Code Review of The Software Project: Productivity Software: Study Outline

Course Title: Software Development Project
Course ID: CSE- 3106

Project By:

M.d. Ashiquzzman Rahad
Student ID: 210201
Jannatul Ferdous Nijhum
Student ID: 210239

Reviewed By:

Sk Miraz Rahman Ani
Student ID: 210211
Mohommad Ibnsina
Student ID: 200210

Submitted To:

Dr. Amit Kumar Mondal
Associate Professor
Computer Science & Engineering Discipline
Khulna University,
Khulna.

Introduction

This code review evaluates the productivity application: **Study Outline**. The review identifies areas for improvement, adherence to best practices, and suggestions for enhancing maintainability, security, and overall code quality. In this code review, bad smells of the code, architecture evaluation, modularity checking, condition statements of the code & other related sections are evaluated.

Code Smells

1. **Large or complex methods:**
   In terms of method size and complexity, the codebase generally maintains a balance, with few instances of excessively large or intricate methods that might pose readability challenges. On average, methods consist of around 10 lines of code, adhering closely to the standard size. The largest method, found in the pomodoro.py module, spans approximately 30 lines, notably in the update_timer function. Conversely, the syllabus.py module predominantly features smaller methods, contributing to a more streamlined and modular structure overall.

2. **Long parameter lists:**
   There is no method with long parameter lists.

3. **Excessive comments:**
   Inconsistencies in the usage of comments are apparent across various modules. While "goal.py" and "pomodoro.py" are inundated

with excessive comments, "syllabus.py" stands in stark contrast with not a single comment to be found.

4. **Duplicate code:**
   There is no duplicate code in the methods. Effective code reusability has been achieved across the majority of modules.

5. **Inconsistent naming conventions:**
   Naming conventions, in many instances, adhere to standardized practices. However, there are instances where variations exist, leading to diverse approaches in naming. For example: update_timer(), save_tasks(), etc.

6. **Incomplete error handling:**
   Errors are primarily managed within each module, with special attention paid to file operations to mitigate potential issues. These operations appear to be handled meticulously, minimizing the likelihood of encountering errors.

7. **Too many if/else statements:**
   In the pomodoro.py module, there are some excessive use of if/else statements. But in other modules, the usage of if/else statements are moderated.

8. **Poor use of inheritance:**
   In this project, there isn't a singular instance of inheritance misuse that's causing significant issues.

9. **Unnecessary dependencies:**
   There are various unnecessary external libraries and frameworks imported in various modules in this project which are not being used. For example, in pomodoro.py module -

```
from tkinter import *
from ttkbootstrap.constants import *
from ttkbootstrap import Style
import json
from ttkbootstrap.dialogs import Messagebox
```

   These libraries/ frameworks are not being used but still imported unnecessary

10. **Magic numbers or hard-coded values:**
   In most of the cases, there is no sign of hard coding or magic numbers. Instead, contents are introduced and used in the project.

Proposed Architecture Evaluation

The proposed architecture of the project is **"Layered Architecture"**. In the project, we can see the reflection of the proposed architecture. There are different layers or modules in the project.

There is a **main.py** module which has the main window with different frames for the application, which serves as the basic user interface layer. The detailed user interface is actually enclosed in each of the modules.

The **pomodoro.py, goal.py and the syllabus.py** modules serves as the main application functionality layer. These modules have their individual application functionalities included in them. The **study.py** module serves the application functionality of launching the application as a whole.

In each of the module, there are required file operations included which serve as the locally data storing layer.

So, we can say that the project reflects the proposed **Layered Architecture** more or less.

Modularity Check

The project comprises five distinct modules: main.py, pomodoro.py, goal.py, syllabus.py, and study.py. Each module serves a specific purpose within the project, contributing to its overall functionality and organization.

main.py module has the main window and frames of the user interface. **pomodoro.py** module serves the necessary features of the pomodoro timer. **goal.py** module serves the necessary features of the goal setter. **syllabus.py** module serves as the syllabus tracker features. **study.py** accumulates the modules and launch them all together.

If/else Condition to Switch statement

Python does not include a built-in switch case statement, consequently, there exists no alternative method to implement required conditions in the code besides using if/else statements.