

# Requirements Specification

## IU Physics Gradebook System

Yu Feng<sup>1</sup>

Rohit Chandran<sup>2</sup>

Naga Rekha Malae<sup>3</sup>

Rev 374, November 1, 2008

<sup>1</sup>Physics Department, Indiana University, Bloomington IN 47401

<sup>2</sup>Department of Computer Science, Indiana University, Bloomington IN 47401

<sup>3</sup>Department of Computer Science, Indiana University, Bloomington IN 47401



## **Executive Summary**

Rev 374, November 1, 2008.

The Physics Department at Indiana University is a non-profit education institution. It provides quality education in physics to students on campus. The department also offers undergraduate and graduate level programs in the physics major.

The proposed system will be used by the Physics Department as a means of entering and storing grades based on the organizational structure. After studying the client environment, the proposed system's characteristics, the possible technical issues in the development of the system and evaluating possible alternatives in the feasibility study, the team found that such a system can be successfully built.

The client currently uses OnCourse to achieve this task of managing grades for students. However, due to complexities in the organizational structure and shortcomings of OnCourse, the current system does not achieve this task very efficiently. It has, therefore, become necessary that a suitable system be developed to meet the needs of the client.

The team studied the client background, emphasizing on organizational structure and the problems that the client faces with the current system. At each level of the structure, the problems are made evident. An understanding of daily activities of the client has been taken into account.

Next, the team analyzed specific goals of the system in terms of the needs of the client with respect to setting up the organizational structure and assigning grades. Also, the client environment has been carefully understood, specifically with respect to roles of users in the system. These roles have been understood in terms of user activity and frequency of use. Also, the system's hardware and software environments have been studied along with the interaction of the system with other information systems, mainly the Central Authentication Service (CAS).

The team has highlighted the functioning of the current system with the help of Entity-Relationship (ER) diagrams and Data Flow Diagrams (DFD's). The availability of the source code and documents is minimal but a process of reverse engineering covers some of the more important aspects of the current system.

The ER diagram captures most of the relationships and constraints in the proposed system. Some of the constraints cannot be modeled and are, therefore, explicitly stated. The DFD's capture the essential data flowing in the system specific to the various users of the system. In addition, important functional, qualitative and maintenance requirements are considered with special focus on client needs. The functional requirements have been prioritized and stated clearly.

A user interface that meets the client requirements has been proposed. Additionally, other factors such as training after the installation and limitations of the proposed system have been analyzed.

Also, a Quality Assurance Plan (QAP) is proposed to ensure the quality of the system. A Project Plan is also attached to this document. The roles and tasks of each team member are also stated.

Client questions, considerations and concerns regarding different aspects of the document have been noted at the end.



# 1 Introduction

Rev 374, November 1, 2008.

The Physics Department at Indiana University is a non-profit education institution. It provides quality education in physics to students on campus. The department also offers undergraduate and graduate level programs in the physics major.

The proposed system will be used by the Physics Department as a means of entering and storing grades based on the organizational structure mentioned in Section 2. After studying the client environment, the proposed system's characteristics, the possible technical issues in the development of the system and evaluating possible alternatives in the feasibility study[1], the team found that such a system can be successfully built.

This document begins with the client background in Section 2, where the client organizational structure and the problems that the client faces with the current system are analyzed. The goals of the proposed system are then stated in Section 3. In the following Section 4, the team studied the client environment, including the users, the hardware and software environments, and the interaction of the system with other information systems. In Section 5, the current is analyzed with the help of Entity-Relationship (ER) diagrams and Data Flow Diagrams (DFD's).

The requirements of the proposed system is finally examined in details in Section 6. The ER diagram, DFDs and their interactions are modeled in Section 6.2, Section 6.3 and Section 6.4, respectively. The functionality requirements, business rules, maintenance requirements and qualitative requirements are specified through Section 6.5 to 6.8. A user interface design that can achieve these requirements is proposed in Section 6.9. Other factors, including the training after the installation and the limitations of the proposed system are stated in Section 6.10, 6.11.

A draft Quality Assurance Plan (QAP) is covered in Section 7.

The comments and feedbacks from the primary client are summarized in Section 9.

## 2 Client Background

Rev 374, November 1, 2008.

The primary client of the project is the Physics Department at Indiana University, Bloomington. The Physics Department provides quality education to students on campus in broad topics of physics in undergraduate and graduate level programs for physics majors and minors. The client organization is divided into two primary divisions, the Research Division and the Teaching Division. The Teaching Division is the main target of this project. The Research Division is not of concern for the proposed system.

The Teaching Division consists of professors and associate instructors. A professor and several associate instructors form a Teaching Unit, who is responsible for offering a course to students. Each individual member in the Teaching Division can participate in multiple units. On the course hierarchy side, a course consists of various components, including but not limited in lecture component, lab component, discussion component and recitation component. Students enrolled in each component are organized by sections. Also, the set of sections in one component need not be the same as the set of sections in another component.

It must be noted that a professor is responsible for the entire course, whereas each instructor may be responsible for one or more sections.

The primary client, Dan Beeker, coordinates the teaching units and ensure the teaching division survives after every semester. The current system, OnCourse, does not allow for this to be done efficiently. The problem is that OnCourse only allows for the creation of courses and then for the assignment of professors, instructors and students to each of the courses. It does not allow for the hierarchy shown above. Therefore, the client organization currently creates a new “course” for each section that is needed within an actual course. Students are also also assigned to each such new “course”.

Also, each teaching unit requires efficiency for assigning grades to students for each assignment which varies with respect of the context of the course component. Currently, the client organization uses various workarounds to accomplish the above task. Many grades from previous semesters have been lost and the assigning grades for the semester in progress has been a cumbersome task. This is because the number of grade entries to be made for each student is very large and confusing.

All of this, in turn, become a very difficult management and storage issue for the administrator at the end of the semester with respect to the grades of each of the students.

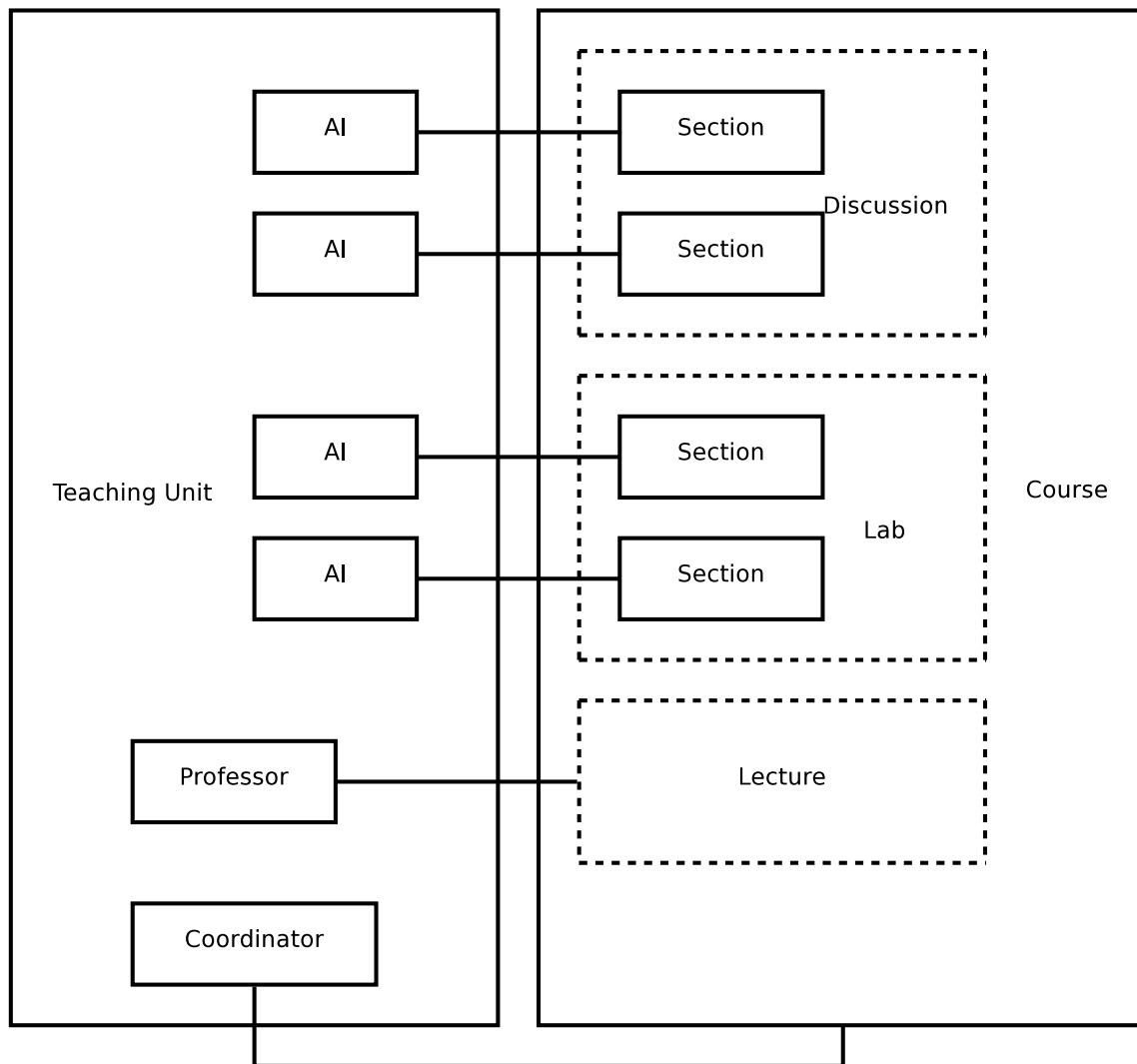


Figure 1: The Client's Business Model

### 3 Goals of the Project

Rev 374, November 1, 2008.

The goal of the project is to precisely represent the unique course structure and efficiently manage the grades assigned to the students over years.

**Handle Complexity** The Physics Department has a fairly unique course structure as explained in (2). The unique structure implies the complexities of allocation of entities for the administrator and also that of assigning grades. The new system should be able to help the client to handle this complexity.

**Provide Efficiency** The primary client specifically requested the system to be direct and efficient. The user interface should be simple; the model should be concise, and the interaction between the system and users should contain as few steps as possible.

**Ensure Privacy** The system will deal with students' grades, which is generally regarded as a privacy of individual students. Therefore, ensuring the privacy is also an implied goal of the system.



## **4 Environment**

Rev 374, November 1, 2008.

### **4.1 Users and Roles**

Every semester, the system is expected to interact with approximately 1000 students, 20 professors, 40 instructors and the primary client (administrator).

The administrator, professors and instructors form the internal users of the system and the students form the external users of the system.

#### **4.1.1 Primary Client and Contact**

The Primary Client for the proposed system is Daniel Beeker, the Undergraduate Laboratory Coordinator. Daniel Beeker is the head staff for undergraduate education in the teaching unit. He has requested a new gradebook system with very clear objectives to the team.

The primary client has sufficient resources for ongoing operation and maintenance of the proposed system. He is also generally available in his office. It is unlikely that he will be absent for any extended periods during the course of the project.

The Primary Client can be reached at

Daniel Beeker, Undergraduate Physics Laboratory Coordinator  
Phone Number: 812-855-5903  
SW 115 727 E. Third St. Bloomington, IN 47405-7105  
debecker@indiana.edu

#### **4.1.2 Information Systems Staff**

The client already has an IT staff to maintain servers and computers, and does not intend to hire new staff. The current staff will be responsible for maintaining the proposed system.

#### **4.1.3 Professors**

Professors in the system are in charge of each of the courses that they are teaching. The professors are fully responsible for all such courses.

A Professor give assignments to each of the components within the course. This would generally be done as each of the assignments are to be given to the students in the semester. It may also be possible that a few assignments may be in one go by the professor. An error in the assignment name would mean that the professor might change the name (modify) of the assignment. The Professor can also possibly revoke an inappropriate assignment.

A professor acquires the student roster at both the course and component levels.

He/she may also assign grades for any student for any assignment in any section in the course. This is generally done after the completion of each assignment. In practice, the professor does not assign grades for the lab, discussion and recitation components. He could, however, also assign grades for students in the lab component if required. Cases should be rare because a professor directly conducts the lab sessions.

The final grades of all students for each component and also the final course grade can be obtained by the professor.

#### **4.1.4 Instructors**

Instructors can be assigned to multiple sections across components in a course. They assign grades of any student for any assignment within the sections that he is assigned. This is generally done after the completion of each assignment. This may also mean that they overwrite grades that were previously assigned by the professor.

#### **4.1.5 Administrator**

The primary client is the Administrator.

He has complete access to the system. He accomplishes the following:

- Setting up the course structure for the current course offering: This is generally just before the beginning of the semester.
- Assigning professor(s) to a course offering: This is just before the beginning of the semester.
- Assigning instructors to various sections: This may be just before the beginning of the semester or just after the semester begins. This process may, however, be done later in the semester too.
- Adding students to the roster for both course and components: This is just before the beginning of the semester. This may, however, be done later in the semester too.
- Reviewing gradebooks for all components and courses and generating reports: This is generally at the end of the semester.
- Making backups for the system: This may be done at regular intervals of time.

The administrator is very familiar with the above mentioned tasks.

#### **4.1.6 Students**

Students are external users of the system. Students in a course simply view the list of assignments and the grades assigned to them for each of the assignments. Each student can only view his/her own grades. The grades would be available for viewing as soon as the professor or instructor puts them up. The final grades are generally available in the week following the end of the semester.

It must be noted, however, that a student can also be an undergraduate instructor.

### **4.2 Hardware Platforms**

The client currently has PCs with Pentium 4 CPUs that are suitable for running Windows 2000 Server. These machines can be used for development purposes. The computing capacity of the PCs is sufficient for the proposed system. The final hardware environment might differ from that of the development machine, and therefore an installation stage is expected.

### **4.3 Software Platforms**

The software platform of the proposed system will be built upon productive versions of several pieces of Free Software.

- for the Database management system, MySQL should be deployed;
- for the HTTP service daemon, Apache should be deployed;
- for the Web applications, PHP run-time (with development run-time) should be deployed.

A software environment that is comparable with the one mentioned above is easy to set up, and can be accessible at any personal or commercial web hosting service.

### **4.4 Interaction with Other Information Systems**

The proposed system has interactions with the Central Authentication Service (CAS) for authenticating regular users. [5]

### **4.5 Impact on Operations**

After the deployment of the proposed system, the team will prepare the primary client on the usage of the system. The primary client in turn will be responsible for training the other users of the system. The team will also provide a user manual for the proposed system.

This initial familiarization with the system may slow down operations a little, but is not expected to last too long.

Also, the Physics Department is expected to stop using OnCourse for updating and keeping track of grades. Final grades are sent to the professors that offer the courses.

## 5 Current System

Rev 374, November 1, 2008.

The client currently take the advantages of the Gradebook module of the OnCourse system. The gradebook system is simple, and most functionalities, especially

- Course/Section management,
- Roster management,
- Instructor/TA assignment,

are cascaded to other subsystems of OnCourse.

The direct document of OnCourse is not available; however documentation of the origin of OnCourse, Sakai Project is available. Since there are no substantial differences between OnCourse and Sakai, in this section Sakai is analyzed.

### 5.1 Current Entity-Relationship Model

In the current system, there are three types of users: Student, Instructor and TA. The modification history of grades are not stored. An offered course is called Section.

Although Sakai has its own user authentication system, OnCourse utilizes the IU itaccount for identifying users. This fact is also captured in the ER Model

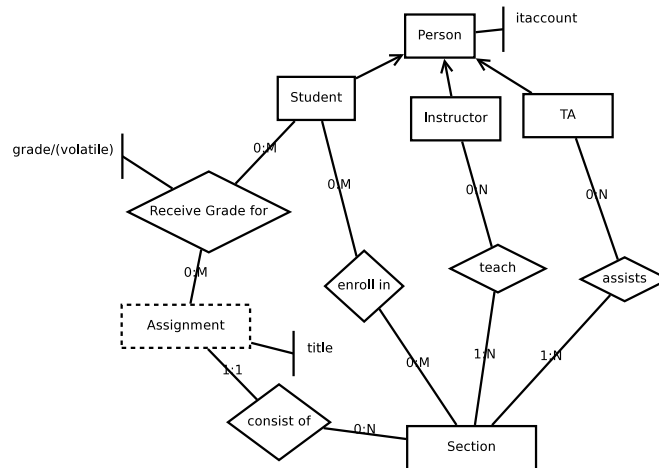


Figure 2: ER Current System

## 5.2 Data Flow Diagrams for Current System

In this section, the data flow model for the current system is described in Data Flow Diagrams in the second section; the definitions of the data flow are defined in the first section. The data flow in Current system is simple.

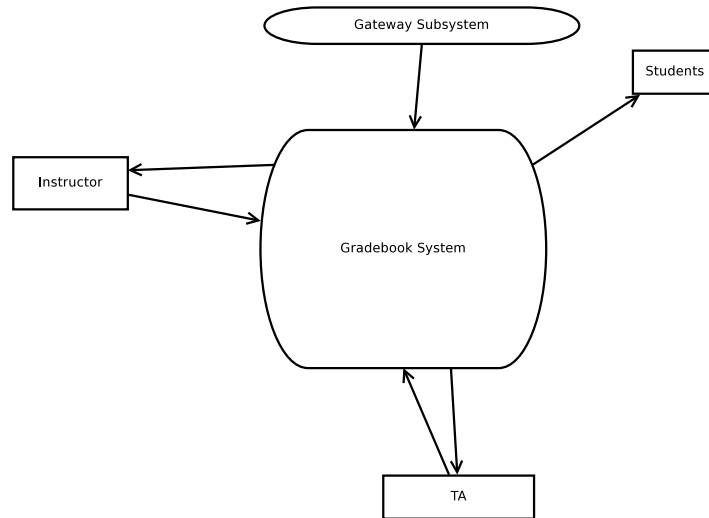


Figure 3: ContextLevel Current System

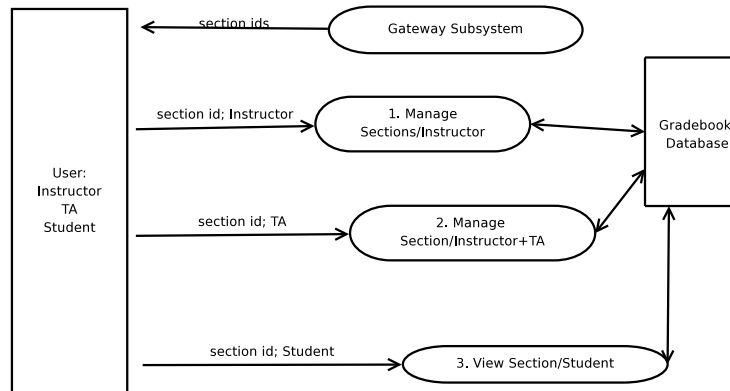


Figure 4: L0 Current System

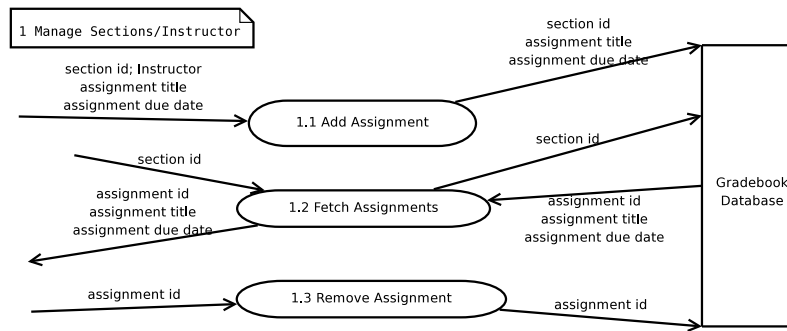


Figure 5: L1-1 Current System

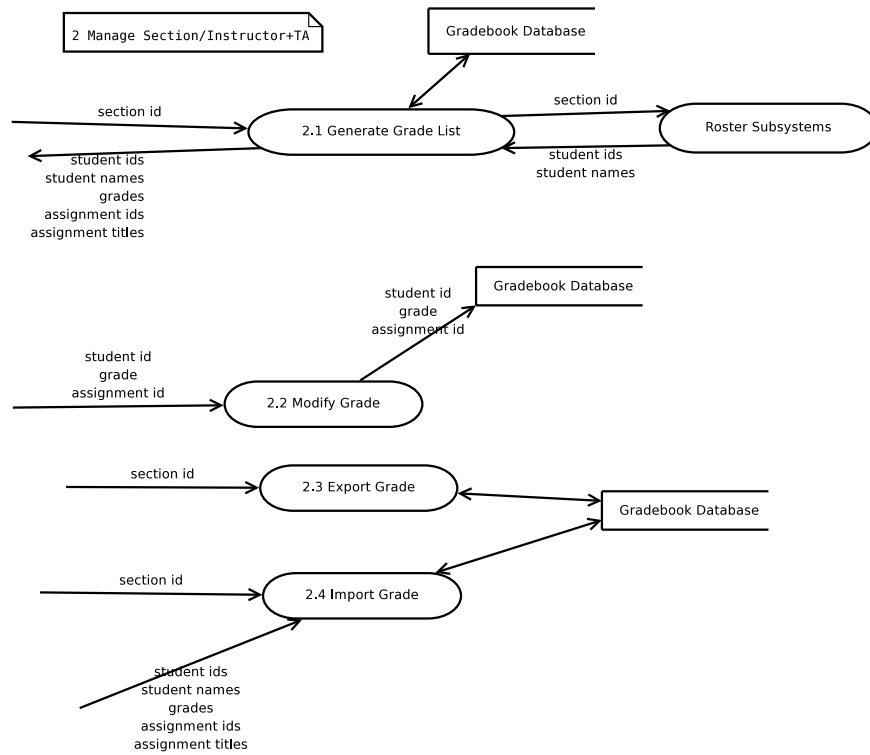


Figure 6: L1-2 Current System

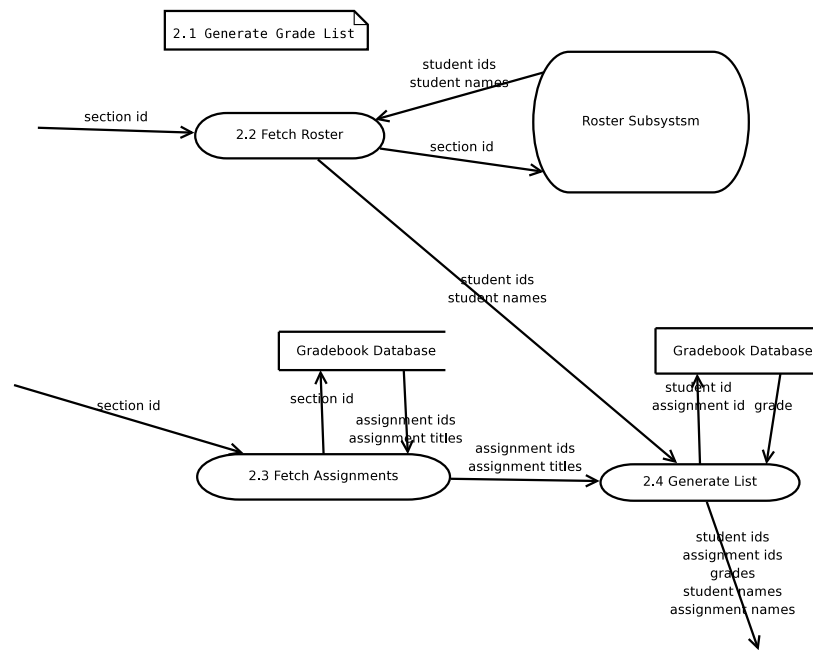


Figure 7: L2-2.1 Current System

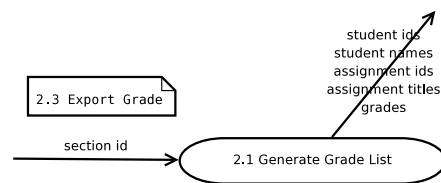


Figure 8: L2-2.3 Current System

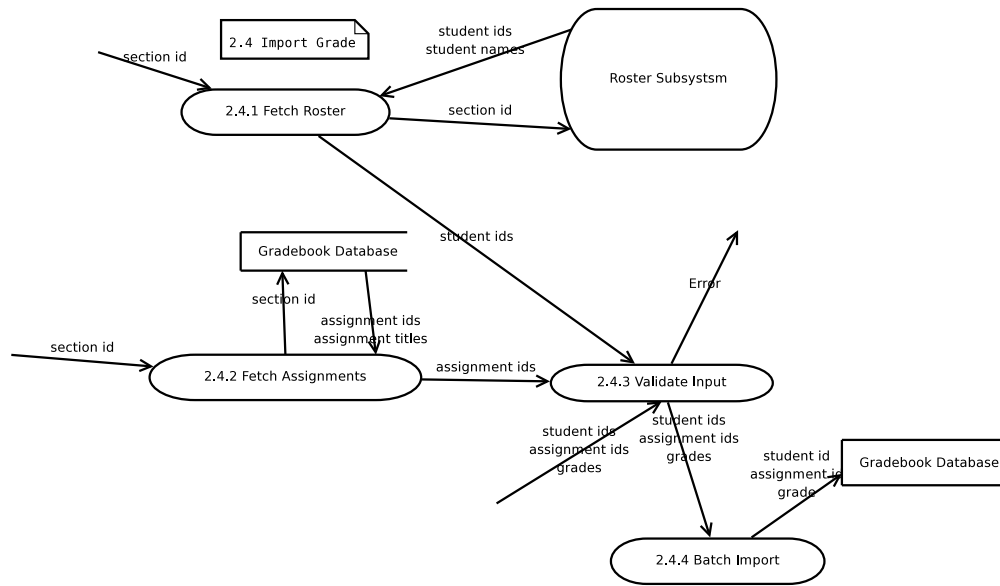


Figure 9: L2-2.4 Current System

### 5.3 Other Available Documents

Sakai<sup>1</sup> Project provides a description of the functional requirements of their gradebook subsystem[4].

---

<sup>1</sup>OnCourse is a derivative of Sakai.





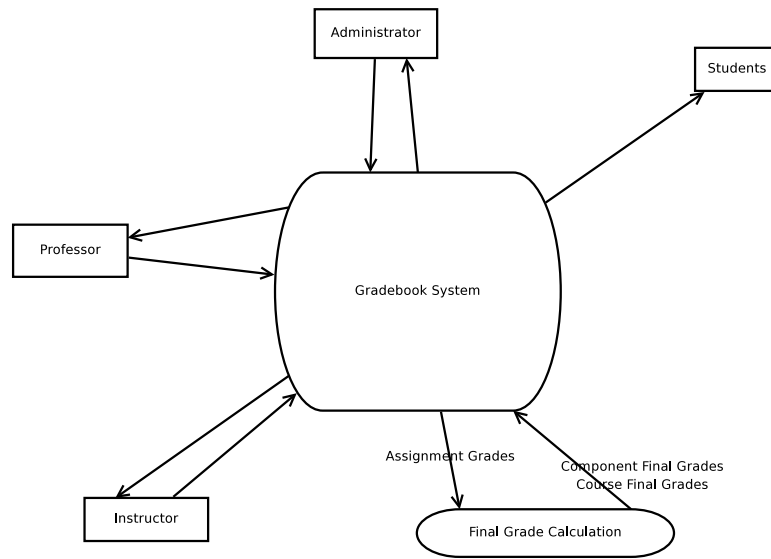


Figure 12: ContextLevel Proposed System

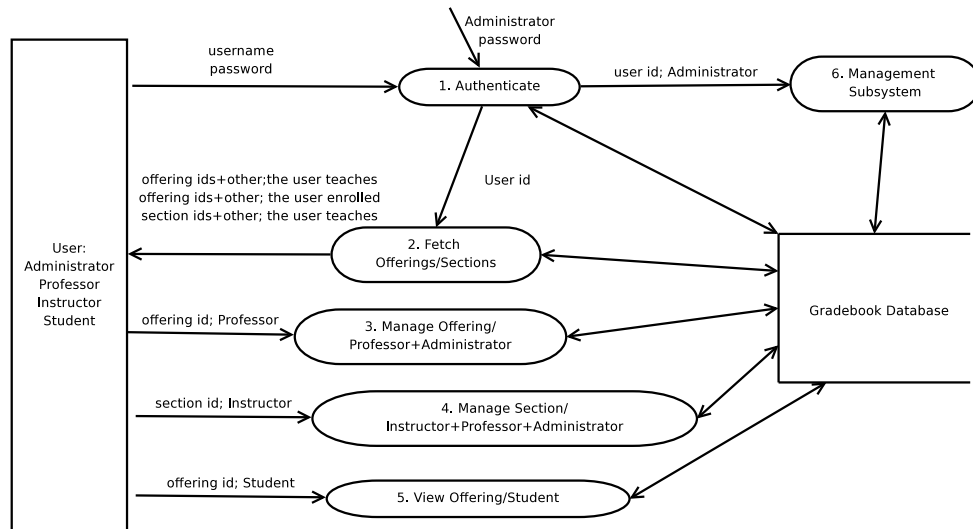


Figure 13: L0 Proposed System

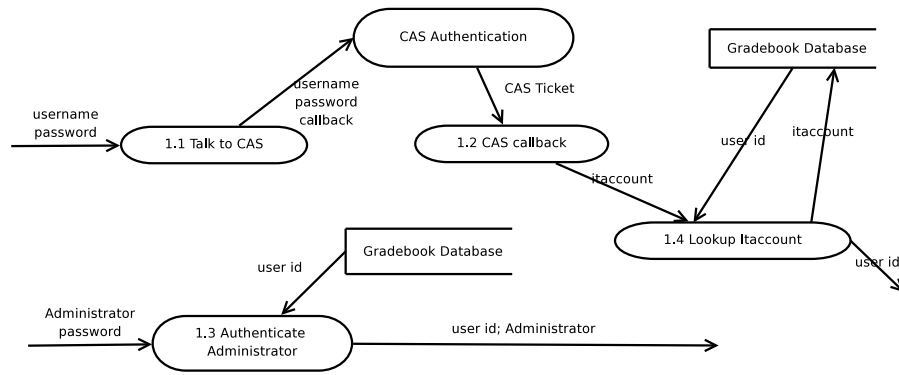


Figure 14: L1-1 Proposed System

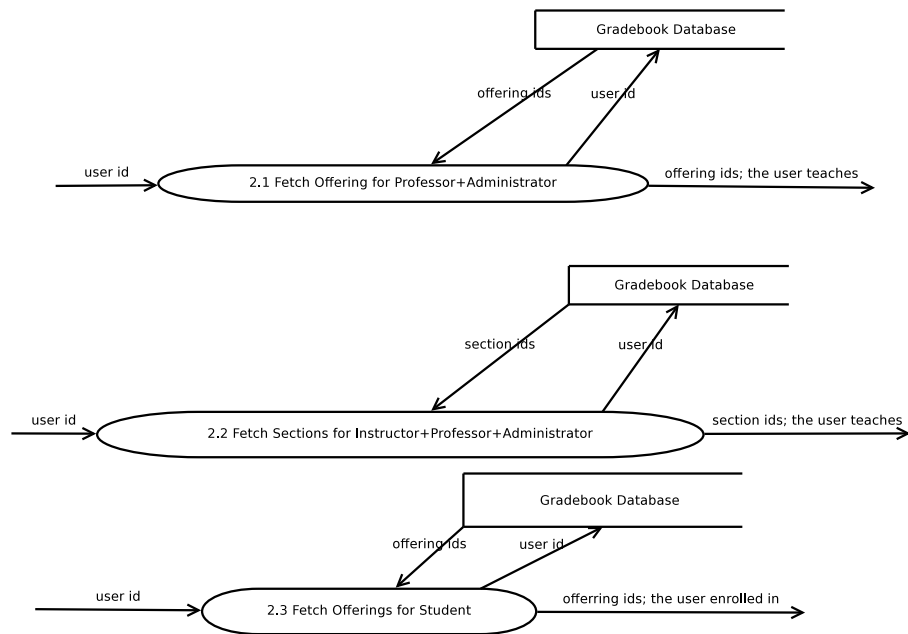


Figure 15: L1-2 Proposed System

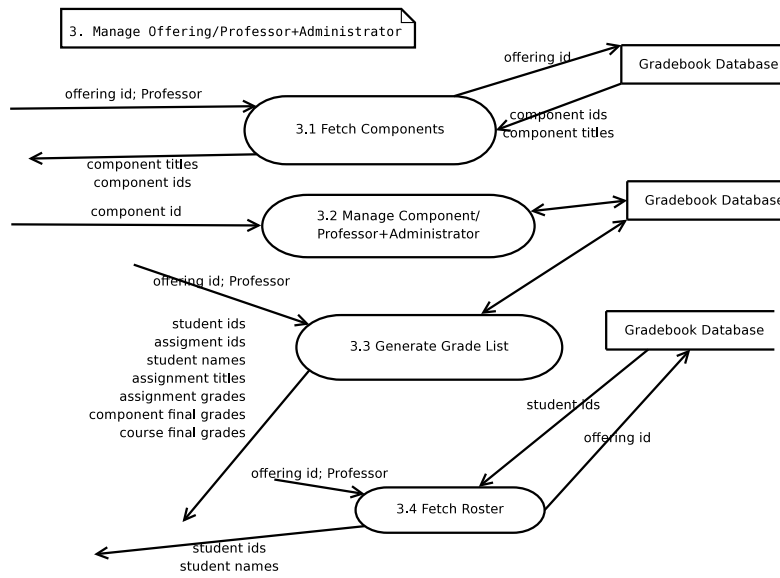


Figure 16: L1-3 Proposed System

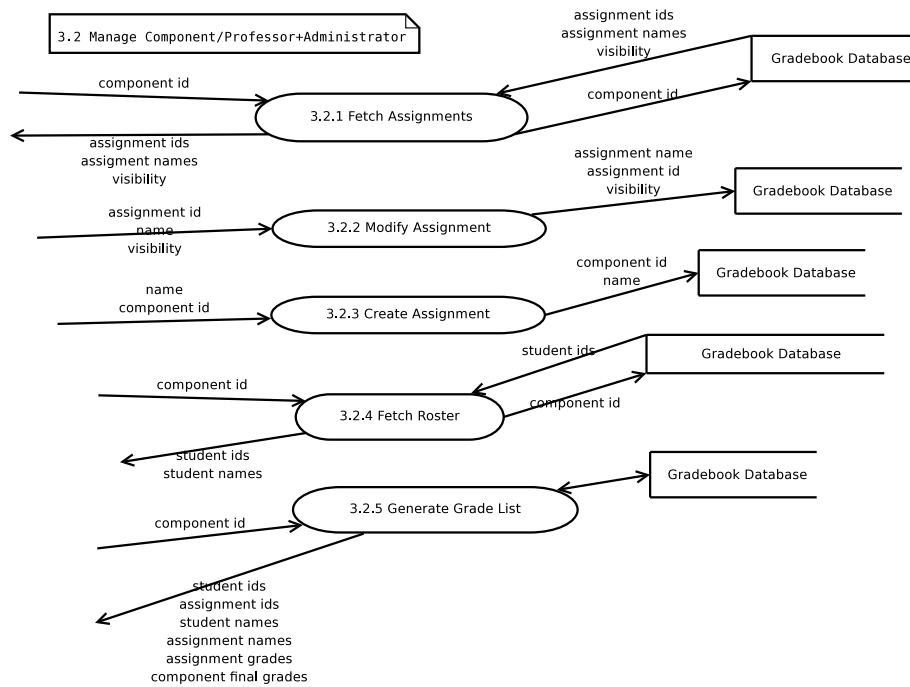


Figure 17: L2-3.2 Proposed System

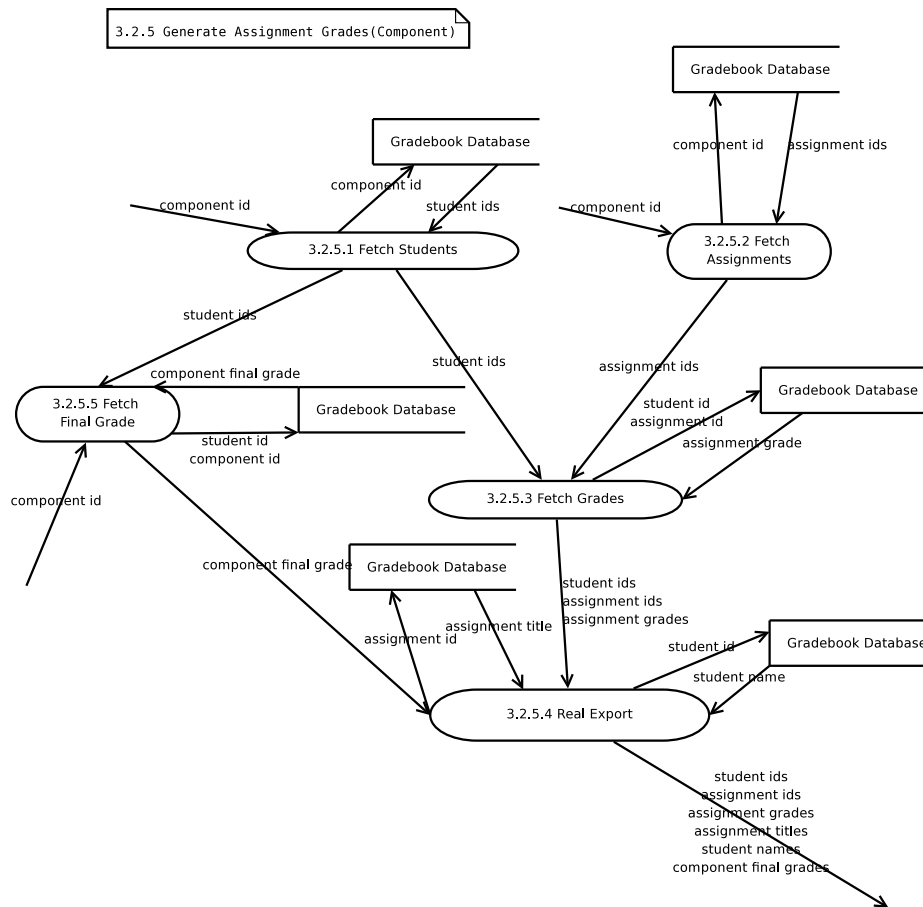


Figure 18: L3-3.2.5 Proposed System

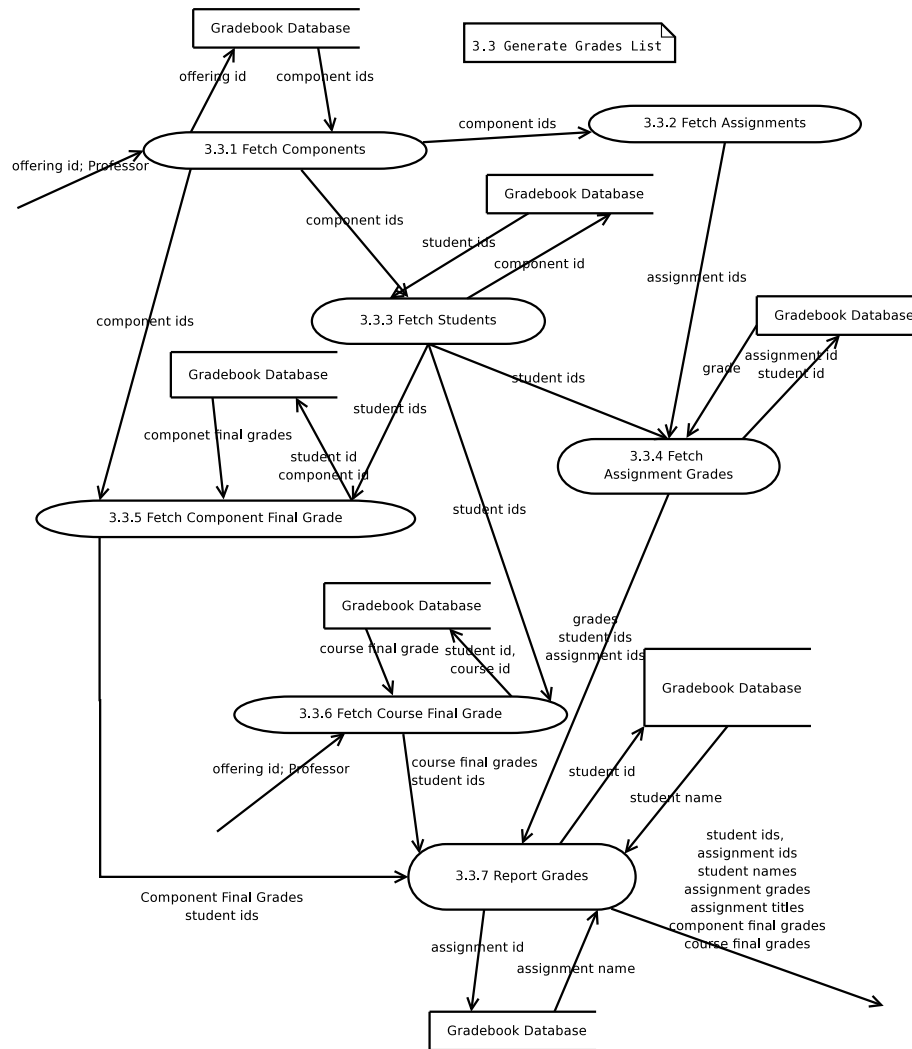


Figure 19: L2-3.3 Proposed System

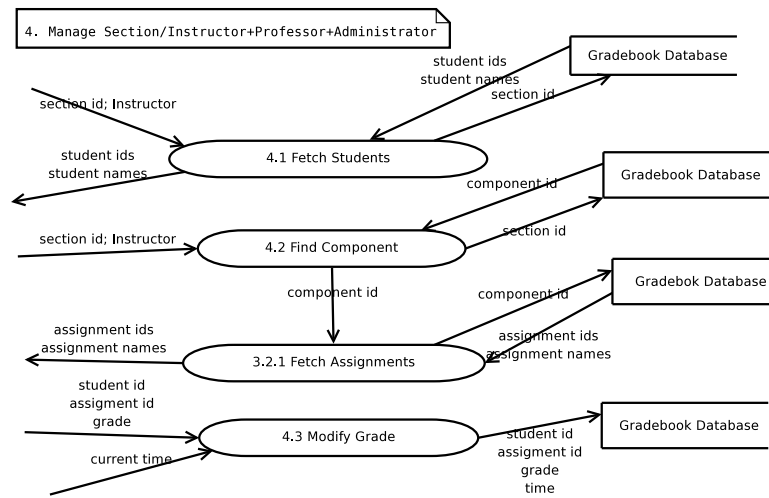


Figure 20: L1-4 Proposed System

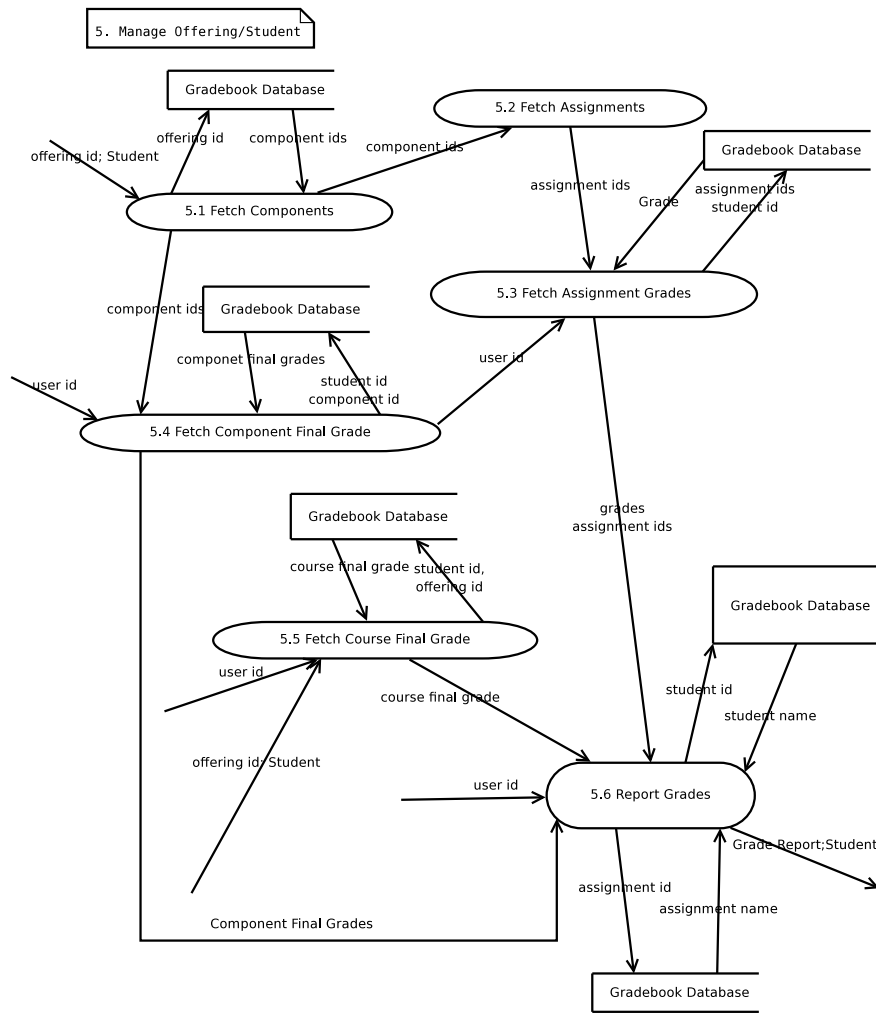


Figure 21: L1-5 Proposed System



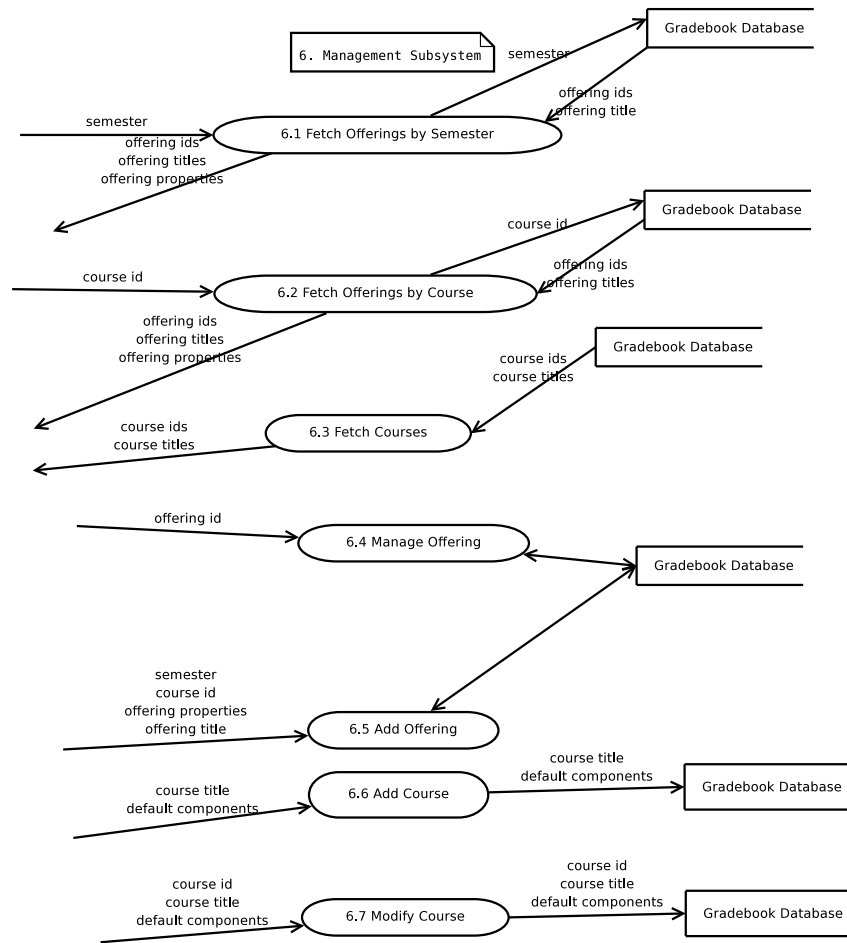


Figure 22: L1-6 Proposed System

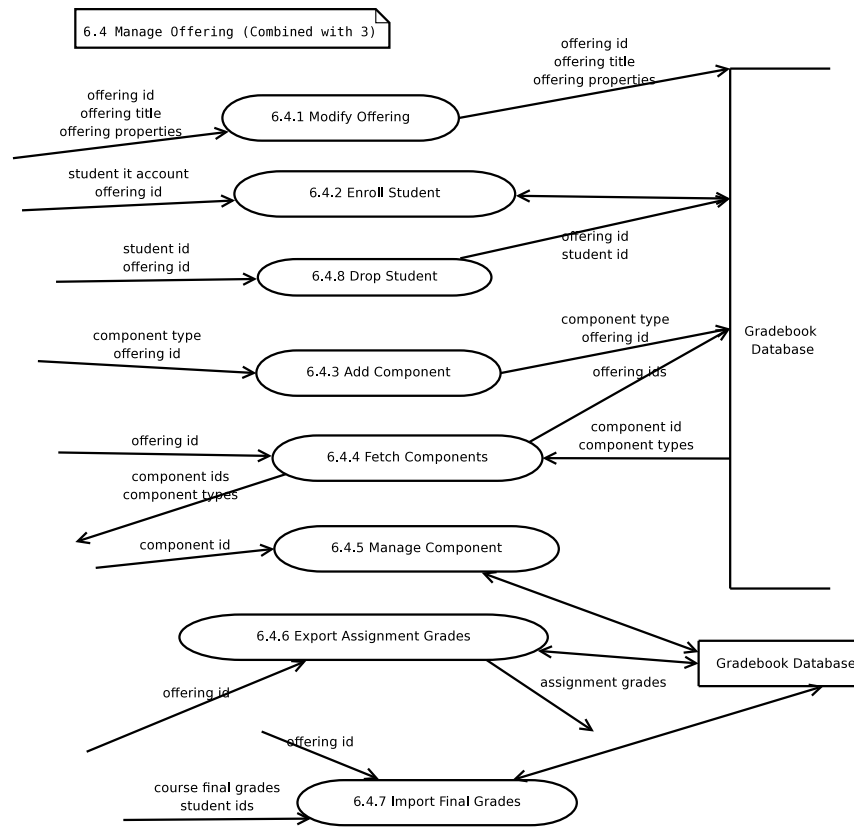


Figure 23: L2-6.4 Proposed System

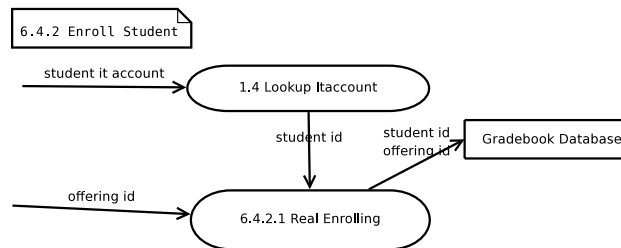


Figure 24: L3-6.4.2 Proposed System

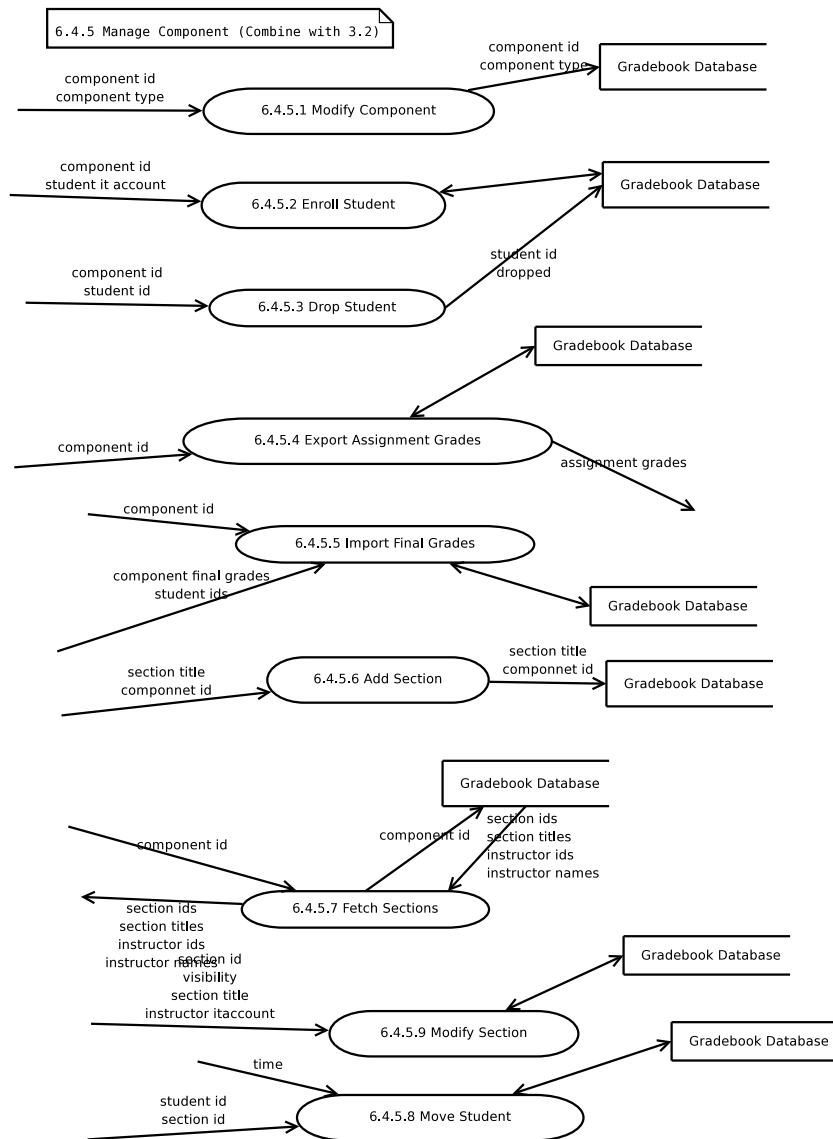


Figure 25: L3-6.4.5 Proposed System

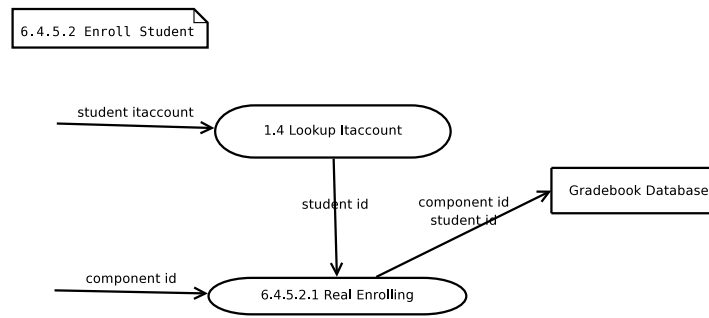


Figure 26: L4-6.4.5.2 Proposed System

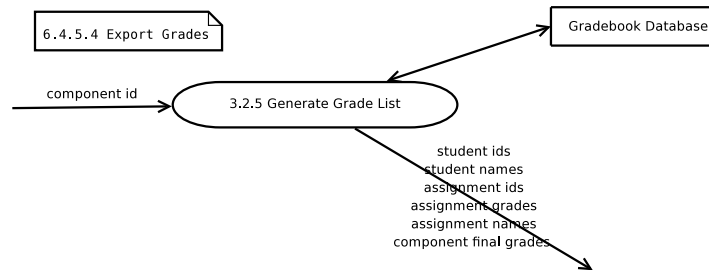


Figure 27: L4-6.4.5.4 Proposed System

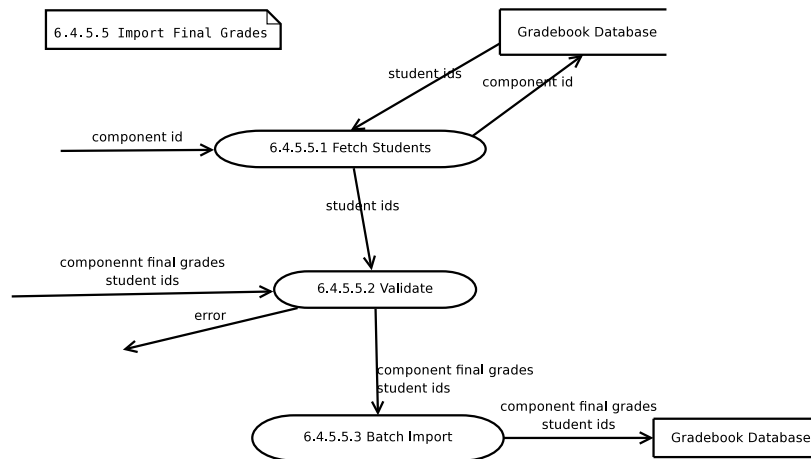


Figure 28: L4-6.4.5.5 Proposed System

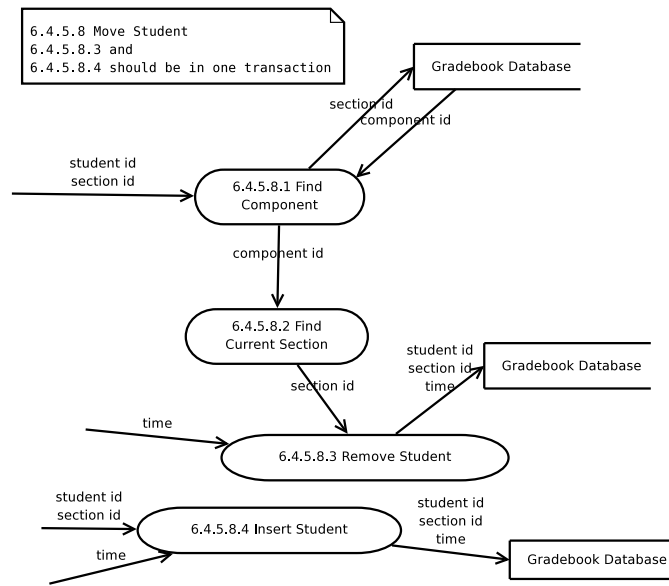


Figure 29: L4-6.4.5.8 Proposed System

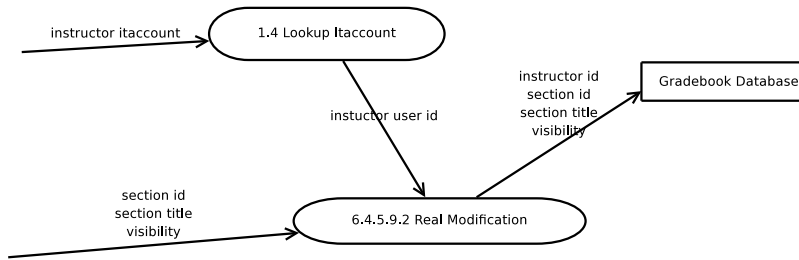


Figure 30: L4-6.4.5.9 Proposed System

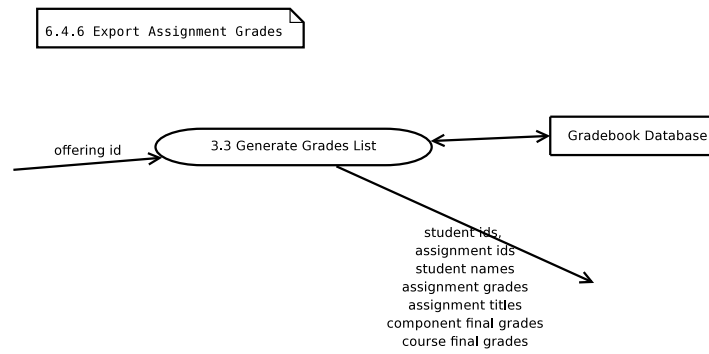


Figure 31: L3-6.4.6 Proposed System

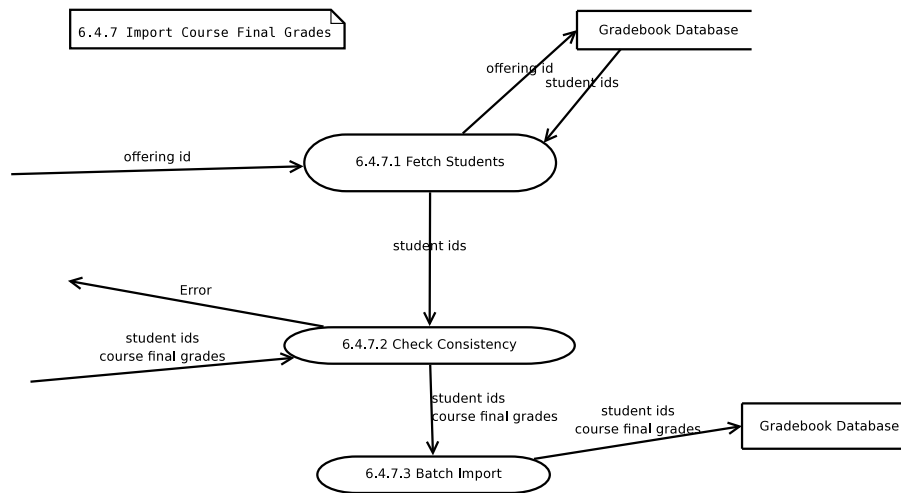


Figure 32: L3-6.4.7 Proposed System

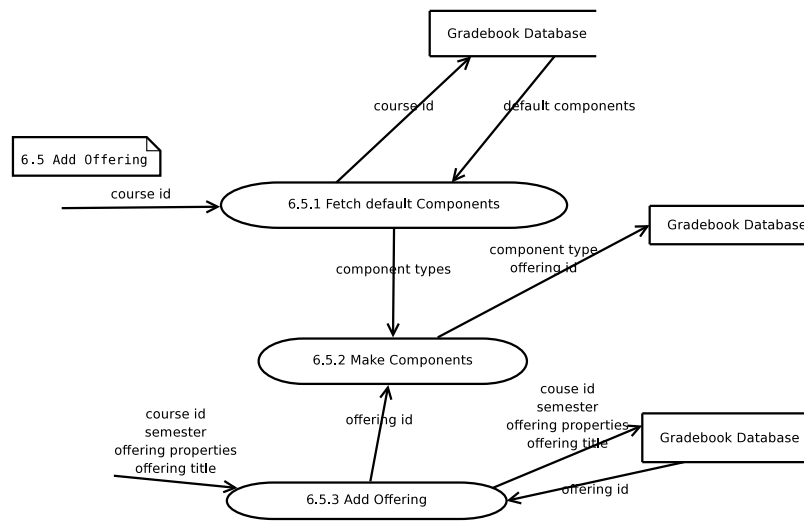


Figure 33: L2-6.5 Proposed System

## 6.4 Interaction of ER and DF Models

In this section, the DFD processes and the entities on which they operates are listed.

To rapidly locate the DFD process, strip the last division in the number to obtain a parent DFD number, and look the process in the parent DFD.

1	Person
1.1	N/A
1.2	N/A
1.3	Person
1.4	Person
2	Offering, Section, Person
2.1	Offering, Person
2.2	Section, Person
2.3	Offering Person
3	Offering, Person, Component, Assignment
3.1	Person, Component, Offering
3.2	Component, Assignment, Person

3.2.1	Component, Assignment
3.2.2	Assignment
3.2.3	Component, Assignment
3.2.4	Component, Person
3.2.5	Component, Person, Assignment
3.2.5.1	Component, Person
3.2.5.2	Component, Assignment
3.2.5.3	Person, Assignment
3.2.5.4	Person, Assignment, Component
3.2.5.5	Person, Assignment, Component
3.3	Person, Assignment, Component, Offering
3.3.1	Offering, Component
3.3.2	Component, Assignment
3.3.3	Person, Component
3.3.4	Person, Assignment
3.3.5	Component, Person
3.3.6	Offering, Person
3.3.7	Person, Assignment, Component, Offering
3.4	Offering, Person
4	Assignment, Person, Component, Section
4.1	Section, Person
4.2	Section, Component
4.3	Person, Assignment, Component
5	Offering, Component, Assignment, Person
5.1	Offering, Component
5.2	Component, Assignment
5.3	Person, Assignment
5.4	Component, Person



5.5	Offering, Person
5.6	Person, Assignment, Component, Offering
6	Offering, Course, Person, Component, Assignment, Section
6.1	Offering
6.2	Course, Offering
6.3	Course
6.4	Offering, Person, Component, Assignment, Section
6.4.1	Offering
6.4.2	Person, Offering
6.4.3	Offering, Component
6.4.4	Offering, Component
6.4.5	Component, Person, Assignment, Section
6.4.5.1	Component
6.4.5.2	Component, Person
6.4.5.2.1	Component, Person
6.4.5.3	Component, Person
6.4.5.4	Component, Person, Assignment
6.4.5.5	Component, Person
6.4.5.5.1	Component, Person
6.4.5.5.2	Component, Person
6.4.5.5.3	Component, Person
6.4.5.6	Component, Section
6.4.5.7	Component, Section, Person
6.4.5.8	Person, Section, Component
6.4.5.8.1	Section, Person, Component
6.4.5.8.2	Component, Section
6.4.5.8.3	Section, Person
6.4.5.8.4	Section, Person

- 6.4.5.9 Section, Person
- 6.4.5.9.2 Section, Person
- 6.4.6 Student, Assignment, Offering, Component
- 6.4.7 Offering, Course, Person
- 6.4.8 Student, Offering
- 6.5 Course, Component, Offering
- 6.5.1 Course
- 6.5.2 Component, Offering
- 6.5.3 Course, Offering
- 6.6 Course
- 6.7 Component, Section, Person

## 6.5 Functionality Requirements

The client requires the system to maintain the information that he is interested in but not to process them. Therefore all functionality requirements are related to the information content.

### A. Record the ER model

The proposed system should be capable of recording the ER model of the client organization, as captured in 6.2.

### B. Support and Maintain the ER model

#### 1. Student

##### i. View Student Grades

**Information Source** user ⋈ offering ⋈ component ⋈ assignment ⋈ assign\_grade

**Select Condition** by user.id, by offering.id, max assign\_grade.time

**Sort Condition** N/A

**Other** N/A

#### 2. Instructor:

##### i. Assign Section Grades

**Insert** grade, time, teacher\_id, student\_id, assignment\_id

**Information Source** assignment\_grade

#### 3. Professor:

##### i. Assign Section Grades

Refer to 2(b)i

##### ii. Add Assignments

- Insert** assignment\_title, hidden=false  
**Information Source** assignment
- iii. Modify Assignments  
**Update** assignment\_title, hidden=false  
**Select Condition** by assignment\_id  
**Information Source** assignment
- iv. Hide Assignments  
**Update** assignment\_title, hidden=true  
**Select Condition** by assignment\_id  
**Information Source** assignment
- v. View/Export Overall Course Grades<sup>3</sup>  
**Information Source** offering ⋈ component ⋈ assignment ⋈ assign\_grade (.receiver) ⋈ person  
**Select Condition** by offering.id, max assign\_grade.time  
**Sort Condition** by component, by person, by assignment  
**Other** N/A
- vi. View/Export Overall Component Grades  
**Information Source** component ⋈ assignment ⋈ assign\_grade (.student\_id) ⋈ person  
**Select Condition** by component.id, max assign\_grade.time  
**Sort Condition** by person, by assignment  
**Other** N/A
- vii. View/Export Course Rosters  
**Information Source** course ⋈ course\_person ⋈ person  
**Select Condition** by course.id  
**Sort Condition** by person  
**Other** N/A
- viii. View/Export Component Rosters  
**Information Source** component ⋈ component\_person ⋈ person  
**Select Condition** by component.id  
**Sort Condition** by person  
**Other** N/A
4. Administrator (Information Management)
- i. Add Courses  
**Insert** course\_name,  
**Information Source** course
- ii. Modify Courses  
**Update** course\_title, default\_components

---

<sup>3</sup> A Course Grade is actually an Offering Grade. The terminology is so widely used in the client's daily practice that the team decided to preserve it so as to avoid confusion in the client side.

- Select Condition** by course\_id  
**Information Source** course
- iii. Add Course Offerings  
**Insert** course\_id, offering\_title, offering\_properties  
**Information Source** offering
- iv. Modify Course Offerings  
**Update** offering\_title, semester, professor\_id  
**Select Condition** by offering\_id  
**Information Source** offering
- v. Assign Professors to Offerings  
Integrated in 2(d)iv
- vi. Add Students to Offering  
**Insert** student\_id, offering\_id  
**Information Source** student\_offering
- vii. Remove(Hide) Students from Offering Rosters  
**Update** student\_id, hidden=true  
**Select Condition** by student\_id, offering\_id  
**Information Source** offering
- viii. Add Components  
**Insert** offering\_id, component\_type  
**Information Source** component
- ix. Hide Components  
**Update** component\_type, hidden=true  
**Select Condition** by component\_id  
**Information Source** component
- x. Modify Components  
**Update** component\_type, hidden=false  
**Select Condition** by component\_id  
**Information Source** component
- xi. Add Assignments  
Refer to 2(c)ii
- xii. Modify Assignments  
Refer to 2(c)iii
- xiii. Hide Assignments  
Refer to 2(c)iv
- xiv. Move Students between Sections  
**1. Information Source** Section  $\bowtie$  Component  
**Select Condition** by student\_id, by start\_date, by component\_id  
**2. Insert** start\_date, end\_date, student\_id, section\_id  
**Information Source** section

- xv. Add Sections  
**Insert** section\_name, component  
**Information Source** section
- xvi. Modify Sections  
**Update** section\_name, hidden=false, instructor\_id  
**Select Condition** by section\_id  
**Information Source** section
- xvii. Hide Sections  
**Update** section\_name, hidden=true, instructor\_id  
**Select Condition** by section\_id  
**Information Source** section
- xviii. Assign Instructors to Sections  
Integrated in 2(d)xvi.
- xix. Assign Grades  
Refer to 2(b)i
- xx. View/Export Overall Course Grades  
Refer to 2(c)v
- xxi. View/Export Overall Component Grades  
Refer to 2(c)vi
- xxii. View/Export Course Rosters  
Refer to 2(c)vii
- xxiii. View/Export Component Rosters  
Refer to 2(c)viii
- xxiv. Import Final Course Grades  
**Update** final\_grade  
**Select Condition** by offering\_id, by student\_id  
**Information Source** offering\_student
- xxv. Import Final Component Grades  
**Update** final\_grade  
**Select Condition** by component\_id, by student\_id  
**Information Source** component\_student

## 6.6 Business Rules

In this section, the business rules in the client organization are listed. They are classified by the roles in the client organization.

- For Students
  - At any given time, a student can participate in at most one section within each component.
  - A student cannot view the roster of the course, component or the section.
  - A student cannot view the grades for any other students.

- For Instructors
  - An instructor cannot view or modify any grades associated with students belonging to sections other than the ones he/she teaches.
  - An instructor can only view the roster of sections that he/she can access.
  - An instructor can only assign/modify grades of students for sections that he/she can access.
- For Professors
  - A professor can only view and modify assignments, roster and grades within courses that he/she teaches.
- Miscellaneous Constraints
  - A student can be an undergraduate instructor.
  - A student from a previous semester can enroll for the same course in a following semester to finish incomplete work in a subsets of the components.

## 6.7 Maintenance Requirements

Maintenance requirements involved in the project covers the following two aspects:

- Backup
  - The software platform will provide the backup mechanism in the MySQL database management system.
  - The client will come up with a backup policy.
- Clean-up
  - The system will provide a mechanism to hide outdated data entries.
  - The client will come up with a clean-up policy.

## 6.8 Qualitative Requirements

In this section the qualitative requirements, which do not directly correspond to operations or actions are described. As indicated in Section 3, the client organization requires a specifically designed system to precisely represent the unique course structure and to efficiently manage the grades assigned to the students over years. To achieve the goal, the system should meet the following qualitative requirements:

- Efficiency
  - The grades will be managed efficiently.
  - The course structure will be organized and precisely represented by the system.
  - The client will not need any workarounds.
  - Interface will be concise and precise for all users.

- Portability
  - The users will be able to access the system from various software and hardware platforms.
  - The system itself can be deployed to various hardware platforms.
- Reliability
  - The system will not crash on software bugs.
  - Hardware failures and software platform failures will be recoverable.
- Security
  - Grades of individual students are kept private . Each individual student can only view their own grades.
  - Without appropriate permissions, users cannot access any data in the system.

## **6.9 User Interface**

The proposed system is virtually made up by two subsystems, the Gateway subsystem for regular users; and the management system(Administrator Gateway) for the administrator. Only key screens of the user interface design are presented in this document.

### **6.9.1 Conventions**

The user interface follows the conventions of web pages:

- Hyperlinks for switching screens.
- Buttons for posting a request to the system.
- Tab key for switching between Hyperlinks, Buttons and other HTML form elements in page layout sequence.
- The back button of the browser can cancel an unposted request.

## 6.9.2 Login

The Login screen is where a user logs into the system. Depending on the type of user, the screen is redirected to either 6.9.3 or 6.9.8 after authentication.

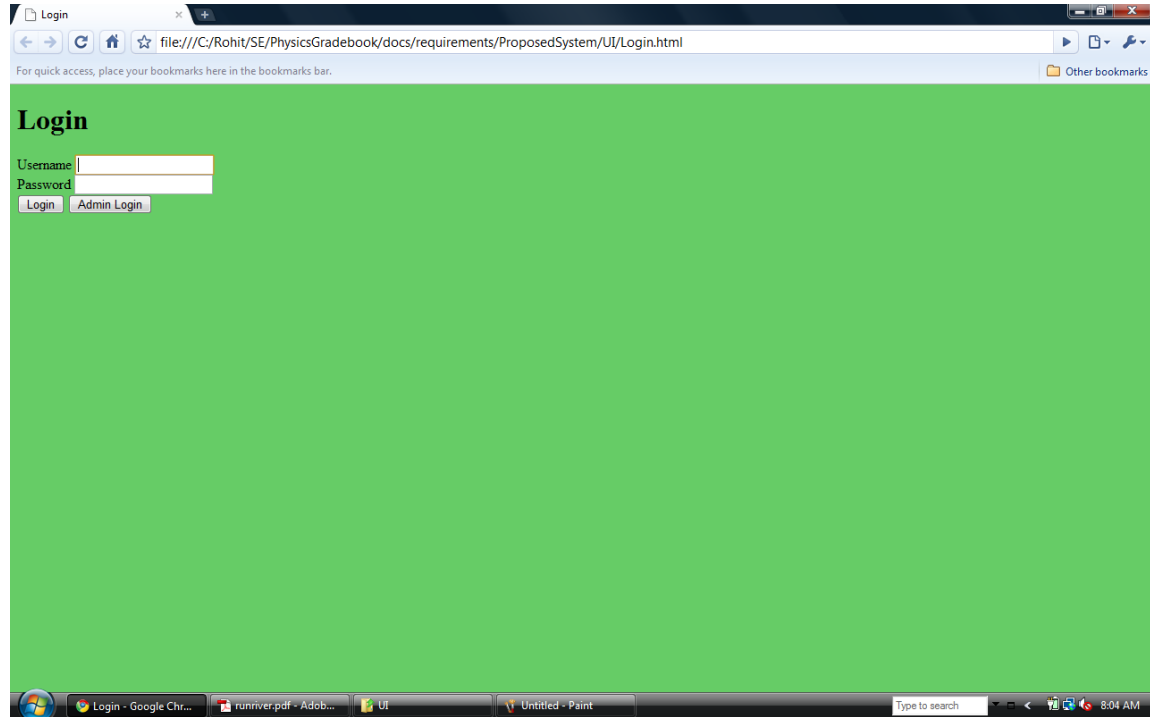


Figure 34: User Interface for Login

Event	Action
Click Login Button	Go to 6.9.3
Click Admin Login Button	Go to 6.9.8

Table 1: Login Screen



### 6.9.3 Gateway

The Gateway screen is the gateway for students, instructors and professors to access the system. The content in this screen depends on the roles the user plays in the system. Three sections are displayed for Instructor, Professor and Student.

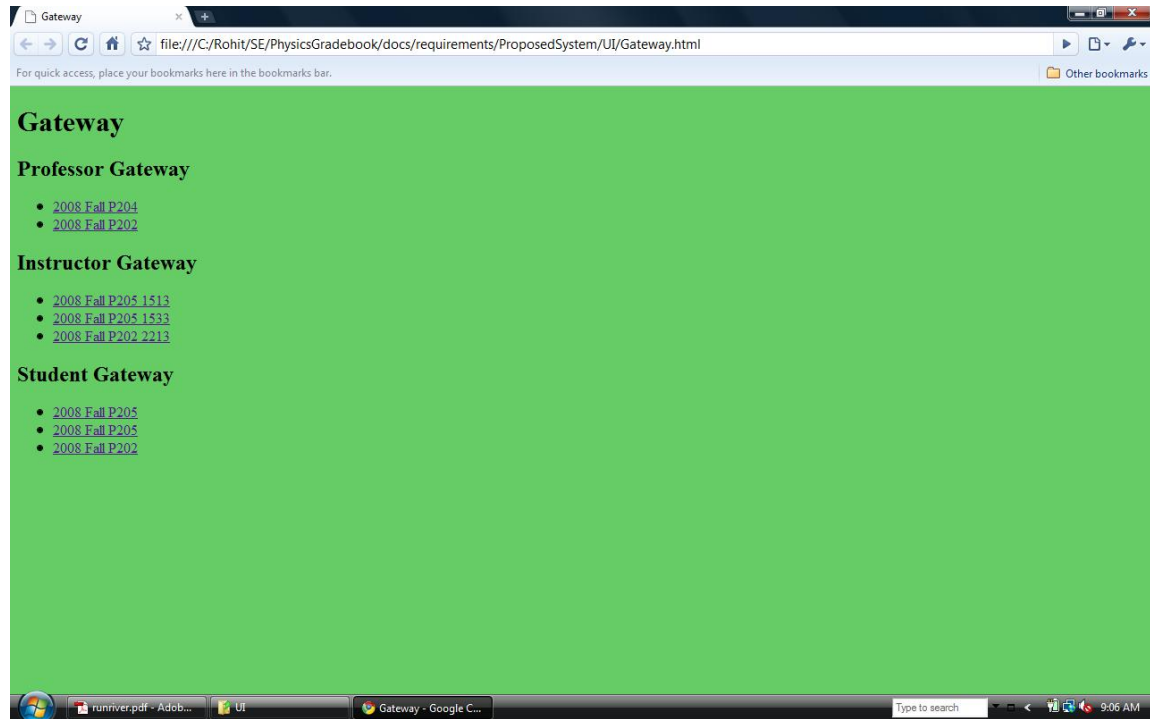


Figure 35: Gateway Screen for Regular Users

Event	Action
Click a Hyperlink in Professor Gateway	Go to 6.9.5
Click a Hyperlink in Instructor Gateway	Go to 6.9.7
Click a Hyperlink in Student Gateway	Go to 6.9.4

Table 2: Gateway Screen for Regular Users

#### 6.9.4 Offering - Student

A student views his grades for all the assignments and final grades in any given offering of a course.

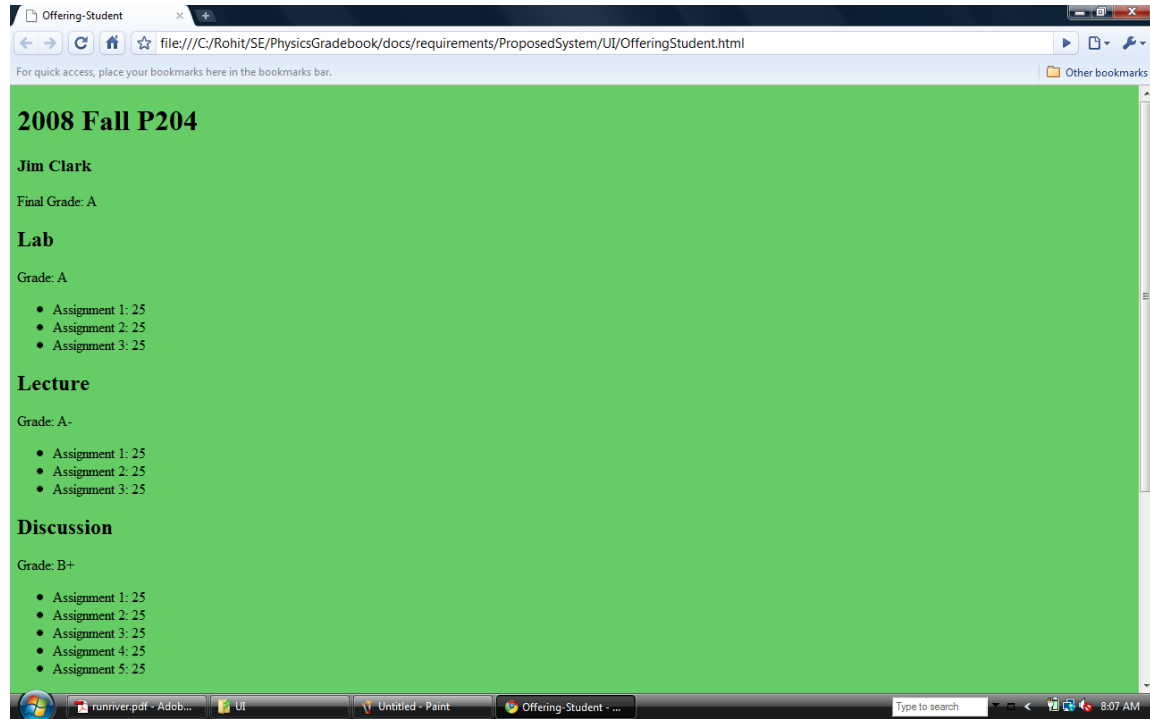


Figure 36: Offering Screen for a Student

Event	Action
N/A	N/A

Table 3: Offering Screen for a Student

### 6.9.5 Offering - Professor

A professor manages the assignments and grades in an offering with this screen.

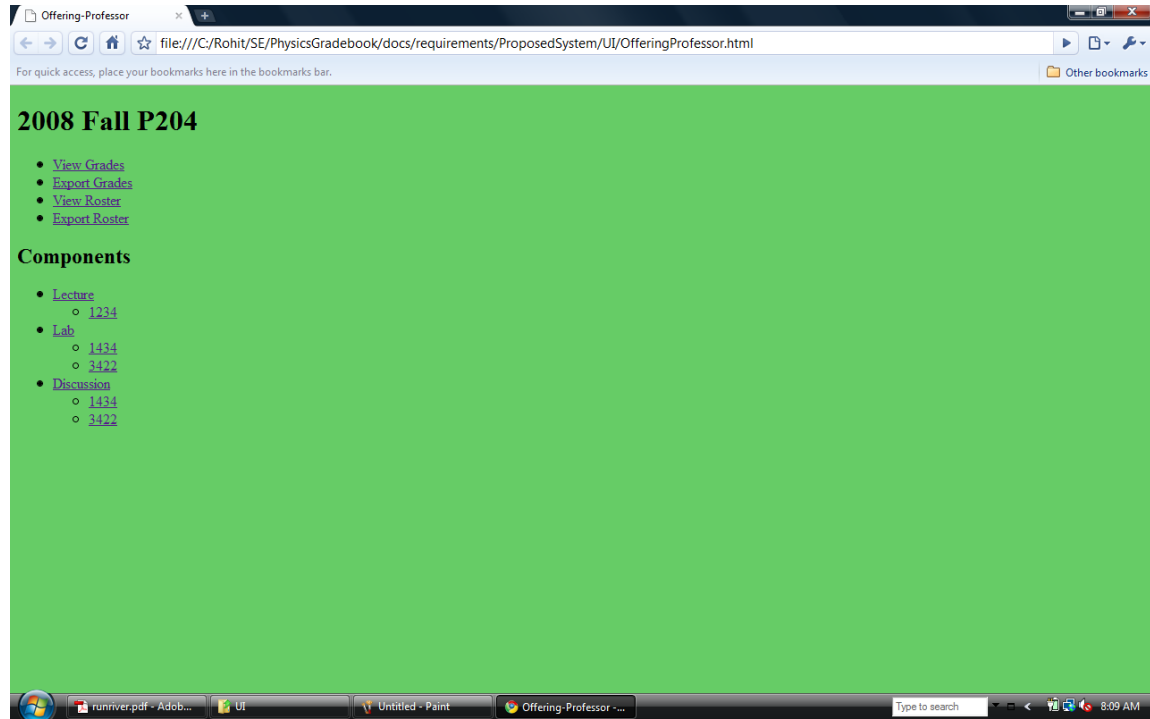


Figure 37: Offering Screen for Professor

Event	Action
Click View Grades	Show a list of all grades in the offering
Click Export Grades	Export a list of all grades in the offering
Click View Roster	Show the roster of the offering
Click Export Roster	Export the roster of the offering
Click Component Hyperlink	Go to 6.9.10
Click Section Hyperlink	Go to 6.9.7

Table 4: Offering Screen for Professor

## 6.9.6 Component - Professor

A professor manages the assignments and grades in a component with this screen.

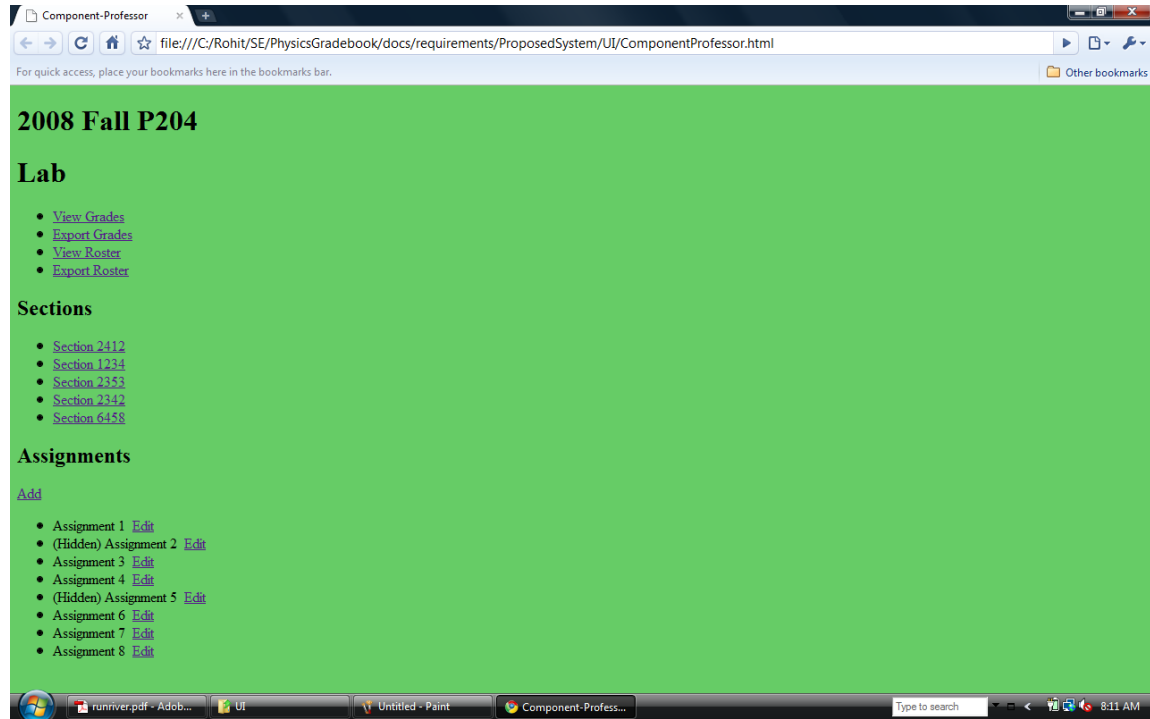


Figure 38: Component Screen for Professor

Event	Action
Click View Grades	Show a list of all grades in the component
Click Export Grades	Export a list of all grades in the component
Click View Roster	Show the roster of the component
Click Export Roster	Export the roster of the component
Click Add	Add a new assignment
Click Edit	Modify an existing assignment
Click Section Hyperlink	Go to 6.9.7

Table 5: Component Screen for Professor

6.9.7 Section - Instructor

An instructor or professor manages the grades in a section with this screen.

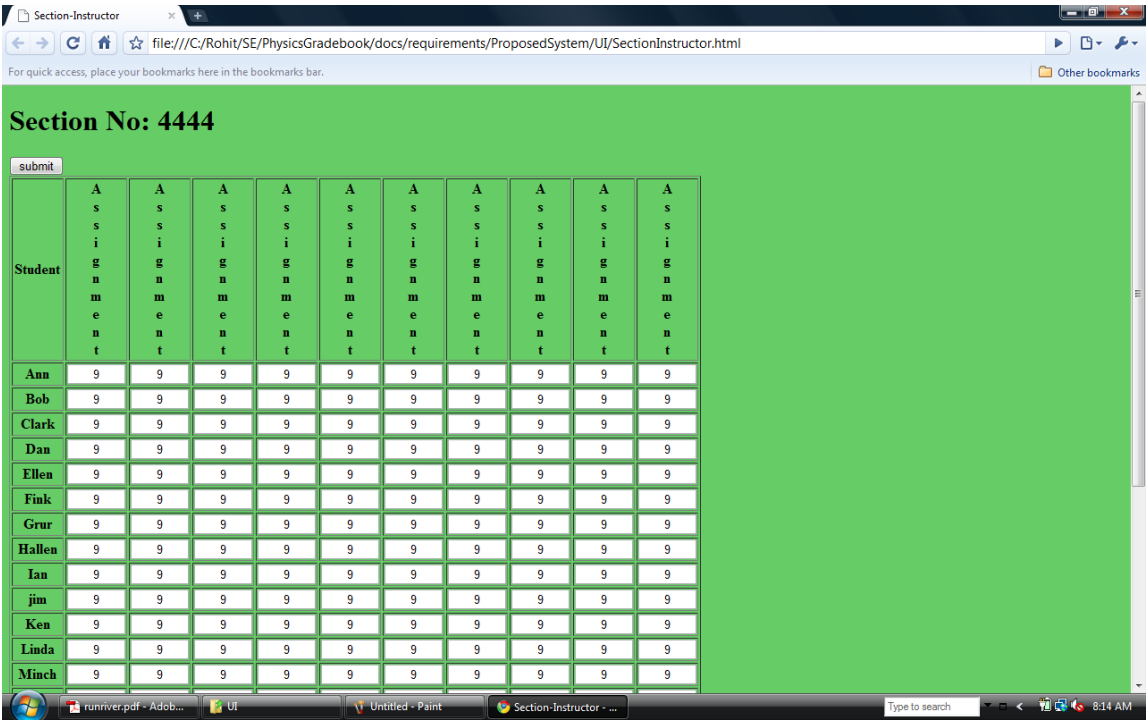


Figure 39: Section Screen for Instructor

Event	Action
Click Submit	Update the grades of students for assignments

Table 6: Section Screen for Instructor

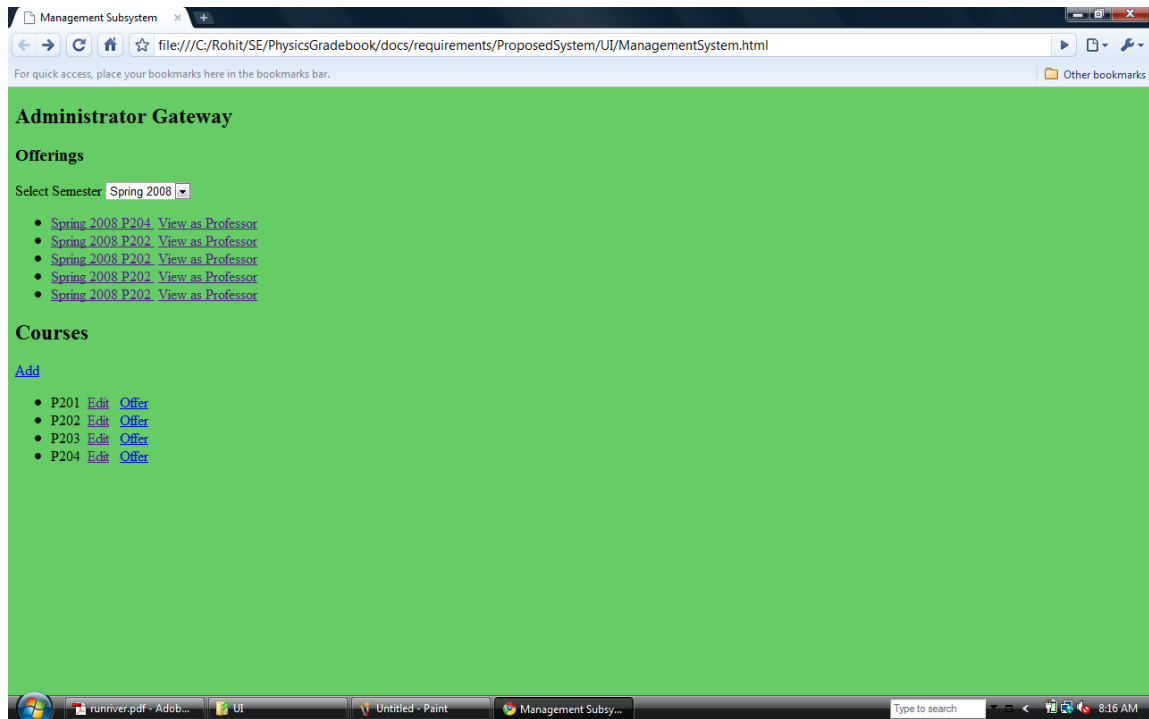


Figure 40: Administrator Gateway Screen

### 6.9.8 Administrator Gateway

Administrator Gateway screen is the screen for the management system.

Event	Action
Click Offering Hyperlink	Go to 6.9.9
Click Add	Add a new Course
Click Edit	Modify an existing Course
Click Offer	Offer a course in a semester

Table 7: Administrator Gateway Screen

## 6.9.9 Offering - Administrator

The administrator manage an offering in this screen.

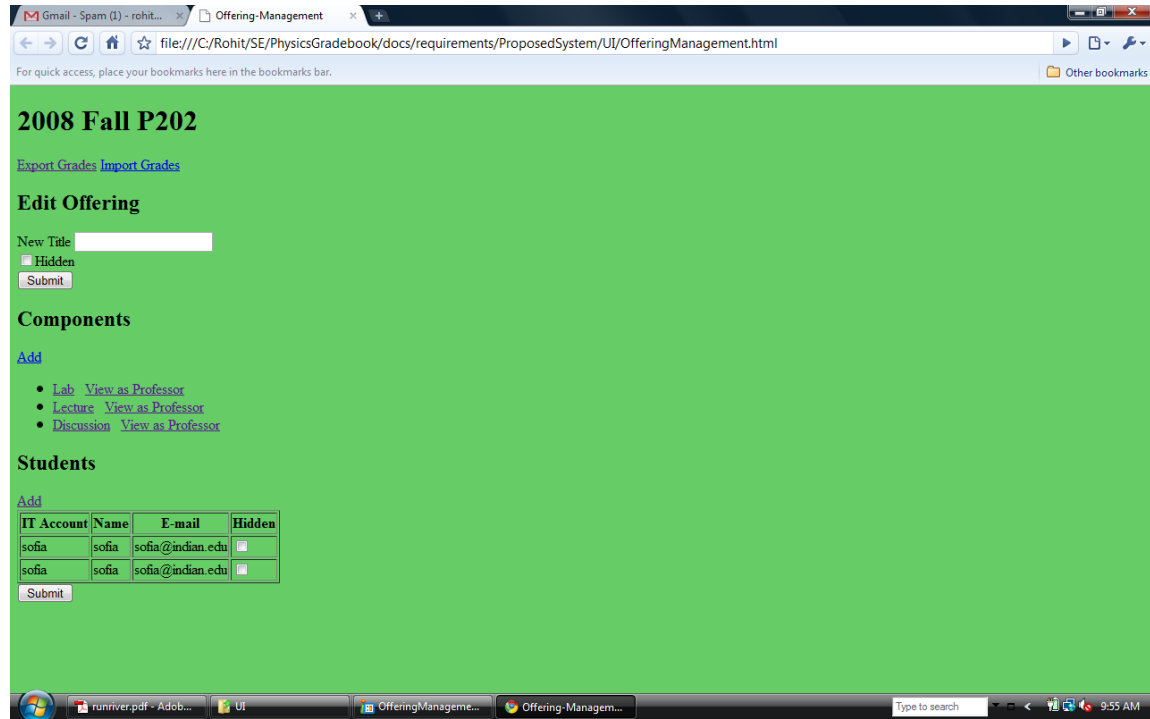


Figure 41: Offering Screen for Administrator

Event	Action
Click Export Grades	Export grades
Click Import Grades	Import grades
Click Hidden checkbox in Edit Offering	N/A
Click Submit button in Edit Offering	Modify existing offering
Click Component Hyperlink	Go to 6.9.10
Click Add in Components	Add new component to a course
Click View as Professor	Go to 6.9.5
Click Add in Students	Add a student to the offering
Click checkbox under Hidden column in Students	N/A
Click Submit button in Students	Hide student from roster of the offering

Table 8: Offering Screen for Administrator

### 6.9.10 Component - Administrator

The administrator manage a component in this screen.

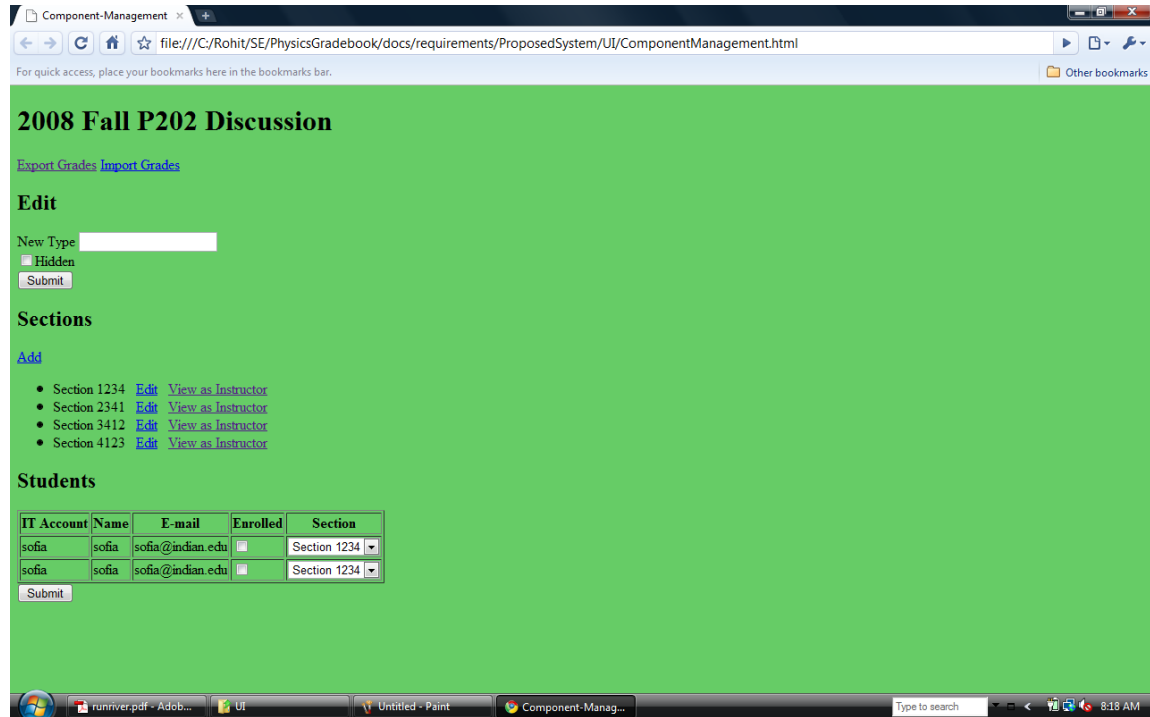


Figure 42: Component Screen for Administrator

Event	Action
Click Export Grades	Export grades
Click Import Grades	Import grades
Click Hidden checkbox in Edit Component	N/A
Click Submit button in Edit Component	Modify this Component
Click Edit in Sections	Modify a Section.
Click Add in Sections	Add a new Section to this Component
Click View as Instructor	Go to 6.9.7
Click Add in Students	Add a student to the offering
Click Checkbox Students	N/A
Click an Item in the Dropdown List in Students	N/A
Click Submit button in Students	Change the section and enrollment of students

Table 9: Component Screen for Administrator



## **6.10 Installation**

The proposed system is a complete replacement of the current system. The client will immediately switch to the proposed system when deployed. A transition phase is not needed, which has two implications:

- The data from the current system is not going to be transported to the proposed system.
- A period of concurrent operation is not expected.

The users of the system are expected to change each semester. The team will prepare the primary client on the usage the entire system. The primary client is responsible for training other users based on roles for the relevant aspects of the system at the beginning of each semester.

## **6.11 Limitations**

The proposed system will not implement the following functionalities:

- Calculating Final Grades.
- Reporting Final Grades to the Registrar Office.

The secretaries and the primary client will implement these functionalities manually as processes external to the proposed system.

## 7 Quality Assurance Plan

Rev 374, November 1, 2008.

### 7.1 Purpose and Scope

This Quality Assurance Plan (QAP) provides a framework for developing an effective, reliable, and maintainable software system for the IU Physics Gradebook System project. The QAP requires that certain procedures be followed and certain documents be written during the course of the project. The QAP is written in general accordance with the IEEE standard STD 730-1984 [3].

This QAP covers the the user interfaces of the IU Physics Gradebook System. The system is used by the Physics Department of Indiana University for keeping the grades for its students over years.  
resersers

### 7.2 Organization, Tasks, and Responsibilities

The project requires the following designated roles:

- Quality Assurance Monitor (QAM) - to be assigned
  - The QAM is appointed for the project by the course management. The role of the QAM will be:
    - \* To ensure that this QAP is in accordance with the general guidelines.
    - \* To ensure that the project is in compliance with its plans.
- Program Librarian (PL) - Yu Feng
  - The PL will maintain the Released Software Library (see 7.6).
- Quality Assurance Recorder (QAR) - Rohit Chandran
  - The QAR will maintain the Change Log (see 7.2) and Software Verification Summary (see 7.8).

### 7.3 Documentation Required

The following documents are required for the project:

- Requirements Specification
- Quality Assurance Plan
- Preliminary Design Description Test Plan (Software Verification/Validation and Acceptance Plan)
- Detailed Design Description
- User's Documentation
- Programmer's Reference Manual
- Change Log

- Software Verification Summary
- Computer Program Development Plan

The Requirements Specification, Preliminary Design Description, Quality Assurance Plan, Detailed Design Description, and Test Plan are the documents produced at project Milestones 2, 3, and 4 [2].

The User's Documentation must include the User's Reference Manual, the training for the users will be done by the Client.

The Programmer's Reference Manual must include sections corresponding to:

- a global static description, with
  - entity-relationship diagrams
  - data dictionary
  - procedure hierarchy charts
- a global dynamic description, with
  - Data Flow Diagrams (DFD)
- per module documentation, with
  - structure charts {HIPO, Warnier-Orr, or equivalent}

The Programmer's Reference Manual must provide explicit connections from items in the specifications and requirements through the logical and physical design to implementation.

The Change Log will document all changes to the final product that affect milestones already past. The Change Log must track the effects of these changes to all past milestones. As appropriate for the span of changed milestones, entries in the Change Log must include sections on requirements, specifications, design, and implementation. The Change Log will also record any reported problems with accepted modules and the corrective actions taken for these problems (see 7.7).

Each entry in the Change Log shall include

- date reported - the date when change was requested or problem observed
- originator - the individual(s) who requested the change or observed the problem reported
- condition - the reasons for the change or symptoms of the problem date
- action taken - when the change was implemented or corrective action completed description of action taken
- components affected - milestones, modules, etc. affected by the action
- implementer - the individual(s) who took the change/corrective action
- tests - description of the tests or verification/validation taken with respect to this action

The Software Verification Summary is discussed in 7.6, 7.9 and 7.12.

## 7.4 Standards, Practices, and Conventions

All programs and modules developed in the course of this project must conform to standards on code development, organization, and internal documentation.

### 7.4.1 Program Standards

- strict convention should be followed,
- all variables should be at least 4 characters long,
- number of lines in a procedure should be limited,
- number of procedures in a file should be limited,
- if Object Oriented method is used in programming, number of methods in a class should also be limited,
- the interface between modules should be minimized,
- the use of global variables should be minimized.

All programs and modules developed in this project will be subject to unit test, acceptance test, and verification and validation according to the procedures described in the Software Verification/ Validation and Acceptance Plan. The Software Verification/Validation and Acceptance Plan will contain the following information: Requirements to be verified Software-code test plan Type of test: function, performance, stress, structure Test assumptions Requirements being tested Test cases used Expected outcome Test completion criteria Acceptance criteria Document verification plan (e.g., user's manual, programmers manual, internal documentation) System integration plan

The general requirements for testing, verification, and validation include the following:

- All modules must undergo unit test before being included in the library of released software (see 7.6).
- The author or authors of a module may not be wholly responsible for unit testing of the module. In particular other team members must participate in defining the tests and evaluating the test results and may, if appropriate, be required to participate in the running of the tests.
- An Acceptance Test, as specified in the Software Verification/ Validation and Acceptance Plan, must be conducted before the completed system is delivered to the client.
- All documents produced in the course of this project must be verified prior to project completion.
- A Software Verification Summary shall be maintained to record all tests and verification/validation steps and the results of these actions. This Software Verification Summary shall be maintained by the QAR (see 7.2).

## **7.5 Reviews and Audits**

The following reviews and audits will be conducted:

- Requirements Specification Review
- Design Review
- Testing Audit

The Reviews will occur during the development of the respective milestone documents. The Reviews will cover a draft of those documents and will be attended by other teams and members of the course management.

The Testing Audit will be conducted by the QAM as part of the product acceptance process. The primary activity of the Testing Audit will be the review of the Software Verification Summary to insure that all required tests and verification steps have been successfully completed. The Change Log will also be reviewed to insure that all issues are resolved.

## **7.6 Configuration Management**

Configuration management must be controlled by the PL (see 7.2) who will maintain a Released Software Library. A software module is "released" after it has passed the unit tests specified in the Software Verification/Validation and Acceptance Plan (see 7.4). All development and testing on a given module must use other modules from the Released Software Library only.

For configuration management, subversion is used. The branching and tagging facilities of subversion meet the configuration management requirements.

Notification emails are sent for new software releases and the release history is captured in a release note.

## **7.7 Problem Reporting and Corrective Action**

All problems with released software modules and any corrective action related to these problems must be recorded in the Change Log (see 7.3).

## **7.8 Tools, Techniques, and Methodologies**

- Project progress control is done with the help of weekly logs made by the secretary, Rohit and provided to the supervisor.
- Project action follow-up is also done with the help of weekly logs made by the secretary, Rohit and provided to the supervisor.
- Documentation is done and internally edited with the use of Latex.

## **7.9 Code Control**

Configuration management will include keeping the Released Software Library, which contains standard, current-version software (see 7.6).

- The subversion facility provided by Google Code is used to ensure that all modifications are identifiable and traceable.
- The development work is distributed on the team members' computers.
- The released modules will be immediately deployed on a machine provided by the primary client in the Physics Department.
- The source code of the standard libraries will reside on Google Code, network-accessible from the stations used in actual development. This will provide all team members with consistent access to the latest versions of released software.

## **7.10 Media Control**

Every team member has his own working copy of the source code. This itself would act as a backup. If Google Code is down for a long period of time, a repository can be reconstructed from any of the the three working copies.

## **7.11 Supplier Control**

Supplier control is not applicable to this project.

## **7.12 Records Collection, Maintenance, and Retention**

The QAR must maintain the Change Log (see 7.3) and Software Verification Summary (see 7.8).

## **8 Appendices**

Rev 374, November 1, 2008.

- The Project Plan is provided as a separate page.

## **9 Client Comments**

Rev 374, November 1, 2008.



## References

- [1] Feasibility Study for Physics Gradebook System.
- [2] P465-6 & P565-6 Software Engineering for Information Systems I & II: Information Packet, Computer Science Department, Indiana University, 2008.
- [3] IEEE Standard for Software Quality Assurance Plans (STD 730-1984), Inst. of Electrical and Electronics Engineers, New York, 1984.
- [4] Requirement Specifications of Sakai Gradebook. <https://source.sakaiproject.org/svn/gradebook/trunk/xdocs/specs23/index.htm>
- [5] Information about Integrating CAS with a website. <http://kb.iu.edu/data/atfc.html>