System Design

Sprint 4

Team: Fifth Order Ignorant

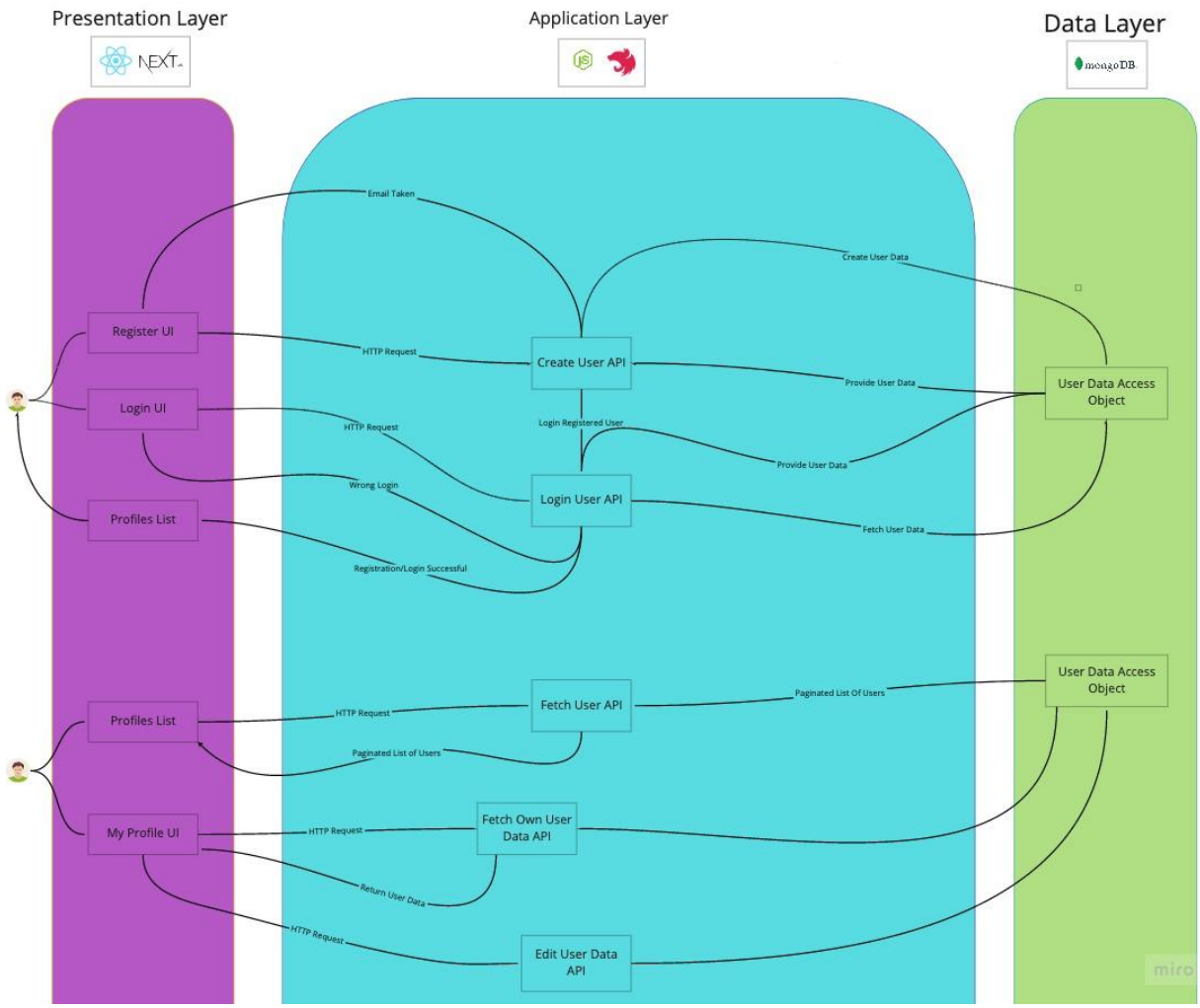
Maryam Gohargani
Michael Sheinman Orenstrakh
Hai Yang Tang
Catherine Xia
Akira Takaki
Jan Garong
Ethan Lam

Table of contents:

- Pages
  - Class Name: Feed
  - Class Name: Profile (./client/profile/[id].tsx)
  - Class Name: Profiles
  - Class Name: MyApp (./client/pages/_app.tsx)
  - Class Name: CreateAssignment
  - Class Name: CreateTeam
  - Class Name: Assignments
  - Class Name: Assignment (pages/assignment/[id].tsx)
  - Class Name: Profiles
  - Class Name: Request
  - Class Name: AdminRequest
- Components
  - Class Name: Navbar
  - Class Name: AuthComponent
  - Class Name: RegisterForm
  - Class Name: RoleRequestForm
  - Class Name: LoginForm
  - Class Name: About
  - Class Name: ChangeNameForm
  - Class Name: Card
  - Class Name: ChangeBioForm
  - Class Name: ChangeJobForm
  - Class Name: SendPost
  - Class Name: CreatePost
  - Class Name: ReplyPost
  - Class Name: Post
  - Class Name: Links
  - Class Name: ChangeLinkForm
  - Class Name: PostFeed
  - Class Name: PostList
  - Class Name: CreateTeamForm
  - Class Name: CreateAssignmentForm
  - Class Name: AssignmentView
  - Class Name: AvatarComponent
  - Class Name: TagList
  - Class Name: SetTagList
  - Class Name: SubmissionTable

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.

Forbidden

A forbidden error occurs when a user attempts to access an API without a corresponding permission. For instance, if a normal user attempts to see another user's assignment submission they will see a corresponding error message. In this case, we return a 403 error code.

Conflict

This happens when the client attempts to send a request but the resource already exists in the database. In this case, we return a 409 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: AssignmentCreateRequest	
Responsibilities: Store schema of an assignment creation request. The request includes the name, description and max mark for the assignment.	**Collaborators**: N/A

Class Name: AssignmentPaginationRequest	
Responsibilities: Store schema of an assignment pagination request. The request includes the start and the end for where to fetch the assignment from.	**Collaborators**: N/A

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: EditLinksRequest	
Responsibilities: Store schema of an edit links request. The request includes the new links.	**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	**Collaborators**: N/A

• Last name • Job Title • Bio	

Class Name: FileRequest	
Responsibilities: Store schema of a file request. A file request should include: • File id	**Collaborators**: N/A

Class Name: GetSingleSubAssRequest	
Responsibilities: Store schema of a get request for a submission or an assignment. A get request should include: • Assignment/Submission id	**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: PostPaginationRequest	
Responsibilities: Store schema of a profile pagination request. The request includes: • Start • End	**Collaborators**: N/A

Class Name: PostRequest	
Responsibilities: Store schema of a post request. The request includes: • Content • Parent 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: ProfilePctureRequest	
Responsibilities: Store schema of a profile picture request used to display a picture. A profile picture request includes the user 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: 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: SubmissionCreateRequest	
Responsibilities: Store schema of a submission creation request. A submission creation request includes: • User Id • Assignment Id	**Collaborators**: N/A

Class Name: SubmissionPaginationRequest	
Responsibilities: Store schema of paginating submissions: • start • end	**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: UploadFileRequest	
Responsibilities: Store schema of an upload file request. The request includes the id to be uploaded.	**Collaborators**: N/A

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

• Id • Start • End	

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: AssignmentResponse	
Responsibilities: Store the format of an assignment response. The response includes: • Id • Name • Description • maxMark	**Collaborators**: N/A

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 request. The response includes: • Author • Timestamp • Content • post id	**Collaborators**: N/A

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

Class Name: SingleAssignmentResponse	
Responsibilities: Store the format of a single assignment response. The response includes assignment id.	**Collaborators**: N/A

Class Name: SubmissionPaginationResponse	
Responsibilities: Store the data and total number of submissions for that assignment.	**Collaborators**: AssignmentSubmissionResponse

CRC Cards: Server

Entities

Class Name: Assignment

Responsibilities: • Store the assignment id, name, description, and max mark.	**Collaborators**: N/A

Class Name: FileInfo	
Responsibilities: • Store information about a file, including the name, size, mile type, and the original name.	**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 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: Request	
Responsibilities: • Store request id, user id, requested role, description of the request, request time, status	**Collaborators**: • RequestStatus • Role

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

• Approved • Rejected	

Class Name: Role (enum)	
Responsibilities: The roles in the program • Default • Entrepreneur • Investor • Service Provider • Admin (to be added in sprint 4) • Mentor (to be added in sprint 4)	**Collaborators**: N/A

Class Name: Submission	
Responsibilities: • Store userId, assignmentId, FileInfo, Feedback, mark.	**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: Tag (enum)	
Responsibilities: Store the tags in the program. The currently supposed tags are: • Business • Fun • Opportunity • Technology	**Collaborators**: N/A

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: User	
Responsibilities: • Store a user's id, email, and hashed password. • Provide a getter for each attribute	**Collaborators**: N/A

Data Access Object

The Data Access Objects (DAOs) are separated into the in memory DAOs, and the MongoDB DAOs. Each of these implement the corresponding DAO interface. Moreover, entries in the MongoDB database have a corresponding schema for data storage.

DAO Interfaces

Interface: AssignmentDAO	
Responsibilities: • Create an assignment • Get an assignment by ID • Get paginated assignments.	**Collaborators**: • Assignment

Interface: MulterDAO	
Responsibilities: • Retrieve a file. • Delete a file.	**Collaborators**: N/A

Interface: PostDAO	
Responsibilities:	• **Collaborators**:

• Store all Post Entities • Implement interface methods in PostDAO	• Post

Interface: RequestDAO	
Responsibilities: • Create a new Request • Get a Request by ID	**Collaborators**: • Role • Request

Interface: TeamDAO	
Responsibilities: • Create a new Team • Get a Team by ID • Check if a team exists • Get all teams	**Collaborators**: • Team

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

Interface: SubmissionDAO	
Responsibilities: • Create submissions, get submission Id, upload submission, upload feedback, replace submission and count of submissions for an assignment	**Collaborators**: • N/A

In Memory DAOs

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

Class Name: MulterLocalDAO	
Implements: MulterDAO	
Responsibilities: ● Read a locally stored file using the NodeJS file system API. ● Delete a locally stored file using the NodeJS file system API.	**Collaborators**: N/A

Class Name: PostInMemory	
Implements: PostDAO	
Responsibilities: ● Store all Post Entities in memory ● Implement interface methods in PostDAO	**Collaborators**: ● Post

Class Name: RequestInMemory	
Implements: RequestDAO	
Responsibilities: ● Store all Request entities in memory ● Implement interface methods in RequestDAO	**Collaborators**: ● Request ● Role

Class Name: TeamInMemory	
Implements: TeamDAO	
Responsibilities: ● Store all Team entities in memory ● Implement interface methods in TeamDAO	**Collaborators**: ● Team

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

MongoDB Schemas

Class Name: AssignmentSchema	
Responsibilities: Allows the user to interact with Assignments in the MongoDB database. The schema defines: ● Assignment ID ● Assignment Name ● Assignment Description ● Assignment maxMark	**Collaborators**: ● Assignment

Class Name: FileInfoSchema	
Responsibilities: Allows the user to interact with files in the MongoDB database. The schema defines: ● File name ● File mime type ● File original name ● File size	**Collaborators**: ● FileInfo

Class Name: PostSchema	
Responsibilities: Allows the user to interact with Posts in the MongoDB database. The schema defines: • Post ID • Post User ID • Post Content • Post Parent • Post timestamp • Post tags	**Collaborators**: • Post • Tag

Class Name: ProfileSchema	
Responsibilities: Allows the user to interact with Profiles in the MongoDB database. The schema defines: • Id • First Name • Last Name • Bio • External Links • Job Title • Role • Picture	**Collaborators**: • Profile • Role • FileInfoSchema

Class Name: RequestSchema	
Responsibilities: Allows the user to interact with Requests in the MongoDB database. The schema defines: • Id • userID • Status • Timestamp • Description • Role	**Collaborators**: • Request • Role • RequestStatus

Class Name: SubmissionSchema	
Responsibilities: Allows the user to store submissions associated with an assignment	**Collaborators**: • Submission

• Id • userId • assignmentId • timestamp • mark • Feedback • file	• FileInfoSchema

Class Name: TeamSchema	
Responsibilities: Allows the user to interact with teams in the MongoDB database. The schema defines: • Id • creatorId • Name • MemberIds • Timestamp	**Collaborators**: • Team

Class Name: UserSchema	
Responsibilities: Allows the user to interact with users in the MongoDB database. The schema defines: • Id • Email • Password	**Collaborators**: • User

MongoDB DAOs

Class Name: AssignmentMongoDAO	
Implements: AssignmentDAO	
Responsibilities: • Store all Assignment entities in a database • Implement interface methods in AssignmentDAO	**Collaborators**: • Assignment

Class Name: MulterConfigService	
Responsibilities: • Initializes the GridFS API.	**Collaborators**: N/A

Class Name: MulterMongoDAO	
Implements: MulterDAO	
Responsibilities: • Read a file stored on a MongoDB server using the GridFS API. • Delete a file stored on a MongoDB server using the GridFS API.	**Collaborators**: N/A

Class Name: PostMongoDAO	
Implements: PostDAO	
Responsibilities: • Store all Post Entities in the MongoDB database • Implement interface methods in PostDAO	**Collaborators**: • Post • PostSchema

Class Name: RequestMongoDAO	
Implements: RequestDAO	
Responsibilities: • Store all Request Entities in the MongoDB database • Implement interface methods in RequestDAO	**Collaborators**: • Request • Role • RequestDAO

Class Name: SubmissionMongoDAO	
Implements: SubmissionDAO	
Responsibilities: • Allows for submission creation, fetching submissions,	**Collaborators**: • FileInfo

replacing submissions, and uploading files	• SubmissionDAO • Submission

Class Name: TeamMongoDAO	
Implements: TeamDAO	
Responsibilities: • Store all Team Entities in the MongoDB database • Implement interface methods in TeamDAO	**Collaborators**: • Team

Class Name: UserProfileMongoDAO	
Implements: UserProfileDAO	
Responsibilities: • Store all User and Profile Entities in the MongoDB database • Implement interface methods in UserProfileDAO	**Collaborators**: • User • Profile • ProfileResponse

Services

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

Class Name: AuthService	
Responsibilities: • Authorize a user's sessions	**Collaborators**: • UserProfileDAO • JwtService

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: SubmissionService	
Responsibilities: ● Create a submission ● Get a user's submission by assignment id ● Get K submissions - pagination ● Get the number of submissions ● Upload submission file ● Upload submission feedback	**Collaborators**: ● SubmissionDAO ● Submission

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

Controllers

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

Class Name: AssignmentController

Responsibilities: • POST: Create assignment • GET: Assignment Pagination	**Collaborators**: • AssignmentService

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

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: • Extends Passport's JWT authentication guard with XSS countermeasure (HTTP only cookie). • Populates request objects received by controllers with the currently logged-in user's information (ID).	**Collaborators**: • AuthService • ConfigService

Class Name: RolesGuard	
Responsibilities: • Ensures a user has the necessary role to access an endpoint before allowing their request to proceed.	**Collaborators**: •

Class Name: PostController	
Responsibilities: • Get Post API • Get all posts API - pagination	**Collaborators**: • PostService

• Get posts by a user - pagination • POST Post API	

Class Name: RequestController	
Responsibilities: • POST: Request	**Collaborators**: •

Class Name: SubmissionController	
Responsibilities: • GET: Submission by ID • PUT: Submission instance • POST: Submission File • GET: Submission pagination	**Collaborators**: • Submission • UserProfileService • Role • JwtAuth • SubmissionService

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

Class Name: UserProfileController	
Responsibilities: • Get a single user • Get all users API - pagination • POST: Register API • GET: Picture API	**Collaborators**: • UserProfileService

Class Name: UserProfileEditController	
Responsibilities: Create API requests for editing profile information.	**Collaborators**: • UserProfileService

• PATCH: EditName API • PATCH: EditJob API • PATCH: EditBio API • PATCH: EditRole API • PATCH: EditLinks API • POST: Picture API	• AuthService

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: AssignmentModule	
Responsibilities: • Group functionality from AssignmentController, AssignmentService, and AssignmentDAO into a Module.	**Collaborators**: • AssignmentController • AssignmentDAO • AssignmentService

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

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

Class Name: SubmissionModule	
Responsibilities: • Group functionality from SubmissionController, SubmissionService, and SubmissionDAO into a	**Collaborators**: • SubmissionController • SubmissionService

module	• SubmissionDAO

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

Class Name: InMemoryDAOModule	
Responsibilities: • Registers in memory DAOs into the NestJS dependency injection system.	**Collaborators**: • MulterLocalDAO • AssignmentInMemory • PostInMemory • RequestInMemory • TeamInMemory • UserProfileInMemory

Class Name: MongoDBDAOModule	
Responsibilities: • Registers MongoDB DAOs into the NestJS dependency injection system.	**Collaborators**: • MulterMongoDAO • UserProfileMongoDAO • PostMongoDAO • RequestMongoDAO • AssignmentMongoDAO

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

Class Name: UserProfileEditModule	
Responsibilities: ● Group functionality from UserProfileEditController and UserProfileService into a module	**Collaborators**: ● UserProfileEditController ● UserProfileService ● AuthModule

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

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: Profiles	
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: Assignments	
Responsibilities:	**Collaborators**:

• Renders the page which contains a list of assignments	• /api/assignment/pagination

Class Name: Assignment (pages/assignment/[id].tsx)	
Responsibilities: • Fetches an individual assignment's info from the backend. • Renders information related to an individual assignment. • Renders submissions if you're an admin or a mentor.	**Collaborators**: • /api/assignment/get/:id • AssignmentView

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

Class Name: AdminRequest	
Responsibilities: • Renders the page for admins to review role requests.	**Collaborators**: • RequestFeed

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:	**Collaborators**:

• Request a role for your account to be changed to	• 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**: ● SetTagList

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 ● TagList ● ReplyPost

Class Name: Links	
Responsibilities: ● Displays links from the database on a userprofile	**Collaborators**: ● ChangeLinkForm

Class Name: ChangeLinkForm	
Responsibilities: ● Displays a form that allows users to change their links	**Collaborators**: ● /api/profile/editlinks

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

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

Class Name: AvatarComponent	
Responsibilities: ● Renders an individual assignment. ● Displays the download link for an individual assignment's attachment.	**Collaborators**: ● /api/assignment/file/:id

Class Name: TagList	
Responsibilities: ● Displays a list of tags	**Collaborators**: ● N/A

Class Name: SetTagList	
Responsibilities: ● Displays a list of tags which can be removed/added.	**Collaborators**: ● N/A

Class Name: SubmissionTable

Responsibilities: ● Paginate through a list of submissions for the given assignment.	**Collaborators**: ● N/A