**MULT213 – A3 – HENRY – SELF REFLECTION**

**Self-Assessment: French Dictionary with Number Game**

For my project, I developed a **French Dictionary** web application using React.js, with an added **French Number Game**. The dictionary allows users to click on French words to display their meaning and an associated image. The number game lets users listen to French number audio (1–10), input the correct number, and track their streak, introducing a fun and interactive learning component.

**1. Strengths and Achievements**

1. **Successfully Met the Requirements**:
   * I achieved all **Tier A** and **Tier B** requirements, including interactivity, logical components, and live data fetching.
   * The dictionary works seamlessly with **dynamic rendering** of words, images, and meanings.
   * The game component allows real-time input validation, timer functionality, and streak tracking.
2. **Challenges and How I Overcame Them**:
   * **Dependency Conflicts**: During project setup, I faced errors due to dependency issues (e.g., React version conflicts). I overcame this by using *--legacy-peer-deps* during package installations.
   * **Image Paths Not Loading**: Initially, images in the dictionary didn’t display because the media folder wasn’t in the correct location. I fixed this by placing all media files in the public folder and referencing them correctly using relative paths.
   * **Audio Playback Issues**: Implementing the audio game required dynamically loading audio files for numbers. I used the browser’s Audio API and organized the audio files properly in the public/audio folder.
3. **New Concepts and Techniques Learned**:
   * **React Hooks**: I improved my understanding of *useState* and *useEffect* to manage dynamic data and component lifecycle events.
   * **Component-Based Development**: I separated the game functionality into a new reusable component, *FrenchNumberGame.js*, improving code structure.
   * **Dynamic File Management**: I learned to manage static assets like images and audio files using the public folder.

**2. Areas for Improvement**

1. **Better UI/UX Design**:
   * While functional, the app’s UI could be enhanced with better styling and animations. For example, the input field could visually highlight incorrect answers.
2. **API Integration for Live Data**:
   * Although I used static JSON-like data for words and local audio files, integrating a real API for fetching French vocabulary or pronunciation would make the app more robust and realistic.
3. **Additional Features**:
   * If I had more time, I would implement:
     + A score leaderboard for the game.
     + An option to expand the game to include numbers beyond 10.
     + A search bar for the dictionary, as initially planned but simplified based on instructor feedback.

**3. Reflection on Goals**

I met the project’s goals by:

* Creating an interactive dictionary that displays images and word meanings.
* Building a functional and engaging number game that enhances the user’s learning experience.
* Maintaining clean, modular code and utilizing React effectively.

However, I recognize the need for more polish in UI/UX and the integration of more advanced features like real APIs.

**4. Self-Assessment Grade**

Based on the assignment rubric and my overall performance, I would give myself a **95/100**.

* I believe I met all the key requirements for **Tier A** and **Tier B**.
* I demonstrated problem-solving skills, learned new concepts, and delivered a fully functional, interactive project.
* With more time, I could enhance the app further, but I am proud of what I accomplished.

**Conclusion**

This project taught me how to plan, execute, and debug a React-based web application. I improved my coding skills, overcame challenges, and delivered a project that meets the assignment goals. Moving forward, I aim to focus more on refining user experience and integrating real-world data sources to create even more dynamic applications.