System Design - Sprint 1

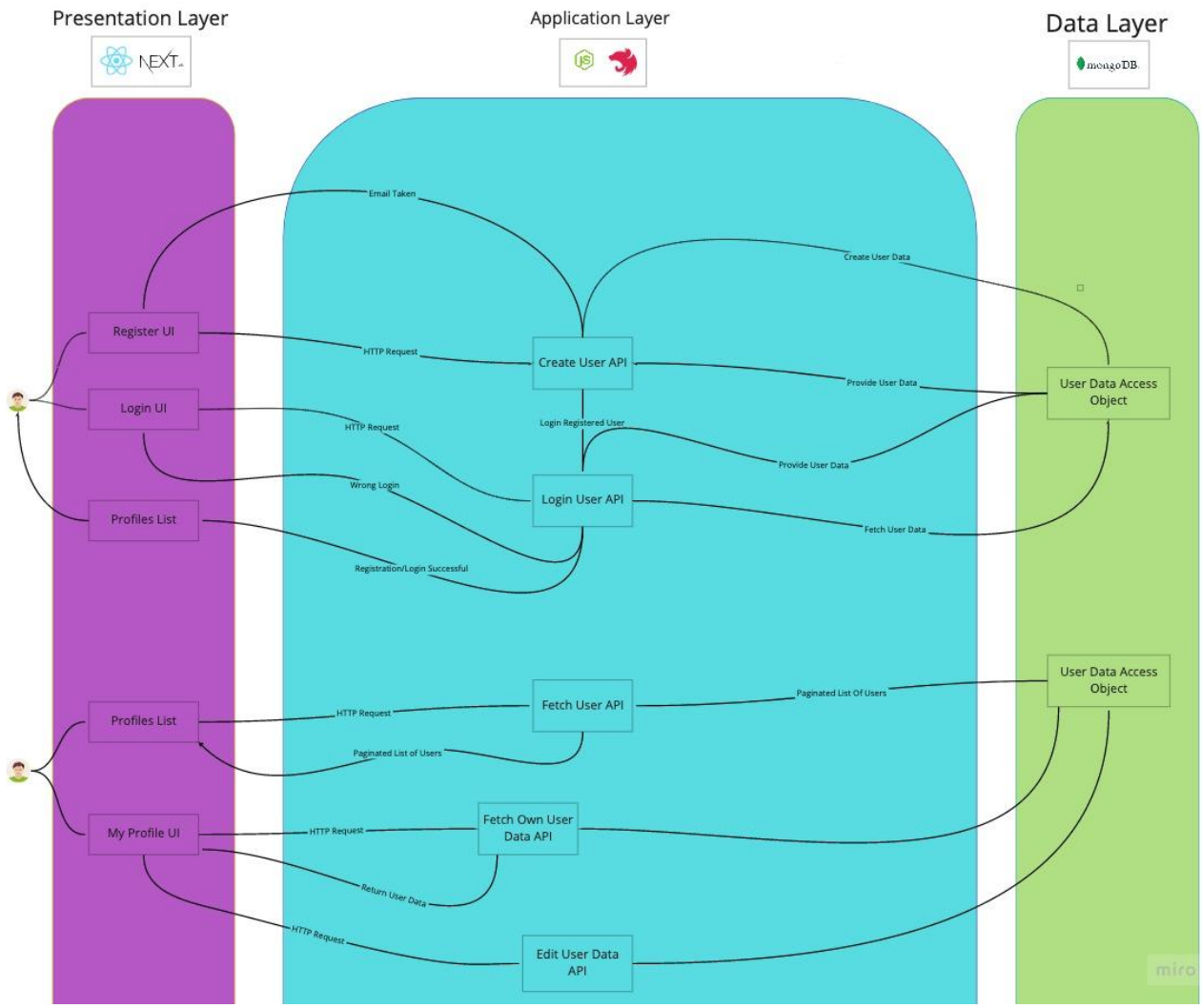
We will create a functional website which should work on any device with a modern web browser. We will use the three-tiered Architecture. The technologies for the three layers are React (Next.js) for the client tier, Node.js (NestJS) for the logic tier, and MongoDB for the Data Tier. However, we will use an in-memory repository for sprint 1 and transition to MongoDB in a future sprint. A high-level diagram of the interactions between components is shown in the next page.

Error Handling

In an event of server-side errors, we return a corresponding error code to indicate an error occurred. We use NestJS passport authentication model which authenticates a user by giving the user a JWT cookie which is maintained to validate requests.

Unauthorized Actions: This happens when a non-logged in user tries to send a request through the server which requires the user to be logged in. In this case, we return a 401 error code.

Network Failure: This happens when the client attempts to send a request but the server runs into an issue. In this case, we return a 500 error code.



CRC Cards: Common

The common part includes requests and responses format shared between the server and the client.

Class Name: LoginRequest

Responsibilities: Store schema of a login request. A login request should include: • Email • Password	**Collaborators**: N/A

Type Name: SessionPayload	
Responsibilities: Store type of a session payload. Includes: • Id • Full name • Integrity hash	**Collaborators**: N/A

CRC Cards: Server

Entities:

Class Name: User	
Responsibilities: • Store a user's id, first name, last name, about, email, interests, hashed password. • Provide a getter for each attribute	**Collaborators**: N/A

Class Name: Profile	
Responsibilities: • Store a user's id, name, bio, external links, job title, role. • Provide a getter for each attribute. • Edit the job title and the bio. • Add an external link. • Remove an external link.	**Collaborators**: N/A

Class Name: Message

Responsibilities: • Store message id, message content, posted by user id, child message ids, parent message id, posted at time.	**Collaborators**: N/A

Class Name: Profile	
Responsibilities: • Store profile name, related organizations, profile bio, all posts made by the profile, external links, profile id.	**Collaborators**: • Message

Class Name: RequestStatus (enum)	
Responsibilities: The statuses a request can have • Pending • Approved • Rejected	**Collaborators**: N/A

Class Name: Role (enum)	
Responsibilities: The roles in the program • Default • Entrepreneur • Investor • Service Provider • Admin (to be added in sprint 2) • Mentor (to be added in sprint 2)	**Collaborators**: N/A

Class Name: Request	
Responsibilities: • Store request id, user id, requested role, description of the request, request time, status	**Collaborators**: • RequestStatus • Role

Data Access Object:

Interface: UserProfileDAO

Responsibilities: • Register and login a user • Create a corresponding profile for a user • Get a user by id and by email • Get a profile, and a paginated profile list.	**Collaborators**: • User • Profile

Class Name: UserProfileInMemory	
Implements: UserProfileDAO	
Responsibilities: • Store all User entities • Implement methods from UserProfileDAO in memory.	**Collaborators**: • User

Class Name: MessageDAO	
Responsibilities: • Create a new Message • Get all child messages for a given message. • Get a message by ID • Get a parent of a message.	**Collaborators**: • Message

Class Name: MessageInMemory	
Implements: MessageDAO	
Responsibilities: • Store all Message Entities • Implement interface methods in MessageDAO	**Collaborators**: • Message

Class Name: RequestDAO	
Responsibilities: • Create a new Request • Get a Request by ID	**Collaborators**: • Request

Services:

Contain DAOs interface, In-Memory implementation of the interfaces, and implementations of the APIs.

Class Name: UserService	
Responsibilities: ● Register and login a user ● Get a user by id ● Get K users - pagination ● Edit user profile information	**Collaborators**: ● UserDAO

Class Name: MessageService	
Responsibilities: ● Create a message ● Get a message by id ● Get K messages - pagination	**Collaborators**: ● MessageDAO

Class Name: ProfileService	
Responsibilities: ● Create a profile ● Get posts by ID ● Get related links by ID ● Get bio by ID	**Collaborators**: ● ProfileDAO

Controllers:

Class Name: UserController	
Responsibilities: ● Get User by id API ● Get all users API - pagination ● POST User API	**Collaborators**: ● UserService

Class Name: MessageController	
Responsibilities: • Get Message API • Get all messages API - pagination • POST Message API	**Collaborators**: • MessageService

Class Name: AuthController	
Responsibilities: • Post: Login API • Get: Session API	**Collaborators**: • AuthService • ConfigService • LoginRequest

Modules:

Class Name: UserProfileModule	
Responsibilities: • Group functionality from UserController and UserService into a module	**Collaborators**: • UserProfileController • UserProfileService • UserProfileInMemory

Class Name: MessageModule	
Responsibilities: • Group functionality from MessageController and MessageService into a module	**Collaborators**: • MessageController • MessageService • UserProfilesInMemory

Class Name: AuthModule	
Responsibilities: • Group functionality from AuthController and AuthService into a module	**Collaborators**: • AuthController • AuthService

CRC Cards: Client

Class Name: AuthProvider	
Responsibilities: ● Stores currently logged in user's information (just name as of sprint 1). ● Subscribes to the frontend's HTTP client and updates currently logged in user's information when necessary (for example when server removes cookies after user logs out). ● Makes currently logged in user's information available to other components via React context.	**Collaborators**: ● SessionPayload

Class Name: ProfileProvider	
Responsibilities: ● Returns the given profile as a JSON file, with the name, role and description of themselves.	**Collaborators**: ● /api/profile

Pages:

Class Name: Feed	
Responsibilities: ● Displays a feed where you can view and make posts	**Collaborators**: ● MessageFeed, ● SendMessage, ● /api/message/create, ● /api/message/paginatio n

Class Name: Profile

Responsibilities: • Displays the profile on the overview tab and recent activity.	**Collaborators**: • About

Class Name: ProfileList	
Responsibilities: • Renders a list of all the profiles	**Collaborators**: • api/user/pagination

Components:

Class Name: Navbar	
Responsibilities: • Displays logo, site navigation links, and an authentication component in a horizontally arranged manner.	**Collaborators**: • AuthComponent

Class Name: AuthComponent	
Responsibilities: • Displays registration and login buttons when no user is logged in. • Stores registration and login forms which the registration and login buttons can open. • Displays name and logout button when a user is logged in. • Sends a logout request to the backend when the logout button is clicked.	**Collaborators**: • AuthProvider • RegisterForm • LoginForm • /api/auth/logout

Class Name: RegisterForm	
Responsibilities: • Displays form for registering a new account on the site. • Validates data a user intends to use to register their account and displays any feedback to the user (e.g. invalid email address). • Sends a registration request to the backend when the submit form button is clicked.	**Collaborators**: • /api/user/register

Class Name: RoleRequestForm	
Responsibilities: ● Request a role for your account to be changed to	**Collaborators**: ● N/A

Class Name: LoginForm	
Responsibilities: ● Displays form for logging into an existing account on the site. ● Validates data a user intends to use to log into their account and displays any feedback to the user (e.g. an incorrect password was entered). ● Sends a login request to the backend when the submit form button is clicked.	**Collaborators**: ● /api/user/login

Class Name: About	
Responsibilities: ● Displays the user's name, role, and a bio about them	**Collaborators**: ● ChangeNameForm ● ChangeBioForm ● ChangeJobForm

Class Name: ChangeNameForm	
Responsibilities: ● Allows a user to change their name	**Collaborators**: ● About

Class Name: Card	
Responsibilities: ● Displays a more rounded ANT-UI box for components to store in.	**Collaborators**: N/A

Class Name: ChangeBioForm

Responsibilities: ● Allows a user to change their bio	**Collaborators**: ● About

Class Name: ChangeJobForm	
Responsibilities: ● Allows a user to change their job	**Collaborators**: ● About

Class Name: SendMessage	
Responsibilities: ● Allows the user to write a message in a text box. ● Allows the user to send the message in the text box through a button.	**Collaborators**: ● Feed

Class Name: Message	
Responsibilities: ● Display the author's name. ● Display the timestamp in which it was posted. ● Display the message.	**Collaborators**: N/A

Class Name: MessageFeed	
Responsibilities: ● Display messages sent to the server. ● Can scroll down to reveal more messages ● Displays a loading pinwheel when trying to get messages.	**Collaborators**: ● MessageList ● Feed

Class Name: MessageList	
Responsibilities: ● Display a list of messages.	**Collaborators**: ● Message