# Design Document DEC15-21
# COGS - Centralized Online Grading System

Version: 1.0

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

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

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.

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
onfused 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

# Use Cases

## Scoring an Assignment:

-No Unit Tests, no automated scoring

    -Code does not require input

        -Grader runs code, creates report for instructor to score

    -Code requires input

        -Student provided input, grader runs code with students input for instructor to score

        -Instructor is prompted for input, report is generated after code terminates

-Unit Tests

    -Grader runs codes with assignment-spec provided inputs, compares outputs to assignment-spec, creates report with cases passed/failed, as well as standard report information.

        -Instructor modifies grades

            -modify auto-graded points

            -add/subtract points for additional criteria

            -modify total score

        -Instructor does not modify grades

## Creating an assignment:

Always included in creating an assignment:

    -Begin Time

-Due Date/Time
-Last Accepted Date/Time
-Assignment Title and Description
-Total Score Possible

Additional Options (Optional)
-Copy a previous assignment , force modify time/date options
- Unit Tests
-Instructor submits inputs and expected outputs
-Instructor configures Unit Test options
- Check-boxes
-Instructor configures checkbox labels and values

## Student submitting an assignment:
Everytime:
Student sees compile and output report.
Student provided input
-Students submits input and code.
No Student provided input
Not Unit Tested
-Student submits code and indicates if the code prompts a user
Unit Tested
-Student submits code, and sees Unit test results with report

## Submitting Grades To Bb:

-When all submissions for an assignment are graded by instructors, the Head
Instructor is allowed to submit scores to Blackboard (note, we want a human to control
the communication with Bb)

## Creating a new class (once per semester):
-Everytime:
-Name
-Semester ID
-Dates?
-Add accounts
-TAs
-Students
-Add sections

**Edit a class (like create a class)**

**Cheating:**
-When grader runs , part of  Instructor's report has cheating information.
-Bulk cheating results are viewable by a certain level of instructors.

**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 compile students' code and generate Report.
  - The system will provide compile errors if code fails to compile.
  - The system will terminate compilation if code is determined to be uncompilable.
  - The system will give students the option to provide inputs that will be used in compilation and Report. The inputs and outputs will be included in the 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 Report.
  - The system will run unit tests if they have been provided by Instructor in the assignment. The system will generate a score according to pass/fail of unit tests, this score will be included in the Report.
  - The system will allow Instructors to modify any scores given by Grader.
  - The system will automatically deduct points from score in Report if an assignment is submitted after Due Date/Time.

- **Web Interface**
  - The system will allow Instructor to create an assignment that has the following properties:
    - i. Begin Date/Time
    - ii. Due Date/Time
    - iii. Last Accepted Date/Time
    - iv. Assignment Title and Description
    - v. Total Score Possible.
  - The system will allow an old assignment to be selected and any of the above values to be modified to create a new assignment.
  - The system will allow Instructor to include Unit Tests with assignment.
  - The system will allow Instructor to include checkboxes that will modify student score when being graded.
  - The system will allow submissions for assignments only between Begin Date/Time and Last Accepted Date/Time.
  - The system will allow multiple submissions and only generate Report for the most recent submission.
  - 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.
  - 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.
  - The system will allow the creation of a new Class with the following properties:
    - i. Name
    - ii. Semester ID
    - iii. Dates(?)
    - iv. Add accounts with following properties:
      1. NetID
      2. Type (Instructor/Student)
      3. Section
      4. Grading Group
  - The system will allow multiple accounts to be added at once via upload file of comma separated values.
  - The system will allow account to be added by manually typing required properties.
  - The system will divide workload of Instructors according to Grading Group.

- **Cheating detection**
  - The system will require authentication.
  - 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 Report.

- ○ The system will generate Report viewable by Instructors if bulk cheating is detected.
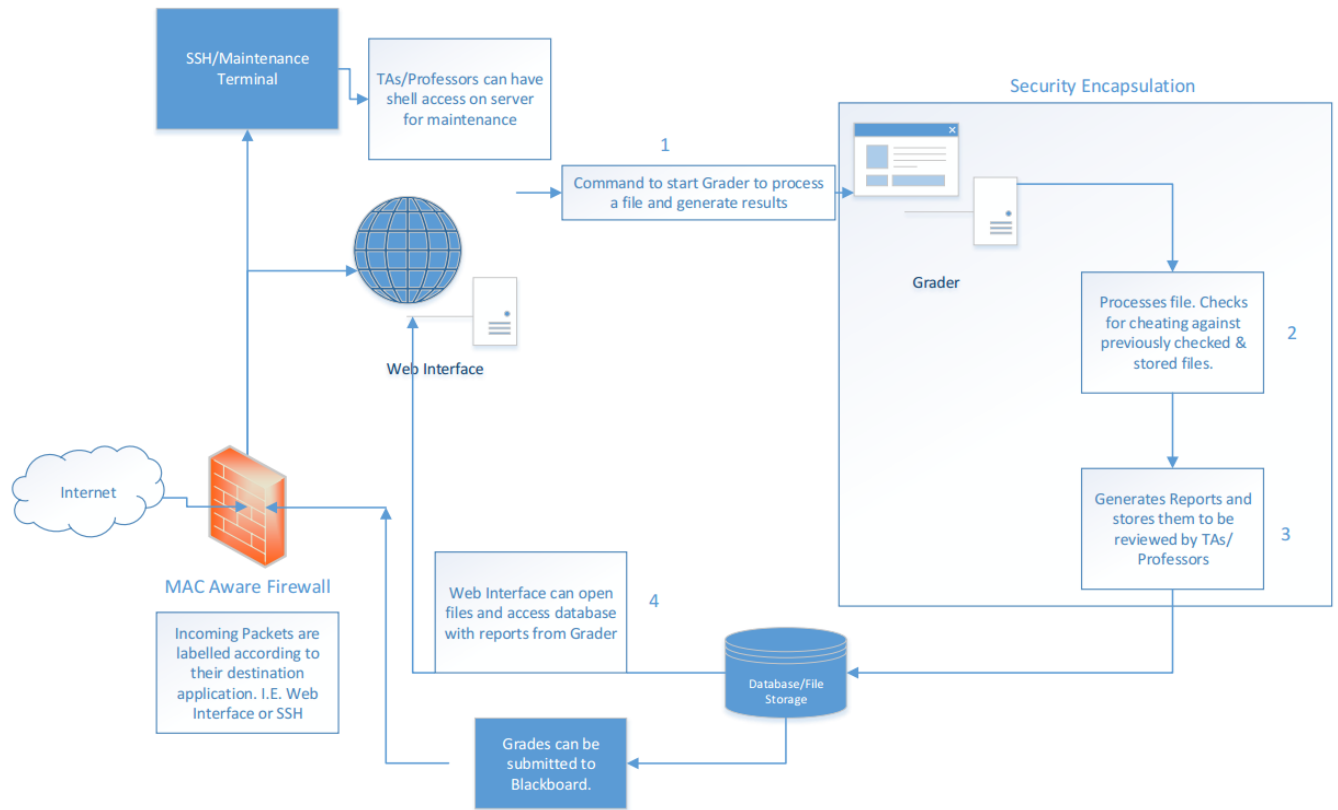
## System Requirements

- ● **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

SSH/Maintenance Terminal

TAs/Professors can have shell access on server for maintenance

1

Command to start Grader to process a file and generate results

Security Encapsulation

Grader

Processes file. Checks for cheating against previously checked & stored files.

2

Web Interface

Internet

MAC Aware Firewall

Incoming Packets are labelled according to their destination application. I.E. Web Interface or SSH

Web Interface can open files and access database with reports from Grader

4

Generates Reports and stores them to be reviewed by TAs/ Professors

3

Database/File Storage

Grades can be submitted to Blackboard.

# 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.
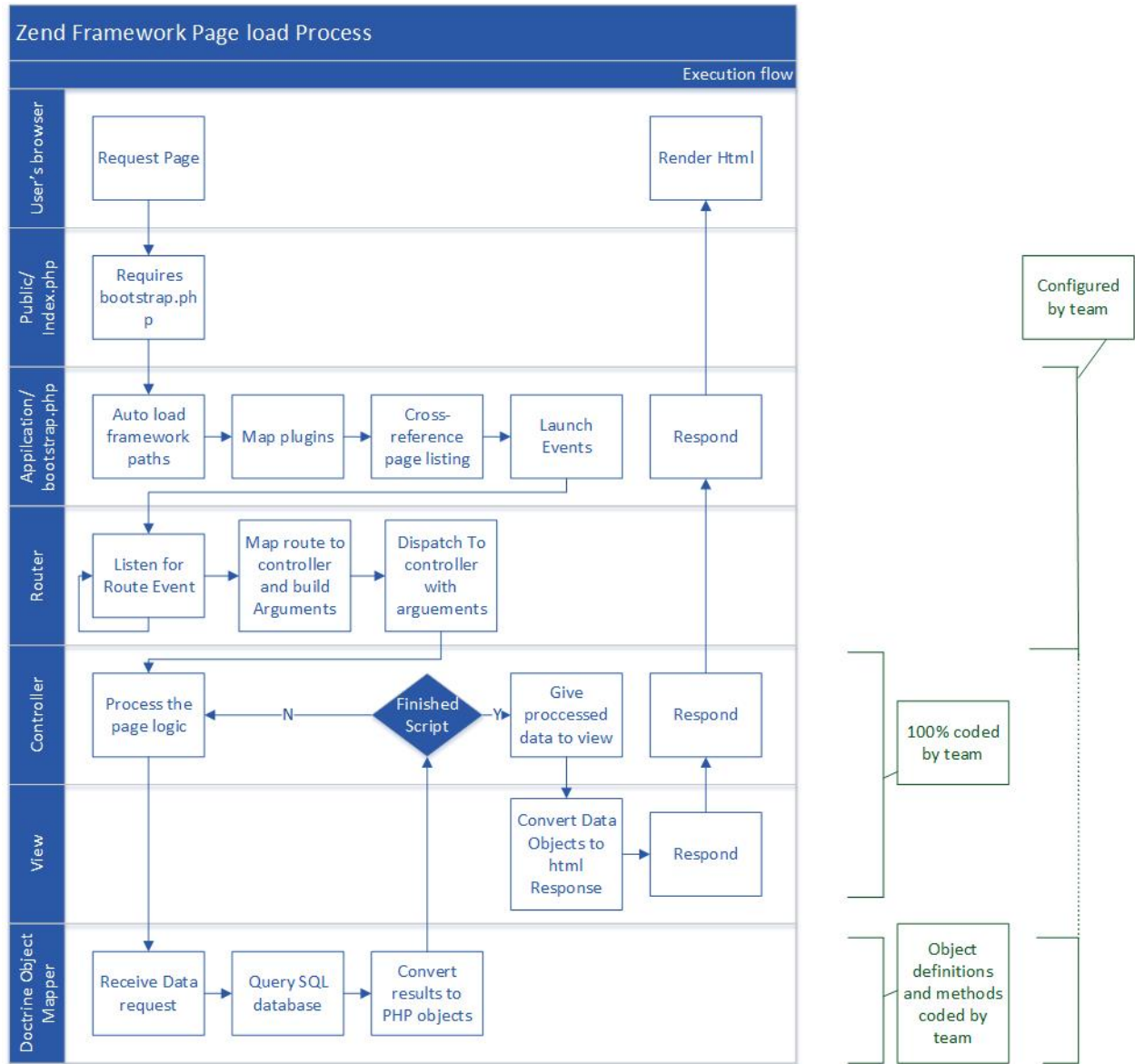
## Controllers - the main brains

        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



**Zend Framework Page load Process**

Execution flow

| Lane | |
|------|---|
| User's browser | Request Page → Render Html |
| Public/Index.php | Requires bootstrap.php |
| Application/bootstrap.php | Auto load framework paths → Map plugins → Cross-reference page listing → Launch Events; Respond |
| Router | Listen for Route Event → Map route to controller and build Arguments → Dispatch To controller with arguements |
| Controller | Process the page logic ← N — Finished Script — Y → Give proccessed data to view; Respond |
| View | Convert Data Objects to html Response → Respond |
| Doctrine Object Mapper | Receive Data request → Query SQL database → Convert results to PHP objects |

Configured by team

100% coded by team

Object definitions and methods coded by team

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.

| Header Bar - You are Instructor |
|---|

| Sidebar | Create a New Assignment |
|---|---|
| Home | |
| Assignments | **Previous Assignments** |
| Create New Assignment | Assignment Name |
| Student Grades | Enter Assignment Name |
| Grade Assignments | Begin Date: 3/14/15 · Due Date: 3/27/2015 · Last Day Availible: 3/30/2015 |
| sidebar | Assignment Description |
| sidebar | This is a description of an assignment. It is either really long or not long at all. |
| sidebar | text / text / text |
| Log Out | |

**Attach Files** — Upload

**Attach Unit Tests** — Upload

**Create Assignment**

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.

| | |
|---|---|
| Home | |
| Assignments | |
| Create New Assignment | |
| Student Grades | |
| Grade Assignments | |
| sidebar | |
| sidebar | |
| sidebar | |
| Log Out | |

# Create a New Class

( Previous Classes )

Class Name

| Enter Class Name |
|---|

Begin Date          End Date

| 1/06/15 | 📅 |    | 5/09/15 | 📅 |

Class Description

This is a description of a class. It is either really long or not long at all.

text

text

text

Class Sections

A (T 2-4)
B (W 12-2)
C (W 2-4)
D F (12-2)

( Create Class )

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.

# New User Sign Up

| First Name | Some Text | Select a Class | Classes ⬍ |
| Last Name | Some Text | Select a Section | Sections ⬍ |
| ISU Login | Some Text | | Submit |

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.

| Header Bar - You are Student |

| Home | Submit Attempt for Assignment [Name] |
| My Classes | Notes |
| My Assignments | text |
| My Grades | text |
| | text |
| sidebar | (Upload Assignment) |
| sidebar | ☐ I have provided input  (Upload input) |
| sidebar | |
| Log Out | Submit |

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

# Welcome to COGS!

**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

**Why is Cogs?**
dictum, praesent nascetur pulvinar sed, in dolor pede in aliquam, risus nec error quis pharetra. Eros metus quam augue suspendisse, metus rutrum risus erat in. In ultrices quo ut lectus, etiam vestibulum urna a est, pretium luctus euismod nisl, pellentesque turpis hac ridiculus massa. Venenatis a taciti dolor platea, curabitur lorem platea urna odio, convallis sit pellentesque lacus proin. Et ipsum velit diam nulla, fringilla

**Where is Cogs?**
vel tincidunt vitae, elit turpis tellus vivamus, dictum adipiscing convallis magna id. Viverra eu amet sit, dignissim tincidunt volutpat nulla tincidunt, feugiat est erat dui tempor, fusce tortor auctor vestibulum. Venenatis praesent risus orci, ante nam volutpat erat. Cursus non mollis interdum maecenas, consequat imperdiet penatibus enim, tristique luctus tellus eos accumsan, ridiculus erat laoreet nunc.

**Who is Cogs?**
vel tincidunt vitae, elit turpis tellus vivamus, dictum adipiscing convallis magna id. Viverra eu amet sit, dignissim tincidunt volutpat nulla tincidunt, feugiat est erat dui tempor, fusce tortor auctor vestibulum. Venenatis praesent risus orci, ante nam