

**NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY**

School of Electrical Engineering and Computer Sciences

*Yours Edu Connect*

*(Courses Management System)*

**WEB ENGINEERING**

**Project Report**

**Group Members**

**Sohaib Ahmed (373937)**

**Muhammad Ehtisham Raza (374965)**

**INTRODUCTION**

Welcome to YoursEduConnect, where education meets simplicity and global opportunities. Navigate a vast array of 40,000 courses worldwide effortlessly using our user-friendly interface and dynamic filters. Discover detailed insights into universities, connect with a diverse global learning community, and propel your educational journey with confidence.

Our commitment is to demystify the complexities of educational exploration. With 2 lac courses in our database and 40,000 courses currently available, YoursEduConnect empowers learners to tailor their educational path. Stay tuned as we add the remaining 1 lac 60 thousand courses, providing you with an even broader spectrum of choices.

Behind the scenes, our admin page is your tool for efficient data manipulation, ensuring the information you access is accurate and relevant. YoursEduConnect is not just a platform; it's your gateway to a world of educational possibilities, combining innovation, user empowerment, and a commitment to shaping the future of global education. Join us on this exciting educational journey!

**USE OF WEB AND OTHER TECHNOLOGIES**

* HTML
* CSS
* React JS
* Flask (python)
* Bootstrap
* MySQL (database)

**Problem Statement**

YourEduConnect aims to alleviate the challenges faced by students, administrators, and educational institutions in navigating and managing course-related information effectively. The platform addresses the following real-life problems:

**1. Navigating Course Information Maze:**

* **Problem:** Prospective students often struggle to find detailed and consolidated information about courses offered by various educational institutions. This leads to time-consuming searches across multiple websites and platforms.
* **Solution:** YourEduConnect streamlines course information, providing a centralized platform where students can easily access comprehensive details such as fees, descriptions, disciplines, and durations.

**2. Administrative Inefficiencies in Course Management:**

* **Problem:** Administrators and educational institutions face challenges in managing and updating course details efficiently. Traditional methods may involve manual data handling, leading to errors and delays in communication.
* **Solution:** The platform offers an intuitive admin interface for seamless course management, enabling administrators to update information promptly and communicate changes effectively.

**3. User-Friendly Access to Educational Resources:**

* **Problem:** Users, including students and administrators, often encounter cumbersome or confusing user authentication processes on educational platforms. This may hinder their access to valuable resources.
* **Solution:** YourEduConnect simplifies the user experience with an efficient and user-friendly sign-up/sign-in process, ensuring easy access to educational resources.

**4. Enhancing Online Learning Experience:**

* **Problem:** Traditional websites may disrupt the user experience, especially when reloading is required, causing inconvenience during online learning and research.
* **Solution:** By leveraging React with Virtual DOM, YourEduConnect minimizes page reloads, creating a seamless and interactive online learning experience, particularly on pages with detailed course information.

**5. Effective Collaboration in Educational Settings:**

* **Problem:** Collaborative efforts within educational settings face challenges due to diverse stakeholder involvement and conflicting requirements. This may result in miscommunication and inefficiencies.
* **Solution:** YourEduConnect emphasizes effective communication, streamlined task assignment, and conflict resolution within groups, fostering a collaborative environment for educational stakeholders.

In summary, YourEduConnect addresses real-life challenges by providing a user-friendly, centralized platform for accessing course information, streamlining administrative processes, enhancing user authentication, improving online learning experiences, and promoting effective collaboration in educational settings.

**Choose of Stack:**

**Benefits of Using Flask:**

* **Lightweight and Flexible:** Flask is a lightweight and flexible web framework that allows developers to build web applications quickly. It provides just what is needed to get started without imposing too much structure.
* **Easy to Learn:** Flask follows a simple and intuitive design, making it easy for developers, especially beginners, to learn and understand. It has a minimalistic approach to building web applications.
* **Powerful Routing:** Flask provides a powerful routing system, allowing developers to define routes and their corresponding actions with ease. This makes it convenient to structure the application and handle different endpoints.
* **Widely Used in Industry:** Flask is widely adopted in the industry and has a large community of developers. This means there are plenty of resources, tutorials, and extensions available, making it easier to find solutions to common problems.

**Benefits of Using React:**

* **Declarative Syntax:** React uses a declarative syntax, making it easier to understand and visualize the structure of UI components. Developers can describe how the UI should look based on the application's state.
* **Component-Based Architecture:** React is built around a component-based architecture, promoting reusability and maintainability. Each component encapsulates its own logic and UI, which can be composed to create complex interfaces.
* **Virtual DOM and Efficient Updates:** React's Virtual DOM enables efficient updates by minimizing direct manipulations to the actual DOM. This leads to improved performance and a better user experience, especially in scenarios where frequent updates are required.
* **Unidirectional Data Flow:** React follows a unidirectional data flow, making it easier to manage the state of an application. Data flows in a single direction, from parent to child components, simplifying the understanding of how changes propagate through the application.

**Use of React in Course Page:**

* **Preventing Page Reloads:** In the course page, react is used to prevent unnecessary page reloads. Traditional web applications might reload the entire page when navigating between different sections or updating content. React, with its virtual DOM, allows for partial updates without refreshing the entire page.
* **Enhancing User Experience:** By leveraging React's ability to handle UI updates efficiently, the course page becomes more responsive. Users can interact with different sections of the page, such as updating course information, without experiencing the delays associated with full page reloads.
* **React Virtual DOM in Action:** The use of React Virtual DOM in the course page ensures that only the necessary parts of the DOM are updated when the state changes. This results in a smoother user experience, reduced server load, and improved overall performance.
* **Dynamic Content Loading:** React enables dynamic loading of content, allowing users to interact with course details, update information, and navigate seamlessly without interruptions. This is achieved by handling changes in the application state through React components.
* **Maintainability and Scalability:** The component-based structure of React promotes maintainability and scalability. Adding new features or making changes to existing ones in the course page becomes more manageable, contributing to a more sustainable and extensible codebase.

In short, the combination of Flask and React offers a balance of simplicity, flexibility, and efficiency in web development. Flask handles the backend logic and routing, while React enhances the frontend with its declarative syntax, component architecture, and virtual DOM, resulting in a modern and responsive user interface.

**World Wide Web Concepts:**

**1. HTML (HyperText Markup Language):**

**Purpose:** HTML provides the structure and content of web pages.

**Usage in Template:**

* Defines the document structure with `<html>`, `<head>`, and `<body>`.
* Includes metadata like character set and viewport settings.
* Uses `<meta>`, `<link>`, `<script>`, and `<title>` elements.

**2. CSS (Cascading Style Sheets):**

**Purpose:** CSS styles HTML elements for a visually appealing layout.

**Usage in Template:**

* External `style.css` for custom styles.
* Bootstrap CSS for predefined styles.
* Inline styles with the `style` attribute.

**3. JavaScript:**

**Purpose:** JavaScript adds interactivity and dynamic behavior.

**Usage in Template:**

* External scripts linked with `<script>` tags.
* Inline scripts for immediate execution.
* Uses the `defer` attribute for deferred script execution.

**4. Bootstrap Framework:**

**Purpose:** Bootstrap simplifies UI design with pre-styled components.

**Usage in Template:**

* Utilizes Bootstrap classes for navigation, cards, buttons, forms, and grid layout.
* Ensures responsive design with the Bootstrap grid system.

5. **Jinja Templating:**

**Purpose:** Jinja dynamically generates HTML content.

**Usage in Template:**

* Embeds Python-like code within `{{ }}` and `{% %}` tags.
* Supports conditional statements and loops.

6. **External Libraries:**

**Purpose:** Enhances functionality with external resources.

**Usage in Template:**

* Leaflet for interactive maps.
* Font Awesome for scalable icons.
* Bootstrap Icons for Bootstrap components.

**7. Client-Side Rendering (JavaScript):**

**Purpose:** Enables dynamic updates without page reload.

**Usage in Template:**

* Handles form submissions with JavaScript.
* Implements dynamic content updates.

**8. Responsive Design:**

**Purpose:** Ensures adaptability to different screen sizes.

**Usage in Template:**

* Leverages Bootstrap's responsive grid system.
* Applies media queries for device-specific styles.

**9. Web Forms:**

**Purpose:** Collects user input through various form elements.

**Usage in Template:**

* Implements forms for user authentication and course manipulation.
* Utilizes `<form>`, `<input>`, `<textarea>`, and `<button>` elements.

**10. HTTP Methods:**

**Purpose:** Specifies the type of action for data submission.

**Usage in Template:**

* `method="POST"` for submitting data.
* `method="GET"` for requesting data.

These concepts collectively form the backbone of a modern web application, enabling a seamless and interactive user experience. They encompass everything from structuring content and styling to dynamic behavior and responsiveness.

**Python File:**

**app.py:**

**Purpose:** This is the main Flask application file.

**Key Features:**

* Handles routing and HTTP request/response handling.
* Integrates with templates and static files.
* Uses Flask's decorators (`@app.route()`) to define routes.
* Includes logic for rendering HTML templates and processing form data.
* Likely contains functions/methods corresponding to different routes.

**Util.py:**

**Purpose:** Encapsulating utility functions and helper classes.

**Key Features:**

* Manages interactions with the database, including query execution and data integrity.
* Contains miscellaneous functions aiding various tasks throughout the application.
* Implements logic for processing and transforming data as required by the application.
* Defines custom classes or models tailored to the application's specific needs.

These Python files collectively contribute to the backend logic of the web application. Flask facilitates the integration of Python with the web, and the other files organize and manage different aspects of the application, such as routing, database interaction, configuration, and template rendering.

**Conflicting Requirements**

**Identifying and Resolving Conflicting Requirements:**

**1. User Authentication and Authorization:**

* **Conflict:** Balancing the need for a secure authentication system while ensuring a seamless user experience.
* **Resolution:** Implement a robust authentication system (e.g., Flask-Login) and carefully design user flows to balance security and usability.

**2. Performance Optimization:**

* **Conflict:** Balancing rich user interfaces with optimized page load times.
* **Resolution:** Implement UIs like bootstrap.

**React Virtual DOM:**

**1. Page Loading:**

* **Conflict:** Our courses page reloads every time a change is done.
* **Resolution:** To resolve this, we have used react virtual DOM which stop this issue automatically with its build in feature and converted it into HTML.

**2. Efficient DOM Manipulation:**

* **Conflict:** Traditional direct DOM manipulation can be slow and lead to a poor user experience.
* **Resolution:** React Virtual DOM allows efficient updates by creating a lightweight in-memory representation of the DOM. Changes are first applied to this virtual representation and then efficiently synced with the actual DOM, minimizing unnecessary reflows.

**3. Automatic UI Updates:**

* **Conflict:** Manually tracking and updating changes in the UI is error-prone and resource-intensive.
* **Resolution:** React's Virtual DOM automatically identifies changes in the component state and efficiently updates the UI. Developers work with a declarative syntax, specifying how the UI should look based on the state, and react takes care of the underlying updates.

**Resolving Conflicts in HTML and Python Files:**

**1. HTML Files (templates):**

* **Conflict:** Balancing dynamic content with maintainability in HTML templates.
* **Resolution:** Utilize Jinja templating in Flask to embed dynamic content. Separate concerns by organizing templates logically and using template inheritance.

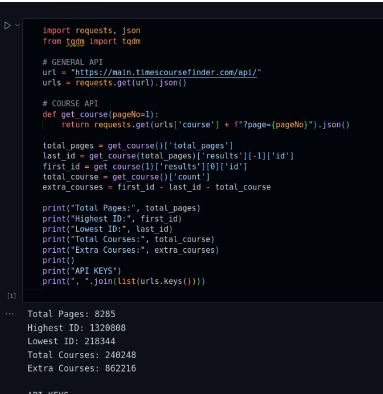
By identifying and resolving these conflicts during the development process, developers can create a robust, efficient, and maintainable web application.

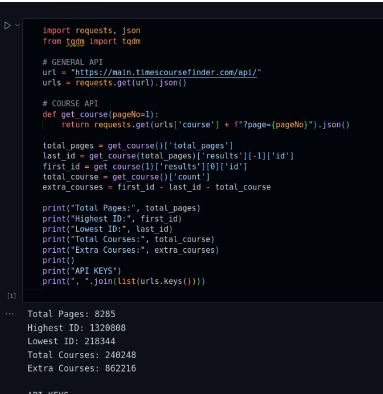
**STEPS/MODULES**

1. **Data scraping:**

We did data scraping with the help of an API and request library in python. We had an API dealing with json files and by using request library, we were able to scrap the data in json format.

We used this request library with an API because it was very efficient and one of the easiest ways of scraping. Some screenshots of this work are attached below:







1. **Data processing and cleaning:**

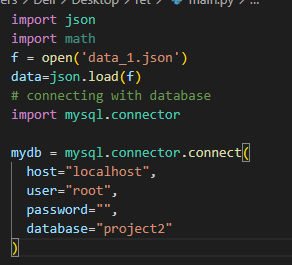
After data scraping, we cleaned up the data and processed all the courses having inappropriate or odd data format. We put all the data in a single json file so that it could be handled easily and could be easily dumped into mysql database. Ease of management is shown in the screenshots below the next section.

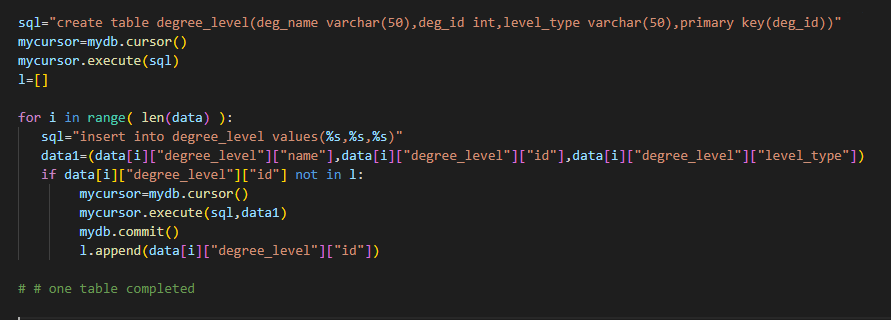
1. **Database design:**

From a single json file made above, we dumped the data into mysql using python mysqlconnector. First, we made tables in python and then using mysqlconnector, we inserted data in tables by iterating through jsonfile. This concept required appropriate json file handling.

This was done in python because we were going to use flask (python) for backend, so python was the preferred option for this task also. Moreover, this was also more efficient and easy, and json file handling was much easier through python.

Some screenshots of this work are as below:





**Normalization:**

The YoursEduConnect database, meticulously crafted with 29 tables, stands as the cornerstone of our dynamic educational platform. This report provides an insight into the creation and structure of the database, highlighting key principles and tables that underpin the seamless functionality of YoursEduConnect.

**Normalization Principles:**

The database follows normalization principles to ensure optimal efficiency and data integrity. By minimizing redundancy and organizing data logically, our design promotes streamlined storage and retrieval operations. This commitment to normalization enhances the overall reliability and performance of the database.

**Key Tables:**

1. **adm\_schedule:** Tracks admission schedule details and foreign key referencing the 'courses' table.

2. **admin\_login:** Manages administrator login credentials.

3. **campus:** Stores information about educational campuses and foreign key references to 'institute' and 'location' tables.

4. **city:** Holds data related to cities.

5. **country:** Contains details about countries.

6. **course\_duration:** Defines the duration of courses.

7. **course\_log:** Logs course-related information.

8. **course\_title:** Manages course titles.

9. **courses:** Central table for course information and utilizes various foreign keys for relationships with other tables.

10. **currency:** Manages currency types.

11. **degree\_level:** Defines different degree levels.

12. **description\_log:** Logs descriptions.

13. **discipline:** Stores discipline-related details.

14. **fee:** Manages course fees and foreign key references to 'courses' and 'currency' tables.

15. **institute:** Contains information about educational institutes.

16. **language:** Defines different languages.

17. **location:** Holds information about locations and foreign key references to 'city', 'state', and 'country' tables.

18. **ranking:** Stores ranking information and foreign key references to the 'institute' table.

19. **source\_links:** Manages source links related to courses and foreign key references to the 'courses' table.

20. **specialization:** Stores details about course specializations and foreign key references to the 'discipline' table.

The creation of the YoursEduConnect database is not merely a technical process; it represents a strategic initiative to build a robust foundation for our dynamic educational platform. The thoughtful application of normalization principles and the establishment of well-structured relationships between tables showcase our commitment to providing a seamless, responsive, and user-centric educational experience.

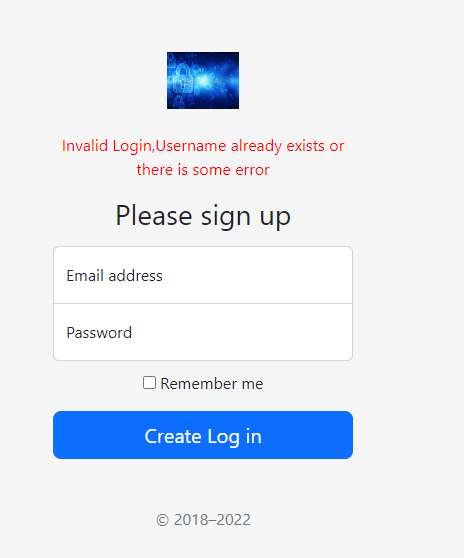
1. **Website design:**

* **FRONT END DESIGN:**

For front-end design, we have used HTML, CSS and React. HTML and CSS were easy to use, and they provide responsive front end, which is why these were used.

At the time of design/production, React is used for user side of website to render filter.html as react helps integrate the code and render dynamic pages easily with virtual DOM concept. Then, **React was converted to build form and now we have pure HTML, CSS files for frontend.**

Beautiful animations were created using key frames in CSS. Different error messages are also handled in frontend using **jinja templates** of flask making html files dynamic. One of the error messages is as below:



Bootstrap is also used for responsive front-end pages as it gives us required front end pages with efficient code which is easy to understand. The combination of Html, CSS, React, Flask and Bootstrap helped us create responsive and decent front-end pages as shown below:

**Home Page:**

A person smiling with headphones on her neck

Description automatically generated

A screenshot of a computer

Description automatically generated

**About Page:**

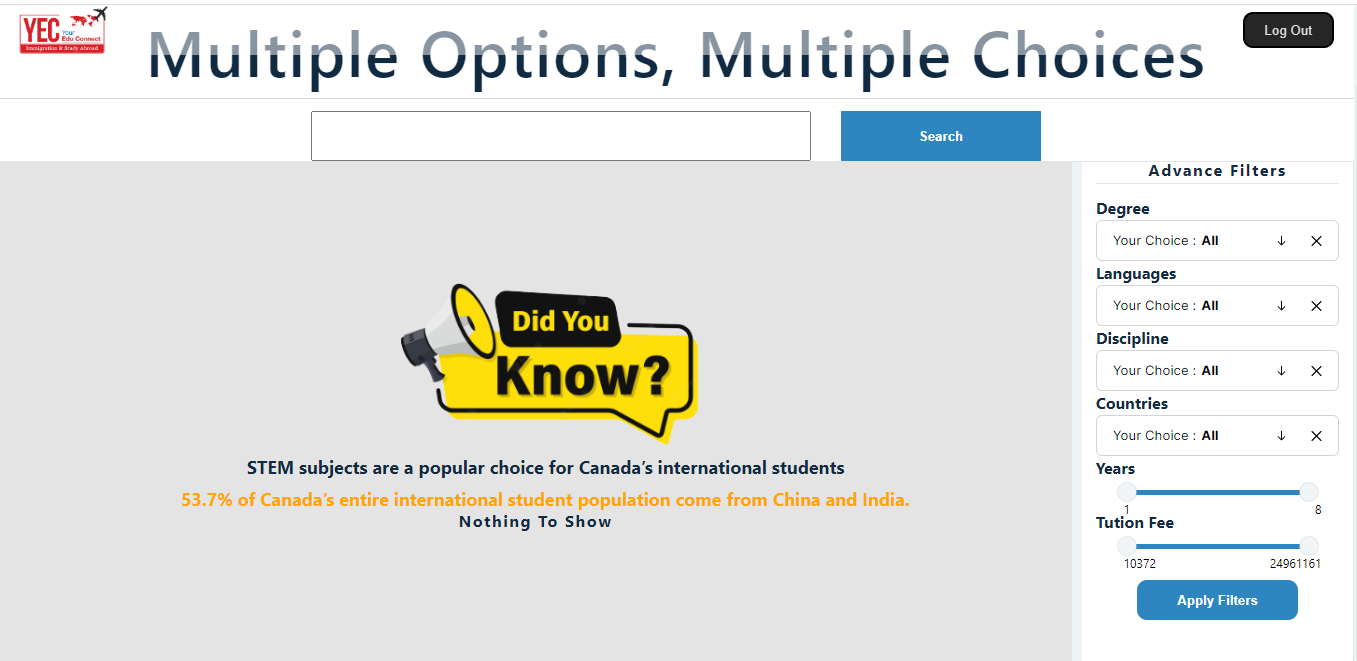
A screenshot of a website

Description automatically generated

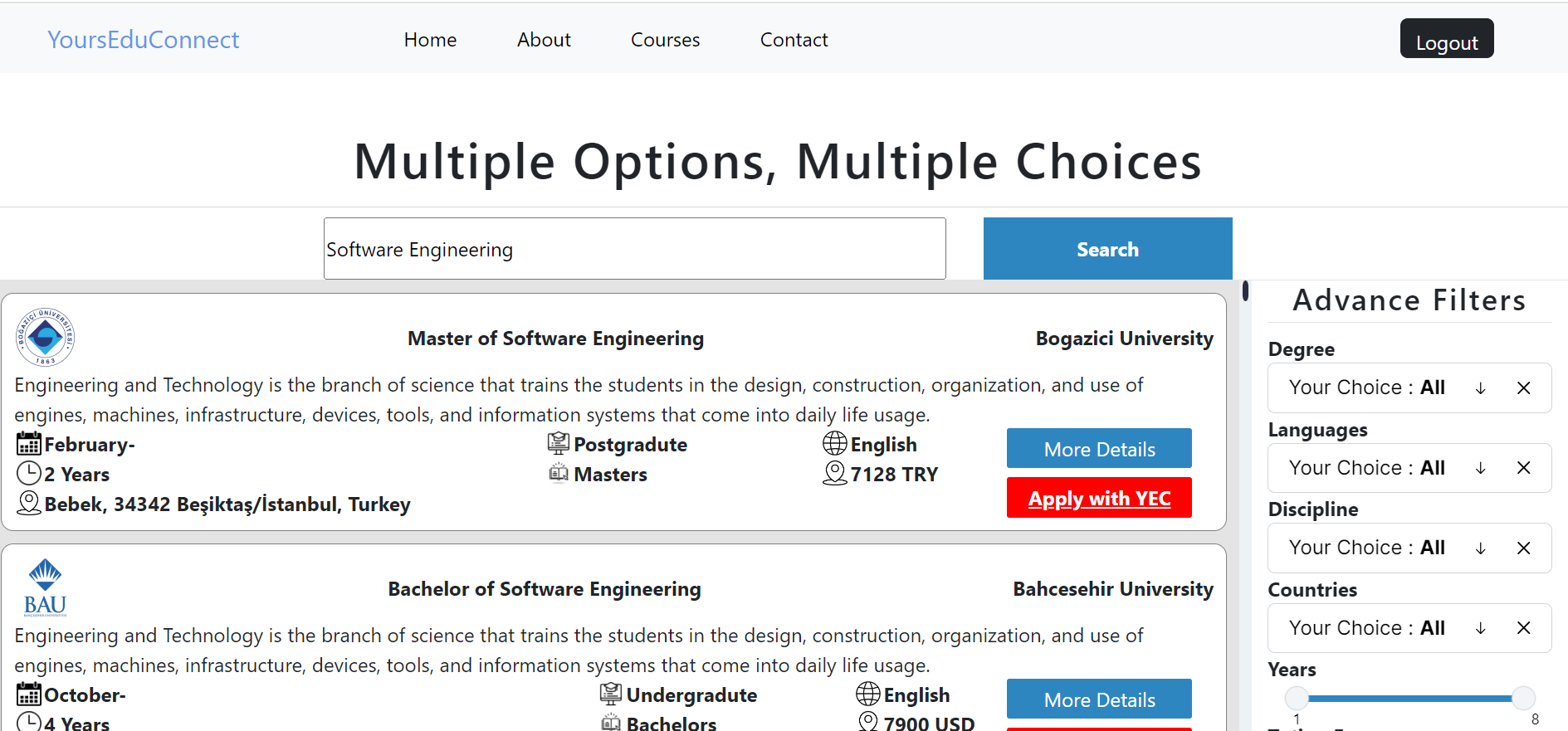
A group of people looking at a computer

Description automatically generated

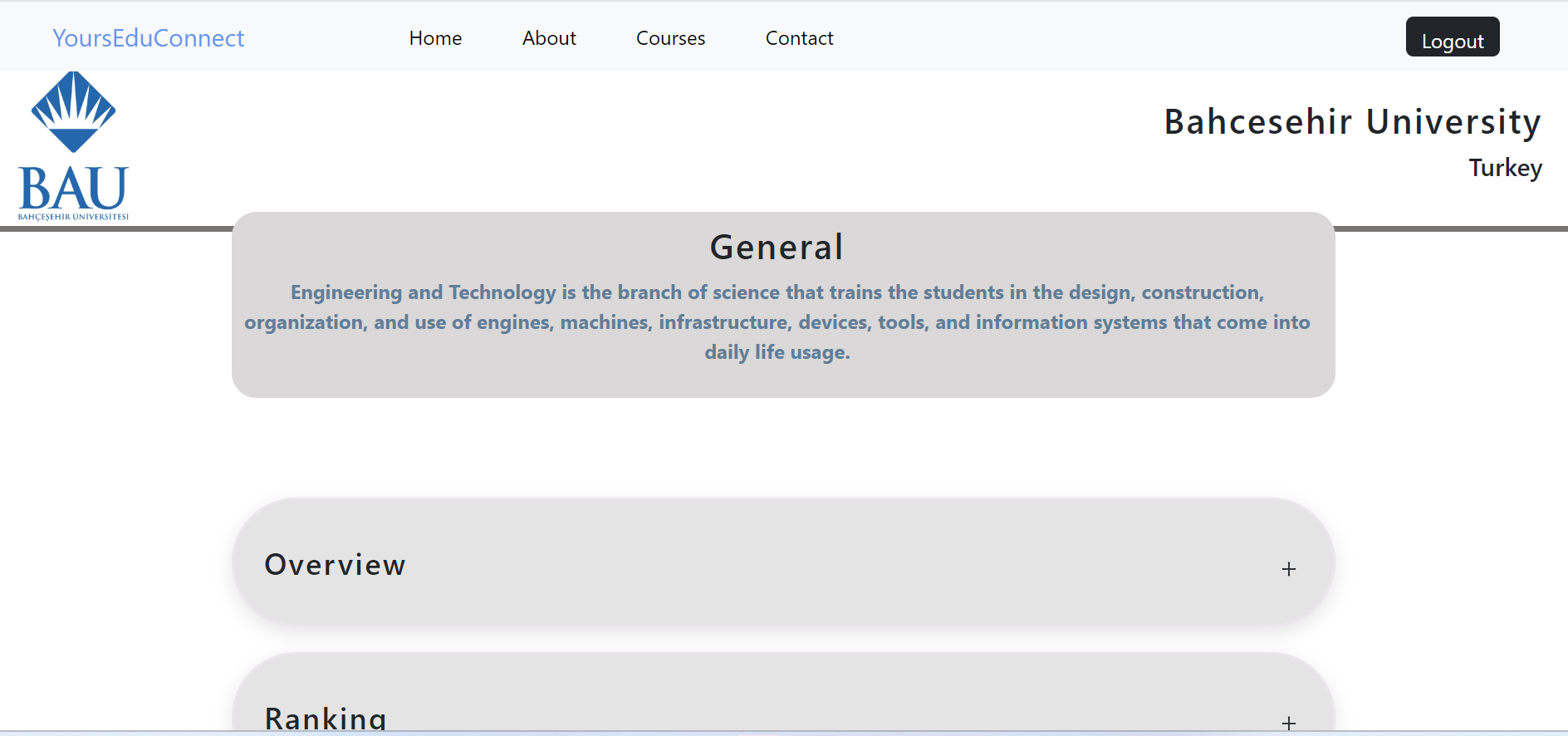
**Courses Page:**

******

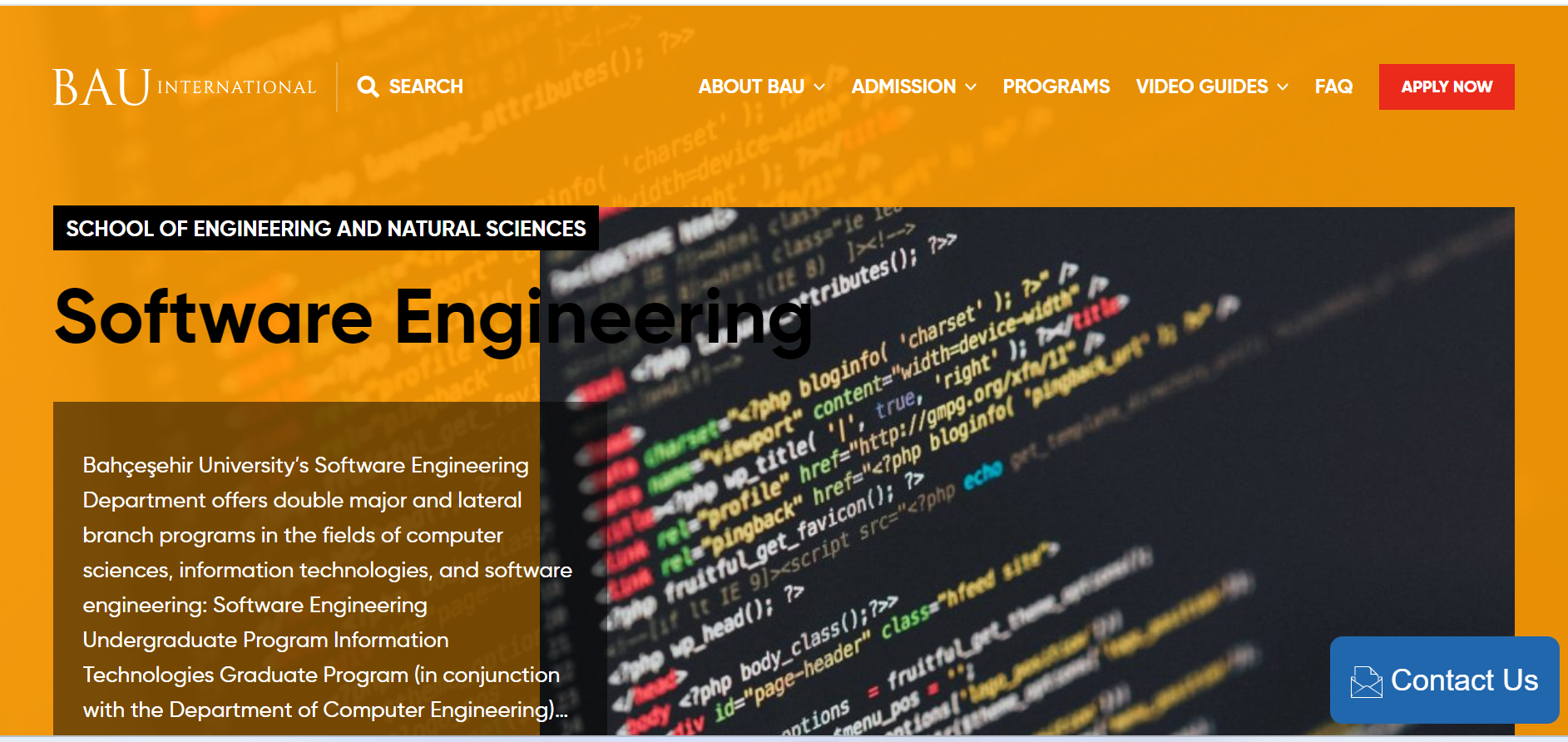
**Searching Course:**

****

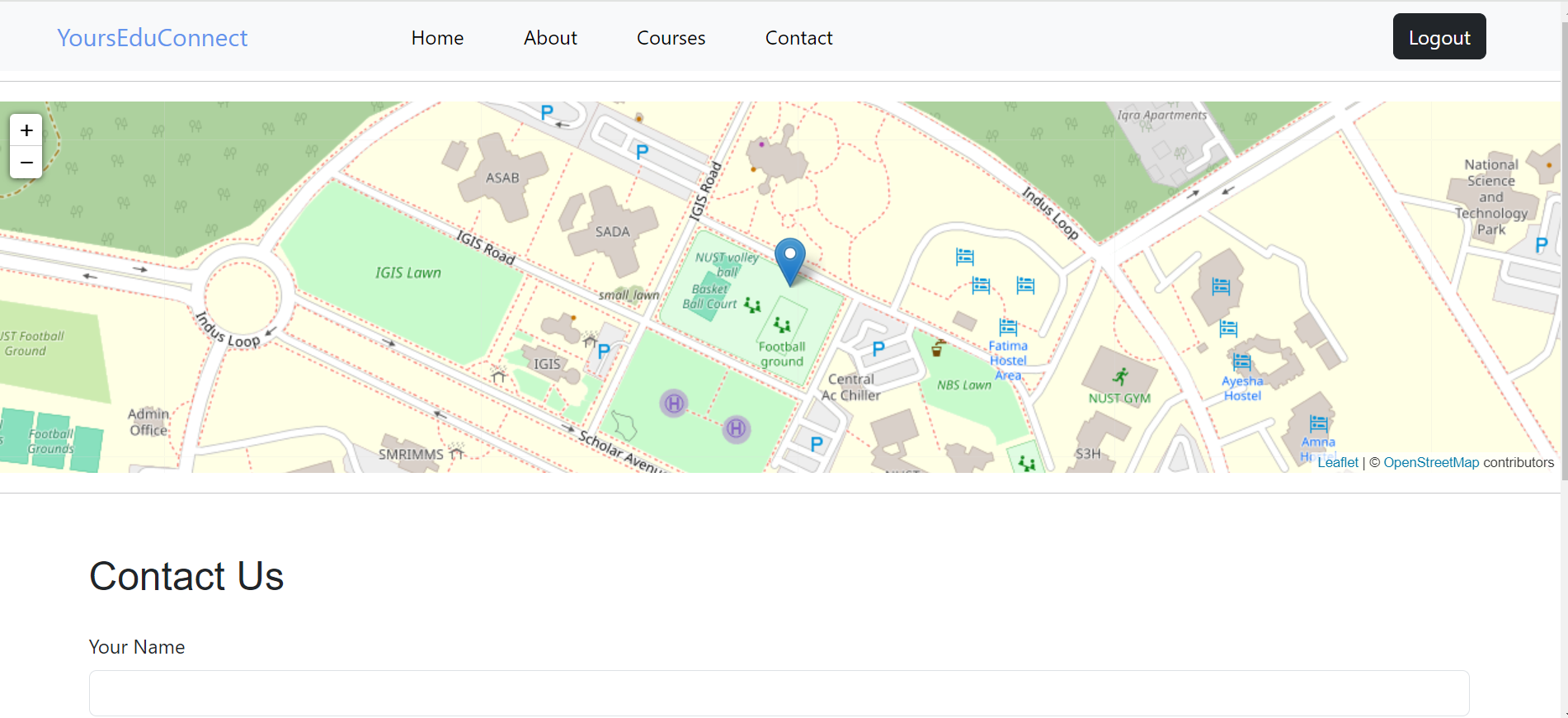
**More Details of Course University:**

****

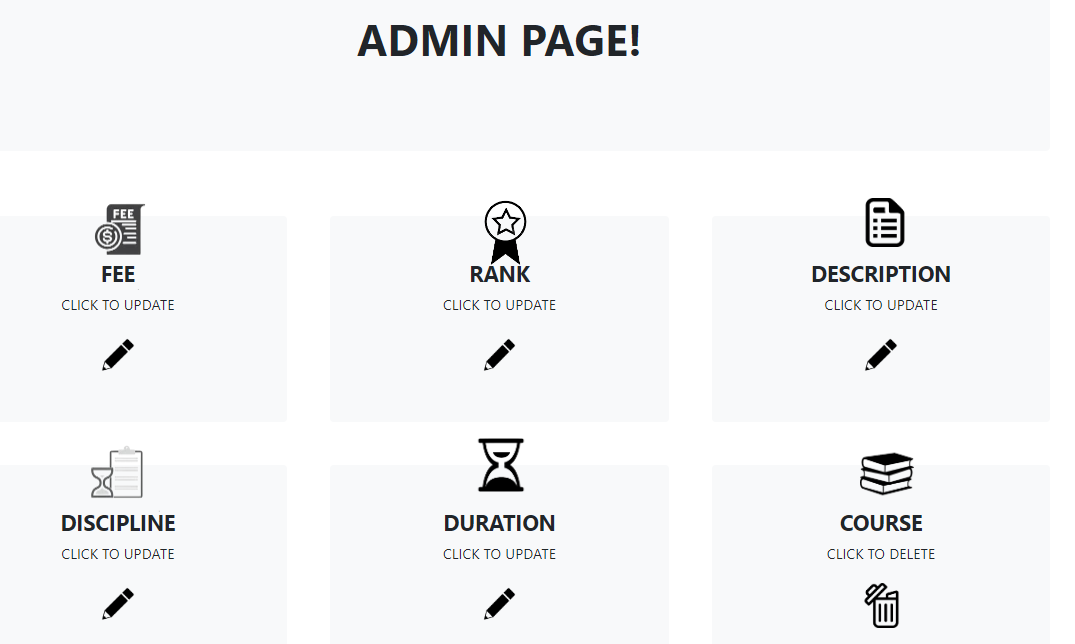
**University URL:**

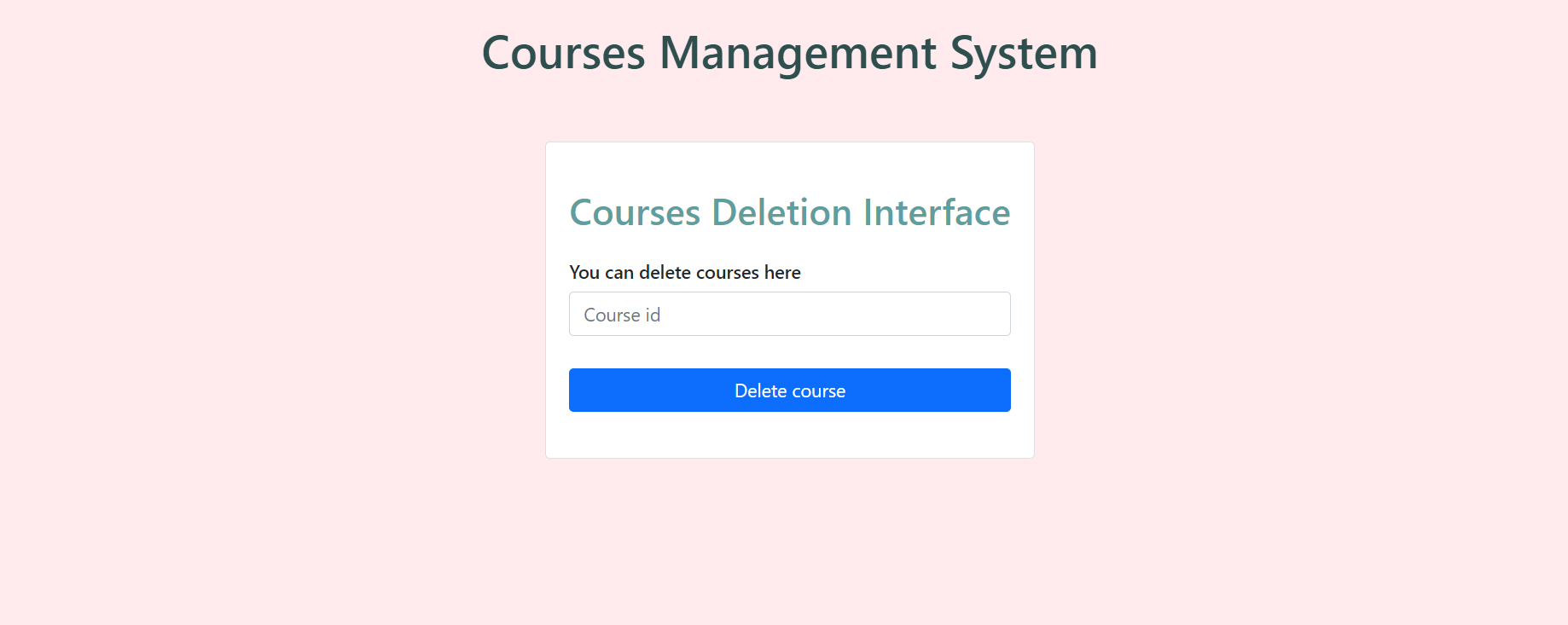
****

**Contact Page:**

****

**Admin side front page:**

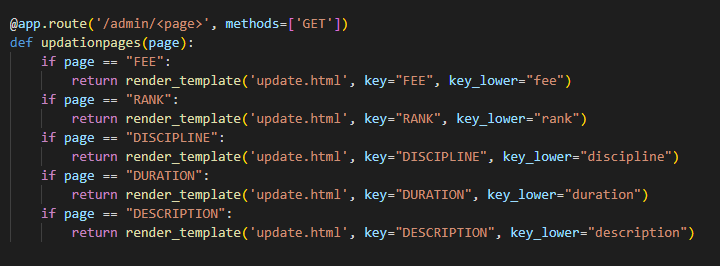
******

****

* **BACK-END DESIGN:**

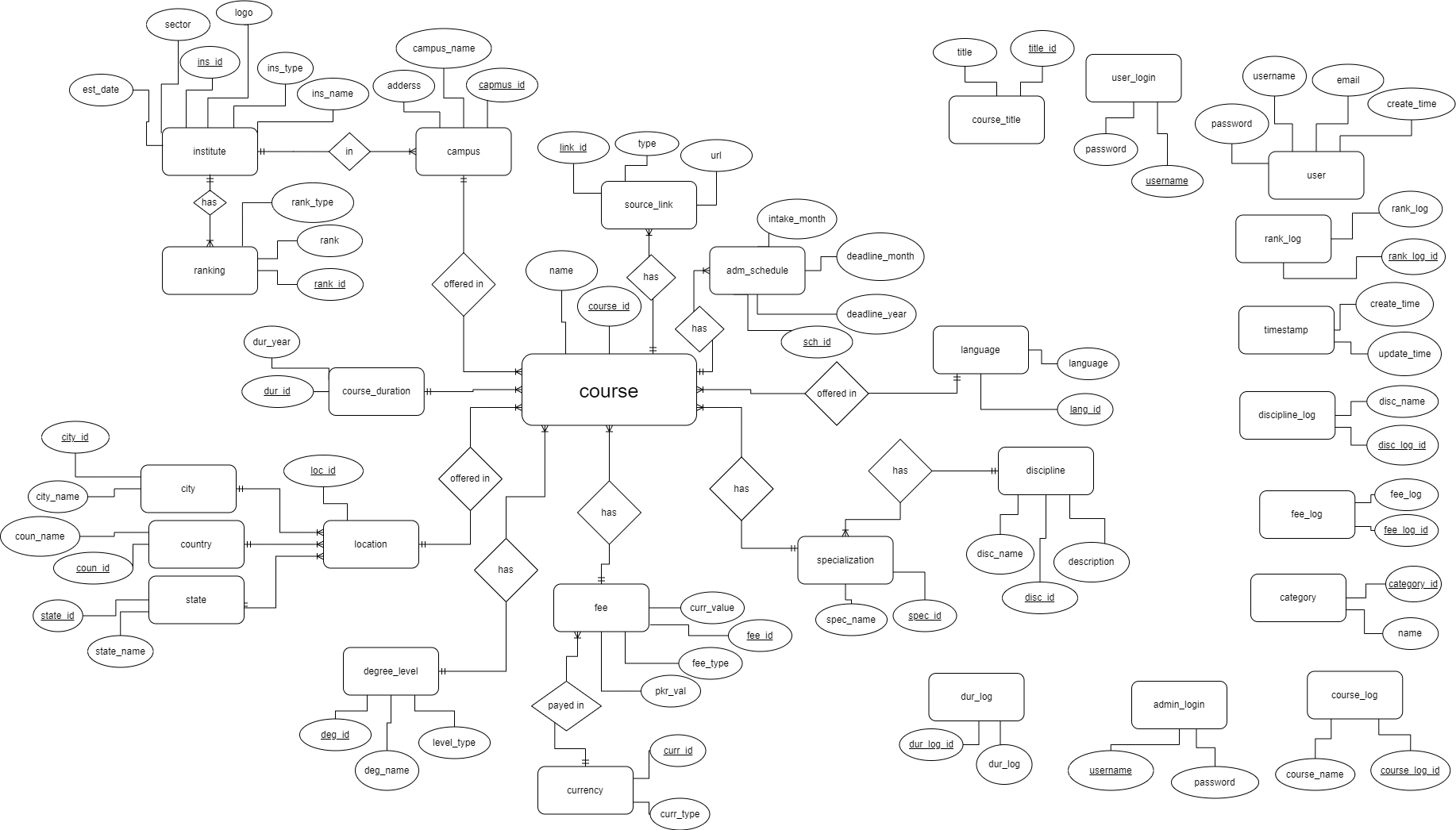
We used flask in python for backend. Flask helped us create dynamic jinja templates and made handling of database easily. It helped us render front end through its simple syntax and being in python, its connection with database and other code was very easy, simple and efficient.

Different functionalities of flask helped render different pages with different URLs dynamically through a simple code shown below:

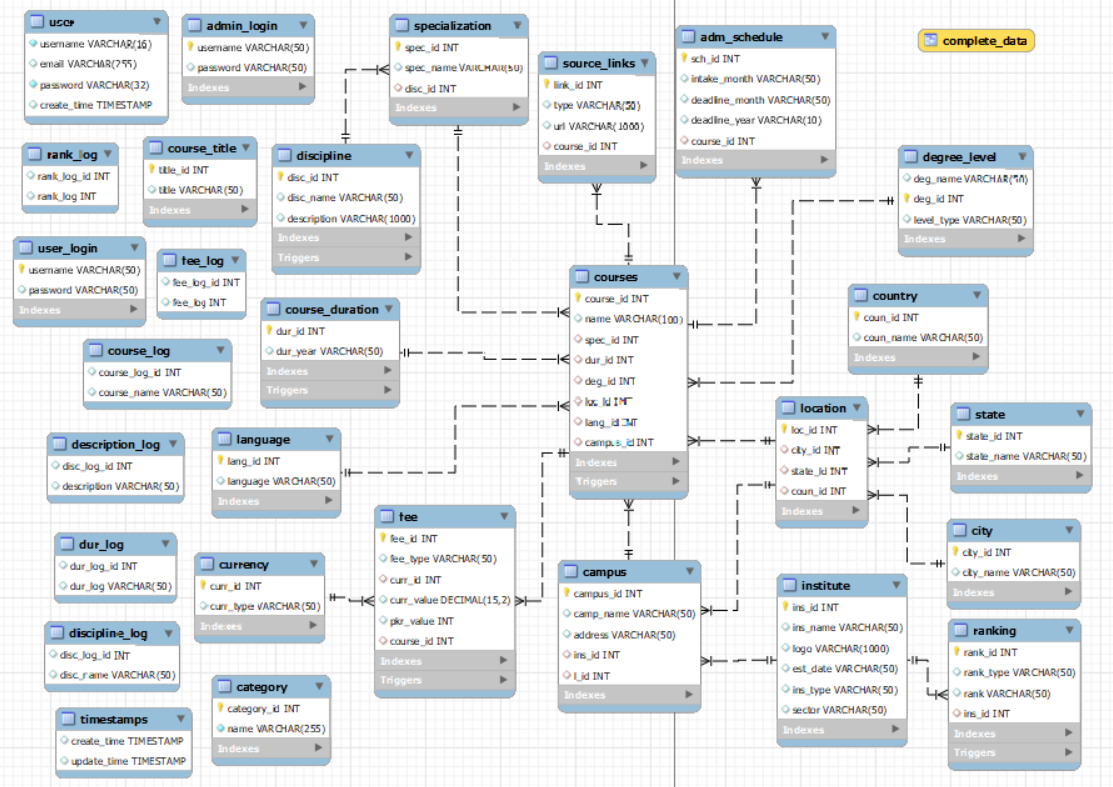


*Above code is used to render 5 front end pages. This is the efficiency and simplicity of flask code which urged us to use this for our backend.*

**Database Diagrams:**

**ERD**

**System Generated ERD:**



The main table in our ERD is the courses table. The table itself has only 2 attributes, course\_id and name. The rest of the attributes are foreign keys of all the other tables that are related to such as fee, duration, specialization etc.

Most of the independent tables are to keep a log of the previous states of certain tables, as well as to keep a record of the user and admin details for authentication.

**How to run code?**

Put all the files as placed in our project zip file. Then, dump our database in a new database named “project2” and lastly, install flask modules and run app.py.

**GitHub**

Link: <https://github.com/pro-Ethical-hacker/web-project.git>