University Management and Scheduling Application

Advisor: Prof Mark Grechanik, Ph.D.

(Requirements Specification Document)

Simant Purohit
Akshay Thirkateh
Prasad Nair
TABLE OF CONTENTS

1. Introduction
   1.1 Purpose of the system
   1.2 Scope of the system
   1.3 Objectives and success criteria of the project
   1.4 Definitions, acronyms, and abbreviations

2. Current system

3. Proposed system
   3.1 Overview
   3.2 Functional requirements
   3.3 Nonfunctional requirements
      3.3.1 Usability
      3.3.2 Reliability
      3.3.3 Performance
      3.3.4 Supportability
      3.3.5 Fault Tolerance
      3.3.6 Interface
      3.3.7 Packaging
      3.3.8 Legal
   3.4 System models
      3.4.1 Use case model
      3.4.2 User interface (screen mock-ups)
1 Introduction

1.1 Purpose of the system

Our motive is to develop ‘University Management and Scheduling’ application that mainly focuses on developing an application which would be used not only by the students but the professors as well. This application covers the areas of course scheduling, on-campus position selection processing and tools used by the professors. The aim is to develop a better system that is promising, reliable and maintainable. Appropriate procedures will be taken keeping in mind the properties required to develop good software and to safeguard the data and its integrity. Its purpose is broadly classified into three diverse implementable areas which would be individually developed as independent modules and then collaborated together as a complete application. An overview of each of the modules is given as follows.

Course scheduling is one of the main prominent features for any university. The idea is to develop a system that lists all the available courses offered in each semester. It would allow the department to add or remove an offered course from the list and allows students to register for them. If the course has reached its limit, then automatically it would add the interested students into a waiting list and automatically sends a mail when there is a vacancy. A system to put/ remove holds on the courses that are open to some/ all students.

Another Category would be on-campus position selection processing. A clear procedure that would select the students based on their skillset for any job opening. Any student that is eligible to apply for a position is made to fill in an application form. A professor/ department posts a job opening. In accordance to the requirements for the position, a list of suitable candidates is generated automatically and then a mail is sent to them for further procedures. This system is mainly intended to be used only for GA, TA, RA positions but can be extended to any related full time/ part-time jobs.

A set of tools for the professor to make their teaching and related duties easier and better is also a very important aspect that is incorporated here. We plan to devise a tool that calculates the grade based on the cut off criteria given by the professor. A system that compares students’ marks and gives a corresponding grade for each exam. This tool also holds the capability that if the professor needs to send course documents and other pertaining information to his class he can do so with this.

1.2 Scope:

The scope of the project is to deliver a high end application which would pragmatically serve many major tasks in a university such as course scheduling, selection of applicants for part time or full time jobs and a number of tools which can be used by the departments or professors. This system would act as a better bridging tool between students, professors and the department also.

1.3 Objectives and success criteria of the project

The objective of the system is to build a system that successfully implements all the aimed functionalities. The system thus built should be easy to maintain and should be flexible enough to add/remove more functionalities as and when required. The aimed system being a application for managing a university should maintain high levels of integrity of data.
1.4 Definitions, acronyms, and abbreviations

University admin: This is a term used to refer to the admin who manages all the different departments in the university and schedules all the courses in the departments among others.

Department admin: Department admin refers to the admin who manages one single department in the university, managing activities like adding courses, assigning professors to courses etc.

Department: A university consists of various departments. But here we use the generalized meaning for the word as every department functionally is the same and also it’s duties and responsibilities too.

GA: This is a term used in reference to Graduate Assistant who assists the department in various technical tasks.

TA: This is a term used in reference to Teaching Assistant who acts as an assist to the professor for a particular course.

RA: This is a term used in reference to Research Assistant who works along with the professor on a research project.

2. Current system

The current system which we have come across as of now does not provide exceptional features and clear understanding for the various assistantship selection procedures. What we feel missing is a transparent process which filters the applicants based on the self-entered information.

When a student tries to register for a course and it is filled up to capacity there is nothing he can do about it in this current scenario rather than just waiting for anyone to drop it and it is opened for grabs by anyone without any preference to the student who just missed it.

The current system can extensively be improved with tools for the professors/ department which would be more efficient in terms of curve calculation, grade calculation and cutoff decision making. This is needed so as to have a clearer understanding about any calculations needed.

3. Proposed system

3.1 Overview

The system proposed aims at implementing features to help professors, student, and the university in itself to ease their workload in terms of managing various aspects of the university. The system aims at providing tools to professors that will streamline the calculation of grades and grading curve keeping in mind the preferences of the professor on how to grade. This system aims to develop grading algorithm to ensure fair grading. Tools for professor will also provide the functionality to communicate with students for announcements, sharing coursework materials etc.

The system will include tools for students to help them in registration for courses they wish to take. The student will be able to sign up on wait list if the class capacity is currently full. The wait list will act as a queue for students wishing to attend the course. The students will be notified by email as and when the seat is up for grabs.
Apart from tools for professors and students, the system will incorporate functionalities for departments in the university offering different courses to add, remove, modify course information. Course information will include parameters such as teaching professor, credits, duration etc.

The system will also include a top-level admin functionality which will be able to manage all the departments, professors and students.

The list of functionalities in not exhaustive and some functionalities might be removed or added depending upon the feasibility and necessity.

### 3.2 Functional requirements

#### 3.2.1 Use Cases

**Use Cases for User Authorization**

**3.2.1.1: Login**

Use case name: Login

Actor: System User

Flow of events:
1. The user starts the application.
2. The system presents the login window to the user.
3. The user enters its username and password details in the provided fields.
4. The system tries to authenticate the user.
5. If the user is authenticated, an appropriate interface is shown to the user.

Alternate flow of events 1:
1. At step 4 of the main flow of events, it may happen that the system is unable to authenticate the user.
2. The system then notifies the user of the unsuccessful login attempt and presents the login page to the user.
3. Now the main flow is repeated from step number 2.

Alternate flow of events 2:
1. The user is only allowed 5 unsuccessful attempts before the system locks for a certain (undecided) period of time before a re-attempt is allowed.

Entry condition: The user has started the system

Exit condition: The user has been successfully logged in or has been notified of the reason for the unsuccessful attempt.
3.1.2.2: Recover password
Use case name: Recover password.
Participating actors: System user.
Flow of events:
1. The user does not remember the password to its account.
2. The user can click on the ‘forgot my password’ button to recover password.
3. The user is asked for email id associated with the account so that the recovery code can be sent to the user.
4. The user is presented with a screen asking to enter the recovery code that was sent. The code has a timed validity and the screen will return to login screen after a specified time.
5. The user copies the recovery code from the email and pastes the code inside the recovery code box.
6. If the code is correct and not timed out, the system asks the user to enter new password.
7. The password is successfully changed after the user presses the accept button.
8. The system notifies the user of a successful password change and returns the user back to login screen.

Alternate flow of events 1:
1. At step 4 of main flow of events, if the user enters an invalid code/expired code for more than 2 times, the password recovery process is terminated and the user will be taken to login screen again.

Entry condition: The user has forgotten account password and wishes the recover the password.

Exit condition: The user successfully recovers account password or is notified of the reason for an unsuccessful attempt.

3.1.2.3: Change password
Use case name: Change password.
Participating actors: System user.
Flow of events:
1. The user is logged into its own profile.
2. The user can click on the ‘Change my password’ button to change password.
3. The user is asked for old password to verify the process.
4. If the old password is correct, the user is asked to enter the new password for the account.
5. The user should press confirm changes button to confirm the change in password.
6. The system notifies the user of a successful password change and returns the user back to login screen.

Alternate flow of events 1:
1. At step 3 of main flow of events, if the user enters an invalid old password for more than 2 times, the password change process is terminated and the user will be taken to login screen again.

Entry condition: The user is logged in and wants to change the account password.
Exit condition: The user successfully changes account password or is notified of the reason for an unsuccessful attempt.

Use Cases for Professor Tools

1. Grade It

3.2.1.4: Add exam/quiz to the course (Grading criteria)

Use case name: Add exam/quiz to the course
Participating actors: Professor
Flow of events:
1. The professor is logged into the system.
2. The professor enters the ‘grade it’ tab.
3. The system opens the tools presenting the list of exams currently added to the course. The professor chooses to add a new grading criteria to the course by pressing ‘add new criteria’ button.
4. The system presents a form to the user asking for details about the grading criteria.
5. The professor enters the details about the criteria and presses submit button.
6. The system adds the new criteria to the course and returns to the list of criteria.

Entry condition: The professor is logged in to the system and is associated with the course.

Exit condition: The professor successfully adds the new grading criteria to the user or is given an appropriate notification about what went wrong with the process.

3.2.1.5: Add marks

Use case name: Add/Change marks.
Participating actors: Professor, TA
Flow of events:
1. The professor/TA is logged into the system.
2. The professor/TA opens the ‘grade it’ tab.
3. The system displays the list of criteria of evaluation that currently exist in the course.
4. The professor/TA press the ‘Add marks’ button along the criteria top add/edit marks.
5. The system presents the user with the name of all the students currently taking this course along with the marks added (if any).
6. The user can add the marks alongside each students’ name.
7. The user presses the submit button to complete the process and save changes.
8. The system returns back to the list of criteria.

Entry condition: The professor/TA is logged in to the system and is associated with the course.
Exit condition: The professor/TA successfully adds/edits the marks associated with grading criteria.

2. Curve It

3.2.1.6: Calculate curve

Use case name: Calculate curve.
Participating actors: Professor
Flow of events:
1. The professor enters the ‘curve it’ tool.
2. If any previous curve calculation for the associated course exists, the system presents it to the user.
3. The user can now either press calculate curve or recalculate curve button to start the curving process.
4. The system presents a form to the user so that the user can enter curving parameters.
5. The user now presses the submit button and the system starts calculation of curve along the given parameters.
6. The system presents the new curve along with details of what the new grades for each student are.
7. The user presses the ‘apply new grades’ button to confirm the grading curve. The user also has the option of pressing cancel to keep the old grading curve.

Entry condition: The professor must be logged into the system and there must be at least one grading criteria defined.

Exit condition: The professor successfully calculates the grading criteria and saves it to the system or is given an appropriate message about what went wrong.

3. Share It

3.2.1.7: Share course documents

Use case name: Share course documents

Participating actors: Professor, TA

Flow of events:
1. The user logs in the system and enters the ‘share it’ tab.
2. The system presents an interface to the user with the facility to attach files and add notes.
3. The user adds appropriate documents and presses send button.
4. The system automatically adds all the students associated with the course to the mailing list and send an email to each student.
5. The system provides a confirmation to the user.
6. The system returns back to the user’s home screen.

Entry condition: The user is logged in and is associated with at least one of the courses.

Exit condition: The user can successfully share files with students or is given a appropriate notification about where the process faltered.
3.2.1.8: Search for courses
Use case name: Search for courses.

Participating Actors: System User: Student

Flow of events:
1. The student first enters the course search tab.
2. The student is given option to search for courses by department or by name.
3. The student presses the search by department button.
4. The student is presented with the list of departments which offer courses.
5. After selecting the department, the student is presented the list of courses offered by the department.
6. The student can go back and forth and select different departments for its search.

Alternative flow of events:
1. The student first enters the course search tab.
2. The student is given option to search for courses by department or by name.
3. The student presses the search by name button.
4. The student types the name of the course in the search box and presses the search button.
5. The student is presented with the courses matching the search query.
6. The student can type a different search name and get other results.

Entry Condition:
Student must be logged in to the system.

Exit Condition:
The student is able to see the list of courses searched for.

3.2.1.9: Register for courses
Use case name: Register for courses.

Participating Actors: Student

Flow of events:
1. The student searches for courses either by department or by name and has a list of courses displayed.
2. The student selects a course from the list.
3. The system displays the course details to the student. The details of the course also includes the current capacity and the filled capacity of the class.
4. If the class has remaining capacity and the student is eligible to register, the register for class button will be enabled and student can press the button to register.
5. The request for the registration is processed and the student is registered for the course.
Alternate flow of events 1:
1. From step 4 of the main flow of events, if the class capacity is full, the register button will be disabled and the ‘add to wait list button’ will be enabled.
2. The student can add self to the wait list for the current course if the eligibility criteria is met by the student.
3. After pressing add to wait list button, the student is added to the wait list for the course.

Alternate flow of events 2:
1. From step 4 of the main flow of if the student is not eligible at all to register for class, the buttons for registration and waiting list will be disabled.
2. The student can now go to a different course where registration is possible.

Entry Condition:
Student must be logged in to the system.

Exit Condition:
The student has either being able to register for the course or the waiting list or is shown proper message regarding the eligibility criteria.

3.2.1.10: Dropping courses/ removing from wait list.

Use case name: Drop course

Participating Actors: System User: Student

Flow of events:
1. The student enters the ‘My courses tab’.
2. The current courses on the registration and the waiting list of student is displayed to the student.
3. The student can select the drop course checkbox for the courses which are supposed to be removed either from main registration or the waiting list the student is subscribed to.
4. The student now presses the save changes button to confirm dropping of courses.
5. The system presents a summary of changes to be made and asks the student to confirm the changes.
6. The student can press the confirm changes button and the changes will be applied to its profile.
7. The ‘My courses tab’ is displayed again to the student.

Alternative flow of events 1:
1. At step 4, the student can press cancel changes button.
2. The system will ignore all the checkbox for dropping courses and refresh the ‘My courses tab’ without applying any changes.

Alternative flow of events 2:
1. At step 6, the student can press cancel changes button.
2. The system will ignore all the selected courses for dropping and refresh the ‘My courses tab’ without applying any changes.

Entry Condition:
Student must be logged in to the system and inside the my courses tab

Exit Condition:
The changes requested by the student have been applied.

3.2.1.11: View grades

Use case name: View grades

Participating actors: Student

Flow of events:
1. The student enters the ‘my grades’ tab.
2. The student is shown the list of courses the student is currently registered for or was previously registered. The system also displays the current cumulative GPA for the student excluding the current courses taken.
3. The student clicks on the course for which grade is to be viewed.
4. The system displays all the grades for the particular course including any exam or homework grade. The grade display depends upon what grades have been posted by the associated professor.

Entry condition:
The student is logged in and is or was registered for at least one course

Exit condition:
The student is able to view the cumulative and course wise grades.

3.2.1.12: See course details

Use case name: View course details

Participating actors: Student

Flow of events:
1. The student enters the ‘my courses’ tab.
2. The student is shown the list of courses the student is registered for.
3. The student clicks on the course for which details are to be viewed.
4. The system displays all the course details to the student including professor details, TA details, syllabus, recommended books, etc.
5. The student can also download attachments from the course page that may have been made available by the professor teaching the course.
Entry condition:
The student is logged in and is registered for at least one course.

Exit condition:
The student is able to view the course details and view/download all the available course material.

3.2.1.13: View Student details
Use case name: View student details

Participating actors: Student

Flow of events:
1. The student enters the ‘my details’ tab.
2. The system shows the student the details about the student in terms of registration, course summary, grade summary etc. (Summary)

Entry condition:
The student is logged in.

Exit condition:
The student is able to view the details about self or an appropriate message indicating the error is shown.

3.2.1.14: Check eligibility to apply
Use case name: Check eligibility to apply

Participating Actors: Student

Flow of events:
1. The student enters the page containing the position description.
2. The student then has to check if he is eligible to apply for the position.
3. ‘Apply’ button then checks if the logged student has the required GPA to apply for a position.
4. If criteria matches then the Application form is displayed.

Alternative flow of events:
1. The student enters the page containing the position description.
2. The student then has to check if he is eligible to apply for the position.
3. ‘Apply’ button then checks if the logged student has the required GPA to apply for a position.
4. If criteria does not match then the application form should not be shown but a pop up message box should address ‘criteria not matching’.

Entry Condition:
Student must have required GPA.
Exit Condition:
The student is able to view the application form if the eligibility criteria is satisfied else the pop up message box should be shown.

3.2.1.15: Enter Application Details

Use case name: Enter application details

Participating Actors: Student

Flow of events:
1. The student enters the page containing application form.
2. The student answers the various required details.
3. Then to save the data “confirm” button is pressed.
4. The confirmation overview page is then opened which shows all the data entered for verification.
5. If there is a need to change any entry ‘Edit’ can then be pressed to go back to the previous page.
6. If the student finds all entries to be correct he can then press “Submit”.

Entry Condition:
Student must be logged in and must be eligible to apply.

Exit Condition:
The student is able to complete the application for open positions.

3.2.1.16: Scale Students

Use case name: Scale students

Participating Actors: Nil (Initiated by the system)

Flow of events:
1. Prior to the student entering the details in the application form and the data is saved in the database.
2. Then the system automatically computes the skillset into points/flags and generates a score.
3. This set of flags and the score is saved into another database table.
4. When the match students is initiated this data is used for matching the requirements.
5. The appropriate matching list is then returned to the user.

Entry Condition:
The student should have filled the application form.

Exit Condition:
The flags and the score is saved in another database table.
3.2.1.17: Match Requirements

Use case name: Match requirements

Participating Actors: Professor, department admin

Flow of events:

1. The professor/department logs into the system.
2. Then by clicking on ‘add position’ will direct them to the appropriate page.
3. The professor/department enters the skillset required for the position.
4. Then presses ‘Match’ to generate a list of students who have similar requirements.

Entry Condition:
Professor/ department must be logged in to add an open position.

Exit Condition:
The Professor/department is able to see the list of matching students.

3.2.1.18: Intimate Students

Use case name: Intimate students

Participating Actors: Professor/ department admin

Flow of events:

1. The professor/department logs into the system.
2. Then the professor/department performs the ‘match requirements’ as explained in the previous use case.
3. The professor/department can select the number of people he wishes to call for an interview.
4. Accordingly a mail is sent to them with further details.
5. A summary report email is then sent to the professor too.

Entry Condition:
Professor/ department must be logged in to add an open position.

Exit Condition:
The students and the professor/department are able to receive a mail.

3.2.1.19 Add a course

Use case name: Add a course

Participating actors: Department admin

Flow of events:

1. The department admin enters the manage courses tab.
2. The system displays the current courses that are under the department.
3. The department admin presses the ‘add a new course’ button.
4. The system produces a form asking for details about the course.
5. The department admin can now add all the details and press the apply button
6. The system now asks for confirmation from the user to add the course.
7. The department admin now can press the confirmation button to add this new course to the course list.
8. The system adds the new course and presents the courses page to the department admin.

Entry condition:
The user is logged in as a department admin.

Exit condition:
The department admin is able to add new course or is given an appropriate notification about where the process faltered.

3.2.1.20 Remove a course

Use case name: Remove a course

Participating actors: Department admin

Flow of events:
1. The department admin enters the manage courses tab.
2. The system displays the current courses that are under the department.
3. The department admin can now select a course from the list that is to be deleted.
4. The system now displays all the information related to the course clicked.
5. The department admin now presses the remove course button.
6. The system asks for confirmation from the user to delete the course.
8. The system removes the course and presents the courses page to the department admin.

Entry condition:
The user is logged in as a department admin.

Exit condition:
The department admin is able to remove course or is given an appropriate notification about where the process faltered.

3.2.1.21 Add a professor

Use case name: Add a professor

Participating actors: Department admin

Flow of events:
1. The department admin enters the ‘manage teaching faculty’ tab.
2. The system displays the current professors that are under the department.
3. The department admin presses the ‘add a new professor’ button.
4. The system produces a form asking for details about the professor.
5. The department admin can now add all the details and press the apply button.
6. The system now asks for confirmation from the user to add the new professor.
7. The department admin now can press the confirmation button to add this new professor to the professors list.
8. The system adds the new professor and presents the current professors page to the department admin.

Entry condition:
The user is logged in as a department admin.

Exit condition:
The department admin is able to add new professor or is given an appropriate notification about where the process faltered.

3.2.1.22 Add a student

Use case name: Add a student

Participating actors: Department admin

Flow of events:
1. The department admin enters the ‘manage students’ tab.
2. The system displays the current students that are under the department.
3. The department admin presses the ‘add a new student’ button.
4. The system produces a form asking for details about the student.
5. The department admin can now add all the details and press the apply button.
6. The system now asks for confirmation from the user to add the new student.
7. The department admin now can press the confirmation button to add this new student to the students list.
8. The system adds the new student and presents the current students page to the department admin.

Entry condition:
The user is logged in as a department admin.

Exit condition:
The department admin is able to add new student or is given an appropriate notification about where the process faltered.

3.2.1.23 Remove a professor/student

Use case name: Remove a professor/student

Participating actors: Department admin

Flow of events:
1. The department admin enters the ‘manage professors/students tab’.
2. The system displays the current professors/students that are under the department.
3. The department admin can now select a professor/student from the list that is to be deleted.
4. The system now displays all the information related to the professor/student clicked.
5. The department admin now presses the remove professor/student button.
6. The system asks for confirmation from the user to delete the professor/student.
8. The system removes the professor/student and presents the current professors/students page to the department admin.

Entry condition:
The user is logged in as a department admin.

Exit condition:
The department admin is able to remove professor/student or is given an appropriate notification about where the process faltered.

3.3 Nonfunctional requirements

3.3.1 Usability
i. The system must be easy to use by all such that they do not need to read an extensive amount of manuals.
ii. The system must be quickly accessible by the users.
iii. The system must be intuitive and simple in the way it displays all relevant data and relationships.
iv. The menus of the system must be easily navigable by the users with buttons that are easy to understand.

3.3.2 Reliability
i. The System must provide with accurate numbers to the user continuously. Any irregularities will be taken care by the administrator or by the program itself.
ii. The System should successfully add courses, students, professors and also the departments along with their access levels. It should also be able to perform the intended functionalities without any data loss.
iii. The system must provide a password enabled login to the user to avoid any unauthorized user changing the data in the system.
iv. The system should provide the user updates on completion of requested processes and if the requested processes fail, it should provide the user the reason for the failure.
v. The system should not update the data in any database for any failed processes.

3.3.3 Performance
i. The system must not lag, because the users don’t have down-time to wait for it to complete an action.
ii. The system must complete updating the databases after each functionality is implemented.
iii. According to their respective access levels, the functions of the system must be available to the user every time the system is in use.
iv. The calculations performed by the system must comply according to the norms set by the user and should not vary unless explicitly changed by the user.

### 3.3.4 Supportability

i. The application is designed such that it works even on systems having the minimum configuration.

ii. The system is adaptable even if additional plugins or modules are added at a later point.

iii. The data on the server can be connected from multiple users, so as to make the system more portable.

iv. The application is designed to support different operating systems.

### 3.3.5 Fault tolerance

i. To handle the errors which may occur during the inputting, the system must be able to intimate the user with either a pop up message box or the related error message. We need to make sure that incorrect data should not be saved into the database.

ii. Under Extreme conditions, the system needs to make sure that there is no integrity loss and there is no data loss also. The data will need to be continuously be updating on the server. Under suspicious circumstances the system should authenticate the user before giving them access.

### 3.3.6 Interface

i. There should be checks done on the kind of format the inputs are accepted by the application. There should be a data type check before any insertion into the database.

ii. The data would be stored centrally into a database server. Upon firing of queries the intended outputs should be obtained.

iii. The input to the application would either the user inputted data or the data that is stored in the SQL servers.

### 3.3.7 Packaging

i. The system must be able to run on the Windows operating systems beginning with Windows 7, and must be able to run on future releases.

ii. The software must incorporate a license key authentication process.

iii. The packaging must come with a manual that details the use of the system, and also the instructions on how to use the program. This manual may be included either in a booklet that comes with the software, or on the disc that the software itself is on.

### 3.3.8 Legal

Every software that is developed should be kept in mind that all its functionalities should be legal and not be counterfeiting any other software. There should also be proper documentation and registration of the product so that it can safely be legal always.
3.4 System models

3.4.1 Use case model
3.4.2 User interface (screen mock-ups)
Assumptions:

The following is the list of assumptions being made in the following project:

- All the students already have their own email Id’s setup.
- The student is enrolled in the university.
- There are a fixed number of classrooms
- The length of each class is one hour.
- We assume that the finances for the university is already taken care of.
- The capacity of every class is same.

Tools Used:

- Eclipse Juno (or) Netbeans 7.4
- JavaFX scene builder 2.2
- GenMyModel.com (For use cases)
- MySQL

References:

- Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd Edition)

Note:

The above given information is not exhaustive. The requirements may change over a period of time. If there are any changes being made it will be reflected in the further iterations of this document.