Professional Networking Application (LinkedIn type)

Nizar Darwish (1115201800286)
Antonios Kalamakis (1115201800056)

Content

1. Introduction

Everything in the exercise was implemented.

Disclaimer: Opening the site you will find a security warning. By pressing "Advanced" and then "Continue" you should be able to advance to the website with no issues. **You should do this with both links:**

1. https://localhost:8080 (frontend)
2. https://localhost:8000 (backend)

The aim of this work is to develop a professional networking application. The users will have access to the application via a modern web browser (Preferred browsers: Google Chrome and Opera).
The project has been implemented in Vue for the front-end and Nodejs (Restful-Api) for the back-end, using MongoDB as a database. In the following sections we will discuss how to run the project, details about the subpages, functionalities of the website, difficulties and challenges we encountered during the implementation.
(It's good to mention that all the popups are closed by pressing the top left X button.)

2. Commands to execute the servers

The project files include all the modules and libraries you need. But if the server is not running due to missing modules run **npm install -g npm** to install the latest npm version.

a. To run the front-end (Vue code) type the command **npm run serve**. The port, the protocol and the address that it listens to will be printed in the console. (You might need to use this command: npm install vue@next to install some of vue's core modules.)
b. To run the back-end (Nodejs code) type the command **npm start**. The server address will be printed in the console.

Run the two servers in different terminals

3. LogIn and SignUp page

a. The user home page has a sign in and register tab, on register every field is required except phone number and profile picture, those could be set on register or/and on the settings tab.

b. When logging in, if the email and password are correct then return the user's info. When registering, make a new user and return the user's info.

4. Main Page

a. The main page has 3 compartments, the top compartment is the navigation bar, the middle compartment is different for each tab and will be analyzed later and the bottom is a footer for the page with some dummy data popup when pressing the terms of use item. Used only for cosmetic purposes. The navigation bar has a search bar for users (once you hover over the search button a search input slides into view). This action works both by pressing the enter button or pressing the search icon. In addition to that. At the left we have a small segment indicating some of the users information and a connections button redirecting to the user's network tab.Finally its good to mention that the left compartment is set to sticky so even when the user scrolls down the compartment is always into view. After that there is an input compartment used to generate new posts, this compartment accepts photos, voice files and video files by pressing the desired media button and as the user appends it there is a preview available showing each image, video and voice file a user has provided. The images and videos are in a group of maximum 4 and if the user wants to preview more than 4 files all there is to do is click on an image and all of the media will popup. In more depth about the post printing, the media files and the text are viewed exactly as previewed while generating the post, each post has a like or dislike button and a comments section. To post a comment we need to press the comment button to print all post's comments and type the new comment we want to append. After that the comment will be only posted with a click of the enter button.
b. Once logged in, his posts, the posts of the users he has connected with, the posts that they liked and the recommended posts from the Matrix Factorization Collaborative Filtering will be presented. The Posts database json model is the following: The author, text, pictures, videos, voice recordings (if uploaded), name, avatar, likes (an array of users), comments (an array of users, text, name, avatar and date) and the date the post was created.

5. Network Page

a. In this page the user can check the users he connected with. The user's photo, name and other information are presented. By clicking on a user, we are redirected to the user's profile, since we are connected with the user every information is available to us regardless of whether the information is private or public. On the contrary, if we found the user's profile by using the search bar and are not connected with the user, only the public information will be

presented to us and the private information will be hidden. In addition to that, we can send a connection request to the user or if we are already connected we could start a conversation with them.

b. When navigating to the Network Page a get request will be made for the connected users. By clicking on a user, a “get profile” request is made and the appropriate information is given to be presented. The Users database json model is the following: The first name, last name, email, number, password, profile picture, education, skills, experience, connected users, pending connection requests, liked posts, notifications (likes and comments).

6. Applications Page

a. Navigating through this page you will find 3 tabs. The “My jobs” tab, where you can post a new job, each job is required to have some description and at least one skill to be accepted to post. Below the job posting compartment you can see every job you have posted. Each job has a button that redirects to the “Requests tab” showing all the applications made for this job. By clicking on an application, a popup will appear showing the applicant's description, the skills he listed and his resume file. At the “Jobs” tab you will find jobs from other users and their information (description). Each job has an “intrest” button opening a popup for the user to add all the necessary information and apply for the job.
b. When clicking the “Jobs” tab, the jobs based on the skills he has declared in his personal data, the job posts of the users he has connected with and the recommended jobs from the Matrix Factorization Collaborative Filtering will be presented. The Job database json model is the following: The author, name, avatar, description, an array of skills and an array of the users applications.

7. Communications Page

a. Here we again have 2 compartments, the left compartment shows a list with all the users that we have started a conversation with. Additionally, we can start a new conversation by clicking the chat icon, it opens a popup search bar, where you can search for any user you have connected and want to chat with. After selecting a user a chat will appear, at the right compartment, showing the conversation and all the previous messages (if any).
b. The Chat database json model is the following: The chaters (an array of users) and the messages (an array of messages). For each message we store the sender, the text, the avatar of the sender and the date the message was sent.

8. Personal Information Page

This profile page is to preview and edit information. The user can update his profile picture, latest work experience, educational background and his list of skills. For each information(except profile picture) there is an edit button at the right side. Once clicked, you can edit the information and choose whether to set it to public or private. When editing the work experience you can see all the history of updates (not only the latest work experience).

9. Settings Page

In settings you have the option to change your information like email, phone number, password and profile picture (by pressing your current picture). The user's email input element is set to disabled since it's pre-filled with information but if the user clicks on the element the email gets cleared and the user can now set the new email if needed. For each change made, it’s necessary to click the update button (only the changed fields will get updated). There is a reset option in case the user doesn't want to apply the changes.

By pressing the logout button, the user will be redirected to the login page.
If the user is not logged in the user won't have access to any of the subpages and will be redirected to the login page. On the other hand, if he is already logged in, he will stay logged in (even if he closed the browser tab) until he logs out.

10. Admin Page

A single user can access the admin page. At the top of the admin page we have a search bar searching by user name or by id. The name search can be written approximately, but when searching by the id, the whole user’s id must be written correctly. Additionally, in order to generate the JSON or XML file we need to click the “select” button and click the users we want to select and the type of the file you want to download. Afterwards by clicking the “export” button the JSON or XML file will be downloaded.
When the “select” button is not clicked, we can click on a user and we get a popup with all the user’s information.

It's good to mention that on login the user is redirected to the admin page but could also go to the other pages as a normal user by changing the link to a /user etc. This is done so the admin can be a normal user if it's desired. Also the admin can logout by /logout or by going to the main site and hitting the logout button.

Conclusion

Nizar Darwish: Implemented the backend
Antonios Kalamakis: Implemented the frontend

Some of the difficulties we faced on the frontend's point of view were to make every subpage's compartments automatically get resized every time the browser is resized as well. In case, the browser window is not big enough to fit all the compartments, each compartment has a minimum size so its content won't wrap and the page will overflow so the user can scroll to review the rest of the page. Another difficulty was to make the preview of the post's media, so it's dynamic in case the user wants to add multiple files and to present them in an organized way.
On the other hand, the backend had its own challenges as well. One of them was the database structure. The models had to be thought of very carefully so we don't have any duplicate data in our database as well as having flexibility in inserting and updating the data. Not to mention that, when presenting the posts to the user, it was a challenge to go through the different models and to correlate them in order to recommend the right posts to the user. Ofcourse, the Matrix Factorization Collaborative Filtering algorithm was one of the hardest things because the implementation had to be made from scratch.
This project was really fun overall and a great challenge!!! :)