LexiLock

Sushanth Reddy Racham Reddy
Clemson university
sushanr@clemson.edu

Joanna Lin
Clemson University
jlin23@clemson.edu

Venkata Sai Ganesh Varun Duggirala
Clemson University
vduggir@clemson.edu

ABSTRACT

In the context of this report, we have crafted a straightforward Python game using Flask and HTML/CSS which is infused with the festive spirit of Halloween. The game's core objective revolves around the task of identifying and recording words from a letter grid known as the wordament. Once all the words on the screen have been successfully collected, they are automatically transferred to a password generator for later use. Remarkably, our game exhibits robust functionality and includes an intelligent computer-controlled generator that aims to provide a challenge for human players. This engaging game is exclusively designed for single-player mode, boasting smooth and user-friendly graphics, with the added convenience of mouse control. What sets our game apart is its adherence to the fundamental principles of software engineering, ensuring a structured and well-designed gaming experience.

Keywords
Python, Flask, HTML/CSS, Software engineering

1. INTRODUCTION

As a trio of developers, we've collaboratively undertaken the creation of this game project, a significant component of our Foundation of Software Engineering course. Throughout this course, we've diligently applied a range of software engineering techniques. Our primary driving force behind embarking on this game development journey was the realization that this particular course, Foundation of Software Engineering, empowers us to apply fundamental Agile[2] principles. Moreover, it underscores the pivotal role we play in Scrum meetings, which, in turn, has a profound impact on our game's development. We've recognized that our focus extends beyond merely coding software; we're deeply invested in various facets of the game, including its narrative, environment, gameplay mechanics, user interface, characters, and more.

In the intersection of linguistic artistry and digital safeguarding, Wordament and the Password Generator stand as pillars of modern utility. Wordament, with its lexical challenges akin to crafting intricate passwords, elevates language skills and requires a nuanced understanding of words. In contrast, the Password Generator assumes the role of guardian, crafting cryptic passcodes that defy intrusion, mirroring

the meticulous selection of words in poetry. This report unveils the symbiotic relationship between these two remarkable creations, showcasing their synergy in empowering individuals to engage creatively with language while fortifying their digital presence in an ever-evolving cybersecurity landscape.

The main objectives of the game are:

- Create a project game in the python programming language with the functionality of traditional games of this type.
- Design a game in such a way that it is easy to control and the game interface is controlled by a single player.
- Develop the game with minimum hardware and software requirements.

2. Gameplay

In this immersive gaming adventure, participants embark on a captivating journey that unfolds through two distinct phases, challenging their linguistic aptitude and deductive reasoning.

In the initial phase, players are warmly welcomed into the game's world, greeted by a 4x4 grid(figure 1) populated with an assortment of randomly generated uppercase letters. Meticulous care has been taken to ensure that this grid conceals a minimum of 50 cleverly hidden 4-letter words, intricately woven within the letter arrangement. The primary objective in this phase is to uncover a total of twenty valid 4-letter words, meticulously concealed within the grid's intricate landscape. Players engage in a strategic endeavor, selecting letters by simply clicking on the grid's buttons, allowing them to construct words according to their preferred sequence. However, a captivating twist adds complexity to the challenge: the chosen letters must be adjacent to one another on the grid, introducing a delightful puzzle-solving element to the gameplay. As players artfully piece together their chosen words, each word dynamically materializes below the grid. The validation of their word selections is as straightforward as a click on the "Submit" button. With each unique and valid word submitted, the game responds with enthusiastic encouragement. An ever-vigilant counter diligently keeps track of the number of valid words yet to be uncovered within the grid, adding to the suspense and motivation. As the game unfolds, players become increasingly engrossed in the labyrinthine array of letters, their determination mounting with each newly discovered word. The ultimate achievement lies in successfully identifying twenty distinct and valid words, a milestone that propels them into the exhilarating realm of the Password Game.

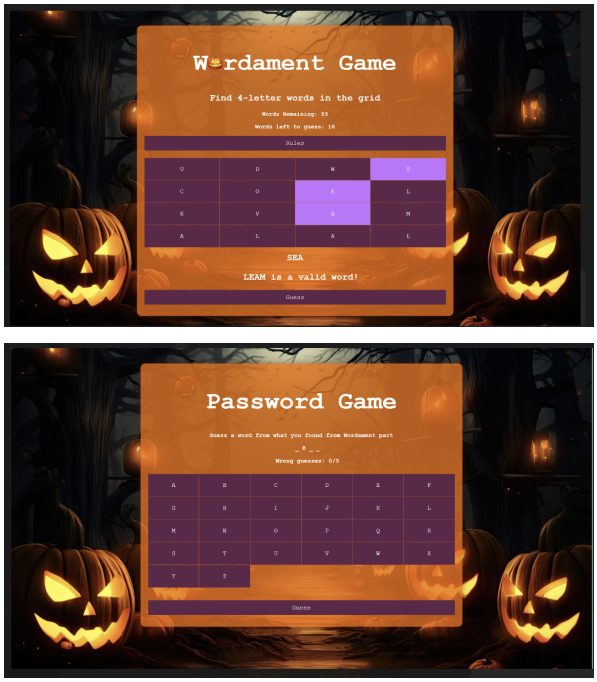

Figure 1: Wordament and Password Game Gameplay

The Password Game(figure 1) introduces a captivating twist as one of the words unearthed in the initial phase is randomly designated as the elusive "mystery word." Initially veiled in secrecy, this word is presented with placeholders, each represented by underscores, symbolizing the letters within it. Players embark on a quest to decipher this enigma, armed with a finite allowance of five incorrect guesses. To aid in their deductive journey, the complete alphabet is elegantly presented as clickable buttons, offering players a rich array of choices. With each accurate guess, the corresponding underscores gracefully yield to reveal the actual letters, providing players with tantalizing insights into their progress. However, the stakes are high, as each erroneous guess inches them perilously closer to the maximum threshold of permissible mistakes, a constant reminder conscientiously displayed for player reference. The game culminates with either the triumphant unveiling of the entire word within the allocated limit of wrong guesses or the bittersweet realization of having exceeded this threshold, resulting in a valiant yet unrewarded effort. Irrespective of the outcome, the game thoughtfully acknowledges players with aptly crafted messages, presenting them with the choice to reset their journey and embark on a fresh adventure or gracefully exit the game, having savored a captivating and intellectually stimulating linguistic escapade.

3. Implementation

Environment: We have used the Visual Studio Code and MacOS to run this project.

Programming Language: We have used Python programming language(Python 3.7.4) for implementation purposes and Flask[1] for micro web framework, it serves as a minimalistic Python web framework, enabling the creation of web applications through its straightforward and adaptable platform, simplifying tasks such as request handling, routing, and HTML template rendering. Also, HTML and CSS are indispensable for UI/UX design, structuring content, and styling interfaces to create visually pleasing and user-friendly web experiences. They enable layout design, interactivity, and accessibility, contributing to a positive user interaction. Additionally, we optimized HTML and CSS[3] to enhance website performance for improved UX.

IDE: We have used Visual Studio notebook while developing the code for the program.

Microsoft word: Documentation for the report is done using Microsoft word.

4. Code Decode

```
{% with messages = get_flashed_messages(category_filter=["word_label"]) %}
    {% if messages %}
        <ul class="flash">
            {% for message in messages %}
                <h3>{{ message }}</h3>
            {% endfor %}
        </ul>
    {%else%}
        <ul class="flash">
            <h3>_ _ _ _</h3>
      </ul>
    {% endif %}
{% endwith %}

{% with messages = get_flashed_messages(category_filter=["hangman_label"]) %}
    {% if messages %}
        <ul class="flash">
            {% for message in messages %}
                <h3>{{ message }}</h3>
            {% endfor %}
        </ul>
        {%else%}
        <ul class="flash">
            <h3>Wrong guesses: 0/5</h3>
      </ul>
    {% endif %}

{% endwith %}
```

Figure: 2 Password.html code

Above code(figure 2) explains about the result of the letter guess. This code snippet manages and

displays flashed messages in a web application. It first checks for messages in the "word_label" category and displays them as a list of h3 elements if any are present. If there are no messages, it shows a default placeholder message ("_ _ _ _"). The second part does the same for messages in the "hangman_label" category, displaying them if available, and showing a default message indicating the number of wrong guesses (initially set to 0 out of 5) if there are no messages. This code is used to provide feedback or information to users in a web application.

```
<form action="/guess" method="post">
    <h2><p id="guess_word" name="guess_word" value = ""></p></h2>
    <input type="hidden" id="guess_word_hidden" name="guess_word_hidden" value = "" />

{% with messages = get_flashed_messages(category_filter=["message"]) %}
    {% if messages %}
        <ul class="flash">
            {% for message in messages %}
                <h2>{{ message }}</h2>
            {% endfor %}
        </ul>
    {% endif %}
{% endwith %}
```

Figure 3 Code in index.html

So in the figure 3 it explains about the Action /guess is to call the function which is mapping in URL “/guess" on python code. (submit_word). This HTML form enables user word guesses in a web app. Entered words are displayed and sent via POST to "/guess," and there's commented-out code for displaying messages, likely for providing feedback within a game or interactive feature.

```
<div class="keybord">

    {% for key in keybord %}
        <form action="/key" method="post">
            <button type="submit" name="key_id" value = "{{key}}" class="subfont">{{key}}</button>
            <input type="hidden" id="guess_word" name="guess_word" value={{word_to_guess}}>
        </form>
    {% endfor %}

</div>
```

Figure 4: code for password button grid

Finally in figure 4, is for the Build a 26 button grid for the letter to guess in the password game. This HTML template functions as a keyboard interface for a word-guessing game, generating buttons for each key. When a user clicks a key button, it triggers a POST request to the "/key" endpoint with the selected key and the current word being guessed ("word_to_guess"). This allows users to submit their word guesses interactively.

5. Project Development Overview

Over a duration of six weeks, our team embarked on a dynamic journey to ensure the triumphant completion of this project. To structure our efforts effectively, we divided this time span into three distinctive two-week sprints, each marked by its unique focal points.

In the inaugural sprint, our primary endeavor revolved around comprehensive planning. We dedicated a significant portion of this phase to strategizing our course of action, determining which game concepts to initiate, and delineating the frameworks required to underpin our project. It was a period characterized by extensive brainstorming sessions and strategic discussions, laying the groundwork for our venture.

The subsequent sprint ushered in a shift in our focus towards the refinement of the user interface. During this phase, we delved into the realms of creativity and design, laboring to craft an immersive and user-friendly interface that

would elevate the overall gaming experience. Concurrently, we undertook the formidable task of seamlessly integrating the two distinct game modes. This stage involved bridging the gap between the initial word discovery phase and the subsequent password game, ensuring a coherent transition.

As we embarked on the final sprint, our attention pivoted towards rigorous testing and meticulous refinement. This phase was dedicated to comprehensive testing to unearth and rectify any potential issues or glitches that might impede the smooth gameplay experience. It also entailed the meticulous fine-tuning of various game elements, ensuring that the project adhered to our exacting standards of quality and excellence. This stage was marked by an unwavering commitment to detail and a relentless pursuit of perfection.

Collectively, these three sprints encapsulated our transformative journey spanning six weeks. It encompassed everything from the conceptualization and strategic planning phase to the creative design and seamless integration phase, culminating in exhaustive testing and the ultimate refinement of our project. The unwavering dedication and collaborative spirit of our team played a pivotal role in steering our game project to a triumphant conclusion.

6. Conclusion and Future Scope

We have effectively developed Lexilock, a game using Python, Flask, HTML/CSS, all while adhering to solid software engineering principles. We conducted comprehensive testing, including the integration of a sophisticated computer-driven password generator designed to challenge players' knowledge. Our game is designed for single-player control and is compatible with video calls, demonstrating our commitment to delivering a seamless user experience. Throughout this project, we gained valuable insights into project management techniques used by professionals in handling large-scale projects. Additionally, we refined our teamwork skills and acquired expertise in collaborative development. Looking forward, there is still an opportunity for improvement, particularly in successfully implementing a feature that mandates players to only connect adjustment letters when guessing the word.

7. REFERENCES

[1] Chauhan, Nidhi & Singh, Mandeep & Verma, Ayushi & Parasher, Aashwaath & Budhiraja, Gaurav. (2019). Implementation of database using python flask framework: college database management system. International Journal of Engineering and Computer Science. 8. 24894-24899. 10.18535/ijecs/v8i12.4390.

[2] Sharma, Sheetal & Sarkar, Darothi & Gupta, Divya. (2012). Agile Processes and Methodologies: A Conceptual Study. International Journal on Computer Science and Engineering. 4.

[3] Peroni, Silvio & Osborne, Francesco & Di Iorio, Angelo & Nuzzolese, Andrea & Poggi, Francesco & Vitali, Fabio & Motta, Enrico. (2017). Research Articles in Simplified HTML: A Web-first format for HTML-based scholarly articles. PeerJ Computer Science. 3. e132. 10.7717/peerj-cs.132.