Double Down TFT

Documentation

Introduction

1.1 Purpose

Double Down TFT is an application that helps users find match data for the Double Up gamemode in Riot's Teamfight Tactics. In Double Up, four teams of two face off in a free for all fight with interacting characters and economies. Players want to see data from their games that reflect both teammates' performance in the game.

1.2 Scope

This proof of concept application uses the Riot developer API to access data about specific accounts and matches. Double Down TFT is a Windows application written in Python 3.12. The function is to make a series of API calls on Riot's developer platform, to aggregate Double up match data.

1.3 Definitions

Summoner Name	Display name and tagline separated by "/"
puuID	ID used to identify accounts
Game	gameID for one specific game
Match	Data associated with a game ID
Player	Account and recent match data
Synergy	All matches from two players on the same team
API key	Key given to access Riot API

System Overview

2.1 System Description

Double Down TFT is a lightweight, client side application used to access TFT match data. Users provide an API key and a summoner name to search for Double Up games and common teammates. The data is aggregated and sorted by gameplay partner to be used for external analysis. This data can give insights into player synergy and compatibility, as well as game strategy.

2.2 Key Features

API integration: The application communicates with the Riot API based on the Key. The user will never interact with web requests or http responses

Summoner Name Search: Search by unique name with tagline to ensure accuracy

Match Data Export: Match data is saved to your PC as a JSON file

User Interface: Basic and intuitive computer interaction to reduce difficulty

Lightweight Processing: Optimized functions, allowing nearly any Windows10 device

2.3 Design Decisions

For the UI, Tkinter was used because of its easy python integration and quick window making capabilities. It will run very light on Windows 10 and record data from small fields.

For HTTP requests, the library Requests was used due to its variability and ease of access. It can read and submit web requests quickly in python.

Component Design

3.1 Module Breakdown

3.1.1 Startup

Description: Main file that initializes the window

Responsibility:

- Main: Keeps program running and UI screens updated
- Window Initialization: Starts “LoginPage” and “SummonerSearch”

Interactions:

- Coordinates other pages

3.1.2 Login Page

Description: Manages UI related to API key validation and information

Responsibility:

- API Key Input: Record and validate API key
- Title Display: Shows title and description of application
- Riod Dev Button: Link to Riot Developer Portal to retrieve API Key
- Help Button: Gives user instructions on using the application

Interactions:

- Records APi key to be used by Summoner Search

3.1.3 Summoner Search

Description: Manages UI related to searching a Summoner, querying and aggregating match data, and synergy statistics

Responsibility:

- Summoner Data Retrieval: Make API calls to access match data
- File Creation: Create JSON files of aggregated match data
- Synergy Statistics: calculate and display synergy information

Interactions:

- Make Match and Player classes to store information
- Receive API key from Login Page

3.1.4 Match

Description: Class object of match data based on one TFT game

Responsibility:

- JSON storage: Hold raw data from Riot

Interactions:

- Used by Summoner Search for data aggregation and synergy statistics
- Used by Player class to hold matches

3.1.5 Player

Description: A summoner and all their matches

Responsibility:

- Summoner info: hold basic info of summoner
- Match data: hold references to all matches played by summoner

Interactions:

- Used by Summoner Search to track player’s games

3.1.6 Constants

Description: Hold strings that are needed to create web request strings

Responsibility:

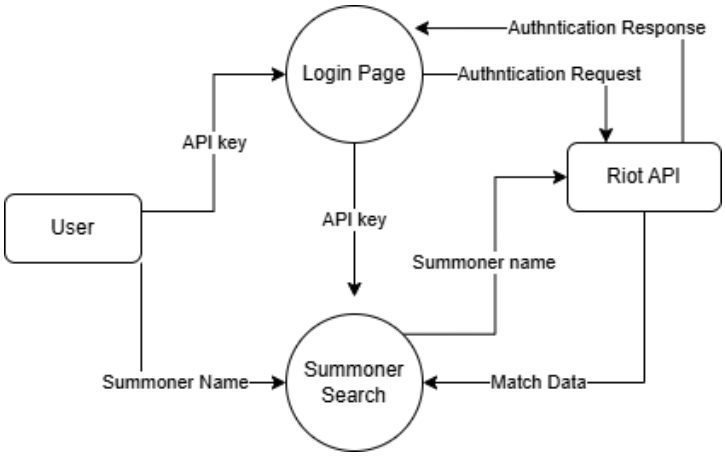
- Riot Keywords: hold keywords that are prone to change but shared across queries
- Web strings: Functions that return http request ready strings for specific queries

Interactions:

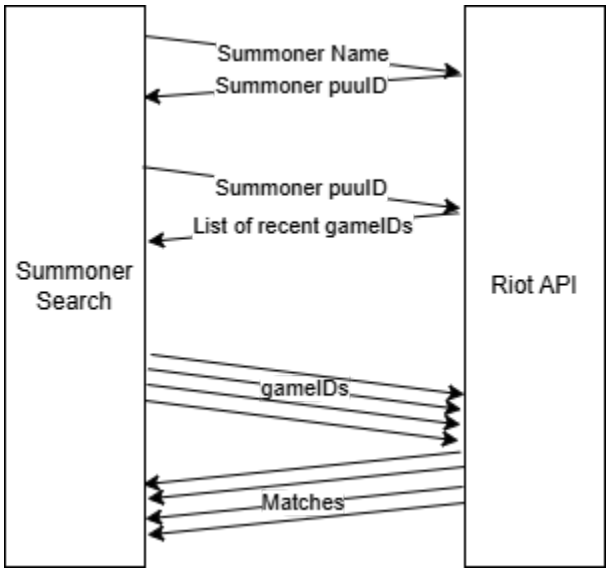
- Login Page, Summoner Search, Matches, and Payers all build their queries from the constants functions

3.2 Data Flow

3.2.1 Data Flow Diagram



3.2.2 Summoner Search Handshake



3.3 Interfaces

3.3.1 Login Page

API Key input field: Text field that records API key for validation

Title: Title of program and description

Riot Developer Portal Button: Button that routes to Riot Developer Portal
Help Button: Button that shows helpful dialogue when clicked

3.3.2 Summoner Search

Summoner Name input field: text field that records Summoner Name for API requests
Synergy Statistics: Statistics from the summoner's most popular double up teammates
File Save Dialogue: Dialogue box that helps you chose a save location for your JSON file

3.4 Components

3.4.1 Startup Components

Class: DoubleDownTFT (Tkinter window)
Attributes:

- title: window title - String
- geometry: resolution - String
- configure: color - String
- key: api key - String
- self.pages: All pages in UI - Dict
- page: current page - Tkinter Page

Functions:

- configure(): changes color and other specs
- show_page(): displays requested page

3.4.2 Login Page Components

Class: LoginPage (Tkinter page)
Attributes:

- controller: parent window - Tkinter window
- help_text: text that displays after help button - String
- title_label: main Double Down Title - Tkinter label
- subtitle_label: subtitle text - Tkinter label

- api_label: text next to input box - Tkinter label
- api_entry: input field for API key - Tkinter widget
- invalid_label: text that displays after invalid API key entry - Tkinter label
- dev_link: button link to Riot Developer Portal - Tkinter widget
- help_link: button to display help text - Tkinter widget

Functions:

- open_riot_dev: opens riot developer portal link in new tab
    - Args: event (button pressed)
- record_input: validate api key. If valid, go to next page, if not call error function
    - Args: API key
- show_help: displays instructions on screen
- badKey: displays error message

3.4.3 Summoner Search Components

Class: summoner Search (Tkinter Page)

Attributes:

- controller: parent window - Tkinter window
- label: input field label - Tkinter label
- name_entry: input field for Summoner name - Tkinter widget

Functions:

- getUserName: get username from input field to pass to synergy function
- printSynergy: prints synergy statistics from summoners past games

3.4.4 Match Components

Class: Match (Custom Class)

- ID: gameID of match - String
- key: api key - String
- raw: raw match data - JSON

Functions:

- getMatches: get match data from gameID

3.4.5 Player Components

Class: Player (Custom Class)

Attributes:

- name: Summoner name - String
- key: API key - String
- raw: raw player data - JSON
- puu: puuID - String
- games: list of players most recent gameIDs - JSON
- matches: list of match data of most recent matches - JSON

Functions:

- getGames: Get recent games from summoner name
    - Returns: List of gameIDs
- makeMatchList: makes list of match data from list of gameIDs
    - Returns: dictionary of match data
- getRaw: get raw player data from summoner name
    - Returns: dictionary of player data
- synergy: saves JSON of match data aggregated by teammate
    - Returns: dictionary with synergy statistics

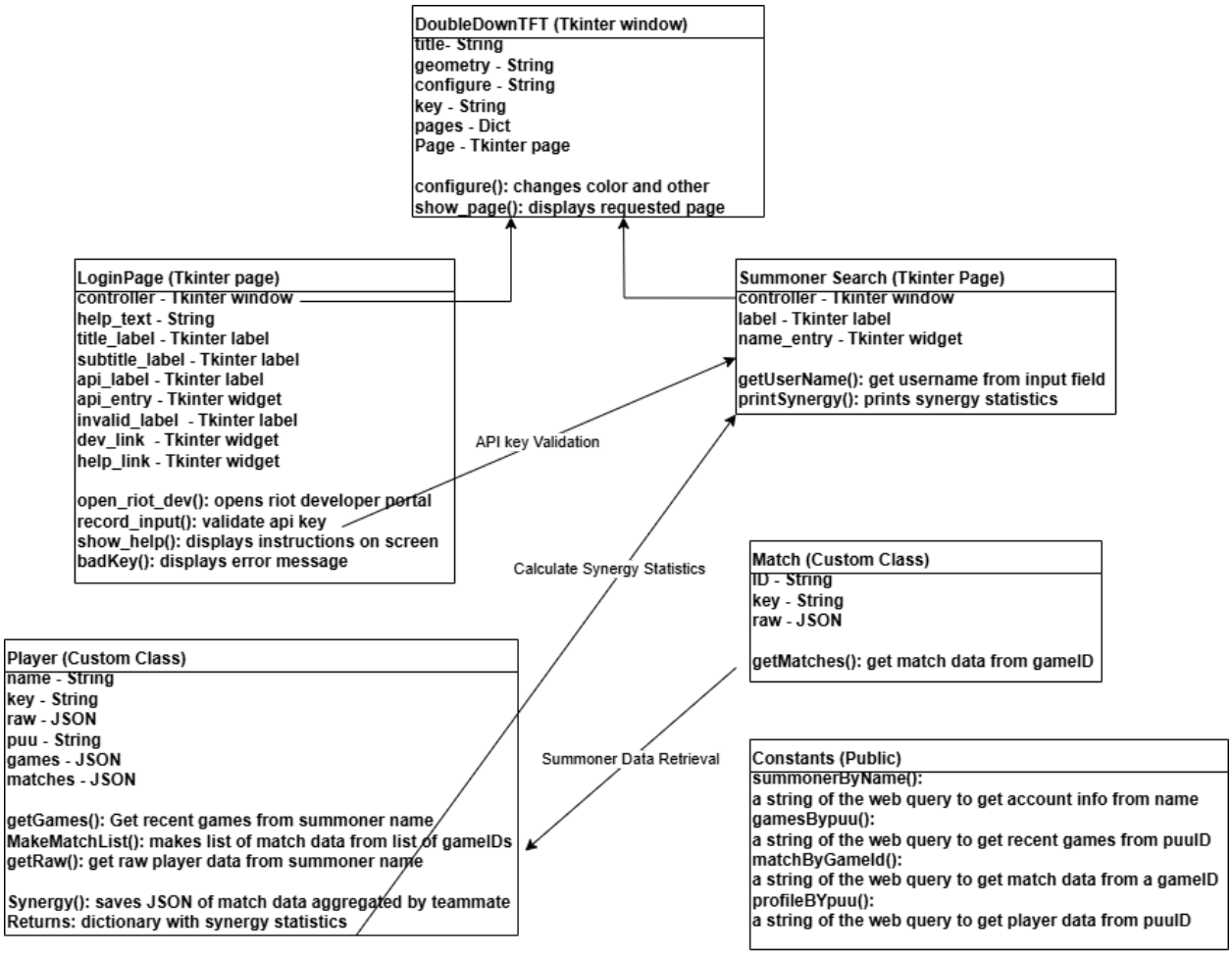
3.4.6 Constants Components

Functions:

- summonerByName
    - Return: a string of the web query to get account info from name
    - Args:
        - sumName
            - Summoner Name - String
        - key
            - API key - String
        - lolRegion
            - Account region (americas, asia, europe) - String
- gamesBypuu

        - Return: a string of the web query to get recent games from puuID
        - Args:
            - puu
                - puuID - String
            - key
                - API key - String
            - games
                - Number of games - Integer
            - tftRegion
                - Region of tft server - String
- matchByGameId
    - Return: a string of the web query to get match data from a gameID
    - Args:
        - gameID
            - gameID - String
        - key
            - API key - String
        - tftRegion
            - Region of tft server - String
- profileBYpuu
    - Return: a string of the web query to get player data from puuID
    - Args:
        - puu
            - puuID - String
        - key
            - API key - String
        - tftRegion
            - Region of tft server - String

3.5 Component Diagram



Process Design

4.1 Key Processes

4.1.1 API Key Validation

Description: The process of recording and verifying the user's API key

1. The User inputs their API Key and presses enter
2. The application sends a web request to Riot
3. The application receives a http request status

4. If validated, The user proceeds to the Summoner Search. If not, the user repeats the validation process

4.1.2 Summoner Data Retrieval

Description: The process of retrieving match data in the Summoner Search

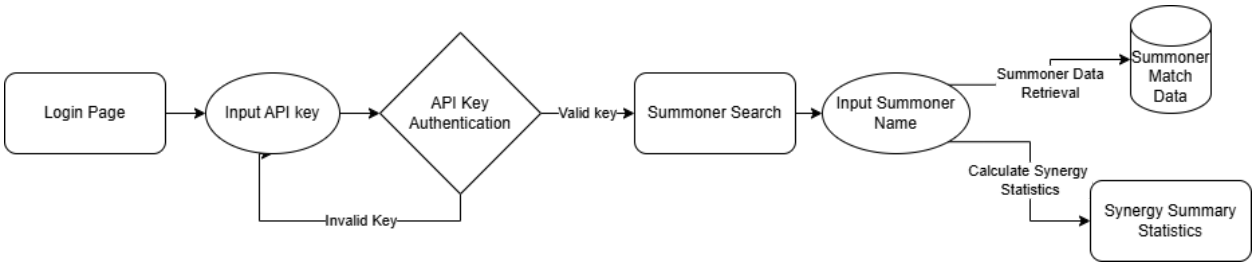
1. The user inputs a Summoner Name in the Summoner Search and presses enter
2. The application requests the user's account data and most recent match data
3. The application aggregates the matches by common teammate in a JSON file
4. The application displays a "save file" dialogue box
5. The user choses a save location on their PC
6. The application saves the JSON file to the chosen location

4.1.3 Calculate Synergy Statistics

Description: The process of calculating summary data after the Summoner Search

1. The user inputs a Summoner Name in the Summoner Search and presses enter
2. The application requests the user's account data and most recent match data
3. The application calculates win rate and total games by synergy
4. The application displays synergy summary Statistics

4.2 Process Flow Diagram



References

5.1 PEP - 8

All code was written in compliance with PEP 8 – Style Guide for Python Code.

5.2 Riot API

ALL API calls are sent to Riot Games.

Adhere to all Riot Games policies while using the application.

5.3 Libraries

All libraries were accessed in 2024 for educational purposes.

Tkinter

Requests

Webbrowser

Python 3.12

Reflection

Over the course of writing this documentation, I realized that I should have started writing my documentation much earlier. To start, it would have saved me a lot of time in the long run, but I think the much more valuable benefit would have been better organization of code. While reviewing my notes and researching online on how to write documentation, I learned more about how to organize and write efficient code. Near the end of my documentation writing I realized that I should have used another class and file for the portions related to saving the JSON file to the computer. It is not a very big deal because this is just a proof of concept, but it really made me think about my workflow. Additionally, setting up the data flow diagram was very fun and I got to use a new type of chart that I had seen in a security class for the handshake diagram.

Additionally, I was able to narrow my program into three main processes. Creating and knowing these processes in my mind made it much easier to navigate and speak about my project as a whole.

All in all, I once again overlooked the importance of keeping good documentation and paid the price later. Regardless, I ended up making a beautiful document which summarizes my project very neatly.