**Table of Contents**

[Persistence 2](#_Toc275996545)

[Code Conventions 3](#_Toc275996546)

[Implementation Status Report 4](#_Toc275996547)

[User Manual for RUM System 5](#_Toc275996548)

[Testing for Graphical User Interface 6](#_Toc275996549)

[Tests for Database System 7](#_Toc275996550)

[Team Contributions 8](#_Toc275996551)

# Persistence

**Database Using Persistence Framework**

For our RUM system we made our database using only the persistence framework. We currently have implemented our entire persistence framework database and it can be found in these files:

* Course.java
* Major.java
* Requirement.java
* CourseGroup.java
* SystemManager.java

All of these files are entities and with the JPA all necessary tables are created automatically.

**File Formats**

All information on file formats can be located on page 7 of the requirements document (or check below). It discusses the three different types of files that will store information.

The following types of files will be uploaded by users for data entry:

Course listing file

Major requirements file

Student info

Course listing file-This file is in CSV format and will be the file used by the super admin to upload course offering information. This file will need to maintain the following information about each course:

Course name

Meeting time (for scheduling)

Semesters offered

Prerequisites (as a non-comma delimited string)

This file be comma and line delimited.

Major requirements file- The major requirements file is in CSV format and will be used by department administrators to upload major requirements. It will be able to be edited by them after upload, and can be re-downloaded and saved. The following data will need to be maintained by the file.

Major name

Minimum GPA

Required Courses

Minimum Grade

Student info file- This file is in CSV format and will be used by the student to upload personal data and course data into the system. The information that will need to be contained in this file is:

Name

Major

Class name

Grade

The student can edit this file after it is uploaded, and the student will be able to re-download this file. It should be in comma separated format.

# Code Conventions

We have followed and plan to continue using the Java Sun code conventions. These can be found at http://java.sun.com/docs/codeconv.

# Implementation Status Report

This document will contain currently implemented features for our RUM system (requirements and design) as well as any bugs or limitations known.

**Fully Implemented Features**:

* Graphical User Interface Layout
* Persistence Database

**Requirements Satisfied/Partially Satisfied**:

* Currently we have not begun to implement specific requirements for our RUM system although our Graphical User Interface contains all needed functions to easily and quickly implement our requirements.

**Design Implemented/Partially Implemented**:

* Graphical User Interface:
  + All screens included in our design are accounted for.
  + All buttons correctly link and change to the associated screen chosen.
  + All screens contain the tables and buttons needed to fully implement all requirements in our design (as mentioned above).
* Persistence Database:
  + Created using ApacheDerby and done through NetBeans.
  + Contains all needed tables with appropriate attributes and primary keys included.
  + All code for the database can be found in the server folder of the submitted files.

For our project we have estimated that we have approximately implemented 25-30% of our project. We came to this conclusion based off the amount of bulk code that the GUI required and the coding of the persistence database. We feel we have set a very good framework for implementing all of our requirements quickly. Overall we have implemented the front and back end of our system with only the middle missing. The major missing part of the system are the algorithms for certain project requirements.

# User Manual for RUM System

This will be a basic guide using our RUM system including: (1) installing the necessary components to your computer; (2) running the server for the application program; and (3) starting up the client program on your computer.

This system is a client/server application that checks requirements for a student’s graduation. It allows students to upload/add courses they have taken including their grades and if they are transfer credits. They can then check these classes against a list of requirements for whatever major they have selected. Then based off what requirements they have not met the system can generate a suggested schedule of classes they can take to graduate. The system also allows for user admins to log in, add/edit/remove majors and departments, add/edit/remove requirements for majors and upload/add course offerings for a school.

**Installing/Configuring/Starting Rum System on Server Side**

To use this system the user must have the following on their computer:

* Apache Derby – See http://db.apache.org/derby/derby\_downloads.html

**Running the Server Program.** User must run this before using the client.

1. Download the server .JAR executable file.
2. Double click the server .JAR executable file.

You can now proceed to running the client application.

**Installing/Configuring/Starting Rum System on Client Side**

To use this system the user must have the following on their computer:

* Java Runtime Environment – See <http://www.java.com/en/download/index.jsp>

**Running the Client.** For the client side there is no actual install on the computer.

Note: User must have the associated server running before using the client.

1. Download the client .JAR executable file.
2. Double click the client .JAR executable file.

You can now begin to use the RUM system.

# Testing for Graphical User Interface

**Student Run Through**

When we first start the application we correctly load our welcome screen. If we begin by clicking the Student option we change screens to the Student page. From there we have multiple options. If we click the add course button we navigate to the add course page. On that page we have a drop down menu to select a grade and a transfer check box. As of now both the ok and cancel button brings us back to the student page. If we click on the edit course button next from the student page we have the same screen as add screen page except the page says edit course page. The remove course button on the student page does not do anything but when it is implemented it will directly remove the highlighted course from the table on the student page. The upload course and download courses as of now do nothing but they will allow the user to download or upload courses from a specific file format once it is implemented. The check requirements and generate schedule also currently do anything.

**Department Admin Run Through**

If we hit back from the student page we go back to the welcome screen. From here we can click on department administrator. This brings us to the log in page where we must enter a username and password and then click log in. As of now we can simply log in regardless of the input but in the future we will check the user name and password. The back button brings us back to the welcome screen. If we click on log in we are brought to the Department Administrator Page. The add major button pops up a dialogue box where we can enter a major. As of now ok and cancel both bring you back to the department administrator page. Edit major button pops up a dialogue box as well where you can enter the major you wish to change. Remove major button does nothing as of now but it should automatically remove a major from the table on highlighted major on the screen. The edit requirements button brings up a new page called requirements. From here we can add remove or edit. Clicking back brings us back to the department administrator page. From here if we click back again we can go back to the welcome screen.

**Registrar Admin Run Through**

We now will choose are last option from the welcome screen which is registrar administrator. We will go through the same login screen which will bring us to the registrar admin page. From here we have three options. The first option is clicking view edit courses button. This brings us to the edit department page. Once again we have several options. Add course brings us to the add course page. In this page there is a bunch of information that can be typed in such as name, number, department, and description. As of now we don’t actually save anything so ok and cancel do the same thing (bring us back to edit department page). The remove course button will delete a course from the table in the future. Currently the edit course button is the same as the add course button with just a different title. Upload courses will allow the user to import a file in the future by clicking the browse button.

# Tests for Database System

**Test 1:** Add Major to Database

This tests whether or not data for majors can be added to the database.

Precondition: User must be logged in as admin(either) and major has valid information in all required fields.

1. Major information is read into system.
2. JDBC inserts the data into the database table through the JPA.

Requirements successfully associated with major.

Outcome: Test was successful. Data inserted into correct table, objects associated with correct major.

**Test 2:** Add Course to Database

This tests whether or not data for courses can be added to the database.

Precondition: User must be logged in as a Registrar Admin and course has valid information in all required fields.

1. Course information is read into system.
2. JDBC inserts the data into the database table through the JPA.

Outcome: Test was successful. Data inserted into correct table, objects associated with correct course.

JUnit Tests

# Team Contributions

William Matrix Peckham

* Created the power point for the design review presentation.
* Coded the GUI in part with John Paul Pennisi.

Chris Scarola

* Spoke for 20% at the design review presentation.
* Wrote the implementation status report.
* Wrote the user manual.
* Edited and formatted all parts of this document.

Tom Biegner

* Spoke for 80% at the design review presentation.
* Created the database using persistence framework.
* Created and used self made testing for the database.

John Paul Pennisi

* Coded the GUI in part with William.
* Wrote the paragraph summary for the GUI testing.

John Paul Pennisi

William Matrix Peckham

Chris Scarola

Thomas Beigner

Homework 6: Persistence and Code 1