**Github Link**

<https://github.com/2216064/Website2/tree/main>

**Cloud Deployment**

For development and scaling I initially used ReplIt to make all necessary files and to start working on the static and templates folders. After that was successfully done I started trying to integrate the backend but I quickly ran into quite a few problems with ReplIt’s IDE and how it treats Flask redirects so I used Python on my local machine to finish off the development. I then used Azure to test it further and make sure everything was working (Microsoft Learn, n.d.).

**Project Overview**

Project Goals and Objectives:

The aim of this project is to develop a job portal application where students and employers can interact. Students can register, create a profile, and search for jobs, while employers can post job listings and manage their profiles. The system supports secure authentication, user management, and job search functionalities.

Scope of the project:

* Development of the front-end user interface.
* Implementation of the back-end server-side logic.
* Users: The application supports two types of users: Students and Employers.
* User registration and login with secure password hashing.
* Password reset functionality via email.
* Profile management for students and employers.
* Job posting and job search functionalities.

**Installation Instructions**

Development Environment setup

1. Download folder “Backend”
2. If you do not already have Python installed, download it from [www.python.org](http://www.python.org) and use the setup wizard making sure to install from path
3. Go to your computer terminal and make sure pip is installed (if unsure type: ”pip --version”) then make sure it is updated using: “python -m pip install --upgrade pip”
4. Once pip is installed open up python shell (not idle) and make sure you have these libraries downloaded via pip: logging, random, string, datetime, flask, flask\_sqlalchemy, flask\_mail, werkzeug.security and werkzeug.utils
5. Setup database using “ python -c 'from app import db; db.create\_all()' “
6. Run app.py in the Backend
7. Navigate to http://127.0.0.1:5000/index

**Application Architecture**

The architecture of the application includes:

* **Frontend:** Developed using basic web technologies like HTML, CSS, and JavaScript
* **Middleware:** Flask and JSONify
* **Backend:** Built with Flask a WSGI web application framework
* **Database:** Managed using SQLAlchemy and SQLite

### **Data Flow**

1. User sends a request from the frontend.
2. The request is processed by Flask and JSONify, which communicates with the database via SQLAlchemy.
3. Flask/JSONify returns the processed data back to the frontend for display.

**Legal and Ethical Considerations**

1. **Data Privacy**Compliance with Data Protection Laws:  
   The application must strictly adhere to data protection laws such as the General Data Protection Regulation (GDPR) in the European Union, the California Consumer Privacy Act (CCPA) in the United States, and other relevant local regulations (GDPR Overview, n.d.; California Consumer Privacy Act, n.d.). This involves implementing appropriate measures to protect user data, providing clear and transparent privacy notices, and obtaining user consent where required.

User Consent: Ensure that the application obtains explicit consent from users before collecting, processing, or sharing their personal information. This includes providing clear options for users to opt in or out of data processing activities.

Data Minimization: Collect only the data necessary for the specific purposes of the application. Avoid gathering excessive or irrelevant personal information.

Right to Access, Modify, and Delete Data: Users should have the ability to access, correct, or delete their personal data from the application. Implement user interfaces and backend functionality to facilitate these requests.

Data Breach Notifications: In the event of a data breach, it is crucial to have procedures in place for timely notification to both the affected users and relevant regulatory authorities, as required by law.

1. **Intellectual Property**Licensed Libraries and Software:

Ensure that all third-party libraries, frameworks, and software components used in the development of the application are properly licensed.

Open Source Compliance: If using open-source software, comply with the terms of the licenses, which may include attribution, sharing modifications, or not using the software for commercial purposes without a proper license (Open Source Initiative, n.d.).  
Copyright for Media:

Any images, videos, fonts, and other media assets used in the application must be sourced from legitimate sources with the appropriate licenses. Avoid using media without proper authorization to prevent copyright infringement.

User-Generated Content: Implement terms of service that require users to ensure that their content does not violate any intellectual property rights as well as any other laws to do with explicit imagery. Additionally, provide mechanisms to remove infringing content if reported.

1. **Security**Encrypted Connections:  
   Use HTTPS (SSL/TLS) to encrypt all data transmitted between the application and its users, ensuring that sensitive information such as login credentials, personal data, and payment details are protected from interception and eavesdropping (Let's Encrypt, n.d.).

Secure Password Storage:  
Implement strong password hashing algorithms using werkzeug.security

Regular Vulnerability Assessments:  
User Education: Educate users about best security practices, such as recognizing phishing attempts and using strong, unique passwords, to minimize the risk of security breaches.

**Future plans for Scaling**Database scaling: potentially migrating the database to MySQL to handle larger datasets  
Load balancing: Using Cloudflare to handle traffic and unwanted DDoS attempts  
More security: to prevent SQL injection, XSS attacks, and malicious file injection with tighter parameters and an additional check of media being uploaded before it is entered into the database. Furthermore, a way of handling the Employer Website entry to prevent unwanted code execution

**Timeline  
Semester 1:**Developed initial frontend to website  
Created index.html, login.html, register.html, platform.html  
Created Platform.js, Script.js and Captcha.js  
Created style.css, platform.css, loginstyle.css

**Semester 2:**Created app.py  
Created Backend directory  
Created student\_dashboard, email\_sent, employer\_dashboard, employer\_edit, forgot\_password, password\_reset, student\_edit  
Created employerdashboard.css, dashboard.css  
Added to index.html, login.html, register.html, platform.html, Platform.js, Script.js and Captcha.js

**Milestones:**11/12/2023: Frontend completion date  
02/08/2024: Backend completion date  
03/08/2024: Website ready for deployment

**Reflection**Throughout the project I met innumerable obstacles. From in the first semester, images weren’t being displayed correctly on the page due to image dimensions to CSS and JavaScript overwriting each other to obscure code errors that took me a week to solve. In the second semester, the obstacles seemed to stack on each other as I had no previous experience of creating a backend to a website. I was slowly able to work through it and solve the issues I encountered. Particularly the issue I had with ReplIt redirects and how it wasn’t running Flask code correctly which disappeared when I migrated platforms. However now I feel fully competent at making a backend on my local machine and I found having a rigorous timetable enabled me to be absorbed by the code and then I was able to understand all of it at once.

**References**

California Consumer Privacy Act (CCPA). (n.d.). *California Department of Justice*. Retrieved August 3, 2024, from<https://oag.ca.gov/privacy/ccpa>

GDPR Overview. (n.d.). *GDPR.eu*. Retrieved August 3, 2024, from<https://gdpr.eu/what-is-gdpr/>

Let's Encrypt. (n.d.). *HTTPS Implementation and Security*. Retrieved August 3, 2024, from<https://letsencrypt.org/getting-started/>

Microsoft Learn. (n.d.). *Azure Testing for Python Apps*. Retrieved August 3, 2024, from<https://learn.microsoft.com/en-us/azure/app-service/quickstart-python>

Open Source Initiative. (n.d.). *Open Source Licensing Compliance*. Retrieved August 3, 2024, from<https://opensource.org/licenses>