DEPARTMENT OF INFORMATICS & TELECOMMUNICATIONS

HELLENIC REPUBLIC
National and Kapodistrian University of Athens
EST. 1837

Internet Application Technologies

LinkedIn Clone

Ioanna Kontemeniotou

1115202000227

Panagiotis Chatzimichos

1115202000211

Table of Contents

Table of Contents **2**
Description **3**
Installation and Execution **4**
- Backend 4
- Frontend 4

Database Details **5**
Backend Details **5**
- Endpoints 5
- Authorization Tokens 6
- Images and media content 6
- Chat 7
- SSL/TLS 7

Frontend **7**
- Components 7
- Local Storage 9

Summary **9**

Description

This project was developed for the course "Internet Application Technologies." It is an employment-focused social media platform that enables users to connect and interact with others in a professional environment. Users can view posts, like and comment on them, and create their own posts. Additionally, they can apply for jobs, post job listings, and view applicants. The platform also facilitates messaging between connections, and each user has a profile showcasing their skills, experience, and education.

This report will provide a detailed guide on how to use the platform, including instructions for installing and running the code. It will also cover the development process, highlighting the key components used in both the frontend and backend.

Installation and Execution

To run the project, open two terminals and navigate to the respective folders for the backend and frontend. In each terminal, execute the following commands:

Backend

- $ cd backend
- $ openssl req -nodes -new -x509 -keyout privatekey.key -out certificate.crt -days 365
- Fill privatekey.key and certificate.crt path in backend/.env file
- $ npm install
- $ npm run dev

Frontend

- $ cd frontend
- $ npm install
- $ HTTPS=true npm start

Note:
Since the SSL/TLS certificate is self-signed, your browser may display a warning. To proceed, visit the following URLs:

- https://localhost:3000 (Frontend)
- https://localhost:3001 (Backend)

Choose "Proceed" or "Accept the Risk" (depending on your browser) to trust the self-signed certificate.

Database Details

We utilized MongoDB as the database for this project, hosted on MongoDB Atlas (`https://www.mongodb.com/products/platform/atlas-database`), eliminating the need for local installation.

The primary models used in the application are:

- **Users:** Stores user data, connection details, and references to the jobs and posts created by the user.
- **Posts:** Contains the post content, associated media, comments, and likes.
- **Conversation:** Tracks messages exchanged between two users.
- **Job:** Holds information about job listings, including the description, location, and applicants.

Backend Details

Endpoints

The backend of the platform is built using JavaScript, with Node.js and and Express.js handling the routing. Backend endpoints are hosted at localhost:3000/api/ and are categorized into five primary sections, each handling specific functionalities:

- **/api/auth:** Manages user authentication and token handling, with routes for login, logout, and token refresh.
- **/api/users:** Handles user-related operations such as connection management, profile updates, and settings adjustments.
- **/api/posts:** Focuses on post creation, editing, commenting, and interactions related to posts.
- **/api/message:** Manages chat and messaging functionalities between users.
- **/api/job:** Handles job listings, applications, and related job advertisement operations.

For more detailed information on the backend, refer to the backend/requests folder, which contains tests for all endpoints. These tests are designed for use with the "REST Client" VS Code extension but provide a clear overview of the available endpoints and their functionalities.

Authorization Tokens

The platform uses JWT (JSON Web Tokens) for authentication and authorization, implementing an access-refresh token model. Upon successful login, two tokens are issued: an access token and a refresh token.

- **Access Token:** This token is short-lived and is used to authorize requests to access protected resources. It is attached to each request via the `Authorization: Bearer {accessToken}` header.
- **Refresh Token:** This token has a longer lifespan and is used to obtain a new access token when the original one expires.

The backend employs a middleware to protect resources, ensuring that the access token is valid and identifying the user who made the request.

When the access token expires, the frontend automatically makes a request to `/auth/refresh` using the refresh token to obtain a new access token. If the refresh token has also expired, the user is logged out and redirected to the login page for re-authentication.

Images and media content

Images and media files are uploaded to the server and stored in the server's file system. The database stores a reference link to each file, allowing the frontend to retrieve and display the content as needed.

Chat

The chat feature is implemented through backend endpoints that save messages in the database. On the frontend, the conversation is periodically refreshed every few milliseconds to ensure real-time communication between users.

SSL/TLS

Communication between the backend and frontend is secured using HTTPS. A self-signed SSL/TLS certificate, stored on the server, is used for this purpose. As mentioned in the Installation section, users may need to manually accept the certificate in their browser due to its self-signed nature.

Frontend

The frontend of the platform was developed using React.js and CSS. The frontend directory contains both the standard folders installed by React and three custom directories.

1. **Components**: This folder stores all the essential components for the project. Each individual page has its own subfolder, containing the corresponding .js file for functionality and a .css file for styling.
2. **Images:** This folder stores the platform's logo and any other static images used in the frontend.
3. **Services:** This folder contains a utility function named *makeRequest*, which handles all API calls to the backend. It manages the user's access and refresh tokens, ensuring proper authentication for each request.

Components

The platform includes multiple pages that users can navigate to, each serving a distinct function. Navigation is handled using *react-router-dom*. Below is a brief description of each page and how navigation works:

- **Welcome:** The landing page of the platform, accessible at /welcome. It provides introductory content and directs users to either log in or register.
- **Administrator:** The administrator page at /administrator is a dedicated interface for administrators, where all registered users are displayed. Administrators can view user details and have the capability to download user information in both XML and JSON formats, either for all users or for a selected subset.

- **Login:** The login page (/welcome/login) allows existing users to access their accounts.
- **Register:** The registration page (/welcome/register) enables new users to sign up for an account on the platform.
- **Profile:** The user's profile page, located at /profile/:userId, displays the profile information of the user, including skills, experience, and education.
- **Edit Profile:** This page (/edit-profile/:userId) allows users to modify their profile details, such as updating their skills, experience, and education.
- **Notifications:** The notifications page (/notifications) shows a list of notifications relevant to the user, including interactions with posts or job applications.
- **Settings:** This page (/settings) lets users configure personal settings for their account, including preferences and security options.
- **Feed:** The main feed page (/feed) displays posts from other users, where users can interact by liking, commenting, or creating new posts.
- **Network:** The network page (/network) allows users to see their connections with other users on the platform.
- **Jobs:** This page (/jobs) lists job openings where users can apply or post job listings as employers.
- **Applicants:** The applicants page (/applicants) displays a list of candidates who have applied to job listings created by the user.
- **Chat:** The chat page (/chat) enables users to communicate with their connections in real-time.

- **Settings**: The settings page (/settings) allows users to make changes to their account details.

Each page on the platform features a persistent header at the top, offering seamless navigation between key sections such as Feed, Network, Jobs, Chat, and Notifications. Additionally, the header includes a dropdown menu that allows users to easily view their profile, access the settings page, and log out of the platform for secure account management.

Local Storage

The platform securely stores the logged-in user's access token and refresh token in local storage to maintain session authentication. Depending on the requirements of specific pages, additional data, such as the user ID or other relevant information, may also be

temporarily stored in local storage to facilitate navigation and functionality across the platform.

Summary

This project is a full-stack application developed for demonstration purposes, showcasing both frontend and backend integration. We encountered numerous challenges throughout the development process, from implementing real-time communication to handling secure authorization. Through continuous testing, we successfully addressed these issues, ensuring a functional platform.