**Technical Stacks Used in Developing the Native Language Learning Website**

1. Frontend Technologies

1.1 HTML5

Purpose: HTML5 is used for structuring the content on the website. It is the backbone of the webpage layout, providing the foundation for elements such as headings, paragraphs, forms, images, and links.

Usage: All the pages of the website, including the homepage, course creation page, and login forms, are structured using HTML5.

1.2 CSS3 and Bootstrap

Purpose: CSS3 is utilized for styling the website to make it visually appealing. Bootstrap 5, a powerful CSS framework, is used for responsive design, ensuring the site looks good on all devices.

Usage:

Bootstrap 5: Provides a grid system, buttons, forms, modals, and other components, which streamline the development process and maintain a consistent look and feel.

Custom CSS: Additional styling is implemented to customize Bootstrap components to fit the specific design needs of the website.

1.3 JavaScript (AJAX)

Purpose: JavaScript is used for client-side scripting to enhance user interactions and improve the website's responsiveness. AJAX is employed to allow asynchronous data submission, enabling parts of the webpage to update without reloading the entire page.

Usage:

AJAX is used on forms like the course creation form to handle submissions asynchronously, improving the user experience by showing modals or error messages without a full page reload.

2. Backend Technologies

2.1 Python

Purpose: Python is the core programming language used for the backend development of the website, chosen for its readability, simplicity, and powerful libraries.

Usage: Python powers the entire backend logic, including data processing, user authentication, and communication with the database.

2.2 Flask

Purpose: Flask is a micro web framework used to build the web application. It is lightweight and provides the necessary tools and libraries to create a robust backend while maintaining flexibility.

Usage:

Routing: Flask manages the URL routing of the website, connecting different URLs to the appropriate functions (e.g., creating a course, managing lessons).

Template Rendering: Flask's Jinja2 template engine is used to render HTML templates with dynamic data passed from the backend.

2.3 Flask-WTF

Purpose: Flask-WTF integrates Flask with WTForms, providing form handling and validation in a more Pythonic way.

Usage:

Form Handling: Used for creating and managing forms such as course creation, lesson count, and login forms. It provides CSRF protection and form validation.

Custom Validations: Custom validators ensure that user inputs meet specific criteria before processing.

2.4 Flask-Login

Purpose: Flask-Login is used to manage user sessions and authentication, ensuring secure access to different parts of the website based on user roles.

Usage: It handles user login, logout, and access control, ensuring that only authenticated users can create courses or manage lessons.

2.5 Flask-Migrate

Purpose: Flask-Migrate, an extension of Alembic, is used to handle database migrations in Flask applications.

Usage: It manages changes to the database schema, enabling smooth updates and versioning of the database as the project evolves.

3. Database Technologies

3.1 SQLAlchemy

Purpose: SQLAlchemy is an ORM (Object-Relational Mapping) library that provides a high-level abstraction for interacting with relational databases using Python objects instead of writing raw SQL queries.

Usage:

Data Models: SQLAlchemy models represent the database schema, including tables like `User`, `Course`, `Lesson`, and `Quiz`.

Database Operations: CRUD (Create, Read, Update, Delete) operations are handled through SQLAlchemy, ensuring efficient interaction with the database.

3.2 SQLite (or MySQL/PostgreSQL)

Purpose: SQLite is a lightweight, disk-based database used during development for its simplicity. For production, more robust databases like MySQL or PostgreSQL may be used.

Usage:

Storage: Stores all persistent data, including user information, course details, lessons, and quizzes.

Queries: Used to retrieve, insert, update, and delete data as needed by the application.

4. Image and Video Handling

4.1 File Uploads (Flask-Uploads)

Purpose: Flask-Uploads is used to handle file uploads in a secure and organized manner.

Usage:

Image Uploads: Course creators can upload images for their courses, which are stored on the server and linked to the course records in the database.

4.2 Video Embedding

Purpose: To provide rich multimedia content, the platform allows embedding of video URLs.

Usage: Video URLs can be added by course creators, and these videos are then embedded in the course content pages, providing an interactive learning experience.

5. Security

5.1 Flask-Bcrypt

Purpose: Flask-Bcrypt is used to hash user passwords, ensuring they are stored securely in the database.

Usage: Passwords are hashed before storage, and the hashes are checked during the login process to authenticate users securely.

5.2 CSRF Protection

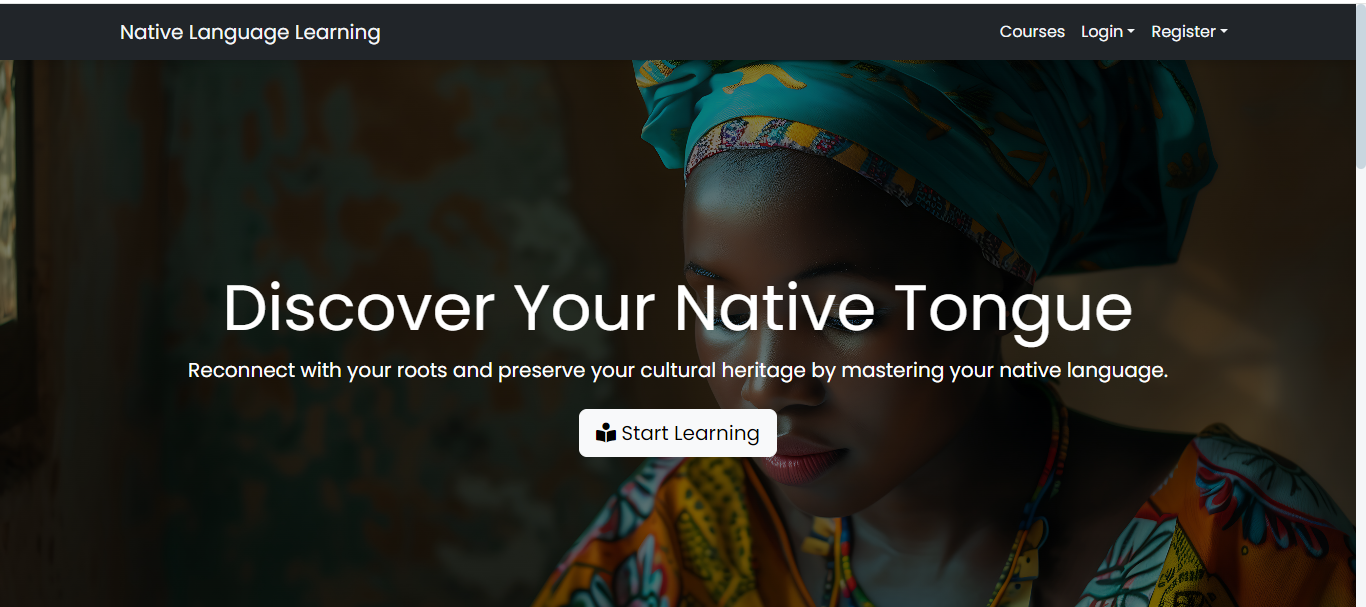
Purpose: Cross-Site Request Forgery (CSRF) protection is implemented to secure forms against unauthorized actions.

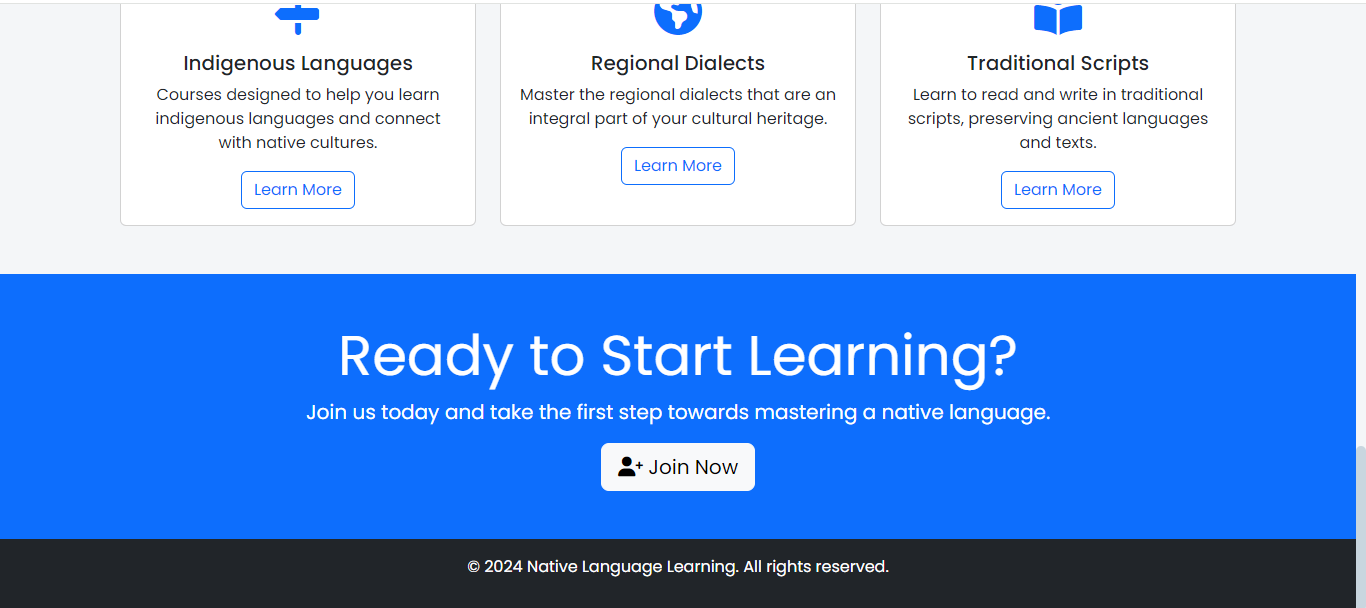
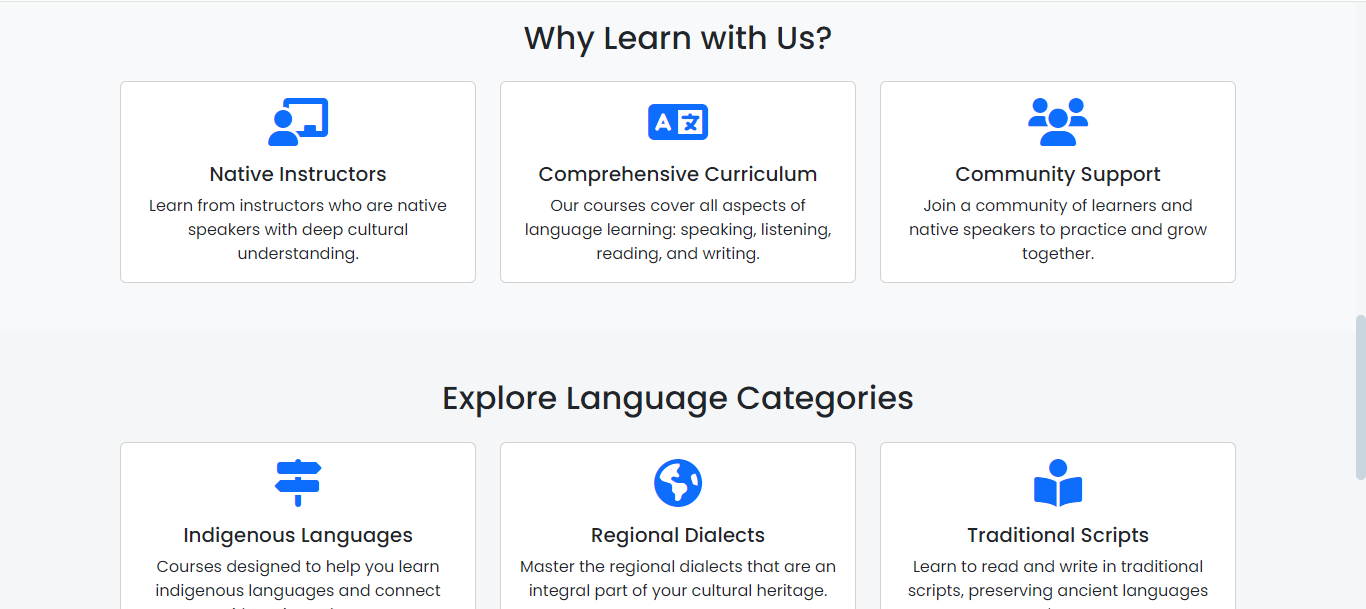
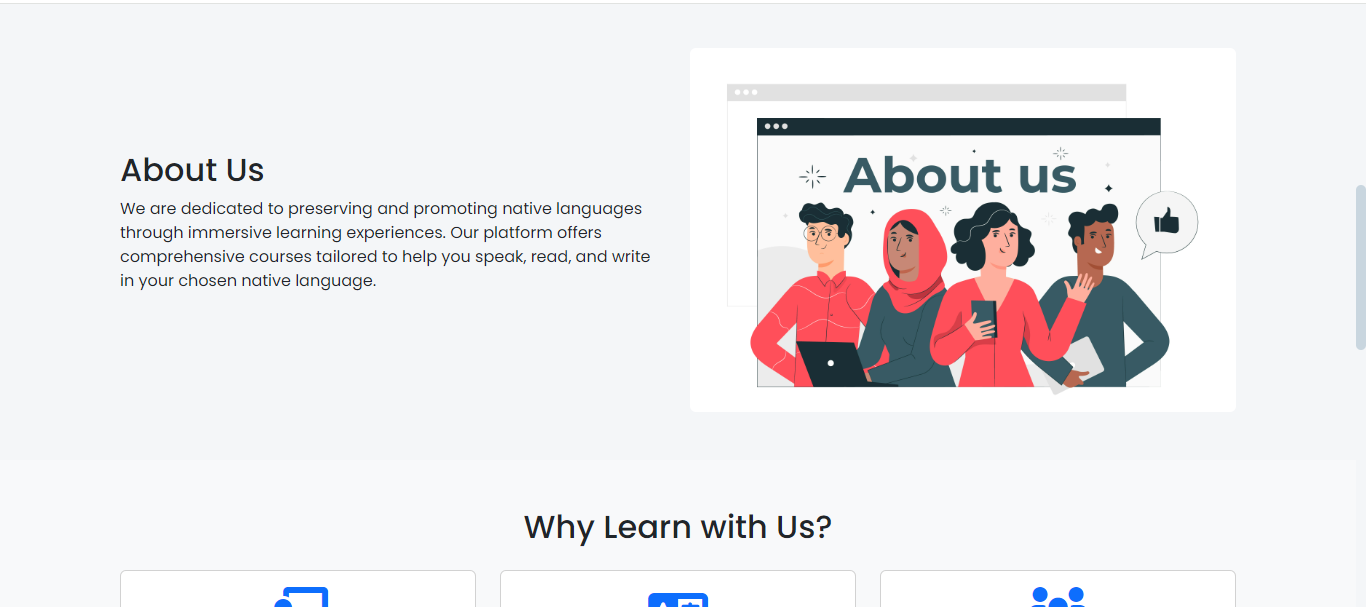
Usage: Flask-WTF automatically includes CSRF protection tokens in forms, ensuring that only legitimate requests from authenticated users are processed.

**Website Pages Overview**

**Home Page (Unauthenticated Users):**

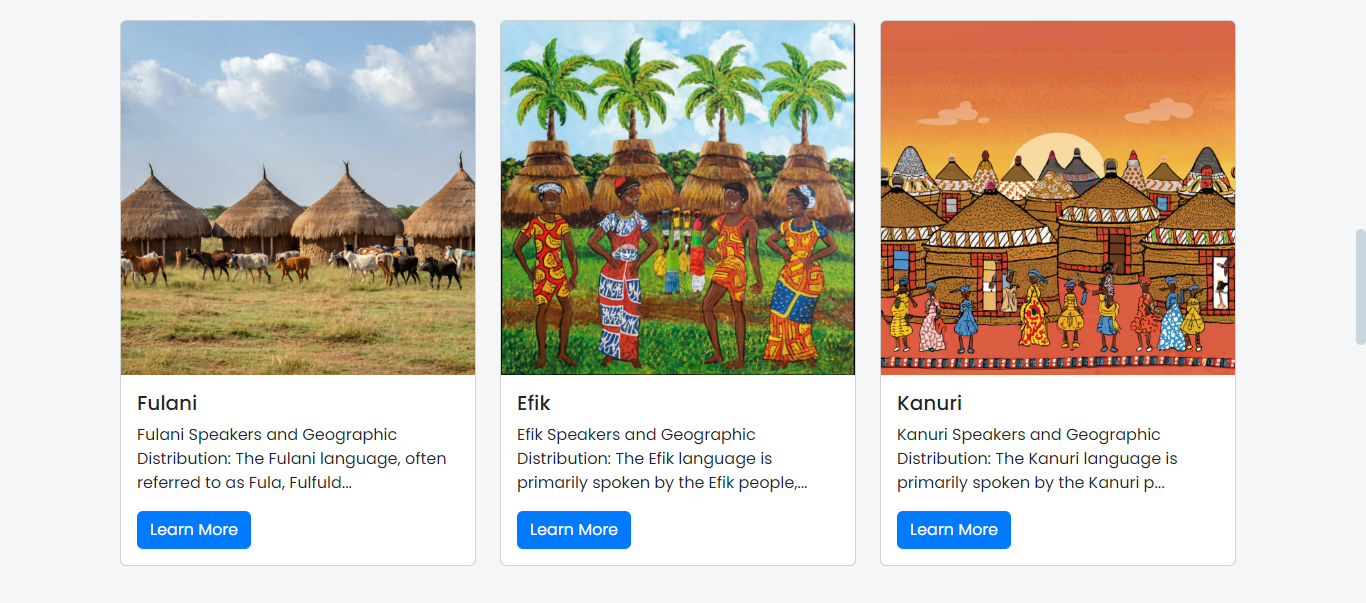
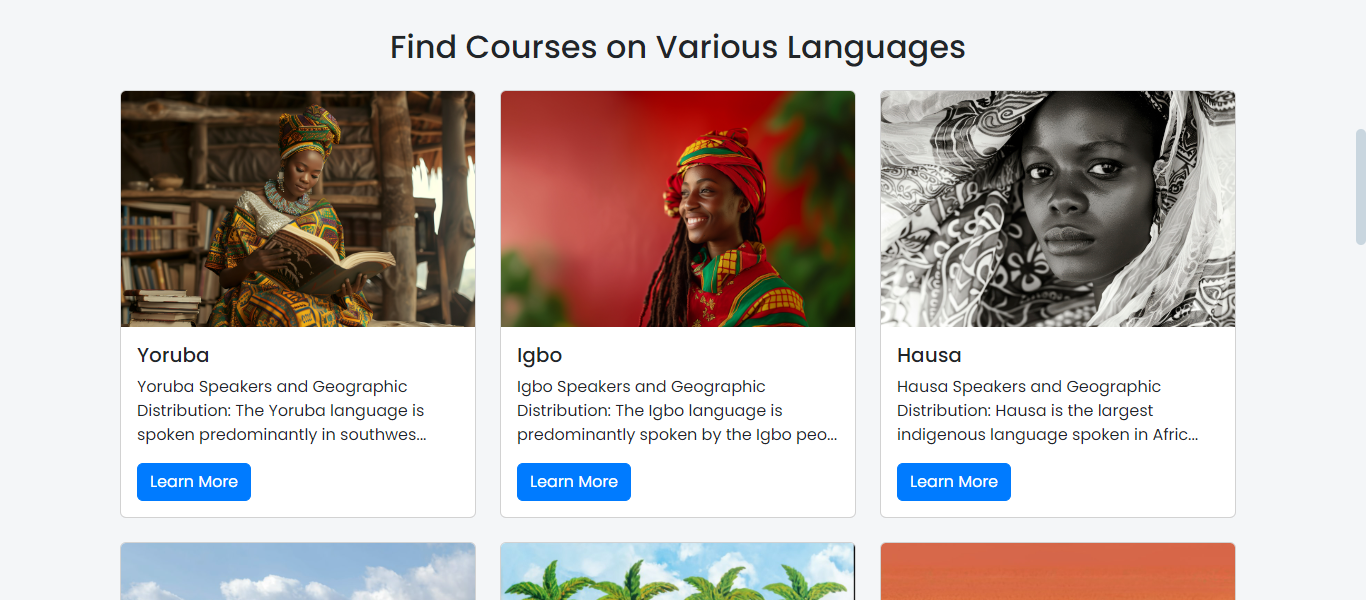
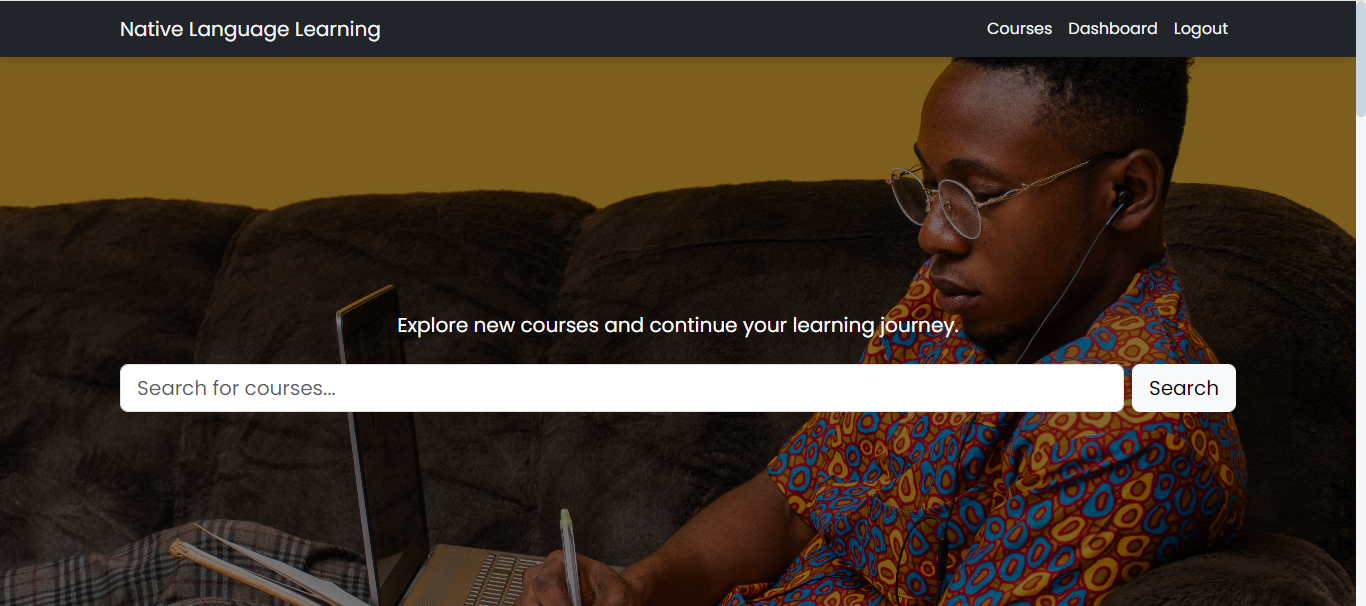
* **Purpose:** This page serves as the landing page for users who are not logged in. It introduces the website, its purpose, and the benefits of using the platform.
* **Features:**
  + **Hero Section:** A prominent hero section showcasing the platform's value proposition.
  + **About Section:** An explanation of the platform's goals, vision, and features.
  + **CTA (Call to Action):** Buttons prompting users to sign up or log in.





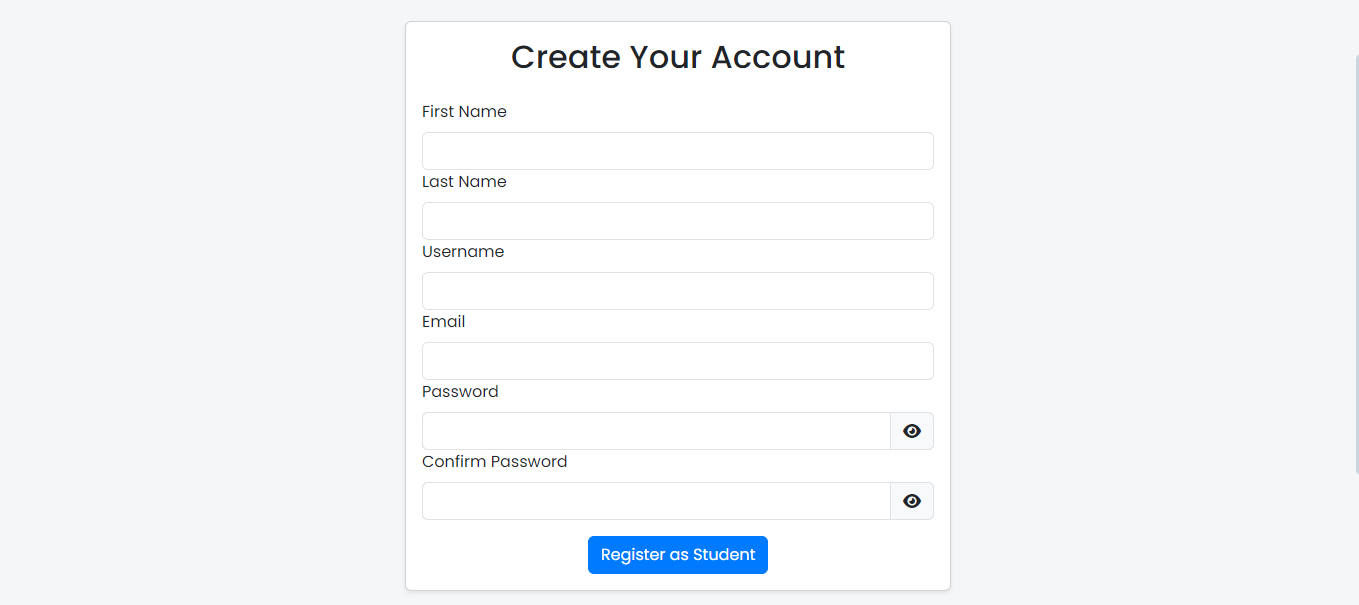
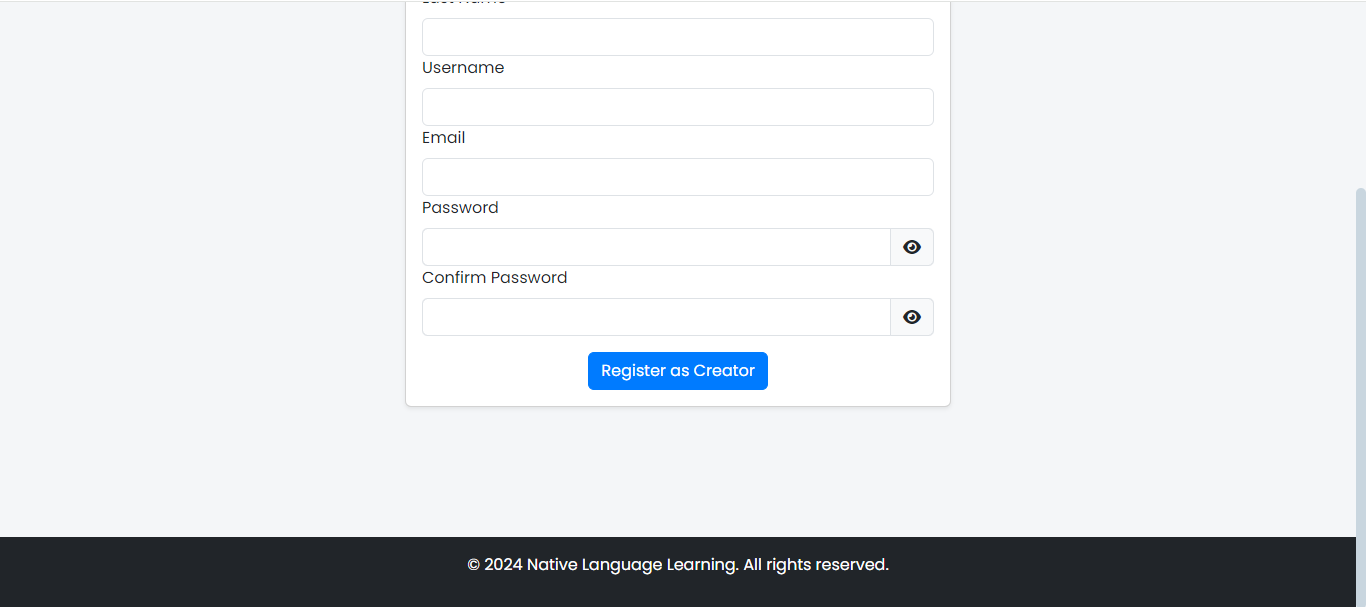
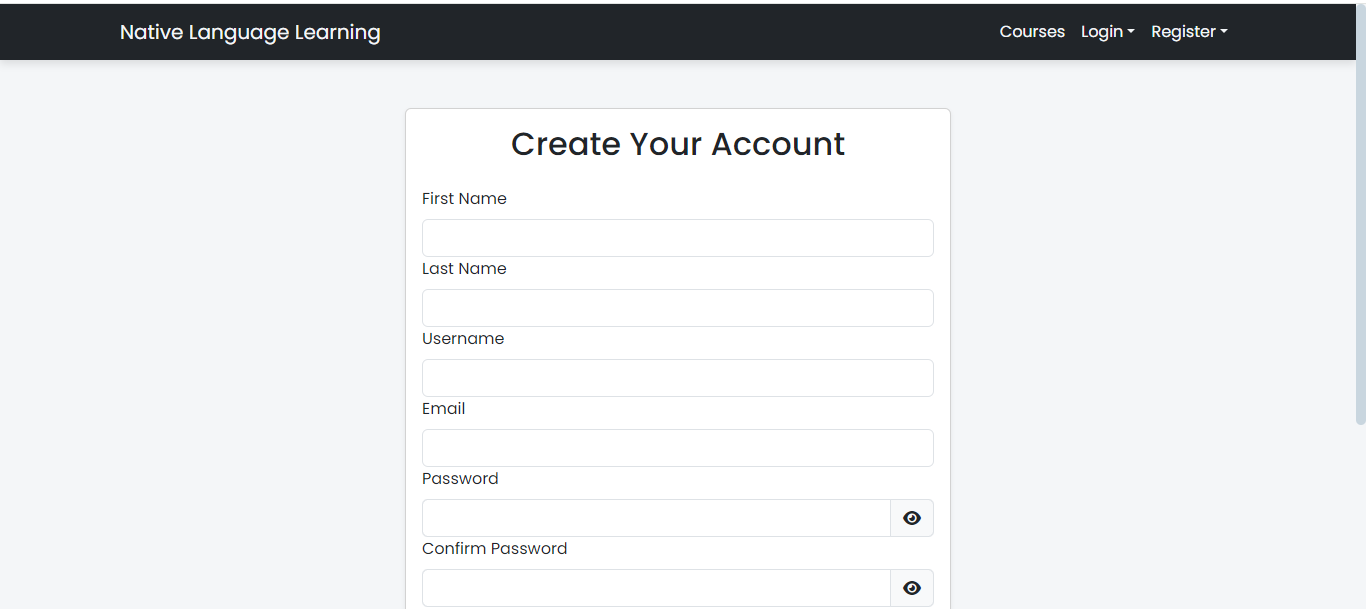
2. **Home Page (Authenticated Users):**

* **Purpose:** This page is tailored for users who are logged in, providing them with quick access to the main functionalities of the website.
* **Features:**
  + **Search Bar:** A prominent search bar for users to find courses quickly.
  + **Featured Courses:** Display of recommended or popular courses based on user preferences.
  + **Dashboard Link:** A link to the user's dashboard where they can manage their courses and profile.



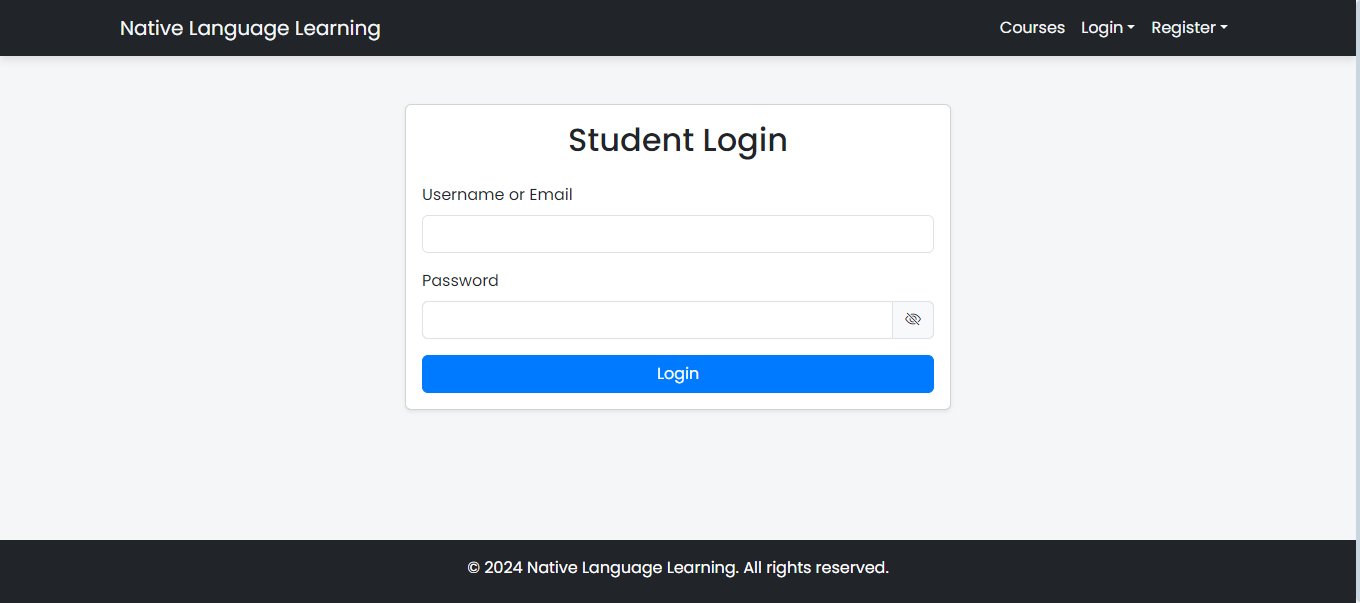
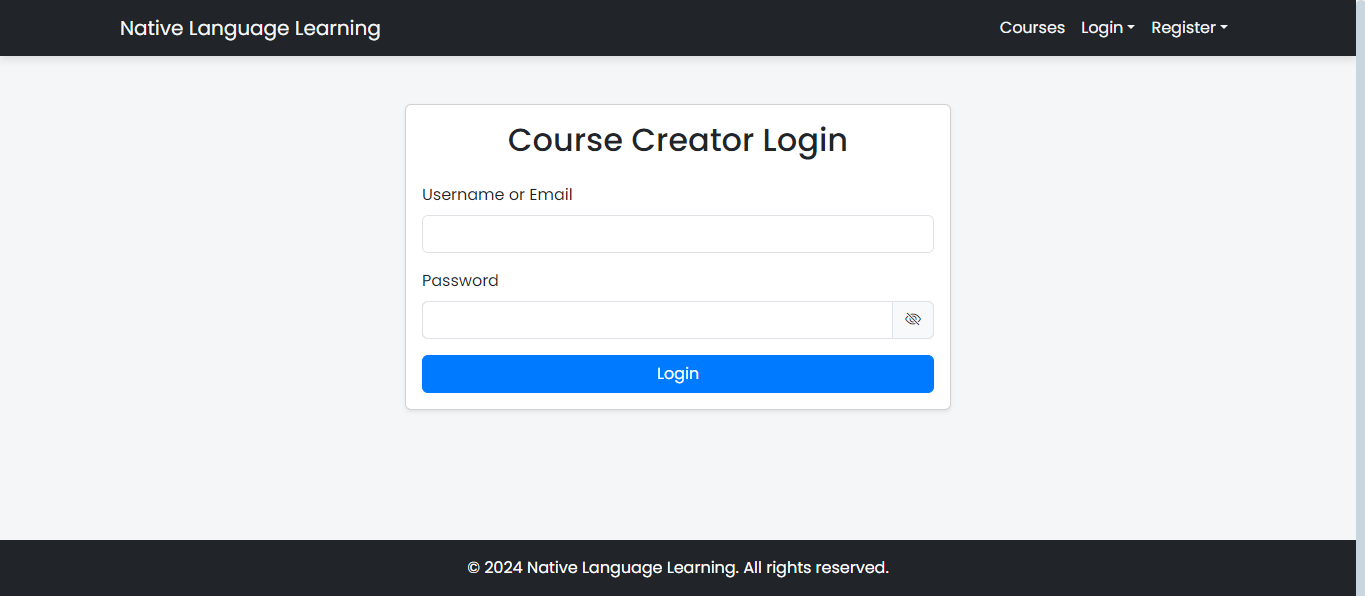
**User Registration Page**

* **Purpose**: Allows new users to sign up by providing their details.
* **Features**:
  + Input fields for username, email, password, and other personal information.
  + Client-side validation using JavaScript and jQuery.
  + Form submission handled by Flask-WTF with backend validation.
* **Technical Implementation**:
  + Bootstrap forms for styling.
  + CSRF protection using Flask-WTF.
  + Password hashing with Bcrypt before storing it in the database.



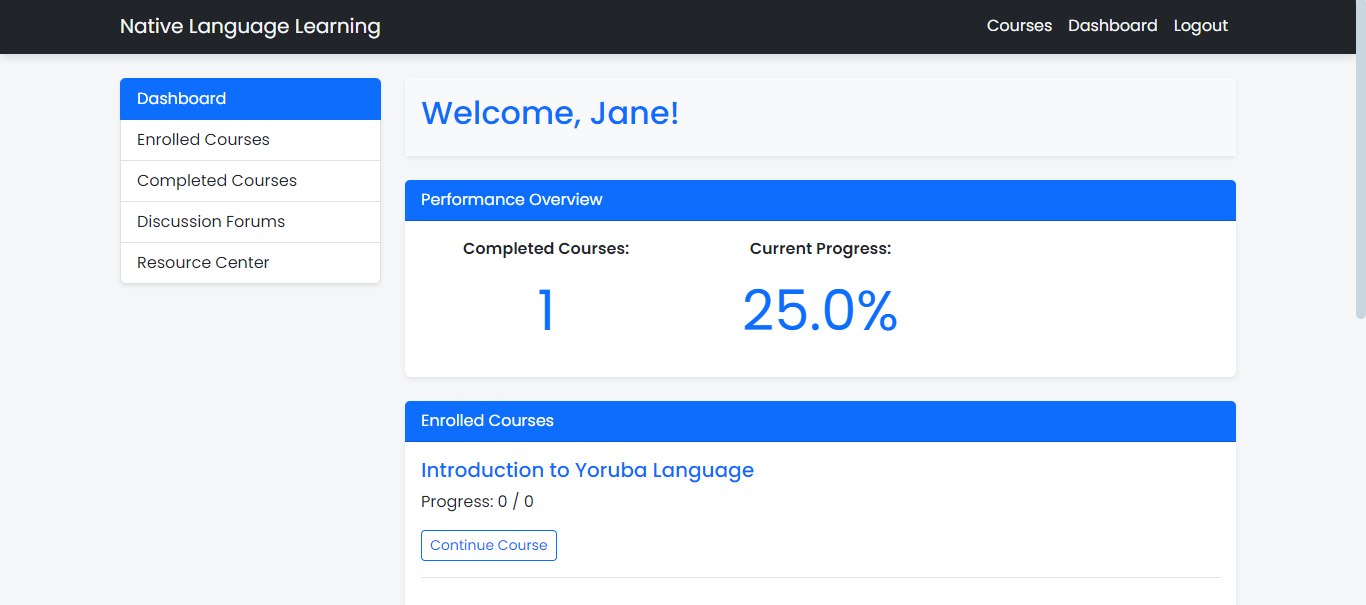
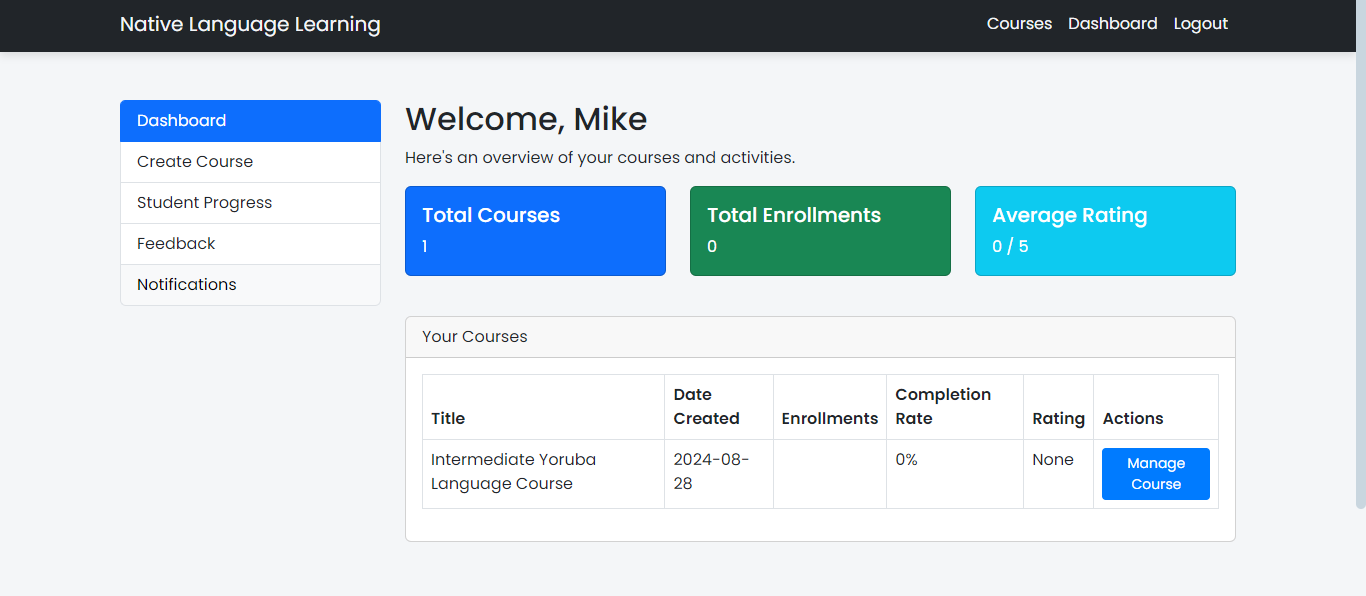
**Login Page**

* **Purpose**: Authenticates users, allowing them to access restricted parts of the website.
* **Features**:
  + Username/Email and password fields.
  + Error messages displayed if authentication fails.
* **Technical Implementation**:
  + User sessions managed by Flask-Login.
  + Custom JavaScript to toggle password visibility.



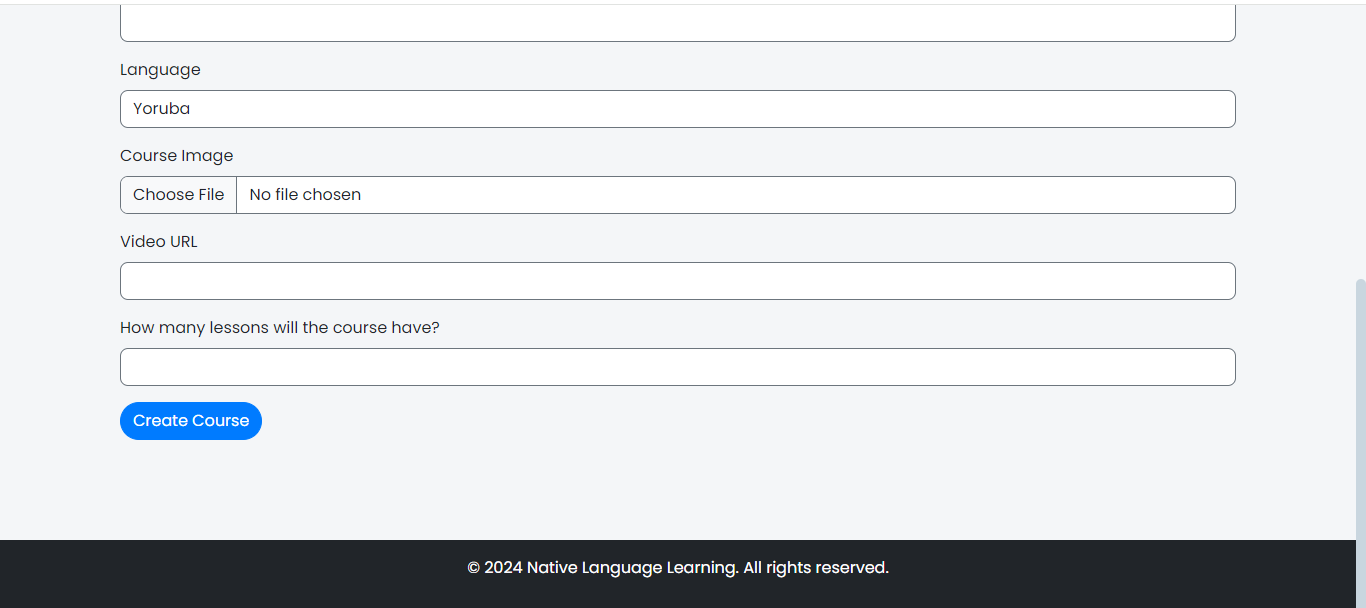
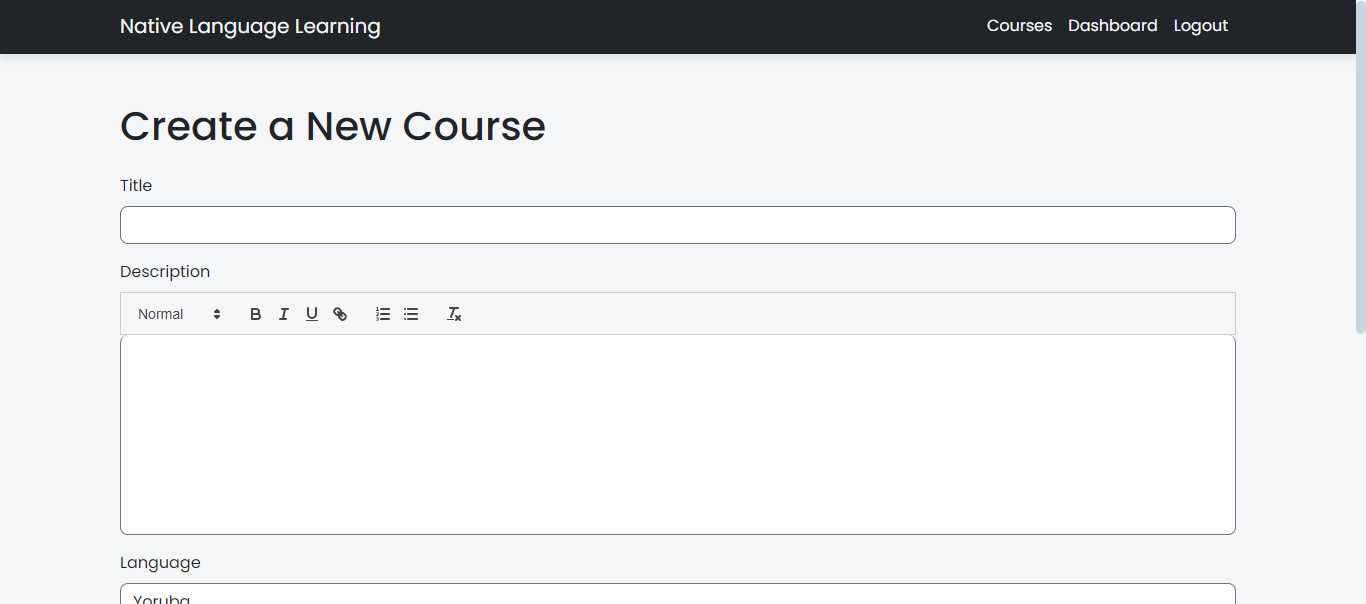
**Dashboard**

* **Purpose**: Provides an overview of user activities and quick access to key functionalities.
* **Features**:
  + Links to create new courses, view existing courses, and manage user settings.
  + Display of recent activities and updates.
* **Technical Implementation**:
  + Dynamic content loaded based on user role (student or instructor).
  + Flask routes for handling various user actions.

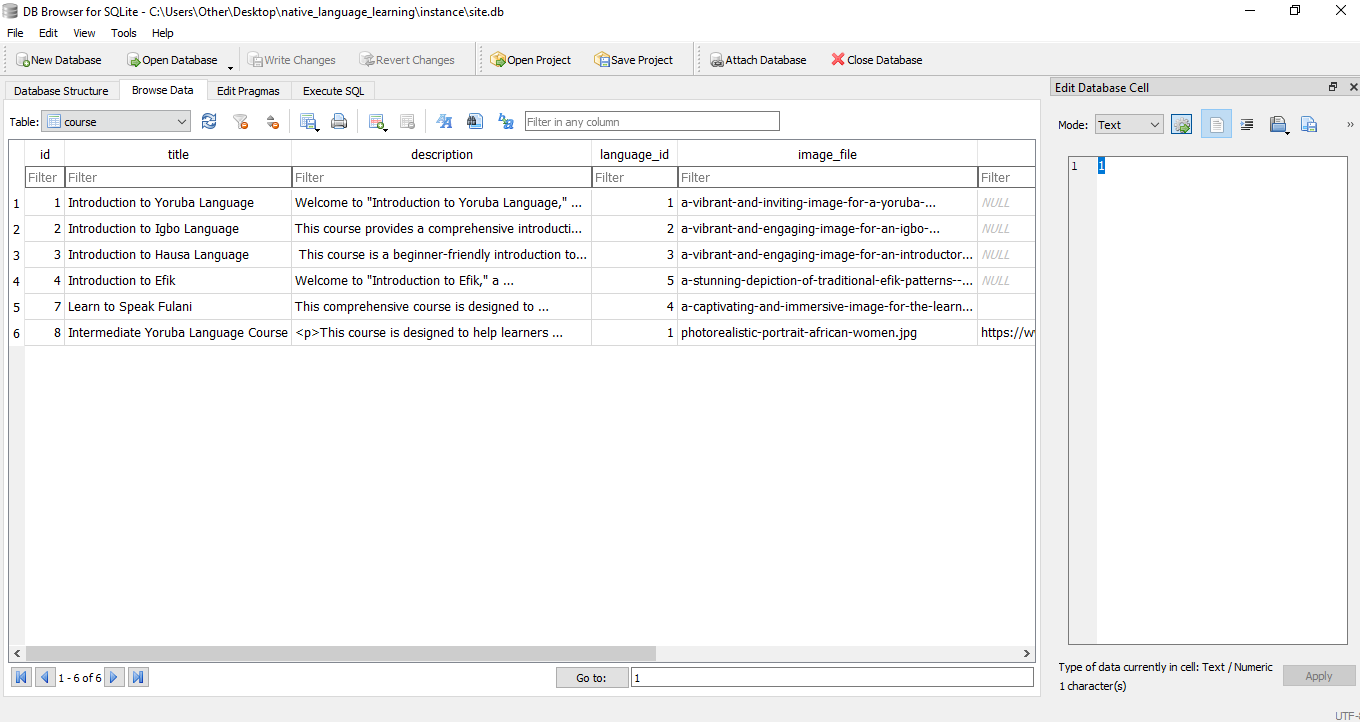


**Course Creation Page**

* **Purpose**: Enables instructors to create and manage their courses.
* **Features**:
  + Form fields to enter course title, description, language, and upload images/videos.
  + Modal to specify the number of lessons in the course.
  + AJAX form submission for a seamless user experience.
* **Technical Implementation**:
  + File uploads handled by Flask-Uploads.
  + Lesson count specified in a modal triggered by AJAX.
  + Backend logic to save course details in the database.



Database(SQLite):



Conclusion:

The development of the Native Language Learning Website leverages a range of modern technologies and frameworks, ensuring a responsive, secure, and efficient application. The frontend stack provides a visually appealing and user-friendly interface, while the backend technologies ensure robust and secure data handling