A SYNOPSIS 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

Supervisor Name

Designation

Project Team ID: ID No.

School of Computing

Graphic Era Hill University, Bhimtal, Uttarakhand

March-2025

Graphic Era Hill University
BHIMTAL CAMPUS

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the Synopsis entitled **"Title of the 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 **Guide Name, Designation**, 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

The above mentioned students shall be working under the supervision of the undersigned on the **"Title of the project"**

Signature

Supervisor

Signature

Head of the Department

Internal Evaluation (By DPRC Committee)

Status of the Synopsis: Accepted / Rejected

Any Comments:

Name of the Committee Members: **Signature with Date**

Table of Contents

First Come First Serve (FCFS) CPU Scheduling Algorithm Implementation

Project Overview

A sophisticated implementation of the First Come First Serve (FCFS) CPU scheduling algorithm with a modern web interface. This project demonstrates efficient process management and real-time scheduling visualization.

Technology Stack

- **Backend**: C++ with Crow Framework
- **Frontend**: HTML5, CSS3, JavaScript
- **Build System**: CMake
- **Version Control**: Git
- **Documentation**: Markdown

Features

- **Interactive Process Management**
  - Dynamic process addition removal
  - Real-time scheduling visualization
  - Intuitive user interface
- **Advanced Scheduling Metrics**
  - Completion Time calculation
  - Turnaround Time analysis
  - Waiting Time optimization
- **Real-time Visualization**
  - Gantt chart representation
  - Process timeline display
  - Performance metrics dashboard
- **Error Handling & Validation**
  - Input validation
  - Process limit enforcement
  - Error message display

OBJECTIVES

The project aims to achieve the following objectives:

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)**

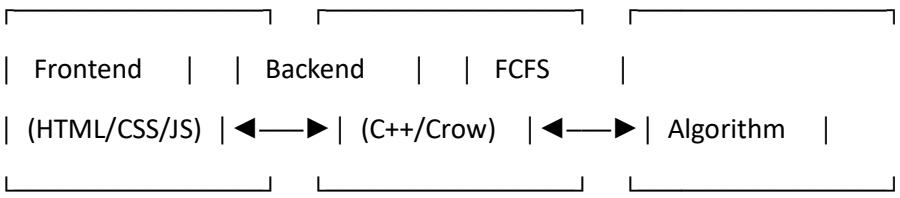
2 Secondary Objectives

- **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 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**

System Architecture



Getting Started

Prerequisites

- C++ compiler (C++17 or later)
- CMake (version 3.10 or later)
- Git
- Modern web browser

Installation

1. Clone the repository:

```
git clone https://github.com/PranjalNG/OS-project.git
cd OS-project
```

2. Build the project:

```
mkdir build
cd build
cmake ..
cmake --build .
```

3.Run the application:

```
./fcfs_scheduler .
```

Adding Processes

1. Enter process details:
    - Process ID (unique identifier)
    - Arrival Time (when process enters the system)
    - Burst Time (CPU time required)
2. Click "Add Process" to include in scheduling queue

Viewing Results

- **Gantt Chart**: Visual representation of process execution
- **Timeline**: Detailed process execution sequence
- **Metrics**: Performance analysis including:

  1.Complrtion Time

  2.Turnaround timed

  3. waiting time

Performance Metrics

Key Calculations

1. **Completion Time (CT**
    - Time when process finishes execution
    - Formula: CT = Previous Process CT

      +Current Process BT
2. **Turnaround Time (TAT)**
    - Total time from arrival to completion
    - Formula: TAT = CT - AT

3. **Waiting Time (WT)**
    - Time process waits in ready queue
    - Formula: WT = TAT - BT

Average Metrics

- Average Turnaround Time = Σ(TAT) / n
- Average Waiting Time = Σ(WT) / n

Technical Implementation

1.Backend Architecture struct

```
Process {
   int pid;
 int arrival_time;
  int burst_time;
  int completion_time;
int turnaround_time;
int waiting_time;
};
```

2.Frontend Components

- Process input form
- Real-time visualization
- Results display panel
- Error handling interface

3.**Testing**

- Unit tests for FCFS algorithm
- Integration tests for API endpoints
- UI/UX testing for web interface
- Performance benchmarking

SYSTEM REQUIREMENTS

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

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