Project Title
ValoFoDummies.gg

Project Summary
It should be a 1-2 paragraph description of what your project is.
Valorant is a free-to-play 5v5 tactical first-person shooter. We aim to build a web app to help Valorant players improve by providing easily accessible information and recommendations. We plan to do this by making Valorant stats more understandable through visual formats, and also analyzing those stats to understand play habits. Using a combination of individual player stats and meta trends, the web app will give recommendations for things such as certain areas to focus on, or agents to play. Utilizing LLMs, the web app will be capable of giving advice for a particular match. It is also a fun site that allows users to draw comparisons between themselves and their favorite pros. They can also create an account to track their progress over time.

Description
State as clearly as possible what you want to do. What problem do you want to solve, etc.?

We want to give new and casual players an easy way to learn more about Valorant and how to improve their gameplay. Decision making will be less overwhelming as the app will recommend a few agents for a given map along with playstyle recommendations using data from pro players. Users will have an easy GUI to understand what agents and abilities are best for what map, a common question frequently asked.

On the other hand, all the operations for data analytics should be efficient, so the users get instant results. The game is very complex and has many variables, very difficult for casual players to sift through the mountain of data on other Valorant sites like blitz.gg and vlr.gg and come to their own conclusions. Using data from pro-player matches as well as casual matches will allow us to analyze patterns and trends for best performing agents and abilities and present this information in a simple and easy to understand format for our users.

We will also be displaying information of performance based on Agents, maps and other relevant variables so players who are more versed with the game can use the information.

Creative Component

What would be a good creative component (technically challenging function) that can improve the functionality of your application? (What is something cool that you want to include? How are you planning to achieve it?)

- Interactive Visualizations: create interactive graphs of relevant statistics such as player, agent and map statistics
    - Complex frontend and backend code to achieve this
- Machine Learning: predict live match win rate using the calculated stats of the other players on the team and the enemy team
    - Need to understand best ML algorithm to achieve this, train and test the ML
- LLM: we can fine-tune an open-source LLM to give specific Valorant insights, like creating live strategies against the opposing team or recommendations on playstyle per agent and per map.
    - Will need to understand LLM API calls, how to train the model, how to interpret output
- Player-lookalikes: A player can find a pro player with similar stats/play-styles as them. If the lookalike is a streamer, they would be able to get better at their preferred playstyle by watching them.
    - Have to tag players as Pros/Streamers/Casual

Usefulness
Explain as clearly as possible why your chosen application is useful. What are the basic functions of your web application? (What can users of this website do? Which simple and complex features are there?). Make sure to answer the following questions: Are there any similar websites/applications out there? If so, what are they, and how is yours different?

Everyone wants to improve their gameplay and analyzing pro stats can help them get insights on how to better play certain maps, agents, understand team comps and so on.
Features

- View own stats
    - Players can link RIOT accounts so we can use Riot API to get data
    - Combination with other data will allow for valuable insights
- User-Friendly GUI for important META information
    - GUI allows users to easily locate information for FAQ such as META team combinations for maps for team agents and roles
    - Personal Agent recommendations based on map/ desired role
    - More detailed information such as match or player-specific also displayed but not as prominent to avoid clutter

    - Display Attacker/ Defender sided maps
- Generate match based predictions for win loss
    - Use data on agent, maps and players to predict win loss %

Similar websites like tracker.gg or blitz.gg give insights into player stats, but they don't give direct tips for their users to improve. They are mainly a clutter of data one has to dig through to gain any real benefit. We offer recommended agents based on stats, desired role and map which are easy to see and interpret. We also display information that these sites do not like attacker/defender sided maps.

Realness

We want you to build a real application. So, make sure to locate real datasets. Describe your data sources (Where is the data from? In what format [csv, xls, txt,...], data size [cardinality and degree], what information does the data source capture?). It would be hard to satisfy stage 2 requirements with one dataset. Thus, we strongly recommend identifying at least two different data sources for your project.

- vlr.gg

This dataset would serve as the basis for recommendations as it is the match data of pro players. It can be scraped directly from the site or we can use past data through Kaggle (https://www.kaggle.com/datasets/visualize25/valorant-pro-matches-full-data). Based on Kaggle, the dataset is in the .sqlite format and has 4 tables.

The Game_Rounds table has 15531 rows and 4 columns; it contains brief information for each game, with the columns GameID, Team1ID, Team2ID and RoundHistory.
The Game_Scoreboard table has 157939 rows and 28 columns. It contains more detailed information on the scores of each game, referring to the GameID as found in Game_Rounds. The columns of this table can be used to find agents played and performance statistics such as kills, deaths and assists.
The Games table has 15888 rows and 36 columns. It contains more details on the games themselves, also referring to the GameID as found in Game_Rounds. The columns of this table can be used to find maps played in each game, which teams are playing, the winner of the game, and more.
Finally, the Matches table has 7818 rows and 12 columns. It contains additional details of each match, referring to the MatchID found in the Games table. The columns of this table contain the date and patch of each match, as well as other information like event details and map scores.

- Finding Popular Agent Picks for Specific Maps
    - Game_scoreboard table

    - Users can query the database to find out which agents are most frequently picked on specific maps by filtering data based on Map from the Games table and Agent from the Game_Scoreboard. Statistical analysis can show trends in agent selection which can be visualized on the site.
- Map Bias:
    - Games table
    - Team1_SideFirstHalf, Team2_SideFirstHalf, and round wins. Analyze match results to determine if a map tends to favor defenders or attackers. This involves comparing the win rates of teams starting on different sides and aggregating this data to show general trends for each map.
- Comparing Users to Pro players:
    - Game_Scoreboard for pro player statistics; integrate user data from Riot account.
    - By comparing user statistics (like ACS, kills, deaths, assists) to those of similar pro players, users can find role models in the pro scene whose play style matches theirs. This can guide users on specific areas to emulate or focus on improving.
- General Match/ Map/ Player Specific Data
    - All tables
    - Provide users with searchable databases where they can look up specific matches, maps, or player statistics. Users can see details round histories, scoreboard data, and match outcomes.
- Live Game Analysis and Recommendations
    - Real-time data integration with Riot's API for live game stats; historical data from Game_Scoreboard and Games
    - When user is in a live game, use machine learning to analyze ongoing match data against historical stats to offer real-time strategic advice and predictions
- Role-based Gameplay suggestions
    - Game_Scoreboard for detailed player roles and performance metrics
    - Analyze roles and corresponding performance indicators (first bloods for duelists, assists for supports) to provide tailored advice for users to adjust gameplay according to preferred role.
- Interactive Statistical Visualisation
    - All tables
    - Implement interactive graphs and charts that allow users to visualize data trends over time, compare between different players or teams and explore statistical relationship in the data
- Machine Learning for Win-Rate predictions

    - Include all gameplay data for training predictive models
    - Use ML to predict win probabilities based on game states, player performances and historical match outcomes. Can help users understand impact of certain strategies or compositions on chances of winning

If our team were to scrape new data from the vlr.gg site, it would likely follow a similar format to the Kaggle dataset, but with a greater number of rows.

- blitz.gg

https://github.com/IronicNinja/valorant-stats
This dataset is scraped from blitz.gg, which contains data about the entire Valorant player base across different ranks. The data is very granular and split by multiple levels. The top level is split by abilities, agents, maps, and weapons. Within each folder, there are separate folders for each of the maps and another folder for combined data across maps. Within those folders, there are individual CSV files, which are split by the ranks of players. To use this data, we would reformat it into tables, so that there would be relationships between the abilities, agents, maps, and weapons.

Functionality
This is where you talk about what the website delivers. Talk about how a user would interact with the application (i.e., things that one could create, delete, update, or search for). Read the requirements for stage 4 to see what other functionalities you want to provide to the users. You should include:

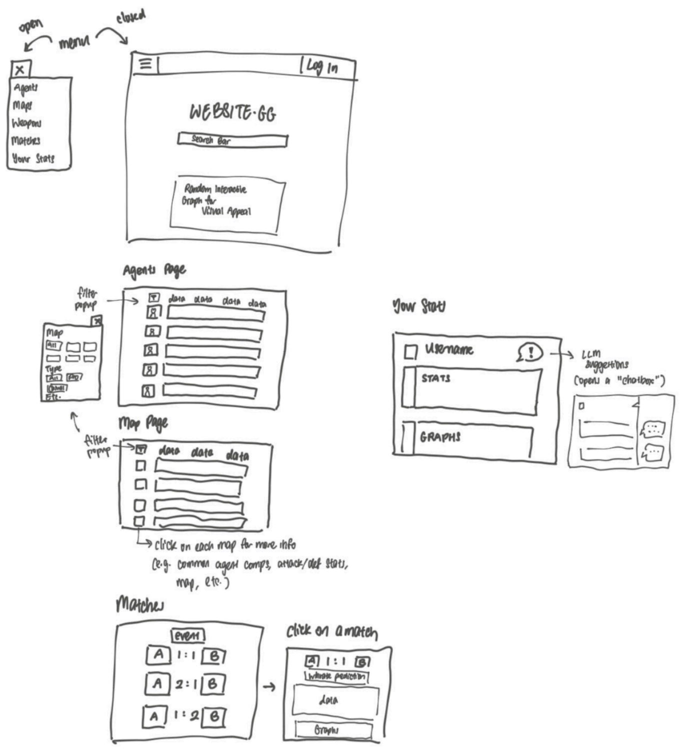
Our website seeks to allow users to understand the Valorant meta and use relevant information to improve their gameplay. For example, they may be able to:
- find popular agent picks for certain maps
- whether the map is defender or attacker sided
- compare themselves to pro players that have a similar playstyle/role to the user.
- See general match/ map/ player specific data

They would start interacting with the website by logging in with their Riot account, so we can get their match history. The API also allows us to see if they are currently in a live game or not, and we can use the match data to generate recommendations and predictions. If they are just browsing, past match history would be visible and it would be correlated to specific improvements based on the role of pro players.
- Duelists should have higher first bloods -> need to be more aggressive

- Support agents generally have higher assists -> Play more backline
- If player has anomalies in stats, could suggest role closest to player's natural playstyle based off of stats and swap

Besides the overall functionalities of the site, the site would also have an interactive feature that visualizes statistics and aims to utilize machine learning and LLMs to provide win-rate predictions and suggestions to improve player performance

A low-fidelity UI mockup
What do you imagine your final application's interface might look like? A PowerPoint slide or a pencil sketch on a piece of paper works!



Project work distribution
Who will be responsible for each of the tasks or subtasks?
Explain how backend systems will be distributed across members. Be as specific as possible as this could be part of the final peer evaluation metrics.

Frontend:

Andrew: React JS, Python, Java, Firebase
Daniel: React, Python, GCP

We will be responsible for making the layout based on agreed mockups and making API calls to the backend. We will primarily use React.js and data visualization libraries like D3.js for interactive datasets.

Backend:
Kris: Python ML & Visualizations
Acelynn: Python, LLM

The backend will be written in Python using the FastAPI library. We will use sqlite to access and query the local datasets and integrate Riot SSO so users can log in using their Riot accounts. We also plan to train and integrate an LLM to give recommendations that are tailored to Valorant. If any data needs to be cleaned and transformed into a different format (CSV to SQL), we will also handle that.

We plan to split the work of the actual app, but we will work on the design components (Stage 2 and Stage 3) together.

ANNEX:

Table: **Game_Rounds**

- GameID TEXT
- Team1ID INTEGER
- Team2ID INTEGER
- RoundHistory TEXT

Table: **Game_Scoreboard**

- GameID TEXT
- PlayerID TEXT
- PlayerName TEXT
- TeamAbbreviation TEXT
- Agent TEXT
- ACS INTEGER
- Kills INTEGER
- Deaths INTEGER
- Assists INTEGER
- PlusMinus INTEGER
- KAST_Percent REAL
- ADR INTEGER
- HS_Percent REAL
- FirstKills INTEGER
- FirstDeaths INTEGER
- FKFD_PlusMinus INTEGER
- Num_2Ks INTEGER
- Num_3Ks INTEGER
- Num_4Ks INTEGER
- Num_5Ks INTEGER
- OnevOne INTEGER
- OnevTwo INTEGER
- OnevThree INTEGER
- OnevFour INTEGER
- OnevFive INTEGER
- Econ INTEGER
- Plants INTEGER
- Defuses INTEGER

Table: **Games**

- GameID TEXT
- MatchID TEXT
- Map TEXT
- Team1ID INTEGER
- Team2ID INTEGER
- Team1 TEXT
- Team2 TEXT
- Winner TEXT
- Team1_TotalRounds INTEGER
- Team2_TotalRounds INTEGER
- Team1_SideFirstHalf TEXT
- Team2_SideFirstHalf TEXT
- Team1_RoundsFirstHalf INTEGER
- Team1_RoundsSecondHalf INTEGER
- Team1_RoundsOT INTEGER
- Team2_RoundsFirstHalf INTEGER
- Team2_RoundsSecondHalf INTEGER
- Team2_RoundsOT INTEGER
- Team1_PistolWon INTEGER
- Team1_Eco INTEGER
- Team1_EcoWon INTEGER
- Team1_SemiEco INTEGER
- Team1_SemiEcoWon INTEGER
- Team1_SemiBuy INTEGER
- Team1_SemiBuyWon INTEGER
- Team1_FullBuy INTEGER
- Team1_FullBuyWon INTEGER
- Team2_PistolWon INTEGER
- Team2_Eco INTEGER
- Team2_EcoWon INTEGER
- Team2_SemiEco INTEGER
- Team2_SemiEcoWon INTEGER
- Team2_SemiBuy INTEGER
- Team2_SemiBuyWon INTEGER
- Team2_FullBuy INTEGER
- Team2_FullBuyWon INTEGER

Table: **Matches**

- MatchID TEXT
- Date TEXT

- Patch TEXT
- EventID TEXT
- EventName TEXT
- EventStage TEXT
- Team1ID INTEGER
- Team2ID INTEGER
- Team1 TEXT
- Team2 TEXT
- Team1_MapScore INTEGER
- Team2_MapScore INTEGER