**InstructED**

**Software Design**

**CSCI-P465/565 (Software Engineering I)**

**Project Team**

**Kevin Cao**

**Andrew Jedlicka**

**Blake Miller**

**Shaun Trimm**

**1. Introduction**

* 1. **System Description**

Our project is a Learning Management System implemented as a web app. We are aiming to create an application that allows instructors and students to interface, allowing for completion of assignments and grading of work.

**1.2 Design Evolution**

**1.2.1 Design Issues**

We want our app to run on any operating system using web browsers that support React and Node.js. Our app requires access to the internet, which will be a requirement. Our app is also intended only for use on personal computers rather than on cell phones or other interfaces, but hopefully will be usable on some other devices.

**1.2.2 Candidate Design Solutions**

Briefly (a paragraph or two) describe each of the possible design solutions that your team discussed/explored.

Our first decision was on our front-end: we debated between using HTML + CSS versus Javascript. We briefly discussed using other front-end architectures. We debated between React, Angular, and Vue. However, we did not discuss functionality on other systems, such as phones or tablets. We did discuss use of other database systems, notably comparing PostgreSQL and MySQL. These solutions would have likely resulted in a similar outcome to what we chose to use for the final version of our project.

**1.2.3 Design Solution Rationale**

Provide some reasoning for the design option that your development team selected.  Tie in the selected approach's ability to meet the requirements, and most effectively address the design issues.

We chose JS because we believed it to be the most powerful and well-documented solution. We used similar logic to choose React, which we chose above Angular and Vue because it was more well-documented and established. PostgreSQL was chosen over MySQL because Heroku, the hosting solution we chose, had support for it rather than MySQL.

**1.3 Design Approach**

**1.3.1 Methods**

We have used design documents as a primary way to maintain our design vision. This has allowed us to modify our design while maintaining a consistent “bible” of design choices we have made, while also giving us an easy way to demonstrate design needs to our customer.

**1.3.2 Standards**

We have consistently used camel-casing for our naming convention. Additionally, we have used a standard page format with a navbar that is consistent across all pages. Finally, our tables adhere to Third Normal Form for consistency.

**1.3.3 Tools**

We are using Heroku to host our site, which does not generate anything other than a working copy of our git branch on a remote site.

**2. System Architecture**

**2.1 System Design**

Provide a description of  the high-level design of your system.  Include a system diagram that captures the major components and the external interfaces.  Describe the operation of the system (high-level) and the interaction between components.

Our site allows for our three types of users, which include students, instructors, and administrators to access a variety of learning tools. Students can create accounts, message professors, complete assignments, view their schedule,

Students can do the following:

* Reset their password, modifying their user info by using a reset-password link sent via email
* View all their courses as well as the assignments in their courses that are upcoming
* Complete assignments, creating an entry in the table for that assignment for their submission which contains student id, assignment id, submission, submission date, and grade (which is initialized to null)
* Chat with other students and professors
* Search for assignments
* Change password

Instructors can do the following:

* View all their courses
* Create assignments, creating a row in a table that contains the course id, assignment id, due date, description of assignment, and submission type
* View individual submissions for assignments and assign them a grade, modifying the entry in submissions table for a given student’s submission
* Chat with students
* Change password

Admins can do the following:

* Change role of existing user
* Enroll student in a course
* Create a course and assign an Instructor to it

**2.2 External Interfaces**

We are currently using Duo authentication’s interface. We have been using this interface to secure our logins, accessing Duo only when we have a successful username/password entry pair. We are also using Facebook authentication, which allows for users to login using Facebook.

**3. Component Design**

The Component Design section details the proposed design of each system component.  A system conponent is a functional partition of the system.  Components may be organized as you see fit - a component may be a collection of objects, or a single object.  However, a system must be composed of multiple components (that is, a system cannot be one component).  The layout of this section is at your discretion, but please include the following (at a minimum) information for each component:

* **Component Name**  
  Login and Registration
* **Component Description**  
  The main page on our site prior to login is the login prompt. The login prompt allows for users to log in using existing credentials, create a new account, login with Facebook, or reset their password. If a user logs in, they are redirected to Duo, which redirects if successful to the dashboard. If they want to create an account, they are directed to that page where they input account information. If they login with Facebook, it is similar to regular log-in. If they choose to reset their password, they enter their email, then go to the emailed URL where they enter their username and new password twice to reset it.
* **Responsible Development Team Member**  
  Kevin Cao
* **Component User Interface**  
  If this component includes a user interface, include some details about the interface including what aspects of the component will be available through the interface, a description of each of the user screens that are expected for this component, and a description of each of the user notifications and/or messages that are planned for this component.  
  Users will receive a message if they incorrectly log in, enter non-matching passwords during registration or password resetting, or fail to login with Facebook or Duo. User screens are login, register, forgot password, and reset password.
* **Component Objects**  
  Describe the objects/classes that comprise this component.  Provide a listing of expected data members and methods for each class.  Note in the description if a given object/method does any of the following (Note: Some of this information may be captured in class diagrams or other parts of the design - simply be sure it is included somewhere):
  + Makes an operating system call (cite expected system calls to be made)
  + Makes a hardware-specific system call (cite calls to be made)
  + Creates/alters/deletes a file (cite file names)
  + Explicitly calls the method of another object (cite name of other object)
  + Is explicitly called by another object (cite name of other object)
  + Passes data to another object (cite data structure and name of other object)
  + Receives data from another object (cite data structure and name of other object)
  + Is derived from another object (cite name of other object)
* DuoLogin
  + Passes DuoAuthState to login class, checking if duo auth succeeds
* ForgotPassword
  + Posts to /forgotPassword
* Login
  + Posts to /authenticate, checking db for password
  + Posts to /authenticate/facebook, checking for successful facebook validation
  + Pushes user to /duologin if successfully authenticated
* Navbar
  + Accesses Login, Create Account, and Forgot Password
* Registration
  + Posts to /users
  + Posts to /authenticate/facebook
* ResetPassword
  + Posts to /updatePassword
  + Gets from /resetPassword/(token)
* Queries:
  + CreateUser updates Users with insert
  + CreateLoginToken creates login token from userID
  + LoginUser checks Users table using email and password
  + loginFacebook checks Users table using email
  + loginFacebook inserts into users email and name
  + ForgotPassword checks Users
  + ForgotPassword creates password reset token
  + CreatePasswordRestToken inserts into PasswordTokens
  + CreatePasswordResetToken updates PasswordTokens
  + CheckResetExpiration checks PasswordTokens
  + UpdatePassword checks PasswordTokens
  + UpdatePassword updates Users
* **Component Interfaces (internal and external)**  
  This component will provide all other components with the current user. This will include their user ID, name, type, and anything else. This component is the first to be loaded, and all other components require this to function. This also interfaces with our Duo and Facebook implementations.
* **Component Error Handling**  
  Error Case 1: Input for username can be SQL injection, prevented by ensuring entry is properly formatted and inserted rather than added to query  
  Error Case 2: Users can access URL’s without being directed to them by our application so we look for queries and cookies to ensure no user can get into another account  
  Error Case 3: Users can input incorrect passwords or inconsistent passwords so we notify them if this is the case and do not allow them to access sensitive information  
    
  **Revision History**

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Change Description** |
| 1 | 10/4/20 | Begun and added Login/Registration |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Page Author/Creator:** [Adeel Bhutta](http://homes.soic.indiana.edu/aabhutta/) **Last Modified:** 8/23/2016