Sprint Reflection

Features, Assumptions & Decisions Made During Development

- **User Authentication**
  - We relied on Django's built-in authentication framework, for username/password authentication, as it proved to be a sufficient connection.
- **Security Measures**
  - Django's default password hashing algorithm was assumed to mask passwords.
- **User Interface and Interaction**
  - Assumed that users would be able to access the web app using their devices and this would redirect them to their respective dashboard if logged in.
- **Backend and Database:**
  - We assumed that the database could handle queries efficiently, and was reliable for all dependent components.
- **Prototype Development:**
  - Card Images for the prototype were assumed, as they were auto-generated by AI.
- **QR Code Feature:**
  - Assumed that users would be able to automatically navigate to an auto-generated HTML page when scanning QR codes.
- **Frontend-Backend Integration:**
  - Understood that the connection between the Front-End and Back-End was not straightforward. Required deeper knowledge of Django views that were used to connect the Database to the Front-End.

Challenges Faced During Development & Rectifications

- **Merge Conflicts & Git Workflow:**
  - **Challenge:** Frequent merge conflicts were occurring during file restructuring and with branches not syncing with the main branch.
  - **Fix:**
    - Initiating pull requests from feature branches.
    - Merging main into feature branches before the final merge.
    - Increasing branching discipline and avoiding direct changes to main.
    - Enhancing communication via detailed Kanban board updates and more frequent standup meetings.
- **Django Infrastructure & Lack of Prior Experience:**
  - **Challenge:** Initial unfamiliarity with Django made it difficult, particularly for user authentication and verification.
  - **Fix:**
    - Studying Django documentation and following multiple online tutorials.
    - Gaining practical understanding of Django's project structure and its authentication system.
- **Front-End Development:**
  - **Challenge:** Inexperience with front-end development posed difficulties in creating the landing page HTML.

        - Learning key concepts through online tutorials, especially using Bootstrap.

Future Development of Existing Capabilities & New Features for Sprint 2

- **Account Security Enhancements:**
    - Implement forgot password functionality
    - Enable user account deletion
    - Add brute force/DoS protection, email verification, and two-factor authentication
- **UI/UX Improvements:**
    - Enhance landing page with animations, loading indicators, and interactive elements
- **Backend & Data Integrity:**
    - Increase database validation and automate tests
    - Extend card model attributes
- **Feature Expansion:**
    - Introduce booster pack creation for gamemasters
    - Create Trading Mechanisms

Sprint 2 Reflection

Assumptions & Decisions

During the development process, there were some critical assumptions that were made in order to render decision-making less complicated and more effective. One major assumption was that certain features were going to take a long time to implement, especially when there were problems that the team had not solved before. This assumption informed task precedence and time management since the team anticipated potential delays and scheduled for limited availability of team members as a result of coursework deadlines.

While developing the leaderboard, it was assumed that the database contained all the necessary information related to players, such as points and levels. This led to directly fetching data from the database and putting it in the context to make it available to the front end. This made data flow simpler and front end and back-end integration simpler.

Similarly, when working on the register form validation, it was presumed that the correct error would be obtained from the back end. So, priority was to display the error visually rather than verifying if it's correct, which streamlined front-end development.

Django framework functionality was also assumed to be functioning as expected, after the success of Sprint One and weekly meeting feedback. Additionally, the assumption that GeoLocation would be simple to implement was also not correct, as the initial method was taking too much time, and a change of approach was needed. These assumptions illustrate the trade-off between efficiency and flexibility in the development process.

Challenges Faced During Development

During the development process, our team encountered several significant challenges and issues that required strategic problem-solving and collaboration to resolve. One major challenge was dealing with merge conflicts that arose from a restructure of the project files. These conflicts disrupted the workflow and caused inconsistencies in the codebase. We rectified this by carefully resolving conflicts through version control and improving team coordination during file restructuring.

Another issue involved redirect problems with the login page, where error messages would not display after an incorrect login attempt. We resolved this by adjusting the error-handling logic and ensuring that the correct error messages were passed from the back end to the front end. A key challenge was aligning old features with the new project scope. A previous feature was no longer sufficient, so we had to modify the existing code to match new expectations. This required careful refactoring and testing to ensure compatibility with the updated system.

One of the biggest obstacles was the time constraints that prevented us from implementing both the battle and trading systems. We chose to focus on the trading system, recognizing it as the more complex and valuable feature. This decision allowed us to concentrate resources on completing one feature thoroughly instead of delivering two incomplete ones. We also faced skill mismatches within the team. Some back-end members had to work on front-end tasks, such as HTML for the leaderboard.

Personally, the team overcame this by using online tutorials and adapting quickly to front-end development, which ultimately enhanced my understanding of this area. Implementing real-time trading was challenging due to limited experience with WebSockets, causing frequent bugs. We resolved this by collaborating with team members and studying WebSocket documentation to gain a deeper understanding. Additionally, GeoDjango presented difficulties, so we switched to Geopy and Leaflet for geolocation functionality, simplifying the process and improving performance.