**Course 360**

**Software Design**

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

**Project Team**

Sai Rohith Achanta

Juhi Deshkar

Aravind Parappil

Kriti Shree

**1. Introduction**

**1.1 System Description**

The main purpose of this project is to develop a Course Registration Management System which would provide students with a unified module, not only letting the students register for the courses and look up the course details but also the facility to chat with the professors and registered students. It would facilitate a consolidated view of the course details, seats available, ratings of the professor and past student grade records. Similarly, the faculty will also be able to view their courses , registered students, etc. The system will also have payment portal for the students to pay their fees. Registrar will be the administrator, who has the central authority to manage the student and professor account.

**1.2 Design Evolution**

Component-based development techniques involve procedures for developing software systems by choosing ideal off-the-shelf components and then assembling them using a well-defined software architecture. It saves time and money when building large and complex systems: Developing complex software systems with the help of off-the-shelf components helps reduce software development time substantially.

**1.2.1 Design Issues**

It works only on LINUX system and compatible with the system having browsers installed.

**1.2.2 Candidate Design Solutions**

Now in an object-oriented language, this one large program will instead be split apart into self contained objects, almost like having several mini-programs, each object representing a different part of the application. And each object contains its own data and its own logic, and they communicate between themselves.

**1.2.3 Design Solution Rationale**

Saves time and money when building large and complex systems: Developing complex software systems with the help of off-the-shelf components helps reduce software development time substantially.

 Autonomous – Components should be independent from each other. Components should be able to continue to function when other components are not available.

 Loosely coupled – Components should be loosely coupled through explicit service definitions and be re-usable across applications.

 Performance – Components should be capable of handling load as multiple applications may use one and the same component.

**1.3 Design Approach**

**1.3.1 Methods**

Hands down mockup , got validated from customers

**1.3.2 Standards**

Top down method and Camel naming convention is being used

**1.3.3 Tool**

Atom, Pycharm, postman, postgresSql, reactjs,nodejs, github

Describe any tools that you plan to use to assist you in developing the design, and specify exactly what products will be generated by the tools.

**2. System Architecture**

**2.1 System Design**

NA

**2.2 External Interfaces**

NA

**3. Component Design**

**Component Name**: Provide a unique component name

Sign in

Forgot password

Registration

Add Course

Delete Course

Edit course

Retrieve all courses

Retrieve course by name

Retrieve all professors

Retrieve all students

Retrieve professor calendar details

Add to shopping cart

Delete from shopping cart

Enroll to courses from shopping cart

Drop courses

Save comment from each user for each course

Retrieve Shopping cart data

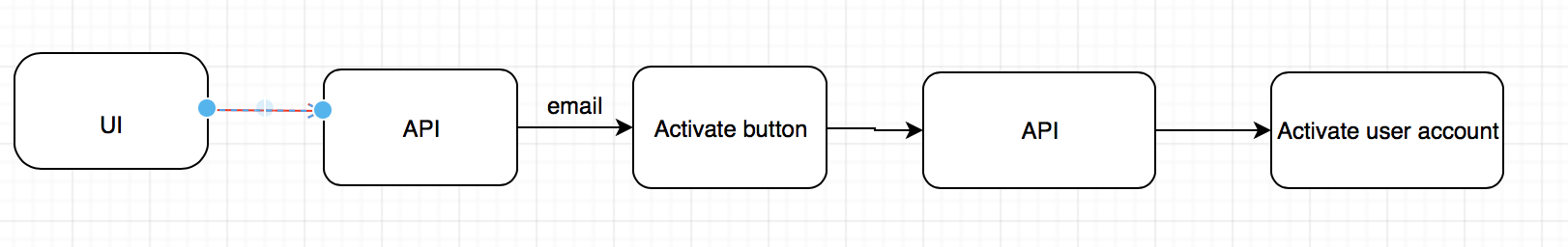
Add financial aid to user table

Pay fees API after checkout

**Component Description**:

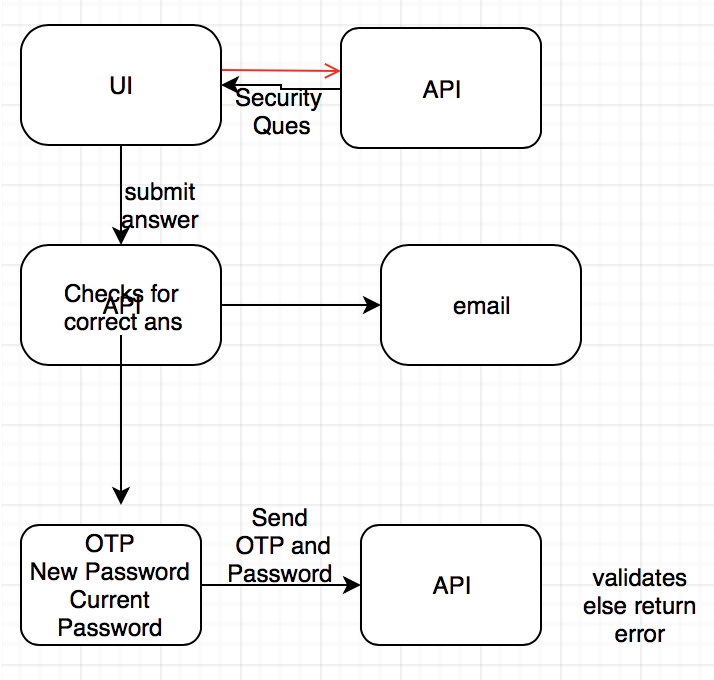
**Registration**

User gets the screen to enter the registration details , activation mail is send to the user’s email id. The user clicks on the link to activate the account.



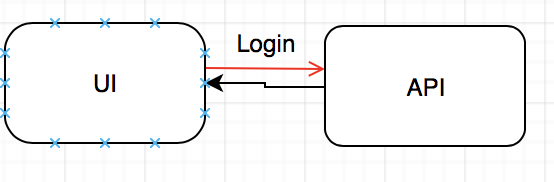
**Forget Password**

Incase the user forgets password, it can be retrieved using the security question. Once the user clicks on the forgot password , one time password is send to their registered email id. The user can use this email to answer the security question to reset the password



**Login**

User needs to enter email id and password to login



**Adding a Course:**

Admin will pass the course details while adding the course

Pass course details

API

UI

**Deleting a Course:**

Admin will delete a course by passing course id.

Pass course id

API

UI

**Updating a Course:**

Admin will be able to update the course by passing the course details

Pass course details

API

UI

**Retrieving all courses**

All Courses are returned when this API is called

API

UI

Return all courses

**Retrieving Professors:**

Admin can retrieve the details of all the professors.

API

UI

Return all Professors

**Retrieving Students:**

Admin can retrieve all the students.

API

UI

Return all Students

**Course Search:**

Courses can be searched by passing a substring to the API. The API will return a list of matching courses.

Send substring of course

API

UI

Return a list of matched courses

**Retrieving calendar data for professor:**

A professor can retrieve his calendar by passing professor id. The API will return his schedule

Send professor id

API

UI

Return schedule for professor

**Add to shopping cart:**

Student can add courses to the shopping cart.

Sending course

API

UI

**Delete from shopping cart**

This API deletes the selected course from shopping cart.

Sending course

API

UI

**Enroll to courses from shopping cart**

This API sends the course ID for the user in which he wants to get enrolled.

Sending course ID

API

UI

**Drop courses**

This API sends the course ID for the user which he wants to drop.

Sending course ID

API

UI

**Save comment from each user for each course**

This API saves comments regarding the courses from each user.

Sending comments

API

UI

**Retrieve Shopping cart data**

This API retrieves data send by the user

Send ID

API

UI

Return courses added in shopping cart

**Add financial aid to user table**

This API sends financial aid if provided to the user.

Send financial aid

API

UI

**Pay fees API after checkout**

This API pays the fees using credit card or e-check after user checkouts after enrolling into the courses and user gets the email with receipt.

API

UI

**Responsible Development Team Member**

**Component Diagram**

Kriti has designed the component diagrams.

**Component User Interface**

UI was developed by Aravind and Kriti

**Component Objects**

Sai and Juhi have developed backend APIs for the components.

**Component Interfaces (internal and external)**

**Component Error Handling**

**Revision History**

First Prepared on 09/28/2018

Revised on 10/28/2018

**Page Author/Creator:** Kriti Shree

**Last Modified:** 10/28/2018