A logo with text on it

Description automatically generated

**National College of Ireland**

**Project Submission Sheet**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name:** | Ansh Ashwini Jain | | |
| **Student ID:** | 23308320 | | |
| **Programme:** | Masters of Science in Cybersecurity | **Year:**2024 |  |
| **Module** | Secure Web Development | | |
| **Lecturer:** | Mr. Rohit Verma | | |
| **Submission Due Date:** | 15th December 2024 | | |
| **Project Title:** | CA2-Simple Bank Application | | |
| **Word Count:** | 2441 | | |

**I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.**

**ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.**

|  |  |
| --- | --- |
| **Signature:** | Ansh Ashwini Jain |
| **Date:** | 15th December 2024 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS:**

1. Please attach a completed copy of this sheet to each project (including multiple copies).

2. Projects should be submitted to your Programme Coordinator.

3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.

4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**

5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

|  |  |
| --- | --- |
| **Office Use Only** | |
| Signature: |  |
| Date: |  |
| Penalty Applied (if applicable): |  |

**AI Acknowledgement Supplement**

**Secure Web Development**

**CA2 – Simple Bank Application**

|  |  |  |
| --- | --- | --- |
| **Your Name/Student Number** | **Course** | **Date** |
| **Ansh Ashwini Jain / 23308320** | Msc in Cybersecurity | 15th December 2024 |

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click [here](https://libguides.ncirl.ie/useofaiinteachingandlearning/studentguide).

**AI Acknowledgment**

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

|  |  |  |
| --- | --- | --- |
| **Tool Name** | **Brief Description** | **Link to tool** |
| **NA** |  |  |
|  |  |  |

**Description of AI Usage**

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used**.

|  |  |
| --- | --- |
| **[Insert Tool Name]** | |
| [Insert Description of use] | |
| [Insert Sample prompt] | [Insert Sample response] |

**Evidence of AI Usage**

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

**Additional Evidence:**

GITHUB Link: <https://github.com/asbhamaan/simpleSecureBankSWD>

YOUTUBE Link: <https://youtu.be/fNoKs3NOljI>

Introduction

The main idea behind this project is to implement a simple bank application. It is a Django based application for managing bank accounts, transactions and users which provides functionalities like deposit, withdrawal, funds transferring and a way to look at the transactions done in the past. The whole goal is also to ensure security and there are various ways in which it is achieved.

The website is originally built on Django web framework. with SQLite and HTML, CSS and Javascript. A user signs up and logs in to the website. And is able to deposit money into their account. Once that is done, they can make a transfer to another user or withdraw that amount. Security implementations earlier included of a basic login, with protection from CSRF token and clickjacking. I contributed with enhancing the login security password functionality, and focused on protecting the website from SSL Redirection, Man in the middle attacks, improved SQL injection, altering the admin path to ensure no external access is allowed to the main application and ensuring safety from XSS attacks. My aim with this website is mainly to enhance security and improve quality of the website without altering the main and simple goal of creating a way to transfer money from one user to other. I have used an existing source code present on GitHub, the link is: <https://github.com/patel-ab/simpleBank>

The project as of now doesn’t have any license of usage and is available on Github for anyone to use.

Security Requirements

For a web-based application there are multiple ways in which it can be called as secured. As security cannot always be 100%, one always tries to achieve it. Following are certain ways in which the website present can be called as a secured website. All of these are restrained to the key point that it’s solely for a bank application.

Authentication

As a user for a bank has their financial stored in a digital site, it is very important to know and only allow that user to gain access to their site. This can be done by verifying the identity of the user using the ways of a username and password combination.

Implementing strong passwords

One of the features I contributed to for this website was the functionality of implementing strong passwords. The original user had put in validators as provided by Django but they didn’t work as the forms created are not Django forms. Implementing strong passwords can really reduce the ways in which a malicious user can gain access to financial and personal information of a user.

Encryption

By default, Django provides PBFK2 algorithm with a SHA256 hash. This is specially designed to ensure that the passwords provided by user are not seen or getting leaked if there is any attack that occurs. An added layer of security which makes much bigger of a difference.

Session Management

Now this application, there are multiple users who will be able to login and transfer money to and fro each other, so session management is one of the greatest requirements. The application is strong enough to provide a secure session for authenticated users to do the actions they want to.

Error Handling

With multiple users able to enter any input they want, there are multiple ways that an error can be occurred, and it needs to be handled in a very certain way so that there are no underlying issues while handling data and responses of the website. With this website, that has been handled and taken care of for most scenarios where an issue can happen.

Form and Input Validation

All of the forms present on the website have input fields where user can type in information which is needed. Validation of these fields and forms is a must for the user to have an experience which makes them feel secured and included and makes the whole of website more responsive and relevant in general.

Testing

I have performed a few positive as well as negative testing for the website which has helped me understand where the website is lacking and needs to improve in, plus also areas where the website has good security implemented.

Design and Architecture

The flow of the application starts from a user required to sign up with their details, then proceed to login to the website. There the user has 2 options, either deposit the money or withdraw. Then there is an option for the user to transfer money to another user present in the bank, and view the transactions done till now.

A diagram of a sign up register with Ice hockey rink in the background

Description automatically generatedFollowing is an example of a use-case with Abuse and Misuse case.

Fig 1. Use Case Diagram for the website

Use-Case: An Actor with all possible functionalities of Sign Up, Login Functionalities and different features available for Deposit and Withdrawing money, then transferring to other user functionality with viewing transactions history.

Abuse Case: A Hacker can gain malicious access and try to modify the records of the users present which can cause financial data to be compromised and displays a weak security of the website.

**A diagram of a server

Description automatically generated**Misuse Case: Any bank official with ill intent, for illegitimate reasons can provide a user with X amount of money or withdraw money from the bank. This is also a display of weak security as user privileges are compromised.

Fig 2. Sequence Diagram for the Login Function

For a specific function of login with password hashing, Following is the Sequence diagram which can be used to explain the function. Main components are Browser, Server, Database and the security element and the relationship between them with relation to the SignUp page.

Implementation

The user uses Django for the backend, SQLite for database and HTML, CSS and Javascript for frontend of the web application. The reason I believe why Django is selected is cause it’s a high-level python web framework which has multiple security features and really helps focus on security completely. With protection from SQL Injections to Cross-site request forgery (CSRF), It has all the accessibilities to security with high protection. The only reason why I believe SQLite is because it’s a lightweight, database engine, which I found suitable for the simple bank application. Django and SQLite together really provided with the most secure foundation which can be found on a web application. For frontend, HTML, CSS and Javascript has been used which makes the web pages more interactive for the user and allows to enhance security by creating more responsive design and highly valid inputs which all works towards making the program secure.

One of the major code change I made was to the password function validation by adding a JavaScript to ensure that the password entered is strong enough and not easily attackable. The user had put validators in the settings file but those were not working as they only work on Django forms and the forms used here are normal HTML forms. Here is a snippet of the function. It is a regex pattern to confirm that the password contains at least one uppercase letter, one lowercase letter, one digit and the length is at least 8 characters strong.

A computer screen shot of a program code

Description automatically generatedFig 3. Code Screenshot

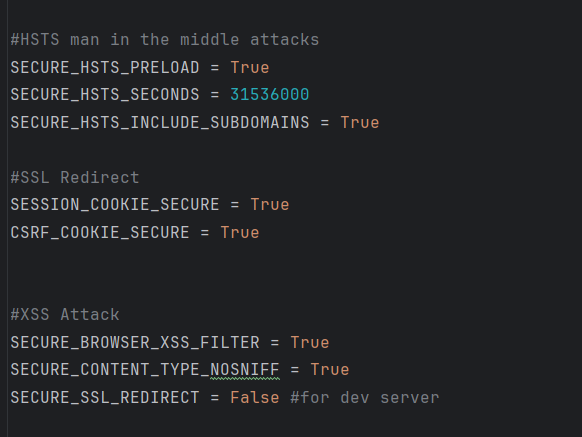
The next code change was to ensure the security of the application is further enhanced by implementing 3 key security features.

Fig. 4. Code Implementation

1) Man in the middle Attacks: HTTP Strict Transport Security (HSTS) is the standard here that is used to prevent man in the middle attacks from occurring. It focuses on creating such settings that doesn’t allow any malicious individual to enter the website. The preload function requests the domain to be added to browser preload lists, enforcing HTTPS right from the start. includeSubDomains ensures all sub domains inherit the HSTS policy. Preload seconds is what sets the duration for which browser enforces the HSTS policy with a minimum of 1 year as set.

2) SSL Redirect: If a browser connects initially via HTTP, which is the default for most browsers, it is possible for existing cookies to be leaked. For this reason, I have set your [SESSION\_COOKIE\_SECURE](https://docs.djangoproject.com/en/5.1/ref/settings/#std-setting-SESSION_COOKIE_SECURE) and [CSRF\_COOKIE\_SECURE](https://docs.djangoproject.com/en/5.1/ref/settings/#std-setting-CSRF_COOKIE_SECURE) settings to True. This instructs the browser to only send these cookies over HTTPS connections.

3) Cross-Site scripting (XSS) Attack: This specific attack occurs when malicious scripts are injected into webpages and executed in the user’s browser. With the features implemented, it can help mitigate most of the features. The NOSNIFF feature helps in avoiding MIME type sniffing [1].

Testing

For testing of the project, I decided to make use of the bandit [2] tool which is specifically designed to run for Python applications to identify common security issues. Bandit tool provides overall issues present on the application. After running the bandit tool on the project, it generated information regarding the severity and confidence levels present for the issues, I was able to perform testing to understand in more detail.

Following is the table for conducting the test on the Transferring Amount Page. Here the user has 3 fields present and transfers money to an existing user, if data is filled incorrectly the transaction should not occur.

**TABLE I**

**TESTING OF MONEY TRANSFER PAGE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Scenario | Test Case | Test Type | Expected Result | Actual Result | Status (Pass/Fail) |
| Transferring Money from one existing user to another | To transfer money with balance available to an existing user | Positive | Able to transfer money with the transaction shown in history | Able to transfer money with the transaction shown in history | Pass |
| To transfer money to another existing user without any balance | Negative | Message ‘Insufficient Balance! Enter details Correctly.’ | Message ‘Insufficient Balance! Enter details Correctly.’ | Pass |
| User input validation for the amount field | Boundary | User should not be able to enter any character at the field. | As Expected. | Pass |
| Transferred Money is displayed as added balance on other user. | Positive | We should see added balance in other user. | We are able to see added balance at the other user. | Pass |

Here one feature that I can implement to make it more ehance is by making the form more unique to the user. Right now the only uniqueness is based on the name and the ID that gets generated at the backend unique for the user. Will have to further work on showing the uniqueness on the front end to make transferring money more fluid and easier.

Conclusion

Due to the time constraints present, I have been able to implement minimal security measures to make the existing project more secure than before. There will be a lot more to work on and dwell more to explore how much more can be done to make a website more secure. My learnings from this project has been to understand the balance between security and functionality implementations and how to ensure both are grown and developed together in hand.

References

[1] <https://www.vskills.in/certification/tutorial/the-theme-of-web-security/>

[2] <https://bandit.readthedocs.io/en/latest/>

Appendices

**TABLE II**

**PLANNED SECURITY REQUIREMENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement ID | Requirement | Status | Percentage of Completion | Pending Implementation |
| 1 | Input validation in All webpages | Completed | 100% |  |
| 2 | Uniqueness of user | In-Progress | 85% | Transfer based on account number instead of unique name. |
| 3 | Strong Passwords | Completed | 100% |  |
| 4 | Increased Authentication | In-Progress | 70% | A way to include OTP’s and security questions while logging in |
| 5 | Password Hashing | In-Progress | 80% | Create a stronger hashing function |
| 6 | SQL Injection | In-Progress | 75% | Need to make it more detailed |
| 7 | Session Management | Completed | 100% |  |
| 8 | XSS Attacks | In-Progress | 75% | Need to make it more detailed |
| 9 | Insecure Design | In-Progress | 90% | To make a clear UI for clear understandability |
| 10 | Security Logging and monitoring | In-Progress | 30% | To create secure log files to ensure data backup |