Software Architecture and Design Document

# 1. Introduction

In large tertiary institutions, exam scheduling can be a cumbersome task. This is a difficult and tricky task, traditionally done manually by a team of tireless employees in the Exams and Graduations Office. The system that is being developed is aimed at solving this problem using a well-known computer science algorithm called Graph Colouring. The system will generate a more optimized exam timetable, that will be in the best interests of students by spreading out their exams over time.

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This Software Architecture Document provides an architectural overview of the Exam Scheduling System. The Exam Scheduling System is being developed by University of the Witwatersrand students to develop more optimised exam timetables.

1.3 Definitions, Acronyms, and Abbreviations

ESS - Exam Scheduling System (project name)

UML - Unified Modeling Language

MVC - Model View Control architecture

1.4 Overview

This report will present a detailed analysis of the architecture of the Exam Scheduling System. Further sections in this document cover the architectural representation of the project including architectural representation, architectural goals and constraints and use case realizations. Later sections cover the details of the main views of the system. Furthermore, the document describes the data view, performance view and quality view.

2. Architectural Representation

This document presents the architecture as a series of views; use case view, logical view, process view, deployment view and implementation view. These are views on an underlying Unified Modeling Language (UML) model. This document also describes the data view, performance and quality views of the system.

### 2.1 Logical View

Audience: Designers, Programmers, Testing staff

Area: Functional requirements, system layers

Describes the design of object model

### 2.2 Process View

Audience: Programmers

Area: Non-functional requirements, describes the design’s concurrency and synchronization aspects.

### 2.3 Deployment View

Audience: DB administrators, system engineers, deployment managers

Area: Persistence- describes architecturally significant persistent elements in the data model. Describes the mapping of the software onto the hardware and shows the system’s distributed aspects.

### 2.4 Implementation View

Audience: Programmers, code testers

Area: software components, describes the modules and subsystem divisions of the system.

6. Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

#### 6.1 Server side

ESS (Exam Scheduling System) will be hosted on a web server. Since this is a web based application, the underlying client OS can be any computer operating system (Linux, MAC, Windows). Apache will be used as the central database server. All communication with client has to comply with public HHPS, TCP/IP communication protocol standards.

### 6.2 Client side

Users will be able to access ESS only online. Initially, this will be hosted as a standalone application but may later be integrated with other university systems.Clients/users are expected to use a modern web browser such as Mozilla Firefox 10, Internet Explorer 9, Google Chrome, or Safari to get full user experience.

### 6.3 Persistence

All the data will be saved in the central server. This is a relational database that implements the 3rd normal form (Apache).

### 6.4 Reliability/Availability

The system will be subjected to several continuous testing (unit testing, integration testing, system testing) before being deployed to make sure that the system is reliable. These tests will be implemented at every phase of development. The Apache server can respond to many clients at the same time while maintaining data integrity.

### 6.5 Performance

The system responds to any request under standard database and web server timeouts (30 seconds). The system performance also depends on available hardware, network and internet connection capabilities. Tasks like the timetable generation on hundreds of courses may take comparatively high time. Therefore, actual performance can only be determined after the system is deployed and tested.

### 6.6 Development tools

The project incorporates many development tools.

Programming: Python (Jupyter notebook), React.js and Express.js (middleware)

Database: MySQL

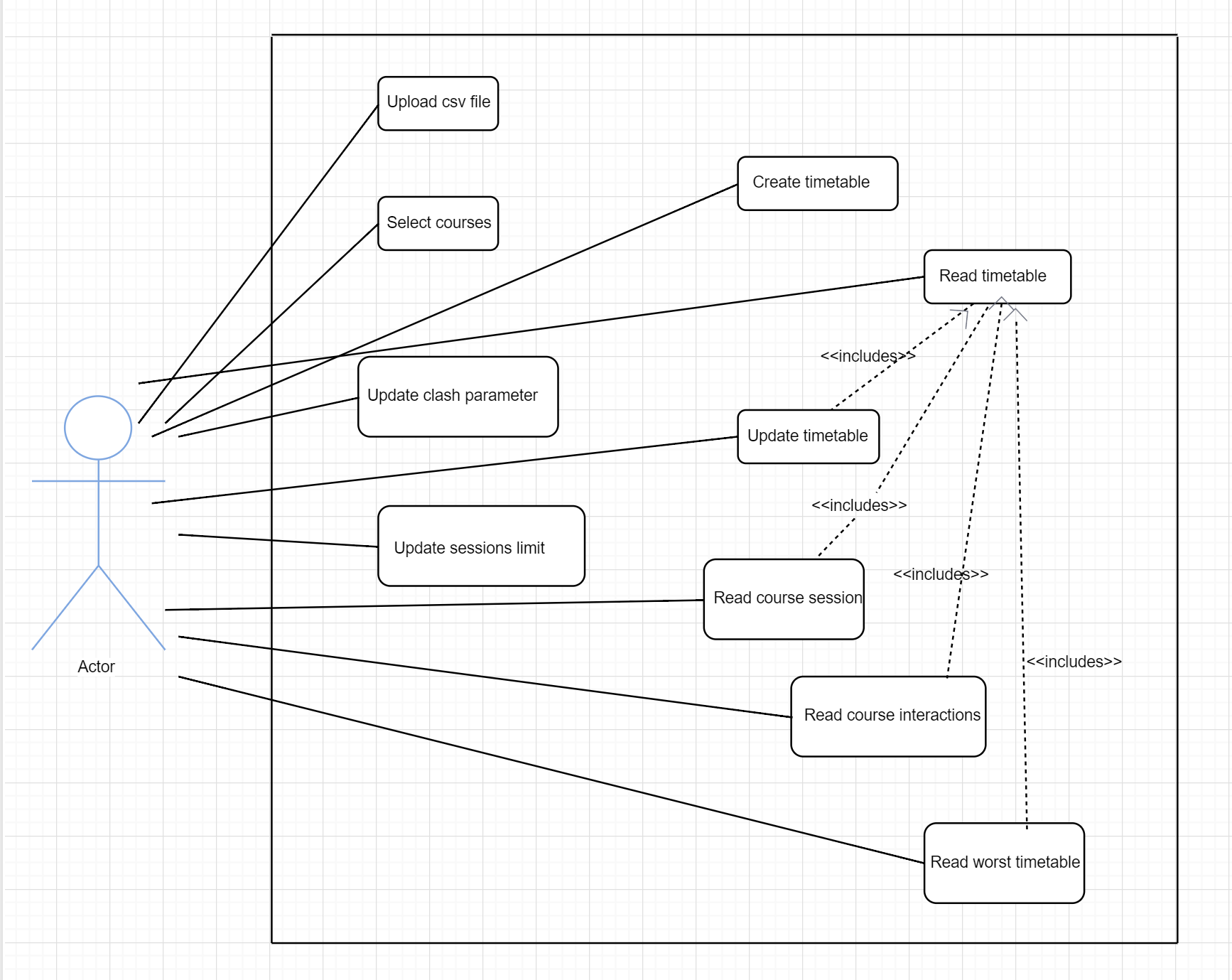
Diagrams: draw.io

Database connection: MySQL connector

Schedule: taiga.io

7. Architecturally Significant Use Cases

Use Case Diagram:



The actor, in this case will be the administrator at the Exams and Graduations Office. The actor will invoke the following use cases:

**Upload csv file**- to upload the courses.csv and students.csv files

**Select courses**- which allows the actor to choose which courses to schedule

**Update clash parameter**- changes the current clash parameter to the new parameter entered by the actor

**Update sessions**- change the max number of exam sessions to the one specified by the actor

**Create timetable**- this is the actual generation of the optimised timetable

**Read timetable**- this displays the generated timetable

**Update timetable**- generates a new timetable after parameters have been changed then it invokes Read timetable to display the updated timetable

**Read course session**- invokes Read timetable to find the course in the generated timetable then it displays the session details for that course

**Read course interactions**- invokes Read timetable to find all the courses that interact with the specified course in the generated timetable then it displays all the related courses

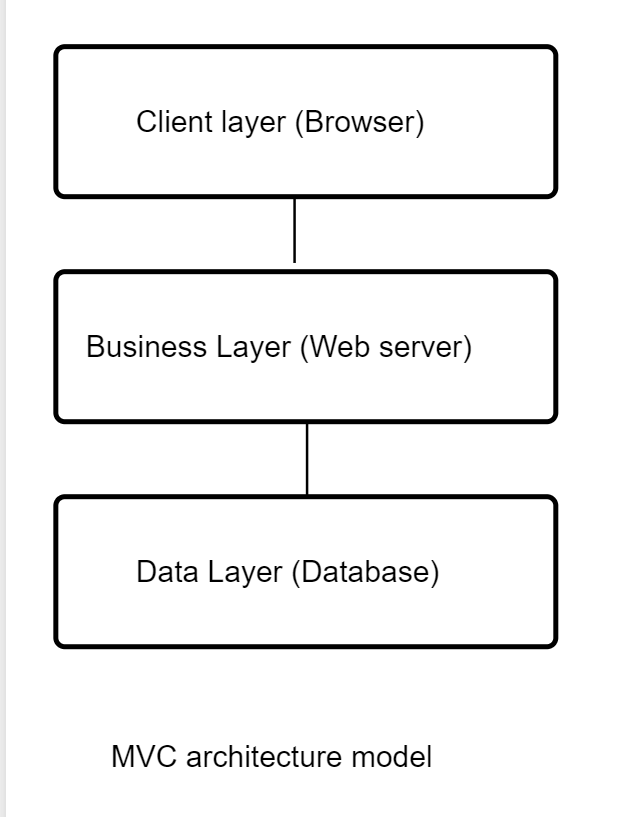
**Read worst timetable**- invokes Read timetable to find the student with the worst timetable and display the worst timetable

8. Logical (Class) View

This is a description of the logical view of the architecture. It describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. It also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. A class diagram is included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

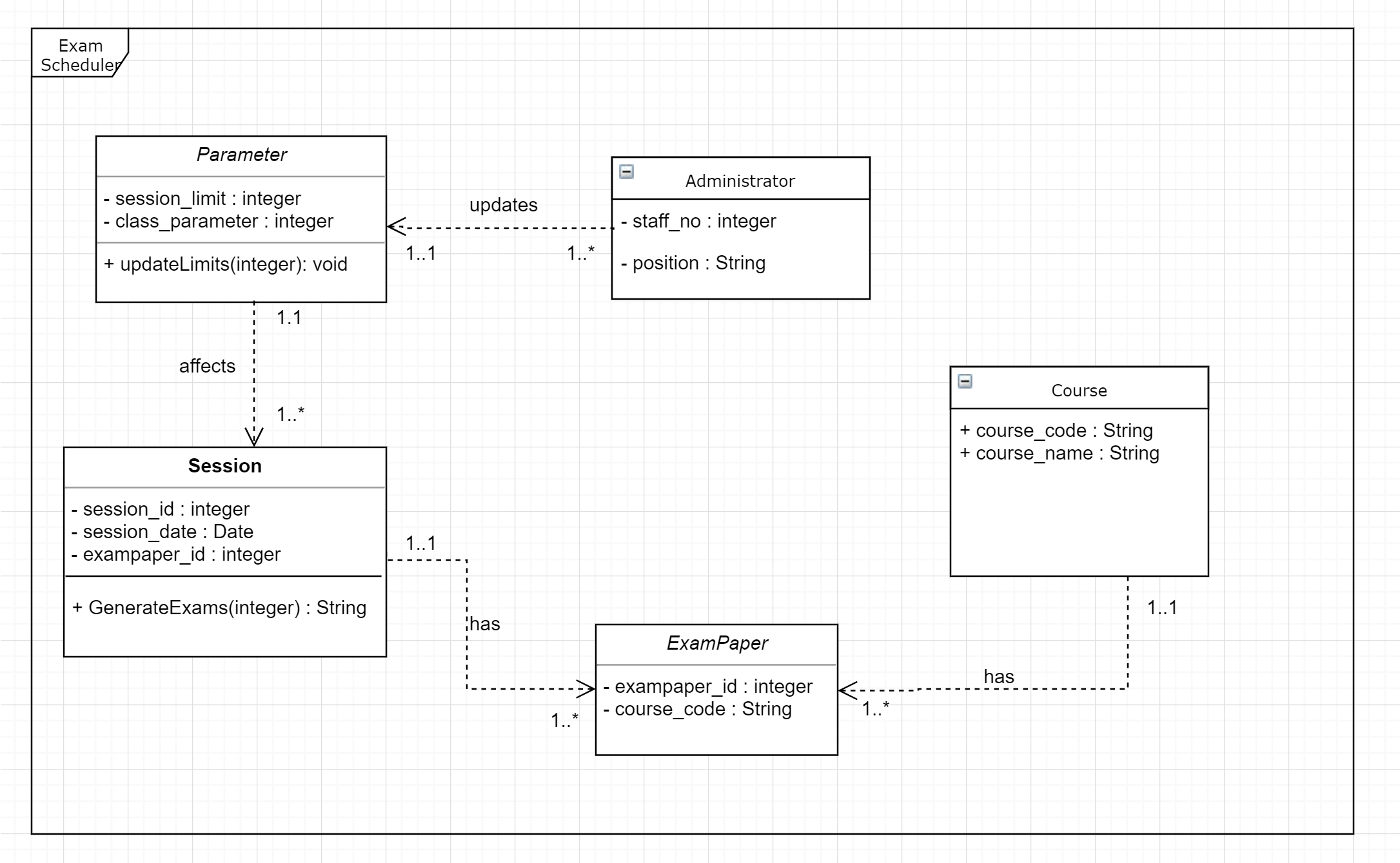
### 8.1 Layering

The ESS is divided into 3 layers. The layering model of the ESS application is based on associating each layer with a particular responsibility. This model was chosen because it separates various system responsibilities from one another so as to improve system development, reusability and maintenance.



### 8.2 Architecturally Significant Design Packages

Class Diagram:



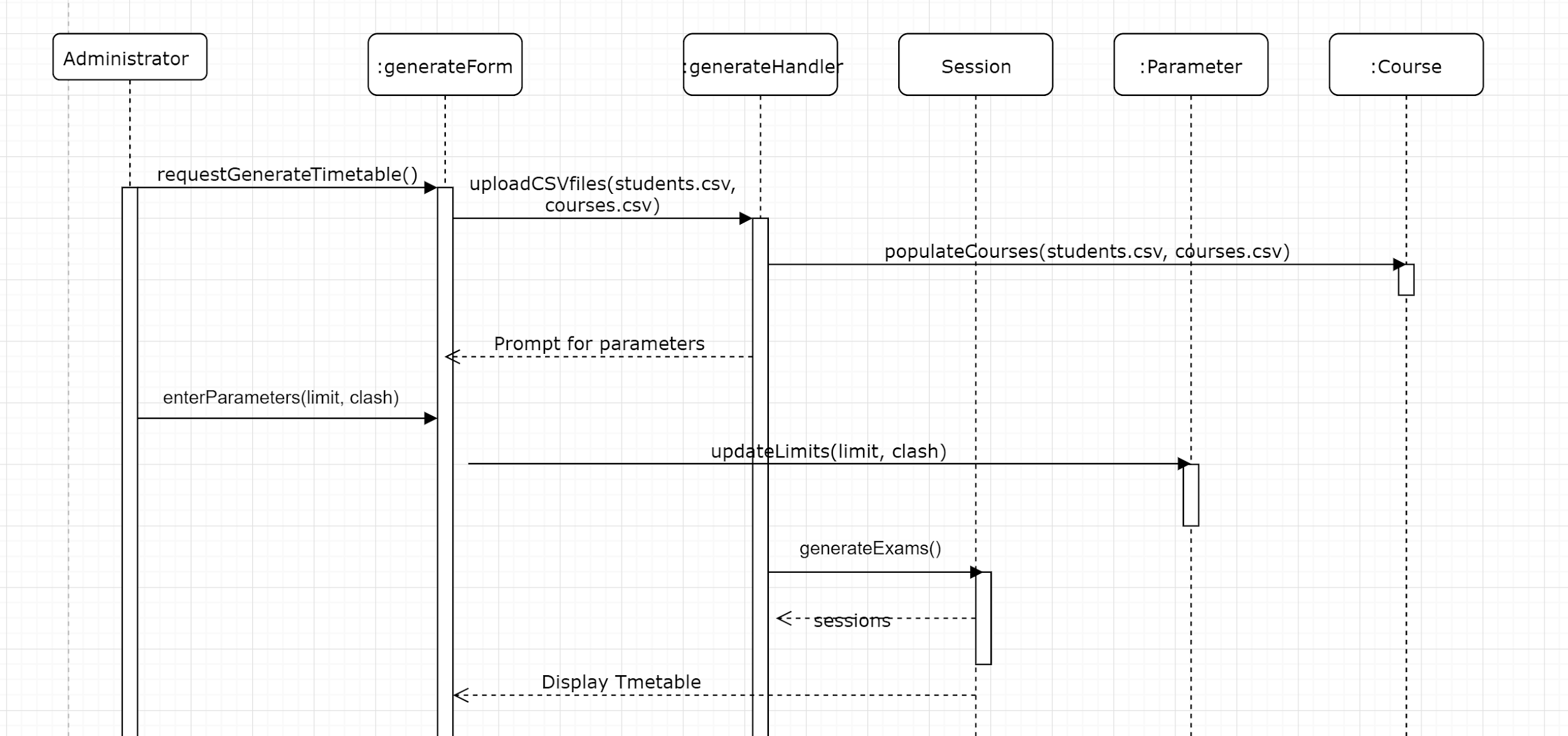
Class diagram : Exam Scheduling System

Especial design decisions:

* All users are derived from the superclass called “Administrator”. Each user of the system has common attributes such as, staff\_no and position.
* Each user can update a parameter set. A parameter set can be updated by many users. Hence the one to many relationship.
* Each parameter affects many exam sessions. Each exam session can be affected by one parameter set. Therefore, one to many relationship.
* An exam paper can only belong to only one session. But a session can have many exam papers written in it. Hence one-to-many relationship.
* A course can have one or many exam papers, but an exam paper can only belong to one course.

9. Process View

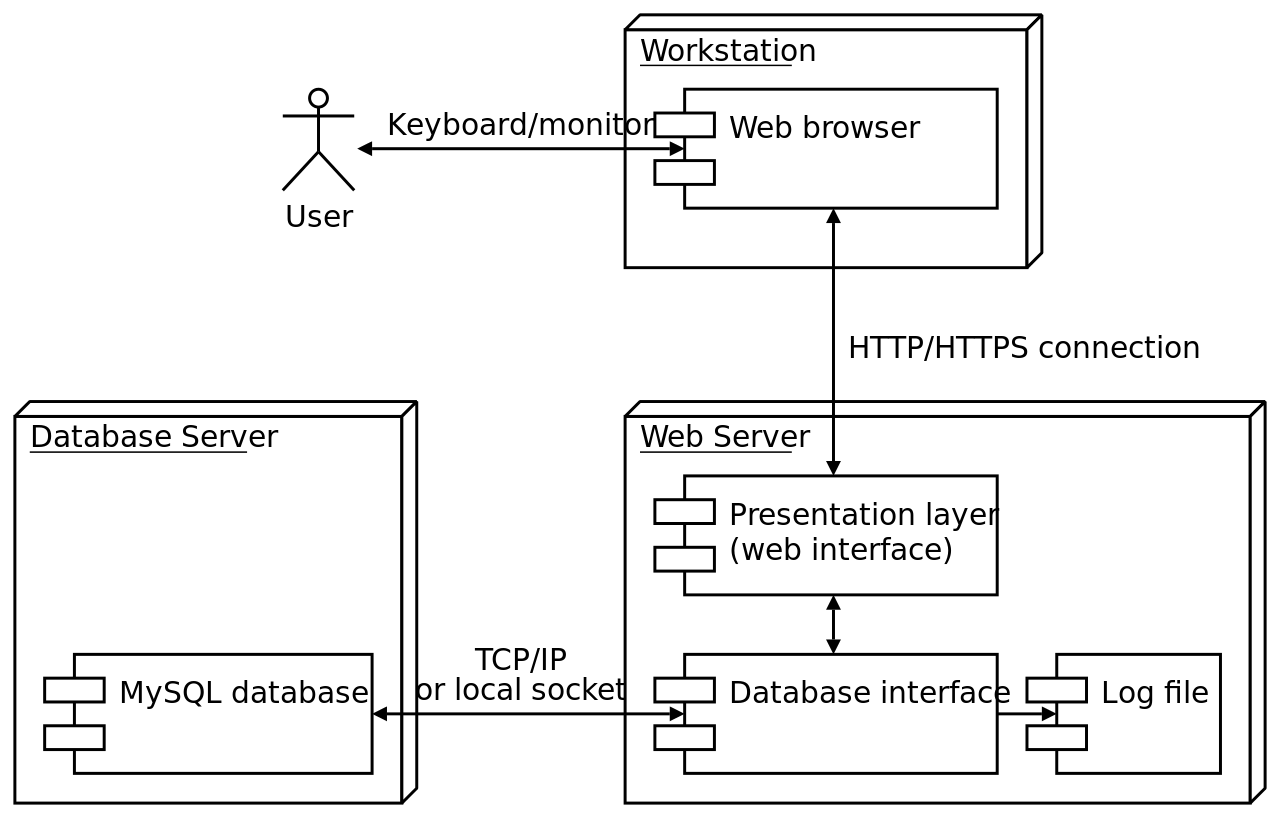
A description of the process view of the architecture. It elaborates the run time behaviour of the system. The Sequence Diagram illustrates the exam scheduling organized as executable processes. Processes exist to support uploading csv files, specifying clash parameters, generating the timetable, viewing the timetable in different formats.



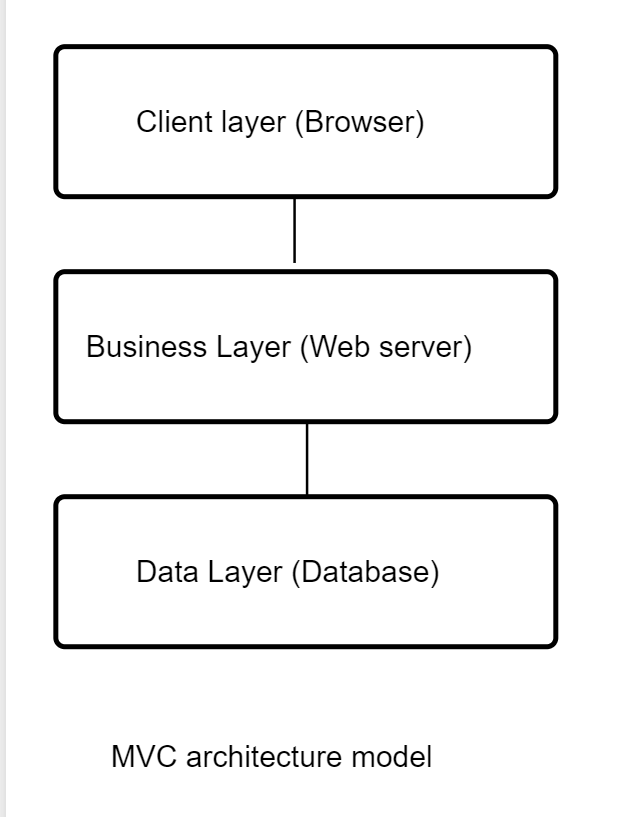
Sequence Diagram : Generate Exams

10. Deployment View

Being a web application, this ESS system is hosted in a remote server. The database is hosted in some other hosting space. Most of the processing (generating the timetable) is done in the backend, so that the client computer does not spend too much of CPU power. A deployment diagram is shown below to visualise the software-to-hardware implementation.



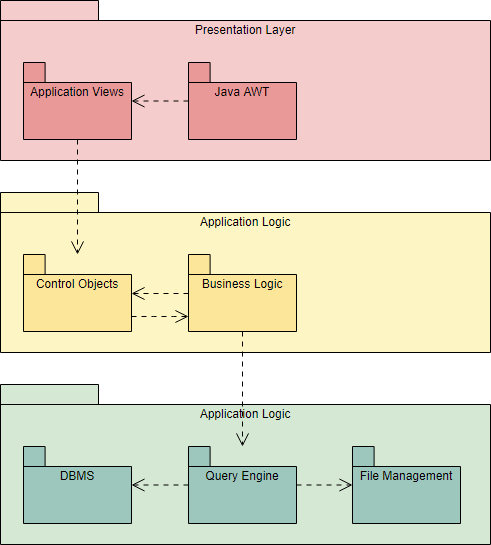
11. Implementation View



ESS (Exam Scheduling System) is a web application that follows the MVC architecture pattern. The reason being that, it separates functions into layers thus improving maintainability and reusability.

The client layer contains the graphical user interfaces (web pages). The actions of these web pages are handled by controller classes. Controller classes invokes and instantiates objects of model classes that contain business logic. Separating packages in this manner reduces complexity. This is depicted diagrammatically in the Package Diagram below.

The model classes can be subdivided into two layers, the business layer and the data layer. Data layer is manipulated through the DBMS framework. The business layer contains the main entity classes that process the actual algorithm.

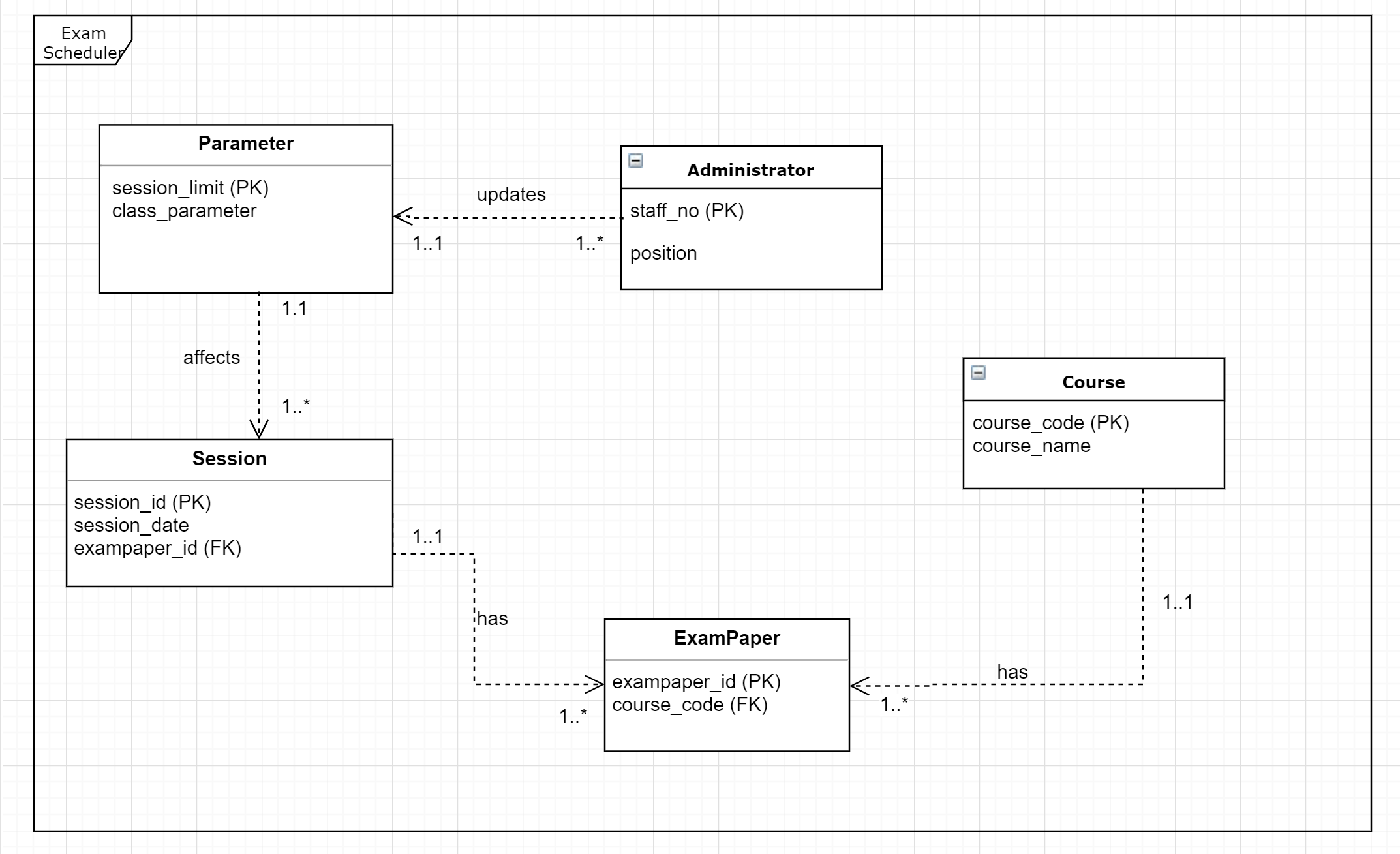


12. Data View

The following is an ER diagram which represents how the database looks and all the attributes and tables. The database implementation for this ER diagram is done using MySQL. The main tables are:

1. Administrator - The EGO staff member
2. Parameter
3. Session
4. ExampPaper
5. Course

From that, a class is made for each table in the database, and that makes it possible to access the database.



13. Size and Performance

The selected architecture supports the sizing and timing requirements through the implementation of a client-server architecture. The client portion is implemented on local campus PCs or remote dial up PCs. The size of the software is still not calculated but will be known by the end of the project. The client computers need to have a web browser to access the system. All the functionalities wiil be processed at the backend. The components have been designed to ensure that minimal disk and memory requirements are needed on the PC client portion.

14. Quality

The software architecture supports the quality requirements

1. The user interface of the ESS shall be designed for ease-of-use and shall be appropriate for a computer-literate user community with no additional training on the System.
2. Each feature of the ESS shall have built-in online help for the user. Online Help shall include step by step instructions on using the System. Online Help shall include definitions for terms and acronyms.
3. The ESS shall be available 24 hours a day, 7 days a week.

More of the quality parameters for the ESS (Exam Scheduling System) are covered in the Architectural goals and constraints section.

15. References

<https://www.uml-diagrams.org/web-application-uml-deployment-diagram-example.html?context=depl-examples>

<http://sce.uhcl.edu/helm/rationalunifiedprocess/process/artifact/ar_sadoc.htm>

<http://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm>