

# **Project Plan: Version 1**

## **December 15-21**

**COGS (Central Online Grading System)**

## Table of Contents:

1. Team Overview
2. Problem Statement
3. System Description
4. System Breakdown
5. Deliverables
  - 5.1. First Semester
  - 5.2. Second Semester
6. Specifications
7. Schedule
  - 7.1. Gantt Chart
8. Resources
9. Testing
10. Risks

## Team Overview

Kathryn Widen - Computer Engineering

[kwiden@iastate.edu](mailto:kwiden@iastate.edu)

Daniel Riechers - Software Engineering

[riechers@iastate.edu](mailto:riechers@iastate.edu)

Daniel McDonough - Computer Engineering

[dmac@iastate.edu](mailto:dmac@iastate.edu)

Forrest Scott - Computer Engineering

[fvscott@iastate.edu](mailto:fvscott@iastate.edu)

Zachary Lones – Computer Engineering

[lones@iastate.edu](mailto:lones@iastate.edu)

-----

Good reference: [http://may1517.ece.iastate.edu/docs/content/lighthouse\\_project\\_plan\\_v3.pdf](http://may1517.ece.iastate.edu/docs/content/lighthouse_project_plan_v3.pdf)

-----

## Problem Statement

The main goal of this project is to create a web application that will drastically reduce the amount of time required for Professor Daniels and his TAs to grade code submitted by students in CPR E 185. Professor Daniels would like for students to be able to come up with their own ideas rather than assigning homework that will give specific outputs given specific inputs, it is therefore highly unlikely that the webapp will be an autograder in the sense that it will grade student's code.

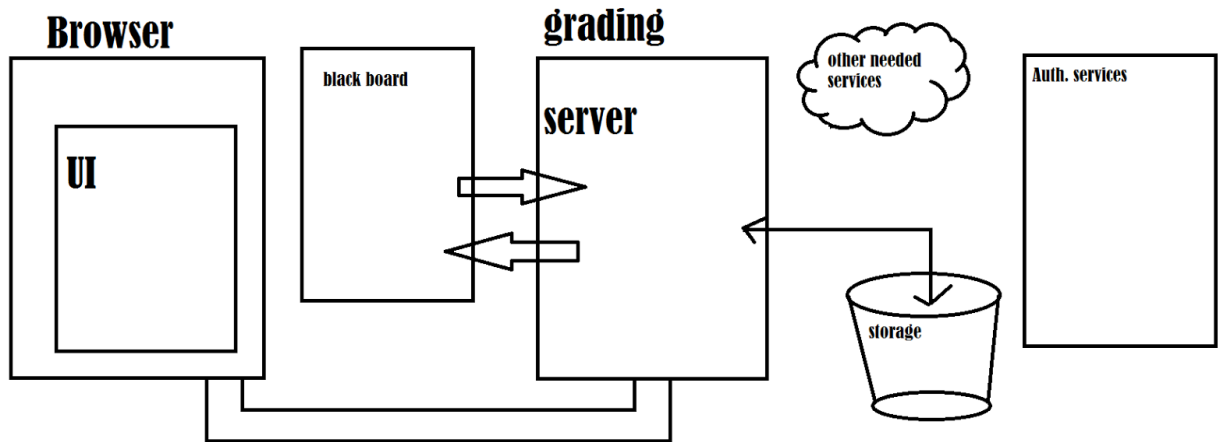
We wish to create an environment where TAs can be assigned groups of students and view the work from each student and grade them as well while minimizing the number of actions needed by the TA to complete the task. The grades submitted by the TA will then be sent to Blackboard for record keeping and so students can view their grade and any comments.

The webapp will adhere to all legalities related to safety of students' grades.

## System Description

COGS consists of several components:

- Back End: The backend system will be managing all of the data between instructors, students, and black board. The system will be capable of:
  - Interfacing with Blackboard to update student grades
  - Use Pubcookie to verify users upon login and classifying the users as admin, instructor/TA, or student.
  - Upload and store student submissions
  - Compile and auto-grade submissions
  - Store or immediately upload student grades to blackboard
- Front End: The system will be accessible through a webpage hosted on ISU servers. The front end will have three distinct parts:
  - Student Front End - Any user that is identified as a student will use COGS to upload assignments.
    - i) Submit/Resubmit assignments
    - ii) View feedback from auto-grader
  - Instructor Front End - The instructor front end will be viewable by those with instructor privileges. The instructor will be the heaviest users of COGS, and so the system should be set up to make their workflow the smoothest
    - i) Add/remove/edit assignments
      - (1) add/change/remove assignment due date
      - (2) mark assignment to be auto-graded or instructor graded
      - (3) set criteria/rubric for assignments
    - ii) view/grade new submissions that they are authorized to see
    - iii) view/grade old submissions that they are authorized to see
    - iv) View assignment submitter name and class section
  - Admin Front End - The admin front end will have all the same functionality as an instructor users with added privileges:
    - i) add/remove instructor or student users
    - ii) add/remove admin users
    - iii) manage individual instructor privileges



## System Breakdown

### Deliverables

#### System Deliverables:

- Iso containing all necessary COTS packages and dependencies
  - Install from iso automated with Kickstart
  - All system configuration automated with Kickstart
- COGS System:
  - Encapsulated environment for running student code
  - Compiler
    - Support for gcc
    - Support for clang
  - Mandatory Access Control Policy
    - Transparent to administrators
    - Tightly secured environment for running encapsulated code.

#### Software Deliverables:

- Software delivered in RPMs
  - RPM ensures correct permissions and configuration
  - SRPM keeps all build configuration in one place so it is easily repeatable
- Software should be added to system iso
  - Added to iso's yum repository as a
- COGS Software:
  - Analysis and scoring program
  - Web UI
  - Blackboard integration software

## Specifications

### User Specifications:

- Students must be able to access the system to submit code for current assignments.
- Students should not be able to submit code for previous assignments past due dates
- Professors and TA's must be able to easily access computed scores and code of students.
- Professors TA's must be able to manually change given scores
- Professors must be able add and modify due dates and assignments

System Specifications

- Cogs must store scores as securely as possible
- Cogs must make the best effort to not be exploited by students' code
- Cogs must automatically analyze submitted code, generate a score and provide any relevant feedback to students.

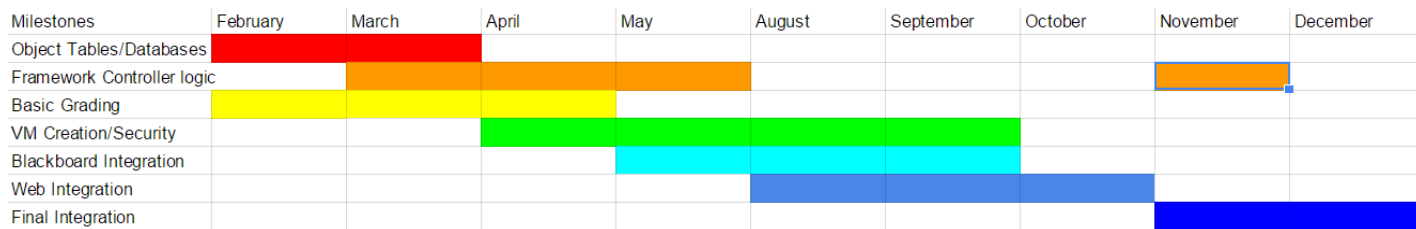
Schedule and Gantt Chart

PHP Framework

- ORM Mapping
- Views
- Controllers
- Blackboard API
- Accounts/Security

Grader

- Cheating Detector
- Scorer/Unit tests
- Command line interface



Resources Required

The system will require a virtual server hosted by ISU. This would allow the system to be scalable across multiple classes and sections.

Testing

Risks

Security Risks

Since the data our server will be processing could contain personal data protected by law, it is of the utmost importance to implement secure designs. While our goals reflect realistic and obtainable security levels, there is always risk involved with possible intrusion attempts on our server.

#### Hosting Risks

There are also risks involved with bureaucracy and possibly unknown rules related to hosting. The university may demand that the server is hosted within a virtual machine stack and tied to protected databases. It will of course be possible to accommodate any hosting requirements, but there are risks involved with establishing a long-term hosting solution

#### BlackBoard Risks

The designs suggest some level of integration with the BlackBoard services. There are risks and involved with obtaining permissions to interface with the BlackBoard system, as well as unknowns about the availability of features and usability with the BlackBoard API. Also, any time our system would write to the BlackBoard system, it is essential that great care is taken in order to prevent any false records or wrongful deletion of current records within the BlackBoard system.

#### Feasibility Risks

The goals of this project are extensive, which means careful planning with have to take place in order for different modules to come together, as well as guaranteeing that the highest priority features are completed first. By implementing the project in phases, the project can prove the feasibility of each added feature while simultaneously providing clear separation within the design layers. Lastly, our plan must reflect the possibility that if not all of the stretch goals are reached, the core functionality will still be provided.

#### Technology Integration Risks

Our project spans many different technologies from information security tools, to compilers, to web frameworks, to possibly chrome addons. There will be risks created at any time there are many different technologies and APIs being bonded together within one endeavor. these risks can stem from version incompatibilities, messaging standards, or just simply from misunderstanding the time needed to work out the complexities of a new technology. Planning out how each of the many pieces will fit together is crucial to minimize the chances of a new technology subverting the entire project.