NUNKI V1.0

CFG Software Stream

Group 6: Alejandra, Alex, Desiree, Kirsty, Maria, Morag

Contents

Introduction

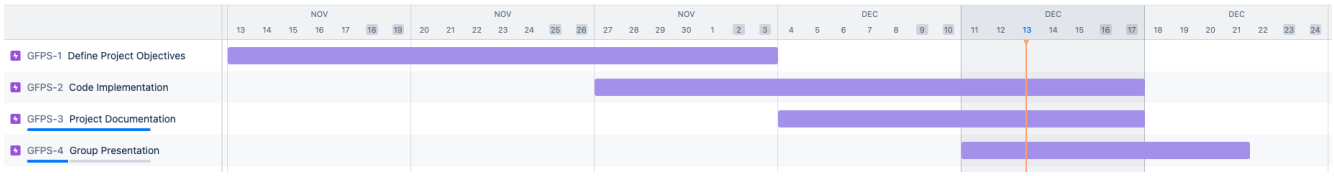
Aims & Objectives

We are a dynamic group of English and Spanish speakers driven by our shared fascination with language and learning. We aimed to create an engaging and positive language learning app to educate users on the different grammatical parts of English sentences, leveraging a sophisticated word generation API. With thorough research, trial and error, meticulous planning and many bugs, we have successfully collaborated to bring to life NUNKI V1.0 - the gamified English learning game.

Roadmap

Below is the project roadmap extracted from Jira showing the 6-week project broken down into:

1. Defining Project Objectives (specification & Design)
2. Code Implementation (& execution)
3. Project Documentation
4. Group Presentation



Background

Learning the parts of speech in English can be challenging for non-native and native speakers.

This game is intended as an educational resource for both learners and teachers and those wanting to improve their skills and knowledge.

The game is set up for beginner (A1) level English where the user completes an activity to practice their skills in recognising parts of speech (e.g. noun, verb, pronoun).

Using the API to generate sentences where each word is paired with its corresponding part of speech, the user can learn and practice correctly identifying these elements in a sentence and improve their understanding of this crucial aspect of language.

Specifications & Design

Key features:

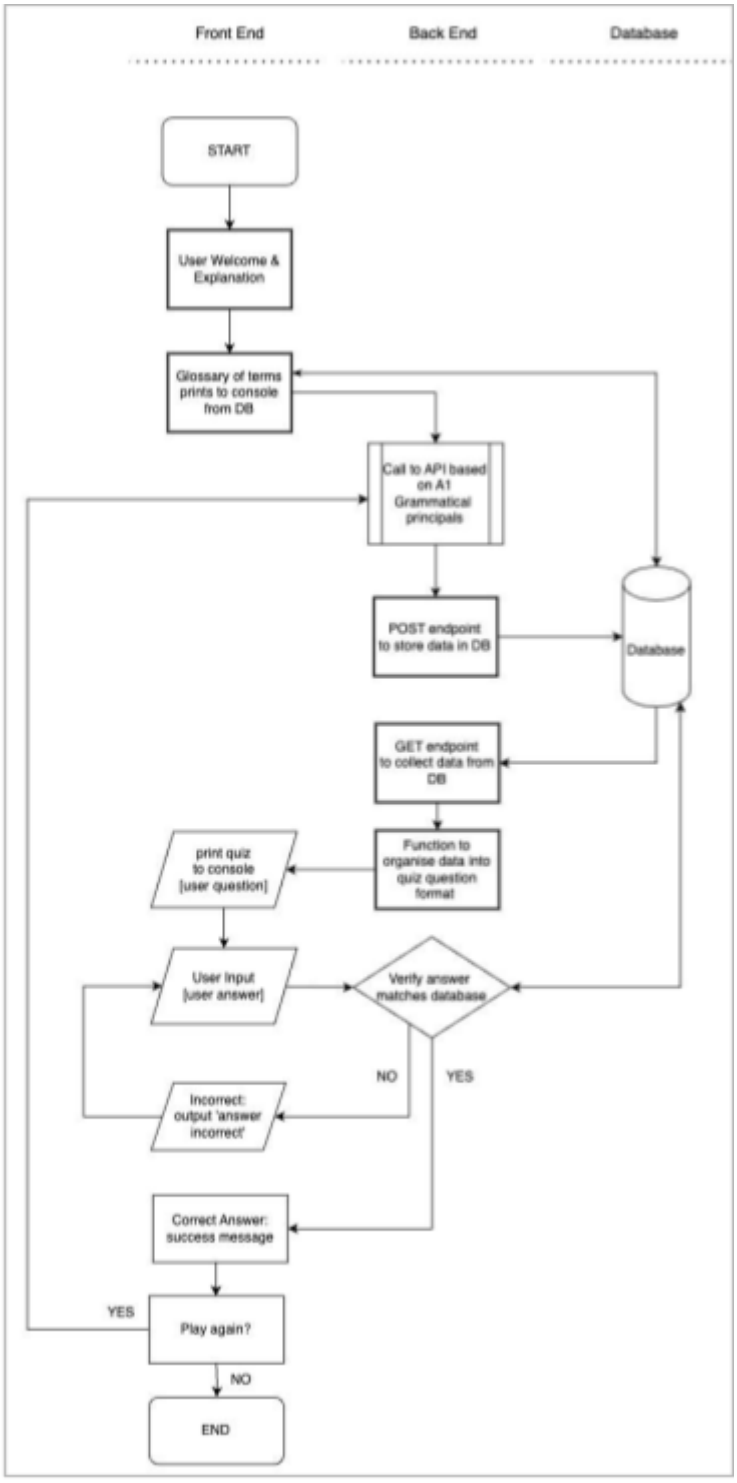
- Console-based Front End: The user interacts with a user-friendly console-based interface, ensuring a straightforward and immersive learning experience.

- API Calls for Sentence Generation: Utilizing the power of API calls, sentences are dynamically generated, with each word thoughtfully paired with its corresponding part of speech (e.g., [cat, noun]).

- Database integration: A robust database integration facilitates efficient word storage, ensuring a diverse and comprehensive learning experience.

- Game logic for sentence-based activities with user input: Engaging game logic guides users through sentence-based activities, prompting them to identify and understand various parts of speech

- Encouraging messages throughout the game: To maintain a positive learning environment, the game incorporates encouraging messages throughout, motivating users as they progress through the language-learning journey.



Flow Chart of Game

Folder Structure [Github]:

Front End

- README.md
- frontend.py

Back End

- db_utils.py
- app.py
- config.py
- __init__.py
- language_game_db.sql

Tests

- game_logic_unittest.py

Implementation & Execution

Development Approach & Roles

Our team's approach to development has been collaborative. We quickly discovered common interests and aligned this to the project then used a SWOT analysis to understand how best to work to individual strengths. During this process, we understood time combined with individual availability to be the primary risk to the project's success.

Once we had agreed on the game we used MoSCoW to define the Minimum Viable Product (MVP). This required an iterative approach and more time than initially planned to focus our idea and ensure deliverability. The idea was tested through code snippets, diagrams and thorough research. This revealed how overly ambitious our initial game concept was.

Having all agreed and understood the game design flow chart a combination of software development methods were used to maintain team collaboration, motivation and a successful output.

Alejandra	Alex	Desiree	Kirsty	Maria	Morag
README.md Presentation frontend.py	Database app.py db_utils.py Endpoints	Jira Board Github app.py config.py db_utils.py Endpoints	PM Jira Board app.py Documentation	database frontend.py Branding Presentation	Database config.py db_utils.py Testing Endpoints

Communication tools: Slack, Google meet, Google Sheets (meeting minutes), Github and Jira.
OpenAI API: https://platform.openai.com/docs/guides/text-generation/chat-completions-api

Tools & Libraries

Project Requirements:

- Python==3.12
- PyCharm Community Edition 2023.3.1
- Flask 2.2.5
- OpenAI==1.3.9
- mysql-connector-python==8.2.0
- requests==2.31.0
- python-dotevn==1.0.0
- pip=23.2.1

Frameworks & Libraries:

- Flask (https://flask.palletsprojects.com/en/2.2.x/)
- MySQL (https://www.mysql.com/products/connector/)
- Requests (https://pypi.org/project/requests/)
- Sys (https://docs.python.org/3/library/sys.html)
- dotevn-python (https://pypi.org/project/python-dotenv/)
- String.io
- os.path
- Json
- unittest

Agile Development

Our team worked together using an agile approach as demonstrated in the roadmap whereby the design phase and code implementation phase overlapped. With time being the primary constraint and the functionality of the end product being critical. Features were removed or adapted as code snippets were developed.

Implementation Process and Challenges

Challenges, outlined below, included an originally over-ambitious scope, aligning of team schedules, code errors and API limitations.

Ambitious original scope

The initial idea had to be scaled back to reach the MVP (Minimum Viable Product). We were ambitious as a team though have learned of and acknowledged the limits of our current skill level. The MoSCoW method was implemented several times when must-have features became could-have features as our understanding evolved. The 'could have' features that can be implemented in future versions include a Spanish version, another API for image generation, a front-end web page with user interactions and alternative games such as a sentence scrambler.

Aligning Schedules

The team's original aim was to meet regularly and divide the project into Jira tickets to allow everyone to work individually and then come together to review. However, due to unforeseen circumstances, the challenges of the team coming together meant adjusting the strategy to suit. We adapted by implementing pair programming - coding together on calls, working together on documentation output and regrouped as a team to design the process. This has suited the team dynamic more than relying solely on the Jira ticket system resulting in more cohesive outputs.

Code syntax

Early in the design phase we discovered several syntax errors. Following some online research it was discovered that OpenAI had recently released a new version of OpenAI. The corresponding documentation had not yet been updated. Therefore syntax in the documentation was different and conflicted with their code syntax. We were then able to coordinate and fix the syntax errors.

API limitations

During development we discovered the OpenAI API had a limit on the number of calls made per day before being required to purchase access. In future versions we would overcome this potentially costly issue by adapting the API request to retrieve 10 sentences and their grammatical components in a single request, enabling the player to play 10 times before another API request is made.

Testing & Evaluation

Testing Strategy

Testing Priorities:

- Priority was given to identifying critical issues that could break the game.
- Minor defects, such as user inputting the wrong data type, were deferred to later versions as they didn't impact the game's functionality.

Custom Errors Handling:

- Unsuccessful API calls
- Correct/incorrect user choice and their answers
- Incorrect data type inputs

Challenges:

- The API's inconsistent output required significant development time to ensure consistent results, impacting the comprehensiveness of testing.

Exception Handling:

- Added to the backend files to manage unsuccessful API or Database calls, ensuring correct execution or error identification.

Function and Unit Tests:

- Functionality testing covered various scenarios like different input cases and incorrect date types, prompting custom error messages.
- Unit tests used hard-coded responses to control testing conditions due to the API's inconsistent responses.
- Further unit tests were planned for API calls, database responses, and error handling, postponed due to time constraints.

Future versions should include more comprehensive testing, including end-user feedback and enhancements to program optimisation.

System Limitations

It was noted during testing that when the user answers a question incorrectly, the same question is not repeated. Whilst this does not break the game, it does impact user experience. Due to time constraints, this was not corrected before launch, but should be considered for future releases.

Conclusion

In conclusion, our group project aimed at developing a language-learning game using a word-generating API was a profound experience. Our primary objective was to create an engaging platform that educates users on various parts of speech within a sentence through an interactive console game.

Throughout the project lifecycle, we encountered several challenges that tested the team's resilience and adaptability. Code coordination emerged as a crucial hurdle, requiring meticulous planning and effective communication to ensure seamless integration of features. Additionally, navigating team member availability required agile scheduling and resource allocation and having initially agreed upon a strategy resulted in adaptation to incorporate pair programming and multiple drop-in sessions to deliver outputs.

The most pivotal phase involved the iterative refinement of our initial idea. We encountered complexities that necessitated a series of simplifications before commencing the development phase. This process ensured our concept aligned with the project's objectives and could effectively meet the 'must have' criteria.

We all recognise the value in continuous improvement and given additional time and resources would further enhance the game, incorporating user feedback and exploring additional features to make it more accessible to a broader audience for example adding various levels to suit the ability of the user, creating a Spanish version of the game and implementing an additional API to incorporate images as flashcards to assist those who learn more effectively with visual aids.

Despite the hurdles, our team triumphed over challenges and accomplished our objectives. The culmination of our efforts is a robust language learning game, NUNKI V1.0, designed to educate users on the intricacies of parts of speech, providing an enriching and positive immersive learning experience.