Group 4 Project Document: Guess The Girl

INTRODUCTION

For this project, we sought to develop **"Guess the Girl"**, a daily music-themed web app game inspired by the logic used in the popular games *Wordle* and *Spotle*. The aim is for players to guess a randomly selected female musical artist of the day. The game is designed to be a single-player experience, with a colour-coded feedback system. Clues about the artist, such as genre or country, are revealed with each guess in the form of colour-coded tiles, providing the player feedback on the accuracy of their guess.

To ensure the game is accessible to users with varying levels of musical knowledge, we introduced three difficulty levels, each offering a different number of guesses:

- Easy: 10 guesses
- Medium: 6 guesses
- Hard: 3 guesses

As a group of female developers, we chose to spotlight female musical artists, celebrating their contributions to the music industry—hence the name *Guess the Girl*. The game primarily targets puzzle enthusiasts and commuters who enjoy quick daily challenges, but it also seeks to engage individuals passionate about discovering and supporting female artists. Designed to appeal to a broad demographic, including music lovers and casual gamers aged 18 to 45, the game remains accessible to both younger and older audiences. With its varying difficulty levels, *Guess the Girl* ensures inclusivity, catering to players with a wide range of musical knowledge.

Furthermore, we have deliberately marketed the game such that there is scope to expand the categories to other professions, such as athletes, actors, influential historical women etc, in the hopes of empowering and educating people on successful women across many fields.

Primary Objectives of the Project:

1. **Gameplay** - Develop a web app that provides a captivating, music-themed gaming experience using familiar logic.
2. **Robust backend** - implement a reliable backend system to:
    - Randomly generate daily artists from MySQL database.
    - Process user guesses and retrieve information (such as genre, country, etc.) from the database based on the artists that have been guessed.
3. **User-friendly interface** - Create a visually attractive and intuitive UI to maintain a seamless and enjoyable user experience.

Roadmap of the project:

1. **Project development** - The group selected topics of mutual interest, such as music, puzzles, and feminism, following which we finalised the game logic, key features, and outlined the required pages to develop.
2. **UI development** - We conducted a competitor analysis to identify popular features from successful games (both direct and indirect competitors) and integrated them into our design, ensuring a user-friendly and engaging interface. Additionally, we also analysed user reviews of similar apps to identify any weaknesses and incorporate this feedback into our design (see 'Specification and Design').
3. **Frontend Development** - The core pages (including the homepage, difficulty selection etc). and the key components (navbar and footer) were designed and developed, ensuring that the user input worked with the endpoints.
4. **Backend Development** - We set up the API server, integrated the database and built the necessary endpoints which interacted with the game page.
5. **Testing and Debugging** - Testing and debugging were done iteratively throughout the process of development. Post-development, we conducted testing to ensure the game's functionality and user experience met our requirements.

BACKGROUND

Topic Background:

With the shift from traditional means of passing time to the current offerings of the digital world, puzzles—an activity often thought to be associated with older generations—have also seen a gradual transition from newspapers to online platforms, such as the New York Times website. Seeing a boom during the lockdowns that followed the COVID-19 pandemic [1], daily puzzle games are now a mindless staple activity for all generations, including Gen Z [2], offering a healthier alternative to other digital time-passers such as 'doom scrolling'. With the popularity of games such as 'Wordle' (which garnered upwards of 2 million daily users as of 2023 [3]) and similar games ('Worder.ly', 'Heardle', 'Worldle', etc.), the demand for online puzzles is clearly on the rise, with the projected market value estimated to reach 22 billion US dollars by 2027 [4]. Given this demand—and a shared affinity by us, the developers—we decided to contribute to this market by offering a feminist, musical twist on the popular logic of 'Wordle'. According to our research, there is a notable lack of puzzles made by women with women at the forefront of development. This gap in the current online puzzle market led us to develop our web app, *Guess The Girl.*

Game Logic:

Guess The Girl is an engaging guessing game that challenges players to use logic and deduction to identify the randomly selected female artist of the day. Below is an outline of how it works:

1. **Artist Selection:** At the start of the game, the system selects an artist from a predefined list of top artists from the SQL database. Along with the artist's name, key attributes are stored to help the player narrow their guess (Artist's Name, Country, Genre, Star Sign, Debut Year, and Spotify Ranking).
2. **Player Input:** Players are given a set number of guesses based on the selected difficulty level (Easy: 10 guesses, Medium: 6 guesses, Hard: 3 guesses). Each guess must be a valid name from the database. Invalid guesses will be rejected.
3. **Feedback System:** After each guess, the game provides feedback on the correctness of the artists attributes and their positions using a color-coded system:
   Green - The attribute is correct and the artist belongs to that particular category.
   Yellow - The attribute is close to the correct answer. This feedback is only applicable to the numerical categories of the game (debut year & spotify ranking). For 'debut year' close means the correct answer is within 5 years of the guess. For 'Spotify Ranking' it means the correct answer is within 50 places of the guess.
   Grey - The attribute is incorrect and the artist does not belong in that category.
4. **Elimination Process -** Players use the feedback to eliminate possible solutions and refine their guesses. For example:

   Grey Attribute: If an attribute (such as **Country**) is marked **grey**, the player knows that the artist does not belong to that category, so they can eliminate all artists from that country in future guesses.
   Yellow Attribute: If an attribute is marked **yellow**, such as **Debut Year** being close to a guess (e.g., 2015), the player knows the correct answer debuted within a 5-year range of the guessed year. This helps narrow the scope of possible artists further.
   Green Attribute: If an attribute is highlighted in **green**, it indicates that the attribute is correct. This helps the player to narrow down guesses based on the correct attributes, and thereby identifying the artist.

Figure 1.
Key of the tile colours and their meanings.

5. **Winning and Losing:**
    - The player wins if they guess the correct artist within a maximum number of attempts.
    - The game ends in a loss if the player does not guess the artist after the maximum attempts.
    - The player can end the game using the 'Give Up' button. This will reveal the correct answer, and they will have the option to go to the home page.
6. **Artist List:**
    - The artists list consists of 60 top female artists in the music industry. These artists are stored in an SQL database along with the key attributes: Country, Genre, Star Sign, Debut Year, and Spotify Ranking.
7. **Gameplay Strategies:**
   Players can start with artist names and choose an attribute to focus on, such as country, where they can narrow it down early to where an artist is from. (e.g., USA or UK). Eliminating unlikely attributes early and focusing on narrowing possibilities is key. This straightforward yet engaging logic makes *Guess the Girl* addictive and fun while also challenging players' music knowledge and deductive reasoning skills.

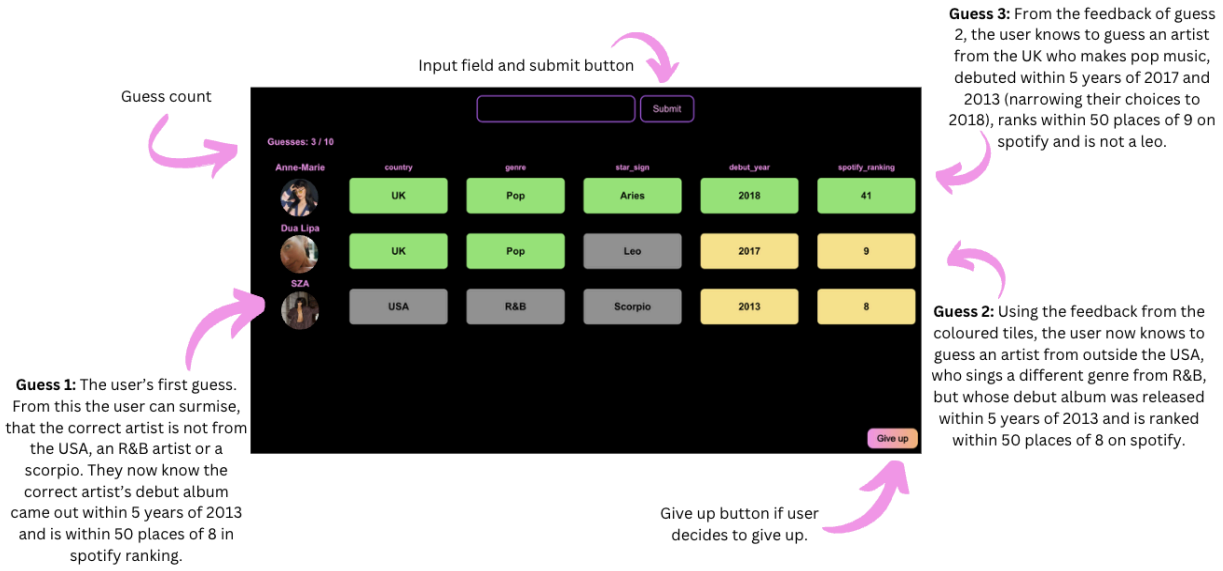

Figure 2.
Example of gameplay with descriptive annotations

SPECIFICATIONS AND DESIGN

Requirements:

Technical: The frontend of the game is built using React, allowing users to select their preferred difficulty level, input guesses, and submit them. The backend, powered by Node.js, retrieves data from a MySQL database to calculate the accuracy of each guess and returns the results to the frontend in a clear, user-friendly format. Version control is managed via Git and GitHub, ensuring seamless collaboration and tracking of changes throughout development. Postman is used for API testing, allowing us to validate the backend endpoints, ensuring they function as expected and handle data efficiently.

Non technical requirements: The web app should provide a seamless navigation experience, allowing users to easily explore the intuitively designed user interface (UI). Users should be able to input guesses for female artists and receive colour-coded feedback reflecting the accuracy of their

guesses based on various metrics, such as genre, location, and debut year. This feedback will help users deduce the correct artist, but if they are unable to, a "Give Up" button will reveal the correct answer, and users can return to the homepage. All wireframes and prototypes were designed on Figma, ensuring a visually appealing and user-friendly interface throughout the app.

Design and Architecture:

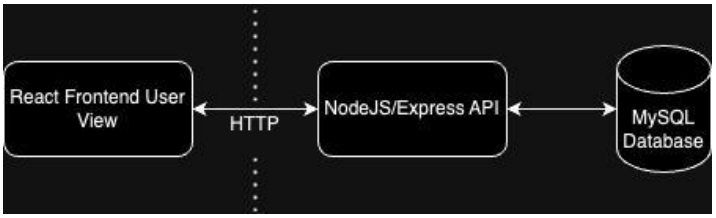

Figure 3.
Architectural diagram of the game

Process: We began by analysing competitors in the online gaming field (e.g. New York Times' puzzles, *Heardle*, etc.) to identify features we liked and disliked, based on both user feedback and our own observations. This was documented in our competitive audit. Each team member created initial wireframes, which we then reviewed and iterated upon, incorporating features from one another's design. We kept in mind the insights we gained from our research, particularly focusing on user preferences such as the desire for no timed limits, the importance of a 'give up' button, and a more engaging webpage. From this, we could select different aspects of each wireframe that improved our UI. Following which, we finalised our ultimate wireframe (click here for prototype).
Heuristics: Drawing from the work of Norman and Nielsen, we employed heuristics to increase the user-friendliness and intuitiveness of the web app. A key strategy was to opt for design choices familiar to users (e.g. hamburger menu), which minimises the cognitive load. Additionally, we adopted a colour-coding scheme similar to that of *Wordle,* ensuring a familiar experience while still staying true to our design. We also followed the unconscious Z-pattern that most users follow when scanning websites, ensuring it informed the design of each page layout. Additionally, we maintained consistent navigation, button, text and colour styles to reduce cognitive load, making the game more intuitive. Furthermore, the pointer changes from an arrow to a pointer to clearly indicate to the user its functionality. The game also offers real-time feedback when the player enters their guess with immediate color-coded feedback which improves the overall user experience. The system has built in error-prevention which prevents the user from entering invalid artist names not available in the database. Lastly, with the ability to adjust the game difficulty, allows flexibility for the user whether they be casual or more experienced players.
Colour Scheme: Following our market research we opted for 'dark mode' (dark background with light lettering) for our primary UI, as it is commonly used for media related websites and is often preferred by modern users (whom we expect to be our primary demographic). Moreover, dark mode reduces eye-strain and can enhance readability due to its highly contrasting nature. We particularly felt that a dark background highlighted the colours of the green and yellow tiles which, as a key component of our game, was a top priority.

Having opted for a dark background, we then looked at which colours would complement the overall theme of our game while maintaining a visually engaging contrast, ensuring a highly impactful UI. This led us to picking light pink as our primary colour, with purple and orange as secondary colours. Crucially this colour scheme provided the contrast required to highlight key UI components, thereby improving the usability and navigation of our web app. It was also important to avoid colours that were similar to the game tile colours, so as to not confuse the visual aspect of the game page. Moreover, as colours often associated with feminism and girlhood, we felt these colours affirmed our brand identity, aligning with the feminist values that underpin our game and further differentiated us from other similar games in the market.

Beyond the scope of this project, we would plan to incorporate a light mode toggle (which would invert the colours, such that the background was light and the lettering was dark pinks and purples), and an accessibility toggle which would use a non-colour coded system to give the user clues. This would ensure the highest level of accessibility for our game, inclusive of those with colour blindness.

IMPLEMENTATION AND EXECUTION

Tools and libraries:

Git/GitHub - Version control system used to manage code, track changes, and collaboration.
Figma - Used to create wireframes, prototypes and UI designs.
SQL Database - Used to store a database of artists and their relevant data.
React - Used for developing the frontend of the app and responsive user experience (dependencies: react, react-dom, react-router-dom, react-scripts, web-vitals, axios, @testing-library/user-event, @testing-library/react, @testing-library/jest-dom, @fortawesome/react-fontawesome, @fortawesome/free-solid-svg-icons, @fortawesome/free-brands-svg-icons, @fortawesome/fontawesome-svg-core).
Node.js - Used for API development (dependencies: cors, dotenv, express, MySQL2).
Postman - Used for testing endpoints.
Slack has been our primary method of communication, with **Google Meet** being used for regular meetings.
Jira was used for task tracking and team coordination, and **Google Drive** was used to manage and store our project-related documents e.g. minutes, research.
Jest was used for testing.

Team member roles:

- All team members contributed to the initial brainstorming and wireframe design.
- **Annie:** Finalised the wireframe, developed the API server and endpoints, integrated the API with the database and frontend, worked on the game page (SJX), and contributed to the report (Introduction, Background, and Specifications and Design).
- **Erin**: Developed the API server and endpoints, integrated the API with the database and frontend, performed testing, wrote readme.md instructions, and contributed to the report (Testing and Evaluation).
- **Joanna:** Worked on the homepage (JSX), the “How to Play” page (JSX), routing, and the game page, and contributed to the report (Background - Game Logic).
- **Laura:** Collated information on the top 60 female artists on spotify based on monthly listeners and developed the SQL Database, worked on HTML and CSS for the difficulty page, about us, navbar, footer, under construction component which was linked to the ‘statistics’, ‘feedback’ and ‘Suggest an artist’ pages, created the Guess the Girl logo using Canva and routing.
- **Nesrin**: Finalised the wireframe and created the prototype, worked on CSS and HTML for the frontend homepage and game page, worked on the game page (JSX), and contributed to the report (Implementation and Execution).
- **Sruti**: Organised the tasks in JIRA, created the image database (which was ultimately not used), developed the progress bar, made code readability edits/bug fixes and contributed to the game page development and README.md.

Achievements:

Integration of Frontend and Backend: The team was able to effectively link the frontend, built using React, with the backend API built with Node.js.
Feature Development: Priority and key features such as the difficulty level selection, colour-coded feedback system, and the ‘Give Up’ button were successfully implemented.

Responsive Design: The game is fully responsive, providing an optimal experience across various devices and screen sizes.
Data management: The creation of an SQL database that stored artist data was completed smoothly, providing quick retrieval for user guesses and real-time feedback system.
Implementation of Finalised Wireframe: The web app closely resembled the finalised wireframe. This alignment between design and implementation will ensure UI consistency and an intuitive user experience.

Challenges:

1. **Time Constraints:** Early in development, the team identified core priorities and "nice-to-have" features. Optional elements such as a multiplayer mode, the ability for users to create their own quiz, and a stats page, were considered depending on available time. The group focused on the essential features, such as the core gameplay and UI, leaving advanced features for future updates.
2. **Data Handling:** The team initially planned to use Spotify's API for dynamic data like monthly listeners, but considered integration issues, therefore decided to create a static database of 60 artists for simplicity and ease of development.
3. **Integration and Merge Conflicts:** The team was concerned about potential merge conflicts when multiple members were working on the same files simultaneously. To avoid this, we prioritized clear communication, regularly updating each other on our work in GitHub. Pull requests were reviewed frequently, and team members were proactive in ensuring there were no conflicts before merging, which helped maintain a smooth workflow and prevented issues with file integration.

Changes:

Promises in Back-End Development: Use of promises when writing back-end. This required research and practice of using a different syntax to the callbacks we used for class exercises and the API assignment, but led to much more readable and organised code.
Refining Difficulty Levels: A key adjustment made during development was the modification of the difficulty levels. An initial idea was to reduce the number of attributes as the difficulty increased. This was changed to make the harder levels involve fewer guesses while keeping the number of attributes the same for each level to maintain the UI of the webpage.

Agile Development:

Iterative Approach: The development process was carried out in an iterative manner. For example, the wireframes and game logic were continuously iterated based on research and team's feedback.
Code reviews: The group adhered to good coding practices by ensuring clear communication when working with shared files, particularly on GitHub, to avoid merge conflicts. Additionally, a different person from the one who created the pull request handled the merge to ensure objective code reviews.
Sprint-like approach: With each team member working independently on specific pages and/or features and collaborating with others as needed. We all joined and set up a sprint in JIRA, but found our regular group calls, and messages in slack to be a more convenient and efficient way of communication, so we did not end up using this tool.

TESTING AND EVALUATION

Testing strategy: Each component (and page) has an accompanying jest test file to ensure they render. MemoryRouter, a React Router component, was used to test rendering within routing and the application's URL. This not only ensured each component rendered successfully, but ensured they were routed as expected. **'**toBeInTheDocument()' was used within tests with 'getByText' or expect() statements containing constant variables. '{ selector: 'class_name' }' was also used to get elements by class name where text was not specific enough to locate elements.
To ensure that the database connected successfully, the database was mocked and tested. Documentation on how to test API endpoints in Postman is contained within the README in the backend folder.

Functional and user testing: A git clone of the repository in its final state was carried out by each group member. The README.md files were followed to ensure correct setup and installation of dependencies; all links were clicked on the webpage; and the game was played, to ensure that the application functions as expected.

System limitations:

The application runs and functions as expected without any delays. The only main concept in the game which we did not deliver on was the daily artist, who the player has to guess, not updating daily. Instead it changes when the game is restarted/the page is refreshed.

Extra pages and components: We have pages on our site which are 'under construction' - *Statistics, Feedback* and *Suggest an Artist*. We planned to include a progress bar to the game page as a design element which we did not manage to include in the final code. We left these until the end in order to focus on the functionality of the game and assessment criteria.

Extended game functionality: We currently have the standard 'easy (10 guesses)' level game available for play. On the difficulty select page, we have 'medium (6 guesses)' and 'hard (3 guesses)' levels displayed at 'coming soon'. If time allowed we would have duplicated the 'easygame.js' file with a limited number of guesses for each level, and linked them to the difficulty select page.

Testing: If time allowed, test files for end-points would have been created, as opposed to the tests being carried out through Postman. More thorough testing of the game page would have been carried out including testing of the css background-color changes for correct, incorrect and close guesses.

Design: Due to time constraints we did not get a chance to add images to the game, which is a key design feature. This makes the UI less engaging.

CONCLUSION

In conclusion, the *Guess the Girl* project successfully merges entertainment, education, and a feminist ethos to create an engaging music-based game. Through the integration of a user-friendly interface, a robust back-end, and varying difficulty levels, the game caters to a broad audience. Technical challenges, including merge conflicts and database integration, were effectively addressed through agile development and iterative processes. Looking ahead, future expansions will focus on enhancing accessibility and broadening the game's categories. This project exemplifies our ability to collaborate effectively, navigate challenges, and create a meaningful and enjoyable experience for users.