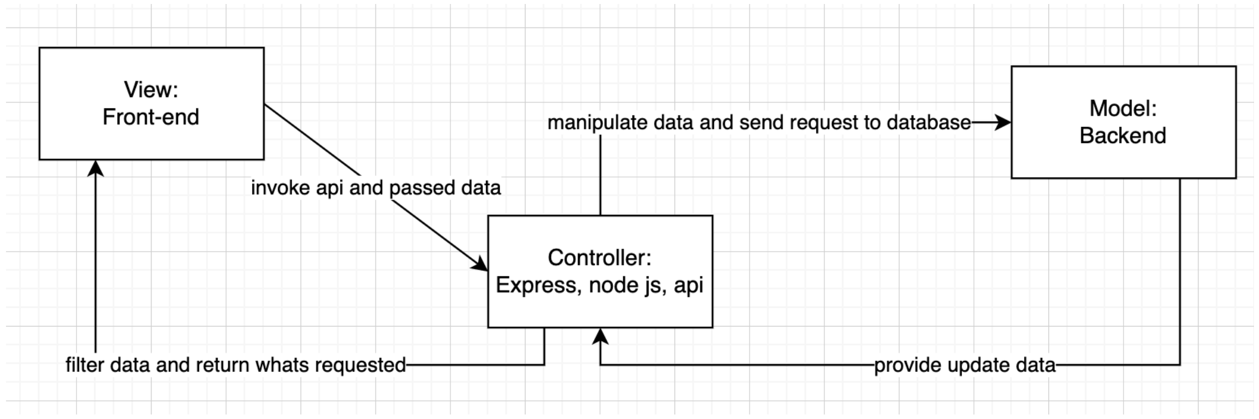
System Design Document

Table Of Content

Software Architecture Diagram


View:
Front-end
Model:
Backend
Controller:
Express, node js, api
manipulate data and send request to database
invoke api and passed data
filter data and return whats requested
provide update data

Our architecture design follows mvc pattern such that the front-end will invoke api and pass data to the controller which will manipulate data and send it to our back end database. The backend would then return updated data that the controller requested. The controller would then update the front-end.

https://www.geeksforgeeks.org/mvc-design-pattern/

CRC Cards

Backend CRC cards

API name: question Url: GET http://127.0.0.1:8000/api/questions	
Required field: None	
Responsibilities: - Returns all the data related to each questions - Keeps tracks on the status of the question, which user are associated with it	Collaborators: - userID

API name: question Url: POST http://127.0.0.1:8000/api/questions	
Required field: Title, status, taID, userID, body, bounty, date	
Responsibilities: - Create and track all the given data under a new question	Collaborators:

API name: question Url: PUT http://127.0.0.1:8000/api/questions	
Required field: Question ID, and fields that needs to be change	
Responsibilities: - Find the questions with the given ID and update the data	Collaborators:

API name: question Url: DELETE http://127.0.0.1:8000/api/questions/:id:	
Required field: Question ID	
Responsibilities: - Delete the question associated with this ID - Return the question ID	Collaborators:

API name: setProfile Url: POST http://localhost:8000/api/profile	
Required field: Name:str ID:num	
Responsibilities: - Create a user with name and id	Collaborators: - userID

API name: getProfile Url: GET http://localhost:8000/api/profile/id:	
Required field: ID:num	
Responsibilities: - Get stored user information from id	Collaborators: - userID

API name: deleteProfile Url: DELETE http://localhost:8000/api/profile/id:	
Required field: ID:num	
Responsibilities: - Delete stored user information from id	Collaborators: - userID

API name: updateProfile Url: PUT http://localhost:8000/api/profile/:id	
Required field: Name:str Image: Buffer	
Responsibilities: - Delete stored user information from id - Change user name and profile picture in this case	Collaborators: - userID

API name: updateRating Url: PUT http://localhost:8000/api/profile/:id/:rating	
Required field: Rating: str ID: str	
Responsibilities: - Change the specified user's rating based on its current rating	Collaborators: - userID

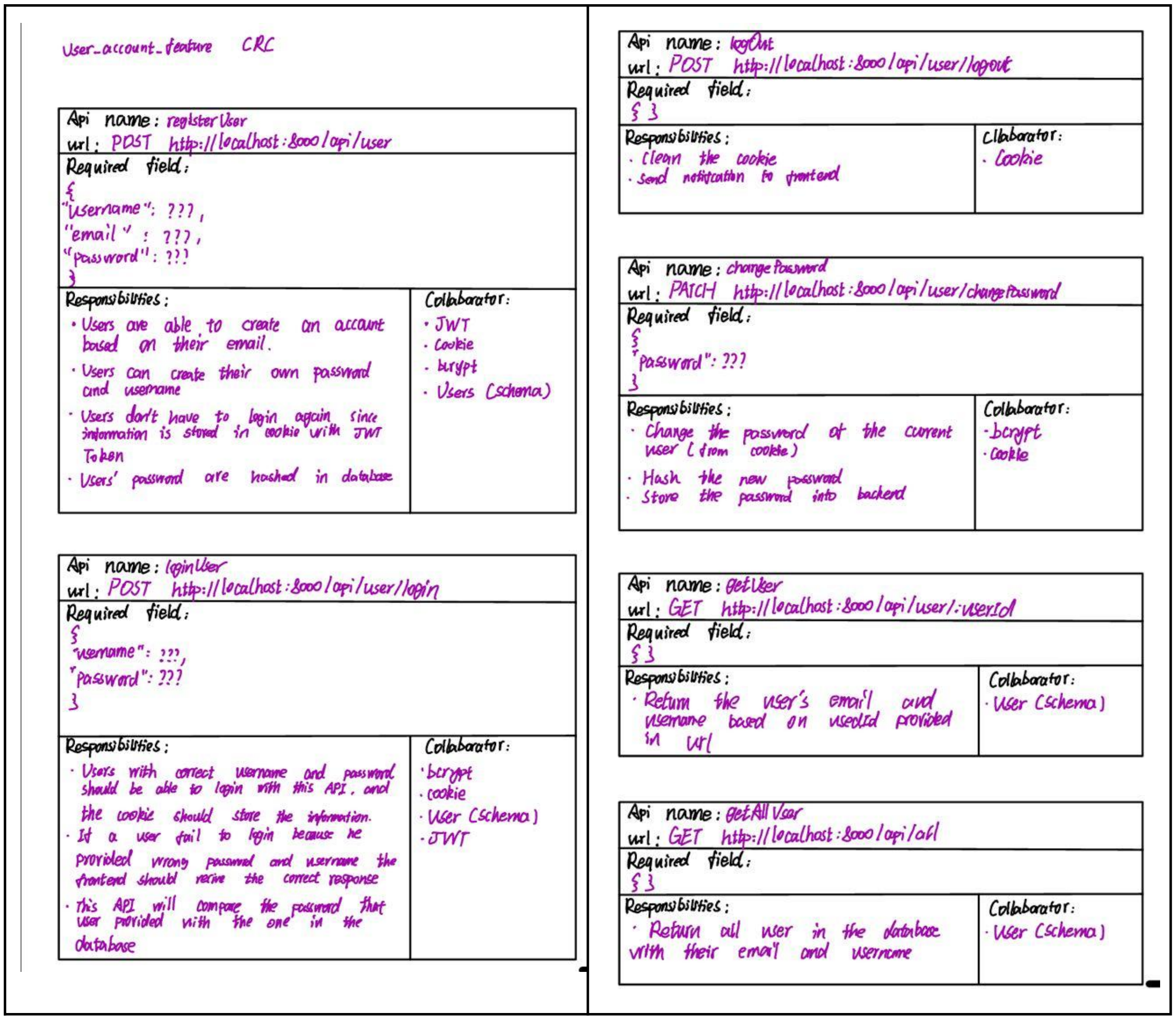
API name: updateAnsweredQuestions Url: PUT http://localhost:8000/api/profile/:id/a_question/:question_id	
Required field: ID: str question_id: str	
Responsibilities: - Change what's in user's answered questions list	Collaborators: - userID - question_id

API name: currency Url: POST http://127.0.0.1:8000/api/currency	
Required field: None	
Responsibilities: - Create an api related to this user - Keeps tracks on the amount the user have	Collaborators:

API name: currency Url: GET http://127.0.0.1:8000/api/:userID	
Required field: UserID	
Responsibilities: - Return the amount this user has	Collaborators:

API name: currency Url: PATCH http://127.0.0.1:8000/api/add/:userID	
Required field: UserID	
Responsibilities: - Add to the amount this user has	Collaborators:

API name: currency Url: PATCH http://127.0.0.1:8000/api/subtract/:userID	
Required field: UserID	
Responsibilities: - Subtract the amount of this user - Return error if not enough amount	Collaborators: Question, user



User_account_feature CRC

Api name: registerUser
url: POST http://localhost:8000/api/user

Required field:
```
{
"username": ???,
"email" : ???,
"password": ???
}
```

Responsibilities:	Collaborator:
· Users are able to create an account based on their email.	· JWT
· Users can create their own password and username	· Cookie
· Users don't have to login again since information is stored in cookie with JWT Token	· brypt
· Users' password are hashed in database	· Users (schema)

Api name: loginUser
url: POST http://localhost:8000/api/user/login

Required field:
```
{
"username": ???,
"password": ???
}
```

Responsibilities:	Collaborator:
· Users with correct username and password should be able to login with this API, and the cookie should store the information.	· bcrypt
· If a user fail to login because he provided wrong password and username the frontend should recive the correct response	· cookie
· This API will compare the password that user provided with the one in the database	· User (schema)
	· JWT

Api name: logOut
url: POST http://localhost:8000/api/user/logout

Required field:
```
{ }
```

Responsibilities:	Cllaborator:
· Clean the cookie	· Cookie
· Send notification to frontend	

Api name: changePassword
url: PATCH http://localhost:8000/api/user/changePassword

Required field:
```
{
"password": ???
}
```

Responsibilities:	Collaborator:
· Change the password of the current user (from cookie)	-bcrypt
· Hash the new password	· Cookie
· Store the password into backend	

Api name: getUser
url: GET http://localhost:8000/api/user/:userId

Required field:
```
{ }
```

Responsibilities:	Collaborator:
· Return the user's email and username based on usedId provided in url	· User (schema)

Api name: getAllUsar
url: GET http://localhost:8000/api/all

Required field:
```
{ }
```

Responsibilities:	Collaborator:
· Return all user in the database with their email and username	· User (schema)

Frontend CRC Cards

Class Name: Register page	
Parent Class: None Sub Class: None	
Responsibilities: - A form with name, email, password, re-enter password and submit button input for the registration process. - Send the form data to the backend by calling the backend API. - A navigation bar to toggle from register page to login page.	Collaborators: - registerUser - React - Tailwindcss - React-redux

Class Name: Login page	
Parent Class: None Sub Class: None	
Responsibilities: - A form with name, password and submit button input for the login process. - Send the form data to the backend by calling the backend API. - A navigation bar to toggle from register page to login page.	Collaborators: - loginUser - React - Tailwindcss - React-redux

API name: getQuestion Url: GET http://127.0.0.1:8000/api/questions/:id/:category?	
Required field: title	
Responsibilities: - Return the matching question for corresponding category/title, whichever one is included	Collaborators: ● questionSchema ● MongoDB ● Javascript

API name: getQuestionById Url: GET http://127.0.0.1:8000/api/questions/:id/q_id/:question_id	
Required field: None	
Responsibilities: - Return the matching question for corresponding question_id.	Collaborators: ● questionSchema ● MongoDB ● Javascript

Frontend CRC Cards

Class Name: Dashboard	
Parent Class: App Sub Class: Sidebar, NewQuestion, Profile	
Responsibilities: - A main stage to display different pages when it is called.	Collaborators: - React - Tailwindcss - React-redux - react-router-dom

Class Name: NewQuestion	
Parent Class: Dashboard Sub Class: None	
Responsibilities: - A form with title, body and bounty with a button “publish” for posting a new question process.	Collaborators: - React - Tailwindcss - React-redux - axios

Class Name: Profile	
Parent Class: Dashboard Sub Class: None	
Responsibilities: - A form with current username, email, new password and confirm new password with a button “save chage” for updating a user’s profile process.	Collaborators: - React - Tailwindcss - React-redux - axios

Api name: changUserInfo
url: PATCH http://localhost:8000/api/user/changeUserInfo

Required field:

```
{
 "id": ???,
 "password": ???,
 "username": ???,
 "email": ???
}
```

Responsibilities:	Collaborator:
· Based on the given field change the user infornations	-bcrypt ·Cookie

Api name: google
url: GET http://localhost:8000/outh/google

Required field:

```
{
}
```

Responsibilities:	Collaborator:
· use passport to authenticate with scope of profile and email · Get a google user's information from google provider	· passport

Api name: google
url: GET http://localhost:8000/outh/google/callbach

Required field:

```
{
}
```

Responsibilities:	Collaborator:
· Callback function of google passport · whenever google Oauth is invoked handle it's action if succeed, save user id to cookie	· passport · cookie