A

ProjectReport

On

CUSTOMTASK SCHEDULAR

Submittedinpartialfulfillmentoftherequirementforthedegreeof

BachelorofTechnology

In

ComputerScienceandEngineering

By

Akshit Dumka(2261081)

Diyansh Rana(2261203)

Piyush Kumar(2261419)

SirajMehra(2261542)

UndertheGuidanceof

Mr. Prince Kumar

ASSISTANT PROFESSOR

DEPARTMENTOFCOMPUTERSCIENCEANDENGINEERING

DEPARTMENTOFCOMPUTERSCIENCE&ENGINEERING

GRAPHICERAHILLUNIVERSITY,BHIMTALCAMPUS

SATTALROAD,P.O.BHOWALI,

DISTRICT- NAINITAL-263132

2024-2025

STUDENT’SDECLARATION

We, **Akshit Dumka , Diyansh Rana , Piyush Kumar andSiraj Mehra**hereby declare the work, which is being presented in the project, entitled ‘**Custom Task Schedular**’ in partial fulfillmentoftherequirement fortheawardofthedegree **BachelorofTechnology(B.Tech.)** in the session **2024-2025**, is an authentic record of my work carried out under the supervision of Mr. Prince Kumar

Thematterembodiedinthisprojecthasnotbeensubmittedbymefortheawardofanyother degree.

Date:

AkshitDumka(2261081)

DiyanshRana(2261203)

PiyushKumar(2261419)

SirajMehra(2261542)

CERTIFICATE

The project report entitled "Custom Task Schedular" being submitted by by Diyansh Rana (2261203) s/o Mr, Ishwar Singh Rana , Akshit Dumka (2261081) s/o Mr. D.N. Dumka , Piyush Kumar(2261419)s/oMr.SanjayKumarandSirajMehra(2261542)s/oMr.SurendraSinghMehra **ofB.Tech.(CSE)toGraphic Era HillUniversity BhimtalCampus forthe award ofbonafide work carried out by them. They have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.**

Mr.PrinceKumar

(ProjectGuide)

Dr.AnkurSinghBisht

(Head, CSE)

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director **‘Prof. (Col.) Anil Nair (Retd.)’,** GEHUBhimtalCampus to permit me and carryoutthis project workwith his excellent and optimistic supervision. This has all beenpossible due to his novelinspiration, able guidance, andusefulsuggestionsthathelpedmetodevelopasacreativeresearcherandcompletetheresearch work, in time.

WordsareinadequateinofferingmythankstoGODforprovidingmewitheverythingthat we need. We again want to extend thanks to our president **‘Prof. (Dr.) Kamal Ghanshala’** for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to **‘Dr. Ankur Singh Bisht’** (Head, Department of Computer Science and Engineering, GEHUBhimtalCampus), ourproject guide **‘<NameofProjectGuide>’** (Assistant Professor,DepartmentofComputerScienceandEngineering,GEHUBhimtalCampus)andother faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this report.

Finally, yet importantly, We would like to express my heartiest thanks to our beloved parents, for their moral support, affection, and blessings. We would also like to pay our sincere thanks to all my friends and well-wishers fortheir help and wishes for the successfulcompletion of this project.

AkshitDumka(2261081)

DiyanshRana(2261203)

PiyushKumar(2261419)

SirajMehra(2261542)

Abstract

The**CustomTaskScheduler** isasoftwareengineeringproject developedto efficientlymanage and automate the execution of user-defined tasks based on specified conditions such as time, priority, or dependencies. This scheduler provides a flexible and user-friendly interface for task creation, modification,deletion,andscheduling,ensuringtasksareexecutedinanorganizedand timely manner.

The project is built using **Object-Oriented Programming (OOP)** principles, ensuring modularity,reusability,andscalability. KeyOOPconcepts suchas**classes,objects,inheritance, polymorphism, encapsulation**, and **abstraction** are applied throughout the design and implementation of the system. For example, different types of tasks can be represented as subclasses inheriting from a common Task base class, allowing for easy extension and maintenance.

Thistaskschedulercanbeused invariousreal-lifeapplications, includingpersonalproductivity tools, automated software systems, or enterprise-leveltask management. It improves efficiency by automating routine tasks and helps users prioritize and track their workflow effectively.

Thegoalofthisproject istodemonstratehowOOPcanbeusedtocreatearobust, maintainable, and extensible software solution that solves real-world problems in task management and automation.

TABLEOF CONTENTS

LISTOF ABBREVIATIONS

- **OOP(Object-OrientedProgramming):** Aprogrammingparadigmbasedontheconceptof "objects"that containdataand methods. It helps in designing modular and reusablecodeusing classes, inheritance, and polymorphism.

- **GUI (Graphical User Interface):** A visual interface that allows users to interact with the softwarethroughbuttons,forms,and other graphicalelementsinsteadoftext-based commands.

- **IDE(IntegratedDevelopmentEnvironment):** Asoftwaresuitethatprovidestoolslikecode editor, debugger, and compiler in a single interface to help developers write and manage code efficiently.

- **CPU(CentralProcessingUnit):**Thebrainofthecomputerwheremost calculationstake place. It executes instructions from the programs.

- **API(ApplicationProgrammingInterface):** Asetoftoolsandprotocolsthat allowdifferent software components to communicate with each other.

- **DBMS(DatabaseManagementSystem):** Softwareusedtostore,retrieve,andmanagedata in databases. It supports functions like data insertion, updates, and queries.

- **UI(UserInterface):** Thepartofthesoftwarewithwhichtheuser interacts,includingscreens, buttons, icons, and layout.

- **CLI(CommandLineInterface):** Atext-basedinterfacewhereuserstypecommandsto interact with the software or operating system.

- **JSON(JavaScript Object Notation):** Alightweight data-interchange format thatiseasyfor humanstoreadandwrite,and formachinestoparseandgenerate.Oftenused fordataexchange between front-end and back-end systems.

- **XML(eXtensibleMarkupLanguage):**Amarkuplanguageusedto encodedocumentsina readable and structured format. Often used in configuration files or data exchange.

- **UML(UnifiedModelingLanguage):** Astandardizedmodelinglanguageusedtovisualizethe design of a system, including its classes, objects, and workflows.

- **CRUD(Create,Read,Update,Delete):** Thefourbasicoperationsperformedondatabase records.

- **JVM(JavaVirtualMachine):** ApartoftheJavaRuntimeEnvironment thatexecutesJava bytecode, allowing Java programs to run on any platform.

- **SDK(SoftwareDevelopmentKit):**Acollectionoftools, libraries,anddocumentationusedto develop software for a specific platform or framework.

- **OS(OperatingSystem):**Systemsoftwarethatmanagescomputerhardwareandsoftware resources and provides common services for programs.

INTRODUCTION

Prologue

The **CustomTaskScheduler**project aims to simplifyand automatetask management througha user-friendlyandefficient softwaresystem. Built using **Object-OrientedProgramming(OOP)** principles, itdemonstrateshowcoreconceptslike **abstraction,encapsulation,inheritance,and polymorphism** can be applied in real-world applications.

Thisschedulerallowsuserstocreate,prioritize,andscheduletasksbasedonspecificconditions. It reflects not onlyour understanding of OOP and software engineering but also our ability to design practical and maintainable solutions.

Thisproject marksasignificantstepinbridgingacademiclearningwithpractical implementation, enhancing both our technical and problem-solving skills.

BackgroundandMotivations

In both personal and professional life, managing multiple tasks efficiently is a common challenge. Missed deadlines, overlapping work, and disorganized schedules can reduce productivity. Traditionalto-dolistsormanualremindersoftenfailtoprovidetheautomationand flexibility needed in today’s fast-moving world.

Thisprobleminspiredthedevelopmentofa **CustomTaskScheduler**—asoftwaresolutionthat not only organizes tasks but also automates their execution based on user-defined priorities, timings, and dependencies. Unlike basic schedulers, this project focuses on building a modular andscalablesystemusing**Object-OrientedProgramming(OOP)**,making it easyto extendand maintain.

The motivation behind this project is to apply OOP concepts in a meaningfulway, while also solving a real-world problem. It provides a hands-on opportunity to implement software engineeringprinciples,improveprogrammingskills,andcreateatoolthataddsvalueto everyday life and project management.

ProblemStatement

In modern computing environments, managing and executing multiple tasks efficientlyisacommonrequirement.However, mosttraditionaltaskmanagement methodslack automation, flexibility, and the ability to prioritize or schedule tasks dynamically. Users often face difficulties inorganizing their tasks based onurgency, dependency, and timing, leading to missed deadlines and reduced productivity.

Thereisaneedfora**customizabletaskscheduler**that allowsusersto create,modify,prioritize, and automate task execution based on specific criteria. The system should be easy to use, extensible, and capable of handling real-world scheduling requirements.

Thisproject aimsto developa**CustomTaskScheduler**using**Object-OrientedProgramming (OOP)** principles, offering a structured and efficient wayto manage tasks with features suchas time-basedtriggers,priorityhandling,andtaskdependencies,allthroughauser-friendly interface.

ObjectivesandResearchMethodology

ThemainobjectivesoftheCustomTask Scheduler**projectare:**

- Todesignanddevelopasoftwareapplicationthatallowsusersto create,manage,and schedule tasks efficiently.
- Toapply**Object-OrientedProgramming(OOP)**conceptssuchas**classes,inheritance, polymorphism, and encapsulation** in building the system.
- Toprovidefeatureslike**taskprioritization,time-basedscheduling,anddependency management**.
- Tobuilda**user-friendlyGUI**forinteractionandtaskvisualization.
- Toensurethesystemis**modular, scalable,andeasytomaintain**,supporting future enhancements.
- Tohelpusers improve**productivityandtimemanagement**throughautomation.

Tosuccessfullyimplementtheproject,thefollowingresearchmethodologywasfollowed:

1. **LiteratureReview:**Studiedexistingtaskmanagementtoolsandschedulersto understand their features, limitations, and user requirements.
2. **RequirementAnalysis:** Gatheredfunctionalandnon-functionalrequirementsofthe system, focusing on usability, performance, and flexibility.

3. **DesignPhase:**UsedOOPprinciplesto designsystemarchitecturewithmodularclasses for tasks, schedulers, and interfaces. UML diagrams were used to model the system.
4. **Development:**Implementedthesystemusingaprogramminglanguagethatsupports OOP (e.g., Java/Python) and GUI frameworks for the user interface.
5. **Testing:**Performedunit and integrationtestingtoverifythecorrect functioningofeach component and ensure overall system reliability.
6. **Evaluation:** Assessedthesystembasedoncriteria suchasusability,efficiency,andhow well it meets the intended objectives.

ProjectOrganization

Thisproject isorganized intoseveralkeyphasesandcomponentsto ensureasmooth development process and effective management:

1. **RequirementAnalysis:**

   Understandingtheuserneeds,definingfunctionalandnon-functionalrequirements,and outlining the scope of the task scheduler.
2. **SystemDesign:**

   Creating architectural diagrams such as class diagrams and flowcharts using **Object-OrientedProgramming(OOP)**conceptstomodeltasks,scheduler,anduser interface components.

3. **Implementation:**

   WritingthecodeusinganOOP-basedprogramminglanguage(e.g.,Java,Python), developing modules fortask creation, scheduling, prioritymanagement, and GUI.

4. **Testing:**

   Conductingunit testsonindividualmodules,integrationtestingtoverifyinteractions between components, and system testing to validate overall functionality.

5. **Deployment:**

   Packagingthesoftwareforuse,ensuringit runsontheintendedplatform,andpreparing user documentation.

6. **MaintenanceandFuture Enhancements:**

   Planning forupdates,bugfixes,andpotentialadditionofnew featuresbasedonuser feedback.

PHASESOFSOFTWAREDEVELOPMENTCYCLE

HARDWAREANDSOFTWAREREQUIREMENTS

HardwareRequirement

Windows	OSX	Linux
MicrosoftWindows 8/7/Vista/10 (32or64bit)	MacOSX10.8.5or higher,up to10.9 (Mavericks)	GNOME orKDEorUnity desktop
4GBRAM minimum,8GB RAMrecommended	2GBRAM minimum,4GB RAMrecommended	2GBRAM minimum,4GB RAMrecommended
400MBharddiskspaceplus additionalspaceforappdata and cache	400MBharddiskspaceplus additionalspaceforappdata and cache	400MBharddiskspaceplus additionalspaceforappdata and cache
JavaDevelopmentKit(JDK)8 orhigher	OracleJavaDevelopment Kit (JDK)	GNUCLibrary(Glibc)2.11 orlater
Optional: Intel processor with IntelVT-xsupportforenhanced performance	Optional: Intel processor with IntelVT-xsupportforenhanced performance	Optional:IntelVT-xsupport recommended

SoftwareRequirement

AndroidSoftwareRequirements

- **AndroidStudioIDE**(latest stableversion)

- **JavaDevelopmentKit(JDK)**8orhigher,or **OpenJDK**
- **AndroidSDK** withnecessaryplatformtoolsandbuildtools
- **GradleBuildSystem**forprojectautomation
- **EmulatorSystemImages**fortestingdifferentAndroidversionsanddeviceconfigurations
- **Optional:** Additional libraries or plugins required by your project (e.g., support libraries, UI frameworks

CODINGOF FUNCTIONS

1. FirstComeFirstServed(FCFS)SchedulingAlgorithm

```
def fcfs(processes):
    processes.sort(key=lambda x: x.arrival_time)
    current_time = 0
    for process in processes:
        if current_time < process.arrival_time:
            current_time = process.arrival_time
        current_time += process.burst_time
        process.completion_time = current_time
        process.turnaround_time = process.completion_time - process.arrival_time
        process.waiting_time = process.turnaround_time - process.burst_time
```

The provided Python code defines a function `fcfs(processes)`that implements the **First-Come, First-Served (FCFS)** CPU scheduling algorithm. It takes a list of `processes`as input, sortsthembased on their arrivaltime, and then simulates their execution in that order. For each process,itcalculatesandassignsthe`completion_time`,`turnaround_time`,and`waiting_time` as attributes of the process object. This function is a basic building block for a task scheduler, demonstrating a simple scheduling strategy

2. ShortestJobSchedulingAlgorithm

```
def sjf(processes):
    n = len(processes)
    completed = 0
    current_time = 0
    is_completed = [False] * n

    while completed != n:
        idx = -1
        min_burst = float('inf')

        for i in range(n):
            if (processes[i].arrival_time <= current_time) and (not is_completed[i]):
                if processes[i].burst_time < min_burst:
                    min_burst = processes[i].burst_time
                    idx = i
                elif processes[i].burst_time == min_burst:
                    if processes[i].arrival_time < processes[idx].arrival_time:
                        idx = i

        if idx != -1:
            p = processes[idx]
            current_time += p.burst_time
            p.completion_time = current_time
            p.turnaround_time = p.completion_time - p.arrival_time
            p.waiting_time = p.turnaround_time - p.burst_time
            is_completed[idx] = True
            completed += 1
        else:
            current_time += 1
```

This Python code implements the **Shortest Job First (SJF)** scheduling algorithm. It iterates throughthe given list of `processes`, at each step selecting the process that has arrived and has the shortest remaining burst time for execution. It keeps track of the `current_time`, marks processes as `completed`, and calculates their `completion_time`, `turnaround_time`, and `waiting_time`.Thealgorithmprioritizesshortertasksto potentiallyreducetheaveragewaiting time.

3. PrioritySchedulingAlgorithm

```python
def priority_scheduling(processes):
    processes.sort(key=lambda x: (x.arrival_time, x.priority))
    current_time = 0
    completed = 0
    n = len(processes)
    is_completed = [False] * n

    while completed != n:
        idx = -1
        min_priority = float('inf')

        for i in range(n):
            if processes[i].arrival_time <= current_time and not is_completed[i]:
                if processes[i].priority < min_priority:
                    min_priority = processes[i].priority
                    idx = i
                elif processes[i].priority == min_priority:
                    if processes[i].arrival_time < processes[idx].arrival_time:
                        idx = i

        if idx != -1:
            p = processes[idx]
            current_time += p.burst_time
            p.completion_time = current_time
            p.turnaround_time = p.completion_time - p.arrival_time
            p.waiting_time = p.turnaround_time - p.burst_time
            is_completed[idx] = True
            completed += 1
        else:
            current_time += 1
```

This Python code implements a **Priority Scheduling** algorithm. It first sorts the incoming `processes`based ontheir arrivaltime, and thenprimarily bytheir priority(lower value indicates higherpriority).Thealgorithmtheniterativelyselectsthehighest priorityprocessthat hasarrived and executes it. It tracks the `current_time`, marks completed processes, and calculates their `completion_time`, `turnaround_time`, and `waiting_time`. Incase ofatie inpriority, processes are chosen based on their arrival time (FCFS for equal priority).

4. ShortestRemainingTimeFirst(SRTF)

```
def srtf(processes):
    n = len(processes)
    current_time = 0
    completed = 0
    remaining_times = [p.burst_time for p in processes]
    is_completed = [False] * n

    while completed != n:
        idx = -1
        min_remaining = float('inf')

        for i in range(n):
            if processes[i].arrival_time <= current_time and not is_completed[i]:
                if remaining_times[i] < min_remaining:
                    min_remaining = remaining_times[i]
                    idx = i

        if idx != -1:
            remaining_times[idx] -= 1
            current_time += 1

            if remaining_times[idx] == 0:
                p = processes[idx]
                p.completion_time = current_time
                p.turnaround_time = p.completion_time - p.arrival_time
                p.waiting_time = p.turnaround_time - p.burst_time
                is_completed[idx] = True
                completed += 1
```

This Python code implements the **Shortest Remaining Time First (SRTF)** scheduling algorithm, which is a preemptive version of Shortest Job First. It keeps track of the `remaining_times`foreachprocess.At eachtimestep,it selectstheprocessthat hasarrivedand has the smallest remaining burst time to execute. If a new process arrives with a shorter remainingburst timethanthecurrentlyexecuting process,thecurrent processispreempted.The code updates `remaining_times`and `current_time`, and once a process completes (remaining time becomes 0), it calculates its `completion_time`, `turnaround_time`, and `waiting_time`.

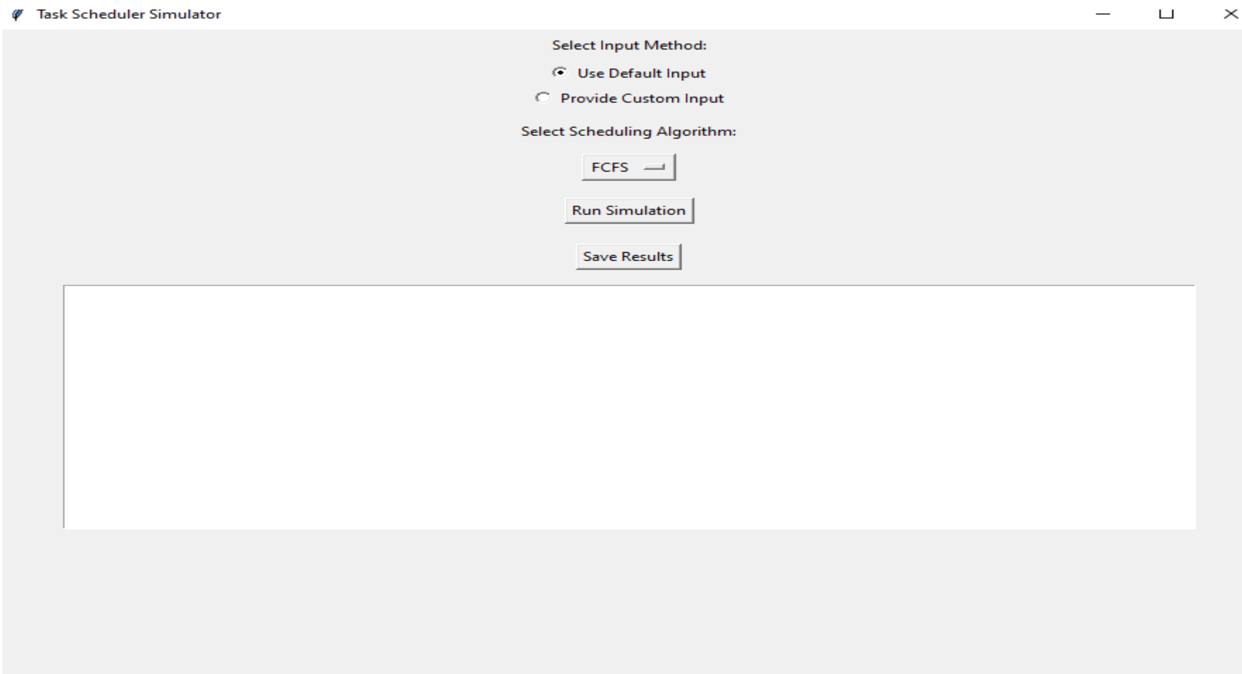
5.

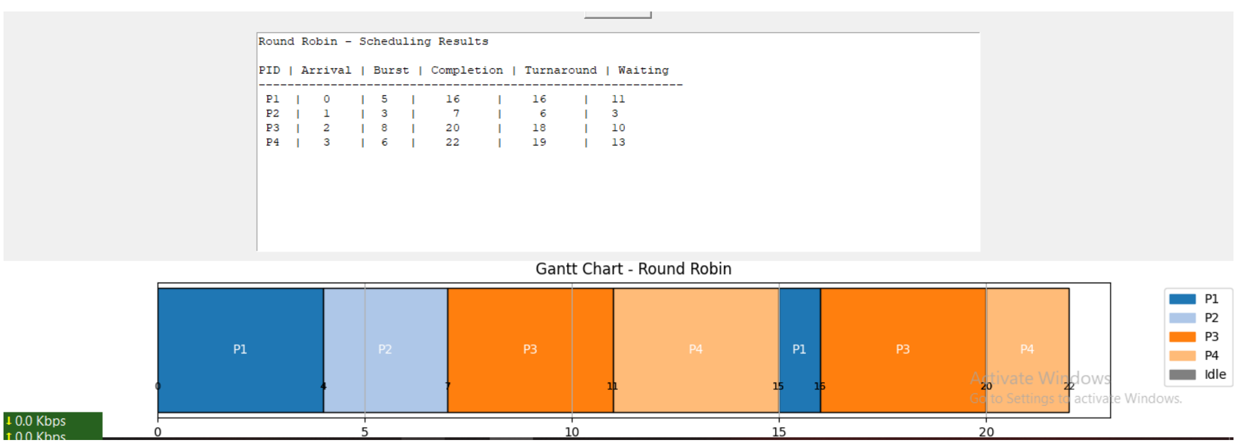
```python
def print_results(processes):
    print("PID | Arrival | Burst | Completion | Turnaround | Waiting")
    print("------------------------------------------------------------")
    for p in processes:
        print(f"{p.pid:^5}|{p.arrival_time:^8}|{p.burst_time:^6}|{p.completion_time:^11}|{p.turnaround_


if __name__ == "__main__":
    process_list = [
        Process("P1", 0, 5),
        Process("P2", 1, 3),
        Process("P3", 2, 8),
        Process("P4", 3, 6),
    ]

    # Example usage:
    fcfs(process_list)
    print_results(process_list)
```

This Python code defines a function `print_results(processes)` that takes a list of process objectsasinput andneatlyprintsatableoftheirscheduling metrics.Thetable includescolumns for Process ID (PID), ArrivalTime, Burst Time, Completion Time, Turnaround Time, and Waiting Time.

The `if name== "main":` block demonstrates example usage. It creates a list of `Process`objects (assuming a `Process`class with attributes like `pid`, `arrival_time`, `burst_time`, `completion_time`, `turnaround_time`, and `waiting_time`exists). The commented-out lines show how you would typically call a scheduling algorithm (like `fcfs`) to processthislist andthenuse `print_results`todisplaythecalculatedresults.Thispart servesas a simple driver or test case for your scheduling functions.

SNAPSHOT





PID	Arrival	Burst	Completion	Turnaround	Waiting
P1	0	5	16	16	11
P2	1	3	7	6	3
P3	2	8	20	18	10
P4	3	6	22	19	13

LIMITATIONS

- Thescheduler maynothandleextremelylarge numbersoftasksefficientlywithout performance degradation, depending on system resources.

- Taskschedulingislimitedtopredefinedcriteriasuchastime,priority,anddependencies;it may not support complex or dynamic scheduling rules.

- Theapplicationreliesonthehostsystem'sclock andresources,so inaccuraciesor interruptions in the system may affect task execution.

- Itmaylack integrationwithexternalcalendarornotificationsystemsunlessspecifically extended.

- TheGUIandfeaturesmight belimitedtodesktopenvironmentsandmaynot beoptimizedfor mobile platforms unless separately developed.

- Persistent storage(ifused)dependsonthechosendatabaseorfilesystemand might have constraints regarding scalability and concurrency.

- Real-timetaskexecutionwithstricttimingguaranteesmaynotbefeasibleinnon-real-time operating systems.

ENHANCEMENTS

Futureenhancementsofthisprojectarethat–

- **IntegrationwithCloudServices:**Enablesynchronizationoftasksacrossmultipledevices using cloud storage for seamless access anywhere.

- **MobileAppDevelopment:**CreatemobileversionsforAndroidandiOStoallowusersto manage tasks on the go.

- **AdvancedSchedulingFeatures:**Supportrecurringtasks,flexibletimeintervals,and conditional triggers for more dynamic task management.

- **NotificationandAlertSystem:** Addpushnotifications,emailreminders,andalertsto keep users informed about upcoming tasks and deadlines.

- **Multi-UserCollaboration:**Introduceuseraccountswithsharedtasklists,permissions,and collaborative scheduling for team projects.

- **AI-PoweredTaskManagement:**Incorporateartificialintelligencetoautomaticallyprioritize tasks and predict user needs based on past behavior.

- **IntegrationwithThird-PartyTools:**Connectwithpopularcalendarapps,project management software, and communication platforms for enhanced workflow.

- **PerformanceImprovements:**Optimizetheschedulertoefficientlyhandlealargernumberof tasks and complex dependencies without lag

CONCLUSION

The **Custom Task Scheduler** project successfully demonstrates the practical application of Object-OrientedProgrammingconceptsindevelopingausefulandefficient softwaretool.By enablinguserstocreate,prioritize,andautomatetasks,thesystemhelpsimproveproductivity and time management.

Through this project, we gained hands-on experience in software design, development, and testing,whileemphasizingmodularityandscalability. Althoughtherearesomelimitations,the foundation laid bythis project opens up manypossibilities for future enhancements and real-world usage.

Overall,theproject servesasavaluable learningexperienceandasteptowardsbuilding more complex and user-centric software solutions.

REFERENCES

GitHubRepository– *CustomTaskSchedulerProject*

Availableat:https://github.com/Akshit-Dumka/Akshit-dumka13e