Individual Report: Python Software Development Project

Name: Ishan

Student ID: 240013

Course: Computing

Module: ST4000CEM Programming and Algorithms

Project Title: Basic Movie Booking App using Tkinter

1. Methodology

Our group followed the **Agile methodology** to complete this project, using a simplified sprint-based approach. We divided the work into small tasks and held regular team meetings to check progress and resolve issues. Tools like GitHub Classroom enabled us to collaborate and manage code versions effectively. Task allocation was dynamic—members took ownership of tasks based on their strengths and availability. I was primarily responsible for the GUI logic and database integration, including booking flows and profile management.

2. Individual Contributions

I contributed primarily to the GUI development using **Tkinter**, and database operations using **SQLite**. I created the windows for movie selection, booking, and user profile management. I also implemented event-driven functions such as button click responses, data fetching from the database, and dynamic screen updates without object-oriented programming (OOP).

I integrated the booking form with the SQLite database, including insert/update queries and retrieval of available seats. The code was procedural and function-based due to the project's simplicity and our decision to avoid OOP. I also ensured user input validation (e.g., no empty fields,

invalid selections) and meaningful error messages through `try-except` blocks.

3. Tools and Technologies Used

- **Programming Language:** Python (without OOP)
- **GUI Framework:** Tkinter
- **Database:** SQLite
- **Version Control:** Git & GitHub Classroom
- **Testing Framework:** Python `unittest`
- **IDE/Editor:** Visual Studio Code

4. Advanced Skills Used

Despite the project not involving OOP, we applied advanced programming skills such as:

- **GUI Programming with Tkinter**, including window management, layout control, and event-driven logic.
- **Database Integration** using `sqlite3`, enabling data persistence.
- **Error Handling** with `try-except` to handle database failures or user input errors.
- **Version Control Workflow** using Git: branching, committing, pulling, and merging.
- **Unit Testing** of backend functions to check data operations and edge cases.

---

5. Reflection

This project helped me understand how to build a functional desktop GUI app with no object-oriented design. Initially, managing multiple Tkinter windows procedurally was challenging, but I gradually learned how to structure the code cleanly using functions and modular files. I also gained experience in integrating SQLite databases into real-time user flows.

Team collaboration via GitHub taught me essential practices like using commit messages effectively, resolving merge conflicts, and reviewing each other's code. In future iterations, I would recommend using OOP to make the code more scalable and maintainable. Additionally, migrating to frameworks like PyQt or adding file-based logs would enhance the project.

---

6. References

1. Python Software Foundation. (n.d.). *Tkinter — Python interface to Tcl/Tk*. https://docs.python.org/3/library/tkinter.html
2. Downey, A. (2015). *Think Python: How to Think Like a Computer Scientist* (2nd ed.). O'Reilly Media.
3. GitHub. (n.d.). *Git Documentation*. https://git-scm.com/doc
4. Python Software Foundation. (n.d.). *sqlite3 — DB-API 2.0 interface for SQLite databases*. https://docs.python.org/3/library/sqlite3.html
5. Coventry University (2021). *APA Referencing Guide*. https://libguides.coventry.ac.uk/apa