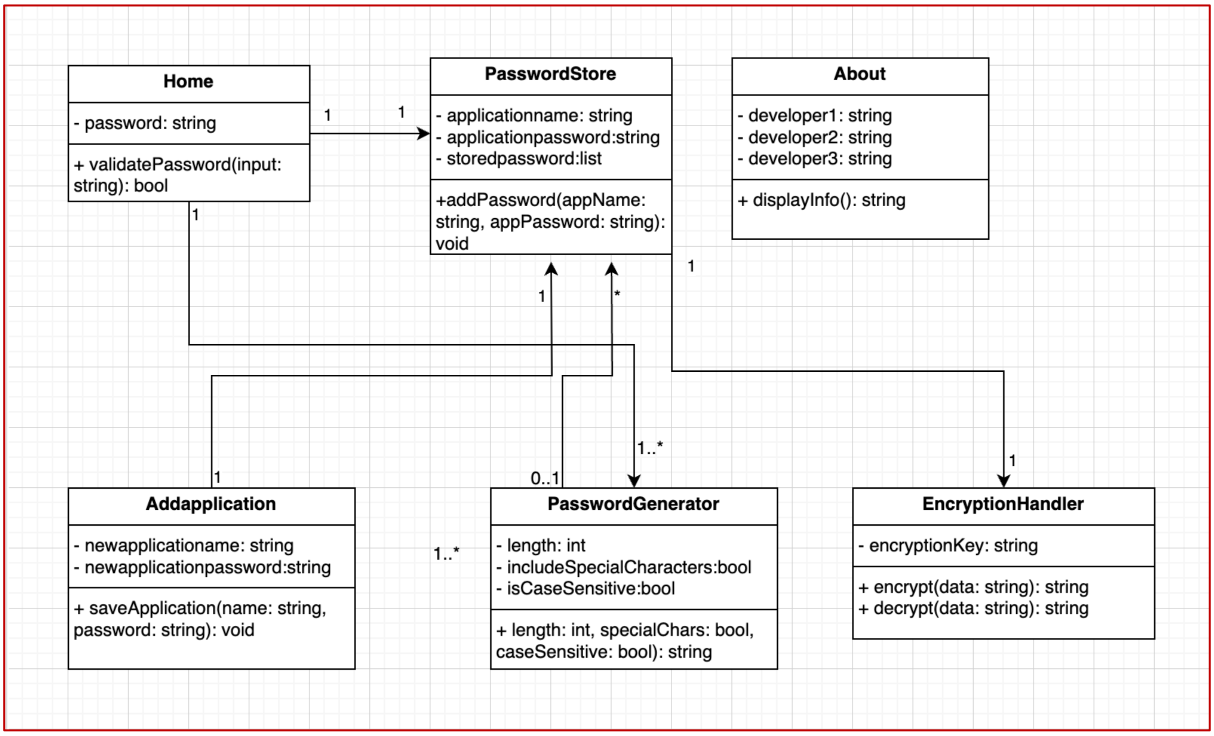
CIS 641 01 - IP - Systems Analysis and Design (F24)

Team Members:

- Likitha Magham
- Swethasarayu Simhadri
- Bhavana Arla

Class Diagram:


Home
- password: string
+ validatePassword(input: string): bool
PasswordStore
- applicationname: string
- applicationpassword:string
- storedpassword:list
+addPassword(appName: string, appPassword: string): void
About
- developer1: string
- developer2: string
- developer3: string
+ displayInfo(): string
Addapplication
- newapplicationame: string
- newapplicationpassword:string
+ saveApplication(name: string, password: string): void
PasswordGenerator
- length: int
- includeSpecialCharacters:bool
- isCaseSensitive:bool
+ length: int, specialChars: bool, caseSensitive: bool): string
EncryptionHandler
- encryptionKey: string
+ encrypt(data: string): string
+ decrypt(data: string): string
1
1
1
1
*
1
1..*
1
0..1
1..*
1

1. **Home**

Description: This class represents the main entry point for the application where users validate their passwords to access the system.

Methods:

- validatePassword(input: string): bool
  - Preconditions: A user must provide a valid password.
  - Postconditions: Returns true if the password matches, granting access to the system.

2. **PasswordStore**

Description: This class is responsible for storing and managing passwords for different applications.

Attributes:

- applicationname: string – The name of the application.
- applicationpassword: string – The password associated with the application.
- storedpassword: list – A collection of all stored application passwords.

Methods:

- addPassword(appName: string, appPassword: string): void
  - Preconditions: Application name and password must be provided.
  - Postconditions: Saves the application's password in the storedpassword list.

3. **Addapplication**

Description: This class handles adding new applications and their associated passwords to the password store.

Attributes:

- newapplicationname: string – The name of the new application to add.
- newapplicationpassword: string – The password for the new application.

Methods:

- saveApplication(name: string, password: string): void
  - Preconditions: The application name and password must be provided.
  - Postconditions: Adds the application details to the password store.

4. **PasswordGenerator**

Description: Responsible for generating secure passwords based on user-defined criteria.

Attributes:

- length: int – The desired length of the generated password.
- includeSpecialCharacters: bool – Whether to include special characters in the password.
- isCaseSensitive: bool – Whether the password should include both uppercase and lowercase letters.

Methods:

- length, specialChars, caseSensitive: bool): string
  - Preconditions: User must specify the required password criteria (length, special characters, case sensitivity).
  - Postconditions: Generates a secure password that matches the given criteria.

5. **EncryptionHandler**

Description: This class provides functionality for encrypting and decrypting sensitive data.

Attributes:

- encryptionKey: string – The key used for encryption and decryption.

Methods:

- encrypt(data: string): string
    - Preconditions: A valid encryption key must be set.
    - Postconditions: Returns the encrypted version of the input data.
- decrypt(data: string): string
    - Preconditions: Encrypted data and a valid encryption key must be provided.
    - Postconditions: Returns the decrypted version of the input data.

6. **About**

Description: This class provides information about the developers of the application.

Attributes:

- developer1: string – Name of the first developer.
- developer2: string – Name of the second developer.
- developer3: string – Name of the third developer.

Methods:

- displayInfo(): string
    - Preconditions: The application must have developer information available.
    - Postconditions: Displays the names of the developers.