PassMan

BU MET CS 673 Summer ‘21 Project (Team 2)

PassMan - The Team

- Francis Xavier Joseph Pulikotil - Team/Requirements leader
- Alexander Dewhirst - Design/Implementation leader
- Zhaowei Gu - Quality Assurance leader
- Andrew Klimentyev - Configuration leader
- Kayla Bayusik - Security leader

PassMan - Introduction

A lot of people use the same passwords on multiple websites. This is a problem because of the many leaks which occur each year. If hackers are able to recover a password for one website, they can simply try it with other websites and thus gain access to multiple accounts of the user.

To reduce such damage, it is advisable to use unique passwords on every website. These unique passwords should also be sufficiently strong to maximize security. A stronger password will be longer, and contain a mix of letters, digits, and special characters to make it more unpredictable.

Even the average person will have tens of accounts with different websites, and it becomes humanly difficult to keep track of so many strong passwords. This is where a password manager is useful - you keep all your strong passwords secure in the password manager application, and the password manager itself is secured with a single strong “master” password. Our project PassMan is a password manager application whose goal is to store passwords securely, while being easy to use.

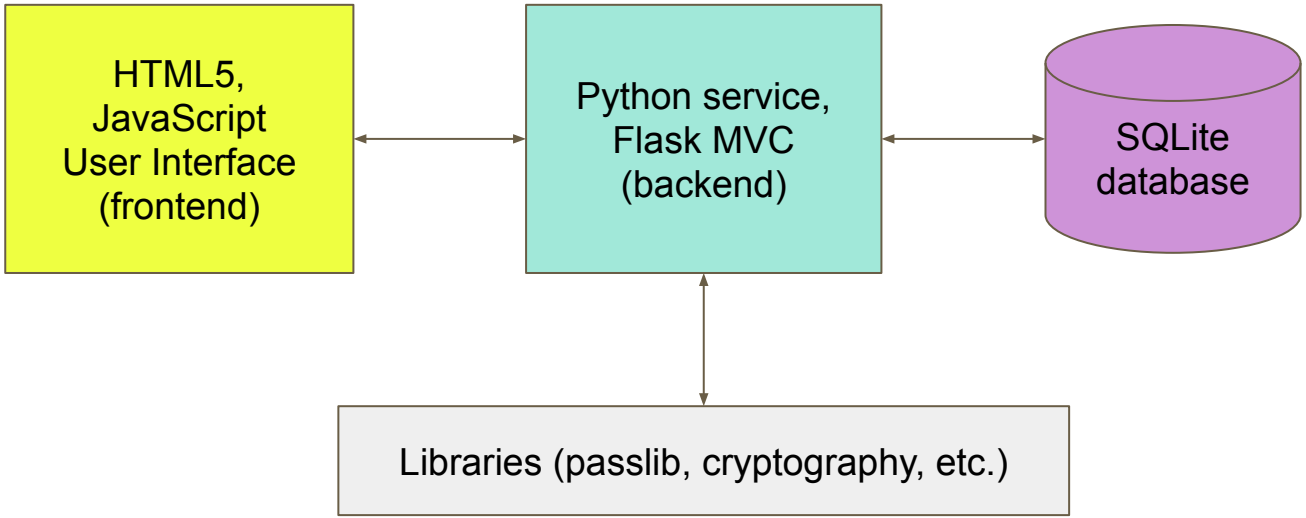
PassMan - Features

- Store passwords securely
  - Encrypt and store passwords in the database.
  - Decrypt them only while showing them to the user.
- Easy to use
  - Essential features are presented in an easy to understand manner.
  - Clean style - prioritize readability and simplicity of the interface.
- Strong password generator
  - Includes a password generator which can be used to generate strong, unique passwords.
  - The generator is configurable to be able to include a mix of letters, digits, and special characters, depending on the user’s requirements.

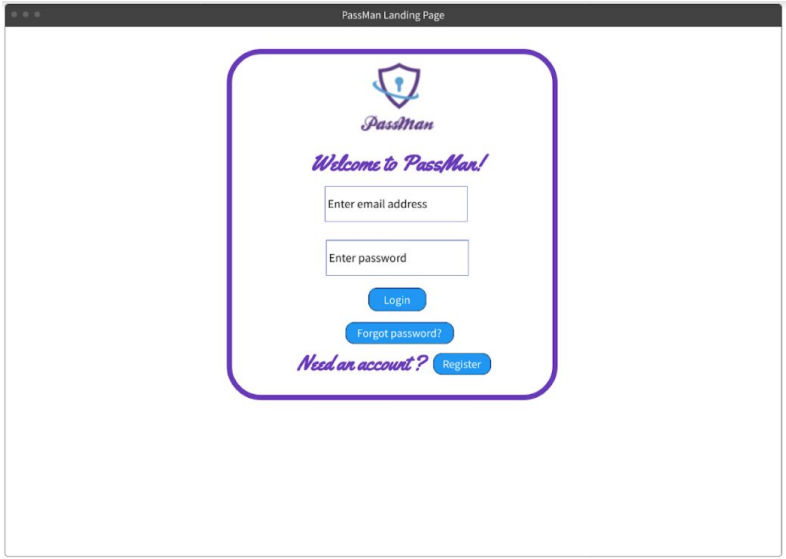
PassMan - Technology

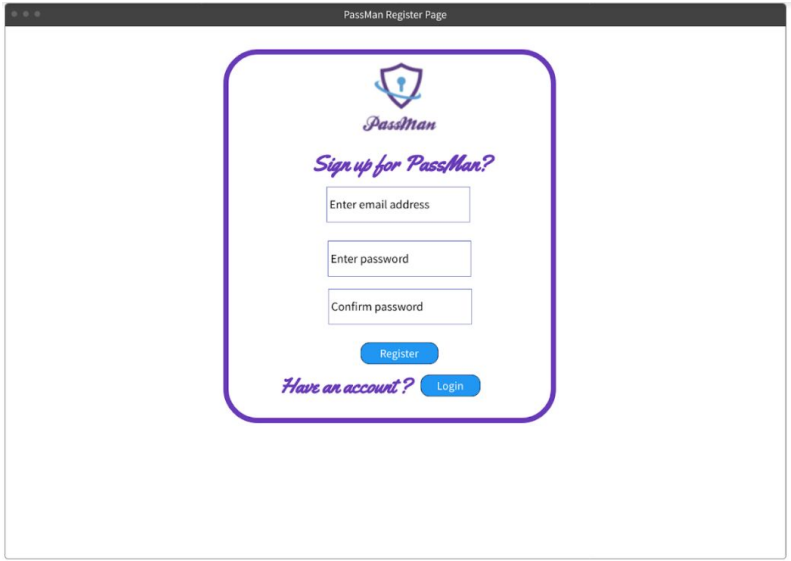
- We used HTML5 and Javascript to develop the PassMan user interface (frontend). We also used CSS for achieving an attractive, clean style.
- The user interface interacts with a Python service which was developed using Flask (backend).
- SQLite is used as the database for persisting data.
- The Python library, *cryptography*, is used for encryption and decryption of sensitive data.
- Use Flask WTF for CSRF protection.
- Python testing framework, *unittest*, is used to test our application.
- We are leveraging *bcrypt* from the *passlib* library to hash user passwords.

PassMan - Architecture

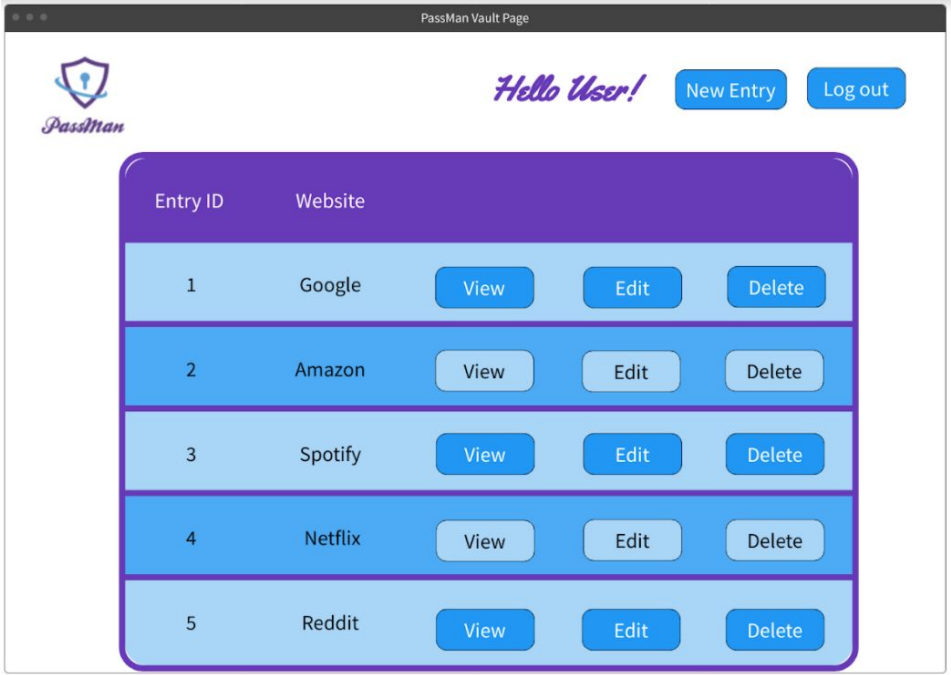


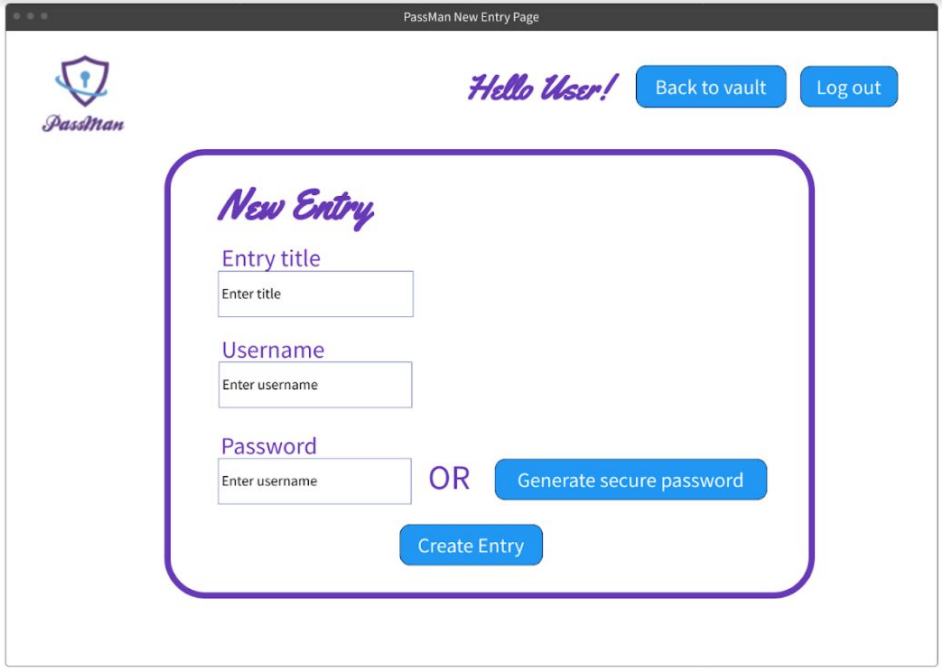
PassMan - Wireframes





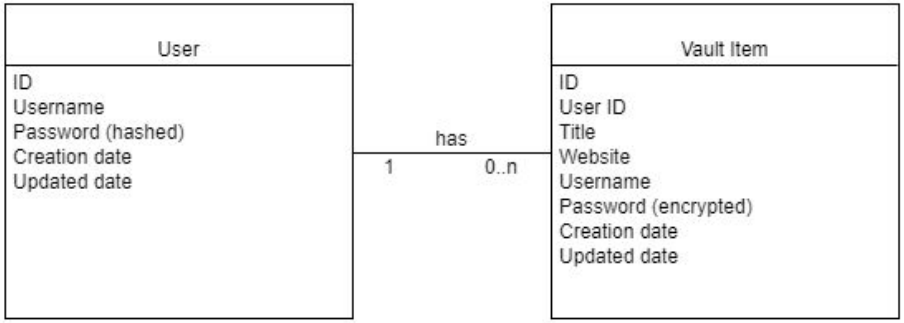
PassMan - Wireframes (cont.)



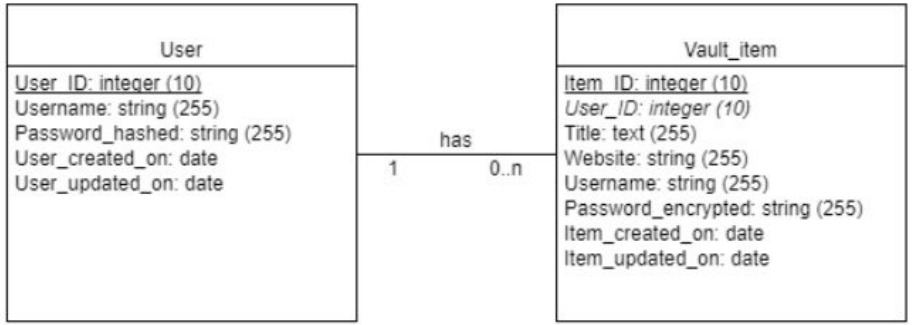


PassMan - Database Models (initial)

- Database Logical Model



- Database Physical Model



PassMan - Database Design

- Database Design for user

```sql
CREATE TABLE user (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   username TEXT NOT NULL COLLATE NOCASE UNIQUE,
   password_hash TEXT NOT NULL,
   salt BLOB NOT NULL,
   created_on TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   updated_on TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

- Database Design for user's Vault

```sql
CREATE TABLE vault (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   user_id INTEGER NOT NULL,
   data BLOB NOT NULL,
   created_on TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   updated_on TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

   FOREIGN KEY (user_id) REFERENCES user (id)
);
```

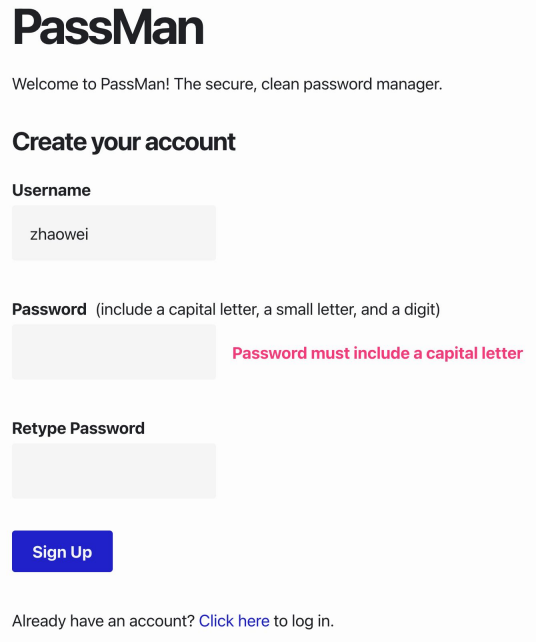
- Database Design for user's Vault Items

```sql
CREATE TABLE vault_item (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   vault_id INTEGER NOT NULL,
   data BLOB NOT NULL,
   created_on TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   updated_on TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

   FOREIGN KEY (vault_id) REFERENCES vault (id)
);
```

PassMan - Login Security

- Password complexity



- Password hashing
- Passlib[bcrypt]

```python
@staticmethod
def create(username, password):
    password_hash = bcrypt.hash(password)
    statement = 'INSERT INTO user(username, password_hash) VALUES (:username, :password_hash) '
    params = {
        'username': username,
        'password_hash': password_hash,
    }
    try:
        with get_db() as db:
            db.execute(statement, params)
    except sqlite3.IntegrityError as ex:
        raise RuntimeError(f'Username {username} is already taken')
```

PassMan - Vault Security

- 16-character salt using *os* library's *urandom*
- *Cryptography* library's *PBKDF2HMAC* generates encryption key from hashing user password and salt
  - With SHA256 algorithm and 100,000 iterations
- Encryption key stored in session cookie
- Key used to encrypt data blob in vault with *Cryptography* library's *Fernet*

PassMan - Testing

Automated Testing

- We have written automated tests using the *flask-unittest* library.
- We also setup *GitHub Actions* to automatically run our automated tests whenever we submit a pull request.
  - This ensures that the code in the main branch is always tested and free of known bugs.
- We added a status badge to our application's README, which shows the current status of the build. PassMan Automated Tests passing

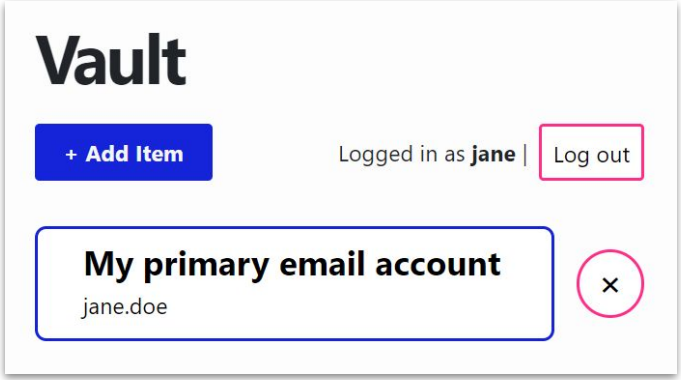
Manual Testing

- Using information from the User Stories in *PivotalTracker*, we created a document with manual tests to be performed: Link to Google Doc
  - We made notes of which tests passed and which failed.

Application Demo

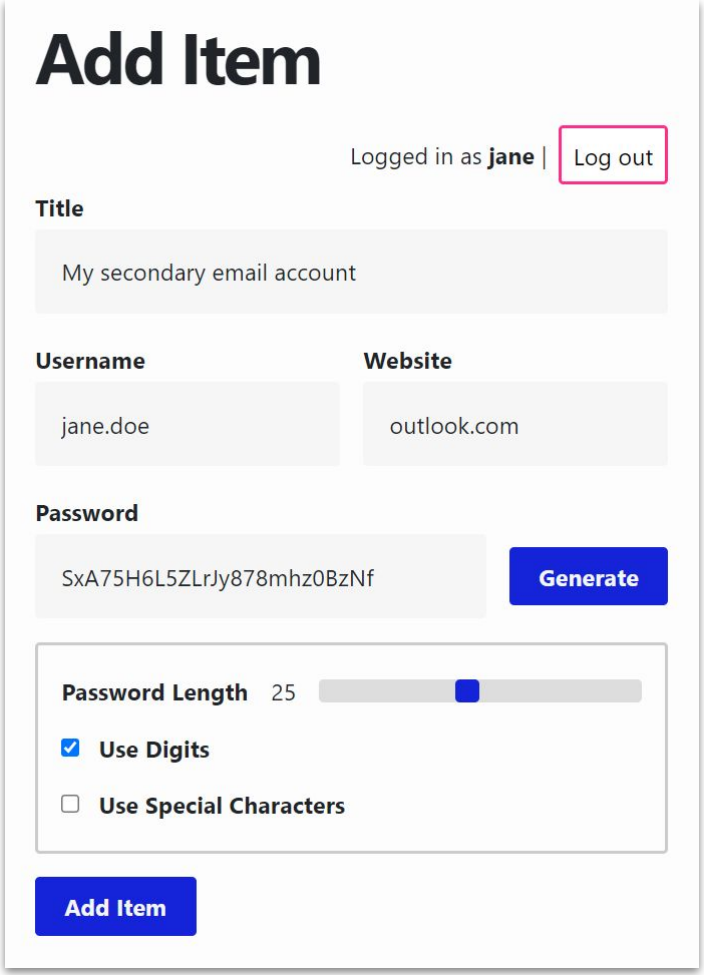
PassMan - Adding Vault Items

- After logging in, the user can add new vault items by clicking the **Add Item** button in their vault.
  - See next slide for details on adding an item.
- To edit an item, simply click on it.
- To delete an item, click the “X” button to the right of that item.



PassMan - Add Item (details)

- Clicking on the **Add Item** button will lead the user to a new page where they can fill in details about that item.
- This same page format is also used when editing an existing vault item.
- To quickly generate a strong password, click the **Generate** button.
- The generated password complexity can be configured using the controls provided (see screenshot on the right).



Code walkthrough

PassMan - Links

- Project documents link: https://drive.google.com/drive/u/4/folders/1gn_6U3sAqs-RFrnpoC6cqk64--MJHEHK
- Project source code link: https://github.com/BUMETCS673/BUMETCS673OLSum21P2