**FIRST COME FIRST SERVE (FCFS) CPU SCHEDULING ALGORITHM**

**A Comprehensive Implementation with Web Interface**

PROJECT SYNOPSIS

**Submitted to:**  
Mr. Prabodh Kumar Sharma  
Assistant Professor  
Department of Computer Applications  
Maharaja Surajmal Institute

**Submitted By:**  
Pranjal Nath Goswami (32101360020) - Project Lead & Backend Developer  
Anjali Singh (32101360005) - Frontend Developer  
Aryan Gupta (32101360008) - Algorithm Implementation  
Deepak Kumar (32101360015) - Testing & Documentation

**Academic Year:** 2023-2024  
**Semester:** VIII  
**Course Code:** BCA404  
**Department:** Bachelor of Computer Applications

**DECLARATION**

We hereby declare that this Project Synopsis titled **"FIRST COME FIRST SERVE (FCFS) CPU SCHEDULING ALGORITHM: A Comprehensive Implementation with Web Interface"** submitted to Maharaja Surajmal Institute is a record of original work done by us under the guidance of Mr. Prabodh Kumar Sharma, Assistant Professor, and this project work has not been submitted elsewhere for any degree or diploma.

Team Members:

1. Pranjal Nath Goswami (32101360020)
2. Anjali Singh (32101360005)
3. Aryan Gupta (32101360008)
4. Deepak Kumar (32101360015)

Date: April 26, 2024  
Place: New Delhi

**ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to our project guide, Mr. Prabodh Kumar Sharma, Assistant Professor, for providing invaluable guidance, comments, and suggestions throughout the course of this project. We are extremely thankful for the consistent support and encouragement that has greatly contributed to the successful planning of this project.

We would also like to thank the Head of Department and all faculty members of the Department of Computer Applications for their continuous support and facilities provided for our project work.

Finally, we extend our thanks to our parents, friends, and all those who directly or indirectly helped us in the preparation of this project synopsis.

Team Members:

1. Pranjal Nath Goswami
2. Anjali Singh
3. Aryan Gupta
4. Deepak Kumar

**TABLE OF CONTENTS**

1. Introduction
2. Problem Statement
3. Objectives
4. Project Scope
5. Literature Review
6. Methodology
7. System Requirements
8. System Architecture
9. Implementation Plan
10. Expected Outcomes
11. Project Timeline
12. Risk Assessment
13. Budget
14. Conclusion
15. References

**1. INTRODUCTION**

The First Come First Serve (FCFS) CPU Scheduling Algorithm is one of the fundamental scheduling algorithms in operating systems. This project implements a comprehensive FCFS scheduling system with a modern web interface, providing both educational value and practical utility.

FCFS operates on a simple principle: processes are executed in the order they arrive in the ready queue. The first process to arrive is the first to be allocated CPU time, regardless of burst time or priority. While simple in concept, FCFS implementation requires careful consideration of process management, timing calculations, and user interface design.

This project bridges theoretical concepts with practical implementation, offering a platform for understanding CPU scheduling algorithms through interactive visualization and real-time calculations.

**1.1 Background of the Project**

CPU scheduling is a core function of operating systems, determining how processor time is allocated to processes. The First Come First Serve algorithm, being the simplest scheduling algorithm, serves as an excellent starting point for understanding scheduling concepts.

In academic settings, the theoretical understanding of scheduling algorithms is often separated from their practical implementation. This project aims to bridge this gap by providing a comprehensive implementation with visual representation, helping users gain both theoretical and practical knowledge.

**1.2 Significance of the Project**

This project holds significance in multiple domains:

* Educational Value: Provides a clear, interactive demonstration of FCFS scheduling principles
* Practical Implementation: Demonstrates real-world application of theoretical concepts
* Technical Skill Development: Combines multiple technologies (C++, Web Development, Algorithms)
* Interface Design: Showcases effective visualization of complex processes

**2. PROBLEM STATEMENT**

CPU scheduling is a critical aspect of operating system design, directly impacting system efficiency and user experience. Traditional approaches to teaching CPU scheduling algorithms often rely on static examples that fail to demonstrate the dynamic nature of process execution.

Specific problems addressed by this project include:

**2.1 Accessibility Gap**

Existing implementations often lack user-friendly interfaces, making them inaccessible to beginners and students learning operating system concepts.

**2.2 Visualization Deficiency**

Many implementations fail to provide clear visual representations of scheduling processes, making it difficult to understand the temporal aspects of process execution.

**2.3 Integration Challenges**

Few solutions integrate theory with practical, interactive implementations, creating a gap between conceptual understanding and applied knowledge.

**2.4 Educational Limitations**

Current educational tools often separate theory from implementation, making it difficult for learners to connect concepts with actual execution.

This project aims to bridge these gaps by creating an intuitive, web-based implementation of the FCFS algorithm that visualizes the scheduling process and provides detailed performance metrics.

**3. OBJECTIVES**

The project aims to achieve the following objectives:

**3.1 Primary Objectives**

* Implement the FCFS CPU scheduling algorithm in C++
* Develop a responsive web interface for interaction with the algorithm
* Provide real-time visualization of the scheduling process
* Calculate and display key performance metrics (completion time, turnaround time, waiting time)

**3.2 Secondary Objectives**

* Create an educational tool for understanding CPU scheduling
* Demonstrate client-server architecture in a web application
* Showcase modern web technologies integrated with C++ backend
* Implement effective error handling and input validation
* Provide detailed documentation for future enhancements

**3.3 Learning Objectives**

* Gain practical experience in algorithm implementation
* Develop skills in full-stack development
* Understand operating system concepts in depth
* Practice software development lifecycle management
* Enhance technical documentation skills

**4. PROJECT SCOPE**

**4.1 In Scope**

* Implementation of FCFS algorithm with all standard metrics
* Development of a web-based user interface
* Real-time calculation and visualization of scheduling metrics
* Support for dynamic process addition and management
* Comprehensive documentation and user guide
* Testing across modern web browsers

**4.2 Out of Scope**

* Implementation of other scheduling algorithms (SJF, Round Robin, etc.)
* Mobile application development
* Distributed system implementation
* Operating system integration
* Real hardware deployment considerations

**5. LITERATURE REVIEW**

**5.1 Theoretical Background**

CPU scheduling is a fundamental concept in operating systems, responsible for determining which process runs at a given time. The FCFS algorithm, being the simplest scheduling algorithm, serves as an excellent starting point for understanding scheduling principles.

**5.2 Key Research Papers**

1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). Wiley.
   * This seminal textbook provides comprehensive coverage of CPU scheduling algorithms including FCFS
   * Chapter 5 specifically discusses CPU scheduling with detailed examples and analysis
   * The book presents theoretical foundations and practical considerations for implementation
2. Tanenbaum, A. S., & Bos, H. (2014). Modern Operating Systems (4th ed.). Pearson.
   * Discusses implementation considerations for CPU scheduling algorithms
   * Offers comparative analysis between scheduling approaches
   * Presents case studies of real-world scheduling systems
3. Stallings, W. (2018). Operating Systems: Internals and Design Principles (9th ed.). Pearson.
   * Offers detailed analysis of performance metrics for scheduling algorithms
   * Discusses efficiency considerations and optimization techniques
   * Provides mathematical analysis of scheduling behavior

**5.3 Similar Implementations**

1. OS Scheduling Simulator (GitHub)
   * Web-based implementation of multiple scheduling algorithms
   * Lacks detailed performance metrics and real-time visualization
   * Limited to predefined scenarios with minimal user customization
2. Process Scheduler Visualizer (SourceForge)
   * Desktop application for visualizing scheduling algorithms
   * Limited to single-platform deployment, no web interface
   * Provides basic visualization but lacks comprehensive metrics
3. CPU Scheduling Algorithms (GitHub)
   * Command-line implementation of multiple scheduling algorithms
   * No visual interface or interactive elements
   * Requires technical knowledge to use and interpret results

**5.4 Gap Analysis**

Our implementation improves upon these existing solutions by combining:

* Web accessibility - making the tool available across platforms
* Real-time visualization - providing dynamic visual representation
* Detailed performance metrics - offering comprehensive analysis
* Modern user interface - ensuring usability for technical and non-technical users
* Comprehensive documentation - facilitating learning and extension

**6. METHODOLOGY**

**6.1 Development Approach**

The project will follow an incremental development approach, with four main phases:

**6.1.1 Analysis & Design Phase**

* Requirements gathering and specification
* System architecture design
* Interface design and prototyping
* Data structure and algorithm planning

**6.1.2 Implementation Phase**

* Backend development (C++ with Crow Framework)
* Frontend development (HTML, CSS, JavaScript)
* Integration of frontend and backend
* Implementation of FCFS algorithm

**6.1.3 Testing Phase**

* Unit testing of algorithm components
* Integration testing of frontend and backend
* User acceptance testing
* Performance testing

**6.1.4 Deployment & Documentation Phase**

* Deployment preparation
* User documentation creation
* Technical documentation finalization
* Project presentation preparation

**6.2 Tools and Technologies**

**6.2.1 Backend Development**

* C++ programming language
* Crow Framework for web server
* JSON for data exchange
* CMake for build management

**6.2.2 Frontend Development**

* HTML5 for structure
* CSS3 for styling
* JavaScript for interactivity
* Fetch API for server communication

**6.2.3 Version Control & Collaboration**

* Git for version control
* GitHub for repository hosting
* Markdown for documentation

**6.2.4 Testing Tools**

* Catch2 for C++ unit testing
* Chrome DevTools for frontend debugging
* Postman for API testing

**7. SYSTEM REQUIREMENTS**

**7.1 Hardware Requirements**

**Development Environment:**

* Processor: Intel Core i3 or equivalent (dual-core)
* RAM: 4GB or higher
* Disk Space: 1GB for development tools and project files
* Internet Connection: Required for package installation and repository access

**Deployment Environment:**

* Any system capable of running a modern web browser and C++ applications

**7.2 Software Requirements**

**Development Tools:**

* C++ Compiler (G++ or Visual C++ 14.0+)
* CMake 3.10 or higher
* Git 2.20 or higher
* Text Editor or IDE (Visual Studio Code, Visual Studio, or equivalent)
* Web Browser (Chrome, Firefox, or Edge)

**Runtime Dependencies:**

* C++ Runtime Libraries
* Modern Web Browser with JavaScript enabled
* Local network for server-client communication

**8. SYSTEM ARCHITECTURE**

The project follows a client-server architecture with three main components:

**8.1 Component Diagram**

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐

│ Frontend │ │ Backend │ │ FCFS │

│ (HTML/CSS/JS) │◄───►│ (C++/Crow) │◄───►│ Algorithm │

└─────────────────┘ └─────────────────┘ └─────────────────┘

**8.2 Data Flow**

1. User enters process details in the web interface
2. Frontend sends process data to backend via HTTP requests
3. Backend processes the data using the FCFS algorithm
4. Algorithm calculates completion, turnaround, and waiting times
5. Results are sent back to the frontend as JSON
6. Frontend displays the results with visualization

**8.3 Key Components**

**8.3.1 Frontend Layer**

* Process Input Form: Collects process details
* Visualization Component: Displays Gantt chart and timeline
* Results Display: Shows calculated metrics
* Error Handling: Validates input and displays messages

**8.3.2 Backend Layer**

* HTTP Server: Handles incoming requests
* API Endpoints: Define routes for different operations
* JSON Parser: Processes input and formats output
* CORS Handler: Manages cross-origin requests

**8.3.3 Algorithm Layer**

* Process Structure: Defines process attributes
* Sorting Function: Orders processes by arrival time
* Calculation Functions: Computes scheduling metrics
* Data Management: Handles process arrays and counters

**9. IMPLEMENTATION PLAN**

**9.1 Phase 1: Setup and Basic Structure (Week 1-2)**

* Set up development environment
* Create repository structure
* Implement basic C++ backend with Crow
* Create HTML/CSS structure for frontend
* Establish basic client-server communication

**9.2 Phase 2: Algorithm Implementation (Week 3-4)**

* Implement process data structures
* Develop FCFS algorithm logic
* Implement sorting and calculation functions
* Create test cases for algorithm validation
* Document algorithm implementation

**9.3 Phase 3: Frontend Development (Week 5-6)**

* Develop form for process input
* Implement results display section
* Create visualization components
* Add basic styling and responsiveness
* Implement input validation

**9.4 Phase 4: Integration and Enhancement (Week 7-8)**

* Connect frontend with backend API
* Implement error handling
* Enhance visualization with animations
* Optimize performance
* Refine user interface

**9.5 Phase 5: Testing and Documentation (Week 9-10)**

* Conduct comprehensive testing
* Fix bugs and issues
* Complete user documentation
* Finalize technical documentation
* Prepare presentation materials

**10. EXPECTED OUTCOMES**

**10.1 Deliverables**

1. Functioning web application with FCFS implementation
2. Source code repository with documentation
3. User guide for application usage
4. Technical documentation of the system
5. Project presentation and demonstration materials

**10.2 Metrics for Success**

1. Functionality: All features work as specified
2. Performance: Calculations complete within 2 seconds for up to 100 processes
3. Usability: 90% of test users can successfully create and visualize a schedule
4. Accuracy: Algorithm produces correct results for all test cases
5. Compatibility: Application works on Chrome, Firefox, and Edge browsers

**11. PROJECT TIMELINE**

|  |  |  |
| --- | --- | --- |
| **Week** | **Deliverable** | **Status** |
| 1-2 | Project Setup and Repository Creation | Planned |
| 3-4 | Backend Implementation | Planned |
| 5-6 | Frontend Implementation | Planned |
| 7 | Integration of Frontend and Backend | Planned |
| 8 | Testing and Bug Fixing | Planned |
| 9 | Documentation | Planned |
| 10 | Final Presentation Preparation | Planned |

**Major Milestones:**

* Week 2: Repository setup complete
* Week 4: FCFS algorithm implementation complete
* Week 6: Frontend implementation complete
* Week 8: Integrated system working
* Week 10: Project completion and presentation

**12. RISK ASSESSMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Probability** | **Impact** | **Mitigation Strategy** |
| Technical difficulties with C++ integration | Medium | High | Early prototyping, alternative approaches ready |
| Timeline delays | Medium | Medium | Buffer time included in schedule, prioritize core features |
| Cross-browser compatibility issues | High | Medium | Regular testing on multiple browsers, use standard features |
| Algorithm implementation errors | Low | High | Thorough testing with known examples, code reviews |
| Team coordination challenges | Medium | Medium | Regular meetings, clear task assignments, version control |

**13. BUDGET**

|  |  |  |
| --- | --- | --- |
| **Item** | **Cost Estimate** | **Justification** |
| Development Hardware | Existing Resources | Team members will use their own computers |
| Software Tools | $0 | All tools used are open-source or free for academic use |
| Server Hosting | $0 | Local development server will be used |
| Documentation Materials | $50 | Printing and binding of final documentation |
| Miscellaneous | $50 | Unexpected expenses |
| **Total** | **$100** |  |

**14. CONCLUSION**

The FCFS CPU Scheduling Algorithm implementation project offers both academic value and practical application. By creating an interactive web interface that visualizes the scheduling process, we bridge the gap between theoretical understanding and practical implementation.

This project will demonstrate not only the technical implementation of a fundamental operating system concept but also showcase modern web development practices and client-server architecture. The resulting application will serve as both an educational tool and a demonstration of the team's technical capabilities.

The emphasis on user experience, visual representation, and accurate algorithm implementation will create a complete solution that addresses the identified problems in existing implementations. The project's success will be measured by its functionality, performance, usability, and accuracy.

**15. REFERENCES**

1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). Wiley.
2. Tanenbaum, A. S., & Bos, H. (2014). Modern Operating Systems (4th ed.). Pearson.
3. Stallings, W. (2018). Operating Systems: Internals and Design Principles (9th ed.). Pearson.
4. MDN Web Docs. (2023). Introduction to the DOM. Mozilla. https://developer.mozilla.org/en-US/docs/Web/API/Document\_Object\_Model/Introduction
5. Crow Documentation. (2023). C++ Microframework for Web. https://crowcpp.org/
6. Dahl, R. (2009). Node.js. https://nodejs.org/
7. GitHub. (2023). Git Documentation. https://git-scm.com/doc