**Developing a Contact Management System with Ensured Quality Attributes**

The project is to develop a basic contact management system, emphasizing **defensive programming** principles and considering a provided **quality attributes checklist**.

### The project consists of two main modules:

1. **User authentication**
2. **Contact management**

Upon accessing the application, the user is required to utilize the provided username and password to log in to the system. Once authenticated, the user will be directed to a page where they can Create, Read, Update, and Delete contact records. Each contact must adhere to specified properties and validation criteria, ensuring that the user cannot submit invalid information, as given in the table below.

|  |  |
| --- | --- |
| Field | Validation |
| First name | Text |
| Last name | Text |
| Company | Text |
| Phone | +1 (xxx) xxx-xxxx |
| Email | Email address |
| Website | http(s)://x.x |
| Address – Unit number | Number |
| Address – Civic number | Number |
| Address – Street | Text |
| Address – City | Text |
| Address – Province | 2 Capital character |
| Address – Postal code | ANA NAN (A represents an alphabetical, N represents a number) |

**Rules to follow:**

* Users must receive relevant error messages; generic error messages are not acceptable.
* The UX design and user interaction with the system are managed by the students. Users must be able to view a list of contacts, as well as view, edit, create, and delete each record.
* Server-side errors, such as a database connection error, will return a generic error message, and the error message should be logged on the server (either in a file or in the database).
* Unauthorized users are not permitted to access any records.
* All other relevant code quality attributes specified in this document or the provided checklist must also be adhered to.

**Project Specifications:**

* **Programming Language:** Flexible (e.g., Python, Java, C#), the quality checklist should be adapted by the student to the chosen language.
* **Deliverables:**
  + - Code implementing the chosen functionality, fully functional.
    - Documentation covering:
      * Design decisions and provided quality attributes.
      * Applied defensive programming techniques and their impacts.
* Presentation explaining the code and quality considerations.

**Defensive Programming Focus:**

* **Input Validation:** Thoroughly validate all user input and external data to prevent errors and vulnerabilities.
* **Error Handling:** Implement robust error handling mechanisms to gracefully handle unexpected situations.
* **Bound Checking:** Ensure all array and pointer accesses are within valid bounds to prevent memory corruption.
* **Type Safety:** Utilize type systems effectively to catch potential errors early.
* **Exception Handling:** Use exceptions judiciously for exceptional situations, not for regular control flow.

**Quality Attributes to consider:**

**Use the checklist and below attributes to guide you through the system development process.**

* **Correctness:** Does the code function as intended, meeting all requirements?
* **Reliability:** Does the code perform consistently without failures or crashes?
* **Usability:** Is the code user-friendly and intuitive?
* **Performance:** Does the code execute efficiently with reasonable resource usage?
* **Security:** Is the code secure against vulnerabilities and unauthorized access?
* **Maintainability:** Is the code well-structured, documented, and easy to understand and modify?