Project Review-1
(2320030025 ,2320030038)

Project 1: Develop a Google users AI techniques to optimize the scheduling of the tasks in its data centres to improve efficiency and reduce energy consumption.

This repository contains a program that visualizes various CPU scheduling algorithms, including FCFS, SJF, SRTF, Priority (non-pre-emptive and pre-emptive), and Round Robin. The program provides a visual representation of how these algorithms work and how they schedule processes on a CPU.

Project Configuration:

- The Project section defines a single project within the solution, titled "CPU Scheduling Algorithms". It includes:
    - The project type GUID ({FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}), indicating it's a C# project.
    - The path to the project's file (CPU Scheduling Algorithms\CPU Scheduling Algorithms.csproj).
    - The project GUID ({1F088284-C921-4395-BE46-EE2EFFA6CF77}), which uniquely identifies this project within the solution.

Main Function Points

- Implements and visualizes various CPU scheduling algorithms
- Includes FCFS, SJF, SRTF, Priority (non-preemptive and pre-emptive), and Round Robin algorithms
- Provides a visual representation of the scheduling process

Technology Stack

- C#

License

MIT License

Bug Report Template:

This section provides a template for reporting bugs, ensuring consistent and clear communication.

Custom Issue Template:

This template allows you to define custom issues that might arise, giving flexibility for different types of reports.

Feature Request Template:

A template for suggesting new features or improvements, which helps gather structured feedback.

C# Project Configuration:

The C# project you've shared is likely related to the development of scheduling algorithms. This includes the

project’s entry point, references, and settings for debugging and release builds.

If you need assistance with specific parts of this content, like optimizing the C# code, organizing project documentation, or generating AI models for schedulingsummary for each section of the .gitignore file:

User-Specific Files:

User-specific files, such as *.rsuser, *.suo, and *.user, store individual preferences and settings in Visual Studio, like window layouts or recently opened files. These files are tailored to a specific user's development environment and are not necessary for other users. By ignoring these files, you prevent them from cluttering the repository and avoid potential conflicts when multiple developers work on the same project.

Build Results:

Build results, including directories like Debug/, Release/, bin/, and obj/, contain compiled binaries, object files, and other intermediate outputs generated during the build process. Since these files can be easily regenerated from the source code, they are excluded from version control. Ignoring these files keeps the repository clean and reduces its size, focusing version control on the actual source code rather than temporary build artifacts.

Visual Studio-Specific Files:

Visual Studio generates several files and folders, such as .vs/, *.VC.db, and *.sln.docstates, which store IDE-specific settings, cache files, and solution-level configurations. These

files are specific to the development environment and do not need to be tracked in the repository. Ignoring them ensures that the repository remains focused on the source code, avoiding unnecessary clutter from IDE-related files.

Test Results:

Test results, such as those found in [Tt]est[Rr]esult*/ and *.coverage, are outputs from running tests, including test logs and code coverage reports. These files are typically large, generated frequently, and are not essential for version control. By excluding them, you prevent the repository from becoming bloated with temporary data, ensuring that only meaningful code changes are tracked.

Auto-Generated Files:

Files like *_i.c, *_p.c, and *.meta are automatically generated by tools and compilers during the build process or by various IDEs. These files are not intended to be edited directly by developers and can be regenerated as needed. Ignoring these files prevents them from being mistakenly committed, keeping the repository free of redundant or unnecessary files.

Third-Party Tools and Add-Ons:

Files and directories related to third-party tools, such as _ReSharper*/, node_modules/, and .ntvs_analysis.dat, are often specific to the development environment and are not essential for the project itself. These files can vary greatly depending on the developer's setup and are excluded to prevent them from cluttering the repository and causing inconsistencies across different development environments.

Backup and Log Files:

Backup files and logs, including *.bak, *.log, and Backup*/, are often generated during development or as a result of various operations. These files are not necessary for version control and can unnecessarily increase the size of the repository. Ignoring them ensures that only relevant and meaningful files are tracked, keeping the repository clean and manageable.

Miscellaneous:

The .gitignore file also excludes various other files and folders, such as Python bytecode files (*.pyc), Nvidia Nsight configuration files (*.nvuser), and local history directories (.localhistory/). These files are specific to particular tools or environments and do not need to be tracked in the repository. Ignoring them helps maintain a focused and efficient version control system, avoiding unnecessary clutter from environment-specific or temporary files.

Common Practice

Developers often use .gitignore files to ensure that their repositories contain only the essential files needed to build and run the application, while excluding those that are either irrelevant or can be regenerated. This is especially important in collaborative environments where multiple developers may have different environments and tools.