System Design

Sprint 2

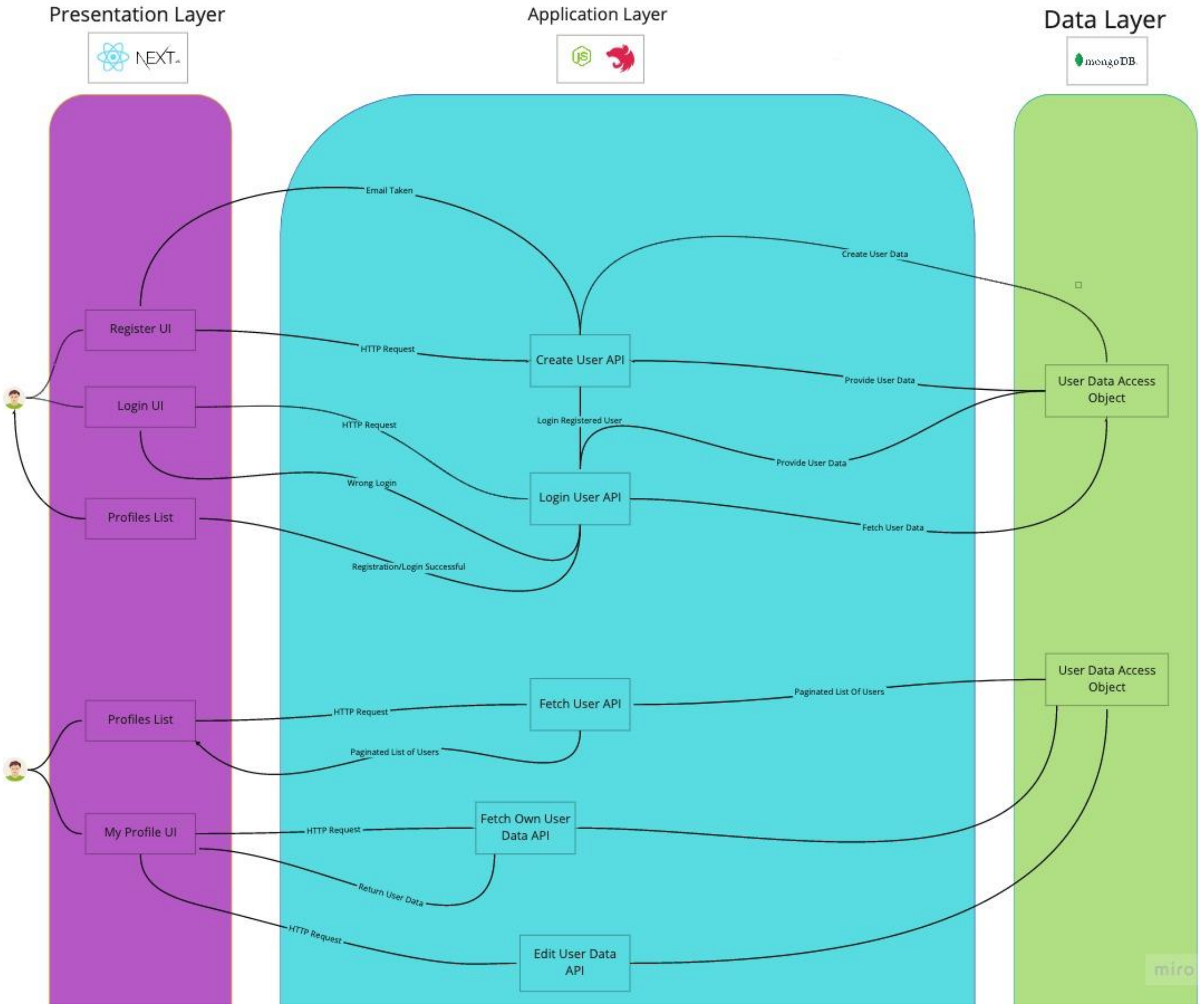
Team: Fifth Order Ignorant

Maryam Gohargani
Michael Sheinman Orenstrakh
Hai Yang Tang
Catherine Xia
Akira Takaki
Jan Garong
Ethan Lam

Table of contents:

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 application tier, and MongoDB for the Data Tier. However, we will use an in-memory repository for sprint 2 and transition to MongoDB in a future sprint. A high-level diagram of the interactions between components is shown below.

Diagram



Error Handling

In the event of server-side errors, we return a corresponding error code to indicate an error occurred. We use the 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

CRC Cards: Common

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

Requests

Class Name: EditBioRequest	
Responsibilities: Store schema of an edit bio request. The request includes a new bio.	**Collaborators**: N/A

Class Name: EditJobRequest	
Responsibilities: Store schema of an edit job request. The request includes a new job.	**Collaborators**: N/A

Class Name: EditNameRequest	
Responsibilities: Store schema of an edit name request. The request includes the new name.	**Collaborators**: N/A

Class Name: EditProfileRequest	
Responsibilities: Store schema of an edit profile request. The request includes: ● First name ● Last name ● Job Title ● Bio	**Collaborators**: N/A

Class Name: LoginRequest	
Responsibilities: Store schema of a login request. A login request should include: ● Email ● Password	**Collaborators**: N/A

Class Name: PostRequest	
Responsibilities: Store schema of a post request. The request includes: ● Content ● Parent id	**Collaborators**: N/A

Class Name: ProfileViewRequest	
Responsibilities: Store schema of a profile view request. A profile view request includes the user id.	**Collaborators**: N/A

Class Name: ProfilePaginationRequest	
Responsibilities: Store schema of a profile pagination request. The request includes: ● Start ● End	**Collaborators**: N/A

Class Name: RegisterRequest	
Responsibilities: Store schema of a register request. The request includes: ● First Name ● Last Name ● Email ● Password	**Collaborators**: N/A

Class Name: RepliesPaginationRequest	
Responsibilities: Store schema of a replies pagination request. The request includes: ● Id ● Start ● End	**Collaborators**: N/A

Class Name: RoleRequest	
Responsibilities: Store schema of a role request. A role request includes: ● Role ● Description	**Collaborators**: N/A

Class Name: TeamRequest	
Responsibilities: Store schema of a team creation request. A team creation request should include: ● Team Name ● User Id	**Collaborators**: N/A

Class Name: UserPostsPaginationRequest	
Responsibilities: Store schema of a pagination request to the user posts pagination request. ● Id ● Start ● End	**Collaborators**: N/A

Class Name: UserRepliesPaginationRequest	
Responsibilities: Store schema of a pagination request to the replies to a given post. The request includes: ● Id ● Start ● End	**Collaborators**: N/A

Responses

Class Name: ErrorResponse	
Responsibilities: Store the format of an error response.	**Collaborators**: N/A

Class Name: PostResponse	
Responsibilities: Store the format of a response to a post	**Collaborators**: N/A

request. The response includes: ● Author ● Timestamp ● Content ● post id	

Class Name: ProfileResponse	
Responsibilities: Store the format of a response to a profile request. The response includes: ● Id ● Bio ● First name ● Last Name ● Job title ● Role	**Collaborators**: N/A

Class 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, email, and 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, interests. ● 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: Post	
Responsibilities: ● Store post id, post content, posted by user id, child post ids, parent post 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**: ● Post

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	**Collaborators**: N/A

● Investor ● Service Provider ● Admin (to be added in sprint 3) ● Mentor (to be added in sprint 3)	

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

Class Name: Team	
Responsibilities: ● Store the team id, team creator id, team name, a list of team members ids, and the creation time.	**Collaborators**: N/A

Class Name: Assignment	
Responsibilities: ● Store the assignment id, name, description, and max mark.	**Collaborators**: N/A

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: PostDAO

Responsibilities:

- Create a new Post
- Get all child posts for a given post.
- Get a post by ID
- Get a parent of a post.

Collaborators:

- Post

Class Name: PostInMemory

Implements: PostDAO

Responsibilities:

- Store all Post Entities
- Implement interface methods in PostDAO

Collaborators:

- Post

Class Name: RequestDAO

Responsibilities:

- Create a new Request
- Get a Request by ID

Collaborators:

- Request

Class Name: RequestInMemory

Implements: RequestDAO	
Responsibilities: ● Store all Request entities ● Implement interface methods in RequestDAO	**Collaborators**: ● Request

Class Name: AssignmentDAO	
Responsibilities: ● Create an assignment ● Get an assignment by ID	**Collaborators**: ● Request

Class Name: AssignmentInMemory	
Implements: AssignmentDAO	
Responsibilities: ● Store all Assignment entities ● Implement interface methods in AssignmentDAO	**Collaborators**: ● Assignment

Services

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

Class Name: UserProfileService	
Responsibilities: ● Register a user ● Get a user by id ● Get K profiles - pagination ● Edit user profile information	**Collaborators**: ● UserProfileDAO

Class Name: PostService	
Responsibilities: ● Create a post ● Get a post by id ● Get K posts - pagination ● Get K posts or replies made by a user - pagination	**Collaborators**: ● PostDAO

Class Name: AuthService	
Responsibilities: ● Authorize a user’s sessions	**Collaborators**: ● UserProfileDAO ● JwtService

Class Name: TeamService	
Responsibilities: ● Create a new team. ● Check if a team exists.	**Collaborators**: ● TeamDAO ● Team

Class Name: AssignmentService	
Responsibilities: ● Create a new assignment.	**Collaborators**: ● Assignment ● AssignmentDAO

Controllers

The controllers control the flow of the program and create the API endpoint for the program.

Class Name: UserProfileController	
Responsibilities: ● Get a single user ● Get all users API - pagination	**Collaborators**: ● UserProfileService

● POST Register API	

Class Name: UserProfileEditController	
Responsibilities: ● Post: EditName API ● Post: EditJob API ● Post: EditBio API	**Collaborators**: ● UserProfileService ● AuthService

Class Name: PostController	
Responsibilities: ● Get Post API ● Get all posts API - pagination ● Get posts by a user - pagination ● POST Post API	**Collaborators**: ● PostService

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

Class Name: JwtAuthGuard	
Responsibilities: ● Extends Passport’s JWT authentication guard with XSS countermeasure (HTTP only cookie).	**Collaborators**: ● AuthService

Class Name: JwtInterceptor	
Responsibilities: ● Renews session cookies when they have been marked as stale or are about to expire.	**Collaborators**: ● AuthService ● ConfigService

Class Name: JwtStrategy	
Responsibilities: ● Populates request objects received by controllers with the currently logged-in user’s information (ID).	**Collaborators**: ● ConfigService

Class Name: TeamController	
Responsibilities: ● POST: Create team	**Collaborators**: ● TeamService

Class Name: AssignmentController	
Responsibilities: ● POST: Create assignment	**Collaborators**: ● AssignmentService

Modules

The modules provide metadata that Nest uses to organize the application structure. This includes grouping controllers, services, and data access objects together.

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

Class Name: PostModule	
Responsibilities: ● Group functionality from PostController and PostService into a module	**Collaborators**: ● PostController ● PostService ● UserProfilesInMemory

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

Class Name: TeamModule	
Responsibilities: ● Group functionality from TeamController, TeamService, and TeamInMemory into a Module.	**Collaborators**: ● TeamController ● TeamInMemory ● TeamService

Class Name: AssignmentModule	
Responsibilities: ● Group functionality from AssignmentController, AssignmentService, and AssignmentInMemory into a Module.	**Collaborators**: ● AssignmentController ● AssignmentInMemory ● AssignmentService

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 the profile.	**Collaborators**: ● /api/profile

Pages

Class Name: Feed	
Responsibilities: ● Displays a feed where you can view and make posts	**Collaborators**: ● PostFeed, ● CreatePost, ● /api/post/pagination

Class Name: Profile (./client/profile/[id].tsx)	
Responsibilities: ● Displays the profile on the overview tab and recent activity. ● Displays recent posts made.	**Collaborators**: ● About ● PostFeed ● /api/post/userposts

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

Class Name: MyApp (./client/pages/_app.tsx)

Responsibilities: - Renders the navbar and layout of the page.	**Collaborators**: - Navbar

Class Name: createAssignment	
Responsibilities: - Renders the page of the assignment form.	**Collaborators**: - CreateAssignmentForm

Class Name: createTeam	
Responsibilities: - Renders the page of the team form.	**Collaborators**: - CreateTeamForm

Class Name: Profiles	
Responsibilities: - Renders the page which contains a list of profiles	**Collaborators**: - /api/user/pagination

Class Name: Request	
Responsibilities: - Renders the page which allows the user to request a role.	**Collaborators**: - RoleRequestForm

Components

Class Name: Navbar	
Responsibilities: ● Displays the 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 the 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 the 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: SendPost	
Responsibilities: ● Displays a textbox used to create original or reply posts.	**Collaborators**: N/A

Class Name: CreatePost	
Responsibilities: ● Submits original post creation request to the backend.	**Collaborators**: ● SendPost ● /api/post/create

Class Name: ReplyPost	
Responsibilities: ● Submits a reply post creation request to the backend.	**Collaborators**: ● SendPost ● /api/post/create

Class Name: Post	
Responsibilities: ● Displays the author's name. ● Displays the timestamp in which it was posted. ● Displays the post's text content. ● Holds the PostList component used to display replies. ● Holds the ReplyPost component used to reply to the post.	**Collaborators**: ● PostList ● ReplyPost

Class Name: PostFeed	
Responsibilities: ● Displays posts sent to the server. ● Triggers the fetching of more posts when the user scrolls to the end of the component.	**Collaborators**: ● PostList

Class Name: PostList	
Responsibilities: ● Displays a list of posts. ● Displays a loading pinwheel when loading more posts.	**Collaborators**: ● Post ● /api/post/replies

Class Name: CreateTeamForm	
Responsibilities: ● Allows a user to create a team given that they fill out the required fields provided in the form.	**Collaborators**: ● /api/team/create

Class Name: CreateAssignmentForm	
Responsibilities: ● Allows a user to create an assignment with a title, mark, and description.	**Collaborators**: ● /api/assignment/create