COGS

(Central Online Grading System)

Dec 15-21

Problem Statement

- Open ended grading homework
- Blackboard is a frustrating interface
  - Doesn’t display .c or .txt files
  - Grading a single assignment is a tedious, multi-stepped process

Old System

TA was responsible for:

- Downloading files
- Compiling Code
- Running Executable
  - Providing Input/Output
- Additional steps to catch cheating
- Grading

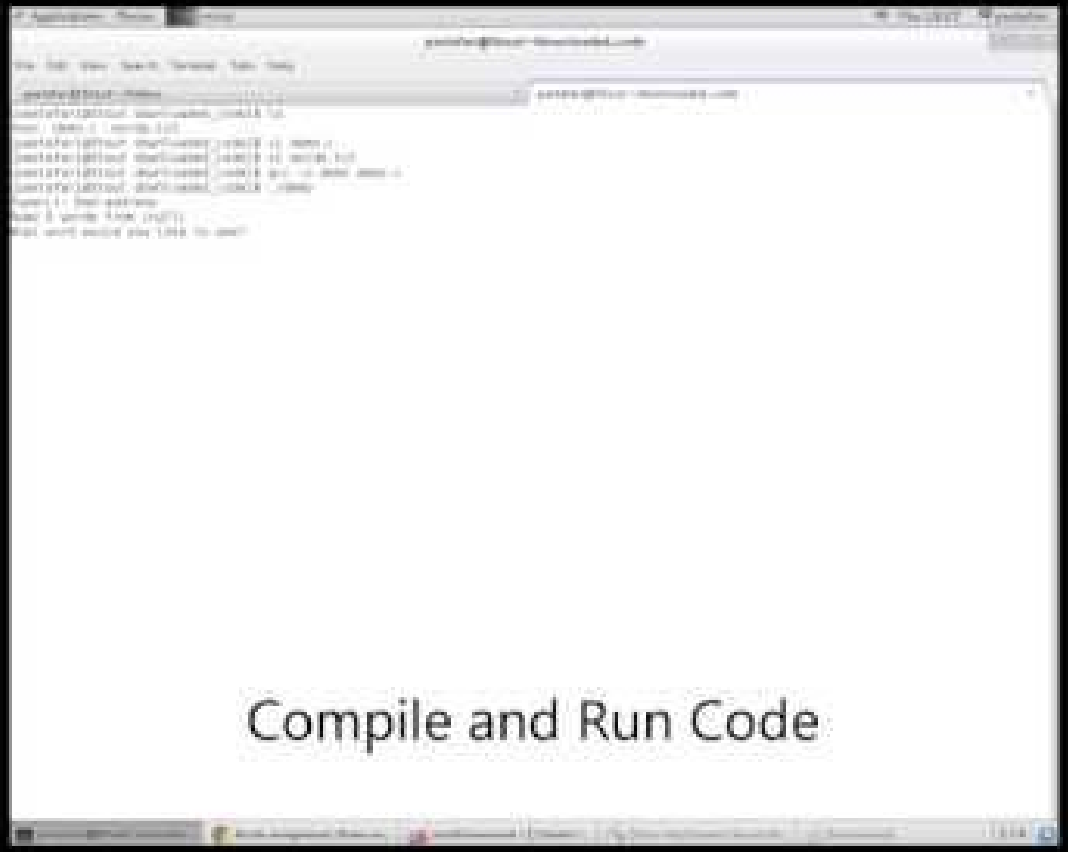
New System

COGS is responsible for:

- Hosting files
- Compiling Code
- Running Executable
  - Using student provided input/output
- Detecting Cheating

TA is responsible for:

- Grading


Compile and Run Code

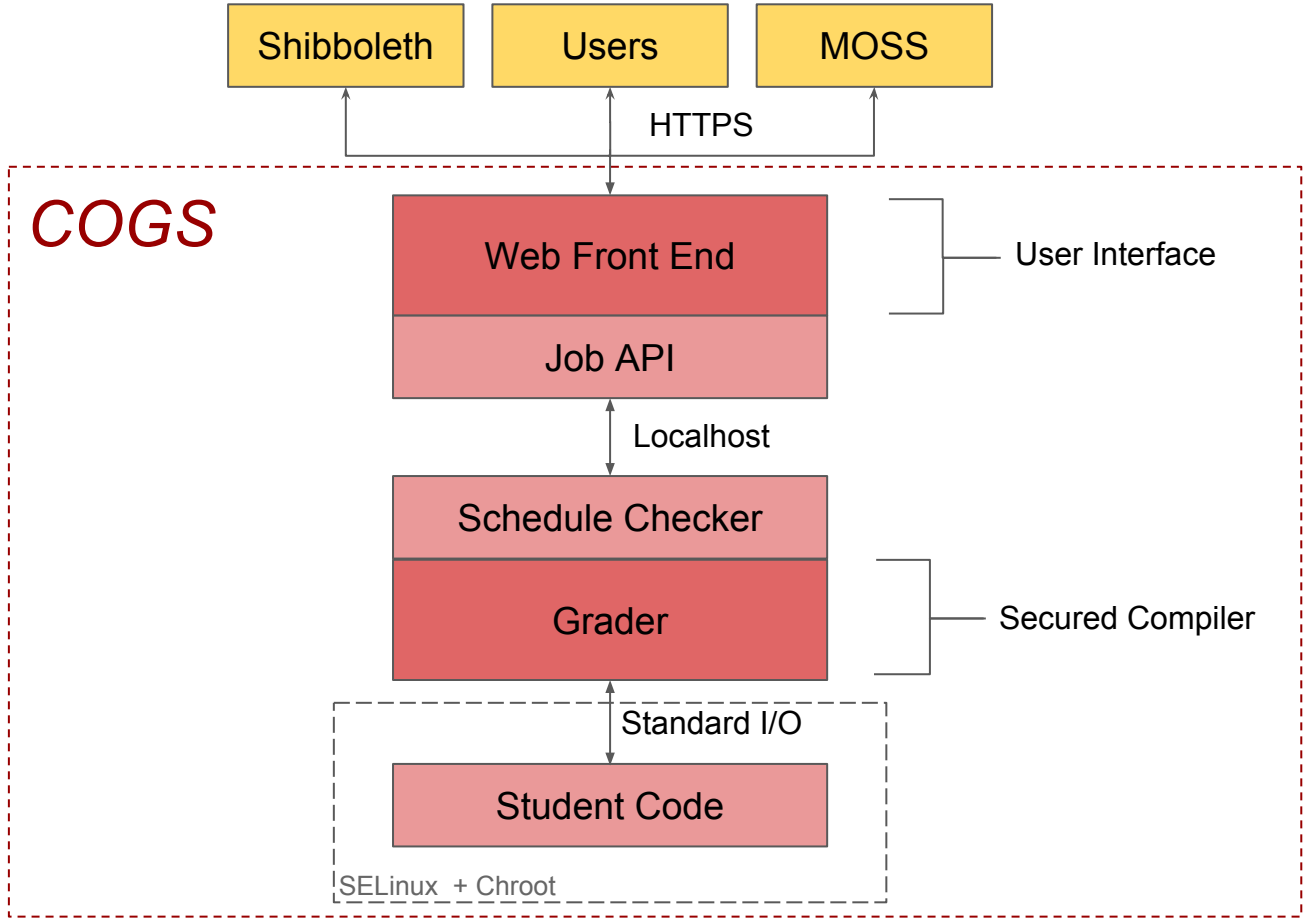
Source Code
Grading Form
Execution

Solution Overview

Goal: Streamline the Grading Process

- Main Features:
  - Easy student submission
  - Secure compiler
  - Automatic testing
  - Cheating detection
  - Streamlined grading

System Sketch



Web Front End

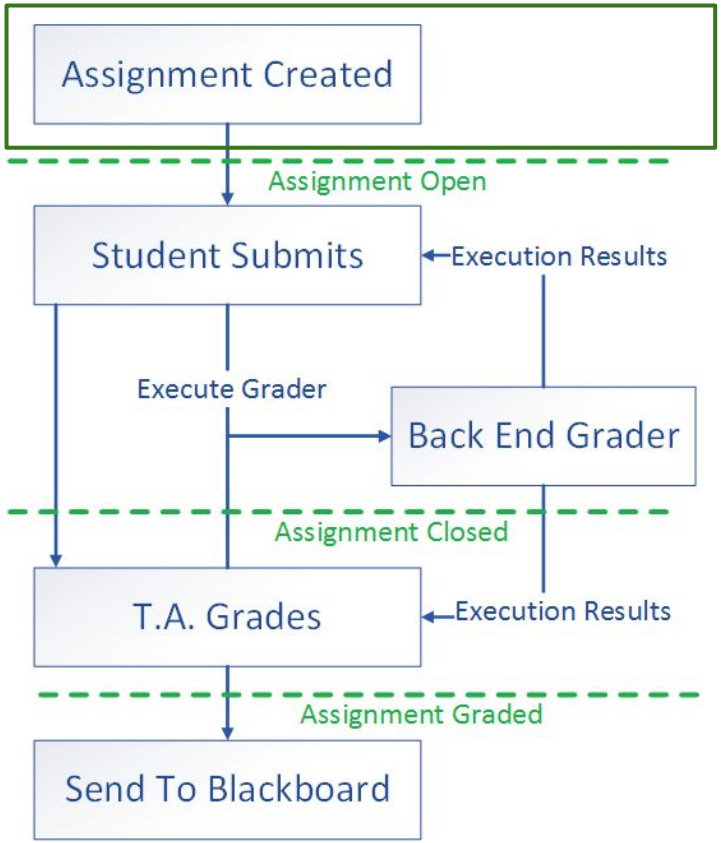
- Represents the User interface
- Used a PHP framework called “Zend Framework”
  - Modularized, secure, and popular
  - Same framework used by Iowa State

Technical Challenge - Integrate ISU Theme

- **Challenge:** Integrate Iowa State University web theme
- **Solution:** Built a multi-layer View Helper system that would intelligently render internal objects like Forms and Tables for the ISU theme.

Grading Timeline

- Creating Assignment
  - Professor only
  - Due dates
  - Grading rubric
  - Attachments



New Assignment

Select Course Course 1

Assignment Name Demo Assignment

Short Description Please use loops and strings

Start Date 12/09/2015

Due Date 12/24/2015

Assignment Close Date 12/24/2015

Assignment Publish ☑

makes the assignment visible to students

Checkbox List

Description Uses Strings **Point Value** 5 X

Description Uses Loop **Point Value** 5 X

Add Checkbox

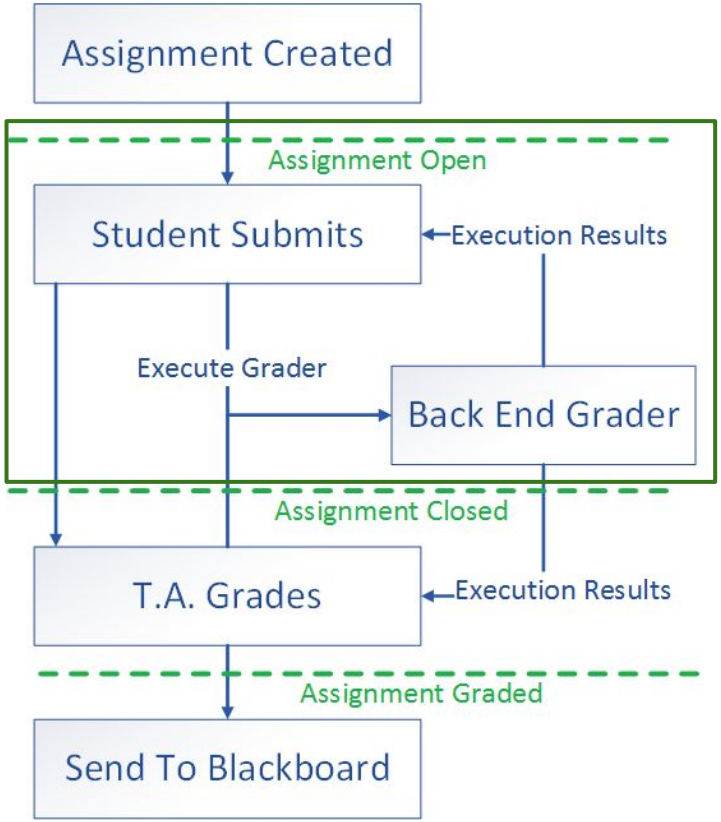
Numberbox List

Description Code Quality **Max Point Value** 5 X

Add Numberbox

Grading Timeline

- Student Submits
  - Custom execution inputs
  - Back End runs
  - Returns errors, outputs
  - Last submission is used



IOWA STATE UNIVERSITY

Central Online Grading System

Search

- Home
- Assignments
  - New Assigment
- Student Grades
- Assignment Grading
- Courses
- Administration
- Log Out

Assignment Submission

Demo Assignment

Submission Comments I hope you like my code!

Source files Choose Files demo.c

Input files Choose Files words.txt

Execution Inputs

Terminal Input -1 **Execution Arguments** words.txt X

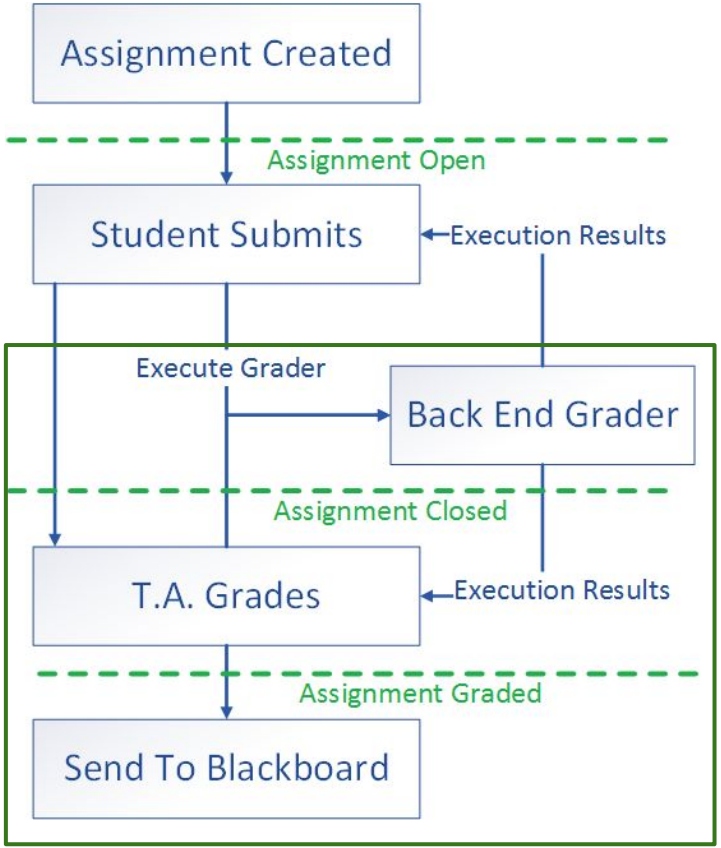
Terminal Input 5 **Execution Arguments** words.txt X

Add Execution Run

Submit

Grading Timeline

- T.A. Grading
  - Streamlined grading pages
  - Can create new execution
  - Notes viewable by students
  - Upload to Blackboard via .csv



Home
Assignments
Student Grades
Assignment Grading
Courses
Administration
Log Out

Submission View

Submission Details

- Assignment: Demo Assignment
- Course: Course 2
- Student: isuStudent2

Student's Comments

Hello! I hope you like my code!

Source Code

demo.c

```
        if (DEBUG) {
                printf("Read %d words from %s \n",word
Count, argv[1]);

                printf("What word would you like to se
e? \n");
                scanf("%d", &i);
                if(i>0 && i<wordCount) printf("%s \n",
 wordlist[i]);
```

Grading Form

Uses Strings	☑ 5 points
Uses Pointers	☐ 5 points
Uses Loops	☑ 5 points
Code Quality	5 / 5 points
Comments	1 / 5 points
Final Score	**16 / 25 points**
Scoring Comments	Please include more comments!

Submit

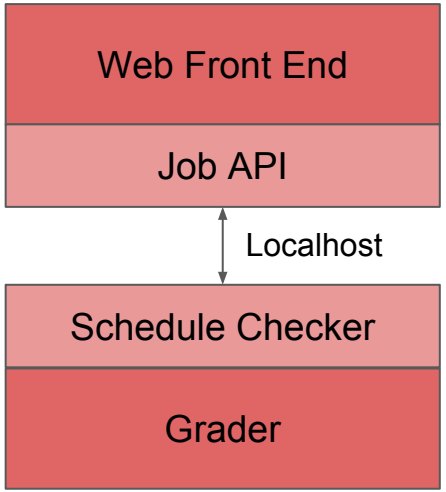
Cheating Detection

Other Features

- Create/Read/Edit/Delete
  - Classes
  - Class Sections
  - Assignments
  - Grades
- Assign TAs, Professors
- Blackboard Integration

Job API + Schedule Checker

- Interfaces between the Front End and Back End
- Back End polls the API periodically for new jobs
- Jobs are assignment submissions
- Handles scheduling and throttling.



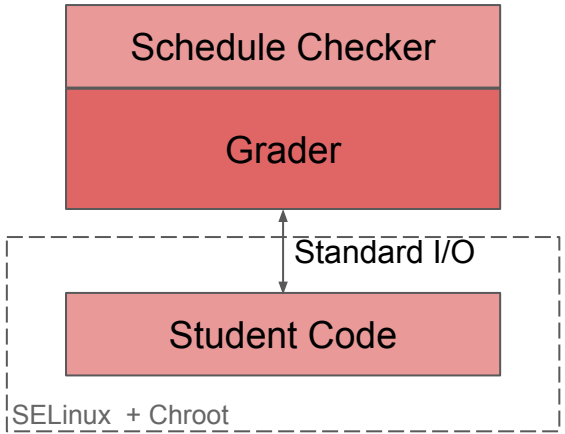
Grader

Grader: Automated system that compiles and runs student code and outputs a report.

Report: Contains compile errors, code input/output, contents of created files/directories, etc.

Runs the student code in a jailed environment.

Developed through test driven development.

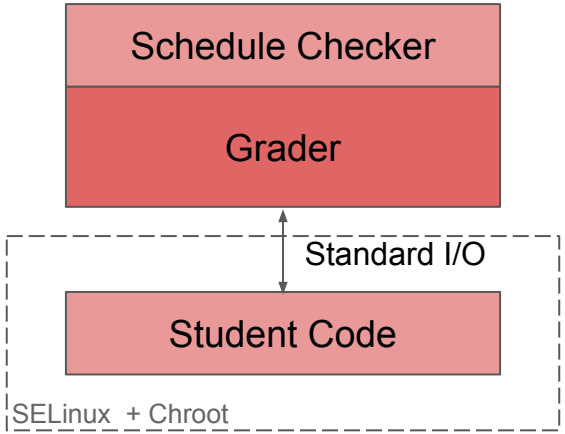


Attack Vectors

- Malicious code submissions
- Cross Site Scripting
- SQL Injection
- Buffer Overruns of grader
- User Masquerading
- Session Hijacking

System Security & SELinux Policy

- System Compartmentalization:
  - System has three distinct compartments with limited interaction between parts
- SELinux Policy:
  - Further restricts the permissions and capabilities of student code
  - Restricts system calls to only necessary ones
- Chroot Jail:
  - A temporary and restricted file system where student code is run to protect the system from the users while allowing the students to access a file system with their code.



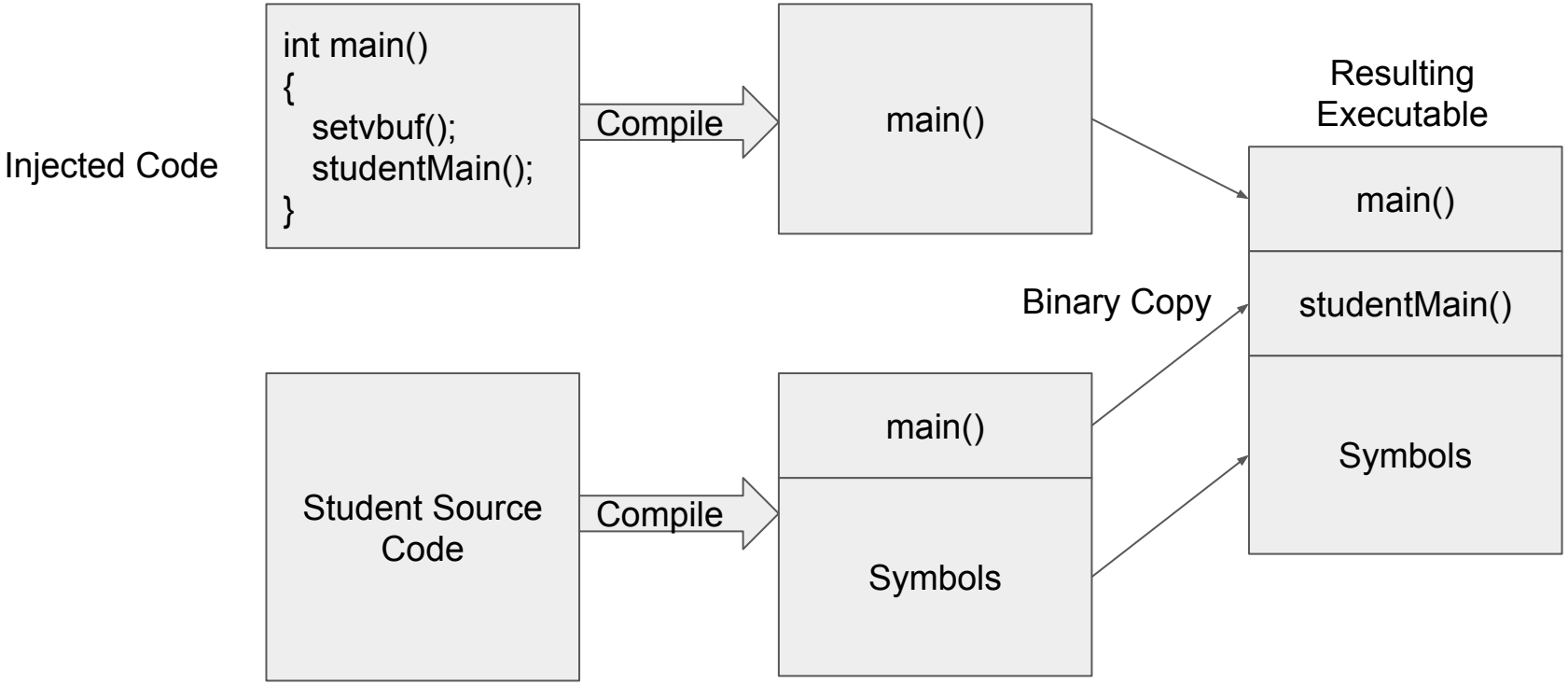
Front End Security

- Primarily best practices
  - Use CSRF tokens
  - Sanitize input to prevent SQL injection
  - Users can only access pages they need
  - Users are authenticated with Shibboleth
  - Focused on OWASP’s Top 10

Technical Challenge - Running Student Code

- Buffering on student code process needed to be turned off to have a “conversational style” of input and output
- Turning off buffering on the parent process and then forking the student code process did not fix the problem
- Solution: Compile time binary injection

The Solution



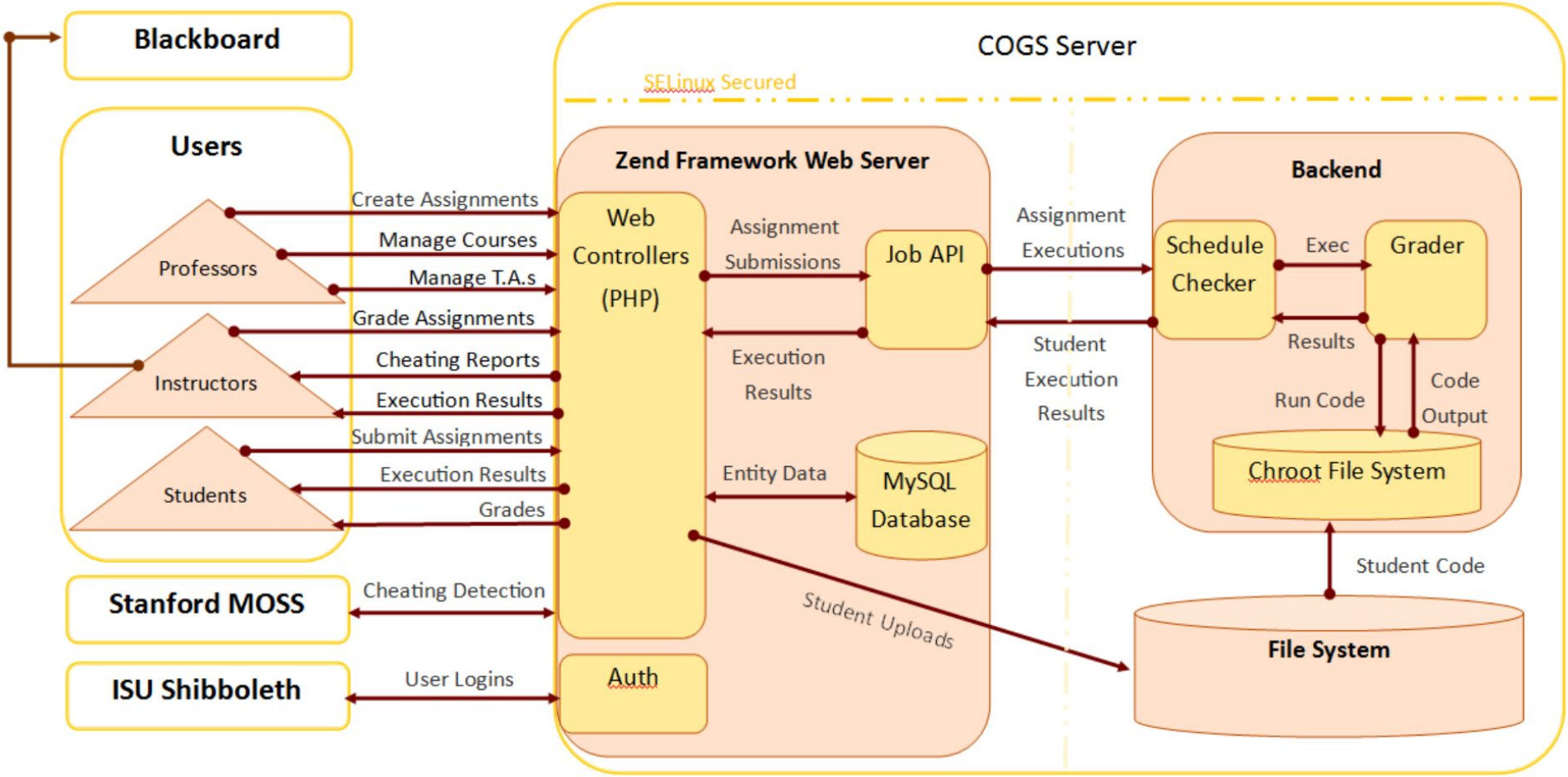
Testing

- Unit testing with Google Tests
- Manual tests
- Alpha testing sometime in the future

Looking Forward

- Continued support
  - Upgrades
  - Bug fixes
- Training TA’s
- Use by more professors

Questions?


Blackboard
COGS Server
SELinux Secured
Users
Professors
Instructors
Students
Create Assignments
Manage Courses
Manage T.A.s
Grade Assignments
Cheating Reports
Execution Results
Submit Assignments
Execution Results
Grades
Zend Framework Web Server
Web Controllers (PHP)
Assignment Submissions
Job API
Execution Results
Entity Data
MySQL Database
Auth
Assignment Executions
Student Execution Results
Backend
Schedule Checker
Exec
Grader
Results
Run Code
Code Output
Chroot File System
Student Code
File System
Student Uploads
Stanford MOSS
Cheating Detection
ISU Shibboleth
User Logins