TO DO LIST

MINOR PROJECT REPORT

By

D.SADWIKA REDDY(RA2211003011072)

K.BHARATH ROYAL (RA2211003011082)

V.YASHWANTH(RA2211003011123)

Under the guidance of

DR.B.KANISHA

In partial fulfilment for the Course

of

21CSC203P – ADVANCED PROGRAMMING PRACTICE

in **DEPARTMENT OF COMPUTING TECHNOLOGIES**

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSC203P ADVANCED PROGRAMMING PRACTICE** entitled in "**TO DO LIST**" is the bonafide work of D.SADWIKA REDDY **(RA2211003011072), BHARATH ROYAL (RA2211003011082)** and **V.YASHWANTH (RA2211003011123)** who carried out the work under my supervision.

SIGNATURE

DR.B.KANISHA
C.TECH
SRM Institute of Science and Technology
Kattankulathur

ABSTRACT

A to-do list is a list of tasks that need to be completed, typically organized in order of priority. It is one of the simplest solutions for task management and provides a minimal and elegant way for managing tasks a person wishes to accomplish. Our aim is to design a simple and elegant website for people to keep a track of the status of their tasks. Making a to-do list is an easy and important task that everyone should do. The immense satisfaction that one gets when completing the task and marking it on the list are incomparable. Moreover, creating a list of tasks ensure you don’t miss out on anything. It’s a scientific fact that when you write the tasks that you need to complete, you are even more motivated to complete it. With this in mind, we come to build a platform which will help people create their own task list. With the help of modern tools and technologies, we strive to build a minimal and efficient to-do list which minimizes distractions and helps people achieve task management with ease and without hassle.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G , Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **Dr.B Kanisha , Assistant professor , C.Tech Department** for her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr.M.Pushpalatha ,Professor &Head , C.Tech** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

CHAPTER NO	CONTENTS	PAGE NO
1	**INTRODUCTION**	**6**
	1.1 Motivation	**6**
	1.2 Objective	**6**
	1.3 Problem Statement	**7**
	1.4 Challenges	**7**
2	**LITERATURE SURVEY**	**8**
3	**REQUIREMENT ANALYSIS**	**9**
4	**ARCHITECTURE & DESIGN**	**10**
5	**IMPLEMENTATION**	**11-15**
6	**EXPERIMENT RESULTS & ANALYSIS**	**16**
7	**CONCLUSION**	**17**
8	**REFERENCES**	**18**

1. INTRODUCTION

1.1 MOTIVATION

A to-do list is a simple prioritized list of the tasks a person must complete. People make a list of everything they need to do, ranked according to priority from the most critical task at the top to the least critical task at the bottom A few of the features of a good to-do list application include:

- Plan and execute simple actions.
- Prioritize, manage, and reason about tasks
- Record notes, action items and ideas.

To-dos are the tasks or the atomic entities that make up a to-do list. To-dos are made quickly, the bulk of them do not specify the work; instead, they are typically just comprehensive enough to serve as a valuable indicator. To be sure, to-do terminology like "Groceries" or "Car Wash" is frequently grammatically correct. Because the signal is so quick, it is only useful for a short period of time while the task is remembered. In certain cases, a simple item like a stack is enough to recall the job without the need for a note. There are clear immediate implications to adding a to-do list to a person's productivity system. The functionalities provided by a good to-do list application/system help declutter the user's mind as their pending tasks are recorded safely and they won't be forgotten. The To-do list project is a user-friendly website which helps them to keep a track of their tasks. It is a simple site which requires no sign-in/log-in or any personal details but still records your task, mark the completed tasks, and stores them even if you visit the site after a few days.

1.2 OBJECTIVE

To-do lists offer a way to increase productivity, stopping you from forgetting things, helps prioritize tasks, manage tasks effectively, use time wisely and improve time management as well as workflow. Making a to-do list is an easy and important task that everyone should do. The immense satisfaction that one gets when completing the task and marking it on the list is incomparable. Moreover, creating a list of tasks ensures you don't miss out on anything. It's a scientific fact that when you write the tasks that you need to complete, you are even more motivated to complete it.

1.3 PROBLEM STATEMENT

The current state of task management through currently available to-do list applications is a hotch-potch, to say the least. used to-do list applications are heavily bloated and provide unnecessary levels of integrations which are usually not required and clutter a user’s productivity system. A lot of good to-do list applications are not free to use and usually run ads to generate revenue, which is a huge negative point when it comes to productivity apps. Ones which are run by large companies are usually trying to pull users towards their own app ecosystem through non-sensical integrations and bloatware. Some of the most glaring issues of to-do list productivity applications are discussed in this chapter as we try to identify the problems and shortcomings of currently available solutions and build our project to overcome those shortcomings.

1.4 CHALLENGES

For most of the to-do list applications available to use, they are not independent entities or applications. Most of these applications are usually meant to be used with other applications to form a “productivity system” where a user’s calendar, clock, mail, notifications, etc. are all linked to their to-do list. While this may be desirable to most people and may work to improve productivity for some, it is very often more distracting than useful and over-integration of applications often leads to sensory overload and overwhelming frustration. Integrations are very often counter-productive and there should exist solutions that are absolutely independent entities, free of all clutter and meant to fulfill a simple and minimal purpose. Unfortunately, not many solutions like this exist currently.

2. LITERATURE SURVEY

As discussed earlier, productivity and task management entails more than just organizing virtual and physical collections and scheduling activities. Recent research has begun to address the problem of generic task management in the context of email. This development is hardly surprising, given that many digital device users are overloaded by the number of chores done through email. According to this research, any successful productivity tool must be tightly connected with email functionalities. Recent researches looked at task management strategies more generally because email and related technologies are unlikely to be the whole picture

3. REQUIREMENT ANALYSIS

1. User Interface (UI):

- Console-Based Interface : As a mini project, you can keep it simple by using the console for input and output.
- Menu System: Implement a menu system to allow users to add tasks, view tasks, mark tasks as complete, and exit the application.
- Input Validation: Ensure that the user input is validated to prevent errors.

2. Task Management:

- Task Class: Create a `Task` class with attributes such as task name, due date, priority, etc.
- Add Task: Allow users to add tasks with relevant details.
- View Tasks: Display a list of tasks with their details.
- Mark as Complete: Allow users to mark tasks as complete.

3. Storage:

- Data Storage: Implement a simple data storage mechanism to persist tasks between sessions (e.g., file storage).
- Read/Write Operations: Develop methods to read tasks from storage and write tasks to storage.

4. Date and Time Handling:

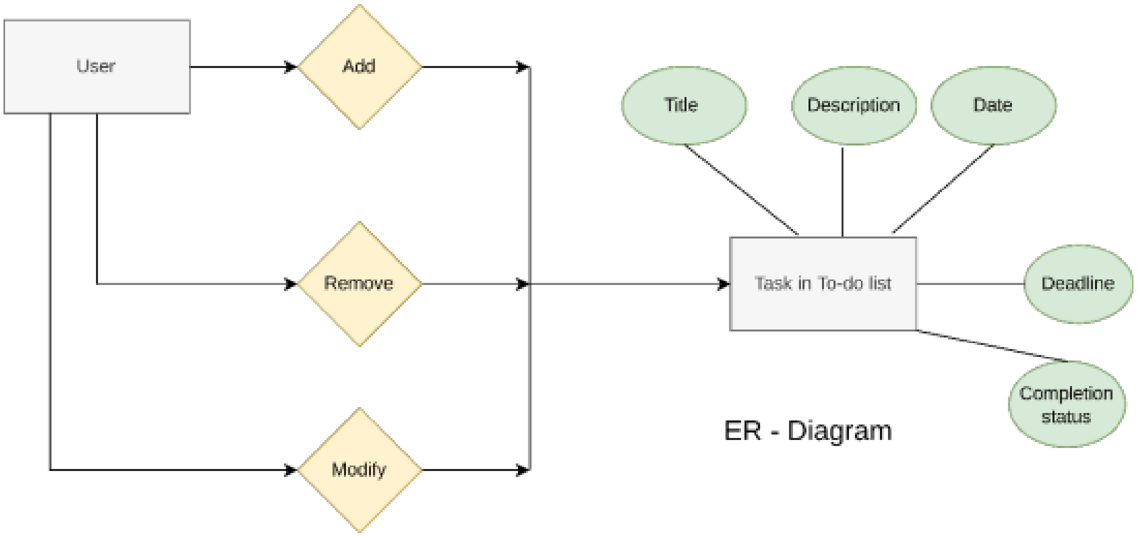
- Due Dates: Allow users to set due dates for tasks.
- Date Validation: Ensure that due dates are validated, and tasks with overdue dates are highlighted.

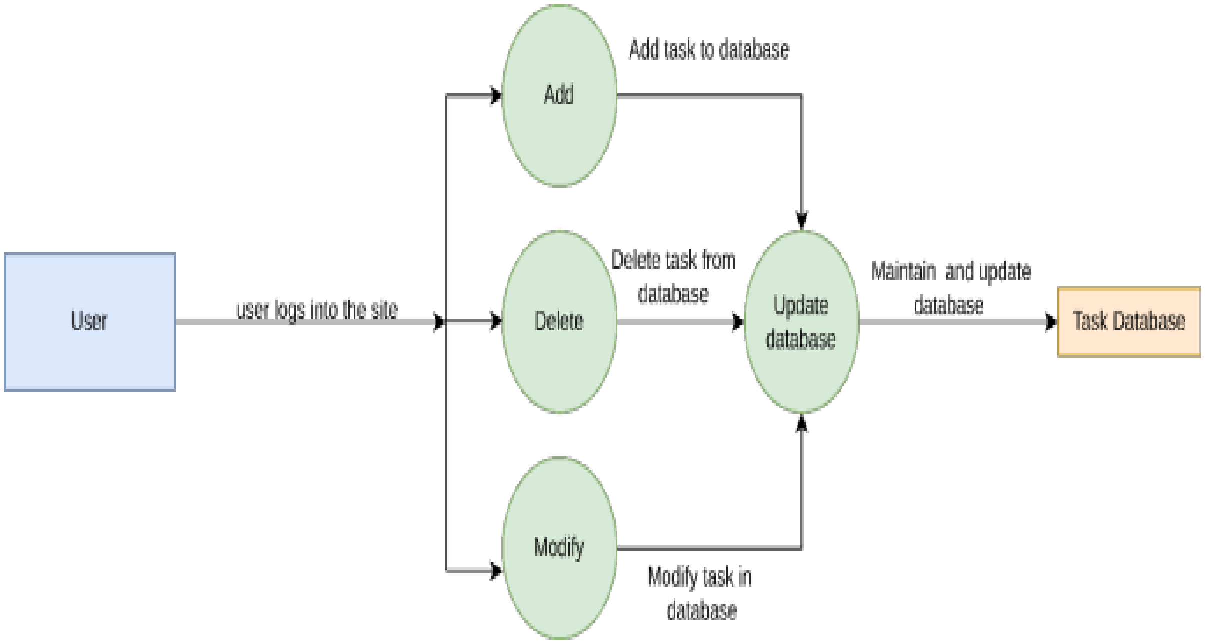
5. Priority Handling:

- Priority Levels: Assign priority levels to tasks (e.g., high, medium, low).
- Sorting: Implement functionality to sort tasks based on priority or due date.

.

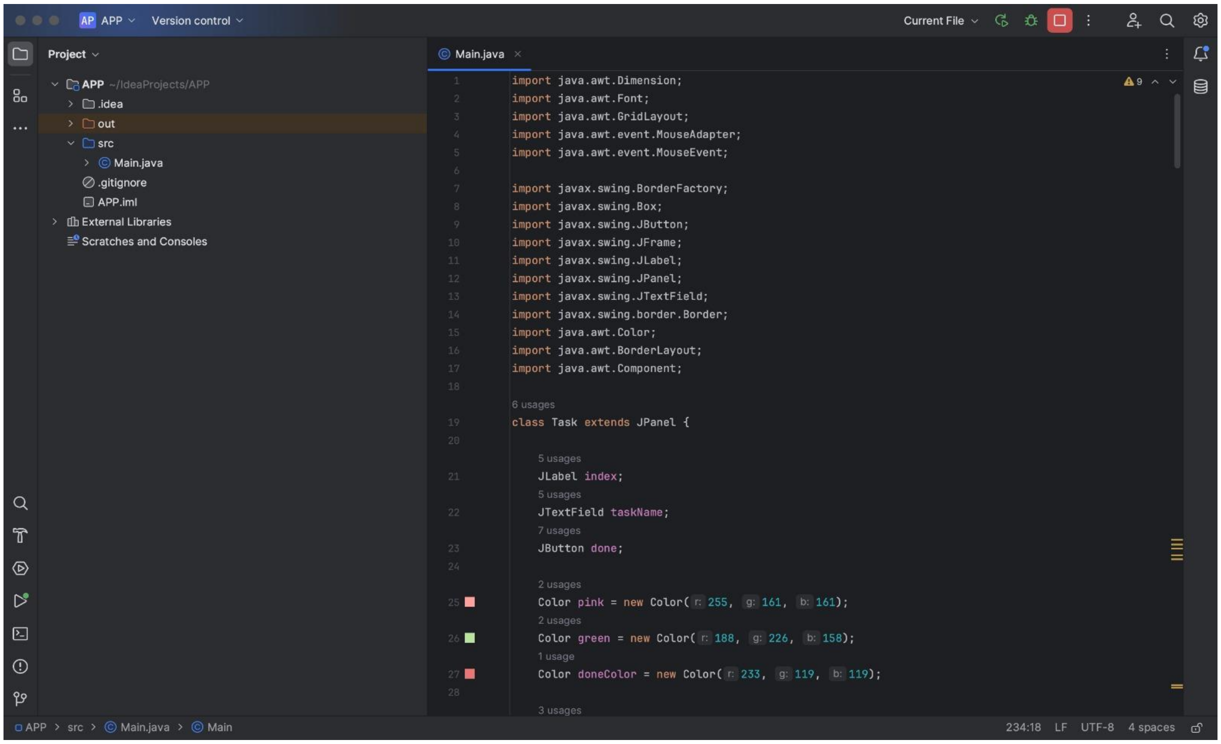
4. ARCHITECTURE AND DESIGN



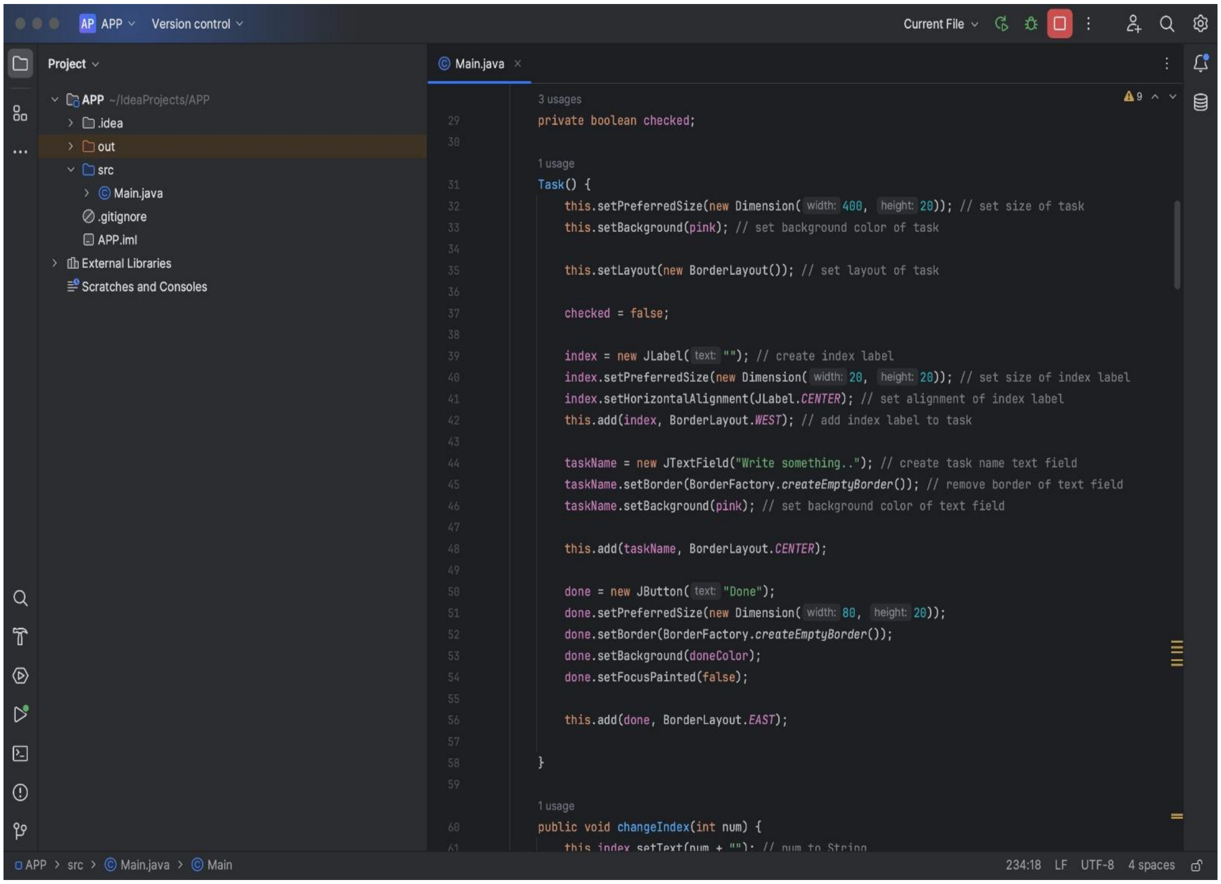


5. IMPLEMENTATION

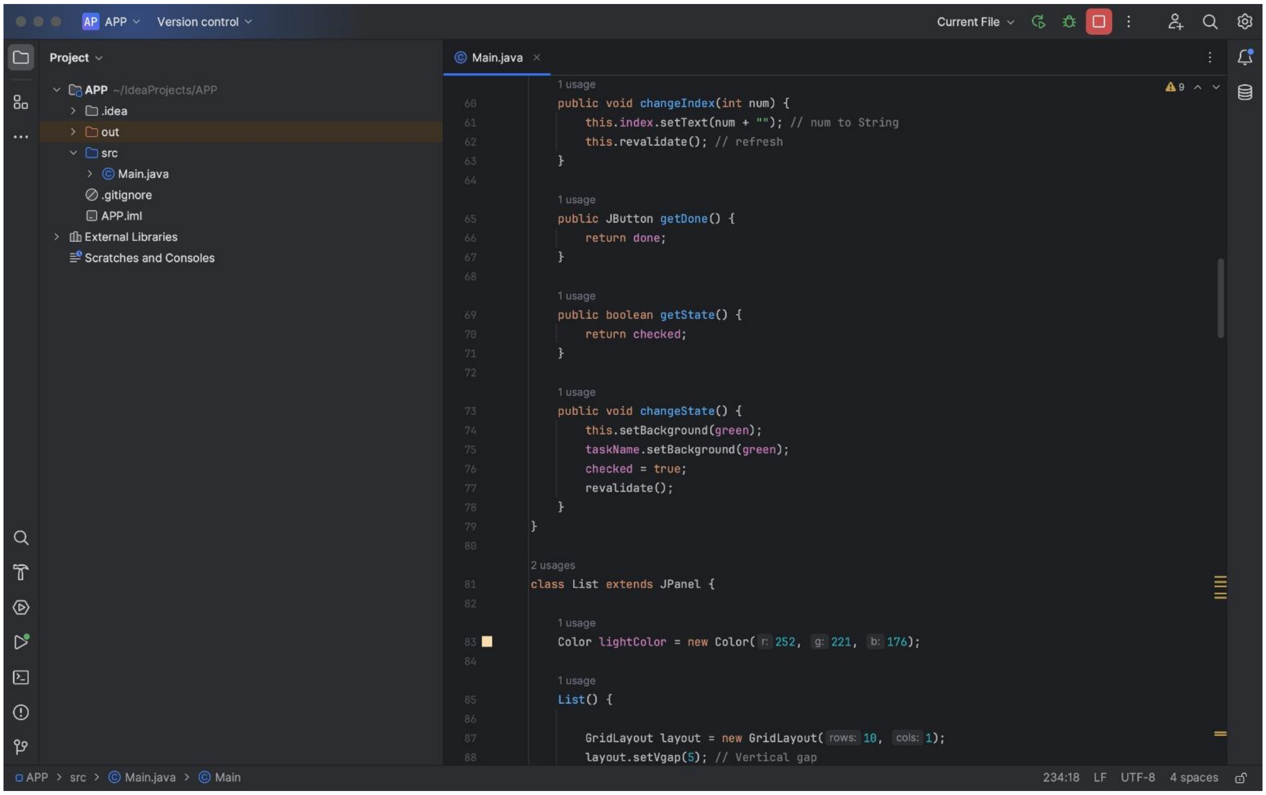
5.1 JAVA CODE:



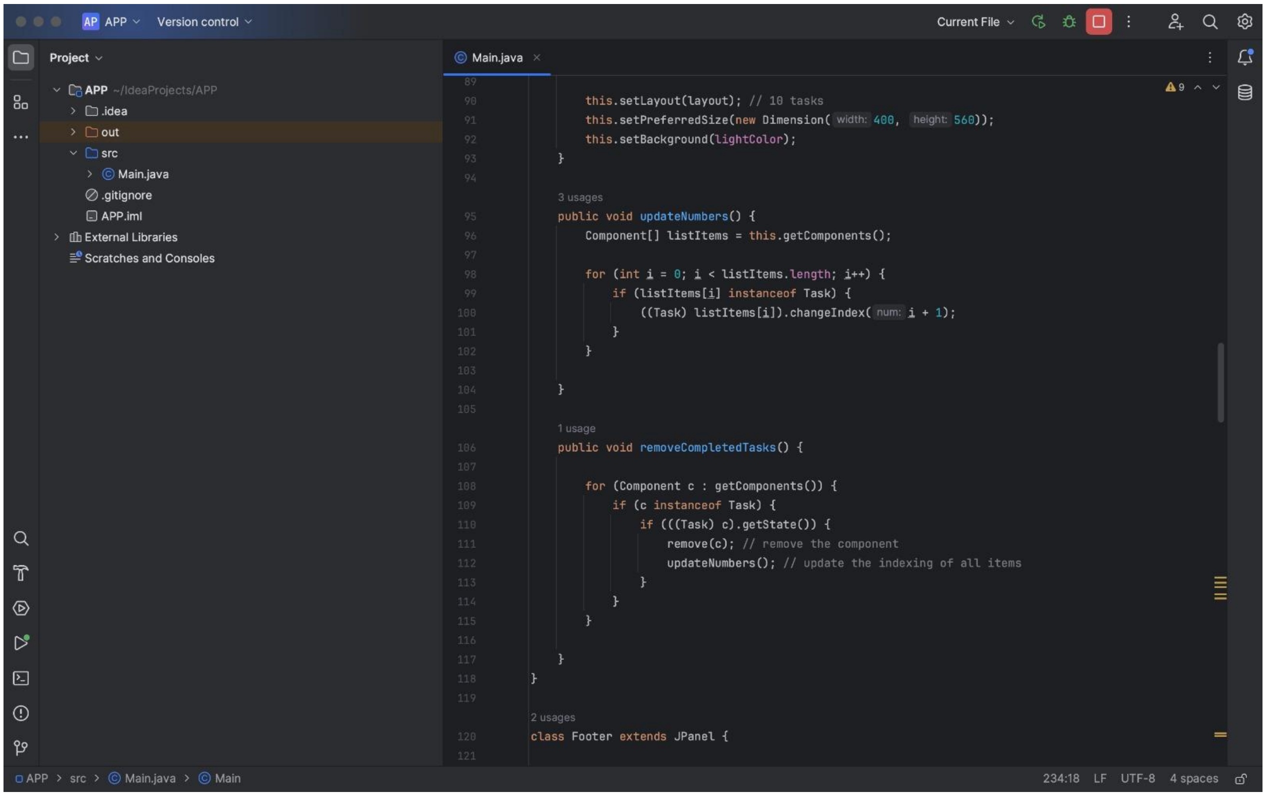
```java
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.Border;
import java.awt.Color;
import java.awt.BorderLayout;
import java.awt.Component;

6 usages
class Task extends JPanel {

    5 usages
    JLabel index;
    5 usages
    JTextField taskName;
    7 usages
    JButton done;

    2 usages
    Color pink = new Color( r: 255,  g: 161,  b: 161);
    2 usages
    Color green = new Color( r: 188,  g: 226,  b: 158);
    1 usage
    Color doneColor = new Color( r: 233,  g: 119,  b: 119);

    3 usages
```



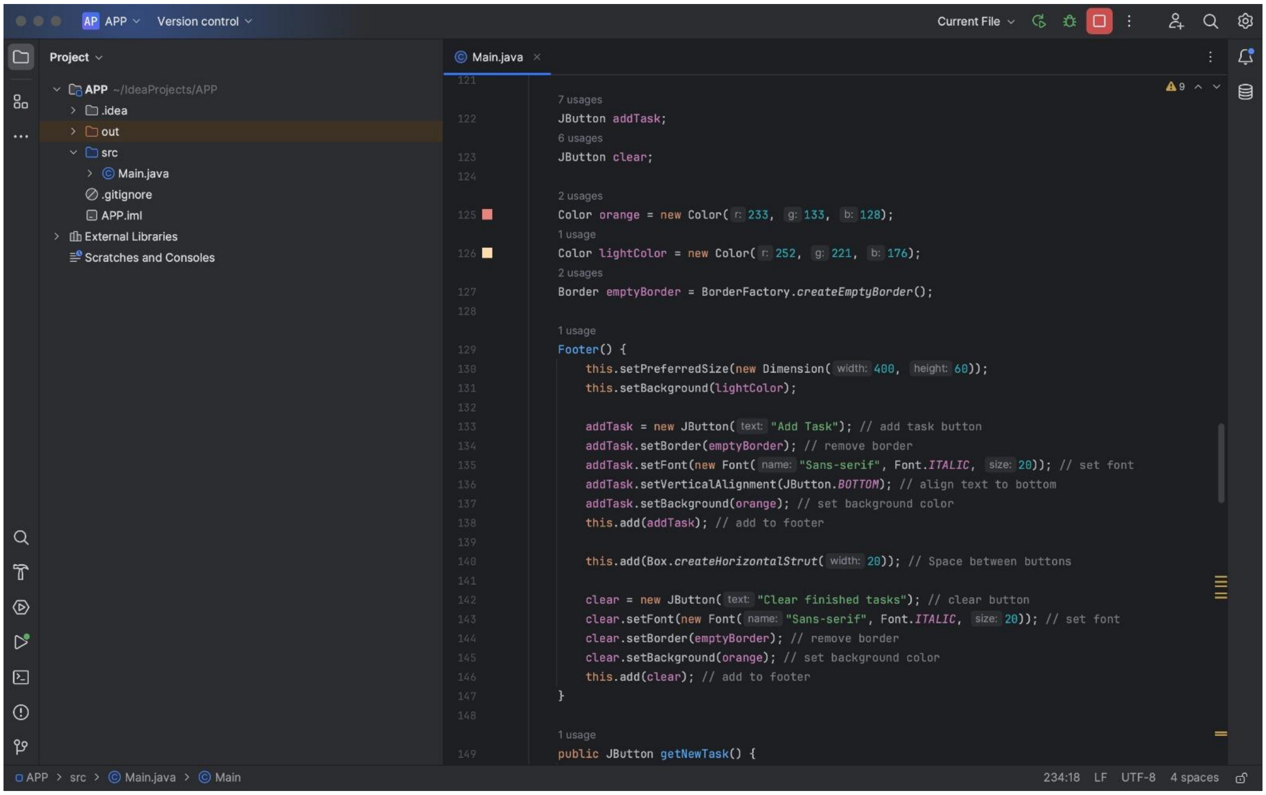
```java
    3 usages
    private boolean checked;

    1 usage
    Task() {
        this.setPreferredSize(new Dimension( width: 400,  height: 20)); // set size of task
        this.setBackground(pink); // set background color of task

        this.setLayout(new BorderLayout()); // set layout of task

        checked = false;

        index = new JLabel( text: ""); // create index label
        index.setPreferredSize(new Dimension( width: 20,  height: 20)); // set size of index label
        index.setHorizontalAlignment(JLabel.CENTER); // set alignment of index label
        this.add(index, BorderLayout.WEST); // add index label to task

        taskName = new JTextField("Write something.."); // create task name text field
        taskName.setBorder(BorderFactory.createEmptyBorder()); // remove border of text field
        taskName.setBackground(pink); // set background color of text field

        this.add(taskName, BorderLayout.CENTER);

        done = new JButton( text: "Done");
        done.setPreferredSize(new Dimension( width: 80,  height: 20));
        done.setBorder(BorderFactory.createEmptyBorder());
        done.setBackground(doneColor);
        done.setFocusPainted(false);

        this.add(done, BorderLayout.EAST);

    }

    1 usage
    public void changeIndex(int num) {
        this.index.setText(num + ""); // num to String
```



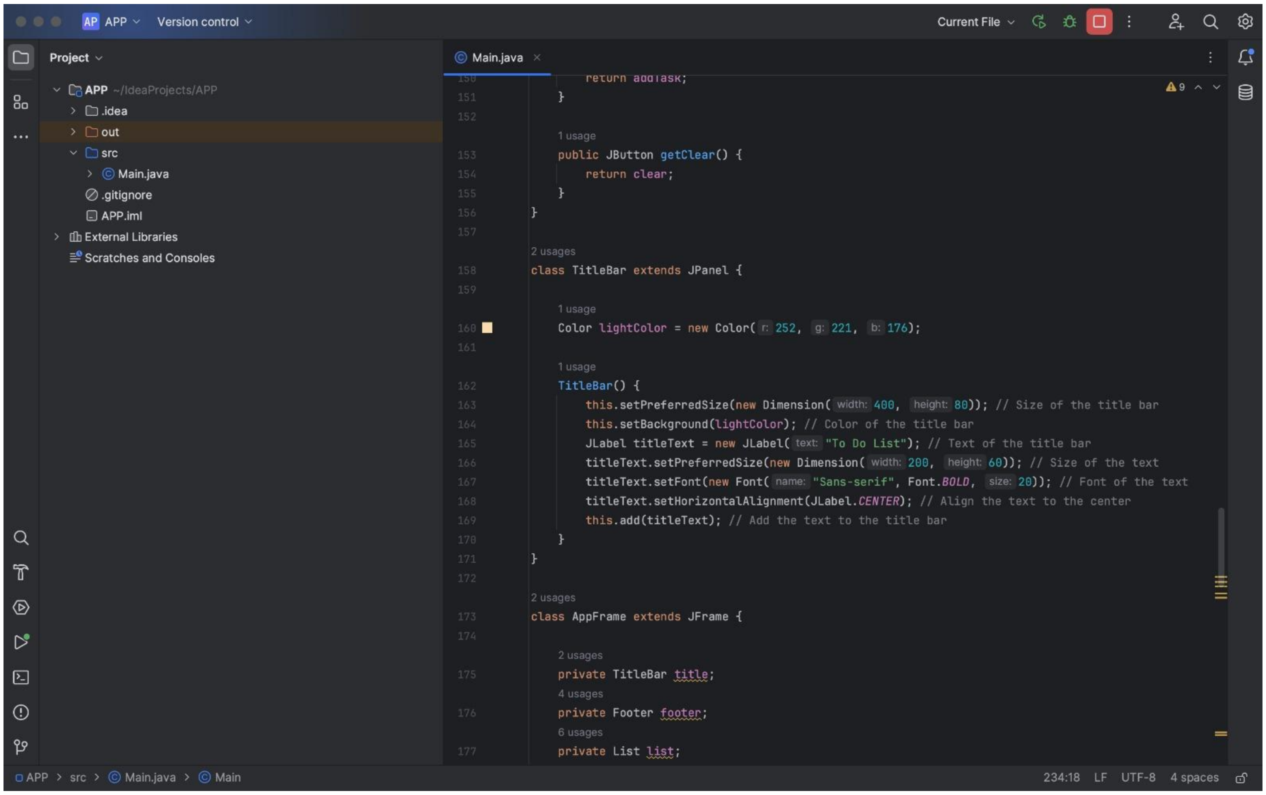
```java
    1 usage
    public void changeIndex(int num) {
        this.index.setText(num + ""); // num to String
        this.revalidate(); // refresh
    }

    1 usage
    public JButton getDone() {
        return done;
    }

    1 usage
    public boolean getState() {
        return checked;
    }

    1 usage
    public void changeState() {
        this.setBackground(green);
        taskName.setBackground(green);
        checked = true;
        revalidate();
    }
}

2 usages
class List extends JPanel {

    1 usage
    Color lightColor = new Color( r: 252,  g: 221,  b: 176);

    1 usage
    List() {

        GridLayout layout = new GridLayout( rows: 10,  cols: 1);
        layout.setVgap(5); // Vertical gap
```



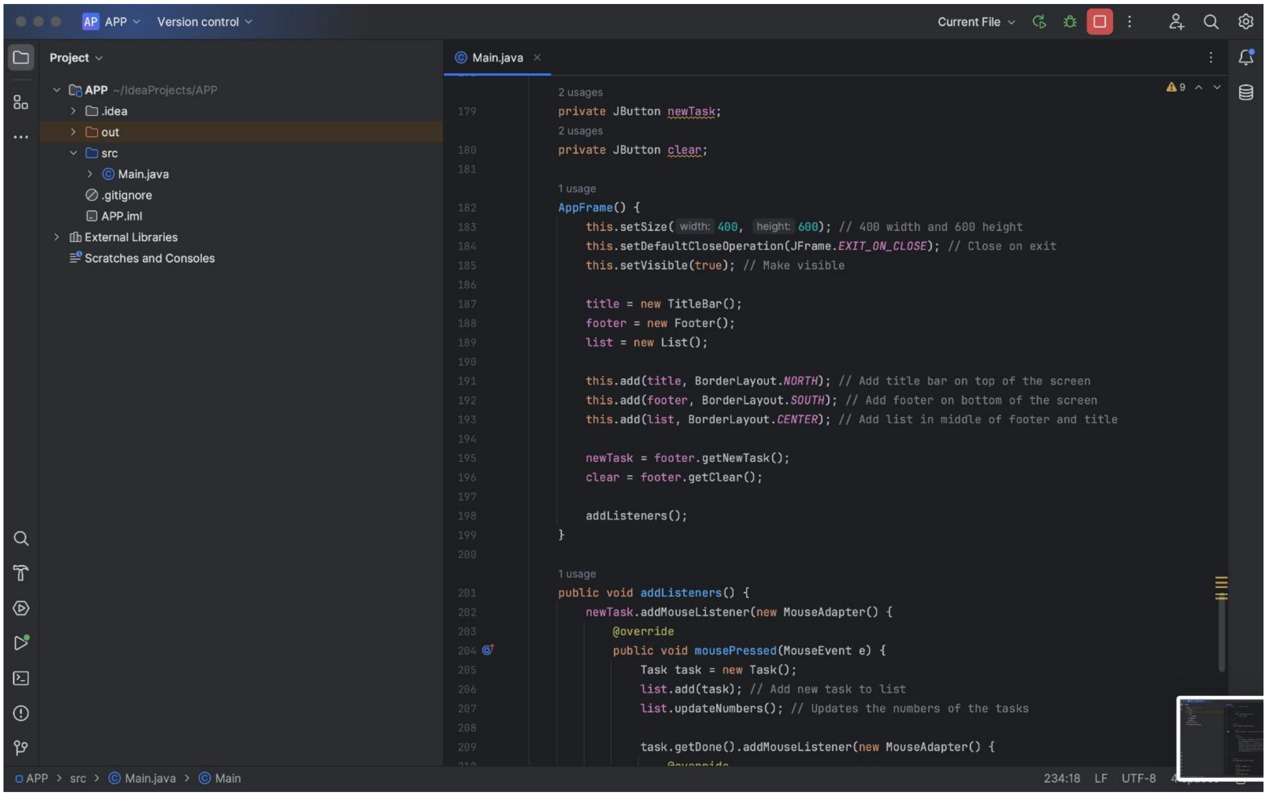
```java

        this.setLayout(layout); // 10 tasks
        this.setPreferredSize(new Dimension( width: 400,  height: 560));
        this.setBackground(lightColor);
    }

    3 usages
    public void updateNumbers() {
        Component[] listItems = this.getComponents();

        for (int i = 0; i < listItems.length; i++) {
            if (listItems[i] instanceof Task) {
                ((Task) listItems[i]).changeIndex( num: i + 1);
            }
        }

    }

    1 usage
    public void removeCompletedTasks() {

        for (Component c : getComponents()) {
            if (c instanceof Task) {
                if (((Task) c).getState()) {
                    remove(c); // remove the component
                    updateNumbers(); // update the indexing of all items
                }
            }
        }

    }
}

2 usages
class Footer extends JPanel {

```



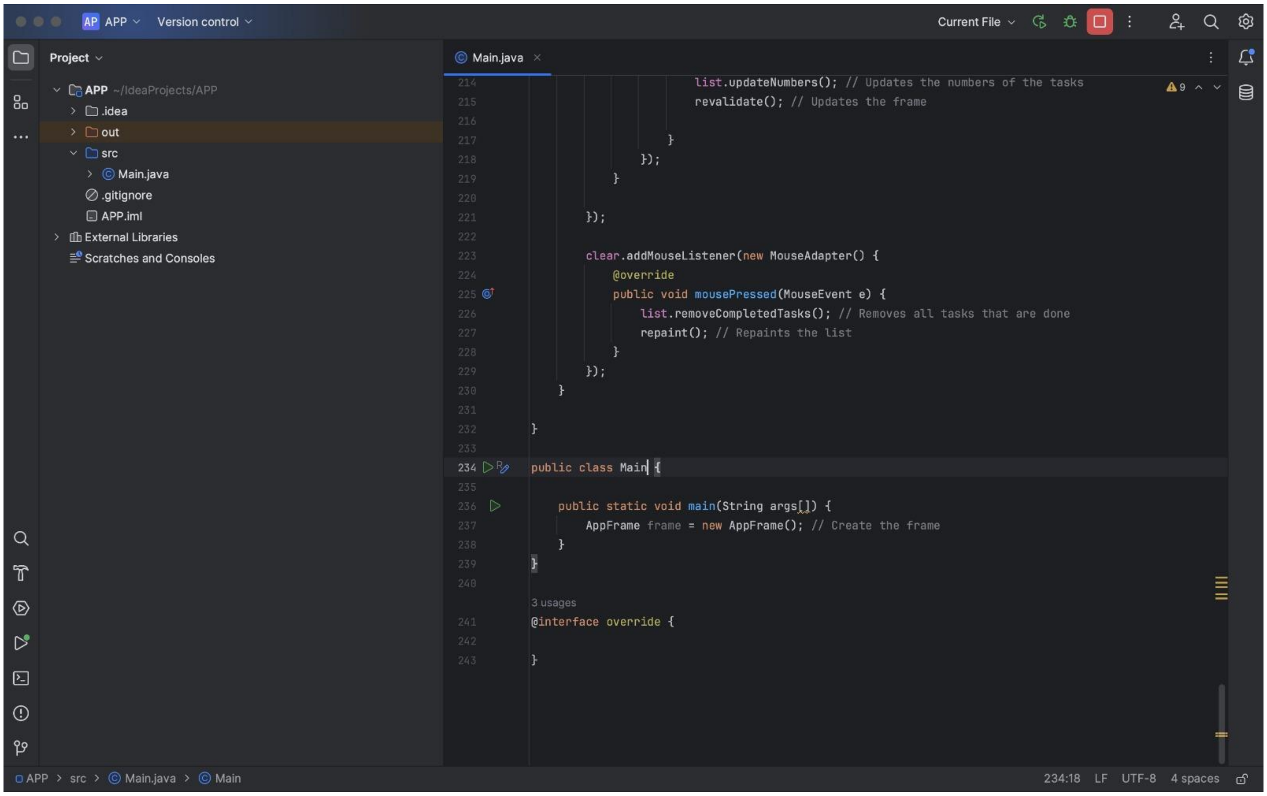
```
    7 usages
    JButton addTask;
    6 usages
    JButton clear;

    2 usages
    Color orange = new Color( r: 233,  g: 133,  b: 128);
    1 usage
    Color lightColor = new Color( r: 252,  g: 221,  b: 176);
    2 usages
    Border emptyBorder = BorderFactory.createEmptyBorder();

    1 usage
    Footer() {
        this.setPreferredSize(new Dimension( width: 400,  height: 60));
        this.setBackground(lightColor);

        addTask = new JButton( text: "Add Task"); // add task button
        addTask.setBorder(emptyBorder); // remove border
        addTask.setFont(new Font( name: "Sans-serif", Font.ITALIC,  size: 20)); // set font
        addTask.setVerticalAlignment(JButton.BOTTOM); // align text to bottom
        addTask.setBackground(orange); // set background color
        this.add(addTask); // add to footer

        this.add(Box.createHorizontalStrut( width: 20)); // Space between buttons

        clear = new JButton( text: "Clear finished tasks"); // clear button
        clear.setFont(new Font( name: "Sans-serif", Font.ITALIC,  size: 20)); // set font
        clear.setBorder(emptyBorder); // remove border
        clear.setBackground(orange); // set background color
        this.add(clear); // add to footer
    }

    1 usage
    public JButton getNewTask() {
```



```
        return addTask;
    }

    1 usage
    public JButton getClear() {
        return clear;
    }
}

2 usages
class TitleBar extends JPanel {

    1 usage
    Color lightColor = new Color( r: 252,  g: 221,  b: 176);

    1 usage
    TitleBar() {
        this.setPreferredSize(new Dimension( width: 400,  height: 80)); // Size of the title bar
        this.setBackground(lightColor); // Color of the title bar
        JLabel titleText = new JLabel( text: "To Do List"); // Text of the title bar
        titleText.setPreferredSize(new Dimension( width: 200,  height: 60)); // Size of the text
        titleText.setFont(new Font( name: "Sans-serif", Font.BOLD,  size: 20)); // Font of the text
        titleText.setHorizontalAlignment(JLabel.CENTER); // Align the text to the center
        this.add(titleText); // Add the text to the title bar
    }
}

2 usages
class AppFrame extends JFrame {

    2 usages
    private TitleBar title;
    4 usages
    private Footer footer;
    6 usages
    private List list;
```

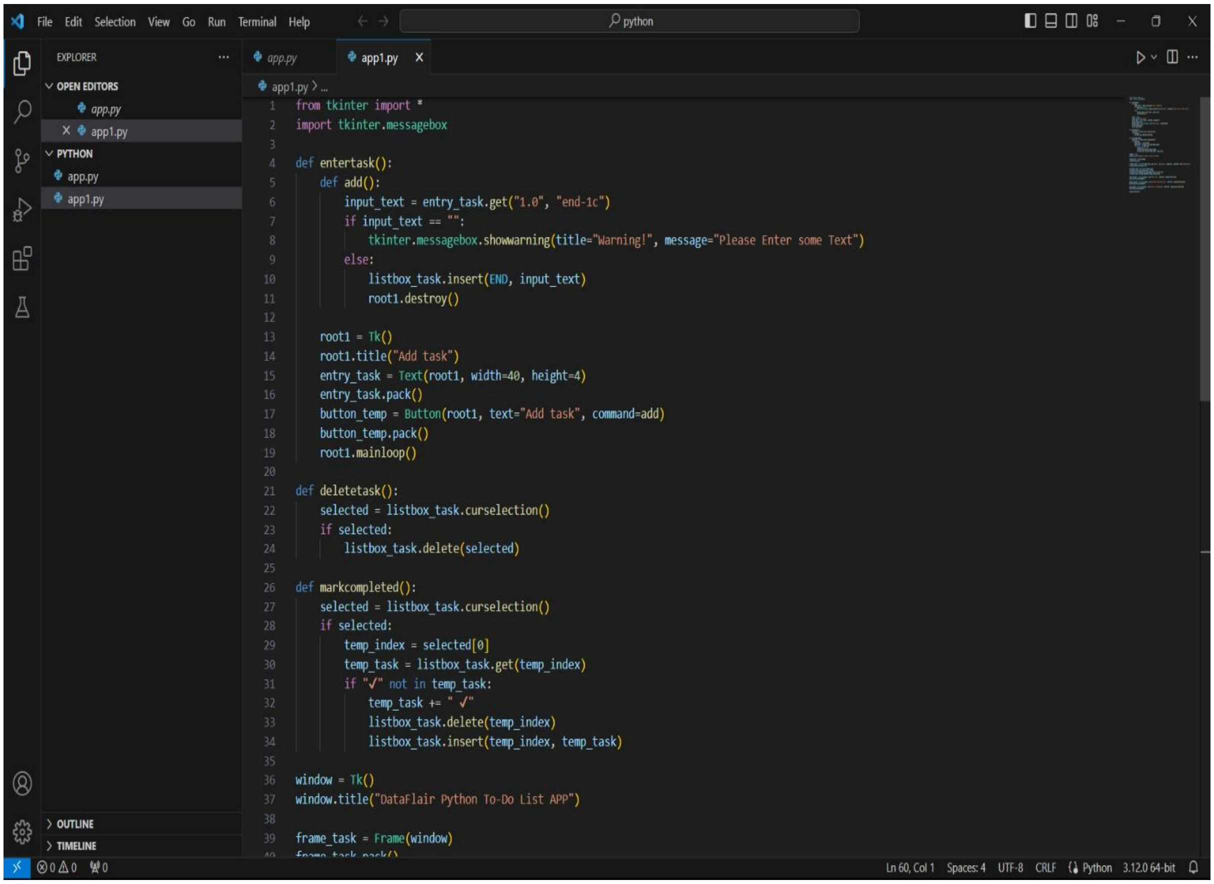


```
    2 usages
    private JButton newTask;
    2 usages
    private JButton clear;

    1 usage
    AppFrame() {
        this.setSize( width: 400,  height: 600); // 400 width and 600 height
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Close on exit
        this.setVisible(true); // Make visible

        title = new TitleBar();
        footer = new Footer();
        list = new List();

        this.add(title, BorderLayout.NORTH); // Add title bar on top of the screen
        this.add(footer, BorderLayout.SOUTH); // Add footer on bottom of the screen
        this.add(list, BorderLayout.CENTER); // Add list in middle of footer and title

        newTask = footer.getNewTask();
        clear = footer.getClear();

        addListeners();
    }

    1 usage
    public void addListeners() {
        newTask.addMouseListener(new MouseAdapter() {
            @override
            public void mousePressed(MouseEvent e) {
                Task task = new Task();
                list.add(task); // Add new task to list
                list.updateNumbers(); // Updates the numbers of the tasks

                task.getDone().addMouseListener(new MouseAdapter() {
                    @override
```

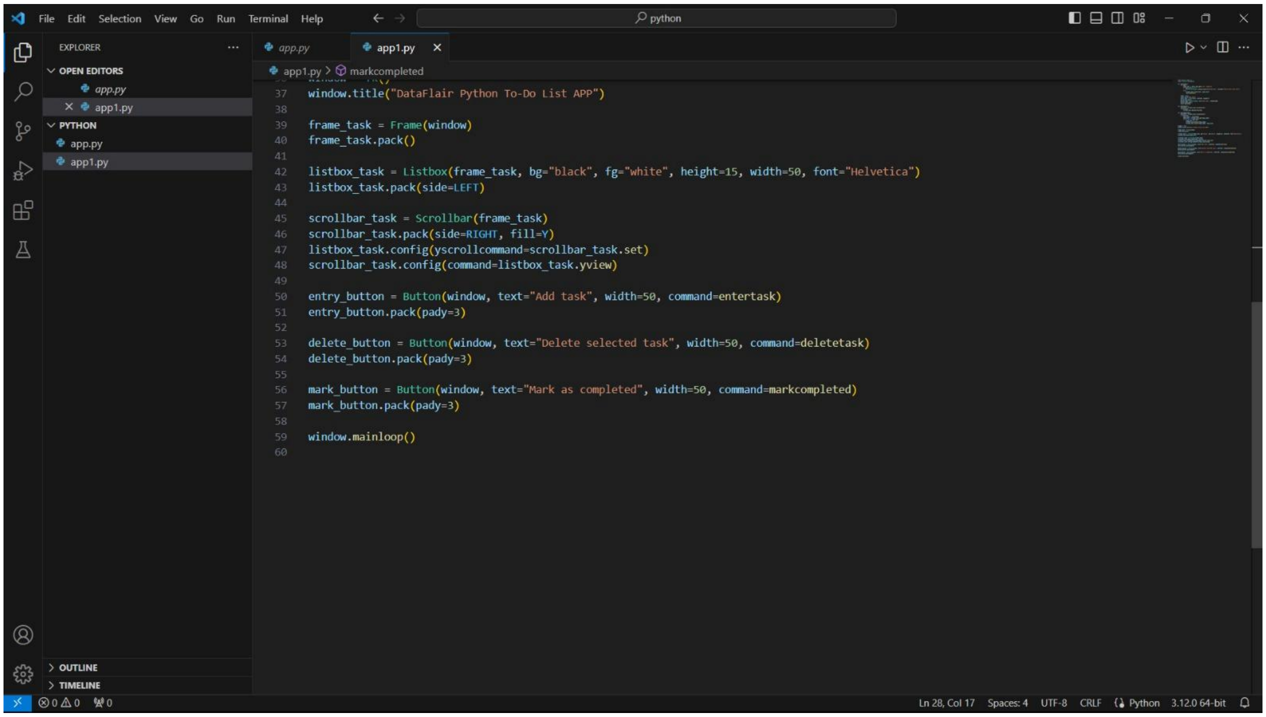


```
                        list.updateNumbers(); // Updates the numbers of the tasks
                        revalidate(); // Updates the frame

                    }
                });
            }

        });

        clear.addMouseListener(new MouseAdapter() {
            @override
            public void mousePressed(MouseEvent e) {
                list.removeCompletedTasks(); // Removes all tasks that are done
                repaint(); // Repaints the list
            }
        });
    }

}

public class Main {

    public static void main(String args[]) {
        AppFrame frame = new AppFrame(); // Create the frame
    }
}

3 usages
@interface override {

}
```

5.2 PYTHON CODE:



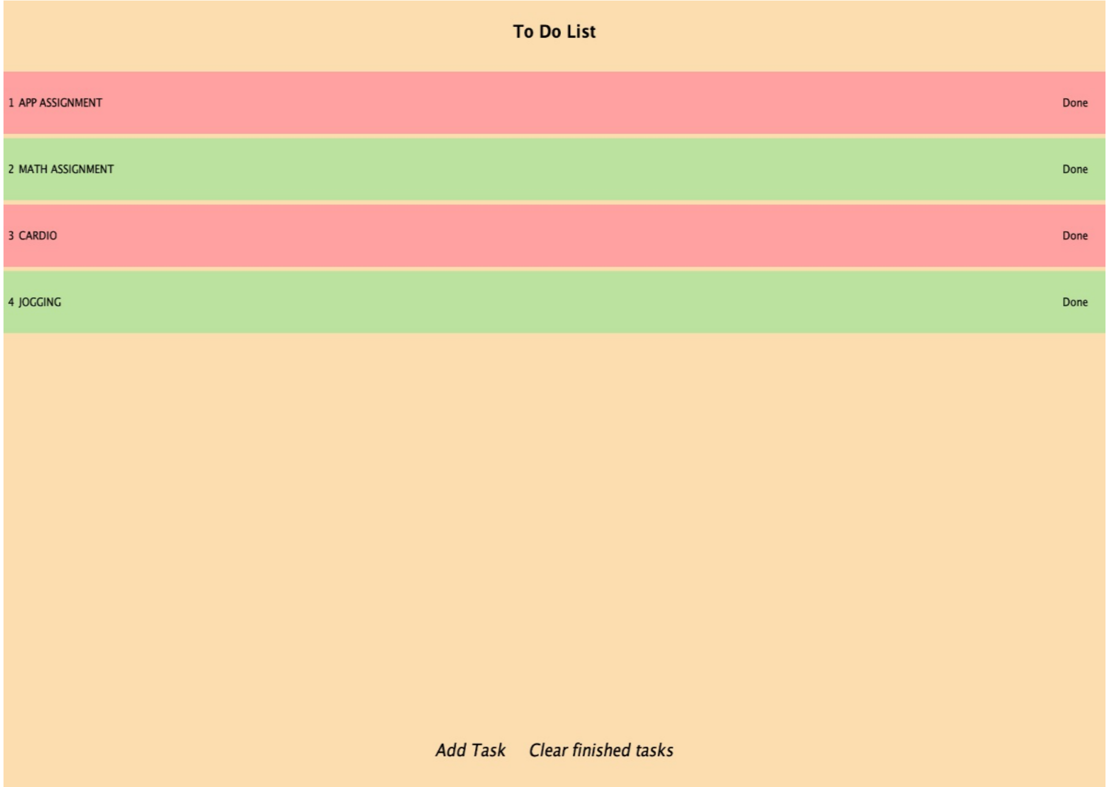
```python
from tkinter import *
import tkinter.messagebox

def entertask():
    def add():
        input_text = entry_task.get("1.0", "end-1c")
        if input_text == "":
            tkinter.messagebox.showwarning(title="Warning!", message="Please Enter some Text")
        else:
            listbox_task.insert(END, input_text)
            root1.destroy()

    root1 = Tk()
    root1.title("Add task")
    entry_task = Text(root1, width=40, height=4)
    entry_task.pack()
    button_temp = Button(root1, text="Add task", command=add)
    button_temp.pack()
    root1.mainloop()

def deletetask():
    selected = listbox_task.curselection()
    if selected:
        listbox_task.delete(selected)

def markcompleted():
    selected = listbox_task.curselection()
    if selected:
        temp_index = selected[0]
        temp_task = listbox_task.get(temp_index)
        if "✓" not in temp_task:
            temp_task += " ✓"
            listbox_task.delete(temp_index)
            listbox_task.insert(temp_index, temp_task)

window = Tk()
window.title("DataFlair Python To-Do List APP")

frame_task = Frame(window)
frame_task.pack()
```



```python
window.title("DataFlair Python To-Do List APP")

frame_task = Frame(window)
frame_task.pack()

listbox_task = Listbox(frame_task, bg="black", fg="white", height=15, width=50, font="Helvetica")
listbox_task.pack(side=LEFT)

scrollbar_task = Scrollbar(frame_task)
scrollbar_task.pack(side=RIGHT, fill=Y)
listbox_task.config(yscrollcommand=scrollbar_task.set)
scrollbar_task.config(command=listbox_task.yview)

entry_button = Button(window, text="Add task", width=50, command=entertask)
entry_button.pack(pady=3)

delete_button = Button(window, text="Delete selected task", width=50, command=deletetask)
delete_button.pack(pady=3)

mark_button = Button(window, text="Mark as completed", width=50, command=markcompleted)
mark_button.pack(pady=3)

window.mainloop()
```

6. EXPERIMENTAL RESULTS AND ANALYSIS

6.1 JAVA OUTPUT:



To Do List

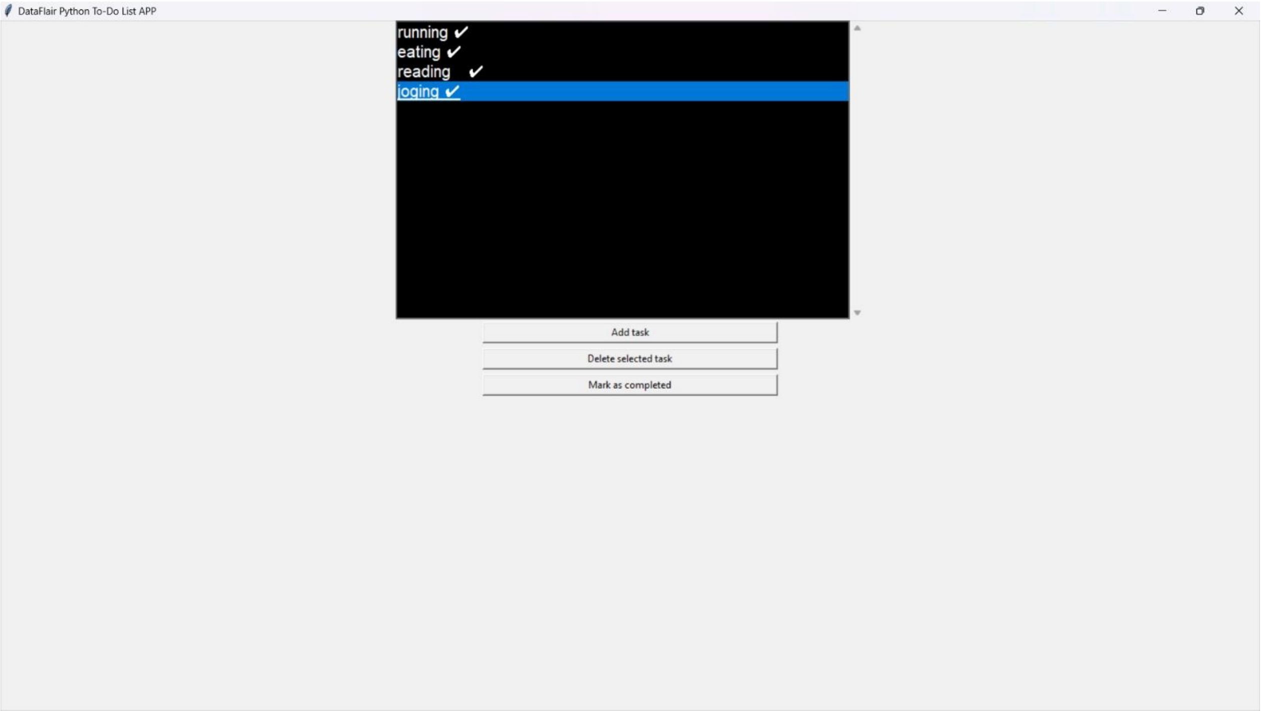
1 APP ASSIGNMENT Done

2 MATH ASSIGNMENT Done

3 CARDIO Done

4 JOGGING Done

6.2 PYTHON OUTPUT:



DataFlair Python To-Do List APP

running ✔
eating ✔
reading ✔
joging ✔

7. CONCLUSION

- The initial plan was to use an SQL database instead of Mongo DB but the Object Relational Mapping(ORM) library we planned to use (Sequelize) with Node.js had issues performing queries on local Postgre SQL database so we tried implementing the project with Mongo DB instead and decided to go on with it.

- The feature to rank and list tasks by their priority was implemented towards the end of the project as we had initially planned to classify tasks into two lists only based on whether the tasks were completed or pending.

- . Several implementation problems did not have any solutions that could be solved by directly using third-party libraries so they had to be solved by manually implementing the functionalities. Improvement in security and integrity .

- Integration with other applications such as calendar, mail, etc.

- Implementation of authentication

- Hosting the web application online to make it accessible to more users

8. REFERENCES

- Staff, C. A. C. M. (2016). React: Facebook's functional turn on writing Javascript. Communications of the ACM, 59(12), 56-62.
- Kvalheim, C. (2015). The Little Mongo DB Schema Design Book. The Blue Print Series.
- Khedkar, S., Thube, S., Estate, W. I., & Naka, C. (2017). Real time databases for applications. International Research Journal of Engineering and Technology (IRJET), 4(06), 2078- 2082.
- Tilkov, S., & Vinoski, S. (2010). Node. js: Using JavaScript to build high-performance network programs. IEEE Internet Computing, 14(6), 80-83.
- Syed, B. A., & Bean, M. (2014). Beginning Node. js (pp. 181-182). New York City: Apress.
- Morales-Morell, A. (2001). Usability aspects of a location-aware TODO list application. University of Puerto Rico, Mayaguez (Puerto Rico).
- Gaspar, F., Taniça, L., Tomás, P., Ilic, A., & Sousa, L. (2015). A framework for application-guided task management on heterogeneous embedded systems. ACM Transactions on Architecture and Code Optimization (TACO), 12(4), 1-25
- . 8. Lin, C. Y., Hung, M. T., & Huang, W. H. (2012, September). A location-based personal task management application for indoor and outdoor environments. In 2012 15th International Conference on Network-Based Information Systems (pp. 582-587). IEEE.
- Bellotti, V., Ducheneaut, N., Howard, M., & Smith, I. (2002). Taskmaster: recasting email as task management. PARC, CSCW, 2.
- Unger, T., & Bauer, T. (2008). Towards a Standardized Task Management. In Multikonferenz Wirtschaftsinformatik (pp. 443-444)