KokoroZen

Virtual Pet Self Care App

Project Documentation

CFG Degree - Summer cohort '23 - Sw3 - Group 7

Team members: Clara Londono, Elena Ailenei, Kelli Garnett, Sara Murray, Zsanett Horvath

Introduction

In our fast-paced lives, taking care of ourselves can often get overlooked. Recognising this modern challenge, our group project introduces a transformative concept—an innovative virtual self-care pet app. This web application merges the joy of caring for a virtual pet with the importance of self-care. Users can nurture a virtual character while exploring various self-care activities, transforming self-care into an engaging journey. By adding a playful touch and holding users accountable, our app encourages them to prioritise their mental well-being.

The core essence of our app is reimagining the concept of self-care through the lens of an interactive virtual companion. Drawing inspiration from the gamified world, the app introduces an engaging twist to self-improvement. Your virtual pet mirrors your well-being, and by taking care of it, you're taking care of yourself. The app's magic lies in its ability to motivate. It prompts users to actively engage in different self-care practices, creating a positive routine. By blending enjoyment with personal growth, it redefines self-care as both fun and meaningful. It's like having a supportive friend reminding you to take care of yourself, leading to a happier and healthier you.

The Virtual Pet Self-Care App boasts an array of practical and engaging features. It includes a comprehensive to-do list equipped with functions for adding, editing, deleting,

and marking tasks as complete. These interactions dynamically update the virtual pet's health and happiness scores, displayed on the user interface in a health and happiness bar. The app also offers feeding, hugging, and watering buttons, encouraging interaction and care. API Ninjas contributes motivational quotes through the Quotes API, infusing positivity and encouragement into users' self-care journeys. The app’s foundation relies on a MySQL database, proficiently storing user profiles, virtual pet data, and activity details. The frontend experience is crafted using HTML, CSS, and Bootstrap, ensuring an intuitive and visually appealing interface. Flask, our chosen backend framework, seamlessly connects all components, creating a harmonious user experience.

This documentation tracks our CFG Degree group project's journey according to the initial plan we outlined. It covers the essentials of our project, how we overcame challenges, and the tools we utilised. You'll get a glimpse of the project's major timeline, its core architecture, and our approach to testing. We'll also touch on important Python libraries that made our code work beyond its initial environment. You'll learn who did what in our team and get a sneak peak into what lies ahead. In essence, this roadmap takes you through our project's evolution, from its beginning to potential future strides.

Background

As a group, we brainstormed ideas that we were interested in which were focused predominantly around music, travel, self-care, and games as these were topics all group members were interested in.

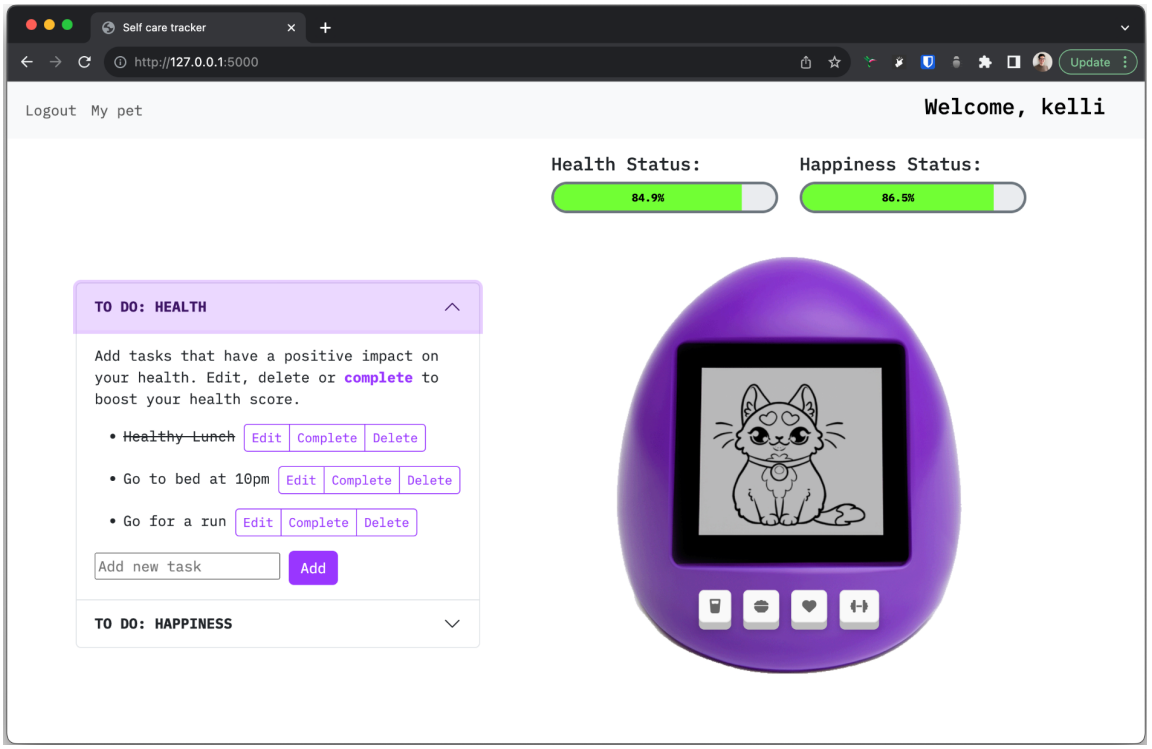
We generated 17 app ideas which were added to our Trello board. We expanded on the details of each idea, scoped whether each project could be made to meet all the above requirements, and within the timeframe. We then went through a series of votes to pare down the list and arrive at a final project choice–to build a web-based app which is a self-care virtual pet, along similar lines of Tamagotchis that we were familiar with from being kids. Building on Kelli's concept, we chose the name 'KokoroZen' for our web application. 'Kokoro' is a Japanese word,' which translates to 'heart' or 'mind' in English. This aligns with the calming essence of 'zen,' signifying peace and serenity. Together, 'KokoroZen' beautifully represents our app's purpose of nurturing self-care and well-being.

Our app operates on simple yet powerful principles. Users sign in to discover a tailored to-do list linked to their virtual pet's well-being. Task completion directly affects the pet's health, reflected in its appearance. This connection enhances the app's sense of accomplishment. We’ve integrated easy-to-use buttons for essential actions like watering, feeding and hugging, which button gives a motivational quote. These actions bridge digital and real-world self-care, boosting the pet's health while symbolising users' self-care

efforts. Neglecting self-care has consequences. If tasks are ignored, the pet's health and happiness naturally decline over time. This gentle reminder reinforces the significance of consistent self-care engagement.

Specifications and design

We developed the web application using the Flask micro framework, a versatile tool renowned for its ability to swiftly create and scale web applications. This framework facilitated an efficient enhancement of the user interface. Our app incorporates a MySQL database, supported by backend logic coded in Python, and the front-end designed using HTML and CSS with the Bootstrap framework. For seamless communication between the user interface, server-side logic, and the database, we implemented our API. Additionally, the app fetches data from an external Quotes API to enrich user experiences.



Overview of the homepage after user login - featuring the to-do lists, the visual health and happiness bars and the tamagotchi toy with the action buttons and the cat on the screen.

The website layout for the project presents the homepage that includes HTML login, supporting both accessibility and usability. After successful login, users are greeted with a left-hand box providing the option to curate their own self-care to-do list, complete with input validation and limitations. This list is linked to the health and happiness scores, prominently displayed at the top right corner in the form of visual bars. The main design element, situated on the right, is an imitation of the iconic tamagotchi toy—a virtual companion representing the user's self-care journey, taking the form of an adorable cat.

The cat's reactions dynamically mirror the user's interactions, shaping its appearance positively or negatively. To encourage engagement, the design integrates 3D buttons labelled "food," "heart," and "water," and “exercise” prompting users to interact further with their virtual pet. A noteworthy aspect of the design becomes apparent when the user engages the "heart" button. This action triggers an external API call that retrieves a motivational quote, adding an uplifting touch to the user experience. When crafting the web app, our intention was to evoke a Y2K design, paying tribute to our childhood and the era during which Tamagotchi emerged.

Our back-end logic supports the app’s essential features including the ability to interact with the virtual pet using the action buttons and effectively manage their own to-do list, with functions such as adding, editing, deleting, and marking tasks as complete. These actions not only maintain health and happiness scores but also engage a gradual decay mechanism when the app is active. This decay, which occurs at intervals of every 30 minutes, moreover, the second decay function addresses the period between app sessions, simulates the passage of time and adds realism to the experience. As briefly mentioned before to enhance user convenience, we've introduced a permanent session feature. Once initiated, these sessions remain active for a duration of 30 days. This strategic decision eliminates the need for frequent logins and allows users to effortlessly pick up where they left off, maintaining an uninterrupted connection with their virtual pets.

In terms of data management, we've integrated an SQL database. This relational database includes 4 tables and effectively manages user profiles, virtual pet information, task and interaction data, contributing to a personalised and engaging experience.

Implementation and Execution

At the start we appointed our Scrum Master Kelli. With her guidance, to allocate roles effectively, we used a combination of SWOT analysis and personal preferences. Our goal was to align individual strengths with corresponding tasks. Flexibility was a guiding principle and we were also open to working together, often coding in pairs or groups, and we used GitHub to collaborate on our code.

Throughout the project each team member had numerous roles and contributions:

- Clara played an important role as our Python and Flask expert. She spearheaded the creation of Python classes and the initial to-do list functionalities. Her expertise extended to implementing sessions and conducting extensive unit testing. Additionally, she provided significant contributions to the login page and refined the code for improved quality and brought together different components to ensure the app worked seamlessly as a unified whole.

- Elena was our cat's 'birth-mum' as she designed the cat visuals and made it a very likeable and 'easy-to-engage-with' character. She was also responsible for managing the API integration and establishing the connection between the API logic and the virtual pet component. She participated alongside the team in the initial stages, contributing to idea generation and expanding on concepts.
- Kelli played a dual role as our Scrum master and front-end guru. Her responsibilities were overseeing administrative aspects and conducting mid-week check-ins to ensure our progress stayed on track. She led the front-end design, commencing with the creation of a Figma Wireframe. This blueprint evolved into a fully functional user interface through the use of HTML, CSS, and Bootstrap. In addition to her front-end proficiency, she actively contributed to the login feature and took charge of managing our GitHub repository.
- Sara was our versatile allrounder, she engaged in multiple facets of the app's development. Originating from her initial concept, she remained involved in diverse areas. She assisted in establishing the API connection, devised the Python-based decay function for health and happiness scores, contributed to database integration, and played a significant role in fostering team communication and collaboration by working in pairs with all team members.
- Zsanett embraced two distinct and contrasting responsibilities. On one hand, she designed the MySQL database and worked in Python to establish the necessary connections. On the other hand, she seamlessly transitioned into a more creative role, contributing to documentation tasks throughout the entirety of the project and actively engaged with the team from the outset, contributing to idea generation and further developing concepts.

In the first week there was some initial confusion and chaos, but we swiftly adopted the Agile development methodology. We started working in weekly sprints and standup meetings to talk about the progress we've made. Any additional meetings were held outside of the sprints either in Slack chats or smaller meetings with only necessary team members.

To keep track of our tasks, we populated a Trello kanban board and designed a comprehensive product backlog. We organised the tasks using MoSCoW prioritisation, segmenting them into four groups: "must have," "should have," "could have," and "would be nice to have." This helped us to see what needed to be done based on how important and doable it was.

For the most part, we were able to avoid the need to wait for specific pieces of code by collaborating concurrently on various parts of the project. This approach allowed us to merge everything together towards the end. Although the process of combining the code took some time, the extra effort was well justified. Opting not to wait for other tasks to conclude allowed us to accomplish a substantial amount of coding work that might otherwise have been delayed.

As team members completed their tasks, they underwent thorough testing using unittest and subsequent error review. Upon completing a task, they transitioned to the review phase. A pull request was made to involve other team members, who reviewed the code and proposed changes. Only when everyone agreed, we integrated our work into the larger project.While all this was happening, we made sure that different parts of our app could work together smoothly. We tested how everything interacted to make sure nothing was broken or causing problems. We kept checking and making improvements to make sure our app was running well. Overall, our way of getting things done involved planning, testing, reviewing, and making sure everything fits together nicely. This approach allowed us to steadily build and improve our app as a team.

To run our code successfully, we've imported a handful of libraries successfully. We have opted for the Flask library as our web framework of choice, utilising its capabilities to build web applications. To automate tasks, we've integrated APScheduler, allowing us to schedule recurring processes. For handling time-related user interactions like tracking last activity for decay functions, we've employed the datetime module. Our configuration needs are managed by the config library, enabling us to adjust values and parameters externally. To facilitate HTTP requests and API calls within Python, we've integrated the requests library. For maintaining code quality, we've leveraged the unittest library for effective testing. Lastly, we've used the mysql.connector library to establish connections with MySQL databases, enabling seamless interaction and retrieval of query results.

Undertaking this project was a significant step for most of us, and we encountered various obstacles along the way. A significant chunk of time was dedicated to selecting and conceptualising the project idea, which initially seemed quite abstract. This made the first week of the project a bit confusing as we adjusted to the new tasks and structure, but we managed by adopting Agile methods. This required good organisation and teamwork since it was a new experience for us. One initial hurdle was bringing together code from different parts of the project into a cohesive document. In our collaborative work process, we faced challenges while using tools like Git and implementing the Flask framework with unfamiliar libraries. We also had to manage our various commitments, such as jobs, education and family commitments which sometimes made it challenging to coordinate group meetings and allocate project work effectively. This highlighted the importance of managing our time and responsibilities well.

Testing and Evaluation

During the planning phase, we drafted a Quality Assurance (QA) and User Acceptance (UA) Test Plan, which outlined the testing strategies, scope, and objectives to ensure the app's quality and reliability. Our primary goal was to subject the application's features and interactions to rigorous testing, guaranteeing the delivery of a top-notch product that aligns

with user expectations and offers an optimal user experience. While time constraints limited our ability to fully execute the plan, we focused on crucial testing aspects.

We carried out functional testing to validate the core functionality of the application. We used print statements and collaborative efforts to ensure the functionality of our code. Two developers reviewed each task, spotting any potential oversights. This rigorous process concentrated on safeguarding the database, API, Python logic, and web app from unexpected errors. The final code assembly was a collaborative effort, employing pair programming for efficient structuring. In terms of unit testing we used a mix of testing methods, conducting tests during and after coding, aligning with personal preferences. Both approaches proved effective, supporting teamwork in reviewing and stress-testing the code. This comprehensive approach aimed to encompass a wide range of potential issues. Upon having a functional prototype, we engaged with external users unfamiliar with the app's development (such as family members and friends) carrying out blackbox testing. Their feedback provided valuable insights, although time limitations prevented immediate implementation of changes. Our intention was to incorporate these insights in a subsequent sprint if more time were available for the project.c

At the time of submission, our web app stands as a fully operational creation; however, it bears the limitations typical of projects developed within constrained timeframes. The imposed time constraint prevented the realisation of all our initially envisioned functions, leading us to opt for a more streamlined version. Within the mere four-week project duration, we confronted the reality of feature exclusion due to time scarcity. Acknowledging our limits, we strategically focused on specific key elements resulting in a thoughtfully refined product. Reflecting on this journey, it's interesting to think about how the project might have evolved with more time.

Conclusion

The Virtual Pet Self Care App was born from a clear objective: to tackle the prevalent issue of neglecting self-care and mental well-being in our fast-paced world.

Our app presents a dynamic solution that resonates with users seeking a path to consistent self-care practices. We successfully completed our desired aims which we set at the beginning - to integrate technology and personal wellness. Our app offers a captivating and user-friendly platform backed up with Python logic, connected to an SQL database and an open API.

We believe this project is a result of teamwork and growth over the past sixteen weeks. It showcases our knowledge from the CFG Degree, as well as our strengths and development. This journey has taught us valuable skills in collaboration, problem-solving,

and innovation. We've enjoyed the process of working together and consider ourselves fortunate to be part of such a communicative team.

Looking ahead, we're enthusiastic about further developing the app, leveraging our newfound knowledge, and expanding its capabilities. This project underscores our dedication to merging technology with well-being, highlighting the significance of positive habits for mental health.