**1. Introduction**

* **Objective**: Develop a responsive web-based student registration system.
* **Devices**: Must support both PCs and mobile devices.

**2. Objectives**

* **User Interface**: Must be user-friendly.
* **Data Security**: Ensure secure data capture and storage.
* **User Updates**: Allow users to update their information.
* **Responsive Design**: Support both PC and mobile devices.

**3. Functional Requirements**

**3.1 User Registration**

* **User Story**: New student Emma registers via mobile, enters personal details, and submits the form. Receives a confirmation email.
* **Fields and Rules**:
  + **Title**: Drop-down list (Mandatory; Options: Mr., Ms., Mrs., Miss, Dr., Prof.)
  + **First Names**: Text input (Mandatory)
  + **Surname**: Text input (Mandatory)
  + **Address Details**: Text input (Mandatory; Fields: Street, City, State, ZIP Code)
  + **Email Address**: Email input (Mandatory; Unique)
  + **Highest Qualifications**: Text input or drop-down list (Optional)
  + **Password**: Password input (Mandatory; Must be entered twice; Strong password policy)

**3.2 User Login**

* **User Story**: Emma logs in using her email and password to view her registration details.
* **Fields and Rules**:
  + **Email Address**: Email input (Mandatory)
  + **Password**: Password input (Mandatory)

**3.3 View User Information**

* **User Story**: Emma reviews her profile details in the "My Profile" section.

**3.4 Update User Information**

* **User Story**: Emma updates her address and email in the "Edit Profile" section and saves the changes.

**4. System Architecture**

**4.1 Front-End**

* **Technologies**: HTML5, CSS3, JavaScript (Frameworks/Libraries: React, Angular, PHP, MVC)
* **Responsiveness**: Bootstrap or equivalent framework.

**4.2 Back-End**

* **Programming Language**: Python, Node.js, PHP, Java
* **Frameworks**: Django (Python), Express (Node.js), Asp.net (Web API)
* **Database**: Relational (MySQL, MSSQL Express, PostgreSQL) or NoSQL (MongoDB)

**4.3 API**

* **Functionality**: RESTful API endpoints for registration, login, information retrieval, and updates.
* **Security**: Token-based authentication (JWT).

**5. Testing**

* **Tests**: Unit tests, integration tests, end-to-end tests.
* **Security Testing**: Identify and fix vulnerabilities.

**Next Steps**

1. **Front-End Development**: Implement user interface using HTML5, CSS3, and JavaScript. Ensure responsiveness with Bootstrap.
2. **Back-End Development**: Set up server using a preferred programming language and framework. Develop API endpoints.
3. **Database Configuration**: Choose and configure a suitable database system.
4. **Testing**: Develop and run tests to ensure functionality and security.
5. **Documentation and Deployment**: Prepare deployment documentation and deploy the system on a suitable server.