System Design Document

Table Of Content

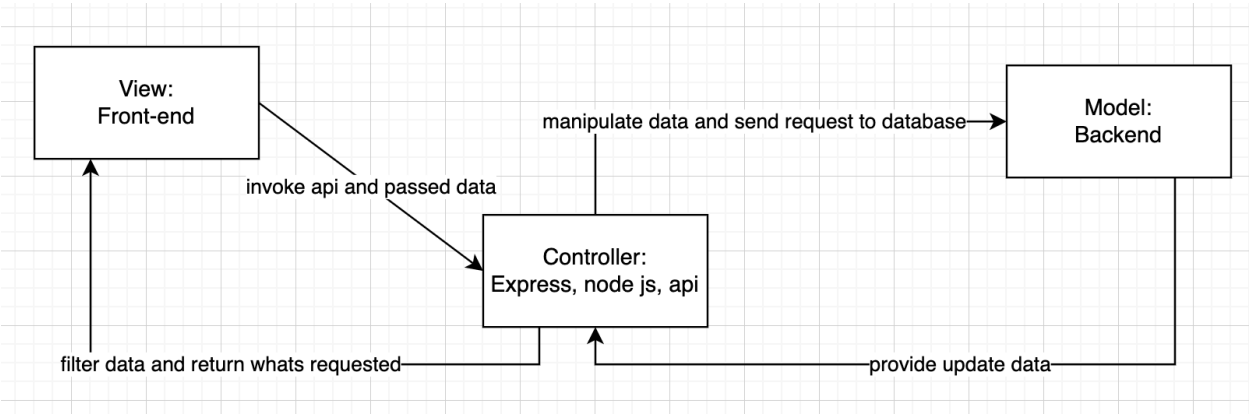
Software Architecture Diagram **3**

CRC Cards **4**

Backend CRC cards 4

Frontend CRC Cards 8

Software Architecture Diagram



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:/name:	
Required field: Name:str ID:num	
Responsibilities: - Delete stored user information from id - Change user name in this case	Collaborators: - userID

User_account_feature CRC

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

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

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

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

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

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

Api name: logOut url: POST http://localhost:8000/api/user/logout	
Required field: { }	
Responsibilities: • Clean the cookie • Send notification to frontend	Collaborator: • Cookie

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

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

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

Api name: getUser url: GET http://localhost:8000/api/user/:userId	
Required field: { }	
Responsibilities: • Return the user's email and username based on userId provided in url	Collaborator: • User (schema)

Api name: getAllUser url: GET http://localhost:8000/api/all	
Required field: { }	
Responsibilities: • Return all user in the database with their email and username	Collaborator: • 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