ValoPlanner - An Intelligent Assistant for VALORANT Team Planning

Daixin Tian
University of Toronto
ali.daixin.tian@gmail.com

Derui Yang
University of Toronto
deruiyang0109@gmail.com

Kefan Li
University of Toronto
m13913013303@gmail.com

Mingjun Zhong
University of Toronto
zhongadiao12@gmail.com

Abstract: This report details the development of an AI agent, ValoPlanner, designed specifically for VALORANT professional and semi-professional games. The agent features a Team Planner tool that aids users in building and evaluating teams based on performance, synergy, and strategy. Additionally, it includes a Fun Fact feature to engage users by providing interesting tidbits related to VCT. We outline our methodologies for data collection and processing, the integration of external data sources, and the deployment of the chatbot. Through rigorous evaluation and testing, we are committed to ensuring that our AI meets user needs and delivers valuable insights into player performance and team dynamics.

1. Introduction

Our project, ValoPlanner, is developing an AI agent specifically designed for professional and semi-professional VALORANT games. Our AI offers a Team Planner tool to help users build and evaluate teams based on performance, synergy, and strategy, making it easier to optimize their rosters. Additionally, for a bit of fun and engagement, we've included a Fun Fact feature that shares interesting tidbits related to VALORANT, keeping users informed and entertained.

In this report, we will introduce the development of our projects from data, methodology, tooling, and aesthetic aspects.

2. Data Processing

This section outlines how we collected, processed, and integrated official VCT data and external sources to build a comprehensive dataset for ValoPlanner.

2.1 AWS S3 Dataset

Devpost and AWS provide resources stored in AWS S3 buckets along with official documentation to manage the large volume of data required for our project.

2.1.1 Data Downloading

The team downloaded data based on the code provided by the tournament officials. Since the data size is very large, which is more than 60GB, the team used the group aws s3 database as the container.

2.1.2 Data Analyze

The team analyzed data from tournament officials, using player, team and league mapping data to collect player basic information including team, region, league level and corresponding icons. The team is selective on game data, which in further steps will feed to the model. The team chooses game data including game config (including participant team, players, map and agent choice, etc), which is used as basic game information; player death and damage event (damage, victim and dealer), which can be used to calculate players' KD, headshot rate and other performances; round and game decided (win and loss, win method), which is used to calculate the winning rate for both players. agents and maps. These are the main information the team used for the core functions of the project. Other information is also collected as development resources.

2.1.3 Data processing

The team downloaded the data separately, and as one file finished unzipping and uploaded to the internal s3 bucket, the file will be deleted locally. The team runs the bucket twice, one round looping all mapping data to generate CSV lookup tables of players' identity data, and another round, which takes the majority of the time, looping all games, accessing tournament and game mappings for basic information, and generating performance lookup tables.

Performance lookup tables include information not just limited to players' KDA, headshot rate, damage per round and more on their performance, agent win rate for each player and each map.

2.2 External Sources

There are two allowed external sources. One is from VALORANT fandom, the other is vlr.gg. We examined both data sources and tried to look for a supplement to Riot's official data. We went through the official data and found out that, for a well-rounded esport chatbot, we need more structured information and VCT knowledge.

2.2.1 VALORANT Fandom Data Assessment

Data from VALORANT fandom and esport fandom is quite messy. Most information is embedded in XML format, plus the files are quite messy and hard to examine. Also, most of its information is included in either the official dataset or vlr.gg. So we decided not to put this dataset into usage.

2.2.2 vlr.gg Data Utilization

The data from VLR.gg has proven instrumental in building the chat-bot, particularly in two key areas: player and game information, as well as recent news articles on VALORANT esports. Our analysis of the website revealed that it offers comprehensive and well-organized information, making it effective for data collection. For instance, the "news" subpage provides a complete list of VALORANT esports news articles, while the "stats" subpage offers an extensive list of VALORANT players, excluding the "CN" region. We prioritized the most recent news articles and selected 5,000 out of 47,500 players based on metrics like rounds played and OPP Rating.

For news data, we crawled the article subpages, extracting the body, author, and publication date of each article. These articles were then stored in Amazon S3 and transformed into vector embeddings using Titan Embedding V2, enabling the chat-bot to retrieve and interpret the most relevant news content related to VALORANT esports. This adds an additional layer of contextual evidence for decision-making, beyond basic numerical data.

Regarding player data, we focused on 5,000 players with dedicated VLR.gg subpages. We collected information on performance, agent usage, team affiliations, and match histories. This data was then integrated with official player data, which offers different dimensions such as player region and league level. All gathered information was consolidated and linked to an Amazon Lambda function, allowing the chat-bot to efficiently query and provide insights into player performance.

2.3 External Sources and Official Sources Matchup

The team checks on both players' full names and acronyms to match up players. There are around 5000 players from official sources and 40000 plus players from external sources. The final matched players are about 3000. The team filtered out players that lack data or only with outdated data, in order to get a dataset with comparatively active players.

3. Methodology

This section outlines the system design, data collection, preprocessing, integration, feature engineering, and deployment methods used to develop the ValoPlanner. Each phase focuses on ensuring efficient data handling, intelligent responses, and scalability through cloud infrastructure. In addition, front-end design features and deployment approaches will be introduced.

3.1 System Design

The development of the VALORANT esports chatbot involves a multi-step process centred around data acquisition, integration, modelling, and optimization. The system architecture is built on cloud infrastructure, primarily leveraging AWS services, to ensure scalability and efficiency. The key components include a backend for data processing, a database for structured storage, a frontend interface for user interaction, and an AI layer for intelligent responses and recommendations. The system uses a serverless architecture powered by AWS Lambda, facilitating real-time processing and response.

3.2 Data Collection and Preprocessing

The initial step involves collecting data from multiple sources, primarily Riot's official datasets and VLR.gg, supplemented by crawled news articles. Data extraction involves both structured data (player and match statistics) and unstructured data (news articles). For structured data, the team employs a code provided by tournament officials to access large datasets stored in AWS S3, which contain over 60GB of information. The data is then unzipped, filtered, and cleaned to ensure the inclusion of high-quality data relevant to professional and semi-professional games. Preprocessing is done in two phases:

3.2.1 Mapping Data

In this phase, all mapping information related to players, teams, leagues, and regions is extracted to create CSV lookup tables. These tables serve as the foundation for linking players' identities with performance metrics, enabling accurate analysis.

3.2.2 Game Data Analysis

The team focuses on extracting relevant match data, such as player and team information, map configurations, agent choices, and in-game events (kills, deaths, damage, etc.). Specific metrics like Kill/Death/Assist (KDA), headshot rate, damage per round, and win rate for players, agents, and maps are computed and stored in performance lookup tables. The data pipeline is designed to handle continuous updates, ensuring the system remains aligned with the latest matches and tournaments.

3.2.3 Data Integration

To create a comprehensive dataset, the information from VLR.gg is integrated with the official Riot data. This is achieved through data matching techniques that reconcile inconsistencies, such as different naming conventions or missing data. Players are matched based on identifiers like full names and acronyms. The team filters the data to retain only active players with sufficient data points, ensuring the accuracy and relevance of the AI model's outputs.

3.3 Feature Engineering

Feature engineering is crucial for both player performance evaluation and the Team Planner tool. Key features include:

1. ***Player Statistics***: Metrics like KDA, headshot rate, and damage per round are extracted from both Riot's official data and VLR.gg. These statistics are used to rank players and suggest optimal team compositions.
2. ***Agent Synergy and Strategy***: The algorithm calculates agent statistics for different players including performance score and detailed event percentage rate , allowing the Team Planner tool to assess potential team synergies and recommend strategic changes.
3. ***News Embeddings***: Using Titan Embedding V2, the team transforms news articles into vector embeddings, enabling the chat-bot to retrieve and present the most relevant news and updates related to specific players or events.

3.4 Knowledge Base

To implement RAG, we require a knowledge base based on data from vlr.gg news articles. Several aspects are taken into consideration. For the embedding model, we applied Amazon Titan V2.

3.4.1 Fixed Size Chunk

This is the default chunking method provided by Amazon. Data is split into fixed size of 300 for embedding. This method is efficient but lacks news metadata such as news title and published date, which could be a problem considering the limited time frame for news information.

3.4.1 No Chunk

No chunk technique basically uses articles as chunks, each article serves as an individual chunk with completed information. This leads to problems of over-sized chunks, which is not practical under the Amazon environment.

3.4.1 Pre-processed Chunk

We combine the above two methods through pre-split all articles into chunks, while maintaining their metadata as an article. These completed chunks are directly fed into embedding models for title and date reference.

3.5 Logic & Reasoning

In order to maximize agent performance, we experienced various techniques on agent reasoning for team formation. Three main methods are brought up and tested.

3.5.1 Direct Inference

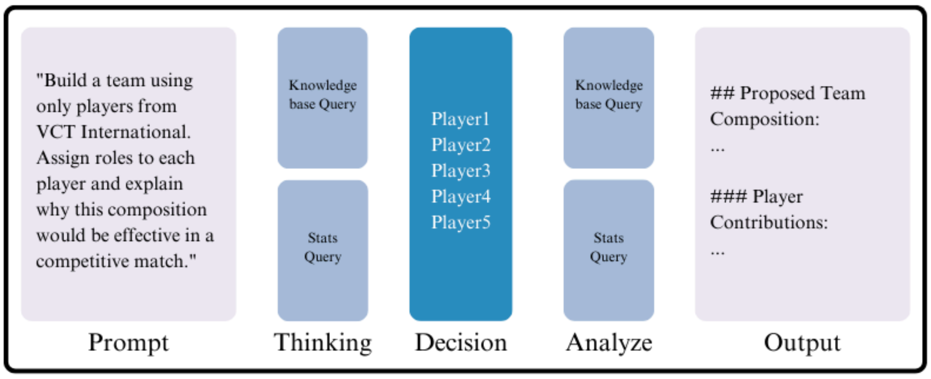

"Build a team using only players from VCT International. Assign roles to each player and explain why this composition would be effective in a competitive match."
Knowledge base Query
Stats Query
Player1
Player2
Player3
Player4
Player5
Knowledge base Query
Stats Query
## Proposed Team Composition:
...
### Player Contributions:
...
Prompt
Thinking
Decision
Analyze
Output

Figure 1: Architecture of Direct Inference

Direct inference basically instructs the agent to generate the team formation all at once, with sufficient reasoning before and after. We abandoned this method at the end for its instability and unreasonable player choices.

3.5.2 Self-revising

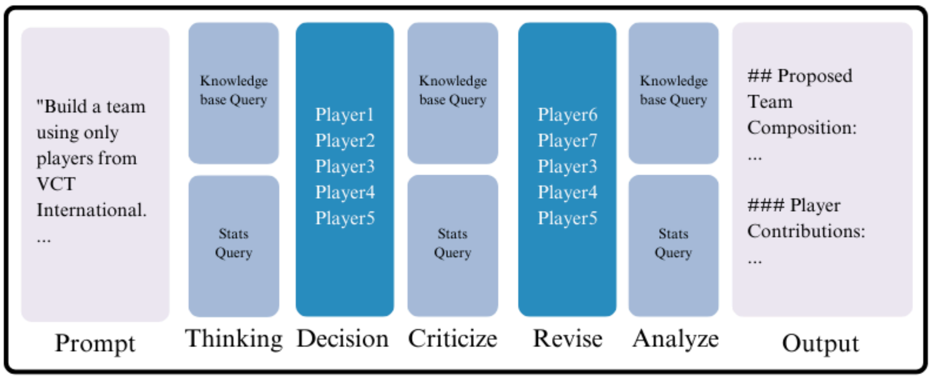

"Build a team using only players from VCT International. ...
Prompt
Knowledge base Query
Stats Query
Thinking
Player1 Player2 Player3 Player4 Player5
Decision
Knowledge base Query
Stats Query
Criticize
Player6 Player7 Player3 Player4 Player5
Revise
Knowledge base Query
Stats Query
Analyze
## Proposed Team Composition: ...
### Player Contributions: ...
Output

Figure 2: Architecture of Self-Revising Inference

Self-revising method improves agent performance on selecting more appropriate players for team formation. Decisions are now carefully made with multi-layer thinking and analysis. However, drawbacks are found for algorithm complexity. It is difficult for the base model to understand the completed procedure of self-revising, or it might require a more in-depth prompting or fine-tuning.

3.5.3 Multi-step Inference

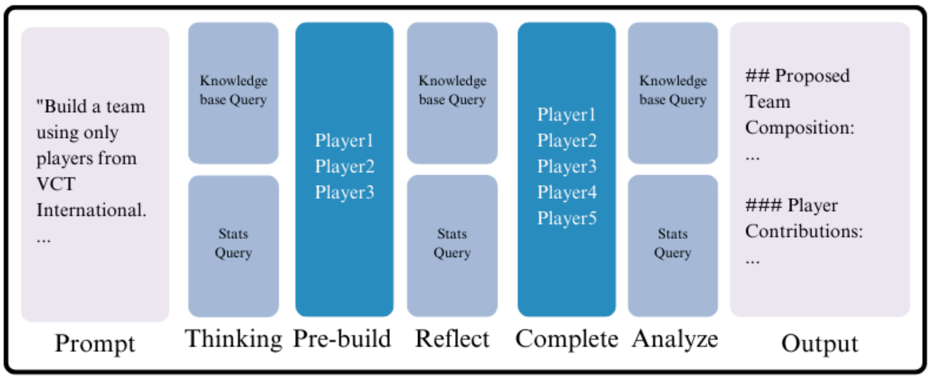


Figure 3: Architecture of Multi-step Inference

Multi-step inference is the pipeline we are applying for our final submission. It drastically improves model understanding on tasks and remains the strength of careful player choices from self-revising. Moreover, it has unignorable advantages in complex downstream tasks, such as mix-gender and cross-regional tasks.

3.6 User Interaction

The front-end interface of the chatbot is designed to be user-friendly, enabling users to interact with the AI through intuitive commands. It supports the following three primary functions:

1. ***Chatbot:*** The chatbot's responses are tailored to user inputs, providing specific insights based on the context of the query, whether it's player analysis, team building, or general VALORANT knowledge. The use of embeddings and vector search ensures that the AI delivers the most relevant information, enhancing the overall user experience.
2. ***Team Planner***: Users can input potential team compositions, and the AI will analyze the lineup based on the latest performance data. The AI will also suggest changes to enhance team synergy, considering factors like agent strengths, map preferences, and player compatibility.
3. ***Fun Fact Feature***: This feature uses embedded news articles to provide users with interesting trivia about VALORANT, players, teams, or recent tournament results, adding an element of engagement to the experience.

3.7 Deployment

The deployment of the VALORANT esports chatbot is built on a robust cloud infrastructure using AWS services to ensure reliability and scalability. For automatic deployment, the system leverages AWS EC2, CodePipeline, and CodeDeploy, enabling a seamless integration and deployment process.

AWS EC2 is used to host the backend services, including the Flask application and the database that powers the chat-bot stored in S3. The EC2 instances are configured to scale as needed, providing flexibility to handle increased traffic during major VALORANT tournaments or events.

Flask serves as the web framework for building the REST APIs that connect the front-end and back-end. It enables the chatbot to process user interactions efficiently and respond with relevant information based on the underlying data and AI models.

AWS CodeDeploy and CodePipeline sets up an automation of the entire release process. The pipeline is triggered automatically whenever there is a new change pushed to the Github repository.

4. Tooling

AWS provided crucial support for our VALORANT esports chat-bot, offering scalable cloud infrastructure to manage large datasets and process information in real time. The integration of AWS tools ensured smooth development and reliable performance throughout the project. Below, we highlight the key tools that made this possible.

4.1 Bedrock with Lambda and Serverless

As required by the competition, this agent service is supported by Amazon Bedrock. According to the methodology description, we introduced Amazon Bedrock Agent (ABA) for high-quality team formation outputs. In order to get access to various information from the dataset and server regarding VALORANT players and stats, action groups and knowledge bases are attached to the agent itself. The knowledge base is held on Amazon Serverless Computing for Retrieval Augmented generation, meanwhile, Amazon Lambda provides functionality to directly query specific data pieces regarding players or agents from the main database. Both services are connected with Amazon S3, which holds most of our processed resources.

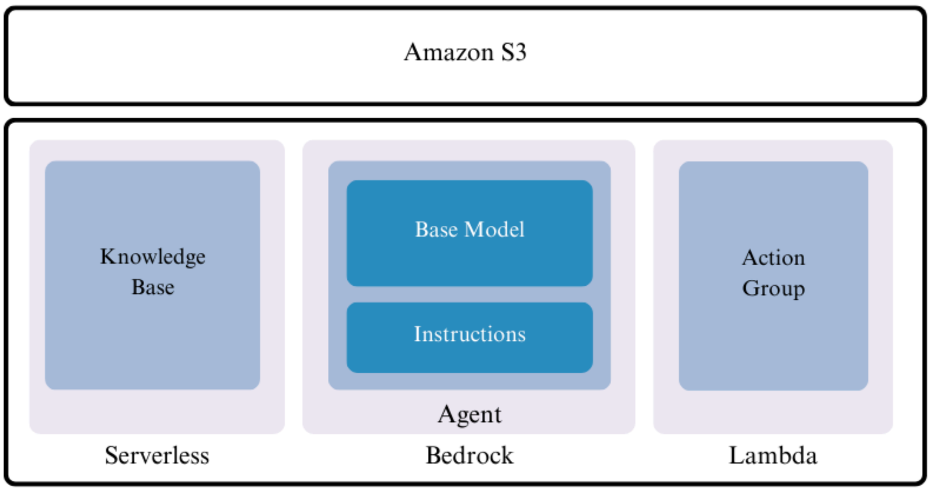


Figure 4: Architecture of usage of AWS related to model and backend

4.2 EC2 & CodePipeline

In line with Section 3.6 Deployment, AWS EC2 and CodePipeline played an essential role in automating and managing our ValoPlanner deployment. EC2 hosted the backend services, ensuring that the system scaled effectively during peak usage, especially during major VALORANT events.

AWS CodePipeline provided continuous integration and delivery (CI/CD) capabilities, automatically triggering deployments whenever new changes were pushed to the GitHub repository. This streamlined the release process, allowing us to quickly implement updates.

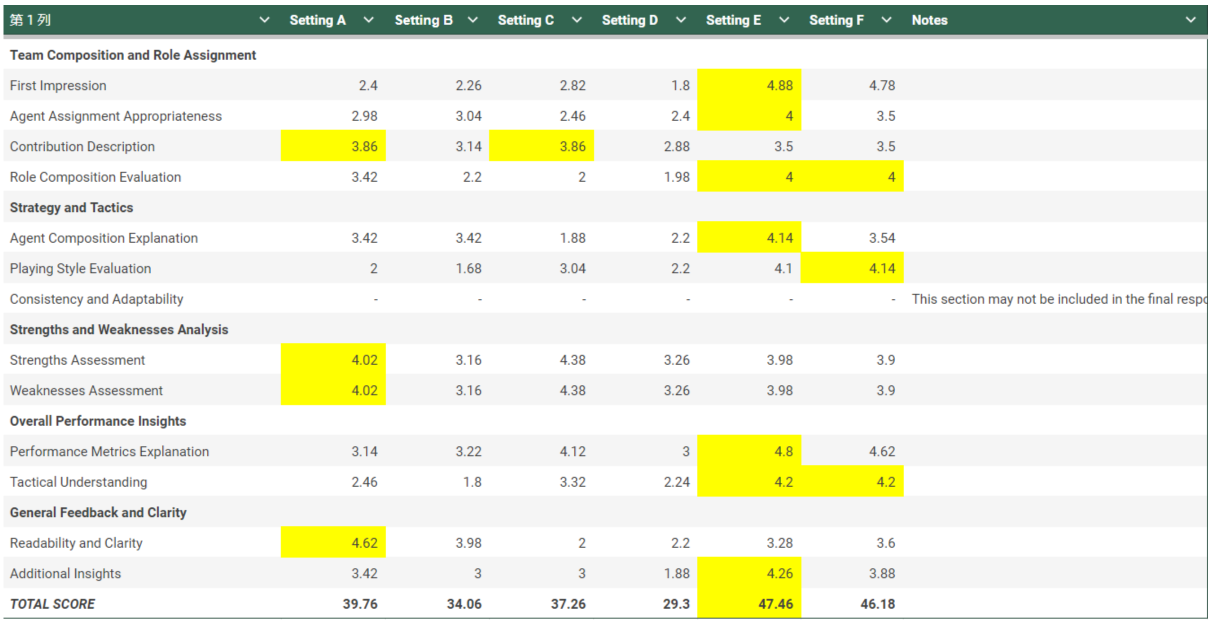
5. Evaluation and Testing

After experimenting with different combinations of models and prompting, we have finalized six distinct solution settings for our agent. By controlling all other variables, we define solution settings by varying inference logic as stated in Section 3.4 Logic & Reasoning and model selection from Claude 3 Sonnet and Claude 3 Haiku.

For our definition, setting A is direct inference and Sonnet; setting B is direct inference and Haiku; setting C is self-revising inference and Sonnet; setting D is self-revising inference and Haiku; setting E is multi-step inference and Sonnet; setting F is multi-step inference and Haiku.

Based on the responses generated by different settings, we decided to conduct an experiment using a questionnaire targeted at 50 VALORANT players, which will assess how well each setting performs across various aspects of the responses. The questionnaire is shown in *Appendix A and B*. By gathering feedback directly from end users, we believe that the results would provide convincing and reliable insights that will guide us in making the final decisions on the settings, ensuring it meets user needs and enhances its professional and convincing reasoning in its responses.

The result of an average of 50 questionnaires is shown in the table below in Figure 5, along with a bar chart for visualization in Figure 6. Eventually, we decided to use the setting E, multi-step inference with Claude 3 Sonnet model, because it gets the highest score and is the most reliable one to end users.



第 1 列	Setting A	Setting B	Setting C	Setting D	Setting E	Setting F	Notes
Team Composition and Role Assignment							
First Impression	2.4	2.26	2.82	1.8	4.88	4.78	
Agent Assignment Appropriateness	2.98	3.04	2.46	2.4	4	3.5	
Contribution Description	3.86	3.14	3.86	2.88	3.5	3.5	
Role Composition Evaluation	3.42	2.2	2	1.98	4	4	
Strategy and Tactics							
Agent Composition Explanation	3.42	3.42	1.88	2.2	4.14	3.54	
Playing Style Evaluation	2	1.68	3.04	2.2	4.1	4.14	
Consistency and Adaptability	-	-	-	-	-	-	This section may not be included in the final resp
Strengths and Weaknesses Analysis							
Strengths Assessment	4.02	3.16	4.38	3.26	3.98	3.9	
Weaknesses Assessment	4.02	3.16	4.38	3.26	3.98	3.9	
Overall Performance Insights							
Performance Metrics Explanation	3.14	3.22	4.12	3	4.8	4.62	
Tactical Understanding	2.46	1.8	3.32	2.24	4.2	4.2	
General Feedback and Clarity							
Readability and Clarity	4.62	3.98	2	2.2	3.28	3.6	
Additional Insights	3.42	3	3	1.88	4.26	3.88	
TOTAL SCORE	**39.76**	**34.06**	**37.26**	**29.3**	**47.46**	**46.18**	

Figure 5: Summary of average scores of questionnaire on different settings

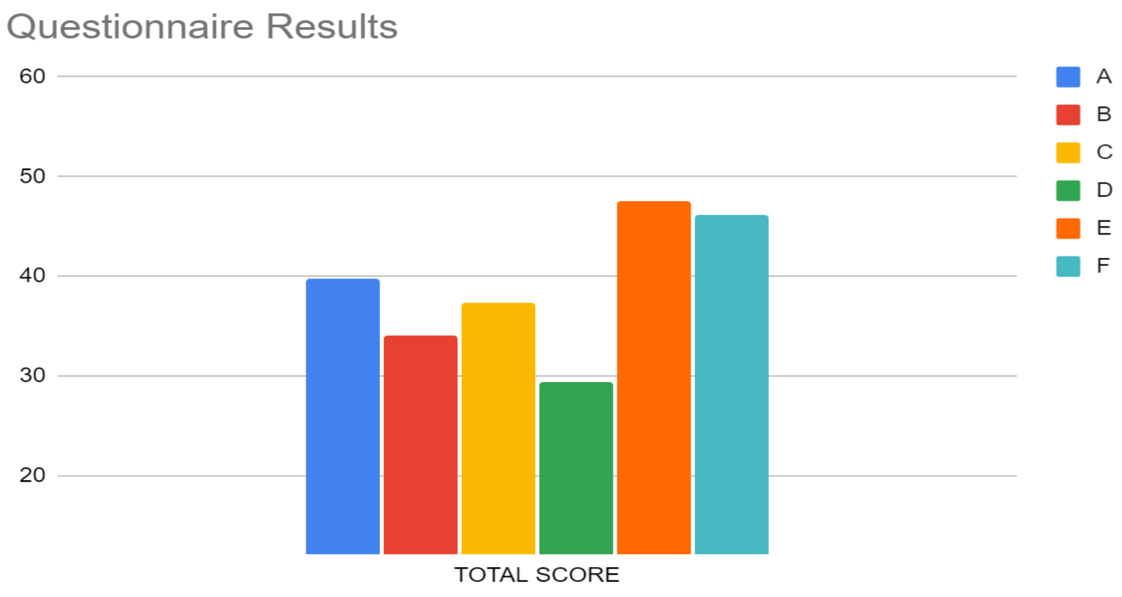


Figure 6: Bar chart of average scores of questionnaire on different settings

6. Aesthetics of Chatbot Interface Design

In this section, I will outline the approach taken to design the front-end of the ValoPlanner project. The design focuses on two key areas: layout and user experience. By implementing a responsive, grid-based structure using Bootstrap's 12-column layout, we ensured seamless adaptability across various devices. The typography, featuring the Roboto font family, enhances readability and visual hierarchy. Additionally, the color palette is inspired by Valorant’s theme, providing high contrast for accessibility. The logo design leverages Gestalt principles, particularly the law of closure, to create a visually distinct and user-friendly interface.

6.1 Layout

In designing the front end for ValoPlanner, I adhered to a standard 12-column layout, leveraging the Bootstrap grid system to ensure flexibility and responsiveness across various screen sizes. Bootstrap’s grid system allows for seamless management of page structure, providing breakpoints that adjust content dynamically depending on the device size. This flexibility was crucial in optimizing the user experience on different devices, from desktops to tablets and smartphones.

The grid system is implemented with container-fluid, ensuring that the layout adjusts fluidly to the entire viewport width. The sidebar is assigned a fixed 2-column width, allowing for easy navigation while keeping the primary content space larger and adaptable to user interactions. This layout structure also utilizes offset classes to ensure balanced spacing and alignment, particularly for the chat window and player card sections. The layout dynamically adjusts when the player cards are expanded, taking up additional space while preserving the intuitive navigation and interaction flow for users.

In terms of typography, the project predominantly uses the Roboto font family, chosen for its modern and clean appearance. By incorporating Roboto with a variation of font weights, we maintain visual hierarchy and clarity, emphasizing important elements like headings, user messages, and call-to-action buttons. The typographical choices help ensure that information is both accessible and aesthetically pleasing.

6.2 User Experience and Accessibility

For the color palette, the primary theme revolves around VALORANT's core color (#fd4556), a bold and energetic red. This is balanced with secondary and neutral tones, such as whites and grays, to ensure contrast and readability. This contrast aids in keeping the UI visually engaging yet not overwhelming. To ensure optimal accessibility, the contrast ratio between text and background was tested to comply with WCAG (Web Content Accessibility Guidelines) standards. This guarantees that users with visual impairments or those viewing on different screen conditions can still comfortably interact with the application.

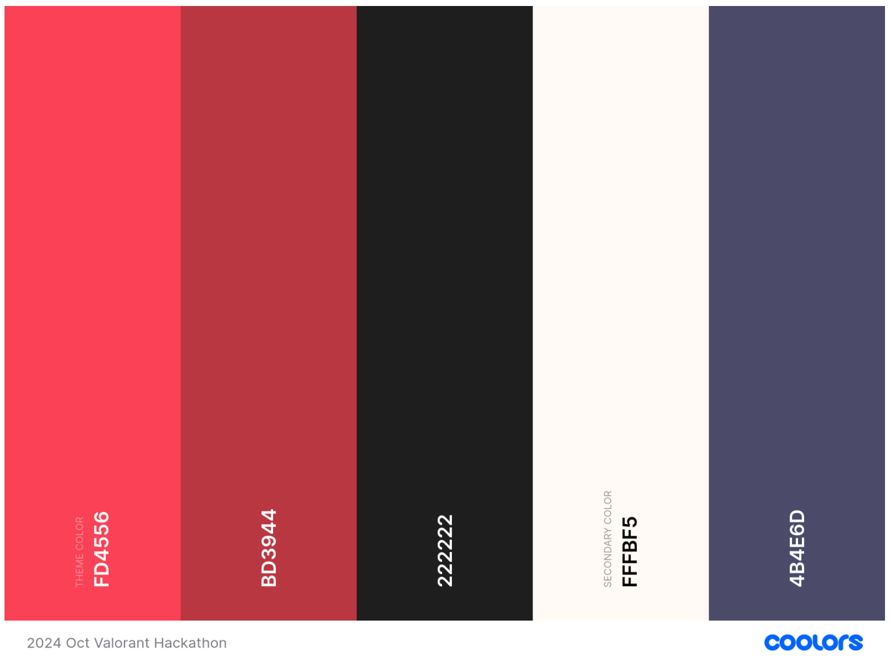


Figure 7: Color Palette of ValoPlanner

The logo design applies the law of closure from Gestalt principles, which plays an essential role in human visual perception. The "V" and "P" in "ValoPlanner" are designed to stand out prominently, using negative space and minimalistic outlines. This subtly emphasizes the brand’s identity without overwhelming the viewer. The clean, minimalist style also contributes to the overall simplicity and elegance of the interface.

Lastly, the layout and color scheme were chosen to ensure consistency throughout the site, improving the overall user experience. The input fields, buttons, and navigational elements are designed with clear, visible states for hover, focus, and disabled conditions. This is crucial for ensuring that the interface remains intuitive and accessible, allowing users to interact effortlessly across all components.

7. Remarks

This section highlights key insights and challenges encountered during the development of the VALORANT esports chatbot. It also reflects on the team's decisions regarding data handling and response streaming, along with the solutions explored.

7.1 Response Streaming

An issue occurred during our development, we planned to make a stream response from the model to improve user experience. However, as we researched, an expert from AWS commented that: "... the final response is typically a product of multiple LLM calls, often chained together so the output of one was used in the input to the next. This substantially reduces the value of response streaming for agents versus the plain LLM-calling … only the last generation in the agent flow can be meaningfully streamed so the client is still waiting with no output through the intermediate steps". Therefore, we gave up the idea of using stream response though we could try to add a layer of using Lambda's response payload streaming feature.

7.2 Backend Code Names

Remarkably, data from s3 are using backend code names. The team uses chatGPT to find out what these code names mean, as shown in picture 1. The team gets code names for maps in the same way. Details can be found in Appendix C.

8. Future Work

As we continue to enhance the VALORANT esports chatbot, several areas have been identified for future development. By focusing on these areas, we aim to refine the accuracy and reliability of our analyses, ultimately enhancing the user experience and the effectiveness of the chatbot in providing valuable insights into player performance and team strategies.

8.1 Data Compensation for Unmatched Players

Currently, some players do not have their information matched in our dataset. That is, some players could be found in vlr.gg but cannot be found in official documents provided by AWS or the other way around. To address this gap, we plan to implement a data compensation strategy that utilizes additional data sources and heuristics. This may include cross-referencing with alternative databases, and leveraging player performance metrics from previous tournaments to ensure that all players can be represented within the system.

8.2 Handling Inactive Players

A significant challenge arises from players who lack any match history within the recent 60 days (data provided by vlr.gg), impacting our ability to analyze their performances and team dynamics effectively. To improve our analysis, we will explore methods to incorporate historical performance data for these players, allowing us to infer trends and agent preferences.

9. Conclusion

In conclusion, our AI agent for VALORANT is built on a robust backend that integrates data from official AWS documents and third-party sources vlr.gg. We dedicated significant effort to processing and refining this data to ensure it is valuable and reliable for team formation.

The selection of Claude 3 Sonnet as our AI model and our instruction/prompting provided to the model as well as agent inference logistics was crucial, supported by extensive prompt engineering and user feedback through a questionnaire experiment. By leveraging advanced AI techniques and comprehensive datasets, our ValoPlanner effectively gives reasonable team formation, provides strong evidence and explanations, evaluates player performance and identifies optimal team strategies.

As we continue to iterate on our design, more comprehensive analysis will be done on the data such that unmatched players and inactive players will receive better representation and insights in our ValoPlanner.

10. Acknowledgments

10.1 Amazon Web Services:

Information and insights were sourced from the official AWS documentation and community discussions available at AWS, which provided valuable context for implementing streaming responses.

10.2 Bedrock Agent Streaming:

The details regarding streaming responses from the Bedrock agent were adapted from a discussion on the AWS forums, specifically found at AWS Repost.

10.3 Riot Games:

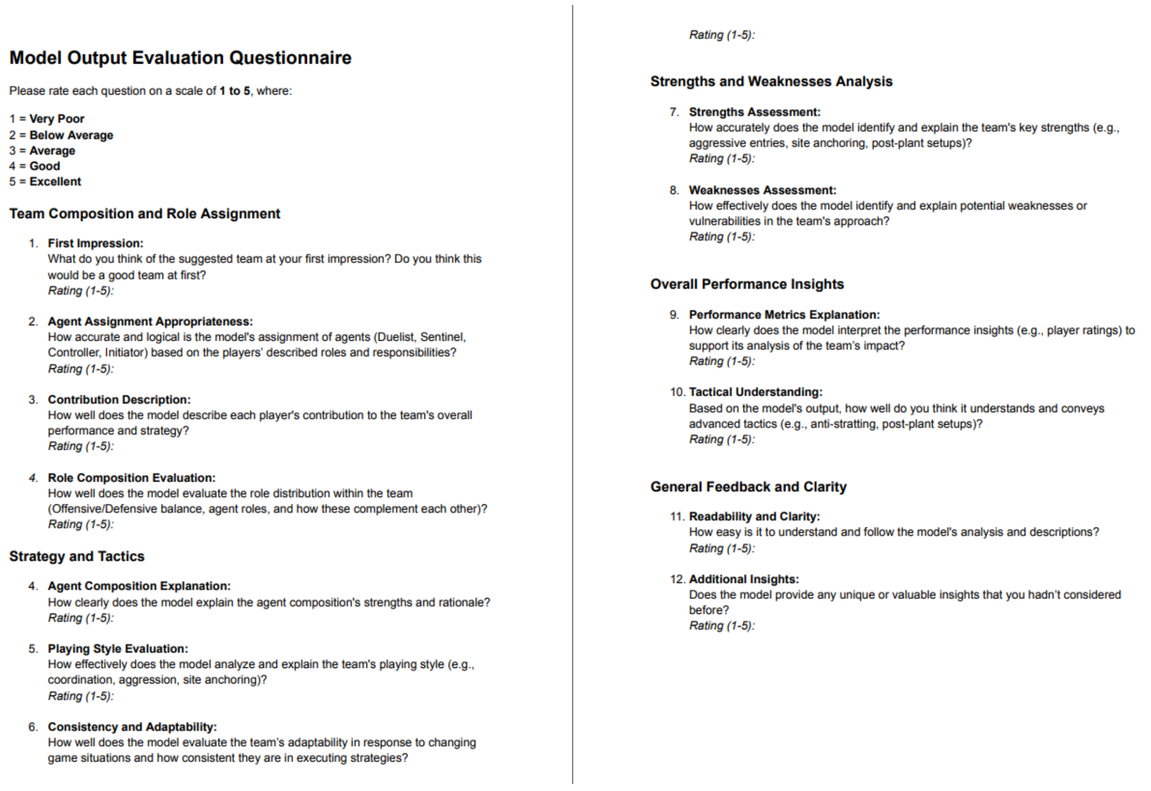
Game mechanics and tournament structures related to VALORANT were informed by resources and updates provided by Riot Games, particularly regarding the VALORANT Champions Tour (VCT).

10.4 VLR.gg:

Competitive data and statistics were referenced from VLR.gg, which is a comprehensive resource for VALORANT esports analytics and community insights.

11. Appendices

A. Questionnaire for evaluating our model setting performance



Model Output Evaluation Questionnaire

Please rate each question on a scale of **1 to 5**, where:

1 = **Very Poor**
2 = **Below Average**
3 = **Average**
4 = **Good**
5 = **Excellent**

Team Composition and Role Assignment

1. **First Impression:**
   What do you think of the suggested team at your first impression? Do you think this would be a good team at first?
   Rating (1-5):

2. **Agent Assignment Appropriateness:**
   How accurate and logical is the model's assignment of agents (Duelist, Sentinel, Controller, Initiator) based on the players' described roles and responsibilities?
   Rating (1-5):

3. **Contribution Description:**
   How well does the model describe each player's contribution to the team's overall performance and strategy?
   Rating (1-5):

4. **Role Composition Evaluation:**
   How well does the model evaluate the role distribution within the team (Offensive/Defensive balance, agent roles, and how these complement each other)?
   Rating (1-5):

Strategy and Tactics

4. **Agent Composition Explanation:**
   How clearly does the model explain the agent composition's strengths and rationale?
   Rating (1-5):

5. **Playing Style Evaluation:**
   How effectively does the model analyze and explain the team's playing style (e.g., coordination, aggression, site anchoring)?
   Rating (1-5):

6. **Consistency and Adaptability:**
   How well does the model evaluate the team's adaptability in response to changing game situations and how consistent they are in executing strategies?
   Rating (1-5):

Strengths and Weaknesses Analysis

7. **Strengths Assessment:**
   How accurately does the model identify and explain the team's key strengths (e.g., aggressive entries, site anchoring, post-plant setups)?
   Rating (1-5):

8. **Weaknesses Assessment:**
   How effectively does the model identify and explain potential weaknesses or vulnerabilities in the team's approach?
   Rating (1-5):

Overall Performance Insights

9. **Performance Metrics Explanation:**
   How clearly does the model interpret the performance insights (e.g., player ratings) to support its analysis of the team's impact?
   Rating (1-5):

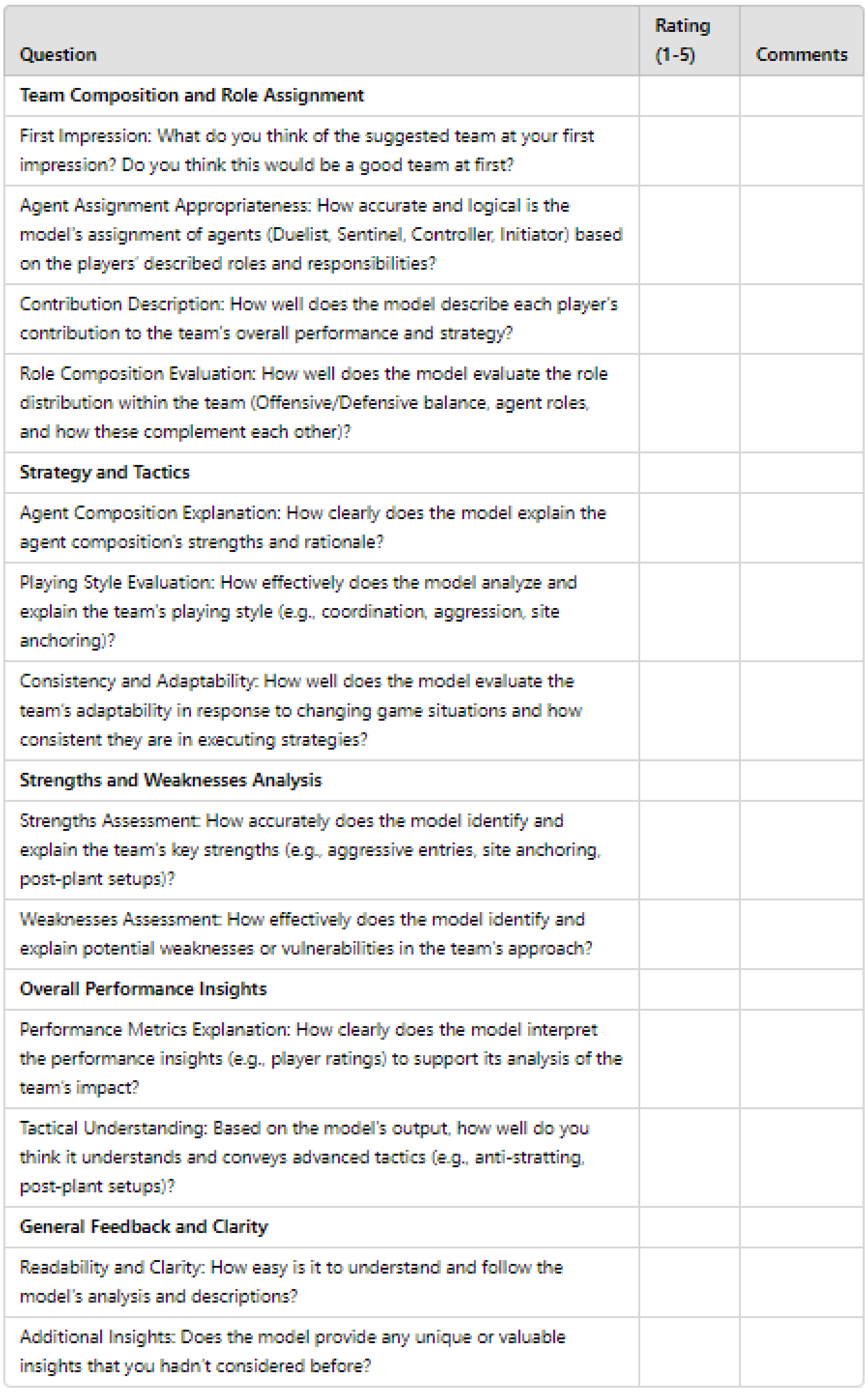
10. **Tactical Understanding:**
    Based on the model's output, how well do you think it understands and conveys advanced tactics (e.g., anti-stratting, post-plant setups)?
    Rating (1-5):

General Feedback and Clarity

11. **Readability and Clarity:**
    How easy is it to understand and follow the model's analysis and descriptions?
    Rating (1-5):

12. **Additional Insights:**
    Does the model provide any unique or valuable insights that you hadn't considered before?
    Rating (1-5):

B. Questionnaire details



Question	Rating (1-5)	Comments
Team Composition and Role Assignment		
First Impression: What do you think of the suggested team at your first impression? Do you think this would be a good team at first?		
Agent Assignment Appropriateness: How accurate and logical is the model's assignment of agents (Duelist, Sentinel, Controller, Initiator) based on the players' described roles and responsibilities?		
Contribution Description: How well does the model describe each player's contribution to the team's overall performance and strategy?		
Role Composition Evaluation: How well does the model evaluate the role distribution within the team (Offensive/Defensive balance, agent roles, and how these complement each other)?		
Strategy and Tactics		
Agent Composition Explanation: How clearly does the model explain the agent composition's strengths and rationale?		
Playing Style Evaluation: How effectively does the model analyze and explain the team's playing style (e.g., coordination, aggression, site anchoring)?		
Consistency and Adaptability: How well does the model evaluate the team's adaptability in response to changing game situations and how consistent they are in executing strategies?		
Strengths and Weaknesses Analysis		
Strengths Assessment: How accurately does the model identify and explain the team's key strengths (e.g., aggressive entries, site anchoring, post-plant setups)?		
Weaknesses Assessment: How effectively does the model identify and explain potential weaknesses or vulnerabilities in the team's approach?		
Overall Performance Insights		
Performance Metrics Explanation: How clearly does the model interpret the performance insights (e.g., player ratings) to support its analysis of the team's impact?		
Tactical Understanding: Based on the model's output, how well do you think it understands and conveys advanced tactics (e.g., anti-stratting, post-plant setups)?		
General Feedback and Clarity		
Readability and Clarity: How easy is it to understand and follow the model's analysis and descriptions?		
Additional Insights: Does the model provide any unique or valuable insights that you hadn't considered before?		

C. Backend Code Names

```
Split - Bonsai
Haven - Triad
Bind - Duality
Icebox - Port
Breeze - Foxtrot
Fracture - Canyon
Pearl - Pitt
Ascent - Venice
Lotus - Jam
```