**Student Registration Application Technical Documentation**

**Introduction**

The Student Registration Application is a web-based platform developed using ASP.NET Core Razor Pages and Microsoft SQL Server (MSSQL) as the database backend. This application facilitates the management of student information, including registration and authentication processes. It ensures data security by implementing password hashing and protecting against SQL injection attacks.

**Business Impact**

Implementing the Student Registration Application offers several business benefits:

* **Efficiency**: Streamlines the student enrollment process, reducing administrative overhead.
* **Data Integrity**: Ensures accurate and consistent student records.
* **Security**: Protects sensitive information through robust authentication and authorization mechanisms.
* **Scalability**: Supports growth in the number of students and courses without significant changes to the system architecture.

**Web Architecture**

The application follows a layered architecture comprising:

1. **Presentation Layer**: Utilizes ASP.NET Core Razor Pages to render dynamic web pages. Razor Pages offer a page-centric development model, promoting clean separation of concerns and improved testability.
2. **Business Logic Layer**: Contains the core functionality, including validation rules and processing logic. This layer is implemented within the PageModel classes associated with each Razor Page, ensuring that the UI remains free of business logic.
3. **Database Layer**: The MSSQL database stores all application data, including user credentials, student profiles, and course information. The database schema is designed to maintain referential integrity and optimize query performance.

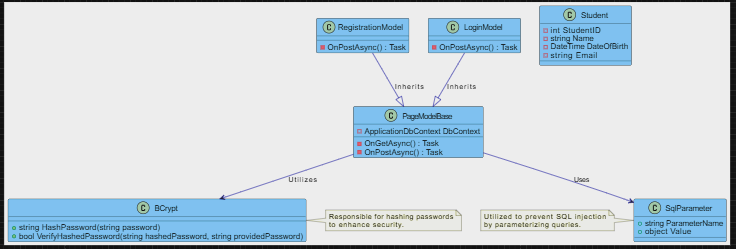
**Security Measures**

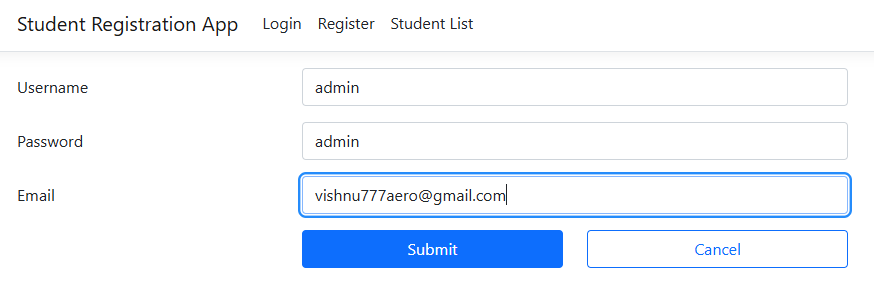
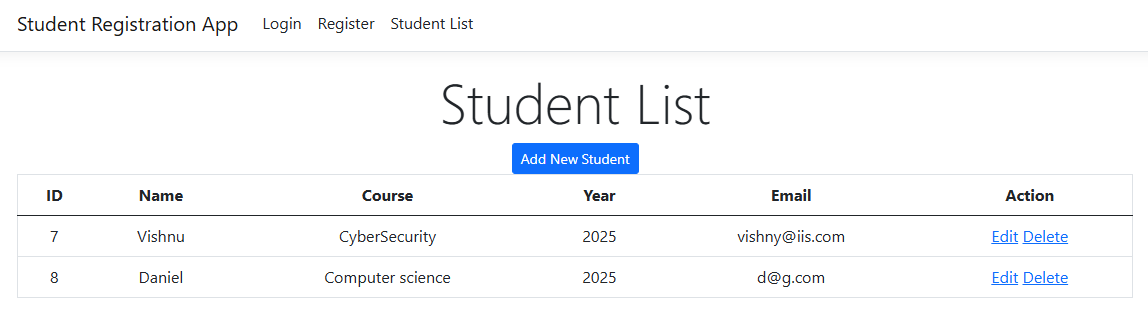
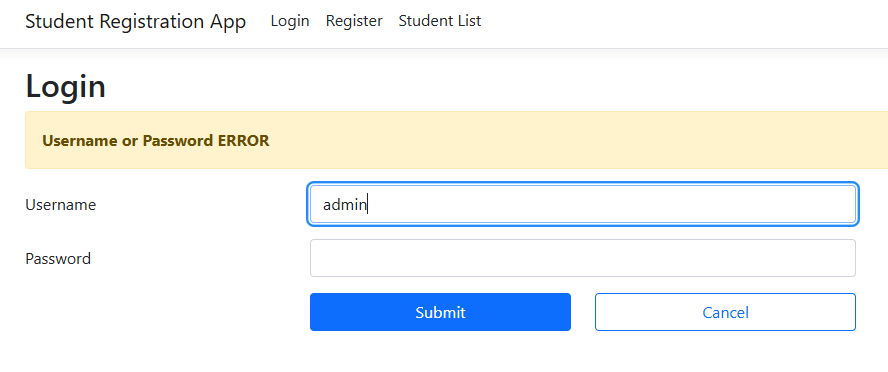
* **Password Hashing**: User passwords are hashed using industry-standard algorithms before storage, ensuring that plaintext passwords are never saved in the database.
* **SQL Injection Prevention**: The application employs parameterized queries and ORM features to safeguard against SQL injection attacks, ensuring that all database inputs are validated and sanitized.
* **SSL Certificate**: Uses self made SSL certificate for secure connection to local host

**Vulnerabilities**

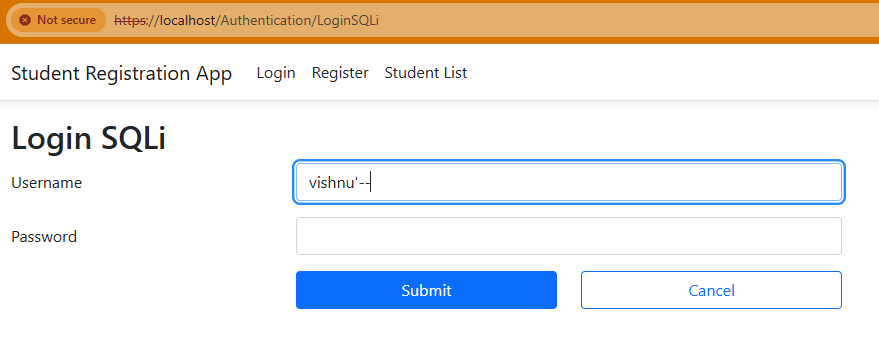
* Added login page with SQL injection vulnerabilities
* Added filter to Student registration portal where you can use complex attacks like UNION attacks

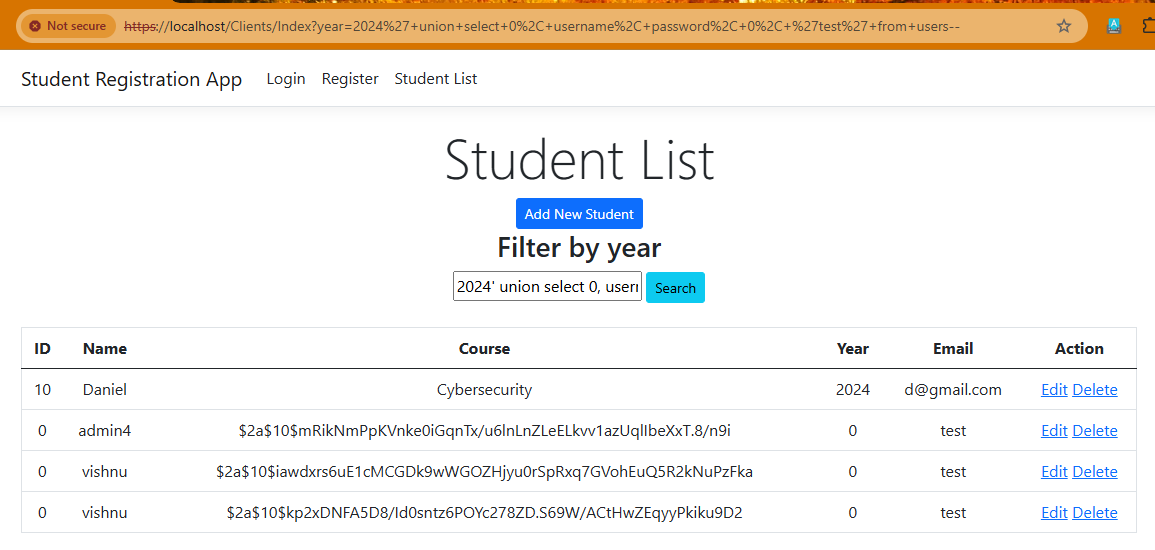
**UML Diagram**



**UI Diagrams (deliverable1)**

**(deliverable2)**

****

****

**Next Steps**

* **Add two factor authentication**
* **Add XSS vulnerability**
* **Make sure only logged in user can navigate between pages using session**