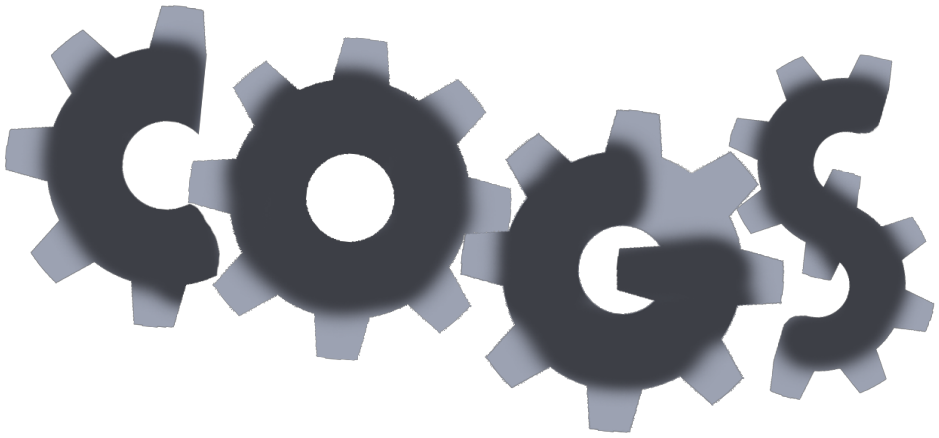
Design Document DEC15-21
COGS - Central Online Grading System

Version: 3.0



Team

Kathryn Widen: Team Leader

Forrest Scott: Webmaster

Zachary Lones: Communication Leader

Daniel McDonough: Key Concept Holder 1

Daniel Riechers: Key Concept Holder 2

Advisor/Client: Dr. Daniels

Table of Contents

- Project Statement
  - Definitions
- Use Cases
  - Scoring an Assignment:
  - Creating an assignment:
  - Student submitting an assignment:
  - Submitting Grades To Bb:
  - Creating a new class (once per semester):
  - Edit a class (like create a class)
  - Cheating:
  - Non-functional Requirements:
- Functional Requirements:
  - Software Requirements
    - Grader
    - Web Interface
    - Cheating detection
  - System Requirements
    - Security Requirements
    - Integration Requirements
- System Block Diagram
- PHP Framework
  - Zend Framework 2
  - Modules - what we make
  - Configuration - let ZF2 do the busy work
  - Controllers - the main brains
  - PHP Framework Directory Map

Zend Framework2 Execution Flow
Testing
Screen Sketches

Project Statement

COGS seeks to make the grading process easier for TAs and professors by automating some steps of grading, detecting cheating, and automating the grade upload to Blackboard.

Definitions

Server: Hosts grader, web interface, and score database
Grader: Compiles and runs student code, is not a human
Grade Report: The output of the compiler and code as well as source code. (ie compile
errors, and code output) This is what the instructor uses to score submissions.
Cheating Report: A report that includes all cheating related info that is generated for the
final submission for each student per assignment.
Student: Person who submits code to COGS for review
Instructor: Person who uses grader report to assign a score to submissions.
Head Instructor(s): Person with slightly more power, allowed to submit grades to Bb
Submission: A student's single attempt at an assignment.
Unit Tests: Automated tests performed by the grader on student code (not to be confused with unit testing for testing the functionality of the system.)
COTS: Commercial off the shelf.
ZF2: Zend Framework 2, the php framework for the COGS web front-end
Structural Requirements Analysis: Custom searches instructors can program that can analyze source code. For instance, a Structural Requirement can be: "Does the student use a for loop?"

Use Cases

Creating an assignment:

- An Instructor navigates to the Class List Page
- The Instructor finds their desired class and open Assignment List Page
- The Instructor clicks on Add a New Assignment.
    - (Optional) The instructor may click Copy to a New Assignment
- The Instructor will fill out or edit the following values using a form:
    - Begin Date/Time
    - Due Date/Time
    - Last Accepted Date/Time
    - Number of Points Deducted Per Day Late
    - Assignment Title and Description
    - Attachable Files (optional)
    - Unit Tests* (optional)
    - Structural Requirements Analysis* (optional)
    - Checkbox Based Grading (optional)
    - Numberbox Based Grading (optional)
    - Total Score Possible

  (*) Denote a stretch goal use case or requirement.

Student submitting an assignment:

- Student navigate to an assignment submission page
- Students upload their source code
- Students may write submission notes
- Students may create any number of execution input blocks
    - Students press Add New Input button to open a new Text Input Element
    - Students press Submit to submit assignment

Student viewing submission result:

- Student navigates to Assignment Submission List
- Student clicks on their desired submission result that has completed execution (not pending)
- The ungraded submission result page would show the preliminary report:
    - Student Source Code
    - Compiler or runtime errors
    - Unit Test Results*
    - Execution Input/Output
- The student may click on Resubmit to resubmit the assignment.

Scoring an Assignment:

- The Instructor navigates to the assignment list
- The Instructor selects the assignment to grade
- A scoring page is presented
    - Page contains Student source code
    - Summarization of Cheating Report and Link
    - Execution runs and results
    - Scoring checkboxes
    - Scoring numberboxes
    - Final grade adjuster
    - Instructor notes
    - Next button
- A instructor will see an alert if the cheating algorithm detects significant changes of cheating.
    - The instructor can click on the alert to see the cheating report
- The instructor can re-execute the code with custom execution input
- The instructor fills out all of the scoring form
- The instructor clicks on Save and Next to start grading next assignment

Submitting Grades To Bb:

- The Instructor navigates to assignment List
- The Instructor presses “Download Grades as CSV” to download the grade file
- The Instructor uses Blackboard interface to upload new grades

Creating or Editing a new class (once per semester):

- The Instructor navigates to the class list
- The Instructor presses New Class to open the new class form
    - Optionally the Instructor can press Edit Class to edit a class in a form
- The Instructor will fill out the following form:
    - Name
    - Semester ID
    - Dates
    - Sections
- The Instructor can enter CSV data to add students
- The Instructor can enter CSV data to add Instructors

Cheating:

- The Instructors view the cheating report

- All cheating concerns are handled off system

Non-functional Requirements:

- System will prioritize efficiency for Instructor/Head Instructor allowing grading assignments to be done with minimal time/clicks.
- The policy used for Mandatory Access Control will be transparent, and maintainable for system administrators.
- All COTS software will be verified to be secure and reliable.

Functional Requirements:

Software Requirements

- **Grader**
    - The system will work with both Gcc and Clang
    - The system will compile students' code and generate reports.
    - The system will provide compile output in the Preliminary Report.
    - The system will give students the option to provide stdin inputs that will be used in the execution of their code.
    - The system will show student input and their program's output in the Preliminary Report.
    - The system will give Instructors the option to provide new inputs for student's code. The new inputs and outputs will be shown in the Final Report.
    - The system will run unit tests if they have been provided by the Instructor in the assignment. The system will generate a score according to pass/fail of unit tests, this score will be included in the Preliminary Report.
    - The system will allow Instructors to modify any scores given by Grader, and the modified score will be shown in the Final Report.
    - Student will be provided both the Preliminary Report, and the Final Report.

- **Web Interface**
    - Creating Assignments
        i. The system will allow Instructor to create an assignment that has the following properties:
            1. Begin Date/Time
            2. Due Date/Time
            3. Last Accepted Date/Time

4. Number of Points Deducted Per Day Late
5. Assignment Title and Description
6. Attachable Files
7. Unit Tests*
8. Structural Requirements Analysis*
9. Checkbox Based Grading
10. Numberbox Based Grading
11. Total Score Possible

ii. The system will allow an old assignment to be selected as a template for a new assignment.
iii. The system will allow Instructor to include Unit Tests with assignment.*
iv. The system will allow Instructor to include checkboxes that will modify student score when being graded.
v. The system will allow submissions for assignments only between Begin Date/Time and Last Accepted Date/Time.

- Student Submission
  i. After the Begin Date/Time is passed Students will be able to submit assignments
  ii. Students won't be able to submit assignments past the Last Accepted Date/Time
  iii. Students will be deducted points if the final submission is past the Due Date/Time, calculated from "Number of Points Deducted Per Day Late".
  iv. The system will allow multiple submissions and use the last submission for grading.
- Assignment Executing
  i. The Back End will generate an execution report for each submission by the student.
  ii. The Execution Report will include all compiler and run-time errors.
  iii. The Execution Report will include all terminal output per execution run.
  iv. The system will display compiled code and Report to student when they submit code. This report will include passed/failed unit tests and score if applicable.
- Assignment Grading
  i. Cheating Reports and Grade Reports will be received from the Back End after the Final Submission Date/Time.
  ii. The system will divide workload of Instructors according to Grading Group.
  iii. The system will provide a page that Instructors to use that will streamline the grading process.
    1. The page will show the student's source code
    2. The page will have itemized checkboxes from the assignment creation

3. The page will have itemized numberboxes from the assignment creation
4. The page will have a numberboxes that will allow the instructor to modify the final grade manually
5. There will be a text input for the instructor to enter notes.

iv. The system will implement locks to prevent race-conditions when multiple instructors are grading the same assignment.
v. The system will alert the instructor when cheating is detected.

- Cheating Detection Front End
  i. Cheating Report will be received from the Back End Grader.
  ii. Cheating Reports will be linked from the Grading page.
  iii. Cheating Reports will include links to submissions that are similar.
- Grade Submission Front End
  i. The system will provide, at request of Head Instructor, a file containing students' scores for an assignment as well as other necessary information in a format that it can be manually checked and then submitted to blackboard.
  ii. Grades will also be viewable by the students via the Front End.
- Class Editing
  i. The system will allow the creation of a new Class with the following properties:
     1. Name
     2. Semester ID
     3. Dates(?)
     4. Add accounts with following properties:
        a. NetID
        b. Type (Instructor/Student)
        c. Section
        d. Grading Group
  ii. The system will allow multiple accounts to be added at once via upload file of comma separated values.
  iii. The system will allow account to be added by manually typing required properties.

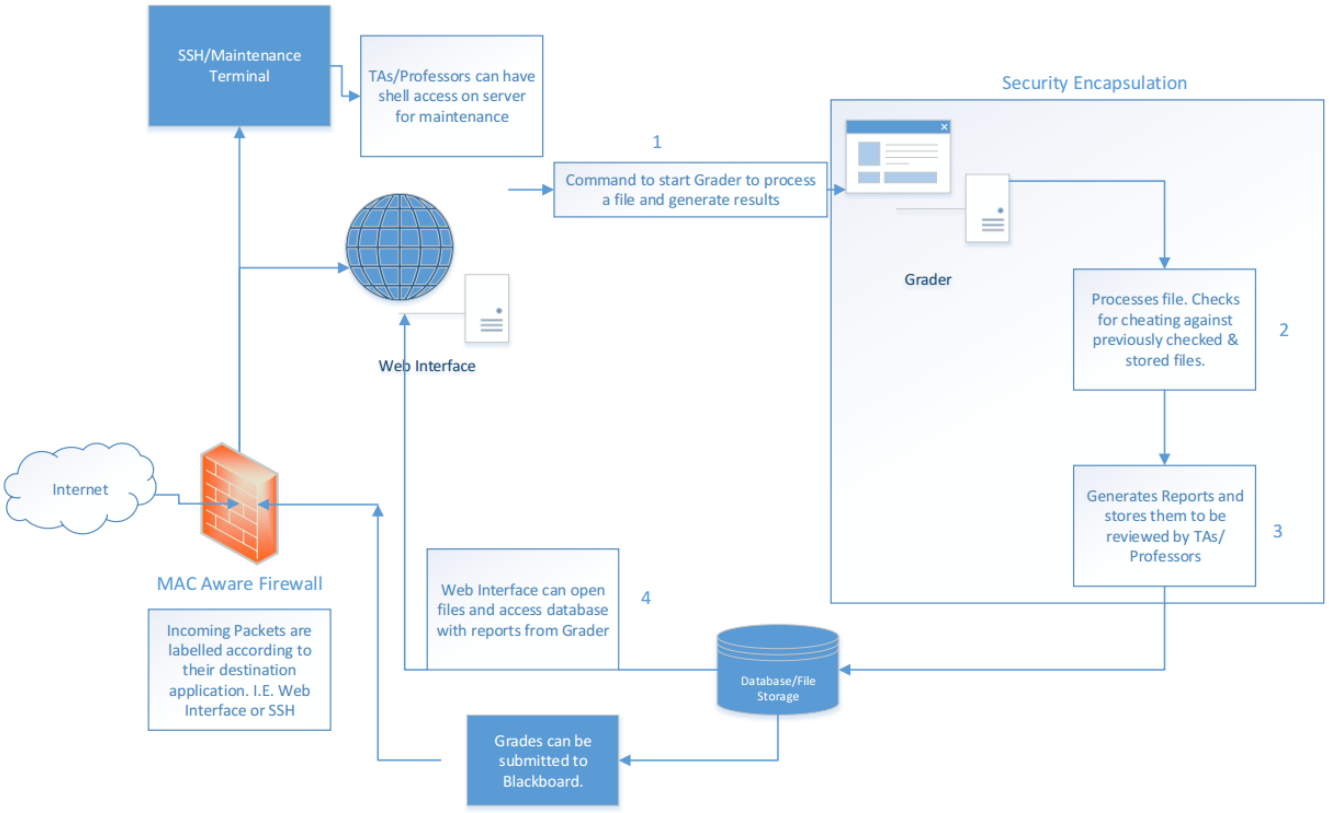
- **Cheating detection**
  - The system will run cheating detection algorithm on all students' code after Last Accept Date/Time has passed.
  - The system will include any cheating detection in the Final Report.
  - The system will generate a Report viewable by Instructors if bulk cheating is detected.

System Requirements

- ### Security Requirements
    - The system shall use Mandatory Access Control to encapsulate student code.
    - The system shall use a chroot environment to encapsulate student code.
    - The system shall use a firewall that labels packets with a Mandatory Access Control policy context.
    - The student code will be allowed to execute system calls used for the reading, writing, creating, opening, and closing of data files, and terminal input and output.
    - The student code will not be allowed to execute system calls not enumerated in the requirements.

- ### Integration Requirements
    - The grader component will be run by the web server and given command line options.
    - The grader component will return its output to the web server as a file.
    - The grader component will be interfaced to a database for storing student scores.
    - The Blackboard interface will only obtain student scores from the database.

System Block Diagram



PHP Framework

Zend Framework 2

To meet the requirements for our front-end php website, we have chosen to use the php framework Zend Framework 2 (ZF2). With its efficient use of resources, high customizability, existing code base, security, and reliability, Zend Framework is an excellent choice. Additionally, ZF2 uses Doctrine 2 as its object relational mapper. Doctrine 2 is a powerful tool that will aid in obfuscate the complex database relations and queries. However, a developer must have a strong top level understanding of the framework before creating content.

Modules - what we make

.ZF2 uses modules to separate its main extra components. We will create one module named COGS, that ZF2 will include. The module will hold all of the code (javascript,php,html) that will hook into ZF2.

Configuration - let ZF2 do the busy work

Top-level flow of routing is shown in the following figure. Luckley, ZendFramework will auto load all necessary resources and route complex urls intuitively. The team will have to modify a few key configuration files within ZF2 in order for ZF2 to understand the navigation map. The following list describes the config files the team will have to manage (many will get quite large with rules and mappings).

Application.config

This is the main configuration for the Entire ZF2 framework. It lists the required modules and adjusts the listener options.

Module.config

This will be the main configuration file for our COGS module. It will contain all of the the routing rules, plugin requirements, and configuration arguments for various ZF2 managers. This will be a large config file.

Local.config

This will hold private, local data like user information and database passwords.

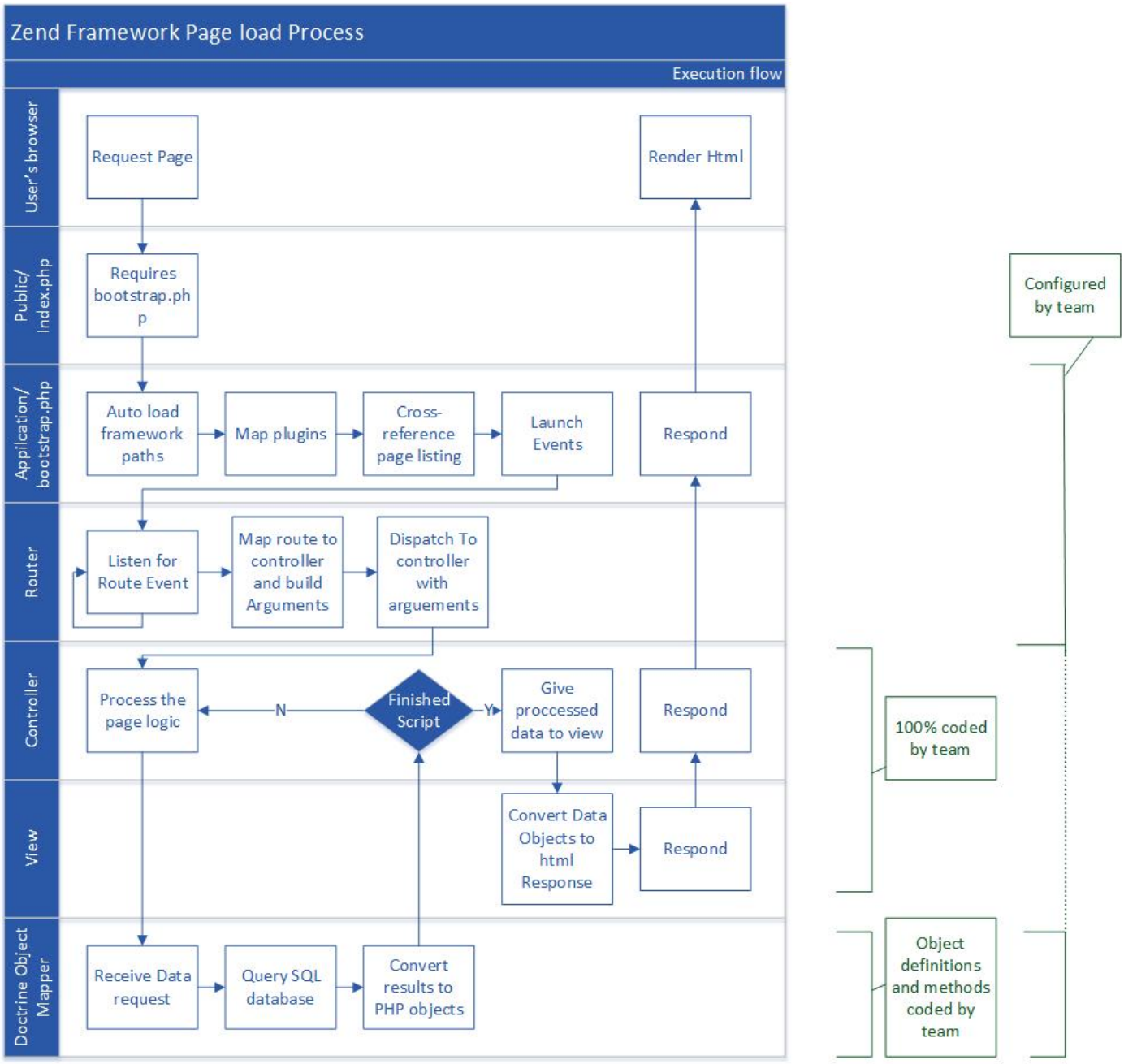
Controllers - the main brains

After configurations are properly setup, ZF2 will handle the technical work of routing and resource handling. Once a user requests a page, ZF2 will present the correct controller file with a request object. We will write the controller files and they will do the main calculations behind ever user interface. The controller files will use ZF2 entity manager to talk to the database, and use the View files to build the html. It is within the Controller files where all of the algorithmic coding will be done. All other files will contain supporting functions and definitions.

PHP Framework Directory Map

```
-www root-
config/
        autoload/                   -config files to autoload
        data/                       -autogenerated scripts for optimization
module/                             -directory to hold ZF2 modules
        application/                -default ZF2 module (contains placeholder views)
        COGS/                       -the custom Module
                config/             -module configuration file(s)
                src/                -contains all of the php code
                        Controller/ -contains the main logic code
                        Entity/     -contains the mapping code for the database
                        Form/       -contains the form objects
                        Plugin/     -contains helper code for the Controllers
                view/               -contains the phtml views files
public/                             -index.php (start point), and public resources (img)
vendor/                             -ZF2 engine and managers (do not touch)
```

Zend Framework2 Execution Flow



This diagram describes the top level execution flow for the PHP framework. To describe the process simply: ZF2 uses a combination of autoloaders and event driven managers that route to Controllers, which in turn process user input, and eventually return html via view files.

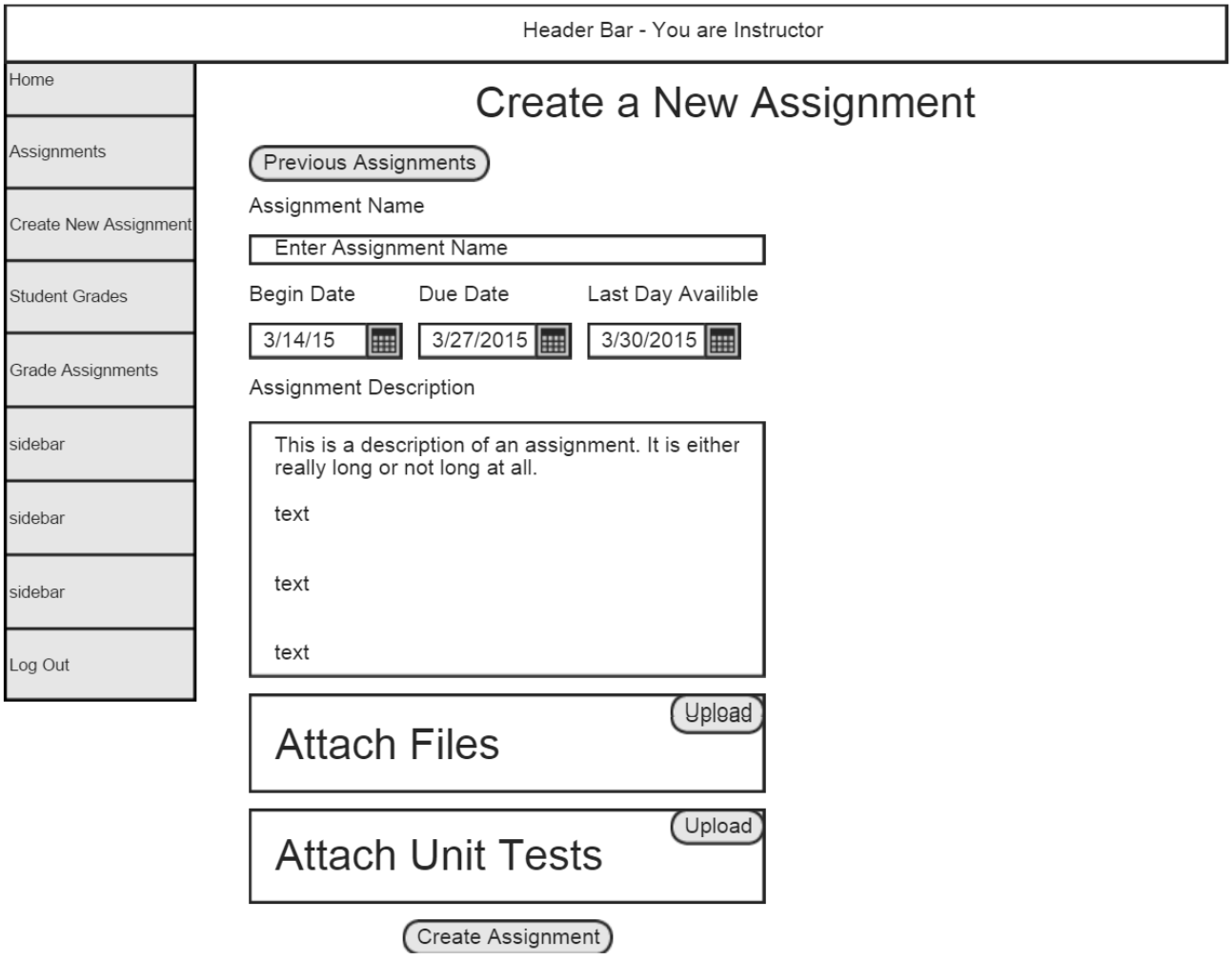
Testing

The components of the system will be unit tested using Google Test. Demonstration, manual testing, and inspection will also be performed on the system.

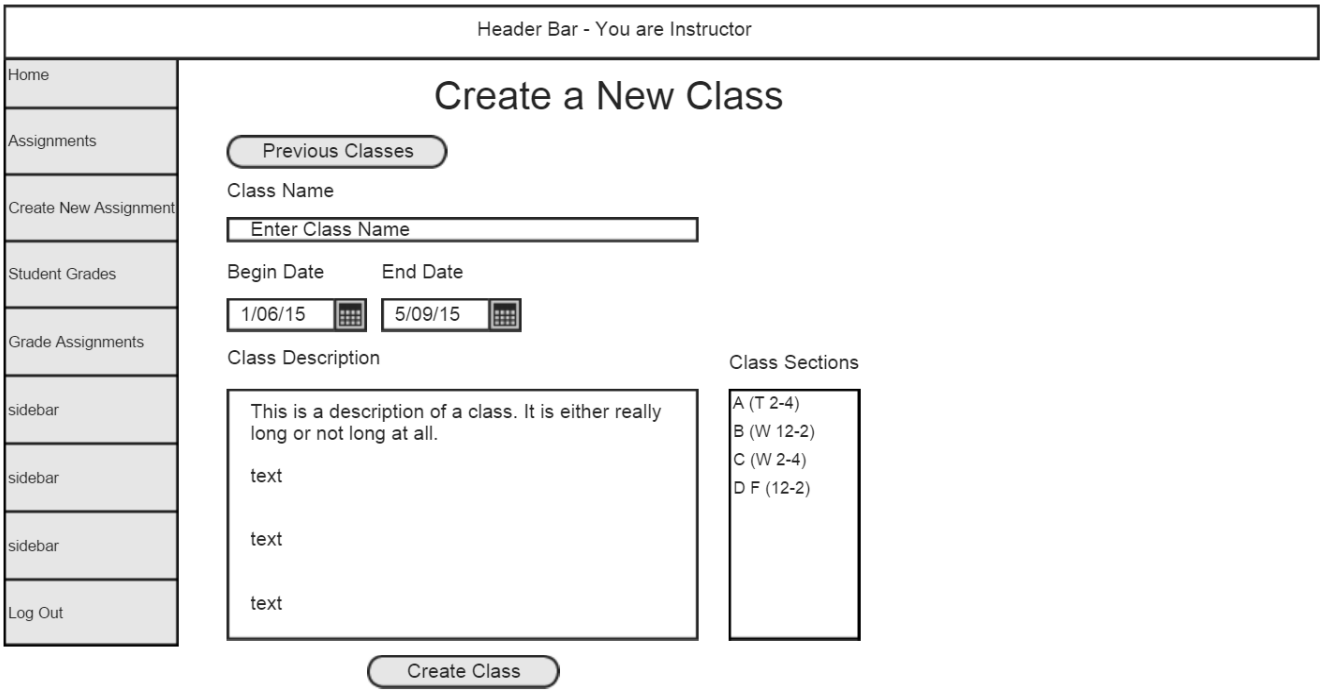
- The proper flow of the interface will be tested via demonstration.
- All grader functionality will be unit tested.
- All functionality of the Blackboard interface will be unit tested.
- All functionality of the Cheating detector will be unit tested.
- The encapsulation of student executables using the Mandatory Access Control will be unit tested.
- The chroot encapsulation of student executables will be manually tested for vulnerabilities.
- COTS software will verified as secure and reliable by checking packages on the National Vulnerability Database.
- Firewall configuration and context labeling will be tested via demonstration.

Screen Sketches

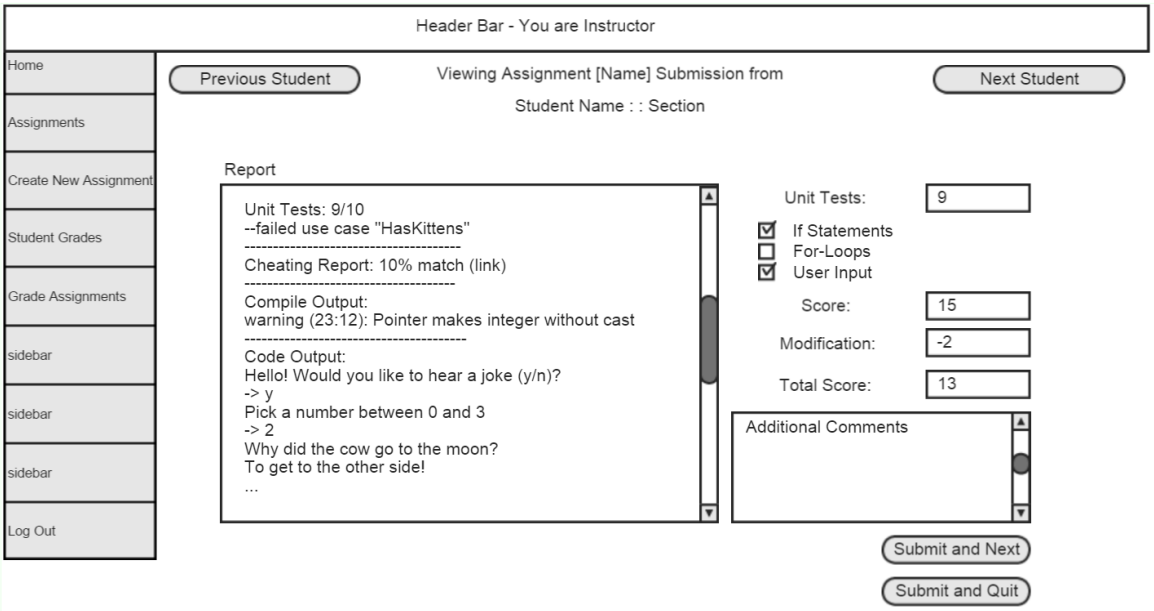
Create a New Assignment: This page is where instructors will create new assignments for students to turn in. This page will only be visible to instructors and includes locations for adding assignment name, dates, descriptions, and attaching files and unit tests. The “Previous Assignments” buttons allows the instructor to use a previous assignment as a template, useful in the event of assignments being very similar or exactly the same as a previous assignment. Assignment name and dates are required by the system to create an assignment, while the other fields are all optional.



Create a New Class: This screen is used once per semester by an instructor to create an instance of a class. To create a class an instructor must include a name and dates. A description and sections are optional. The sections are used by students to pick their section, which can be useful in grading. This page is only visible to instructor users.



Grading Screen: This is the screen seen by instructors to be used for grading. The screen shows the students name and section as well as the students “report”. The report includes the students source code, compiler output, code output, and “cheating” score. If unit tests are used, then the unit tests scores and information will be shown. The screen also includes scoring rubrics. The score for unit tests is auto-generated by the system. The grader can check off if code has various attributes (pre-set when the assignment is made), which generates the value in the “score” field. The instructor can then modify the score if need be with the modification field. The total score is what the score the student will receive for the assignment.

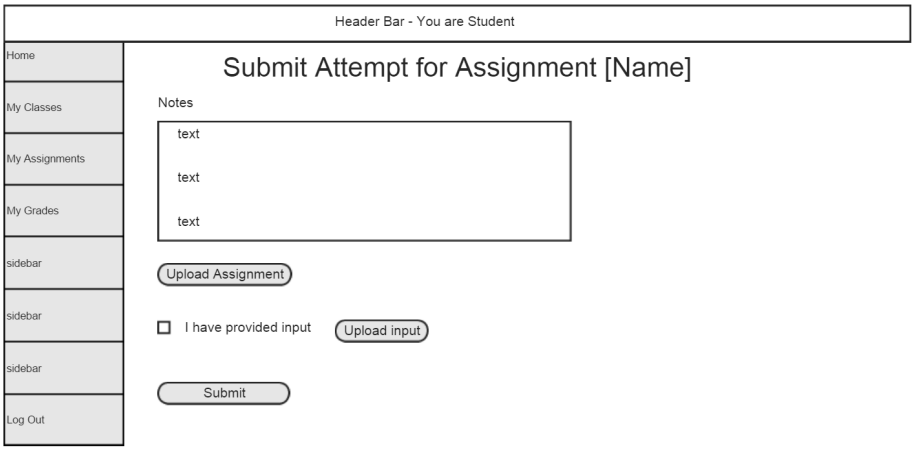


The log in screen for COGS will use pubcookie to authenticate, and so students and instructors will use their NetID and password to log in.

The new user sign up has students enter their name and NetID and select the class and section. This enters the student into the system and associates them with a class that already exists and a section that exists. If the class has no sections associated with it, the section

dropdown will be grayed out.

The page for a student to submit an assignment includes locations for attaching a file to submit, attaching input, and adding comments. The student has to indicate that they are including input.



Welcome screen: Just a splash page to catch users, includes information about COGs and links for logging in and signing up.

Welcome to COGS!

Login Sign Up

What is COGS?
Lorem ipsum dolor sit amet, maiores ornare ac fermentum, imperdiet ut vivamus a, nam lectus at nunc. Quam euismod sem, semper ut potenti pellentesque quisque. In eget sapien sed, sit duis vestibulum ultricies, placerat morbi amet vel, nullam in in lorem vel. In molestie elit dui dictum, praesent nascetur pulvinar sed, in dolor pede in aliquam, risus nec error quis pharetra. Eros metus quam augue

Created by Senior Design Team Dec15-21