



From Profile to Paper: Creating Dynamic Resumes with LinkedIn Integration

Link2Resume

Md Shayemur Rahman

BACHELOR'S THESIS
June 2024

Degree programme in Software Engineering

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Bachelor's the Degree Programme
Software Engineering

Md Shayemur Rahman
From Profile to Paper: Creating Dynamic Resumes with LinkedIn Integration

Bachelor's thesis 29 pages, appendices 2 pages
June 2024

In today's competitive job market, it's crucial to effectively showcase your professional achievements to stand out and advance in your career. This project aims to create a web application that quickly generates CVs using LinkedIn OAuth 2.0 for authentication. By seamlessly integrating with the LinkedIn API, users can easily import their essential profile information for accurate and comprehensive CVs. The app includes a form for users to input their information and easily create PDF versions of their resumes for editing and sharing. This model demonstrates the power of OAuth 2.0 for integrating third-party APIs in CV creation while addressing data privacy, security, and control concerns. Leveraging the latest web technologies, this application represents a significant step towards digitizing professional documentation and empowering individuals to take control of their professional narrative.

CONTENTS

1	INTRODUCTION	5
2	Background and Purpose	7
2.1	Emergence of Dynamic Resume Generation Tools	7
2.2	Leveraging LinkedIn Data for Personalized Resumes	7
2.3	Purpose of the Application	7
3	Technologies	9
3.1	Node.js.....	9
3.2	React.js	9
3.3	MongoDB.....	10
3.4	LinkedIn Authentication and Authorization	10
4	Features.....	11
5	Implementation of LinkedIn to Resume.....	12
5.1	Web Application Architecture	12
5.2	LinkedIn Authentication Integration	15
5.3	User Profile Management	17
6	Challenges and Future Improvements	28
7	DISCUSSIONS	28
	REFERENCES	29
	APPENDICES.....	30
Appendix 1.	Source Code.....	30
Appendix 2.	Final View of The Application.....	30

GLOSSARY

TAMK cr	Tampere University of Applied Sciences. credit.
API	An Application Programming Interface is a set of rules and protocols that enable seamless interaction between different pieces of software.
CRUD	Create, Read, Update, Delete – the four basic functions of persistent storage.
CV	Curriculum Vitae, a detailed document outlining a person's educational background, work experience, skills, and accomplishments, typically used for job applications.
HTTP	HyperText Transfer Protocol – the foundation of data communication for the World Wide Web.
JSON	JavaScript Object Notation – a lightweight data-interchange format that's easy for humans to read and write, and easy for machines to parse and generate.
OAuth	OAuth stands for Open Authorization which enables websites and applications to access a user's information without the need to share their password.
ODM	It is Object-Document Mapping which facilitates mapping between objects in code and documents in a NoSQL database collection.
URL	Uniform Resource Locator helps web browsers locate specific resources on the web.
UI	User Interface – the space where interactions between humans and machines occur.
URL	Uniform Resource Locator helps web browsers locate specific resources on the web.

1 INTRODUCTION

As the jobs are getting more competitive these days, how we present our professional achievements and expertise is more important for our career growth. A good resume shows your experience as well as separates you from other candidates. The creation of a CV has always been more craft than science, putting together myriad data points in a well-ordered, professional package used to take hours of aggregation and promotion from scratch. But with the rise of sophisticated web apps and social media integrations, this process has become much more convenient.

This thesis describes the design and construction of a web application using LinkedIn OAuth 2.0 authentication to help users create and modify CVs. When users authenticate their identity using the LinkedIn API, they are effortlessly able to pull in their core professional profile including full name, email, avatar, headline, and LinkedIn URL. In addition to saving time, this integration guarantees the accuracy and completeness of the core details as they are pulled directly from a user's LinkedIn profile.

Due to limitations in LinkedIn's API access, the application cannot retrieve detailed profile information such as work experience, education, and skills directly. To fix this, the application incorporates a user-friendly form that is very easy to use allowing users to insert this additional data manually. By combining information filled in by LinkedIn and those added manually, this hybrid model makes sure that users can create a richer and more thorough CV.

This web application is built to bring users an interface for easily creating beautiful and content-rich CVs. The imported and manually entered data can be edited and customized by users to suit the job they are applying for, keeping only the most relevant experiences and skills for the role in question on each document. Moreover, the app allows you to export the customized CV to PDF for easier sharing and printing.

The importance of this project lies in its potential to transform the way individuals proceed towards CV creation. Users who keep their profiles updated are more likely to be discovered by recruiters, demonstrating the importance of maintaining accurate and comprehensive professional information online. This application allows users to easily export their online profiles into professional documents by leveraging the authentication and basic data retrieval capabilities of LinkedIn.

Besides its utilitarian purpose, this is another piece that might add to the web development world and application architectures. The application demonstrates how OAuth 2.0 can secure user authentication and the benefits of API integration for a superior user experience and functionality. In addition to the above, the application also addresses fundamental concerns related to data privacy and security, and thus, manages user information in a secure and ensures compliance with industry standards.

The remaining sections of the thesis will detail the technical aspects of the implementation, including the authentication process with LinkedIn OAuth 2.0, the extraction and processing of the user data from the LinkedIn API, and the creation of the manual data entry form. We will further look at the creation of printable PDF resumes and their usability. This study has been designed to show from a thorough analysis and evaluation that the integration of social media data plus tripartite data into a web-based CV creation tool is not just feasible but also a highly effective method for creating comprehensive CVs.

LinkedIn the emergence of such a resume-creation app which is integrated with LinkedIn, is a huge step towards the digitalization of professional documentation. By doing all of this it not only removes complications for the user but uses the latest web technologies to provide a comprehensive, and proactive solution. As the job market shifts and changes, platforms such as this will become ever more necessary in helping people to both tell and control, their professional stories.

2 Background and Purpose

This section provides an overview of the evolution of resume creation and method and the role of LinkedIn data in personalized resumes.

2.1 Emergence of Dynamic Resume Generation Tools

Dynamic resume-generation tools simplify the process of creating professional resumes for individuals offering an efficient way to craft resumes by providing essential resume elements and guiding users through the process (Shivhare et al. 2024). These tools make use of input data in the system to fit up the content automatically so it can be more related and aligned with the job requirements. It demonstrates the broader trend of more personalized and efficient job-hunting methods, indicative of changing recruitment practices in a candidate-driven market.

2.2 Leveraging LinkedIn Data for Personalized Resumes

LinkedIn's platform offers a vast repository of professional data, serving as a valuable resource for creating personalized resumes. Integration with the LinkedIn resume tool allows users to seamlessly import relevant profile information, thereby streamlining the resume creation process. This integration enhances the efficiency of resume development and ensures the completeness and accuracy of content by leveraging the comprehensive data available on the platform. By incorporating LinkedIn data, users can align their resumes with their professional brand, effectively presenting their qualifications to potential employers. Furthermore, maintaining a connection to users' LinkedIn profiles enables automatic updates to the resume, ensuring their ongoing relevance and impact on job applications.

2.3 Purpose of the Application

The web app aims to help users create and manage professional resumes using LinkedIn's network data through its API & OAuth 2.0 service. The app lets users import their LinkedIn profile details to ensure accuracy and make it easier to

create well-crafted resumes. It also automatically updates resumes with any changes in the user's LinkedIn profile, ensuring that job applications always feature up-to-date information.

This web application was created to make the resume generation process easier, by using the information from LinkedIn integrated with the user input. Users can sign in with LinkedIn, upload profile basics, and fill in the rest of the CV details — work experience, education, skills, etc. The web application allows users to customize their resumes, generate them dynamically, and download them in PDF format. The user interface is a React-based single-page application, while the server side is handled by a Node.js backend API. The technology was chosen for its efficiency in creating complex, highly customizable UI and for its powerful, high-performance, and scalable server-side logic. Data Storage MongoDB is used for data storage which gives a database solution with many advantages that can be adaptive and scalable to meet complex data structure and relationships. The application itself is styled using Material UI, which provides a broad collection of components for creating stylish and coordinated interfaces of a more modern nature. This also means that the integration with LinkedIn's API and OAuth 2.0 service allows users to authenticate, generate, and store resumes securely.

3 Technologies

This section explores the technologies employed in web application development, highlighting their roles and contributions to the overall functionality and user experience.

3.1 Node.js

Node.js is one of the commonly used environments for server-side development. Node.js is an open-source JavaScript scalable development environment that allows users to build full-scale web applications (Metwalli, 2023). It is used to ensure the application is quick and responsive, which are key requirements to respond at scale to multiple user requests concurrently — all of course powered by Node.js. With the wide ecosystem of packages and modules via npm (Node Package Manager), Node.js makes it easier to develop the integration of different logical functionalities like authentication, data manipulation, and API interactions of the application. Additionally, the extensive ecosystem of packages and modules available through npm simplifies the development and integration of various functionalities, including authentication and API interactions.

3.2 React.js

React.js is an open-source JavaScript library, crafted with precision by Facebook, that aims to simplify the intricate process of building interactive user interfaces. Imagine a user interface built with React as a collection of components, each responsible for outputting a small, reusable piece of HTML code (Herbert, 2023). The high performance, scalability, and reactivity of this JavaScript library make it capable of enabling the construction of large enterprise-level projects from single-page applications to blogging websites. Reusable components are offered by React, making collaboration easier for large teams. This means that components such as functions, classes, or objects can be reused across applications, reducing the need to rewrite the same code (Bhatt, 2024).

3.3 MongoDB

MongoDB is a cross-platform document-oriented database program. It is classified as a NoSQL database and uses JSON-like documents with optional schemas. (Kugan, 2022). It is the most popular document database, used by major companies like Google, Facebook, and Forbes. It stores data in a binary encoded JSON format (BSON), which offers better data type support and improved indexing and querying capabilities (MongoDB, n.d.). Because of its distributed nature, MongoDB can store and retrieve large amounts of data quickly and efficiently, making it reliable in high-load conditions. In addition, the query language of MongoDB and indexed support provide optimized data retrieval which helps in faster database operations that result from your application getting performance optimization.

3.4 LinkedIn Authentication and Authorization

The web application is connected to the LinkedIn API, and the OAuth 2.0 service since it uses the OAuth 2.0 protocol to authenticate and authorize users who want to log in with their LinkedIn credentials. OAuth 2.0 is a powerful and secure framework that allows different applications to securely interact with each other on behalf of users without sharing sensitive credentials (BYTEBYTEGO, 2023). With OAuth 2.0, user credentials are not stored on the application side, enhancing security. Once a user is successfully authenticated, their session is secured with session tokens and cookies, allowing for continuous login sessions and maintaining the user's authentication state across interactions with the application. This improves the security and performance of the application.

4 Features

LinkedIn Integration: The application is directly integrated with LinkedIn's OAuth 2.0 authentication and LinkedIn API. This will enable users to access their professional profile data such as name, headline, email, avatar, and LinkedIn URL.

Complete CV Creation: users can create comprehensive resumes by combining the content they imported from LinkedIn with manual information. This makes sure that CVs are correct, and current which is personalized to the user based on her history/professional experiences and qualifications.

User-Friendly Interface: The user interface is developed using React and it provides a great UI for generating CVs. Users can effortlessly modify and personalize their CVs to pitch their most relevant experiences and skills for each role.

PDF Export: The application allows users to export their customized CVs to PDF format, simplifying the process of sharing and printing resumes for job applications. This feature makes the application more user-friendly and flexible.

Automatic Updates: Users' CVs are automatically updated whenever there are changes to their LinkedIn profiles, ensuring that their resumes are always accurate and up-to-date.

Secure Authentication: The application used OAuth 2.0 to authenticate and authorize users securely, which means the user credentials are kept safe from others, and the user credentials can be verified before accessing the application.

Data Privacy and Security: The application follows best practices for data privacy and security management making sure any relevant information is stored in compliance with industry regulations.

5 Implementation of LinkedIn to Resume

In this section, we delve into the technical implementation details of the integration of LinkedIn that allows users to log in with their LinkedIn credentials and generate resumes based on their LinkedIn profiles.

5.1 Web Application Architecture

Our application uses express.js on the server side which is a minimalist web application framework of node.js. Express.js enables creating RESTful API which simplifies the communication between the client-servers. We used express.js for initiating LinkedIn authentication, handling LinkedIn callback URLs, and managing the profile data of a user by using CRUD operations.

In Figure 1, we describe an example breakdown of server-side application architecture

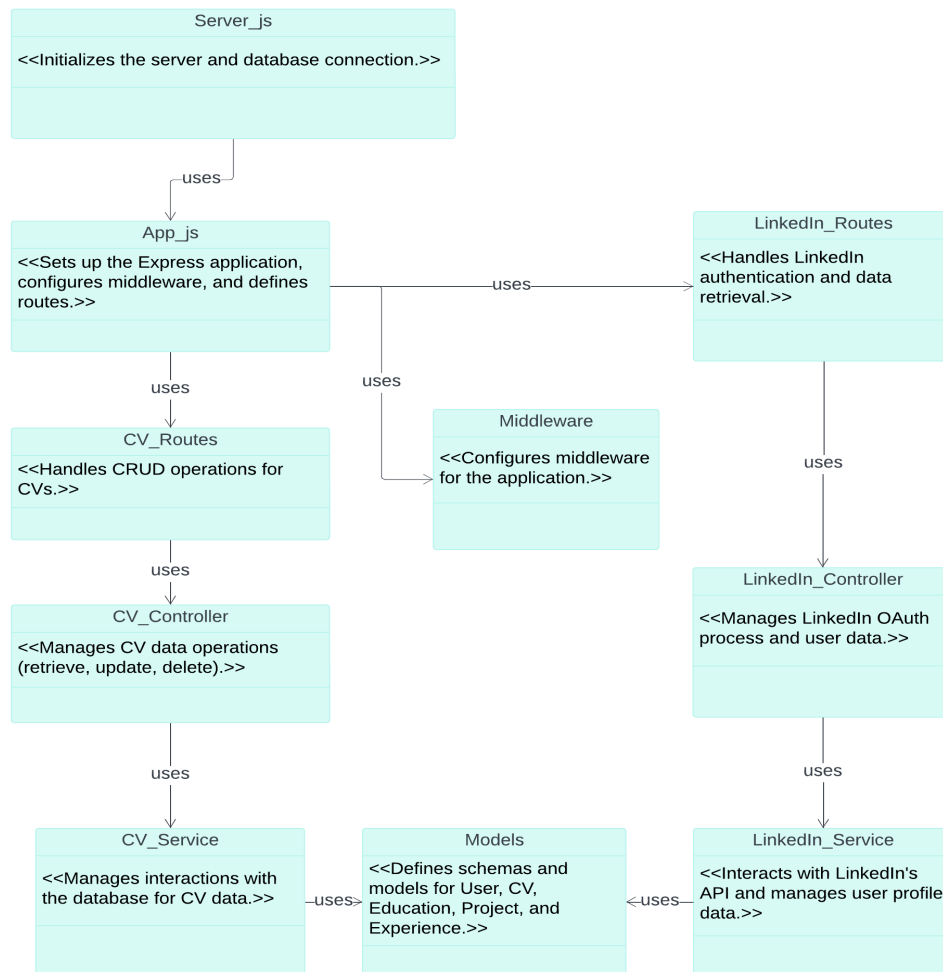


FIGURE 1. Flowchart of server-side application architecture.

Our application uses React.js for client-side rendering. React.js enables the creation of a seamless and interactive user interface. The components of React.js encapsulate and enable modular development and enhance maintainability and reusability.

In Figure 2, we describe an example breakdown of react components used in client-side:

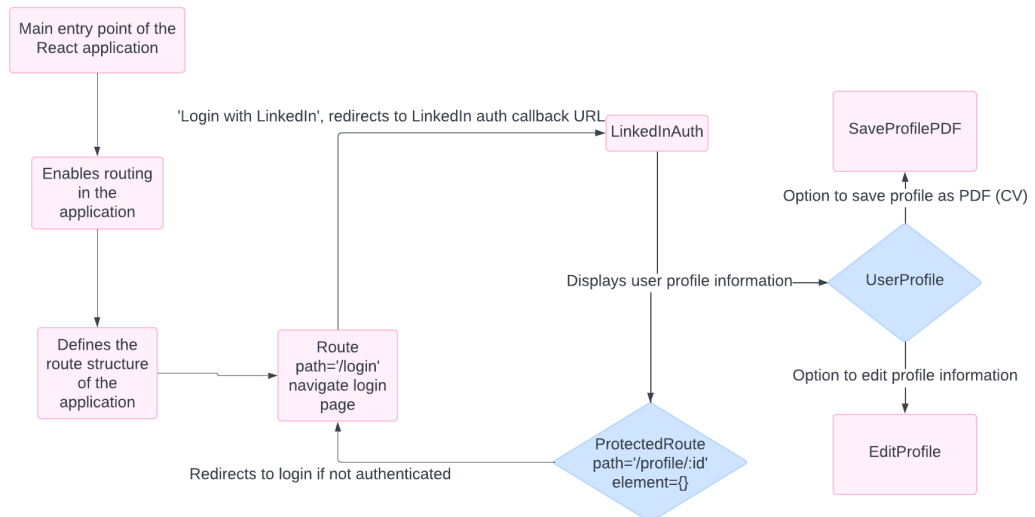


FIGURE 2. Client-side component level breakdown

Our chosen database management system is MongoDB, a very popular NoSQL database. MongoDB is known for its flexibility, scalability, and ease of use.

The code initializes a connection to MongoDB using Mongoose which is an ODM library for MongoDB and Node.js.

In Figure 3, we describe an example breakdown of the Database Relationship between schemas.

Database ER Diagram

Link2Resume | May 20, 2024

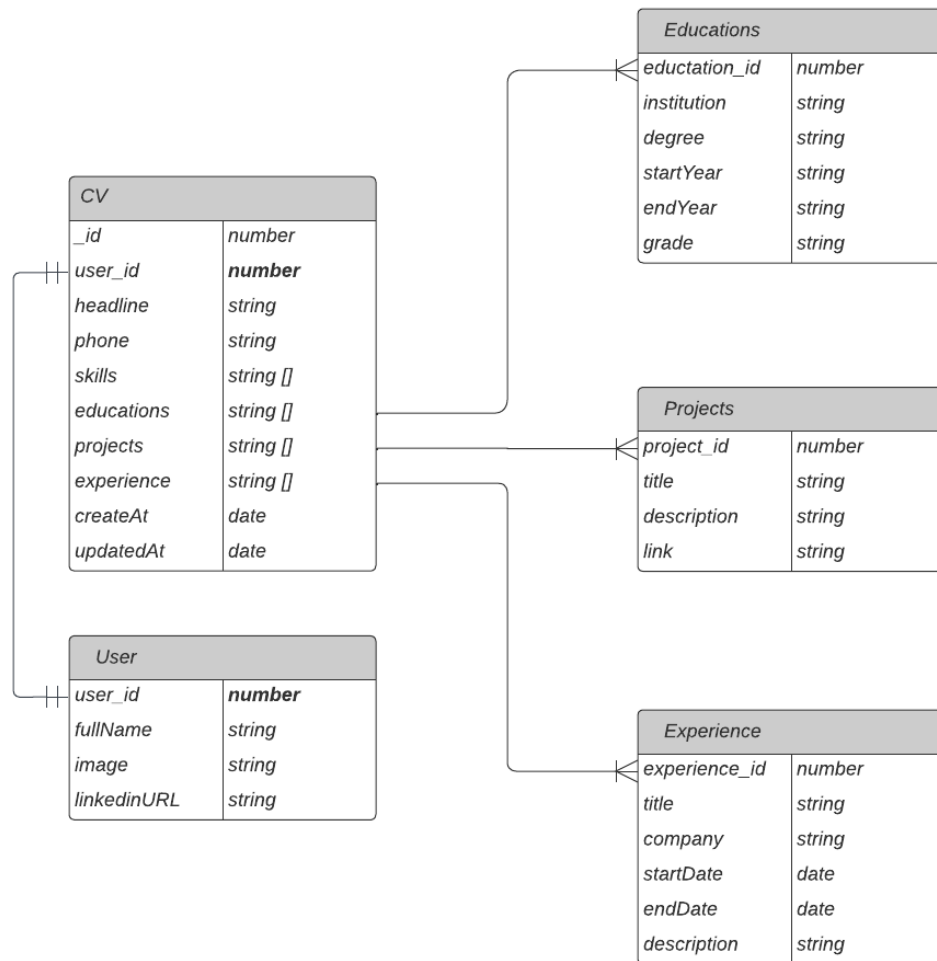


FIGURE 3. Database Schema design diagram.

5.2 LinkedIn Authentication Integration

LinkedIn Authentication allows users to log into our web application securely with the respective configuration, our application provides an easy-to-use and secure user authentication feature based on LinkedIn OAuth 2.0 protocol. In this section, we will discuss the basics of what makes LinkedIn integration into our project, including which pieces and steps are needed.

In Figure 4, we describe the login and creating a CV generation process.

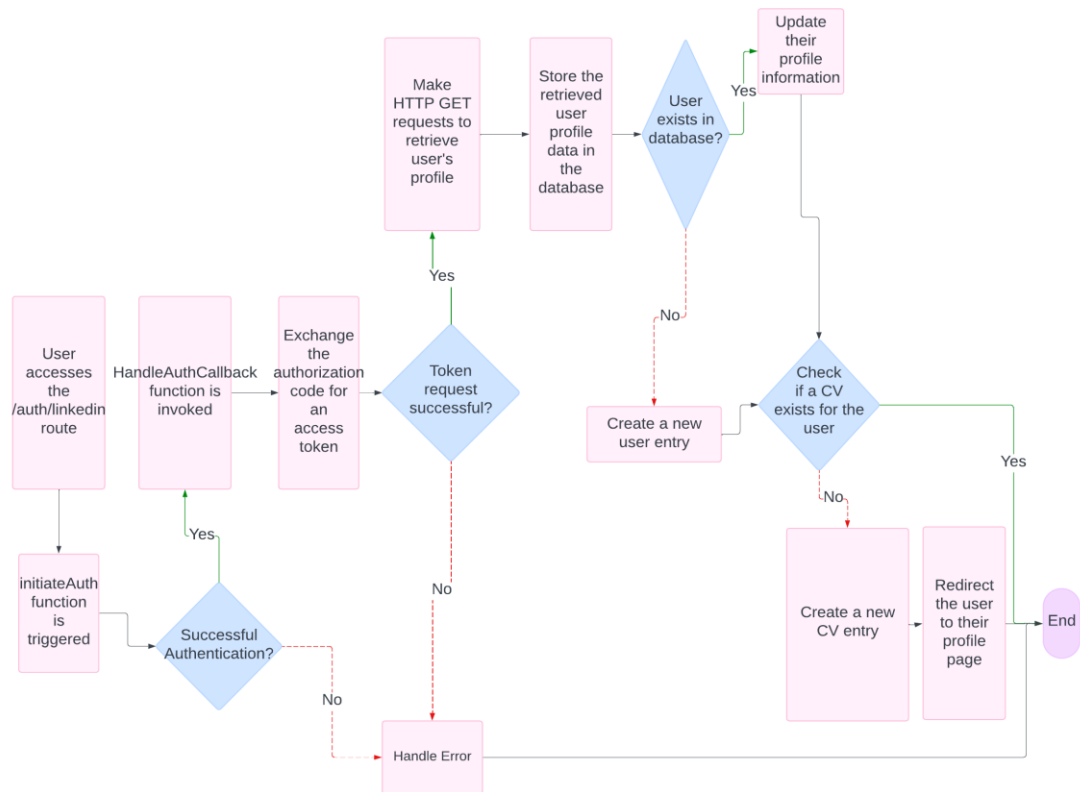


FIGURE 4. User Login and CV Generation Process with LinkedIn Integration.

The user login and CV creation process with LinkedIn integration makes the login to the application faster and easier for the users as well as automates CV creation depending on the information provided for their LinkedIn profile. There is a multi-step process that verifies authentication securely:

1. When a user tries to log in through LinkedIn, the application initiates a function to create a special URL with the required parameters.
2. The user is then redirected to LinkedIn's secure authentication page.
3. Upon successful authentication on LinkedIn, the user is redirected back to our application's designated callback URL.
4. Our application extracts an authorization code from the callback URL and exchanges it for an access token.
5. With an access token, the application makes an HTTP request to retrieve profile information.
6. The retrieved profile data includes the user's full name, headline, profile image, email, and LinkedIn URL.

7. The application then checks if the user already exists on the database based on email. If the user exists, it makes sure the profile information is updated in the database. If not exist, it creates a new user in the database.

5.3 User Profile Management

User Management provides only authorized users to access the application and perform activities based on their roles. Furthermore, user management also serves the role of improving user experience by adding account customization or making sure their data is safe and secure.

To manage user profiles, our application utilizes OAuth 2.0 for user authentication and authorization. This enables users to access their data from service providers such as Google or Facebook without having to directly share their user credentials. With the implementation of OAuth 2.0, we are ensuring a highly secure method for managing user authentication and authorization.

Our application involves several key steps for handling user authentication and authorization. Here are some key steps discussed below:

First, we must create an application on the LinkedIn developer website to ensure access to the LinkedIn profile endpoint. This involves navigating to the LinkedIn Developer Portal, signing in with LinkedIn credentials, and creating a new app (LinkedIn Developer Solutions, 2023). This app will provide the necessary credentials and access tokens to interact with LinkedIn's API.

In Figure 5, we demonstrate the creation of a new application in the LinkedIn Developer Portal.

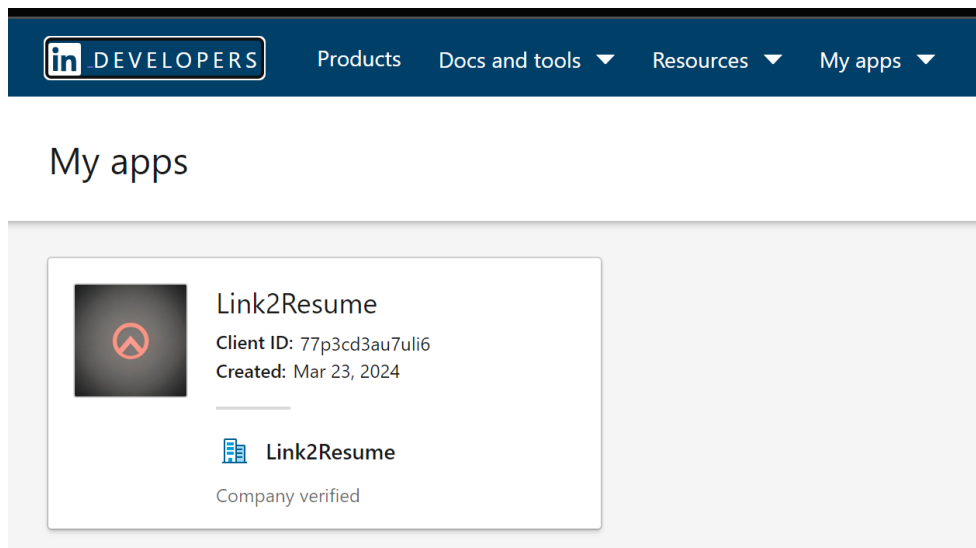


FIGURE 5. Create a new app in the LinkedIn Developer Portal

To effectively manage user profiles and access necessary information from LinkedIn, our application requires specific permissions. These permissions are obtained by requesting access to certain LinkedIn API products. For user profile management we need permissions such as `openid`, `r_basicprofile`, `email` etc., depending on the application requirements. We need to request access to the following products at least to obtain these permissions.

In Figure 6, we demonstrate the required permission we need to obtain from the LinkedIn Developer Portal.

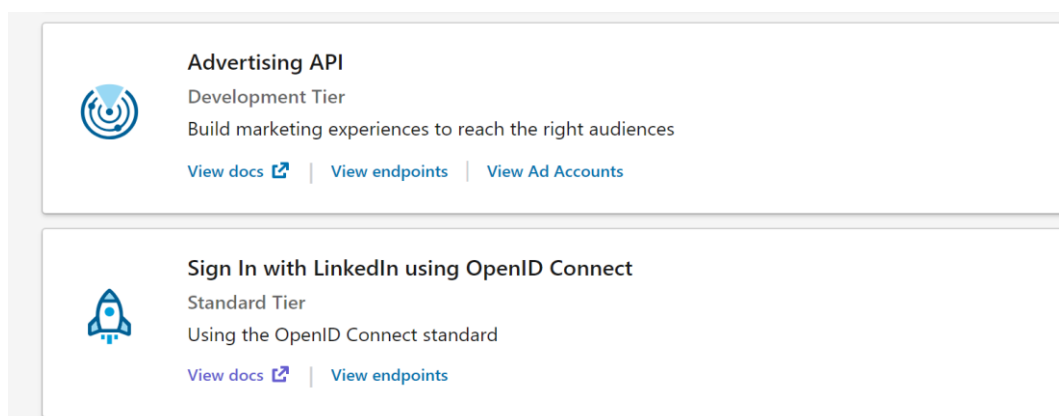


FIGURE 6. LinkedIn API Products Required for User Profile Management

We are required to apply for access to specific LinkedIn API products to obtain the necessary permissions. The image in Figure 6 shows the products that we need, which are "Sign in with LinkedIn" and "Share on LinkedIn." By securing

access to these products, our application will be able to smoothly incorporate LinkedIn's authentication and data retrieval features.

Once we register and are approved for access to our LinkedIn application, we can obtain our application credentials by navigating to the "Auth" tab in our LinkedIn app settings. The basic credentials are your `client_id` and `client_secret`. These credentials which are specific to our application are what we'll use to uniquely identify and authenticate our application whenever we make an API request to LinkedIn.

Now that we have the application credentials and redirect URLs properly configured, we are all set to integrate LinkedIn OAuth 2.0 into our application. This integration will enable users to log in using their LinkedIn accounts and will give our application access to their profile information, as allowed by the requested permissions.

Before handling the LinkedIn OAuth 2.0 workflow, we need to set up the Node.js environment. For setting up Node.js, we need to make sure that Node.js and npm are installed on our machine. We can download and install them from the official Node.js website. After installation, we should initialize a new Node.js project by creating a new project directory and running npm init to generate a package.json file. This can be done with the following commands:

```
mkdir server  
cd server  
npm init -y
```

Next, we need to install the necessary packages for our project. These include Express for server-side handling, Axios for making HTTP requests, and Mongoose for integrating MongoDB with Node.js. We can install these packages by running: `npm install express axios mongoose dotenv`

We need to make sure to configure your environment variables by creating a `.env` file in your project directory. This file will store sensitive information such as your LinkedIn client ID, client secret, callback URL, and other necessary secrets.

Keeping this information in a separate file will help you maintain security and manage your configuration more effectively.

```
LINKEDIN_CLIENT_ID = <your_linkedin_client_id>;  
LINKEDIN_CLIENT_SECRET = <your_linkedin_client_secret>;  
LINKEDIN_CALLBACK_URL = <your_callback_url>;  
DB_URL = <your_mongodb_connection_url>;
```

Now we need to set up our Express server to use the necessary middleware and configure routes for handling LinkedIn OAuth 2.0 authentication in a structured manner.

We need to prepare the `server.js` file for this. This file is responsible for initializing the server and establishing the connection to the MongoDB database. To achieve this, we first establish the port on which the server will be running. Then we take the connection URL from the environment variable to connect to the MongoDB database, and finally, we listen for incoming requests on the specified port.

In Figure 7, we demonstrate initializing the server and establishing a connection with MongoDB in the `server.js` file.

A screenshot of a code editor with a dark background and light-colored text. The code is as follows:

```
1 const port = process.env.PORT || 5000;  
2  
3 const mongoURL = process.env.DB_URL;  
4 mongoose.connect(mongoURL).then(() => console.log('Connected!'));  
5  
6 app.listen(port, () => {  
7   console.log(`🚀 Server is running on http://localhost:${port}`);  
8 });
```

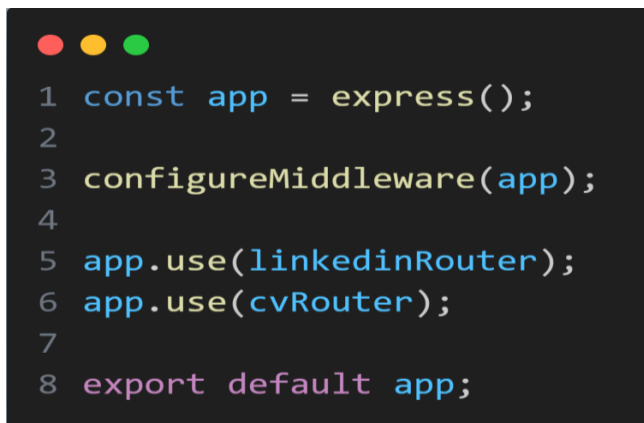
FIGURE 7. `Server.js` file

We'll create an instance of the Express application and configure middleware in a dedicated module called `configureMiddleware.js`. This includes enabling Cross-Origin Resource Sharing (CORS), configuring body parsing middleware, and session management for user authentication.

We'll define two separate routers: `linkedinRouter` for LinkedIn authentication and `cvRouter` for generating and managing user CVs based on their LinkedIn profile information.

Throughout the process, we'll make use of the powerful features provided by Node.js and Express to handle HTTP requests, integrate with external APIs and databases, and simplify the development process.

In Figure 8, in the app.js file we used the necessary middleware for LinkedIn authentication and CV routes.

A screenshot of a code editor showing JavaScript code in a dark theme. The code is as follows:

```
1 const app = express();
2
3 configureMiddleware(app);
4
5 app.use(linkedinRouter);
6 app.use(cvRouter);
7
8 export default app;
```

FIGURE 8. Define middleware in App.js

In our project, we'll create a file called `configureMiddleware.js` to centralize the setup and configuration of middleware components. One important component we'll configure is Cross-Origin Resource Sharing (CORS) to enable seamless communication between the front-end application and the server, regardless of their origins. Also, we used sessions that allow the app to log out after a specific time and keep user data secure.

In Figure 9, in `configureMiddleware.js` we used CORS for connecting with User Interface and session for automatic logout to keep the user data secure.

```
1 const configureMiddleware = (app) => {
2   app.use(cors({ origin: 'http://localhost:3000' }));
3   app.use(express.json());
4   app.use(express.urlencoded({ extended: true }));
5
6   app.use(
7     session({
8       secret: config.SESSION_SECRET,
9       resave: false,
10      saveUninitialized: false,
11      cookie: {
12        maxAge: 5 * 60 * 1000,
13        secure: false,
14      },
15    })
16  );
17 };
18
19 export default configureMiddleware;
```

FIGURE 9. Define CORS and Session-Cookies

The `linkedinRouter.js` file manages the authentication process by creating routes for initiating authentication, handling the callback response, and facilitating user logout. When a user navigates to the authentication route, the `initiateAuth` function constructs the LinkedIn authorization URL with the necessary query parameters. The user is then redirected to LinkedIn, where they can grant permission for our application to access their profile information. Upon consent, LinkedIn redirects the user back to our application with an authorization code.

In Figure 10, we demonstrated `linkedinRouter.js` file is crucial for LinkedIn authentication steps.

```
1 const linkedinRouter = express.Router();
2
3 linkedinRouter.get('/auth/linkedin', linkedinAuthService.initiateAuth);
4 linkedinRouter.get(
5   '/auth/linkedin/callback',
6   linkedinAuthService.handleAuthCallback
7 );
8 linkedinRouter.post('/logout', linkedinAuthService.handleLogout);
9
10 export default linkedinRouter;
```

FIGURE 10. Router file involving LinkedIn authentication steps.

In the `linkedinAuthService.js` file, we begin the authentication process by directing users to LinkedIn's authorization endpoint. Here, we request users to grant permission for our application to access their LinkedIn profile data. We include various parameters such as the response type, client ID, redirect URI, and scope (which defines the level of access requested) in the URL.

After users grant permission and return to our application, we handle the callback. We extract the authorization code from the query parameters in the callback URL. This code is then exchanged for an access token by sending a POST request to LinkedIn's token endpoint. The access token acts as a credential, enabling our application to retrieve the user's LinkedIn data on their behalf.

Upon obtaining the access token, we use it to retrieve the user's profile information from LinkedIn's API. This includes details such as the user's full name, headline, profile picture, email address, and LinkedIn URL. If the user is new to our application, we store their profile information in our database. If they are an existing user, we update their profile accordingly.

Furthermore, we check if the user has a CV linked to their profile. If not, we will create a new CV entry for them.

Finally, we establish the user's session and redirect them to their profile page on our frontend application.

In Figure 11, we demonstrate the process of LinkedIn login and saving data to the database

```

const linkedinAuthUrl = 'https://www.linkedin.com/oauth/v2/authorization';
const linkedinTokenUrl = 'https://www.linkedin.com/oauth/v2/accessToken';
const linkedinProfileUrl = 'https://api.linkedin.com/v2/me';
const linkedinUserInfoUrl = 'https://api.linkedin.com/v2/userinfo';
const linkedinAuthService = {
  initiateAuth: (_, res) => {
    const params = new URLSearchParams({
      response_type: 'code',
      client_id: config.LINKEDIN_CLIENT_ID,
      redirect_uri: config.LINKEDIN_CALLBACK_URL,
      state: config.SESSION_STATE,
      scope: 'openid profile email r_basicprofile',
    });
    res.redirect(`${linkedinAuthUrl}?${params.toString()}`);
  },
  handleAuthCallback: async (req, res) => {
    const { code } = req.query;
    if (!code) return res.status(400).send('Authorization code missing');
    try {
      const tokenResponse = await axios.post(linkedinTokenUrl, null, {
        params: {
          grant_type: 'authorization_code',
          code,
          client_id: config.LINKEDIN_CLIENT_ID,
          client_secret: config.LINKEDIN_CLIENT_SECRET,
          redirect_uri: config.LINKEDIN_CALLBACK_URL,
        },
        headers: { 'Content-Type': 'application/json' },
      });
      const { access_token } = tokenResponse.data;
      const profileResponse = await axios.get(linkedinProfileUrl, {
        headers: { Authorization: `Bearer ${access_token}` },
      });
      const userInfoResponse = await axios.get(linkedinUserInfoUrl, {
        headers: { Authorization: `Bearer ${access_token}` },
      });
      const userProfile = {
        fullName: userInfoResponse.data.name,
        headline: profileResponse.data.headline.localized.en_US,
        image: userInfoResponse.data.picture,
        email: userInfoResponse.data.email ? userInfoResponse.data.email : null,
        linkedinURL: `https://www.linkedin.com/in/${profileResponse.data.vanityName}`,
      };
      let existingUser = await User.findOne({ email: userProfile.email });
      if (existingUser) {
        existingUser.set(userProfile);
        await existingUser.save();
      } else {
        existingUser = new User(userProfile);
        await existingUser.save();
      }
      let cvExists = await CV.exists({ user: existingUser._id });
      if (!cvExists) {
        const newCV = new CV({
          user: existingUser._id,
          headline: userProfile.headline,
        });
        await newCV.save();
      }
      req.session.user = existingUser;
      if (res && userProfile) {
        const redirectUrl = `${process.env.FRONTEND_URL}/profile/${existingUser._id}`;
        res.redirect(redirectUrl);
        return;
      }
    } catch (error) {
      console.error('Error during LinkedIn OAuth:', error);
      res.status(500).send('Authentication failed');
    }
  },
};

```

FIGURE 11. OAuth login and saving data to the database process

Now for storing user profile data, we need to create a database model. In our application, we use Mongoose for this purpose.

In Figure 12, we created a user collection that saves user data from LinkedIn login to our MongoDB database.

```
1 import mongoose from 'mongoose';
2
3 const UserSchema = new mongoose.Schema(
4   {
5     fullName: String,
6     image: String,
7     email: String,
8     linkedinURL: String,
9   }
10 );
11
12 export const User = mongoose.model('User', UserSchema);
```

FIGURE 12. Mongoose User Schema for storing user data.

After successful authentication, the backend redirects users to their profile page on our frontend application. On this page, users can view and manage their profile information, such as their full name, headline, profile picture, and LinkedIn URL.

In our frontend, we use React to create a smooth user experience. Our login page component, `LoginPage.js`, is where users start the authentication process by clicking a button that redirects to LinkedIn's OAuth 2.0 authorization endpoint. This step is essential for securely verifying user identities and obtaining their consent to access LinkedIn profile data.

In Figure 13, we created a simple login page that interacts with our backend app for the login process.

```

1 export const LoginPage = () => {
2   const handleLogin = () => {
3     window.location.href = 'http://localhost:8080/auth/linkedin';
4   };
5
6   return (
7     <div className='login-page'>
8       <h1>Login</h1>
9       <button onClick={handleLogin}>Login with LinkedIn</button>
10    </div>
11  );
12 };

```

FIGURE 13, Login page component in frontend.

After a successful LinkedIn login, the user's profile data is accessed and sent to the backend for validation and storage in our database. The backend checks the information, ensures its validity and uniqueness, and saves new user profiles or updates existing ones. This integration enables efficient user authentication and profile management using LinkedIn's system.

In Figure 14, after successful login user data is saved to the database.

```

_id: ObjectId('66113441215640868000000000000000')
fullName: 'John Doe'
image: "https://media.lidn.com/dms/image/66113441215640868000000000000000"
email: "john.doe@example.com"
linkedinURL: "https://www.linkedin.com/in/johndoe"
createdAt: 2024-05-12T21:56:40.868+00:00
updatedAt: 2024-05-12T21:56:40.868+00:00

```

```

_id: ObjectId('66113441215640868000000000000000')
fullName: 'John Doe'
image: "https://media.lidn.com/dms/image/66113441215640868000000000000000"
email: "john.doe@example.com"
linkedinURL: "https://www.linkedin.com/in/johndoe"
createdAt: 2024-05-13T10:29:20.713+00:00
updatedAt: 2024-05-13T10:29:20.713+00:00

```

FIGURE 14. Database user data collection.

6 Challenges and Future Improvements

During development, we encountered numerous challenges, especially in the aspects of authentication and authorization. The biggest obstacle was LinkedIn's API, which only allows access to a limited amount of personal data contained on user profiles. This was a difficult limitation for us because we could no longer collect the extensive profile information that we used to create full resumes.

To address this problem, we created an archival database using user-defined data and saved it to the database. Additionally, to improve user engagement and guarantee the completion of their resumes, we allowed users to access an editable form in the application. Users can manually enter and edit their profile data in a form that is automatically linked with the information pulled from LinkedIn. By utilizing such a hybrid approach, the application guarantees that users can cover every ground, offering a more precise and individualized resume on the whole.

In the near future, we will be working to add new features and enhance the functionality and user experience of our app. Among the most adventurous is the addition of an AI algorithm that can parse LinkedIn profile data - along with whatever info the user inputs manually. This AI-powered feature is created to generate personalized cover letters tailored to the user's specific experiences and job applications. By implementing AI features, we can offer a strong utility to the users - to autonomously write cover letters on their own, amping up their job-seeking process.

Additionally, to cater to different preferences, we will be introducing more CV templates and unique design options to our template library. This will allow users to have greater flexibility and creativity in creating their resumes and provide them with multiple formats to best represent their professional brand and meet market requirements.

7 DISCUSSIONS

The development of this website has been a breakthrough in the resume creation and management. To simplify the process, we utilized the OAuth 2.0 authentication system along with API from LinkedIn to populate a template with their user's professional profile data, resulting in an accurate and updated resume.

The application merges the information imported from LinkedIn with the data collected directly from users to ensure their CV is as complete as it can be, and relevant to employers. This hybrid model enables users to build resumes based on their overall experiences and qualifications.

It has a user-friendly resume creation and customization user interface implemented using React. Users can edit and personalize their CVs to highlight the most relevant experiences and skills for the specific role someone is applying for.

For job applications, the application also provided the functionality to export personalized CVs in PDF format. This is a huge improvement in the overall functionality and versatility of the application.

Beyond its utilitarian purpose, this project showcases how OAuth 2.0 authentication and API integration can enhance user experience. It also addresses concerns related to data privacy and security, ensuring that user information is managed securely and in compliance with industry standards.

In conclusion, the development of this web application marks a major achievement in the process of digitizing professional documents. By utilizing LinkedIn's authentication system and data retrieval features, we have developed a thorough and proactive solution for creating and managing resumes. This has become increasingly vital in empowering individuals to effectively control and display their professional narratives as the job market undergoes ongoing changes.

REFERENCES

Bhatt, T. (2024). 8 Most Popular Front End Frameworks to Use in 2024. Retrieved from <https://londonappdevelopment.co.uk/blog/best-front-end-frameworks> (Read on 06.06.24).

BYTEBYTEGO. (2023, August 12). EP72: OAuth 2.0 Explained With Simple Terms. Retrieved from <https://blog.bytebytego.com/p/ep72-oauth-20-explained-with-simple> (Read on 06.06.24).

Herbert, D. (2023). What is React.js? Uses, Examples, & More. Retrieved from <https://blog.hubspot.com/website/react-js> (Read on 06.06.24).

LinkedIn. (2023). LinkedIn Developer Solutions. Retrieved from <https://developer.linkedin.com> (Read on 06.04.24).

LinkedIn API Documentation (2023). Authentication with OAuth 2.0 Overview – LinkedIn. Retrieved from <https://learn.microsoft.com/en-us/linkedin/shared/authentication/authentication>

Kugan, S. (2022). Mongo DB. Retrieve from <https://medium.com/@kukan-sakitha/mongo-db-195d6c554d9c> (Read on 09.06.24).

Metwalli, S. A. (2023). What Is NPM? Retrieved from <https://builtin.com/software-engineering-perspectives/npm> (Read on 06.06.24).

MongoDB. (n.d.). JSON Databases Explained. Retrieved from <https://www.mongodb.com/resources/basics/databases/json-database> (Read on 06.06.24).

MongoDB (n.d.), official website. Retrieved from <https://www.mongodb.com>

Node.js. (n.d.). Node.js Documentation. Retrieved from <https://nodejs.org/en/docs>

React (n.d.), official website. Retrieved from <https://react.dev>

Shivhare, K., Shakya, S., & Bhadouria, A. S. (2024). ResumeCraft: A Machine Learning-powered Web Platform for Resume Building. Retrieve from https://www.researchgate.net/profile/Aashi-Bhadouria/publication/380744158_ResumeCraft_A_Machine_Learning-powered_Web_Platform_for_Resume_Building/links/664cc498bc86444c72f5b09a/ResumeCraft-A-Machine-Learning-powered-Web-Platform-for-Resume-Building.pdf (Read on 06.06.24).

APPENDICES

Appendix 1. Source Code

Frontend Link to GitHub: <https://github.com/MdShayemurRahman/resumeGen-frontend>

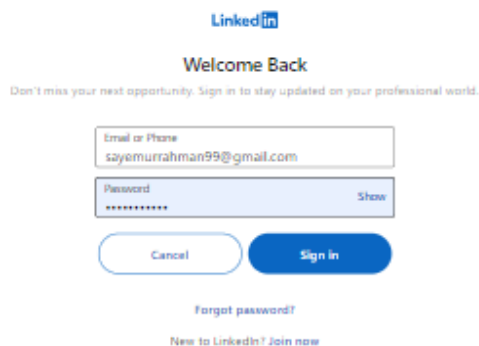
Backend Link to GitHub: <https://github.com/MdShayemurRahman/resumeGen-backend>

Appendix 2. Final View of the Application

Login Page



Login with LinkedIn




Home Page

User Profile LOGOUT

Md Shayemur Rahman

shayemurrahman99@gmail.com
+358449858804
http://www.linkedin.com/in/mdshayemurrahman



[EDIT PROFILE](#)

[SAVE AS PDF](#)

About Me

Student at Tampere University of Applied Science | Full Stack Developer Trainee at Integrify.

Skills

- Python
- React
- Node.js

Experience

Full-Stack Developer - Telus Inc.
worked for 5 years

Projects

CV maker from LinkedIn
<https://example.com>

Education

BSc in Software Engineering
Tampere University
2020 - 2024
Grade: 3.47

Edit Profile

User Profile LOGOUT

Edit Profile

Personal Information

About Me
Student at Tampere University of Applied Science | Full Stack Developer Trainee at Integrify.
1000 words

Phone
+358449858804

Skills

Skill 1: Python | Skill 2: React | Skill 3: Node.js

[ADD SKILL](#)

Education

Institution: Tampere University | Degree: BSc in Software Engineering

Start Year: 2020 | End Year: 2024

Grade: 3.47

[REMOVE](#)

[ADD EDUCATION](#)