (`path/unfinish-milestone/`) (`POST`)

Request Data Fields

- `username` : Username of the user
- `milestone_id` : ID of the milestone

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation

Authorization Level: *Logged in User*

Rate Path (`path/rate-path/`) (`POST`)

Request Data Fields

- `rating` : Point given as rating
- `path_id` : ID of the path
- `username` : Username of the user

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation.

Authorization Level: *Logged in User*

Effort Path (`path/effort-path/`) (`POST`)

Request Data Fields

- `effort` : Point given as effort
- `path_id` : ID of the path
- `username` : Username of the user

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation.

Authorization Level: *Logged in User*

Enroll Path(`path/enroll-path/`) (`POST`)

Request Data Fields

- `username` : Username of the user
- `target` : ID of the path

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation. `ALREADY_ENROLLED` on already existing relation.

Authorization Level: *Logged in User*

Unenroll Path(`path/unenroll-path/`) (`POST`)

Request Data Fields

- `username` : Username of the user
- `target` : ID of the path

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation. `NOT_ENROLLED` on already existing relation.

Authorization Level: *Logged in User*

Get Enrolled Paths(`path/get-enrolled-paths/`) (`POST`)

Request Data Fields

- `username` : Username of the user

Response Data Fields

- `allPaths` : A list consisting of paths, each path has its `path_id` , `rating` , `effort` , `isFollowed` data.

Authorization Level: *Logged in User*

Follow Path(`path/follow-path/`) (`POST`)

Request Data Fields

- `username` : Username of the user
- `target` : ID of the path

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation. `ALREADY_FOLLOWED` on already existing relation.

Authorization Level: *Logged in User*

Unenroll Path(`path/unfollow-path/`) (`POST`)

Request Data Fields

- `username` : Username of the user
- `target` : ID of the path

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation. `NOT_FOLLOWED` on already existing relation.

Authorization Level: *Logged in User*

Get Followed Paths(`path/get-followed-paths/`) (`POST`)

Request Data Fields

- `username` : Username of the user

Response Data Fields

- `allPaths` : A list consisting of paths, each path has its `path_id` , `rating` , `effort` , `isFollowed` data.

Authorization Level: *Logged in User*

Finish Path(`path/finish-path/`) (`POST`)

Request Data Fields

- `username` : Username of the user
- `path_id` : ID of the path

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation.

Authorization Level: *Logged in User*

Get Path(`path/get-path/<path_id>/`) (`GET`)

Request Data Fields

- `username` : Username of the user

Response Data Fields

- `path` : Path with its data and all its milestones including `_id` , `title` , `body` , `isFinished` , `milestones` , `topics` .

Authorization Level: *Logged in User*

Get Related Path(`path/related-path/<topic_id>/`) (`GET`)

Request Data Fields

- `username` : Username of the user

Response Data Fields

- `paths` : A list consisting of paths, each path has its `path_id` , `rating` , `effort` , `isFollowed` , `isEnrolled` data.

Authorization Level: *Logged in User*

Search Path(`path/search-path/<search_text>/`) (`GET`)

Request Data Fields

- `username` : Username of the user

Response Data Fields

- `paths` : A list consisting of paths, each path has its `path_id` , `rating` , `effort` , `isFollowed` , `isEnrolled` data.

Authorization Level: *Logged in User*

My Paths(`path/my-paths/`) (`GET`)

Request Data Fields

- `username` : Username of the user

Response Data Fields

- `paths` : A list consisting of paths, each path has its `path_id` , `rating` , `effort` data.

Authorization Level: *Logged in User*

Path Wordcloud(`path/wordcloud/`) (`POST`)

Request Data Fields

- `width` : (*optional*) integer width in pixels of the image requested
- `height` : (*optional*) integer height in pixels of the image requested
- `path_id` : string key of the path to generate the wordcloud for

Response

A string that contains a BASE64-encoded PNG file.

Authorization Level: *Logged in User*

Finish Task(`path/finish-task/`) (`POST`)

Request Data Fields

- `username` : Username of the user
- `milestone_id` : id of milestone

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation

Authorization Level: *Logged in User*

Unfinish Task(`path/unfinish-task/`) (`POST`)

Request Data Fields

- `username` : Username of the user
- `milestone_id` : id of milestone

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation

Authorization Level: *Logged in User*

Add Resource(`path/add-resource/`) (`POST`)

Request Data Fields

- `username` : Username of the user
- `path_id` : id of path

- `order` : number of order
- `link` : a URL link directing the resource
- `description` : explanation about the resource

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation

Authorization Level: *Logged in User*

Get Popular(`path/popular/`) (`GET`)

Request Data Fields

- `username` : Username of the user

Response Data Fields

- `topics` : List of popular topics
- `paths` : List of popular paths

Authorization Level: *Logged in User*

Get New(`path/new/`) (`GET`)

Request Data Fields

- `username` : Username of the user

Response Data Fields

- `topics` : List of new topics
- `paths` : List of new paths

Authorization Level: *Logged in User*

Get For You(`path/foryou/`) (`GET`)

Request Data Fields

- `username` : Username of the user

Response Data Fields

- `topics` : List of topics suggested for given user
- `paths` : List of paths suggested for given user

Authorization Level: *Logged in User*

Topic (`topic/`)

Favorite Topic (`topic/favorite-topic/`) (`POST`)

Request Data Fields

- `target` : ID of the topic

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation. `ALREADY_FAVORITE` on already existing relation.

Authorization Level: *Logged in User*

Unfavorite Topic (`topic/unfavorite-topic/`) (`POST`)

Request Data Fields

- `target` : ID of the topic

Response Data Fields

- `detail` : A string equal to `SUCCESSFUL` denoting success of the operation. `NOT_FAVORITE` on already non-existing relation.

Authorization Level: *Logged in User*

Search Topics (`topic/search-topic/<search_text>/`) (`GET`)

Request Data Fields

Response Data Fields

- A list of topic objects. A topic object will include `name`, `ID`, `description` and `isFav` fields.

Authorization Level: *Logged in User*

Get Topic (`topic/get-topic/<topic_id>/`) (`GET`)

Request Data Fields

Response Data Fields

- A list of topic objects. A topic object will include `name`, `ID`, `description` and `isFav` fields.

Authorization Level: *Logged in User*

Related Topics (`topic/related-topic/<topic_id>/`) (`GET`)

Request Data Fields

Response Data Fields

- A list of topic objects. A topic object will include `name`, `ID` and `isFav` fields.

Authorization Level: *Logged in User*

MY Topics (`topic/my-topic/`) (`GET`)

Request Data Fields

Response Data Fields

- A list of topic objects. A topic object will include `name`, `ID`, `description` and `isFav` fields.

Authorization Level: *Logged in User*

- URls to access the APIs is provided in the link above.

Postman Collection

- Since requests in public workspaces cannot be sent, firstly, you should fork this collection to your local to test requests.
- After forking, initially, login should be done and authentication token in the response should be pasted to next requests' Bearer Token area.

- Create Path will return a path ID in response. The sample body in collection in 'Create Path' is not saved in database. You can send and try it.
- 'Get Related Paths of Topic' & 'Search Topic' APIs do not require body.
- I'm also adding issues & screenshots of the related requests. Create Path Get Related Paths of Topic Search Topic

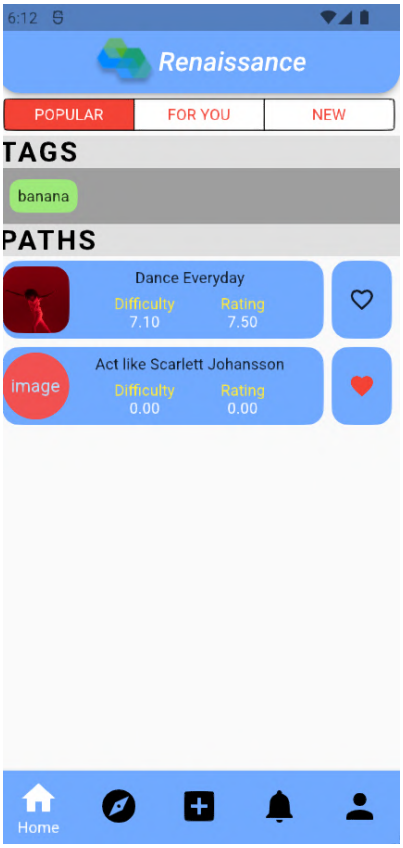
6. Interfaces & Code URLs

Home Page

Mobile Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/home_page.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/tag_container.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/course_container.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/file_converter.dart

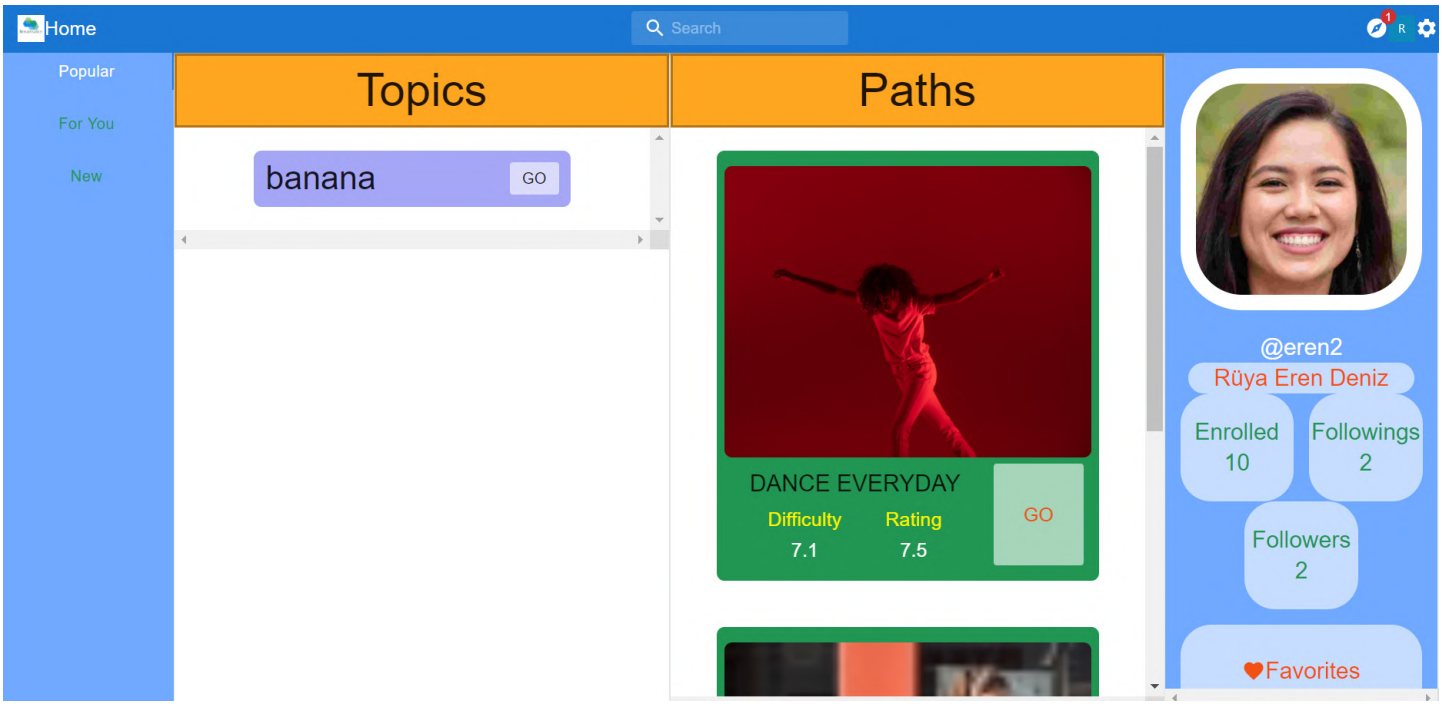
Mobile UI:

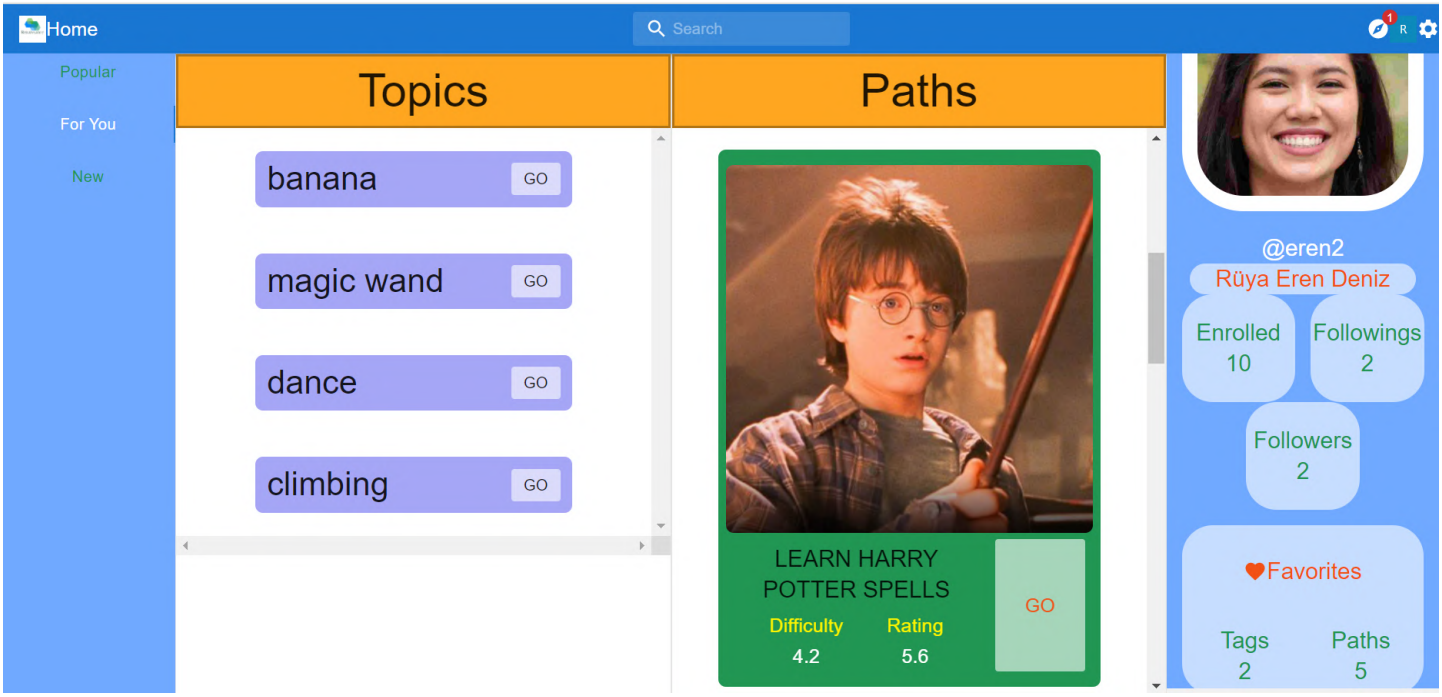


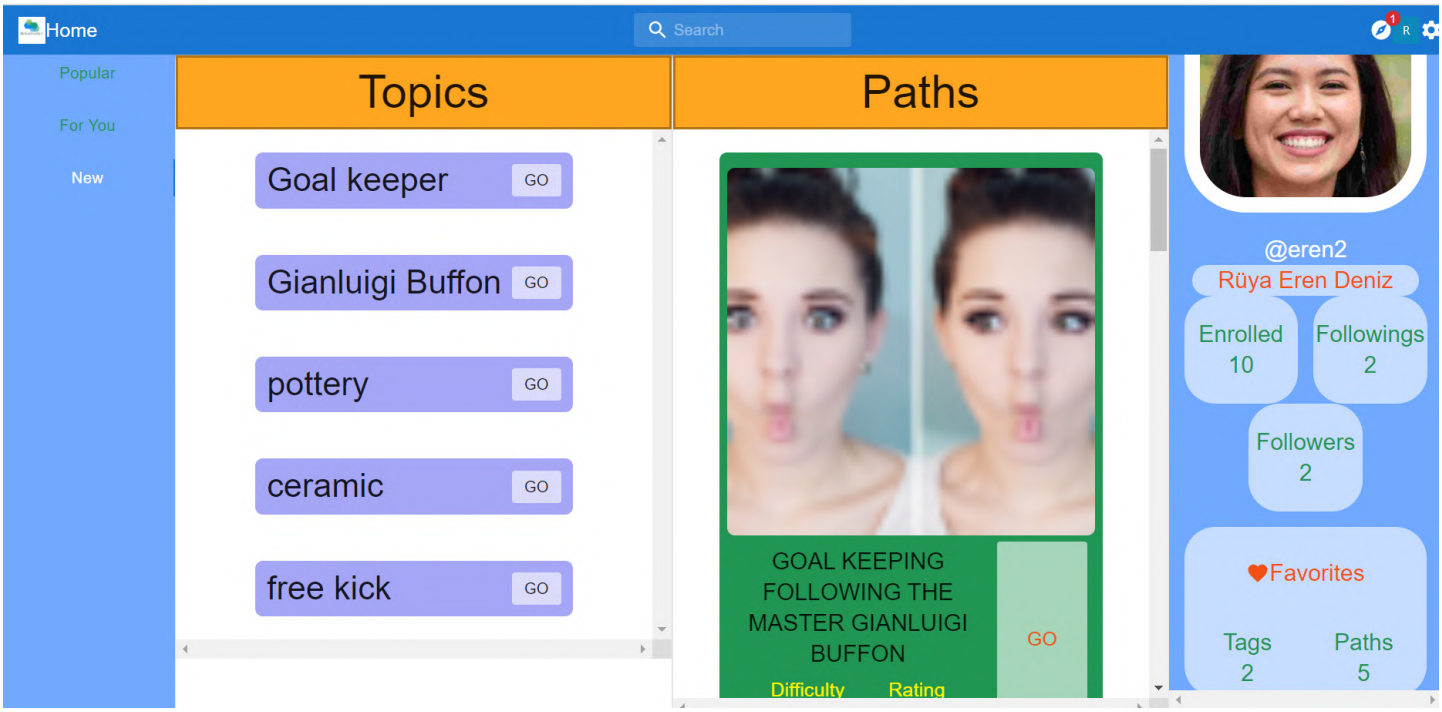
Web Code:

https://github.com/bounswe/2021SpringGroup11/tree/master/source/frontend/src/components/Home

Web UI:





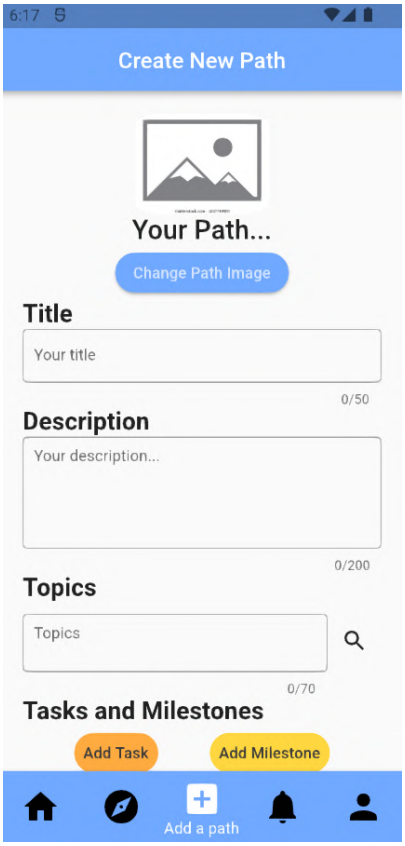


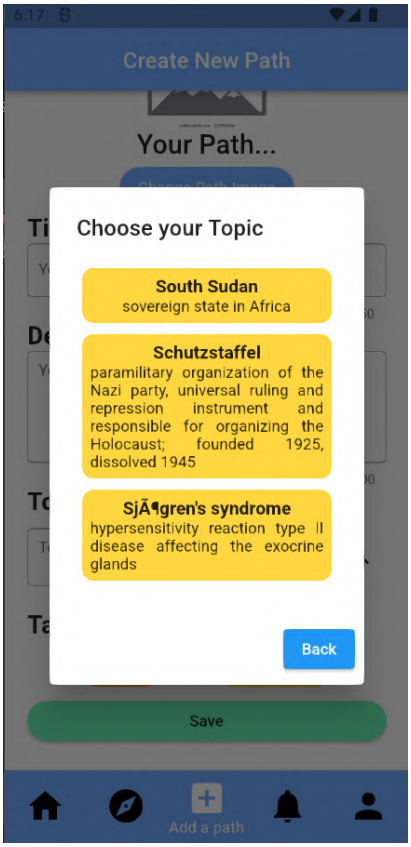
Create Path

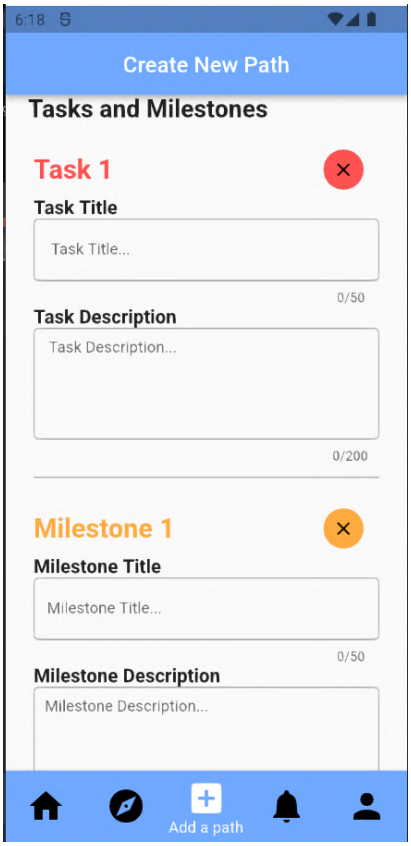
Mobile Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/create_path.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/file_converter.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/tag_desc_container.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/tag_create_container.dart

Mobile UI:


6:17
Create New Path
Your Path...
Change Path Image
Title
Your title
0/50
Description
Your description...
0/200
Topics
Topics
0/70
Tasks and Milestones
Add Task
Add Milestone
Add a path

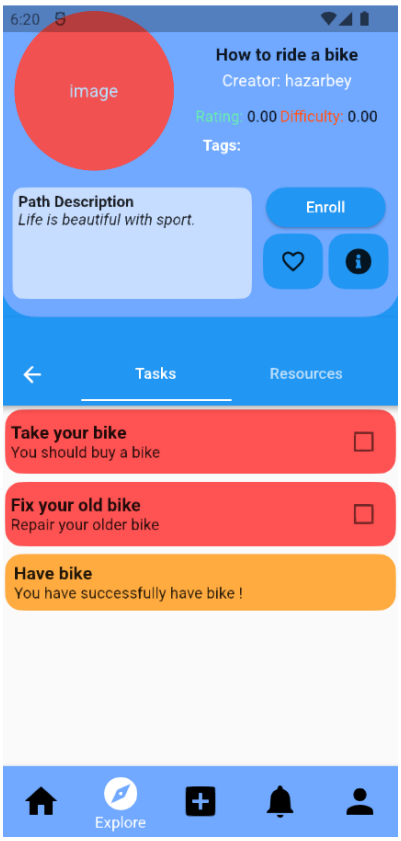

6:17
Create New Path
Your Path...
Choose your Topic
South Sudan
sovereign state in Africa
Schutzstaffel
paramilitary organization of the Nazi party, universal ruling and repression instrument and responsible for organizing the Holocaust; founded 1925, dissolved 1945
SjÃ¶gren's syndrome
hypersensitivity reaction type II disease affecting the exocrine glands
Back
Save
Add a path

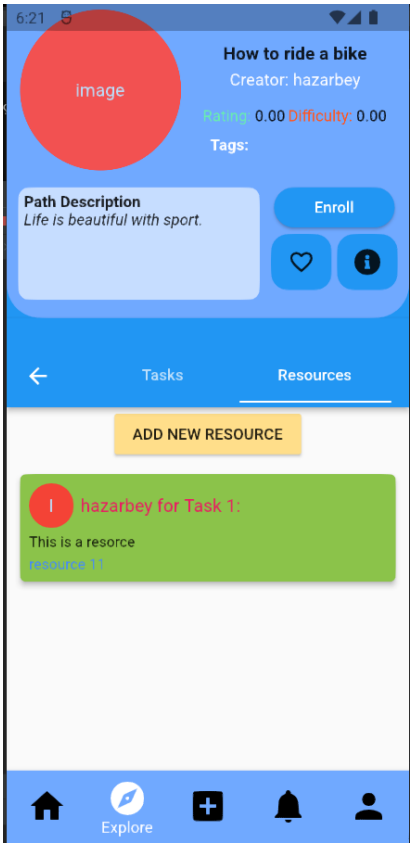


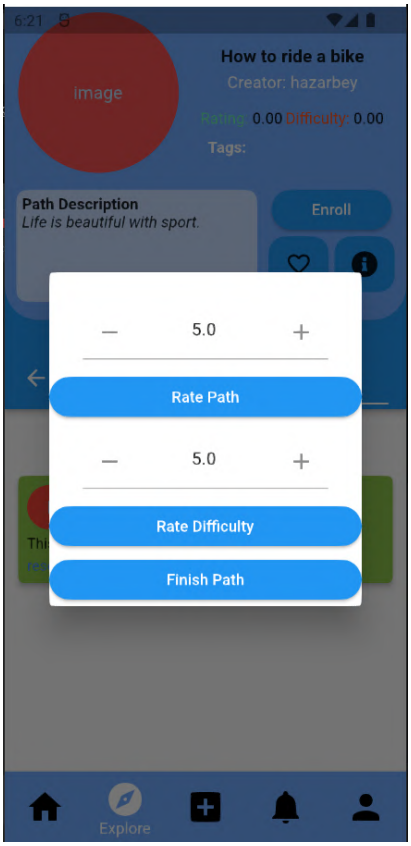
Path Page

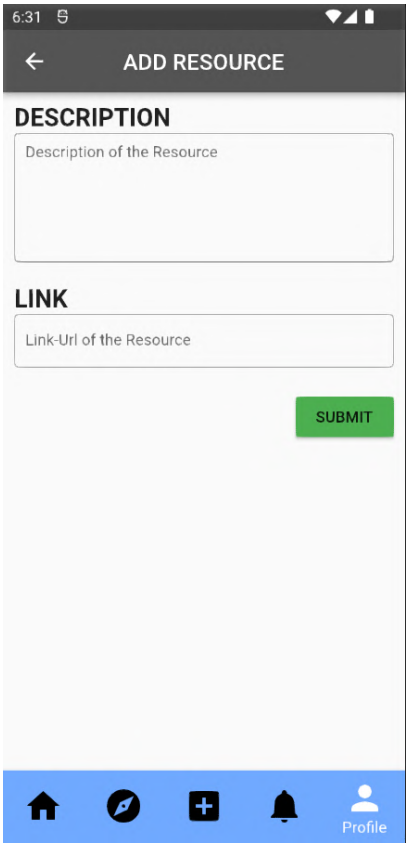
Mobile Code:
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/path_page.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/add_resource_page.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/milestone_widget.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/cutsom_checkbox_widget.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/file_converter.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/resource_container.dart

Mobile UI:


6:20
image
How to ride a bike
Creator: hazarbey
Rating: 0.00 Difficulty: 0.00
Tags:
Path Description
Life is beautiful with sport.
Enroll
Tasks
Resources
Take your bike
You should buy a bike
Fix your old bike
Repair your older bike
Have bike
You have successfully have bike !
Explore


6:21
image
How to ride a bike
Creator: hazarbey
Rating: 0.00 Difficulty: 0.00
Tags:
Path Description
Life is beautiful with sport.
Enroll
Tasks
Resources
ADD NEW RESOURCE
I
hazarbey for Task 1:
This is a resorce
resource 1.1
Explore

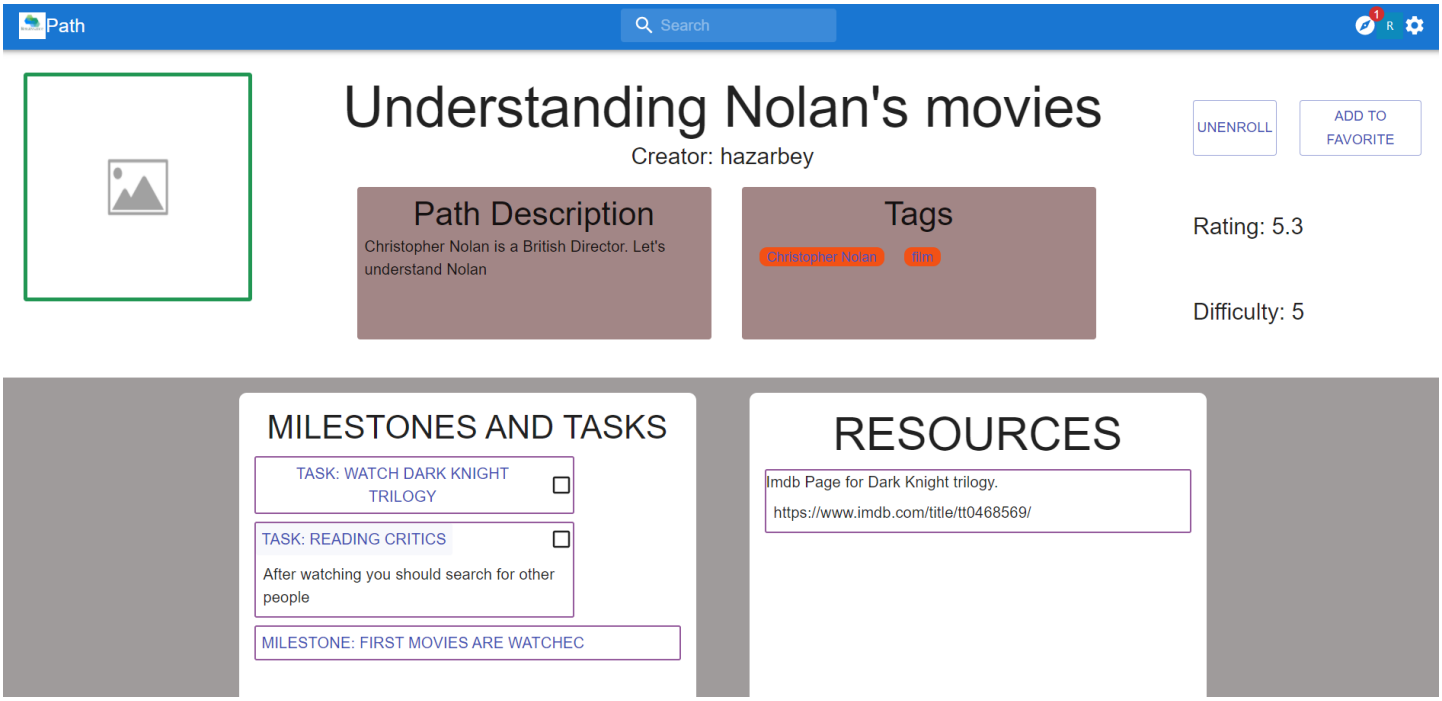




Web Code:

https://github.com/bounswe/2021SpringGroup11/tree/master/source/frontend/src/components/Path

Web UI:

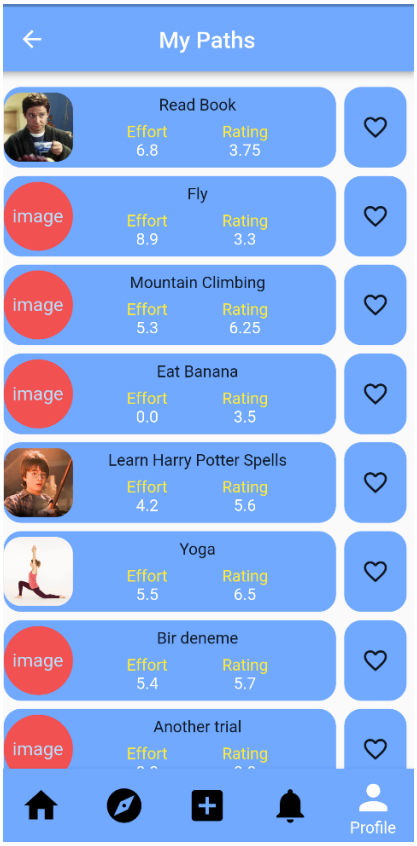


My Paths

Mobile Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/my_paths_page.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/course_container.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/file_converter.dart

Mobile UI:

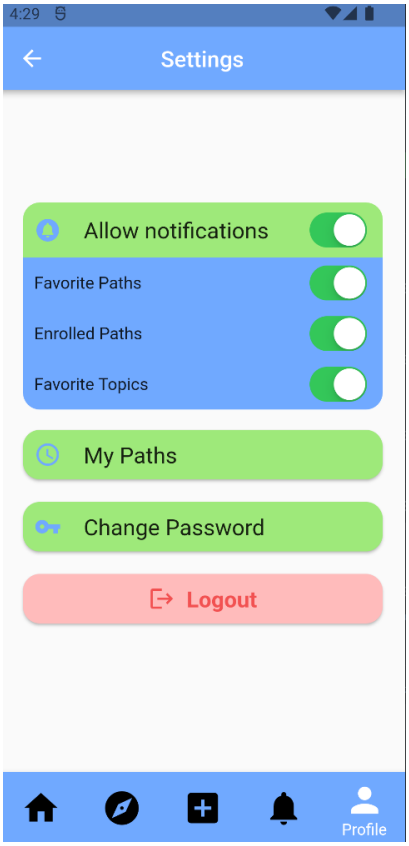


Settings Page

Mobile Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/settings_page.dart

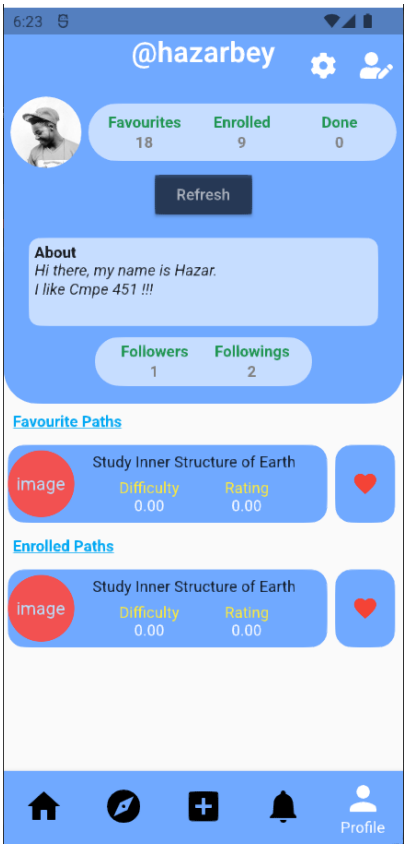
Mobile UI:



Profile Page

Mobile Code:

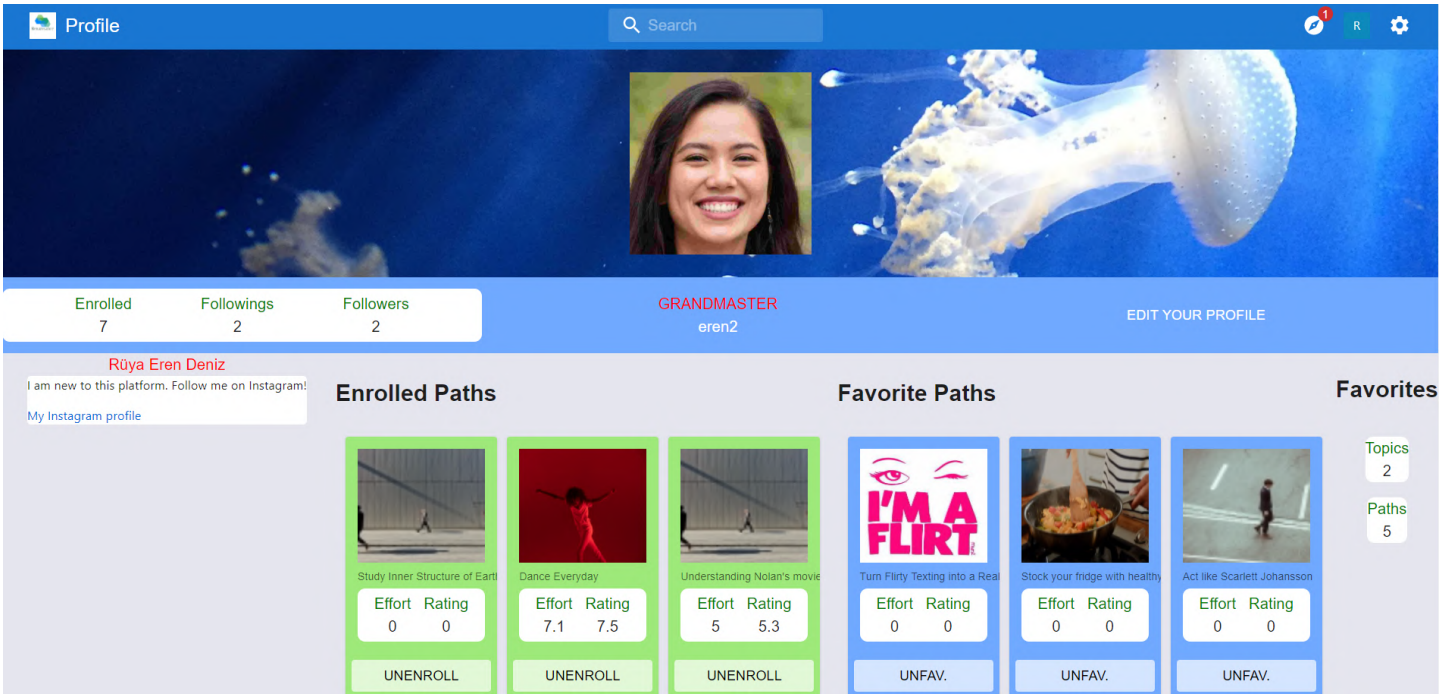
Mobile UI:

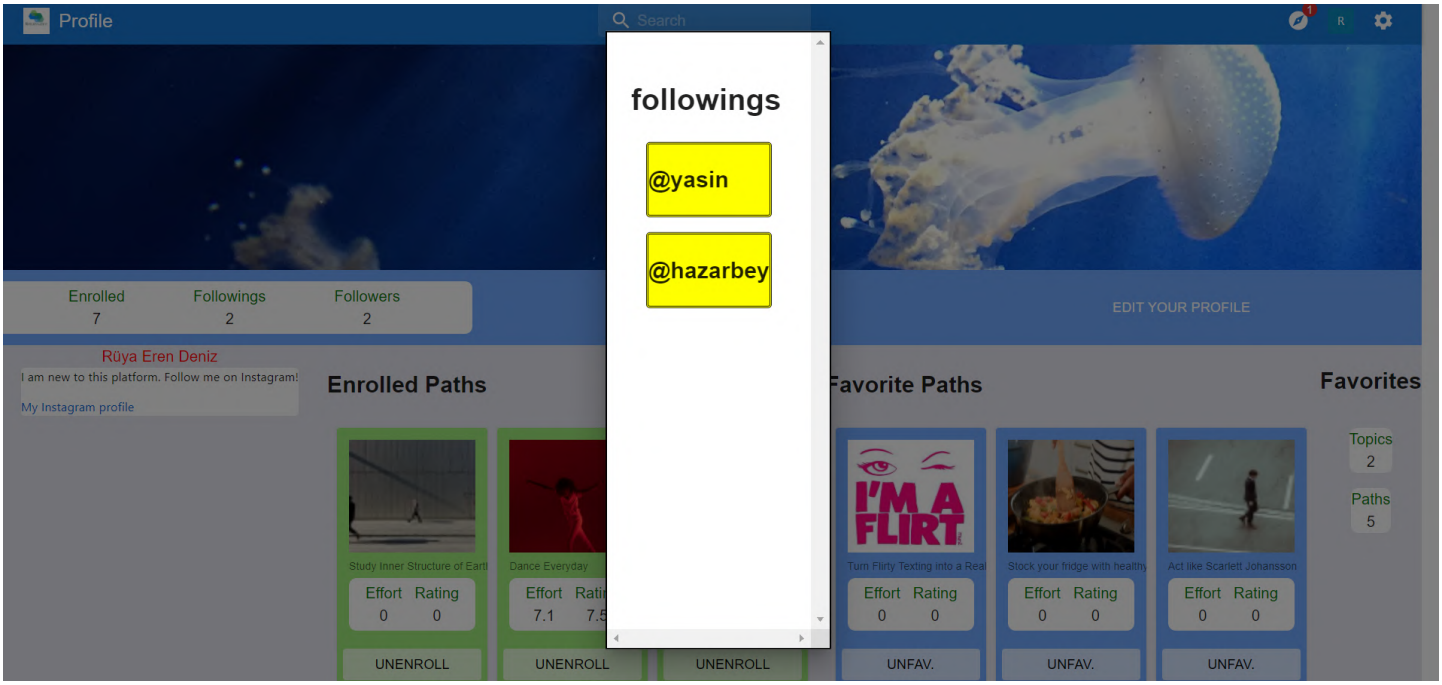


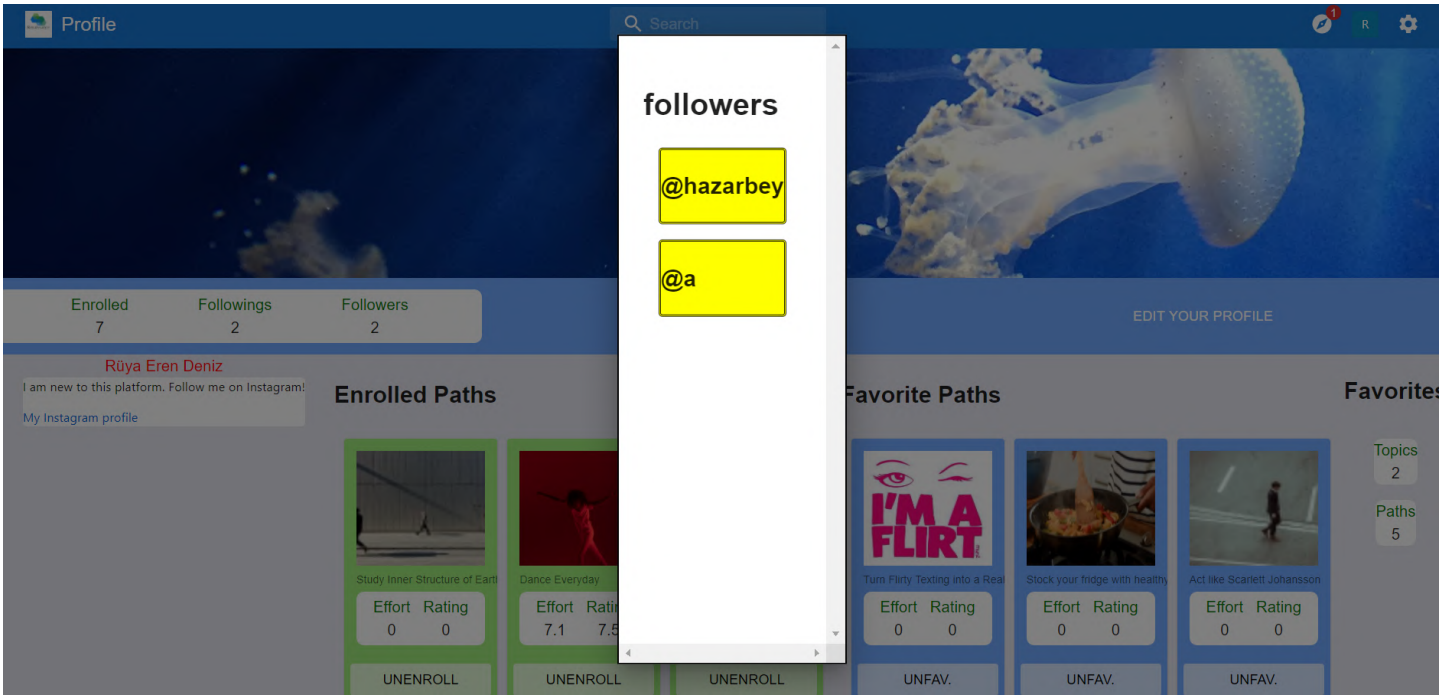
Web Code:

https://github.com/bounswe/2021SpringGroup11/tree/master/source/frontend/src/components/Profile

Web UI:





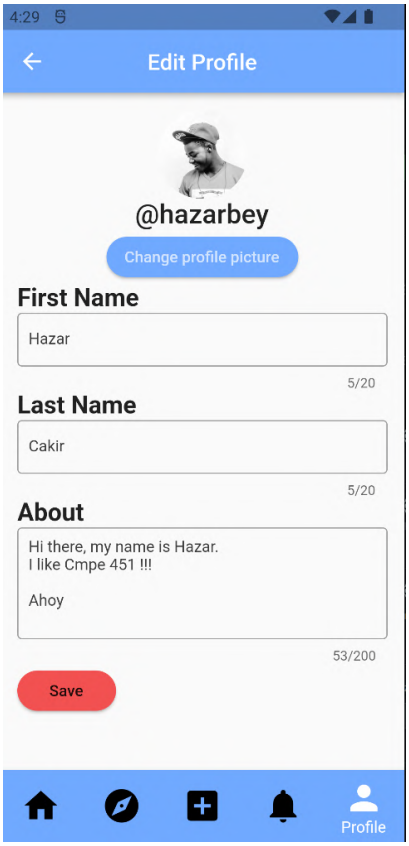


Edit Profile Page

Mobile Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/edit_profile.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/file_converter.dart

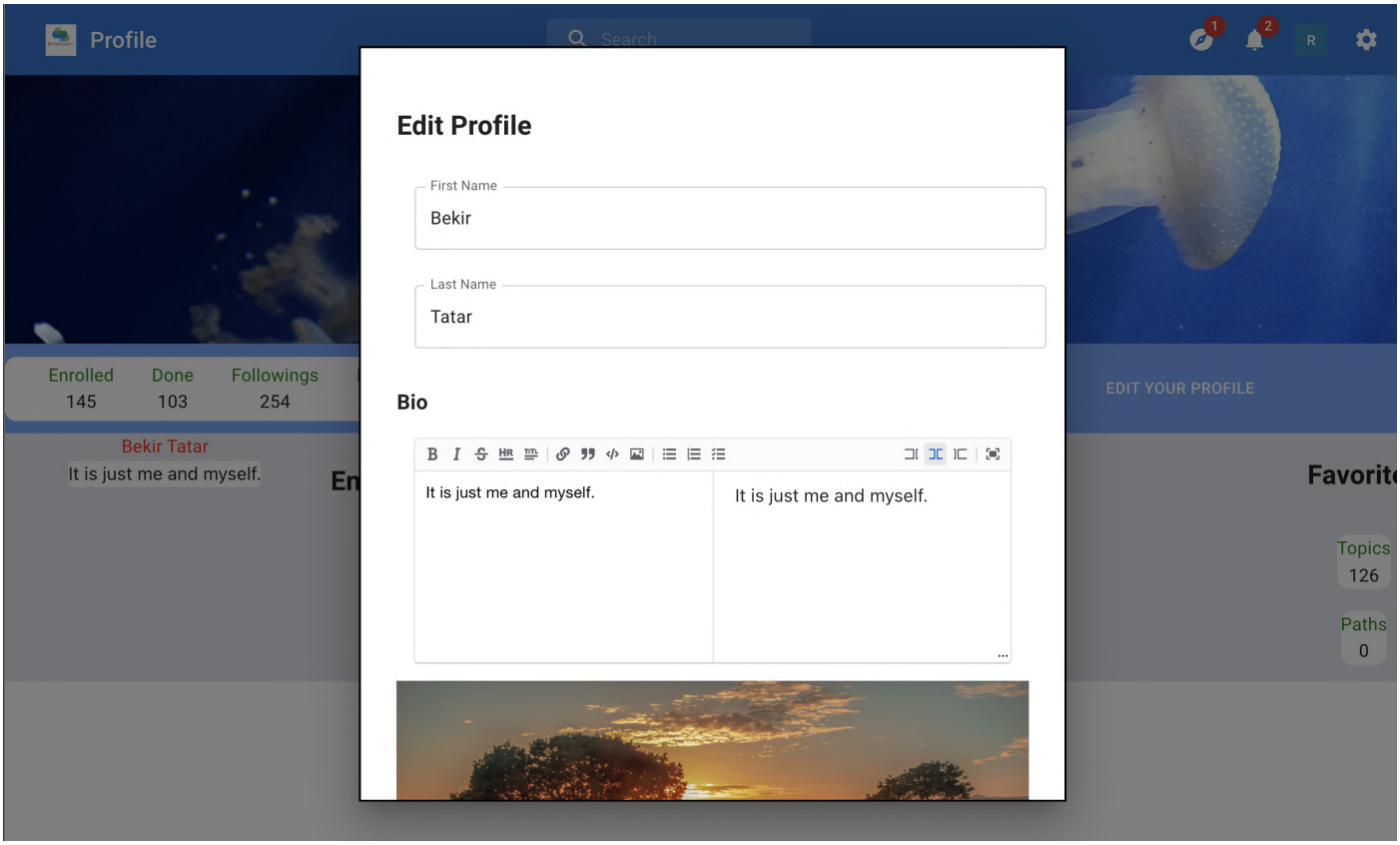
Mobile UI:



Web Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/frontend/src/components/Profile/index.tsx
https://github.com/bounswe/2021SpringGroup11/blob/master/source/frontend/src/components/Profile/helper.ts

Web UI:

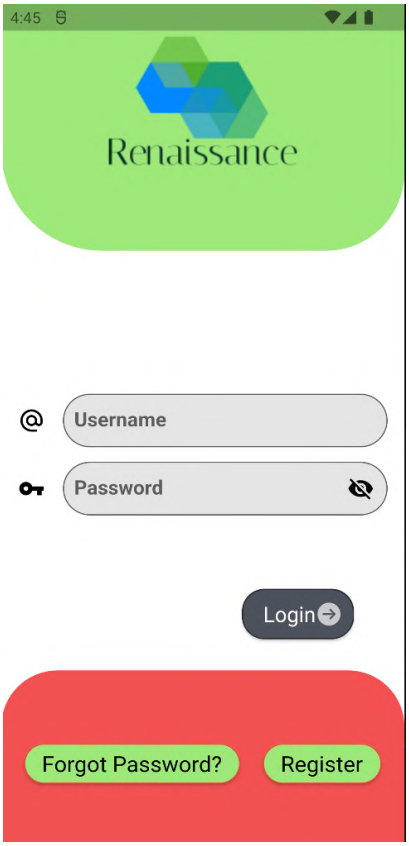


Log In Page

Mobile Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/login_page.dart

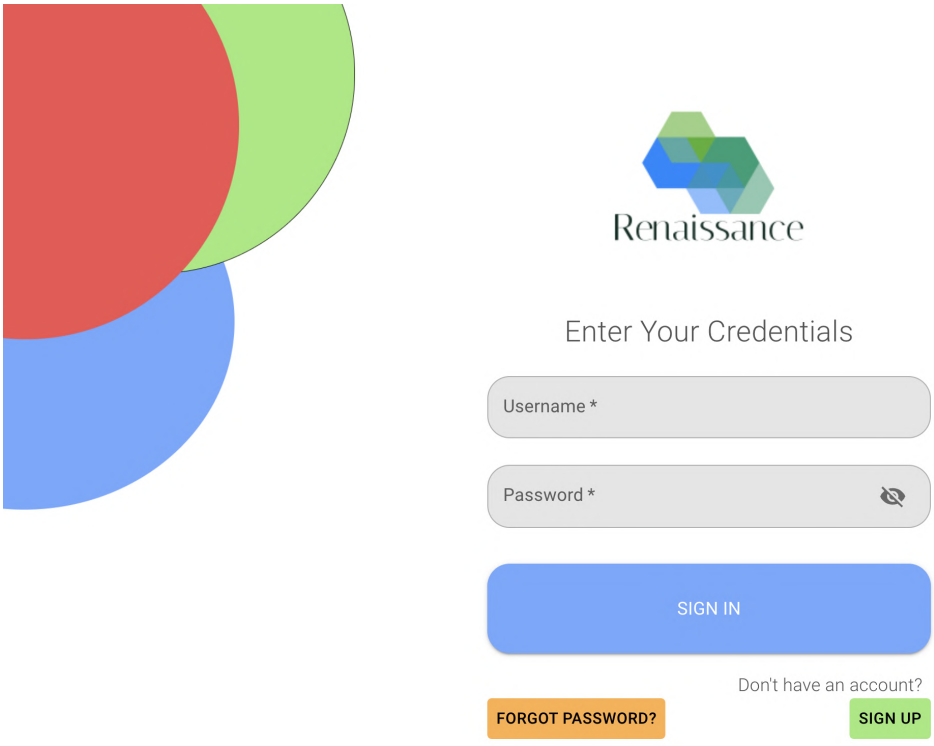
Mobile UI:



Web Code:

https://github.com/bounswe/2021SpringGroup11/tree/master/source/frontend/src/components/Login

Web UI:

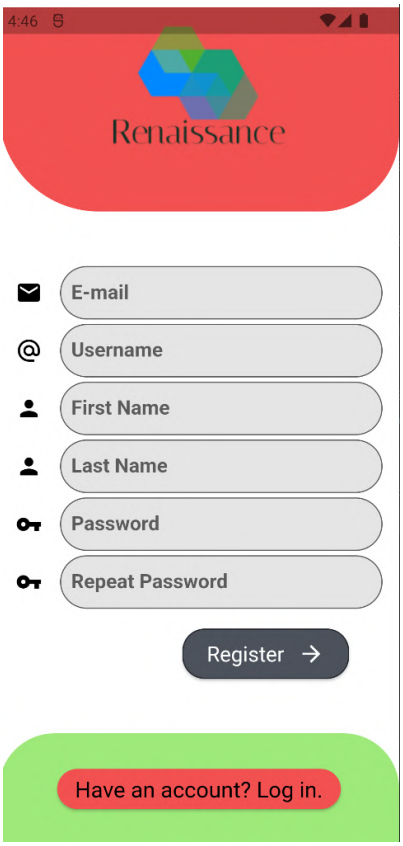


Sign Up Page

Mobile Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/register_page.dart

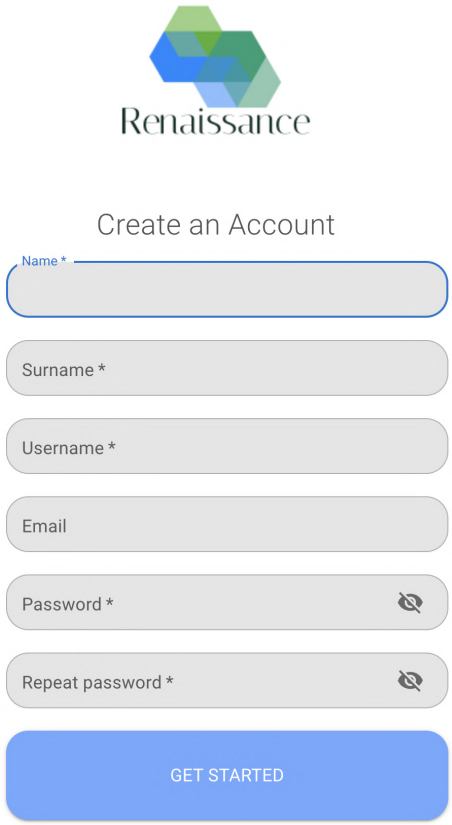
Mobile UI:



Web Code:

https://github.com/bounswe/2021SpringGroup11/tree/master/source/frontend/src/components/SignUp

Web UI:

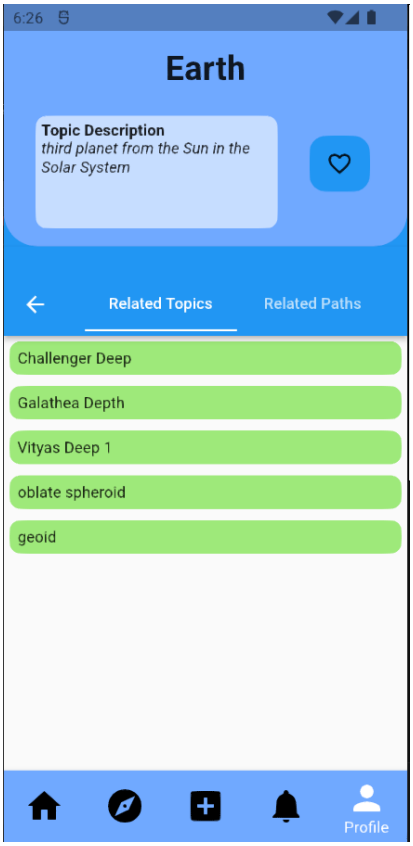


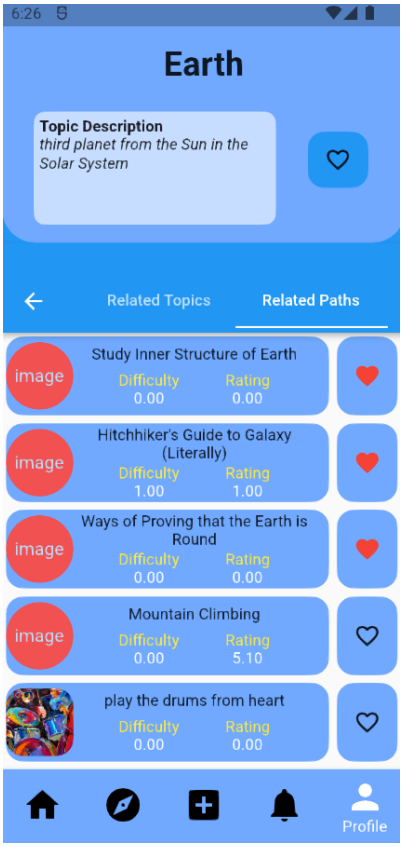
Topic Page

Mobile Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/topic_page.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/tag_container.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/course_container.dart

Mobile UI:

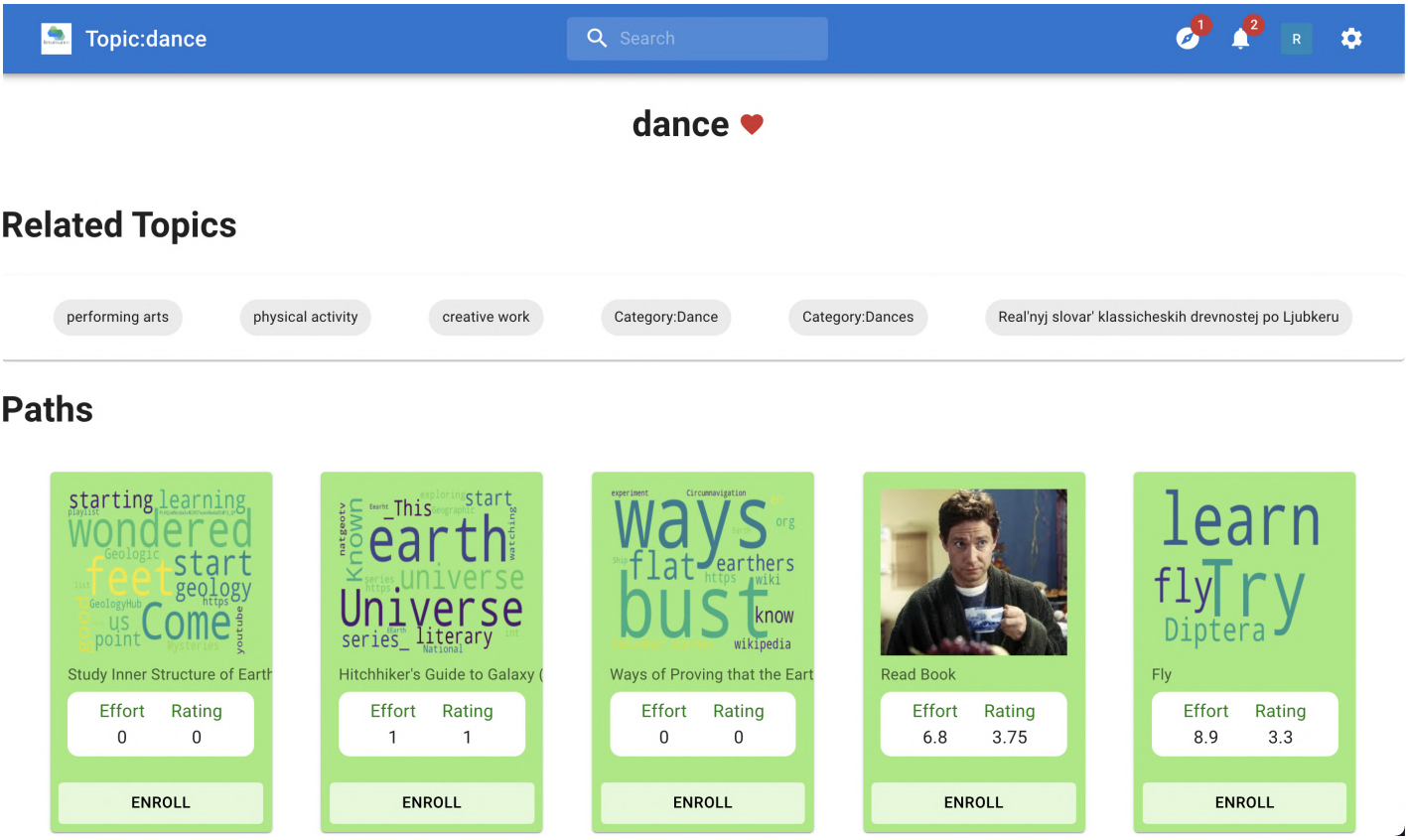




Web Code:

https://github.com/bounswe/2021SpringGroup11/tree/master/source/frontend/src/components/Topic

Web UI:

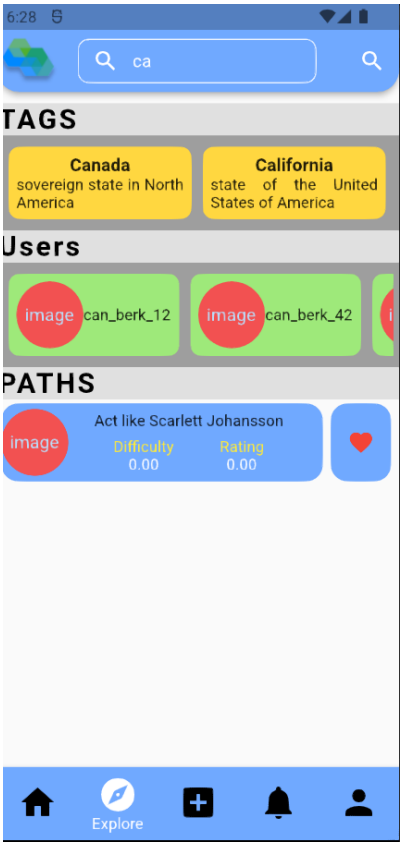


Search

Mobile Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/search_page.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/tag_search_container.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/course_container.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/user_container.dart

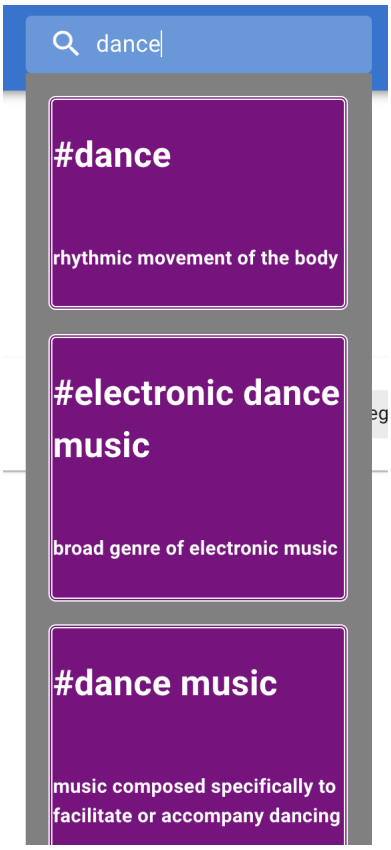
Mobile UI:



Web Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/frontend/src/components/NavBar/SearchBox.tsx
https://github.com/bounswe/2021SpringGroup11/blob/master/source/frontend/src/components/NavBar/search.util.ts

Web UI:

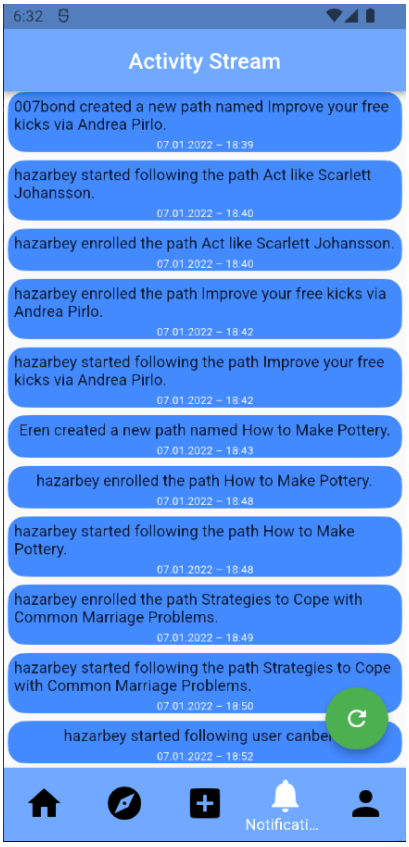


Activity Stream Page

Mobile Code:

https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/activity_stream.dart
https://github.com/bounswe/2021SpringGroup11/blob/master/source/android/Renaissance/lib/widget/activity_container.dart

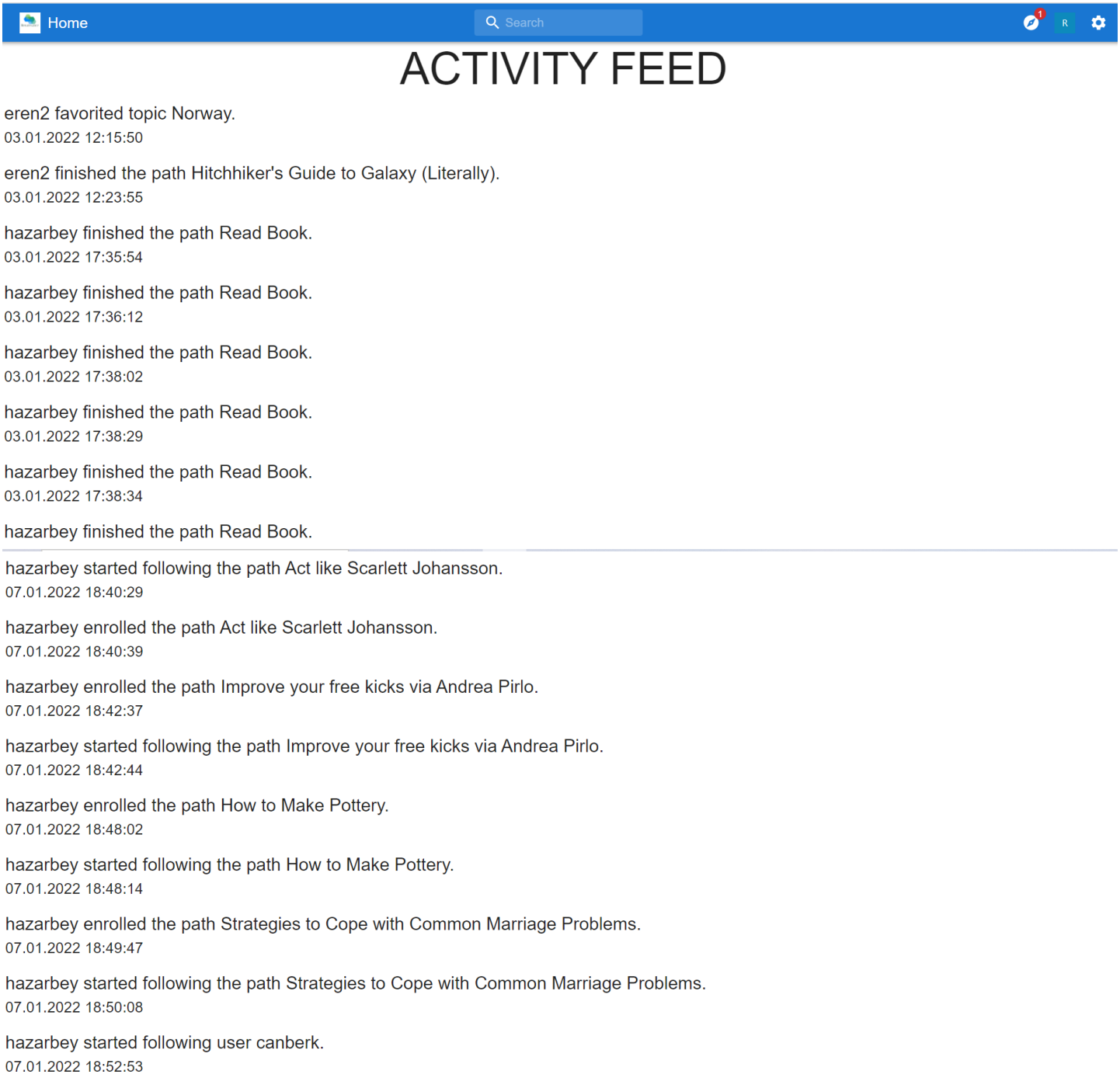
Mobile UI:



Web Code:

https://github.com/bounswe/2021SpringGroup11/tree/master/source/frontend/src/components/Activity

Web UI:

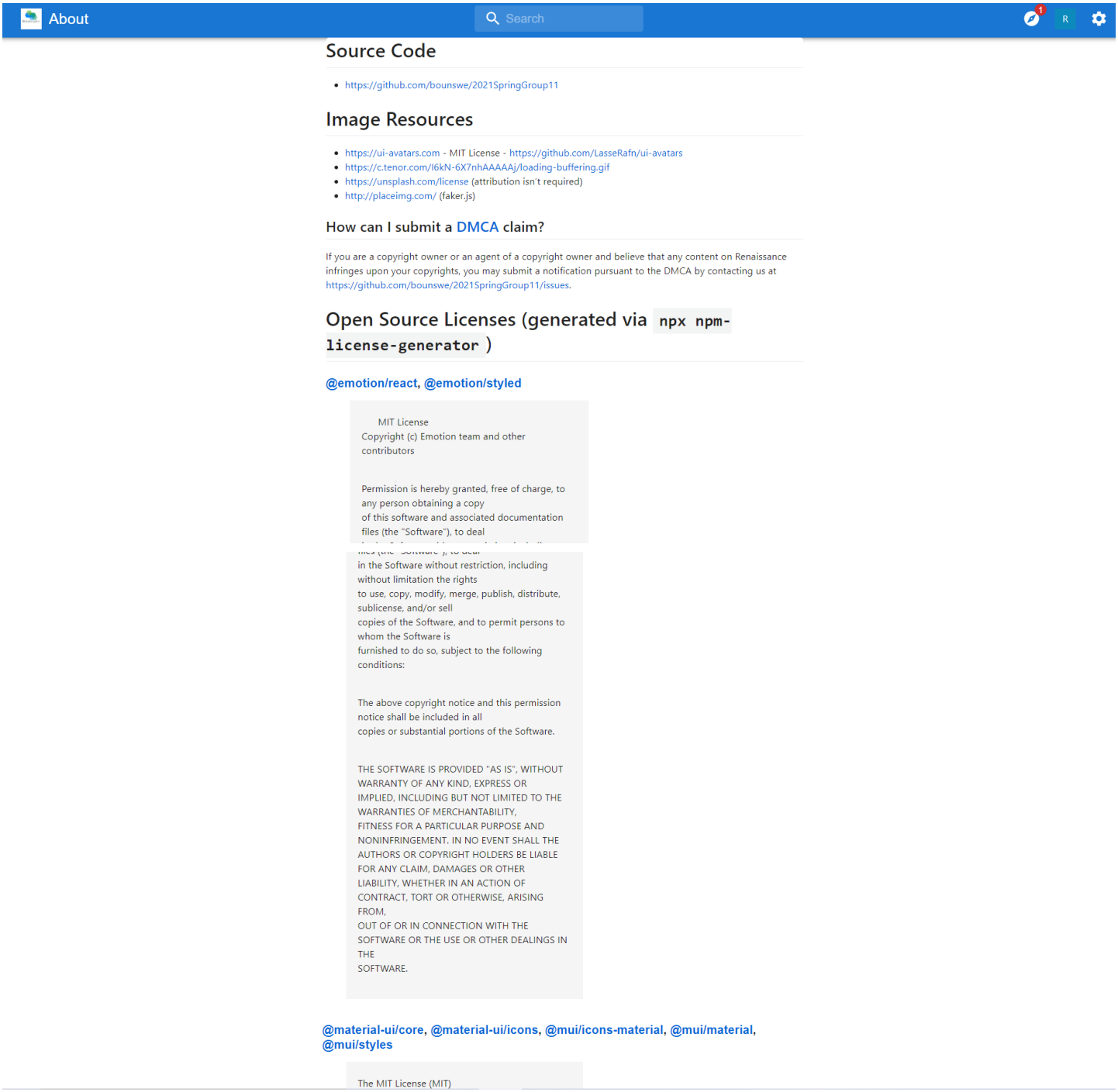


About Page

Web Code:

https://github.com/bounswe/2021SpringGroup11/tree/master/source/frontend/src/components/About

Web UI:



About

Search

Source Code

- https://github.com/bounswe/2021SpringGroup11

Image Resources

- https://ui-avatars.com - MIT License - https://github.com/LasseRafn/ui-avatars
- https://c.tenor.com/I6kN-6X7nhAAAAAj/loading-buffering.gif
- https://unsplash.com/license (attribution isn't required)
- http://placeimg.com/ (faker.js)

How can I submit a DMCA claim?

If you are a copyright owner or an agent of a copyright owner and believe that any content on Renaissance infringes upon your copyrights, you may submit a notification pursuant to the DMCA by contacting us at https://github.com/bounswe/2021SpringGroup11/issues.

Open Source Licenses (generated via `npx npm-license-generator`)

@emotion/react, @emotion/styled

MIT License
Copyright (c) Emotion team and other contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

@material-ui/core, @material-ui/icons, @mui/icons-material, @mui/material, @mui/styles

The MIT License (MIT)

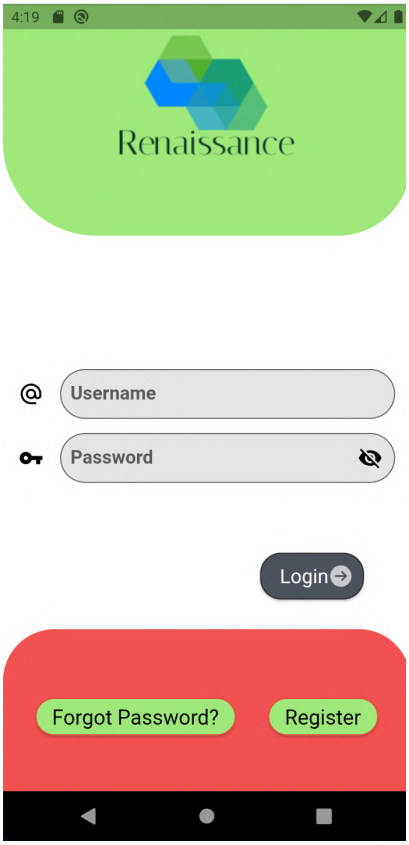
7. User Manual

User Manual - Android:

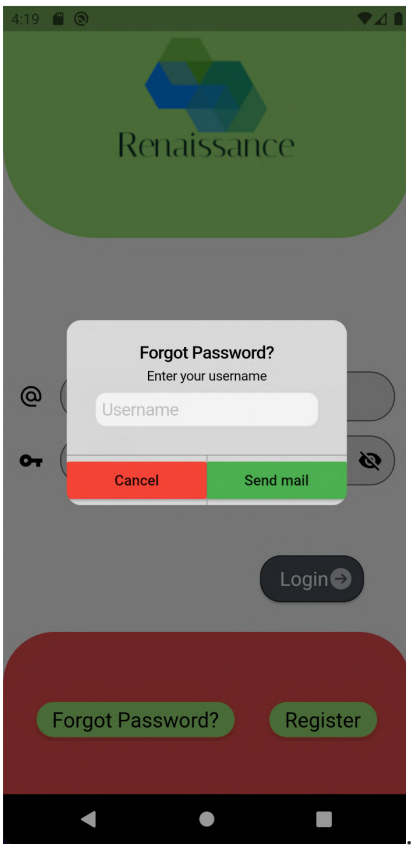
Renaissance is a social self development platform. In order to be able to use Renaissance, one should have an **Renaissance account**. After logging in, all the features will be available.

Register - Login:

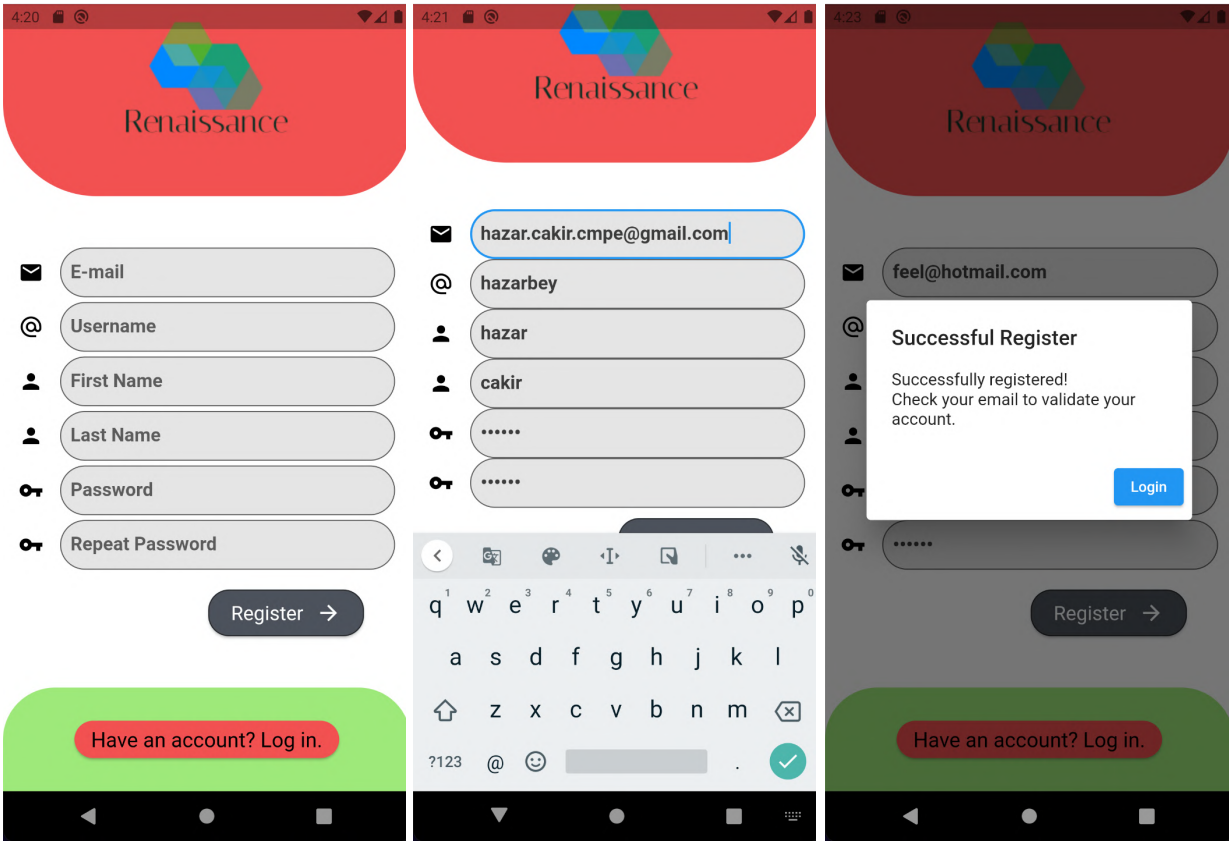
When opening the application first time, a login screen welcomes users.



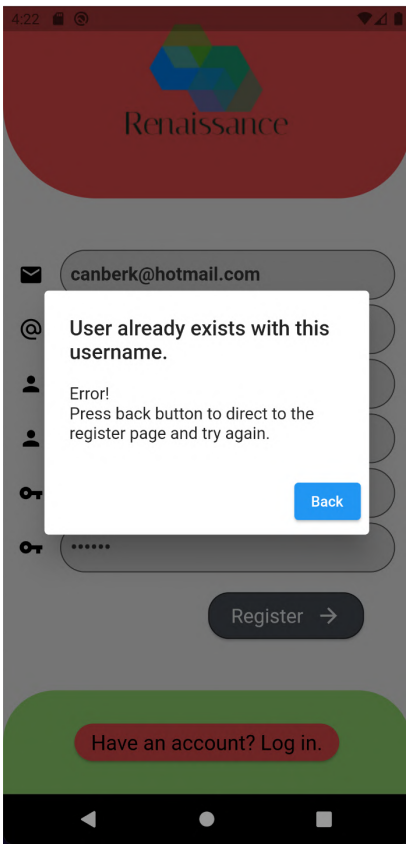
In this screen one can login with their account by filling in the corresponding credentials and tapping "Login" button. In this screen users can interact with two other buttons. While tapping "Forgot Password?", users can reset their credentials with an interface as follows:



Newcoming users should create an account via Register page:

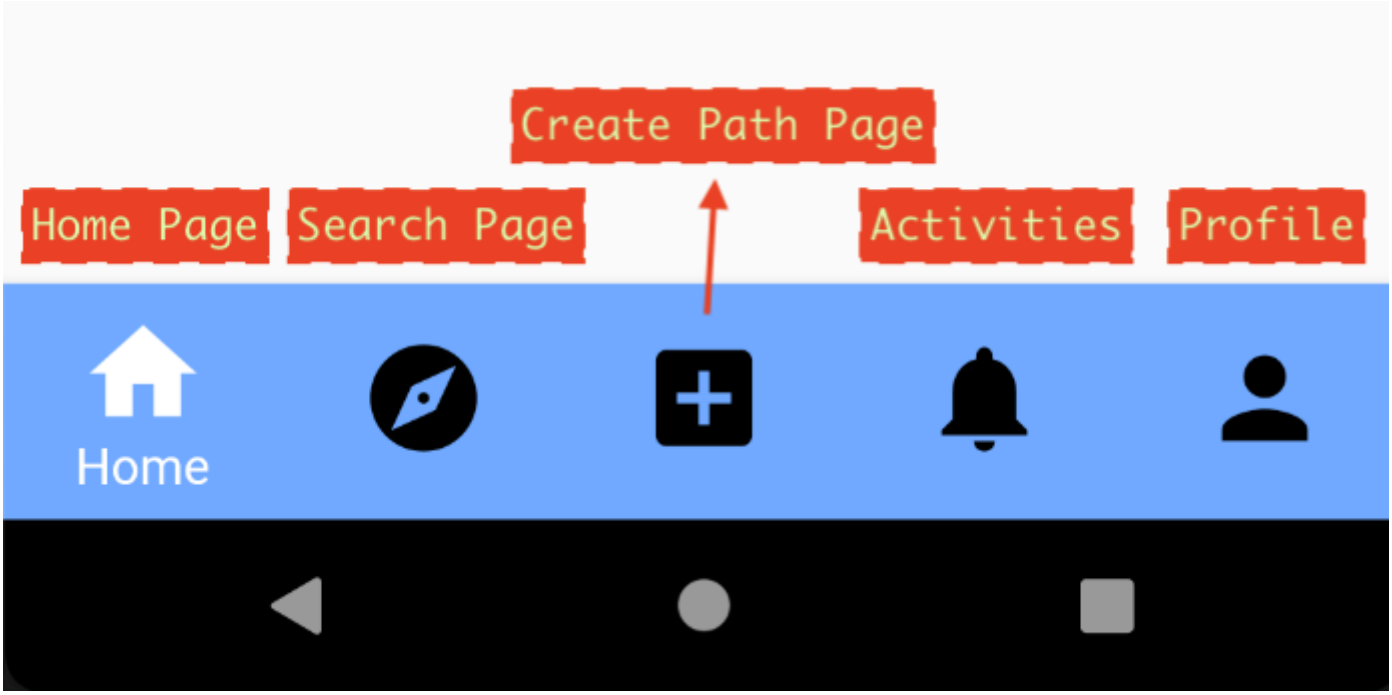


Here the required fields should be filled correctly as username, real name and surname, e-mail and passwords. After filling, user can register via Register button. In case of an error, like existing users, another pop-up notifies users as :

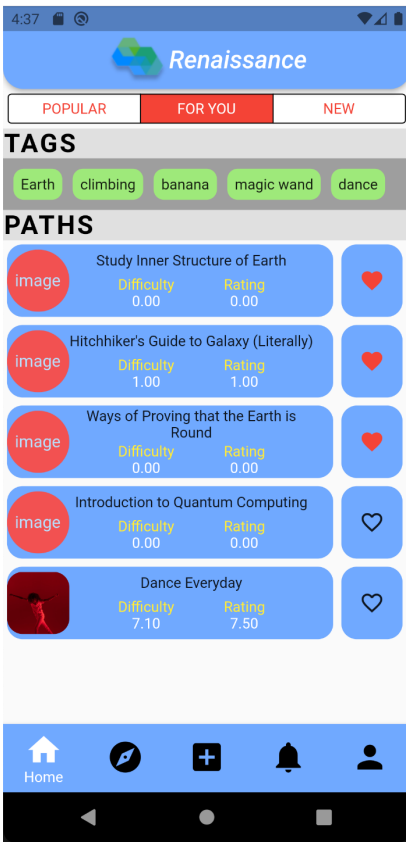


Home Page:

After logging in, the first page users see is Home Page. Before the home page, let us investigate navigation bar at the bottom of the screen.



The bar contains 5 buttons to navigate through "Home Page", "Search Page", "Create Path Page", "Activities Page" and "Profile Page" respectively. This bar is always available to guide users in the application.

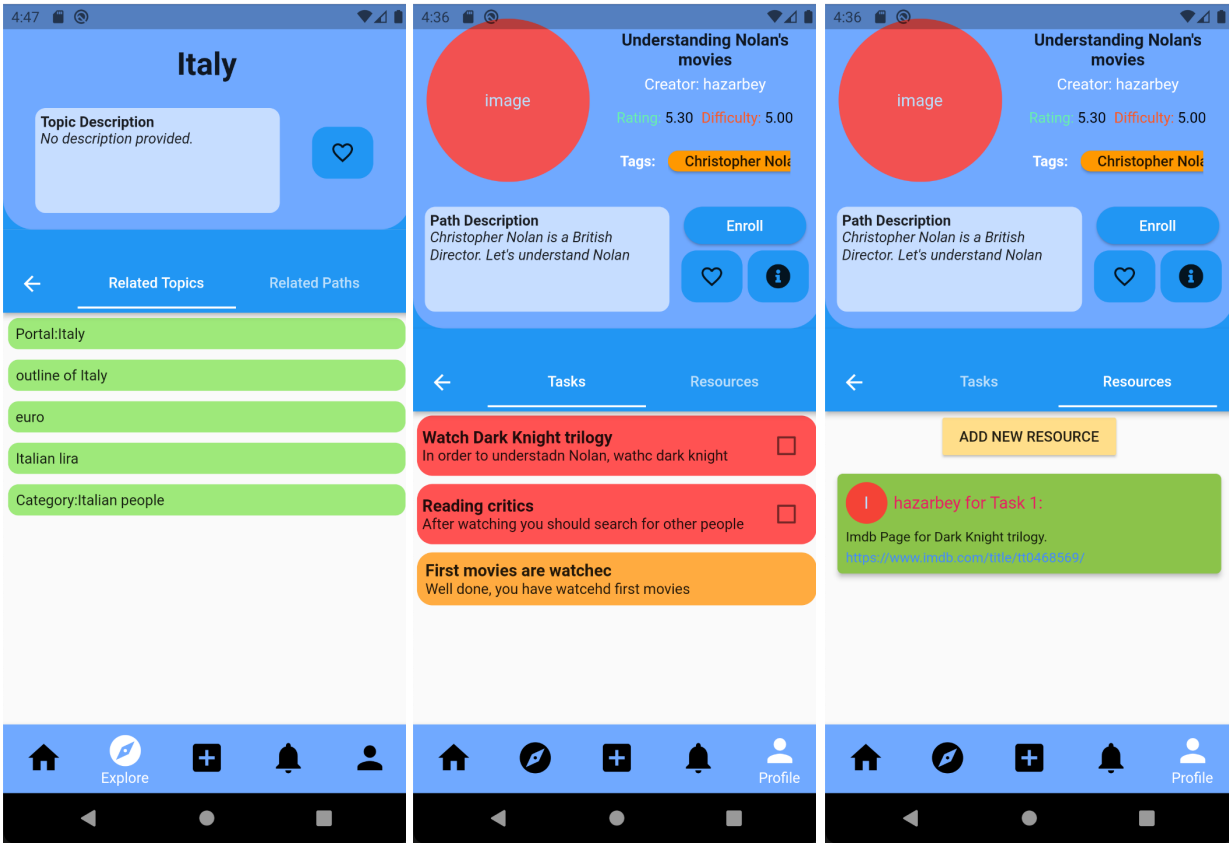


Home Page consist of Tags and Paths in three different class. Users are able to choose the general filters to explore topics and tags in three different way. "Popular" segment allows users to see popular

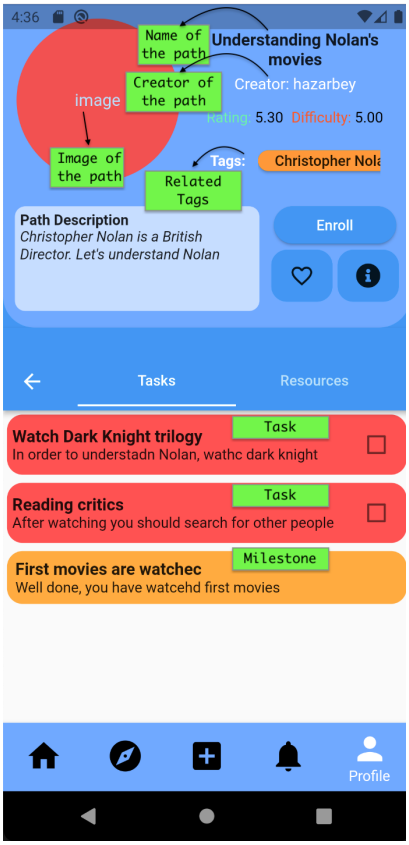
learning paths and tags in Renaissance where "For You" segment shows relevant contents. "New" segment consists of recently created paths and recently used tags.

Topic Page - Path Page:

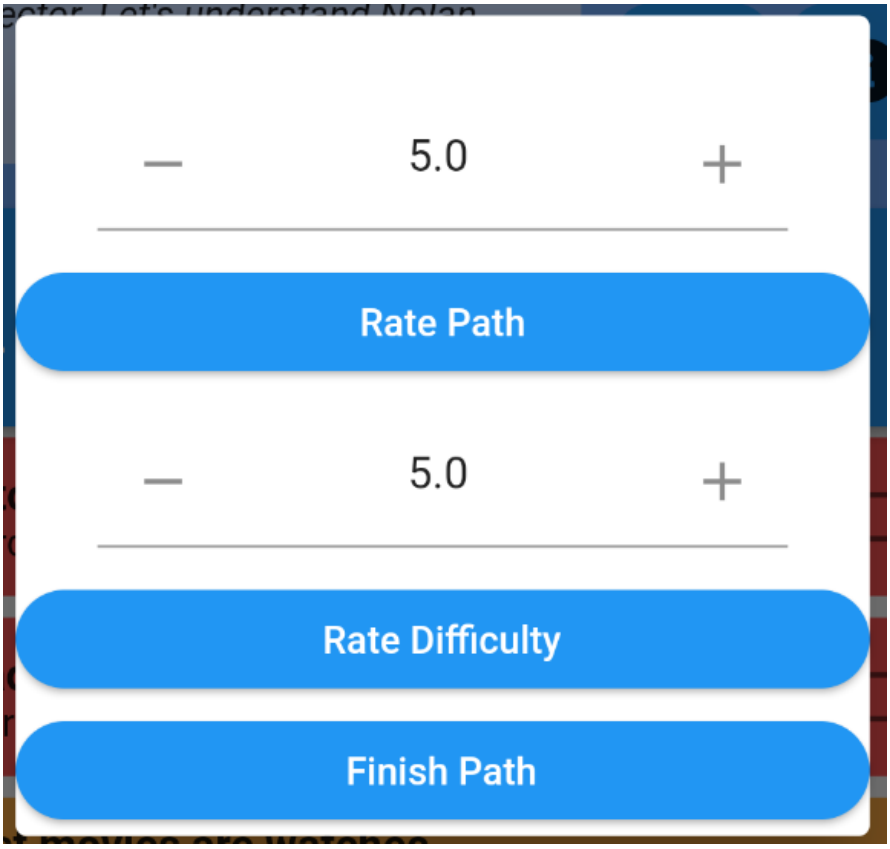
By tapping on paths and tags, users are able to navigate through corresponding Topic's Page or Learning Path's Page. Also the heart button next to each path gives the opportunity to favorite paths. Two examples of Topic page and Learning Path page is as follows.



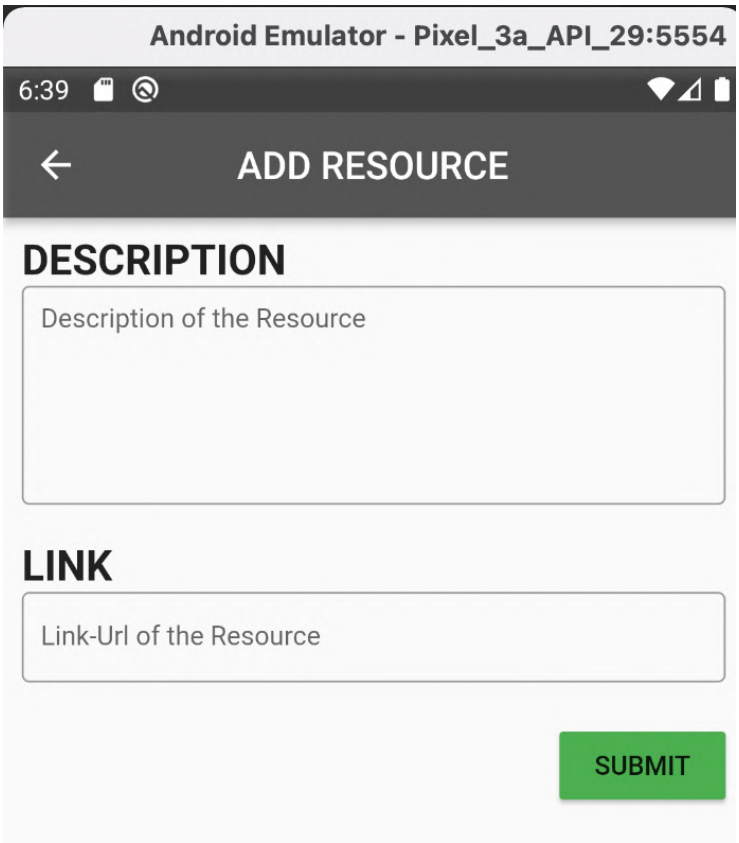
As seen, topic page consists of the name of the topic and the description of it in the header. Users can favorite topics also by tapping the heart button. In addition to those, the related topics and related paths are available to discover. Again, navigation to the paths or topics page via tapping is provided. To navigate back to the Home Page, the back button at the left end of the header should be used.



The Learning Path Page includes so many features. In the header part, the image of the path resists. On the right, one can see the name of the path with its creator. Under creator, the Rating and Difficulty values are available which gives information about the path. Under those, the related topics are located. Here again, via tapping on, one can navigate to the topic page. There is a description and next to the description, action buttons stays. Enroll button enrolls to the path and heart button favs the path. Clicking [i], another pop-up appears to give rating and difficulty values and finish the enrolled path.

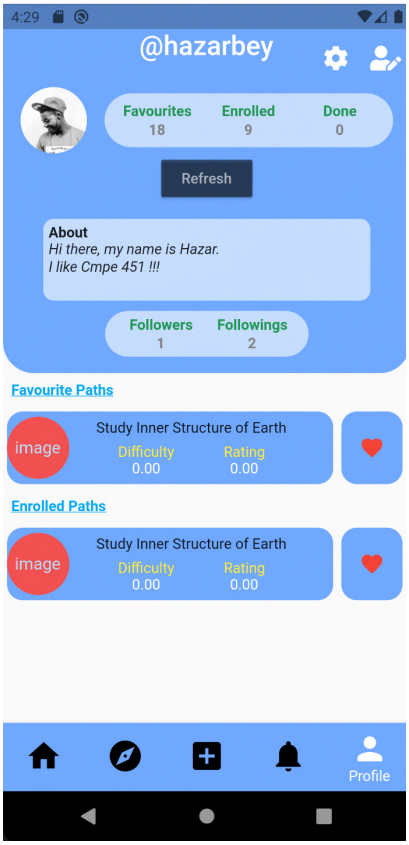


The tasks and milestones are at the bottom of the path screen. Users can mark tasks as finished. Navigation to the Resources tap, it is possible to discover related resources in a path. Also every user is able to create resources via tapping to the "Add New Resource" button. The following screen helps users to create a resource.



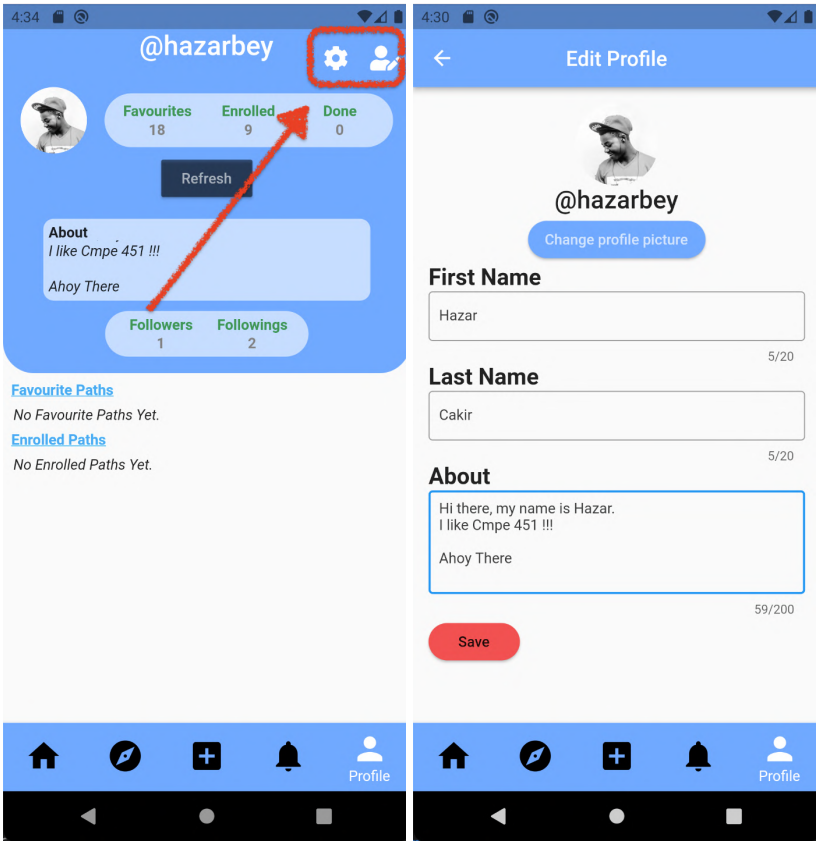
Resources are the links of videos or written resources. So one should give the description and the link to create a resource under a learning path.

Profile Page:



The fifth option on the navigation bar is profile page. Here we can see statistics about user like the number of favorites paths, enrolled paths and paths that are done next to the profile page at the top of the page. The about part follows a refresh button which refreshes the stats about users. The use of the button is not necessary if you do not get huge amount of followers in seconds but yet you can still use the button. Bottom part of the page shows some paths about the user.

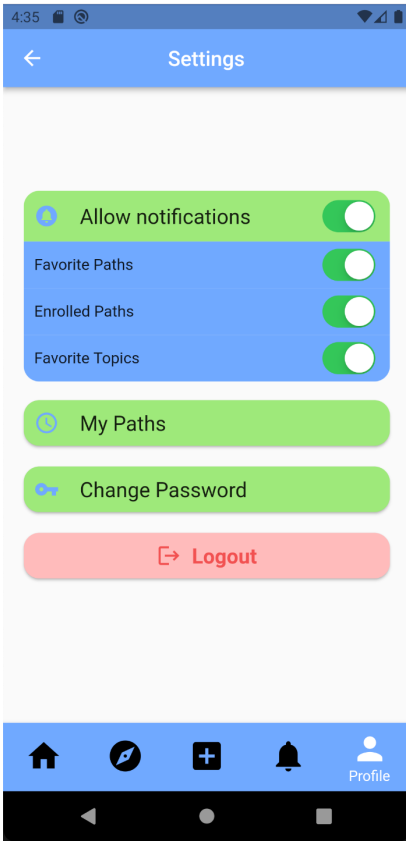
It is possible to update profile page by tapping the button at the top right corner of the page. This redirects to a screen where the information about user can be updated as :



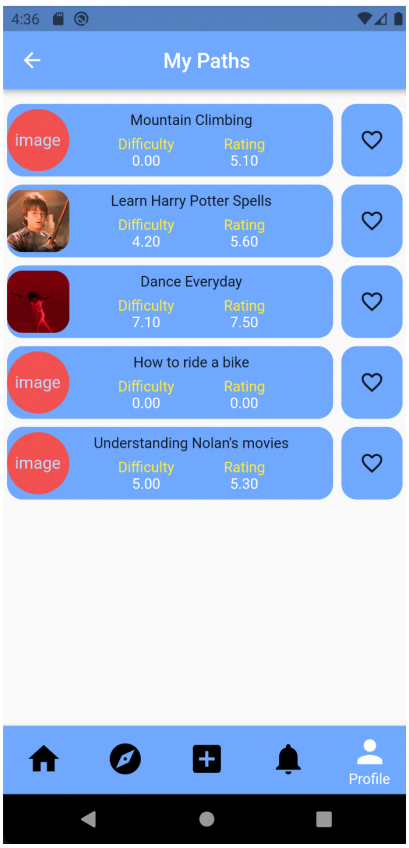
Here the profile picture, first and last name and the about section can be updated via tapping to the Save button.

The other important feature that is reachable from the user page is the settings page. The Settings page button is again at the top right corner of the screen. Via tapping to the button, the settings will be showed as:

Settings

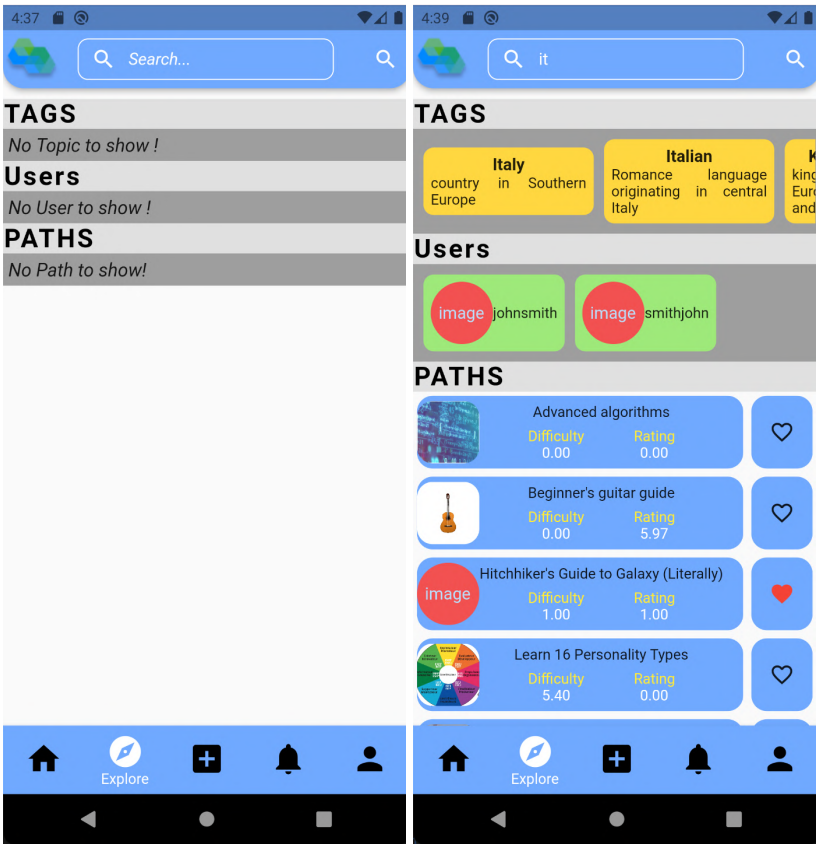


Here one can change its notification settings and change the password of their account. Also the logout functionality lies here. But more importantly, "My Paths" button makes it possible to see all the paths that user created. The resulting screen is as follows:

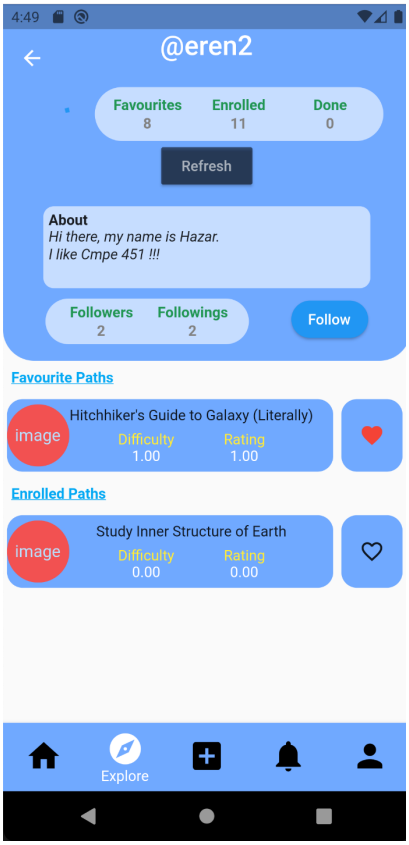


Again, navigating to the path page is possible by tapping on it.

Search Page:



The second available page on the navigation button is search page. Here users can search tags, the users and paths using the search bar at the top of the screen. After writing and tapping the button, the results are loaded one by one. At the top part, one can see Topics includes search term. Under Topics, the other users in Renaissance is listed. At the bottom part, there are all paths includes keyword. Navigation to the topic page, other users page and path is again possible via tapping on them. The Topic page and Path page is already showed. Here is the other users profile page when tapping on:



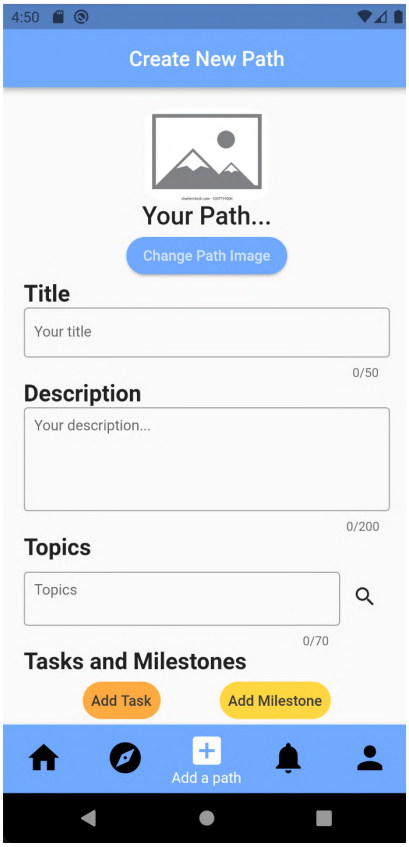
As you can see, configure profile page and setting page features are not shown to the user.

Activity Stream

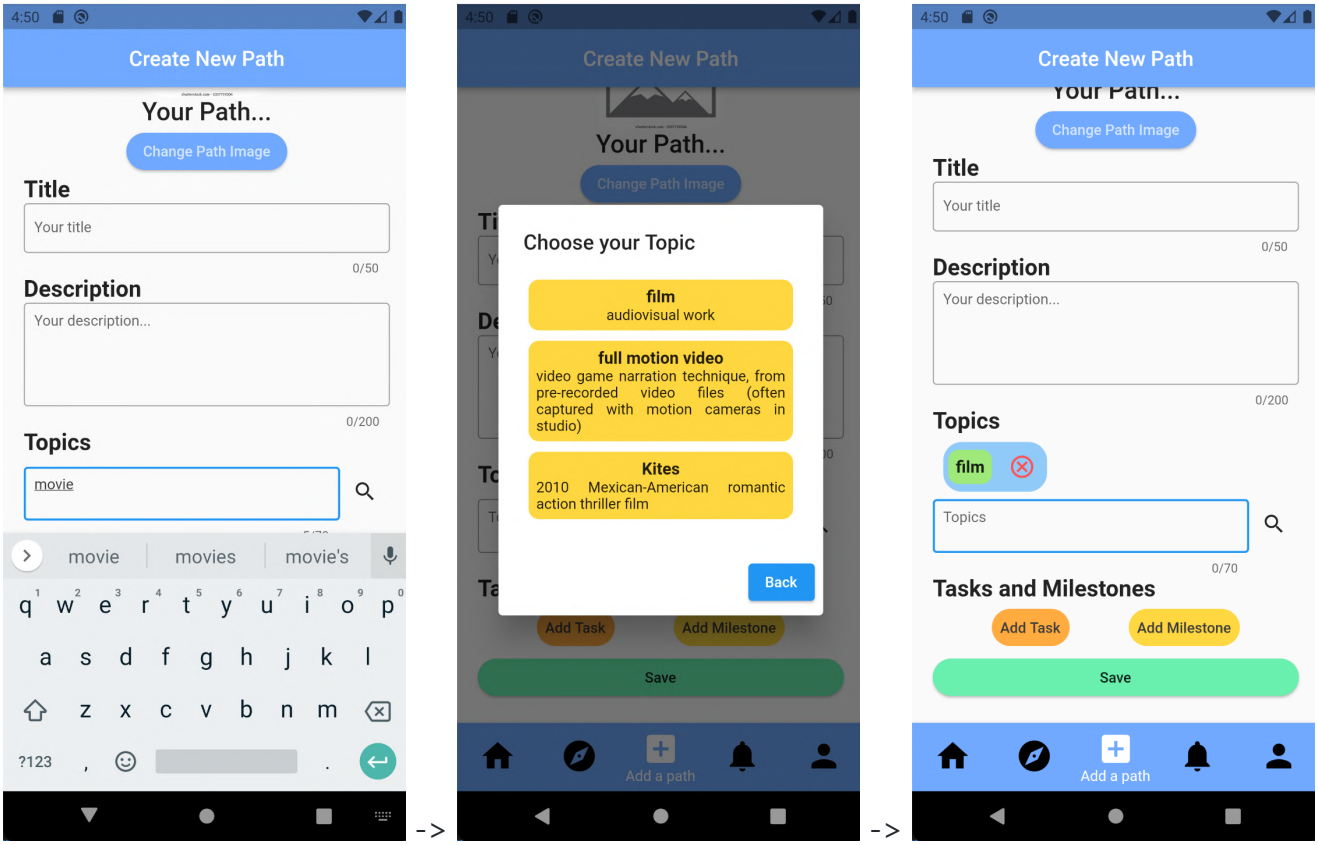
This is the fourth section on the navigation bar and allows user to see the activities related to the user. The green refresh button refreshes the activities seen on the screen and adds if there is any

addition.

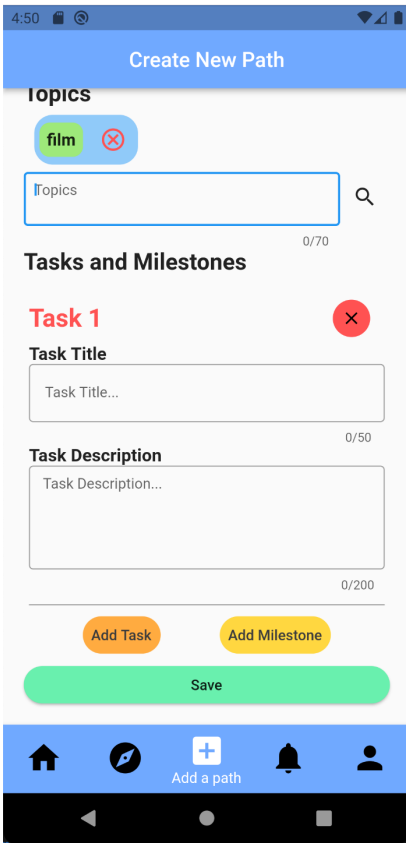
Create Path Page



The middle button at the navigation bar is for creating a new Learning Path. In this page, users are able to give a title and description to the path that will be created as well as adding a path Image. The addition of topic part is a little bit different as, users should insert the topics related to their learning path one at a time and tap search button next to, to choose specific topic really related to their path. Users choose the topic via tapping on it. The process of topic addition is as follows :



After topic part, users are able to add tasks and milestones as much as they wanted. The buttons identifies the functionalities of them. The important thing is the order of creation.



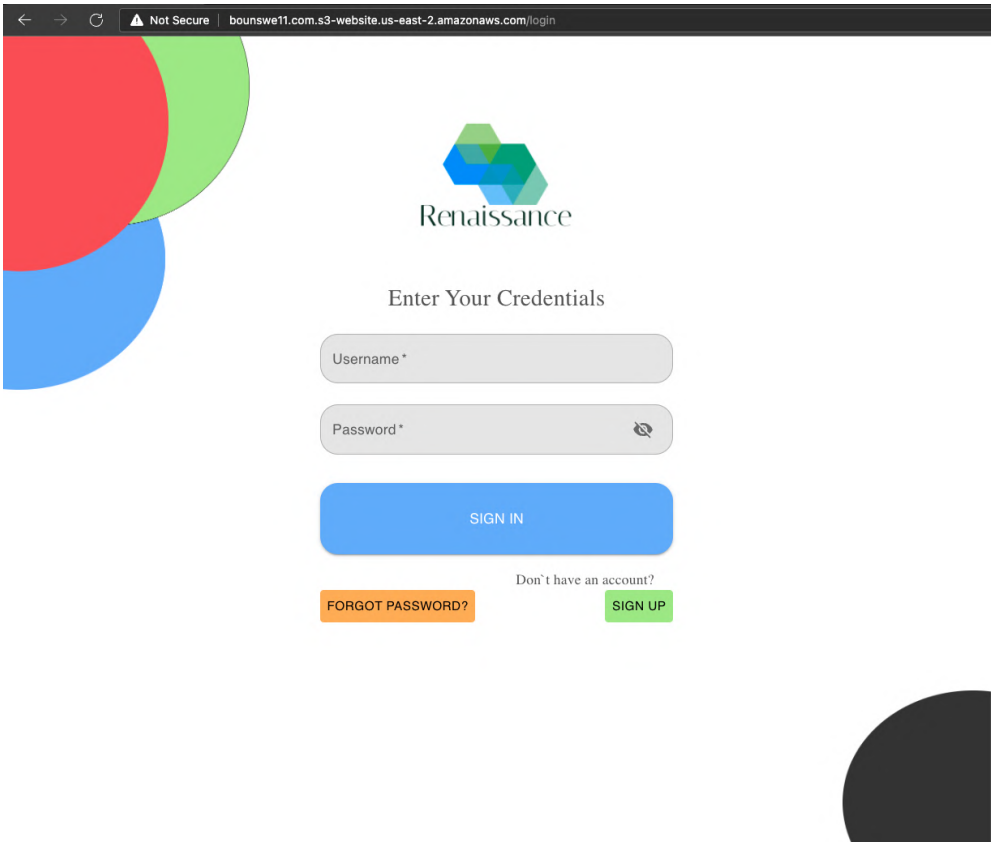
After tapping buttons, the title and description input parts are created. User can input the text fields however they want. After creating and filling all the information, one should tap "Save" button to create Learning path.

User Manual - Web:

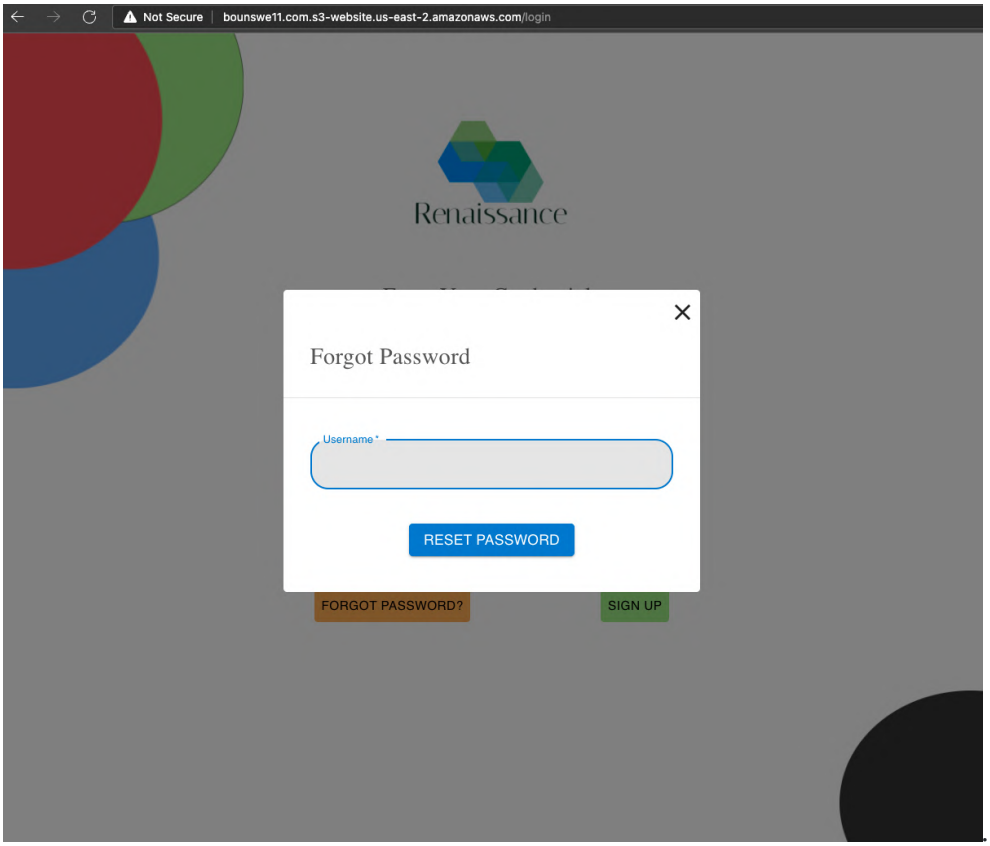
Renaissance is a social self development platform. In order to be able to use Renaissance, one should have an **Renaissance account**. After logging in, all the features will be available.

Register - Login:

When opening the application first time, a login screen welcomes users.



In this screen one can login in its account while filling in the corresponding credentials and tapping "Sign In" button. In this screen users are able to interact with two other buttons. While tapping "Forgot Password?", users can reset their credentials with an interface as follows:



The very first time users should create an account via Register page:

Renaissance

Create an Account

Name *

Surname *

Username *

Email

Password *

Repeat password *

GET STARTED

Renaissance

Create an Account

Name *
Alice

Surname *
Bob

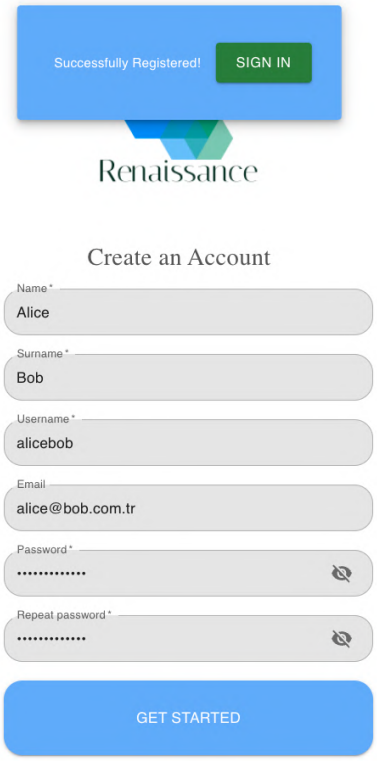
Username *
alice

Email
alice@bob.com

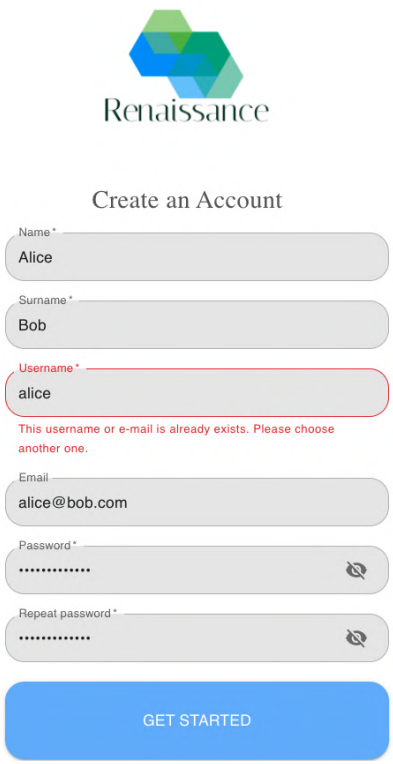
Password *
••••••••••••

Repeat password *
••••••••••••

GET STARTED

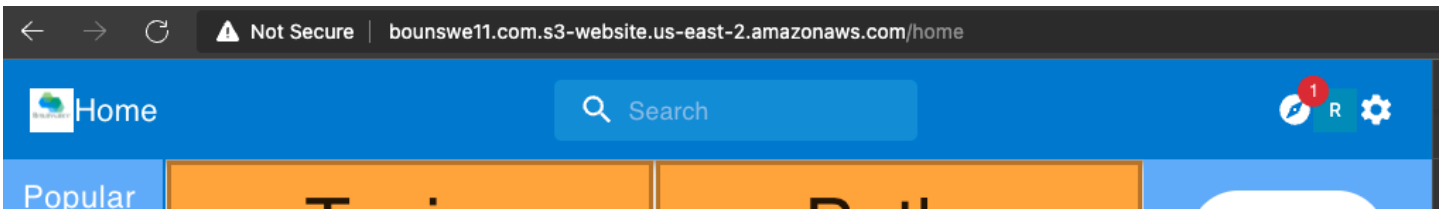


Here the required fields should be filled correctly as username, real name and surname, e-mail and passwords. After filling, user can register via Register button. In case of an error, like existing users, the app notifies users as :

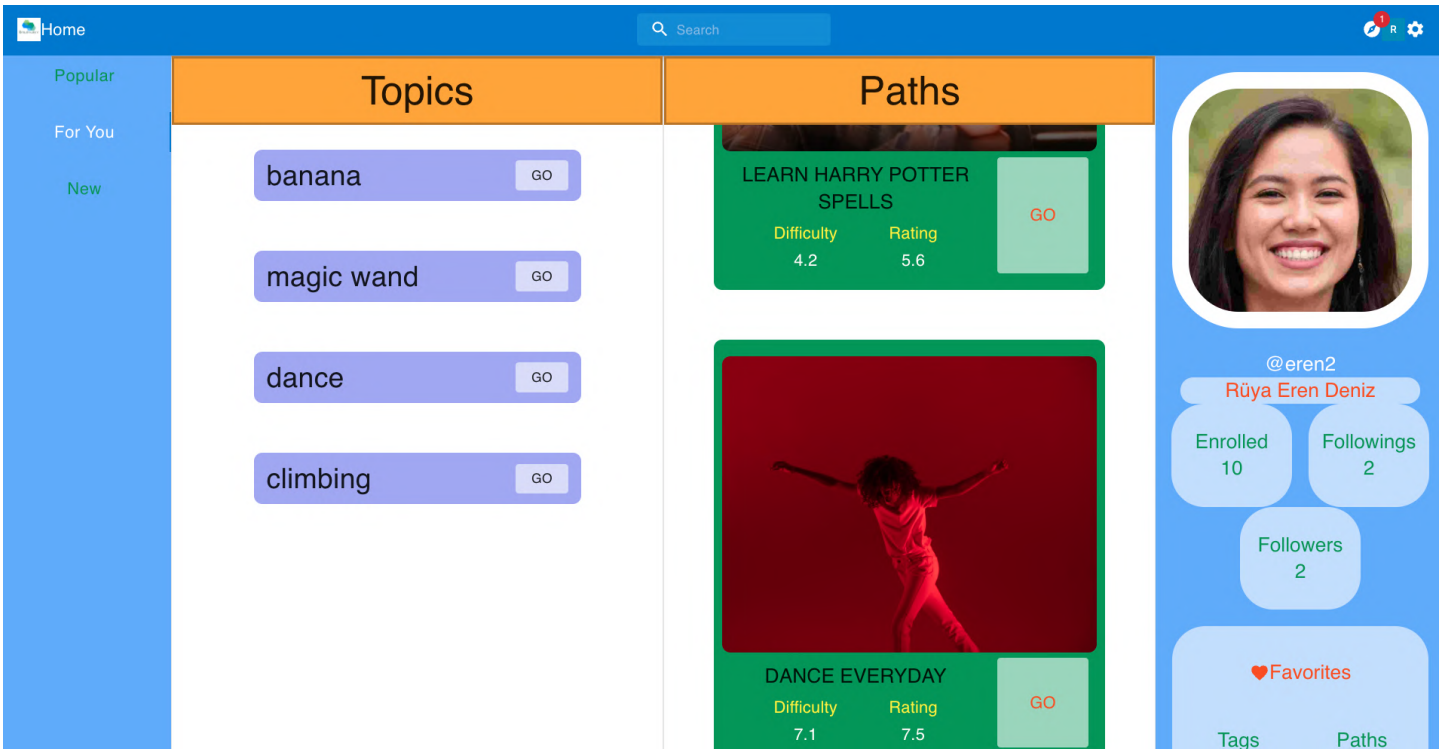


Home Page:

After logging in, the first page users see is Home Page. Before the home page, let us investigate navigation bar at the bottom of the screen.



The bar contains 5 buttons to navigate through "Home Page", "SearchBox", "Activities Page" and "Profile Page", "About" respectively. This bar is always available to guide users in the application.



Home Page consist of Tags and Paths in three different class. Users are able to choose the general filters to explore topics and tags in three different way. "Popular" segment allows users to see popular learning paths and tags in Renaissance where "For You" segment shows relevant contents. "New" segment consists of recently created paths and recently used tags.

Topic Page - Path Page:

Via tapping on paths and tags, users are able to navigate through corresponding Topic's Page or Learning Path's Page. Also the hearth button next to each path gives the opportunity to favorite paths. Two examples of Topic page and Learning Path page is as follows.

Topic:dance

Search

dance ♥

Related Topics

performing arts | physical activity | creative work | Category:Dance | Category:Dances

Paths

Study Inner Structure of Earth	Hitchhiker's Guide to Galaxy	Ways of Proving that the Earth	Mountain Climbing	play the drums from heart	Learn Harry Potter Spells	Dance Everyday	type very fast on keyboard
Effort 0	Effort 1	Effort 0	Effort 0	Effort 0	Effort 4.2	Effort 7.1	Effort 0
Rating 0	Rating 1	Rating 0	Rating 5.1	Rating 0	Rating 5.6	Rating 7.5	Rating 0
UNENROLL	ENROLL	ENROLL	ENROLL	ENROLL	ENROLL	UNENROLL	ENROLL

Path

Search

Understanding Nolan's movies

Creator: hazarbey

UNENROLL | ADD TO FAVORITE

Path Description

Christopher Nolan is a British Director. Let's understand Nolan

Tags

Christopher Nolan | film

Rating: 5.3

Difficulty: 5

MILESTONES AND TASKS

- TASK: WATCH DARK KNIGHT TRILOGY ☐
- TASK: READING CRITICS ☐
  After watching you should search for other people
- MILESTONE: FIRST MOVIES ARE WATCHEC

RESOURCES

Imdb Page for Dark Knight trilogy.

https://www.imdb.com/title/tt0468569/

RESOURCES

Imdb Page for Dark Knight trilogy.

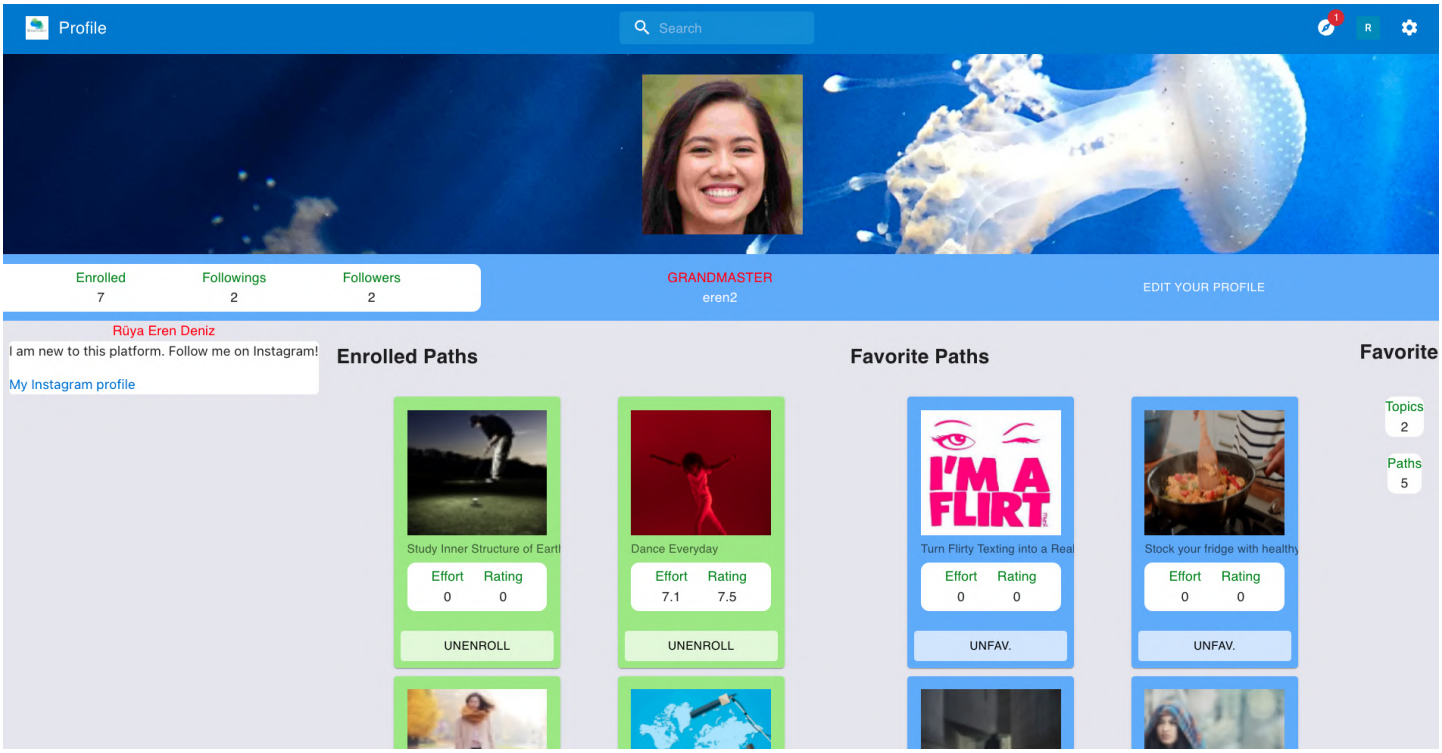
https://www.imdb.com/title/tt0468569/

As seen, topic page is consist of the name of the topic and the description of it in the header. Users can fav topics also via tapping the hearth button. In. addition to those, the related topics and related paths are available to discover. Again, navigation to the paths or topics page via tapping is provided. To navigate back to the Home Page, the home button at the left end of the header should be used.

The Learning Path Page includes so many features. In the header part, the image of the path is resists. One can see the name of the path with its creator. On the right, the Rating and Difficulty values are available which gives information about the path. Under those, the related topics are located. Here again, via tapping on, one can navigate to the topic page. There is a description and next to the description, action buttons stays. Enroll button enrolls to the path and heart button favs the path. The tasks and milestones are at the bottom of the path screen. Users can mark tasks finished. Navigation to the Resources Tab, it is possible to discover related resources in a path.

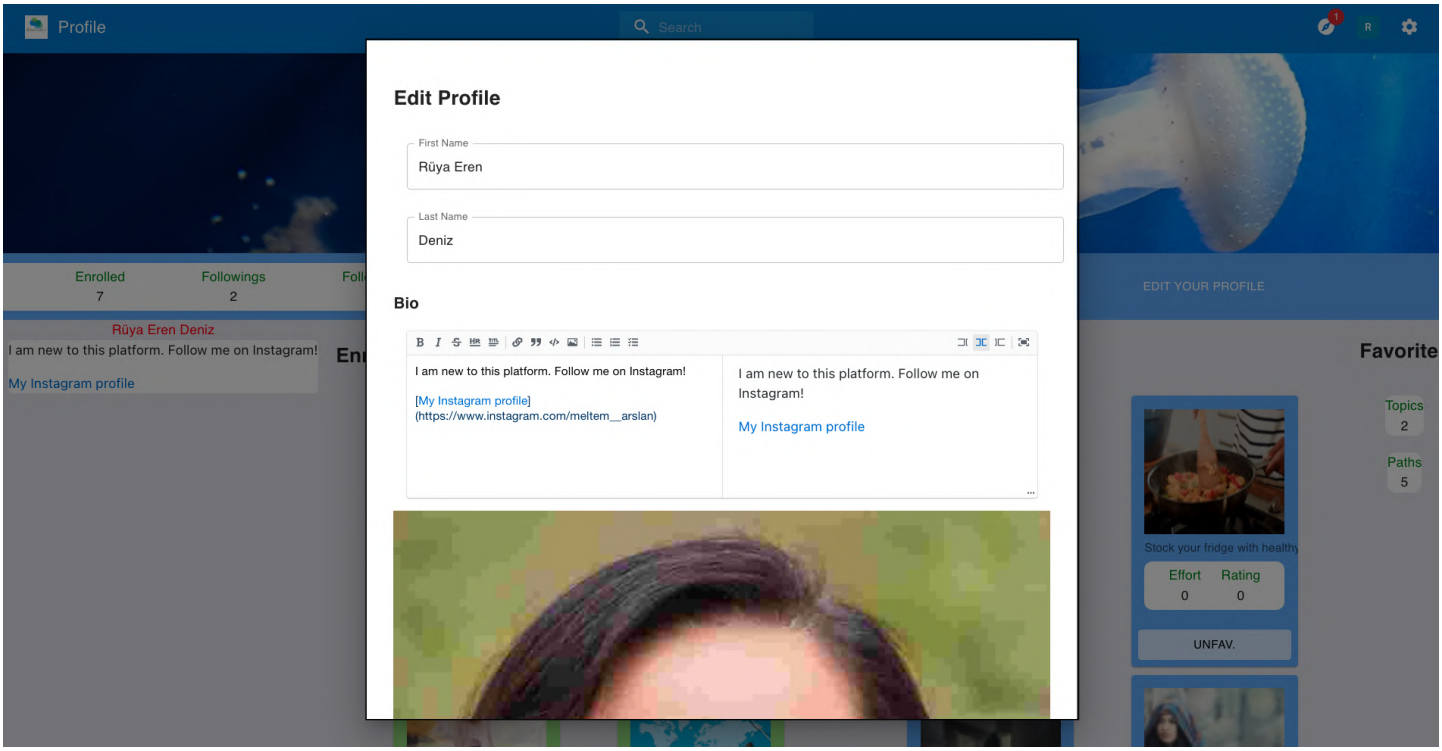
Resources are the links of videos or written sources. So one should give the description and the link of the resource to create a resource under a learning path.

Profile Page:

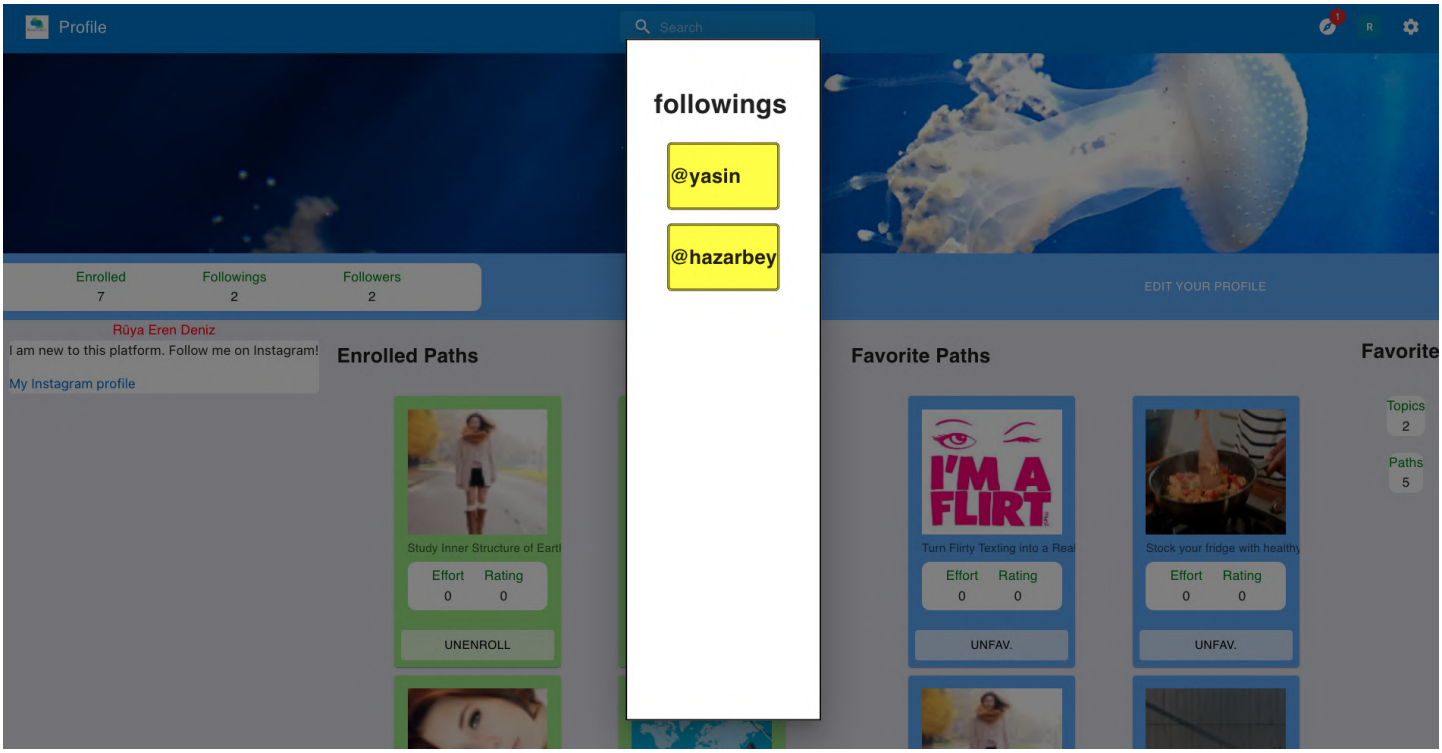


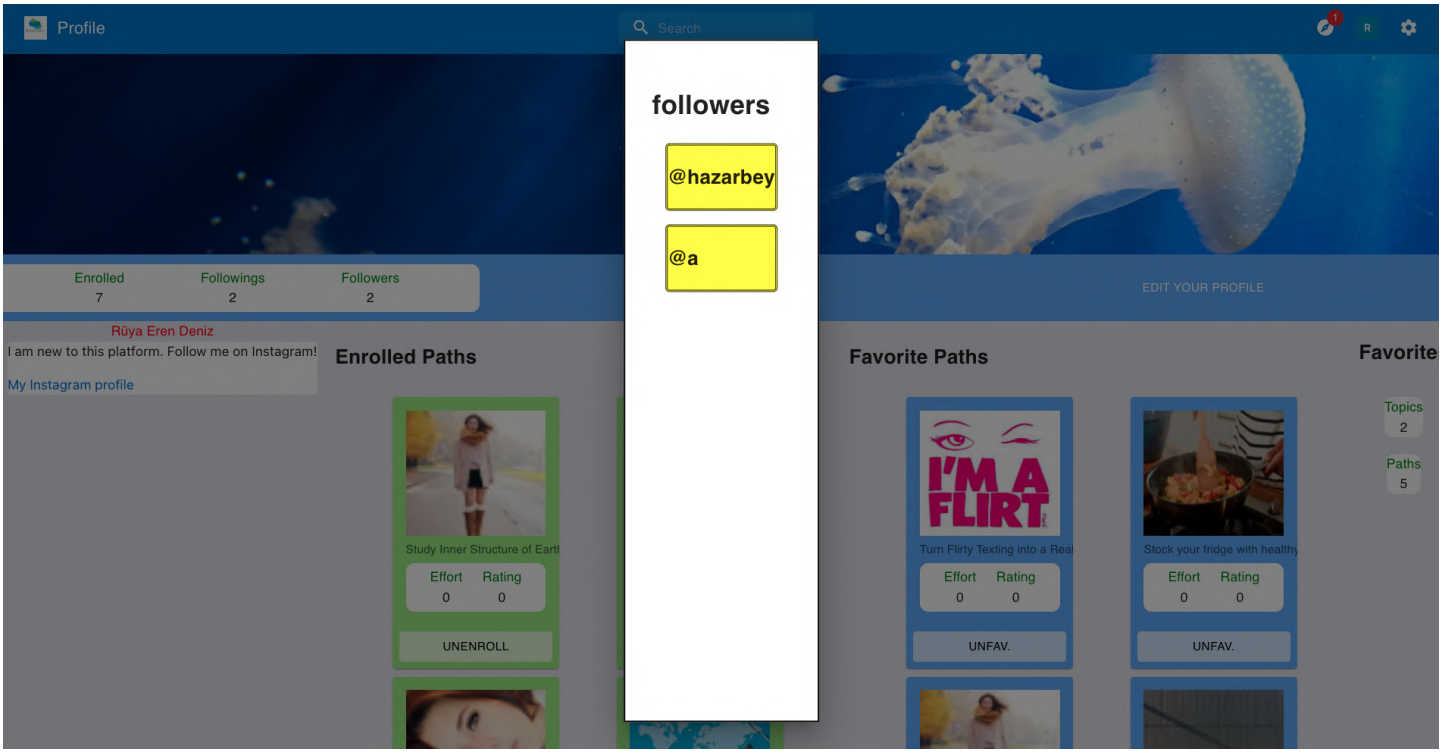
The fourth option on the navigation bar is profile page. Here we can see some statistic about user like the number of favorites paths, enrolled paths and paths that are done next to the profile page at the top of the page. The use of the button is not necessary if you do not get huge amount of followers in seconds but yet you can still use the button. Bottom part of the page shows some paths about the user.

It is possible to update profile page via clicking to the "Edit Profile" button. This opens popup one to a screen where the information about user can be updated as :



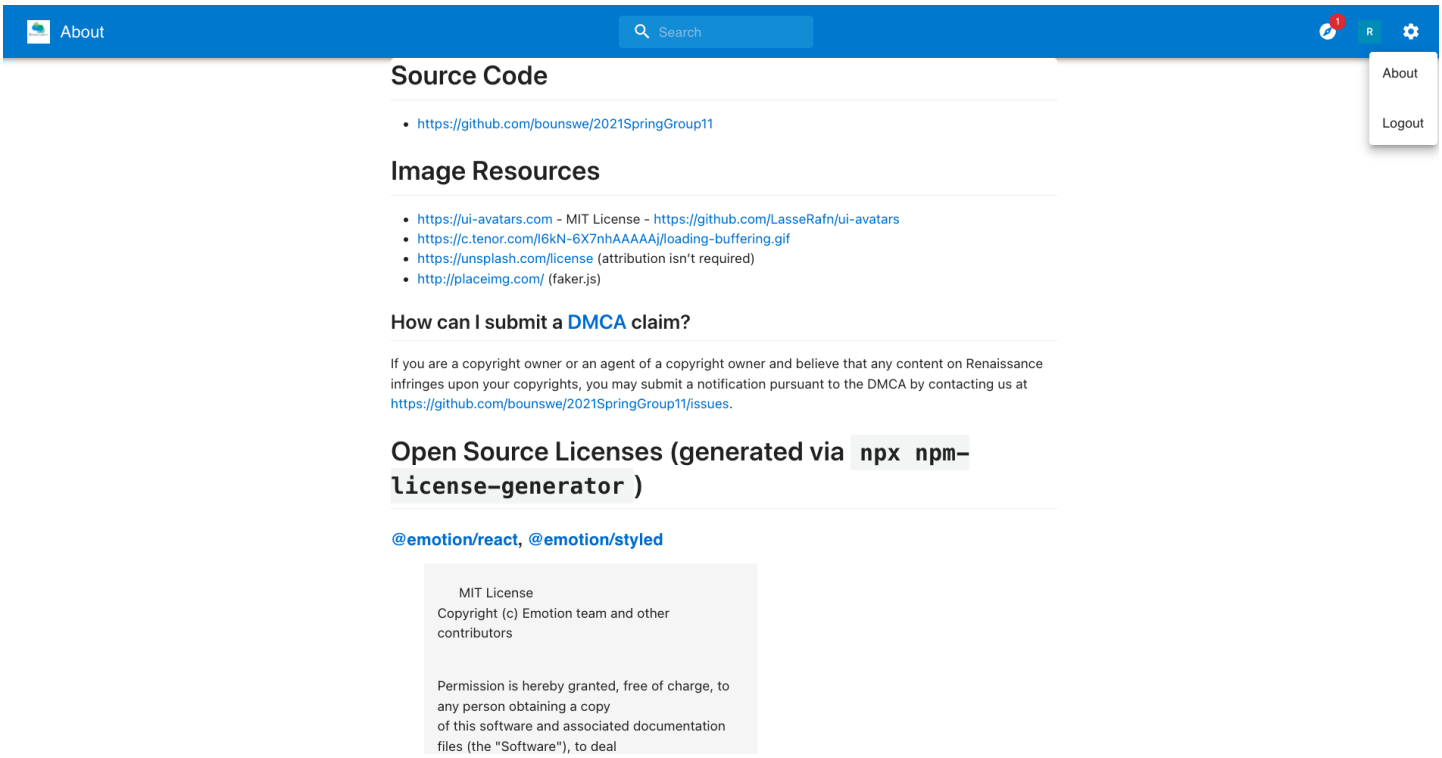
Here the profile picture, first and last name and the about section can be updated via tapping to the Save button. Also, user can list their followers and followings via clicking counters.





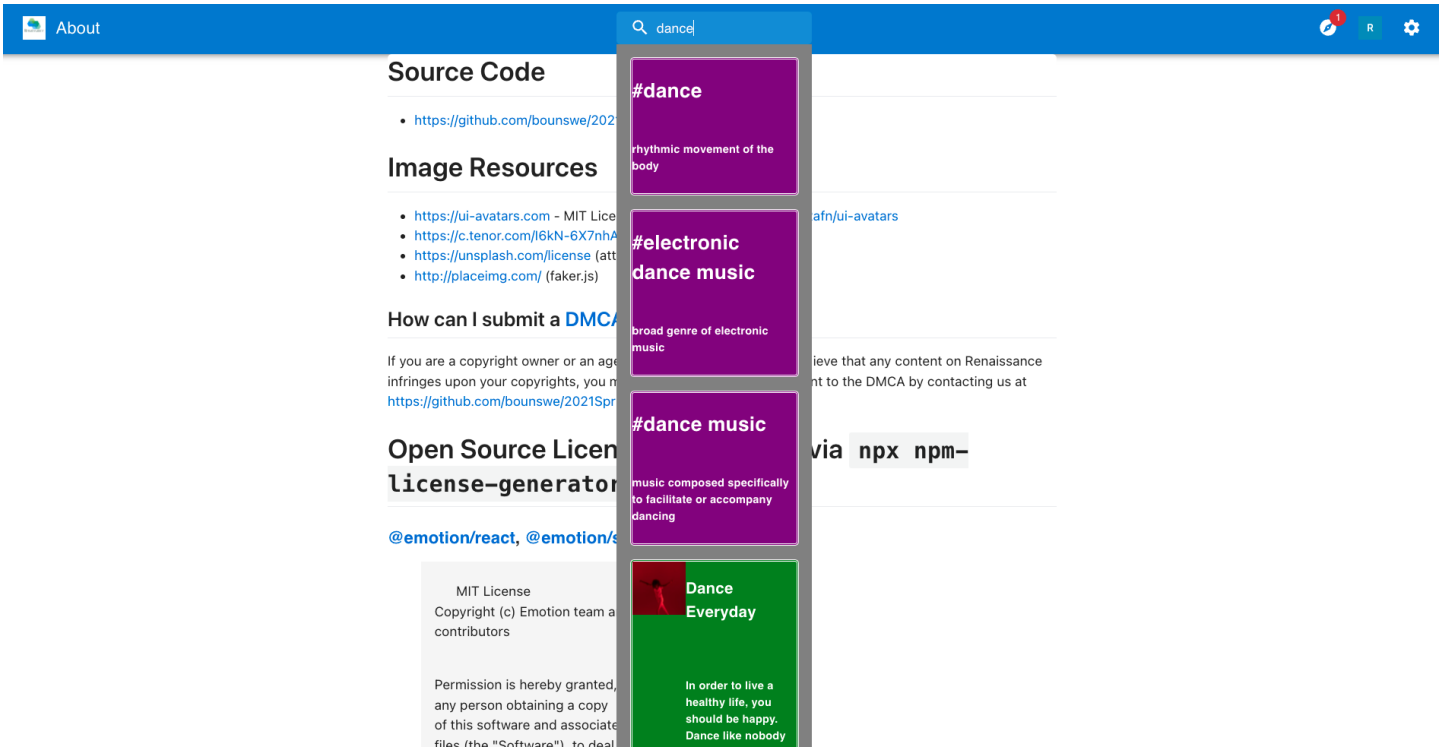
The other important feature that is reachable from the user page is the about page. The about page button is again at the top right corner of the screen via clicking more button. Via tapping to the button, the settings will be showed as:

About Page

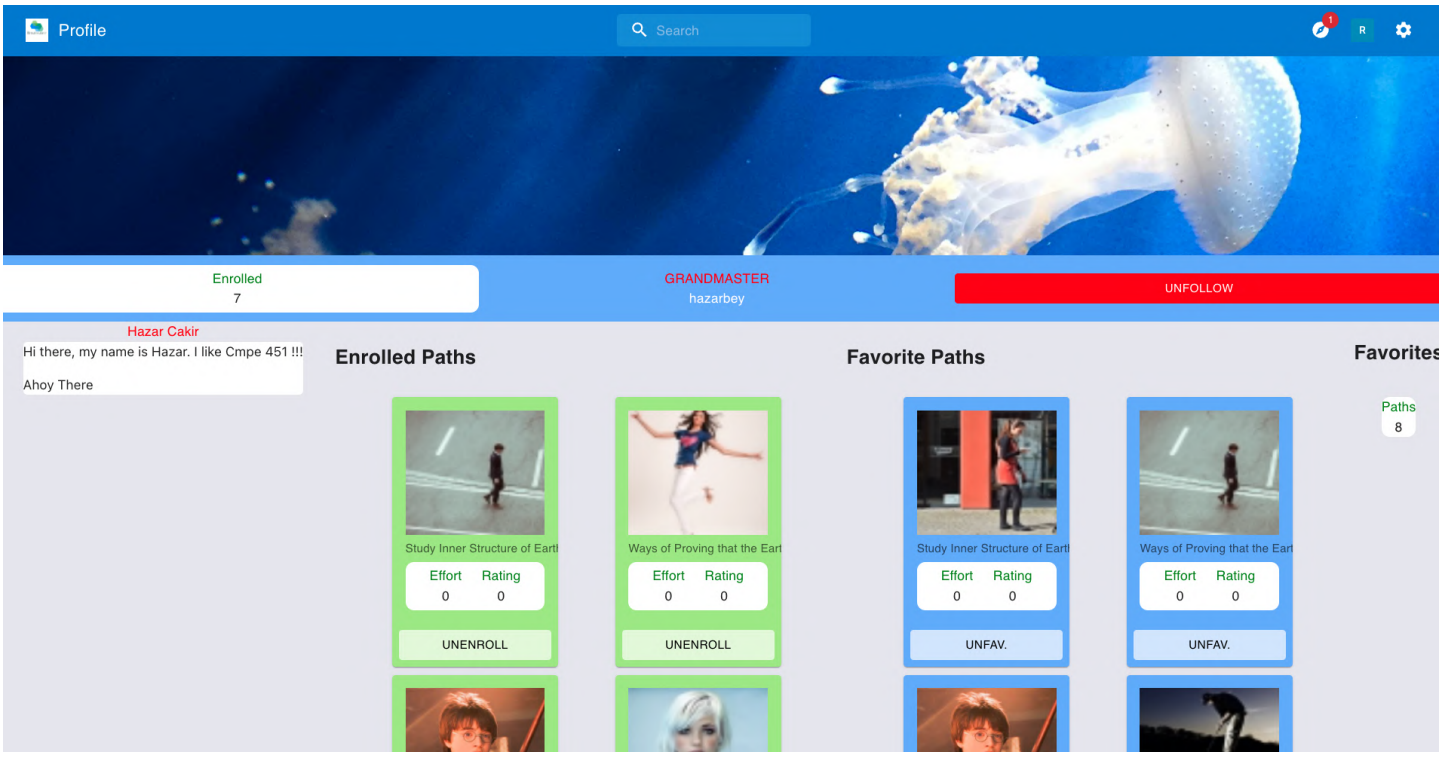


Source code link and DMCA and resource licenses are listed.

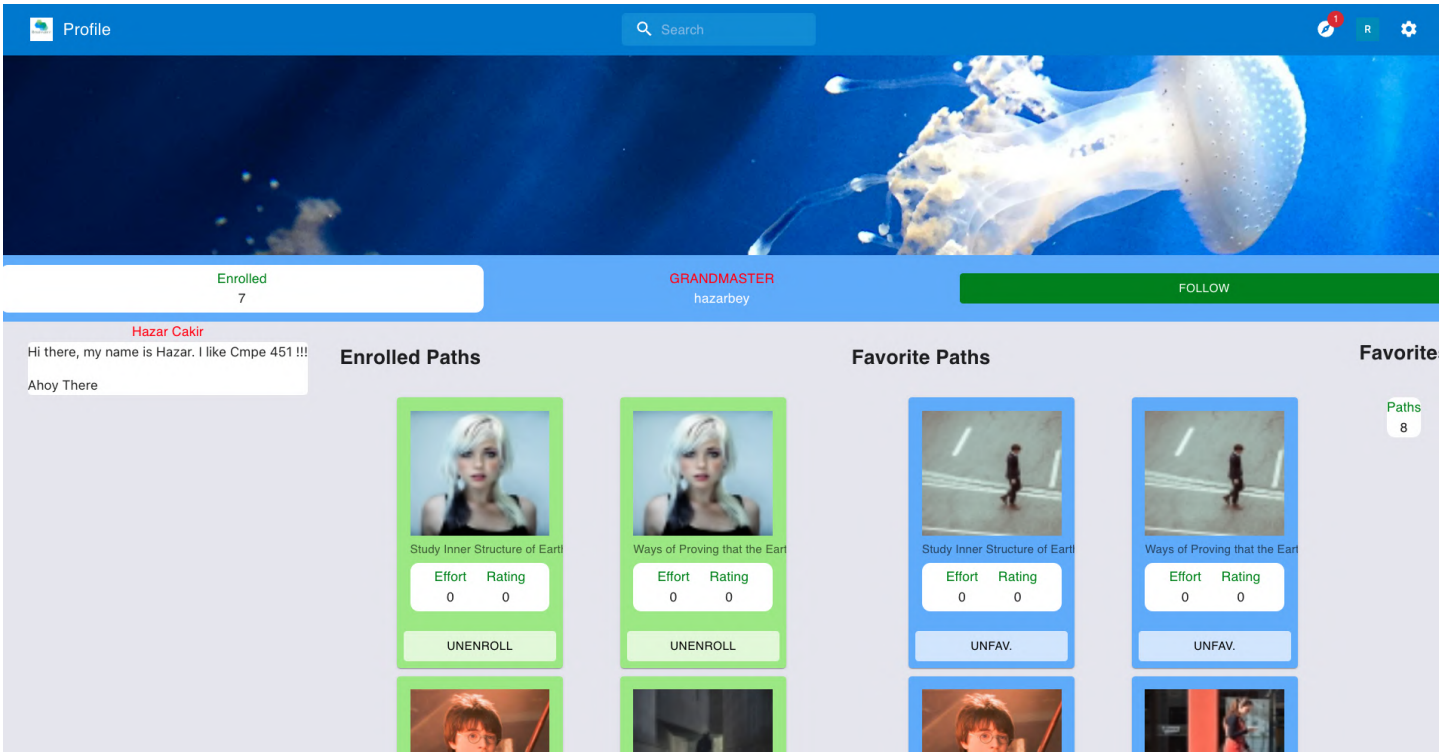
Search Page:



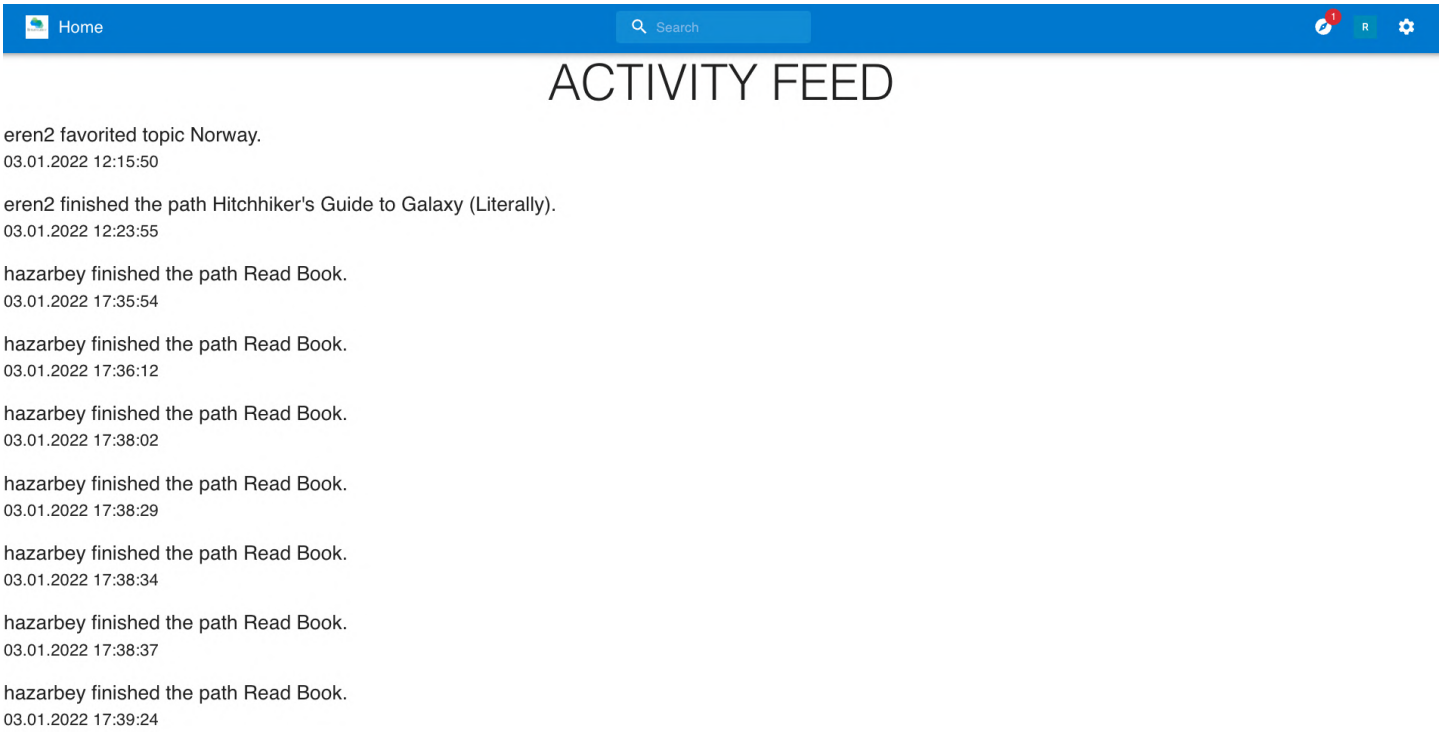
The second available feature on the navigation button is search box. Here users are able to search tags, the users and paths using the search bar at the top of the screen. After writing and waiting, the results are loaded one by one. At the top part, one can see Topics includes search term. Under Topics, the other users in Renaissance is listed. At the bottom part, there are all paths includes keyword. Navigation to the topic page, other users page and path is again possible via tapping on them. The Topic page and Path page is already showed. Here is the other users profile page when tapping on:



As you can see, edit profile page features are not shown to the user. But, there exist, follow and unfollow feature:



Activity Stream



This is the third section on the navigation bar and allows user to see the activities related to the user.

8. System Manual

Android

For installing flutter, we've already provided necessary information on previous report. After installing flutter you can follow the following steps:

For the system requirements, Andorid 7.1.1+ is supported. However, we've generally tested in Android 10+. So, it would be better to use in these devices .

For the installation. We've alredy provided apk in our release tag. You can install the application with just double-clicking to APK on your mobile device. You can also follow same steps on emulator.

PS: If you want to build apk for yourself, Go to 2021SpringGroup11\source\android\Renaissance on terminal and run the following command: `flutter build apk --release`

This will generate apk on 2021SpringGroup11\source\android\Renaissance\build\app\outputs\flutter-apk\apk-release.apk

You can use this apk and follow the same steps provided.

To change the backend url: Go to 2021SpringGroup11\source\android\Renaissance\lib\http_services.dart change the base_url variable and rebuild the apk.

Web Frontend

System requirements (for development)

- Node.js 14+
- developed on windows 10, mac os and ubuntu 20.04
- npm.com network access
- yarn

System requirements (for client app)

- for best performance, Chromium 80+ based browsers

Development Instructions

Available Scripts

In the project directory, you can run:

```
yarn install
```

Installs the necessary npm packages that is specified at packages.json.

```
yarn start
```

Runs the app in the development mode.
Open http://localhost:3000 to view it in the browser.

The page will reload if you make edits.
You will also see any lint errors in the console.

`yarn test`

Launches the test runner in the interactive watch mode.
See the section about running tests for more information.

`yarn build`

Builds the app for production to the `build` folder.
It correctly bundles React in production mode and optimizes the build for the best performance.

Production Instructions with Docker

Build production image

- `docker-compose -f docker-compose.prod.yml build`

To start our production container on port 80 with the name RenaissanceFE

- `docker run -p 80:80 --name RenaissanceFE frontend-prod`

Backend

How to run

- `python3 -m venv env`
- `source env/bin/activate`
- `pip install -r requirements.txt`
- `cd api`
- `cp .env-template.yaml .env.yaml`
- *In this stage, put* `SECRET_KEY: 'Your_Secret_Key'` *in the* `.env.yaml`
- `python manage.py runserver 8000`

9. Project Plan

- PDF
- PDF with gantt chart

		Name	Duration	Start	Finish	Predecessors	Resource Names
1		**Infrastructure Setup**	**6 days?**	**3/23/21 8:00 AM**	**3/30/21 5:00 PM**		
2	✔	Modify the README document	5 days?	3/23/21 8:00 AM	3/29/21 5:00 PM		Bekir Tatar
3	✔	Customize the "issues"	2 days?	3/23/21 8:00 AM	3/24/21 5:00 PM		Selman Berk Özkurt
4	✔	Research on some Repos,Git and Github	5 days?	3/23/21 8:00 AM	3/29/21 5:00 PM		Each Team Member
5	✔	Launch a slack channel	7 days?	3/23/21 8:00 AM	3/29/21 5:00 PM		Fikri Cem Ylmaz;Muhammed Emin Deniz
6	✔	Prepare a meeting template	7 days?	3/23/21 8:00 AM	3/29/21 5:00 PM		Hatice ule Erkul;Alper Canberk Balc
7		Prepare a communication plan	6 days?	3/23/21 8:00 AM	3/30/21 5:00 PM		Egemen Kaplan
8	✔	Create personal wiki pages	2 days?	3/29/21 8:00 AM	3/30/21 5:00 PM		Each Team Member
9	✔	Create a doodle for regular meetings	4 days?	3/23/21 8:00 AM	3/26/21 5:00 PM		Muhammed Emin Deniz
10		**Reqiurement Elicitation**	**10 days?**	**3/31/21 8:00 AM**	**4/9/21 5:00 PM**	**1**	
11	✔	Identify the reqiurements	6 days?	3/31/21 8:00 AM	4/5/21 5:00 PM		Bekir Tatar;Selman Berk Özkurt
12		Examine related software systems	4 days?	4/4/21 8:00 AM	4/8/21 5:00 PM		Each Team Member
13	✔	Identify a list of questions for customer	5 days?	4/4/21 8:00 AM	4/8/21 5:00 PM		Alper Canberk Balc;Selman Berk Özkurt;Fikri Cem Ylmaz
14	✔	Sort the reqiurements	4 days?	4/6/21 8:00 AM	4/9/21 5:00 PM	11	Fikri Cem Ylmaz
15	✔	Prepare a application comparison page	1 day?	4/9/21 8:00 AM	4/9/21 5:00 PM	12	Muhammed Emin Deniz
16	✔	1. Customer Meeting for reqiurements	1 day?	4/9/21 8:00 AM	4/9/21 5:00 PM	13	Each Team Member
17	✔	Document communication plan in a nicer manner	4 days?	4/4/21 8:00 AM	4/8/21 5:00 PM		Hatice ule Erkul
18	✔	**Scenarios and Mockups**	**12 days?**	**4/13/21 8:00 AM**	**4/24/21 5:00 PM**	**10**	
19	✔	Decide on key points about the architecture	2 days?	4/13/21 8:00 AM	4/14/21 5:00 PM		Each Team Member
20	✔	Scenario 1	0.375 da...	4/18/21 2:00 PM	4/18/21 5:00 PM	19	Fikri Cem Ylmaz;Alper Canberk Balc;Mehmet Yasin erem...
21	✔	Scenario 2	0.375 da...	4/18/21 2:00 PM	4/18/21 5:00 PM	19	Fikri Cem Ylmaz;Alper Canberk Balc;Mehmet Yasin erem...
22	✔	Scenario 3	0.375 da...	4/18/21 2:00 PM	4/18/21 5:00 PM	19	Fikri Cem Ylmaz;Alper Canberk Balc;Selman Berk Özkurt;...
23	✔	Complete scenario 1 for web interface	1 day?	4/20/21 8:00 AM	4/20/21 5:00 PM		Eren Altunolu
24	✔	Complete scenario 2 for web interface	4 days?	4/20/21 8:00 AM	4/23/21 5:00 PM		Selman Berk Özkurt
25	✔	Complete scenario 3 for Android app	5 days?	4/20/21 8:00 AM	4/24/21 5:00 PM		Fikri Cem Ylmaz
26	✔	**Design Documents**	**5.375 da...**	**4/25/21 2:00 PM**	**4/30/21 5:00 PM**	**18**	
27	✔	Use case Diagram	0.375 da...	4/25/21 2:00 PM	4/25/21 5:00 PM		Each Team Member
28	✔	Class Diagram	0.375 da...	4/25/21 2:00 PM	4/25/21 5:00 PM		Each Team Member
29	✔	**Sequence Diagram**	**5 days?**	**4/26/21 8:00 AM**	**4/30/21 5:00 PM**	**27;28**	
30	✔	Sign up	2 days?	4/26/21 8:00 AM	4/27/21 5:00 PM		Selman Berk Özkurt
31	✔	Search	3 days?	4/26/21 8:00 AM	4/28/21 5:00 PM		Bekir Tatar
32	✔	Edit Profile	5 days?	4/26/21 8:00 AM	4/30/21 5:00 PM		Alper Canberk Balc
33	✔	Follow / Unfollow Topic	5 days?	4/26/21 8:00 AM	4/30/21 5:00 PM		Eren Altunolu
34	✔	Upvote / Downvote	4 days?	4/26/21 8:00 AM	4/29/21 5:00 PM		Mehmet Yasin eremet
35	✔	Edit Comment	5 days?	4/26/21 8:00 AM	4/30/21 5:00 PM		Muhammed Emin Deniz
36	✔	Edit Resource	4 days?	4/26/21 8:00 AM	4/29/21 5:00 PM		Hatice ule Erkul
37	✔	Post Comment	2 days?	4/26/21 8:00 AM	4/27/21 5:00 PM		Fikri Cem Ylmaz
38	✔	2. Customer meeting for Design Documents	1 day?	4/27/21 8:00 AM	4/27/21 5:00 PM		Each Team Member
39	✔	**Project Planning and RAM**	**5 days?**	**5/4/21 8:00 AM**	**5/8/21 5:00 PM**		
40	✔	Project Planning	5 days?	5/4/21 8:00 AM	5/8/21 5:00 PM		Bekir Tatar
41	✔	RAM	4 days?	5/4/21 8:00 AM	5/7/21 5:00 PM		Fikri Cem Ylmaz
42	✔	**Milestone 1 Report**	**7 days?**	**5/4/21 8:00 AM**	**5/10/21 5:00 PM**	**1;10;18;26**	
43	✔	Reqiurements	6 days?	5/4/21 8:00 AM	5/9/21 5:00 PM		Alper Canberk Balc;Hatice ule Erkul;Muhammed Emin De...
44	✔	**Scenarios and Mockups**	**5 days?**	**5/4/21 8:00 AM**	**5/8/21 5:00 PM**		**Alper Canberk Balc;Hatice ule Erkul;Muhammed...**
45	✔	Scenario 4 (additonal)	5 days?	5/4/21 8:00 AM	5/8/21 5:00 PM		Alper Canberk Balc
46	✔	Scenario 5 (additonal)	5 days?	5/4/21 8:00 AM	5/8/21 5:00 PM		Hatice ule Erkul
47	✔	Scenario 6 (additonal)	5 days?	5/4/21 8:00 AM	5/8/21 5:00 PM		Muhammed Emin Deniz
48	✔	Design documents	6 days?	5/4/21 8:00 AM	5/9/21 5:00 PM		Eren Altunolu;Selman Berk Özkurt
49	✔	Project Plan & RAM	6 days?	5/4/21 8:00 AM	5/9/21 5:00 PM		Fikri Cem Ylmaz;Bekir Tatar
50	✔	3. Customer meeting for milestone report	0 days?	5/8/21 8:00 AM	5/8/21 8:00 AM		Each Team Member
51	✔	Executive Summary & Deliverables	2 days?	5/8/21 8:00 AM	5/9/21 5:00 PM		Mehmet Yasin eremet;Hatice ule Erkul
52	✔	Evaluation of delivarable,tools and processes	0 days?	5/8/21 8:00 AM	5/10/21 5:00 PM		Each Team Member
53	✔	A summary of work done	2 days?	5/8/21 8:00 AM	5/9/21 5:00 PM		Each Team Member
54	✔	**Practice-app Implementation**	**20 days?**	**5/17/21 8:00 AM**	**6/5/21 5:00 PM**		
55	✔	Research about public APIs	11 days?	5/17/21 8:00 AM	5/27/21 5:00 PM		Each Team Member
56	✔	Learning Django and basics of WEB	11 days?	5/17/21 8:00 AM	5/27/21 5:00 PM		Each Team Member
57	✔	Stocks API	6 days?	5/31/21 8:00 AM	6/5/21 5:00 PM		Bekir Tatar
58	✔	Commodities	6 days?	5/31/21 8:00 AM	6/5/21 5:00 PM		Eren Altunolu
59	✔	FXrates	6 days?	5/31/21 8:00 AM	6/5/21 5:00 PM		Muhammed Emin Deniz
60	✔	Spam	6 days?	5/31/21 8:00 AM	6/5/21 5:00 PM		Selman Berk Özkurt
61	✔	ETFSector	6 days?	5/31/21 8:00 AM	6/5/21 5:00 PM		Mehmet Yasin eremet
62	✔	Logosearch	6 days?	5/31/21 8:00 AM	6/5/21 5:00 PM		Fikri Cem Ylmaz
63		**Practice-app Dockerize and Deployment**	**4 days?**	**6/7/21 8:00 AM**	**6/10/21 5:00 PM**		
64	✔	Reseacrh about Docker and AWS	4 days?	6/7/21 8:00 AM	6/10/21 5:00 PM		Each Team Member
65	✔	Docker	4 days?	6/7/21 8:00 AM	6/10/21 5:00 PM		Each Team Member
66		AWS ECS Deployment	4 days?	6/7/21 8:00 AM	6/10/21 5:00 PM		Each Team Member
67	✔	**Milestone 2 report**	**3 days?**	**6/11/21 8:00 AM**	**6/13/21 5:00 PM**	**42;63**	
68	✔	Delivarables	3 days?	6/11/21 8:00 AM	6/13/21 5:00 PM		Each Team Member
69	✔	Report	3 days?	6/11/21 8:00 AM	6/13/21 5:00 PM		Each Team Member
70	✔	**Review And Update Project Material**	**7 days?**	**10/12/21 8:00 AM**	**10/18/21 5:00 PM**		

		Name	Duration	Start	Finish	Predecessors	Resource Names
71		Reqiurements	7 days?	10/12/21 8:00 AM	10/18/21 5:00 PM		Fikri Cem Ylmaz;Selman Berk Özkurt
72		Mockups	7 days?	10/12/21 8:00 AM	10/18/21 5:00 PM		Fikri Cem Ylmaz;Meltem Arslan;Kürat Bak
73		Design	7 days?	10/12/21 8:00 AM	10/18/21 5:00 PM		Kürat Bak;Meltem Arslan;Fikri Cem Ylmaz;Muhammed ...
74		Project Plan	7 days?	10/12/21 8:00 AM	10/18/21 5:00 PM		Eren Altunolu;Bekir Tatar
75		Determine the developer team roles	3 days?	10/13/21 8:00 AM	10/15/21 5:00 PM		Each Team Member
76		**Backend 1**	**15 days?**	**10/19/21 8:00 AM**	**11/2/21 5:00 PM**	**67**	**Backend Team**
77		Decide on API and Database technologies	4 days?	10/19/21 8:00 AM	10/22/21 5:00 PM		Backend Team
78		Configure database for users	11 days?	10/19/21 8:00 AM	11/2/21 5:00 PM		Eren Altunolu
79		API for Signup - Login	5 days?	10/19/21 8:00 AM	10/23/21 5:00 PM		Selman Berk Özkurt
80		API for Delete User	6 days?	10/26/21 8:00 AM	10/31/21 5:00 PM		Mehmet Yasin eremet
81		Admin user functionalities	11 days?	10/19/21 8:00 AM	10/29/21 5:00 PM		Eren Altunolu
82		API for User Search	6 days?	10/26/21 8:00 AM	10/31/21 5:00 PM		Selman Berk Özkurt
83		**Frontend 1**	**13 days?**	**10/19/21 8:00 AM**	**10/31/21 5:00 PM**	**67**	**Frontend Team**
84		Decide on Frontend Technologies	4 days?	10/19/21 8:00 AM	10/22/21 5:00 PM		Frontend Team
85		Initial version of home page	11 days?	10/19/21 8:00 AM	10/29/21 5:00 PM		Meltem Arslan
86		Sgnup - Login	5 days?	10/19/21 8:00 AM	10/23/21 5:00 PM		Bekir Tatar
87		Initial version of profile page	5 days?	10/19/21 8:00 AM	10/23/21 5:00 PM		Muhammed Emin Deniz
88		User Search	7 days?	10/25/21 8:00 AM	10/31/21 5:00 PM		Bekir Tatar
89		Navigation Bar	7 days?	10/25/21 8:00 AM	10/31/21 5:00 PM		Muhammed Emin Deniz
90		**Android 1**	**13 days?**	**10/19/21 8:00 AM**	**10/31/21 5:00 PM**	**67**	**Android Team**
91		Decide on Mobile Technologies	4 days?	10/19/21 8:00 AM	10/22/21 5:00 PM		Android Team
92		Initial version of home page	11 days?	10/19/21 8:00 AM	10/29/21 5:00 PM		Fikri Cem Ylmaz
93		Sign up - Login	5 days?	10/19/21 8:00 AM	10/23/21 5:00 PM		Hazar Çakr
94		Initial version of Profile page	5 days?	10/19/21 8:00 AM	10/23/21 5:00 PM		Kürat Bak
95		User Search	6 days?	10/26/21 8:00 AM	10/31/21 5:00 PM		Hazar Çakr
96		Navigation Bar	6 days?	10/26/21 8:00 AM	10/31/21 5:00 PM		Kürat Bak
97		**Milestone 3 report**	**13 days?**	**11/3/21 8:00 AM**	**11/15/21 5:00 PM**	**76;83;90**	**Each Team Member**
98		Review on initial stage of application	8 days?	11/3/21 8:00 AM	11/10/21 5:00 PM		Each Team Member
99		Connecting backend with frontend and android	8 days?	11/3/21 8:00 AM	11/10/21 5:00 PM		Android Team;Backend Team;Frontend Team
100		Initial deployment of application	7 days?	11/3/21 8:00 AM	11/9/21 5:00 PM		Frontend Team;Backend Team;Android Team
101		Testing	9 days?	11/3/21 8:00 AM	11/11/21 5:00 PM		Each Team Member
102		Delivarables	13 days?	11/3/21 8:00 AM	11/15/21 5:00 PM		Each Team Member
103		Report	13 days?	11/3/21 8:00 AM	11/15/21 5:00 PM		Each Team Member
104		**Backend 2**	**16 days?**	**11/16/21 8:00 AM**	**12/1/21 5:00 PM**	**97**	**Backend Team**
105		Configure database for resources and topics	6 days?	11/16/21 8:00 AM	11/21/21 5:00 PM		Eren Altunolu
106		Wikidata integration	6 days?	11/16/21 8:00 AM	11/21/21 5:00 PM		Selman Berk Özkurt
107		Follow Topics and Paths	6 days?	11/16/21 8:00 AM	11/21/21 5:00 PM		Eren Altunolu
108		Provide path	6 days?	11/16/21 8:00 AM	11/21/21 5:00 PM		Selman Berk Özkurt
109		Consume path	6 days?	11/16/21 8:00 AM	11/21/21 5:00 PM		Selman Berk Özkurt
110		Search topics and path	6 days?	11/16/21 8:00 AM	11/21/21 5:00 PM		Eren Altunolu
111		Rate material	8 days?	11/24/21 8:00 AM	12/1/21 5:00 PM		Mehmet Yasin eremet
112		User profile editing	8 days?	11/24/21 8:00 AM	12/1/21 5:00 PM		Mehmet Yasin eremet
113		Edit path	8 days?	11/23/21 5:00 PM	12/1/21 5:00 PM		Mehmet Yasin eremet
114		**Frontend 2**	**16 days?**	**11/16/21 8:00 AM**	**12/1/21 5:00 PM**	**97**	**Frontend Team**
115		Path and topic pages	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Bekir Tatar
116		Editing profile	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Muhammed Emin Deniz
117		Search topics and path	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Muhammed Emin Deniz
118		Follow topics and path	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Muhammed Emin Deniz
119		Provide path	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Bekir Tatar
120		Consume path	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Bekir Tatar
121		Effort and ratings	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Meltem Arslan
122		Edit path	8 days?	11/23/21 5:00 PM	12/1/21 5:00 PM		Meltem Arslan
123		**Android 2**	**16 days?**	**11/16/21 8:00 AM**	**12/1/21 5:00 PM**	**97**	**Android Team**
124		Resource and topic pages	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Fikri Cem Ylmaz
125		Editing profile	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Hazar Çakr
126		Search topics and paths	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Hazar Çakr
127		Follow Topics and Paths	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Hazar Çakr
128		Provide Paths	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Fikri Cem Ylmaz
129		Consume Path	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Fikri Cem Ylmaz
130		Effort and ratings	14 days?	11/16/21 8:00 AM	11/29/21 5:00 PM		Kürat Bak
131		Edit Path	8 days?	11/23/21 5:00 PM	12/1/21 5:00 PM		Kürat Bak
132		**Milestone 4 report**	**5 days?**	**12/2/21 8:00 AM**	**12/6/21 5:00 PM**	**104;114;123**	**Each Team Member**
133		Review on current version of application	4 days?	12/2/21 8:00 AM	12/5/21 5:00 PM		Each Team Member
134		Second deployment	3.08 days?	12/2/21 8:00 AM	12/5/21 8:38 AM		Frontend Team;Backend Team;Android Team
135		Testing	5 days?	12/2/21 8:00 AM	12/6/21 5:00 PM		Android Team;Backend Team;Frontend Team
136		Delivarables	5 days?	12/2/21 8:00 AM	12/6/21 5:00 PM		Each Team Member
137		Report	5 days?	12/2/21 8:00 AM	12/6/21 5:00 PM		Each Team Member
138		**Backend 3**	**14 days?**	**12/7/21 8:00 AM**	**12/20/21 5:00 PM**	**132**	**Backend Team**
139		Recommendation of materials	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Eren Altunolu
140		Notification services	8 days?	12/7/21 8:00 AM	12/14/21 5:00 PM		Eren Altunolu

		Name	Duration	Start	Finish	Predecessors	Resource Names
141		Comments	8 days?	12/7/21 8:00 AM	12/14/21 5:00 PM		Mehmet Yasin eremet
142		Milestone, Task and Resources	1 day?	12/7/21 8:00 AM	12/7/21 5:00 PM		Mehmet Yasin eremet
143		Activity Streams	12 days?	12/7/21 8:00 AM	12/18/21 5:00 PM		Selman Berk Özkurt
144		**Frontend 3**	**14 days?**	**12/7/21 8:00 AM**	**12/20/21 5:00 PM**	**132**	**Frontend Team**
145		Explore page	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Muhammed Emin Deniz
146		Notification integration	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Meltem Arslan
147		Comments	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Bekir Tatar
148		Milestone, Task and Resources	1 day?	12/7/21 8:00 AM	12/7/21 5:00 PM		Meltem Arslan
149		Activity Streams	12 days?	12/7/21 8:00 AM	12/18/21 5:00 PM		Bekir Tatar
150		Personal page - past efforts	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Muhammed Emin Deniz
151		Share accomplishments and topics	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Bekir Tatar
152		**Mobil 3**	**14 days?**	**12/7/21 8:00 AM**	**12/20/21 5:00 PM**	**132**	**Android Team**
153		Explore page	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Kürat Bak
154		Notification integration	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Hazar Çakr
155		Milestone, Task and Resources	1 day?	12/7/21 8:00 AM	12/7/21 5:00 PM		Hazar Çakr
156		Comments	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Hazar Çakr
157		Activity Streams	12 days?	12/7/21 8:00 AM	12/18/21 5:00 PM		Fikri Cem Ylmaz
158		Personal page - past efforts	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Kürat Bak
159		Share accomplishments and topics	14 days?	12/7/21 8:00 AM	12/20/21 5:00 PM		Fikri Cem Ylmaz
160		**Milestone 5**	**15 days?**	**12/21/21 8:00 AM**	**1/4/22 5:00 PM**	**138;144;152**	**Each Team Member**
161		Review on current version of application	15 days?	12/21/21 8:00 AM	1/4/22 5:00 PM		Each Team Member
162		Final deployment	15 days?	12/21/21 8:00 AM	1/4/22 5:00 PM		Frontend Team;Backend Team;Android Team
163		Testing	15 days?	12/21/21 8:00 AM	1/4/22 5:00 PM		Android Team;Backend Team;Frontend Team
164		Delivarables	15 days?	12/21/21 8:00 AM	1/4/22 5:00 PM		Each Team Member
165		Report	15 days?	12/21/21 8:00 AM	1/4/22 5:00 PM		Each Team Member

451-project-plan - page3

10. Software Requirements Specification

Glossary

- **Admin:** Some users who have the privilege of banning other users.
- **Android Application:** Software developed using Android Studio that can be distributed through Play Store. This is an instance of the interfaces in The System.
- **Ballot:** An object that displays the number of votes with each of positive and negative attributes and exposes a way for any user to add a new vote with positive or negative attributes.
- **Challenge-Response Authentication:** A service internal or external to The System that proves the entity causing requests of interacting with The System is indeed a human.
- **Collective Resource:** Similar to Resource with the additional set called roles, also with additional functionality of creating teams of one User associated with a role and providing Resource functionalities for each of the roles in the team.
- **Dashboard:** A single page in the Interface customized for each user that has a collection of links.
- **Effort:** Data representing a user's accomplishments specific to materials associated with some topics.
- **Infrastructure:** The functional component of The System that is responsible for handling, processing, recording synchronized data and making any programmed operations affecting overall functionality of the system. This is an application stack to be run on one or more computers connected to the global IP network.
- **Interface:** The functional component of The System that deals with representation of the logic and supplying a means of bidirectionally communicating with the infrastructure.
- **Material:** Some documents, applications or web-links associated with some topics users may benefit by studying.
- **Milestone:** An atomic objective in a path resource typically associated with a material.
- **Path Resource:** Similar to resource with the additional set called milestones, providing mark functionalities of the Resource for each of milestones separately.
- **Resource:** Container for material providing functionality to follow, mark as in progress, mark as finished and comment threads.
- **Role:** An atomic objective in a collective resource typically associated with a material.
- **Scheduled Resource:** Similar to path resource with each milestone associated with a period measured since the initial mark-in-progress event of the Scheduled resource.
- **The System:** The project in its all functional components including interface and back end.
- **Topic:** Abstract structure used for labeling most data flowing and being recorded in The System

- **User:** Entity accessing The System through its interfaces and the intended utilizer of overall functionality of The System. Users are associated with and are represented as *accounts* in The System
- **User Identifier:** A ISO/IEC 10646:2009 string that is associated with a user and is unique to that user.
- **Verified Contact:** Means of communication assumed to be accessible only by the User in real life which was proven by the user during signing up to be accessible by it.
- **Web Application:** Instance of the interface that can be run by a generic web browser. This shall be distributed on-demand through IP network using a DNS domain.

1. Functional Requirements

1.1. Accounts

- **1.1.1.** User *shall* be able to register by providing a valid and verified e-mail address, username, and a strong password.
- **1.1.2.** User *shall* be able to login by providing verified e-mail address.
- **1.1.3.** The user *shall* be able to delete their accounts.
- **1.1.4.** The user *shall* be able to reset its password via providing its e-mail address and via passing a challenge-response test such as "captcha" .
    - **1.1.4.1.** In this case, System *should* send a mail for resetting password. In order for resetting to happen, user should initiate resetting process from the e-mail that is specifically sent from the system.
- **1.1.5.** System *shall* include admin privileged user whose privileges are as follows:
    - **1.1.5.1** An admin privileged user *should* be ban or permanently delete users
    - **1.1.5.2** An admin privileged user *should* be remove comments, materials and resources.
    - **1.1.5.3** An admin privileged user *should* be able to give admin privileges to a normal user while declaring it as another admin.
- **1.1.6.** Every user *shall* have a unique and non-alterable username called user identifier which will *shall* be determined at the creation of the account.
    - **1.1.6.1.** System *shall* not allow new accounts to use an user identifier associated with a deleted account, or that is not associated with any account anymore for any reason .

1.2. Self-Development

- **1.2.1.** User *shall* create a resource which includes a description, image and materials. The details of resource is explained in the glossary.
- **1.2.2.** The System *shall* reliably record any progress data supplied by the users grouped by materials and topics.
    - **1.2.2.1.** The System *should* display at least one central statistic of this information grouped by materials and topics.
    - **1.2.2.2.** The System *should* display progress data associated by a user to the users associated with that user.
- **1.2.3.** The System *should* provide the user with a graphical overview of their past effort. There *should* be a page to show this information to the user.
- **1.2.4.** The System *shall* determine and inform its users about topics they could be potentially interested.
    - **1.2.4.1.** The System Interface *shall* provide a page that the users can intentionally navigate to and see this information that may also contain other functionality.
- **1.2.5.** The System *shall* supply its users with recommendation of resources.
    - **1.2.5.1.** The System Interface *shall* provide a page that the users can intentionally navigate to and see this information that may also contain other functionality.
- **1.2.6.** The System *should* offer a means of reliably proving realized effort and any accomplishment of a user to any third party authorized by that user.
- **1.2.7.** The users *shall* be able to reliably record and edit some objectives they determine as a set of material-deadline pairs.
- **1.2.8.** The users *should* be able to record topics they want to improve on, or break/develop a habit.
- **1.2.9.** The users *shall* be able to work on their determined objectives as a group.
    - **1.2.9.2.** Users *should* be able to create groups under collective resources that can be only visible to users that are invited to this group.

1.3. Social Interaction

- **1.3.1.** The users *should* have profile pages, and be able to modify their profiles.
- **1.3.2.** The users *should* be able to share their accomplishments and preferred topics via other applications.
- **1.3.3.** The users *should* be able to share their comments on specific topics or resources.

    - **1.3.3.1.** Any comment *shall* display user identifier of the user that posted that comment
        - **1.3.3.1.1.** If a user account is deleted, user identifier associated with a comment *shall* become "deleted user".
    - **1.3.3.2.** A comment *should* be able to have a comment list associated with it, which *should* have nested aligning with respect to the owning comment.
    - **1.3.3.3.** Any user *shall* have the option to delete any comment posted by that user.
        - **1.3.3.3.1.** If a comment is deleted by the posting user and there exists a nonempty comment list associated with it, then the Interface *shall* display the deleted parent comment as the message "deleted comment".
    - **1.3.3.4.** A comment *should* have a ballot associated with it which shows rate of the comment. Detailed ballot description is done at the glossary.
- **1.3.4.** The users *should* be able to search for other users via topics that is associated with them or via username.
- **1.3.5.** Users *shall* be able to mark any resource, topic or milestone as "in progress", "following" or "done".
    - **1.3.5.1.** The system interface *shall* provide a means of marking and unmarking.
    - **1.3.5.2.** The system interface *shall* display number of users marking each object.
    - **1.3.5.3.** System *shall* denote an association with the user and marked object as any of the above
- **1.3.7.** System *should* include two ballots denoting "effort" and "rating" associated with every resource.

1.4. User Experience

- **1.4.1.** The System interface *should* give its users notification messages.
    - **1.4.1.1.** If implemented, those notifications *shall* be given for scheduled resource milestones reminding the deadlines.
- **1.4.2.** The users *should* be able to customize the way the notifications are sent.
- **1.4.3.** The users *should* be able to search data assigned to topics using strings.
- **1.4.4.** The users *should* be able to access other users' profiles.
    - **1.4.4.1.** The System Interface *should* display user's associations with topics, resources and milestones in the profile.

2. Non-Functional Requirements

2.1. Availability

- **2.1.1.** An Android Application and a Web Application *shall* be implemented as instances of the Interface.
    - **2.1.1.1.** Android operating systems of version 10 and above *shall* be supported.
- **2.1.2.** The Interface *shall not* require any user to give text commands for any of its functionalities. It *shall* be possible to give all commands through the interface using a graphical interface.
    - *2.2.2.1.* Those commands *should* be either the standard input types provided by HTML 5 standard or those with equivalent functionality.
- **2.1.3.** The System *shall* express topics using semantic representation as defined in Wikidata.
    - **2.1.3.1.** Topics *shall* be indexed using both the integer semantic ID as indexed in Wikidata in the Infrastructure.
    - **2.1.3.2.** The Interface *should* represent the topics as strings in the preferred language of the operating system of the user and a representation in whatever language is available if it is not available.
        - **2.1.3.2.1.** If the representation of the user's language is not found and there are multiple languages for which a representation is available, English *shall* be prioritized.
- **2.1.4.** The Web Application *should* only use standard HTML5 and Javacscript code that is compatible with Chromium and Firefox .

2.2. Performance

- **2.2.1.** The Infrastructure *shall* be able to support at least 100 user actions per second.
- **2.2.2.** The Infrastructure *shall* be able to support at least 10000 accounts.
- **2.2.3.** The Infrastructure *shall* have a response time of less than 1.0 second excluding network delay.
- **2.2.4.** The Infrastructure *shall* have an uptime of at least 99%.

2.3. Privacy

- **2.3.1.** The System *shall* be compliant with GDPR.
    - **2.3.1.1.** Users *shall* be given explicit and concise explanation of how their data may be used.
    - **2.3.1.2.** The interface *shall* give the users a means of approving that they understood how their data may be used at the time of account creation.
    - **2.3.1.3.** If any change of how data may be used, all users *shall* be given a notice and prohibited from using the System until they approve it.
- **2.3.2.** Users *shall* have authority over visibility by third entities of data related to their accounts.

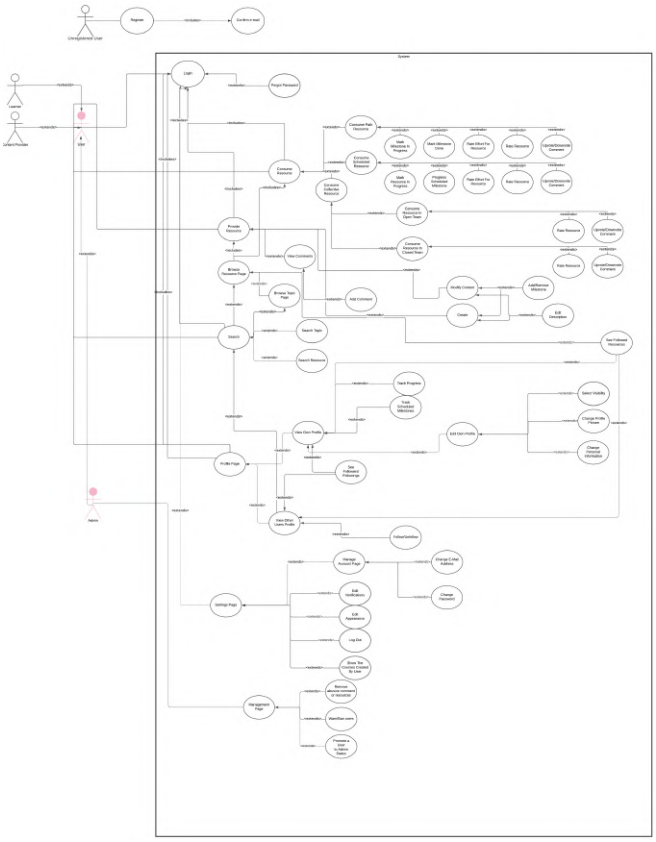
- **2.3.2.1.** The interface *should* provide a simple page in which the users can select this.

2.4. Security

- **2.4.1.** Data that shall be reliably recorded in the Infrastructure *should* be recorded using a means of redundancy, such as using RAID.
- **2.4.2.** Application layer protocol used by The System *shall* be restricted to HTTPS.
- **2.4.3** Passwords *shall* be hashed before being stored and verified using a NIST-approved cryptographically secure hash function using a salt that depends on the exact user and a system-wide nonce acquired from a reliable entropy source.

11. Software Design Documents

Use Case Diagram

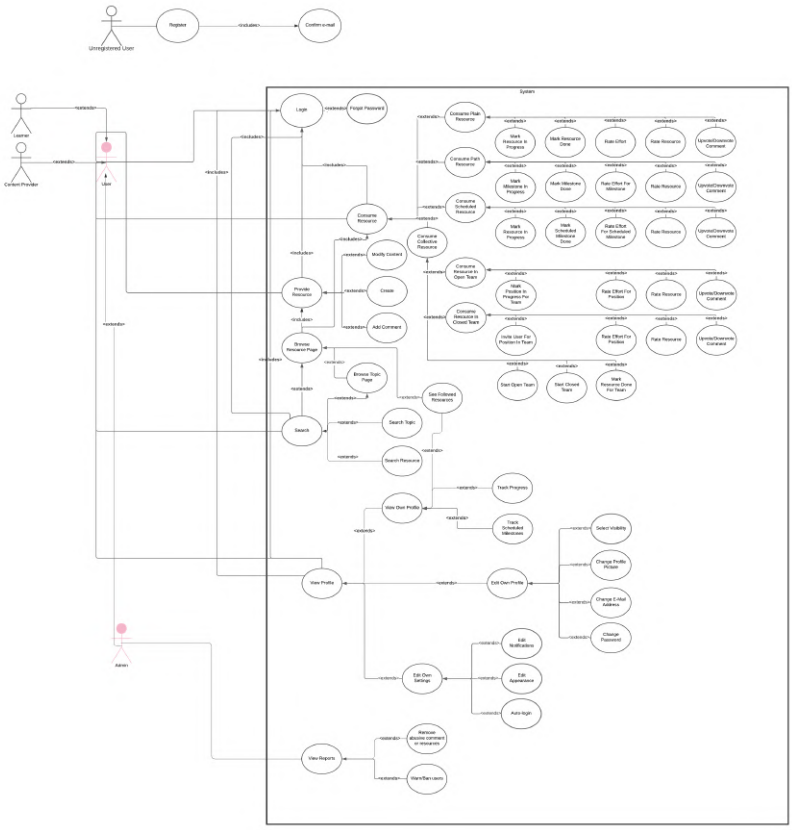


Old Versions

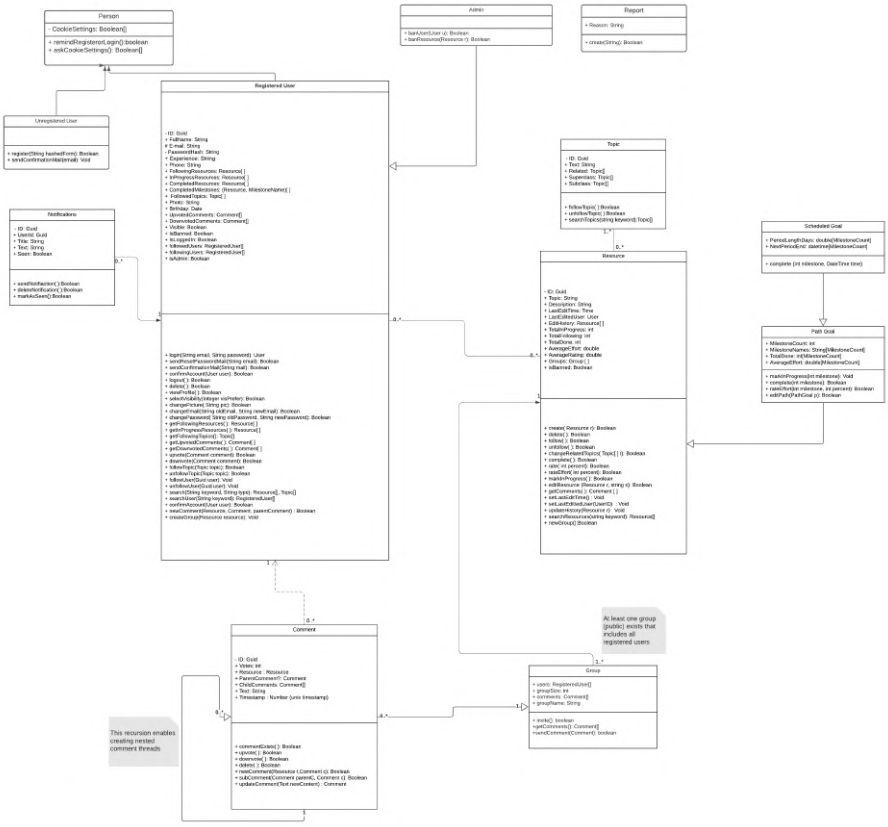
Changelog: 31.10.2021

- Admin can add new admin
- Admin has management page
- Direct access settings page after login
- They can see your courses from the settings page

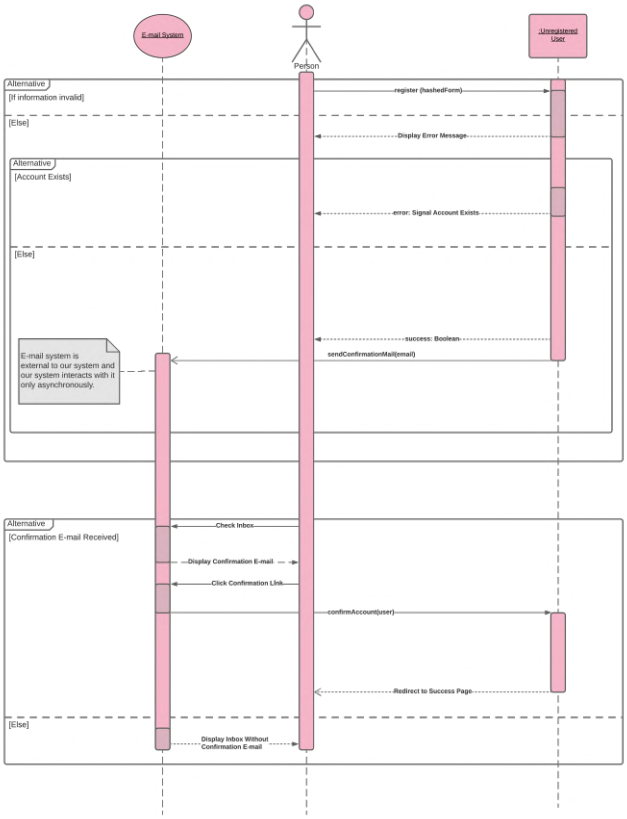
- Move some features from edit own profile to new manage account page
- View comment under resource



Class Diagram

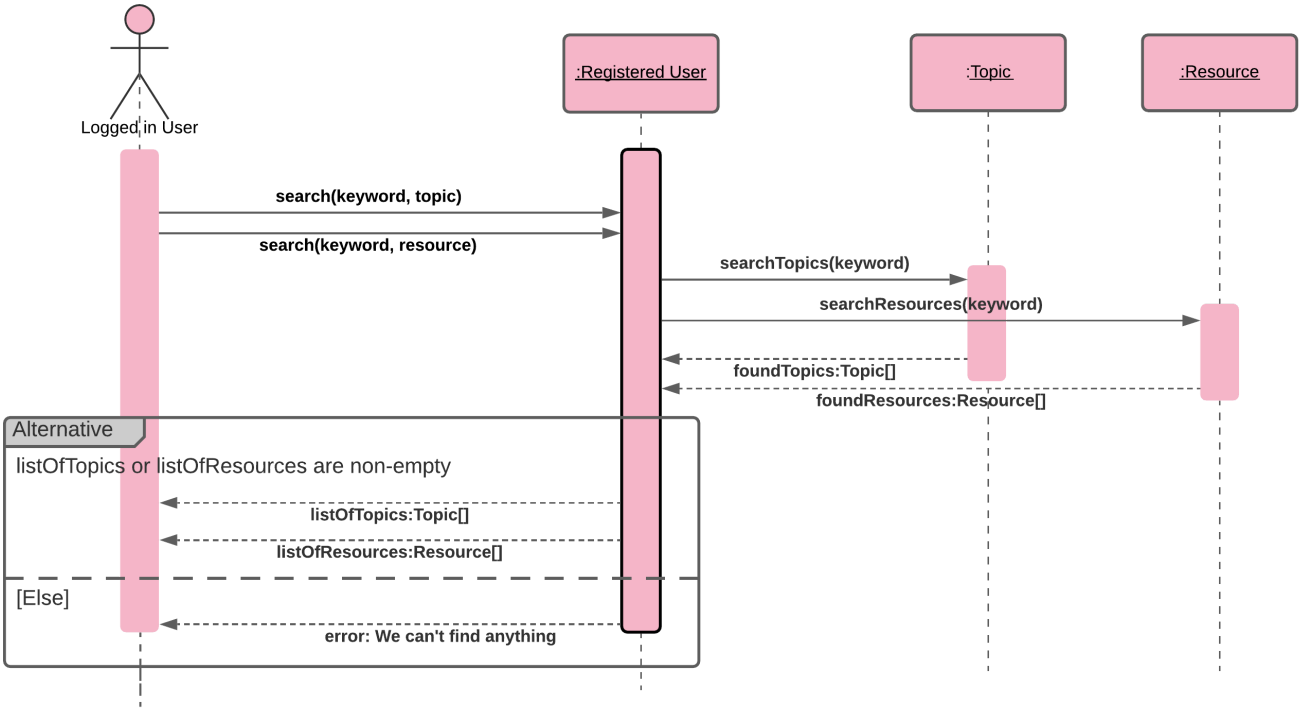


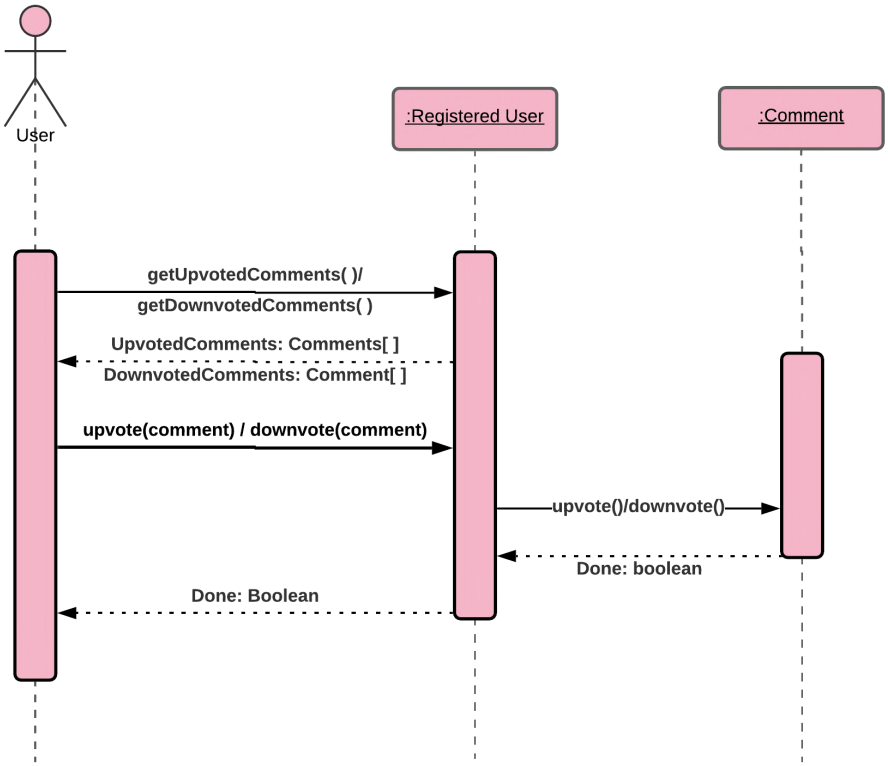
Sequence Diagrams



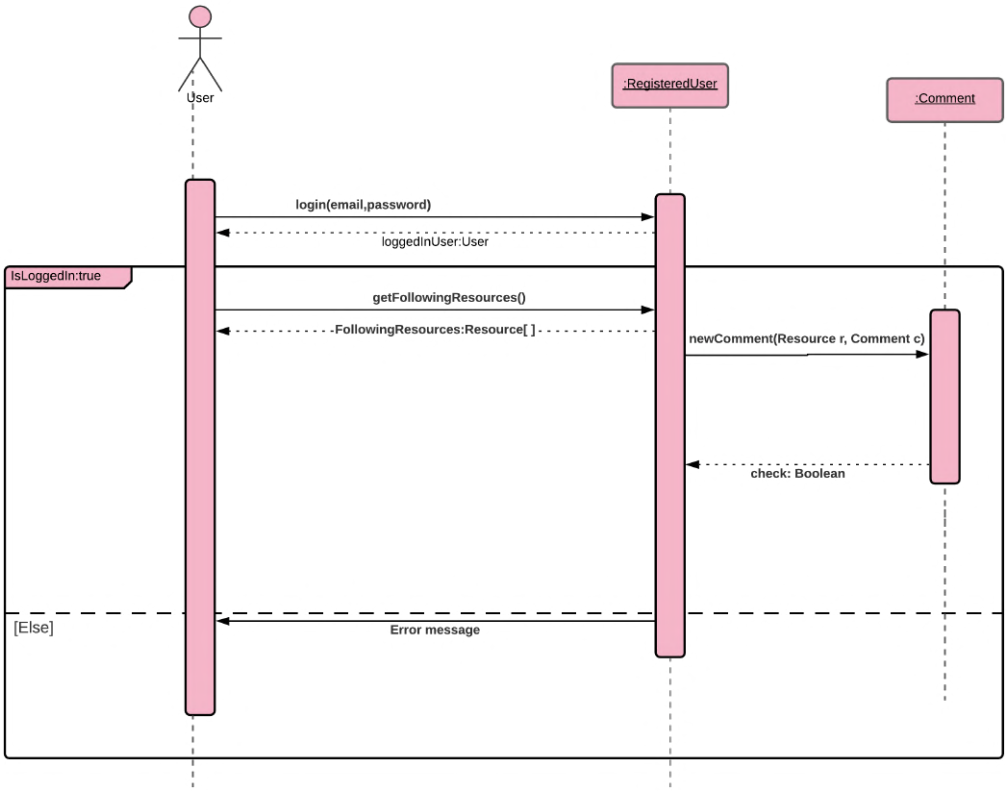
Sign Up

Search

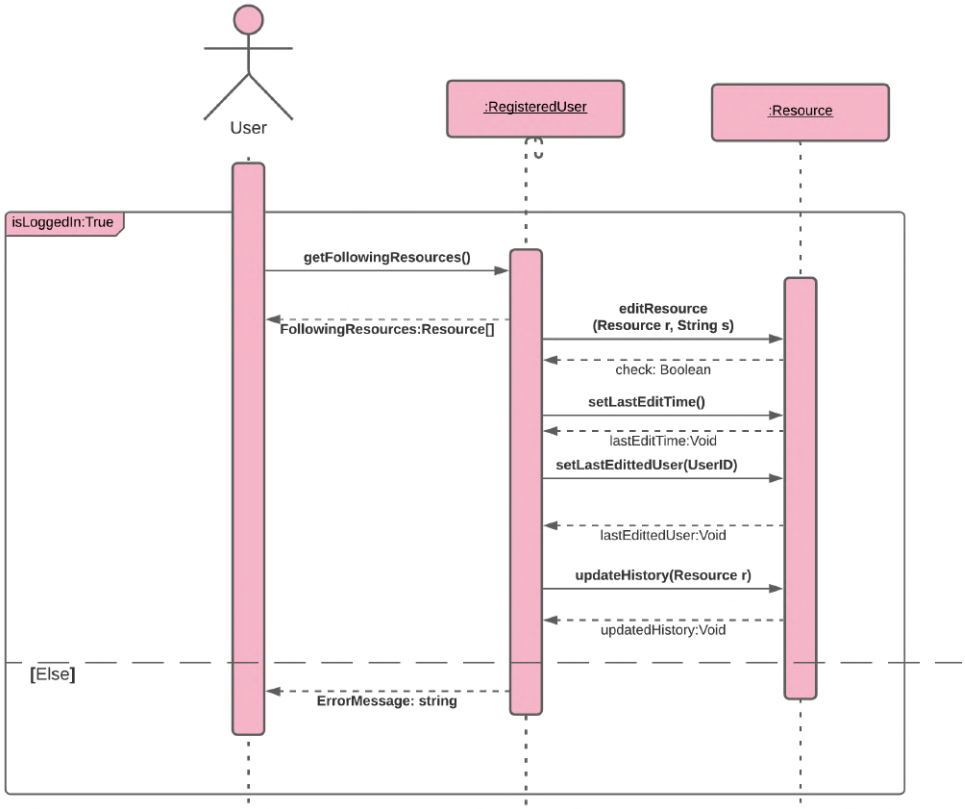




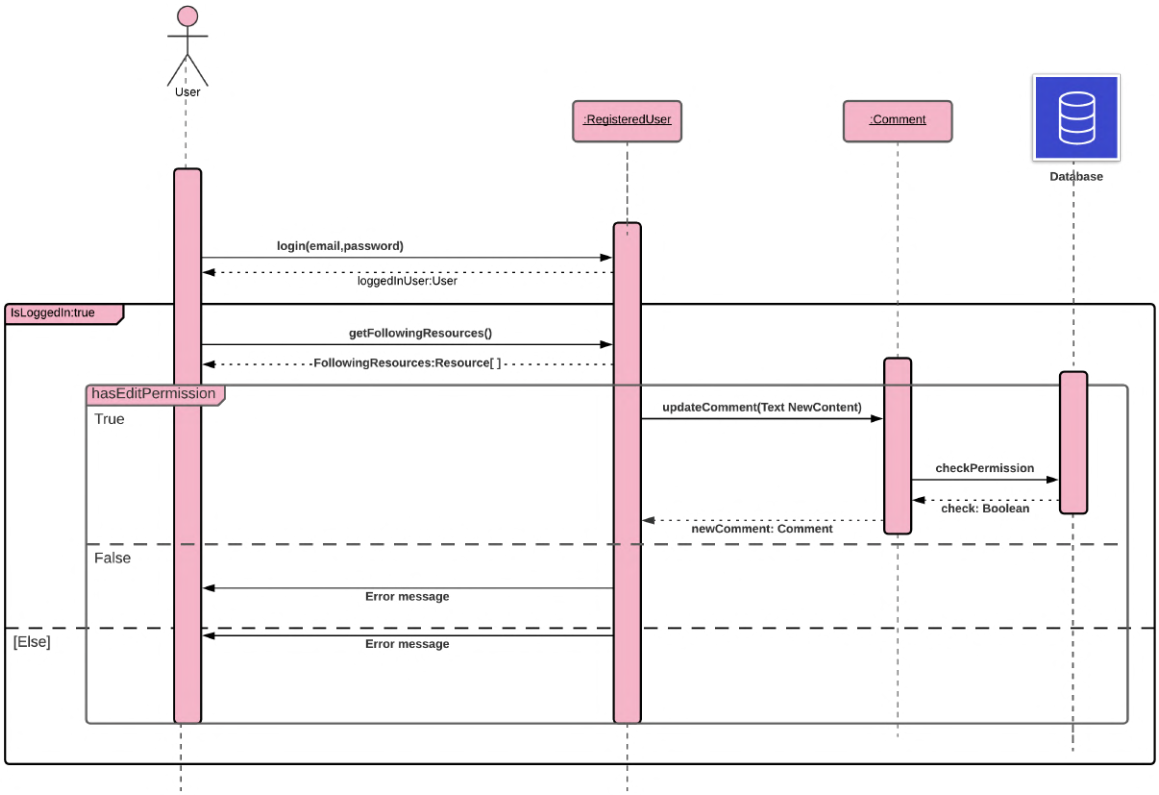
Upvote/Downvote



Post a Comment

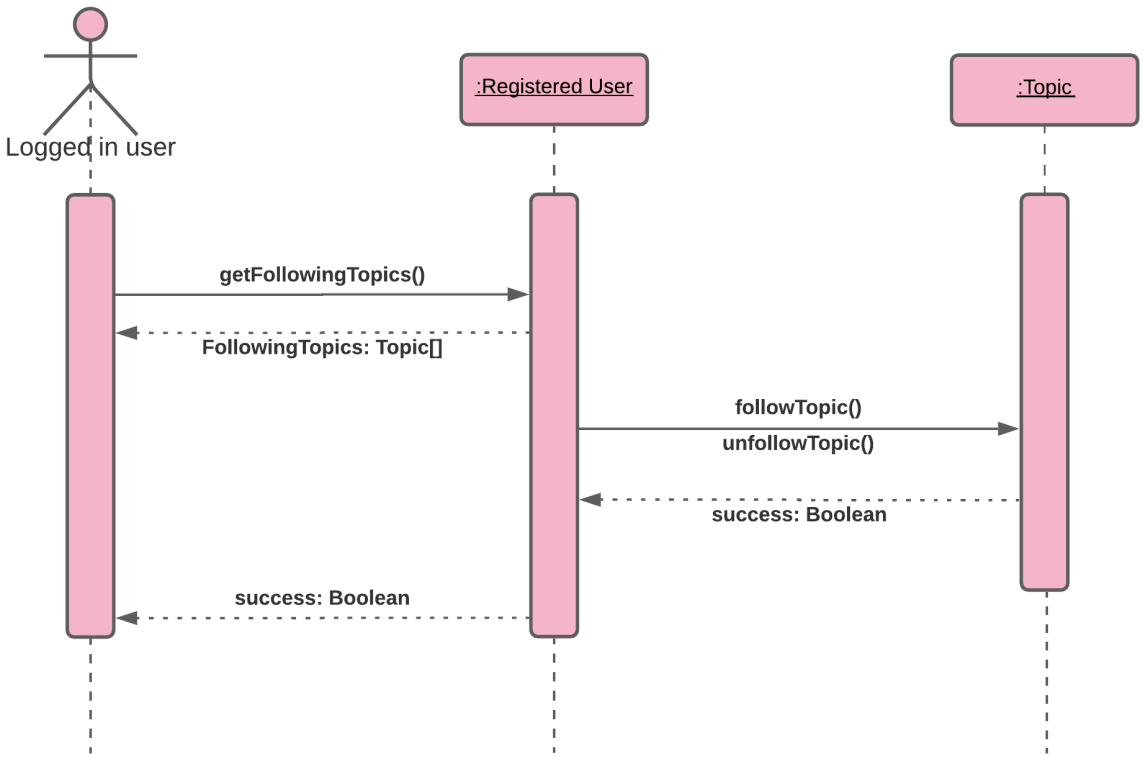


Edit Resource

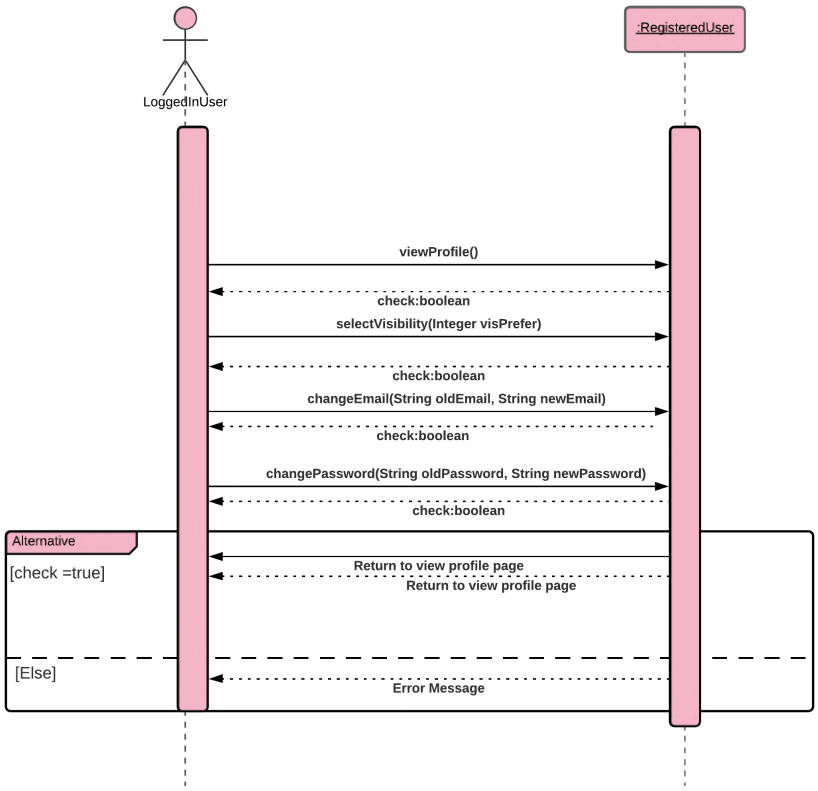


Edit Comment

Follow/Unfollow Topic



Edit Profile



Mockups

Scenario 1

User Persona

- **Photo:**
- **Name:** Mohammad Lee
- **Age:** 42
- **Job:** Businessperson
- **Interest:**

1. Mohammed tries to improve himself in various programming topics in order to strengthen his communication with the technical team of his company.
2. At the weekends, he likes to cook for his wife and his little daughter.
3. He workouts for 30 minutes every morning as starting the day.

Preconditions

- Mohammad is already signed up and logged in the android app.
- The app is started in Home page.

Goals

- Mohammad has more leisure time nowadays, so he wants to explore some new topics to be acknowledged about them according to his past interest areas.
- He decides to enter the his favorite app and see what it suggests.

Acceptance Criteria

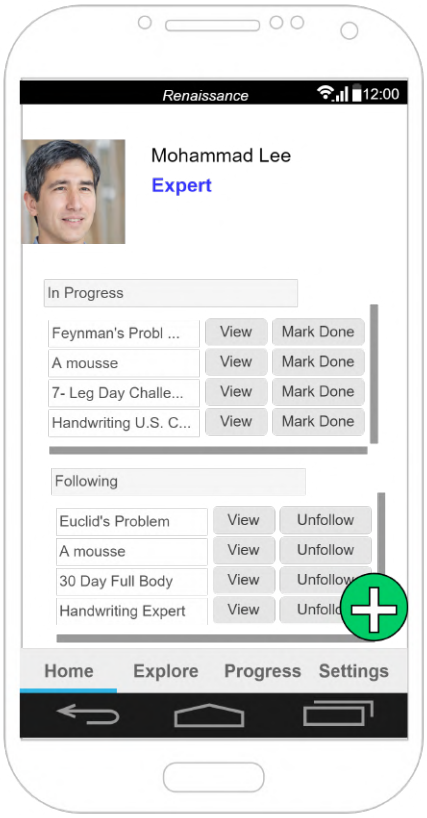
In reference to the Revision 1.1 of the requirements

- **(Requirement 1.1.2.)** The System shall provide its users access to it by means of specialized accounts.
- **(Requirement 1.2.3.)** The System shall determine and inform its users about topics they could be potentially interested.
- **(Requirement 1.2.4.)** The System shall supply its users with recommendation of materials.
- **(Requirement 1.2.7.)** The users should be able to reliably record topics they want to improve on.
- **(Requirement 1.3.1.)** The users should be able to create custom pages representing their accounts and those pages that ease their access to other functionalities, like a dashboard.
- **(Requirement 1.3.4.)** The users should be able to share their comments on specific topics.

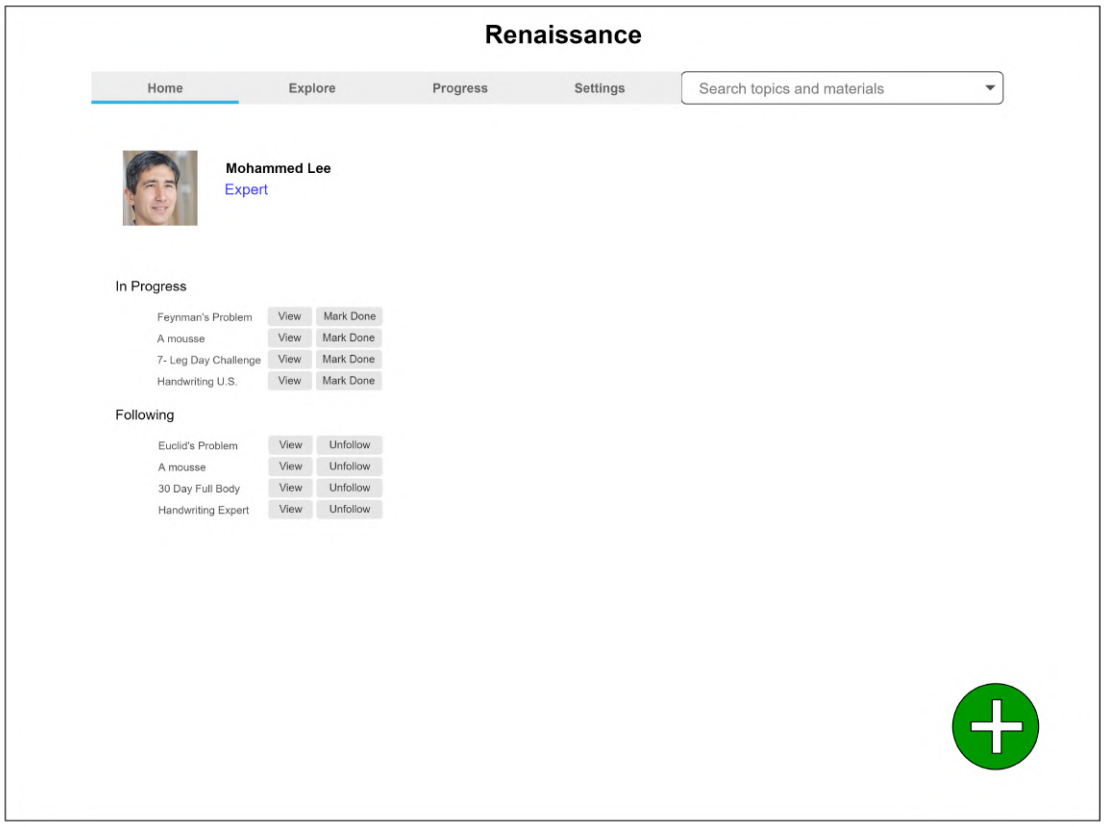
- **(Requirement 1.4.2.)** The users should be able to customize the way the notifications are sent.
- **(Requirement 2.1.1.)** An Android Application and a Web Application shall be implemented as instances of the Interface.

Actions & Mockups

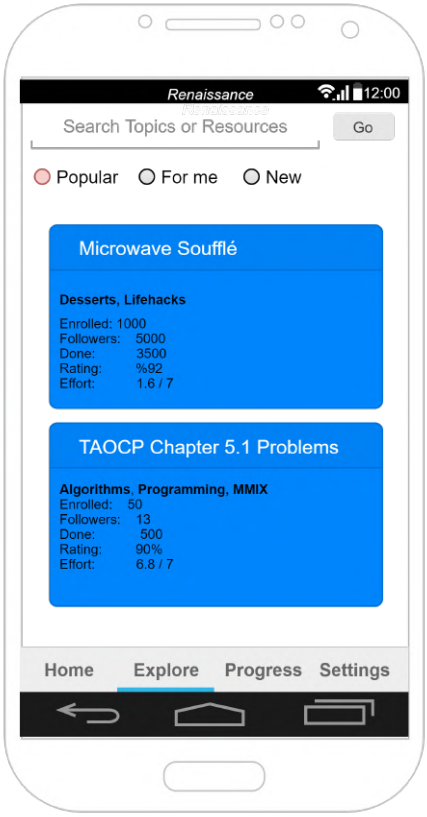
1. Mohammad is in pursuit of meeting a new topic increasing his technical knowledge. He is already an experienced user of our app. Since he is auto logged in to the app, when he clicks, the



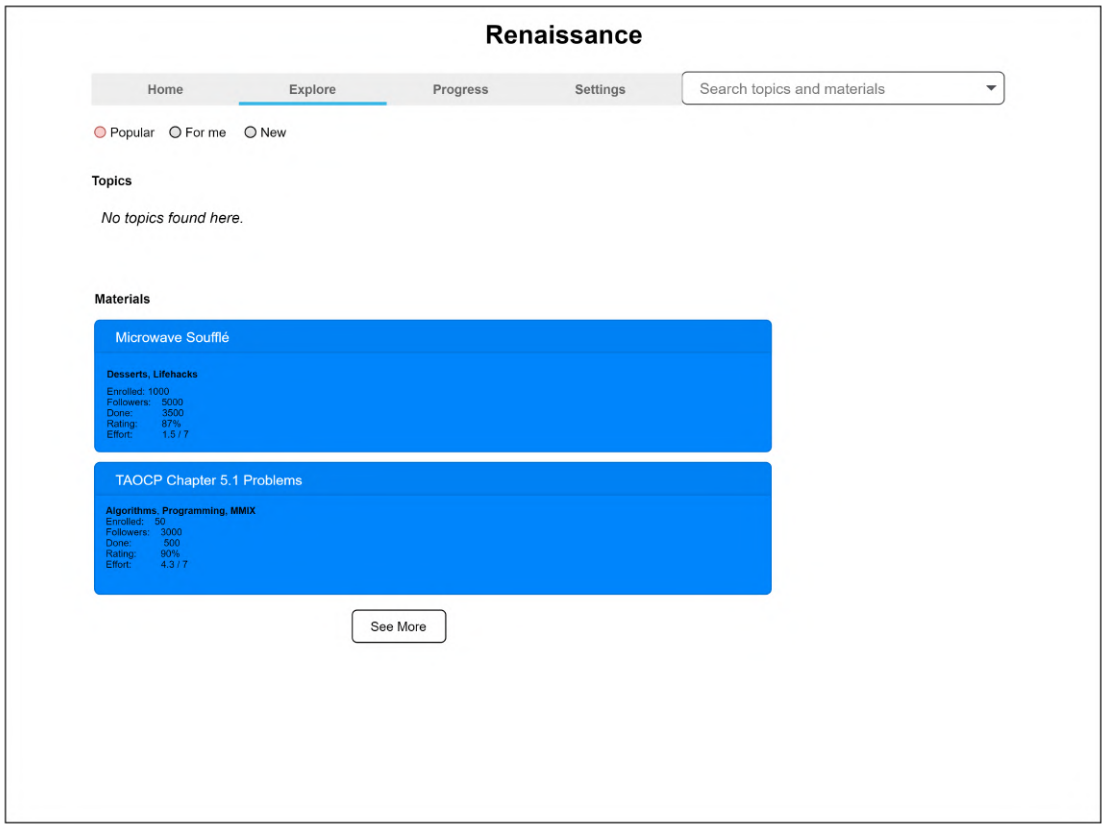
home page is opened.



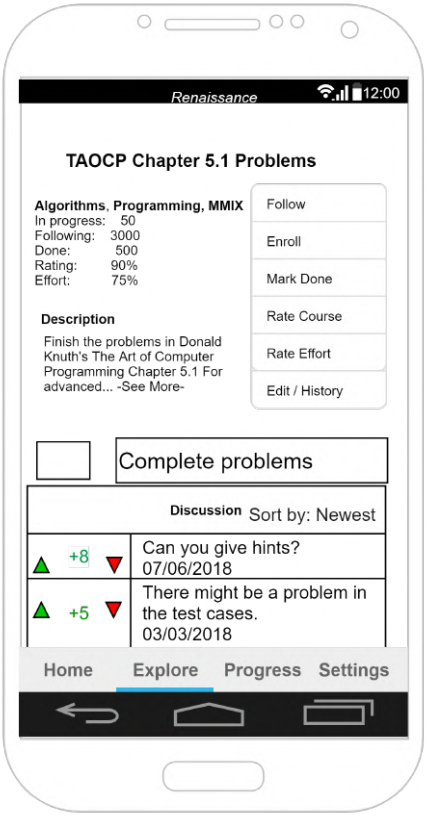
2. He goes to the Explore tab to see which topics the app suggests him. There are some options



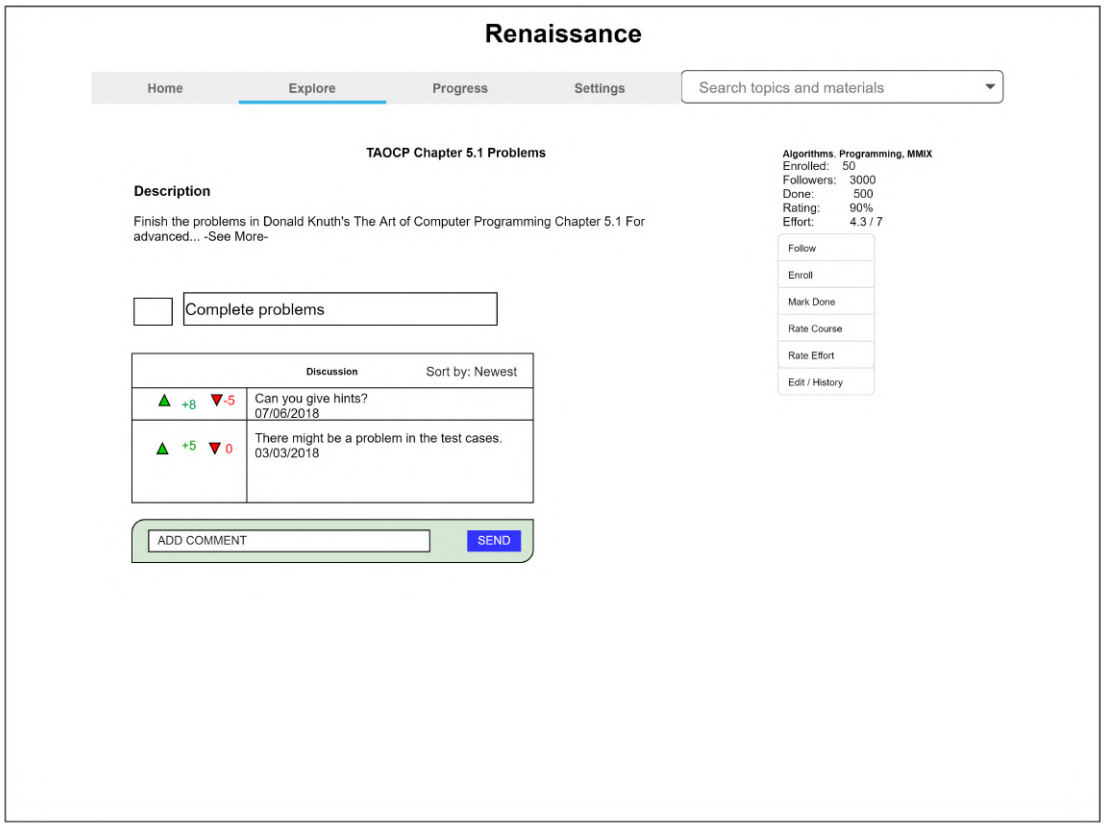
according to his past interests.



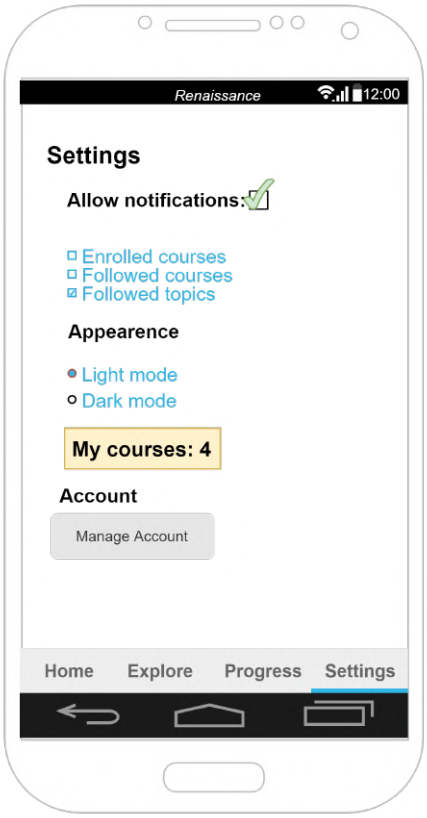
3. He clicks the second topic to view the related page including discussion part and various



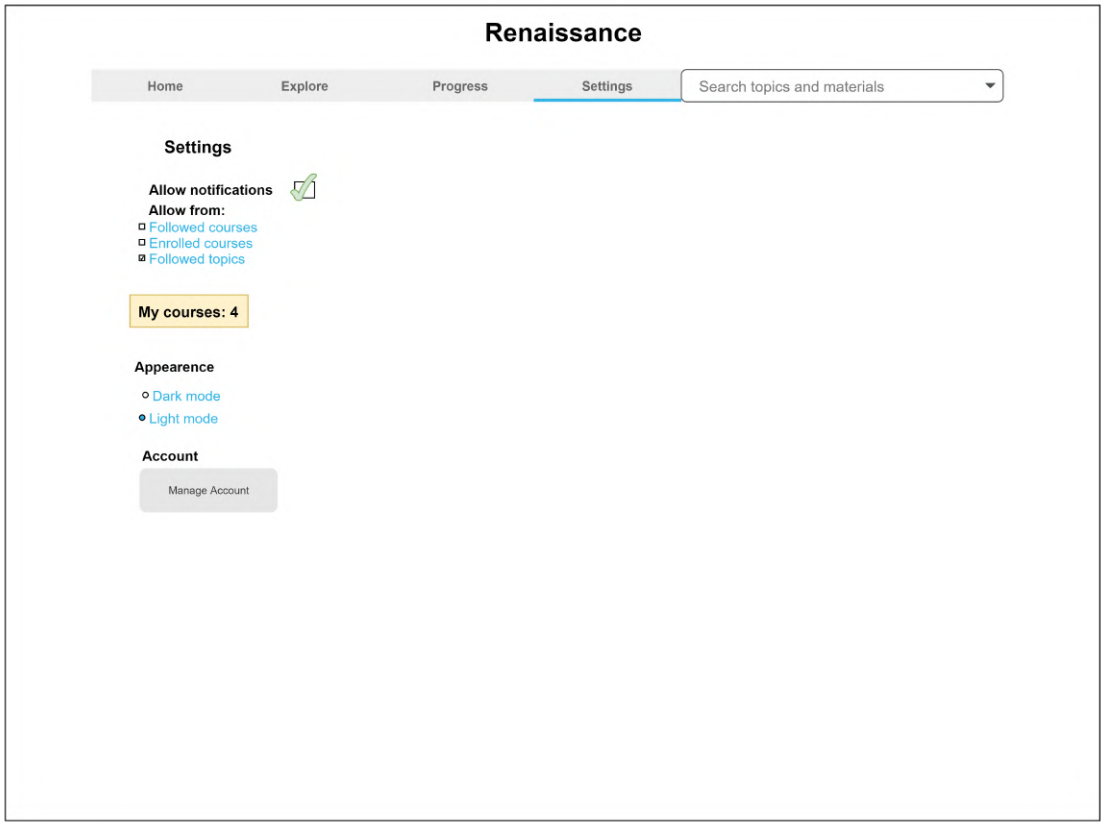
materials of this topic and follows it.



4. After he finishes his job in the Explore tab, he goes to the Settings tab. He turns on his notifications of the topics he follows with the curiosity of starting following a new topic before



quitting the app.



Scenario 2

User Persona

- **Photo:**
- **Name:** Ramesh Kardashayan
- **Age:** 32
- **Job:** Computer Engineer
- **Interest:** Programming, Handwriting, Rick Astley

Preconditions

- Ramesh is already signed up and logged in the android app.

Goals

- He wants to change the classification of an HTML exercise.

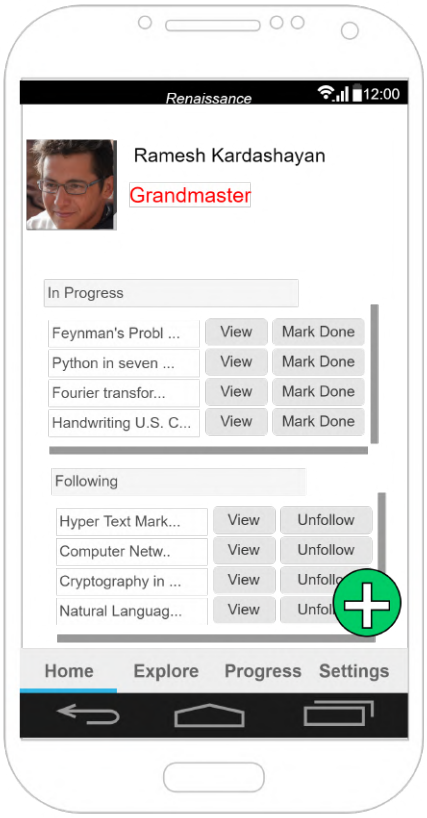
Acceptance Criteria

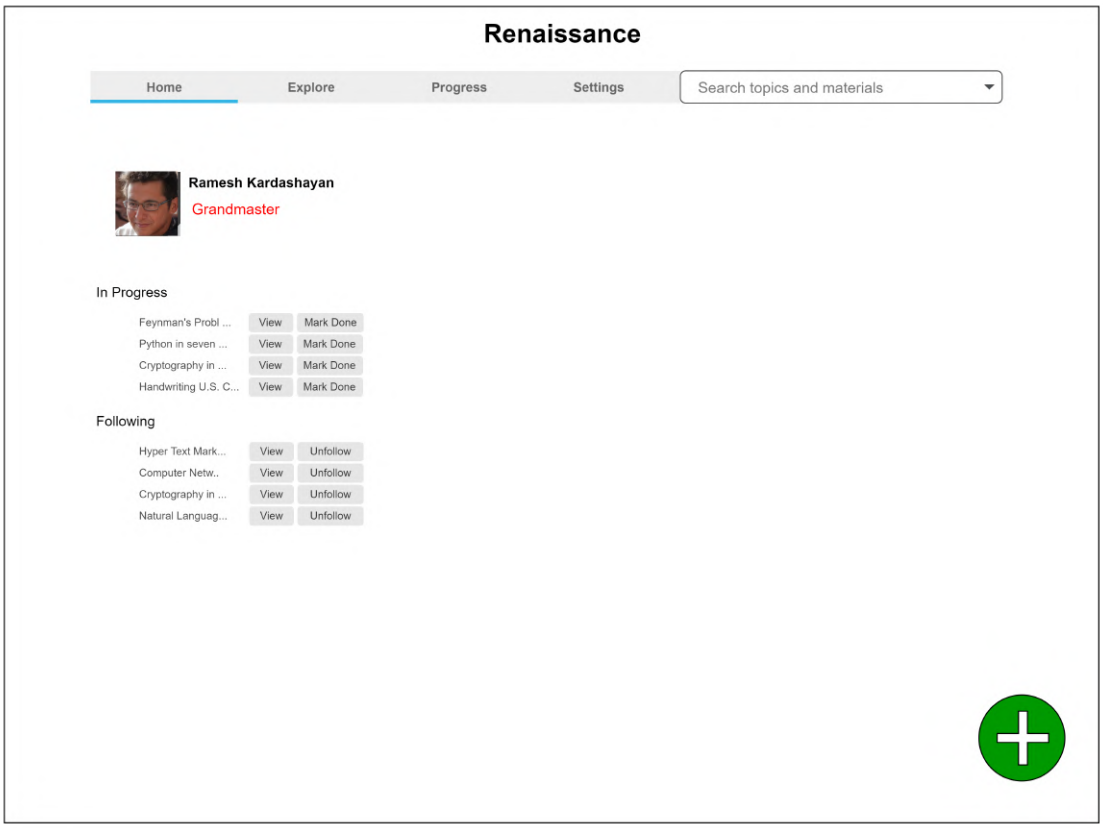
In reference to the Revision 1.1 of the requirements

- **(Requirement 1.1.2.)** The System shall provide its users access to it by means of specialized accounts.
- **(Requirement 1.2.4.)** The System shall supply its users with recommendation of materials.
- **(Requirement 1.2.7.)** The users should be able to reliably record topics they want to improve on.
- **(Requirement 1.3.4.)** The users should be able to share their comments on specific topics.
- **(Requirement 2.1.1.)** An Android Application and a Web Application shall be implemented as instances of the Interface.

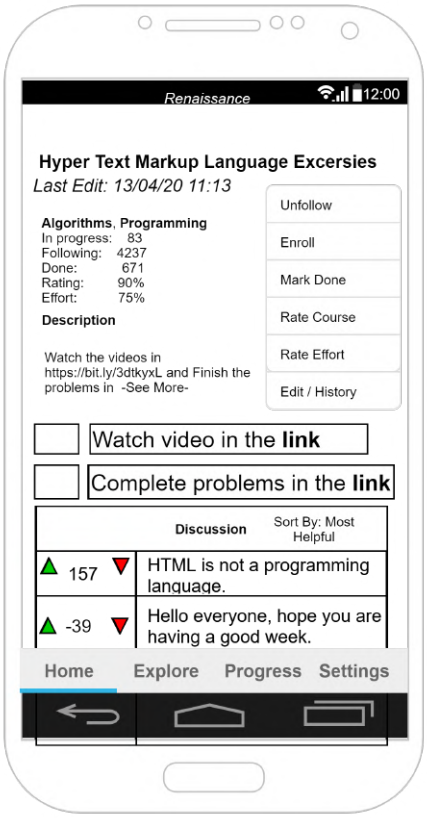
Actions & Mockups

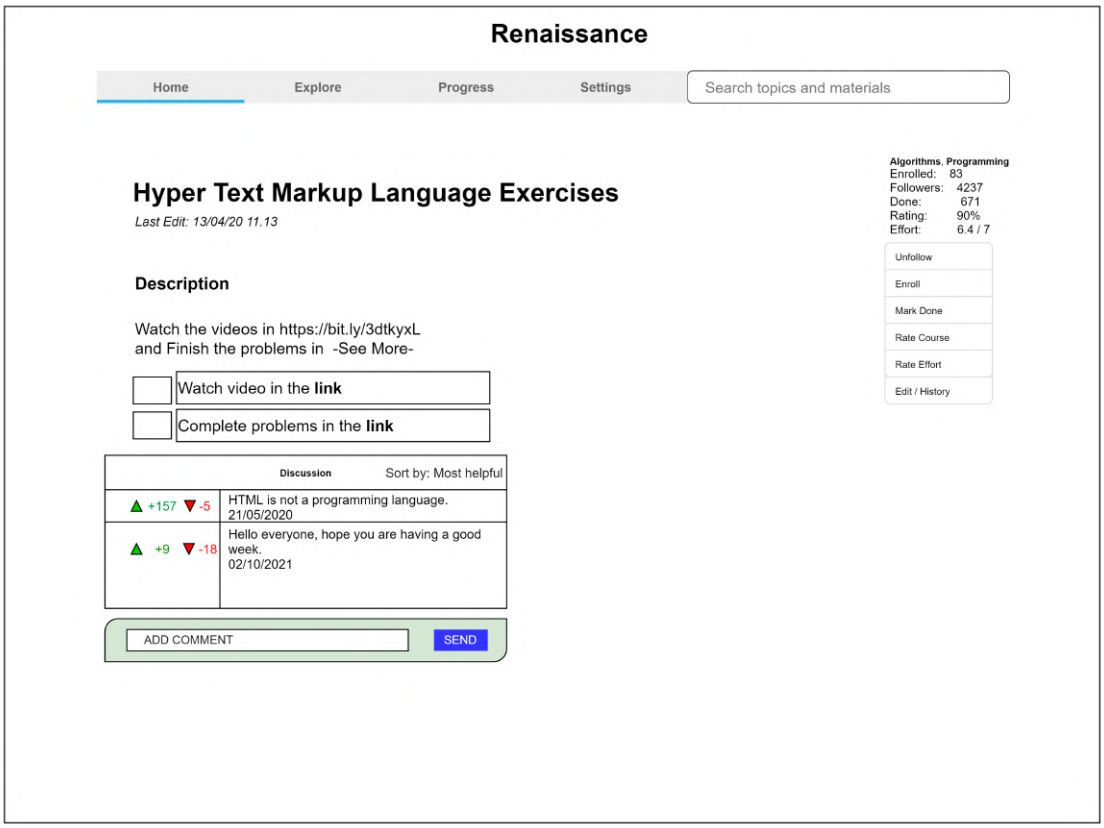
1. Ramesh opens his app and home page appears (Figure 1). He wonders what's new in his following content Hyper Text Markup Language Exercises and clicks the View Button.



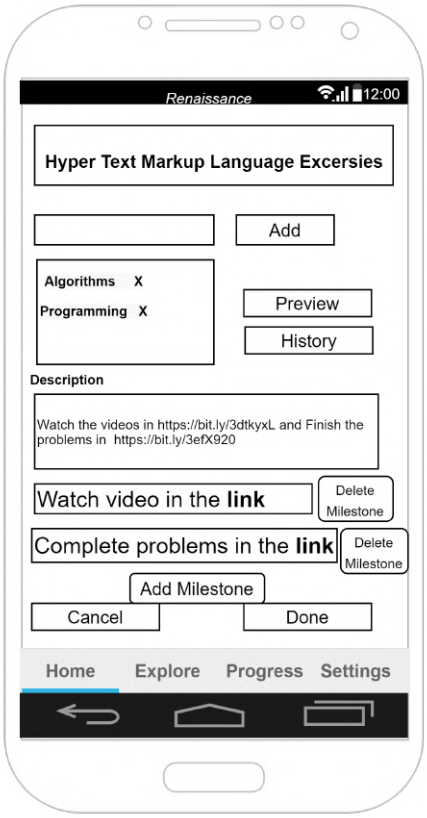


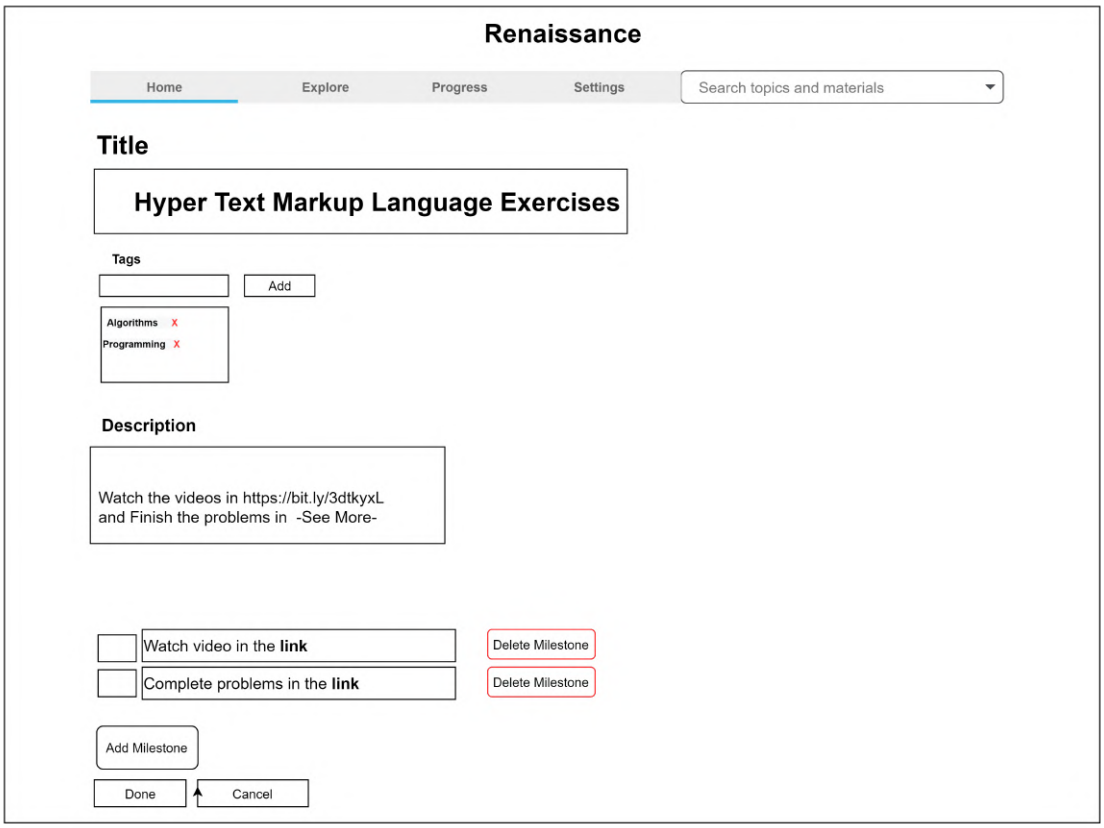
2. Hyper Text Markup Language Exercises page appears and he noticed that related topics of this page are wrong. So, he decides to change the related topics and press the Edit/History button.

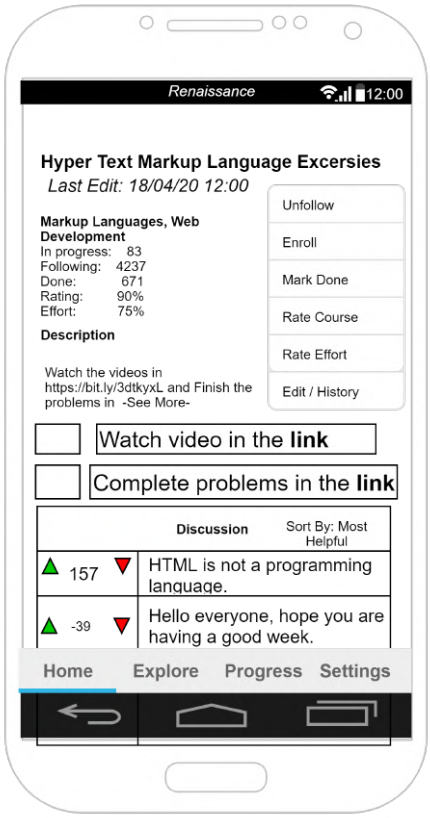




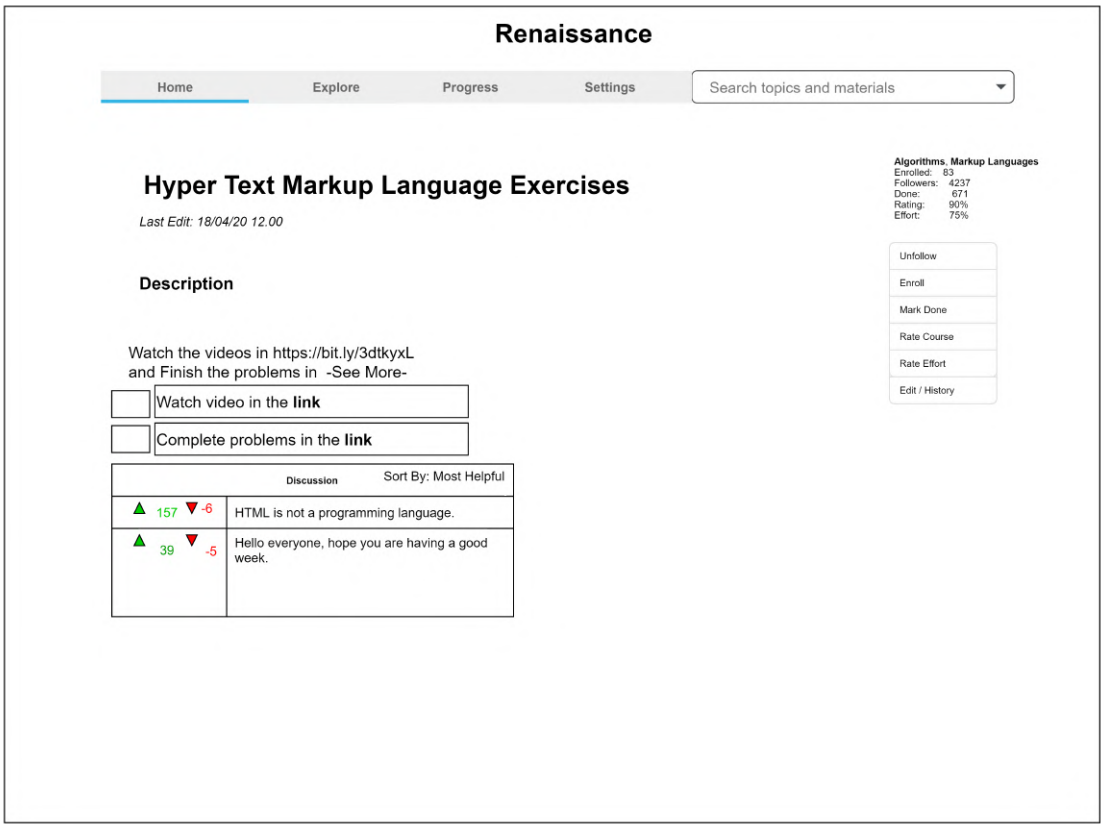
3. Edit page appears. He can add/delete topics, change the description or header. He deletes the related topics and add the new ones. Then press the Done Button.


Renaissance
12:00
Hyper Text Markup Language Excersies
Add
Algorithms X
Programming X
Preview
History
Description
Watch the videos in https://bit.ly/3dtkyxL and Finish the problems in https://bit.ly/3efX920
Watch video in the link
Delete Milestone
Complete problems in the link
Delete Milestone
Add Milestone
Cancel
Done
Home
Explore
Progress
Settings


Renaissance
Home
Explore
Progress
Settings
Search topics and materials
Title
Hyper Text Markup Language Exercises
Tags
Add
Algorithms X
Programming X
Description
Watch the videos in https://bit.ly/3dtkyxL
and Finish the problems in -See More-
Watch video in the link
Delete Milestone
Complete problems in the link
Delete Milestone
Add Milestone
Done
Cancel



4. Related topics and the last edit date changed.



Scenario 3

User Persona

- **Name:** Tom Marvolo Riddle
- **Age:** 16

- **Job:** Student
- **Interest:** Dark Magic, Magic, Duelling

Preconditions

- Tom is in the web app.
- Tom has not any account and is not signed up.

Goals

- To become immortal. (to find a material about a topic he is interested in)
- To find out about how to create a crucifix to ensure that.

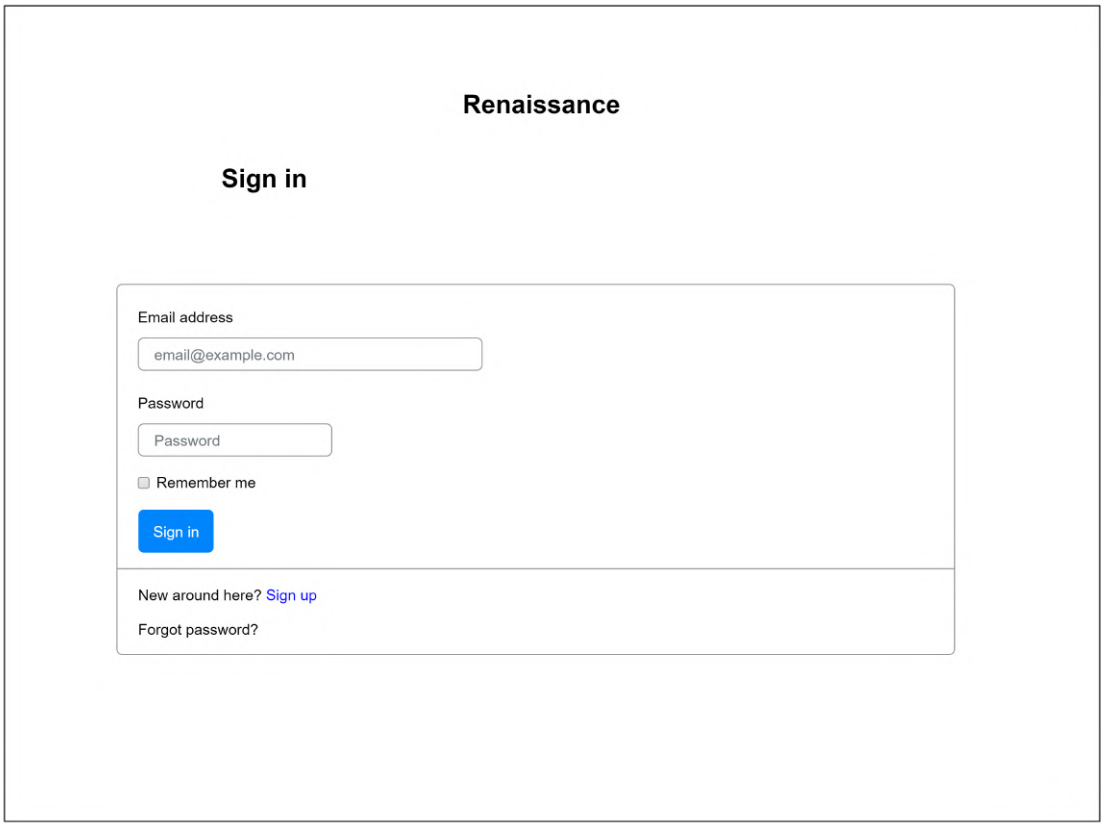
Acceptance Criteria

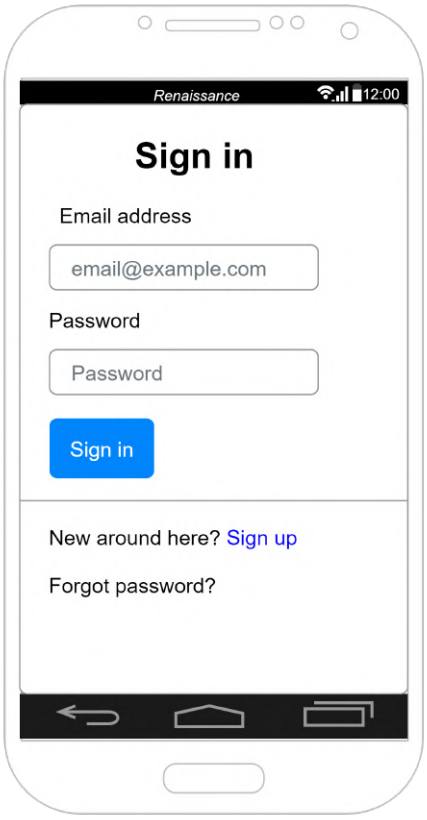
In reference to the Revision 1.1 of the requirements

- **(Requirement 1.1.2.)** The System shall provide its users access to it by means of specialized accounts.
- **(Requirement 1.1.3.)** The users shall be able to create and delete their accounts and be able to gain access if they prove identity in case they forget their password.
- **(Requirement 1.2.7.)** The users should be able to reliably record topics they want to improve on.
- **(Requirement 1.4.3.)** The users should be able to search data assigned to topics using keywords.
- **(Requirement 2.1.1.)** An Android Application and a Web Application shall be implemented as instances of the Interface.
- **(Requirement 2.1.3.)** The System shall express topics using semantic representation as defined in Wikidata.

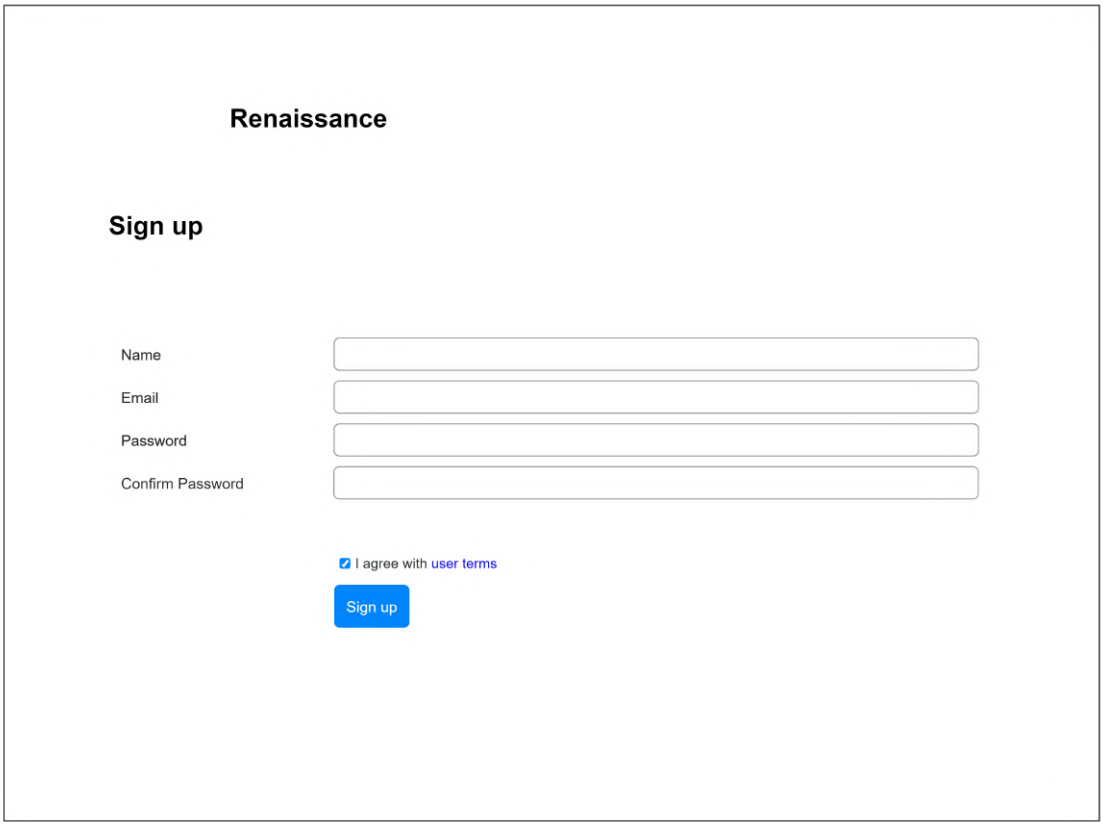
Actions & Mockups

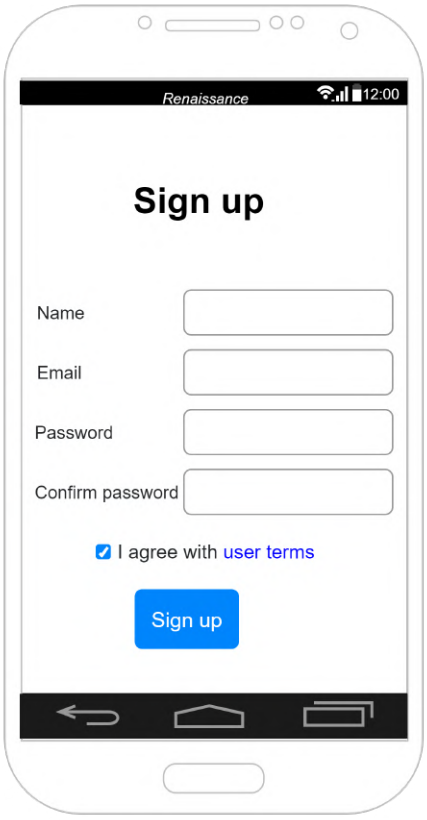
1. He opens the site.



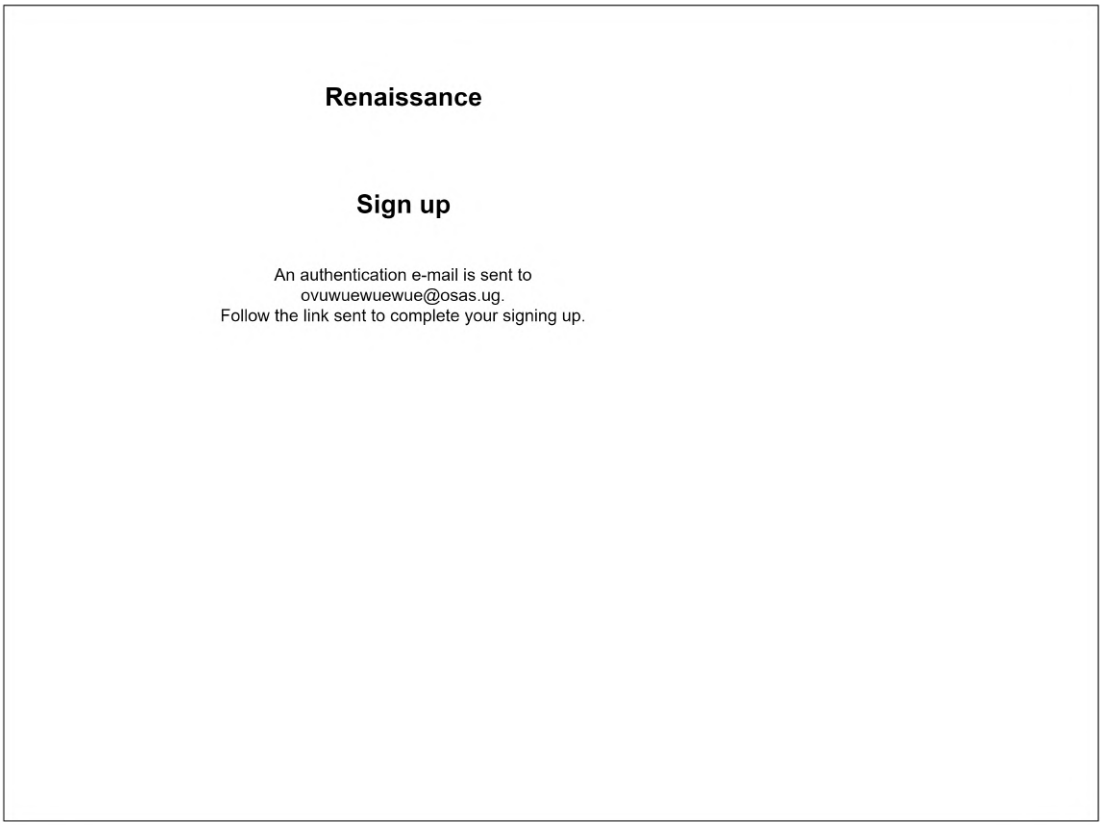


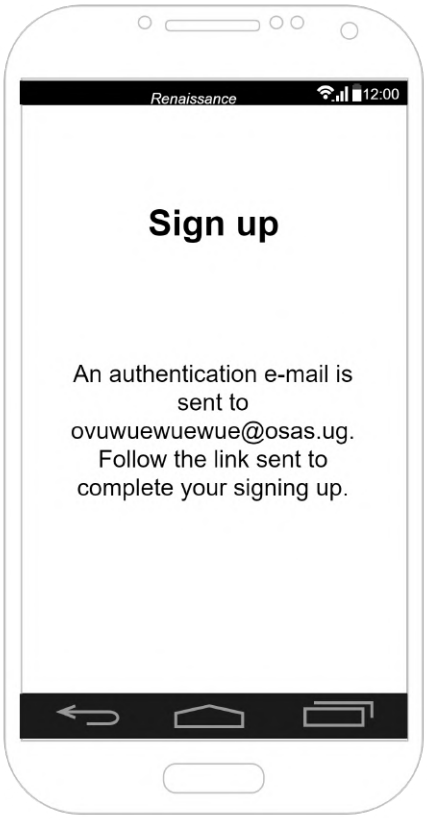
2. He realizes that he does not have an account. He clicks the sign up button then redirected to sign up page.



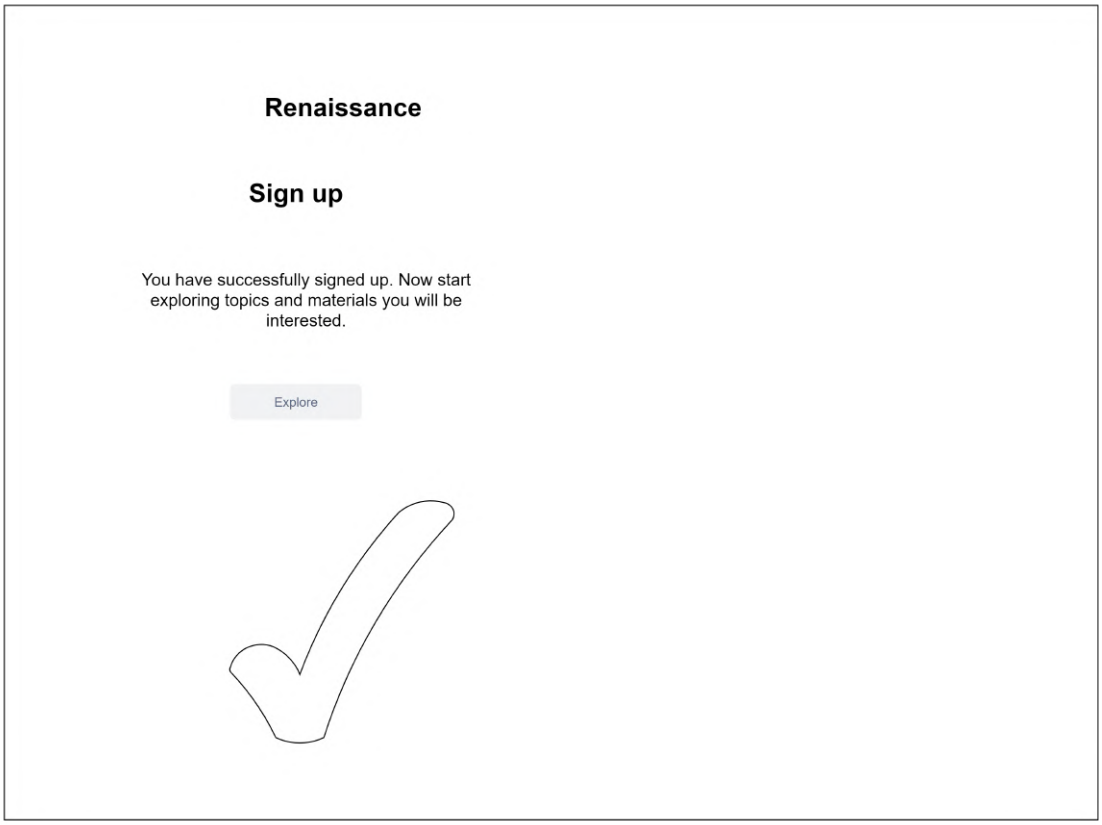


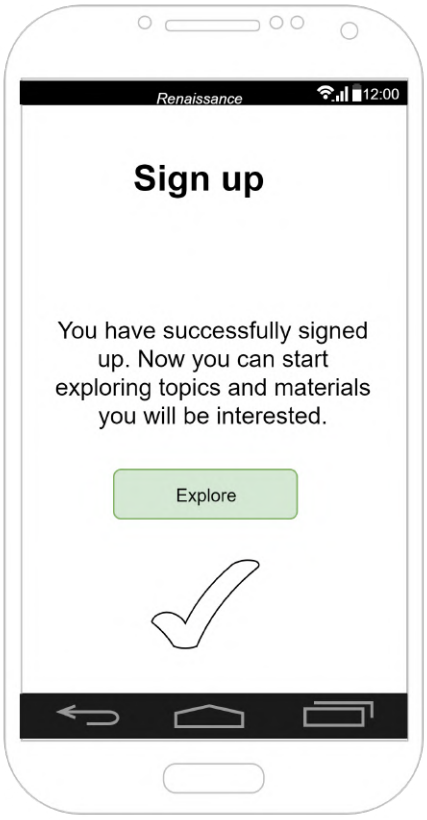
3. He provides the required information and by clicking the sign-up button he is redirected to the e-mail confirmation page.



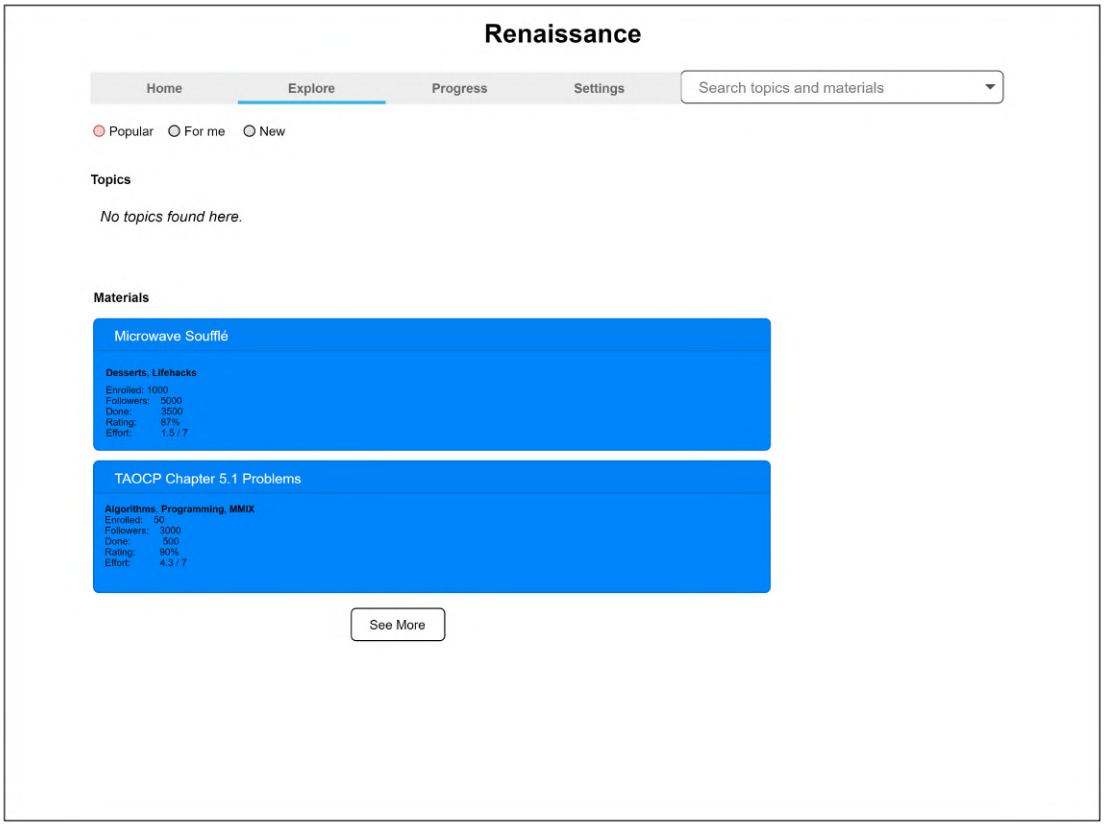


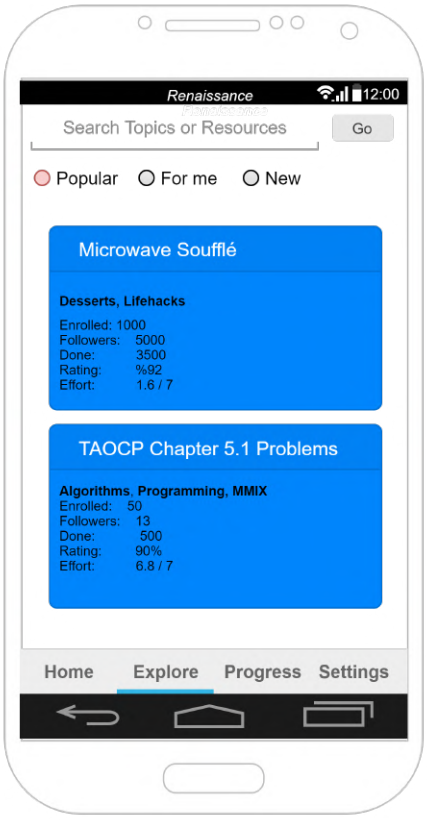
4. After confirming his e-mail he is redirected to the following page.



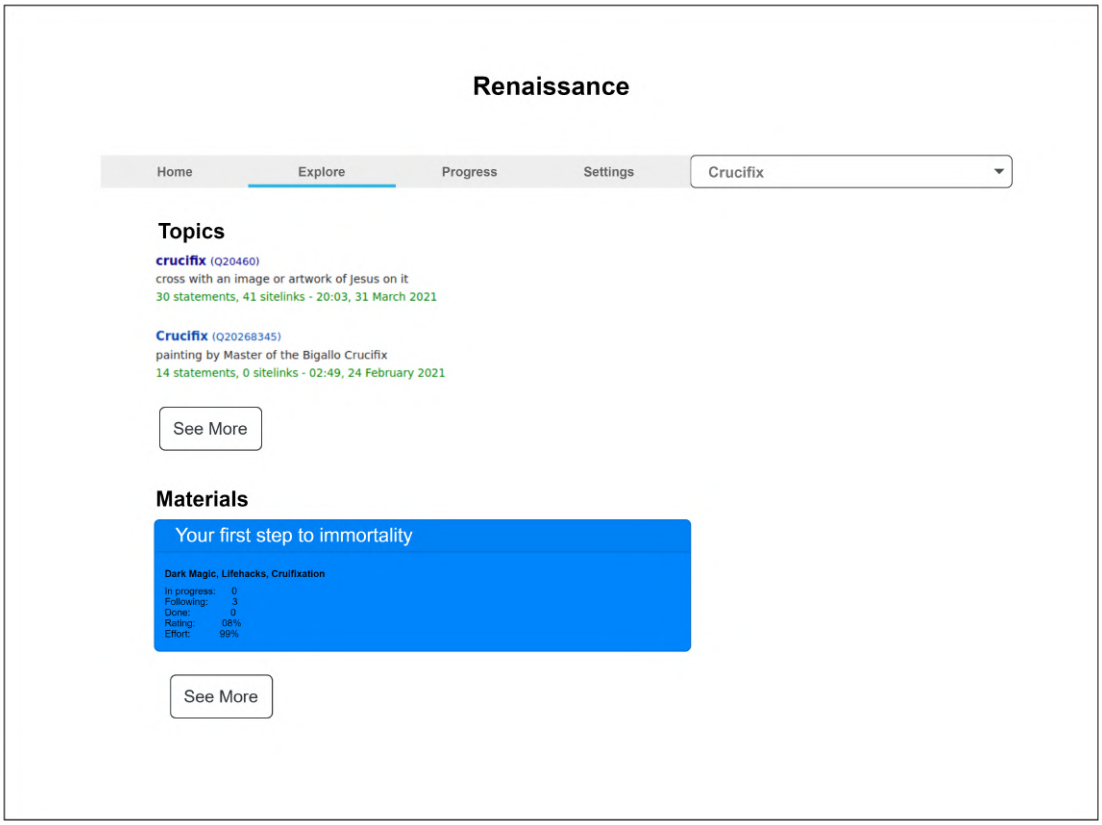


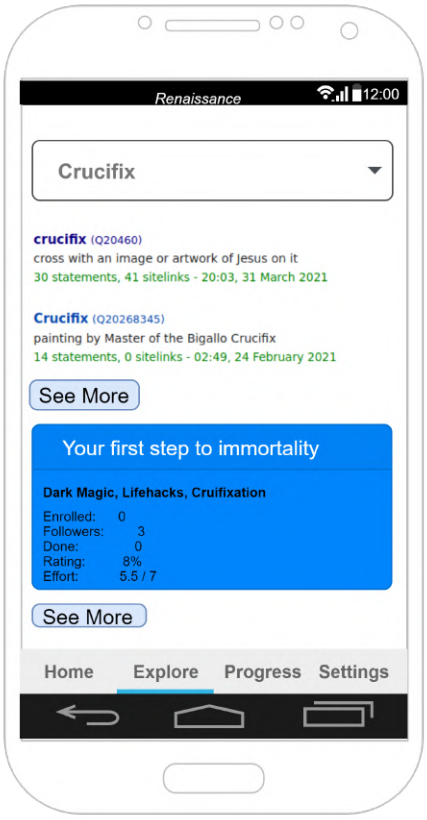
5. He does not have any topics or material followed in his account. (Figure 5)



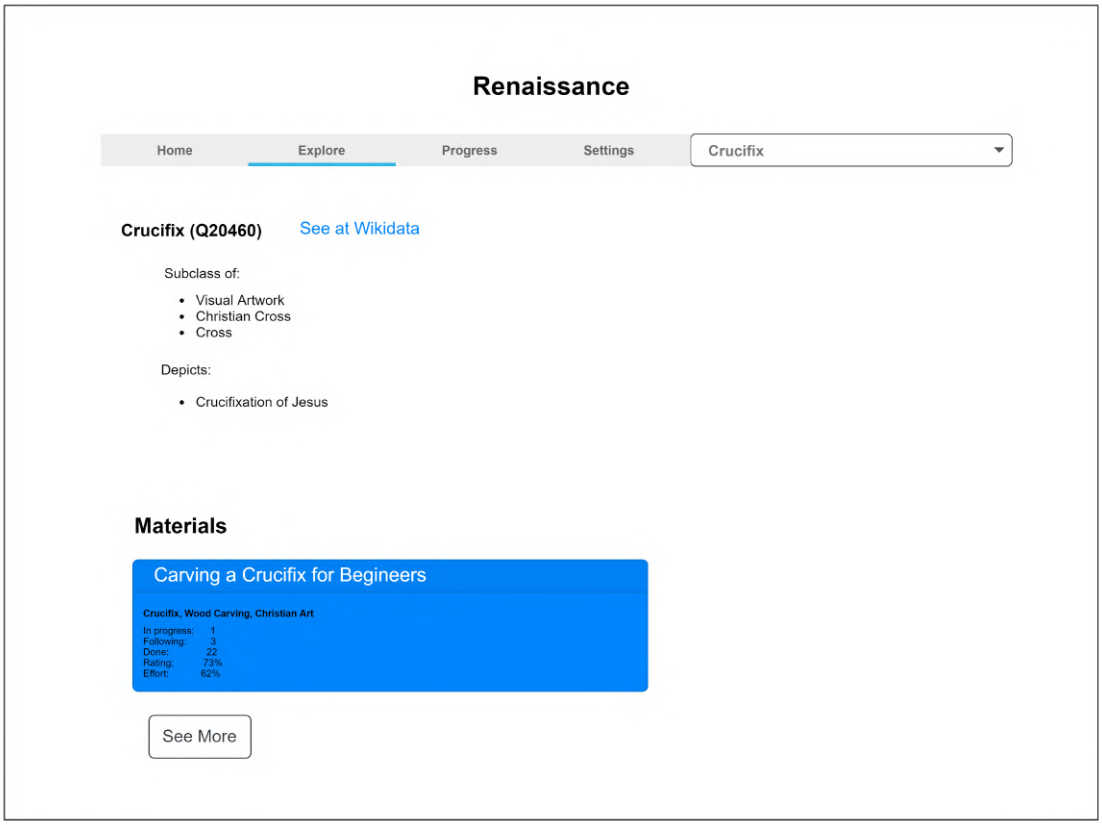


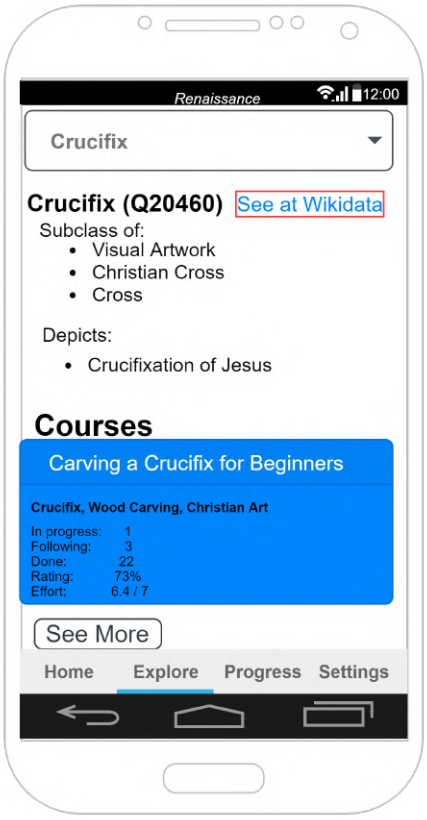
6. He searches for crucifix on the search bar on top, and he is redirected to the page in Figure 6. (Figure 6)



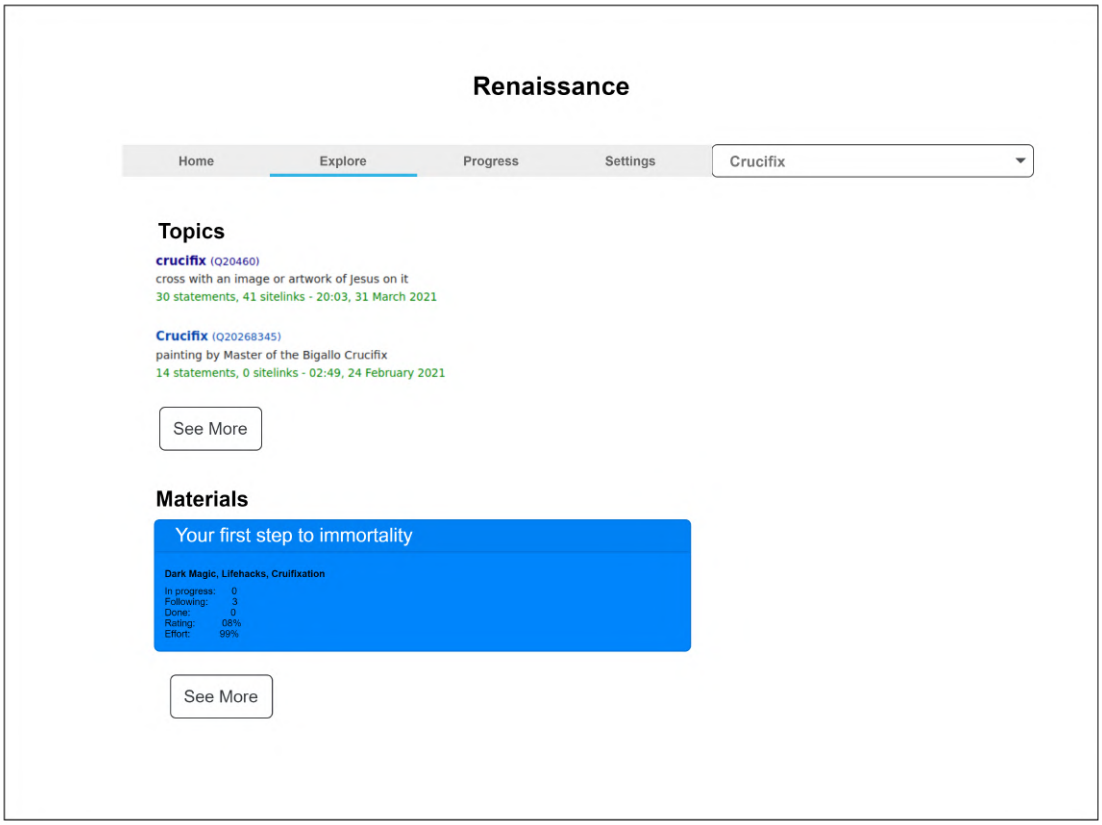


7. He clicks the first topic result and he is prompted to the page in Figure 7, which he is not satisfied with. (Figure 7)



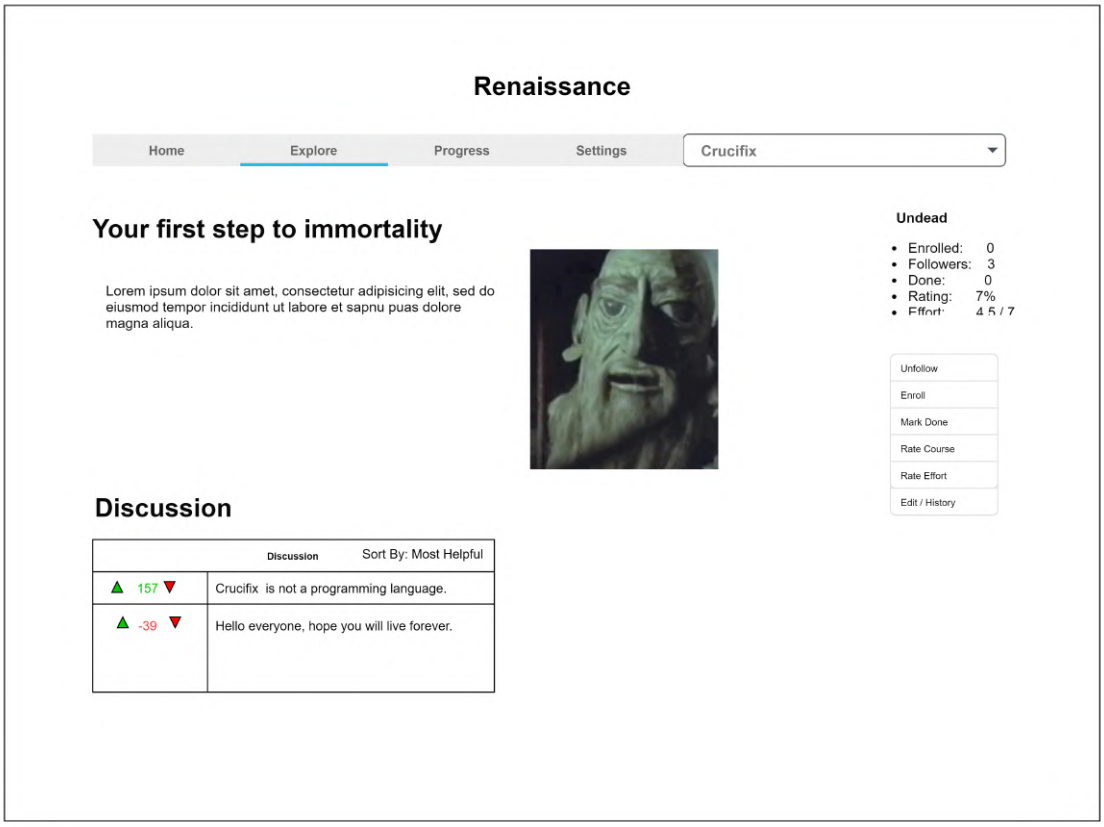


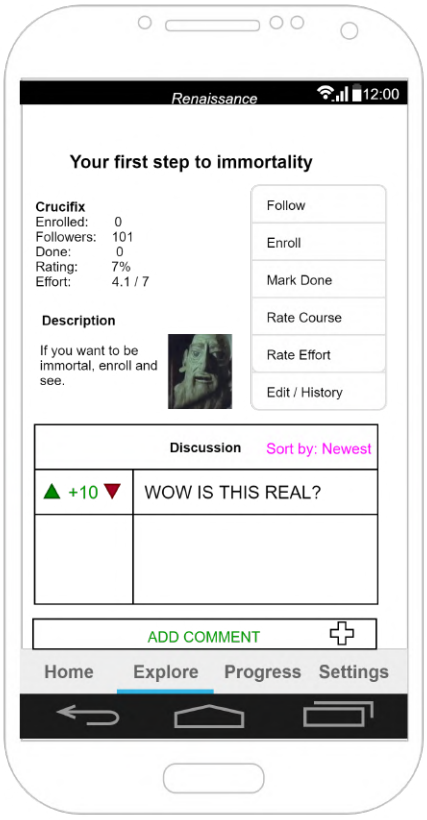
8. So he goes back to the search results, where he sees the material in the bottom, which he finds interesting.


Renaissance
Home
Explore
Progress
Settings
Crucifix
Topics
crucifix (Q20460)
cross with an image or artwork of Jesus on it
30 statements, 41 sitelinks - 20:03, 31 March 2021
Crucifix (Q20268345)
painting by Master of the Bigallo Crucifix
14 statements, 0 sitelinks - 02:49, 24 February 2021
See More
Materials
Your first step to immortality
Dark Magic, Lifehacks, Cruifixation
In progress: 0
Following: 3
Done: 0
Rating: 08%
Effort: 99%
See More

Renaissance
12:00
Crucifix
Topics
crucifix (Q20460)
cross with an image or artwork of Jesus on it
30 statements, 41 sitelinks - 20:03, 31 March 2021
Crucifix (Q20268345)
painting by Master of the Bigallo Crucifix
14 statements, 0 sitelinks - 02:49, 24 February 2021
See More
Courses
Your first step to immortality
Dark Magic, Lifehacks, Cruifixation
Enrolled: 0
Followers: 3
Done: 0
Rating: 8%
Effort: 5.1 / 7
See More
Home
Explore
Progress
Settings

9.

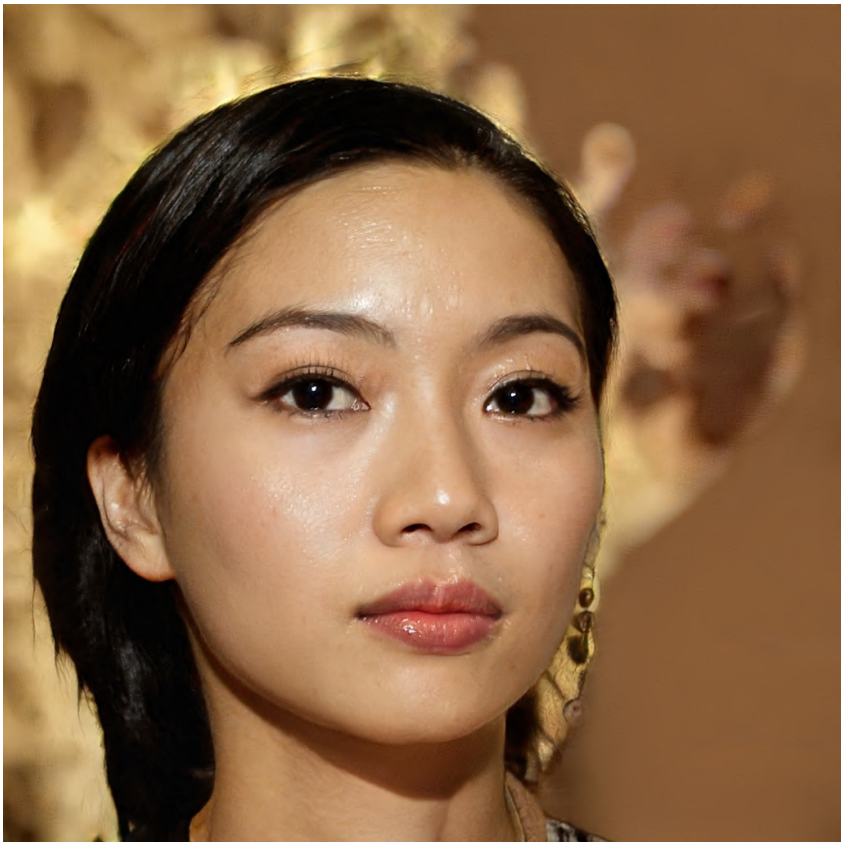




Scenario 4

User Persona

- Photo:



- **Name:** Yolanda Graver
- **Age:** 35
- **Job:** Baker
- **Interest:** Baking, Boxing, Cleaning

Preconditions

- Yolanda is already signed up and logged in the android app.

Goals

- She wants to create a new resource including a path.

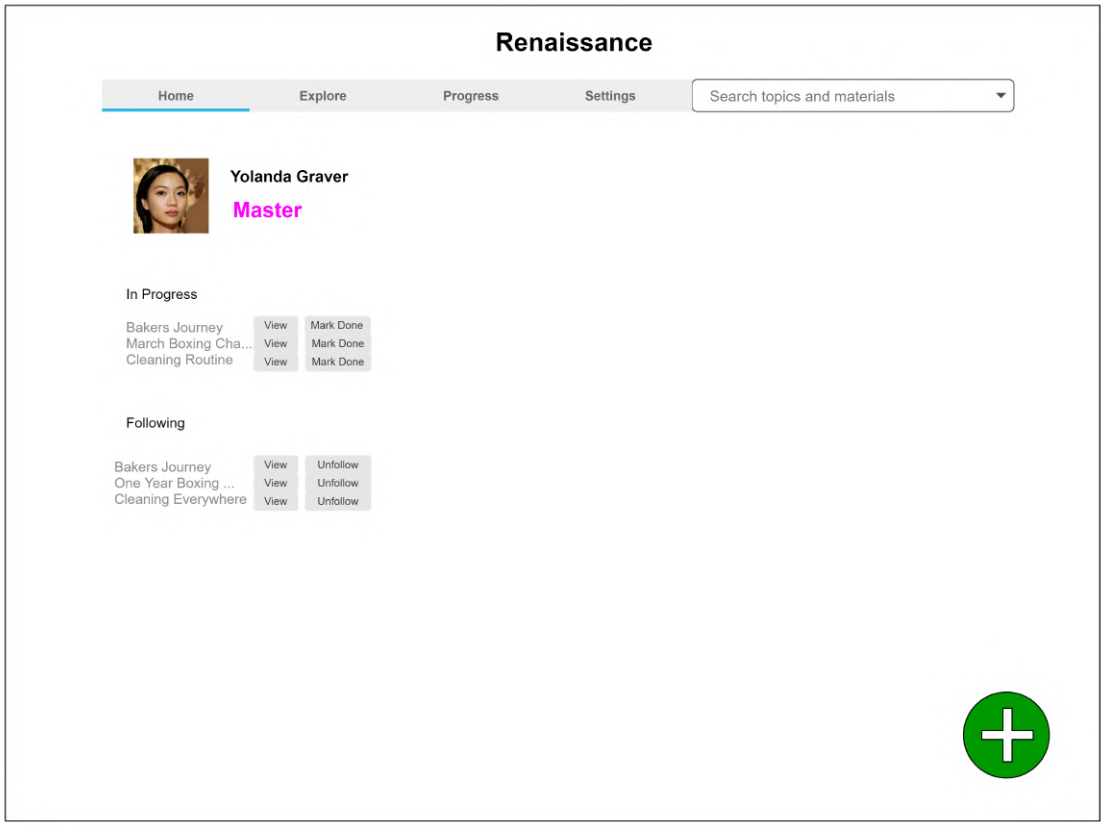
Acceptance Criteria

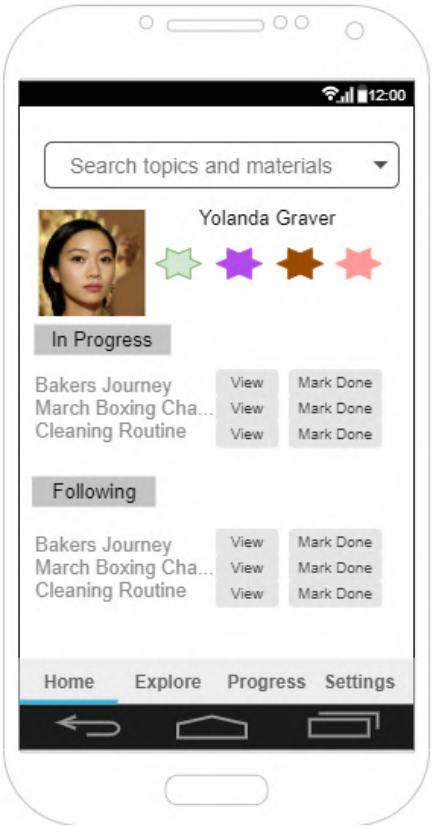
In reference to the Revision 1.1 of the requirements

- **(Requirement 1.1.2.)** The System shall provide its users access to it by means of specialized accounts.
- **(Requirement 1.2.6.)** The users shall be able to reliably record and edit some goals they determine as a set of material-deadline pairs.
- **(Requirement 1.2.7.)** The users should be able to reliably record some topics they want to improve on.
- **(Requirement 2.1.1.)** An Android Application and a Web Application shall be implemented as instances of the Interface.

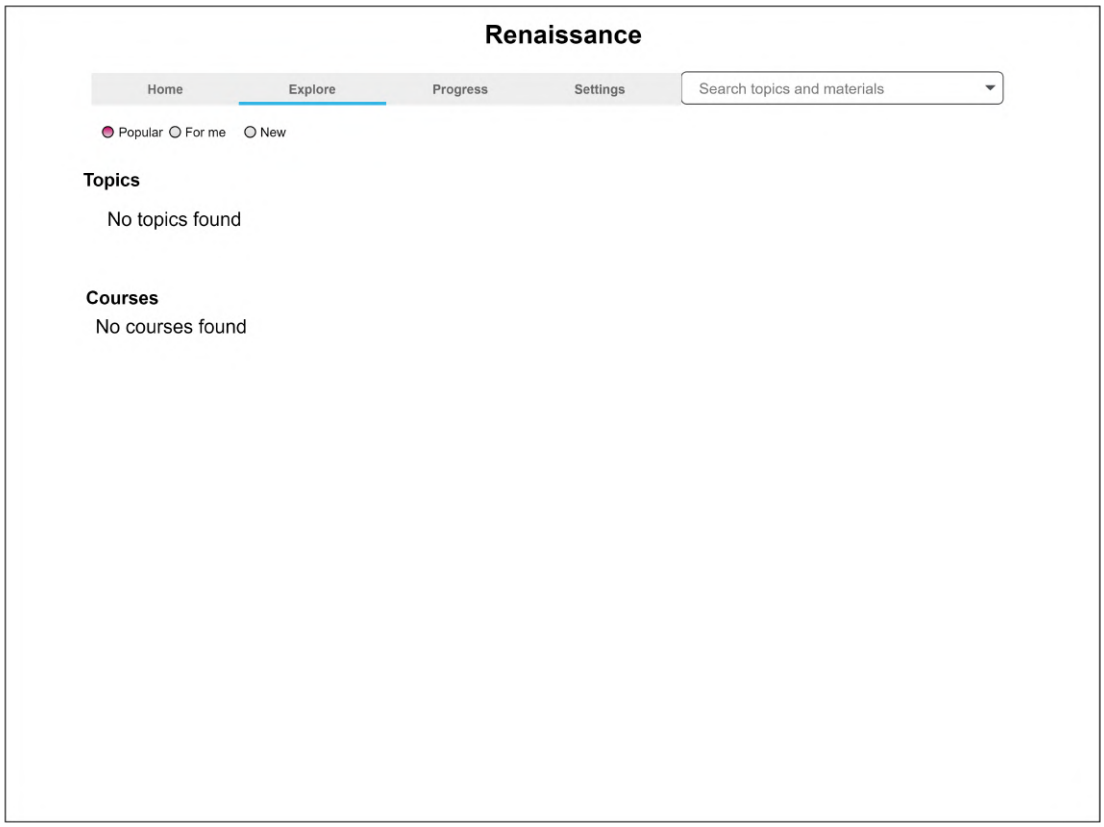
Actions & Mockups

1. Yolanda opens his app and home page appears (Figure 1). She wants to create a new resource to share her mouse shaped cake recipe.

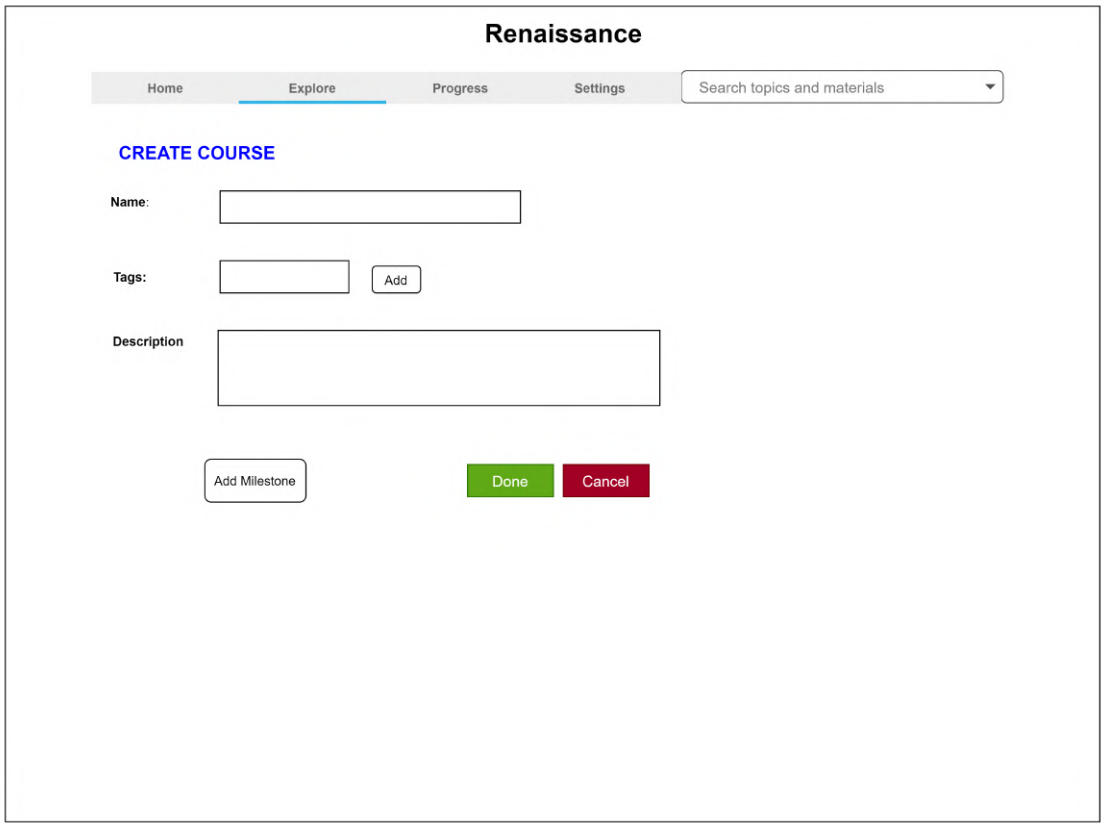


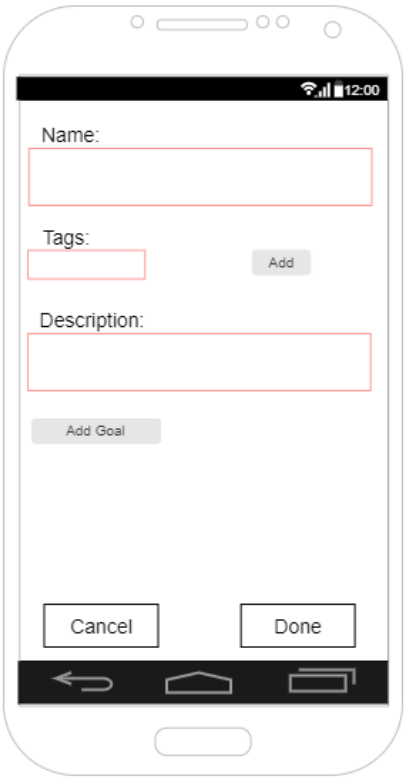


2. She goes to explore page. (Figure 2) Then she clicks on the "Create" button.

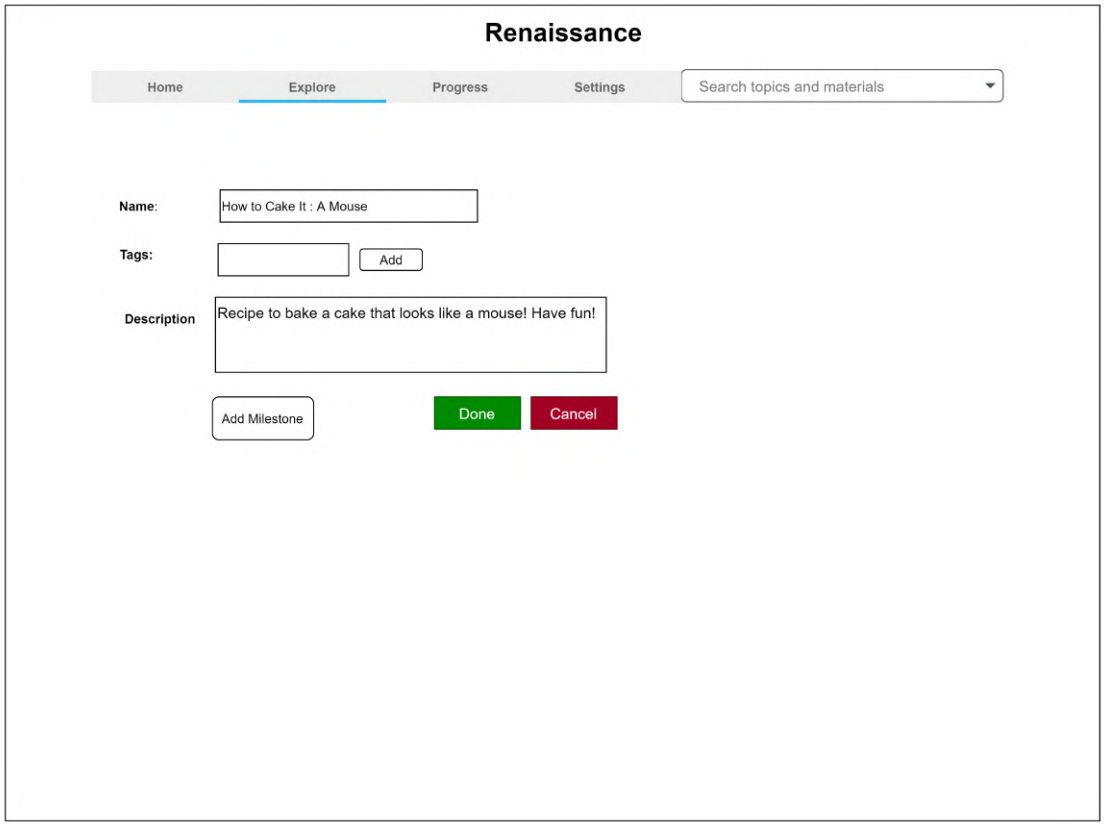


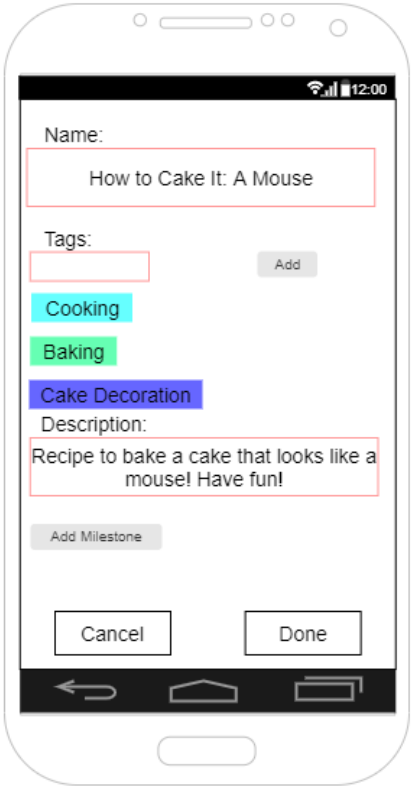
3. Edit page appears. (Figure 3)



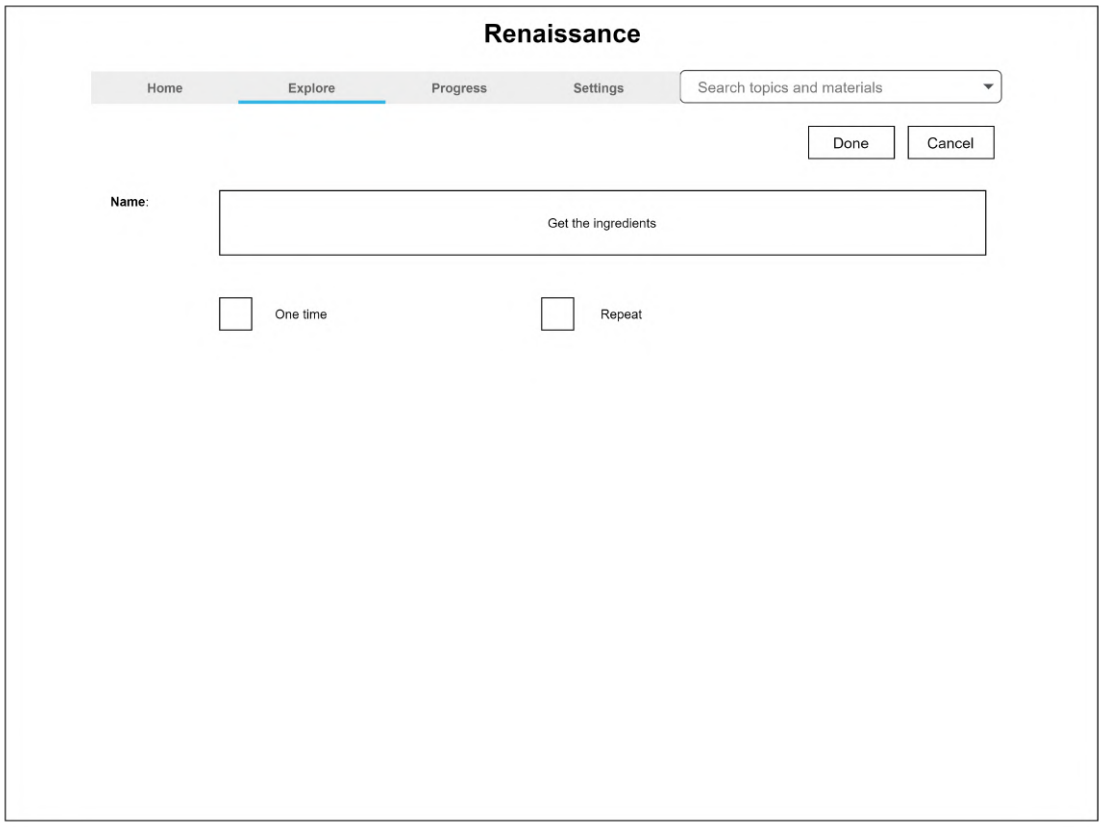


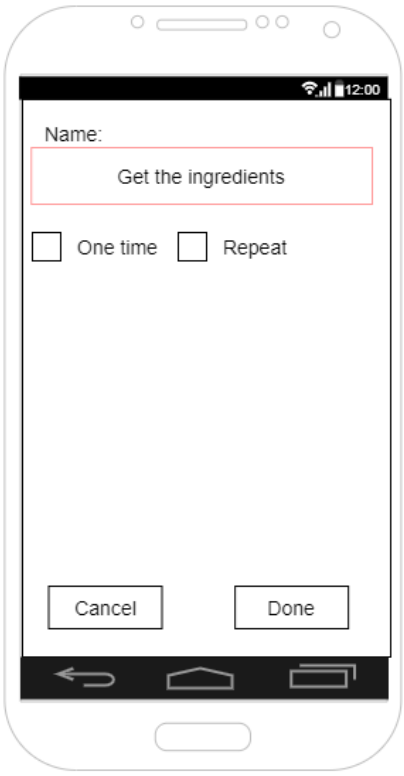
4. She names the resource , adds tags and description. Then she clicks on to the "Add Milestone" button. (Figure -4)


Renaissance
Home
Explore
Progress
Settings
Search topics and materials
Name:
How to Cake It : A Mouse
Tags:
Add
Description
Recipe to bake a cake that looks like a mouse! Have fun!
Add Milestone
Done
Cancel

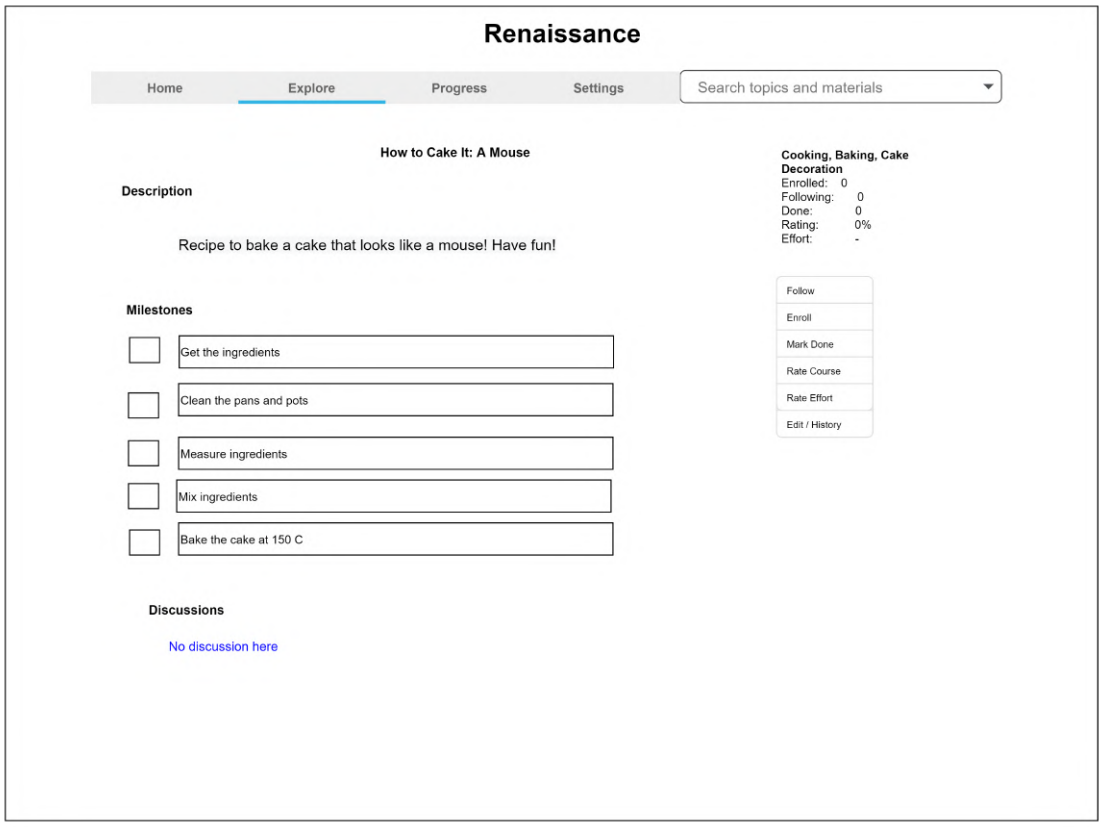

12:00
Name:
How to Cake It: A Mouse
Tags:
Add
Cooking
Baking
Cake Decoration
Description:
Recipe to bake a cake that looks like a mouse! Have fun!
Add Milestone
Cancel
Done

5. She adds the new milestone, sets it recurring type and clicks done. (Figure-5)





6. Final resource page is as shown below.



Scenario 5

User Persona

- **Photo:**
- **Name:** John Jones
- **Age:** 50
- **Job:** Teacher
- **Interest:** Dance, Group Activities

Preconditions

- John is already signed up and logged in to the android and web app.
- The app is started at in-progress goals.

Goals

- John tries to learn dancing. He wants his friend to learn dancing, so they can dance after the pandemic together.

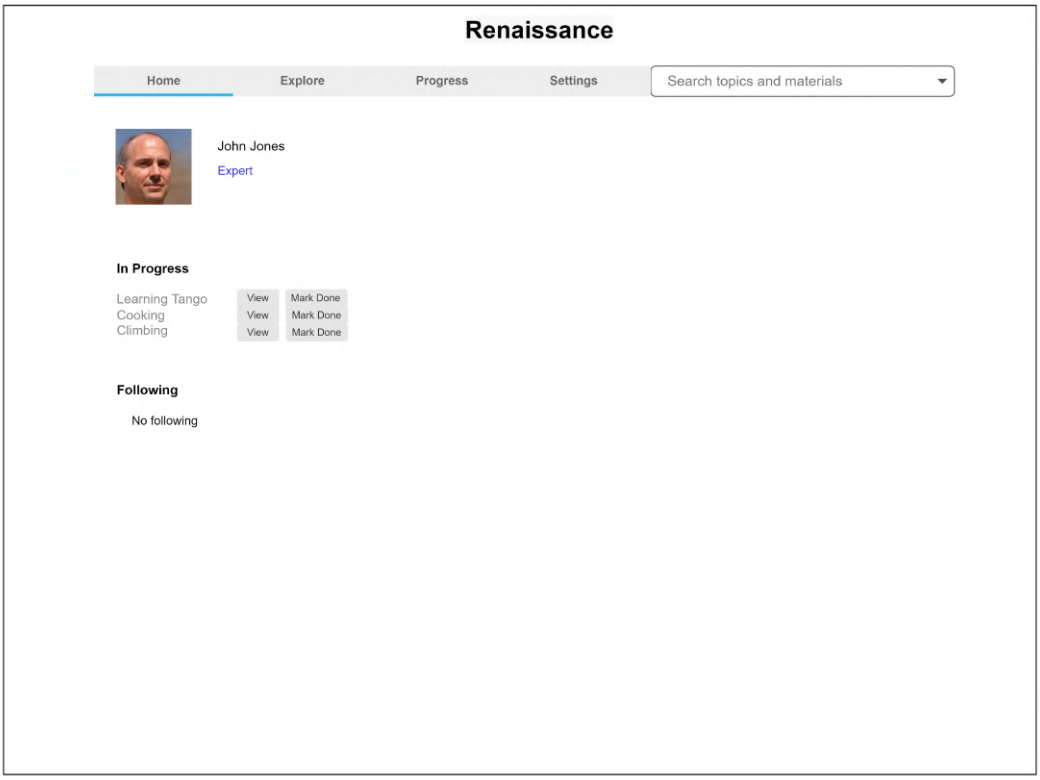
Acceptance Criteria

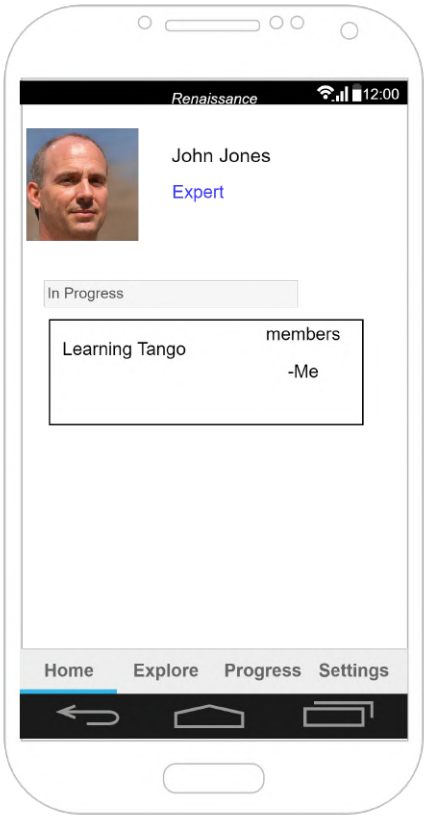
In reference to Revision 1.1 of the requirements

- **(Requirement 1.2.8.)** The users *shall* be able to work on their determined goals as a group.
- **(Requirement 1.3.4.)** The users *shall* be able to search for other users.
- **(Requirement 2.1.1.)** An Android Application and a Web Application shall be implemented as instances of the Interface.

Actions & Mockups

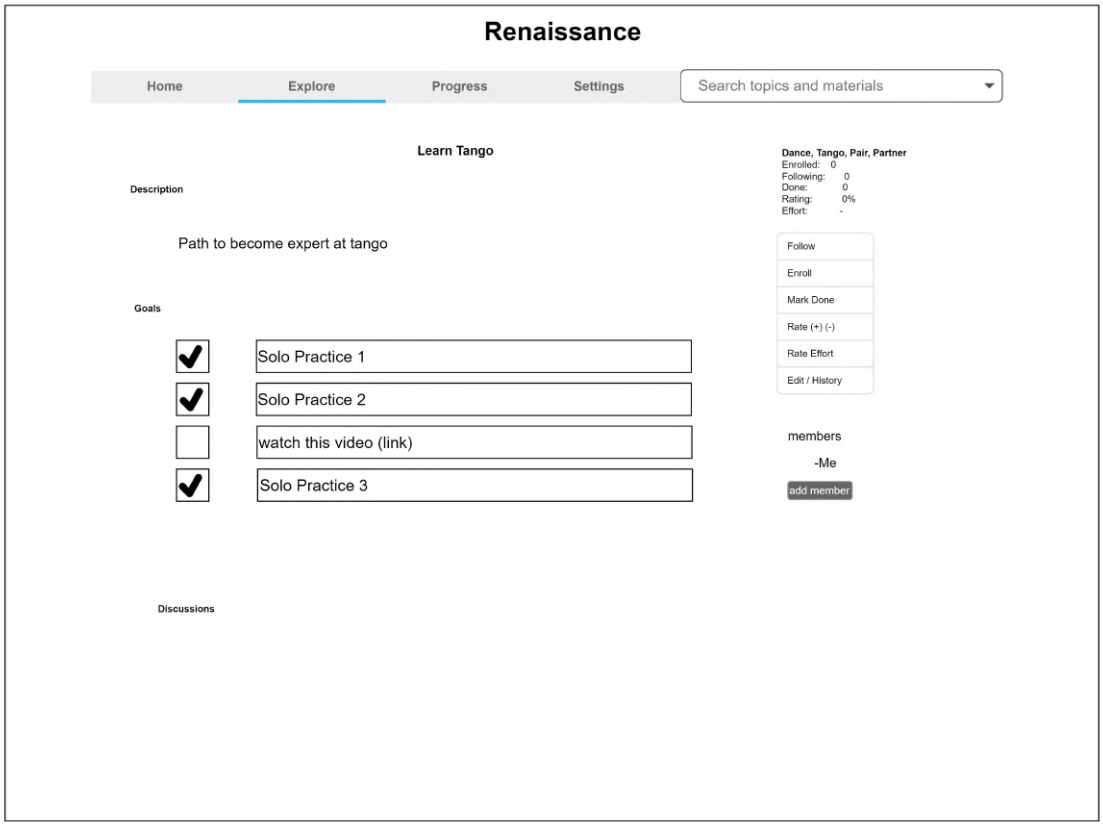
1. John opens the app and sees in-progress goals, and opens one relating to tango.

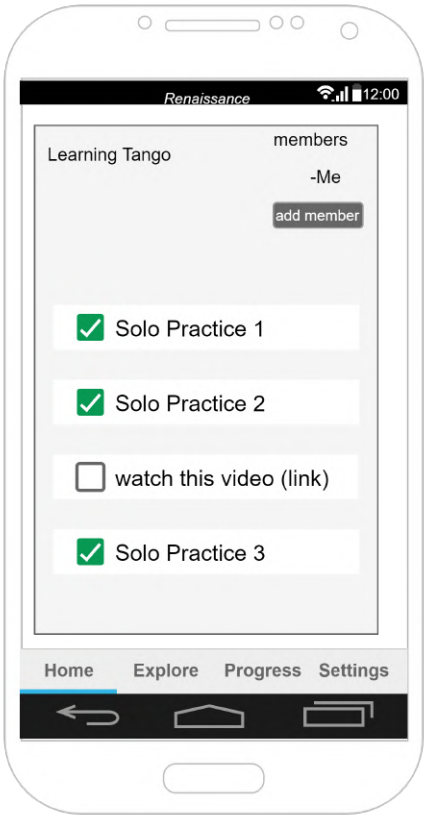




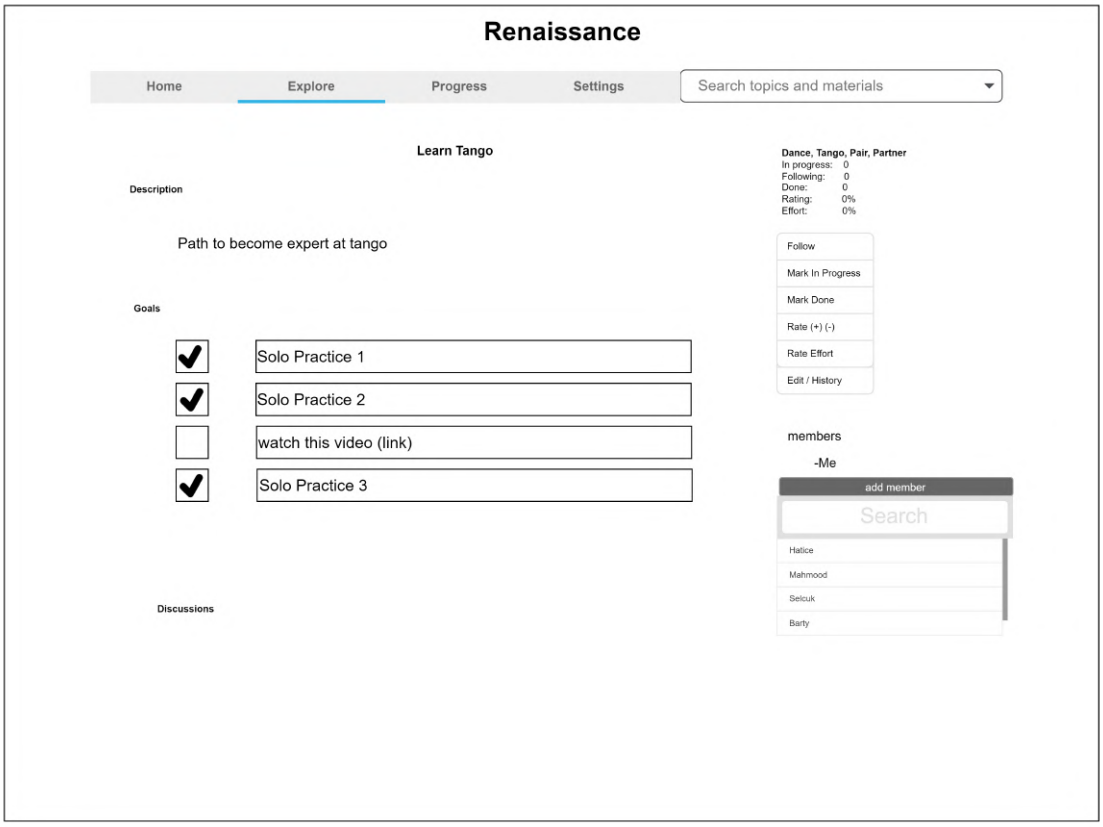
2. He presses add member button to add his friend Hatice

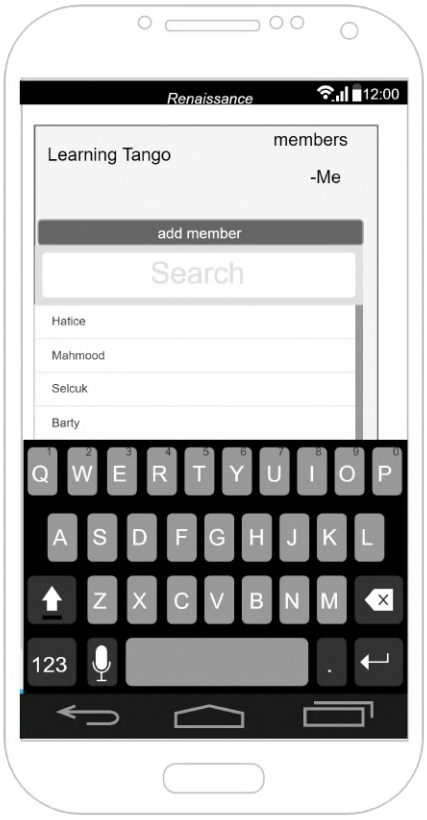
2



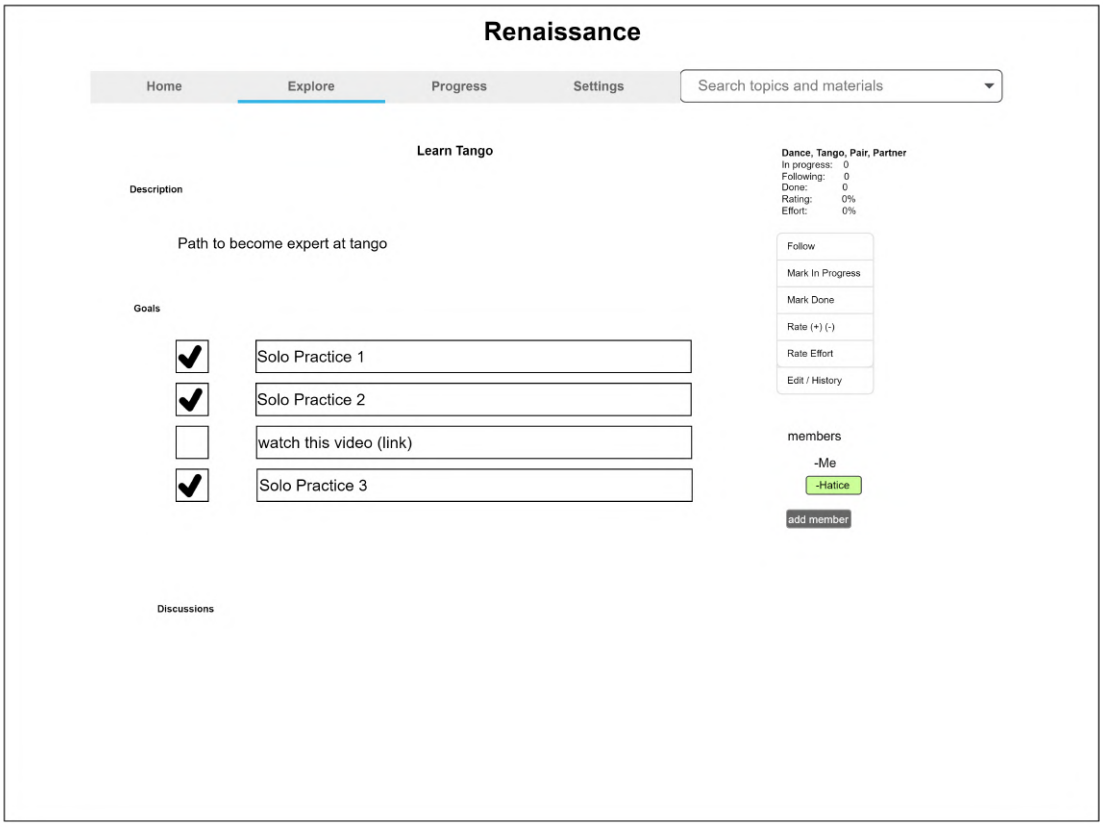


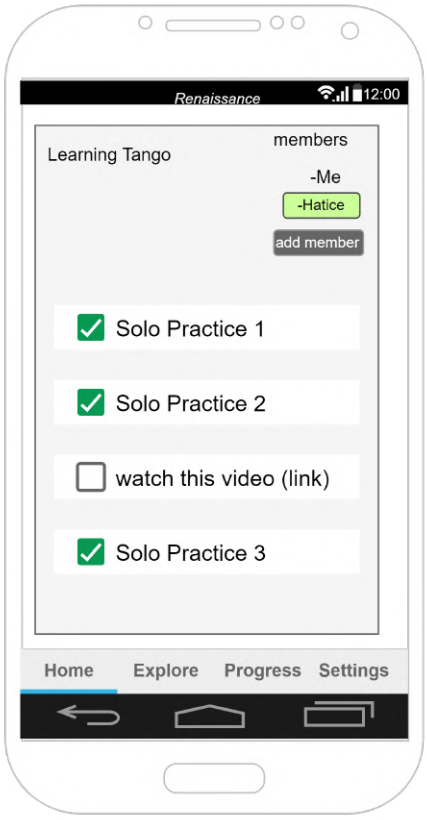
3. He searches, finds the friend and selects it.



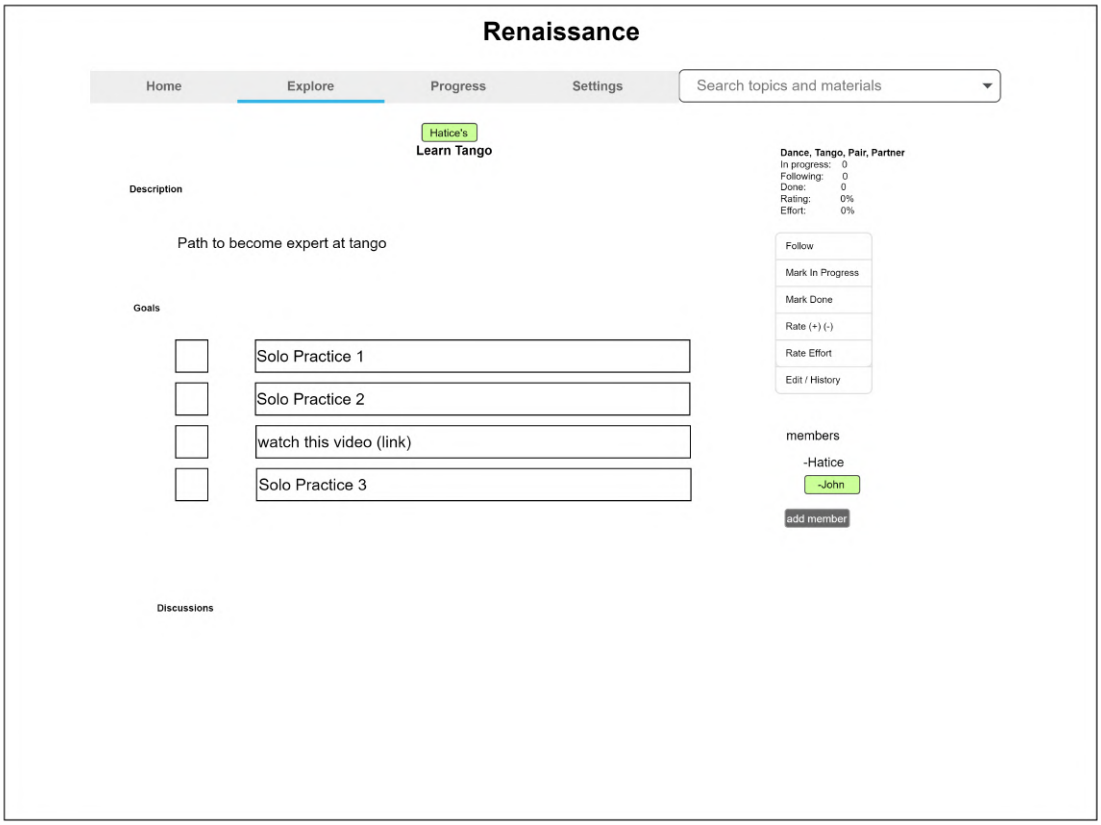


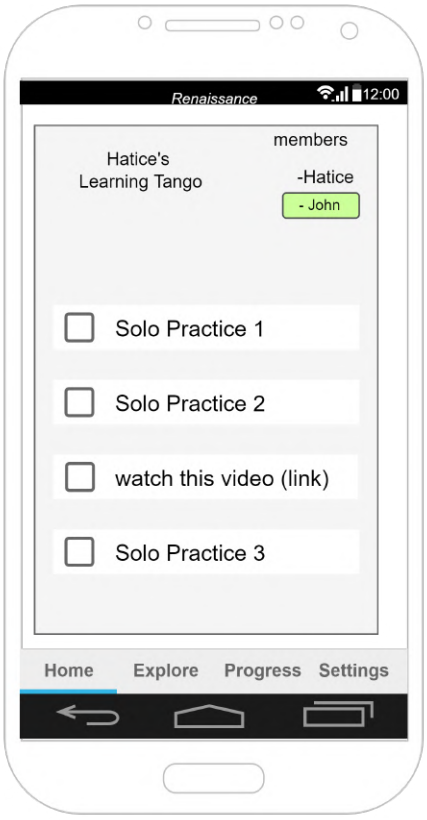
4. The new member added to this path, he can see members now.





5. He goes to his friend's path to see her progress


Renaissance
Home
Explore
Progress
Settings
Search topics and materials
Hatice's
Learn Tango
Description
Path to become expert at tango
Goals
Solo Practice 1
Solo Practice 2
watch this video (link)
Solo Practice 3
Discussions
Dance, Tango, Pair, Partner
In progress: 0
Following: 0
Done: 0
Rating: 0%
Effort: 0%
Follow
Mark In Progress
Mark Done
Rate (+) (-)
Rate Effort
Edit / History
members
-Hatice
-John
add member


Renaissance
12:00
Hatice's
Learning Tango
members
-Hatice
- John
Solo Practice 1
Solo Practice 2
watch this video (link)
Solo Practice 3
Home
Explore
Progress
Settings