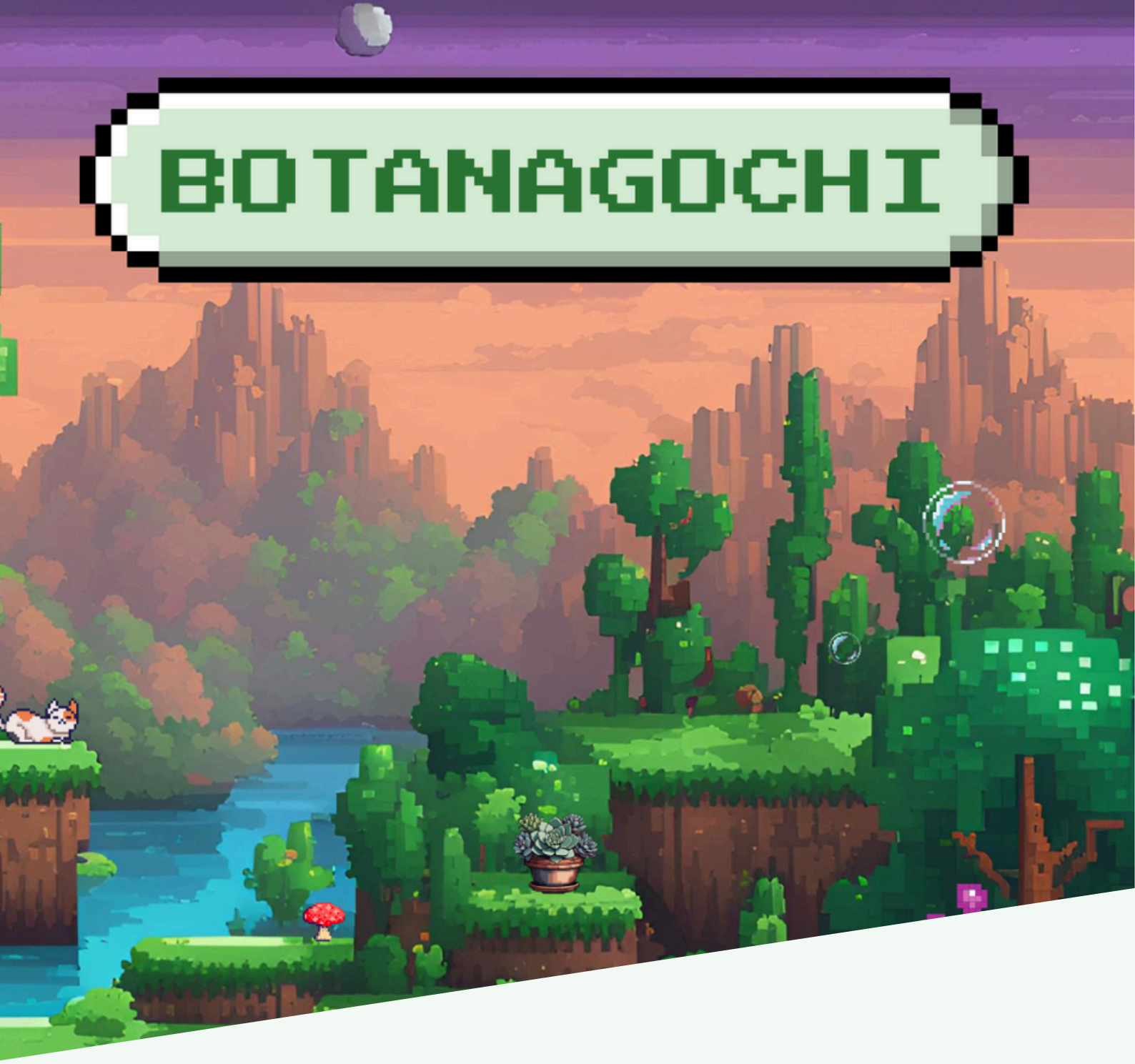


REFİYE KEVSER TÜRKER | 3047413

TB II PROJECT REPORT

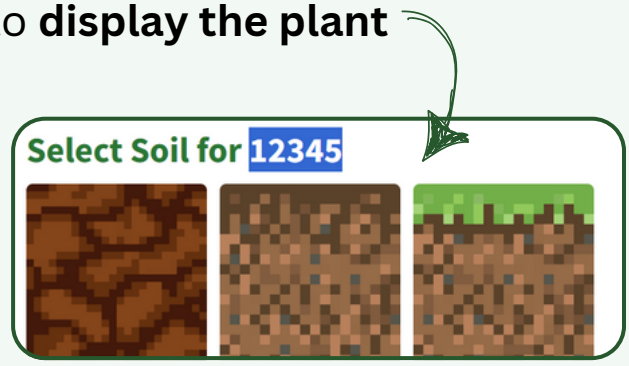
WINTER SEMESTER 2024-25

• Introduction

Botanagochi is an interactive virtual plant care application that I further developed using Streamlit on top of the concepts from last semester. In this part of the project, I tried to make updates on user experience, add more functionality, and make it more than a local app by using a cloud database and deploying it!

• Added Features

- Previously, the app only supported a spider plant. I added a **succulent** option to provide users with more variety.
- I added a **Login** and **Registration Page**, using MongoDB connection for cloud storage. This was one of the easiest parts of the project as we had already covered it in class together.
- By using **"session_state"**, I store the plant name and fetch it from the database. This allowed me to **display the plant names** depending on the user input.
- I added **reward pages** specific to the plants, which display a care video when the user completes the game by using **st.video widget**!



- More features were added to the layout apart from the game:
  - ***Plant Chores Checklist:*** By using Streamlit's **checkbox** widget, a page was created where the user can track what they need to do for real-life plant care. Unfortunately, I did not have the time to arrange it in a way to store the progress of this checklist, but this can be added for future versions.
  - ***Plant Community****:* This page displays a QR code to a shared Telegram community for plant enthusiasts where they can exchange plants.
  - ***AI Chatbot:*** I created an initial **enhanced prompt** for the AI to be a plant expert and assist users in identifying plants. To make it more accurate, I gave **pre-existing options** by using different widgets and **injected them into the prompt string** for response generation. Next time I can add an image input, as I learned Hugging Face has image classification abilities. I think this was a creative way to integrate AI.

- **Design**

Layout: I used columns for better alignment and organization of UI elements as I did not have enough knowledge on HTML and CSS.

I used **st.markdown** to apply color changes and improve visual appeal according to the theme.

In some of the cases, I used my previous HTML experience to play with the size, and the alignment of the text. Unfortunately, I could not use it to align some of the images as they were mainly in GIF format, and they lost their motions when I tried.

```
st.markdown(
    "<h1 style='text-align: center; color: #2b7735;'>How would you like to proceed with your virtual world? 🌱</h1>",
    unsafe_allow_html=True
```

I added an audio player by using st.markdown and adding audio controls at the top of the app to play relaxing background music. This was the simplest method to incorporate sound into the experience.

Streamlit's default button styles were limited, and did not match the theme. So, I made additional research on CSS customization for a more personalized design. I added it at the top of the code as a button style, and allowed HTML so that the customization is applied to all the buttons added. However, it was not applied to the login and registration buttons due to time limitations.

The visuals were again created by me using Canva. I tried to implement the pixel theme as much as possible, but it was a bit more difficult due to the design restrictions of Streamlit.

Challenges and Limitations

- While integrating the AI chatbot, I mishandled the secret file containing the MongoDB password. I resolved this issue easily by defining **unique credentials** for Hugging Chat, which made the AI bot work without an issue.
- Initial testing with friends exposed issues when registering usernames containing capital letters, special characters, or other unexpected inputs. I fixed these problems by using **.strip() and .lower()**. I had to validate these carefully to ensure compatibility with MongoDB in the registration page code.
- There was an issue with the navigation between the pages: The user needed to click twice on every button to proceed. After some research, the most likely reason was that the session state was not updated immediately. Therefore, I needed to use st.rerun() while defining the page navigation function to force a rerun after updating the session state.

References

- Audio player
- Personalized buttons
- Makıng sure the customization is not overridden
- Solution to prevent double clicking to proceed
- Making sure that the response of the AI bot includes emojis: ChatGPT
- Changing the color of the titles
- Checkboxes
- Key parameters for checkboxes
- Visual designs: Canva
- Help for connecting the prompt, selection, and response generating: DeepSeek