Architecture Design Document for Case Study 1 (Course Scheduling System)

1. Overview

1.1. System Overview

The course scheduling system takes input from the Professors about their preferences, about course offerings and class rooms from the department secretary, and then proposes a suitable schedule for the courses.

1.2. System Context

The system context is defined clearly in the SRS. Basically, the department is the main sink of the information. The main sources of information are the Professors (who provide information about their time preferences and maximum enrollment,) and the department secretary (which provides information about courses begin offered, time slots, and available class rooms.)

1.3. Stakeholders of PIMS

The main stakeholders and their concerns are:

- **Professors:** Their main concern is that their highest priority should be satisfied. This means that the algorithm for scheduling should be such that it can easily be changed with a better algorithm later.
- **Department secretary/Head:** The schedule should be fair and should utilize the resources well. (Again this means that the scheduling algorithm should be upgradable.)

1.4. Scope of this Document

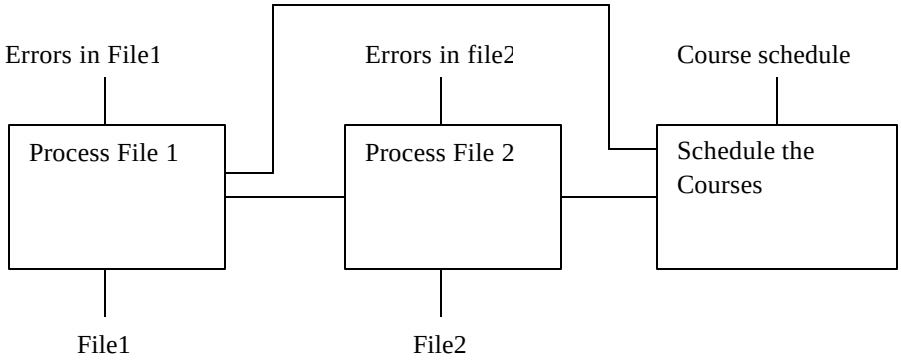
This document describes the proposed architecture for the course scheduling system. For architecture, we consider only the component and connector view.

1.5. Definitions and Acronyms

As given in the SRS.

2. Architecture Design

As this is a batch processing-type system with inputs coming and output being produced, the most natural style will be the pipe-and-filter style. We use this style for the architecture of the system. The proposed architecture is shown below.



This architecture has three filter components – one to process the information provided in File 1, the other to process the information provided in File 2, and the third to produce the schedule. As the information produced by the first component is also used in the processing of information from File 2 as well as for scheduling, the connections are set accordingly.

In this architecture, we do not require that the components be executed in parallel. For ease of implementation, they may be executed in a sequential order, particularly since the file processing modules are not likely to consume much time – the main processing time will be needed by the third component and that remains the same in parallel or serial execution. Any (synchronous or asynchronous) method can be used to support the pipes.

3. Evaluating the Architecture

Let us evaluate this architecture with respect to some properties.

Criteria	**Evaluation of the proposed Architecture**
Change in some file format	Good – change should impact only the filter for that file.
Change in scheduling algorithm	Good – only the scheduling component needs to be changed.
Adding new constraints for scheduling	Good – only the scheduling component needs to be changed
Replacing files with GUI	Poor – switching to a GUI interface will require changing almost the complete implementation with this architecture. The scheduling component can still be used.

Extension to web based	Poor – This architecture does not support this change; will require a complete rewrite of the system.
Provision of additional securities (passwords, etc)	Average – The architecture is not designed with security in mind. However, it is possible to add a security component for verification in the start.

So the architecture satisfies the current main criteria of being able to change the scheduling algorithm. However, if the system is to be later enhanced to newer technologies or approaches, then most likely the system built using this architecture will not be reusable, and might have to be built afresh.