A PROJECT REPORT ON

First Come First Served

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF COMPUTER APPLICATION

Submitted by:

Student Name 1 - Pranjal Nath Goswami	**University Roll No. 2371289**
Student Name 2 - Diya Bisht	**University Roll No.2371101**
Student Name 3 - Karan Singh	**University Roll No.2371171**
Student Name4 – Pradeep Singh bora	**University Roll No.2371283**

Under the Guidance of

Dr. Mukesh Joshi

PBL faculty

Project Team ID: Group 44

School of Computing

Graphic Era Hill University, Bhimtal, Uttarakhand

June-2025

CANDIDATE’S DECLARATION

We hereby certify that the work which is being presented in the Synopsis entitled **“FCFS Scheduling project”** in partial fulfilment of the requirements for the award of the Degree of Bachelor Computer Application of the Graphic Era Hill University, Bhimtal shall be carried out by the undersigned under the supervision of **Dr. Mukesh Joshi, PBL faculty**, School of Computing, Graphic Era Hill University, Bhimtal.

Name1-Pranjal Nath Goswami	University Roll no1-2371289	signature
Name2- Diya Bisht	University Roll no2-2371101	signature
Name3-Karan Singh	University Rollno3-2371171	signature
Name4- Pradeep Singh bora	University Roll no4-2371283	signature

CERTIFICATE

The project report entitled “ FCFS scheduling ” being submitted by Pranjal Nath Goswami(2371289), Diya Bisht (2371101), Karan Singh(2371171) and Pradeep Singh bora (2371283) of BCA to Graphic Era Hill University Bhimtal Campus for the award of Bonafide work carried out by them. They have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.

Dr. Mukesh Joshi **Dr. Sandeep Kumar Budhani**

(Project Guide) **(Head, SOC)**

TABLE OF CONTENTS

First Come First Server (FCFS) CPU Scheduling

Team Members:

1. Pranjal Nath Goswami (University Roll No: 2371289)
   - Role: Project Lead, Backend Development & Documentation
   - Responsibilities:
     * Overall project coordination
     * Crow framework implementation
     * Server-side logic
     * Documentation

2. Karan Singh (University Roll No: 2371171)
   - Role: Frontend Development
   - Responsibilities:
     * User interface design
     * HTML/CSS implementation
     * JavaScript functionality

3. Diya Bisht (University Roll No: 2371101)
   - Role: Algorithm Implementation
   - Responsibilities:
     * FCFS algorithm coding
     * Time calculations
     * Process management

4. Pradeep Singh Bora (University Roll No: 2371283)
   - Role: Testing
   - Responsibilities:
     * Test case development
     * Quality assurance

University: Graphic Era Hill University (GEHU), Bhimtal
Department: School of Computing
Course: Bachelor’s in computer application (BCA)
Semester: 4th (2nd Year)
Subject: Software Engineering
Project Group No.: 44
Project Faculty: Dr. Mukesh Joshi

PROJECT OVERVIEW

- **Objective:**

  - Implementation of FCFS CPU scheduling algorithm with interactive web interface

- **Technology Stack:**

  - Backend: C++ with Crow Framework
  - Frontend: HTML, CSS, JavaScript

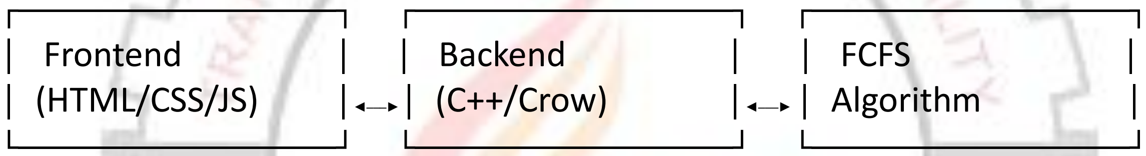
- **Purpose:**

  - Demonstrate CPU scheduling concepts through interactive visualization

SYSTEM ARCHITECTURE

DIAGRAM: Simple flow chart showing:

[Frontend Layer (HTML/CSS/JS)]
    ↓ HTTP Requests
[Backend Layer (Crow Server)]
    ↓ Algorithm Processing
[Process Management & Calculations]



• Data Flow:

- User inputs processes via web interface
- Data sent to C++ backend via HTTP
- Algorithm calculates results
- Results returned to frontend for display

BACKEND IMPLEMENTATION

• Process Structure:

```
struct Process {
    int pid;              // Process ID
    int arrival_time;     // When process arrives
    int burst_time;       // How long it needs to run
    int completion_time;  // When it finishes
    int turnaround_time;  // Total time taken
    int waiting_time;     // Time spent waiting
};
```

• Data Storage:

- Array of processes (max 10 processes)
- Each process contains 6 key attributes

FCFS Algorithm Implementation

• Sorting Processes:

```
// Sort processes by arrival time (bubble sort)
for(int i = 0; i < n-1; i++) {
    for(int j = 0; j < n-i-1; j++) {
        if(proc[j].arrival_time > proc[j+1].arrival_time) {
            Process temp = proc[j];
            proc[j] = proc[j+1];
            proc[j+1] = temp;
        }
    }
}
```

• Calculating Completion Time:

```
// Calculate completion times
proc[0].completion_time = proc[0].arrival_time + proc[0].burst_time;
for (int i = 1; i < n; i++) {
    if (proc[i].arrival_time > proc[i-1].completion_time) {
        proc[i].completion_time = proc[i].arrival_time +
proc[i].burst_time;
    } else {
        proc[i].completion_time = proc[i-1].completion_time +
proc[i].burst_time;
    }
}
```

TIME CALCULATIONS

• Turnaround Time:

```
// Calculate turnaround time
void findTurnaroundTime(Process proc[], int n) {
   for (int i = 0; i < n; i++) {
      proc[i].turnaround_time = proc[i].completion_time -
proc[i].arrival_time;
   }
}
```

• Waiting Time:

```
// Calculate waiting time
void findWaitingTime(Process proc[], int n) {
   for (int i = 0; i < n; i++) {
      proc[i].waiting_time = proc[i].turnaround_time -
proc[i].burst_time;
   }
}
```

FRONTEND IMPLEMENTATION

• HTML Structure:

```
<form id="processForm">
  <div class="form-group">
    <label>Process ID:</label>
    <input type="number" id="pid" required>
  </div>
  <div class="form-group">
    <label>Arrival Time:</label>
    <input type="number" id="arrival" required>
  </div>
  <div class="form-group">
    <label>Burst Time:</label>
    <input type="number" id="burst" required>
  </div>
  <button type="submit">Add Process</button>
</form>
```

JAVASCRIPT FUNCTIONALITY

• Process Management:

```
// Store processes
let processes = [];

// Add process to list
function addProcess() {
    const pid = document.getElementById('pid').value;
    const arrival = document.getElementById('arrival').value;
    const burst = document.getElementById('burst').value;

    processes.push({
        pid: parseInt(pid),
        arrival: parseInt(arrival),
        burst: parseInt(burst)
    });

    updateProcessList();
}
```

API INTEGRATION

• Server Communication:

```
async function calculateFCFS() {
    try {
        const response = await fetch('http://localhost:5000/fcfs', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ processes: processes })
        });
        const results = await response.json();
        displayResults(results);
    } catch (error) {
        console.error('Error:', error);
        alert('Error calculating FCFS');
    }
}
```

CROW FRAMEWORK IMPLEMENTATION

• Server Setup:

```
// Include Crow header
#include "crow.h"

// Initialize Crow application
crow::SimpleApp app;

// Define routes
CROW_ROUTE(app, "/")([](){
   return "FCFS Scheduler";
});

// FCFS calculation endpoint
CROW_ROUTE(app, "/fcfs").methods("POST"_method)
([](const crow::request& req){
   auto json = crow::json::load(req.body);
   // Process FCFS calculation
   return crow::response(result);
});

// Start server
app.port(5000).multithreaded().run();
```

TESTING & RESULTS

• Test Cases:

- Case 1: Sequential Arrival
  Process 1: Arrival = 0, Burst = 5
  Process 2: Arrival = 5, Burst = 3
  Process 3: Arrival = 8, Burst = 2

- Case 2: Overlapping Arrival
  Process 1: Arrival = 0, Burst = 5
  Process 2: Arrival = 2, Burst = 3
  Process 3: Arrival = 4, Burst = 2

• Results Analysis:

- Average Waiting Time
- Average Turnaround Time
- Visual Representation of Process Execution

CHALLENGES & SOLUTIONS

• Challenge 1: Process Synchronization

  - Solution: Implemented proper sorting algorithm

• Challenge 2: Real-time Updates

  - Solution: Used async/await in JavaScript

• Challenge 3: Error Handling

  - Solution: Implemented comprehensive error checks

• Challenge 4: Cross-Origin Requests

  - Solution: Added appropriate CORS headers

FUTURE SCOPE

• Planned Enhancements:

- Support for multiple scheduling algorithms
- Comparative performance visualization
- Gantt chart representation

• Technical Improvements:

- Database integration for process history
- User accounts and saved configurations
- Mobile-responsive design

LEARNING OUTCOMES

• Technical Skills:

- C++ programming expertise
- Web development fundamentals
- Algorithm design and analysis

• Soft Skills:

- Project planning and execution
- Technical documentation
- Problem-solving approach

THANK YOU

Contact Information:

Email: pranjalnathgoswami@gmail.com
GitHub: https://github.com/PranjalNG/OS-project
Questions & Discussion