Local Markdown Wiki and Editor

COP701: Software Systems Laboratory
Assignment 1

by
Dharmisha Sharma Entry No. 2022MCS2062
Raajita Bhamidipaty Entry No. 2022MCS2053
Tisha Madame Entry No. 2022MCS2067

Guide: Prof Rahul Narain

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY Delhi
Hauz Khas, INDIA.
2022-2023

Acknowledgement

We would like to thank **Prof Rahul Narain** for their invaluable help and guidance. We would like to express our deep sense of gratitude to **our fellow classmates** for their important help, technical suggestions and discussions. Finally we would like to thank our family and friends for their constant moral support.

Abstract

Local Markdown wiki and editor , this project requires us to create our own Wikipedia. We have implemented this by making our own **Sportspedia** where a user could find all of the articles and information related to various sports.

The articles are supposed to be in markdown format, which are then to be parsed to HTML by our own **MARKDOWN PARSER** and then render by our framework. The framework used here is the **tkinter.**

Contents

List of Figures

Chapter 1

Introduction

Wikipedia is one of the most visited websites on the internet. All of us have used it, some of us have even edited it. Wikipedia is an example of a wiki, a collection of easy-to-edit pages which have links to each other. In this assignment,we have created a similar application which allows a user creating, updating, removing and viewing operations for articles . More precisely , we have created a **Sportspedia** which contain articles related to sports.

Our Sportspedia built in python language,maintains a wiki as a repository of articles in Markdown format. It provides a rendered view of the articles, and permits creation of new articles and editing and removal of existing articles.

Links in the Markdown files can be linked to other pages in the wiki as well as the browser. The user gets redirected to the relevant page upon clicking on the link. If the page does not exist, the user gets a choice to create it.

Chapter 2

Design

Our Sportspedia was built using the Tkinter framework .

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and macOS installs of Python.

We start from creating our main window for our application and setting its properties including size and title.Next we need to have a drop down box , side panel so that we could choose our category and access our articles.These articles would have a link to other articles as well as to those on the web also . We now proceed to adding title to our window , in a top bar frame -**SPORTSPEDIA**. Summarizing our steps-

- Create window and set its size
- Inserting GUI elements like drop down box, side panel and top panel along with title sportspedia

Our project at this step would look like :-

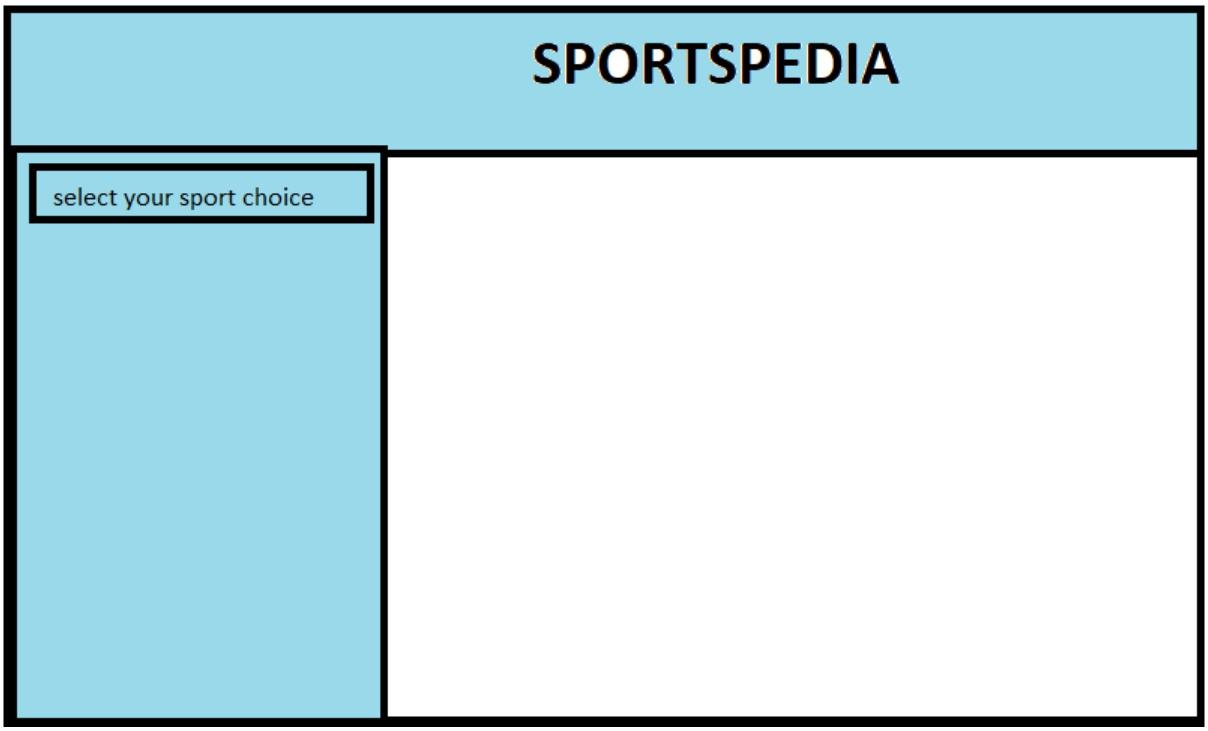

Figure 2.1: Initial Step

Next , we need to add few buttons to our project so that we could edit ,delete and add articles. Also we need to add our home page label, which is used to access home page file .The process for this would be explained in the working section . Thus we need to add-

- Add , delete and buttons to work with our articles.
- Adding home page label object .

The window after this would look like -

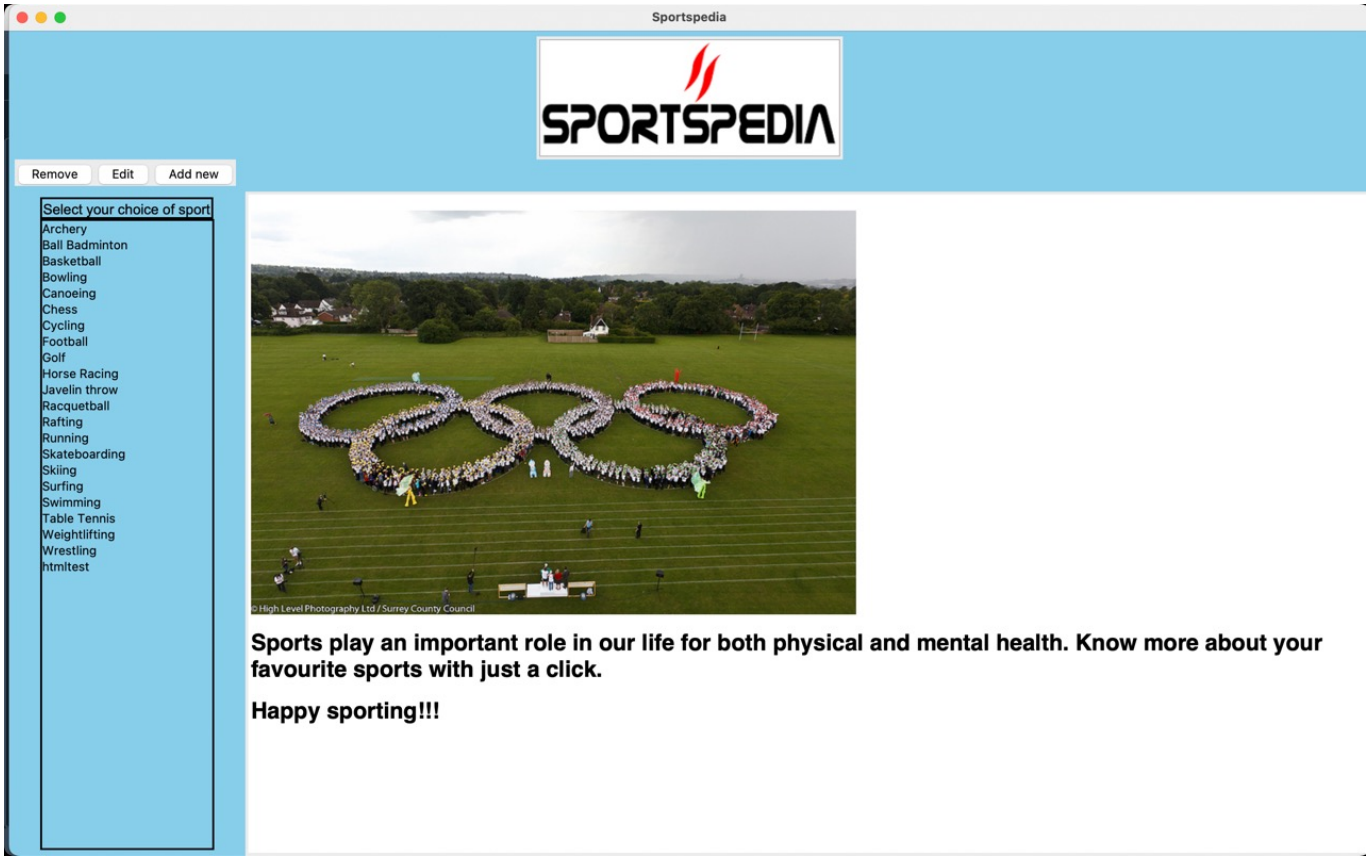

Figure 2.2: After adding buttons

Chapter 3

Working Of Components

In this section we would discuss the working of our four main events with the help of few flowcharts and diagrams

3.1 Markdown Parser

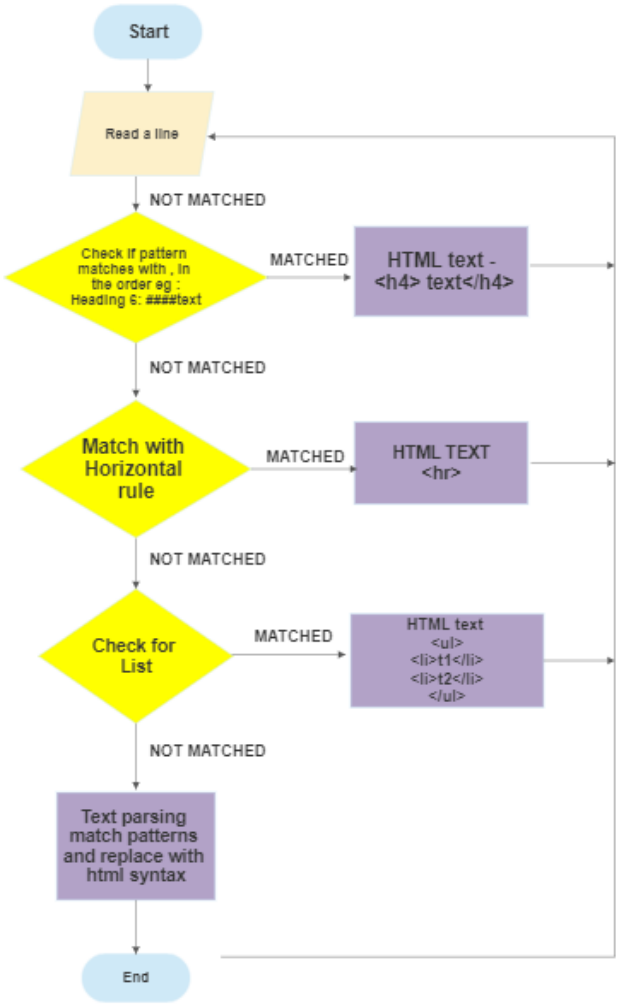

Figure 3.1: Flowchart describing parser functionality

Our parser would parse the text in the following order

- Italics and Bold

text is converted to text

- Bold

  text is converted to text

- Italics

  text is converted to text

- Image

  [text](path) is converted to <img src=path alt =text>

- Hyperlink

  [text](path) is converted to text

3.2 Event 1 - Selecting an article

User selects an article which it wants to visit. As soon as it selects an article all the elements on the central frame are cleared . The article which was selected by the user is stored in our memory in **MARKDOWN** format, which upon selection by the user is parsed to HTML by our **compiler** and then rendered and displayed on our frame.This article if the user wishes can then be further edited or removed.

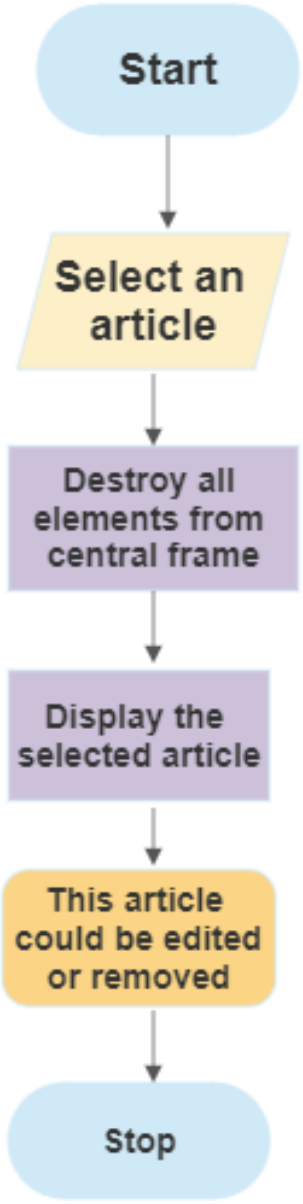

Figure 3.2: Flowchart describing Software Architecture of Application

Figure 3.3: Screenshot for select functionality

3.3 Event 2 - Add an article

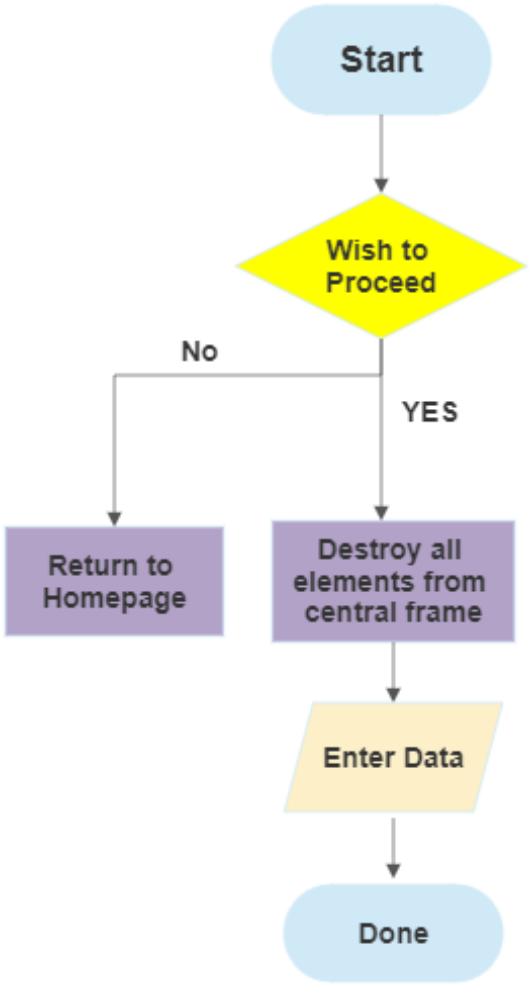

Figure 3.4: Flowchart describing procedure during adding an article

This event is triggered when we click on to the add button.After the button is clicked all the elements on the central frame are cleared and user will be asked if it wants to proceed with a dialog box. If yes then a text

box is popped up to enter the details of the new article and subsequently are saved, with a DONE button

After the user clicks done button , we fetch the title of the article and create an HTML file of format "title.md". Next to this we fetch the contents from textbox and write them down to the markdown file and then parse it to generate an HTML file.This will then be modified to be added to our side panel in a sorted way .

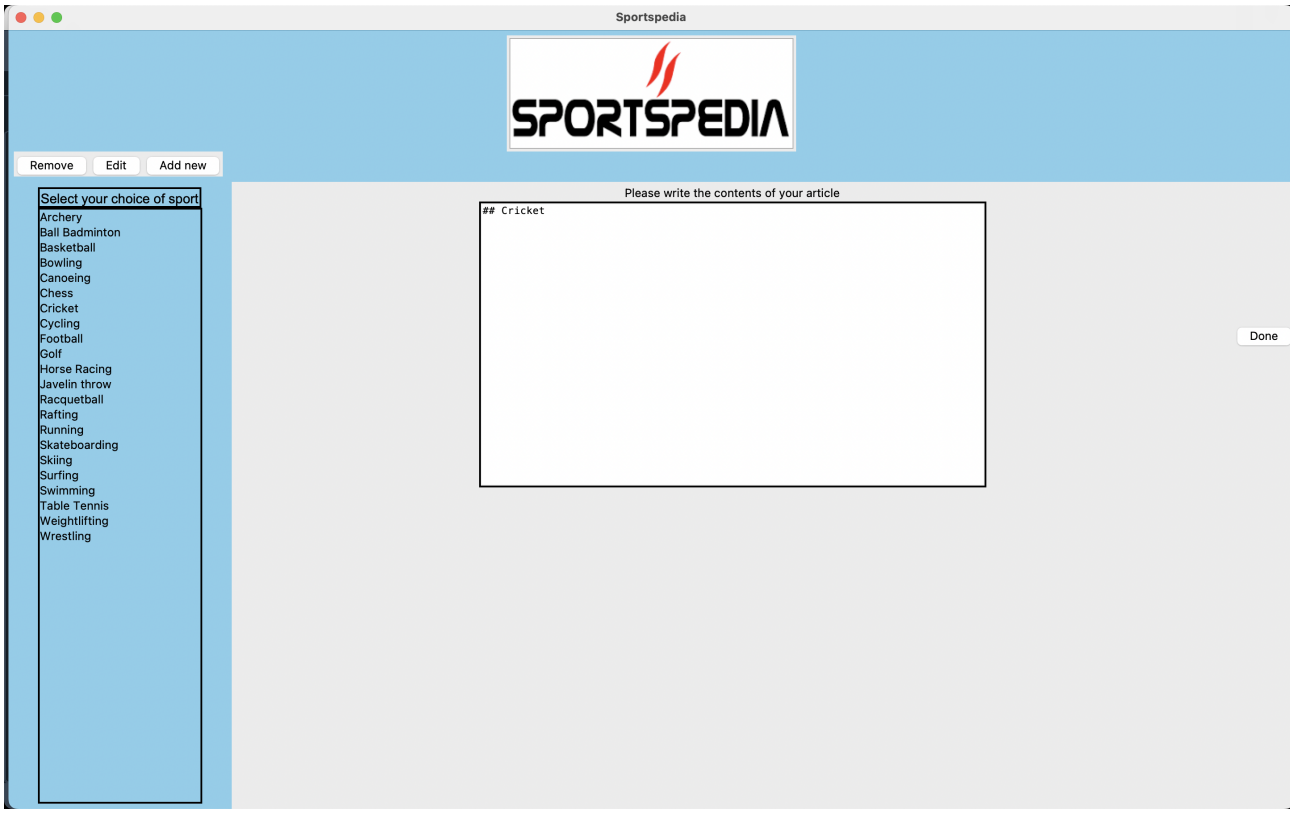

Figure 3.5: Screenshot1 for add functionality

Figure 3.6: Screenshot2 for add functionality

3.4 Event 3 - Edit an article

This event is triggered when we click on to the edit button.After the button is clicked all the elements on the central frame are cleared and user would select the article that it desired to be updated.The file is then fetched from the local directory and its contents are visible to us on a textbox which is is popped on the central frame. Here we could edit any changes that we wish to do in the article .

After the user starts editing a keypress event is generated which updates

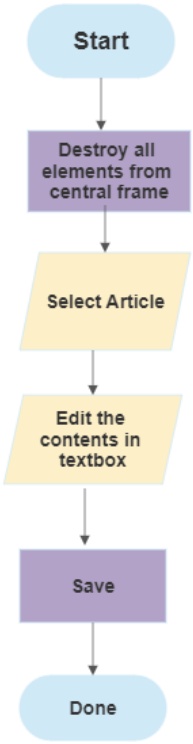

Figure 3.7: Flowchart describing procedure during editing an article

our article in runtime. Here our markdown parser parses the text, which is later converted to html and then displayed to us on the label.

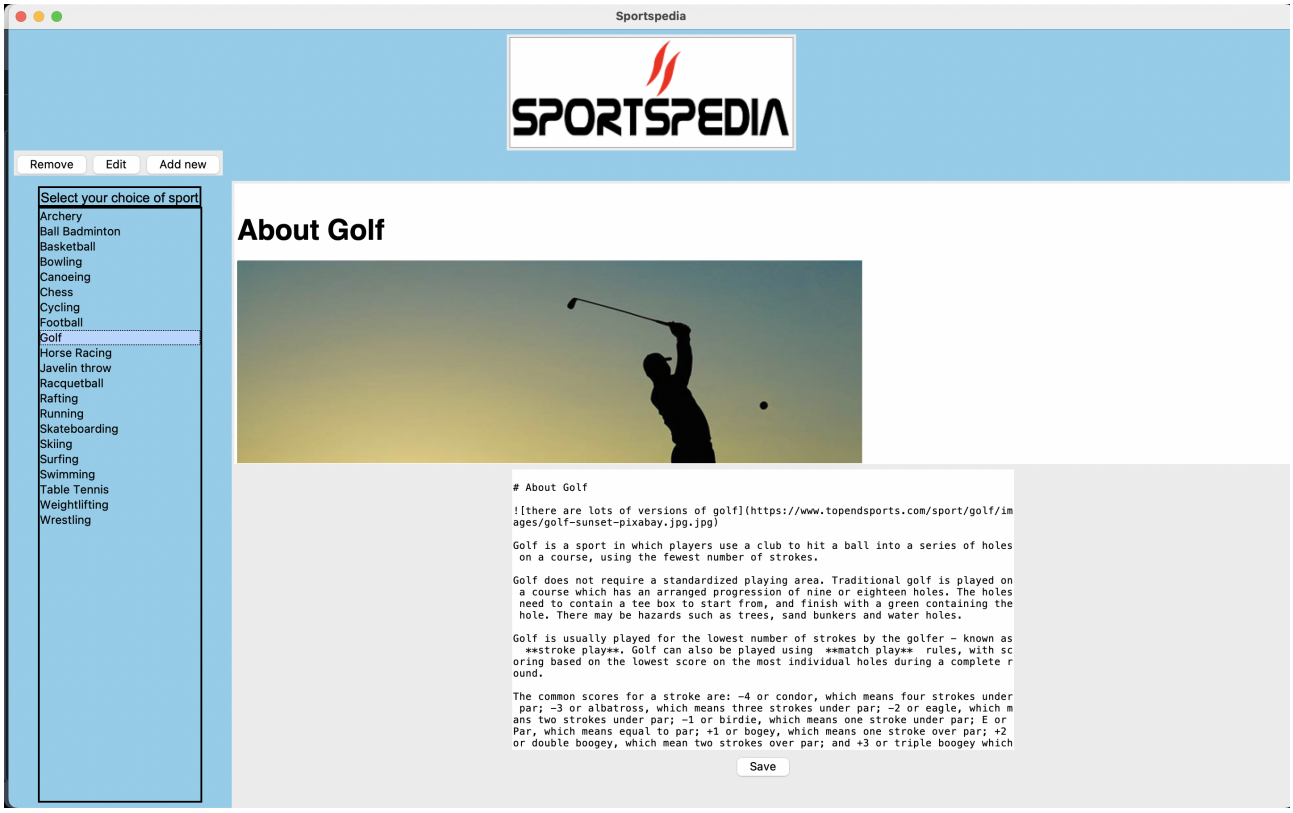

Figure 3.8: Screenshot for edit functionality

3.5 Event 4 - Remove an article

This event is triggered when we click on to the remove button.After the button is clicked the user is asked for confirmation and upon selecting yes,the article is removed from the local directory and the side panel as well.

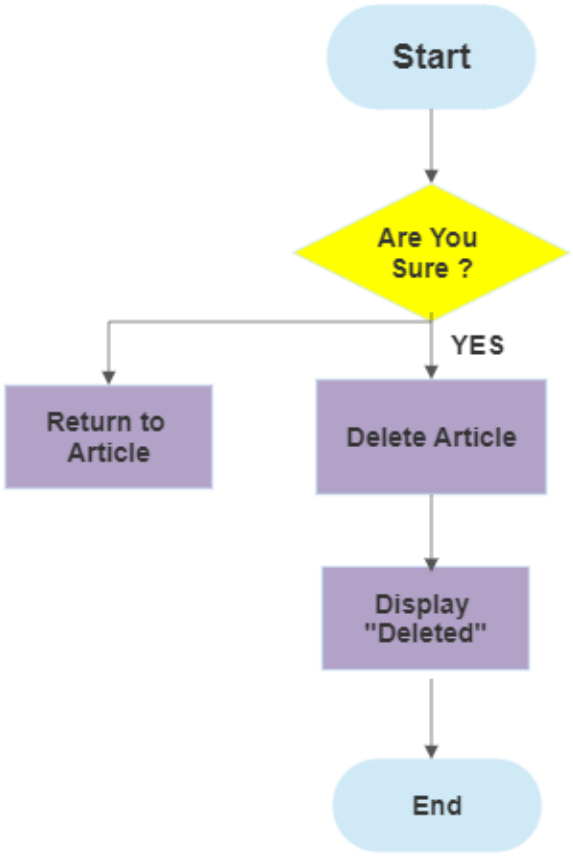

Figure 3.9: Flowchart describing procedure during removing an article

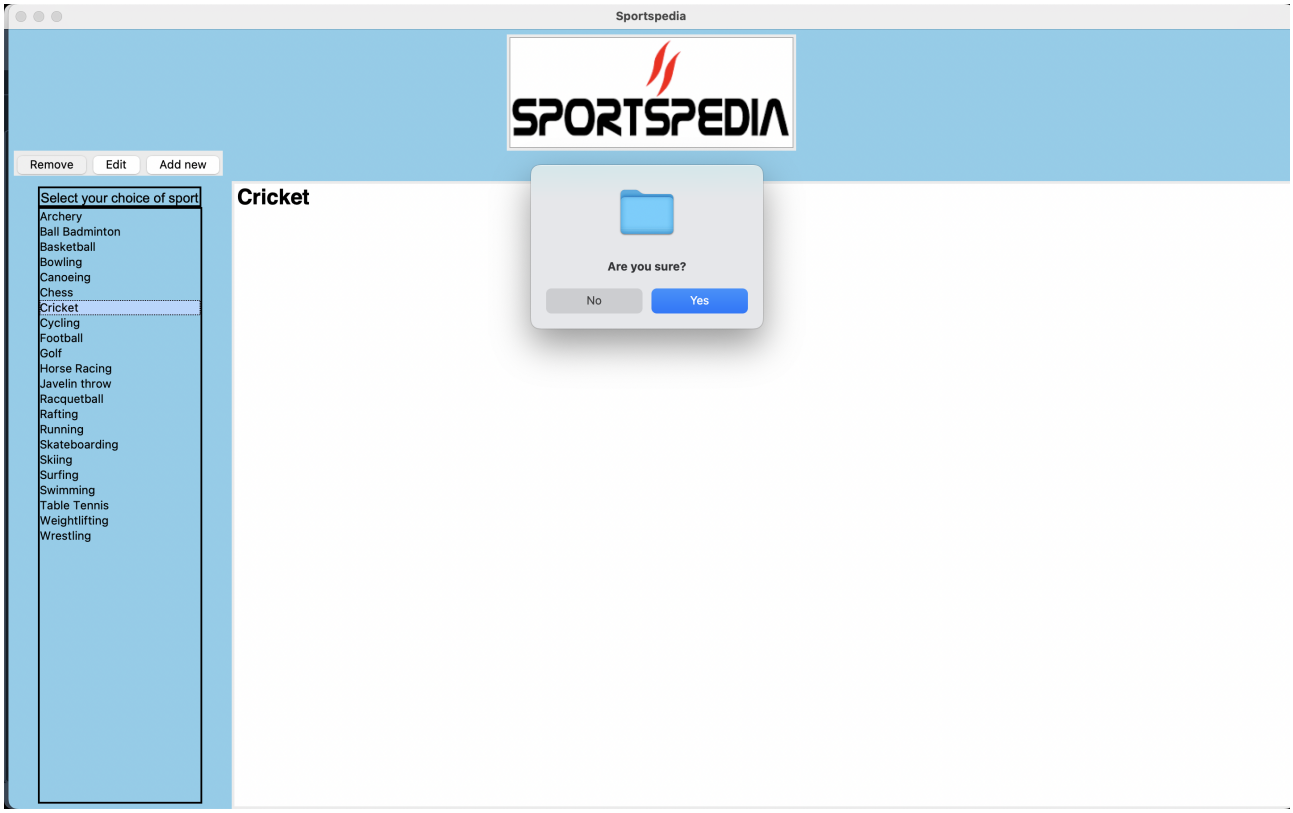

Figure 3.10: Screenshot1 for delete functionality

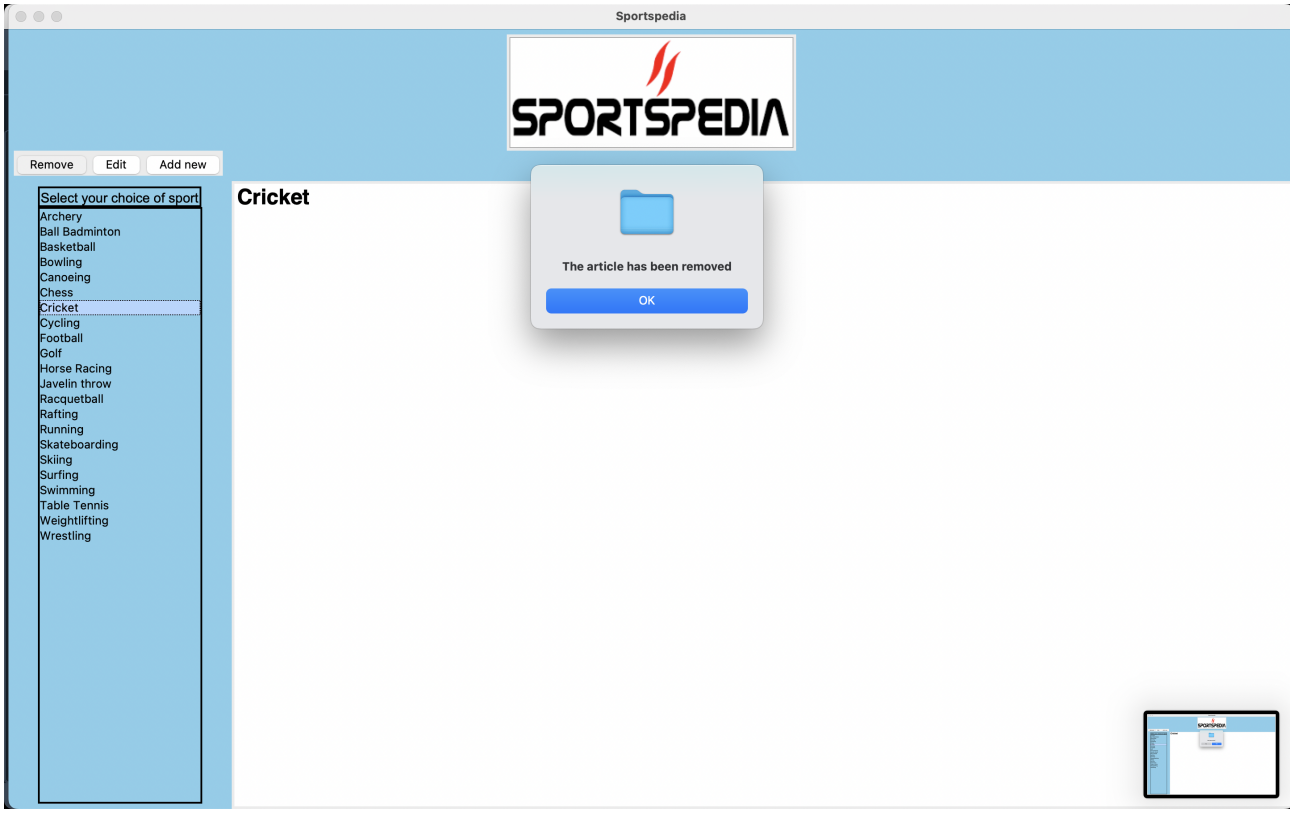

Figure 3.11: Screenshot2 for delete functionality

Chapter 4

Software Architecture

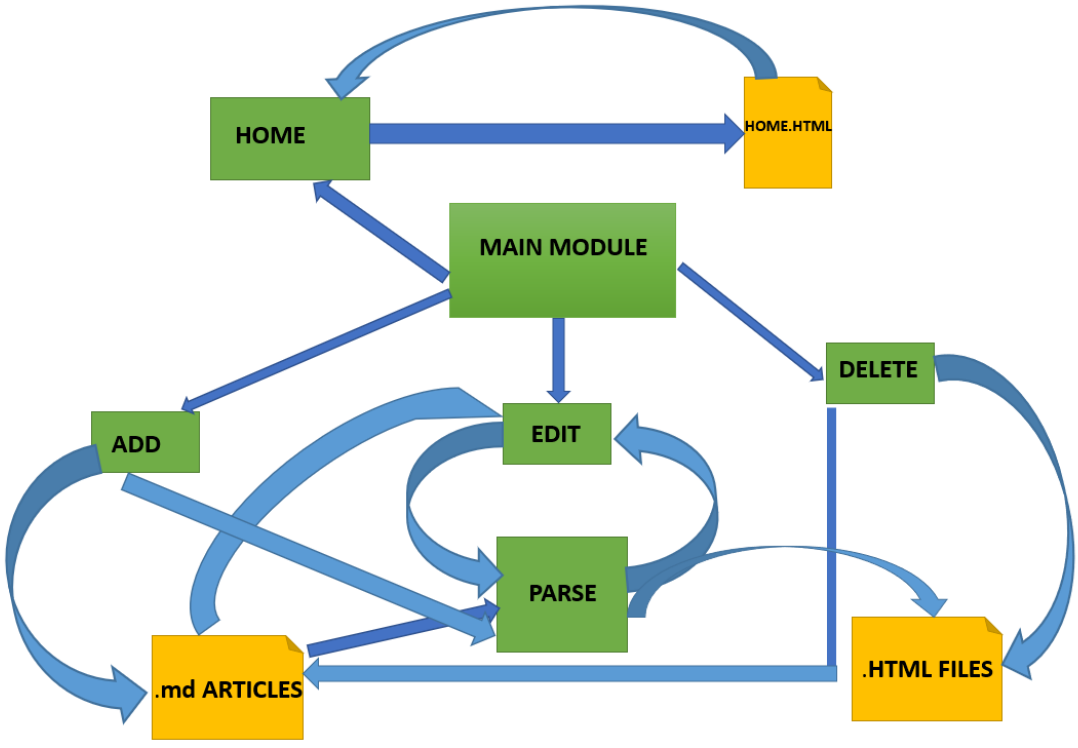

Figure 4.1: Flowchart describing Software Architecture of Application

4.1 Rationale behind the Software Architecture

The main requirement was to provide the following functionalities:

- Parsing the markdown file.
- Opening an article.
- Editing the current article.
- Adding a new article.
- Removing an existing article.

For implementing each of the above functionalities one module has been created. This constitutes the high level design. Further, within each of these the code is modularized cleanly for example in the Parse module there are separate functions to parse headings, lists, text styles etc these are called with the right precedence. In the Add module there are functions to save a file, add it to the list of files etc. This constitutes the low level design.

Chapter 5

Individual Contribution

Individual Contribution

- **Entry No - 2022MCS2062**

  Built the GUI as well as application and built functionalities like selecting an article, adding a new article, editing the current article and deleting the current article, their working and testing.

- **Entry No - 2022MCS2067**

  Worked on creating all markdown articles and their corresponding links, satisfying the two essential requirements and creating report .

- **Entry No - 2022MCS2053**

  Worked on building the parser to convert Markdown to HTML and integrating it within our project.

Chapter 6

Conclusion

Sportspedia is thus a mini wikipedia concerning about sports related articles. In our project , we have successfully built our **own MARKDOWN parser** to parse our markdown (.md) files to html and then render them using our tkinter framework in the python language. We have used few tkinter modules namely tkhtmlview ,tkhtmlweb and other too , for their specific advantages.All the requirements of adding a new article, editing and removing an existing one , selecting a new one , implementing home page are satisfied. The other two required conditions of version control , and Automation building are also satisfied parallelelly using **GITHUB** and **pybuilder** .

Chapter 7

References

For all of the articles used -

- https://www.topendsports.com/sport/list/index.htm
- https://pybuilder.io/documentation/tutorial
- https://en.wikipedia.org/wiki/List_of_sports
- https://pypi.org/project/tkhtmlview/