

# Development of Exam Timetable Generation Software [ETS] for College

by

OGBONDA GLORY (09094338741)

Software Engineering – Revolutionary Master Mind Series

NIIT- Port-Harcourt  
2016-2017

# Abstract

A college timetable is a temporal arrangement of a set of lectures and classrooms in which all given constraints are satisfied. Creating such timetables manually is complex and time-consuming process. By automating this process with computer assisted timetable generator can save a lot of precious time of administrators who are involved in creating and managing course timetables. Since every college has its own timetabling problem, the commercially available software packages may not suit the need of every college. Hence I have developed practical approach for building lecture course timetabling system, which can be customized to fit to any colleges timetabling problem.

This project introduces a practical timetabling algorithm capable of taking care of both strong and weak constraints effectively, used in an automated timetabling system. So that each teacher and student can view their timetable once they are finalized for a given semester but they can't edit them.

## Contents

Introduction .....	5
1.1    Introduction .....	5
1.2    Project Plan .....	5
1.2.1    Scope of Project .....	6
1.2.2    Estimation of the resources .....	6
1.3    General Problem Formulation .....	7
Review of Literature .....	8
2.1    Timetabling .....	8
2.1.1    Timetabling problems .....	8
The Constraint Programming (CP) .....	9
2.1.2    The General View .....	10
2.2    Constraints .....	10
2.2.1    Hard Constraints .....	10
2.2.2    Soft Constraints .....	11
Description of Project .....	13
3.1    Proposed System .....	13
3.2    ETS Features .....	14
3.3    Hardware & Software Details .....	16
Hardware Requirement .....	16
Software Requirement .....	16
3.4    Analysis & Design .....	16
3.4.1    Java & XML .....	16
3.4.2    Data Structure & Flow Diagram .....	18
3.4.3    Class Diagram .....	20
3.4.4    Use Case Diagram .....	22
Design Approach .....	23
4.1    Design Consideration: XML Knowledgebase .....	23
Implementation .....	24
5.1    User Interface Code .....	24
5.2    Output Generator Code .....	26
5.3    Timetable Generator Code .....	26
Testing and Results .....	30

6.1	Screen Shot of ETS .....	30
6.1.1	User Interface Screen.....	30
6.1.2	Timetable View Screen .....	39
Conclusion.....		42

# Chapter 1

## Introduction

### 1.1 Introduction

Even though most college administrative work has been computerized, the lecture/exam timetable scheduling is still mostly done manually due to its inherent difficulties. The manual lecture/exam-timetable scheduling demands considerable time and efforts. The lecture/exam-timetable scheduling is a constraint satisfaction problem in which I found a solution that satisfies the given set of constraints.

The college lecture/exam-timetabling problem asks us to find some time slots and classrooms which satisfy the constraints imposed on offered courses, lecturers, classrooms and so on. Since the problem is a combinatorial optimization problem belonging to NP-hard class [1], the computation time for timetabling tends to grow exponentially as the number of variables increase.

There have been a number of approaches made in the past decades to the problem of constructing timetables for colleges and schools. Timetabling problems may be solved by different methods inherited from operations research such as graph coloring and mathematical programming, from local search procedures such as tabu search and simulated annealing, from genetic algorithms or from backtracking-based constraint satisfaction manipulation. In our project, timetabling problem is formulated as a constraint satisfaction problem and we proposed a practical timetabling algorithm which is capable of taking care of both strong and weak constraints and finding variables' instantiation, which is based on the forward search method.

### 1.2 Project Plan

The first activity in software project planning is the determination of software scope. The second software planning task is estimation of the re-sources.

### 1.2.1 Scope of Project

**Timetable Generation System** generates timetable for each class and teacher, in keeping with the availability calendar of teachers, availability and capacity of physical resources (such as classrooms, laboratories and computer room) and rules applicable at different classes, semesters, teachers and subjects level.

Best of all, this Timetable Generation System tremendously improves resource utilization and optimization.

### 1.2.2 Estimation of the resources

I used the resources as shown below:

#### PEOPLE:

- Practitioners who deliver the technical skills that are necessary to engineer a product or application (Project Team)
- End-users who interact with the software once it is released for production use (Timetable Users)

#### Project Team:

- Ogbonda Ebube Glory (Testing, Analysis and Implementation, Coding, Design and Implementation, Design and Testing)

#### REUSABLE SOFTWARE COMPONENTS:

##### Java Packages:

Com.lowagie.text {itext -1.3.jar for .pdf file creation}

Org.dom4j {jaxen-1.1.1.jar and dom4j-1.6.jar for .xml file reading}

#### HARDWARE/SOFTWARE TOOLS:

##### Hardware Requirement:

- Pentium IV processor machine
- Minimum RAM: 256 MB to 512 MB
- Disk free space: 2250 MB+

##### Software Requirements:

- Operating System: Windows 2000/XP
- Other Software JDK 1.5+
- IDE Used: NetBeans

## 1.3 General Problem Formulation

The considered college is a four-year college which has the following characteristics on its course administration.

1. The college offers courses for daytime students.
2. The classes for daytime students are scheduled in the weekday's daytime and Saturday morning.
3. The types of lectures are:
  - Theory lecture
  - Practical
4. The class size of theory lectures is from 40 to 60.
5. The group size for practical hours is from 15 to 25.
6. A minimum timeslot is a 1 Hour interval. For theory classes takes 1 timeslot and for practical takes 2 timeslots.

Once a lecturer decides to offer a course for a specific year-session of a department, an offered course  $X_i$  takes place in the timetabling problem, which is expressed as a tuple of attributes. Except Timeslots and Rooms, all attributes of  $X_i$  are determined at the time the course is decided to be offered. Both Timeslots and Rooms are list fields to contain assigned time slots and classrooms for the course  $X_i$ . To indicate an attribute attr of an offered course  $X_i$ , I use the notation  $X_i.attr$ . The time slots are generally assigned from 8.30 AM to 5.00 PM for weekdays. The time slots are labeled as  $T_i$  ( $i = 1\ 2\ 3\ \dots$ ).

# Chapter 2

## Review of Literature

### 2.1 Timetabling

A timetable construction is an NP-complete scheduling problem. It is not a standard job-shop problem because of the additional classroom allocation. It is large and highly constrained, but above all the problem differs greatly for different colleges and educational institutions. It is difficult to write a universal program, suitable for all imaginable timetabling problems. Although manual construction of timetables is time-consuming, it is still widespread, because of the lack of appropriate computer programs.

#### 2.1.1 Timetabling problems

There exist many different timetabling problems such as:

- University Timetabling
- Examination Timetabling
- School Timetabling
- Sports Timetabling
- Employee Timetabling

Furthermore, there exist many problem solving methods, which usually use the concepts of standard optimization algorithms such as Backtracking, Evolutionary Algorithms or Constraint Logic Programming.

In recent years two main approaches seem to have been successful.



- The first approach is based on Local Search Procedures
- The second approach is based on Constraint Programming (CP) **The Local Search**

### **Procedures**

The local search procedures such as Simulated Annealing, Tabu Search and Genetic Algorithms. These methods express constraints as some cost functions, which are minimized by a Heuristic Search of better solutions in a neighborhood of some initial feasible solution. Their greatest disadvantages are:

1. The difficulty of taking into account hard constraints
2. The need to determine their parameters through experimentation.

Although they are good for optimizing the initial feasible solution, they have problems with finding it.

### **The Constraint Programming (CP)**

Its main advantage is declaratively: a straightforward statement of the constraints serves as part of the program. This makes the program easy to modify, which is crucial in timetabling problems. The constraints are handled through a system of constraint propagation, which reduces domains of variables, coupled with backtracking search. In modern CP languages, both features do not need to be programmed explicitly. The main disadvantages of this approach are:

1. The difficulties with expressing soft constraints
2. The possible problems with improving the initial feasible solution, which - as a rule - may be determined without difficulties.

The ability to express complex constraints in a simple, declarative way is crucial for introducing the requirements of the colleges and university timetabling problem into the program and is crucial for their successful solution, - a custom-tailored distribution strategy is able to introduce soft constraints during a search, leading quickly to a "Good" timetable, - incorporation of local search into CP gives the ability to optimize effectively the timetable.

## 2.1.2 The General View

As mentioned above, many types of timetabling problems exist. But all these problems have several properties in common. One of these similarities is that certain entities have to be scheduled. e.g., the college timetabling problem has several entities such as students, lecturers, courses, practical and classes and labs. All these entities have properties e.g. classes are linked to the course and the students of this class are taught. Constraints Assignments usually cannot be done arbitrarily, but many constraints have to be considered. We distinguish two different types, namely hard and soft constraints. A solution is feasible if no hard constraints are violated. A feasible solution is better than another if fewer soft constraints are violated.

A timetabling algorithm can use different strategies to get a solution without violations of hard constraints. Violations can either be avoided from the outset or penalized to lead the algorithm towards better solutions and introduce repair mechanisms.

## 2.2 Constraints

There are various constraints to be satisfied at the time to instantiate variables about time slots and classrooms. The constraints can be categorized into Hard and Soft constraints.

### 2.2.1 Hard Constraints

A timetable which breaks a hard constraint is not a feasible solution, and must be repaired or rejected by the timetabling algorithm. Hard constraints include “First Order Conflicts”,

HC1	A classroom is not assigned to more than one lecture at the same time.
HC2	An lecturer cannot teach more than one class at the same time.
HC3	Courses for the same year-session students of a department cannot take place at the same time.
HC4	The classroom for a course should have enough capacity to take students registered in the course.
HC5	The classroom should be well equipped with required facilities for the classes.

Table 2.1: Hard Constraints

### 2.2.2 Soft Constraints

Soft constraints are less important than hard constraints, and it is usually impossible to avoid breaking at least some of them. Whichever timetabling method is applied, timetables are usually rated by a penalty function, which calculates the extent to which a timetable has violated its soft constraints. Some soft constraints are more important than others, and this is often specified with a priority value.

SC1	The lectures are not assigned to time slots, which are in the lecturer's forbidden time zones.
SC2	Lecturers' daily lecture hours should be restricted to be within the allowed maximum hours.
SC3	As far as possible, classes are scheduled in the lecturer's preferred time zones.
SC4	A lunch break must be scheduled.
SC5	The practical courses are scheduled in morning session, and the theory courses are scheduled in afternoon session.
SC6	If possible, the lecture hours for a course should be scheduled consecutively.
SC7	As far as possible, classes should be scheduled in their corresponding department's exclusive-use classrooms.
SC8	The classrooms should be allocated in a manner to minimize the distances between adjacent classes' classrooms.

Table 2.2: Soft Constraints

It is desirable for timetables to satisfy all hard and soft constraints. However, it is usually difficult to meet all these constraints. Any hard constraint must not be violated in any case, but some soft constraints can be sacrificed to find feasible timetables.

The timetabling process is made more difficult by the fact that so many people are affected by its outcome. I identified three main stakeholders in this process each with their own set of aims and wants [2]:

1. The administration sets the minimum standards that the timetable must conform to.

2. The departments concerns are more likely to be prominent in the course timetable. They will want the schedule to be consonant with the development of the subject taught as well as making more specific demands for particular classrooms or labs.

The third group of stakeholders are the students, whose view of the timetable will be restricted to the part that affects them. Given the number of students involved it is difficult to obtain specific criteria as to what is the best timetable for students.

# Chapter 3

## Description of Project

### 3.1 Proposed System

My Timetabling Algorithm is main component of my project which produces the HTML based timetable even / odd semester sheet as the output.

My project takes various inputs from the user such as Teacher List, Course List, Semester List, Room List, Day List and Timeslot as well as various rules, facts and constraints using web based forms, which are stored in XML based knowledge base. This knowledge base serves as input to my Timetable Generator Algorithm residing on server machine.

My knowledgebase is in the middle, because it is between my timetabling algorithm and GUI front end which is designed in the last. After the representation of KB is standardized, I designed the timetabling algorithm. The design of timetabling algorithm took most of my total time. During design of algorithm, first problem was, from where to start? Second problem was, does it really going to work? But after all due to my superior design of knowledgebase, flowcharts and enough thinking on timetabling data structure representation helped me to really boost building my fine working algorithm.

fig. shows Proposed System - Timetable Generation System

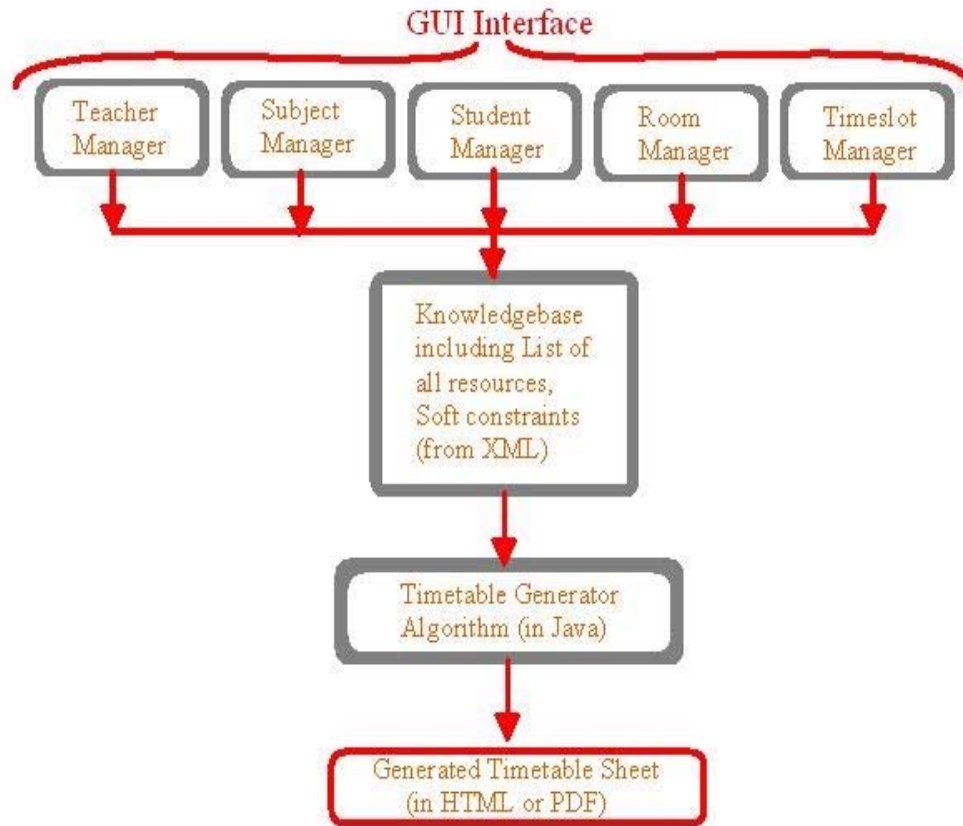


Figure 3.1: Proposed System

## 3.2 ETS Features

- *Simple...* Easy-to-understand & simple-to-operate with intuitive user interface, guides and samples
- *Fast...* Generates timetable in minutes, saving your time, efforts and money
- Rules configuration and verification

- Different views of timetable depending upon who is viewing timetable with my Timetable Generation System, timetable generation process involves just a few simple steps:

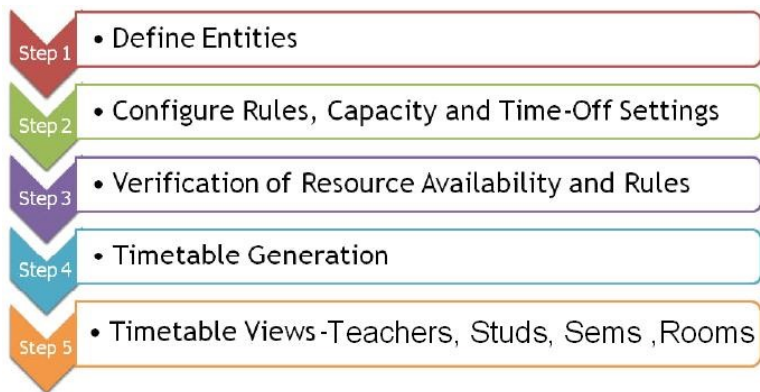


Figure 3.2: Timetable Generation process

fig. shows General View of Timetable Generation System

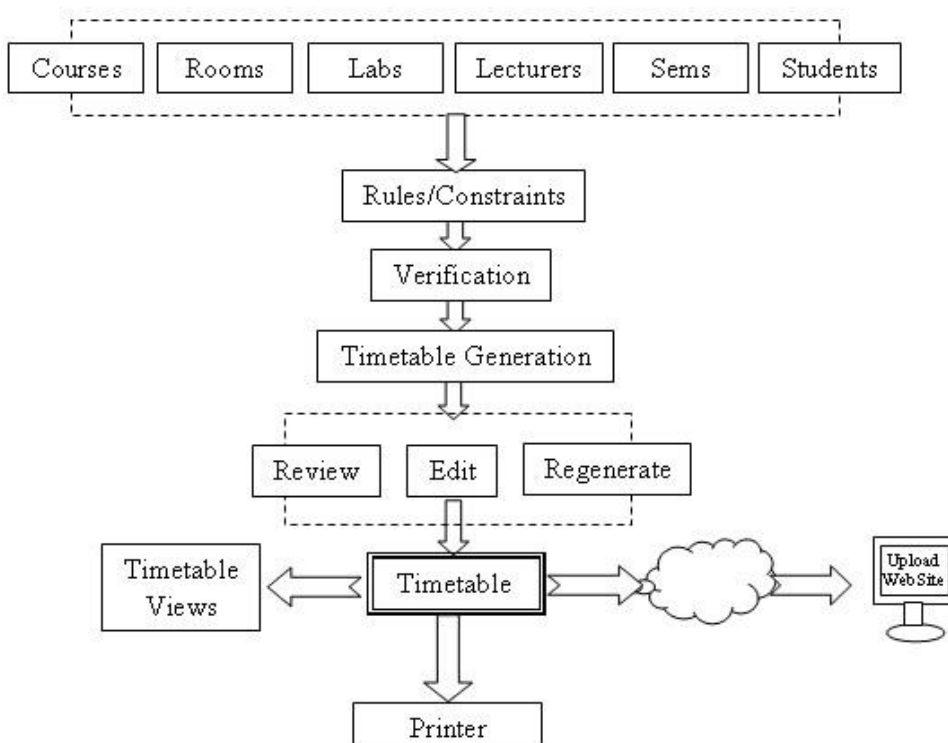


Figure 3.3: General View of ETS

## **3.3 Hardware & Software Details**

### **Hardware Requirement**

- Pentium IV Processor machine
- Minimum RAM: 256 MB to 512 MB
- Disk Free space: 250 MB+

### **Software Requirement**

- Operating System: Windows 2000/XP
- Other Software: JDK 1.5+
- IDE Used: NetBeans

## **3.4 Analysis & Design**

### **3.4.1 Java & XML**

#### **Java:**

Java is widely adopted and there is a vast range of both commercial and open source libraries available for the platform, making it possible to include support for virtually any system, including native applications via JNI or JNA. When it comes to RIAs Java's main weakness is its multimedia support. Java 6 Update N improves some features that have hindered the use of Java for RIAs including startup time and download size, and Sun may even include new multimedia support in this release (due Q2,2008).

#### **Details about Java:**



Java is a programming language originally developed by Sun Microsystems and released in 1995 as a core component of Sun's Java platform. The language derives much of its syntax from C and C++ but has a simpler

object model and fewer low-level facilities. Java applications are typically compiled to bytecode which can run on any Java virtual machine (JVM) regardless of computer architecture.

#### **Platform independence:**

One characteristic, platform independence, means that programs written in the Java language must run similarly on any supported hardware/operating system platform. One should be able to write a program once, compile it once, and run it anywhere.

#### **XML:**

The Extensible Markup Language (XML)[3] is a general-purpose specification for creating custom markup languages. It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet, and it is used both to encode documents and to serialize data.

It started as a simplified subset of the Standard Generalized Markup Language (SGML), and is designed to be relatively human-legible. By adding semantic constraints, application languages can be implemented in XML. These include XHTML, RSS, MathML, GraphML, Scalable Vector Graphics, MusicXML, and thousands of others. Moreover, XML is sometimes used as the specification language for such application languages.

XML is recommended by the World Wide Web Consortium. It is a free open standard. The W3C recommendation specifies both the lexical grammar and the requirements for parsing.

#### **Advantages of XML:**

1. It is text-based.
2. It can represent common computer science data structures: records, lists and trees.
3. The strict syntax and parsing requirements make the necessary parsing algorithms extremely simple, efficient, and consistent.
4. XML is heavily used as a format for document storage and processing, both online and offline.

5. The hierarchical structure is suitable for most (but not all) types of documents.

6. It is platform-independent, thus relatively immune to changes in technology.

### **3.4.2 Data Structure & Flow Diagram**

XML File contains following sections as shown in Figure.3.4

1. Teacher List

2. Subject List

3. Room List

4. Timeslot List

5. Years/Students List

6. Days Options

7. Constraints, Rules



Figure 3.4: Data Structure

**Flow of Data within project:** Following Figure.3.5 shows the flow of data from user to timetable generator algorithm and back to user

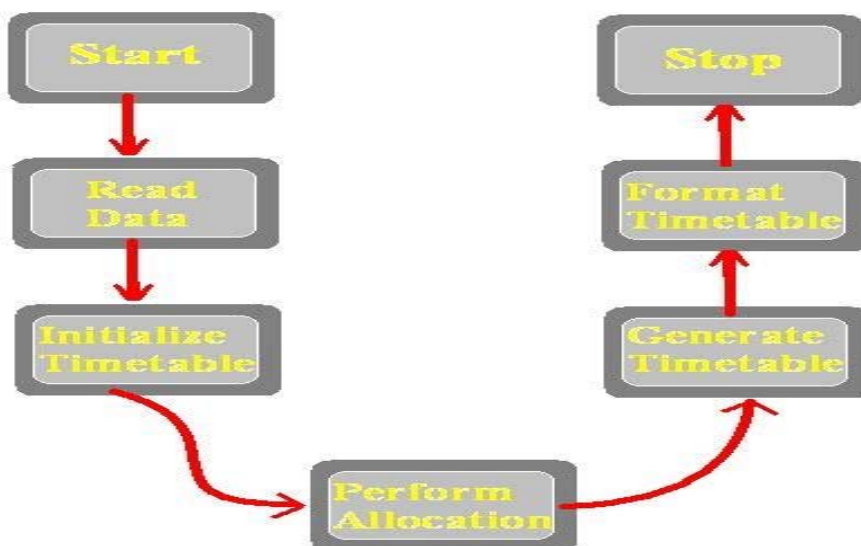


Figure 3.5: Flow of Data within project

### 3.4.3 Class Diagram

**Packages:** Packages allow us to break up a large number of objects into related groupings. In many object oriented languages (such as Java), packages are used to provide scope and division to classes and interfaces.

**Classes:** The core element of the class diagram is the class. In an object oriented system, classes are used to represent entities within the system; entities that often relate to real world objects.

The following Figure.3.6 shows the package DataSourceKB which contains classes to handle Back End (Knowledge Base)

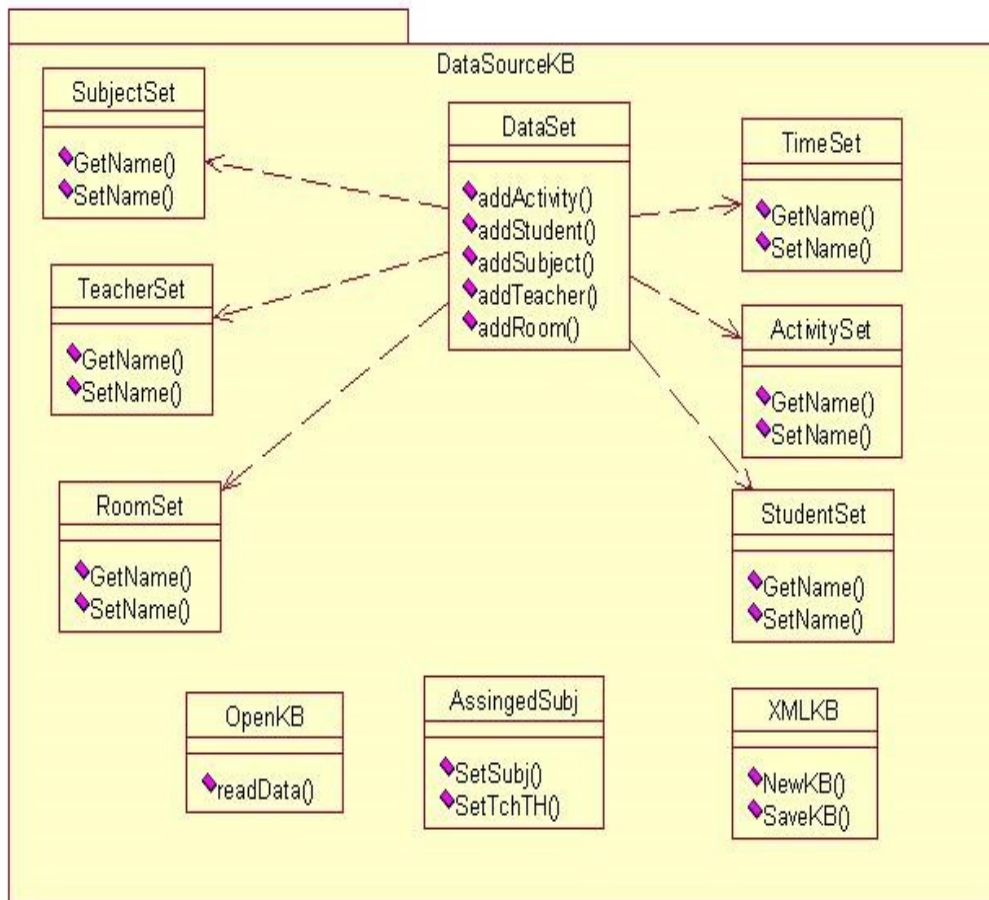


Figure 3.6: Class Diagram 1

The following Figure.3.7 shows the package ETS which contains sub packages and classes to create Front End.

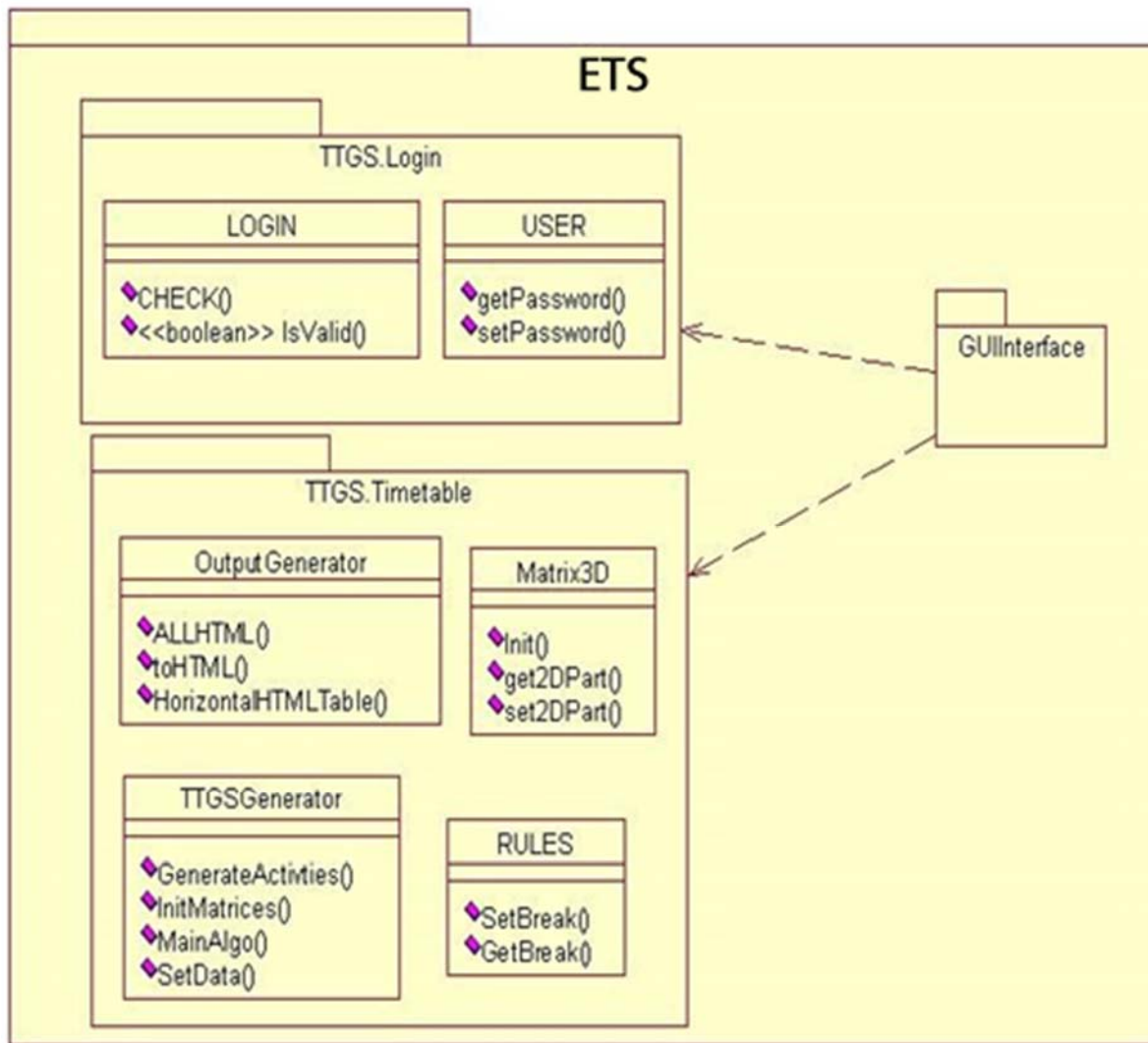


Figure 3.7: Class Diagram 2

### 3.4.4 Use Case Diagram

The following Figure.3.8 shows the use case diagram for my timetable generation system.

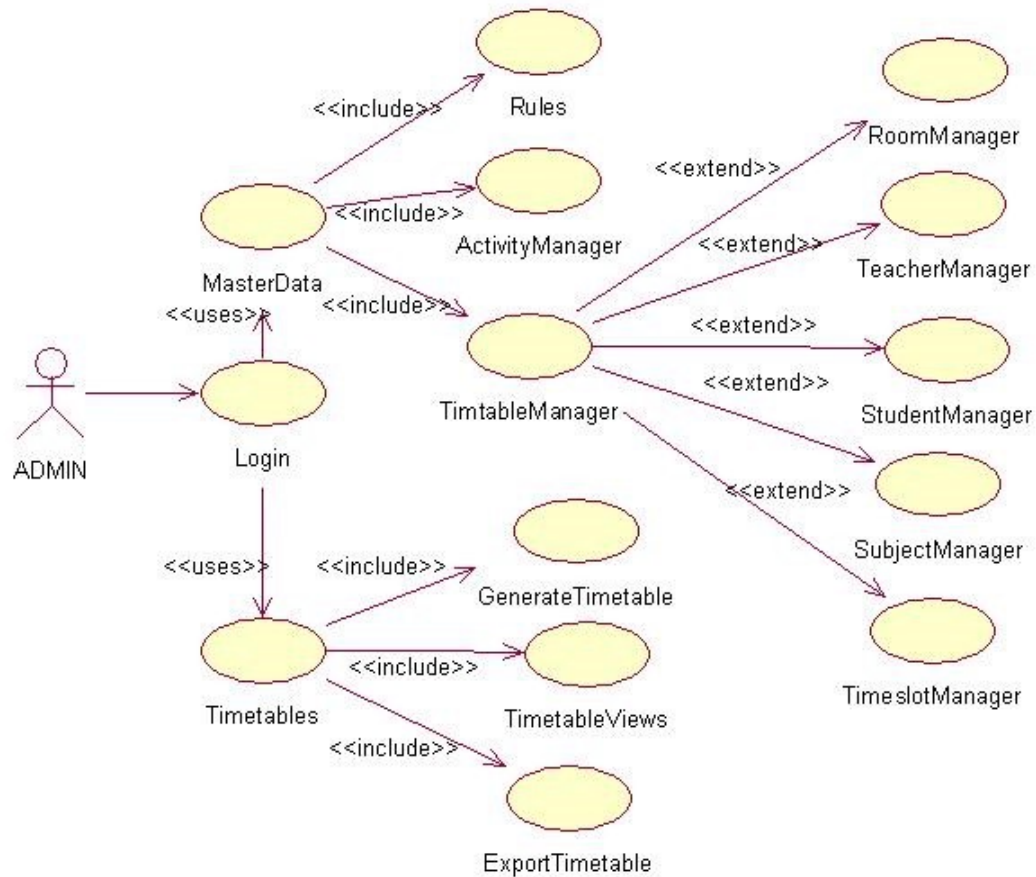


Figure 3.8: Use Case Diagram

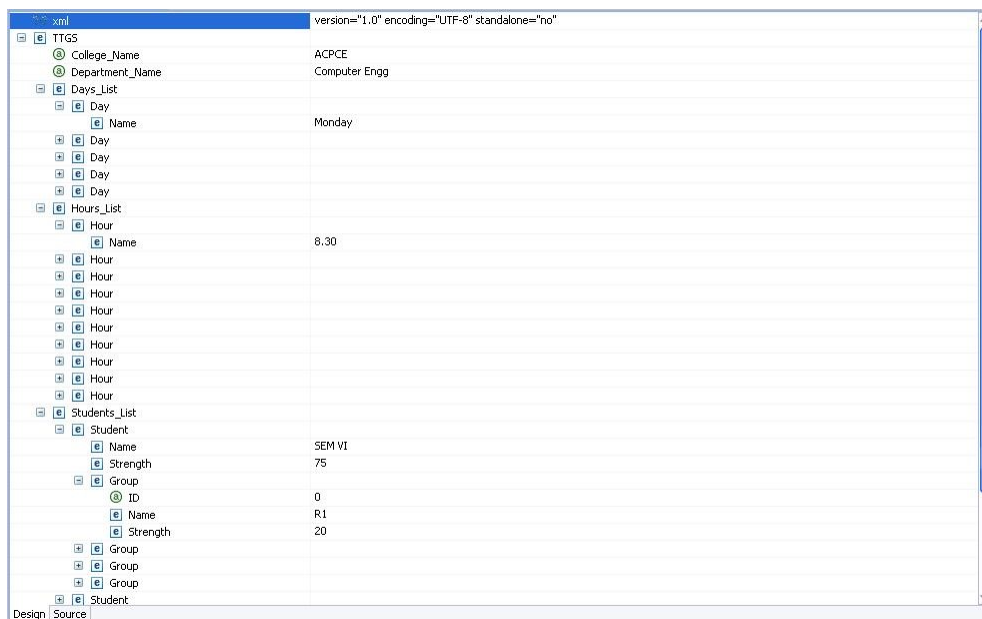
A use case is made up of a set of scenarios (such as: Login, Data Management, Rules settings etc.). Each scenario is a sequence of steps that encompass an interaction between a user and a system.

# Chapter 4

## Design Approach

### 4.1 Design Consideration: XML Knowledgebase

As we preferred the XML as our back-end we have to design it first then retrieve it using Java into our application.



version="1.0" encoding="UTF-8" standalone="no"	
TTGS	
College_Name	ACPCE
Department_Name	Computer Engg
Days_List	
Day	
Name	Monday
Day	
Day	
Day	
Day	
Hours_List	
Hour	
Name	8.30
Hour	
Hour	
Hour	
Hour	
Hour	
Hour	
Hour	
Hour	
Students_List	
Student	
Name	SEM VI
Strength	75
Group	
ID	0
Name	R1
Strength	20
Group	
Group	
Group	
Student	

Figure 4.1: Computer.ets.xml - XML Knowledgebase

# Chapter 5

## Implementation

### 5.1 User Interface Code

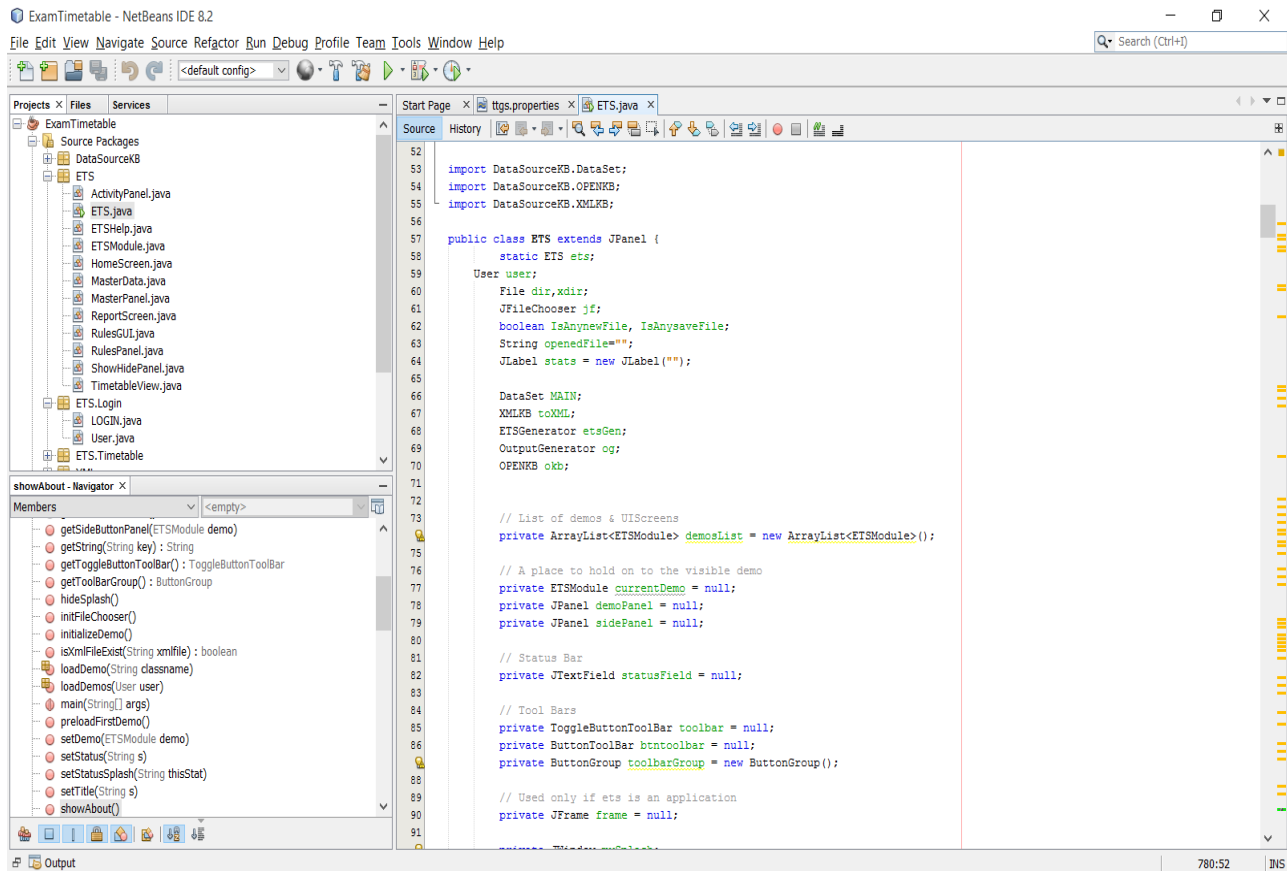


Figure 5.1: ETS Main Class

The Figure.5.1 shows main class ETS, which can be seen as entry class.



The following Figure.5.2 shows Matrix3D class, which can be viewed as Template Class. Template class can take many forms, i.e. we can instantiate this class by any data type (such as String, Integer, Boolean).

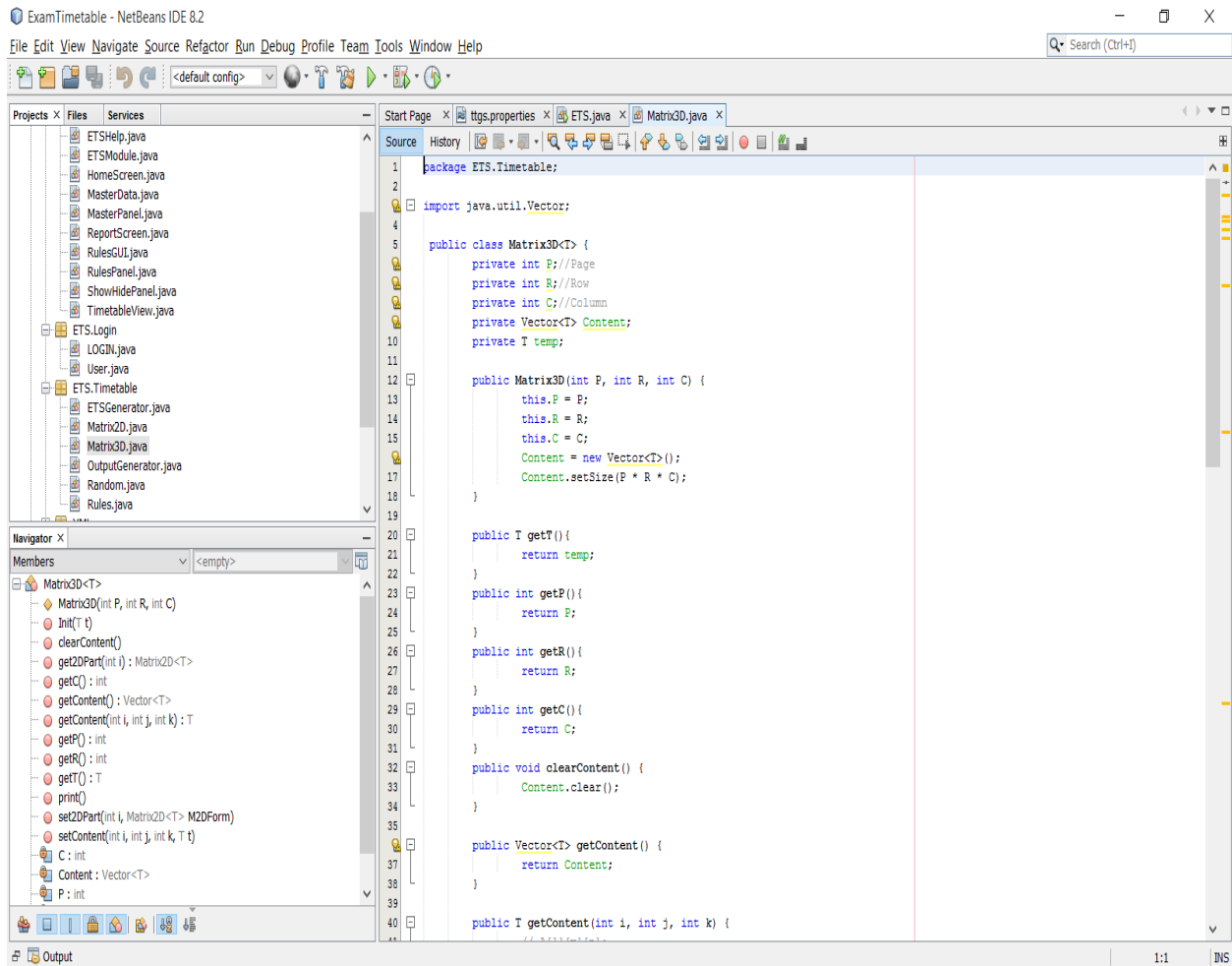


Figure 5.2: Matrix3D Class

## 5.2 Output Generator Code

The following Figure.5.3 shows how the generated timetable can be represented in HTML.

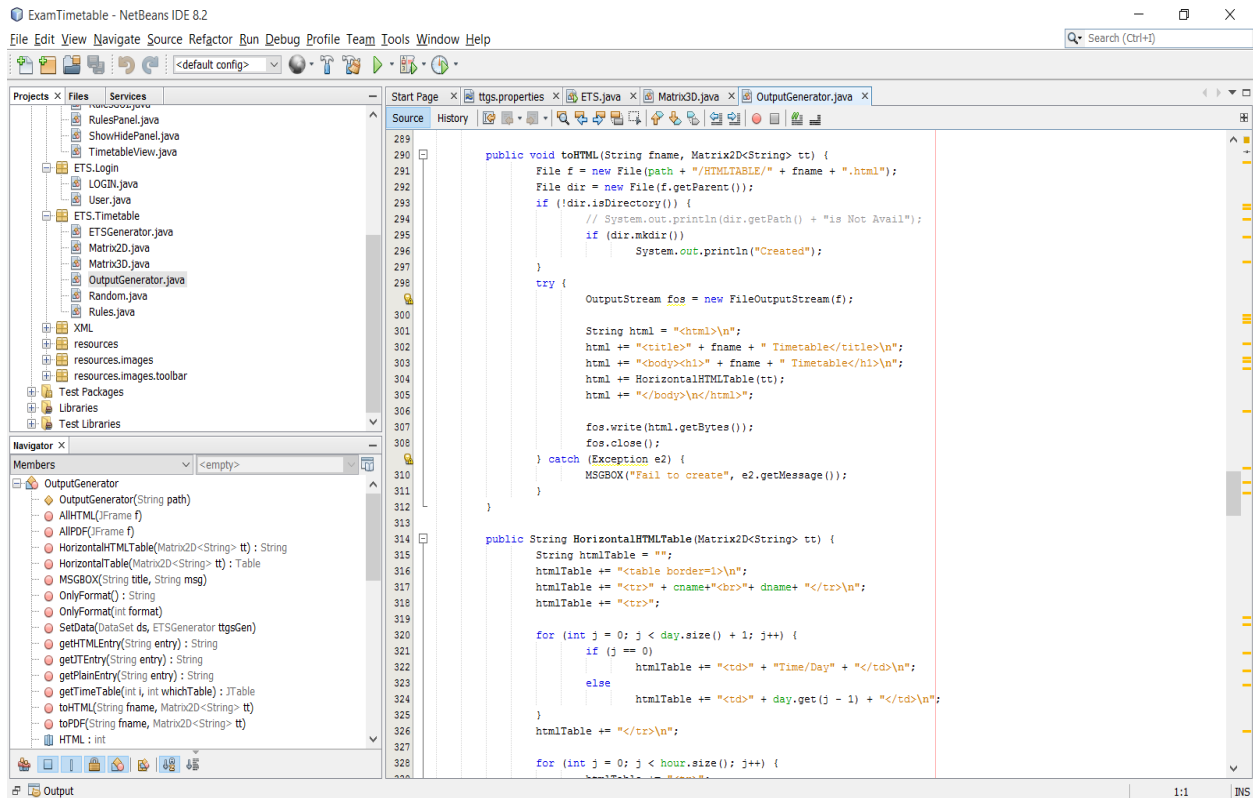


Figure 5.3: Output Generator - Export to HTML file

## 5.3 Timetable Generator Code

The following Figure.5.4 shows the Main algorithm. This is heart of my project. This works in following way:

1. Initiate storage matrices
2. Perform allocation by checking TSR (Teacher, Student, Room) resources.
3. If any resource is not available, then it exits else it places the activity current timeslot and day

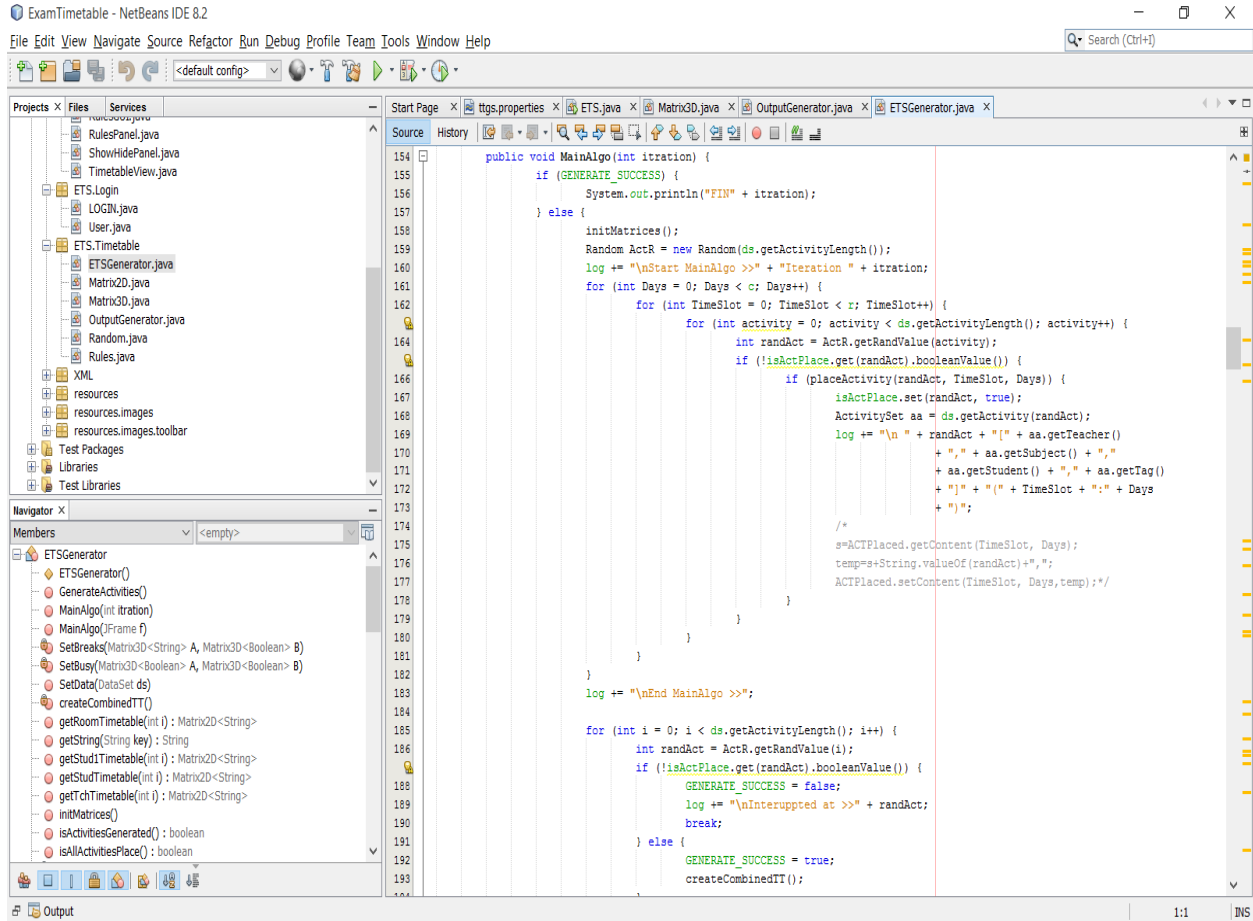


Figure 5.4: ETS Main Algorithm Code

The following Figure.5.5 shows how activities should be generated.

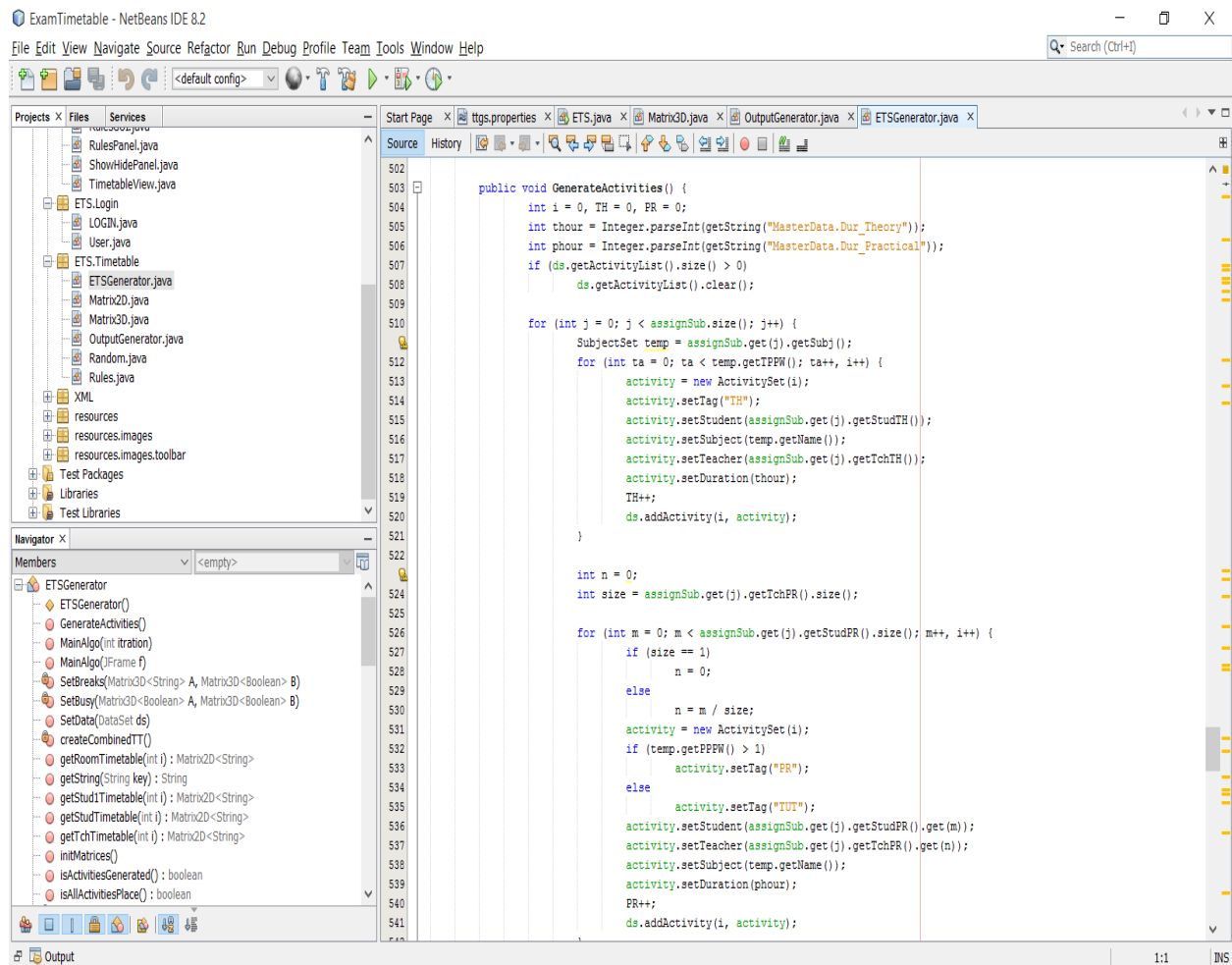


Figure 5.5: ETS Activity Generation

The following Figure.5.6 shows how the particular activity can be placed at (Day, Timeslot). It is called in Main Algorithm.

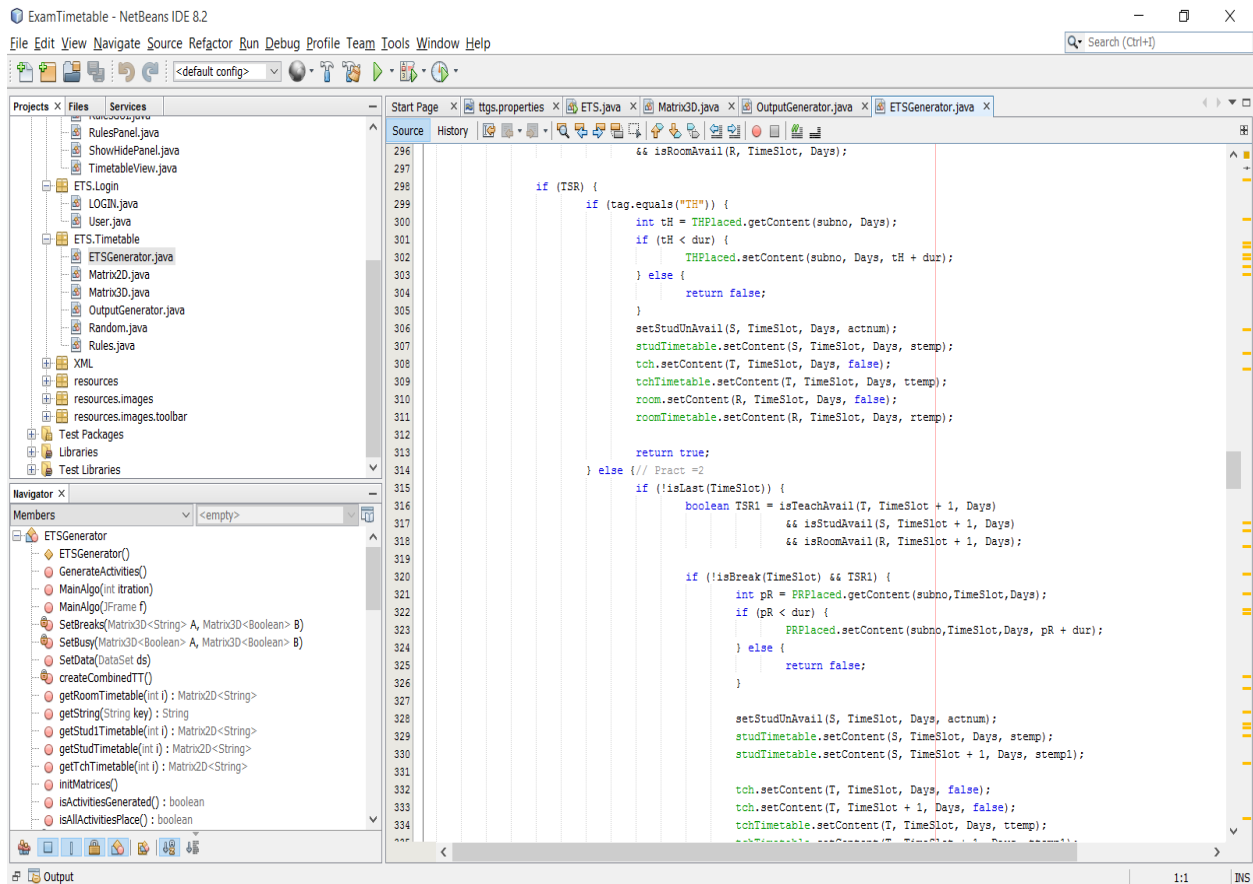


Figure 5.6: ETS Placement of activity

# Chapter 6

## Testing and Results

### 6.1 Screen Shot of ETS

The Screen Shots of my project is as following Fig.6.1 to Fig.6.13

As GUI is main entry point to any software application, I also tried to make my Timetabling System to make it easy to interact with Naive Users as well as Sophisticated Users.

I have designed my ETS front-end in Java (Swing set), that's why I was able to make interfaces according to my ease.

#### 6.1.1 User Interface Screen

The following Figure.6.1 shows Home Screen which is entry point to my project (ETS Application).

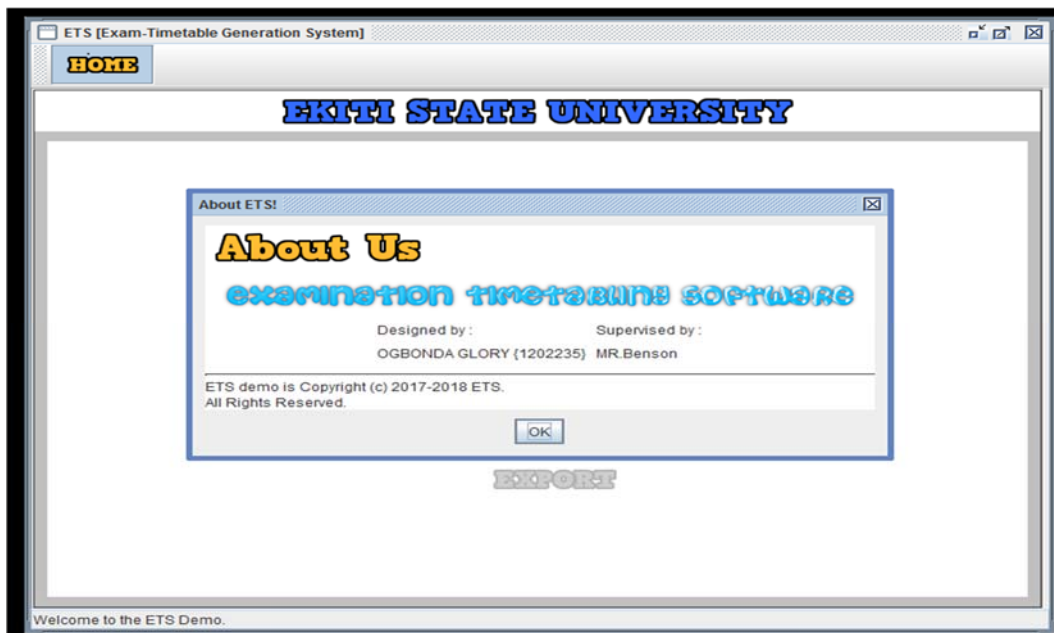


Figure 6.1: Home Screen

The following Figure.6.2 shows Login Screen which allow authorized access to ETS Application.

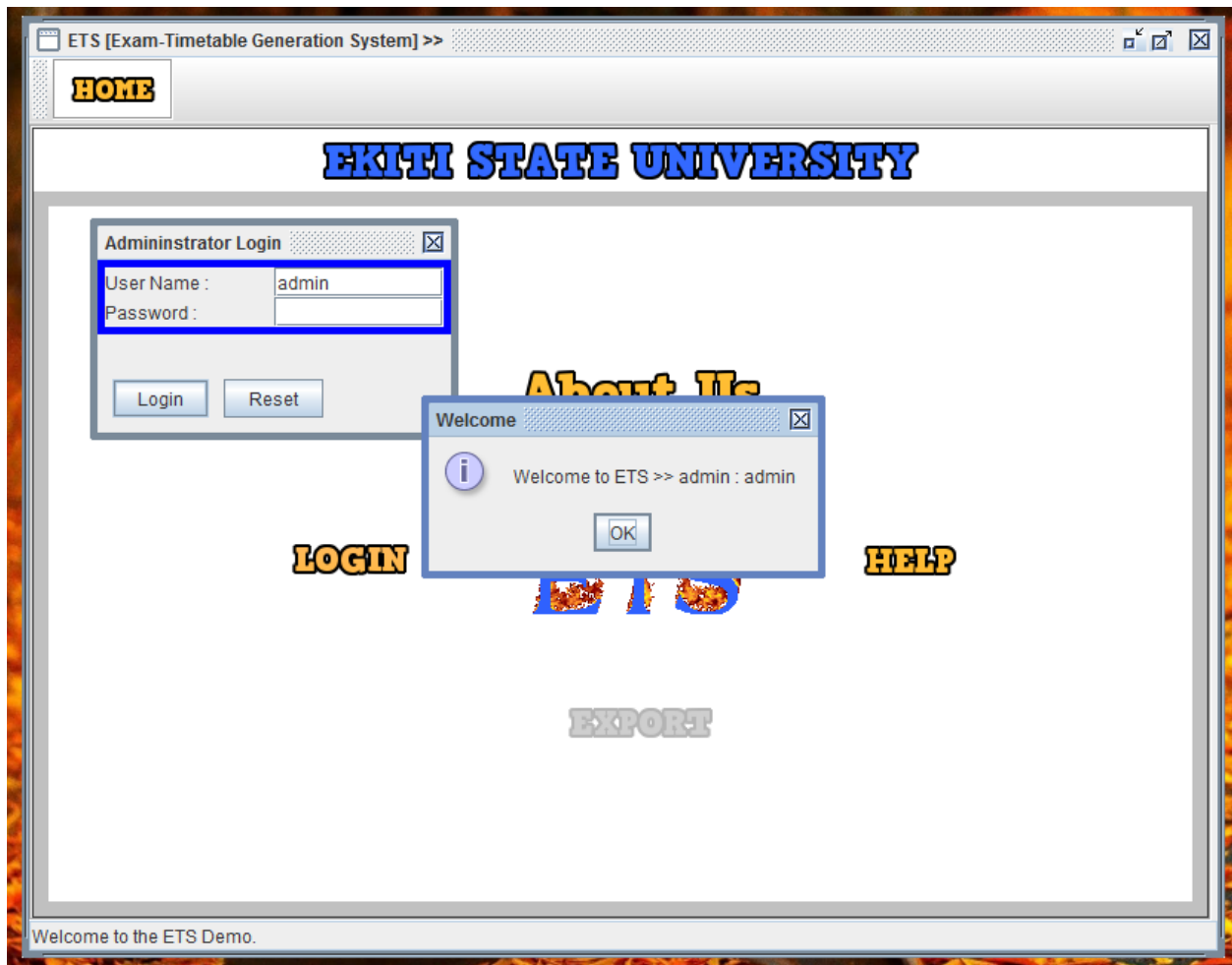


Figure 6.2: Login Screen

The following Figure.6.3 shows Teacher Manager Screen which provides the facility to enter all data entities as shown in fig.3.4 The other manager screen similar to this one e.g. Student, Subject, Room Manager Screens.

ETS [Exam-Timetable Generation System] >>

**HOME** **MASTER DATA** **TIMETABLES**

Step 1 >> Timetable Data Manager Step 2 >> Activity Manager Step 3 >> Rule Settings

Invigilator Manager Course Manager Student Manager Hall Manager Timeslot Manager

Invigilator Manager

Invigilator Name : Mr. Eric

Invigilator Type : Asst. prof

Add Invigilator Show

Invigilator Name	Invigilator Type
Mr. Benson	Prof.
Mrs. Regina	Prof.

Edit Invigilator Remove Invigilator Cancel

Invigilator Manager List :

0]Mr. Benson  
1]Mrs. Regina  
2]Mr. Alalibo  
3]Mr. Eric

Welcome to the ETS Demo. You are on MasterData

Figure 6.3: Teacher Manager Screen



The following Figure.6.4 shows Timeslot Manager Screen which provides way to set working days & timings.

ETS [Exam-Timetable Generation System] >>

**HOME** **MASTER DATA** **TIMETABLES**

Step 1 >> Timetable Data Manager Step 2 >> Activity Manager Step 3 >> Rule Settings

Invigilator Manager Course Manager Student Manager Hall Manager Timeslot Manager

Days :

Day 1: Monday Day 2: Tuesday

Day 3: Wednesday Day 4: Thursday

Day 5: Friday Day 6:

No. of Days/Week : 5

No. of Hours/Day : 9

Add

Hours :

Hour 1: 7:30-8:30am Hour 2: 8:30-9:30am Hour 3: 9:30-10:30am

Hour 4: 10:30-11:30am Hour 5: 11:30-12:30pm Hour 6: 12:30-1:30pm

Hour 7: 1:30-2:30pm Hour 8: 2:30-3:30pm Hour 9: 4:00pm

Hour 10: Hour 11: Hour 12:

New Open Save Close

Welcome to the ETS Demo. You are on MasterData

Figure 6.4: Timeslot Manager Screen

The following Figure.6.5 shows Course-Student Assignment Screen which provides administrator to assign the set of Course to Students.

ETS [Exam-Timetable Generation System] >>

**HOME** **MASTER DATA** **TIMETABLES**

Step 1 >> Timetable Data Manager Step 2 >> Activity Manager Step 3 >> Rule Settings

Select an operation Add, Remove, Modify the activity from following list

New

### Course - Student Assignment

Subject Settings :  
Select a course and an activity tag from given:

Theory Java567

Duration 1

Activity/Week 5

Student List :  
Select a student set of students from following list:

100 Level  
200 Level  
300 Level  
400 Level

300 Level

Clear

Subjects List :

[Subject : Java101][TH: [Teacher : null][Students : 100  
[Subject : Java102][TH: [Teacher : null][Students : 200  
[Subject : Java103][TH: [Teacher : null][Students : 200  
[Subject : Java1056][TH: [Teacher : null][Students : 30  
[Subject : Java254][TH: [Teacher : null][Students : 400  
[Subject : Java345][TH: [Teacher : null][Students : 100  
[Subject : Java567][TH: [Teacher : null][Students : 300  
[Subject : Java5670][TH: [Teacher : null][Students : 10

Subjects Description :

Subjects Details:  
=====

Subject : Java567.  
TH:  
Teacher : null.  
Students : null.  
PR:

**Subject-Student Assignment** Assign It


Figure 6.5: Course-Student Assignment

The following Figure.6.6 shows Course -Teacher Assignment Screen which allows to assign the Course to Teacher.

ETS [Exam-Timetable Generation System] >>

**HOME** **MASTER DATA** **TIMETABLES**

Step 1 >> Timetable Data Manager Step 2 >> Activity Manager Step 3 >> Rule Settings

Select an operation Add, Remove, Modify the activity from following list:  New

**Course - Invigilator Assignment**

Subject Settings :  
Select a course and an activity tag from given:

Theory  Java5670

Duration

Activity/Week

Teacher List :  
Select an invigilator/set of invigilator from following list:

Mr. Benson Mrs. Regina Mr. Alalibo Mr. Eric	Mr. Alalibo
--	-------------

Subjects List :

[Subject : Java101][TH: [Teacher : Mr. Benson][Stude  
[Subject : Java102][TH: [Teacher : Mr. Benson][Stude  
[Subject : Java103][TH: [Teacher : Mr. Benson][Stude  
[Subject : Java1056][TH: [Teacher : Mrs. Regina][Stud  
[Subject : Java254][TH: [Teacher : Mr. Alalibo][Student  
[Subject : Java345][TH: [Teacher : Mr. Eric][Students :  
[Subject : Java567][TH: [Teacher : Mrs. Regina][Stude  
[Subject : Java5670][TH: [Teacher : Mr. Alalibo][Stude

Subjects Description :  
Subjects Details:  
=====

Subject : Java5670.  
TH:  
Teacher : null.  
Students : 100 Level.  
PR:

**Subject-Invigilator Assignment**

Figure 6.6: Course -Teacher Assignment

The following Figure.6.7 shows Activity Manager Screen which allows to generate activities based on Teacher-Student-Course Assignment.

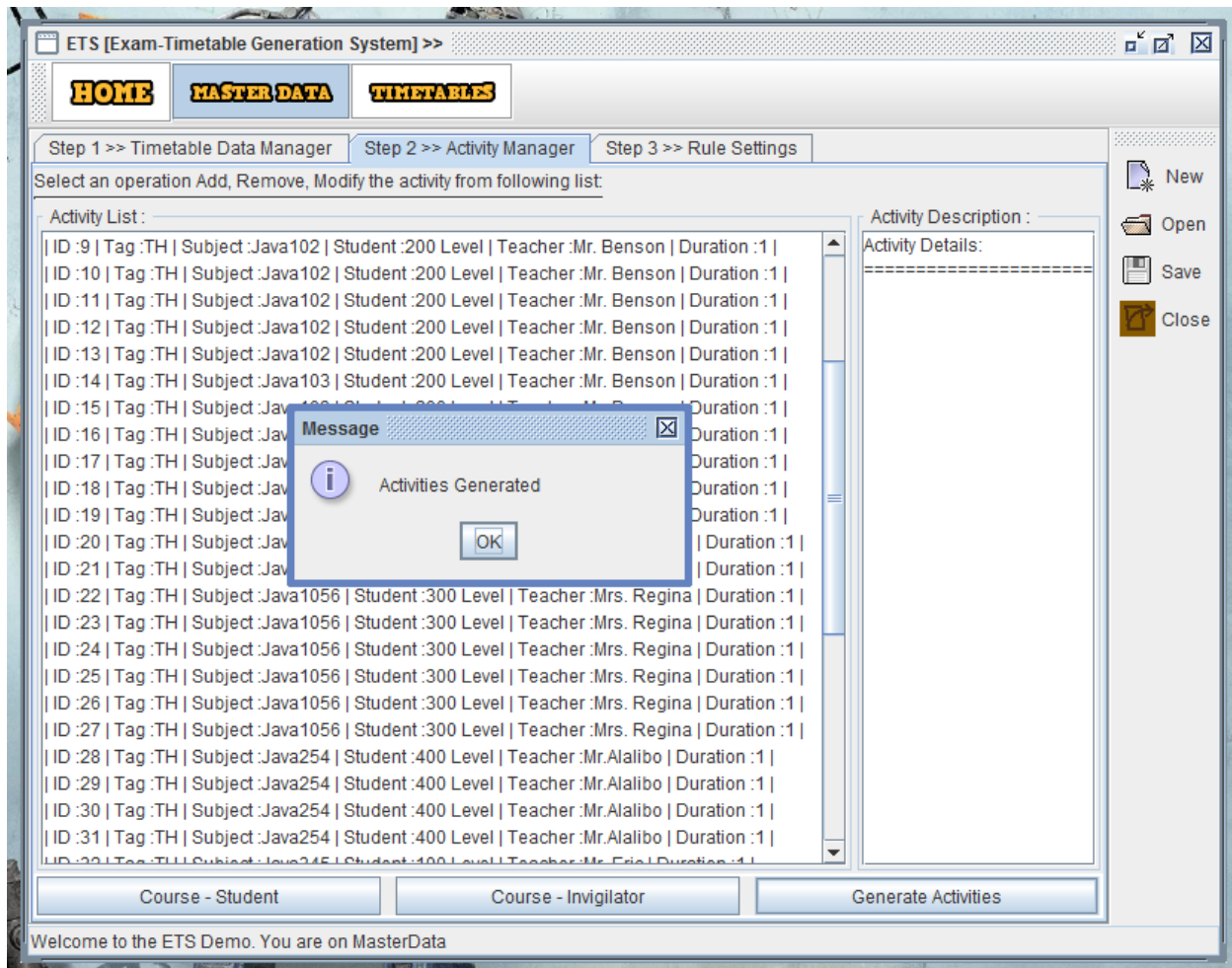


Figure 6.7: Activity Manager Screen

The following Figure.6.8 shows Rules Manager Screen which allows to set the soft constraints such as Break Time, Teacher Availability, Student Availability etc.

ETS [Exam-Timetable Generation System] >>

HOME MASTER DATA TIMETABLES

Step 1 >> Timetable Data Manager Step 2 >> Activity Manager Step 3 >> Rule Settings Step 4 >> Generate Timetable

## Rules List:

☒ Set University Name

NIIT Set University

Set University Name

By User set to : Na

☒ Set Department Name

Software Engineering Set Department

Set Department Name

By User set to : Na

- ☐ Hall or Lab is not assigned to more than one Activity at the same timeslot
- ☐ Invigilator is not assigned to more than one Activity at the same timeslot
- ☐ Students are not assigned to more than one Activity at the same timeslot
- ☐ Hall for an Activity should have enough capacity to fit No.of Students
- ☐ Basic compulsory time constraint
- ☐ Basic compulsory space constraint
- ☐ Break time constraint
- ☐ Minimum interval between Activities

New Open Save Close

Welcome to the ETS Demo. You are on MasterData

Figure 6.8: Rules Manager Screen

The following Figure.6.9 shows Timetable Generation Screen which allows to generate timetable. If timetable successfully generated, then it shows success message else you would get a non-success message as shown below. If you want different output, then you have to regenerate it.

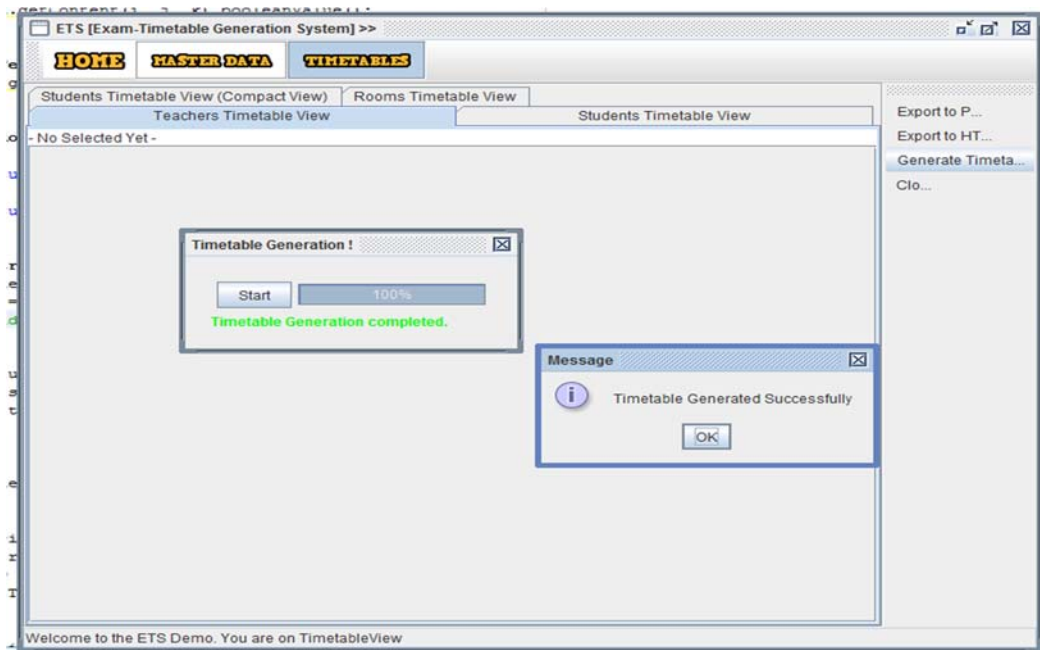
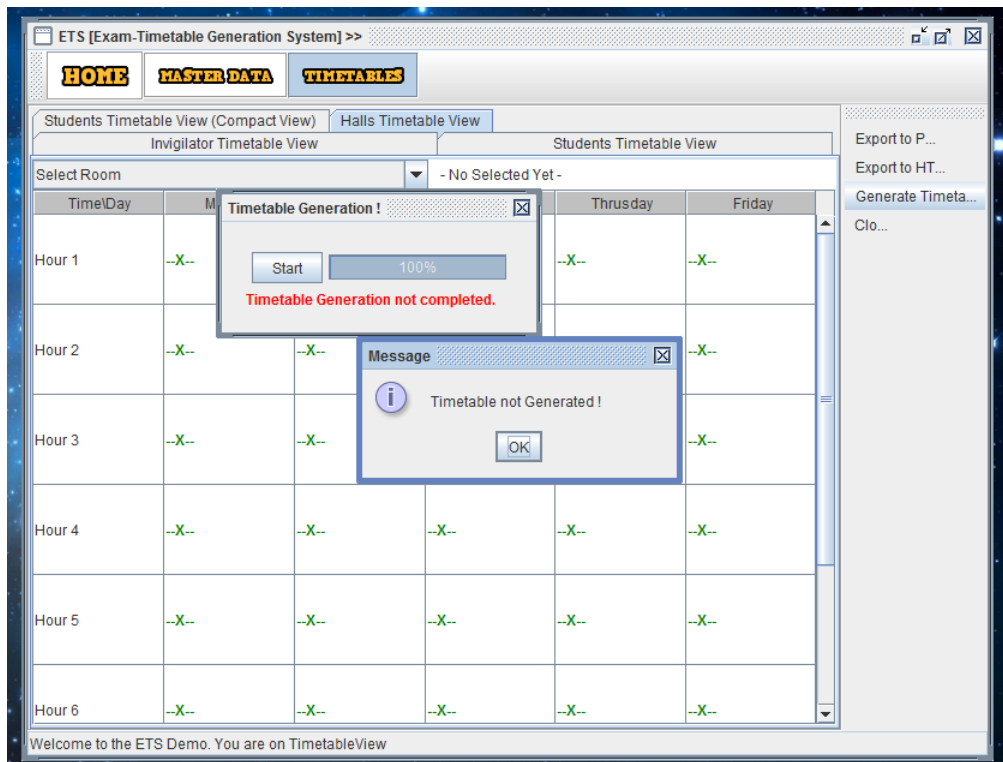


Figure 6.9: Timetable Generation

## 6.1.2 Timetable View Screen

The following Figure.??, shows various Timetable views which shows timetables, if generated.

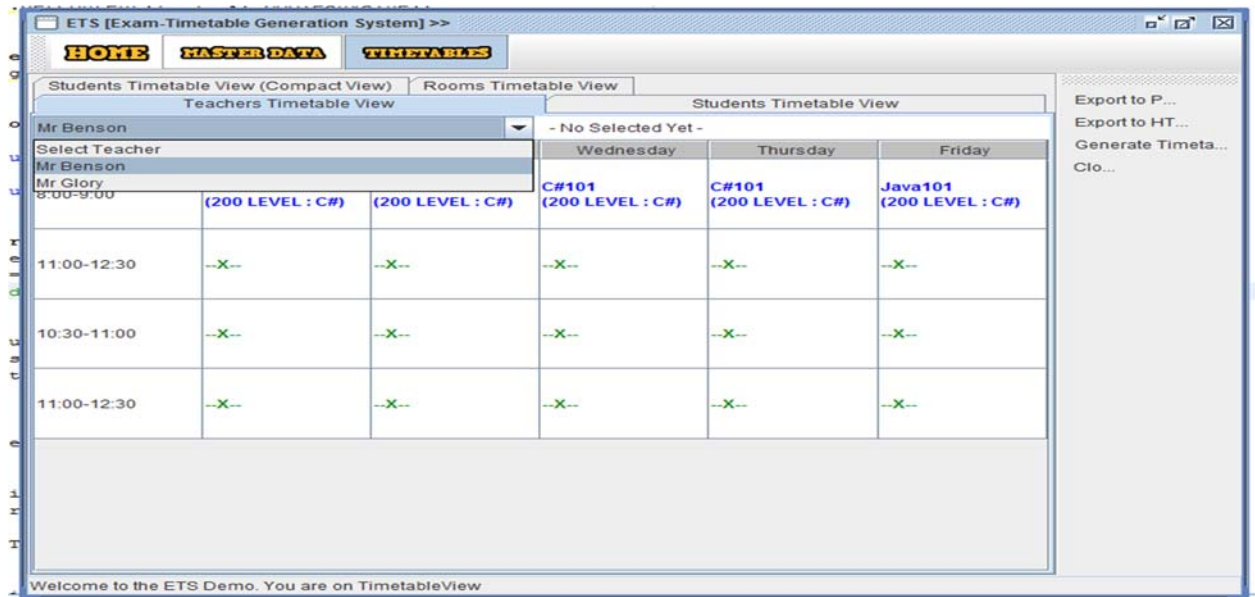


Figure 6.10: Teacher Timetable View

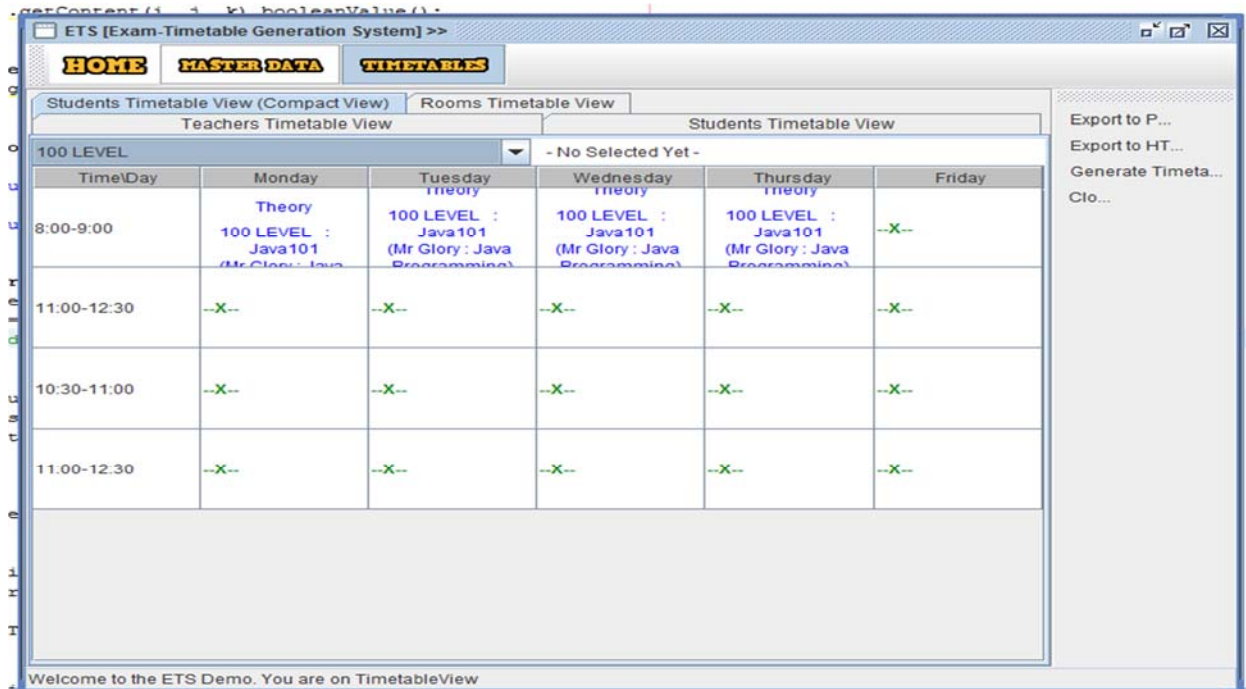


Figure 6.11: Student Timetable View



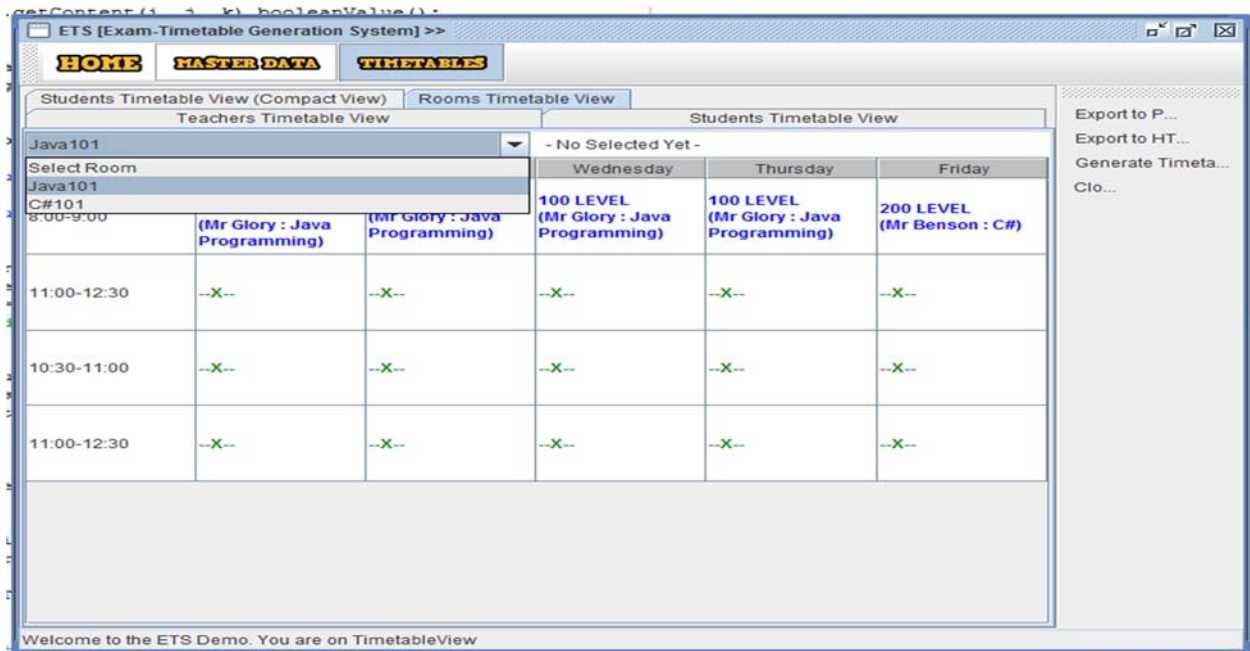


Figure 6.12: Room Timetable View

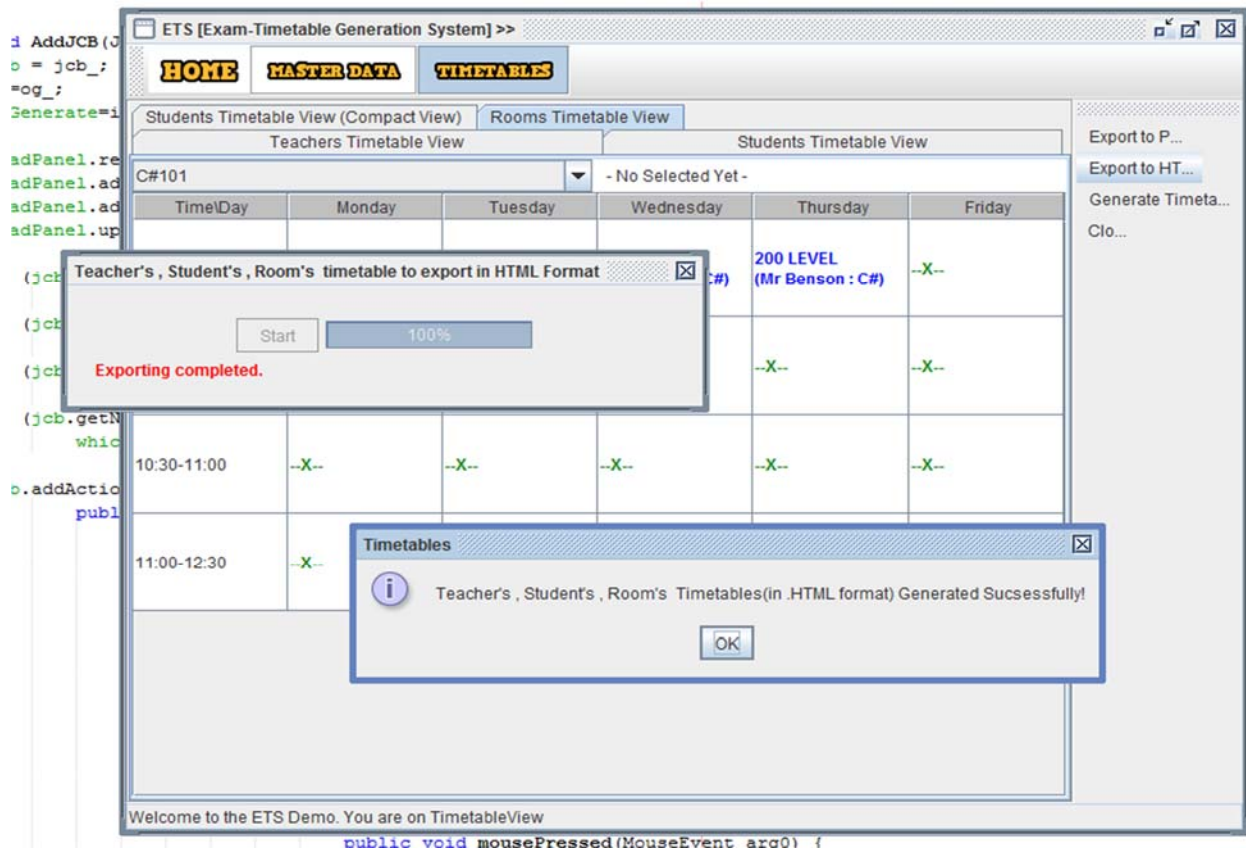


Figure 6.13: Timetables Export to HTML



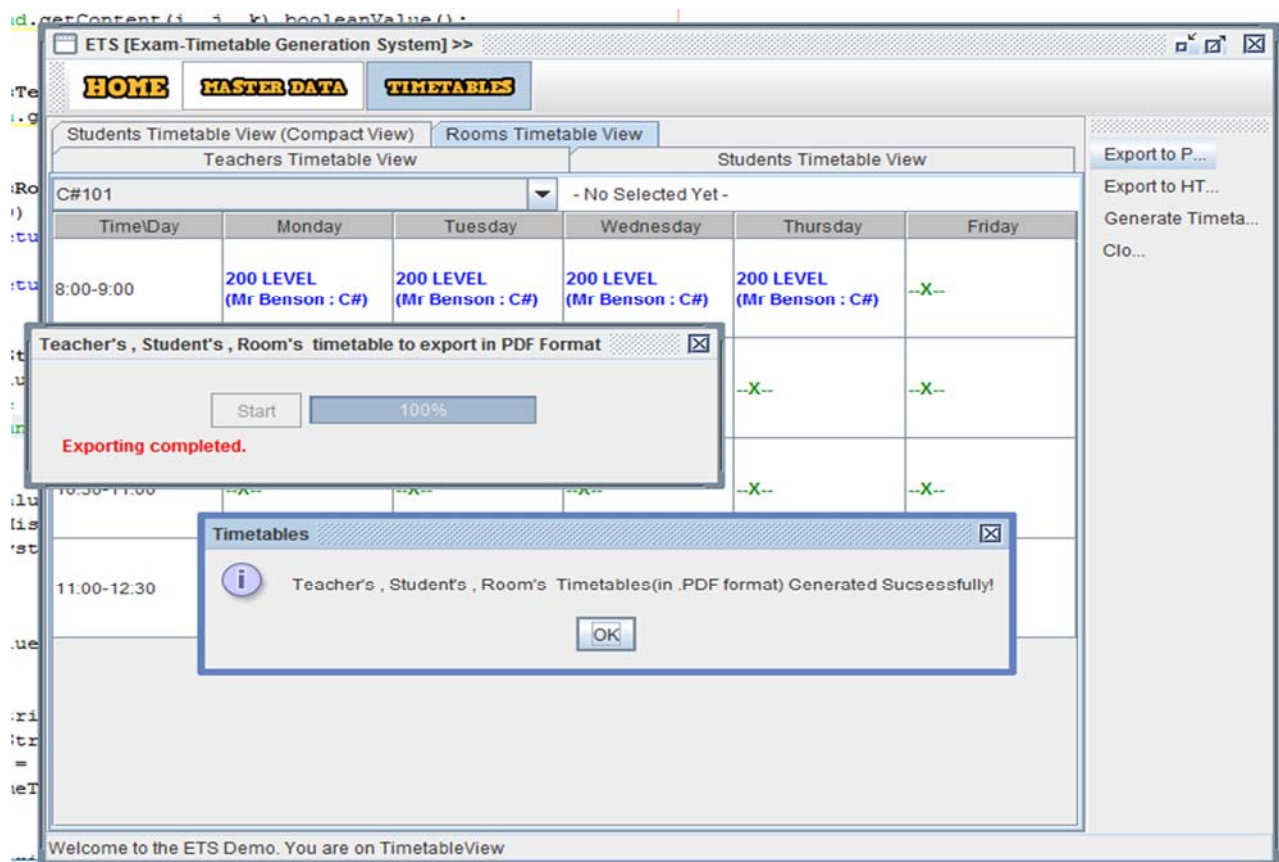


Figure 6.14: Timetable Export to PDF

# Chapter 7

## Conclusion

In this **Phase** of my project, I used JAVA as front-end & XML as back-end as I decided. I came across many difficulties and problems while implementing this software but my tremendous desire to complete my project, made me to overcome all difficulties and problems.

I can ensure that, this software will be used in other colleges.

At last I have implemented and successfully executed my project and now I am going to handover this software to my college. Thanks.