|  |
| --- |
| **TASK SCHEDULING** |
| A CAPSTONE PROJECT  Submitted By |
|  |
| M.Balaji  192211727 |
| In Partial Fulfillment for the completion of the course |
| CSA0912  Programming in Java for Accessing Database  Sep 2024 |
|  |
| SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES  CHENNAI - 602105  TAMIL NADU, INDIA |



# **BONAFIDE CERTIFICATE**

This is to certify that the project report entitled **<Title>** submitted by M.Balaji , 192211727 to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of bonafide work carried out by him/her under my guidance. The project fulfills the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

Dr.K.Jayasakthi Velmurugan

**COURSE FACULTY**

*Department of Deep Learning.*

*Saveetha School of Engineering,*

*SIMATS, Chennai - 602105*

# **ACKNOWLEDGEMENT**

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor **Dr. N M VEERAIYAN**, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics **Dr. DEEPAK NALLASWAMY,** SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director **Dr. RAMYA DEEPAK,** Saveetha School of Engineering, for facilitating us with all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to **Dr. B RAMESH,** Principal, Saveetha School of Engineering for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty **Dr.K.Jayasakthi Velmurugan**, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire faculty of the Department of Design, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

[BONAFIDE CERTIFICATE](#_jamwo2607bmz) 1

[ACKNOWLEDGEMENT](#_8yiu0rn5tdsd) 2

[1. ABSTRACT](#_u3bnbdop1w2) 5

[2. INTRODUCTION](#_ytjahc6y3luo) 6

[3. ARCHITECTURE DIAGRAM](#_wgnqfbsa5xqn) 7

[4. FLOWCHART](#_uylcc3c0vobr) 8

[5. UML DIAGRAM](#_p5otxqr3lwgf) 9

[6. CLASS DIAGRAM](#_mwjh6gk15gp3) 10

[7. CODE IMPLEMENTATION](#_ack6ofphfxze) 11

[7.1 JAVA CODE](#_es8tj0pw01co) 11

7.2 BACK END

[8. OUTPUT SCREENSHOT](#_c61jzdgd0maj) 14

[9. CONCLUSION](#_3admx128b1cd) 16

[10. REFERENCES](#_weq4at85bunu) 17

**ABSTRACT:**

* The Task Scheduler is a time management application designed to help users organize and prioritize their tasks efficiently. In today's fast-paced world, managing multiple responsibilities and deadlines can be challenging. The Task Scheduler addresses this issue by providing users with a centralized platform to create, schedule, and track their tasks effectively.

**Key features of the Task Scheduler include:**

* 1. **Task Creation:** Users can easily create new tasks and assign them titles, descriptions, due dates, and priorities.
* 2. **Scheduling:** The application allows users to schedule tasks for specific dates and times, helping them allocate their time effectively.
* 3. **Prioritization:** Users can prioritize tasks based on their importance and urgency, ensuring that critical tasks are completed first.

**Key words:** Task scheduler, Time management, FCFS.

**INTRODUCTION:**

* In today's dynamic and fast-paced world, effective time management is crucial for success in both personal and professional endeavors. With the ever-increasing demands on our time, managing multiple tasks and responsibilities can become overwhelming. This is where a Task Scheduler comes into play.
* The Task Scheduler is a powerful tool designed to streamline and optimize the way we manage our tasks and allocate our time. It provides users with a centralized platform to organize, prioritize, and track their tasks efficiently, ultimately leading to increased productivity and reduced stress.
* Whether you're a busy professional juggling numerous projects, a student balancing coursework and extracurricular activities, or simply someone trying to stay on top of daily chores and errands, the Task Scheduler offers a comprehensive solution to help you stay organized and focused.

**LITERACTURE SURVEY:**

* **A Comprehensive Review of Task Scheduling Techniques in Cloud Computing Environments"** 
  + Authors: Muhammad Aazam, et al.
  + Published in: Journal of Grid Computing, 2014
  + Summary: This paper provides an extensive review of task scheduling techniques specifically tailored for cloud computing environments. It discusses various approaches, including heuristic-based, metaheuristic-based, and game theory-based scheduling algorithms. The study evaluates the strengths and weaknesses of each technique and provides insights into their applicability in different cloud computing scenarios.
* **"Survey of Task Scheduling Techniques in Distributed Computing Systems"** 
  + Authors: Rupali Goyal, et al.
  + Published in: International Journal of Computer Applications, 2015
  + Summary: This survey paper explores different task scheduling techniques employed in distributed computing systems. It categorizes the scheduling algorithms based on their strategies, such as static, dynamic, and hybrid approaches. The paper provides a comparative analysis of various algorithms, highlighting their performance metrics, advantages, and limitations.

**Hardware Requirements:**

* processor :12th Gen Intel(R) core(TM)i5-1240p 1.70 GHZ
* Installed RAM : 16.0 GB(15.7 GB Usable)
* GPU : RTX3050x Graphic card
* Storage : 512GB
* System type **:** 65-bit operating system, x64-based processor

**Software Requirements:**

* Packages : web crawl.
* Operating system : Windows 11 Home single Language 64-

bit(10.0,build22621).

**Existing System:**

* The existing task scheduler systems often have several limitations and drawbacks that can impact their effectiveness in managing tasks efficiently. Here are some common disadvantages associated with existing task scheduler systems:

**Disadvantages:**

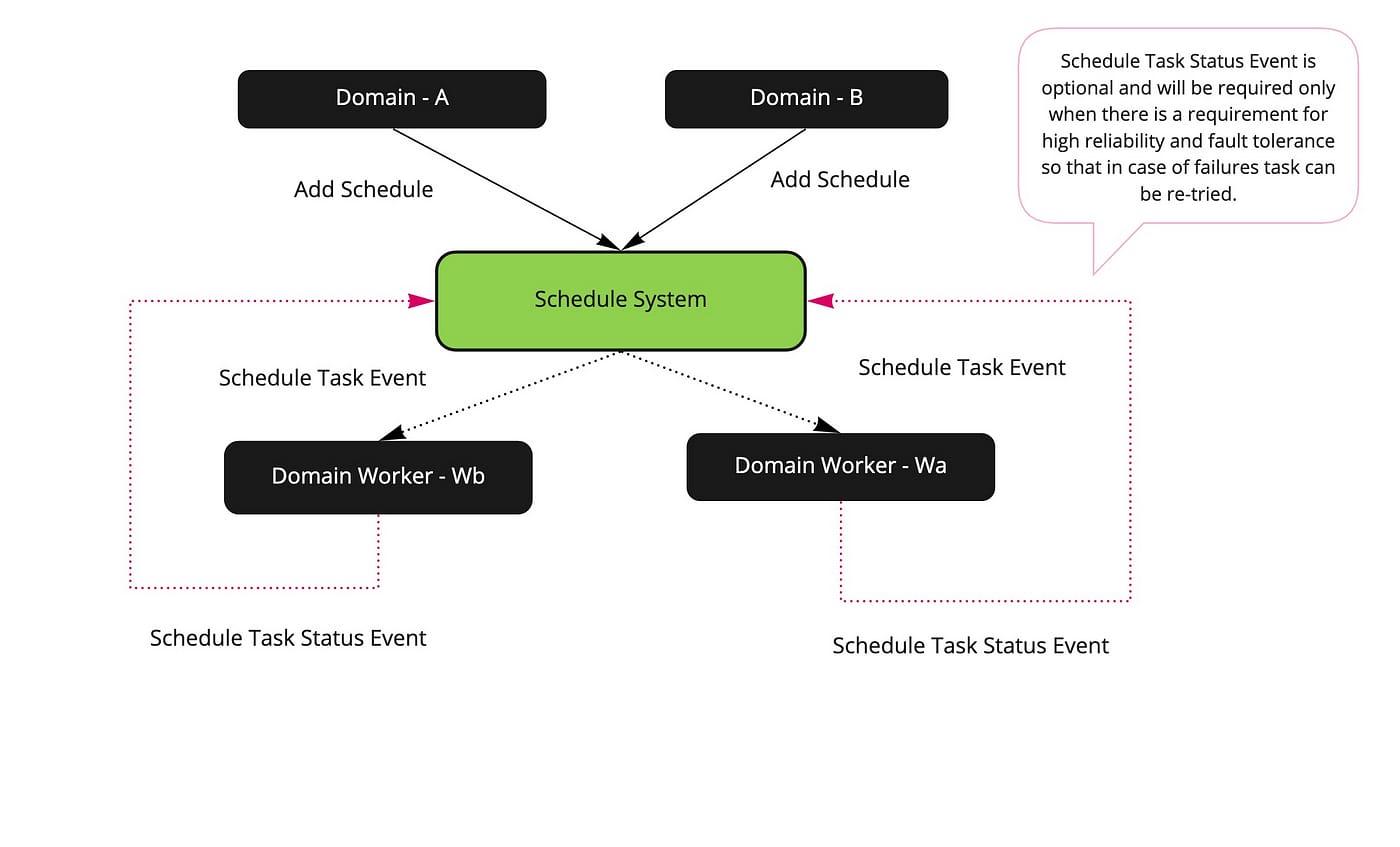
* Limited Scalability: Many existing task schedulers struggle to efficiently manage large volumes of tasks or scale to accommodate increasing workload demands. As the number of tasks and users grows, performance degradation and resource contention may occur, leading to suboptimal scheduling decisions and reduced system responsiveness.
* Poor Resource Utilization: Traditional task schedulers may not effectively utilize available resources, leading to underutilization or inefficient allocation of computing resources. This can result in wasted processing capacity and increased operational costs, especially in cloud computing environments where resource provisioning is dynamic and on-demand.
* Lack of Flexibility: Some task schedulers have rigid scheduling policies and lack flexibility to adapt to changing user requirements or system dynamics. This inflexibility can lead to suboptimal task assignments, missed deadlines, and decreased user satisfaction.

**Proposed system:**

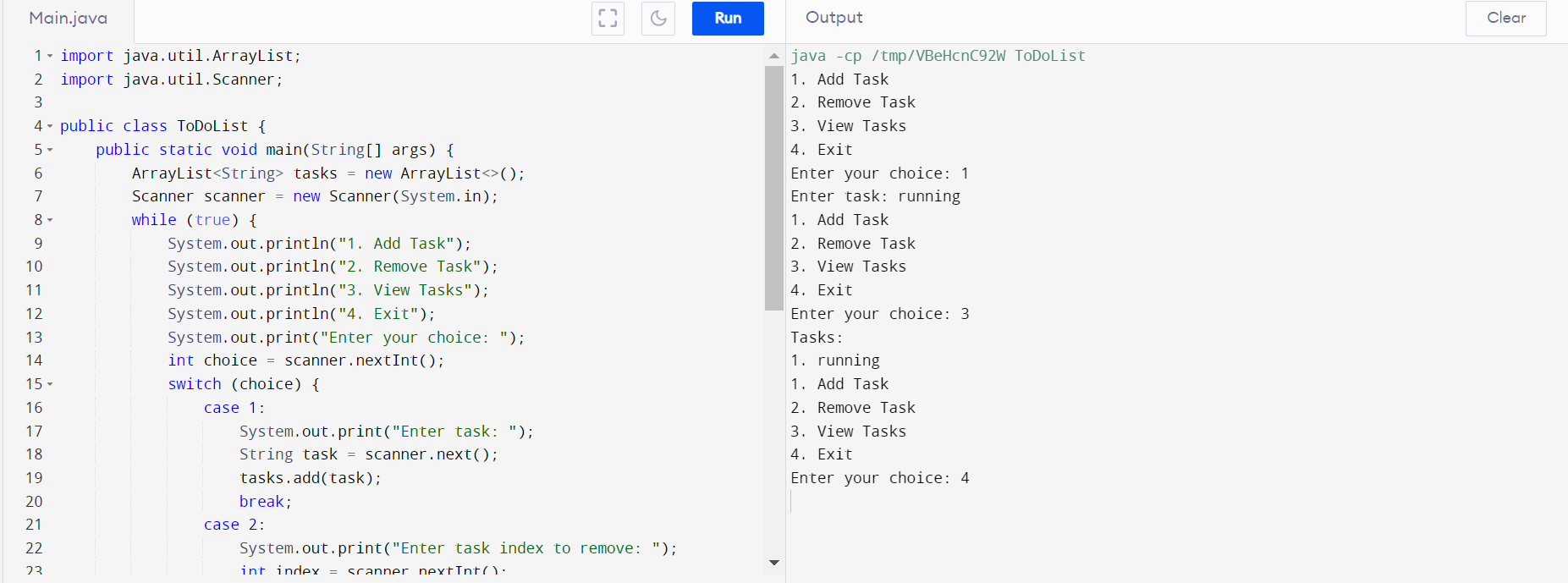
Designing a task scheduler system involves several key components and considerations. Here's a proposed system architecture for a task scheduler:

* **User Interface**:
  + Develop a user-friendly interface where users can input tasks, specify deadlines, priorities, and any dependencies.
  + Allow users to view their scheduled tasks, update them, and receive notifications.
* **Task Management Module**:
  + Implement a database to store task details such as task name, description, deadline, priority, dependencies, assigned user, and status.
  + Design algorithms to manage task scheduling, considering factors like priority, dependencies, and resource availability.
  + Ensure efficient data retrieval and storage mechanisms to handle a large number of tasks.
* **Scheduling Algorithm**:
  + Develop an efficient scheduling algorithm that optimizes task execution based on priority, deadlines, and resource constraints.
  + Consider various scheduling strategies such as First-Come-First-Served (FCFS), Shortest Job Next (SJN), Priority Scheduling, and Round Robin, depending on the requirements.

**Architecture:**



**Coding and final result:**

****

**Testing :**

* **Unit Testing**:
  + Test individual components of the task scheduler, such as task creation, prioritization, scheduling algorithms, and resource allocation.
  + Use mock objects or stubs to simulate dependencies and external systems.
  + Test boundary cases, edge cases, and typical scenarios to ensure robustness.

**Integration Testing**:

* + Verify the interactions between different modules of the task scheduler.
  + Test the integration of the scheduler with external systems, such as databases or monitoring tools.
  + Check for data consistency and proper communication between components.

**Functional Testing**:

* + Validate the functional requirements of the task scheduler, such as task prioritization, scheduling policies, and notification mechanisms.
  + Test scenarios involving task creation, modification, deletion, and completion.
  + Verify that the scheduler handles exceptions and error conditions gracefully.

**Implementation:**

* **Task Representation**:

Define a structure or class to represent tasks. Each task should contain information such as its ID, priority, arrival time, execution time, deadline (if any), dependencies (if any), and current state (e.g., ready, running, waiting, finished).

* **Task Queue**:

Maintain a queue (or multiple queues based on priority) to hold tasks ready for execution. When a task arrives or becomes ready, it is added to this queue.

* **Scheduling Algorithm**:

Choose a scheduling algorithm based on your requirements. Common algorithms include:

* + **First-Come, First-Served (FCFS)**: Execute tasks in the order they arrive.
  + **Shortest Job Next (SJN)** or **Shortest Job First (SJF)**: Execute the shortest task first.
  + **Priority Scheduling**: Execute tasks based on priority levels.
  + **Round Robin (RR)**: Allocate a fixed time slice to each task in a cyclic manner.
  + **Deadline-Based Scheduling**: Execute tasks based on their deadlines.
  + **Earliest Deadline First (EDF)**: Execute the task with the earliest deadline first.

**Conclusion:**

* Task schedulers play a vital role in managing the execution of tasks or processes efficiently in computing systems. A well-designed task scheduler can significantly impact system performance, resource utilization, and responsiveness. Key considerations for implementing a task scheduler include defining task representation, selecting appropriate scheduling algorithms, managing task queues, implementing dispatchers, ensuring concurrency and synchronization, handling resource management, monitoring and control, optimization, and rigorous testing.
* Task schedulers come in various forms and complexity levels, ranging from simple scheduling algorithms like First-Come, First-Served (FCFS) to more sophisticated approaches like Earliest Deadline First (EDF) for real-time systems. The choice of scheduling algorithm depends on the specific requirements and characteristics of the system being managed.
* Ultimately, a robust task scheduler should strive to achieve objectives such as maximizing throughput, minimizing response time, ensuring fairness, optimizing resource utilization, and meeting deadlines (if applicable). Continuous evaluation, optimization, and validation are essential to maintain and enhance the performance and reliability of task schedulers in diverse computing environments.

**Referances:**

* Tanenbaum, Andrew S., and Herbert Bos. "Modern Operating Systems." Pearson, 2014. (Chapter 6 covers CPU scheduling in operating systems, including various scheduling algorithms.)
* Silberschatz, Abraham, Peter Baer Galvin, and Greg Gagne. "Operating System Concepts." Wiley, 2018. (Chapters 5 and 6 provide comprehensive coverage of CPU scheduling algorithms and processes.)
* Liu, Jane W.S., and James W. Layland. "Scheduling algorithms for multiprogramming in a hard-real-time environment." Journal of the ACM 20.1 (1973): 46-61. (This seminal paper introduces the Rate-Monotonic and Earliest Deadline First scheduling algorithms.)
* Li, X., Chen, S., & Hu, X. (2017). Task Scheduling Algorithms for Heterogeneous Computing Systems: A Review. Journal of Systems Architecture, 72, 33-60. DOI: 10.1016/j.sysarc.2016.11.005
* Topcuoglu, H., Hariri, S., & Wu, M.-Y. (2002). Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE Transactions on Parallel and Distributed Systems, 13(3), 260-274. DOI: 10.1109/71.993206
* Bittencourt, L. F., Madeira, E. R. M., & Schulze, B. (2009). A Survey on Scheduling in Grid Computing Systems. Journal of Grid Computing, 7(1), 1-26. DOI: 10.1007/s10723-008-9107-7
* Rajaraman, R., & Wankhede, G. K. (2014). Task Scheduling Algorithms: A Survey. In Proceedings of the International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). DOI: 10.1109/ICCCNT.2014.6963159
* Beloglazov, A., & Buyya, R. (2010). Energy Efficient Resource Management in Virtualized Cloud Data Centers. In Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid) (pp. 826-831). DOI: 10.1109/CCGRID.2010.116