**IT5016**

**Assessment 2 software Project Part B**

**Requisition System – Software Development Lifecycle (SDLC)**

1. **Planning**

During this phase the software goals & purpose were defined. We knew the system had to be structured so that staff could submit requisition requests, see status of the approvals and managers could review and respond to those requests. The necessity of some statistical reporting was also recognized.

**(2) Requirement Analysis**

We analysed the features we needed: Staff should be able to insert their requisition and personal information. The system should compute total cost

Approvals must be based on set conditions (e.g., total below $500).

Managers should be able to respond to pending requests.

System must track and display statistics (approved, pending, rejected).

Information must be clearly displayed in a structured format.

**(3) Design**

During design, we settled on using classes in Python which made it easier to structure our code base to be modular and maintainable. Each of the parts of the system (staff input, approval process, statistics, etc.) was broken down into its own method. We were also going to create a class variable to track all requisitions and statistics releveant over the lifetime of the app (since it would only run for a while normally).

**(4) Implementation (Coding)**

This phase was the actual writing of the Python code: The Requisition System class was created. I used functions to divide the job clearly: staff\_info, requisition\_description, requisition\_approval, respond\_requisition, display\_requisition, requisition\_piece.

Added a test function to simulate submissions of multiple requisitions.

**(5) Testing**

To manually test the system, we ran the function test\_requisition\_system() and entered the various test cases: Requisitions with totals under and over $500. Pending items are responded to by managers.

Checking that status updates and stats matched the changes.

**(7) Deployment**

This software isn't fully functional, but can be executed easily in any Python environment, and thus may be useful in internal applications or education environments. Considerations for future deployment:

Porting to a GUI/web version. Implementing permanent storage (e.g., writing to file or database).

**(8) Maintenance**

Through the final review, areas of improvement were noted:

Dynamic Requisition ID. Username or password incorrect, Send the User Back.

Showing a list of availed items Improving users’ interface and experience.

Refactoring and updates will follow as the software evolves.

**Here is the conclusion**

After researching agile methodology and waterfall model methodology, we decided to go with the waterfall model methodology, the five phases of software development lifecycle of the requisition system are planning design development testing deployment and Maintenance which further assist us in forming a working model reliable system that is up to specification and allows for further development, we made sure that the program is working and as expected.