CPU SCHEDULING

A MOBILE APPLICATION TO VISUALIZE OS JOB/PROCESS SCHEDULING ALGORITHMS.

Developed By

Nisarg Dave [18BCP070]
Pallav Chauhan [18BCP072]
Neev Shah [18BCP067]
Hetvi Patel [18BCP039]
Paritosh Joshi [18BCP074]

Pandit Deendayal Petroleum University

Overview

CPU Scheduling is a mobile based application which can run and visualize the Job Scheduling Algorithms. Users can enter the details of the algorithm and visualize how the algorithm works with the help of the Visualization tab available.

Technical Specifications

Language: Dart
IDE: Android Studio

Getting your Windows Machine ready:

1) Installation of **Flutter**

1.1) System requirements:

To install and run Flutter, your development environment must meet these minimum requirements:

- Operating Systems: Windows 7 SP1 or later (64-bit), x86-64 based
- Disk Space: 1.32 GB (does not include disk space for IDE/tools).
- Tools: Flutter depends on these tools being available in your environment.
    - Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)
    - Git for Windows 2.x, with the Use Git from the Windows Command Prompt option.
      If Git for Windows is already installed, make sure you can run `git` commands from the command prompt or PowerShell.

1.2) Download **Flutter SDK:**

i. Download the following installation bundle to get the latest stable release of the Flutter SDK: flutter_windows_1.22.4-stable.zip
ii. Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (for example, `C:\src\flutter`).

1.3) Update your path

If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the `PATH` environment variable:

- From the Start search bar, enter ‘env’ and select **Edit environment variables for your account.**
- Under **User variables** check if there is an entry called **Path**:

- If the entry exists, append the full path to `flutter\bin` using `;` as a separator from existing values.
- If the entry doesn't exist, create a new user variable named `Path` with the full path to `flutter\bin` as its value.

You have to close and reopen any existing console windows for these changes to take effect.

1.4) Run **flutter doctor:**

From a console window that has the Flutter directory in the path (see above), run the following command to see if there are any platform dependencies you need to complete the setup:

```
C:\src\flutter>flutter doctor
```

This command checks your environment and displays a report of the status of your Flutter installation. Check the output carefully for other software you might need to install or further tasks to perform (shown in **bold** text).

For example:

[-] Android toolchain - develop for Android devices

- **Android SDK at D:\Android\sdk**

✗ Android SDK is missing command line tools; download from https://goo.gl/XxQghQ

- **Try re-installing or updating your Android SDK,**

visit https://flutter.dev/setup/#android-setup for detailed instructions.

1.5) Install **Android Studio**

i. Download and install Android Studio.
ii. Start Android Studio, and go through the 'Android Studio Setup Wizard'. This installs the latest Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools, which are required by Flutter when developing for Android.

1.6) Set up your **Android device**

To prepare to run and test your Flutter app on an Android device, you need an Android device running Android 4.1 (API level 16) or higher.

i. Enable **Developer options** and **USB debugging** on your device. Detailed instructions are available in the Android documentation.
ii. Windows-only: Install the Google USB Driver.
iii. Using a USB cable, plug your phone into your computer. If prompted on your device, authorize your computer to access your device.

iv. In the terminal, run the flutter devices command to verify that Flutter recognizes your connected Android device. By default, Flutter uses the version of the Android SDK where your adb tool is based. If you want Flutter to use a different installation of the Android SDK, you must set the ANDROID_SDK_ROOT environment variable to that installation directory.

1.7) Set up the **Android emulator:**

To prepare to run and test your Flutter app on the Android emulator, follow these steps:

i. Enable VM acceleration on your machine.
ii. Launch **Android Studio**, click the **AVD Manager** icon, and select **Create Virtual Device…**
    - In older versions of Android Studio, you should instead launch **Android Studio > Tools > Android > AVD Manager** and select **Create Virtual Device…**. (The Android submenu is only present when inside an Android project.)
    - If you do not have a project open, you can choose Configure > AVD Manager and select Create Virtual Device…
iii. Choose a device definition and select **Next**.
iv. Select one or more system images for the Android versions you want to emulate, and select **Next**. An *x86* or *x86_64* image is recommended.
v. Under Emulated Performance, select **Hardware - GLES 2.0** to enable hardware acceleration.
vi. Verify the AVD configuration is correct, and select **Finish**.
For details on the above steps, see Managing AVDs.
vii. In Android Virtual Device Manager, click **Run** in the toolbar. The emulator starts up and displays the default canvas for your selected OS version and device.

2) Set up an **editor**(using Android Studio is preferable)

2.1) Install **Android Studio:**

Download and Install Android Studio, version 3.0 or later

2.2) Install the **Flutter and Dart plugins:**

To install these:

i. Start Android Studio.
ii. Open plugin preferences (**Configure > Plugins** as of v3.6.3.0 or later).
iii. Select the Flutter plugin and click **Install**.
iv. Click **Yes** when prompted to install the Dart plugin.
v. Click **Restart** when prompted.

For more details and to solve some error you can visit to Flutter Installation

How to run?

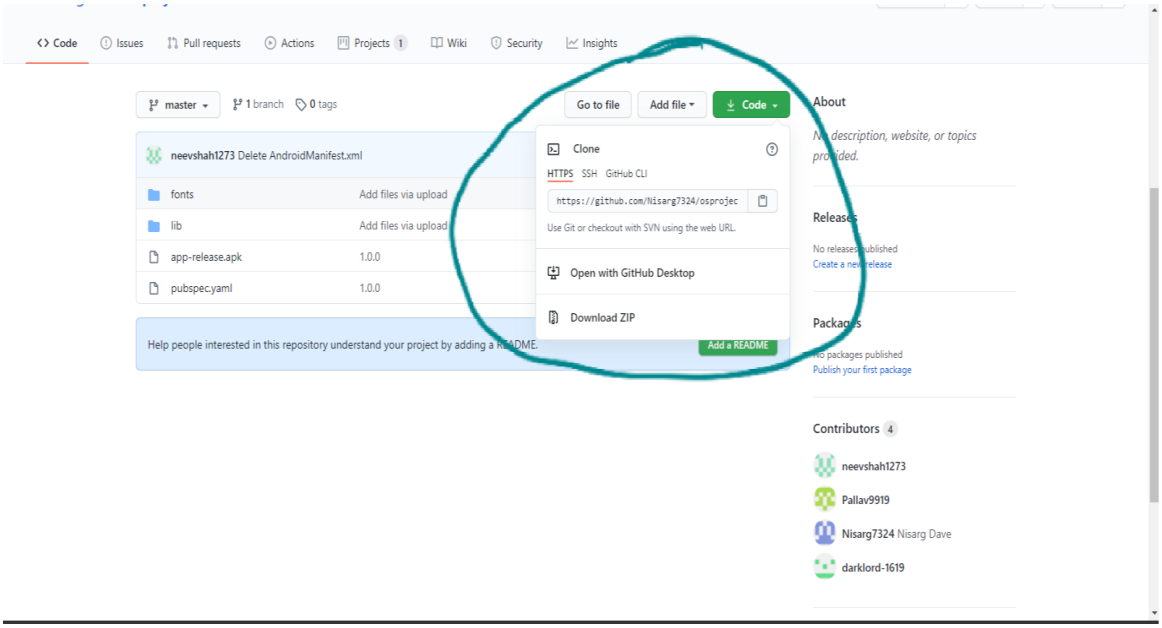
Method 1:

1. Download .apk file for CPU Scheduling
2. Install this .apk file in your Android Device and you can run CPU Scheduling application directly on you Android Device

Method 2:

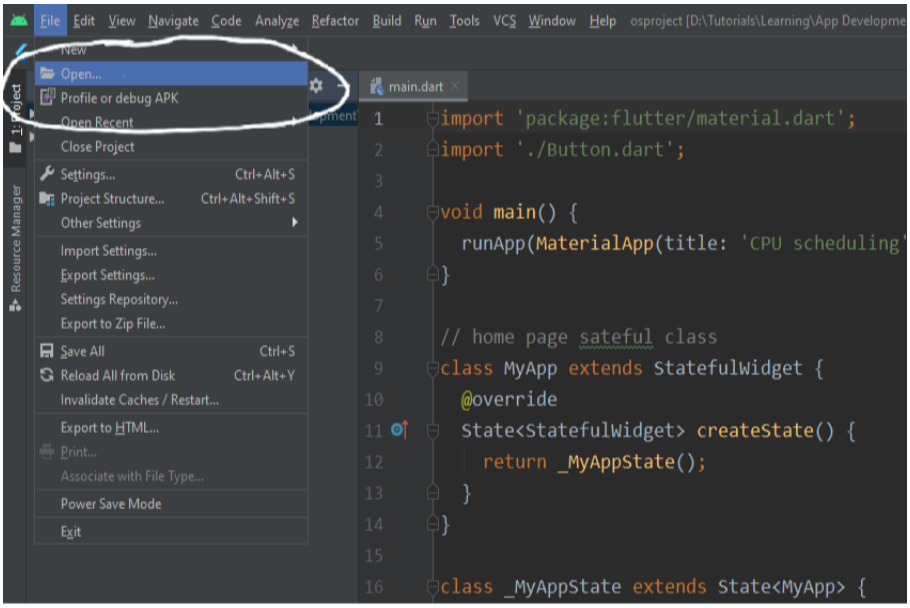
Follow the commands to run this application:

1. Go to GitHub to download main files for project
2. Click on **Code** as shown below

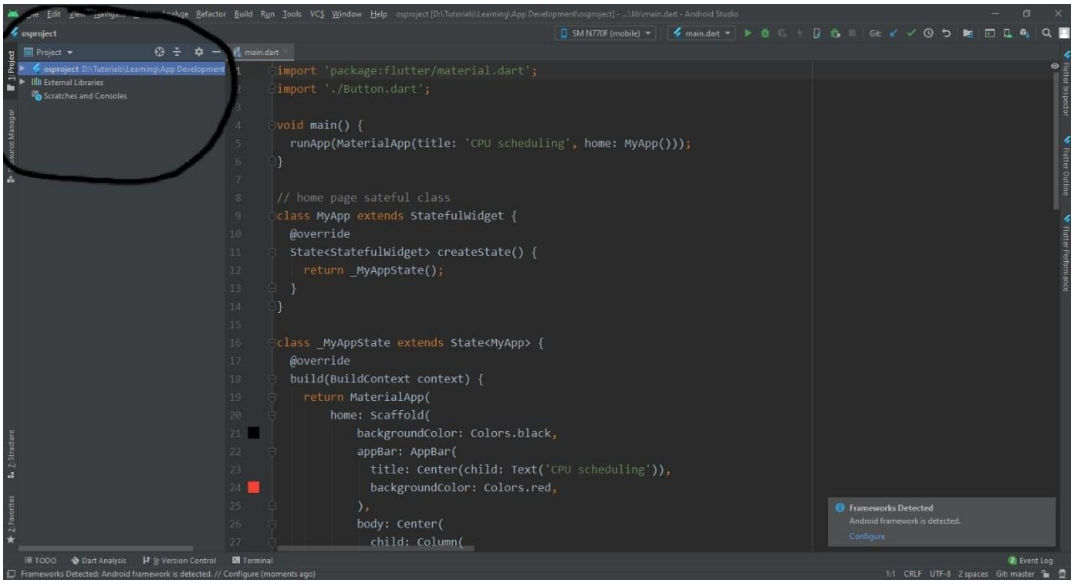


3. Click **Download ZIP** to download Zip folder for our project
4. **Extract** file contents of Zip folder to any desired location.
5. Now, open **terminal** to move to Flutter Project location using cd command
6. Once you are in project's directory use command

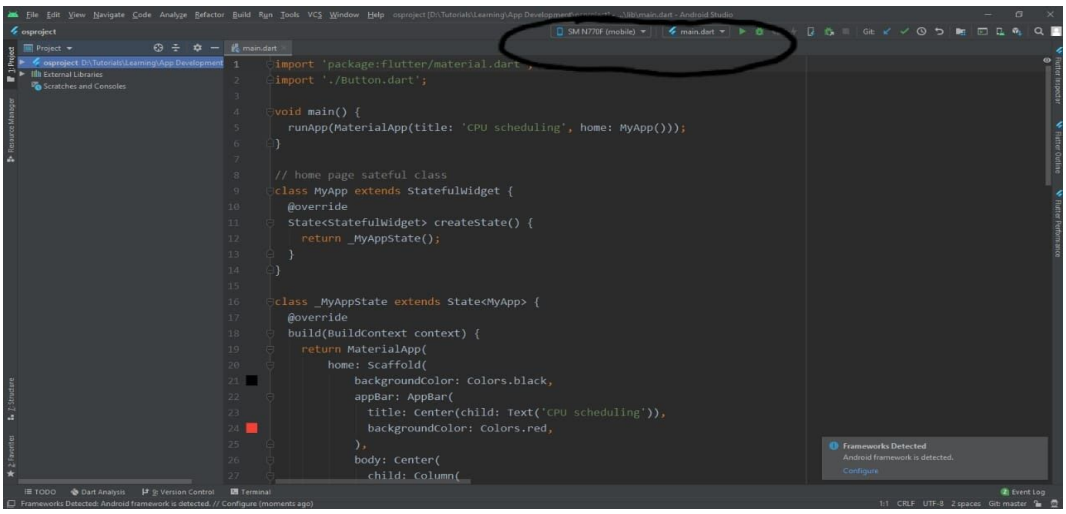
   $ **flutter pub get**
7. After completing all the above steps you can open Android Studio and open your Project
8. To open your project click **File > Open > Flutter Project**



9. Now, our project is completely loaded your machine
10. By clicking on **project resource** you can see all the files related to this project and using editor you can edit code as you wish



11. To run this project, check if your Android device is visible and the file selected is **main.dart** from **lib** folder(To changes if required)



12. Now click **play** and you can see a **Virtual Android device** running with our application **'CPU Scheduling'**

How to use it?

We have uploaded a working video of the project on Youtube which will give you all the information on how to use the CPU Scheduling application and how to take benefit of all the features provided.

Link of the video:

Even you can refer to the step listed below about how to use this application

1. Select the **algorithm** displayed on your screen
2. Use the **slider** on top-right corner to switch between inputs with/without IOBT
3. Enter the **values** of **Arrival Time**, **Burst Time** and also **I/O Burst Time** if selected
4. As you enter values, you can see result of **Completion Time**, **Turn Around Time**, **Waiting Time** change according to the inputs
5. To add new process click on **'Add Process'**
6. To delete last process added click on **'Delete Process'**
7. **'Average Waiting Time'** and **'Average Turn Around Time'** is also visible below
8. For **visualization** there are two options available
   i) Click on **Gantt Chart** to see the gantt chart of how all the above processes are getting processed. Gantt Chart includes **Process No.**, **Start Time** and **End Time**. On clicking on any cell of Gantt Chart you can also see **Ready Queue** and **Finished Queue** for that particular

cell. To know about the state of the process we have added a color scheme for better understanding.

a) **Green** :- After end time mentioned, this process has finished execution and will go to Terminated Queue.
b) **Grey** :- After the end time mentioned, this process will go to IOBT Queue.
c) **Red** :- After end time mentioned, this process is preempted and will go to Ready Queue. Also, the process has not yet finished execution.

ii) Click on **Visualization** to see exactly how the algorithm works.

a) On the left side you can see 4 different segments, **'Not Arrived'**, **'Ready'**, **'Running'**, **'Terminated'**, defining states of each particular process.
b) By clicking the **'+'** button as shown below you can **increase time**.
c) By clicking the **'-'** button as shown below you can **decrease time**.
d) Using these two operations you can increase or decrease time and **find the state** of any process at that **particular time interval.**

Future Work

Some more features can be implemented and added to make the application more better and user friendly. Here are those which can be a part of the application in future.

Tutorial

Instead of going through the entire tutorial to understand the working of our project we can add a tutorial section at the starting of application so, whosoever is interested in understanding the working of application or understand cpu scheduling algorithms implemented in this project.

Available to all

By creating an apk file we will like to make this application available to download directly via Play Store. This feature will be dedicated to all those who only want to use the final product of this project and are not interested in code written to create this application.