RTS Analyzer

Software Engineers:

- *Cody Strange – ETL Developer*
- *Jaden Albrecht – Database Administrator*
- *Hoyoung Kim – Server-Side Developer*

Institution: Utah Valley University

Course: CS4550-601 2024 Spring

Abstract

The RTS Analyzer is a comprehensive software tool developed by students at Utah Valley University for the CS4550-601 course in Spring 2024. This application is designed to process and analyze StarCraft II replay data to extract and display strategic gameplay metrics such as build orders and win rates. Utilizing technologies like Django for backend development and Flutter for frontend interface design, the RTS Analyzer aims to enhance real-time gaming strategies by allowing users to overlay build orders onto live game scenarios. The tool leverages a Python module, sc2reader, for accessing replay data, and implements rigorous coding standards to ensure robustness and maintainability. The project integrates advanced data processing techniques to determine similarities between player actions and predefined optimal build orders, offering a significant tactical advantage in gameplay. The application is structured to meet high standards of performance, scalability, and security, addressing both functional and non-functional requirements meticulously. This paper will discuss the design, development, and operational strategy of the RTS Analyzer, showcasing its potential to revolutionize game strategy development and analysis.

Acknowledgements (from Cody Strange)

I extend my heartfelt thanks to several professors whose guidance was instrumental not only in shaping the design of this project but also in providing the foundational knowledge that made it possible.

- **Professor Reza Senati**: His review and advice on our ER Diagram and database schema were invaluable, as were the lessons in his Database Theory class that introduced me to relational databases. Our collaborative work on a database research paper bolstered my confidence and ability to implement these designs effectively.

- **Professor Lynn Thackery**: He meticulously reviewed our UML, data flow, and architecture diagrams, ensuring we could properly design and document our entire project. His teachings in the Principles and Software Patterns class, especially the principle "Program to an interface, not an implementation," have been the cornerstone of this project's maintainability and scalability. His influence has significantly reduced the time I spend debugging, making this complex project the smoothest I have undertaken.

- **Professor Craig Sharp**: From his Software Engineering One class, I learned the crucial importance of design and documentation in software projects. His class not only deepened my love for software engineering but also inspired me to switch my major from Computer Science to Software Engineering, marking a pivotal point in my academic and professional life.

- **Professor Brian Knaeble**: He provided critical feedback on the algorithm for determining user build orders. While I have not yet incorporated all his recommendations, they are slated for future implementation. His Analyzing Algorithms class transformed my problem-solving approach and sparked my interest in enhancing my mathematical skills.

- **Professors Frank Jones, JP Tang, and Peter Aldous**: Each of them imparted valuable lessons and insights that have been integral to my development.

My experiences with these educators have profoundly shaped my approach to software development. I am deeply grateful for their instruction and support, which have left an indelible mark on my educational journey.

Organization

Coding Standards

Commenting

Functions

- Brief description of function
- Parameters
- Return Values

Classes

- Brief description of class

ChatGPT

- Highly recommend to let ChatGPT do most of the commenting
- Double check any comments by ChatGPT

Programming Paradigm

- Object oriented

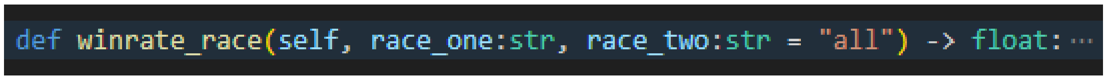
Naming Conventions

- Classes: CapWords
- Functions: snake_case
- Variables: snake_case
- Constants: ALLCAPS
- Files/Folders: snake_case

Code Formatter

- Run all python files through black

Type Safety

- Every function should have the parameters types listed and the return type of the function listed
- 

```python
def winrate_race(self, race_one:str, race_two:str = "all") -> float:···
```

Software Requirements

Data Collection and Processing

Sc2reader

- Utilize python module for accessing SC2 replay data.

Backend Development

Python 3.12

- Our programming language of choice

Black

- A python code formatter

Pytest

- Used for python unit testing

SQL Alchemy

- For storing replay data
- For storing build order data

Frontend Development

Flutter

- Used for building dynamic and responsive user interfaces.

Tkinter

- Used for overlaying build orders to user screen

Version Control

GitHub

- Used to store project, allows for collaboration and version control

Communication

Discord

- Used for meetings and messaging

Microsoft Teams

- Used for meetings

Message App

- Used messaging

Task Organization

Trello

- Used to track tasks and progress on the project

Documentation

Word

- Used to document and organize the process of creating and maintaining the application

Excel

- Used to document testing plans and similar documents

Lucid Chart

- Used to create diagrams like UML, architecture, and schemas

Draw.io

- Used to create ER Diagrams

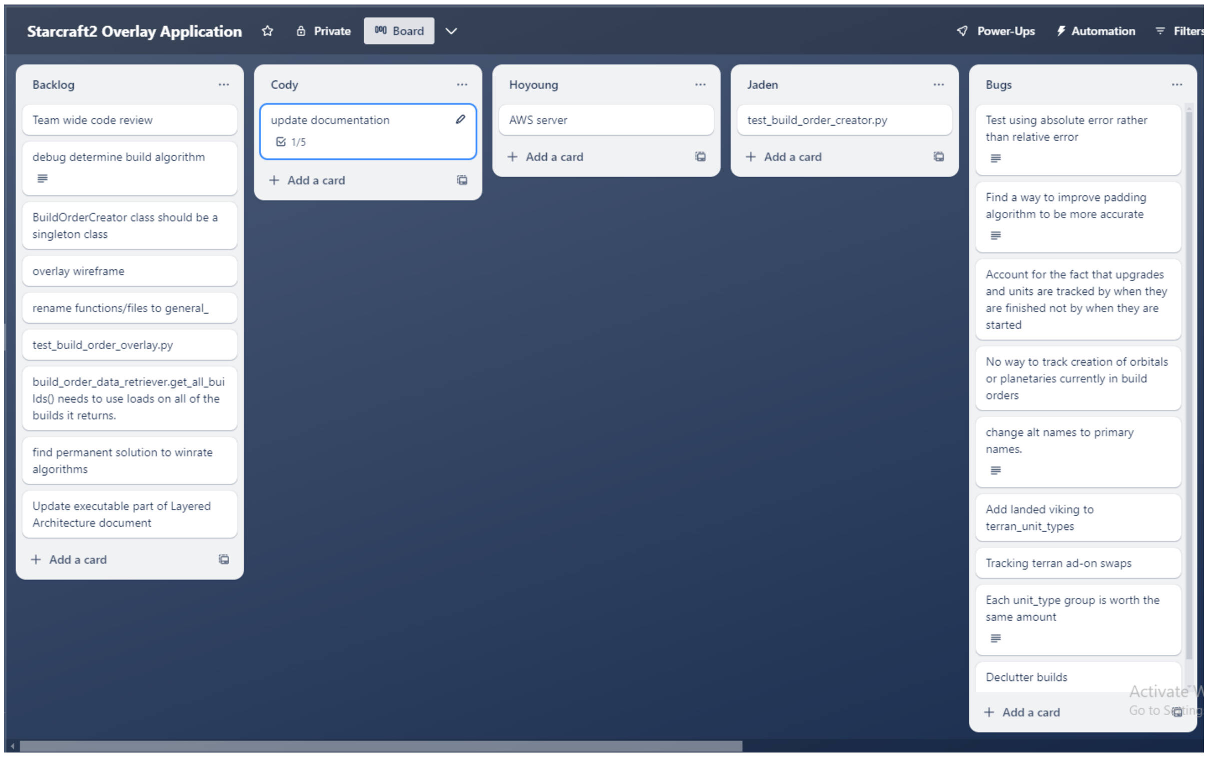
Scheduling

Meetings

- About every other day
- Otherwise by Appointment

Backlog

Description: This contains the tasks that we have completed each iteration as well as what we are currently working on, what known bugs exist, and what remains to be done. We add to the backlog as we discover new tasks but ignore them until we finish what we are working on.



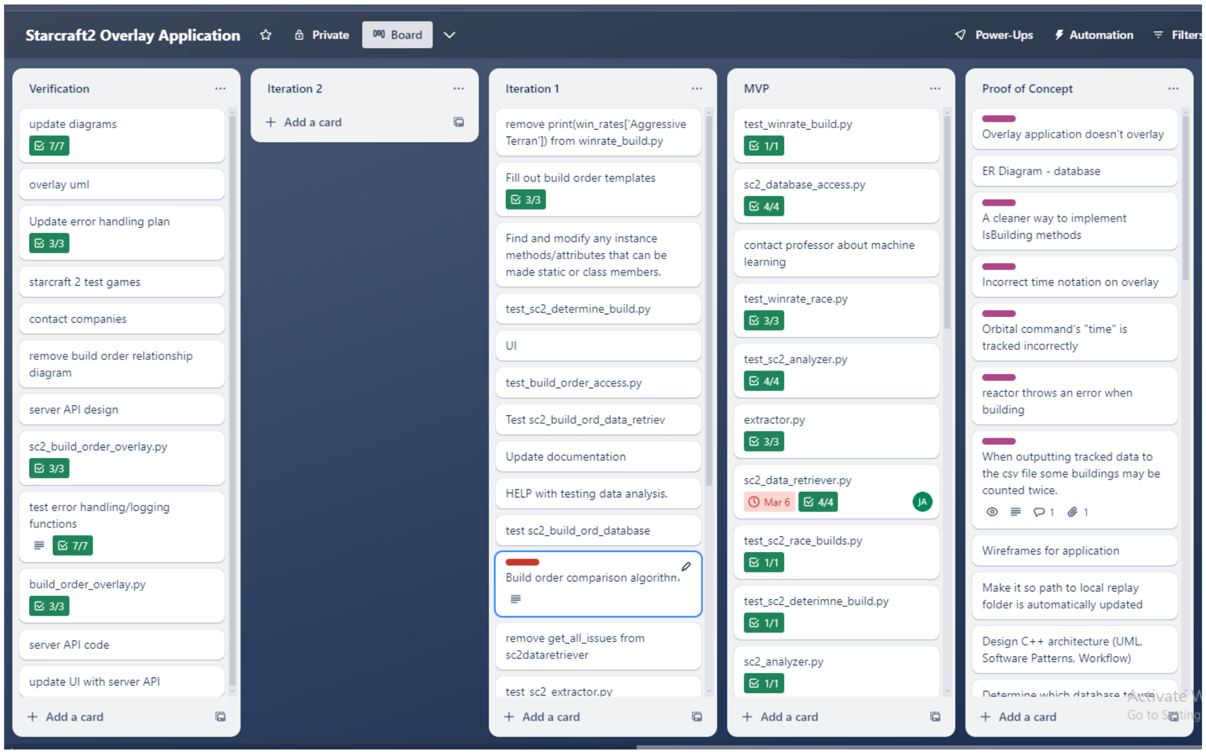
Backlog
- Team wide code review
- debug determine build algorithm
- BuildOrderCreator class should be a singleton class
- overlay wireframe
- rename functions/files to general_
- test_build_order_overlay.py
- build_order_data_retriever.get_all_builds() needs to use loads on all of the builds it returns.
- find permanent solution to winrate algorithms
- Update executable part of Layered Architecture document
- Add a card

Cody
- update documentation 1/5
- Add a card

Hoyoung
- AWS server
- Add a card

Jaden
- test_build_order_creator.py
- Add a card

Bugs
- Test using absolute error rather than relative error
- Find a way to improve padding algorithm to be more accurate
- Account for the fact that upgrades and units are tracked by when they are finished not by when they are started
- No way to track creation of orbitals or planetaries currently in build orders
- change alt names to primary names.
- Add landed viking to terran_unit_types
- Tracking terran ad-on swaps
- Each unit_type group is worth the same amount
- Declutter builds
- Add a card



Verification
- update diagrams 7/7
- overlay uml
- Update error handling plan 3/3
- starcraft 2 test games
- contact companies
- remove build order relationship diagram
- server API design
- sc2_build_order_overlay.py 3/3
- test error handling/logging functions 7/7
- build_order_overlay.py 3/3
- server API code
- update UI with server API
- Add a card

Iteration 2
- Add a card

Iteration 1
- remove print(win_rates['Aggressive Terran']) from winrate_build.py
- Fill out build order templates 3/3
- Find and modify any instance methods/attributes that can be made static or class members.
- test_sc2_determine_build.py
- UI
- test_build_order_access.py
- Test sc2_build_ord_data_retriev
- Update documentation
- HELP with testing data analysis.
- test sc2_build_ord_database
- Build order comparison algorithm
- remove get_all_issues from sc2dataretriever
- test sc2 extractor.py
- Add a card

MVP
- test_winrate_build.py 1/1
- sc2_database_access.py 4/4
- contact professor about machine learning
- test_winrate_race.py 3/3
- test_sc2_analyzer.py 4/4
- extractor.py 3/3
- sc2_data_retriever.py Mar 6 4/4 JA
- test_sc2_race_builds.py 1/1
- test_sc2_deterimne_build.py 1/1
- sc2_analyzer.py 1/1
- Add a card

Proof of Concept
- Overlay application doesn't overlay
- ER Diagram - database
- A cleaner way to implement IsBuilding methods
- Incorrect time notation on overlay
- Orbital command's "time" is tracked incorrectly
- reactor throws an error when building
- When outputting tracked data to the csv file some buildings may be counted twice. 1 1
- Wireframes for application
- Make it so path to local replay folder is automatically updated
- Design C++ architecture (UML, Software Patterns, Workflow)
- Determine which database to use
- Add a card

Requirements Gathering

Functional Requirements

Description: These requirements are the general goals that our application should be able to meet. How they are met is determined later.

- Analyze groups of replays.
  - Determine build order
  - Determine win rates based on races
  - Determine win rates of build 'A' vs build 'B'
- Import and display build order to live game.

Non-Functional Requirements

Description: These requirements do not pertain to specific behaviors or functionalities of the application but rather to its overall attributes and characteristics.

Performance and Responsiveness

- The application should be capable of processing and displaying data with minimal latency.
- It should handle high volumes of concurrent users and data requests efficiently.

Scalability

- The system should be scalable to accommodate a growing number of users and an increasing amount of data.

Reliability and Availability

- The application should have high uptime, with minimal downtime for maintenance or updates.
- It should be reliable in delivering accurate and consistent analytics data.

Security

- Strong measures for data security, including encryption of sensitive data and secure handling of user information.
- Implementation of proper authentication and authorization mechanisms to protect user accounts and data.

Maintainability and Modularity

- The codebase should be well-organized and documented for ease of maintenance and updates.

Usability and Accessibility

- The user interface should be intuitive and user-friendly, catering to both novice and experienced gamers.
- The application should be accessible to users with disabilities, complying with relevant accessibility standards.

Compliance and Legal Requirements

- Adherence to legal and regulatory requirements, such as data protection laws (e.g., GDPR, if applicable).

Risk Analysis

Technical Risks

Risk of Inaccurate Analysis

There's a risk that the program may not accurately analyze replays due to incorrect logic, outdated algorithms, or compatibility issues with different SC2 versions.

- Mitigation: Regularly update the program to align with the latest game patches, and thoroughly test the program with a variety of replays
- Contingency: Temporarily remove feature that is inaccurate until we can guarantee accuracy

Risk of Incompatibility with Future SC2 Updates

Future updates to SC2 might change the replay format or introduce new features not supported by the current program.

- Mitigation: Plan for regular updates and maintenance and stay informed about upcoming SC2 updates.
- Contingency: Make it so the program doesn't accept replays past the date of the new update until the program is compatible with the new version of sc2 replays.

Legal and Compliance Risks

Risk of Data Privacy Violations

If the analytics tool collects user data, it must comply with data protection regulations like GDPR or CCPA.

- Mitigation: Implement strong data privacy policies and only collect necessary data with user consent.
- Contingency: Shut down program until it complies with data protection regulations.

Operational Risks

Risk of Dependency on External Libraries

The project might rely on external libraries (like sc2reader) which could become outdated or unsupported.

- Mitigation: None
- Contingency: Drop project

Risk of Insufficient Testing

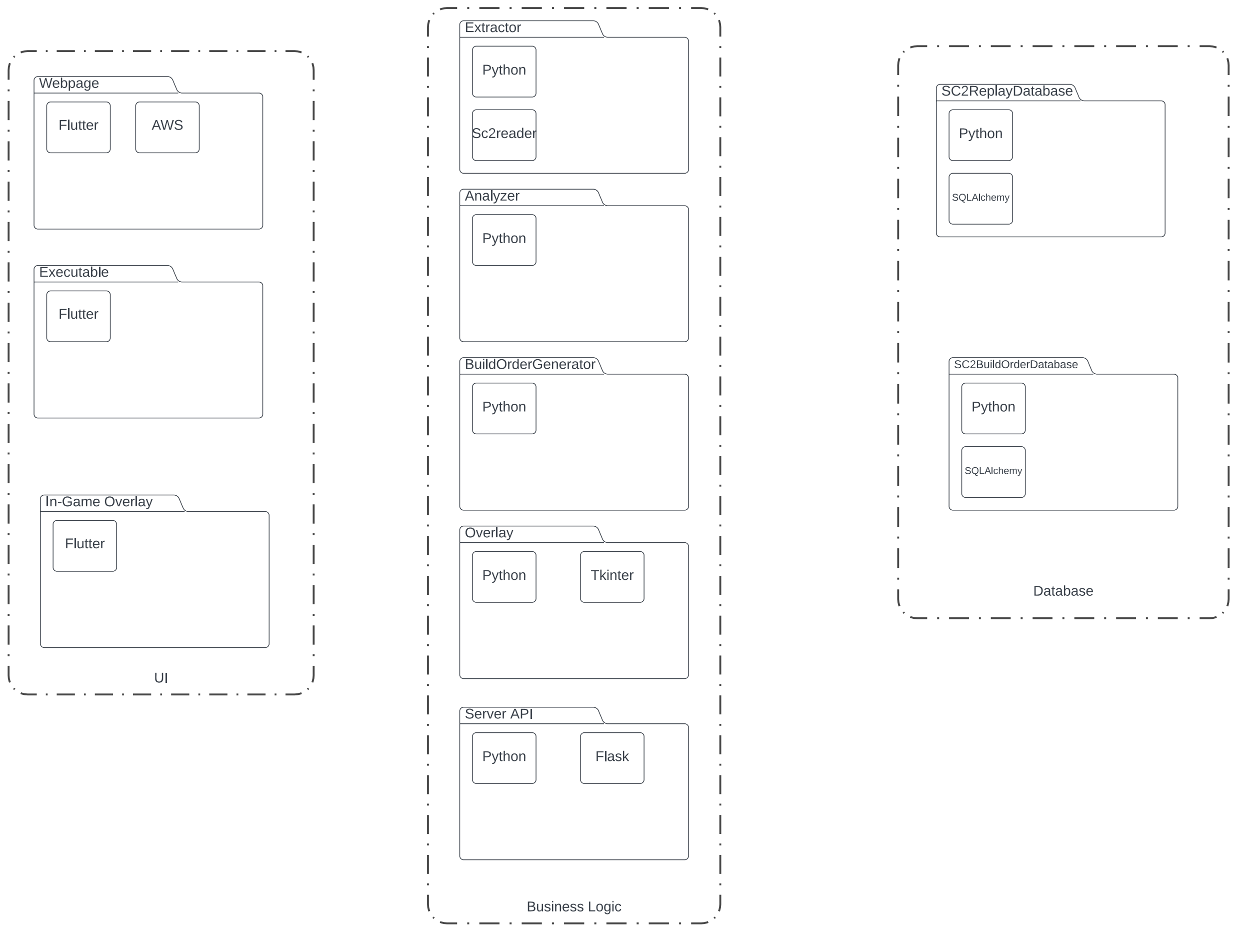
Inadequate testing can lead to undetected bugs and issues in production.

- Mitigation: Implement comprehensive testing strategies, including unit tests, integration tests, and user acceptance tests.
- Contingency: Fix bugs, possibly rollback to previous version of product and add more comprehensive testing

High Level Design

Layered Architecture

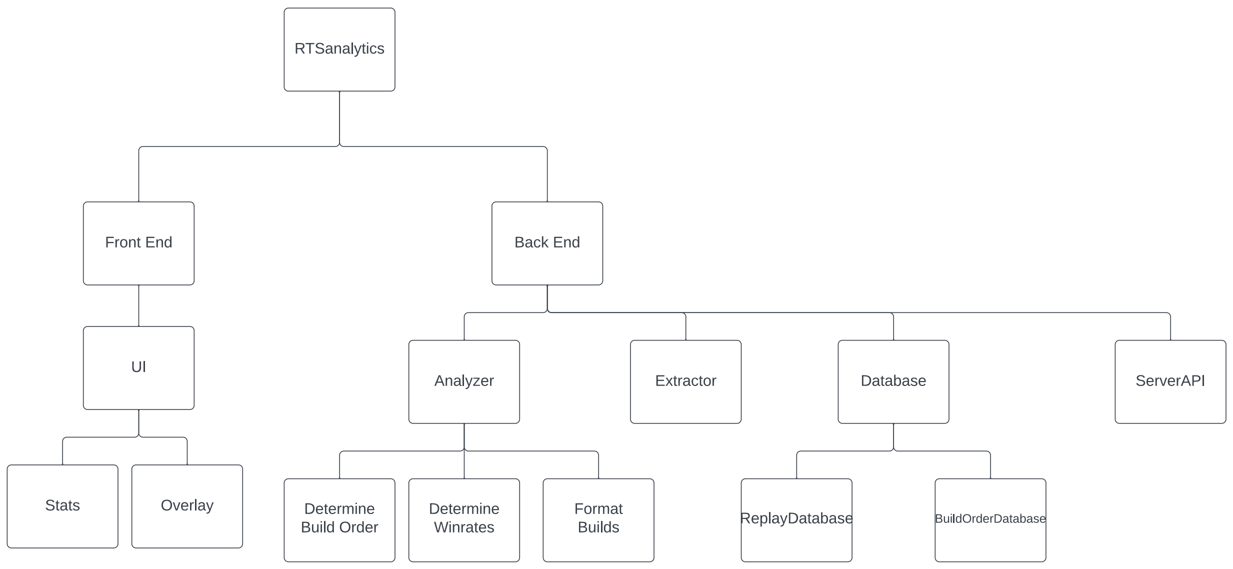
Description: Depicts all the high-level modules, what tools are used to create them and how they can be categorized



RTS Analyzer

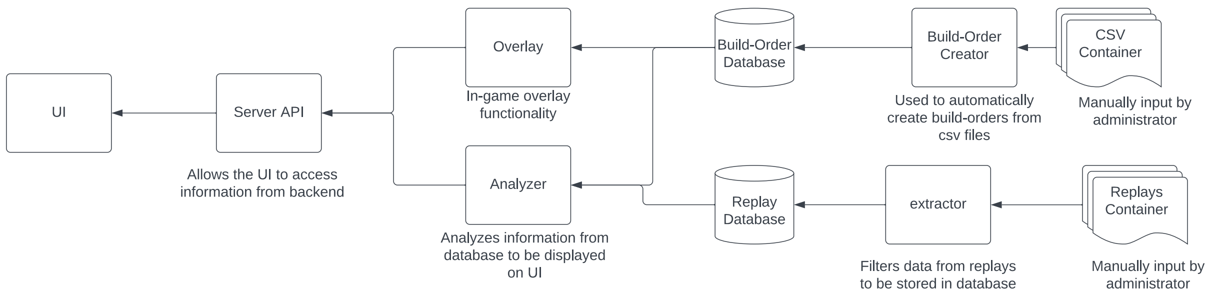
Hierarchical Architecture

Show how all the high-level modules are split into low-level modules



Data Flowchart

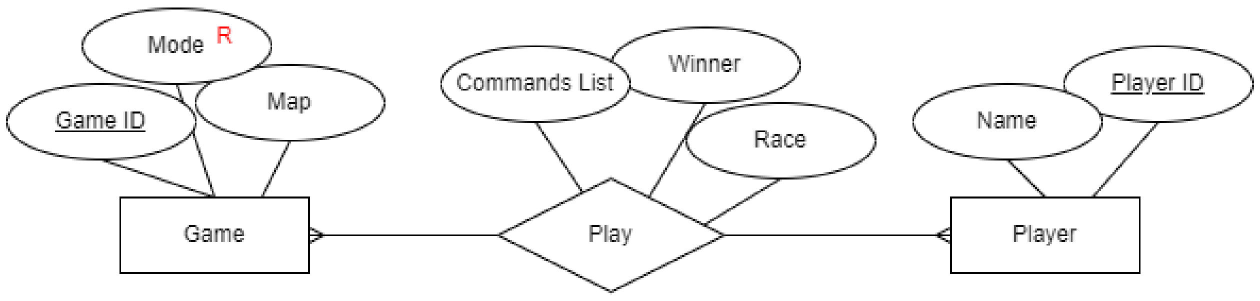
Description: Depicts all the high-level modules in the project and how data flows between them



UI
Server API
Allows the UI to access information from backend
Overlay
In-game overlay functionality
Analyzer
Analyzes information from database to be displayed on UI
Build-Order Database
Replay Database
Build-Order Creator
Used to automatically create build-orders from csv files
extractor
Filters data from replays to be stored in database
CSV Container
Manually input by administrator
Replays Container
Manually input by administrator

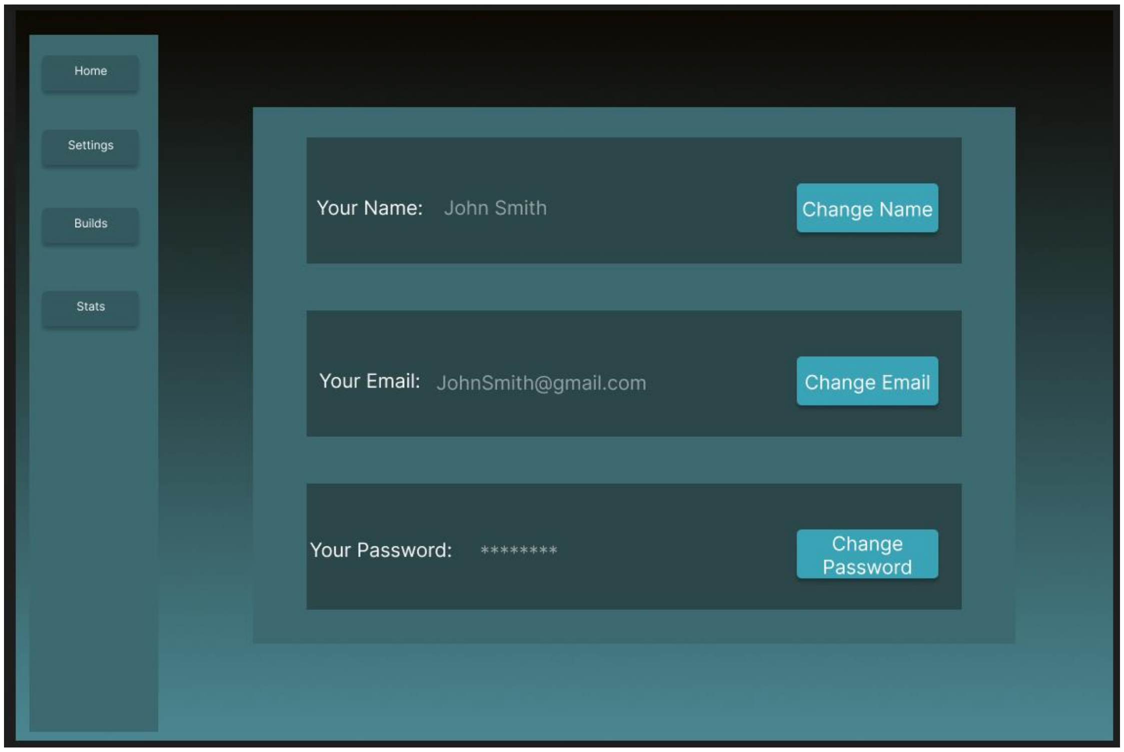
SC2 Replay ER Diagram

Description: Depicts all the entities in the database and their relationships to one another

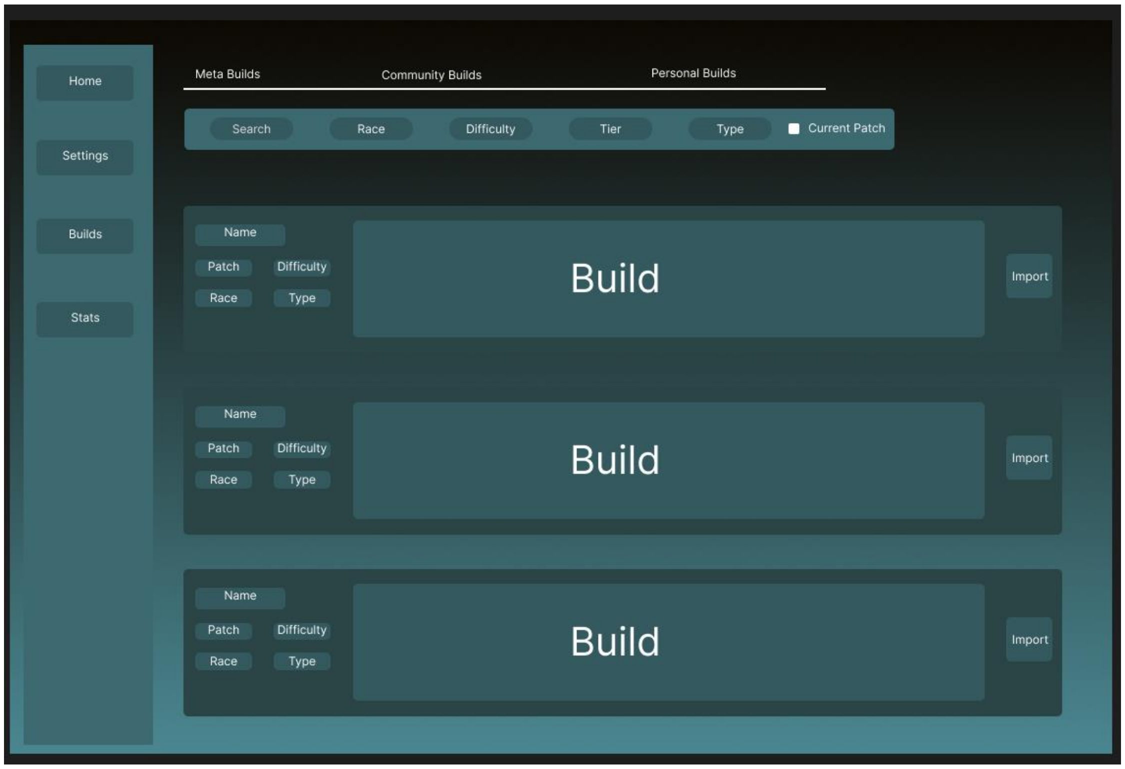


Wireframes

Accounts

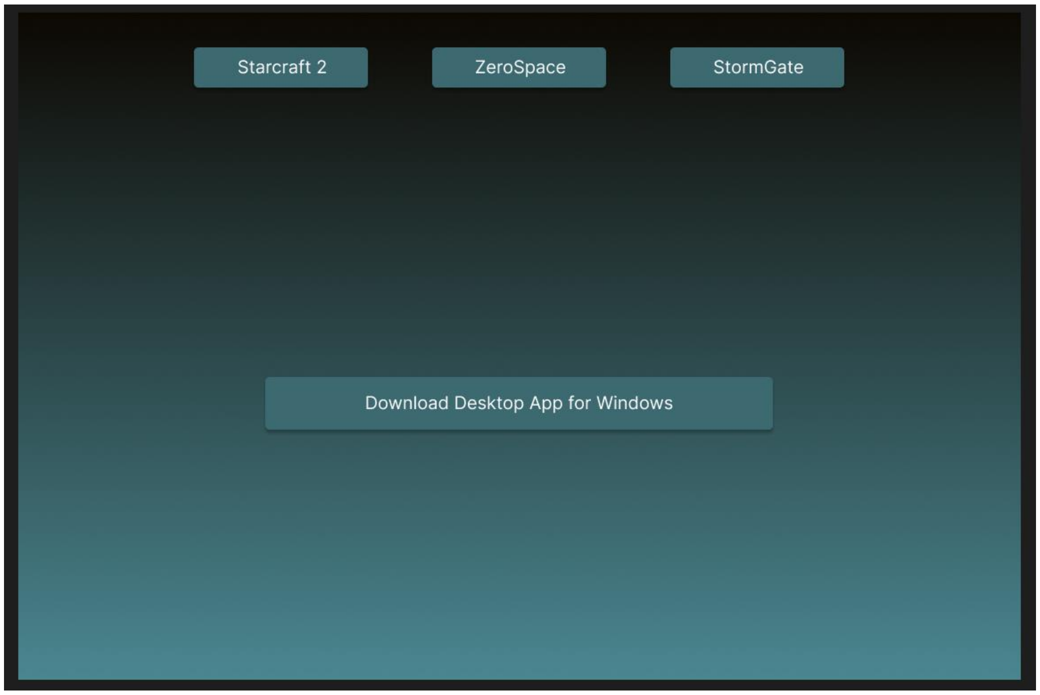


Builds

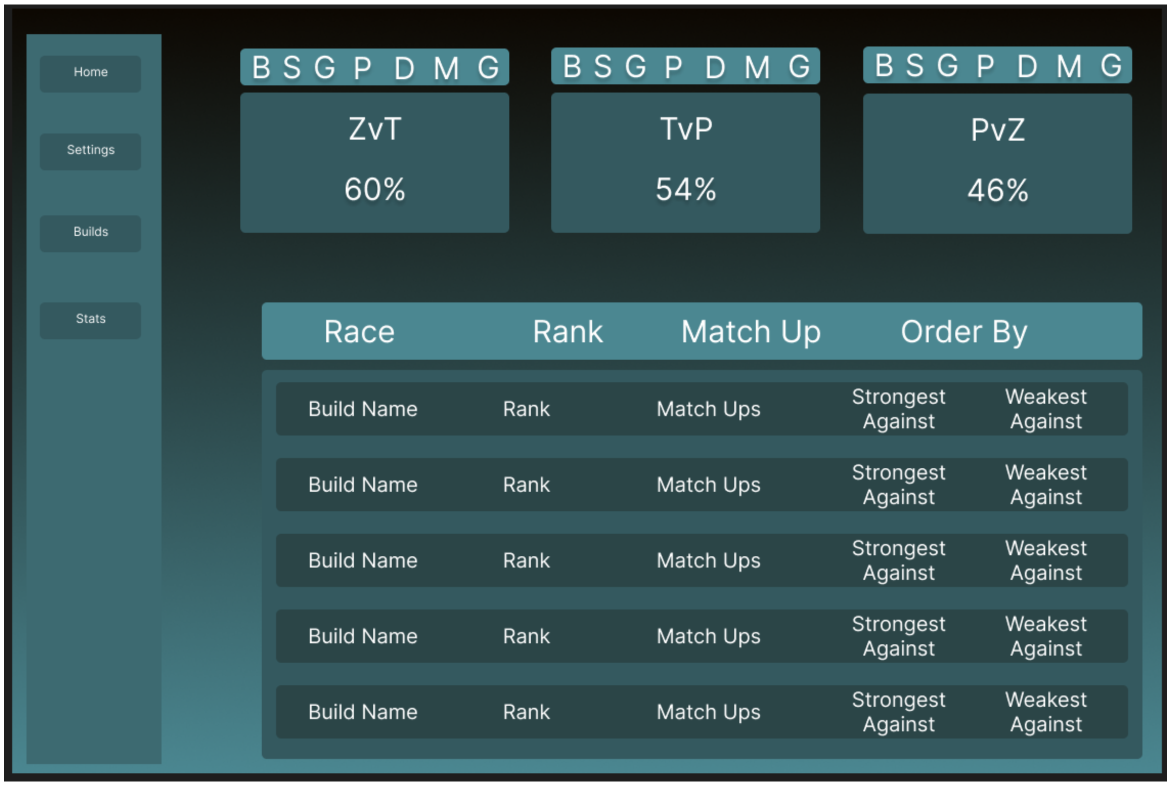


RTS Analyzer

Home



Stats Page

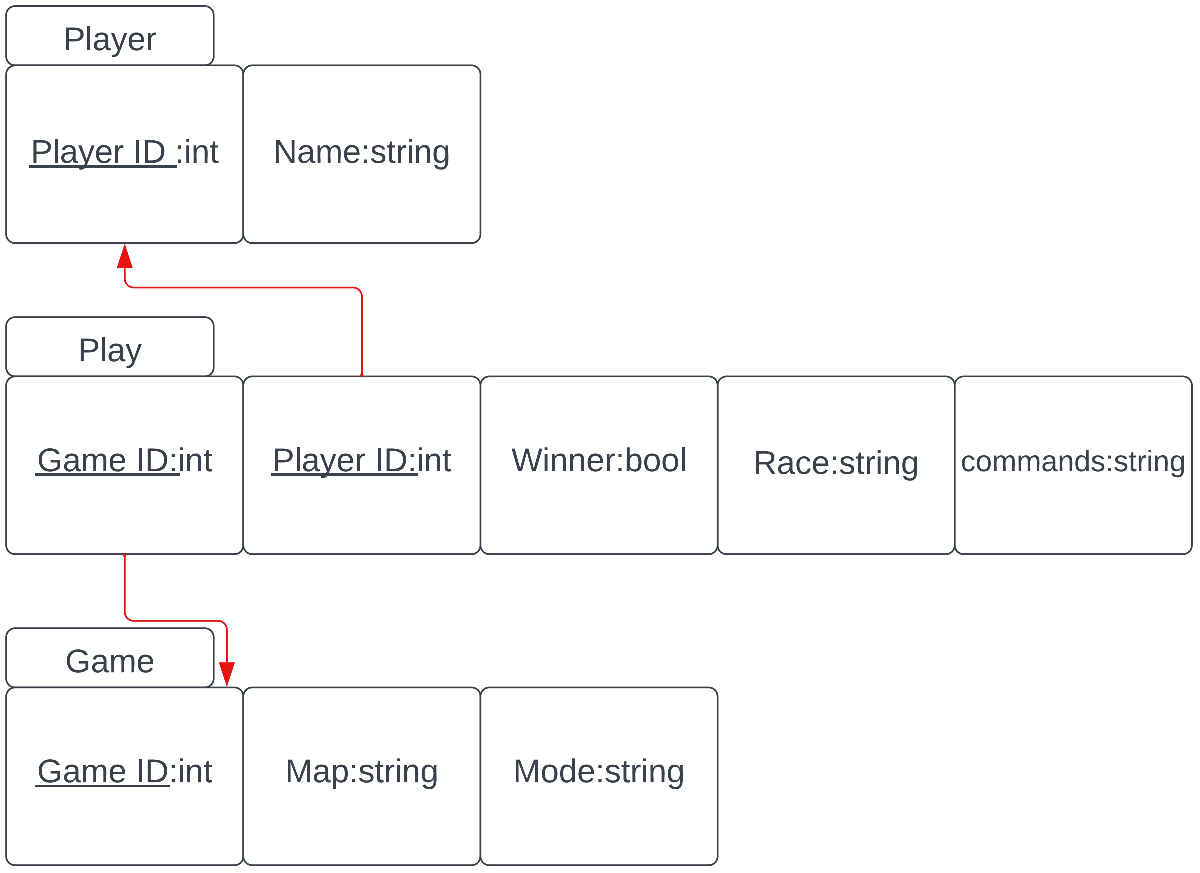


Build Name	Rank	Match Ups	Strongest Against	Weakest Against
Build Name	Rank	Match Ups	Strongest Against	Weakest Against
Build Name	Rank	Match Ups	Strongest Against	Weakest Against
Build Name	Rank	Match Ups	Strongest Against	Weakest Against
Build Name	Rank	Match Ups	Strongest Against	Weakest Against

Low Level Design

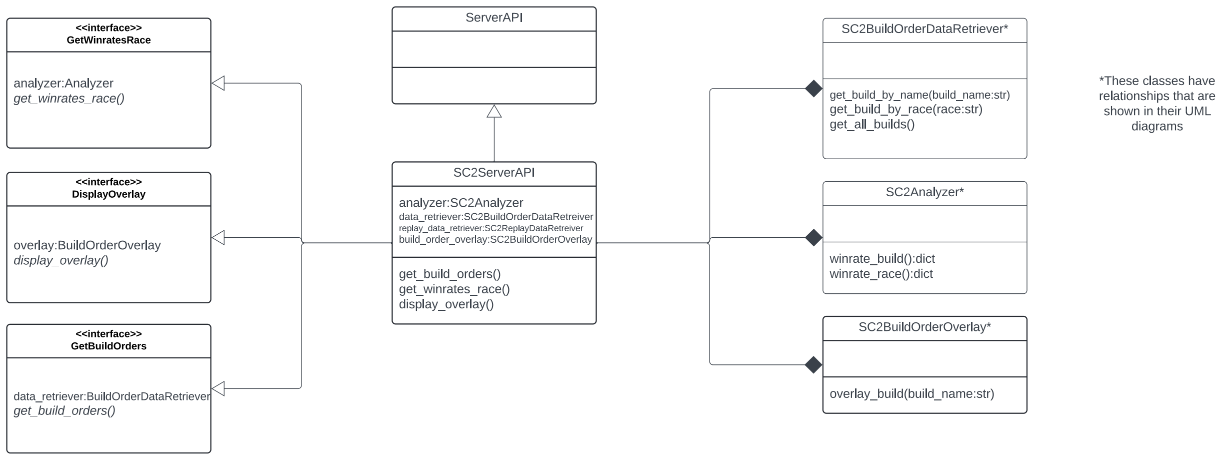
SC2 Replay Schema

Description: Depicts how the entities and relationships from the SC2 Replay ER Diagram will be converted into tables for the database



Server API Class UML

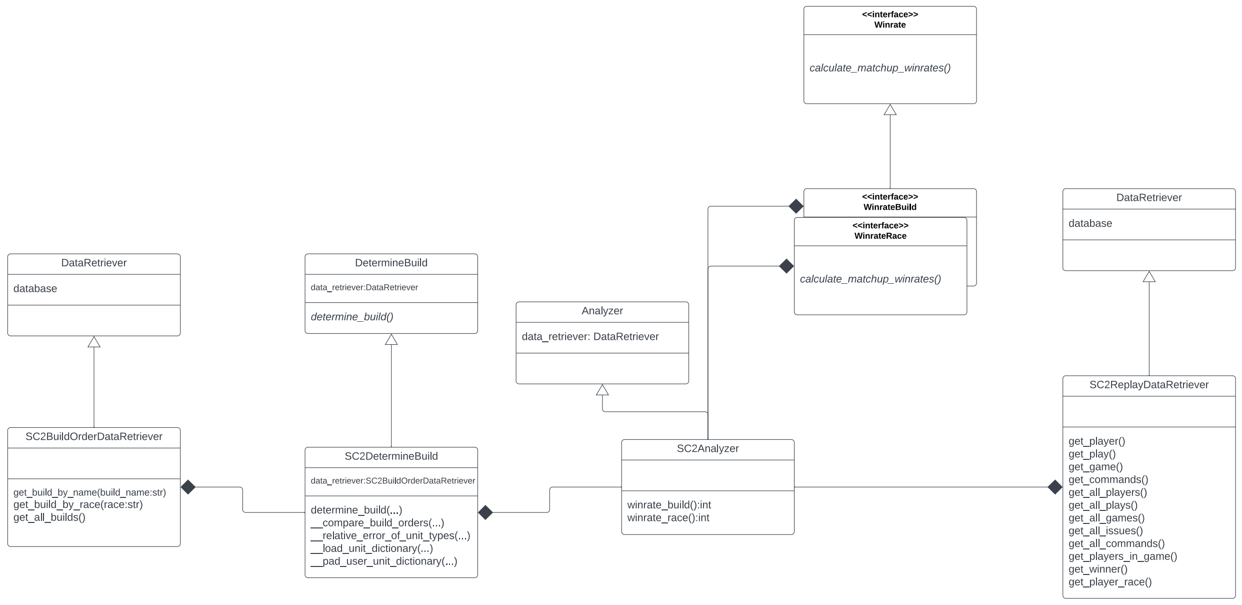
Description: UML class diagram of the python server API portion of the project


<<interface>>
GetWinratesRace
analyzer:Analyzer
get_winrates_race()
<<interface>>
DisplayOverlay
overlay:BuildOrderOverlay
display_overlay()
<<interface>>
GetBuildOrders
data_retriever:BuildOrderDataRetriever
get_build_orders()
ServerAPI
SC2ServerAPI
analyzer:SC2Analyzer
data_retriever:SC2BuildOrderDataRetreiver
replay_data_retriever:SC2ReplayDataRetreiver
build_order_overlay:SC2BuildOrderOverlay
get_build_orders()
get_winrates_race()
display_overlay()
SC2BuildOrderDataRetriever*
get_build_by_name(build_name:str)
get_build_by_race(race:str)
get_all_builds()
SC2Analyzer*
winrate_build():dict
winrate_race():dict
SC2BuildOrderOverlay*
overlay_build(build_name:str)
*These classes have relationships that are shown in their UML diagrams

RTS Analyzer

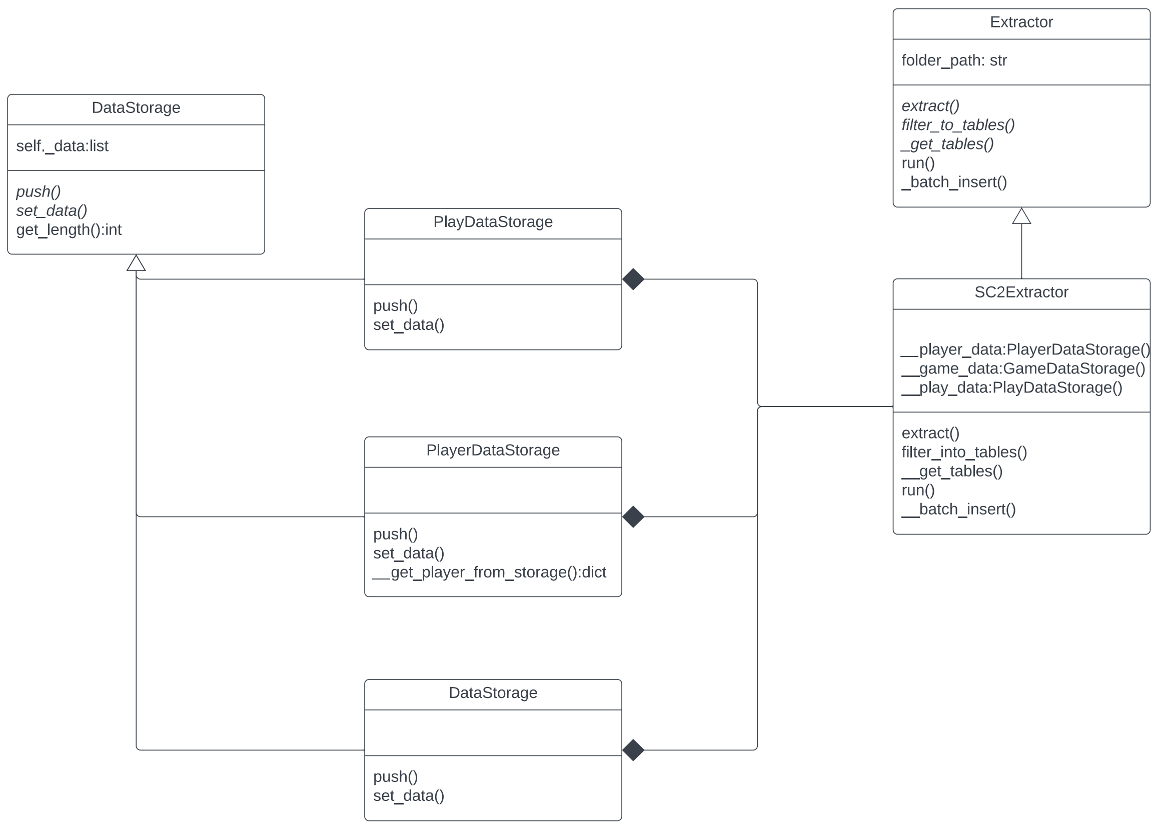
Analyzer Class UML

Description: UML class diagram of the analyzer portion of the project



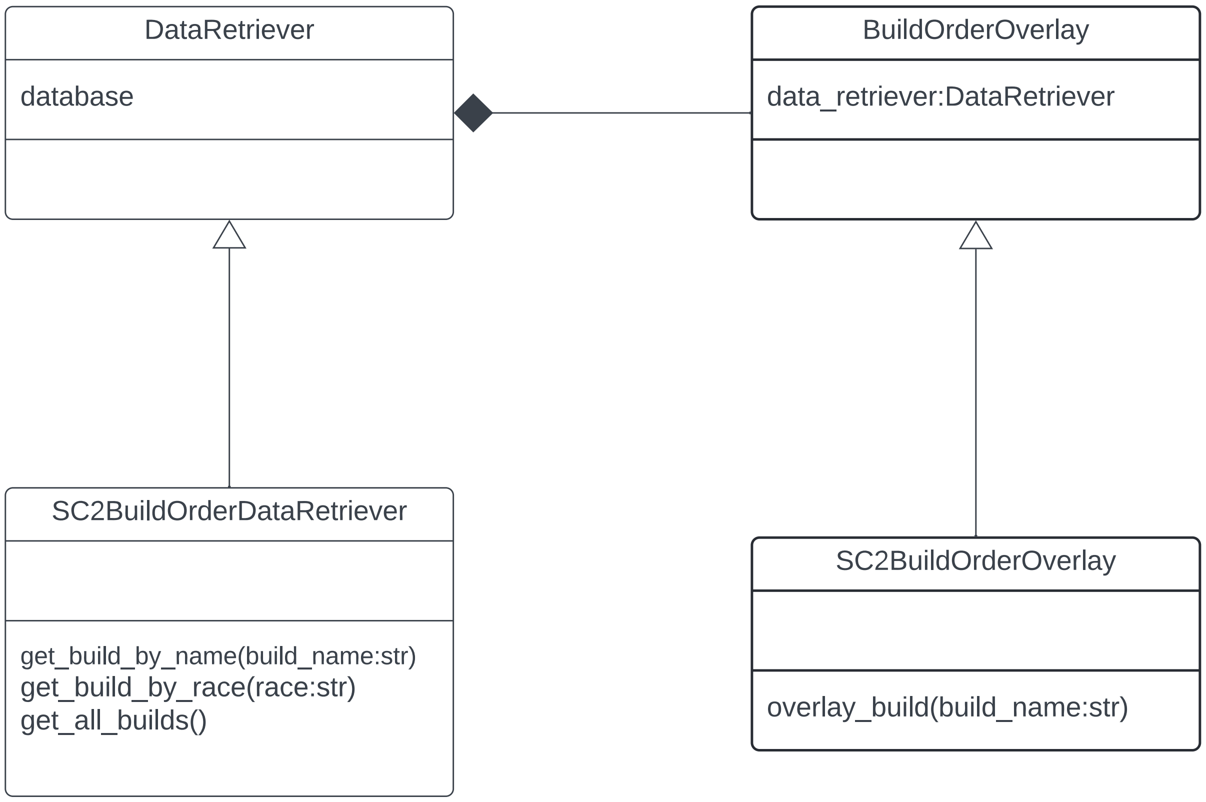
Extractor Class UML

Description: UML class diagram of the extractor portion of the project



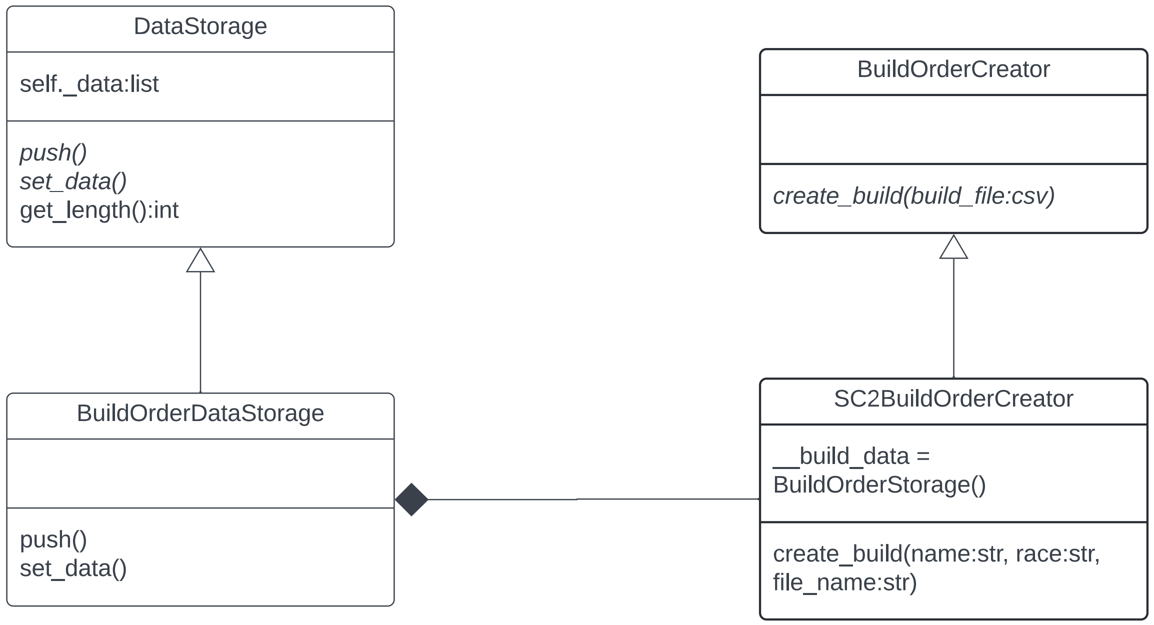
Overlay Class UML

Description: UML class diagram of the overlay portion of the project


DataRetriever
database
BuildOrderOverlay
data_retriever:DataRetriever
SC2BuildOrderDataRetriever
get_build_by_name(build_name:str)
get_build_by_race(race:str)
get_all_builds()
SC2BuildOrderOverlay
overlay_build(build_name:str)

Build Order Creator Class UML

Description: UML class diagram of the build order creator portion of the project



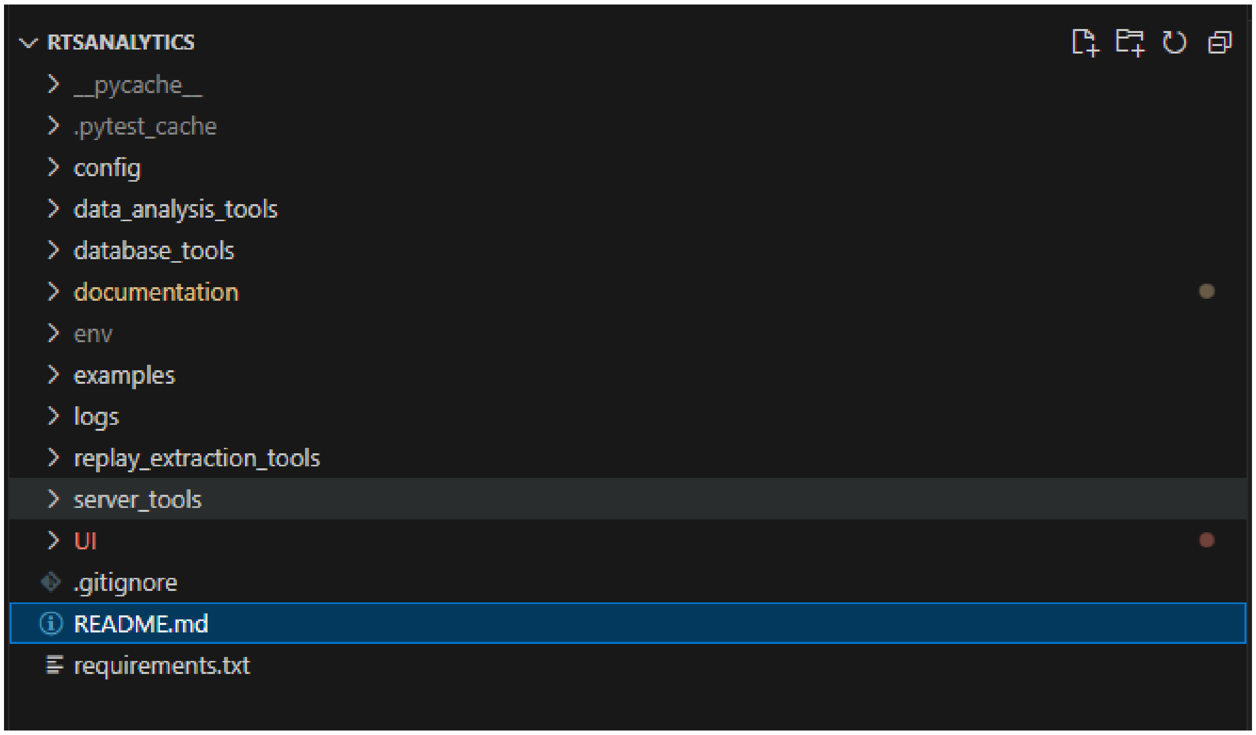
Development

File Structure

Root

Description: We organized the files by their functionality at the highest level the functionalities they are split on are, configuration, analysis, database, data extraction, user interface, and server integration.

We also have folders for documentation, logging, and running examples.



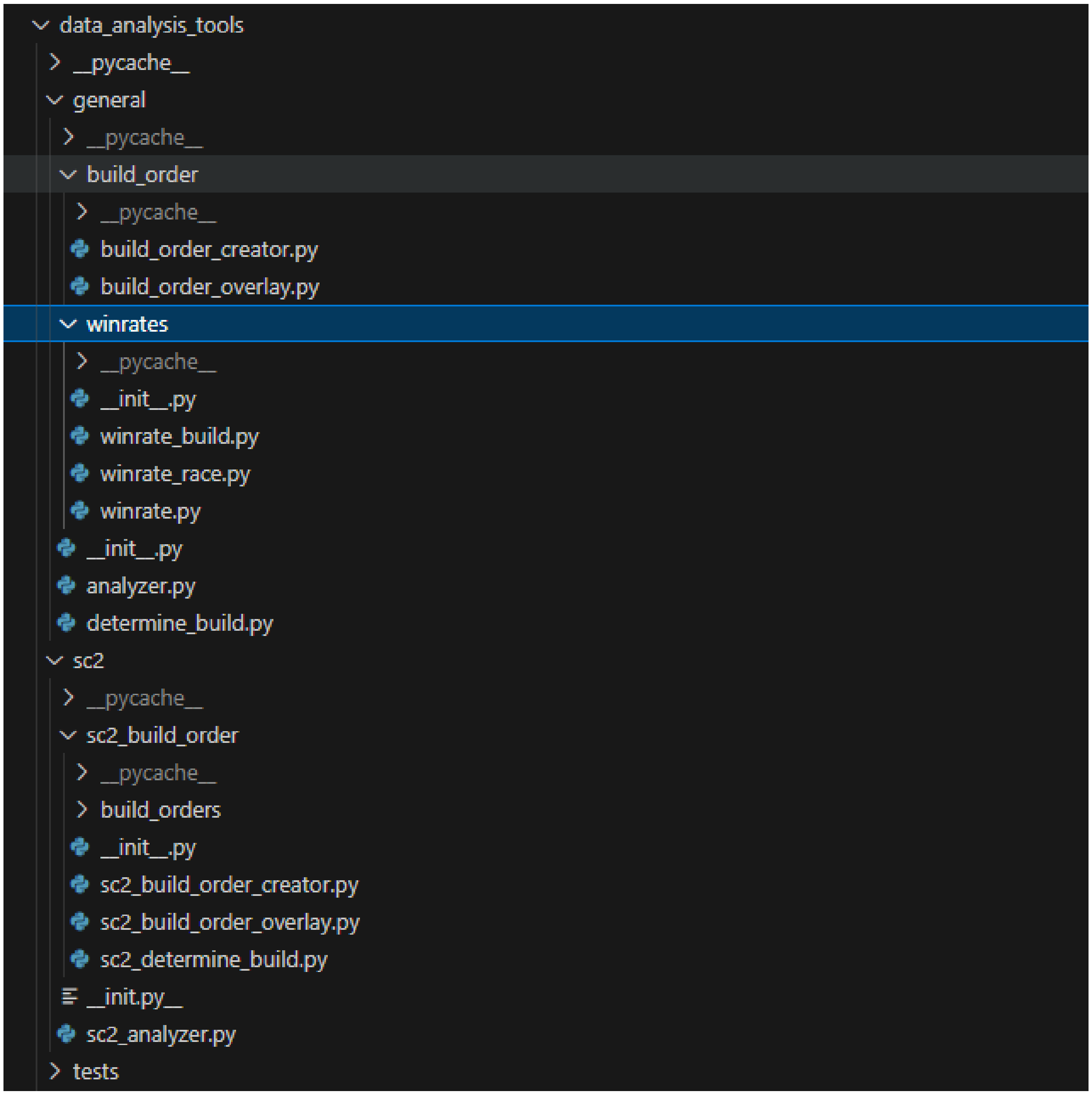
```
RTSANALYTICS
  > __pycache__
  > .pytest_cache
  > config
  > data_analysis_tools
  > database_tools
  > documentation
  > env
  > examples
  > logs
  > replay_extraction_tools
  > server_tools
  > UI
  .gitignore
  README.md
  requirements.txt
```

Data Analysis Tools

Description: Data analysis contains code for analyzing information, such as being able to determine what build order a user is doing, and win rates based on multiple factors.

It also includes code for creating and overlaying build orders, it is debatable whether that code should be under data analysis tools or under its own dedicated folder.

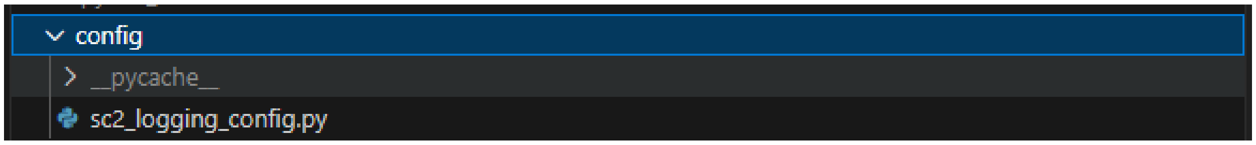
The general folder contains the parent classes and interfaces that RTS specific classes can inherit from. For now we only have SC2 specific classes.



```
data_analysis_tools
  __pycache__
  general
    __pycache__
    build_order
      __pycache__
      build_order_creator.py
      build_order_overlay.py
    winrates
      __pycache__
      __init__.py
      winrate_build.py
      winrate_race.py
      winrate.py
    __init__.py
    analyzer.py
    determine_build.py
  sc2
    __pycache__
    sc2_build_order
      __pycache__
      build_orders
      __init__.py
      sc2_build_order_creator.py
      sc2_build_order_overlay.py
      sc2_determine_build.py
    __init.py__
    sc2_analyzer.py
  tests
```

Configuration

Description: The config file currently only contains the logging configuration for all sc2 specific logging. We will likely create RTS specific folders as we need more configuration files.

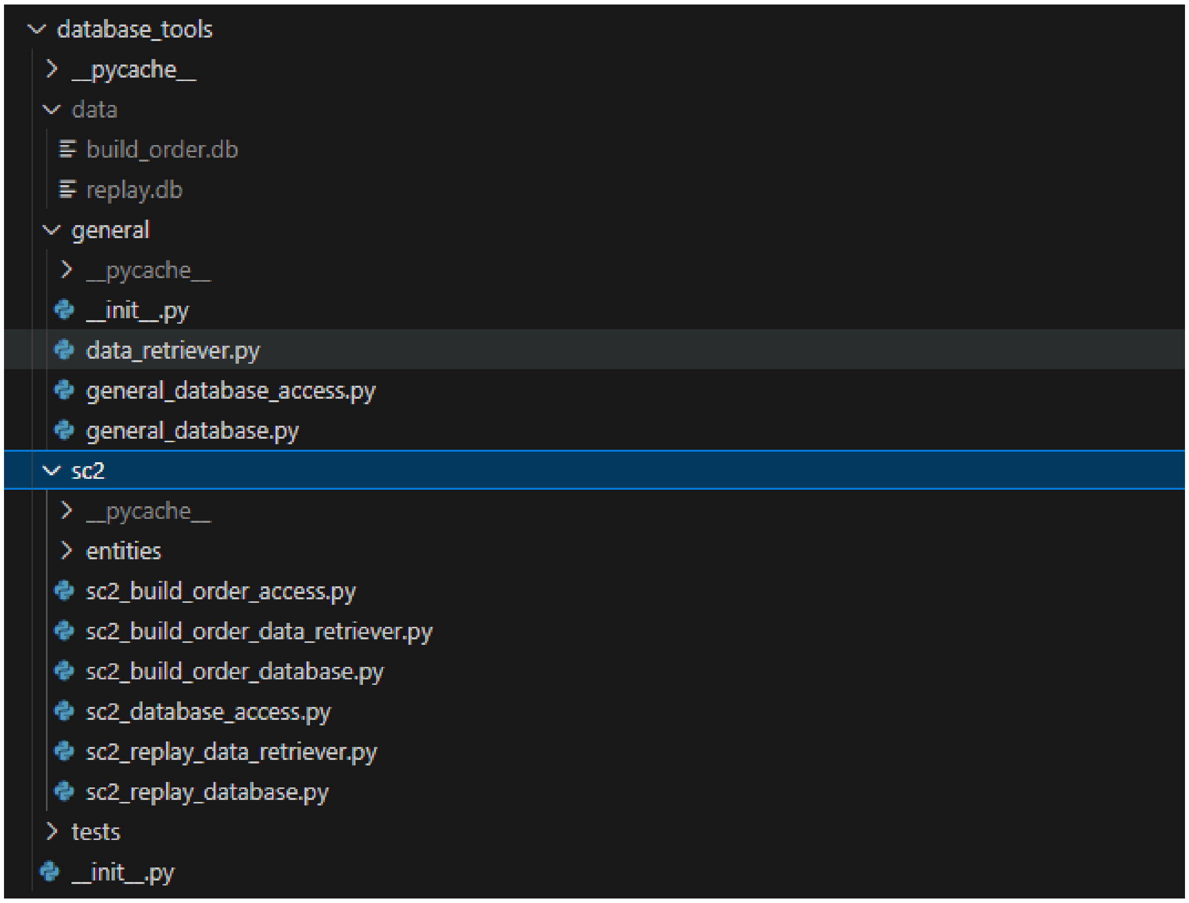


```
config
  __pycache__
  sc2_logging_config.py
```

Database Tools

Description: This contains all database related code, the data folder contains all of our database files. Each database has three file dedicated to accessing it, first is the database.py files these files are what directly create and access the .db files. However in order to guarantee that we can easily change databases when needed we have database_access.py and database_retriever.py files that act as a middle man for any files that want to push or get data from the database.

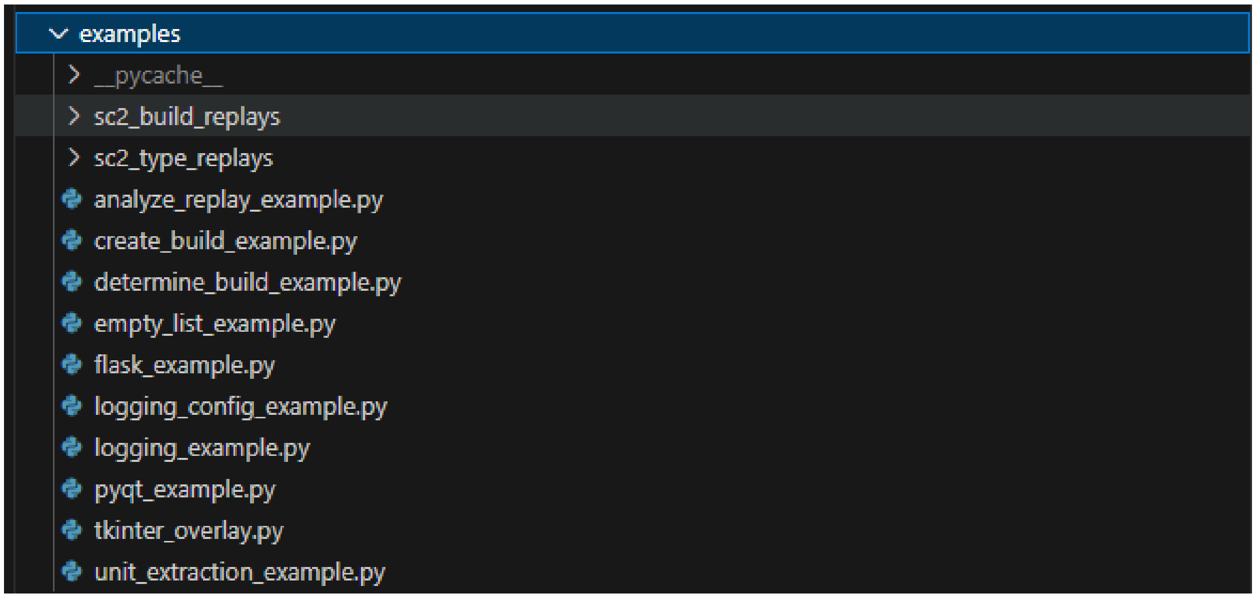
The general folder contains the parent classes and interfaces that RTS specific classes can inherit from. For now we only have SC2 specific classes.



```
database_tools
  __pycache__
  data
    build_order.db
    replay.db
  general
    __pycache__
    __init__.py
    data_retriever.py
    general_database_access.py
    general_database.py
  sc2
    __pycache__
    entities
    sc2_build_order_access.py
    sc2_build_order_data_retriever.py
    sc2_build_order_database.py
    sc2_database_access.py
    sc2_replay_data_retriever.py
    sc2_replay_database.py
  tests
  __init__.py
```

Examples

Description: The examples folder contains code that helps show how to run certain functions or how to implement certain python libraries. As well as containing folders of data needed for the example file to work

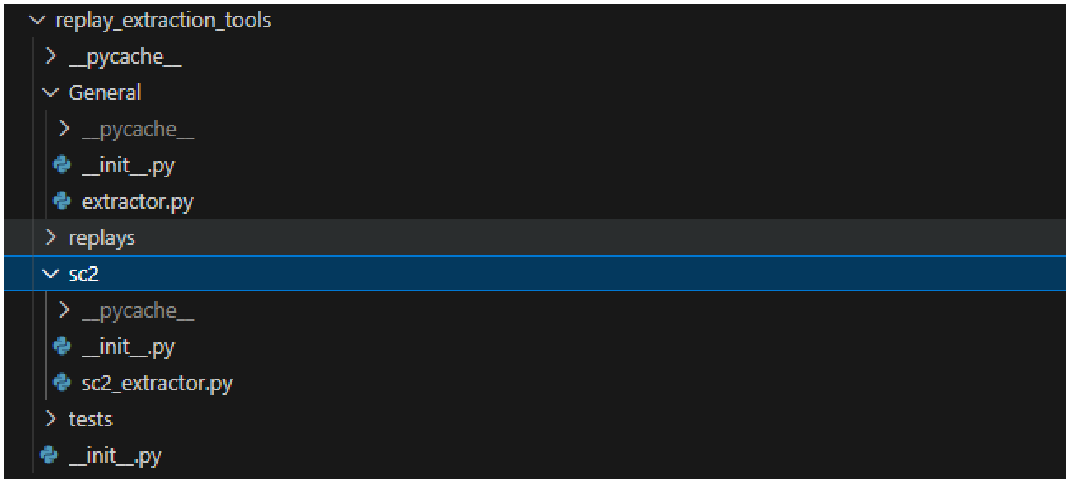


```
examples
  __pycache__
  sc2_build_replays
  sc2_type_replays
  analyze_replay_example.py
  create_build_example.py
  determine_build_example.py
  empty_list_example.py
  flask_example.py
  logging_config_example.py
  logging_example.py
  pyqt_example.py
  tkinter_overlay.py
  unit_extraction_example.py
```

Replay Extraction Tools

Description: This contains files that are used to pull and filter information from RTS replays. It pushes the relevant data to database tools files. Currently it is also where we store the replays that we want to get information from

The general folder contains the parent classes and interfaces that RTS specific classes can inherit from. For now, we only have SC2 specific classes.

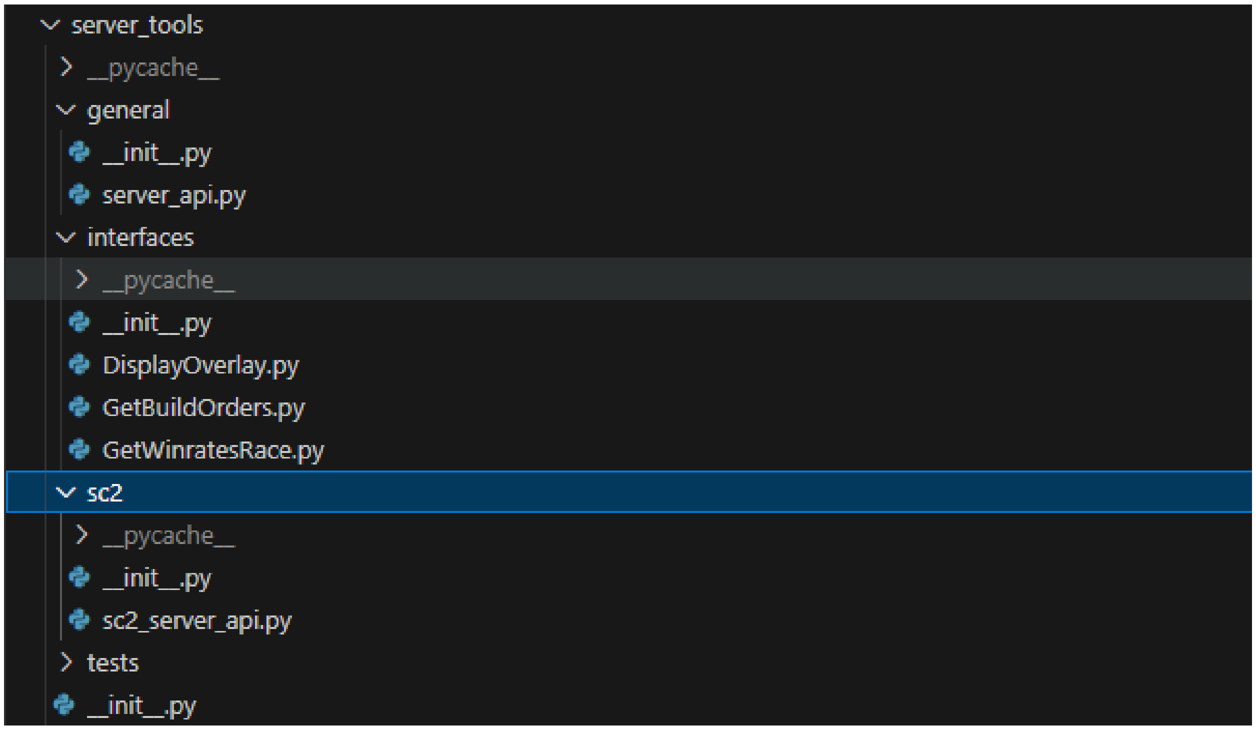


```
replay_extraction_tools
  __pycache__
  General
    __pycache__
    __init__.py
    extractor.py
  replays
  sc2
    __pycache__
    __init__.py
    sc2_extractor.py
  tests
  __init__.py
```

Server Tools

Description: This contains all the code to assist in integrating the python back-end code with the UI.

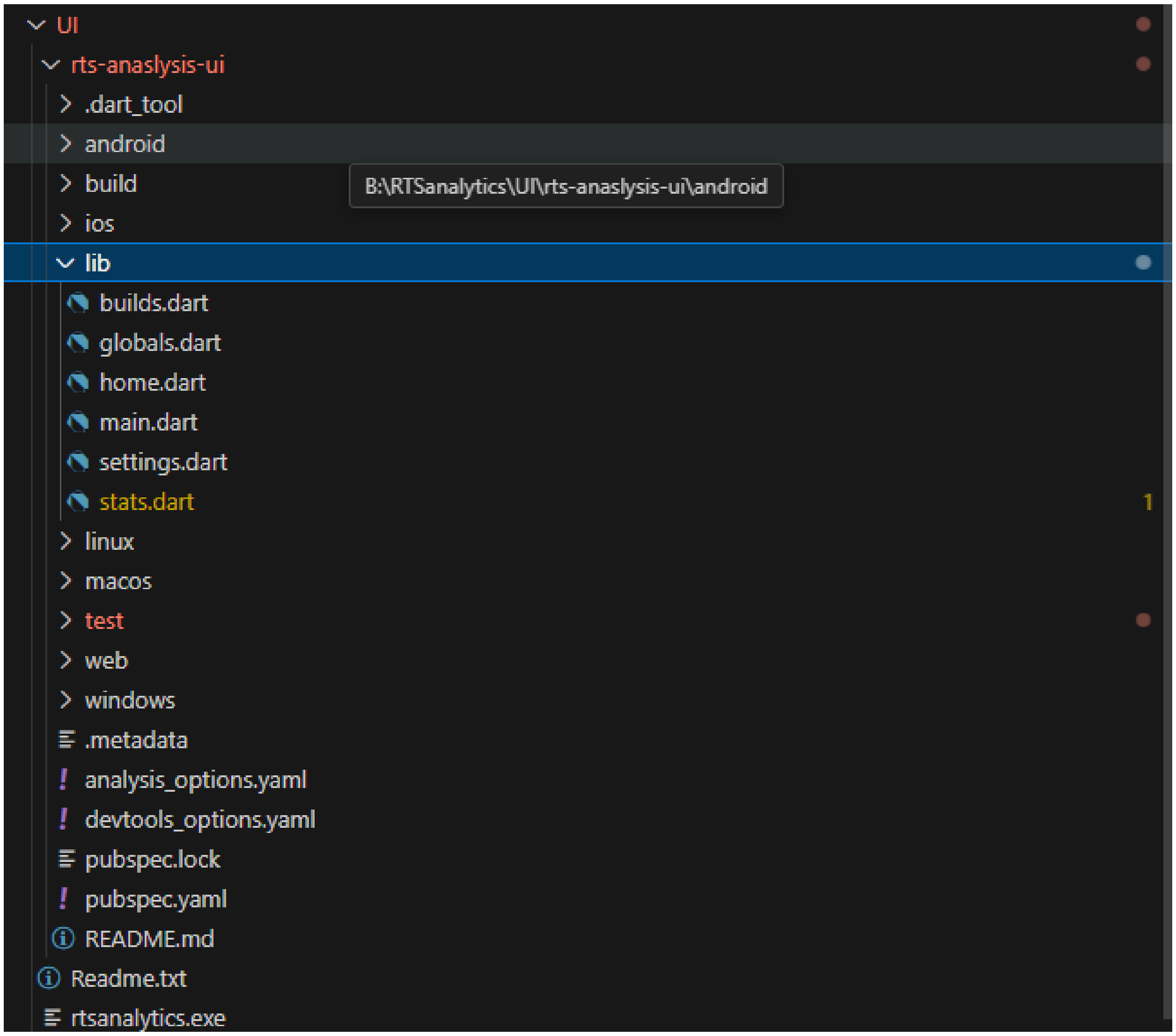
The general folder contains the parent classes and interfaces that RTS specific classes can inherit from. For now we only have SC2 specific classes.



```
server_tools
  __pycache__
  general
    __init__.py
    server_api.py
  interfaces
    __pycache__
    __init__.py
    DisplayOverlay.py
    GetBuildOrders.py
    GetWinratesRace.py
  sc2
    __pycache__
    __init__.py
    sc2_server_api.py
  tests
  __init__.py
```

User Interface

Description: This is the flutter code that builds the user interface, each of the .dart files are a page in the UI. However currently only the stats.dart is up and running.

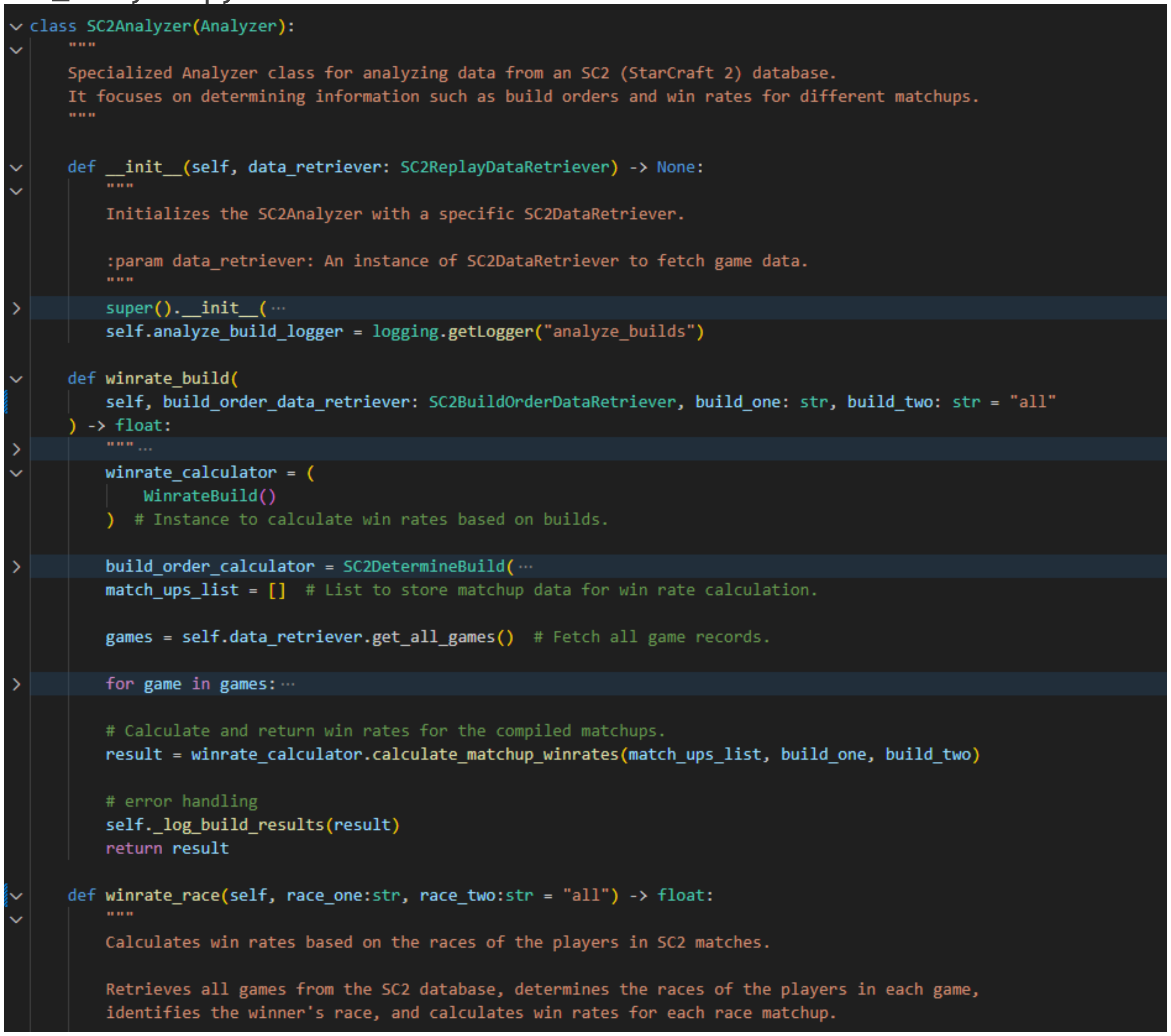


```
UI
  rts-anaslysis-ui
    .dart_tool
    android
    build
    ios
    lib
      builds.dart
      globals.dart
      home.dart
      main.dart
      settings.dart
      stats.dart                                   1
    linux
    macos
    test
    web
    windows
    .metadata
    analysis_options.yaml
    devtools_options.yaml
    pubspec.lock
    pubspec.yaml
    README.md
  Readme.txt
  rtsanalytics.exe
```

B:\RTSanalytics\UI\rts-anaslysis-ui\android

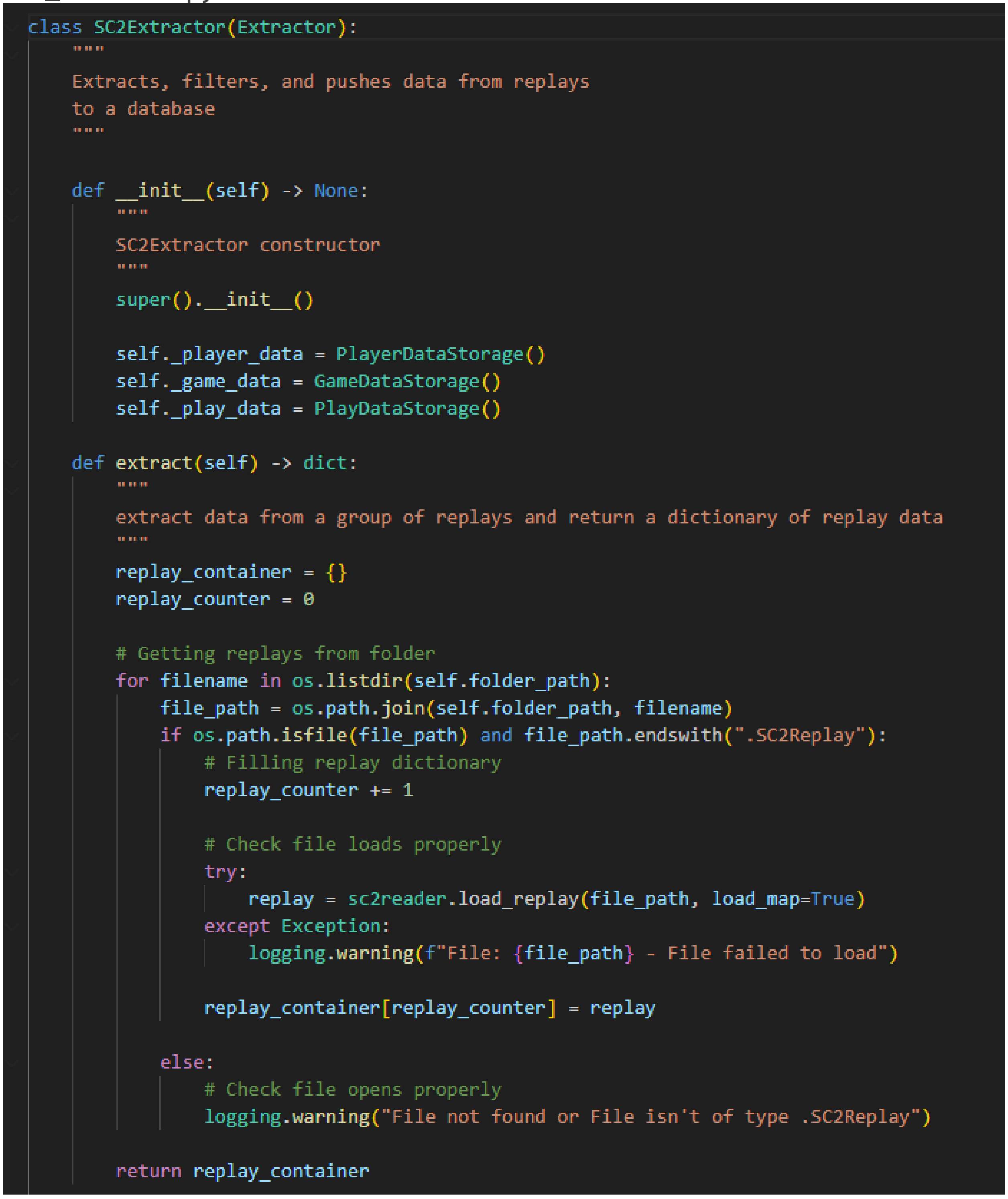
Code Snippets

Sc2_analyzer.py



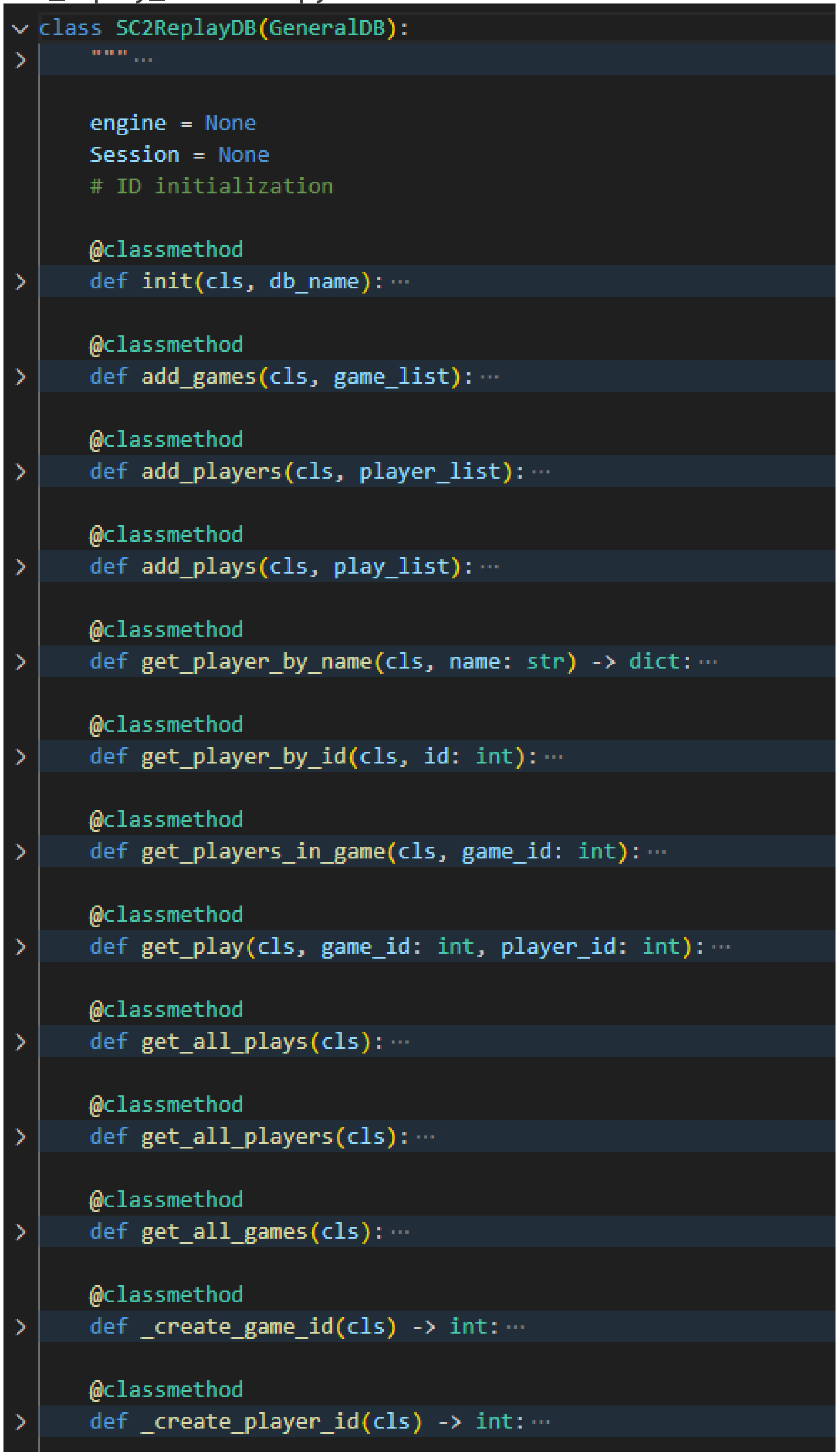
```python
class SC2Analyzer(Analyzer):
    """
    Specialized Analyzer class for analyzing data from an SC2 (StarCraft 2) database.
    It focuses on determining information such as build orders and win rates for different matchups.
    """

    def __init__(self, data_retriever: SC2ReplayDataRetriever) -> None:
        """
        Initializes the SC2Analyzer with a specific SC2DataRetriever.

        :param data_retriever: An instance of SC2DataRetriever to fetch game data.
        """
        super().__init__(…
        self.analyze_build_logger = logging.getLogger("analyze_builds")

    def winrate_build(
        self, build_order_data_retriever: SC2BuildOrderDataRetriever, build_one: str, build_two: str = "all"
    ) -> float:
        """…
        winrate_calculator = (
            WinrateBuild()
        )  # Instance to calculate win rates based on builds.

        build_order_calculator = SC2DetermineBuild(…
        match_ups_list = []  # List to store matchup data for win rate calculation.

        games = self.data_retriever.get_all_games()  # Fetch all game records.

        for game in games:…

        # Calculate and return win rates for the compiled matchups.
        result = winrate_calculator.calculate_matchup_winrates(match_ups_list, build_one, build_two)

        # error handling
        self._log_build_results(result)
        return result

    def winrate_race(self, race_one:str, race_two:str = "all") -> float:
        """
        Calculates win rates based on the races of the players in SC2 matches.

        Retrieves all games from the SC2 database, determines the races of the players in each game,
        identifies the winner's race, and calculates win rates for each race matchup.
```

Sc2_extractor.py



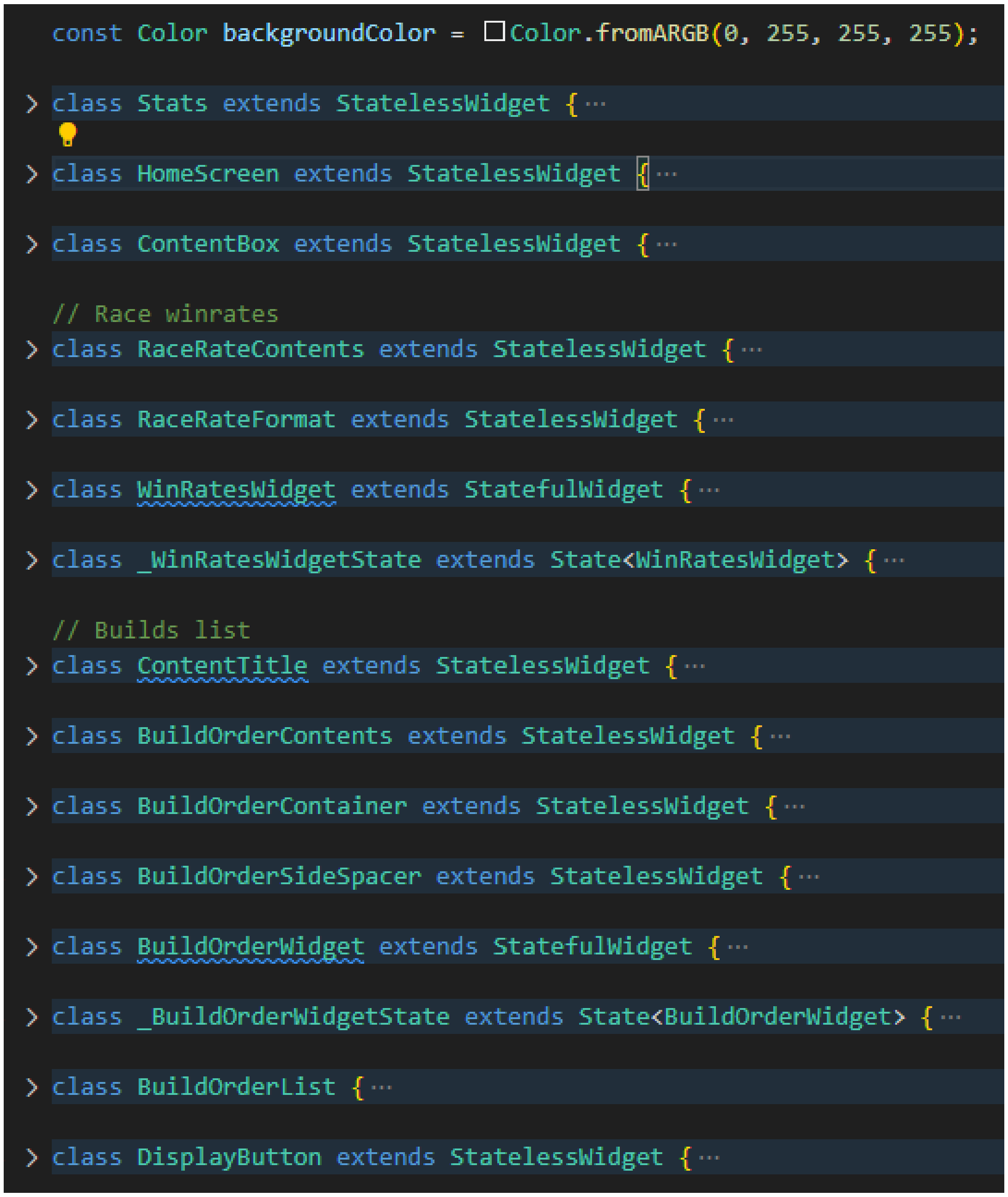
```python
class SC2Extractor(Extractor):
    """
    Extracts, filters, and pushes data from replays
    to a database
    """

    def __init__(self) -> None:
        """
        SC2Extractor constructor
        """
        super().__init__()

        self._player_data = PlayerDataStorage()
        self._game_data = GameDataStorage()
        self._play_data = PlayDataStorage()

    def extract(self) -> dict:
        """
        extract data from a group of replays and return a dictionary of replay data
        """
        replay_container = {}
        replay_counter = 0

        # Getting replays from folder
        for filename in os.listdir(self.folder_path):
            file_path = os.path.join(self.folder_path, filename)
            if os.path.isfile(file_path) and file_path.endswith(".SC2Replay"):
                # Filling replay dictionary
                replay_counter += 1

                # Check file loads properly
                try:
                    replay = sc2reader.load_replay(file_path, load_map=True)
                except Exception:
                    logging.warning(f"File: {file_path} - File failed to load")

                replay_container[replay_counter] = replay

            else:
                # Check file opens properly
                logging.warning("File not found or File isn't of type .SC2Replay")

        return replay_container
```

Sc2_replay_database.py



```
class SC2ReplayDB(GeneralDB):
    """...

    engine = None
    Session = None
    # ID initialization

    @classmethod
    def init(cls, db_name):...

    @classmethod
    def add_games(cls, game_list):...

    @classmethod
    def add_players(cls, player_list):...

    @classmethod
    def add_plays(cls, play_list):...

    @classmethod
    def get_player_by_name(cls, name: str) -> dict:...

    @classmethod
    def get_player_by_id(cls, id: int):...

    @classmethod
    def get_players_in_game(cls, game_id: int):...

    @classmethod
    def get_play(cls, game_id: int, player_id: int):...

    @classmethod
    def get_all_plays(cls):...

    @classmethod
    def get_all_players(cls):...

    @classmethod
    def get_all_games(cls):...

    @classmethod
    def _create_game_id(cls) -> int:...

    @classmethod
    def _create_player_id(cls) -> int:...
```

Stats.dart



```
const Color backgroundColor = Color.fromARGB(0, 255, 255, 255);

> class Stats extends StatelessWidget {···

> class HomeScreen extends StatelessWidget {···

> class ContentBox extends StatelessWidget {···

// Race winrates
> class RaceRateContents extends StatelessWidget {···

> class RaceRateFormat extends StatelessWidget {···

> class WinRatesWidget extends StatefulWidget {···

> class _WinRatesWidgetState extends State<WinRatesWidget> {···

// Builds list
> class ContentTitle extends StatelessWidget {···

> class BuildOrderContents extends StatelessWidget {···

> class BuildOrderContainer extends StatelessWidget {···

> class BuildOrderSideSpacer extends StatelessWidget {···

> class BuildOrderWidget extends StatefulWidget {···

> class _BuildOrderWidgetState extends State<BuildOrderWidget> {···

> class BuildOrderList {···

> class DisplayButton extends StatelessWidget {···
```

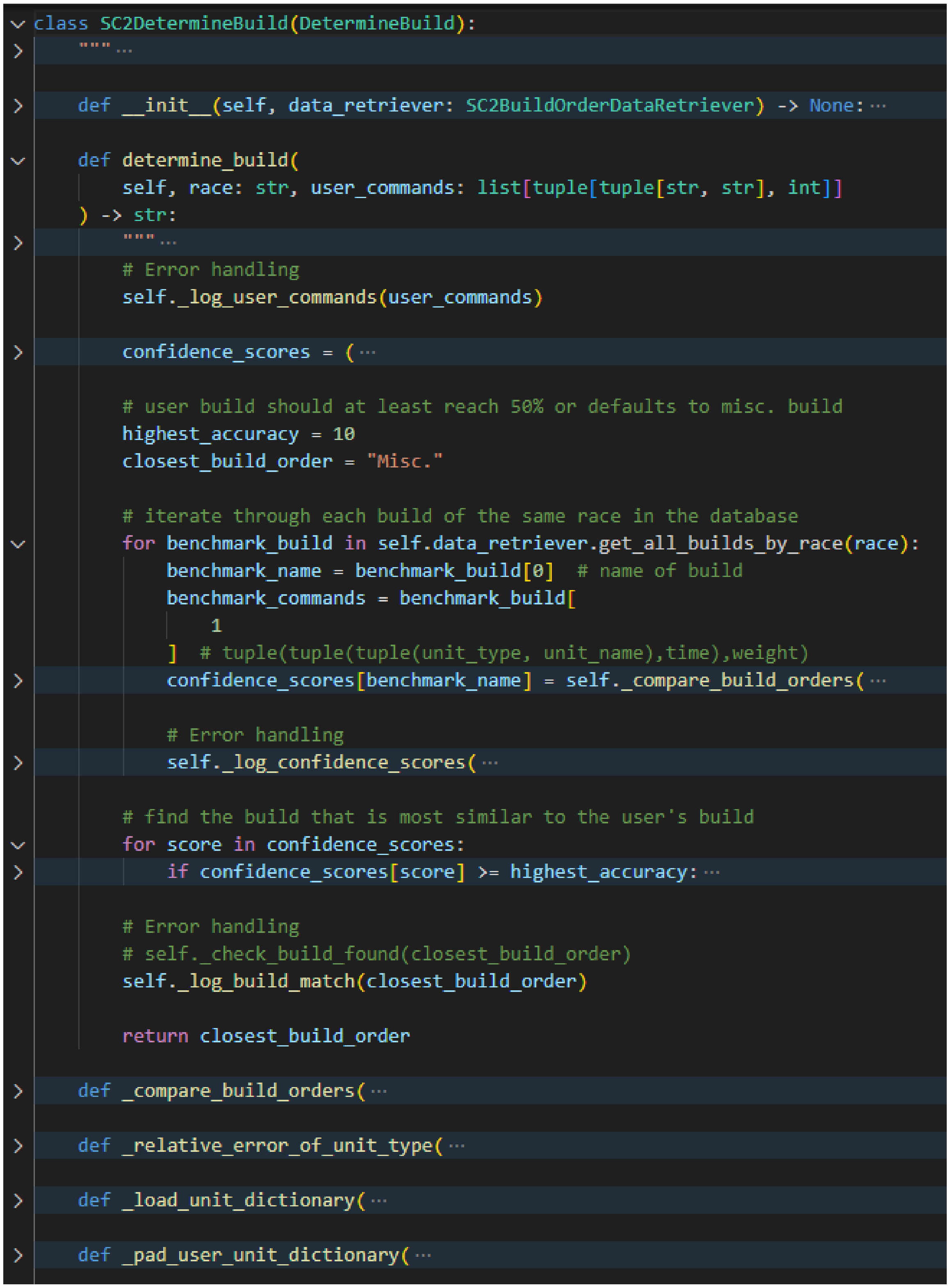
Determine Build Orders

Algorithm

The goal of the algorithm is to find out build order that the user is attempting to go in their game by analyzing the commands the user input in the game (note that each command consists of (TypeOfCommand, TimeCommandWasGiven). This is accomplished by:

1. Creating a database of build orders that we will call benchmark builds
2. Compare the user commands against each benchmark build and get a similarity score, this is done by
    a. Sorting all commands into containers by type, e.g. ‘scv’ commands get grouped with other ‘scv’ commands
    b. Sort each container in ascending order by when each command was given
    c. Compare how similar user’s containers are to benchmarks by
        i. Compare the size of each container
        ii. If the user’s container is smaller pad it with **large** numbers until it is the same size as benchmark’s container
        iii. Take the relative error between command 1’s time in benchmark to command 1’s time in user
        iv. Repeat for all commands in benchmark’s container
        v. Get the average relative error of all the commands between the two containers
    d. Repeat for all containers getting the average relative error of each container
3. Get the average relative error of all the containers
4. Repeat this for each build
5. Whichever build has the smallest average relative error is considered the most similar

Code



```
class SC2DetermineBuild(DetermineBuild):
    """...

    def __init__(self, data_retriever: SC2BuildOrderDataRetriever) -> None:...

    def determine_build(
        self, race: str, user_commands: list[tuple[tuple[str, str], int]]
    ) -> str:
        """...
        # Error handling
        self._log_user_commands(user_commands)

        confidence_scores = (...

        # user build should at least reach 50% or defaults to misc. build
        highest_accuracy = 10
        closest_build_order = "Misc."

        # iterate through each build of the same race in the database
        for benchmark_build in self.data_retriever.get_all_builds_by_race(race):
            benchmark_name = benchmark_build[0]  # name of build
            benchmark_commands = benchmark_build[
                1
            ]  # tuple(tuple(tuple(unit_type, unit_name),time),weight)
            confidence_scores[benchmark_name] = self._compare_build_orders(...

            # Error handling
            self._log_confidence_scores(...

        # find the build that is most similar to the user's build
        for score in confidence_scores:
            if confidence_scores[score] >= highest_accuracy:...

        # Error handling
        # self._check_build_found(closest_build_order)
        self._log_build_match(closest_build_order)

        return closest_build_order

    def _compare_build_orders(...

    def _relative_error_of_unit_type(...

    def _load_unit_dictionary(...

    def _pad_user_unit_dictionary(...
```

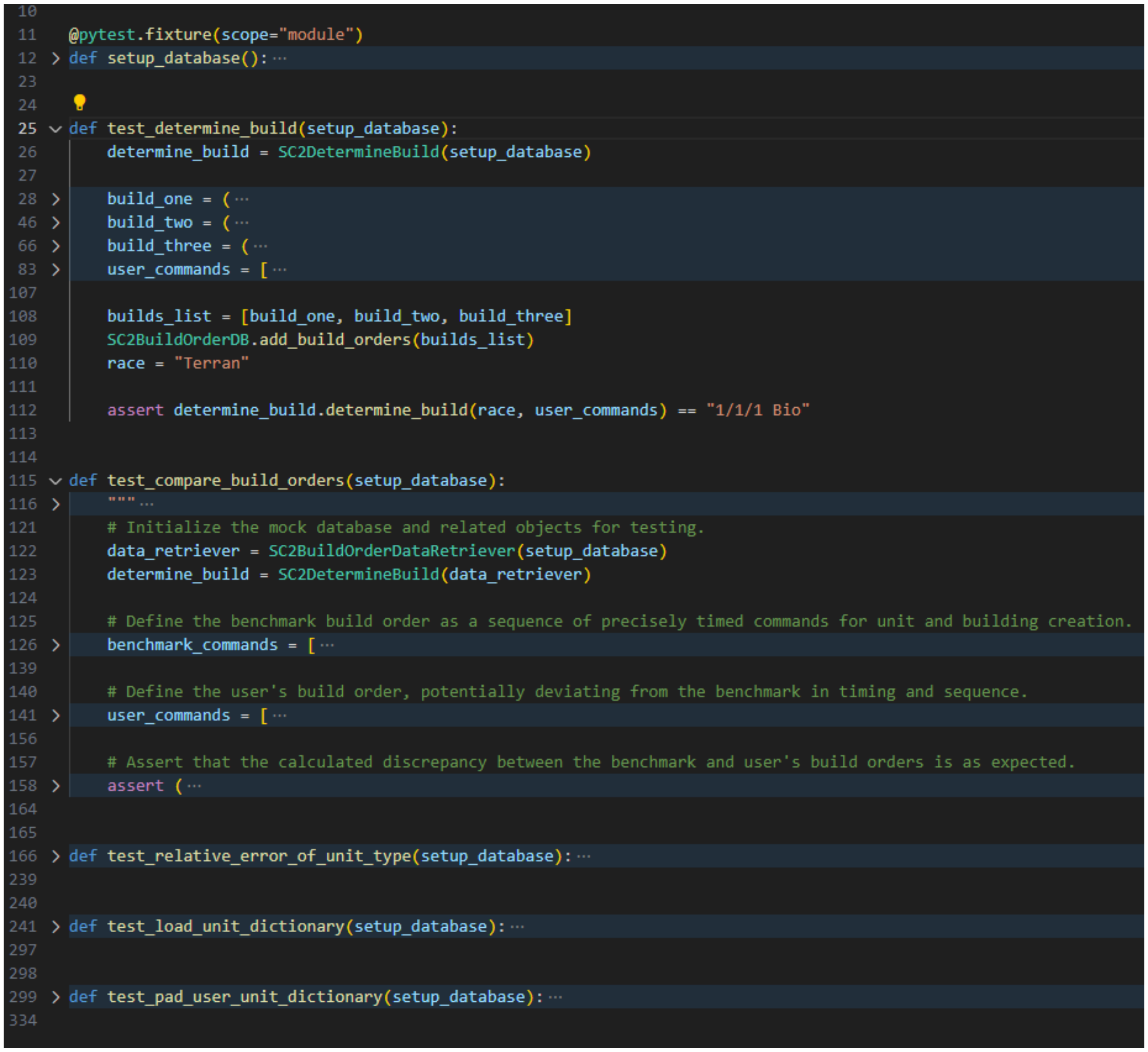
Testing

Pytest

Description: We created unit tests for all the methods in most of the python programming files.

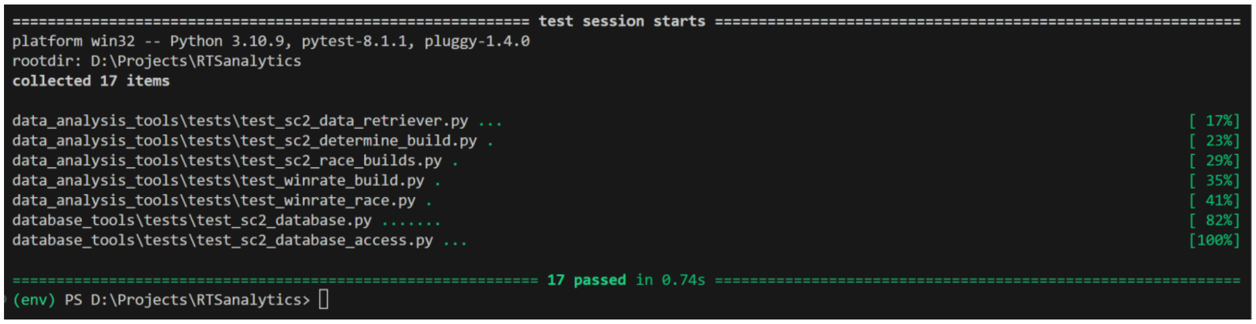
Rather than a formal testing plan we just created a test file for all relevant python files

Code



```
@pytest.fixture(scope="module")
def setup_database():···


def test_determine_build(setup_database):
    determine_build = SC2DetermineBuild(setup_database)

    build_one = (···
    build_two = (···
    build_three = (···
    user_commands = [···

    builds_list = [build_one, build_two, build_three]
    SC2BuildOrderDB.add_build_orders(builds_list)
    race = "Terran"

    assert determine_build.determine_build(race, user_commands) == "1/1/1 Bio"


def test_compare_build_orders(setup_database):
    """···
    # Initialize the mock database and related objects for testing.
    data_retriever = SC2BuildOrderDataRetriever(setup_database)
    determine_build = SC2DetermineBuild(data_retriever)

    # Define the benchmark build order as a sequence of precisely timed commands for unit and building creation.
    benchmark_commands = [···

    # Define the user's build order, potentially deviating from the benchmark in timing and sequence.
    user_commands = [···

    # Assert that the calculated discrepancy between the benchmark and user's build orders is as expected.
    assert (···


def test_relative_error_of_unit_type(setup_database):···


def test_load_unit_dictionary(setup_database):···


def test_pad_user_unit_dictionary(setup_database):···
```

Results



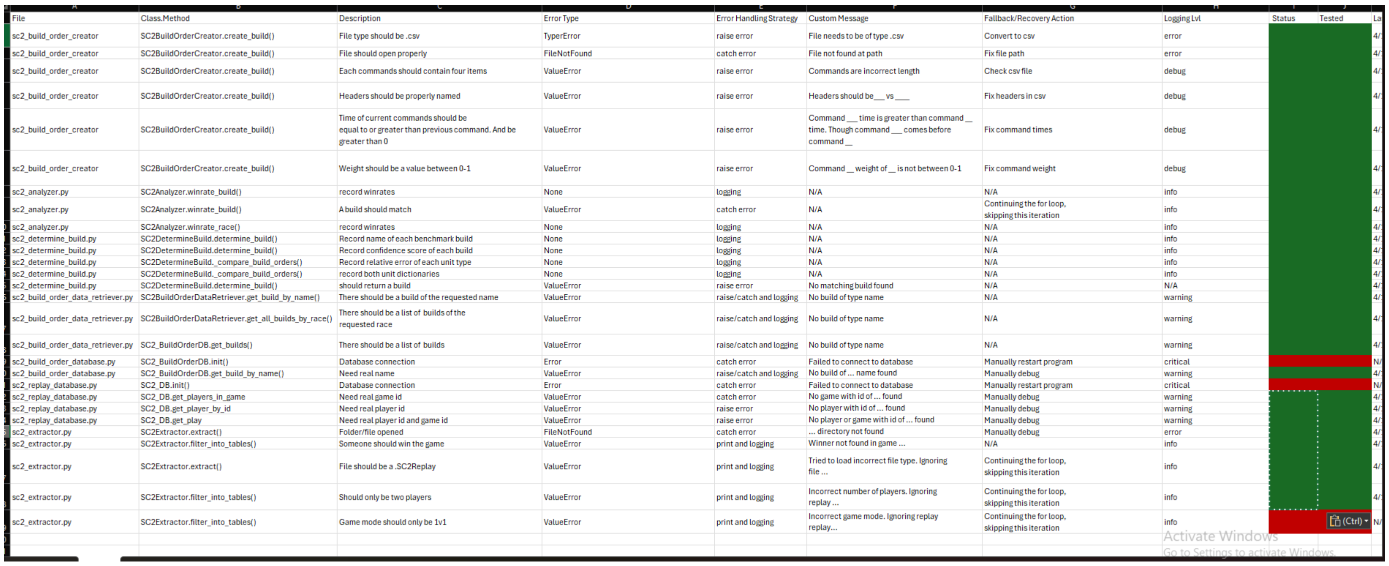
```
============================================================ test session starts ============================================================
platform win32 -- Python 3.10.9, pytest-8.1.1, pluggy-1.4.0
rootdir: D:\Projects\RTSanalytics
collected 17 items

data_analysis_tools\tests\test_sc2_data_retriever.py ...                                                                              [ 17%]
data_analysis_tools\tests\test_sc2_determine_build.py .                                                                               [ 23%]
data_analysis_tools\tests\test_sc2_race_builds.py .                                                                                   [ 29%]
data_analysis_tools\tests\test_winrate_build.py .                                                                                     [ 35%]
data_analysis_tools\tests\test_winrate_race.py .                                                                                      [ 41%]
database_tools\tests\test_sc2_database.py .......                                                                                     [ 82%]
database_tools\tests\test_sc2_database_access.py ...                                                                                  [100%]

============================================================= 17 passed in 0.74s =============================================================
(env) PS D:\Projects\RTSanalytics> 
```

Error Handling + Logging

Description: This is the error handling and logging plan, we went through each file searching for likely exceptions and errors that could occur. We documented where these were at later we came back at wrote error handling for the exceptions that we could. We also setup python logging so that each error will be logged, on top of this we added additional logging in certain areas to make it easier to debug in the future.

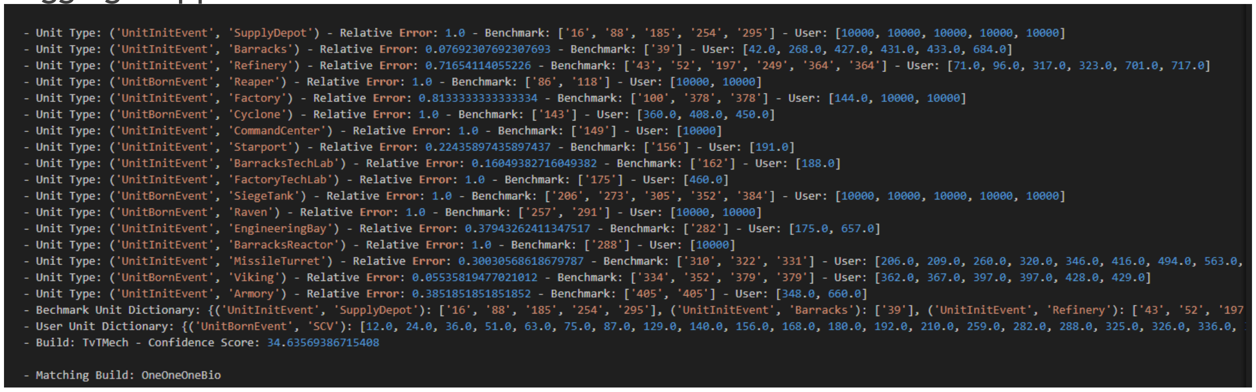
Document located at RTSanalytics\documentation\finished\error_handling_plan.xlsx

Plan

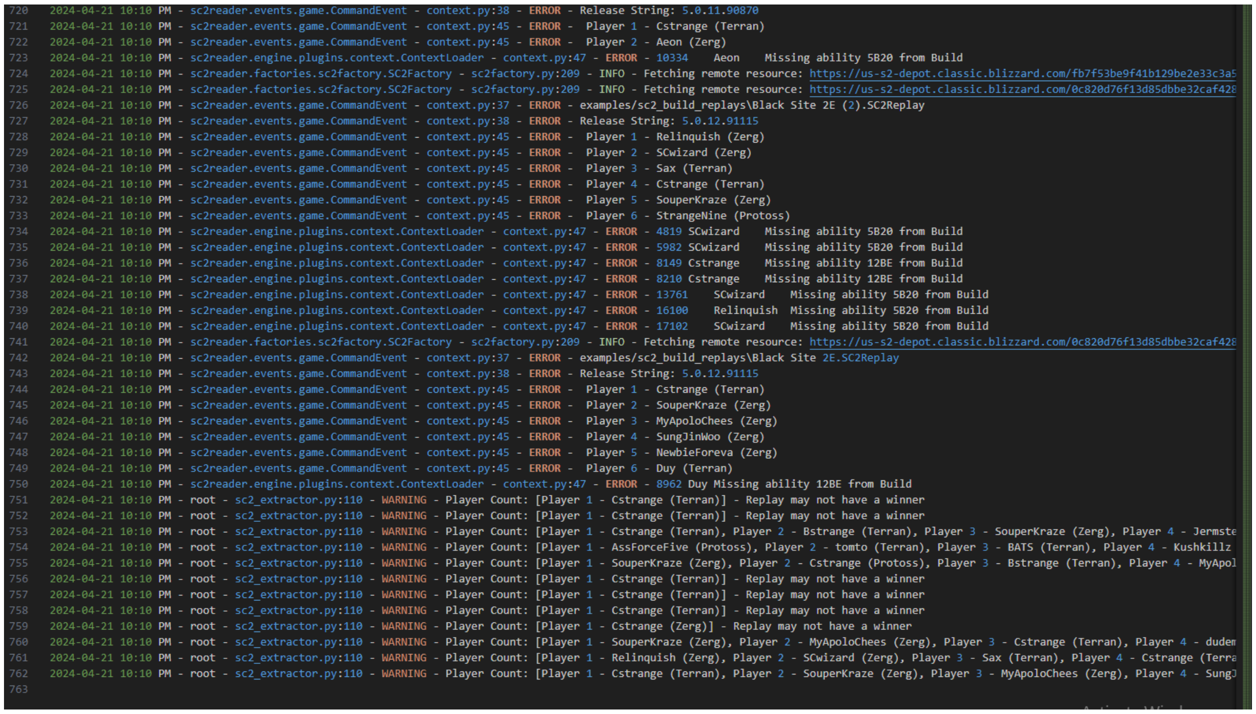


File	Class.Method	Description	Error Type	Error Handling Strategy	Custom Message	Fallback/Recovery Action	Logging Lvl	Status	Tested	La
sc2_build_order_creator	SC2BuildOrderCreator.create_build()	File type should be .csv	TyperError	raise error	File needs to be of type .csv	Convert to csv	error	🟩	🟩	4/
sc2_build_order_creator	SC2BuildOrderCreator.create_build()	File should open properly	FileNotFound	catch error	File not found at path	Fix file path	error	🟩	🟩	4/
sc2_build_order_creator	SC2BuildOrderCreator.create_build()	Each commands should contain four items	ValueError	raise error	Commands are incorrect length	Check csv file	debug	🟩	🟩	4/
sc2_build_order_creator	SC2BuildOrderCreator.create_build()	Headers should be properly named	ValueError	raise error	Headers should be___ vs ____	Fix headers in csv	debug	🟩	🟩	4/
sc2_build_order_creator	SC2BuildOrderCreator.create_build()	Time of current commands should be equal to or greater than previous command. And be greater than 0	ValueError	raise error	Command ___ time is greater than command __ time. Though command ___ comes before command __	Fix command times	debug	🟩	🟩	4/
sc2_build_order_creator	SC2BuildOrderCreator.create_build()	Weight should be a value between 0-1	ValueError	raise error	Command __ weight of __ is not between 0-1	Fix command weight	debug	🟩	🟩	4/
sc2_analyzer.py	SC2Analyzer.winrate_build()	record winrates	None	logging	N/A	N/A	info	🟩	🟩	4/
sc2_analyzer.py	SC2Analyzer.winrate_build()	A build should match	ValueError	catch error	N/A	Continuing the for loop, skipping this iteration	info	🟩	🟩	4/
sc2_analyzer.py	SC2Analyzer.winrate_race()	record winrates	None	logging	N/A	N/A	info	🟩	🟩	4/
sc2_determine_build.py	SC2DetermineBuild.determine_build()	Record name of each benchmark build	None	logging	N/A	N/A	info	🟩	🟩	4/
sc2_determine_build.py	SC2DetermineBuild.determine_build()	Record confidence score of each build	None	logging	N/A	N/A	info	🟩	🟩	4/
sc2_determine_build.py	SC2DetermineBuild._compare_build_orders()	Record relative error of each unit type	None	logging	N/A	N/A	info	🟩	🟩	4/
sc2_determine_build.py	SC2DetermineBuild._compare_build_orders()	record both unit dictionaries	None	logging	N/A	N/A	info	🟩	🟩	4/
sc2_determine_build.py	SC2DetermineBuild.determine_build()	should return a build	ValueError	raise error	No matching build found	N/A	N/A	🟩	🟩	4/
sc2_build_order_data_retriever.py	SC2BuildOrderDataRetriever.get_build_by_name()	There should be a build of the requested name	ValueError	raise/catch and logging	No build of type name	N/A	warning	🟩	🟩	4/
sc2_build_order_data_retriever.py	SC2BuildOrderDataRetriever.get_all_builds_by_race()	There should be a list of builds of the requested race	ValueError	raise/catch and logging	No build of type name	N/A	warning	🟩	🟩	4/
sc2_build_order_data_retriever.py	SC2_BuildOrderDB.get_builds()	There should be a list of builds	ValueError	raise/catch and logging	No build of type name	N/A	warning	🟩	🟩	4/
sc2_build_order_database.py	SC2_BuildOrderDB.init()	Database connection	Error	catch error	Failed to connect to database	Manually restart program	critical	🟥	🟥	N/
sc2_build_order_database.py	SC2_BuildOrderDB.get_build_by_name()	Need real name	ValueError	raise/catch and logging	No build of ... name found	Manually debug	warning	🟩	🟩	4/
sc2_replay_database.py	SC2_DB.init()	Database connection	Error	catch error	Failed to connect to database	Manually restart program	critical	🟥	🟥	N/
sc2_replay_database.py	SC2_DB.get_players_in_game	Need real game id	ValueError	catch error	No game with id of ... found	Manually debug	warning	🟩	🟩	4/
sc2_replay_database.py	SC2_DB.get_player_by_id	Need real player id	ValueError	raise error	No player with id of ... found	Manually debug	warning	🟩	🟩	4/
sc2_replay_database.py	SC2_DB.get_play	Need real player id and game id	ValueError	raise error	No player or game with id of ... found	Manually debug	warning	🟩	🟩	4/
sc2_extractor.py	SC2Extractor.extract()	Folder/file opened	FileNotFound	catch error	... directory not found	Manually debug	error	🟩	🟩	4/
sc2_extractor.py	SC2Extractor.filter_into_tables()	Someone should win the game	ValueError	print and logging	Winner not found in game ...	N/A	info	🟩	🟩	4/
sc2_extractor.py	SC2Extractor.extract()	File should be a .SC2Replay	ValueError	print and logging	Tried to load incorrect file type. Ignoring file ...	Continuing the for loop, skipping this iteration	info	🟩	🟩	4/
sc2_extractor.py	SC2Extractor.filter_into_tables()	Should only be two players	ValueError	print and logging	Incorrect number of players. Ignoring replay ...	Continuing the for loop, skipping this iteration	info	🟩	🟩	4/
sc2_extractor.py	SC2Extractor.filter_into_tables()	Game mode should only be 1v1	ValueError	print and logging	Incorrect game mode. Ignoring replay replay...	Continuing the for loop, skipping this iteration	info	🟥	🟥	N/

Logging Snippets

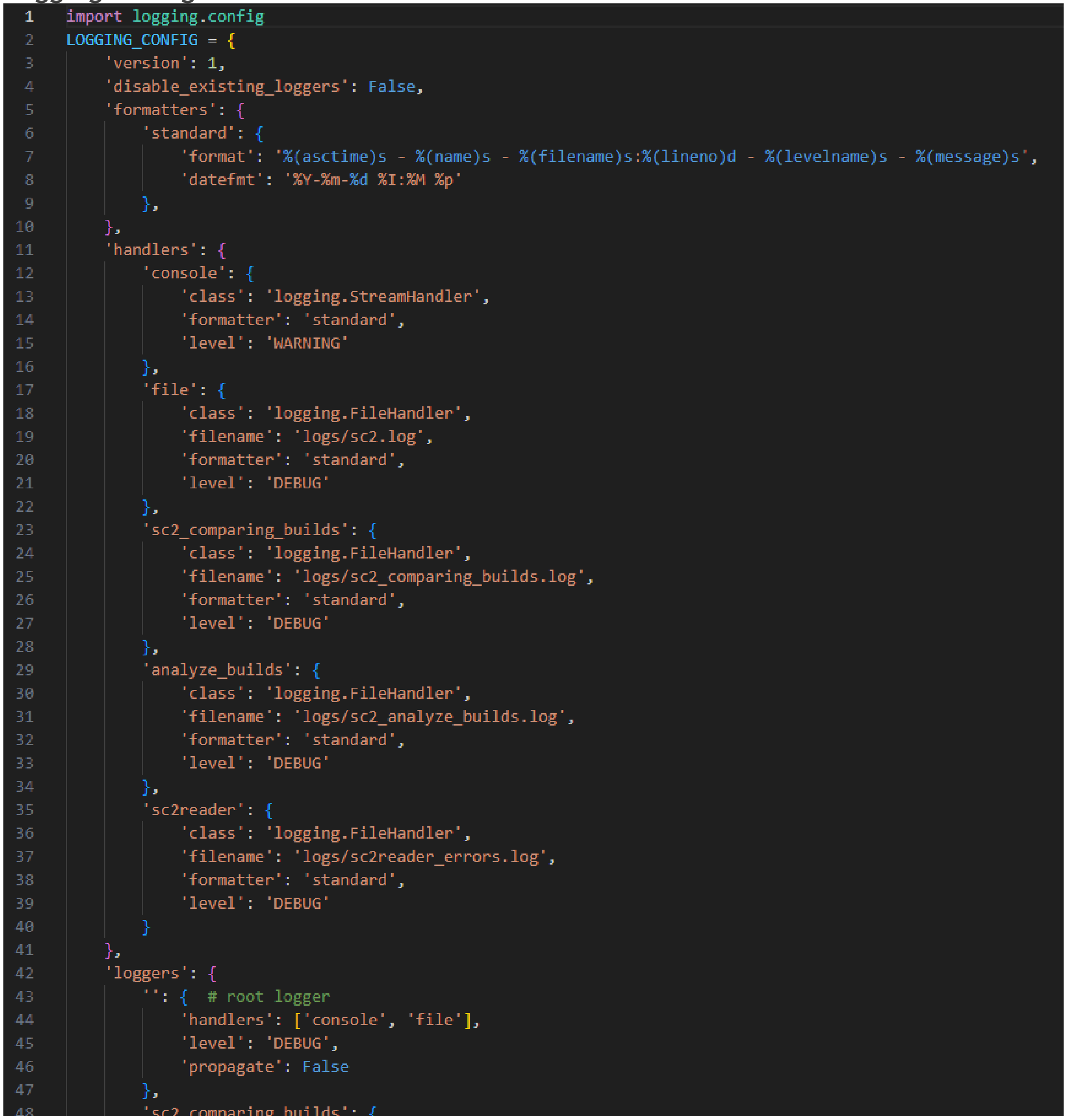


```
 - Unit Type: ('UnitInitEvent', 'SupplyDepot') - Relative Error: 1.0 - Benchmark: ['16', '88', '185', '254', '295'] - User: [10000, 10000, 10000, 10000, 10000]
 - Unit Type: ('UnitInitEvent', 'Barracks') - Relative Error: 0.07692307692307693 - Benchmark: ['39'] - User: [42.0, 268.0, 427.0, 431.0, 433.0, 684.0]
 - Unit Type: ('UnitInitEvent', 'Refinery') - Relative Error: 0.71654114055226 - Benchmark: ['43', '52', '197', '249', '364', '364'] - User: [71.0, 96.0, 317.0, 323.0, 701.0, 717.0]
 - Unit Type: ('UnitBornEvent', 'Reaper') - Relative Error: 1.0 - Benchmark: ['86', '118'] - User: [10000, 10000]
 - Unit Type: ('UnitInitEvent', 'Factory') - Relative Error: 0.8133333333333334 - Benchmark: ['100', '378', '378'] - User: [144.0, 10000, 10000]
 - Unit Type: ('UnitBornEvent', 'Cyclone') - Relative Error: 1.0 - Benchmark: ['143'] - User: [360.0, 408.0, 450.0]
 - Unit Type: ('UnitInitEvent', 'CommandCenter') - Relative Error: 1.0 - Benchmark: ['149'] - User: [10000]
 - Unit Type: ('UnitInitEvent', 'Starport') - Relative Error: 0.22435897435897437 - Benchmark: ['156'] - User: [191.0]
 - Unit Type: ('UnitInitEvent', 'BarracksTechLab') - Relative Error: 0.16049382716049382 - Benchmark: ['162'] - User: [188.0]
 - Unit Type: ('UnitInitEvent', 'FactoryTechLab') - Relative Error: 1.0 - Benchmark: ['175'] - User: [460.0]
 - Unit Type: ('UnitBornEvent', 'SiegeTank') - Relative Error: 1.0 - Benchmark: ['206', '273', '305', '352', '384'] - User: [10000, 10000, 10000, 10000, 10000]
 - Unit Type: ('UnitBornEvent', 'Raven') - Relative Error: 1.0 - Benchmark: ['257', '291'] - User: [10000, 10000]
 - Unit Type: ('UnitInitEvent', 'EngineeringBay') - Relative Error: 0.37943262411347517 - Benchmark: ['282'] - User: [175.0, 657.0]
 - Unit Type: ('UnitInitEvent', 'BarracksReactor') - Relative Error: 1.0 - Benchmark: ['288'] - User: [10000]
 - Unit Type: ('UnitInitEvent', 'MissileTurret') - Relative Error: 0.30030568618679787 - Benchmark: ['310', '322', '331'] - User: [206.0, 209.0, 260.0, 320.0, 346.0, 416.0, 494.0, 563.0,
 - Unit Type: ('UnitBornEvent', 'Viking') - Relative Error: 0.05535819477021012 - Benchmark: ['334', '352', '379', '379'] - User: [362.0, 367.0, 397.0, 397.0, 428.0, 429.0]
 - Unit Type: ('UnitInitEvent', 'Armory') - Relative Error: 0.3851851851851852 - Benchmark: ['405', '405'] - User: [348.0, 660.0]
 - Bechmark Unit Dictionary: {('UnitInitEvent', 'SupplyDepot'): ['16', '88', '185', '254', '295'], ('UnitInitEvent', 'Barracks'): ['39'], ('UnitInitEvent', 'Refinery'): ['43', '52', '197
 - User Unit Dictionary: {('UnitBornEvent', 'SCV'): [12.0, 24.0, 36.0, 51.0, 63.0, 75.0, 87.0, 129.0, 140.0, 156.0, 168.0, 180.0, 192.0, 210.0, 259.0, 282.0, 288.0, 325.0, 326.0, 336.0,
 - Build: TvTMech - Confidence Score: 34.63569386715408

 - Matching Build: OneOneOneBio
```



```
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:38 - ERROR - Release String: 5.0.11.90870
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 1 - Cstrange (Terran)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 2 - Aeon (Zerg)
2024-04-21 10:10 PM - sc2reader.engine.plugins.context.ContextLoader - context.py:47 - ERROR - 10334    Aeon    Missing ability 5B20 from Build
2024-04-21 10:10 PM - sc2reader.factories.sc2factory.SC2Factory - sc2factory.py:209 - INFO - Fetching remote resource: https://us-s2-depot.classic.blizzard.com/fb7f53be9f41b129be2e33c3a5
2024-04-21 10:10 PM - sc2reader.factories.sc2factory.SC2Factory - sc2factory.py:209 - INFO - Fetching remote resource: https://us-s2-depot.classic.blizzard.com/0c820d76f13d85dbbe32caf428
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:37 - ERROR - examples/sc2_build_replays\Black Site 2E (2).SC2Replay
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:38 - ERROR - Release String: 5.0.12.91115
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 1 - Relinquish (Zerg)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 2 - SCwizard (Zerg)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 3 - Sax (Terran)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 4 - Cstrange (Terran)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 5 - SouperKraze (Zerg)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 6 - StrangeNine (Protoss)
2024-04-21 10:10 PM - sc2reader.engine.plugins.context.ContextLoader - context.py:47 - ERROR - 4819 SCwizard    Missing ability 5B20 from Build
2024-04-21 10:10 PM - sc2reader.engine.plugins.context.ContextLoader - context.py:47 - ERROR - 5982 SCwizard    Missing ability 5B20 from Build
2024-04-21 10:10 PM - sc2reader.engine.plugins.context.ContextLoader - context.py:47 - ERROR - 8149 Cstrange    Missing ability 12BE from Build
2024-04-21 10:10 PM - sc2reader.engine.plugins.context.ContextLoader - context.py:47 - ERROR - 8210 Cstrange    Missing ability 12BE from Build
2024-04-21 10:10 PM - sc2reader.engine.plugins.context.ContextLoader - context.py:47 - ERROR - 13761    SCwizard    Missing ability 5B20 from Build
2024-04-21 10:10 PM - sc2reader.engine.plugins.context.ContextLoader - context.py:47 - ERROR - 16100    Relinquish  Missing ability 5B20 from Build
2024-04-21 10:10 PM - sc2reader.engine.plugins.context.ContextLoader - context.py:47 - ERROR - 17102    SCwizard    Missing ability 5B20 from Build
2024-04-21 10:10 PM - sc2reader.factories.sc2factory.SC2Factory - sc2factory.py:209 - INFO - Fetching remote resource: https://us-s2-depot.classic.blizzard.com/0c820d76f13d85dbbe32caf428
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:37 - ERROR - examples/sc2_build_replays\Black Site 2E.SC2Replay
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:38 - ERROR - Release String: 5.0.12.91115
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 1 - Cstrange (Terran)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 2 - SouperKraze (Zerg)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 3 - MyApoloChees (Zerg)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 4 - SungJinWoo (Zerg)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 5 - NewbieForeva (Zerg)
2024-04-21 10:10 PM - sc2reader.events.game.CommandEvent - context.py:45 - ERROR -  Player 6 - Duy (Terran)
2024-04-21 10:10 PM - sc2reader.engine.plugins.context.ContextLoader - context.py:47 - ERROR - 8962 Duy Missing ability 12BE from Build
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - Cstrange (Terran)] - Replay may not have a winner
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - Cstrange (Terran)] - Replay may not have a winner
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - Cstrange (Terran), Player 2 - Bstrange (Terran), Player 3 - SouperKraze (Zerg), Player 4 - Jermste
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - AssForceFive (Protoss), Player 2 - tomto (Terran), Player 3 - BATS (Terran), Player 4 - Kushkillz
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - SouperKraze (Zerg), Player 2 - Cstrange (Protoss), Player 3 - Bstrange (Terran), Player 4 - MyApol
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - Cstrange (Terran)] - Replay may not have a winner
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - Cstrange (Terran)] - Replay may not have a winner
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - Cstrange (Terran)] - Replay may not have a winner
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - Cstrange (Zerg)] - Replay may not have a winner
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - SouperKraze (Zerg), Player 2 - MyApoloChees (Zerg), Player 3 - Cstrange (Terran), Player 4 - dudem
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - Relinquish (Zerg), Player 2 - SCwizard (Zerg), Player 3 - Sax (Terran), Player 4 - Cstrange (Terra
2024-04-21 10:10 PM - root - sc2_extractor.py:110 - WARNING - Player Count: [Player 1 - Cstrange (Terran), Player 2 - SouperKraze (Zerg), Player 3 - MyApoloChees (Zerg), Player 4 - SungJ
```

Logging Config

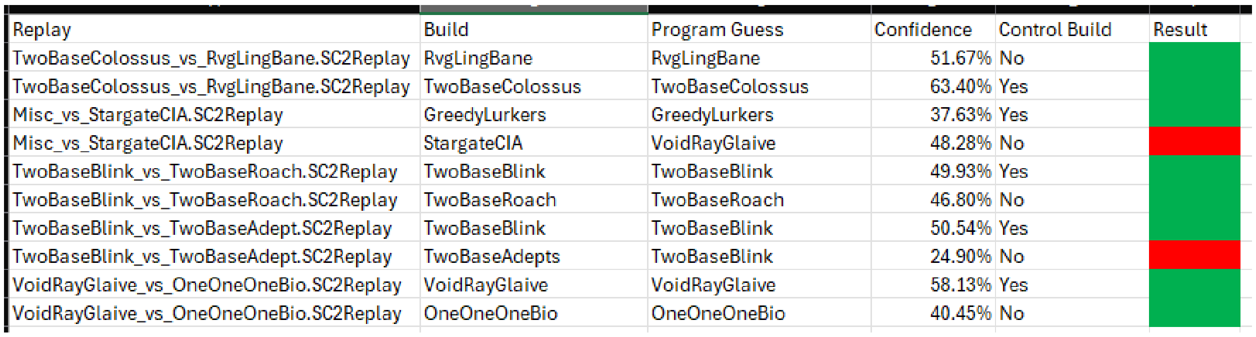


```python
import logging.config
LOGGING_CONFIG = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'standard': {
            'format': '%(asctime)s - %(name)s - %(filename)s:%(lineno)d - %(levelname)s - %(message)s',
            'datefmt': '%Y-%m-%d %I:%M %p'
        },
    },
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
            'formatter': 'standard',
            'level': 'WARNING'
        },
        'file': {
            'class': 'logging.FileHandler',
            'filename': 'logs/sc2.log',
            'formatter': 'standard',
            'level': 'DEBUG'
        },
        'sc2_comparing_builds': {
            'class': 'logging.FileHandler',
            'filename': 'logs/sc2_comparing_builds.log',
            'formatter': 'standard',
            'level': 'DEBUG'
        },
        'analyze_builds': {
            'class': 'logging.FileHandler',
            'filename': 'logs/sc2_analyze_builds.log',
            'formatter': 'standard',
            'level': 'DEBUG'
        },
        'sc2reader': {
            'class': 'logging.FileHandler',
            'filename': 'logs/sc2reader_errors.log',
            'formatter': 'standard',
            'level': 'DEBUG'
        }
    },
    'loggers': {
        '': {  # root logger
            'handlers': ['console', 'file'],
            'level': 'DEBUG',
            'propagate': False
        },
        'sc2_comparing_builds': {
```

Determine Build Accuracies

Description: This is used to track the accuracy of our 'build order determining algorithm' currently we only have anecdotal evidence for its success. Of the 10 builds we ran through the algorithm 8 were determined correctly however not a high enough variety of games were played to confidently say the algorithm is 80% accurate.

Control build refers to whether we played the build or if a random person did.



Replay	Build	Program Guess	Confidence	Control Build	Result
TwoBaseColossus_vs_RvgLingBane.SC2Replay	RvgLingBane	RvgLingBane	51.67%	No	🟩
TwoBaseColossus_vs_RvgLingBane.SC2Replay	TwoBaseColossus	TwoBaseColossus	63.40%	Yes	🟩
Misc_vs_StargateCIA.SC2Replay	GreedyLurkers	GreedyLurkers	37.63%	Yes	🟩
Misc_vs_StargateCIA.SC2Replay	StargateCIA	VoidRayGlaive	48.28%	No	🟥
TwoBaseBlink_vs_TwoBaseRoach.SC2Replay	TwoBaseBlink	TwoBaseBlink	49.93%	Yes	🟩
TwoBaseBlink_vs_TwoBaseRoach.SC2Replay	TwoBaseRoach	TwoBaseRoach	46.80%	No	🟩
TwoBaseBlink_vs_TwoBaseAdept.SC2Replay	TwoBaseBlink	TwoBaseBlink	50.54%	Yes	🟩
TwoBaseBlink_vs_TwoBaseAdept.SC2Replay	TwoBaseAdepts	TwoBaseBlink	24.90%	No	🟥
VoidRayGlaive_vs_OneOneOneBio.SC2Replay	VoidRayGlaive	VoidRayGlaive	58.13%	Yes	🟩
VoidRayGlaive_vs_OneOneOneBio.SC2Replay	OneOneOneBio	OneOneOneBio	40.45%	No	🟩

Deployment