Building a New Grading System for CS 2110: Software Development Methods

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Stephen Shiao

Spring, 2020.

Technical Project Team Members

Stephen Shiao

Kenneth Chen

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Dr. Nada Basit, Department of Computer Science

Abstract

User interfaces can make or break a system. Developers need to build interfaces that conform to user mental models, look modern, and function without error, otherwise users are pushed to find related products to accomplish their tasks. Web-CAT, an assignment submission system used for CS 2110: Software Development Methods, unfortunately frustrates instructors and students with its out-of-date interface and inherent errors. In this project, we aimed to build a similar system with an improved interface and additional functionality to benefit all users. Utilizing the Angular framework for front-end development, we implemented the necessary functionality included in Web-CAT as well as fixed problems that Web-CAT had. This new system decreased the number of clicks needed to accomplish a certain task to allow for more efficient assignment grading. It also integrates browser functionality and implements additional quality-of-life features for professors and graders. If a robust back-end component is implemented and integrated with this front-end, this submission system would be able to be used in any class for any programming language.

Introduction

The current online auto-grading system that the instructors for the class CS 2110: Software Development Methods use is known as Web-CAT, developed by Virginia Tech. Students submit code for their assignments to the system, which automatically runs that code on tests uploaded by professors to output a “correctness score” based on how many tests pass and fail. Instructors, including professors and teaching assistants, can view student submissions, view their files, and assign them grades and comments. Unfortunately, Web-CAT is sometimes frustrating to use from a User Interface/User Experience standpoint.

Problem Statement

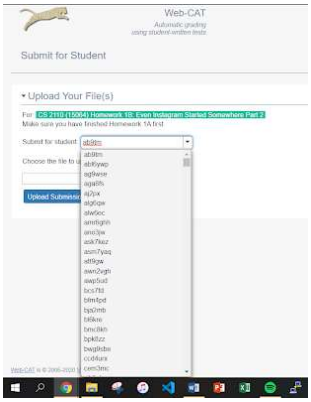
The main problem with this system is that, as an instructor grading an assignment, Web-CAT caused many problems when viewing student files. The main problem is that it relies on a form submission between the assignments page and the file page, so pressing the "Back" button on a web browser caused a form resubmission issue. This makes grading submissions significantly slower as instructors need to press the "Back" button again to reach the desired page, and sometimes the site even sends them back to the home page, causing even more frustration.

Another frustrating issue with Web-CAT is that their search function fails for most cases. When searching for a student to grade, the system searches for only first and last names, without trimming spaces from the search input. With UVA using unique computing IDs for each student, instructors are unable to search for students by computing ID, even though on the submissions page each student has their computing ID next to their name in parentheses (e.g. Stephen Shiao (ss2sc)). Further, simply adding a space at the end of a valid search, the system would no longer find any students.

Other issues we've noticed include the following:

- The system inserts visually-distracting highlights when viewing files for lines not tested. For grading purposes, these are unneeded and inconvenient to look at.

- The system forces graders to navigate to a separate page to submit for a student and asks the grader to select a student (out of over 500 students in the class) to submit for. This adds extra time and effort to graders who need to remember the student they want to submit for as well as searching to said student.



Our system aimed to fix these issues as well as provide additional quality of life features for all users. We wanted professors to be able to track grading progress for homeworks to keep graders accountable for their assigned work. Further, we wanted to streamline the creation of a grading distribution by programmatically assigning students to graders based on the number of hours a grader has.

Contributions

At the beginning, we wrote the user stories that we wanted to accomplish, outlined the functionalities for each page we needed to build, and built wireframes using Figma to show what the pages would look like. When building our backend structure, we also outlined the database structure to store the associated data for our web page.

Stephen

I served as the project manager, interacting with external contacts needed to obtain certain resources, like a web server, and updated our advisor during our biweekly meetings. I also served as project lead, organizing and connecting pages together and integrating Kenny's

pages with the project. Finally, I also built out some of the pages, like the Submit page, and dynamically generated pages per assignment for pages like Modify Assignment, Submit page, and Grade Assignment.

Kenny

My main role outside the high-level structuring was being the lead page designer and builder. I had lots of experience from internships building out web pages, so I applied that knowledge to building out the pages for this project and styling them using SCSS. Specifically, I built out the Create Assignment and Instructor Listing pages as well as the student and instructor home page views.

Related Work

The most popular grading system used to run and test student code is Web-CAT, developed at Virginia Polytechnic University (Edwards & Perez-Quinones, 2008). It was designed to push students to write more thorough tests for their own code and to grade submissions based on the code coverage of those tests. When students write more tests for their own code, they learn much more than if the system tests the code for them (Edwards & Perez-Quinones, 2008). Further, the system accepts many different programming languages and allows for instructors to assess their students' code in any method they wish (Edwards & Perez-Quinones, 2008). For these reasons, it won the 2006 Premier Award for its high-quality, non-commercial courseware (Edwards & Perez-Quinones, 2008) and is used by more than 1,300 courses across 39 universities as of March, 2019 (Tech). It supports 8 programming languages and has processed over 3.5 million student submissions (Tech).

The professors of CS 2110: Software Development Methods currently use Web-CAT for their automated assignment grading and it has done its job for the most part. However, as described above, the front-end interface has aged poorly since it was built before 2006 and has caused much frustration for graders. The system we have designed is intended to take the functionality and dependability of Web-CAT and add many quality-of-life improvements as well as give it a modern user interface.

Other related systems are sites that help programmers hone their skills, including LeetCode.com, HackerRank.com, CodingBat.com, and Codecademy.com. These sites allow users to choose a problem to solve about a wide range of programming topics, from basic data structures to recursion and complex algorithms. Users can then write code in their language of choice to solve said problem, compile and/or run it, and see their results. Since many of the problems these sites have are common questions asked in coding interviews, college students seeking jobs in computer science often use these sites to practice and hone their skills to prepare.

These systems show some promise for use as an assignment submission system. In some cases, such as CodingBat, professors can have students work on specific questions and view their progress. Some sites even allow for file upload and submission. However, they often can't handle much more complex homework assignments requiring multiple files. The systems also don't evaluate the code quality, which is an important aspect of grading for CS 2110. Further, topics such as Graphical User Interfaces (GUIs) and Concurrency are difficult, if not impossible, for an automated system to grade on its own.

System Design

The system was intended to provide a seamless experience to students and instructors for submitting and grading assignments. Students should be able to view current and previous

assignments as well as upload and submit files for assignments and receive feedback on how their code performed. Instructors should be able to grade student submissions quickly and provide comments to students about where points were taken off. Professors, specifically, should be able to create assignments and input a rubric for each assignment.

Angular was the chosen web framework as it provided an easy and efficient way to create and build new pages. It would allow development of a single page application, which would make the user experience both fluid and convenient to use. Angular has proven to be a robust Model-View-Controller framework through its usage by products like Youtube. This framework is under the MIT license.

System Requirements

Gathering system requirements is an important aspect of any team as it allows all involved to see the overall and specific objectives of a project and work toward clear objectives. Under the Agile methodology, system requirements come in the form of user stories (Ibanez, 2018). These are often written as detailed descriptions about what the users should be able to accomplish when interacting with a system.

For this project, the requirements were listed on a Google Spreadsheet. To clarify terminology, “instructors” refers to both teaching assistants and professors.

Minimum requirements:

- All users should be able to login to the system
- All users should be able to view current and previous assignments
- Students should be able to submit code for assignments and receive feedback on
- Professors should be able to create and modify assignments

- Instructors able to view and grade student submissions
- A server should be set up to store a database and files

Desired requirements:

- Professors should be able to build out their own rubrics and deductions for each assignment
- Professors should be able to view, add, and modify current instructors (both professors and teaching assistants)
- Professors, teaching assistants, and students should all have different home pages
- Instructors should be able to submit files for a student on the same page that they grade that student.
- Professors should be able to export assignment grades and upload it directly to UVA's Collab system
- Student code should be displayed for instructors to view when grading

Optional Requirements:

- Professors should be able to track the grading progress on homeworks
- The system should generate a grading distribution based on instructor grading hours
- Users should be able log in using the UVA Netbadge system
- Grading instructors should see the students they've been assigned to first, but also be able to view all student submissions

Wireframes

Wireframes are a simplified outline of the product (Cao, 2020). It allows developers to create and visualize how they want to organize their content. This allows flexibility in finding

errors making changes early on, without having to prototype entire pages. It also allows developers to ensure that all of their requirements were fulfilled early on, and add more requirements that they may have been missing. Wireframes are also easily built, allowing for multiple iterations to create the best user experience. The following are the wireframes we drafted, although they are a bit more detailed than traditional wireframes. These helped us have a clearer picture of the pages that we built later on.

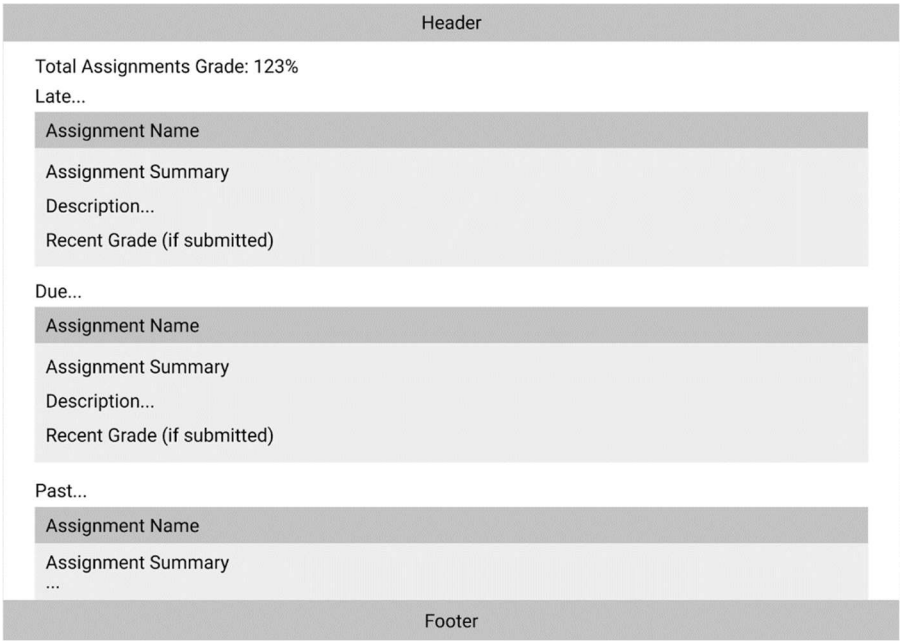


Figure 1: Student Home: Students should be able see an overall summary and of their individual assignments. Having it ordered “Late”, “Due”, “Past” ensures that the most relevant assignments are shown first.

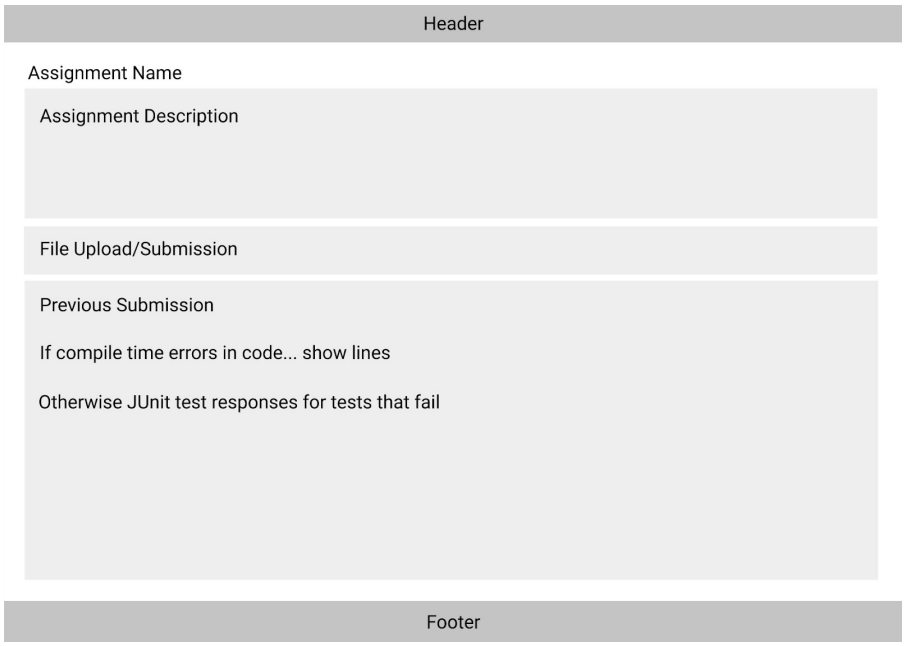


Figure 2: Student Assignment Submission

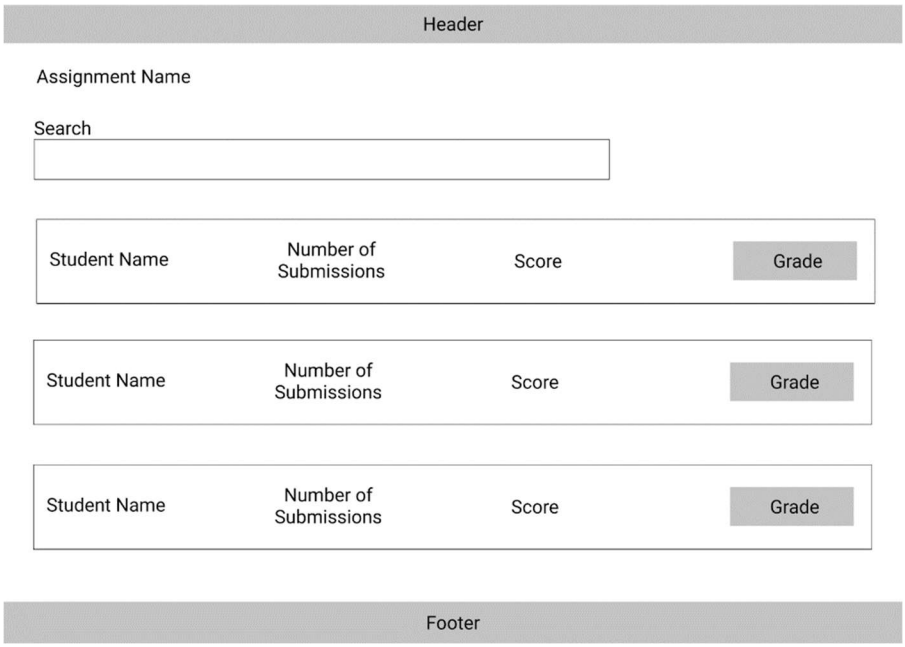


Figure 3: Grading Submission List

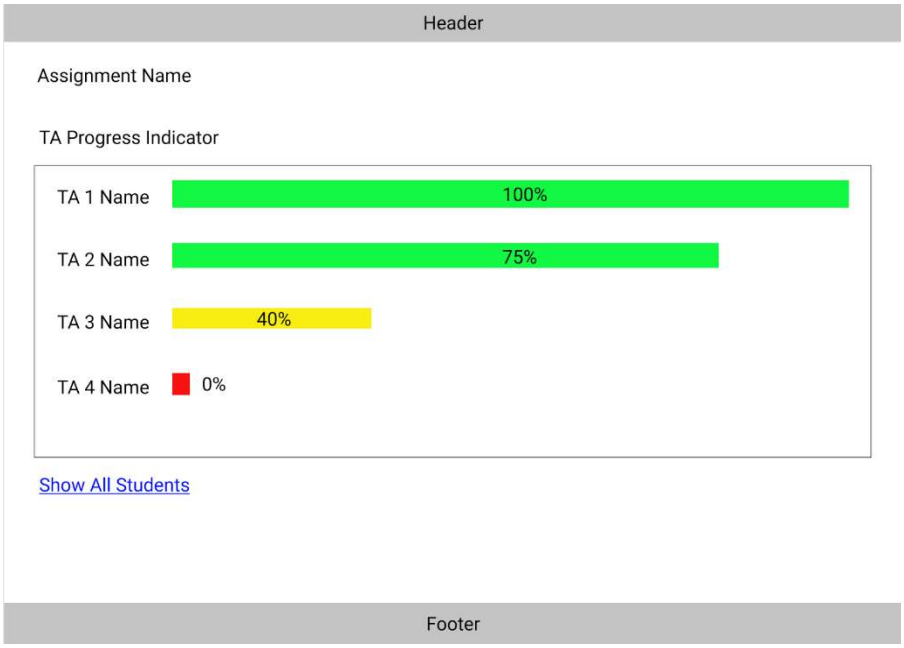


Figure 4: Assignment Grading Progress: Additional feature in our system not present in WebCat. The goal here was to quickly see TAs who weren't quite done yet so it would be easy to shoot them reminders.

Instructors

Upload Spreadsheet (.csv, .xls, ...)

No file uploaded | Upload File

Add Single Instructor:

Name	Computing ID	Grading Hours	Permissions	Add

Instructor List

Search:

Name | Computing ID

Filter by:

Permissions

Name ▲	Computing ID ▲	Grading Hours ▲	Permissions ▲	
TA1	ta1po	3	TA	Remove/Edit
Professor1 (You)	pr1of		Professor	
Professor2	es2so		Professor	Remove/Edit
TA2	ta2oo	2	TA	Remove/Edit
Professor3	rpr3of		Professor	Remove/Edit
TA3	ta3fd	2	TA	Remove/Edit
Professor4	ess4or		Professor	Remove/Edit
TA4	ta4ll	1	TA	Remove/Edit
TA5	ta5ty	2	TA	Remove/Edit
TA6	ta6kn	3	TA	Remove/Edit
...	...		...	...

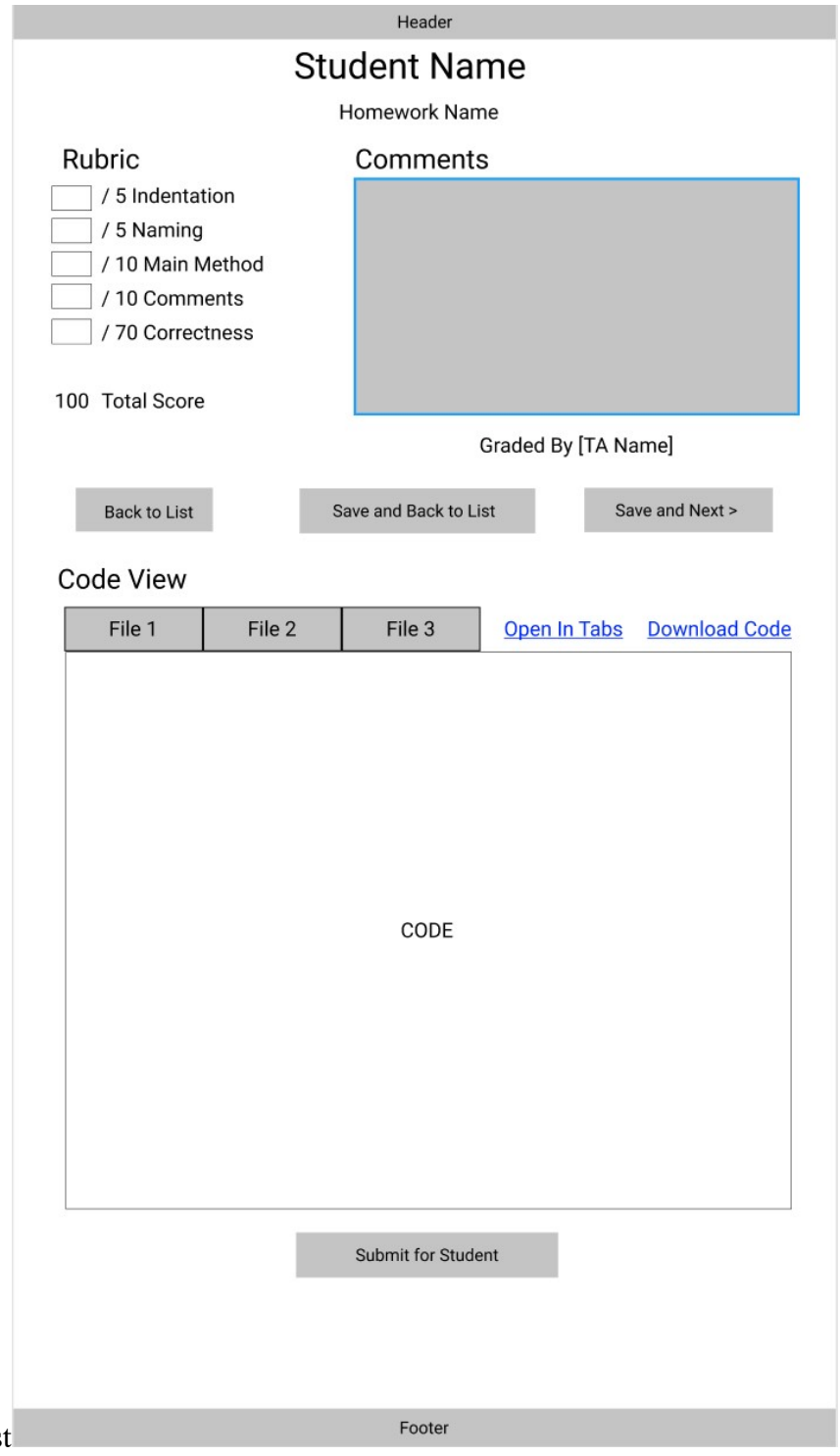


Figure 5: Instructor List

Figure 6: Grading Submission: The key feature added on this page was the Code View. This allows for graders to view code easily on the same page, instead of opening a new window to view code.

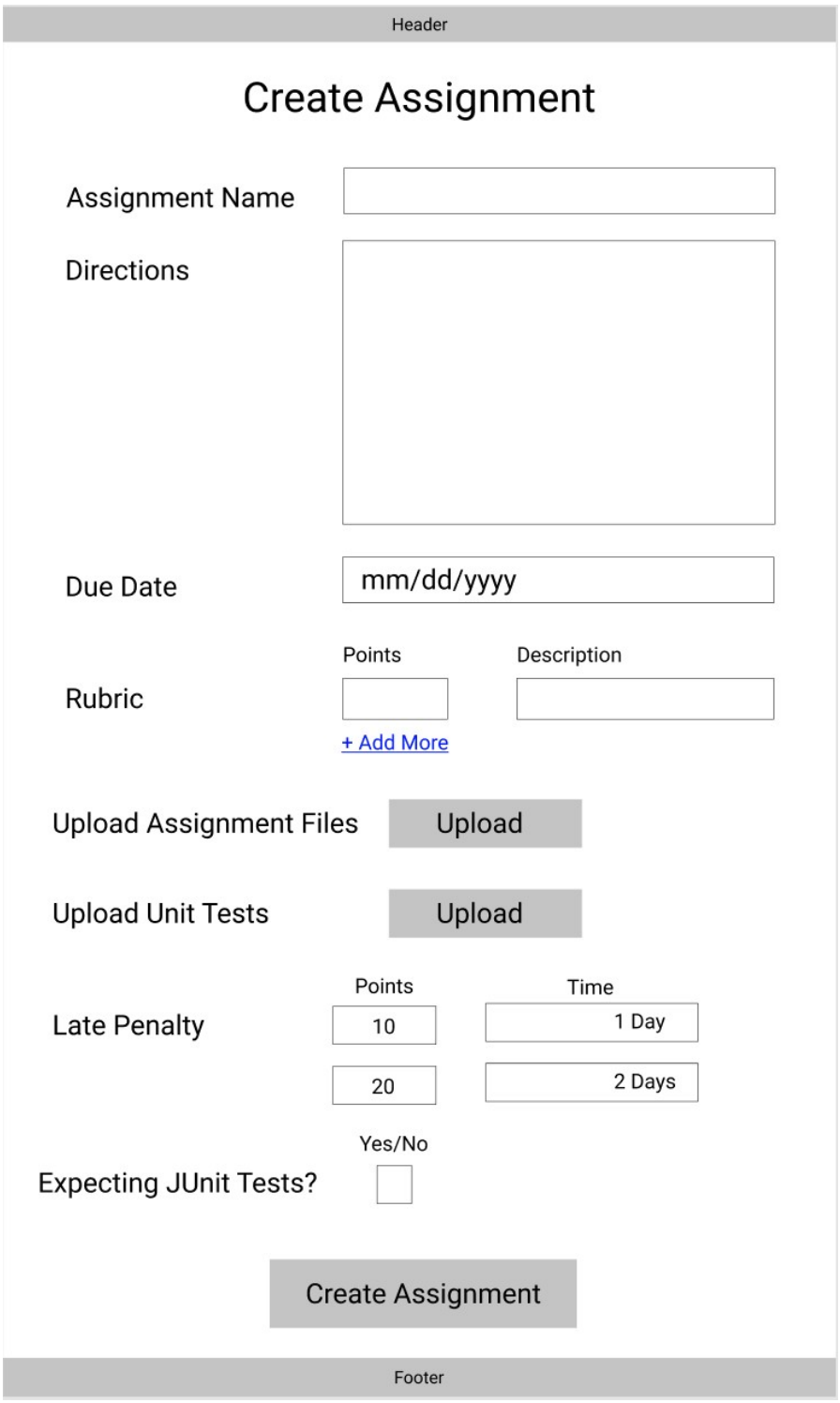


Figure 7: Create Assignment: A key feature added on this page is the rubric section. It allows for flexible rubrics for an assignment, so users don't have to break up sections on their own under comments.

Sample Code

```ts
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { LoginComponent } from './login/login.component';
import { HomeComponent } from './home/home.component';
import { CreateAssignmentComponent } from './create-assignment/create-assignment.component';
import { ModifyAssignmentComponent } from './modify-assignment/modify-assignment.component';
import { AddInstructorsComponent } from './add-instructors/add-instructors.component';
import { SubmitAssignmentComponent } from './submit-assignment/submit-assignment.component';


const routes: Routes = [
  {path: '', component: LoginComponent},
  {path: 'home', component: HomeComponent},
  {path: 'instructors', component: AddInstructorsComponent},
  {path: 'createAssignment', component: CreateAssignmentComponent},
  {path: 'modifyAssignment/:id', component: ModifyAssignmentComponent},
  {path: 'submitAssignment/:id', component: SubmitAssignmentComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

Figure 8: app-routing.module.ts: The general routing of our web application. This screenshot does not include all routes - those can be seen on the GitHub repository.

```scss
.Home {
    margin: 32px;

    &-title,
    &-grade {
        text-align: center;
    }

    &-subtitle {
        font-size: 36px;
        text-decoration: underline;
    }

    &-assignments {
        margin: 16px;
    }

    &-courses {
        margin: 0 16px;

        &-title {
            margin: 0;
            display: flex;
            justify-content: space-between;
            background-color: rgb(180, 180, 180);
            padding: 8px 16px;
        }

        &-name {
            font-weight: bold;
            padding: 0.375rem 0.75rem;
        }
    }
}
```

Figure 9: home.component.scss: Code to style the home page.

```
<div class="Home">
    <!-- Student Experience -->
    <div *ngIf="this.globals.user.role == 'student'">
        <h1 class="Home-title">Welcome {{ name }} to [Course]</h1>
        <h3 class="Home-grade">Your Current Homework Grade is: [Grade]</h3>
        <div class="Home-openAssignments">
            <p class="Home-subtitle">Open Assignments</p>
            <assignment-teaser *ngFor="let assignment of currentAssignments(courses[1].assignments).slice().reverse()"
                [assignment]="assignment">
            </assignment-teaser>
        </div>
        <div class="Home-pastAssignments">
            <p class="Home-subtitle">Past Assignments</p>
            <assignment-teaser *ngFor="let assignment of pastAssignments(courses[1].assignments).slice().reverse()"
                [assignment]="assignment">
            </assignment-teaser>
        </div>
    </div>

    <!-- TA Experience -->
    <div *ngIf="this.globals.user.role == 'teaching assistant'">
        <h1 class="Home-title">Welcome {{ name }}</h1>
        <div class="Home-courses" *ngFor="let course of courses">
            <a class="Home-courses-title">{{ course.courseTitle }} {{ course.semester }}</a>
            <assignment-teaser *ngFor="let assignment of pastAssignments(course.assignments).slice().reverse()"
                [assignment]="assignment">
            </assignment-teaser>
        </div>
    </div>

    <!-- Professor Experience -->
    <div *ngIf="this.globals.user.role == 'professor'">
        <h1 class="Home-title">Welcome {{ name }}</h1>
        <div class="Home-courses" *ngFor="let course of courses.slice().reverse()">
            <div class="Home-courses-title">
                <a class="Home-courses-name">{{ course.courseTitle }} {{ course.semester }}</a>
                <div>
                    <a class="btn btn-primary" href="/createAssignment">Create New Assignment</a>
                     
                    <a class="btn btn-primary" href="/instructors">Instructor List</a>
                </div>
            </div>
            <assignment-teaser *ngFor="let assignment of pastAssignments(course.assignments).slice().reverse()"
                [assignment]="assignment">
            </assignment-teaser>
        </div>
    </div>
</div>
```

Figure 10: home.component.html: The basic structure of the home page. Lots of if statements in order to split the user experience by user type (student, TA, or professor)

```typescript
import { Component, OnInit } from '@angular/core';
import { GlobalsService } from '../common/services/globals.service';
import DummyData from '../../assets/dummydata.json';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.scss']
})
export class HomeComponent implements OnInit {

  name: string = this.globals.user.firstName + this.globals.user.lastName;
  courses: any[] = DummyData.courses;

  constructor(private globals: GlobalsService) { }

  ngOnInit() {
  }

  pastAssignments(assignments: any[]) {
    var a = [];
    var now = new Date();
    for(var i = 0; i < assignments.length; i++) {
      var due = new Date(assignments[i].dueDate);
      if(now > due) {
        a.push(assignments[i]);
      }
    }
    return a;
  }

  currentAssignments(assignments: any[]) {
    var a = [];
    var now = new Date();
    for(var i = 0; i < assignments.length; i++) {
      var due = new Date(assignments[i].dueDate);
      if(now <= due) {
        a.push(assignments[i]);
      }
    }
    return a;
  }
}

```

Figure 11: home.component.ts: The scripts run on our home page. This code grabs assignments that were already done, and assignments that are due in the future but not done yet.

Installation Instructions

1. In the web browser of your choosing, navigate to the webpage

   https://github.com/ss2sc/CS2110HWSubmission

2. Using Terminal (Mac) or Windows PowerShell (Windows), navigate to your desired folder (suggested folder: Documents/Github) using the cd command.
3. Type the command “git clone git@github.com:ss2sc/CS2110HWSubmission.git”. This will download the code from the repository into the file you navigated to.
4. Navigate into the new folder with “cd CS2110HWSubmission”.
5. Install dependencies with “npm install”
6. Install Angular Materials with “ng add @angular/material”. When prompted to install packages, type “yes”. When asked to select a theme, select the first one that says “Indigo/Pink”
7. To run the application, type “ng serve --open”. The page should open in your default web browser.

Results

The system solves some of the problems that Web-CAT had with its user interface, providing a more fluid user experience, particularly for graders.

- Search function: Utilizes one of Angular’s libraries, Angular material, providing a more robust way to search for student names. It allows for strings including spaces, and trims off any whitespace on the ends. It also searches all relevant user fields - first name, last name, and username. This is a large improvement over the single word search that Web-CAT implemented.
- Viewing submission code: Reduced number of clicks needed to view each file. The system provides a way to see every file’s code right on the submission page, instead of having to go to a new page to view a single file, and having to go back to the submission page by resubmitting form data.

- Grading progress indicator: Provides a way to see the progress of each grader, to be able to efficiently remind graders to finish grading, and releasing grades to students in a timely manner.
- Browser capabilities: Web-CAT required lots of form resubmission, rendering the back and forward buttons on browsers useless. This new system allows for usage of the back and forward buttons without running into page errors.
- Retained most front-end capabilities of Web-CAT: Much of the front-end of Web-CAT is necessary, being able to view assignments, submit assignments, etc. This system retains all of those necessary functionalities.
- More efficient grading: By providing a more efficient way to view code, as well as a more organized manner of showing submissions to graders (assigned students at the top), this system allows graders to work more quickly.
- Custom rubric: This system allows for each assignment to have a custom rubric, allowing graders to put less comments in order to provide more detailed feedback on Web-CAT's fixed two-category rubric.

Conclusion

Modernizing the user interface of websites is important in order to minimize frustration from using a website and provide an efficient way to submit, grade, and return assignments. Additionally, having a systematic design process in building a front-end makes building the website a smooth task, and ensures that all features are incorporated in a well-thought-out manner.

We faced many challenges in building this system. Primarily, building an entirely new product similar to an old one is difficult, as we had no access to the backend of Web-CAT. Thus,

we needed to be able to integrate this system with some external department, which impeded our progress severely. Due to COVID-19, the department had to assist the university with transitioning courses online for the Spring 2020 semester, at the same time as we were working on this project. Overall, many factors can slow down the progress of a project, forcing adaptations in order to accomplish some goal. For this project, it meant only being able to develop a front-end interface, instead of the whole product, and creating mock back-ends in order to fully develop a front-end.

If this system were to be completed, it would be robust enough to handle assignment submission of multiple classes. The front-end is built to show different classes, as well as any files, so it would be able to be used for any class, not just the original intent of CS 2110: Software Development Methods.

Future Work

As stated previously, the system currently has no backend server to store data and submission files. Future work needs to be done to set up the server to allow for database creation, file storage, and the ability to compile and run Java files and JUnit tests. Further, such a backend would require an interface to send and receive data from the front-end webpage. Integration work would need to be done to ensure that the data transmitted is in a format that both parties can process. The front-end should be able to take lists generated by SQL queries to the back-end database and display it on the existing template. A web server would also be needed to ensure that the system can be deployed for the public to use.

Currently, the system was only intended to support compiling and running Java files. Work can be done to extend that to support other programming languages such as Python and

C/C++. This way the system isn't only usable by a single class; many other intro- or higher-level computer science classes can also utilize its functionality.

The system can also be expanded to allow students and instructors from other schools to access it by integrating with security systems that other universities utilize.

If the system could support all the above additions, it could possibly rival Web-CAT in its popularity and even take users away from using Web-CAT as it provides a similar service, but with a much more intuitive and modern user interface.

References

Angular. (n.d.). Retrieved April 24, 2020, from https://angular.io/

Cao, J. (2016, January 22). What is a Wireframe: Designing Your UX Backbone. *Studio by UXPin*. https://www.uxpin.com/studio/ui-design/what-is-a-wireframe-designing-your-ux-backbone/

Edwards, S. H., & Perez-Quinones, M. A. (2008). Web-CAT: Automatically grading programming assignments. *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, 328. https://doi.org/10.1145/1384271.1384371

Ibanez, L. (2018, February 24). *Agile Development: User stories are the new requirements document*. Medium. https://medium.com/theagilemanager/agile-development-user-stories-are-the-new-requirements-document-c105947c9291

Tech, V. (n.d.). What is Web-CAT? Retrieved from http://web-cat.org/projects/