***NAME : NOFAL ZULFIKAR***

***ID : F2023065363***

***ASSIGNMENT***

***WORK BREAK DOWN FOR PASSWORD GENERATOR***

**1. Requirements Analysis**

**Functional Requirements**

* Define core functionalities:
  + Password generation with customizable options.
  + Support for uppercase, lowercase, numbers, and special characters.
  + Adjustable password length.
* Identify additional features:
  + Clipboard support.
  + Bulk password generation.
  + Password strength indicator.
* Validate requirements with stakeholders.

**Non-Functional Requirements**

* Define security requirements (e.g., avoid predictable patterns).
* Ensure usability and accessibility.
* Define performance benchmarks (e.g., < 1 second to generate passwords).

**2. Design Phase**

**System Design**

* Define architecture:
  + Modular design for password generation logic.
  + Separate components for user input and output.
* Identify key modules:
  + Input module: User selects password criteria.
  + Password generator module: Implements randomization and character selection.
  + Output module: Displays results or saves to file.

**User Interface Design**

* **Command Line Interface (CLI)**:
  + Create a flow for user prompts and responses.
* **Graphical User Interface (GUI)**:
  + Sketch wireframes for:
    - Input options (checkboxes, sliders).
    - Generate button and password display area.
* Include validation and error messages in UI.

**Algorithm Design**

* Define character pools:
  + Uppercase letters (A-Z), lowercase letters (a-z), digits (0-9), and symbols.
* Implement randomization logic.
* Ensure inclusion of at least one character from each selected pool.
* Optimize for performance and security.

**3. Implementation Phase**

**Backend Development**

* Develop password generation logic:
  + Implement random character selection.
  + Handle user-defined constraints (e.g., minimum length, specific characters).
* Add input validation:
  + Ensure valid password length and options.
  + Handle edge cases (e.g., zero-length password).
* Ensure compatibility with the randomization library for chosen language.

**Frontend Development**

* Implement CLI:
  + Create interactive prompts for user input.
  + Display generated password(s).
* Develop GUI (if applicable):
  + Create forms for user input.
  + Add interactive elements (e.g., sliders, checkboxes).
  + Integrate backend logic with GUI components.

**Integration**

* Connect frontend with backend.
* Test the flow from user input to password generation and output.

**4. Testing and Debugging**

**Unit Testing**

* Test individual functions:
  + Randomization logic.
  + Character pool handling.
  + Input validation.

**Integration Testing**

* Validate communication between components:
  + Ensure UI input passes correct parameters to backend.
  + Verify output is correctly displayed.

**User Testing**

* Conduct user tests:
  + Test usability and ease of navigation.
  + Gather feedback for improvements.

**Security Testing**

* Ensure passwords meet strength requirements.
* Test for randomness and unpredictability.
* Verify absence of vulnerabilities.

**5. Deployment Phase**

**Packaging**

* For CLI:
  + Create an executable file or script.
* For GUI:
  + Build the application using a GUI framework.
  + Package as a distributable file (e.g., .exe, .dmg).

**Documentation**

* Write a user manual:
  + Instructions for using the password generator.
  + Troubleshooting guide.
* Create technical documentation:
  + Architecture diagrams.
  + Code explanations and API references.

**Distribution**

* Host the project on a public repository (e.g., GitHub).
* Share the application through download links or app stores.

**6. Maintenance and Future Enhancements**

**Bug Fixes**

* Monitor feedback and fix reported issues.

**Feature Upgrades**

* Add optional features:
  + Integration with password managers.
  + Customizable rules (e.g., avoid specific characters).
  + Multi-language support.

**Code Optimization**

* Refactor code for improved performance and readability.

|  |  |  |
| --- | --- | --- |
| **Level** | **Task** | **Subtasks** |
| **1** | Requirements Analysis | Functional and non-functional requirements. |
| **2** | Design | System design, UI design, and algorithm design. |
| **3** | Implementation | Backend, frontend, and integration development. |
| **4** | Testing and Debugging | Unit, integration, user, and security testing. |
| **5** | Deployment | Packaging, documentation, and distribution. |
| **6** | Maintenance and Future Enhancements | Bug fixes, feature upgrades, and optimizations. |