SR.NO	NAME	PRN	EMAIL ID
1	AARUSHI DHAWAN	22070122001	aarushi.dhawan.btech2022@sitpune.edu.in
2	GEHNA VITHALANI	22070122070	gehna.vithal.btech2022@sitpune.edu.in
3	ARGYA VIJAY SINGH	22070122027	argya.singh.btech2022@sitpune.edu.in

Programming paradigms report

TASK MANAGER

Problem Statement

In the modern world, effective task management is crucial for productivity and time management. To address this need, we are embarking on a project to develop a Task Manager system that harnesses the principles of inheritance and polymorphism to create a flexible and efficient task scheduling and management tool. The Task Manager will allow users to create, organise, and manage various types of tasks, providing timely reminders and prioritisation options.The primary objective of this project is to design and implement a Task Manager system that leverages the power of object-oriented programming and advanced software engineering concepts such as inheritance and polymorphism. This system will enable users to create, categorise, and prioritise different types of tasks, manage due dates, and provide timely reminders.

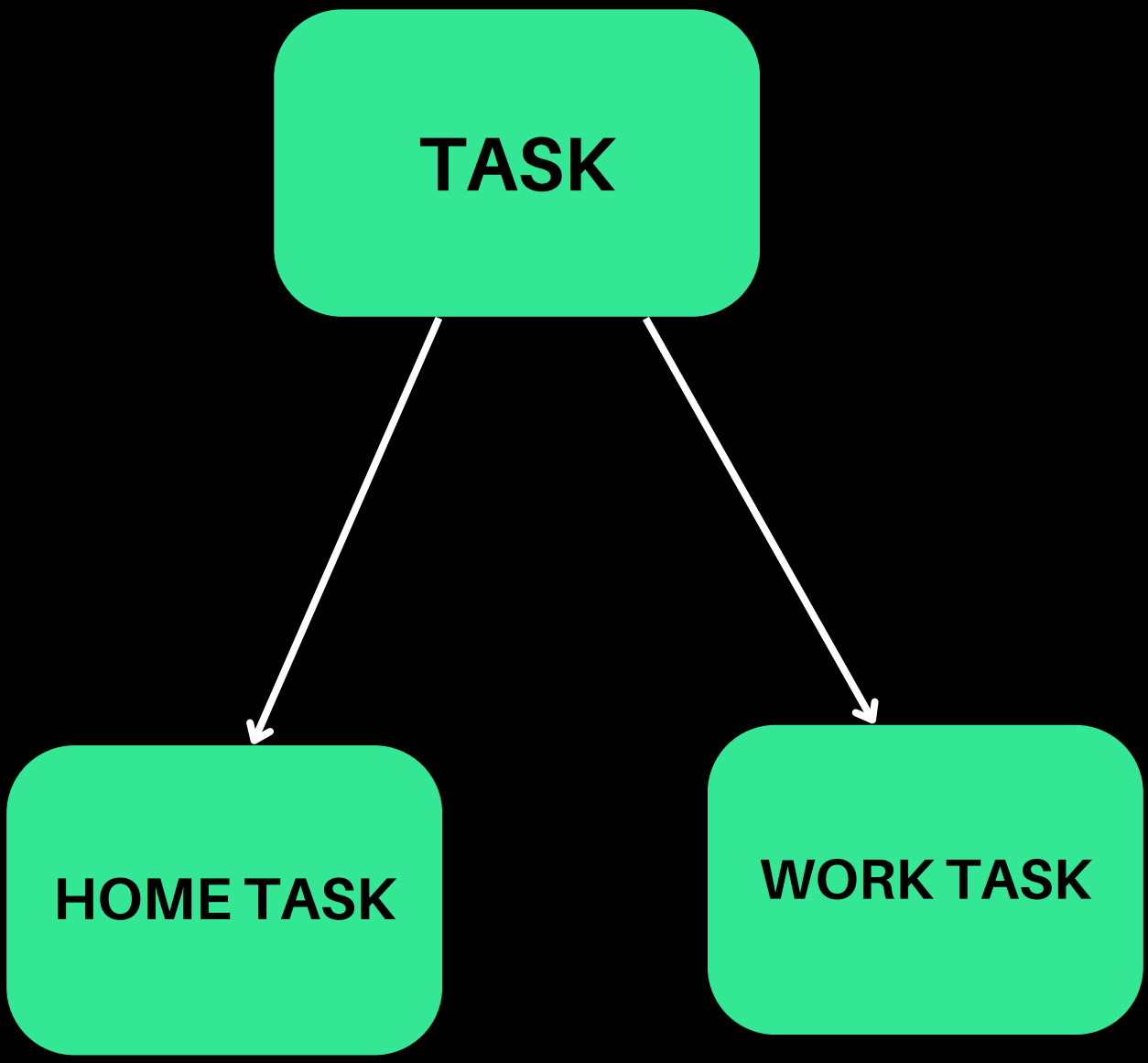
Project objectives and scope :

The primary objectives of this project are as follows:

- Flexibility through Inheritance: Inheritance is a core component of our system. It allows us to create a hierarchy of task classes, providing a structure that encompasses a wide range of task types. Common attributes and behaviours shared among various tasks are encapsulated at the superclass level, while subclasses can introduce specific attributes and methods for different task categories. This flexibility ensures that the Task Manager can handle various tasks, such as personal, work-related, academic, or project tasks, without the need for extensive code duplication.

- Consistency via Polymorphism: Polymorphism is another vital element. By defining a consistent interface for task management, regardless of task type, we ensure that users can interact with the system in a uniform manner. Users can add, edit, delete, and prioritise tasks without needing to understand the specific details of each task class. The system adapts dynamically, enabling these operations on different task types.

CLASS DIAGRAM



Code snippets:

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <ctime>
#include <chrono>
#include <thread>

class Task {
public:
    Task(const std::string& title, const std::string& description, const std
        ::string& priority)
        : title(title), description(description), priority(priority),
            reminderSet(false) {}

    virtual void displayInfo() const {
        std::cout << "Title: " << title << std::endl;
        std::cout << "Description: " << description << std::endl;
        std::cout << "Priority: " << priority << std::endl;
        if (reminderSet) {
            std::cout << "Reminder: Set" << std::endl;
        }
    }

    virtual std::string getTaskDate() const {
        return ""; // Default implementation returns an empty string
    }

    std::string getPriority() const {
        return priority;
    }
```

```
    }

    bool isReminderSet() const {
        return reminderSet;
    }

    void setReminder() {
        reminderSet = true;
    }

private:
    std::string title;
    std::string description;
    std::string priority;
    bool reminderSet;
};

class WorkTask : public Task {
public:
    WorkTask(const std::string& title, const std::string& description, const
        std::string& deadline, const std::string& priority)
        : Task(title, description, priority), deadline(deadline) {}

    void displayInfo() const override {
        Task::displayInfo();
        std::cout << "Deadline: " << deadline << std::endl;
        std::cout << "Task Type: Work Task" << std::endl;
    }

    std::string getTaskDate() const override {
```

```
    std::string getTaskDate() const override {
        return deadline;
    }

private:
    std::string deadline;
};

class HomeTask : public Task {
public:
    HomeTask(const std::string& title, const std::string& description, const
        std::string& dueDate, const std::string& priority)
        : Task(title, description, priority), dueDate(dueDate) {}

    void displayInfo() const override {
        Task::displayInfo();
        std::cout << "Due Date: " << dueDate << std::endl;
        std::cout << "Task Type: Home Task" << std::endl;
    }

    std::string getTaskDate() const override {
        return dueDate;
    }

private:
    std::string dueDate;
};

int getDayOfWeek(int year, int month, int day) {
```

```
int getDayOfWeek(int year, int month, int day) {
    if (month < 3) {
        month += 12;
        year--;
    }

    int K = year % 100;
    int J = year / 100;

    int dayOfWeek = (day + 13 * (month + 1) / 5 + K + K / 4 + J / 4 - 2 * J)
        % 7;

    if (dayOfWeek <= 0) {
        dayOfWeek += 7;
    }

    return dayOfWeek - 1;
}

void printCalendar(int year, int month, const std::vector<Task*>& tasks) {
    int daysInMonth;
    int currentDayOfWeek = getDayOfWeek(year, month, 1);

    switch (month) {
        case 1:
            daysInMonth = (year % 4 == 0 && (year % 100 != 0 || year % 400 ==
                0)) ? 29 : 28;
            break;
        default:
            int daysInMonths[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
                30, 31};
            daysInMonth = daysInMonths[month];
            break;
    }

    std::cout << "Calendar for the specified month:" << std::endl;
    std::cout << " Year: " << year << " Month: " << month << std::endl;
    std::cout << "----------------------------" << std::endl;
    std::cout << " Sun Mon Tue Wed Thu Fri Sat" << std::endl;
    std::cout << "----------------------------" << std::endl;

    for (int i = 0; i < currentDayOfWeek; i++) {
        std::cout << "    ";
    }

    for (int day = 1; day <= daysInMonth; day++) {
        if (day < 10) {
            std::cout << " ";
        }

        bool hasTasks = false;
        for (const Task* task : tasks) {
            if (task->getTaskDate() == std::to_string(year) + "-" + std
                ::to_string(month) + "-" + std::to_string(day)) {
                std::cout << "[" << day << "]";
                hasTasks = true;
                if (task->isReminderSet()) {
                    std::cout << " (R)";
```

```
136                     std::cout << " (R)";
137                 }
138                 break;
139             }
140         }
141
142         if (!hasTasks) {
143             std::cout << day << "  ";
144         }
145
146         if (++currentDayOfWeek > 6) {
147             currentDayOfWeek = 0;
148             std::cout << std::endl;
149         }
150     }
151
152     std::cout << std::endl;
153     std::cout << "----------------------------" << std::endl;
154
155     for (int day = 1; day <= daysInMonth; day++) {
156         for (const Task* task : tasks) {
157             if (task->getTaskDate() == std::to_string(year) + "-" + std
                    ::to_string(month) + "-" + std::to_string(day)) {
158                 std::cout << "Tasks on " << year << "-" << month << "-" <<
                        day << ":\n";
159                 task->displayInfo();
160                 std::cout << std::endl;
161             }
162         }
```

```
164 }
165
166 void setTaskReminder(std::vector<Task*>& tasks, int taskIndex) {
167     if (taskIndex >= 0 && taskIndex < tasks.size()) {
168         tasks[taskIndex]->setReminder();
169         std::cout << "Reminder set for the task." << std::endl;
170     } else {
171         std::cout << "Invalid task index. Reminder not set." << std::endl;
172     }
173 }
174
175 int main() {
176     std::vector<Task*> tasks;
177     int choice;
178
179     do {
180         std::cout << "Task Manager Menu" << std::endl;
181         std::cout << "1. Add a Work Task" << std::endl;
182         std::cout << "2. Add a Home Task" << std::endl;
183         std::cout << "3. Display Tasks" << std::endl;
184         std::cout << "4. Print Calendar" << std::endl;
185         std::cout << "5. Set Task Reminder" << std::endl;
186         std::cout << "6. Exit" << std::endl;
187         std::cout << "Enter your choice: ";
188         std::cin >> choice;
189
190         switch (choice) {
191             case 1: {
192                 std::string title, description, deadline, priority;
```

```
192                 std::string title, description, deadline, priority;
193                 std::cout << "Enter title: ";
194                 std::cin.ignore();
195                 std::getline(std::cin, title);
196                 std::cout << "Enter description: ";
197                 std::getline(std::cin, description);
198                 std::cout << "Enter deadline (YYYY-MM-DD): ";
199                 std::cin >> deadline;
200                 std::cout << "Enter priority (High/Medium/Low): ";
201                 std::cin >> priority;
202                 Task* task = new WorkTask(title, description, deadline,
                        priority);
203                 tasks.push_back(task);
204                 std::cout << "Work Task added." << std::endl;
205                 break;
206             }
207             case 2: {
208                 std::string title, description, dueDate, priority;
209                 std::cout << "Enter title: ";
210                 std::cin.ignore();
211                 std::getline(std::cin, title);
212                 std::cout << "Enter description: ";
213                 std::getline(std::cin, description);
214                 std::cout << "Enter due date (YYYY-MM-DD): ";
215                 std::cin >> dueDate;
216                 std::cout << "Enter priority (High/Medium/Low): ";
217                 std::cin >> priority;
218                 Task* task = new HomeTask(title, description, dueDate,
                        priority);
219                 tasks.push_back(task);
```

```
219                 tasks.push_back(task);
220                 std::cout << "Home Task added." << std::endl;
221                 break;
222             }
223             case 3:
224                 std::cout << "Tasks:" << std::endl;
225                 for (size_t i = 0; i < tasks.size(); i++) {
226                     std::cout << "Task " << i + 1 << ":" << std::endl;
227                     tasks[i]->displayInfo();
228                     std::cout << std::endl;
229                 }
230                 break;
231             case 4: {
232                 int year, month;
233                 std::cout << "Enter year (e.g., 2023): ";
234                 std::cin >> year;
235                 std::cout << "Enter month (1-12): ";
236                 std::cin >> month;
237                 if (month < 1 || month > 12) {
238                     std::cout << "Invalid month. Please try again." << std
                            ::endl;
239                 } else {
240                     printCalendar(year, month, tasks);
241                 }
242                 break;
243             }
244             case 5: {
245                 int taskIndex;
246                 std::cout << "Enter the task index to set a reminder: ";
```

```
238                     std::cout << "Invalid month. Please try again." << std
                            ::endl;
239                 } else {
240                     printCalendar(year, month, tasks);
241                 }
242                 break;
243             }
244             case 5: {
245                 int taskIndex;
246                 std::cout << "Enter the task index to set a reminder: ";
247                 std::cin >> taskIndex;
248                 setTaskReminder(tasks, taskIndex - 1);
249                 break;
250             }
251             case 6:
252                 std::cout << "Exiting the program." << std::endl;
253                 break;
254             default:
255                 std::cout << "Invalid choice. Please try again." << std::endl
                        ;
256         }
257     } while (choice != 6);
258
259     for (Task* task : tasks) {
260         delete task;
261     }
262
263     return 0;
264 }
```

OUTPUT

```
/tmp/uUDRoEt0Gj.o
Task Manager Menu
1. Add a Work Task
2. Add a Home Task
3. Display Tasks
4. Print Calendar
5. Set Task Reminder
6. Exit
Enter your choice: 1
Enter title: Write a report
Enter description: Report based on digital marketing and finances
Enter deadline (YYYY-MM-DD): 2023-11-9
Enter priority (High/Medium/Low): High
Work Task added.
Task Manager Menu
1. Add a Work Task
2. Add a Home Task
3. Display Tasks
4. Print Calendar
5. Set Task Reminder
6. Exit
Enter your choice: 2
Enter title: Mom's birthday
Enter description: Order her favourite watch online
Enter due date (YYYY-MM-DD): 2023-11-22
Enter priority (High/Medium/Low): Medium
Home Task added.
Task Manager Menu
1. Add a Work Task
2. Add a Home Task
```

```
2. Add a Home Task
3. Display Tasks
4. Print Calendar
5. Set Task Reminder
6. Exit
Enter your choice: 4
Enter year (e.g., 2023): 2023
Enter month (1-12): 11
Calendar for the specified month:
Year: 2023 Month: 11
---------------------------
 Sun Mon Tue Wed Thu Fri Sat
---------------------------
                1   2   3   4
 5   6   7   8  [9]10  11
12  13  14  15  16  17  18
19  20  21 [22]23  24  25
26  27  28  29  30
---------------------------
Tasks on 2023-11-9:
Title: Write a report
Description: Report based on digital marketing and finances
Priority: High
Deadline: 2023-11-9
Task Type: Work Task

Tasks on 2023-11-22:
Title: Mom's birthday
Description: Order her favourite watch online
Priority: Medium
Due Date: 2023-11-22
Task Type: Home Task

Task Manager Menu
1. Add a Work Task
2. Add a Home Task
3. Display Tasks
4. Print Calendar
5. Set Task Reminder
6. Exit
Enter your choice: 5
Enter the task index to set a reminder: 1
Reminder set for the task.
Task Manager Menu
1. Add a Work Task
2. Add a Home Task
3. Display Tasks
4. Print Calendar
5. Set Task Reminder
6. Exit
Enter your choice: 3
Tasks:
Task 1:
Title: Write a report
Description: Report based on digital marketing and finances
Priority: High
Reminder: Set
Deadline: 2023-11-9
Task Type: Work Task
```

```
4. Print Calendar
5. Set Task Reminder
6. Exit
Enter your choice: 3
Tasks:
Task 1:
Title: Write a report
Description: Report based on digital marketing and finances
Priority: High
Reminder: Set
Deadline: 2023-11-9
Task Type: Work Task

Task 2:
Title: Mom's birthday
Description: Order her favourite watch online
Priority: Medium
Due Date: 2023-11-22
Task Type: Home Task

Task Manager Menu
1. Add a Work Task
2. Add a Home Task
3. Display Tasks
4. Print Calendar
5. Set Task Reminder
6. Exit
Enter your choice: 6
Exiting the program.
```

Conclusion

In conclusion, this project is dedicated to developing a Task Manager that harnesses the power of inheritance and polymorphism in object-oriented programming. The system offers a flexible, efficient, and user-friendly solution for managing tasks of various types. The key features, including task classification, scheduling, reminders, and data persistence, make this Task Manager a valuable tool for enhancing productivity and time management. Through this project, we demonstrate the significant benefits of applying advanced software engineering concepts to real-world problem-solving, paving the way for a more organized and productive future.