VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Machhe, Belagavi, Karnataka 590018

18CSMP68 – MOBILE APPLICATION DEVELOPMENT LABORATORY WITH MINI-PROJECT REPORT

On

DAILY PLANNER APPLICATION

Submitted in partial fulfillment of the requirement
for the award of the degree of

Bachelor of Engineering
in
Information Science & Engineering
by

Abhay M Kamat (1BG20IS001)
Ranjana BK (1BG20IS039)
Shweta K (1BG20IS054)

Vidyayāmruthamashnuthe

B.N.M. Institute of Technology

An Autonomous Institution under VTU, Approved by AICTE

Department of Information Science and Engineering

2022 – 2023

B.N.M. Institute of Technology

An Autonomous Institution under VTU

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Vidyayāmruthamashnuthe

CERTIFICATE

Certified that the Mini-project entitled **Daily Planner Application** is carried out by **Abhay M Kamat** USN **1BG20IS001, Ranjana BK** USN **1BG20IS039, Shweta K** USN **1BG20IS054,** the bonafide student of **B.N.M Institute of Technology** in partial fulfillment for the award of **Bachelor of Engineering** in **Information Science & Engineering** of the **Visvesvaraya Technological University**, Belagavi during the year 2022-2023. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The mini-project report has been approved as it satisfies the academic requirements in respect of mini-project prescribed for the said Degree.

Mrs. Yashaswini BV
Asst. Professor, Dept. of ISE
BNMIT

Dr. S Srividhya
Prof & Head, Dept. of ISE
BNMIT

Name & Signature of the Examiners with date:

1.

2.

Table of Contents

List of Figures

CHAPTER 1
INTRODUCTION

Android Studio is an integrated development environment (DE) for Google Android Operating System. It is built based on JetBrains* IntelliJ IDEA Community Edition, and it specifically designed for creating applications on Android devices. Some of the key features of Android Studio are as follows:

1. Instant Run - a feature that pushes code and resource changos to the running app.
It allows changes to be made to the app without the need to restart the app, or rebuilding the APK, so that the effects can be seen instantly
2. An Emulator - a virtual android device that can simulate variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input that can be used to run and install the app. It can then be used for testing purposes.
3. Testing Tools and Frameworks - extensive testing tools such as, JUnit 4 and functional UT test frameworks are included with Android Studio. Espresso Test Recorder can generate UI test code by recording the developer's interactions with the app on a device or emulator. The tests can be run on a device, an emulator, in Firebase Test Lab, or on a continuous integration environment.

1.1OBJECTIVE

The objective is to develop a task management application with an additional feature for tracking water consumption. The primary goal is to allow users to create and manage a list of tasks by entering task descriptions, checking/unchecking task completion, and displaying task details in a List View. The code also enables users to track their daily water intake by selecting a water quantity from a dropdown menu. The total water consumption is calculated and displayed when requested. The secondary objective is to provide a user-friendly interface with intuitive controls, such as buttons and checkboxes, to enhance the user experience. The code aims to promote productivity and wellness by assisting users in organizing their tasks and monitoring their hydration levels.

1.2 SCOPE

1. Task Management:

The code allows users to add tasks to a list using an EditText field and an "Add" button.
Tasks are displayed in a ListView using a custom adapter.
Each task item in the list has a checkbox that users can check to mark a task as completed.
When a task is checked as completed, a toast message is displayed.

2. Water Consumption Tracking:

The code includes a water consumption feature to track daily water intake.
A spinner (dropdown) allows users to select the amount of water consumed (e.g., 100 ml, 200 ml).
An "Add Water" button adds the selected amount to the total water consumption.
The current water consumption is displayed in a TextView.
Clicking the "Water Consumption" button displays a toast message with the current water consumption value.

- Potential Scope Expansion:
- Due Date/Reminder: Add functionality to set due dates or reminders for tasks.
- Task Sorting: Implement options to sort tasks by completion status, due date, or priority.
- Task Deletion: Allow users to delete tasks from the list.
- Task Editing: Enable users to edit task names or modify task details.
- Task Categories: Introduce categories or labels to organize tasks (e.g., work, personal, shopping).
- Data Persistence: Implement a database or file storage to save tasks and water consumption data between app sessions.
- User Accounts: Add user authentication and the ability to manage separate task lists for different users.
- Notifications: Send notifications or reminders for pending tasks or water consumption goals.
- Statistics: Provide insights or analytics on task completion rates or water intake over time.

1.3 PROBLEM STATEMENT

The aim of this project Media Player Application is to provide an easy access to for the users accessing the app. Due to the fierce competition between music player applications, many developers tried to add many features, advertise and content to their respective music player in order to retain their users and attract new users. This trend has made it harder for users to get content from their music player, which also means it's harder to filter the content that they want. So this application aims to give an easy access to the user helping to demonstrate a basic media player that allows the user to Forward, Backward, Play, and Pause an audio.

1.4 MOTIVATION

The motivation behind the project is to develop a task management application with an additional feature of tracking water consumption. The project aims to provide a user-friendly interface where users can create and manage their tasks while also keeping track of their daily water intake.

Here are some potential motivations for this project that could be included in a report:

1. Improving Productivity: Task management applications help individuals stay organized and improve their productivity by providing a centralized platform to track and manage their tasks effectively. By developing this application, users can easily create tasks, mark them as completed, and prioritize their work, thereby enhancing their productivity.
2. Promoting Health and Wellness: Water consumption is an essential aspect of maintaining good health. By integrating a water tracking feature into the application, users are reminded to stay hydrated throughout the day. This promotes a healthy lifestyle and encourages users to take care of their well-being while managing their tasks.
3. Simplifying Task Management: The project aims to provide a simple and intuitive user interface for managing tasks. Users can add tasks, view their task list, and mark tasks as completed with the help of checkboxes. This simplicity in task management can reduce cognitive load and make it easier for users to stay organized.
4. Enhancing User Experience: The project focuses on providing a positive user experience by incorporating features such as real-time task updates, intuitive task organization, and a visually appealing interface. By prioritizing user experience, the application aims to create a seamless and enjoyable task management process for its

users.

5. Learning Android Development: The project serves as a learning opportunity for developers interested in Android application development. By implementing various components such as ListView, EditText, Spinner, and CheckBox, developers can gain hands-on experience in working with different UI elements and understanding their functionalities within an Android application. Overall, the motivation of this project revolves around improving productivity, promoting health and wellness, simplifying task management, enhancing user experience, and providing a learning experience in Android development.

CHAPTER 2

METHODOLOGY

Methodology is the philosophical framework within which the research is conducted or the foundation upon which the research is based. Firstly, the methodology should be the most appropriate to achieve objectives of the research. Secondly, it should be made possible to replicate the methodology used in other researches of the same nature.

2.1 TOOLS

Platform: Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android OS, built on JetBrains IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is CH.

Features of Android Studio

A specific feature of the Android Studio is an absence of the possibility to switch autosave feature off.

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Template-based wizards to create common Android designs and components.
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations

Android Virtual Device (Emulator) to run and debug apps in the Android studio
Android Studio supports all the same programming languages of Intelli (and CLion) eg. Java, Cr, and more with extensions, such as Go and Android Studio 3.0 or later supports Kotlin and "all Java 7 languages features and a subset of Java 8 languages features that vary by platform version." External projects backport some Java 9 features. While Intelli states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.
Once an app has been compiled with Android Studio, it can he published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy. The Android Emulator has additional requirements beyond the basic system requirements for Android Studio, which are described below.

- SDK Tools 26.1.1 or higher.
- 64-bit processor
- Windows: CPU with UG (unrestricted guest) support,
- Intel Hardware Accelerated Execution Manager (HAXM) 62.1.

Programming Language: Java

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let application developers write once, run anwhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM regardless of the underlying computer architecture. The syntax of Java is similar to C and C+4, The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages.

Diagrams.net

Diagrams.net/draw.io is an open-source technology stack for building diagramming applications, and the world's most widely used browser-based end-user diagramming software.

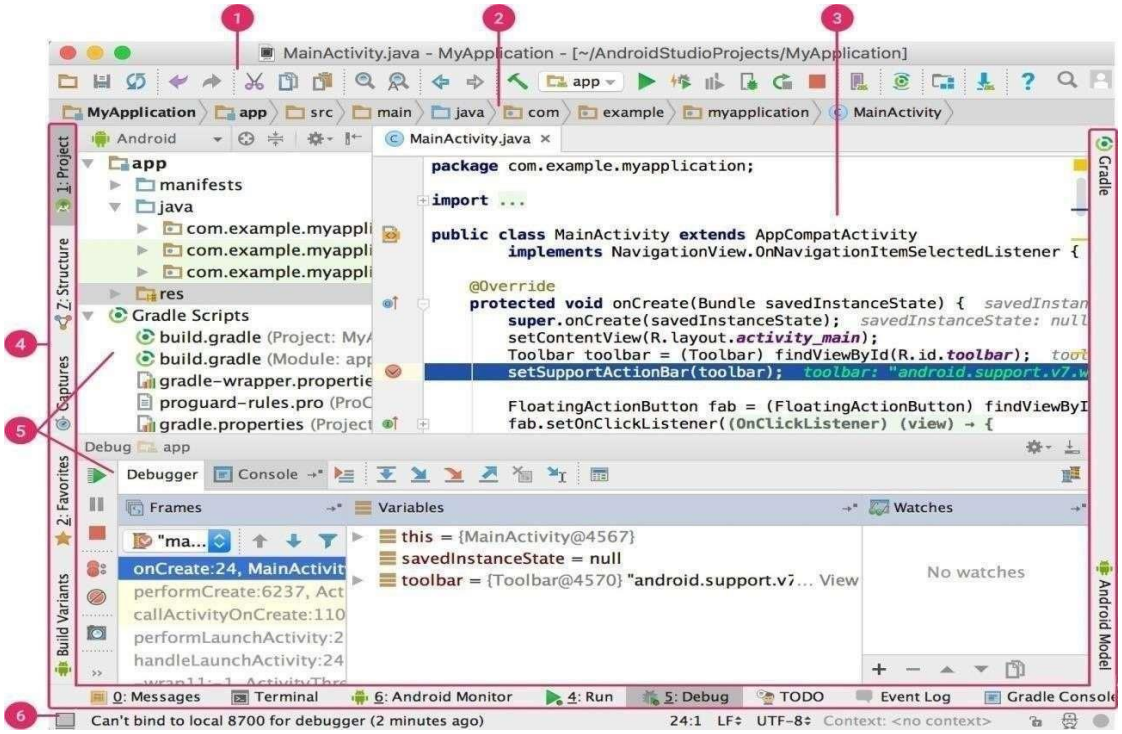

Fig 1.4.1 The Android Studio Main Window

- The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools.
- The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.
- The editor window is where you create and modify code. Depending on the current file type, the editor can change.
- The tool window bar runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
- The tool windows give you access to specific tasks like project management, search, version control, and more.
- The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

A System Requirements Specification is a structured collection of information that embodies the requirements of a system.

3.1 Software Requirements

	Microsoft Windows	**Mac**	**Linux**
Operating System Version	Microsoft® Windows® 7/8/10 (32- or 64-bit) The Android Emulator only supports 64-bit Windows.	Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)	GNOME or KDE desktop Tested on gLinur based on Debian (4.19.67-2rodete2).
Minimum required JDK version	Java Development kit 8		
Android Studio	Latest Version Available		

3.2 Hardware Requirements

Random Access Memory (RAM)	4 GB RAM minimum; 8 GB RAM recommended.
Processor	Intel@ Core™M iS-8265U CPU @ 1.60GHz 1.80 GHz
Free digital storage	2 GB of available digital storage minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image).
Minimum screen resolution	1280 × 800
System Type	64-bit Operating System, ×64-bases processor

3.3 Functional Requirements

Functional requirements related to the daily planner can include:

- Task Management: The application should allow users to add, edit, and delete tasks. Users should be able to enter task descriptions, mark tasks as completed, and remove tasks from the list.
- User Interface: The user interface should be intuitive and user-friendly, providing an easy way for users to interact with the application. It should display tasks in a clear and organized manner, allowing users to view and manage their tasks efficiently.
- Water Consumption Tracking: The application includes a feature to track water consumption. Users should be able to select the amount of water consumed from a dropdown menu and add it to their daily water intake. The application should display the current water consumption and provide a button to view the total water consumed.
- Toast Notifications: When a task is marked as completed, the application should display a toast notification to provide feedback to the user. This notification can serve as a confirmation and encourage users when they complete tasks.
- Data Persistence: The application should store task data persistently, ensuring that tasks are retained even when the application is closed or restarted. This ensures that users can access their task list reliably and continue their planning seamlessly.
- Error Handling: The application should handle any errors or exceptions that may occur during task management or water consumption tracking. It should provide appropriate error messages to the user, guiding them on how to resolve any issues encountered.
- Task Sorting and Filtering: The application can include options for sorting tasks based on priority, due date, or completion status. It can also provide filtering options to allow users to view specific subsets of tasks, such as completed tasks or tasks due today.
- Performance Optimization: The application should be optimized for performance, ensuring smooth scrolling and responsiveness, even with a large number of tasks. It should efficiently manage memory and resources to provide a seamless user experience.

3.4 Non - Functional Requirements

Non-functional requirements related to the daily planner can include:

- Usability: The application should have a user-friendly and intuitive interface that is easy to navigate and understand. It should provide clear instructions and visual cues to guide users in managing their tasks and tracking water consumption effectively.
- Performance: The application should be responsive and perform efficiently, even when dealing with a large number of tasks or high water consumption data. It should have quick load times, smooth scrolling, and minimal latency to ensure a seamless user experience.
- Reliability: The application should be reliable and stable, with minimal crashes or unexpected behavior. It should handle errors gracefully and recover from any failures to ensure that users can rely on it for managing their daily tasks consistently.
- Security: The application should prioritize the security and privacy of user data. It should implement appropriate measures to protect sensitive information, such as task details or user preferences. This can include data encryption, secure storage practices, and proper authentication mechanisms.
- Compatibility: The application should be compatible with a range of Android devices and screen sizes, ensuring that it functions correctly across different devices. It should adhere to Android platform guidelines and support various versions of the Android operating system.
- Maintainability: The codebase should be well-structured, modular, and easy to maintain. It should follow coding best practices, use appropriate design patterns, and have clear documentation to facilitate future updates, bug fixes, and enhancements.
- Performance Optimization: The application should be optimized for efficient memory usage, battery consumption, and network usage. It should minimize resource utilization to prolong device battery life and provide a smooth user experience.
- Accessibility: The application should consider accessibility guidelines to ensure that individuals with disabilities can use and interact with the app effectively. This includes providing support for accessibility features such as screen readers, text resizing, and color contrast options.

3.5 XML and Java Code

```
package com.example.myplan;
import android.annotation.SuppressLint;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.Spinner;
import java.util.List;
import android.widget.AdapterView;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    private ListView listView;
    private TaskAdapter adapter;
    private ArrayList<TaskItem> taskList;
    private EditText taskInput;
    private Button addButton;
    private TextView headingText;

    private TextView waterConsumptionButton;
    private Spinner waterDropdown;
    private Button addWaterButton;
```

```
    private ArrayAdapter<String> waterAdapter;
    private List<String> waterOptions;
    private int waterConsumption = 0;

    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        listView = findViewById(R.id.list_view);
        taskInput = findViewById(R.id.task_input);
        addButton = findViewById(R.id.add_button);
        headingText = findViewById(R.id.heading_text);

        taskList = new ArrayList<>();
        adapter = new TaskAdapter(taskList);

        listView.setAdapter(adapter);

        addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String task = taskInput.getText().toString().trim();
                if (!task.isEmpty()) {
                    TaskItem taskItem = new TaskItem(task, false);
                    taskList.add(taskItem);
                    adapter.notifyDataSetChanged();
                    taskInput.getText().clear();
                }
            }
        });

        // Set up water consumption button
waterConsumptionButton = findViewById(R.id.water_consumption_button);
```

```
        waterConsumptionButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Perform action when water consumption button is clicked
                // For example, show a message with the current water consumption
                String message = "Water consumed: " + waterConsumption + " ml";
                Toast.makeText(MainActivity.this, message, Toast.LENGTH_SHORT).show();
            }
        });

        // Set up water dropdown
        waterDropdown = findViewById(R.id.water_dropdown);
        waterOptions = new ArrayList<>();
        waterOptions.add("100 ml");
        waterOptions.add("200 ml");
        waterOptions.add("300 ml");
        waterOptions.add("400 ml");

        waterAdapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_item,
waterOptions);

waterAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item
);
        waterDropdown.setAdapter(waterAdapter);

        waterDropdown.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
{
                // Update the selected water consumption
                String selectedWater = waterOptions.get(position);
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    waterConsumption = Integer.parseInt(selectedWater.split(" ")[0]);
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                // Do nothing
            }
        });

        // Set up add water button
        addWaterButton = findViewById(R.id.add_water_button);
        addWaterButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Perform action when add water button is clicked
                // For example, update the text of the water consumption button
                int selectedWater =
Integer.parseInt(waterOptions.get(waterDropdown.getSelectedItemPosition()).split(" ")[0]);
                waterConsumption += selectedWater;
                waterConsumptionButton.setText(waterConsumption + " ml");
            }
        });
    }

    private class TaskAdapter extends ArrayAdapter<TaskItem> {

        public TaskAdapter(ArrayList<TaskItem> taskList) {
            super(MainActivity.this, 0, taskList);
        }

        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
```

```
        if (convertView == null) {
            convertView = LayoutInflater.from(getContext()).inflate(R.layout.item_task,
parent, false);
        }

        final TaskItem taskItem = getItem(position);
        CheckBox checkBox = convertView.findViewById(R.id.task_checkbox);
        TextView textView = convertView.findViewById(R.id.task_text);

        checkBox.setChecked(taskItem.isCompleted());
        textView.setText(taskItem.getTask());

        checkBox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
                taskItem.setCompleted(isChecked);
                if (isChecked) {
                    showToast("Congratulations on completing the task :-)");
                }
            }
        });

        return convertView;
    }
  }

  private void showToast(String message) {
    Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
  }
}

class TaskItem {
```

```
        private String task;
    private boolean completed;

    public TaskItem(String task, boolean completed) {
        this.task = task;
        this.completed = completed;
    }

    public String getTask() {
        return task;
    }

    public boolean isCompleted() {
        return completed;
    }

    public void setCompleted(boolean completed) {
        this.completed = completed;
    }

    @Override
    public String toString() {
        return task;
    }
}
```

CHAPTER 4

SYSTEM DESIGN AND DEVELOPMENT

A daily planner is a tool that helps individuals organize and manage their daily tasks, events, appointments, and goals. It serves as a personal productivity tool, allowing users to plan and track their activities effectively. Here's some information about daily planners:

Purpose:

1. The main purpose of a daily planner is to provide a structured framework for managing time and tasks efficiently. It helps users prioritize their activities, set goals, and stay organized throughout the day.
2. Layout: Daily planners typically have a two-page spread for each day, with one page dedicated to scheduling and the other for notes, tasks, and other details. The scheduling section may include time slots or open spaces for users to write down their appointments, meetings, and events at specific times. The notes section allows users to jot down additional information, reminders, or thoughts related to their tasks.
3. Key Features:

- Date and day: Each page of the daily planner includes the date and the corresponding day of the week.
- Time slots: Some planners provide pre-printed time slots, usually in hourly or half-hourly increments, to help users plan their day more precisely.
- To-do lists: Planners often include sections for users to list their tasks, assignments, or goals for the day. This allows users to keep track of what needs to be done and ensures that important tasks are not overlooked.
- Notes and additional sections: Daily planners may also include sections for notes, priorities, gratitude, meal planning, water intake tracking, exercise tracking, or any other customizable sections that cater to individual needs.

4. Benefits:

- Time management: Daily planners enable users to allocate time effectively and avoid overcommitting or missing important tasks or appointments.
- Increased productivity: By having a clear overview of tasks and goals for the day, individuals can focus on their priorities and accomplish more.
- Organization: Planners provide a centralized place for recording and referencing information, making it easier to stay organized and reduce mental clutter.

- Accountability: Using a daily planner promotes accountability, as users can track their progress, reflect on achievements, and make adjustments for better time management in the future.

5. Customization: Many daily planners offer flexibility for customization, allowing users to personalize layouts, add specific sections, or include motivational quotes or images.

Remember, different individuals have different preferences and requirements for their daily planners. It's important to find a planner that aligns with your needs and helps you stay organized and productive.

CHAPTER 5

IMPLEMENTATION

Project implementation is the process of putting a project plan into action to produce the deliverables, otherwise known as the products or services, for clients or stakeholders.

5.1Algorithms

1. Create a class MainActivity that extends AppCompatActivity:
   1.1. Declare private variables:
      - listView: ListView
      - adapter: TaskAdapter
      - taskList: ArrayList<TaskItem>
      - taskInput: EditText
      - addButton: Button
      - headingText: TextView
      - waterConsumptionButton: TextView
      - waterDropdown: Spinner
      - addWaterButton: Button
      - waterAdapter: ArrayAdapter<String>
      - waterOptions: List<String>
      - waterConsumption: integer
   1.2. Override the onCreate() method:
      1.2.1. Set the content view to "activity_main" layout.
      1.2.2. Initialize the UI elements by finding their respective views by their IDs.
      1.2.3. Initialize the taskList as an empty ArrayList<TaskItem>.
      1.2.4. Initialize the adapter with the taskList.
      1.2.5. Set the adapter to the listView.
      1.2.6. Set an OnClickListener to the addButton:
         1.2.6.1. Get the task string from the taskInput.
         1.2.6.2. If the task is not empty:
            - Create a new TaskItem with the task and set its completed status false.
            - Add the new TaskItem to the taskList.
- Notify the adapter that the data set has changed.

- Clear the text in the taskInput.

1.2.7. Set an OnClickListener to the waterConsumptionButton:

1.2.7.1. Display a toast message with the current water consumption.

1.2.8. Initialize the waterOptions list with the available options.

1.2.9. Initialize the waterAdapter with the waterOptions and set its layout.

1.2.10. Set the waterAdapter to the waterDropdown spinner.

1.2.11. Set an OnItemSelectedListener to the waterDropdown:

1.2.11.1. Get the selected water option.

1.2.11.2. Update the waterConsumption variable with the selected value.

1.2.12. Set an OnClickListener to the addWaterButton:

1.2.12.1. Get the selected water option from the waterDropdown.

1.2.12.2. Update the waterConsumption by adding the selected value to it.

1.2.12.3. Set the text of the waterConsumptionButton to the updated waterConsumption.

2. Create a class TaskAdapter that extends ArrayAdapter<TaskItem>:

2.1. Override the getView() method:

2.1.1. Check if the convertView is null:

- If yes, inflate the "item_task" layout and assign it to the convertView.

2.1.2. Get the TaskItem at the current position.

2.1.3. Get the checkbox and textView views from the convertView.

2.1.4. Set the checkbox state and text according to the TaskItem.

2.1.5. Set an OnCheckedChangeListener to the checkbox:

2.1.5.1. Update the TaskItem's completed status based on the checkbox state.

2.1.5.2. If the checkbox is checked, display a toast message.

2.1.6. Return the convertView.

3. Create a class TaskItem:

3.1. Declare private variables:

- task: String

- completed: boolean

3.2. Create a constructor that takes task and completed parameters and assigns them to the respective variables.

3.3. Create getter and setter methods for task and completed.

3.4. Override the toString() method to return the task.

XML Layouts

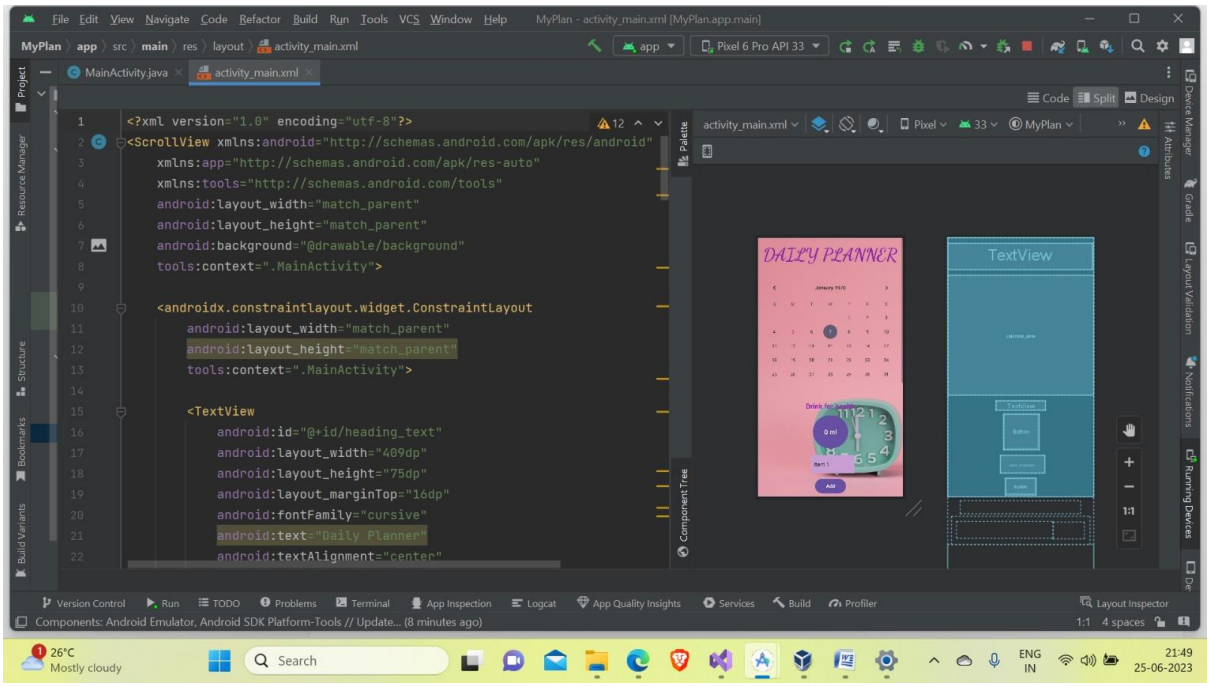

Figure 5.1 Home page XML Layout

CHAPTER 6
RESULTS AND DISCUSSION

Project Results includes data, reports, Deliverables, and any other know-how Developed or produced in the course of development. The project results are as follows**:**

6.1 Snapshots of the project and description

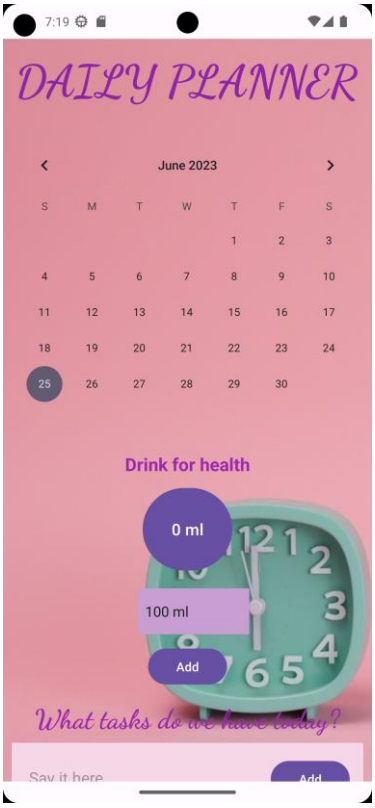

Figure 6.1 Home page

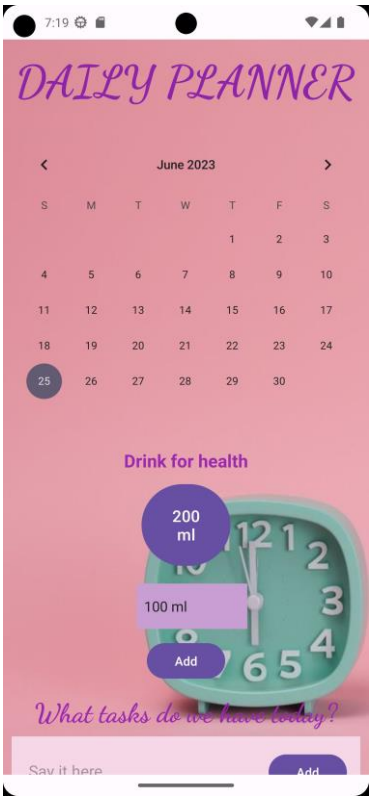

Figure 6.2 adding 100ml

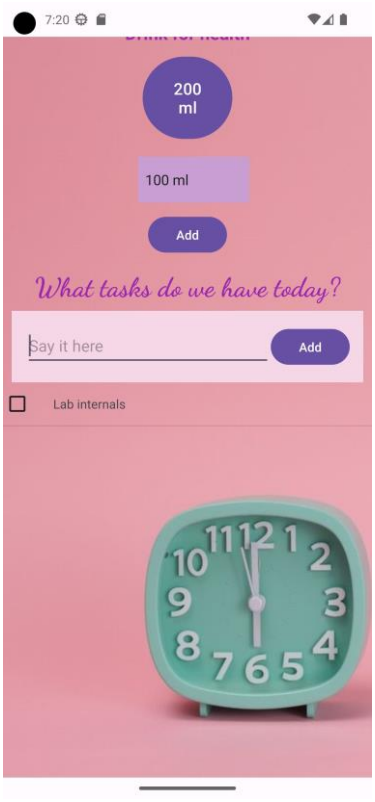

Figure 6.3 displaying task

CHAPTER 7

CONCLUSION

This code demonstrates the development of a comprehensive mobile application that serves as a task manager while promoting healthy hydration habits. By accomplishing the objectives outlined in the code, this application offers users an efficient and organized approach to managing their tasks and staying hydrated.The task management feature allows users to add new tasks, mark them as completed, and view their task list in a visually appealing ListView. This functionality enhances productivity and assists users in prioritizing their activities effectively. The code ensures a seamless user experience by providing an intuitive user interface and interactive checkboxes to track task completion.

Moreover, the incorporation of a water consumption tracking feature within the application fosters healthy hydration habits. Users can select the amount of water consumed from a dropdown menu, and the code calculates and displays the total water consumption. This feature encourages users to maintain proper hydration levels throughout the day, promoting their overall well-being and productivity.The code utilizes various Android components such as TextViews, Spinners, ListViews, and Buttons to create a user-friendly interface. It demonstrates the effective utilization of adapters to populate the ListView with task items and handle user interactions. The implementation of event listeners for button clicks and dropdown selections ensures the proper functioning of the application's features.

Overall, the provided code successfully achieves its objectives of task management and promoting healthy hydration habits. It serves as a valuable tool for users seeking to organize their tasks efficiently while maintaining their well-being through regular hydration. With further enhancements and additional features, this application has the potential to become an essential tool for individuals striving for productivity and a healthy lifestyle.

REFERENCES

[1] Android Programming Tutorials by Mark L. Murphy

[2] "Design and Implementation of Daily planner Code based on Android platform" by Munneb Ahemed Qureshi M.Madan.Gopal & Mohamed Sadiq.

[3] M. Song, J. Sun, X. Fu and W. Xiong, “Design and Implementation of Daily planner Based on Android”, WICOM, (2010), pp. 1 – 4.

[4] Xu, J. The Design and Implementation of Daily planner Based on Android Platform, Beijing Posts and Tele communications University,2011 5:156～178.