

Preliminary Design IU Physics Gradebook System

Yu Feng¹

Rohit Chandran²

Naga Rekha Malae³

¹Physics Department, Indiana University, Bloomington IN 47401

²Department of Computer Science, Indiana University, Bloomington IN 47401

³Department of Computer Science, Indiana University, Bloomington IN 47401

Executive Summary

This document covers certain technical aspects such as the architecture and database design of the IU Physics Gradebook system.

The architecture of the proposed system is described by means of a structural system model, a control model and a subsystem decomposition model. In addition, modules to be used for the system and the interaction between the modules are explained. These modules are derived from the Data Flow Diagrams specified in the Requirements Specification Document. These include the template library, user interface generators, servlets, error handling library and the core libraries.

The ER diagram specified in the requirements specification document is translated into an equivalent relational database schema. This schema identifies the fields for each of the tables along with the relationships amongst these tables. For the purpose of identifying each row in a table uniquely, a primary key is also specified for each table. The system will make use of the advantages of the database platform for data consistency and automatic crash recovering.

The user interface of the proposed system is described in order to meet the client's functionality requirements. The operations performed on each screen and the navigation path between these screens is also explained. Certain common appearance and layout characteristics are mentioned.

The preliminary test plan for the system has been proposed. The plan spans from the testing operations to how these operations will be done.

Coding specifications for module decomposition rules, module header formats, in-line commenting and indentation are standardized. These coding standards are required by the QA plan to guarantee the quality of the information system.

Exceptions that may occur are mentioned and how they are handled by the system is also explained.

Additional and modified requirements of the client have been appended to the end of the document.

Contents

1	Introduction	5
2	Requirement Specifications	6
2.1	Client Background	6
2.2	Project Goals	6
2.3	Environment	6
2.4	Data Flow Diagrams	6
2.5	Entity-Relationship Diagram	6
2.6	Functionality Requirements	6
3	Architecture and Modules	7
3.1	Architecture	7
3.1.1	Structural System Model	7
3.1.2	Control Flow Architecture	8
3.1.3	Component Architecture	9
3.2	Module Coupling	10
3.2.1	Interaction between Modules	10
3.2.2	Convention Type Names	10
3.3	Template Library	12
3.4	User Interface Generator	13
3.4.1	Trivial Generator	13
3.4.2	Gateway Generator	13
3.4.3	Administrator Gateway Generator	13
3.4.4	Modify Grades Generator	13
3.4.5	Edit Roster Generator	13
3.4.6	Offering Management Generator	14
3.4.7	Report Grades Generator	14
3.5	Servlets	14
3.5.1	Login Servlet	14
3.5.2	CAS Callback Servlet	14
3.5.3	Modify Grades Servlet	14
3.5.4	Component Management Servlet	15
3.5.5	Export Servlet	15
3.5.6	Import Servlet	15
3.5.7	Course Management Servlet	15
3.5.8	Offering Management Servlet	15
3.5.9	User Management Servlet	16
3.6	Error Handling Module	16
3.7	Core Libraries	16
3.7.1	Session Management	16
3.7.2	User and User Permission Authorization	17
3.7.3	Data Manipulation	18

4	Data Design	21
4.1	Introduction	21
4.2	PERSON Table	21
4.3	COURSE Table	22
4.4	OFFERING Table	22
4.5	SEMESTER Table	23
4.6	COMPONENT Table	24
4.7	COMPONENT_TYPE Table	25
4.8	COURSE_DEFAULT_COMPONENTS Table	25
4.9	ENROLL_COMPONENT Table	26
4.10	ENROLL_OFFERING Table	27
4.11	JOIN_SECTION Table	28
4.12	SECTION Table	30
4.13	TEACH_SECTION Table	31
4.14	SESSIONS Table	32
4.15	ASSIGN_GRADES Table	32
4.16	LATEST_ASSIGNED_GRADES Table	34
4.17	ROLE_TYPE Table	35
4.18	ASSIGNMENT_TYPE Table	35
4.19	ASSIGNMENT Table	36
5	User Interface	38
5.0.1	Welcome	39
5.0.2	Non-CAS Login	40
5.0.3	CAS Login	41
5.1	Administrator	42
5.1.1	Administrator Gateway	42
5.1.2	Add Offering	43
5.1.3	Offering Management	44
5.1.4	Add Component to Course Offering	46
5.1.5	Add Section to Component	46
5.1.6	Add Student to a Course Offering	47
5.1.7	Add Assignment to Component	47
5.1.8	Modify Assignment in Component	48
5.1.9	Modify Course	48
5.1.10	Export Grades	50
5.1.11	Export Offering Roster	50
5.1.12	Export Component Grades	50
5.1.13	Export Component Roster	50
5.1.14	Import Offering Grades	51
5.1.15	Import Component Grades	51
5.1.16	Modify Grades	51
5.1.17	View Offering Grades	52
5.1.18	View Component Grades	53
5.1.19	View Section Grades	53
5.2	General Users	54

5.2.1	User Gateway	54
5.2.2	Report Grades Screen	55
6	Exception Handling	56
6.1	Platform Failures	56
6.2	Runtime Exceptions	57
6.3	Error Handling Schema	57
7	Test Plan	59
7.1	Unit Testing	59
7.2	Security Testing	59
7.3	Functionality Testing	59
7.4	Integration Testing	59
7.5	User Acceptance Testing	60
7.6	Performance Testing	60
7.7	Regression Testing	60
7.8	Multi-User Testing	60
7.9	Usability Testing	60
7.10	Operation Acceptance Testing	60
7.11	Database testing	61
	Index	67
	Nomenclature	69
A	List of Requirements Changes	71
	Appendices	71
B	Coding Standards and Conventions	72
B.1	Introduction	72
B.2	Modules	72
B.3	Module Decomposition	72
B.4	Headers	72
B.4.1	PHP Headers	72
B.4.2	JavaScript Headers	73
B.4.3	CSS Headers	73
B.5	Commenting and Indentation	73
B.5.1	File Header	73
B.5.2	Member Function Header	73
B.5.3	In-line Commenting	74
B.5.4	Indentation Rules	74
B.5.5	Line Breaking Rules	75
B.5.6	Blank/Space Rules	75
B.6	Naming Conventions	75
B.7	Database Schema	75
B.8	File Naming Conventions	76

B.9 Local Variables	76
B.10 Global Variables	76
B.11 User Interface	76
B.12 Form Layout	76
B.13 Form Navigation	77
B.14 Informative Messages	77
B.15 Key Bindings	77

1 Introduction

The IU Physics Gradebook system is being developed in order to replace OnCourse for the purpose of storing and managing grades more efficiently.

As part of this effort, this document highlights and elaborates on certain design details of the proposed system.

Section 2 of the document is carried forward from the Requirements Specification and these bring to light details about the client background, project goals, environment along with Data Flow Diagrams and Entity-Relationship diagrams for both the current and proposed system.

Section 3 details the architecture design of the proposed system. This is done by means of a structural system model, which is explained in terms of the services provided by it, a control flow model and also a component architecture. The Controller within this component architecture consists of five modules, namely, the template library, user interface generators, servlets, error handling library and the core libraries. Details of these modules along with their interaction are specified.

The ER diagram specified in the requirements specification document is translated into an equivalent relational database schema in Section 4. This schema identifies the fields for each of the tables along with the relationships between these tables. For the purpose of identifying each row in a table uniquely, a primary key is also specified for each table. Foreign key constraints are also specified. The tables are designed with the InnoDB structure that keeps the data consistent and also automatically recovers from a crash.

The user interface of the proposed system that meets the client's functionality requirements is brought to fore with the help of details of operations performed on each screen. In addition, navigation between these screens along with certain appearance and layout characteristics are mentioned in section 5.

Exceptions that may occur are mentioned and how they are handled by the system is also explained in detail with error handling modules in section 6.

Section 7 has the test plan for the system which mentions the different testing operations to be carried out and how these will be done.

Coding specifications for module decomposition rules, module header formats, in-line commenting and indentation are standardized in Appendix B. These coding standards, as required in the Quality Assurance plan in previous documents, provide a quality information system to the client.

Additional and modified requirements of the client have been added to the end of the document as the Appendix A.

2 Requirement Specifications

Client Background, Project Goals, Environment, Data Flow Diagrams, Entity Relation Diagram, Functionality Requirements are not included in the Preliminary design document because there are no specific changes made to these sections. However, significant changes have been made to the User Interface which are covered in this document.

2.1 Client Background

The client background is described in [2].

2.2 Project Goals

The project goals is described in [2].

2.3 Environment

The environnement is described in [2].

2.4 Data Flow Diagrams

The Data Flow diagrams are described in [2].

2.5 Entity-Relationship Diagram

The ER diagram are described in [2].

2.6 Functionality Requirements

The functionality requirements are described in [2]. The list of changes to the requirements are listed in the appendix of this document (Appendix A)

3 Architecture and Modules

3.1 Architecture

A structural system model, a control model and a sub-system decomposition model are designed to accomplish the overall architecture design of the system.

3.1.1 Structural System Model

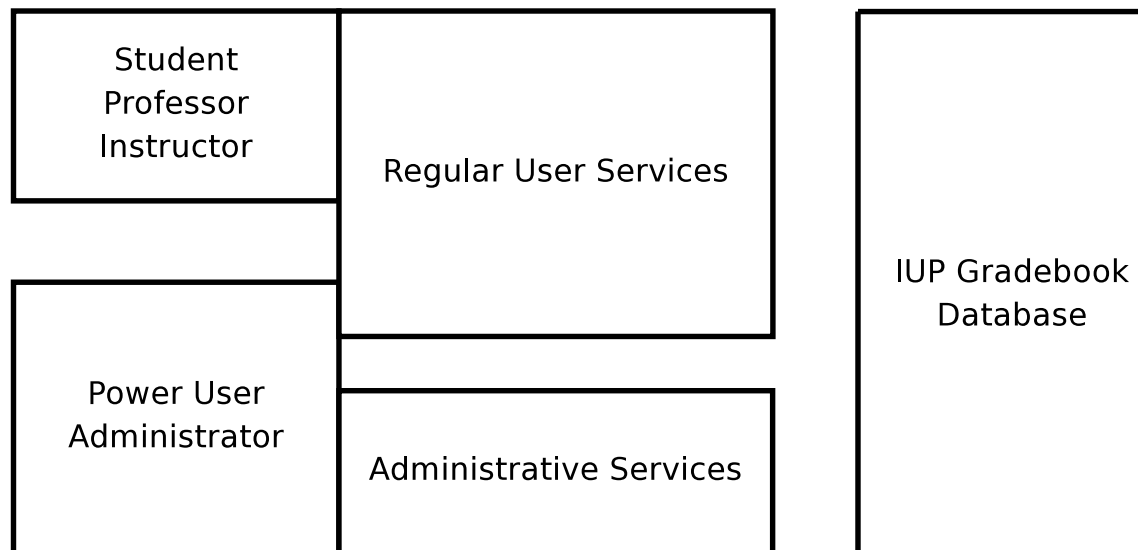


Figure 1: Architecture of the Services

In the above figure the common boundary stands for the interaction between users and the functionalities of the system.

From the user's perspective, the system is composed of two major services:

- Regular user services are focused towards the students, professors and instructors. The system provides services to enable
 - Viewing grades
 - Assigning grades
 - Manipulating assignments
- Administration services are focused towards the departmental staff. The system provides services to enable
 - Managing course hierarchy
 - Managing roster
 - Batch import/export of grades

3.1.2 Control Flow Architecture

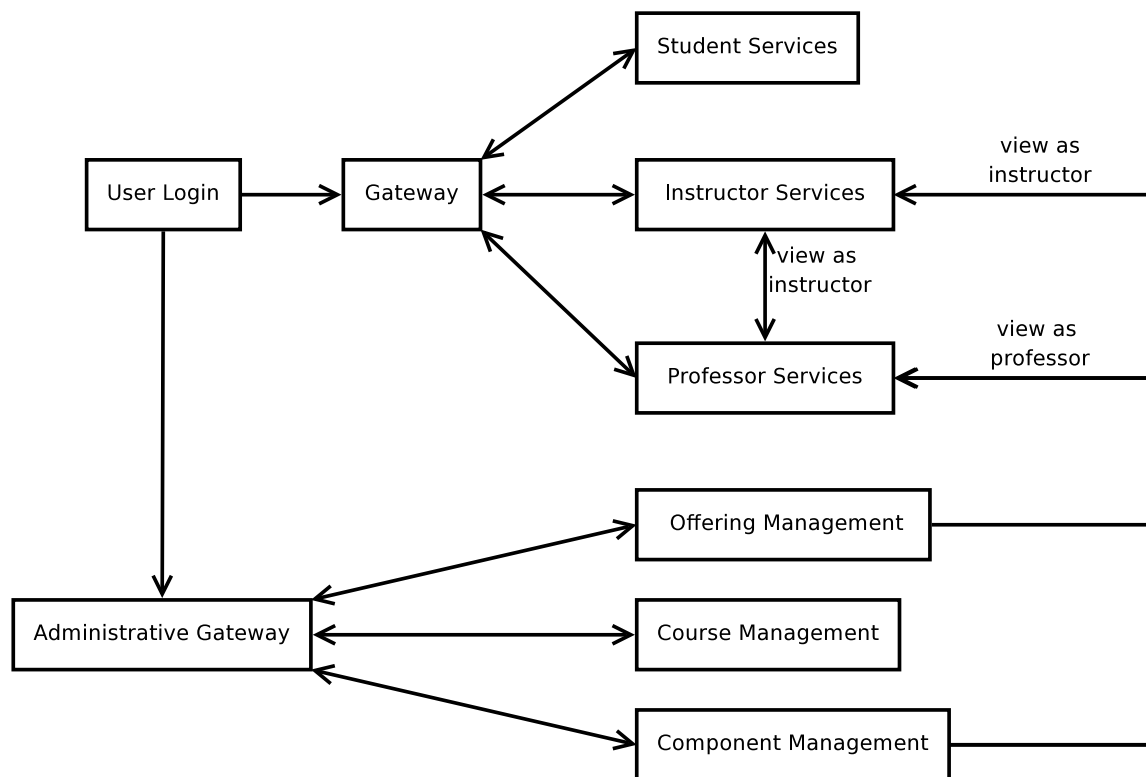


Figure 2: Architecture of the Control Flow

The control flow models the transition of the active service in the system for each interactive session.

3.1.3 Component Architecture

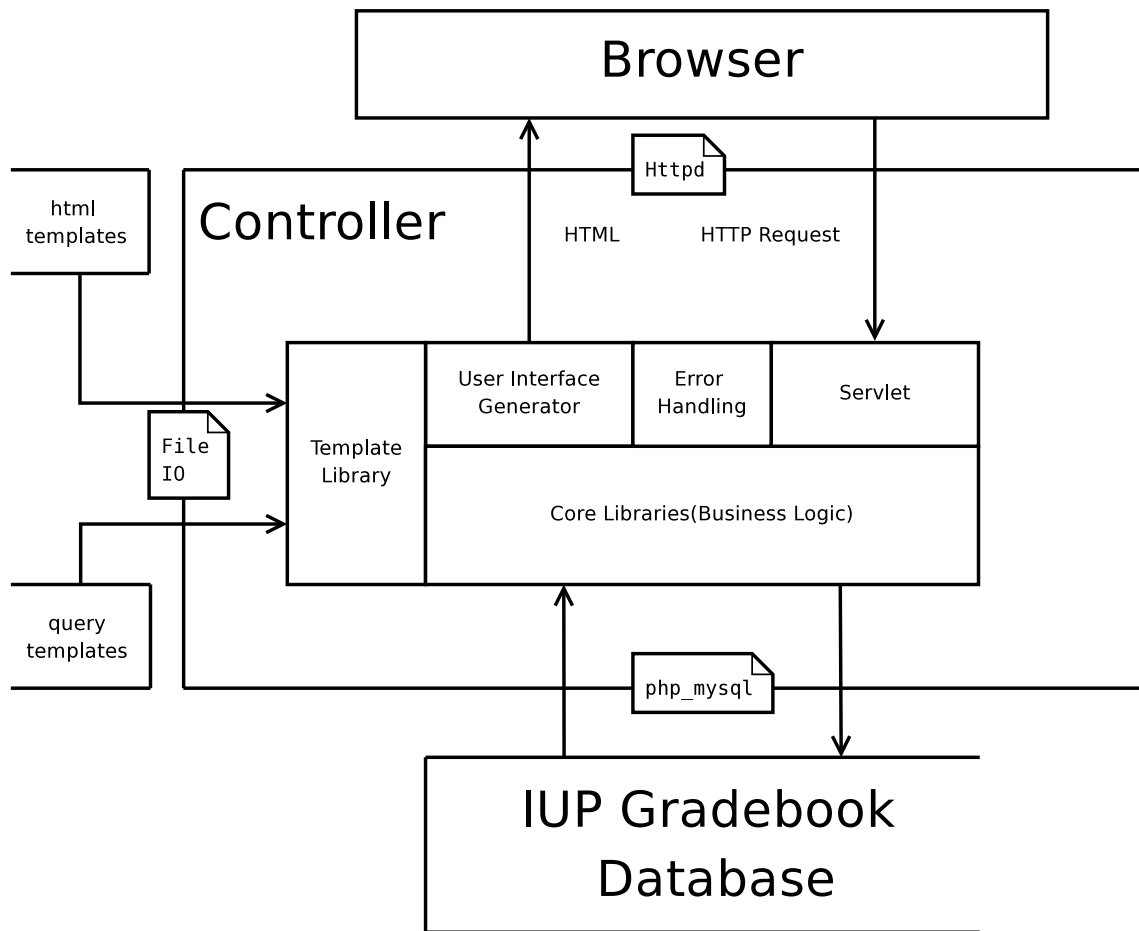


Figure 3: Architecture of the Components

The system is composed of three components which are shown in the above Figure.

Browser The web browser on the user's workstation directly interacts with the user. It must be noted that the development team has no control over the user's choice of browser. The team assumes the user of the system will use Mozilla Firefox(Gecko), Safari(WebKit), or Internet Explorer (Trident).

Database The DBMS system which provides fundamental storage services.

Controller The controller contains the business logic. It is responsible for resolving the requests from the browser, checking the validity of the requests, querying the database, and pushing the user interface to the browser.

The Controller contains five relatively independent modules:

- Template Library, a utility library that builds SQL queries or HTML pages from template files.
- User Interface Generators, server side scripts that generate the HTML files by merging the data obtained from the database and the page templates from the templates.
- Servlets, server side scripts that respond to the HTTP requests from the client browser but do not directly push any content to the browser.
- Error Handling library, the platform to support the error handling policies.
- Core Libraries
 - Session Management, an extension to the PHP session management library
 - User and Permissions, a collection of functions which are used for user validation and user permission authorization.
 - Data Manipulation, a collection of data manipulation routines.

The detailed hierarchy of the module design in the Controller is described in the following sections.

3.2 Module Coupling

3.2.1 Interaction between Modules

The adjacent boundaries of the modules indicate inter-module interactions.

The Template Library is used by the Core Libraries and User Interface Generators to generate the SQL queries and HTML pages respectively. The User Interface Generators and Servlets, where the client business is implemented, are supported by the functionalities of the Core Libraries. The Error Handling library provides a mechanism for error throwing and forwarding between the three libraries.

3.2.2 Convention Type Names

PHP Objects are used for communication between modules. These objects are built on-the-fly by associating the property name with property values. Classes are not implemented. Instead, a convention for the property names associated with named objects is set in this section.

The objects postfixed with 'key' should be sufficient to identify the corresponding entity in the database. A reference to the corresponding table structure in the data design is given when it is appropriate.

course_key An object representing the primary key of the COURSE table in the database design.

Field	Explanation
course_number	Refer to the Data Design

offering_key An object representing the primary key of the OFFERING table in the database design.

Field	Explanation
course_key	The foreign key to the course
professor_id	Refer to the Data Design
semester	Refer to the Data Design

component_key An object representing the primary key of the COMPONENT table in the database design.

section_key An object representing the primary key of the SECTION table in the database design.

Field	Explanation
component_key	The foreign key to the component
section_number	Refer to the Data Design

person An object to represent a person. Related to, but not limited to an entry in the PERSON Table.

Field	Explanation
id	The person_id of the person
last_name	Refer to the Data Design
first_name	Refer to the Data Design
email	Refer to the Data Design
cas_account	Refer to the Data Design
iup_account	Refer to the Data Design
active	Refer to the Data Design; can be undefined
grades	An array of grades; See below ????? for the definition of grade; can be undefined

grade An object representing an entry in the ASSIGN_GRADES table.

Field	Explanation
grade	Refer to the Data Design
comment	Refer to the Data Design
timestamp	Refer to the Data Design; can be undefined if it is for the latest grade
assigner	A person object describing the person that assigns the grade; grades and active should be undefined
assignment	An assignment object describing the assignment for which the grade is assigned

assignment An object representing an entry in ASSIGNMENT Table.

Field	Explanation
component_key	The foreign key to a component
title	Refer to the Data Design. Also referred as assignment_title
type	Refer to the Data Design

template An object representing a template. Created by the Template Library.

Field	Explanation
template_name	The name of the template
result	The result after the template substitution; private member and should not be accessed

error An object to encode a error. Created by the Error Handling module.

Field	Explanation
code	The code of the error; a string describing the main feature of the error, in capital letters, and separated by the underscore character
message	The formatted error message
level	The severity of the error; integer; lower the integer, greater is the severity

result A data type provided by `php_mysql` module to represent the result of an SQL query.

3.3 Template Library

Introduction The template library is a utility library that build SQL queries or HTML pages from template files.

Initialization Find the location for storing template files in the file system.

Finalization N/A (Not Applicable)

Interface Design

Name	<code>template_load</code>
Description	Load the template given by <code>template_name</code> into memory and return the template object
IN	<code>template_name, type</code>
OUT	<code>template</code>
Name	<code>template_get_child</code>
Description	Obtain a child template in a template
IN	<code>template, child_template_name</code>
OUT	<code>child_template</code>
Name	<code>template_reset</code>
Description	The template to an unparsed state, invalidate the result field
IN	<code>template</code>
OUT	N/A
Name	<code>template_assign_prepend</code>
Description	Parse the template, prepend template variables with their values; therefore, the template variables are preserved
IN	<code>template, array (variable => value)</code>
OUT	N/A
Name	<code>template_assign</code>
Description	Assign values to the template variables
IN	<code>template, array (variable => value)</code>
OUT	N/A
Name	<code>template_get_result</code>
Description	Obtain the result of the template substitution
IN	<code>template</code>
OUT	<code>string</code>

3.4 User Interface Generator

User Interface Generators are server side scripts that generate the HTML files by merging the data obtained from the database and the page templates from the templates. The persistence of sessions and user permissions checking are performed at this level. For a generator, the IN parameters are stored in the HTTP Request header, and the only OUT parameter is the generated HTML page.

3.4.1 Trivial Generator

Generator	Trivial Generator
Description	The trivial generator generates pages without accessing the database
IN	template_name, (other parameters depending on the template)
OUT	N/A
ERROR	ACCESS_DENIED, RESOURCE_NOT_FOUND if the template is not found

3.4.2 Gateway Generator

Generator	Gateway Generator
Description	It generates the Gateway page. Refer to the User Interface Section
IN	N/A
OUT	N/A
ERROR	N/A

3.4.3 Administrator Gateway Generator

Generator	Administrator Gateway Generator
Description	It generates the Gateway page for the Administrators. Refer to the User Interface Design
IN	N/A
OUT	N/A
ERROR	ACCESS_DENIED if the user is not a power user or administrator

3.4.4 Modify Grades Generator

Generator	Modify Grades Generator
Description	It generates the Modify Grades pages and View Grade pages. Refer to the User Interface Design
IN	offering_key, component_key, section_key, readonly
OUT	N/A
ERROR	ACCESS_DENIED, RESOURCE_NOT_FOUND

3.4.5 Edit Roster Generator

Generator	Edit Roster Generator
Description	It generates the View Roster page. Refer to the User Interface Design
IN	readonly, offering_key, component_key, section_key
OUT	N/A
ERROR	ACCESS_DENIED, RESOURCE_NOT_FOUND

3.4.6 Offering Management Generator

Generator	Offering Management Generator
Description	It generates the Offering Management page. Refer to the User Interface Design
IN	offering_key
OUT	N/A
ERROR	ACCESS_DENIED, RESOURCE_NOT_FOUND

3.4.7 Report Grades Generator

Servlet	Report Grades Generator
Description	It reports the grades of a student in an offering.
IN	offering_key
OUT	N/A
ERROR	ACCESS_DENIED, INVALID_INPUT

3.5 Servlets

Servlets are server side scripts that respond to the HTTP requests from the client browser but do not directly push any content to the browser. For servlets, the IN parameters are stored in the HTTP request header. They do not have OUT parameters. Rather, they redirect the browser to send requests to other Servlets or User Interface Generators. The persistence of sessions and user permission checking are performed at this level.

3.5.1 Login Servlet

Servlet	Login Servlet
Description	Perform the login actions, validating the user name and password, establish the session
IN	username, password
OUT	N/A
ERROR	INVALID_USERNAME, INVALID_PASSWORD

3.5.2 CAS Callback Servlet

Servlet	CAS Callback Servlet
Description	Finish the CAS login process, establish the session
IN	CASTicket
OUT	N/A
ERROR	

3.5.3 Modify Grades Servlet

Servlet	Modify Grades Servlet
Description	Modify the grades
IN	array(student with grades)
OUT	N/A
ERROR	ACCESS_DENIED, INVALID_INPUT

3.5.4 Component Management Servlet

Servlet	Component Management Servlet
Description	Modify the component, adding assignments etc
IN	component_key, operation, operation_parameters (depending on the operation)
OUT	N/A
ERROR	ACCESS_DENIED, INVALID_INPUT

3.5.5 Export Servlet

Servlet	Export Servlet
Description	Generate files for exporting
IN	offering_key/section_key, type_of_export
OUT	The generated file, containing the requested information.
ERROR	ACCESS_DENIED, INVALID_INPUT

3.5.6 Import Servlet

Servlet	Import Servlet
Description	Import information from a file.
IN	offering_key/section_key, type_of_import, file_content
OUT	Bach import information from a given file.
ERROR	ACCESS_DENIED, INVALID_INPUT

3.5.7 Course Management Servlet

Servlet	Course Management Servlet
Description	Modify the course
IN	course_key, operation, operation_parameters (depending on the operation)
OUT	N/A
ERROR	ACCESS_DENIED, INVALID_INPUT

3.5.8 Offering Management Servlet

Servlet	Offering Management Servlet
Description	Modify the offering, add components, etc
IN	offering_key, operation, operation_parameters (depending on the operation)
OUT	N/A
ERROR	ACCESS_DENIED, INVALID_INPUT

3.5.9 User Management Servlet

Servlet	User Management Servlet
Description	Add/Remove users
IN	user_id, operation, operation_parameters (depending on the operation)
OUT	N/A
ERROR	ACCESS_DENIED, INVALID_INPUT

3.6 Error Handling Module

Introduction The error handling module is the platform to support the error handling policies described in Section . An error typically contains an error code and an error message.

Initialization Initialize the error table with error codes and the format of the error messages.

Finalization N/A (Not Applicable)

Interface Design

Name	error_new_from_http_request
Description	Create a new error by decoding variables in the HTTP Request
IN	N/A
OUT	error
Name	error_new_from_code
Description	Create a new error by specifying the error code and error parameters
IN	error_code, array(parameters)
OUT	error
Name	error_to_url_postfix
Description	Encode an error to a string that can be postfixed to the URL
IN	error
OUT	string
Name	panic
Description	Stop the execution of current UI Generator or Servlet with the given error message
IN	error_code, array(parameters)
OUT	N/A

3.7 Core Libraries

3.7.1 Session Management

Introduction The session management module is an extension to the PHP session management library. It overrides the default handlers in the library for starting and ending sessions, storing and removing session variables, and garbage collecting so that these session data are stored into the a database table described in Section 4.14.

Initialization Setup the PHP session management library to use database for session storage.

Finalization N/A

Interface Design

Name	session_set_current_person
Description	Store the person id to a session variable
IN	person_id
OUT	N/A
Name	session_get_current_person
Description	Read the person id in current session
IN	N/A
OUT	person_id

3.7.2 User and User Permission Authorization

Introduction This module is a collection of functions for user validation and user permission authorization.

Initialization N/A

Finalization N/A

Interface Design

Name	person_is_offering_student
Description	Test if a person is a student in an offering
IN	person_id, offering_key
OUT	boolean
Name	person_is_section_instructor
Description	Test if a person is an instructor of a section
IN	person_id, section_key
OUT	boolean
Name	person_is_offering_professor
Description	Test if a person is a professor for the given offering
IN	person_id, offering_key
OUT	boolean
Name	person_is_power_user
Description	Test if a person is a power user
IN	person_id
OUT	boolean
Name	person_is_administrator
Description	Test if a person is an administrator
IN	person_id
OUT	boolean

Name	person_authorize_by_iup
Description	Authorize a person with its iup account and password
IN	iup_account, password(plaintext)
OUT	person_id or FALSE(if failure)
Name	person_authorize_by_cas
Description	Authorize a person with CAS system; executed asynchronously and a callback servlet has to be specified
IN	cas_account, password, url of CAS callback servlet
OUT	N/A

3.7.3 Data Manipulation

Introduction This module collects the data manipulation routines. These routines do not validate if the caller has sufficient permissions to perform the given operations. Most of the functions are convenient wrappers over `database_query_from_template` or `database_query_from_string`. Notice that data entry routines are not encapsulated in the module. Data entry is directly resolved in the Servlets via `database_query_from_template`.

Initialization Establish a connection to the SQL database and select the database schema.

Finalization N/A

Interface Design

Name	database_query_from_template
Description	Construct a query by template substitution and execute the query
IN	template_name, array(variable => value)
OUT	result
Name	database_query_from_string
Description	Query the database with an SQL query string
IN	string
OUT	result
Name	database_start_transaction
Description	Start a database transaction
IN	N/A
OUT	N/A
Name	database_commit_transaction
Description	Commit a database transaction
IN	N/A
OUT	N/A
Name	database_select_students_from_offering
Description	Return an array of students enrolled in the given offering
IN	offering_key
OUT	array(person_id => person)

Name	database_select_students_from_component
Description	Return an array of students enrolled in the given component
IN	component_key
OUT	array(person_id => person)
Name	database_select_students_from_section
Description	Return an array of students enrolled in the given section
IN	section_key
OUT	array(person_id => person)
Name	database_select_assignments_from_component
Description	Return an array of assignments in the given component
IN	component_key
OUT	array(assignment_title => assignment)
Name	database_select_components_from_offering
Description	Return an array of components in the given offering
IN	offering_key
OUT	array(component_type => component)
Name	database_select_sections_from_component
Description	Return an array of sections in the given component
IN	component_key
OUT	array(section_number => section)
Name	database_select_courses
Description	Return an array of all courses
IN	N/A
OUT	array(course_number => course)
Name	database_select_semesters
Description	Return an array of all semesters
IN	N/A
OUT	array(semester)
Name	database_select_offerings_from_course
Description	Return an array of all offerings of an course
IN	course_key
OUT	array(semester => offering)
Name	database_select_offerings_from_professor
Description	Return an array of all offerings taught by an professor
IN	person_id
OUT	array(offering)
Name	database_select_sections_from_instructor
Description	Return an array of all sections taught by an instructor
IN	person_id
OUT	array(section_number => section)
Name	database_select_offerings_from_student
Description	Return an array of all offerings in which a student is enrolled
IN	person_id
OUT	array(offering)

Name	database_select_grades_from_students_assignments
Description	An array of grades associated with the direct product of the students and the assignments
IN	array(person), array(assignment)
OUT	array(person)
Name	database_select_grade_history_from_student_assignment
Description	An array of history of grades associated with a given student for a given assignment
IN	person_id, assignment_key
OUT	array(grade)

4 Data Design

4.1 Introduction

A database schema is designed according to the ER model (Refer to Section 2.5) captured in the requirement analysis stage.

The module for data manipulation is defined in section 3.7.3.

4.2 PERSON Table

Introduction

The PERSON table contains all the details of the users and is used to validate the user during login. The default value for the role field is user. The person_id field is auto incremented by default.

Fields

Field	Datatype	Comment
itaccount	varchar(20)	IT account of the user
iupaccount	varchar(20)	IU Physics account of the user that does not have an IU account (non-IU user)
password	varchar(100)	Password to validate the user
first_name	varchar(20)	First name of the user
last_name	varchar(20)	Last name of the user
middle_name	varchar(20)	Middle name of the user
email	varchar(20)	Email ID of the user
person_id	bigint(20)	ID internally generated by the database engine to uniquely identify each user
role	varchar(20)	Type of user

Table 1: PERSON Table Fields

Indexes

- role in the PERSON table is a foreign key that references the name field of the ROLE_TYPE table.
- itaccount in the PERSON table is a unique key because no two persons can have the same itaccount.
- iupaccount in the PERSON table is a unique key because no two persons can have the same iupaccount.
- person_id in the PERSON table is a primary key that is internally generated by the database engine and it is used to uniquely identify a person in the system.

Keyname	Type	Field
itaccount	UNIQUE	itaccount
iupaccount	UNIQUE	iupaccount
primary	primary	person_id
role	INDEX	role

Table 2: PERSON Table Indexes

4.3 COURSE Table

Introduction

The COURSE table contains information regarding the courses which are being offered by the professor.

Fields

Field	Datatype	Comment
course_name	varchar(50)	Name of the course being offered by the professor
course_comment	longtext	Course description
course_number	varchar(20)	Uniquely identifies each course

Table 3: COURSE Table Fields

Index

- course_number is the primary key of the COURSE table.

Keyname	Type	Field
primary	primary	course_number

Table 4: COURSE Table Indexes

4.4 OFFERING Table

Introduction

The OFFERING table contains information about the offerings, such as the professor offering the course and the semester in which it is being offered.

Fields

Field	Datatype	Comment
offering_semester	varchar(20)	Year and semester in which the course is being offered
course_number	varchar(20)	Uniquely identifies a course
professor_id	bigint(20)	ID internally generated by the database engine to identify each professor uniquely

Table 5: OFFERING Table Fields

Indexes

- course_number is a foreign key that references the course_number field in the COURSE table.
- semester is a foreign key that references the name field in the SEMESTER table.

Keyname	Type	Field
primary	primary	offering_semester, course_number, professor_id
course_number	INDEX	course_number
semester	INDEX	offering_semester

Table 6: OFFERING Table Indexes

4.5 SEMESTER Table**Introduction**

The SEMESTER table contains information regarding the year and the semester in which the courses are being offered.

Fields

Field	Datatype	Comment
name	varchar(20)	Year and semester in which the course is being offered

Table 7: SEMESTER Table Fields

Indexes

Keyname	Type	Field
primary	primary	name

Table 8: SEMESTER Table Indexes

4.6 COMPONENT Table

Introduction

The COMPONENT table contains information regarding the component such as component name, the corresponding course number, who is offering that course and when is it being offered.

Fields

Field	Datatype	Comment
course_number	varchar(20)	Uniquely identifies a course
offering_semester	varchar(20)	Year and semester in which the course is being offered
professor_id	bigint(20)	ID internally generated by the database engine to identify each professor uniquely
component_type	varchar(20)	Name of the component in a course

Table 9: COMPONENT Table Fields

Indexes

- course_number, offering_semester, professor_id in the COMPONENT table together form a foreign key that references course_number, offering_semester, professor_id fields in the OFFERING table and which together acts as a primary key for the COMPONENT table.
- component_type in COMPONENT table is a foreign key that references the name field in the COMPONENT_TYPE table.

Keyname	Type	Field
primary	primary	course_number, offering_semester, professor_id
course_number	INDEX	course_number, offering_semester, professor_id
component_type	INDEX	component_type

Table 10: COMPONENT Table Indexes

4.7 COMPONENT_TYPE Table

Introduction

The COMPONENT_TYPE table contains information regarding the type of component.

Fields

Field	Datatype	Comment
name	varchar(45)	Type of component

Table 11: COMPONENT_TYPE Table Fields

Indexes

Keyname	Type	Field
primary	primary	name

Table 12: COMPONENT_TYPE Table Indexes

4.8 COURSE_DEFAULT_COMPONENTS Table

Introduction

The COURSE_DEFAULT_COMPONENTS table contains information regarding default component setup of a course.

Fields

Field	Datatype	Comment
course_number	varchar(20)	Uniquely identify a course
component_type	varchar(20)	Type of component

Table 13: COURSE_DEFAULT_COMPONENTS Table Fields

Indexes

- course_number in COURSE_DEFAULT_COMPONENTS table is a foreign key that references the course_number field in the COURSE table.
- component_type in COURSE_DEFAULT_COMPONENTS table is a foreign key that references the name field in the COMPONENT_TYPE table.
- course_number, component_type together act as a primary key for the COURSE_DEFAULT_COMPONENTS table.

Keyname	Type	Field
primary	primary	course_number, component_type
course_number	INDEX	course_number
component_type	INDEX	component_type

Table 14: COURSE_DEFAULT_COMPONENTS Table Indexes

4.9 ENROLL_COMPONENT Table

Introduction

The ENROLL_COMPONENT table contains information regarding the students enrolled in a component along with the component information such as the course to which the component belongs, in which semester the course is being offered and which professor is offering that course. This table is also used to check the existence of the student in the component and also holds information regarding the student's final grade for the component.

Fields

Field	Datatype	Comment
course_number	varchar(20)	Uniquely identifies a course
offering_semester	varchar(20)	Year and semester in which the course being offered
professor_id	bigint(20)	ID internally generated by the database engine to identify each professor uniquely
component_type	varchar(20)	Type of component
student_id	bigint(20)	ID internally generated by the database engine to identify each student uniquely
component_grade	varchar(10)	Final component grade assigned to each student
active	tinyint(1)	Hides/unhides a student from the component

Table 15: ENROLL_COMPONENT Table Fields

Indexes

- student_id field in the ENROLL_COMPONENT is a foreign key that references the person_id field in the PERSON table.
- course_number, offering_semester, professor_id, component_type fields together is a foreign key that references course_number, offering_semester, professor_id, component_type fields in the COMPONENT table.

- course_number, offering_semester, professor_id, component_type, student_id fields together acts as a primary key for the ENROLL_COMPONENT table.

Keyname	Type	Field
component	INDEX	course_number, offering_semester, professor_id, component_type
student	INDEX	student_id
primary	primary	course_number, offering_semester, professor_id, component_type

Table 16: ENROLL_COMPONENT Table Indexes

4.10 ENROLL_OFFERING Table

Introduction

The ENROLL_OFFERING table contains information regarding the students enrolled in the course along with the course information such as in which semester the course is being offered and which professor is offering that course. This table is also used to check the existence of the student in the course and also holds information regarding the student's final course grade.

Fields

Field	Datatype	Comment
offering_semester	varchar(20)	Year and semester in which the course being offered
course_number	varchar(20)	Uniquely identifies a course
professor_id	bigint(20)	ID internally generated by the database engine to identify each professor uniquely
student_id	bigint(20)	ID internally generated by the database engine to identify each student uniquely
offering_final_grade	varchar(10)	Final component grade assigned to each student
active	tinyint(1)	Hides/unhides a student from the course offering

Table 17: ENROLL_OFFERING Table Fields

Indexes

- offering_semester, course_number, professor_id, student_id fields together acts as a primary key for the ENROLL_OFFERING table.

- offering_semester, course_number, professor_id fields together acts as a foreign key for the ENROLL_OFFERING table that references the offering_semester, course_number, professor_id fields of the OFFERING table.
- student_id field in the ENROLL_OFFERING table is a foreign key that references the person_id field of the PERSON table.

Keyname	Type	Field
primary	primary	offering_semester, course_number, professor_id, student_id
offering	INDEX	offering_semester, course_number, professor_id
student	INDEX	student_id

Table 18: ENROLL_OFFERING Table Indexes

4.11 JOIN_SECTION Table

Introduction

The JOIN_SECTION table contains information regarding the student such as when the student joined the section, the component to which the section belongs, the course to which the component belongs, which professor is offering the course and in which semester the course is being offered. It also holds information regarding the student withdrawing from the section. The default value for the start_timestamp field is CURRENT_TIMESTAMP.

Fields

Field	Datatype	Comment
start_timestamp	timestamp	Records the date and time that the student is enrolled into the section
end_timestamp	timestamp	Records the date and time that student leaves the section. NULL if student is currently part of the section
course_number	varchar(20)	Uniquely identifies a course
offering_semester	varchar(20)	Year and Semester in which the course is being offered
professor_id	bigint(20)	ID internally generated by the database engine to identify each professor uniquely
component_type	varchar(20)	Type of component
section_number	varchar(20)	Section number of the section in a component
student_id	bigint(20)	ID internally generated by the database engine to identify each student uniquely

Table 19: JOIN_SECTION Table Fields

Indexes

- start_timestamp, course_number, offering_semester, professor_id, component_type, section_number, student_id fields together act as a primary key for the JOIN_SECTION table.
- start_timestamp, course_number, offering_semester, professor_id, component_type, section_number fields together acts as a foreign key for the JOIN_SECTION table that references the start_timestamp, course_number, offering_semester, professor_id, component_type, section_number fields of the SECTION table.
- student_id field is a foreign key of the JOIN_SECTION table that references the person_id field of the PERSON table.

Keyname	Type	Field
primary	primary	start_timestamp, course_number,offering_semester, professor_id, component_type, section_number, student_id
section	INDEX	start_timestamp, course_number, offering_semester, professor_id, component_type, section_number
student	INDEX	student_id

Table 20: JOIN_SECTION Table Indexes

4.12 SECTION Table

Introduction

The SECTION table contains information about a section such as section number, the component to which the section belongs, the course to which the component belongs, which professor is offering the course and in which semester the course is being offered.

Fields

Field	Datatype	Comment
section_number	varchar(20)	Section number of the section in a component
component_type	varchar(20)	Type of component
course_number	varchar(20)	Uniquely identifies a course
offering_semester	varchar(20)	Year and semester in which the course being offered
professor_id	bigint(20)	ID internally generated by the database engine to identify each professor uniquely

Table 21: SECTION Table Fields

Indexes

- section_number, component_type, course_number, offering_semester, professor_id fields together act a primary key for the SECTION table.
- component_type, course_number, offering_semester, professor_id fields together acts as a foreign key for the SECTION table that references component_type, course_number, offering_semester, professor_id fields of the COMPONENT table.
- section_number is also made unique because a course can have a component without sections in it.

Keyname	Type	Field
primary	primary	section_number, component_type, course_number, offering_semester, professor_id
component	INDEX	component_type, course_number, offering_semester, professor_id
section_number	UNIQUE	section_number

Table 22: SECTION Table Indexes

4.13 TEACH_SECTION Table

Introduction

The TEACH_SECTION table contains information regarding the instructor such as the section to which the instructor is assigned, the component to which the section belongs, the course to which the component belongs, which professor is offering the course and in which semester it is being offered.

Fields

Field	Datatype	Comment
course_number	varchar(20)	Uniquely identifies a course
offering_semester	varchar(20)	Year and semester in which the course being offered
professor_id	bigint(20)	ID internally generated by the database engine to identify each professor uniquely
component_type	varchar(20)	Type of component
section_number	varchar(20)	Section number of the section in a component
instructor_id	bigint(20)	ID internally generated by the database engine to identify each instructor uniquely

Table 23: TEACH_SECTION Table Fields

Indexes

- course_number, offering_semester, professor_id, component_type, section_number, instructor_id fields together acts as a primary key for the table TEACH_SECTION.
- course_number, offering_semester, professor_id, component_type, section_number fields together is a foreign key for the table TEACH_SECTION which references the course_number, offering_semester, professor_id, component_type, section_number fields of the SECTION table.
- instructor_id field is also a foreign key for the table TEACH_SECTION which references the person_id field of the PERSON table.

Keyname	Type	Field
primary	primary	course_number, offering_semester, professor_id, component_type, section_number, instructor_id
section_number	INDEX	course_number, offering_semester, professor_id, component_type, section_number,
instructor_id	INDEX	instructor_id

Table 24: TEACH_SECTION Table Indexes

4.14 SESSIONS Table

Introduction

The SESSION table stores information regarding each session.

Fields

Field	Datatype	Comment
session	varchar(255)	name of the session
session_expires	int(10)	the time when the session expires
session_data	text	data what session would contain

Table 25: SESSIONS Table Fields

Indexes

- session field in the SESSION table is a primary key

Keyname	Type	Field
primary	primary	session

Table 26: SESSIONS Table Indexes

4.15 ASSIGN_GRADES Table

Introduction

The ASSIGN_GRADES table contains information regarding grades assigned to students for different assignments in a section by instructors belonging to that section. The default value of the grade_timestamp field is CURRENT_TIMESTAMP.

Fields

Field	Datatype	Comment
teacher_id	bigint(20)	ID internally generated by the database engine
student_id	bigint(20)	ID internally generated by the database engine to uniquely identify each student
course_number	varchar(20)	Uniquely identifies a course
offering_semester	varchar(20)	Year and semester in which the course being offered
professor_id	bigint(20)	ID internally generated by the database engine to uniquely identify each professor
component_type	varchar(20)	Type of component
assignment_title	varchar(100)	Title of the assignment
grade_timestamp	timestamp	Records timestamp of the submission of the grade
grade	varchar(10)	Grade assigned to a given student for an assignment
grade_comments	text	Comments related to the grade for each student

Table 27: ASSIGN_GRADES Table Fields

Indexes

- teacher_id, student_id, course_number, offering_semester, professor_id, component_type, assignment_title, grade_stamp fields together act as a primary key for the ASSIGN_GRADES table.
- course_number, offering_semester, professor_id, component_type, assignment_title fields together act as a foreign key of the ASSIGN_GRADES table that references the offering_semester, professor_id, component_type, assignment_title fields of the ASSIGNMENT table.
- teacher_id is a foreign key for the ASSIGN_GRADES table that references the person_id field in the PERSON table.
- student_id is a foreign key for the table ASSIGN_GRADES that references the person_id field in the PERSON table.

Keyname	Type	Field
primary	primary	teacher_id, student_id, course_number, offering_semester, professor_id, component_type, assignment_title, grade_timestamp
teacher_id	INDEX	teacher_id
student_id	INDEX	student_id
assignment	INDEX	course_number, offering_semester, professor_id, component_type, assignment_title

Table 28: ASSIGN_GRADES Table Indexes

4.16 LATEST_ASSIGNED_GRADES Table

Introduction

The LATEST_ASSIGN_GRADES table contains information regarding the latest grades assigned to students for different assignments in different sections for which different instructors are responsible.

Fields

Field	Datatype	Comment
teacher_id	bigint(20)	ID internally generated by the database engine
student_id	bigint(20)	ID internally generated by the database engine
course_number	varchar(20)	Uniquely identifies a course
offering_semester	varchar(20)	Year and semester in which the course is being offered
professor_id	bigint(20)	ID internally generated by the database engine
component_type	varchar(20)	Type of component
assignment_title	varchar(100)	Name of the assignment
grade	varchar(10)	Latest grade assigned to a student for an assignment

Table 29: LATEST_ASSIGNED_GRADES Table Fields

Indexes

- teacher_id, student_id, course_number, offering_semester, professor_id, component_type, assignment_title fields together act as a primary key for the LATEST_ASSIGN_GRADES table.
- course_number, offering_semester, professor_id, component_type, assignment_title fields together act as a foreign key in the LATEST_ASSIGN_GRADES table that references offering_semester, professor_id, component_type, assignment_title fields of the ASSIGNMENT table.

- teacher_id is a foreign key for the LATEST_ASSIGN_GRADES table that references the person_id field of the PERSON table.
- student_id is a foreign key for the LATEST_ASSIGN_GRADES table that references the person_id field of the PERSON table.

Keyname	Type	Field
primary	primary	teacher_id, student_id, course_number, offering_semester, professor_id, component_type, assignment_title
teacher_id	INDEX	teacher_id
student_id	INDEX	student_id

Table 30: LATEST_ASSIGNED_GRADES Table Indexes

4.17 **ROLE_TYPE Table**

Introduction

The ROLE_TYPE table contains information regarding the roles the user can have. The user of the system can only take the roles which are specified in this table.

Fields

Field	Datatype	Comment
name	varchar(20)	Type of user

Table 31: ROLE_TYPE Table Fields

Indexes

- name field is a primary key for the ROLE_TYPE table

Keyname	Type	Field
primary	primary	name

Table 32: ROLE_TYPE Table Indexes

4.18 **ASSIGNMENT_TYPE Table**

Introduction

The ASSIGNMENT_TYPE table contains information regarding the type of assignment such as clicker questions and home work.

Fields

Field	Datatype	Comment
name	varchar(20)	Type of assignment

Table 33: ASSIGNMENT_TYPE Table Fields

Indexes

- name field is a primary key for the table ASSIGNMENT_TYPE table.

Keyname	Type	Field
primary	primary	name

Table 34: ASSIGNMENT_TYPE Table Indexes

4.19 ASSIGNMENT Table**Introduction**

The ASSIGNMENT table contains information regarding the assignment such as name and type of assignment, the component to which the assignment belongs, the course to which the component belongs, which professor is offering the course and in which semester is it being offered.

Fields

Field	Datatype	Comment
course_number	varchar(20)	Uniquely identifies a course
offering_semester	varchar(20)	Year and semester in which the course being offered
professor_id	bigint(20)	ID internally generated by the database engine to uniquely identify each professor
component_type	varchar(20)	Type of component
assignment_title	varchar(100)	Name of the assignment
assignment_type	varchar(20)	Type of assignment

Table 35: ASSIGNMENT Table Fields

Indexes

- course_number, offering_semester, professor_id, component_type, assignment_title fields together act as a primary key for the ASSIGNMENT table.
- course_number, offering_semester, professor_id, component_type fields together act a foreign key of the ASSIGNMENT table that references the course_number, offering_semester, professor_id, component_type fields of the COURSE table.

- assignment_type field is a foreign key of the ASSIGNMENT table that references the name field in the ASSIGNMENT_TYPE table.

Keyname	Type	Field
primary	primary	course_number, offering_semester, professor_id, component_type, assignment_title
offering	INDEX	course_number, offering_semester, professor_id, component_type
assignment_type	INDEX	assignment_type

Table 36: ASSIGNMENT Table Indexes

5 User Interface

The user interface of the proposed system is described in terms of what the user must do to use the system.

Figure 4 on page 38, Figure 5 on page 39 and Figure 6 on page 39 show the menu hierarchy for different users in the system. Clearly, we pick an entity first and then an operation for that entity.

The User Interfaces used in the system are of two types:

1. Regular page which is in the form of a regular browser window.
2. Dialog box which is in the form of a pop-up window. When an operation is performed in a dialog box, the user returns to the regular browser window.

The following is the standard layout for all screens:

- The banner of the website is shown at the top of all pages. However, we do not explicitly show this banner in this document.
- The page heading is always at the top left corner of the screen.
- The name of the user, links for Help and Logout are always on the top right corner of the screen, except in the Welcome and Login pages where only the link for Help is shown.
- Any error message is displayed just below the title for the page.
- All Submit Buttons are displayed below the text fields.

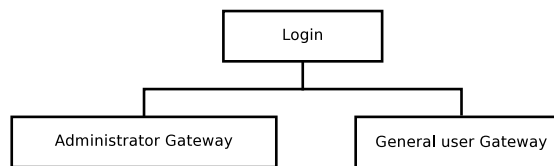


Figure 4: Menu Hierarchy at Login

As shown in this figure, the user is directed to either of the two gateways after login.

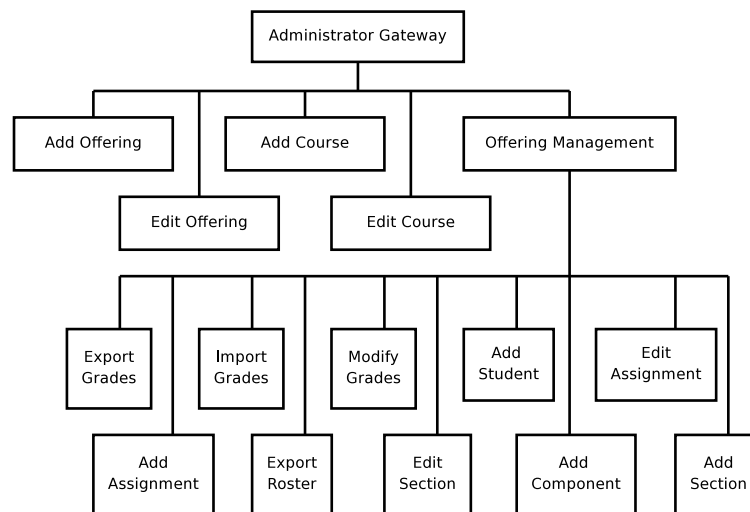


Figure 5: Menu Hierarchy for Administrator

As shown in this figure, the forms for Administrator are organized first by entity and then by operation. Here, the entities are Offering, Component and Section. The operations are View, Add, Export, Import, and Modify.

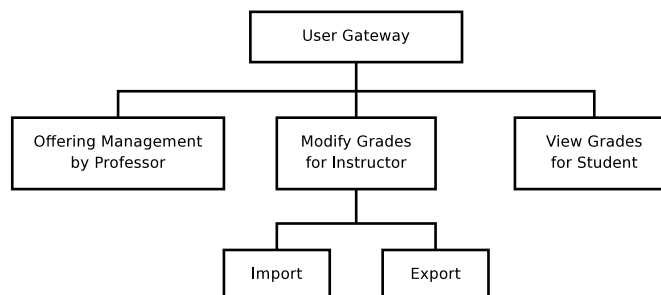


Figure 6: Menu Hierarchy for General Users

As shown in this figure, the forms for General Users are organized first by entity and then by operation. Here, the entities are Offering, Component and Section. The operations are View, Export, Import, and Modify.

The following user interfaces are common to all users of the system:

5.0.1 Welcome

This screen is merely a page that asks the user to go the appropriate page for login. The user may go to either the CAS login or the non-CAS login page. This page is generated by the Trivial Generator module (Section 3.4.1).

Welcome to IU Physics Gradebook System

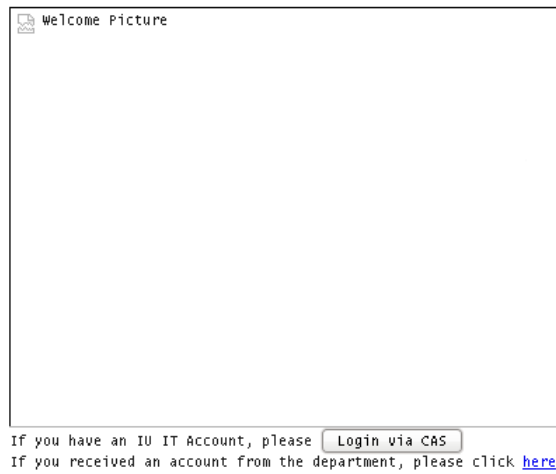


Figure 7: Welcome screen

Operation

A user with a valid IU account clicks on the “Login via CAS” button. This takes the user to the CAS login screen. If not, the user chooses to go to the non-CAS login page.

Navigation

Hyperlink	Destination
here	Non-CAS login page

Table 37: Navigation from Welcome screen

Appearance

A Welcome picture is shown on this page immediately below the page heading.

5.0.2 Non-CAS Login

This screen enables a non-CAS user to log in to the system. This page is generated by the Trivial Generator (Section 3.4.1) module.

Welcome To the IU Physics [Help](#) Gradebook System

ERROR MESSAGE

If you have an account and password provided by the department, please submit it here.

Username

Password

Figure 8: Non-CAS Login screen

Operation

The user types the username and password and then clicks on the IUP login button. This invokes the Login Servlet (Section3.5.1).

A correct username and password takes the user to either 5.1.1 or 5.2.1. An incorrect username and/or password entered by the user means that the user is shown the login screen again with a message indicating this invalid entry.

5.0.3 CAS Login

This page allows CAS users to log in to the system.

INDIANA UNIVERSITY

Central Authentication Service
Please enter your username and passphrase.

Username:

Passphrase:

- For security reasons, you will need to close your web browser when you finish using services that require authentication.
- If you need assistance, view [login help](#).

INDIANA UNIVERSITY Copyright © 2008 The Trustees of Indiana University | Copyright Complaints

Figure 9: CAS Login screen

Operation

The user types the IU username and password and then clicks on the “login” button. The CAS Callback Servlet (Section3.5.2) is invoked at this time.

5.1 Administrator

5.1.1 Administrator Gateway

This page is generated by the Administrator Gateway Generator (Section3.4.3) module.

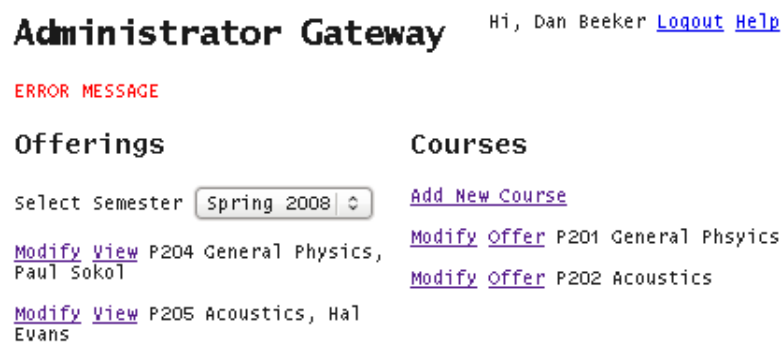


Figure 10: Administrator Gateway screen

Operations

- Add Offering
- View Offering
- Modify Offering
- Offer Course
- Modify Course
- Select Semester
- Add New Course

Navigation

Hyperlink	Destination
Modify under Offerings	Modify Offering screen
View	Offering Management screen
Modify under Courses	Modify Course screen
Offer	Add Offering screen
Add New Course	Add Course screen

Table 38: Navigation from Administrator Gateway

5.1.2 Add Offering

This page is generated by the Trivial Generator (Section3.4.1) module.

Offer a Course

Hi, Dan Beeker [Logout](#) [Help](#)

ERROR MESSAGE
The course P204 General Physics is going to be offered.

Please select a semester or [Add](#) a new semester
Spring 2008

Which professor is offering the course?
If the professor has an IU IT account:
rc5

or alternatively
If the professor uses an IUP account ([Add User](#))

Submit

Figure 11: Add Offering screen

Operation

The Administrator selects the semester in which the course needs to be offered. The Administrator also assigns a professor to the Offering and then clicks the Submit button to make the course offering and returns to the Administrator Gateway screen.

Navigation

Hyperlink	Destination
Add	Add Semester screen
Add User	Add Non-CAS User screen

Table 39: Navigation from Add Offering screen

5.1.3 Offering Management

This page is generated by the Manage Offering Generator module (Section3.4.6).

Manage Offering

Hi, Dan Beeker [Logout](#) [Help](#)

ERROR MESSAGE

P204 General Physics, Paul Sokol

[Modify Grades](#) [Export Grades](#) [Import Grades](#) [Export Roster](#)

Add or Select Component

Sections

[Add New Section](#)

[Modify View Grades](#) Section 12345, Yu Feng, 23 Students

[Modify View Grades](#) Section 12346, Peng Guo, 22 Students

Assignments

[Add New Assignment](#)

[Modify View Grades](#) Clicker: Clicker 38.1

[Modify View Grades](#) Clicker: Clicker 38.2

[Modify View Grades](#) Homework: 3.8, Problem 3

[Modify View Grades](#) Homework: 3.9, Problem 4

Roster

[Add New Student](#)

Move Selected Students to Section in Component

	Last Name	First Name	Email	Lab Section	Lecture Section	Recitation Section
<input type="checkbox"/>	Feng	Yu	feng@g.com		12345	12346
<input type="checkbox"/>	Tank	Chintan	ct@g.com	12222	12345	12346
<input type="checkbox"/>	Liu	Baoyi	lby@g.com		12345	12346

Figure 12: Offering Management screen

Operations

- Modify Grades
- Export Grades
- Import Grades
- Export Roster
- Add Component
- Add New Section
- Modify Section
- View Grades for Section
- Add New Assignment
- Modify Assignment
- View Grades for Assignment
- Add New Student to Roster
- Move Students between Sections

Navigation

Hyperlink	Destination
Modify Grades	Modify Grades screen
Export Grades	Export Grades screen
Import Grades	Import Grades screen
Export Roster	Export Roster screen
Add	Add Offering Component screen
Add New Section	Add Section screen
Modify under Sections	Modify Section screen
View Grades under Sections	View Grades for Section screen
Modify under Assignments	Modify Assignment screen
View Grades under Assignments	View Grades for Assignment screen
Add New Student	Add New Student screen

Table 40: Offering Screen for Administrator

Add a Component to P202 General Physics, Paul Sokol

Hi, Dan Beeker [Logout](#) [Help](#)

ERROR MESSAGE

You are about to add a new component to an offering. This will make the offering different from other offerings of P202 General Physics.

Please select the type of the component.

If the component is not listed, please type it here

Figure 13: Add Offering Component screen

5.1.4 Add Component to Course Offering

This page is generated by the Trivial Generator module (Section3.4.1).

Operation

The administrator enters the component type in the text box provided. On clicking the “Submit” button, the Offering Management Servlet (Section3.5.8) is invoked.

5.1.5 Add Section to Component

This page is generated by the Trivial Generator module (Section3.4.1).

Add Section to Lab Component of 2008 Fall P202 General Physics, Paul Sokol

Hi, Dan Beeker [Logout](#) [Help](#)

ERROR MESSAGE

Please type the section number obtained from the registrar:

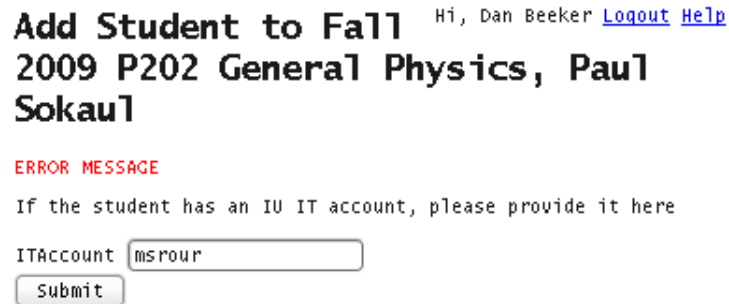
Figure 14: Add Section To Component screen

Operation

The administrator enters the section number in the text box provided. On clicking the “Submit” button, the Offering Management Servlet (Section 3.5.8) is invoked.

5.1.6 Add Student to a Course Offering

This page is generated by the Trivial Generator module (Section 3.4.1).



Add Student to Fall 2009 P202 General Physics, Paul Sokaul Hi, Dan Beeker [Logout](#) [Help](#)

ERROR MESSAGE

If the student has an IU IT account, please provide it here

ITAccount

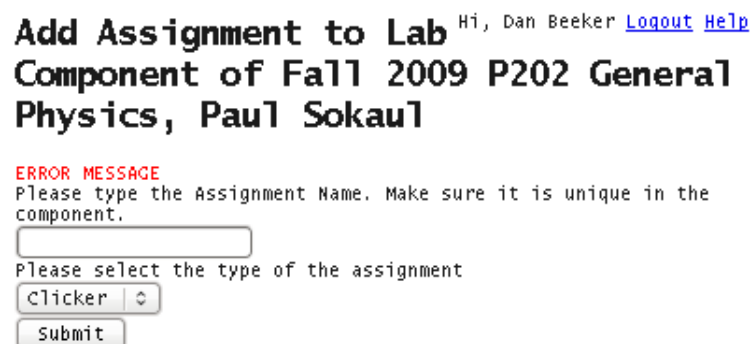
Figure 15: Add Student screen

Operation:

The administrator enters the student’s IU account in the text box provided. On clicking the “Submit” button, the Offering Management Servlet (Section 3.5.8) is invoked.

5.1.7 Add Assignment to Component

This page is generated by the Trivial Generator module (Section 3.4.1).



Add Assignment to Lab Component of Fall 2009 P202 General Physics, Paul Sokaul Hi, Dan Beeker [Logout](#) [Help](#)

ERROR MESSAGE

Please type the Assignment Name. Make sure it is unique in the component.

Please select the type of the assignment

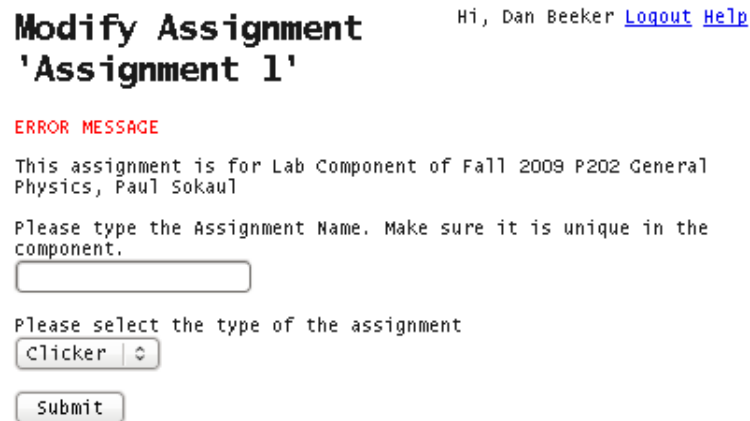
Figure 16: Add Assignment screen

Operation

The administrator enters the assignment title in the text box provided and also selects the type of assignment. On clicking the “Submit” button, the Offering Management Servlet (Section 3.5.8) is invoked.

5.1.8 Modify Assignment in Component

This page is generated by the Trivial Generator module (Section 3.4.1).



Modify Assignment Hi, Dan Beeker [Logout](#) [Help](#)

'Assignment 1'

ERROR MESSAGE

This assignment is for Lab Component of Fall 2009 P202 General Physics, Paul Sokaul

Please type the Assignment Name. Make sure it is unique in the component.

Please select the type of the assignment

Clicker

Submit

Figure 17: Modify Assignment screen

Operation

The administrator enters the new assignment title in the text box provided and chooses the type of assignment. On clicking the “Submit” button, the Offering Management Servlet (Section 3.5.8) is invoked.

5.1.9 Modify Course

This page is generated by the Trivial Generator module (Section 3.4.1).

Modify P202 Acoustics Hi, Dan Beeker [Logout](#) [Help](#)

ERROR MESSAGE

You are about to modify the course P202 General Physics. The changes on the course number and the name of the course will immediately show up on the offerings of the course. The changes on the default components will only affect future offerings of the course.

What is the number of the Course? (e.g., P202)

What is the name of the Course? (e.g., Acoustics)

Please list the default components included by the course. An usual list will be "Lab, Lecture, Discussion"

[More](#)

Figure 18: Modify Course screen

Operation

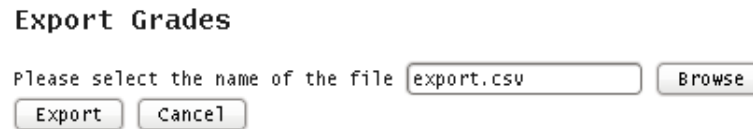
The Administrator enters the new course number and title in the appropriate text box and also provides the new default components of the course in the text boxes for Default Components. On clicking the “Submit” button, the Course Management Servlet (Section 3.5.7) is invoked.

Navigation

Hyperlink	Destination
More	Modify Course screen with additional blank text boxes to add default components

Table 41: Navigation from Modify Course screen

5.1.10 Export Grades



Export Grades

Please select the name of the file

Figure 19: Export Grades screen

Operation

The Administrator clicks the “Browse” button to choose the file that needs to be exported. Clicking on the Export button invokes the Export Servlet (Section 3.5.5).

5.1.11 Export Offering Roster



ExportRoster

Please select the name of the file

Figure 20: Export Roster screen

Operation

The Administrator clicks the “Browse” button to choose the file that needs to be exported. Clicking on the Export button invokes the Export Servlet (Section 3.5.5).

5.1.12 Export Component Grades

Refer to 5.1.10.

5.1.13 Export Component Roster

Refer to 5.1.11.

5.1.14 Import Offering Grades

Import Grades

Please select the name of the file

Figure 21: Import Offering Grades screen

Operation

The Administrator clicks the “Browse” button to choose the file that needs to be imported. Clicking on the “Import” button invokes the Trivial Generator module on page 13.

5.1.15 Import Component Grades

Refer to 5.1.14.

5.1.16 Modify Grades

This page is generated by the Modify Grades Generator (Section 3.4.4). This is the same as the View Grades screen but in modify mode. To switch to this modify mode, the user must click on the “here” hyperlink in the View Grades screen.

Modify Grades for O

Hi, Dan Beeker [Logout](#) [Help](#)

ERROR MESSAGE
 This page is read-only to protect the grades from unwillingly modifications. To unlock, click [here](#).
[Import](#) [Export](#)

Find Student Find Assignment

		Lecture: Homework		Lecture: Clicker		Lab	
Last Name	First Name	Assignment1	Assignment2	Clicker1	Clicker2	Lab1	Lab2
Liu	Baoyi	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Tank	Chintan	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Feng	Yu	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Figure 22: Modify Grades screen

Operation

The user assigns/modifies grades for any student for any assignment by editing the appropriate text boxes. The user can use the Student or Assignment filter by typing something to search for a specific subset of

students.

Navigation

Hyperlink	Destination
here	View Grades screen (same as Modify Offering Grades screen but in read-only mode)
Import	Import Offering Grades screen
Export	Export Offering Grades screen

Table 42: Navigation from Modify Offering Grades screen

Appearance and Layout

- During filtering, the results are highlighted in yellow color.
- A valid grade entry in a text box is shown in green color.
- An invalid grade in a text box is highlighted in red color.

5.1.17 View Offering Grades

This page is generated by the Modify Grades Generator. This is the same as the Modify Grades screen but in read-only mode.

Modify Grades for Hi, Dan Beeker [Logout](#) [Help](#)

ERROR MESSAGE
This page is read-only to protect the grades from unwillingly modifications. To unlock, click [here](#).
[Import](#) [Export](#)

Find Student Find Assignment

Last Name	First Name	Lecture: Homework		Lecture: Clicker		Lab	
		Assignment1	Assignment2	Clicker1	Clicker2	Lab1	Lab2
Liu	Baoyi	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Tank	Chintan	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Feng	Yu	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Figure 23: View Offering Grades screen

Operation

The user can use the Student or Assignment filter to search for a specific subset of students.

Navigation

Hyperlink	Destination
here	Modify Grades screen (same as View Offering Grades screen but in modify mode)
Import	Import Offering Grades screen
Export	Export Offering Grades screen

Table 43: Navigation from View Offering Grades screen

Appearance and Layout

- During filtering, the results are highlighted in yellow color.

5.1.18 View Component Grades

Refer to 5.1.17.

5.1.19 View Section Grades

Refer to 5.1.17.

5.2 General Users

5.2.1 User Gateway



Figure 24: User Gateway screen

Operation

The user views the list of course offerings associated with that user for that semester.

- If the user is a Professor, then only the Professor Gateway is shown to the user.
- If the user is a Student, then only the Student Gateway is shown to the user.
- If the user is an Instructor, then only the Instructor Gateway is shown to the user.
- A user may be both a Student and an Instructor, in which case both the Student and Instructor Gateways are shown to the user.

Navigation

Hyperlink	Destination
links to Offering (Professor)	Offering Management screen
links to Offering (Student)	Report Grades screen
links to Section (Instructor)	Modify Grades screen

Table 44: Navigation from Gateway screen

5.2.2 Report Grades Screen

This page is generated by the Report Grades Generator (Section 3.4.7).

Grade Report		Hi, Paul Sokaul Logout Help
ERROR MESSAGE		
Grade Report for Justin Stevens		
Fall 2009 P202 General Physics		
Final Grade: (Pending)		
Lab	Lecture	Discussion
Final Grade: 9.0	Final Grade: 25.0	Final Grade: 25.0
Lab1: 9.0	Cpcker1: 25	Assignment 1: 25
Lab2: 8.0	Cpcker2: 25	Assignment 2: 25
Lab3: 10.0	Assignment 1: 25	Assignment 3: 25
		Assignment 4: 25
		Assignment 5: 25

Figure 25: Report Grades screen

Operation

The user can print the page from the browser menu.

6 Exception Handling

This section covers the policy and schema for handling exceptions.

6.1 Platform Failures

A platform failure occurs either because of wrong configuration of the system or malfunction of the platform. The platform failures are resolved at the platform levels. A typical list of possible platform failures and how they will be handled is provided in the following. Usually, a platform failure should stop the system from working, and until the cause of the failure is solved in the platform, the system cannot come back online.

Power Failure Power Failure is handled at the operating system and database system level.

Database Failure Temporary database failures are handled at the database system level. MySQL is a mature database management solution and the client can rely on its failure recovery mechanism. In rare cases, the IT staff in the client organization has to manually fix the database.

A special case is that the exceptions on the foreign key constraints are not database failures. Instead, they are handled in the Core Libraries and translated into runtime exceptions.

File System Failure File system failures are handled at the operating system level. A corrupt file system does not always put the system offline immediately, but if the corrupt blocks span the files required to support the software system, it will fail. The recovery usually also requires the intervention of the IT staff in the client organization.

Web Server Failure Web server failures are handled at the web server level. A web server failure is also not self-recoverable.

- Internal Server Error
The request was not completed; the server met an unexpected condition
- Not Implemented
The request was not completed; the server did not support the functionality required
- Bad Gateway
The request was not completed; the server received an invalid response from the upstream server
- Service Unavailable
The request was not completed; the server is temporarily overloaded or down
- Gateway Timeout
The gateway has timed out.
- HTTP Version Not Supported
The server does not support the "HTTP protocol" version

6.2 Runtime Exceptions

Runtime exceptions occur during the online period of the system when the users are interacting with the system in sessions. Usually the runtime exceptions are generated (thrown) in the Servlets, and handled in the User Interface Generators. Runtime exceptions do not put the system offline. The Error Handling module provides the mechanisms for runtime exception handling.

- ACCESS_DENIED
The user is not allowed to access the requested information
- INVALID_USERNAME
The provided user name does not exist
- INVALID_PASSWORD
The provided password does not match the user name
- INVALID_INPUT
The input is invalid; the definition of invalidity depends on the context
- RESOURCE_NOT_FOUND
The template or the requested database entry is not found

6.3 Error Handling Schema

The platform to support exception handling is described in the Error Handling Module in Section 3.6. Runtime Exceptions are thrown by User Interface Generator and servlets. They are then caught by User Interface Generator. Refer to Figure 26 for a graphical explanation.

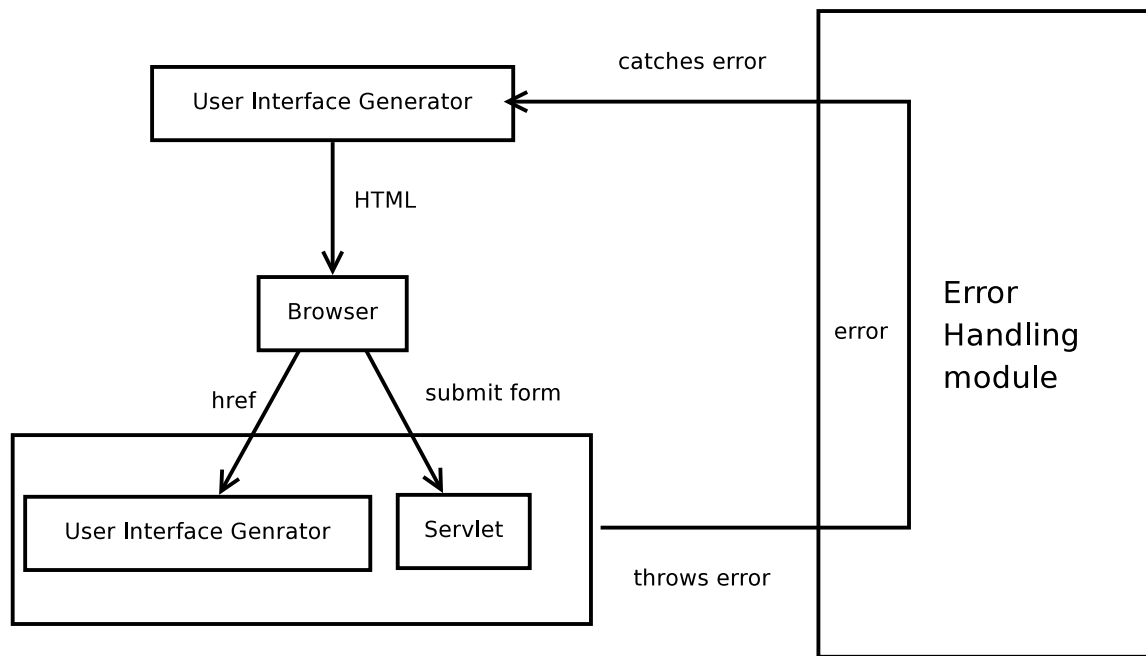


Figure 26: Error Handling Schema

The figure demonstrates the production and consumption of exceptions. Exceptions are thrown by User Interface Generator and servlets. They are then caught by User Interface Generator.

7 Test Plan

Developed from the requirement specification document[2], the testing scope outlined below describes the types of testing that will be performed on the proposed information system. For each testing type, a policy for performing the test is given.

7.1 Unit Testing

The objective of unit testing is to verify that the individual units of source code are working properly. In this project, a unit is equivalent to a function in a module. Considered as a white box testing technique, unit testing is generally done before integration testing.

Policy This testing will be carried out before the code enters the library. PHP-Unit[7] will be used to perform the unit testing.

PHP-Unit is a member of the xUnit family of testing frameworks and provides both a framework that makes the writing of tests easy as well as the functionality to easily run the tests and analyze their results.

7.2 Security Testing

The objective of security testing is to test factors such as confidentiality, integrity, authentication, authorization, availability and non-repudiation of the system. The client specifically mentioned about the safety of the system from external hackers; to ensure this safety, the team will perform regular security testings with standard tools.

Policy This testing will be carried out after the implementation phase. Nikto[10] is used to perform the security testing.

Nikto is a tool for finding default web files and examining web server and CGI security.

7.3 Functionality Testing

The objective of the functionality testing is to verify whether the system meets the functionality requirements specified in the requirement specification document[2].

Policy Functionality testing is performed after the entire system is implemented. Open source tool Selenium[8] is used to carry out the functionality testing.

Selenium is a suite of tools to automate web app testing across many platforms.

7.4 Integration Testing

This test proves that all components of the system interface with each other correctly without gaps in the data flows. The final integration testing will prove that system works as an integrated unit when all the bugs are fixed.

Policy Integration testing is performed after the libraries are frozen, when no more unit testing is required.

7.5 User Acceptance Testing

User Acceptance testing ensures that the system operates in the expected manner, and any supporting materials such as procedures, forms are accurate and suitable for the intended purpose. This testing also ensures that there are no gaps between the functionality of the entire system and the user's conceptions.

Policy User acceptance testing is performed after the integration testing.

7.6 Performance Testing

These tests ensure that the response time of the system is acceptable. The proposed system will have multiple users using the system simultaneously. The system should respond within acceptable time to all the users under an average load.

Policy Performance testing is performed after the integration testing. Open source tool OpenWebLoad[9] is used to carry out the Performance testing.

OpenWebLoad is a tool for load testing web applications. It aims to be easy to use and providing near real-time performance measurements of the application under test.

7.7 Regression Testing

The objective of regression testing is to ensure that new functionality and improved stability of the software does not compromise the existing functionalities.

Policy Regression testing is not carried out because there is only one release cycle.

7.8 Multi-User Testing

Multi-user testing will attempt to prove that it is possible for an acceptable number of users to work with the system at the same time. The objective of the test is to break the system.

Policy Multi-user testing is performed after the integration testing.

7.9 Usability Testing

The objective of Usability testing is to ensure the user friendliness of the screens. As there are several roles for the users of the system, the testing should be performed with respect to the user interfaces for various roles in order to ensure the quality of the system.

Policy This testing is performed through all stages of the development.

7.10 Operation Acceptance Testing

The objective of Operation Acceptance testing is to ensure that processes and procedures are in place to allow the entire system to be used and maintained.

Policy This testing phase is to be performed prior to deploying the system to a live site.

7.11 Database testing

The objective of Database testing is to ensure whether the queries are retrieving data from the database as expected.

Policy This testing is performed after the database and all the queries that the project requires are created. This is performed manually.

References

- [1] Feasibility Study for Physics Gradebook System.
- [2] Requirement Specification for physics Gradebook System.
- [3] P465-6 & P565-6 Software Engineering for Information Systems I & II: Information Packet, Computer Science Department, Indiana University, 2008.
- [4] IEEE Standard for Software Quality Assurance Plans (STD 730-1984), Inst. of Electrical and Electronics Engineers, New York, 1984.
- [5] Requirement Specifications of Sakai Gradebook. <https://source.sakaiproject.org/svn/gradebook/trunk/xdocs/specs23/index.htm>
- [6] Information about Integrating CAS with a website. <http://kb.iu.edu/data/atfc.html>
- [7] PHP-Unit Project, <http://www.phpunit.de/manual/3.3/en/index.html>
- [8] Selenium Project, <http://selenium.seleniumhq.org/>
- [9] OpenWebLoad Project, <http://openwebload.sourceforge.net/>
- [10] NIKTO Project, <http://www.cirt.net/nikto2>

List of Tables

1	PERSON Table Fields	21
2	PERSON Table Indexes	22
3	COURSE Table Fields	22
4	COURSE Table Indexes	22
5	OFFERING Table Fields	23
6	OFFERING Table Indexes	23
7	SEMESTER Table Fields	23
8	SEMESTER Table Indexes	24
9	COMPONENT Table Fields	24
10	COMPONENT Table Indexes	24
11	COMPONENT_TYPE Table Fields	25
12	COMPONENT_TYPE Table Indexes	25
13	COURSE_DEFAULT_COMPONENTS Table Fields	25
14	COURSE_DEFAULT_COMPONENTS Table Indexes	26
15	ENROLL_COMPONENT Table Fields	26
16	ENROLL_COMPONENT Table Indexes	27
17	ENROLL_OFFERING Table Fields	27
18	ENROLL_OFFERING Table Indexes	28
19	JOIN_SECTION Table Fields	29
20	JOIN_SECTION Table Indexes	30
21	SECTION Table Fields	30
22	SECTION Table Indexes	31
23	TEACH_SECTION Table Fields	31
24	TEACH_SECTION Table Indexes	32
25	SESSIONS Table Fields	32
26	SESSIONS Table Indexes	32
27	ASSIGN_GRADES Table Fields	33
28	ASSIGN_GRADES Table Indexes	34
29	LATEST_ASSIGNED_GRADES Table Fields	34
30	LATEST_ASSIGNED_GRADES Table Indexes	35
31	ROLE_TYPE Table Fields	35
32	ROLE_TYPE Table Indexes	35
33	ASSIGNMENT_TYPE Table Fields	36
34	ASSIGNMENT_TYPE Table Indexes	36
35	ASSIGNMENT Table Fields	36
36	ASSIGNMENT Table Indexes	37
37	Navigation from Welcome screen	40
38	Navigation from Administrator Gateway	43
39	Navigation from Add Offering screen	44
40	Offering Screen for Administrator	45
41	Navigation from Modify Course screen	49
42	Navigation from Modify Offering Grades screen	52
43	Navigation from View Offering Grades screen	53
44	Navigation from Gateway screen	55

45	List of Requirements Changes	71
----	--	----

List of Figures

1	Architecture of the Services	7
2	Architecture of the Control Flow	8
3	Architecture of the Components	9
4	Menu Hierarchy at Login	38
5	Menu Hierarchy for Administrator	39
6	Menu Hierarchy for General Users	39
7	Welcome screen	40
8	Non-CAS Login screen	41
9	CAS Login screen	41
10	Administrator Gateway screen	42
11	Add Offering screen	43
12	Offering Management screen	44
13	Add Offering Component screen	46
14	Add Section To Component screen	46
15	Add Student screen	47
16	Add Assignment screen	47
17	Modify Assignment screen	48
18	Modify Course screen	49
19	Export Grades screen	50
20	Export Roster screen	50
21	Import Offering Grades screen	51
22	Modify Grades screen	51
23	View Offering Grades screen	52
24	User Gateway screen	54
25	Report Grades screen	55
26	Error Handling Schema	58

Index

Administration services, 7
Administrator, 39
Administrator Gateway, 43
Assignment, 36, 52

banner, 38
browser, 9, 14

CAS, 39, 40
Component, 24, 39
control model, 7
controller, 9
Core Libraries, 10
Course, 22–24, 27
course, 43
course hierarchy, 7

Data Manipulation, 10
Database testing, 61
DBMS, 9
Dialog box, 38

ER model, 21
error code, 16
Error Handling library, 10
error handling module, 16
error message, 16, 38
Exception Handling, 56

File system failure, 56
filtering, 52, 53
functionality testing, 59

gateways, 38
General Users, 39

heading, 38
HTML page, 13
HTML pages, 10, 12
HTTP Request header, 13
HTTP requests, 14

Instructor, 31, 32, 34, 54
integration testing, 59
Internet Explorer, 9

Multi-user testing, 60

objects, postfixed with 'key', 10
Offering, 22, 39, 43
offerings, 54
Operation Acceptance testing, 60
ozilla Firefox, 9

password, 42
Performance testing, 60
PHP Classes, 10
PHP Objects, 10
PHP session, 10, 16, 32
PHP sessions, 13, 14
picture, 40
Power Failure, 56
Professor, 22, 27, 54
property name, 10

redirect, 14
regression testing, 60
Regular page, 38
Regular user services, 7
Runtime exception, 57

Safari, 9
Section, 28, 30, 39
section number, 30
security testing, 59
Semester, 22, 23
semester, 43, 54
server side scripts, 14
Servlets, 10, 14
Session Management, 10
SQL queries, 10, 12
structural system model, 7
Student, 26, 52, 54
sub-system decomposition model, 7
Submit, 38, 43, 46

template files, 12
Template Library, 10
template library, 12
Temporary database failure, 56

testing, 59

unit testing, 59

Usability testing, 60

User Acceptance testing, 60

User and Permissions, 10

user interface, 38

User Interface Generators, 10, 13

user permission, 14

user permissions, 13

username, 42

Web server failure, 56

Nomenclature

banner The picture to indicate the existence of the system.

Camel Case A mixture of upper and lower case letter, e.g GObjectClass.

course hierarchy Each year, a course is offered in an Offering. An Offering consists of Component(s), and students in each component were divided into Section.

ER model Entity Relation model, a common methodology for information system modeling.

Gecko Mozilla web browser engine.

Nikto Nikto is a tool for finding default web files and examining web server and CGI security.
Reference link:<http://www.cirt.net/nikto2>

OpenWebLoad OpenWebLoad is a tool for load testing web applications. It aims to be easy to use and providing near real-time performance measurements of the application under test.
Reference link:<http://openwebload.sourceforge.net/>

PHPUnit It is a member of the xUnit family of testing frameworks and provides both a framework that makes the writing of tests easy as well as the functionality to easily run the tests and analyse their results.
reference link:<http://www.phpunit.de/manual/3.3/en/index.html>

Selenium Selenium is a suite of tools to automate web app testing across many platforms.
Reference link:<http://selenium.seleniumhq.org/>

Servlets Servlets are server side scripts that respond to the HTTP requests from the client browser but do not directly push any content to the browser.

Session A session is a semi-permanent interactive information exchange, between two or more communicating devices, or between a computer and user.

Trident known as MSHTML, the html layout engine from Microsoft.

WebKit Google Chrome and Safari are based on WebKit browser engine.

A List of Requirements Changes

New Requirement	Old Requirement	Reason for Change
Assignments are classified by types	N/A	In a lecture component, assignments can be either homework or “clicker questions”
Roles: users, power users and administrators	Single Administrator	Secretaries also do the management tasks, but they do not manage user privileges; client requirement
IUP users and CAS users	All CAS users and the administrator authorized internally	Some students do not have CAS account; specified by client
Basic validation for the grades	N/A	Some grades are obviously ill-formed and should be excluded, e.g., “A B”, “3.B”, and extra spaces
A report for number of students in each section	N/A	New client requirement
Grades can be as long as 20 characters	N/A	To allow for verbose grades like “Excellent”.
Calculate the average grade in a component (per student)	N/A	The simple average grade helps in calculating the final grades; new client requirement
The system must be made secure from regular external hackers	N/A	New client requirement

Table 45: List of Requirements Changes

B Coding Standards and Conventions

B.1 Introduction

The goal of these guidelines is to facilitate the programmers to create uniform code that is easier to understand and maintain. In this document, the guidelines on PHP modules, database naming and user interface design are listed.

B.2 Modules

B.3 Module Decomposition

All source code will be grouped into modules. The team is not going to adopt Object Oriented features of PHP; rather, each module corresponds to a name-space or class that deals with a single, unique domain.

- One source code file corresponds to one module in the system.
- If the module deals with the objects, the handler of the object is passed in function calls instead of global variables.

B.4 Headers

Headers in PHP and JavaScript are different from other languages like C or C++. In PHP and JavaScript, the modules themselves are contained in the headers.

B.4.1 PHP Headers

- All the headers are included by the same entrance file "include.php" which resolves the location of other headers and includes them. This is to avoid troubles for source code in different sub-directories to locate the headers.
- Use "include" statement to include the "include.php" file.
- Site configuration files are also provided as PHP headers. Missing site configurations should not cause the system to panic in a runtime-error manner.
- All global variables should be declared in a single file; the initialization is in each module.
- Each file begins with the includes, followed by context lines. Then the declarations of functions follows.
- Public member functions are prefixed with the module name.
- Private member functions are prefixed with an underscore ("_").

B.4.2 JavaScript Headers

- HTML pages include JavaScript headers with relative path names.
- Dependencies are resolved in the HTML pages. JavaScript headers never resolve the header dependencies.
- No global variables are preserved between JavaScript headers.
- Public members are prefixed with module name.
- Private members are prefixed with an underscore (“_”).

B.4.3 CSS Headers

- CSS headers are included statically in the HTML pages.

B.5 Commenting and Indentation

B.5.1 File Header

Each source file begins with a header section consisting of comments. The purpose of the header is to describe the general characteristics of the file and the module.

```

/*****
 * Reference to the Copyright Licenses (GPL)
 *
 * FILE: sample.php
 * MODULE NAME: sample_module
 *
 * DESCRIPTION: This is a sample module (DEPRECATED).
 *
 * NOTES: If the file is included, the application dies.
 *****/

```

Fortunately, PHP, JavaScript and CSS share the same syntax of commenting.

B.5.2 Member Function Header

Each member function in a module should have a comment header to describe the behavior of the member function. The gtk-doc¹ convention is used.

```

/*****
 * sample_function:
 *   @self: the handler of sample object.
 *   @param2(out): the second parameter.
 *
 * sample_function do nothing. It should do something.
 *****/

```

¹<http://www.gtk.org/gtk-doc/>

```

*      Refer to @sample_function2.
*
* returns: TRUE if success, FALSE if not.
*
* Changes: added in rev 212.
*
*****/

```

B.5.3 In-line Commenting

In complicated member functions, appropriate in-line comments should be written to describe the process. These in-line comments begins with "/*", follows by a line to describe the comment and stops with "*/". The in-line comments should always follow the same indenting rules of the code it is explaining.

- “FIXME” stands for known rare wrong behavior.
- “NOTE” stands for special notes about the code.
- “TODO” stands for unfinished implementations.

```

.....
....
/* FIXME: divided by zero is not handled */
$i = $i / $number;
....
....

```

B.5.4 Indentation Rules

Indentation are by Tabs (ASCII code 08). Every programmer's editor should be configured to visualize Tabs with 4 spaces.

Context elements that increase the indentation level

- Context level
- Member functions
- Code blocks

```

/*****
* FILEHEADER
*****/
$GLOBAL_VAR = "something";
/*****
* HEADER
*****/
function sample_function() {
    .....
}

```

```
    /* TODO: write stuff */  
    .....  
  
    .....  
}
```

B.5.5 Line Breaking Rules

- Between comment header and member functions there should be no blank lines.
- Between in-line comments and the code there should be no blank lines.
- Add a line break after a curly brace.
- Add a line break after each code block.
- Use common sense to add extra line breaks when semantic groups of code finish.

B.5.6 Blank/Space Rules

- Operators and operands are generally separated by a space (ASCII 32).
- Commas should be followed by a space.
- Avoid continuous spaces.
- In the function definition, a space should be inserted between the function name and the formal parameter list.
- In a function invocation, no space should be inserted between the function name and the argument list.
- A space should always be inserted before an opening curve brace (“{”).

B.6 Naming Conventions

The general guidelines for naming is to use descriptive names. Therefore the names tend to be long and verbose.

B.7 Database Schema

- Table names should be in upper case. Multiple words are separated by underscores (“_”).
- Field names should be in lower case. Multiple words are separated by underscores (“_”). In a field name, the name of the table should be prefixed in lower case if possible. E.g., `table_field` is a field for table `TABLE`.

B.8 File Naming Conventions

- File names for internal modules are in lowercase. If it contains multiple words, separate them by underscores (“_”). No spaces are allowed. For example, `sql_template_manager.php` is a valid file name.
- File names for external modules (UI and Servlets) are in Camel Case.
- File names for JavaScript modules are in lowercase.
- File names for SQL query templates are in lower case.
- File names for HTML templates are the same as the UI module names, with an extension “`tpl`”.

B.9 Local Variables

- Local variables are in lowercase, separated by underscores (“_”).
- Module static(private) variables are in uppercase, separated by underscores (“_”) and prefixed with underscores (“_”).

```
$_SAMPLE_LOCAL_VARIABLE = "local";
function sample_function () {
    $local_variable = "value";
}
```

B.10 Global Variables

- Global variables are in uppercase, separated by underscores (“_”).

```
$DB_USERNAME = "dbadmin";
```

B.11 User Interface

User interface modules require special standardization.

B.12 Form Layout

- The banner² should always be at the top of the form.
- Color highlighting should not be abused; and special considerations for visually handicapped people should be taken.
- The submit button should always be on the left bottom of the form.
- Graphics should have alternate text.
- Input fields should have labels.
- Tables should have captions.
- Large tables should be scrollable and limited to the browser area of the user.

²The picture to indicate the existence of the system.

B.13 Form Navigation

- Forms are linked with plain hyperlinks.
- JavaScript should be avoided for submitting forms and form navigation.

B.14 Informative Messages

- Informative messages should be present just below the title of the form.
- A box or similar structure should be used to separate the informative messages from the other elements of the form.
- Informative messages should not be highlighted.

B.15 Key Bindings

- Default key bindings of Web Browser should not be altered.
- Tabs sequence should be redefined for modifying grades.