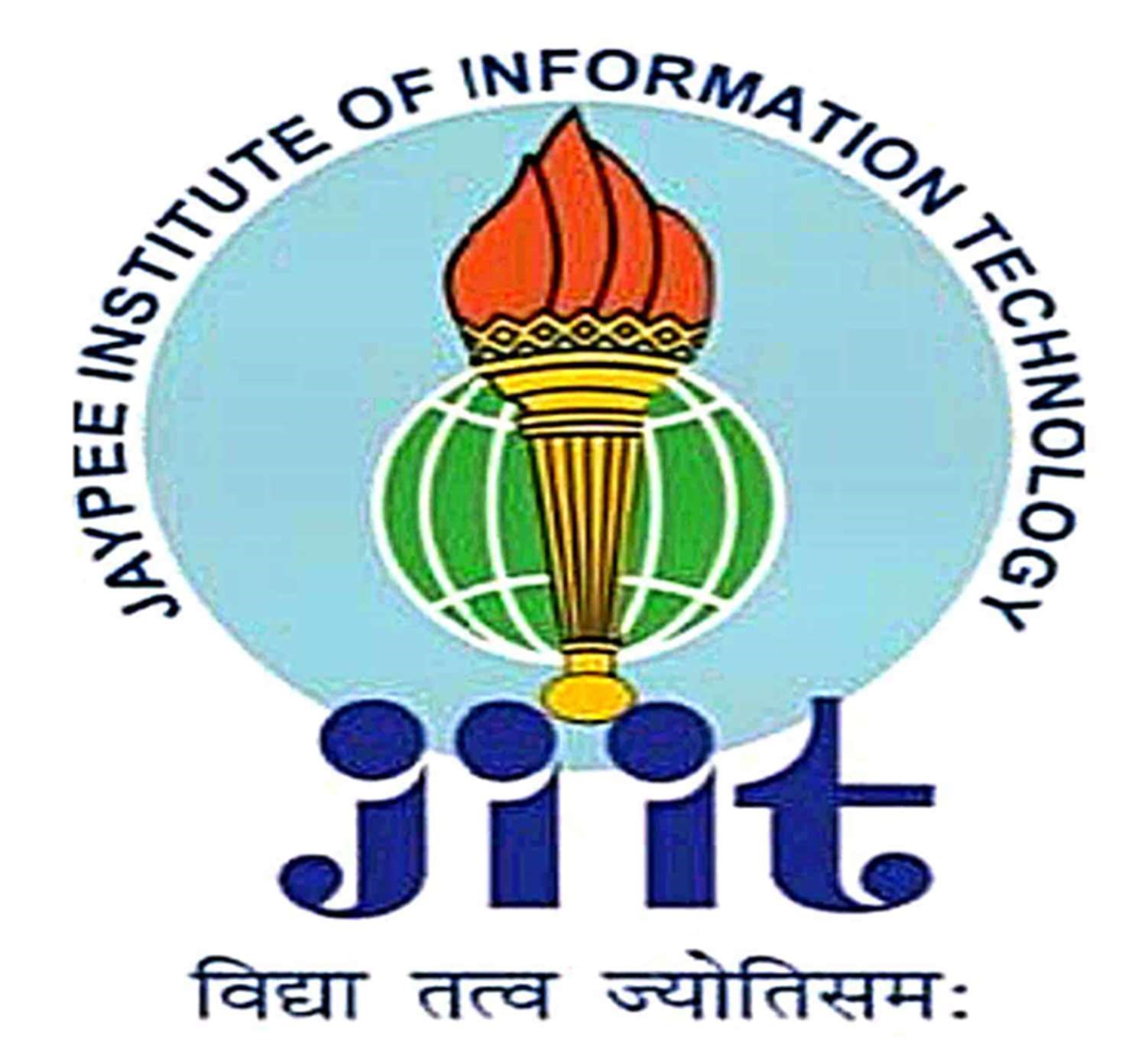
JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

A-10, Sector-62, Noida-201 309, Uttar Pradesh, India.

Project Report: Query Scheduler with

Round Robin Scheduling



Prepared By :

Ananya Mittal (21103305)

Manya Jain (21103313)

Mohmmad Yusuf (21103314)

Vaishnavi Agrawal (21103320)

Batch : B9

Query Scheduler with Round Robin Scheduling

* Abstract Of The Project

The "Query Scheduler with Round Robin Scheduling" is a software application designed to efficiently manage and process queries from two different user groups: Faculty and Students. The program employs the Round Robin scheduling algorithm to ensure fair processing of queries, streamlining query handling, and providing valuable insights into query performance. The objectives include automating query handling, ensuring fairness and efficiency, calculating essential performance metrics, and presenting summary statistics. Key features comprise a user-friendly interface, input validation, query sorting, merging of Faculty and Student queries, Round Robin scheduling, performance metric calculations, and summary statistics display. The technology stack includes C programming language, data structures using structures, Quick Sort for sorting, a command-line interface, and GCC (GNU Compiler Collection) for compilation. Implementation involves user input collection, query details provision (type, query ID, arrival time, and burst time), input validation, sorting using Quick Sort, query merging with priority, Round Robin scheduling, and summary statistics computation. Benefits encompass efficient query handling, automation, insightful statistics, and time savings. Future enhancements may include user authentication, database integration, real-time monitoring, and visualization. In conclusion, the "Query Scheduler with Round Robin Scheduling" is a valuable tool for query-intensive organizations, offering automation, fairness, and performance insights.

* Scope Of The Project

## **Functional Requirements**

1. **Query Input:** Users can input details for Faculty and Student queries, including type (Faculty or Student), query ID, arrival time, and burst time.

2. **Query Validation:** The system validates input data to ensure it adheres to specific constraints and correctness.

3. **Query Sorting:** Queries are sorted based on their arrival times using QuickSort, ensuring a fair execution order.

4. **Query Merging:** Faculty and Student queries are merged into a single queue with priority given to Faculty queries.

5. **Round Robin Scheduling:** Implement Round Robin scheduling to simulate query execution, ensuring fairness and efficiency.

6. **Performance Metrics:** Calculate waiting time, turnaround time, and completion time for each query.

7. **Summary Statistics:** Display summary statistics, including total time spent handling queries, average turnaround time, and average waiting time.

8. **User Interface:** Develop a user-friendly command-line interface for input and display of results.

* 1. **Non-functional Requirements**

1. **Efficiency:** Ensure efficient query handling and execution.

2. **User-friendliness:** Design a user-friendly interface for ease of use.

3. **Performance:** Optimize performance to minimize query execution times.

4. **Reliability:** The system should operate reliably without unexpected crashes.

5. **Scalability:** Handle a significant number of queries gracefully.

6. **Security:** Implement measures to protect against unauthorized access and data breaches.

* 1. **Description of the Modules of the Project**

1. **User Input Module:** Handles user input collection for query details.

2. **Query Validation Module:** Validates user input to ensure data correctness.

3. **Query Sorting Module:** Implements QuickSort for sorting queries based on arrival times.

4. **Query Merging Module:** Merges Faculty and Student queries with Faculty query priority.

5. **Round Robin Scheduling Module:** Executes queries using the Round Robin scheduling algorithm.

6. **Performance Metrics Module:** Calculates waiting time, turnaround time, and completion time.

7. **Summary Statistics Module:** Computes and displays summary statistics.

8. **User Interface Module:** Develops a user-friendly command-line interface for interaction.

To automate the handling of Faculty and Student queries.

* + To ensure that queries are processed fairly and efficiently.
  + To calculate waiting time, turnaround time, and completion time for each query.
  + To provide a summary of the execution, including average statistics.
* Features
  + User-friendly interface for inputting query details.
  + Validation checks to ensure correct input.
  + Sorting of queries based on their arrival times.
  + Merging of Faculty and Student queries with priority given to Faculty.
  + Round Robin scheduling for query execution.
  + Calculation of waiting time, turnaround time, and completion time.
  + Display of summary statistics, including total time spent handling queries, average turnaround time, and average waiting time.
* Technology Stack
  + Programming Language: C
  + Data Structures: Structures
  + Algorithms: QuickSort for sorting
  + User Interface: Command-line interface

Compilation: GCC (GNU Compiler Collection)

* Design Of The Project

The project will follow a modular design, with each module responsible for specific functionality. User inputs will be collected and validated, followed by sorting, merging, Round Robin scheduling, performance metric calculations, and summary statistics generation. The system's architecture will be structured to ensure efficiency, reliability, and scalability.

* Implementation
  + The program takes user inputs for the number of queries and time quantum.
  + Users provide details for each query, including type (Faculty or Student), query ID, arrival time, and burst time.
  + Input validation ensures that the data adheres to specific constraints.
  + Queries are sorted based on arrival times using QuickSort for both Faculty and Student queries.
  + Sorted queries are merged into a single queue with Faculty queries given priority.
  + Round Robin scheduling simulates the execution of queries, tracking waiting times and completion times.
  + The program calculates and displays summary statistics.
  + **Structure Definitions:** The code defines a C structure named `Query` to store query attributes, such as Query ID, Arrival Time, Burst Time, Completion Time, and Total Time. It also initializes arrays to store Faculty, Student, and merged queries (`Mix`).
  + **Global Variables:** Key variables such as `TotalQueries`, `TimeQuantum`, and counters are initialized globally to manage the query handling process.
  + **Input Validation:** The `Inputs For Process` function collects user inputs for the number of queries, time quantum, and query details (Query Type, Query ID, Arrival Time, Burst Time). It includes input validation to ensure data correctness and adherence to constraints.
  + **Query Sorting:** Queries from Faculty and Student groups are sorted separately based on their arrival times using the Quick Sort algorithm. The sorted queries are then merged into a single queue (`Mix`), with Faculty queries given priority.
  + **Round Robin Scheduling:** The `Round Robin` function simulates query execution using the Round Robin scheduling algorithm. It calculates waiting times, turnaround times, and completion times for each query. The results are printed in a tabular format, including Query ID, Arrival Time, Burst Time, Waiting Time, Turnaround Time, and Completion Time.
  + **Maximum Completion Time:** The `Max CT` function calculates the maximum completion time among all queries.
  + **Printing Results:** The `Print Result` function summarizes the execution, displaying the total time spent handling queries, average turnaround time, and average waiting time.
  + **Main Function:** The main function orchestrates the entire program. It takes user inputs, sorts queries, merges them, applies Round Robin scheduling, and prints the results.
* Testing Details

The testing phase will involve the creation and execution of test cases for each module and the system as a whole. Test cases will cover various scenarios, including valid and invalid input, different query types, and a range of query arrival times and burst times. The testing process will ensure the system operates reliably and meets functional and non-functional requirements.

To create test cases for the provided code, you should cover various scenarios and inputs to ensure the correctness of the program. Here are some test cases to consider:

**Test Case 1: Basic Test Case**

- Input:

- Number of Queries: 2

- Time Quantum: 5

- Faculty Query 1:

- Query Type: 1 (Faculty)

- Query ID: F1

- Arrival Time: 1000

- Burst Time: 10

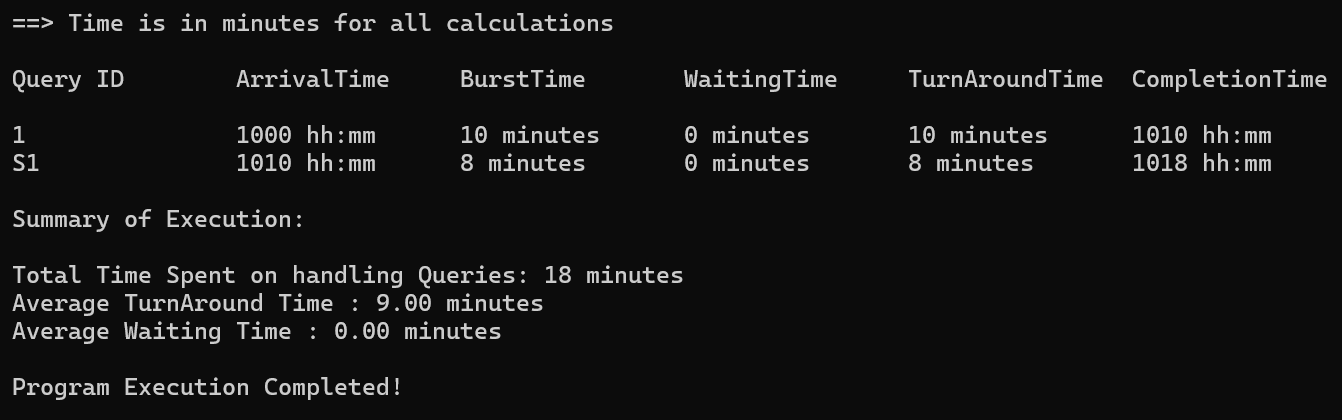
- Student Query 1:

- Query Type: 2 (Student)

- Query ID: S1

- Arrival Time: 1010

- Burst Time: 8



**Test Case 2: Query Merging**

- Input:

- Number of Queries: 3

- Time Quantum: 5

- Faculty Query 1:

- Query Type: 1 (Faculty)

- Query ID: F1

- Arrival Time: 1000

- Burst Time: 10

- Student Query 1:

- Query Type: 2 (Student)

- Query ID: S1

- Arrival Time: 1005

- Burst Time: 8

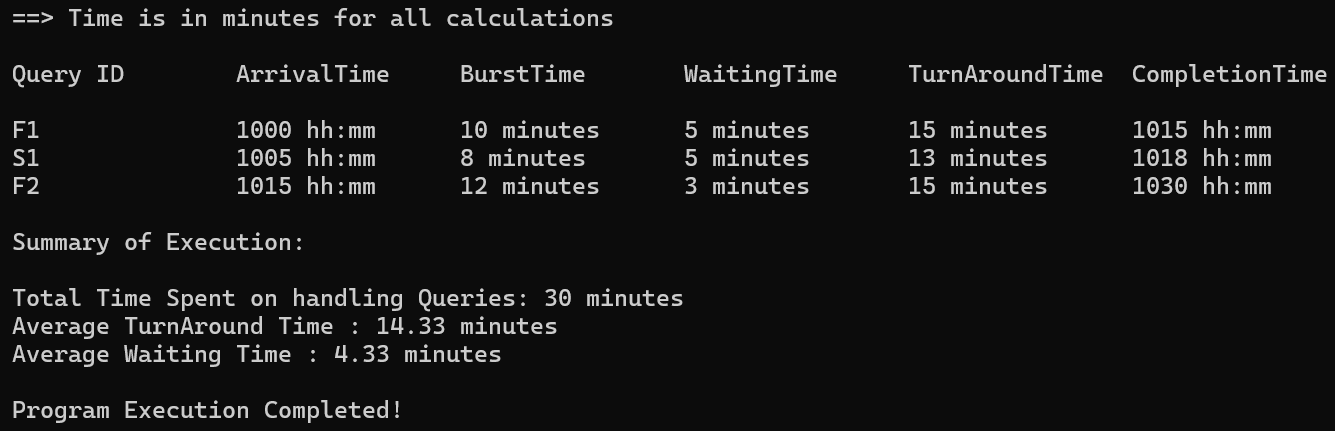
- Faculty Query 2:

- Query Type: 1 (Faculty)

- Query ID: F2

- Arrival Time: 1015

- Burst Time: 12



**Test Case 3: Invalid Burst Time**

- Input:

- Number of Queries: 1

- Time Quantum: 5

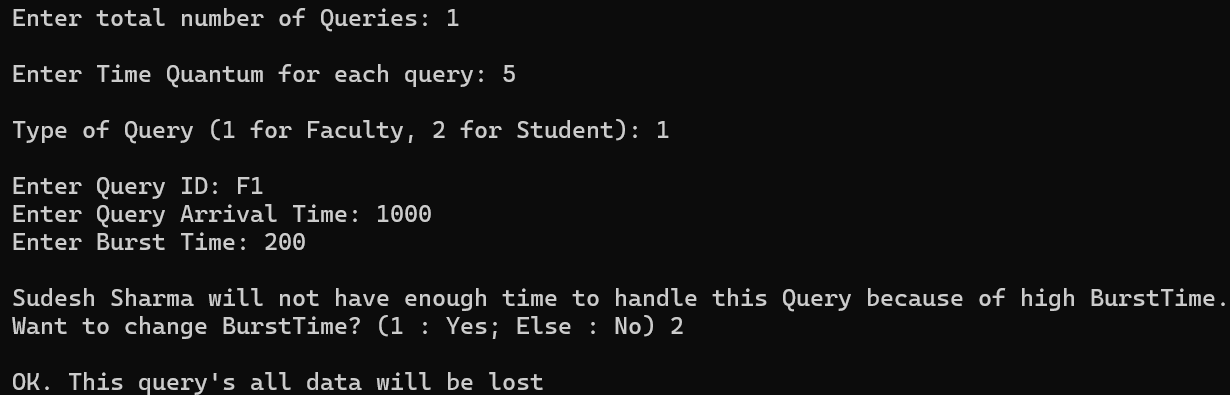
- Faculty Query 1:

- Query Type: 1 (Faculty)

- Query ID: F1

- Arrival Time: 1000

- Burst Time: 200



**Test Case 4: No Faculty Queries**

- Input:

- Number of Queries: 2

- Time Quantum: 5

- Student Query 1:

- Query Type: 2 (Student)

- Query ID: S1

- Arrival Time: 1005

- Burst Time: 8

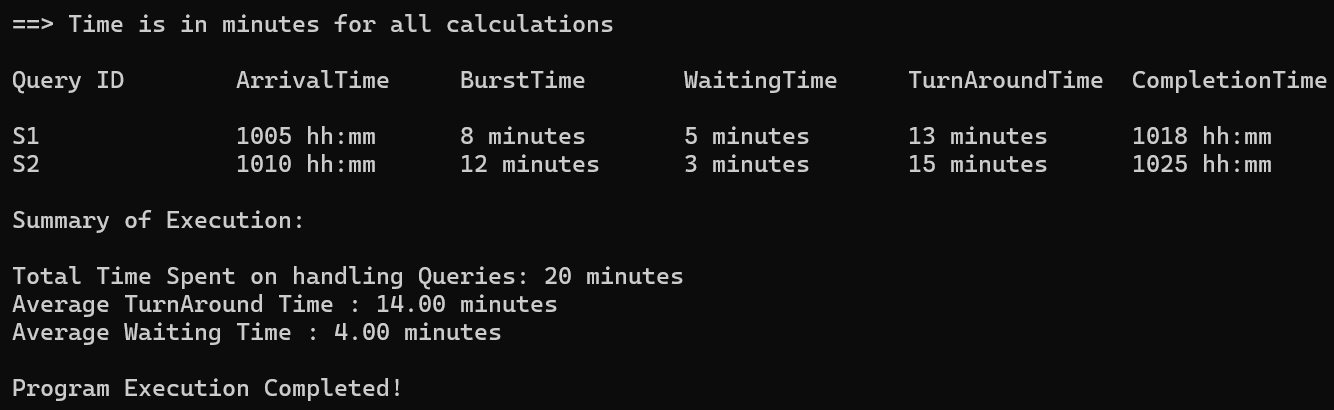
- Student Query 2:

- Query Type: 2 (Student)

- Query ID: S2

- Arrival Time: 1010

- Burst Time: 12

****

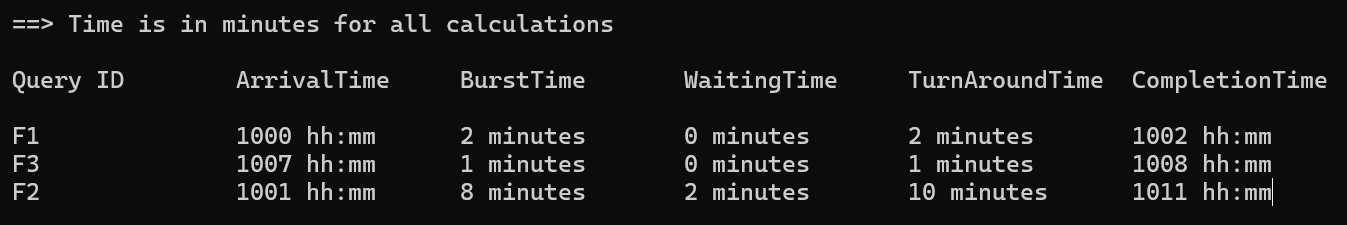
**Test Case 5: Large Number of Queries**

- Input:

- Number of Queries: 10

- Time Quantum: 5

- Create a mix of Faculty and Student queries with varying arrival times and burst times.

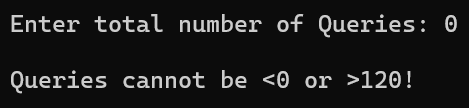


**Test Case 6: Edge Case**

- Input:

- Number of Queries: 0

- Time Quantum: 5



* Benefits
  + Efficient query handling: Queries are processed in a fair and efficient manner.
  + Automation: Manual query handling is eliminated, reducing errors.
  + Insightful statistics: Users can gain insights into query execution performance.
  + Time savings: The system saves time in managing and processing queries.
* Future Enhancements
  + User authentication and access control.
  + Database integration for query storage and retrieval.
  + Real-time monitoring of query status.
  + Visualization of query execution using graphs or charts.
* Conclusion

The "Query Scheduler with Round Robin Scheduling" is a valuable tool for organizations dealing with a large number of queries from Faculty and Students. It automates the query handling process, ensures fair execution, and provides valuable insights into query performance. This system can be further enhanced and customized to meet specific organizational requirements.

* References

1. Tanenbaum, A. S., & Bos, H. (2014). \*\*Modern Operating Systems\*\*. Pearson.

2. Kernighan, B. W., & Ritchie, D. M. (1988). \*\*The C Programming Language\*\*. Prentice Hall.

3. Garg, R. K. (2007). \*\*Operating System Concepts\*\*. Galgotia Publications.

4. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). \*\*Database System Concepts\*\*. McGraw-Hill Education.

5. QuickSort Algorithm Reference: [https://en.wikipedia.org/wiki/Quicksort](https://en.wikipedia.org/wiki/Quicksort)

6. GCC (GNU Compiler Collection) Documentation: [https://gcc.gnu.org/onlinedocs/](https://gcc.gnu.org/onlinedocs/)