# MONOPOLY

## A PROJECT REPORT

***Submitted by***

**VAMSHI GOPARI (G01355694)**<br>
**SHREENATH SIVADAS (G01269232)**

***in partial fulfillment for the award of the master's degree***

**CLASS: SWE 681 SECURE SOFTWARE DESIGN AND PROGRAMMING**

**VOLGENAU SCHOOL OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GEORGE MASON UNIVERSITY, FAIRFAX - 22030**

**DATE: DEC 11, 2022**

TABLE OF CONTENTS

## 1. INTRODUCTION

The economics-themed board game Monopoly is designed for numerous players. The goal of the game is to buy, sell, and build houses, hotels, and other real estates by rolling two dice around the game board. Players collect rent from their opponents to bankrupt them. Tax squares, community chest cards, and chance cards are among the several methods to earn or lose money. Players are rewarded for every "Go" they pass.

They could also be locked up and not let out until one of three things happens. Monopoly has become a part of popular culture all over the world. It has local licenses in more than 103 countries and is printed in more than 37 languages. There are house rules, numerous spin-offs, countless editions, and connected media. (1)

Lizzie Magie created the Landlord's Game in the United States in 1903. She did this to show that a system in which individuals are rewarded is better than one in which monopolies control all the money. She also did this to promote Henry George's economic ideas, especially his ideas about taxes. The game's name references the economic notion of "Monopoly," in which a single business controls the market. Initially, there were two sets of rules for The Landlord's Game, one with a tax and one on which the current regulations are primarily based. In the 1935 version of Monopoly, there was no less capitalistic taxation clause, so the game was more competitive. In 1991, Hasbro acquired Parker Brothers. (1)

### a. Setup

Each player begins with a starting bankroll of $3000. Each participant is given a number and a color that are unique to them. Users will get numbers and colors randomly since neither color nor number has any particular significance. The game will begin as soon as all the players have been assigned a color and a number. The game's objective is for everyone to relocate, construct and sell homes, and buy and sell areas in the game world. (1)

### b. Play

Each participant, beginning with the banker, rolls the dice. Then, each player's unique number and color will be placed on the "START" corner, and the dealer will roll the dice and move on to the number of slots shown on the die. The number remains in the occupied spots until the next player's turn. Multiple numbers may occupy the same place simultaneously. Depending on the spot the token reaches, the player may purchase the property, be required to pay rent or taxes, check their luck at the "PASS" field (either get $300 or lose $300), or be sent to jail.

## 2. INSTALLATION INSTRUCTIONS

- This game was created utilizing numerous programming languages and frameworks, including Python, Flask, JavaScript, CSS, HTML, and MySQL.
- Install the requirement.txt file in the project folder using the command `pip install -r requirement.txt` as part of the Prerequisites. This game requires Python 3.8 or a later version to function.
- In the requirement.txt file, the MySQL connector is listed as one of the components that must be present for us to create a connection with the Database and store data in it.
- To establish the DB connection, you need the MySQL server running on the machine. Please see the instructions in the following link to install and run the MySQL server [https://dev.mysql.com/doc/mysql-getting-started/en/](https://dev.mysql.com/doc/mysql-getting-started/en/) .
- Execute the following commands to establish a connection with our application once the MySQL server configuration has been completed.
- To create the schema:

    -
      ```
      CREATE DATABASE `monopoly` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;
      ```
- To create the necessary tables in the database schema, execute the following SQL commands:

    -
      ```
      CREATE TABLE `user` (
      `id` bigint NOT NULL AUTO_INCREMENT,
      `username` varchar (255) DEFAULT NULL,
      `email` varchar (255) DEFAULT NULL,
      `password` varchar (255) DEFAULT NULL,
           PRIMARY KEY (`id`)
      ) ENGINE = InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
      ```

    -
      ```
      CREATE TABLE `game` (
      `id` int NOT NULL AUTO_INCREMENT,
      `code` varchar (45) DEFAULT NULL,
      `user_id` bigint DEFAULT NULL,
      `mode` varchar (45) DEFAULT NULL,
      `status` varchar (45) DEFAULT NULL,
      `isHost` tinyint DEFAULT NULL,
      `is_winner` tinyint DEFAULT '0',
      `is_losser` tinyint DEFAULT '0',
      PRIMARY KEY (`id`),
      KEY `id_idx` (`user_id`),
      ```

```
CONSTRAINT `id` FOREIGN KEY (`user_id`) REFERENCES `user`
(`id`)) ENGINE=InnoDB AUTO_INCREMENT=118 DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

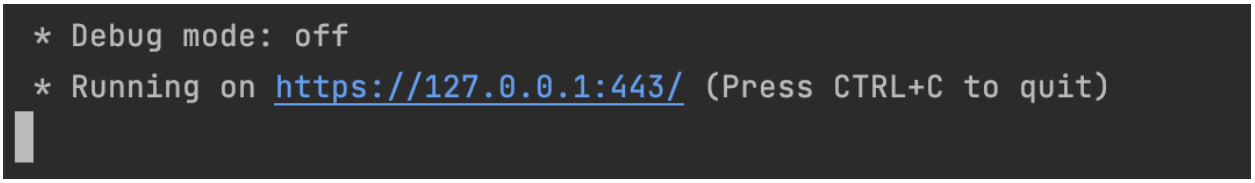
  -
    ```
    CREATE TABLE `game_file` (
    `id` bigint NOT NULL AUTO_INCREMENT,
    `file` blob,
    `updated_at` datetime DEFAULT NULL,
    `code` varchar (125) DEFAULT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `code_UNIQUE` (`code`),
    KEY `code_idx` (`code`)
    ) ENGINE=InnoDB AUTO_INCREMENT=33 DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_0900_ai_ci;
    ```

- We can now configure the application's DB connection and store data in the Database.

## 3. OPERATING INSTRUCTIONS

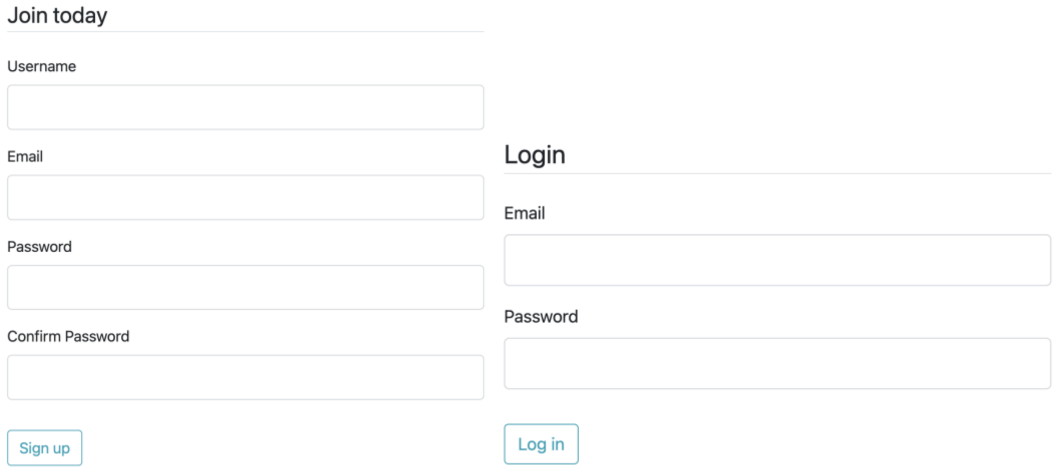
- Once all prerequisites have been met, execute the command below to launch the game.
  - `sudo flask run --cert=cert.pem --key=key.pem --port=443`

- The preceding command will launch the server for the application as detailed below:

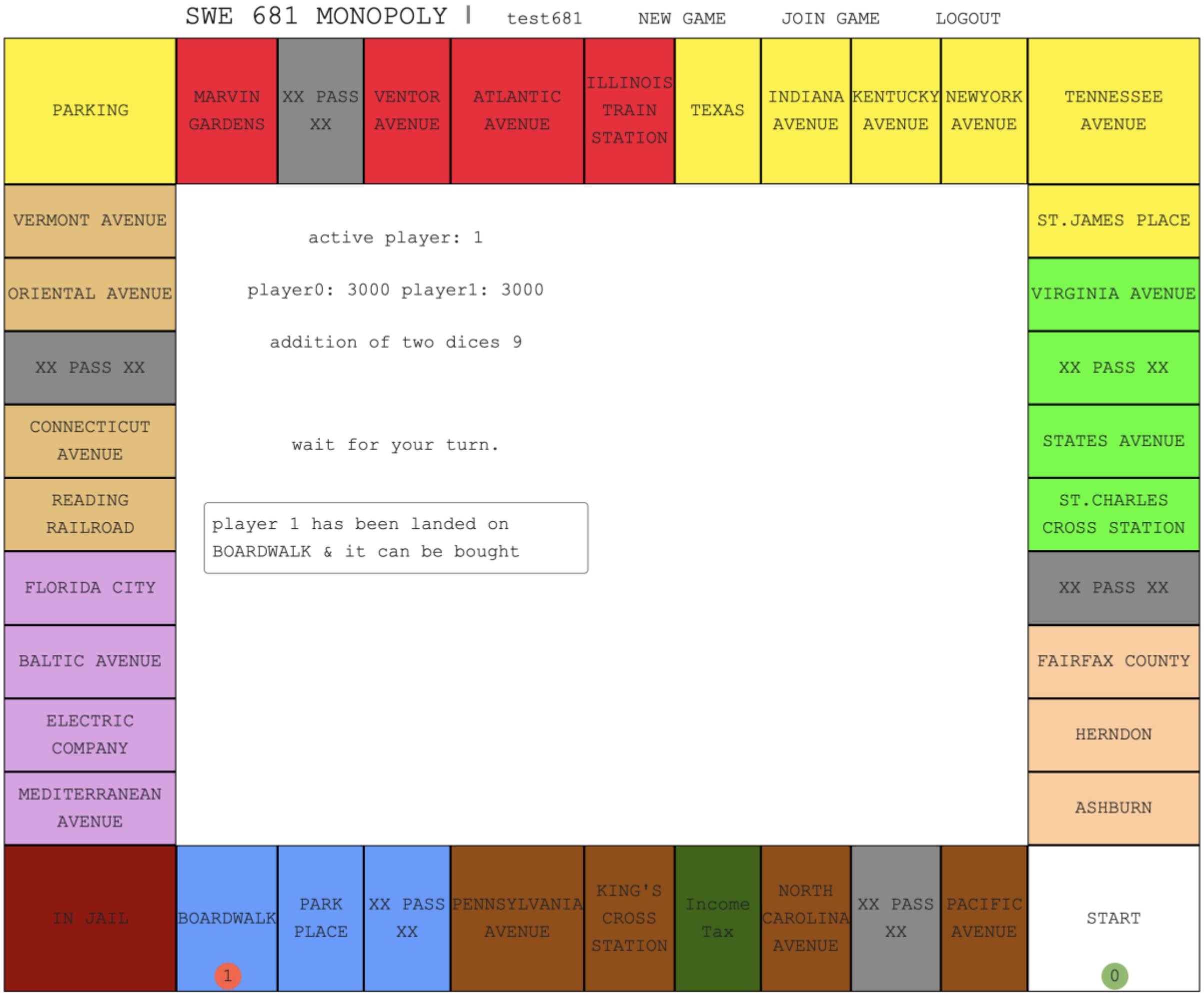


```
* Debug mode: off
* Running on https://127.0.0.1:443/ (Press CTRL+C to quit)
```

- After accessing the URL https://127.0.0.1:443, the homepage will display two options: login or register. After creating an account, we can access it by logging in.



- The user will have the option to either host or join a game after logging in. You can select either of them to be taken to a virtual game board where you can compete against your opponent.



active player: 1

player0: 3000 player1: 3000

addition of two dices 9

wait for your turn.

player 1 has been landed on BOARDWALK & it can be bought

- The player's and opponent's moves are displayed in the middle of the user interface page, where the logs (moves) of both players can be found.
- In order to construct houses and hotels, you must click on the fields you own.
- To sell houses and hotels, you must hold down the shift key and then click on the fields that you own that contain a house or hotel.
- After the game concludes, you can either continue playing or log out of your account.

## 4. DESIGN/ARCHITECTURE OF THE GAME

Python and flask framework were used to develop the game. By setting the ssl_context argument in app, we used the self-signed certificate that was generated by the Openssl command (5) in our Flask app.run() to a tuple with the filenames of the certificate and private key files to run the application on https protocol that uses Transport Layer Security (TLS) that encrypts and secures any communication space and ensures the integrity of the communication. HTML, CSS, and JS were used to develop the application's frontend in order to handle all user events. MySQL is used to store user information such as username, email, password, and game data. The game statistics were stored in the table game, each move was recorded in the pickle files, and the same file was then saved to the database's game file table.

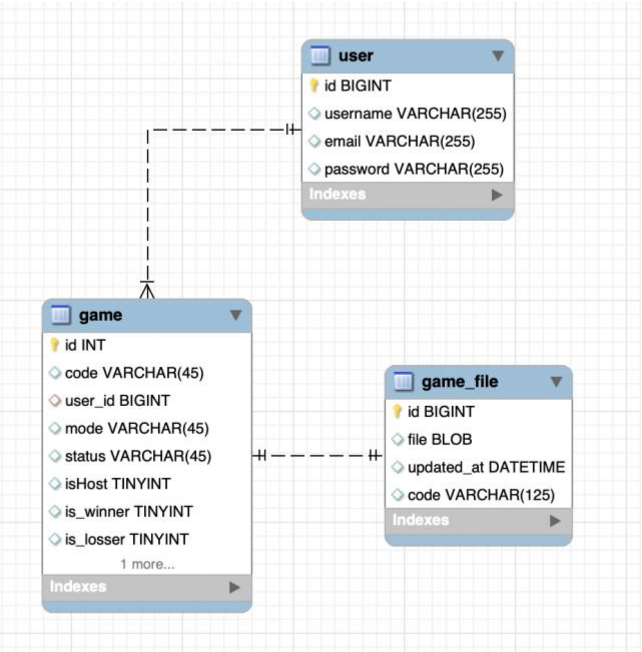


Fig: Database Architecture

## 5. MAJOR COMPONENT OF APPLICATION

### a. Register/Login

These elements permit users to register and log in to the game. Each utilizes the database user table to store and retrieve user information. BCrypt is used to encrypt passwords for added security.

### b. Profile

The incomplete games can be found on this profile page. The pickle file allowed access to the incomplete games. This file will contain every game record and will be stored in the database.

### c. New game

From this portion of the application, the user will be able to host the new game. The game code will be generated so that additional players can join.

**d. Join game**

The website provides a form for the user to enter the code and join the existing game.

**e. Board**

At the beginning of the game, city names, train station names, passes, income tax, jail, and parking fields will be entered into each field. In addition, the assigned amount, current player, and dice value will be displayed in the center of the board. The board will also display the unique id of each player via a background color distinction. On each turn, the board will be updated to reflect the user's actions, such as buying, selling, and the next turn. All events will be recorded in the pickle file and displayed in the board's center.

**f. GitHub link of the code -** https://github.com/swe-monopoly/swe-monopoly

## 6. GAME RULES

Each player's starting bank balance is $3000. The order of the players' turns is determined by chance before the game begins. A typical turn begins by rolling the dice and moving a piece the corresponding number of squares clockwise around the board. When a player lands on or passes the "START" space, the bank gives them $300. Players who land on either Income Tax pay the bank the indicated amount. In earlier editions of the game, the only option for Income Tax was to pay a $300 flat fee.

### a. Jail

When you are sent to jail, you must move immediately into jail, and you cannot collect $200 in salary if you need to pass Go. Additionally, your turn ends when you are placed in jail. If you are not "sent to jail" but land on that space in the normal course of play, you are "Just Visiting" and proceed as normal on your next turn.

### b. Buying Property

When you land on an unclaimed property, you can purchase it from the Bank at the price listed.

### c. Paying Rent

When you land on a property owned by another player, the owner collects rent according to the list on the property's Title Deed card. If the property is mortgaged, the Title Deed card is flipped over and the owner cannot collect rent. The owner may charge double rent for unimproved properties in a color group if he or she possesses all Title Deed cards in that color group. This rule applies to unencumbered properties regardless of whether another property in the same color group is encumbered. It is advantageous to have houses or hotels on a property, as the rents will be considerably higher.

### d. Free Parking

This is a "free" resting spot, so the player who lands here receives no money, property, reward, or penalty.

### e. Bankruptcy

If you owe more than you can pay to another player or the Bank, you are declared bankrupt. If you do not have sufficient funds to pay the other player's rent, you are declared bankrupt and you lose the game.

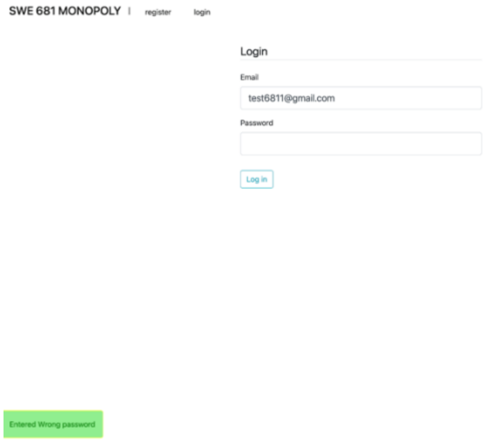
## 7. ASSURANCE CODE

| | | |
|---|---|---|
| 1 | Vulnerability | **CWE-613: Insufficient Session Expiration** |
| | Argument | • Code has been implemented to prevent attackers from utilizing previous session credentials or tokens.<br>• Implemented the expiration date for sessions and tokens.<br>• A token's timeout interval is set to 60 minutes for each session that is started when a user logs in.<br>• The token expires and the user's session is ended if they are continuously inactive for 60 minutes.<br>• After logging out, the user session is invalidated |
| | Justification/Sample code | `@auth.before_request`<br>`def make_session_permanent():`<br>`    session.permanent=True`<br>`    app.config['PERMANENT_SESSION_LIFETIME'] =`<br>`      timedelta(minutes=60)` |
| 2 | Vulnerability | **Broken authentication**<br>• **CWE – 287: Improper Authentication**<br>• **CWE – 306: Missing Authentication for Improper function** |
| | Argument | • Provided centralized authentication at the login page.<br>• Credentials such as password is hashed and then stored which avoids an attacker to decrypt the password.<br>• Regex is used to accept only valid email addresses and passwords containing Small, Big and special characters are only accepted. |
| | Justification/Sample code | `@auth.route('/login', methods=['GET', 'POST'])`<br>`@login_forbidden`<br>`def login():`<br>`   form = LoginForm()`<br>`   if form.validate_on_submit():` |

| | | |
|---|---|---|
| | | `        user =`<br>`User.query.filter_by(email=form.email.data).first(`<br>`)`<br>`        if user and`<br>`bcrypt.check_password_hash(user.password,`<br>`form.password.data):`<br>`            login_user(user, duration=1,`<br>`remember=form.remember.data)`<br>`            session.permanent = True`<br>`            app.permanent_session_lifetime =`<br>`timedelta(minutes=5)`<br>`            next_page = request.args.get('next')`<br>`            return redirect(next_page) if next_page`<br>`else redirect(url_for('game.home',`<br>`user_id=current_user.id))`<br>`        flash('error', 'Something went wrong.')`<br>`    return render_template('auth/login.html',`<br>`form=form)` |
| 3 | Vulnerability | **Using Components with Known Vulnerabilities**<br>**CWE---561: Dead Code** |
| | Argument | Used a static application security testing tool called spectral to identify dead code. Remove dead code before deploying the code. |
| 4 | Vulnerability | • **CWE---862: Missing Authorization**<br>• **CWE---863: Incorrect Authorization** |
| | Argument | • The game will not expose sensitive information to a user who is not explicitly authorized to have access to that information.<br>• Spectators cannot join a game that they do not have the code to.<br>• Users can only see previous game results but cannot modify them.<br>• In login page, when user's enter duplicate names, software performs authorization check and restricts the user from login |
| | Justification/Sample code | If you add @login_required to a view, it will make sure that the current user is logged in and authenticated before calling the view itself. (If they are not, it calls the LoginManager.unauthorized callback.) For example:<br>`@app.route('/post')`<br>`@login_required`<br>`def post():`<br>`    pass`<br>The other way to check if the user is not authorized is shown below: |

<table>
<tr><td></td><td></td><td><pre>
if not current_user.is_authenticated:
    return current_app.login_manager.unauthorized()
</pre></td></tr>
<tr><td>5</td><td>Vulnerability</td><td><strong>CWE – 209 : Generation of Error Message Containing Sensitive Information</strong></td></tr>
<tr><td></td><td>Argument</td><td>

- The amount of error information returned to the user is limited
- The sensitive information may be valuable information on its own (such as a password), or it may be useful for launching other, more serious attacks. The error message may be created in different ways:



SWE 681 MONOPOLY | register login

Login

Email

test6811@gmail.com

Password

Log in

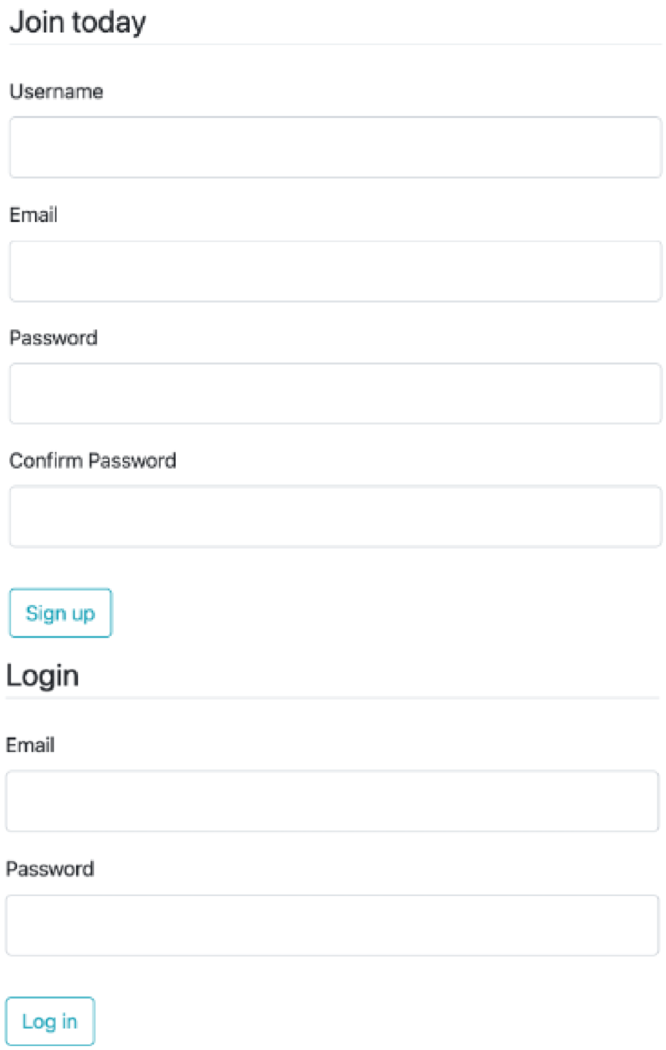
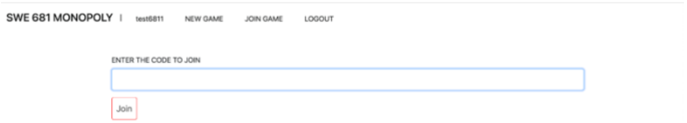
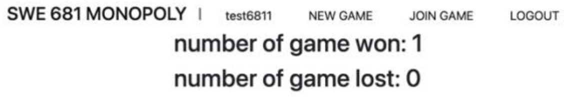
Entered Wrong password

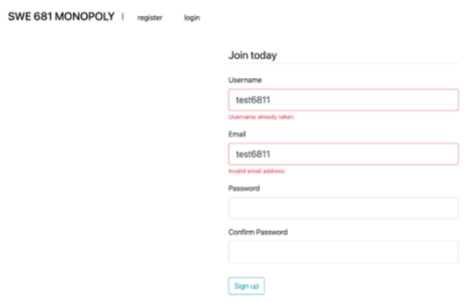
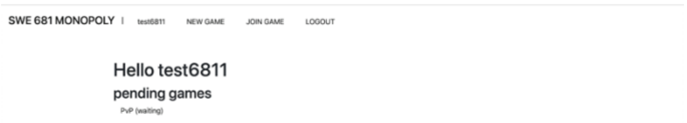
</td></tr>
<tr><td></td><td>Justification/Sample code</td><td><pre>
@auth.route('/login', methods=['GET', 'POST'])
@login_forbidden
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user =
User.query.filter_by(email=form.email.data).first(
)
        if user and
bcrypt.check_password_hash(user.password,
form.password.data):
            login_user(user, duration=1,
remember=form.remember.data)
            session.permanent = True
            app.permanent_session_lifetime =
timedelta(minutes=5)
            next_page = request.args.get('next')
</pre></td></tr>
</table>

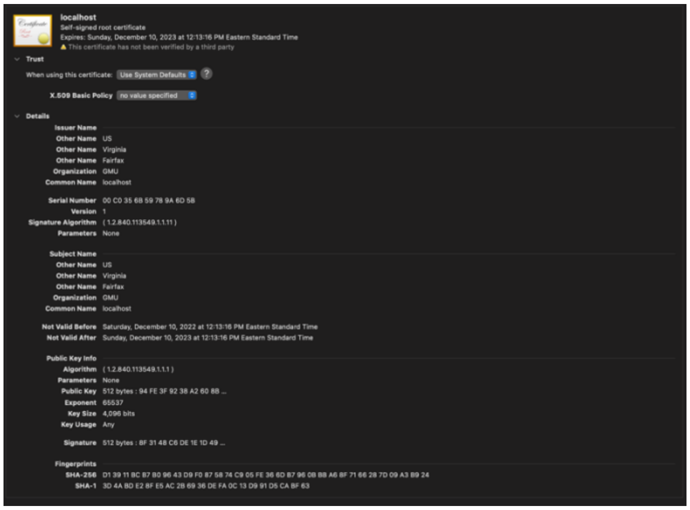
| | | |
|---|---|---|
| | | `        return redirect(next_page) if next_page else redirect(url_for('game.home', user_id=current_user.id))`<br>`      flash('Entered Wrong password', 'Something went wrong.')`<br>`   return render_template('auth/login.html', form=form)` |
| 6 | | **CWE-319: Cleartext Transmission of Sensitive Information** |
| | Argument | • The software transfers sensitive or security-critical data across a secure communication channel that is inaccessible to unauthorized parties.<br>• Many communication channels can be "sniffed" by attackers during data transmission. This significantly lowers the difficulty of exploitation by attackers. |
| | Justification | • Used the HTTPS protocol by default the application traffic.<br>• Did not include sensitive information in log messages or files.<br>• Encrypted data transmitted on internal networks. |
| 7 | Vulnerability | • **CWE – 120 Buffer Copy Without Checking Size of Input (‘Classic Buffer Overflow’)**<br>• **CWE – 131 Incorrect Calculation of Buffer Size** |
| | Justification | • Buffer overflows are not susceptible in Python.<br>• The program verifies the size of input buffer and restricts the user to provide large inputs which leads to buffer overflow. |

### a. Basic Requirements

| | | |
|---|---|---|
| 1 | Requirement | When potential users connect to the game system server, they must be able to either login with their username or password or create a new account. |

| | | |
|---|---|---|
| | Justification | <br>Join today<br>Username<br>Email<br>Password<br>Confirm Password<br>Sign up<br>Login<br>Email<br>Password<br>Log in |
| 2 | Requirement | Once a user logs in, they must be able to: Join an existing game that needs players. |
| | Justification | <br>SWE 681 MONOPOLY \| test6811 NEW GAME JOIN GAME LOGOUT<br>ENTER THE CODE TO JOIN<br>Join |
| 3 | Requirement | Once user logs in, they must be able to see Win/Loss game statistics of themselves |
| | Justification | <br>SWE 681 MONOPOLY \| test6811 NEW GAME JOIN GAME LOGOUT<br>number of game won: 1<br>number of game lost: 0 |

| | | |
|---|---|---|
| 4 | Requirement | Must not let someone create an account that already exists |
| | Justification | <br>SWE 681 MONOPOLY \| register login<br>Join today<br>Username<br>test6811<br>Username already taken.<br>Email<br>test6811<br>Invalid email address.<br>Password<br>Confirm Password<br>Sign up |
| 5 | Requirement | Must store passwords using some salted hash system; you must not store passwords in the clear or as bare hashes. |
| | Justification | Passwords are encrypted using BCrypt hashing function and then stored in the database. |
| 6 | Requirement | Once a player has joined a game, they must be able to re-log in and rejoin |
| | Justification | <br>SWE 681 MONOPOLY \| test6811 NEW GAME JOIN GAME LOGOUT<br>Hello test6811<br>pending games<br>PvP (waiting)<br>By clicking on the game name from the user tab the user can access a game if he is logged out unexpectedly |
| 7 | Requirement | If a player leaves, there must be a timeout where eventually they will forfeit |
| | Justification | If the player leaves unexpectedly and the user doesn't join the game for two hours then the opposite player will automatically be the winner. |
| 8 | Requirement | The program must reject invalid inputs that are syntactically correct but are semantically incorrect because of the current state of the game play. |
| | Justification | • The dice will not function for a player, until the other player's turn is completed.<br>• If a finished player tries to make a move, code will reject it.<br>• Only the active player can build house/sell. |
| 9 | Requirement | implementation of a TLS/SSL certificate. |
| | Justification | • Implemented TLS certificate for secure communication run the application in local machine. |

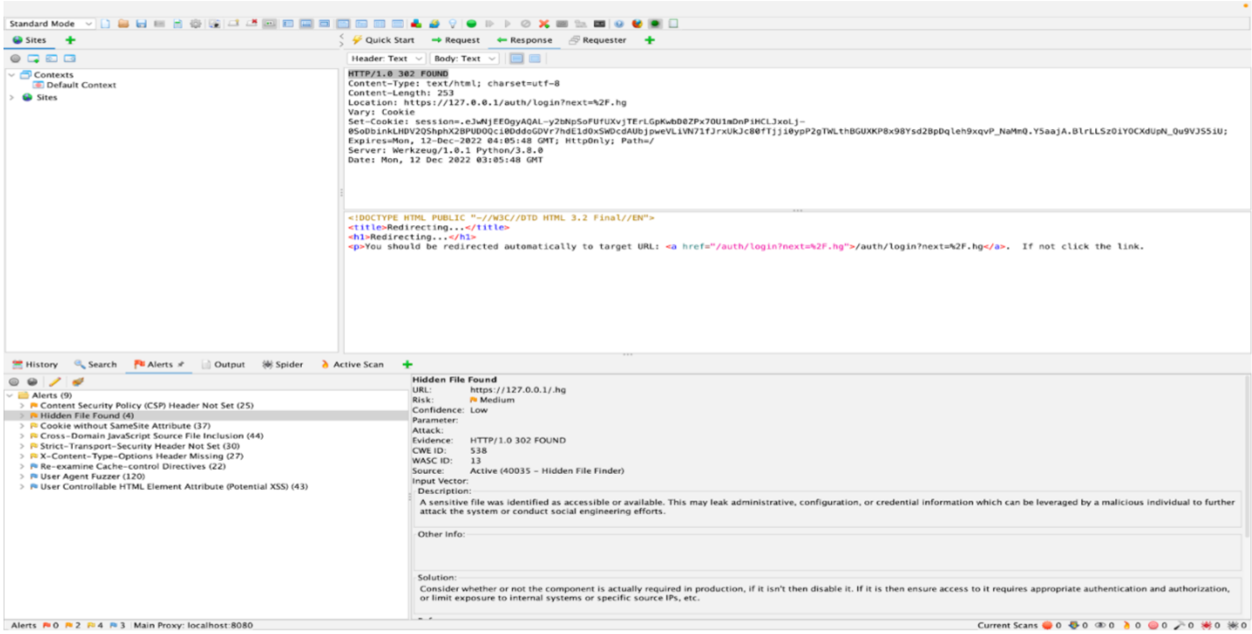
| | | |
|---|---|---|
| | | <br>localhost<br>Self-signed root certificate<br>Expires: Sunday, December 10, 2023 at 12:13:16 PM Eastern Standard Time<br>⚠ This certificate has not been verified by a third party<br>Trust<br>When using this certificate: Use System Defaults<br>X.509 Basic Policy no value specified<br>Details<br>Issuer Name<br>Other Name US<br>Other Name Virginia<br>Other Name Fairfax<br>Organization GMU<br>Common Name localhost<br>Serial Number 00 C0 35 6B 59 78 9A 6D 5B<br>Version 1<br>Signature Algorithm ( 1.2.840.113549.1.1.11 )<br>Parameters None<br>Subject Name<br>Other Name US<br>Other Name Virginia<br>Other Name Fairfax<br>Organization GMU<br>Common Name localhost<br>Not Valid Before Saturday, December 10, 2022 at 12:13:16 PM Eastern Standard Time<br>Not Valid After Sunday, December 10, 2023 at 12:13:16 PM Eastern Standard Time<br>Public Key Info<br>Algorithm ( 1.2.840.113549.1.1.1 )<br>Parameters None<br>Public Key 512 bytes : 94 FE 3F 92 38 A2 60 8B ...<br>Exponent 65537<br>Key Size 4,096 bits<br>Key Usage Any<br>Signature 512 bytes : 8F 31 48 C6 DE 1E 1D 49 ...<br>Fingerprints<br>SHA-256 D1 39 11 BC B7 B0 96 43 D9 F0 87 58 74 C9 05 FE 36 6D B7 96 0B BB A6 8F 71 66 28 7D 09 A3 B9 24<br>SHA-1 3D 4A BD E2 8F E5 AC 2B 69 36 DE FA 0C 13 D9 91 D5 CA BF 63 |

## b. Security Requirements (CIA)

| | | |
|---|---|---|
| 1 | Confidentiality | Must maintain records of every game move and it should be available only to the authenticated users. The game moves should only be displayed after the game is completed. Authenticated users must have access to each user's win-loss report. |
| | Justification | • Game moves are visible to authenticated users.<br>• Win/loss statistics of previous games are available in the authenticated player’s dashboard. |
| 2 | Integrity | In every game, only the current player can make a move, and it must be legal. Win-loss results must not be changed by any player. |
| | Justification | • Upon a player's turn, he/she can be able to make their move.<br>• Win/loss results will be updated automatically once a game completes; it cannot be modified by any player |
| 3 | Availability | A player must not be able to leave the game forever. A timeout should be enabled to eventually forfeit unresponsive players. If a player gets logged out due to wireless failure they must be able to reconnect to the game and continue to play |
| | Justification | • Game is designed in a way that will forfeit the users out of the game if they are idle for more than 60 minutes |

### c. Dynamic Analysis Tool – OWASP ZAP

After performing Static Code Analysis using OWASP ZAP, we identified 2 medium and 4 low level alerts related to Hidden file disclosure and some false positive errors related to X-Content -Type-Options.



```
HTTP/1.0 302 FOUND
Content-Type: text/html; charset=utf-8
Content-Length: 253
Location: https://127.0.0.1/auth/login?next=%2F.hg
Vary: Cookie
Set-Cookie: session=.eJwNjEEOgyAQAL-y2bNpSoFUfUXvjTErLGpKwbD0ZPx7OU1mDnPiHCLJxoLj-
0SoDbinkLHDV2QShphX2BPUDOQci0DddoGDVr7hdE1d0xSWDcdAUbjpweVLiVN71fJrxUkJc80fTjji0ypP2gTWLthBGUXKP8x98Ysd2BpDqleh9xqvP_NaMmQ.Y5aajA.BlrLLSz0iYOCXdUpN_Qu9VJS5iU;
Expires=Mon, 12-Dec-2022 04:05:48 GMT; HttpOnly; Path=/
Server: Werkzeug/1.0.1 Python/3.8.0
Date: Mon, 12 Dec 2022 03:05:48 GMT
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to target URL: <a href="/auth/login?next=%2F.hg">/auth/login?next=%2F.hg</a>.  If not click the link.
```

- Alerts (9)
  - Content Security Policy (CSP) Header Not Set (25)
  - Hidden File Found (4)
  - Cookie without SameSite Attribute (37)
  - Cross-Domain JavaScript Source File Inclusion (44)
  - Strict-Transport-Security Header Not Set (30)
  - X-Content-Type-Options Header Missing (27)
  - Re-examine Cache-control Directives (22)
  - User Agent Fuzzer (120)
  - User Controllable HTML Element Attribute (Potential XSS) (43)

Hidden File Found

| | |
|---|---|
| URL: | https://127.0.0.1/.hg |
| Risk: | Medium |
| Confidence: | Low |
| Parameter: | |
| Attack: | |
| Evidence: | HTTP/1.0 302 FOUND |
| CWE ID: | 538 |
| WASC ID: | 13 |
| Source: | Active (40035 – Hidden File Finder) |
| Input Vector: | |

Description:
A sensitive file was identified as accessible or available. This may leak administrative, configuration, or credential information which can be leveraged by a malicious individual to further attack the system or conduct social engineering efforts.

Other Info:

Solution:
Consider whether or not the component is actually required in production, if it isn't then disable it. If it is then ensure access to it requires appropriate authentication and authorization, or limit exposure to internal systems or specific source IPs, etc.

Alerts 0 2 4 3 Main Proxy: localhost:8080

Current Scans 0 0 0 0 0 0 0 0

## 8. CONCULUSION

We have created a working version of Monopoly that multiple users can play simultaneously. We accounted for the security flaws that were discussed in class. Using the static analysis tool "OWASP-ZAP" to examine the application. We eliminated a number of vulnerabilities and reran the application after removing them. We also implemented SSL certification, which protects sensitive data and secures the connection. The application has authentication and authorization checks at every level, ensuring that only authorized users can access data and preventing the display of any sensitive information. All forms (such as user ID and password) are equipped with regex validators to eliminate invalid credentials.

## 9. REFERENCES

1. https://en.wikipedia.org/wiki/Monopoly_(game)
2. https://www.ultraboardgames.com/monopoly/game-rules.php
3. https://www.zaproxy.org/getting-started/
4. https://dev.mysql.com/doc/mysql-getting-started/en/
5. https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https