TeachEzie

Design Document

10/22/2018

Version <1.0>

TeamRSS

Jackson Schuur

Sophia Schuur

Kathy Freund

Austin Ryf

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

I. Introduction

The purpose of this design document is to outline the progress our team has made while working through Iteration 1.

Currently, our project meets all specifications outlined in the Term Project Description for Iteration 1. We've successfully build a working frontend and backend to initiate Post requests. Both an instructor and student can create a new account. We also set up the GUI and web design from scratch in HTML and CSS. Our end project goal is to complete all the requirements asked of us between now and the due date for Iteration 3. We plan to add many features, including the ability of the user to login, and for a user to view and edit their profile. We plan to make it possible for Instructors to add new courses for which they require a Student to teach, and Students to view and apply to said TA positions. Instructors will be able to view and approve of applications, and Students will be able to see the status of their applications or cancel them.

Section II includes a description of our chosen architectural model and a corresponding UML component diagram. We detail what functions each section performs and why it is important as a whole. For this Iteration, our UML component diagram is small, and will be expanded as we progress through the iterations.

Section III includes detailed outlines of all major subsystems. It discusses each component's role in the project. It goes over the routes we implemented in the Backend and some routes we will implement in the future. It identifies the communication between the client and server, what kind of data those messages contain, and in what format they are sent.

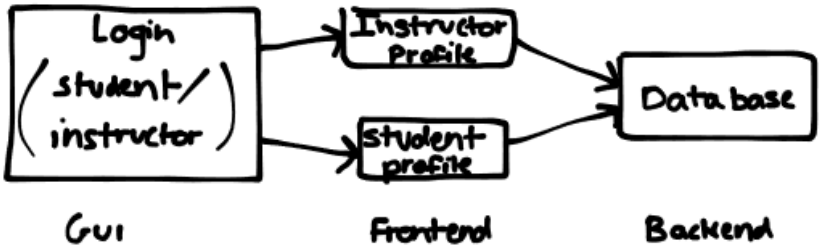
Document Revision History

Rev 1.0 <10/22/2018> <Iteration 1 Design Document>

II. Architecture Design

II.1. Overview

For this project, we utilized the Client-Server Architectural Pattern. Similar to the SmileApp Project we completed earlier in the semester, we separated our project into three main components: The GUI, written in HTML and CSS, the Frontend, written in JavaScript, and the Backend, written in Python. The GUI is responsible for displaying information in an appealing manner, the Frontend processes user input data (such as creating a new Instructor or Student profile), and the Backend centralizes all data management and ensures data integrity and database consistency. This pattern fits well for a system such as ours where different types of users need to access specific, sensitive data. We also needed a central database of all users and their information, and a dedicated server suited this task best. By utilizing this pattern, we neatly kept the dependencies of the main components at a minimum by assigning each one a separate, but equally important task.



III. Design Details

III.1. Subsystem Design

We decoupled each major component of our project using certain aspects of the MVC architectural pattern as well as basic mannerisms to good software design.

III.1.1: For the GUI, we organized our information neatly into an HTML file and its corresponding CSS file. It has been done completely from scratch. All menus and formats are written on the same HTML file and are hidden and shown in the JavaScript accordingly as a user selects options from the GUI. It contains no hyperlinks to any other part of the project.

III.1.2: For the Frontend, we made sure that nothing in the JavaScript code ever sees anything in the Backend. That is to say, the Frontend does not read from or write to the database in any way. The Frontend simply receives user information and places it in the correct URL and performs the proper requests.

III.1.3: The Backend handles all database design. Each user, either a Student or Instructor, has its own database class and respective attributes. This data holds the profile information of either a Student or Instructor. There exists one database entry per one profile.

For this iteration, the backend currently implements model database entries for a Student and Instructor profile and POST routes to create new profiles of each. Messages between the client and server consist of these POST routes and contain JSON formatted profile data. Responses include a success value web response (200).

createStudent(): Create a new Student profile. Return 200 status if successful. POST

createInstructor(): Create a new Instructor profile. Return 200 status if successful. POST

We will implement multiple more routes as we come to Iteration 2 and 3. These routes include, but are not limited to, verifying login (GET/POST), routes to retrieve Instructor and Student profile data (GET), route to retrieve the list of all CptS courses (GET), route for Instructors to add new courses in need of a TA (POST), route to retrieve courses that are not yet assigned TAs (GET), route for Students to submit a

TA application (POST), route for an Instructor to approve an application (POST), route for an Instructor to get the list of the TAs assigned to a class (GET), route for a Student to get the status of his TA application (GET), and a route for a Student to remove/cancel an application (DELETE).

III.2. Data design

We implemented two database entry types for each kind of profile. One for Students, one for Instructors.

The Student model consists of:

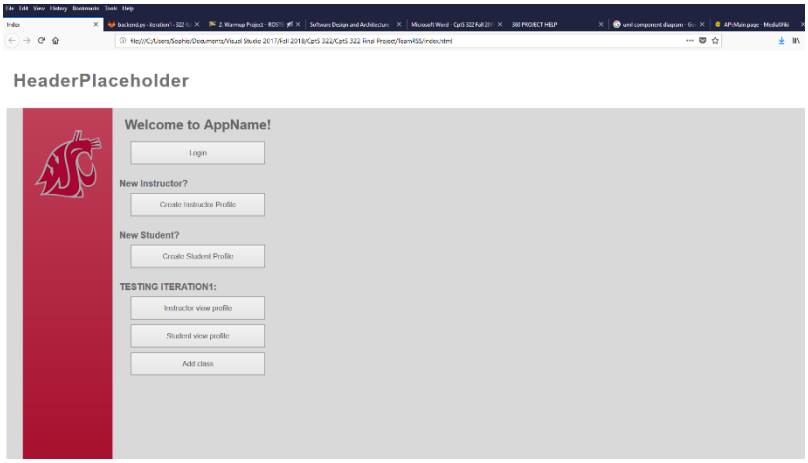
```
class Student(db.Model):
    id = db.Column(db.Integer, nullable=False, primary_key=True)
    studentId = db.Column(db.Integer, nullable=False)
    firstName = db.Column(db.String(128), nullable=False)
    lastName = db.Column(db.String(128), nullable=False)
    email = db.Column(db.String(128), nullable=False)
    major = db.Column(db.String(128), nullable=False)
    gpa = db.Column(db.String(128), nullable=False)
    graduationDate = db.Column(db.String(128), nullable=False)
    password = db.Column(db.String(128), nullable=False)
```

The Instructor model consists of:

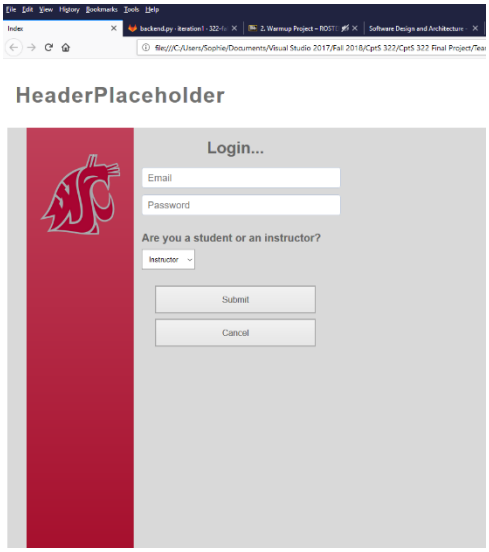
```
class Instructor(db.Model):
    id = db.Column(db.Integer, nullable=False, primary_key=True)
    facultyId = db.Column(db.Integer, nullable=False)
    firstName = db.Column(db.String(128), nullable=False)
    lastName = db.Column(db.String(128), nullable=False)
    email = db.Column(db.String(128), nullable=False)
    phone = db.Column(db.String(128), nullable=False)
    office = db.Column(db.String(128), nullable=False)
    password = db.Column(db.String(128), nullable=False)
```

III.3. User Interface Design

We deployed a clean, simple, easy-to-use interface we feel users will be able to pick up quickly. As we progress through the project, we may make changes to the User Interface as we delve deeper into the kind of data we will be displaying. Under TESTING ITERATION1 are the mock Frontend interfaces required by the Project Description.

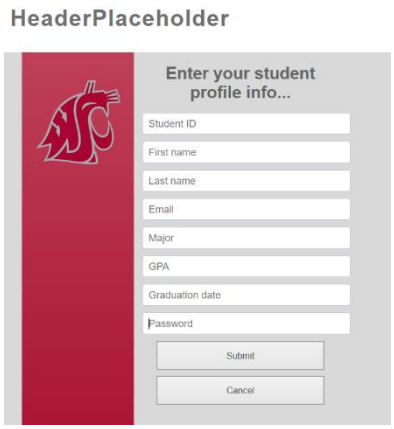


Homepage



Login

New Instructor Profile



New Student Profile

Testing Plan

(in iteration 2)

N/A, see iteration 2 later.

IV. References

Our references include our own coding experience and background knowledge.

V. Appendix: Grading Rubric

(Please remove this page in your final submission) (since there exists an entry for this page in the table of contents I will leave it here just in case.)

These is the grading rubric that we will use to evaluate your document.

Iteration-1

Max Points	**Design**
10	Are all parts of the document in agreement with the product requirements?
15	Is the architecture of the system described, with the major components and their interfaces?
5	Is the rationale for subsystem decomposition and the choice for the architectural pattern explained well?
	Are all the external interfaces to the system (if any) specified in detail?
15	Are the major internal interfaces (e.g., client-server) specified in detail?
15	Are the subsystems that the team has started to implement are described in the document? Are the algorithms and protocols for those subsystems explained in sufficient detail?
10	Is there sufficient detail in the design to start Iteration 2?
	Clarity
5	Is the solution at a fairly consistent and appropriate level of detail?
3	Is the solution clear enough to be turned over to an independent group for implementation and still be understood?
12	Is the document making good use of semi-formal notation (UML, diagrams, etc)
5	Is the document identifying common architectural or design patterns, where appropriate?
5	Is the document carefully written, without typos and grammatical errors?

Grading rubric for iteration-2

Max Points	**Revision**
15	Did the team revise their iteration1 design document and addressed the issues the instructor commented on?
	Design
	Are all parts of the document in agreement with the product requirements?
5	Is the architecture of the system described, with the major components and their interfaces?
5	Is the rationale for subsystem decomposition and the choice for the architectural pattern explained well?
	Are all the external interfaces to the system (if any) specified in detail?
5	Are the major internal interfaces (e.g., client-server) specified in detail?
10	Are the subsystems that the team has started to implement are described in the document? Are the algorithms and protocols for those subsystems explained in sufficient detail?
20	Did the team provided a UML class diagram showing the system's classes, their attributes, operations (or methods), and the relationships among objects? Did the team briefly explain the classes included in the diagram?
5	Is there sufficient detail in the design to start Iteration 3?
	Clarity
5	Is the solution at a fairly consistent and appropriate level of detail?
5	Is the solution clear enough to be turned over to an independent group for implementation and still be understood?
	Is the document making good use of semi-formal notation (UML, diagrams, etc)
	Is the document identifying common architectural or design patterns, where appropriate?
5	Is the document carefully written, without typos and grammatical errors?
	Testing

~~10~~	~~Is there a discussion of how unit testing for the API requests be done automatically? Did the team list the API routes that will be tested and and did the team explain how they will test each route?~~
~~5~~	~~Is there a discussion of how functional testing will be done?~~
~~5~~	~~Is there a discussion of how UI testing will be done?~~