Az-Eddine Elaamry: 10014270

Zakaria El Alaoui: 10017917

Aufbau des Projekts – CPU Task Scheduler Simulator

Das Projekt ist so aufgebaut, dass es klar zwischen **Simulationskern** und **Benutzeroberfläche (GUI)** trennt.

1. Simulation Core (Kernlogik)

Im Kern steckt die eigentliche **Scheduling-Simulation**.
Hier werden die Prozesse, Threads und die gewählten Algorithmen verarbeitet.
Wichtige Bestandteile:

- **Process**
  Repräsentiert einen einzelnen Prozess mit Attributen:
  - Ankunftszeit
  - Burst Time (benötigte CPU-Zeit)
  - Priorität
  - Berechnete Werte: Wartezeit, Durchlaufzeit, Fertigstellungszeit
- **Thread**
  Untereinheit eines Prozesses (z. B. mehrere Aufgaben innerhalb eines Programms).
  Je nach Modus behandelt der Scheduler Threads wie eigene Prozesse oder lässt sie im User-Mode vom Prozess selbst verwalten.
- **GanttEntry**
  Struktur, die speichert, **welcher Prozess/Thread zu welcher Zeit auf der CPU läuft**.
  Daraus wird das Gantt-Diagramm gebaut.
- **SchedulingAlgorithm (Interface)**
  Abstrakte Schnittstelle, die definiert, wie ein Algorithmus aufgebaut sein muss.
  Alle Algorithmen erben davon und implementieren ihre eigene Logik:
  - **FCFS** – First Come First Serve
  - **SJF** – Shortest Job First
  - **RR** – Round Robin (mit Quantum)
  - **RRP** – Priority Round Robin
- **Scheduler**
  Zentrale Steuereinheit:
  - Verwaltet die Liste der Prozesse und Threads
  - Ruft den ausgewählten Algorithmus auf

- Speichert die Ergebnisse im Gantt-Diagramm
- Unterstützt verschiedene Modi:
  - ProcessOnly
  - ThreadKernel
  - ThreadUser

2. API / Erweiterbarkeit

Die Trennung durch die SchedulingAlgorithm-Schnittstelle macht es sehr einfach, **eigene Algorithmen hinzuzufügen**.
Man muss nur eine neue Klasse erstellen, die von SchedulingAlgorithm erbt, und die Methode execute(...) implementieren.

3. GUI (Benutzeroberfläche)

Die Oberfläche ist mit **Qt** umgesetzt und besteht aus:

- **MainWindow**
  Hauptfenster mit allen Eingabeelementen (Arrival Time, Burst Time, Priority, Algorithmus-Auswahl etc.).
  Es enthält die Buttons *Insert Process*, *Start Simulation*, *Reset*.
- **Controller**
  Vermittler zwischen GUI und Simulationskern.
  - Empfängt Benutzereingaben aus dem MainWindow
  - Übergibt die Daten an den Scheduler
  - Holt Ergebnisse zurück und zeigt sie im Fenster an
- **GanttChartWidget**
  Spezial-Widget, das das **Gantt-Diagramm** zeichnet (also den Zeitplan als Balkendiagramm).

Fazit:

Der CPU Task Scheduler Simulator macht die abstrakten Konzepte des Prozess- und Thread-Schedulings greifbar. Mit Prozesstabelle, Statistik und Gantt-Diagramm lassen sich die Auswirkungen von Algorithmen wie FCFS, SJF, RR und Priority RR direkt vergleichen.
Die klare Trennung von Simulationskern und Benutzeroberfläche sorgt für Übersichtlichkeit, und dank des flexiblen API-Designs lassen sich weitere Strategien leicht integrieren. Damit ist das Projekt nicht nur ein nützliches Lernwerkzeug für Studierende, sondern auch eine solide Grundlage für Experimente und Erweiterungen im Bereich Betriebssysteme.