Capstone Final Submission

PixelCola

Alex Reichel, Josh Davis, Jon Burstein, Jacob Miller

Capstone Project Fall, 2023

The University of West Florida

11/25/2023

Dr. Bernd Owsnicki-Klewe

Abstract

In the evolving landscape of cybersecurity, education and awareness are paramount. "PixelCola" emerges as an innovative online learning platform designed to enhance cybersecurity knowledge through interactive gaming. Developed using Python and JavaScript, this platform introduces a unique approach to learning by integrating educational content into engaging gameplay.

The journey begins in Pensacola, Florida, where users, assuming the role of a cybersecurity analyst, navigate through various incidents across the state. The platform features a user-friendly login interface, securely storing user data in a database, ensuring a personalized and safe experience.

PixelCola comprises three distinct games, each tailored to teach specific cybersecurity concepts. The first game, located in Pensacola, is a word scramble that educates players on essential cybersecurity terms and acronyms. As the journey progresses to Tampa, the platform offers a cybersecurity-themed hangman game, further broadening the user's vocabulary with various terms and attacks. Finally, in Miami, users engage in "Portris," a Tetris-like game where matching protocols to their corresponding ports is the key challenge. This innovative approach not only makes learning about network protocols and ports engaging but also enhances memory retention.

By situating the learning experience in a virtual representation of Florida, PixelCola provides a contextual and narrative-driven approach to cybersecurity education. This project not only serves as a testament to the effective use of gaming in education but also as a valuable resource for individuals seeking to enhance their cybersecurity knowledge in an enjoyable and interactive manner.

Table of Contents

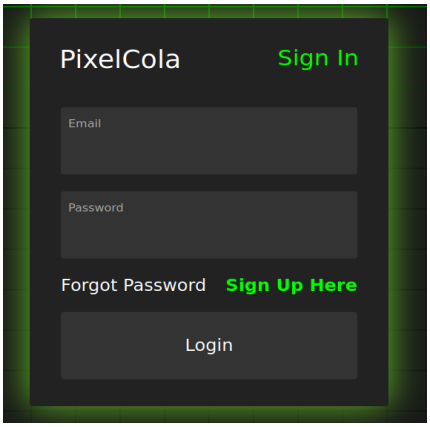
Table of Contents

1 Project description - Ideas, Concepts and Functions

PixelCola is a gamified cyber security learning platform. You play the role of a cyber security analyst responding to incidents across the state starting in Pensacola. “Responding to the events” is playing the mini games that teach different security concepts. In Pensacola you solve word scrambles teaching different cyber terms. In Tampa you play a hangman game that teaches various types of attacks. In Miami you play Portris, a tetris like game where you drop protocols onto the correct ports.

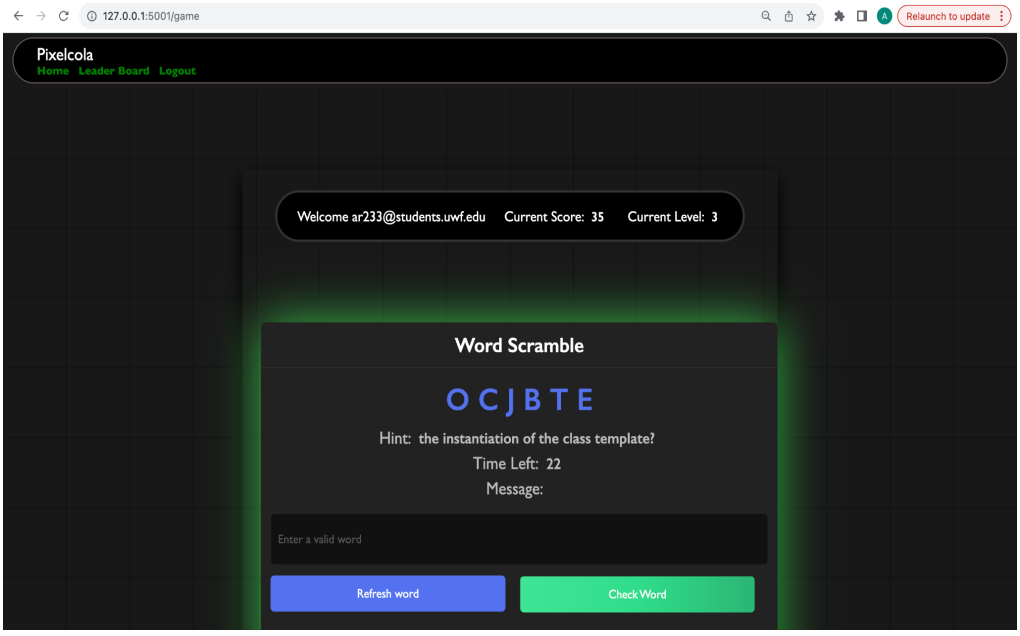
1.1 Log In Screen

The player is greeted with a login screen which is created in auth.py. This part of the project is written in python. This Python code is a Flask-based web application, specifically focusing on user authentication and interactive features. It defines a blueprint, auth which organizes functionalities like user login, registration, logout, score updating, and game access. Key features include handling user login with password verification, registering new users with password hashing, and securely managing user sessions. Additionally, it allows logged-in users to update their scores, and access different gaming stages, each rendered with distinct templates. A leaderboard route is also included, displaying user scores in a competitive format.



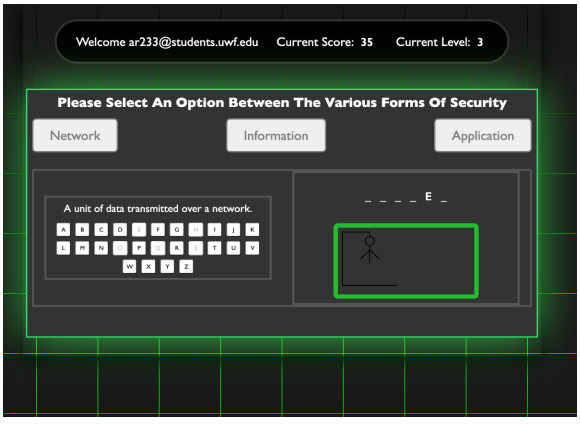
1.2 Game 1 - Word Scramble

The code for game 1, written in javascript, dynamically manages game elements like scrambled words, hints, and user inputs within a web interface. The game is structured around a set of cybersecurity-related words and hints, randomly chosen and scrambled for the player to guess. Key features include a countdown timer that adds urgency to the guessing game, and a scoring mechanism that updates based on the player's success in correctly identifying words. If the player guesses correctly, a congratulatory message is displayed, and the score is updated via an AJAX request, reflecting real-time interaction with the server. Additionally, the game includes level progression and animations, enhancing user engagement. This script effectively combines educational content with interactive gameplay, making learning about cybersecurity both fun and engaging. The seamless integration of front-end dynamics with backend data processing highlights its sophistication as an educational tool within a web application.



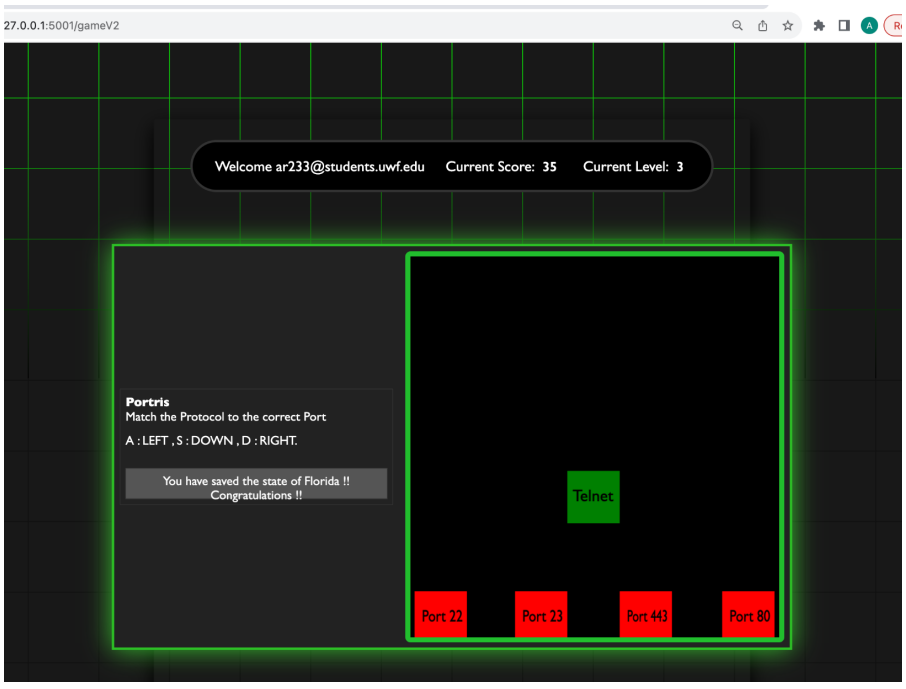
1.3 Game 2 - Hangman

Game 2 is a hangman-style game, focused on educating players about cybersecurity concepts. It integrates a wide range of HTML elements, including sections for hints, letters, user input, and options, as well as a canvas for visual feedback. Central to the game is an options object, which categorizes words under themes like Network, Information, and Application, each accompanied by a hint and an image. The game kicks off with the initializer function, which sets up the user interface, populating it with alphabet buttons for guessing and category options for word selection. When a category is chosen, the generateWord function randomly picks a word from the selected category, presenting it as a series of dashes along with a relevant hint and image. User interaction revolves around guessing the letters of the word; correct guesses reveal the letter, while incorrect ones progressively draw the hangman on the canvas. This drawing aspect is handled by the canvasCreator and its related functions, which intricately render the hangman based on the user's incorrect guesses. The game keeps track of the player's successes and failures, celebrating victories with a win message or marking defeats with a loss announcement in a modal. Additionally, the script features an animation element for visual appeal and an AJAX request to update the player's score and level, indicating a connection to a backend server. Together, these components create a captivating learning experience, blending the classic hangman challenge with the intricacies of cybersecurity, thereby transforming education into an enjoyable activity.



1.4 Game 3 - Portris

Portris is a Tetris-like educational game, focusing on teaching network protocols and their corresponding ports through interactive gameplay on an HTML5 Canvas. In the game, questions about various network protocols are represented as blocks, which the player maneuvers using keyboard inputs to align with the correct answer blocks at the bottom of the canvas. The game mechanics include collision detection functions that handle interactions between the question blocks and the canvas boundaries or answer blocks. Scoring is managed through the QandACheck function, which verifies if the question block correctly aligns with its answer, triggering an AJAX request to update the player's score and level on the server. The game flow is controlled through functions that start the game, progress to the next question, and manage end-game scenarios. Visual feedback is a key aspect, with the canvas being dynamically redrawn in response to player actions and game events, creating an engaging visual experience. Overall, the script effectively merges educational content with the compelling gameplay of a block-matching game, transforming learning about computer networking into an enjoyable and interactive experience.



2 Timeline

Our original timeline consisted of 6 sprints that lasted two weeks each. The first three were spent on each individual game, and the last three were spent improving not only the games but the website and presentation.

Sprint	Week	Deliverable
	09/11/2023- 09/17/2023	Presentation 1 - Project Plan
1	09/18/2023-09/24/2023	Story and Game Planning
1	09/25/2023-10/01/2023	Mini Game 1
2	10/02/2023-10/10/08/2023	Technical Document
2	10/09/2023-10/15/2023	Mini Game 02
3	10/16/2023-10/22/2023	Mini Game 03 Updating story Documentation
3	10/23/2023-10/29/2023	Presentation 2
4	10/30/2023-11/05/2023	Logic and Design
4	11/06/2023-11/12/2023	Review/Repurposing
5	11/13/2023-11/19/2023	Cohesiveness between Games
5	11/20/2023-11/26/2023	
6	11/27/2023-12/03/2023	Final Presentation and Deliverables
	12/04/2023-12/09/2023	

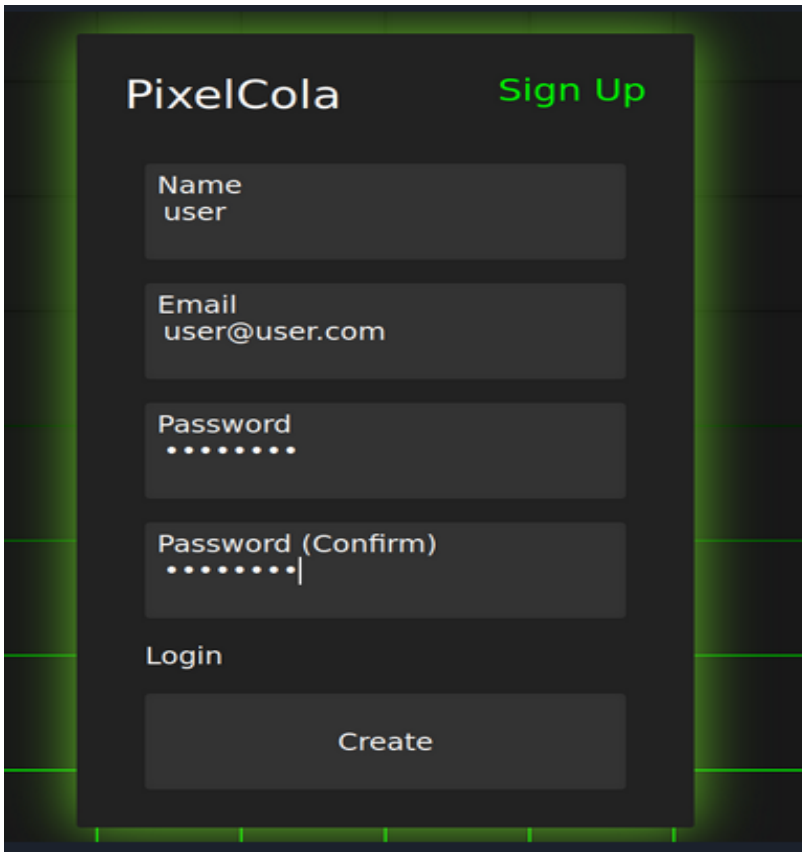
In actuality we stuck to the timeline almost verbatim. There were very few deviations and parts of the software were all delivered on time. Anytime we had issues with development we evaluated our goals again and spent time later on adding functionality and fixing bugs.

Sprint	Week	Deliverable
	09/11/2023- 09/17/2023	Presentation 1 - Project Plan
1	09/18/2023-09/24/2023	Story and Game Planning
1	09/25/2023-10/01/2023	Mini Game 1
2	10/02/2023-10/10/08/2023	Technical Document
2	10/09/2023-10/15/2023	Mini Game 2
3	10/16/2023-10/22/2023	Mini Game 2 and 3
3	10/23/2023-10/29/2023	Presentation 2
4	10/30/2023-11/05/2023	Updating story Documentation
4	11/06/2023-11/12/2023	Logic and Design
5	11/13/2023-11/19/2023	Review/Repurposing Adding variety of content to Games
5	11/20/2023-11/26/2023	Bug Fixing Cohesiveness between Games
6	11/27/2023-12/03/2023	Final Presentation and Deliverables
	12/04/2023-12/09/2023	

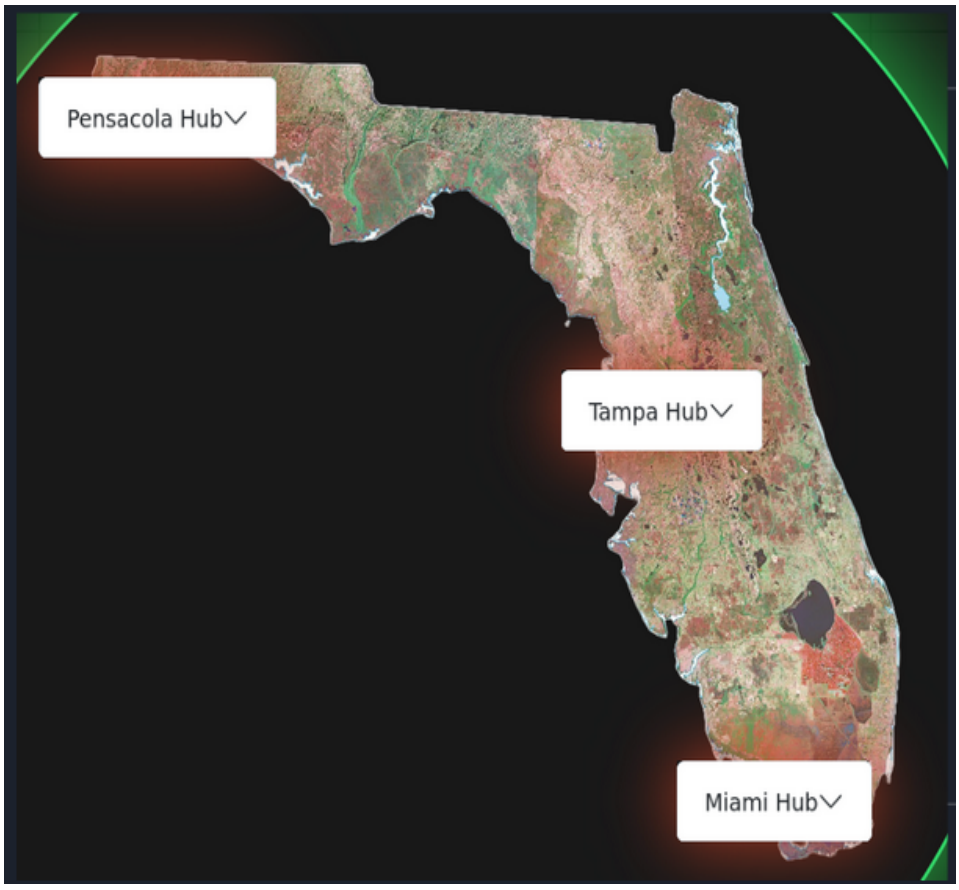
3 Project Results Compared with Expectations

Below are the most important use cases that were created when the project was at the idea phase and below that use case is the final product that the use case eventually became. We implemented these use cases according to our technical documentation. Design changes were made when needed.

Use Case	
Actor	user
Preconditions	The user has accessed the application
Basic Flow	1. The user has gone to the website. 2. The user has clicked new user 3. The user enters their information. 4. The information is stored in the database.
Alternate Flow	If the user already has an account they can skip this.
Exceptions	If the user enters information that is already in the database or a password that does not meet security requirements they will receive an error
Post conditions	The user's account is created and they can log in using these credentials.



Use Case	
Actor	user
Preconditions	The user has created an account and logged in.
Basic Flow	1. The user logs in. 2. The user clicks on a flashing area of the map. 3. The user begins playing the associated game.
Alternate Flow	The user can select a different game to start with.
Exceptions	If the user has not created an account the user will not be able to play games
Post conditions	The user finishes the game or quits.



4 Software Evaluation

Development Platform : Visual Studio Code

Version Control Platform : Github

Languages : Python , JavaScript, HTML and CSS

Framework : Flask library of Python allowing us to present our application via web browser. Our application tracks user progress via mySQL-Alchemy database. Flask is a WSGI application, used to run the application, converting incoming HTTP requests to the standard WSGI environment, and converting outgoing WSGI responses to HTTP responses.

Database : SQLAlchemy, an SQL toolkit that provides efficient and high-performing database access for relational databases.It gives you access to the database's SQL functionalities. It also gives you an Object Relational Mapper (ORM), which allows you to make queries and handle data using simple Python objects and methods

4.1 Software evaluation - Testing

Testing was done alongside development. After the basic code was written, tests were written to ensure that the code functioned as desired. Python was used for the creation of the website using flask, authentication, and database communication. These aspects of the code were tested using a framework called Pytest. Pytest was chosen due to its easy to use syntax and detailed and informative test results. Test results were recorded in a Requirements Traceability Matrix that includes a test number, name for the test, expected outcome, actual outcome, status and date.

test__init__.py

Test ID	Description	Expected Outcome	Actual Outcome	Status	Date	Comments
py000	test_create_app	Pass	Pass	Done	10/09/2023	All tests passed
py001	test_create_database	Pass	Pass	Done	10/09/2023	

```
  ✓ checkUserWord returns correct message and score for empty word (1 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.358 s
Ran all test suites matching /game.test.js/i.
```

PythonChatGame > jstests > JS game.test.js > ...

```
1   const { checkUserWord } = require('./regame.js');
2
3   test('checkUserWord returns correct message and score for correct word', () => {
4     const result = checkUserWord('test', 'test');
5     expect(result.message).toBe('Congrats! test is the correct Word');
6     expect(result.score).toBe(5);
7   });
8
9   test('checkUserWord returns correct message and score for incorrect word', () => {
10    const result = checkUserWord('test', 'wrong');
11    expect(result.message).toBe('Oops! wrong is not the correct Word');
12    expect(result.score).toBe(0);
13  });
```

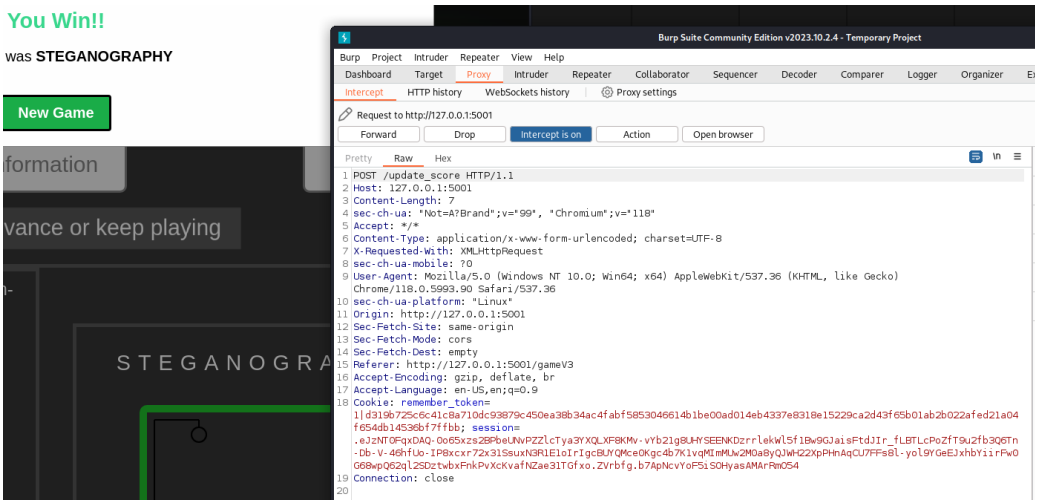
The games themselves were written in javascript that was embedded in HTML For this testing we used the jest framework. Various aspects of the game's code were tested for their functionality ensuring variables are updated correctly as players score and the collision would work as expected. We also conducted play and usability tests with our peers to get their feedback on how player friendly the interface was and how effective the actual training content was. We had a peer play each game and then fill out a questionnaire. Their feedback was recorded and then used to make improvements to the game.

4.2 Software Evaluation - Security

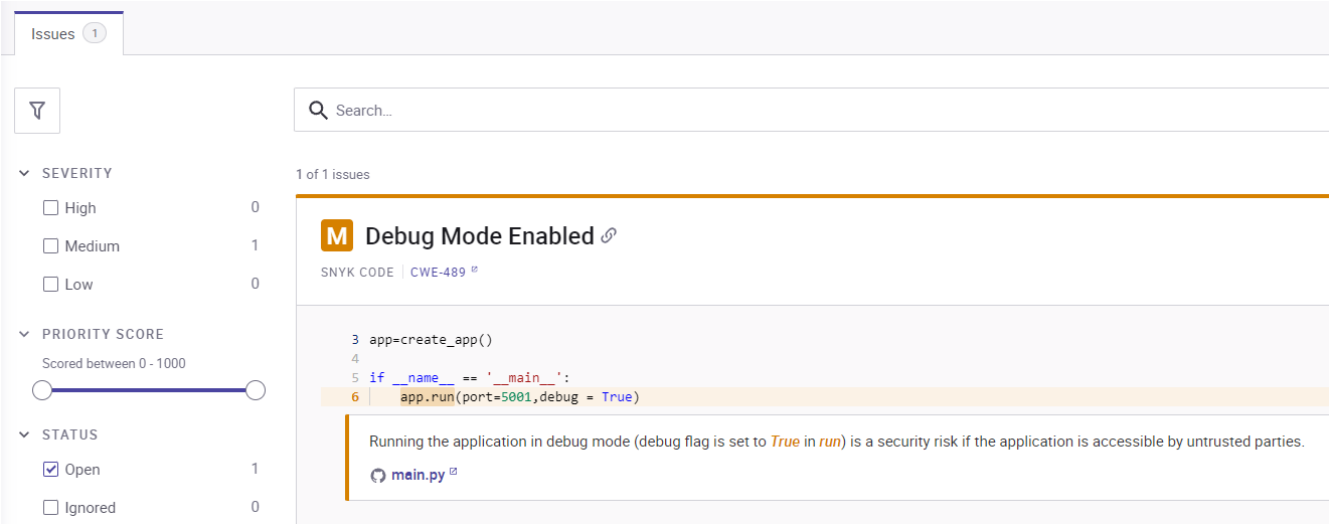
Security was evaluated at all steps of project development by using a security-by-design approach. Observed places to add security tests and where there was already built in security from our development tools with python flask. Implemented security considerations that were recommended.

```
if user:
    flash('Email already exists.', category='error')
elif len(email) < 4:
    flash('Email must be greater than 3 characters.', category='error')
elif len(first_name) < 2:
    flash('First name must be greater than 1 character.', category='error')
elif password1 != password2:
    flash('Passwords don\'t match.', category='error')
elif len(password1) < 7:
    flash('Password must be at least 7 characters.', category='error')
elif re.search(r'[\';"<>]',first_name):
    flash("Name contains potentially malicious characters. Please enter valid input.", category='error')
elif re.search(r'[\';"<>]',email):
    flash("Email contains potentially malicious characters. Please enter valid input.", category='error')
```

Burp Suite for Web Application Testing. Due to built in security it was not integral to our uses and more for consideration of hosting our application on a website.



Snyk as a code analysis tool to keep in line with Secure Coding Standards and to find any potential vulnerabilities.



Security issues that are still open include making error not be too descriptive and that debug mode is off for all parts or the project. Also adding brute force attack prevention by implementing lockout policies. Documentation for Security Requirements and Threat Identification can be found in Appendix A and B respectfully.

5 Work to be Done

Work to be done includes hosting the application on a different server, A dedicated web server will be required to host the website on as well as a database to store player information. There are many services available for hosting web applications without needing to maintain our server, network, domain, etc. Some services may have a free tier up to a certain time or bandwidth. Many services that use one of the WSGI servers described above, or a similar interface include PythonAnywhere, Google App Engine, Google Cloud Run, AWS Elastic Beanstalk and Microsoft Azure. Also adding sprites/characters to the Portris game along with animations and creating an administrator portal to manage existing user accounts.

6 Guide

A set up guide is provided with the following youtube link.

https://youtu.be/iRApakP4RJg setup and installation

https://youtu.be/FDtDLhdMrBI Walkthrough

https://github.com/AlexReichel02/PixelCola

Product name:

PixelCola

Product features:

Ability to create an account and play a series of mini games while learning about cybersecurity. Progress is tracked and saved upon closing the application

Product functions and capabilities:

The product functions as a learning platform for cybersecurity concepts. It has the capabilities to prepare people transitioning from other fields into cyber security to take the CompTIA security+ test as well as gain baseline knowledge in the field.

Installation instructions:

- Download latest version of Python("pip" will be automatically installed),
- Download the latest version of Git Bash
- Navigate to the Github repository https://github.com/AlexReichel02/PixelCola
- Clone the github repository to your local machine.
- Following packages that need to be installed through PIP :
    - pip install flask
    - pip install flask_login
    - pip install sqlalchemy.sql
- In your terminal (command prompt, git bash, etc.) change directories to the PixelCola directory then type the following:
    - For Linux or MAC - python3 main.py
    - For Windows - python main.py
- Copy the url the terminal provides, the application will be running on http://127.0.0.1:5001 copy this into your browser of choice.
- Create an account and begin playing.

Frequently asked questions:

- I don't know anything about computers. Can I start with PixelCola?

Yes absolutely, PixelCola is meant for users of all users to learn as well as retain information.

- Is my computer powerful enough to run PixelCola -

Yes, it runs in the browser and can run on any operating system.

- Is there replayability?

PixelCola is very replayable, multiple playthroughs will help with retention of concepts or brushing up.

Troubleshooting:

- If the program will not run, try reinstalling the following:
    - the latest version of Python
    - pip install flask
    - pip install flask_login
    - pip install sqlalchemy.sql

Appendix A: Software requirements

Req ID	Requirement Name	Description
SR-01	Set secure password requirements	Protect against brute force attacks by having complex password requirements
SR-02	Protect Session	Use Session protection against users' sessions being stolen
SR-03	The database has accurate account data	No errors when being asked to validate or login
SR-04	Input Validation	Validate and sanitize all input data to prevent SQL injection, Cross-Site Scripting (XSS), and other injection attacks.
SR-05	Data Encryption	Data should be encrypted in transit and at rest.
SR-05	Hashing	Hashing methods should be secure
SR-06	Error Handling	Customize error messages to reveal minimal information to attackers.

Appendix B: Threat Identification

#	Threats (types of attacks)	Vulnerabilities	Security Controls (safeguards/countermeasures)
1	SQL Injection	Improper Viewing of Database	Checking for malicious characters in input
2	Cross-Site Scripting (XSS)	Improper handling of user input in web pages	Input validation
3	Session Hijacking	Weak session management	Use secure session management techniques and monitor for unusual activity
4	Man-in-the-Middle Attacks	Lack of encryption for data in transit	Implement SSL/TLS (HTTPS)
5	Brute Force Attacks	Weak or no account lockout policies	Implement account lockout
6	Information Disclosure	Exposing sensitive information in error messages	Customize error messages to be general