Technical Documentation: Equity Community App

Overview

The Equity Community app is a welcoming and collaborative platform for the Equity organization, with a focus on Wings to Fly/Elp scholars, moderators, and administrators. The app provides a user-friendly design and effective communication channels to link persons within the Equity community by leveraging modern online technologies.

The application's frontend is built with React.js, a popular JavaScript toolkit for creating user interfaces. React.js takes a modular and component-based approach to development, allowing for reusable code and a pleasant user experience. Users can access the application's numerous features and functionalities via the interface.

The backend is created with Node.js, a server-side JavaScript runtime, and Express.js, a Node.js web application framework. This combination allows for the development of strong and scalable backend services. Express.js simplifies the routing and handling of HTTP requests, while Node.js ensures the server-side logic's speed and performance.

The role-based access control (RBAC) mechanism is a critical component of the Equity Community app. The RBAC system divides users into three roles: user, moderator, and administrator. Within the application, each position has varied rights and privileges. For example, a person may only see announcements, comment and liking them. Moderators, on the other hand, have additional rights, such as posting updates and communicating critical information to the community. Admins are in charge of managing user accounts and can promote a user to the moderator status.

The application is separated into several key sections, which are accessible via a sidebar navigation menu. These components are as follows:

The Feed: This area displays key Equity announcements. Regular users can read the announcements, leave comments, and interact with the content by like it. Moderators have greater access to publish announcements, ensuring that scholars are effectively communicated with.

The Search tool allows all users to search the platform for specific communications, events, or other users. It improves discoverability and helps users find relevant content.

The Forum: Scholars can post their concerns, questions, or discussions in the forum section. Other users can interact with these posts by liking them or suggesting remedies. This encourages active participation in the community and fosters a collaborative environment.

Events: There is a dedicated page for events where moderators can publish forthcoming events. These events are also displayed in the stream for optimal visibility. Scholars can participate in activities, and the system sends them email reminders so they don't miss out.

Chapters: This section contains information on university chapters that are currently available. Scholars can join a chapter and gain access to chapter-specific correspondence and updates.

While the Feed and Forum features are complete, the Search, Chapters, and Events features are still in the works owing to time constraints. The app's modular architecture and micro services approach, on the other hand, ensure that these functionalities can be enhanced and completed in future revisions.

The Equity Community app uses JSON Web Tokens (JWT) for login and authorization to maintain security. When users sign in or sign up, access tokens and refresh tokens are generated. While refresh tokens are safely saved in the database, access tokens contain encoded user information and roles. The app stores and transmits the refresh token using cookies, ensuring safe connection between the frontend and backend services. Token rotation systems are used to refresh access tokens on a regular basis while maintaining a safe environment.

The application's portability and scalability are essential features. The separation of frontend, backend, and database components enables flexibility and easy swapping of alternative technologies if necessary. The software is containerized using Docker, allowing for easy deployment to several cloud platforms and maintaining consistent behavior across all environments. Furthermore, the micro services architecture allows for vertical scaling, which allows the app to accommodate greater traffic by spinning up multiple server instances.

To summarize, the Equity Community app is a robust and secure platform developed with React.js for the frontend, Node.js with Express.js for the backend, and Azure MongoDB for database administration. It offers a variety of services to help the Equity community communicate, collaborate, and interact. The software provides as a solid platform for continuous development and future expansions due to its modular architecture, emphasis on security, and potential for scalability.

Roles of Users

The Equity Community app distinguishes three types of users:

User (ELP Scholar/Wings to Fly): These are the app's regular users, primarily Equity scholars. Users have less access than moderators and administrators, yet they can still actively engage with the platform. They can examine Equity's announcements, comment on them, and like them to demonstrate their interest. Users can also visit the forum section to submit their complaints and questions, as well as interact with the posts of other users.

Moderators: These are the Equity employees that are in charge of providing updates, communication, and critical information. They have more privileges and access than regular users. Moderators can generate and publish announcements on the feed page in addition to the functionality given to users. This enables them to share pertinent information with the scholars, such as events, scholarships, internships, and other updates. Moderators play an important role in ensuring that Equity and the user community communicate effectively.

Admin: The admin job is in charge of maintaining the Equity Community app and managing users. Administrators have access to user-specific features and permissions. They can promote a regular user to the role of moderator, allowing them to contribute to the communication and dissemination process. Admins are also in charge of app maintenance, ensuring that everything runs well and fixing any technical difficulties that may emerge.

Limitations

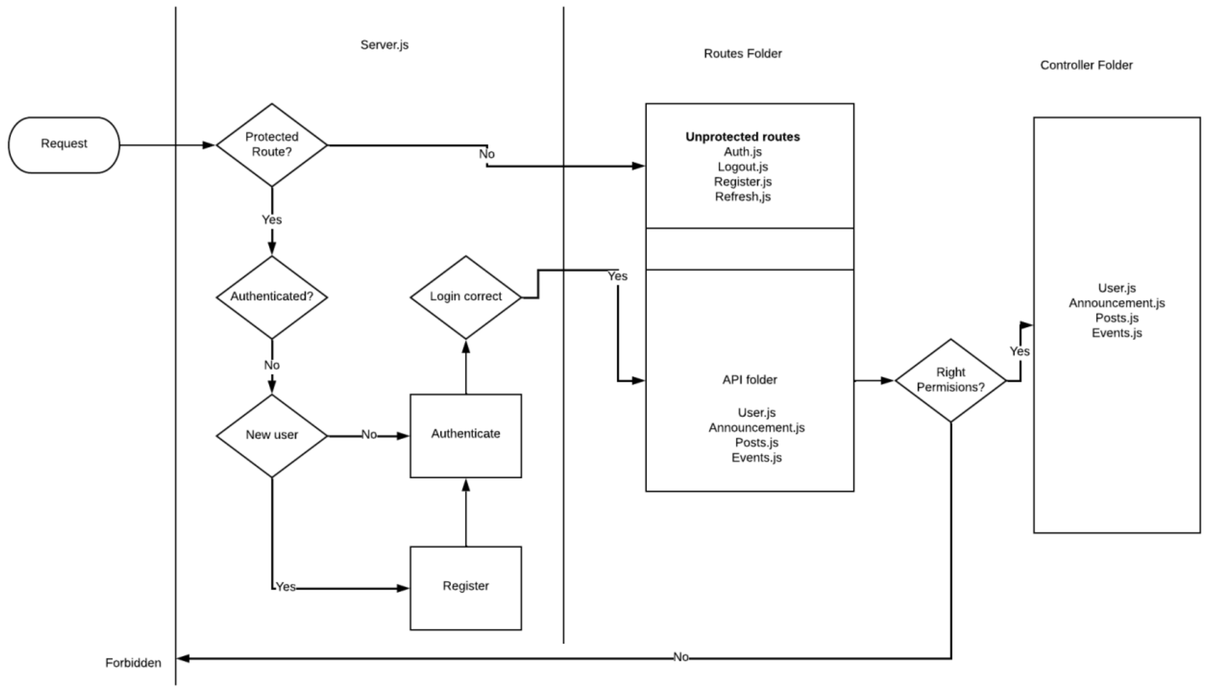
Some Equity Community app functionalities are not fully completed due to time restrictions. The search, chapters, and events features are currently in development and are not yet fully functional. However, the feed and forum's fundamental functions are built to maximize value and stimulate user participation and dialogue.

The development team recognizes the value of these features and may prioritize their completion in future app upgrades or iterations. The app's modular architecture enables easy integration and enhancement of these functionalities as development proceeds.

Backend Architecture

The Equity Community app's backend has a micro services architecture, with discrete services for the frontend, backend, and database components. The backend service, built using Node.js and Express.js, follows the Model-View-Controller (MVC) design paradigm.

How the application is configured?



MVC Design Pattern

The MVC design pattern is used as follows:

1. The **Model** in an Express.js application typically consists of the database and data access layer, which are handled by a database management system such as MongoDB or MySQL. In this application, we chose to use MongoDB. We set up a models folder in the root folder that defines the schema for the objects to be stored in the MongoDB collections.
2. The **View** in an Express.js application is often implemented using a templating engine such as Pug, EJS or Handlebars, which allows developers to generate HTML dynamically. In this case, for scalability, we chose to implement the app as an Application programming interface(API) thus the view and the rendering are to be done by a front end application using React which will consume the API's. Therefore, the application is pluggable and can be used by any frontend service or other existing systems that could consume the API endpoints. This is particularly important because in the case of increased load, many instances of the server can be spun up (vertical scaling) to handle the load and scaled down when the load is low.
3. The **Controller** in an Express.js application is implemented using the routes, which map URLs to functions that handle HTTP requests and responses. The routes can be organized into separate files or modules, and can be used to implement business logic, middleware, or any other functionality required by the application. This is how we set up our controller:

The middleware folder. This folder details any custom middleware that our app would need. These include, logging, error handling, Token authentication and role verification middleware. All these are then imported to different parts of the application.

The controller folder - This folder contains the logic of the applications. Each controller is responsible for the addition, deletion, updating and getting of all entities in the app. The entities are like users, events, posts etc.

The Server,js file is the entry point of the application. It listens for any requests and uses the routes to forward them to the right controller. It initializes the express application and imports the middleware then configures the app to use the middleware. It also details the endpoints exposed by the app. There are protected and unprotected endpoints. The app places the verify token middleware before the protected routes to make sure a user is authenticated before accessing them. For any request, the server forwards them to the API in the routes folder.

The routes folder - This folder contains files that check if the user is authorized to access the routes. For example, it has a User file in the Api folder that imports the verify roles middleware that checks if the sender of that request is authorized to perform that action. It then forwards the request to the controller to perform the action if authorized. If not, it throws an unauthorized error.

Config folder that contains the three roles: Admin, moderator and user, that are to be used by the verify middleware to implement authorization.

A logs folder that collects all the logs and stores then in a txt file.

The Database

We used MongoDB database managed on azure cloud. We deployed the database after which we obtained the database connection string and placed it in a .env file in the root folder. In the server.js file, we wrote a function that connects the application to the database by reading the .env file to get the connection string.

Security

- The Equity Community app emphasizes security through the implementation of JWT (JSON Web Tokens) for authentication and authorization. Key security components include:
- Access Tokens and Refresh Tokens: When users sign in or sign up, unique email addresses and corresponding roles (e.g., user, moderator) are encoded into access tokens. These tokens are sent to the frontend service and used for authentication and authorization. Refresh tokens are also generated and stored in the database for requesting new access tokens.
- Cookie-based Authentication: The server responds with the access token in the response and embeds the refresh token in an HTTP-only cookie. Subsequent requests from the frontend service include the access token, enabling the server to authenticate and authorize users. The use of HTTP-only cookies ensures that the refresh token is not accessible to frontend JavaScript, enhancing security.
- Token Rotation: Access tokens have a limited lifespan (e.g., 30 minutes) to ensure security. When an access token expires, the frontend service uses the refresh token stored in the cookie to request a fresh access token from the server. The server generates a new access token and sends it in the response, maintaining a secure and uninterrupted user experience.

Portability

Separation: we separated the frontend from the backend as well as the database, these components can be swapped for other alternatives. For example, we realized that since we were dealing with people and networks, it would be beneficial to employ a graph database to explore connections between people. This is possible with less hassle because the database exists as a component by itself. Therefore, the application is pluggable and can be used by any frontend service or other existing systems that could consume the API endpoints.

Dockerization: We packaged the application as Docker images and pushed to Docker hub. This enables us to deploy to any cloud on and virtual machine since all the dependencies are packaged with the application. It also allows for versioning for future releases. We published the backend server as a Docker container on azure container service.

Scalability

for scalability, we chose to implement the app as an Application programming interface(API) thus the view and the rendering are to be done by a front end application using React which will consume the API’s. This is particularly important because in the case of increased load, many instances of the server can be spun up (vertical scaling) to handle the load and scaled down when the load is low.

Conclusion

In conclusion, the Equity Community app is a comprehensive solution that leverages modern technologies to create a collaborative and inclusive environment for Equity scholars. The app's frontend, developed with React.js, ensures a smooth and interactive user interface, while the backend, powered by Node.js and Express.js, provides a robust and scalable foundation for handling user requests and business logic. The integration of MongoDB as the database management system enables efficient data storage and retrieval.

The RBAC system implemented in the app ensures that users have appropriate access and permissions based on their roles, promoting a secure and controlled environment. By following the MVC pattern and utilizing middleware and controllers, the app's backend is well-structured and maintainable.

The use of JSON Web Tokens for authentication and authorization enhances the security of the application, ensuring that only authenticated users with valid access tokens can interact with the app's features.

The Equity Community app also demonstrates portability through its separation of frontend, backend, and database components. The micro services architecture and Docker containerization enable easy deployment to various environments and cloud platforms, facilitating scalability and adaptability to changing requirements.

While the app's current functionalities focus on the Feed and Forum sections, the modular design allows for future expansion of features such as search, chapters, and events. With ongoing development and improvement, the Equity Community app will continue to evolve as a powerful tool for communication, information sharing, and engagement within the Equity community.

Overall, the Equity Community app serves as an effective platform for fostering collaboration, providing valuable resources, and promoting a sense of belonging among Equity scholars, moderators, and administrators.