New CSCI Website
Software Requirement Document

Date
March 13, 2017

Team Lead
Peter Alford

Team Members
Alex Patton, Mark Gong-Guy, Frank Navarro-Velasco, Philip Stout

Prepared for
Professor Long Ho
CSCI 450 Software Engineering Course

Table of Contents

Table of Tables

Table of Figures

1 Scope

1.1 Identification

This Software Requirement Document (SRD) defines the software requirements for the new Computer Science at Biola website. It includes use cases and sequences of events by users and administrators interfacing with the website and its components.

1.2 System Overview

The new site will include private and public code repositories, class information, assignment submission, code review, discussion boards, and chat features. There are three main areas of added functionality. First is the ability for instructors to host information which is not publically available on a secure site without risk of copyright infringement. This will centralize all class offerings so that information is readily available to all students. Second, the code collaboration tool will allow students to share code with a TA and get help regardless of their ability to be physically present in the math lab. This is uniquely beneficial to off campus students. Thirdly, the discussion boards and chat features will allow students of all years to assist one another with the learning process, encouraging cross-year engagement.

2 Referenced Documents

The following documents by reference form a part of the system.

- JavaScript: Google style guide
    - https://google.github.io/styleguide/jsguide.html
- Python: PEP 8
    - https://www.python.org/dev/peps/pep-0008/
- HTML/CSS: W3Schools Style Guide and Coding Conventions
    - https://www.w3schools.com/html/html5_syntax.asp
- Canvas API
    - https://canvas.instructure.com/doc/api/index.html
- CodeMirror API
    - https://codemirror.net/doc/manual.html

3 FUNCTIONAL Requirements

3.1 General Requirements

3.1.1 Requirements

- The website shall be compliant to W3C HTML standards.
- The website system shall operate within the most recent releases of Chrome, Internet Explorer, Safari, and Firefox.
- The website system shall be developed in accordance with the agile development process.

3.1.2 Development Goals

- The website will be developed using an Object Oriented Design paradigm where applicable, and according to W3C standards otherwise.
- The website will be developed using an HTML/CSS framework with bootstrap, as well as Javascript, PHP, Python, and mySQL.
- The website will be tested continuously with specific unit tests, and prior to each iteration will be integration tested in a jenkins build server.

3.2 Architecture

3.2.1 Homepage:

3.2.1.1 Language:

- HTML, Javascript

3.2.1.2 Aspects

3.2.1.2.1 Navigation bar:

- Pre-login, will have login form button on right hand side of navbar
- Stays at top of screen.
- Post-login, will have drop down menu options to navigate through site.

3.2.1.2.2 Login form:

- Button located within navbar
    - Post login, will be a message that reads “welcome <username>!”
- Clicked login button will show Popup login box
- Box for entering Biola NetID
- Box for entering NetID Password
- Button to submit form

3.2.1.2.3 Link to https://www.biola.edu/computer-science-bs

3.2.2 Virtual Classroom

3.2.2.1 Language:

- PHP, JS, HTML/CSS, mySQL

3.2.2.2 Aspects

- 3.2.2.2.1 Attendance Aspect
    - UI Elements
        - HTML and CSS Styled button for starting/ending class
        - Popup for attendance code display for teacher
        - Dashboard for displaying classes
        - UI for virtual classroom
    - Attendance Class (Allows teachers and students to keep track of attendance for a class without having to waste time on roll call)
        - startClass(int classID) -> returns int
            - Starts class session and returns sessionID
        - getAttendanceCode(int sessionID) -> returns int
            - Used for both retrieving current attendance codes and generating new code based on last code timestamp
        - markStudentPresent(int classID, classSessionID, attendanceCode, userID) -> return bool
            - Checks if user is in class and if attendanceCode is correct and then marks student present and returns true
        - markStudentTarty(int classID, int sessionID, int userID) -> return bool
            - Checks if user is in class and class is in session and then marks them tardy
        - endClass(int sessionID) -> returns bool
            - Ends the session and places any remaining students on role in absent array.
- 3.2.2.2.2 Whiteboard Aspect
    - UI Elements
        - HTML and CSS Styled button for starting/ending screenshare
        - UI for virtual classroom
    - Whiteboard Class (Allows teacher to display his screen as well as mirror our Collaboration tool to all the students)
        - startShareScreen(int SessionID, int userID) -> return bool
            - Enables screen share for session
        - startCodeScreen(int SessionID, int userID) -> return bool
            - Enables code share for session
        - endScreen(int SessionID, int userID) -> return bool
            - Validates the user and ends either the code screen or share screen

- 3.2.2.2.3 Quiz App/Q&A
    - UI Elements
        - HTML and CSS Styled button and text area for asking and answering questions
        - UI for virtual classroom
        - HTML popup for teacher to see who has answered the question and to mark as correct or incorrect
    - Quiz Class (Allows teacher to ask question to class)
        - askQuestion(int userID, char question, int sessionID) -> return bool
            - Poses a question to the session in the form of an html popup
        - answerQuestion(int userID, char answer,int sessionID, int questionID) -> return bool
            - Answers a question from the session
        - checkwhoAnswered(int questionID) -> return array
            - Returns who has answered the question
        - markCorrectAnswer(int answerID) -> return bool
            - Mark an answer as correct
        - markIncorrectAnswer(int answerID) -> return bool
            - Mark an answer incorrect
        - retrieveAnswers(int questionID) -> return array
            - Returns answers for questionID with answerID in array

3.2.3 Collaboration Tool

3.2.3.1 Language:

- PHP, JS, HTML/CSS

3.2.3.2 Aspects

3.2.3.2.1 CodeMirror Interface Aspect

- Shared text editor in order to allow the student to show a TA their code from a remote location.
    - The initiator of the session will insert a recipient's name (either actor could be a TA or a student) and press a "start session" button.
    - The recipient of the request will receive a hyperlink in their inbox to the shared collaboration page.

3.2.3.2.2 GitHub access Aspect

- Allows student to upload a saved GitHub file and upload it to the shared collaboration session, as opposed to simple copy-paste procedure.
    - The student will have a few options for inserting code into the session
        - One of them would be a button allowing them to upload a file from GitHub
        - Upon selecting this option, the student would be able to look through their GitHub repository and select the file they want to upload

3.2.3.2.3 Private Sessioning Aspect

- Creates a private user-user screen sharing session that will be limited to the given student and TA.

    - Both student and TA will be signed into their unique accounts, and only the given recipient would receive a hyperlink unique URL.
    - This specific URL would expire as soon as the TA ends the session

3.2.3.2.4 Simultaneous editing

- Collaboration document issued at the time an invitation is sent.
    - Unique URL opens to collaborative file

3.2.3.2.5 Upload from System Aspect

- Allows the student to select a file from his/her machine to upload and display in the text editor
    - “Upload from System” button will be an option along with “Upload from GitHub” button

3.2.3.2.6 Download to system Aspect

- Allows the student to download the modified code to his/her machine and then save it within their program’s path.
    - Download Page button will be present and visible to the student.
        - Upon pressing this button, the contents of the page will be downloaded, allowing the student to save the contents under any given path.

3.2.3.2.7 UI

- Layout framework will be grid layout
- Will include: Download, upload, and end session buttons, perhaps in a toolbar format.

3.2.4 Class Creation Tool

3.2.4.1 Languages:

- PHP, JS, HTML/CSS

3.2.4.2 Aspects

3.2.4.2.1 Creation Aspect

- Allows the professor to create a class by inputting all of the important data into a form.
    - Form includes: Title, Subject, Description, Teacher, Semester, Permissions, etc.
    - Create button at the bottom create the class using the data in the form
        - This creates a class page with all of the pertinent aspects described below.

3.2.4.2.2 Calendar Aspect

- Displays a simple calendar that can be view by the students and the profs
    - Will include all the class assignments and important dates

3.2.4.2.3 Assignment Aspect

- A professor can create an assignment to appear in the assignment tab of the class page.
    - This is a form that includes: Title, Description, and Submission details
    - Clicking the create button will generate the assignment.
        - It will create a submission button with determined details
        - It will populate the assignment to the calendar

3.2.4.2.4 Announcement Aspect

- A professor creates an announcement for the class about an opportunity or class cancellation
    - The form will include: title and description

3.2.4.2.5 Rubric Aspect

- A button that allows the view to look at and edit a class rubric
    - All of the inputted assignments factor into this rubric
    - Can edit weightings for different assignments

3.2.4.2.6 UI

- Layout framework will be grid layout
- The class page will include all known classes
- Each class page has announcements on the front page and a panel of buttons on the left that include: assignments, files, create announcement, calendar, rubric.
- Clicking on these pages will reroute you to their respective pages with their own editing data.

3.2.5 Forum

3.2.5.1 Languages:

PHP, JS, HTML/CSS

3.2.5.2 Aspects

3.2.5.2.1 Menu Aspect

- The Menu will be a vertical menu on the left side of the screen
- It shows all channels to the user.
- There will be a button to create a channel
    - Once pressed, a pop-up window will allow the user to enter the channel name.
    - Once created, it will create a new History table in the database, and will appear in everyone's menu once they refresh their page.

3.2.5.2.2 Viewer Aspect

- Comments will be shown in chronological order
- Comments will be shown with other users pinned to the left side of the window and the user's comments pinned to the right
- Load history into window.
- Window will be embedded in the HTML page.
- When a user changes the channel via the menu, the new channel content will be uploaded into the window dynamically without refreshing the whole page.
- The history window needs to be up to date with the latest posts, therefore there will be a listener querying the server at least every second to check if the last commentID it recorded is the most recent commentID. If not, it retrieves all more recent comments and uploads them to the page.

3.2.5.2.3 Input Aspect

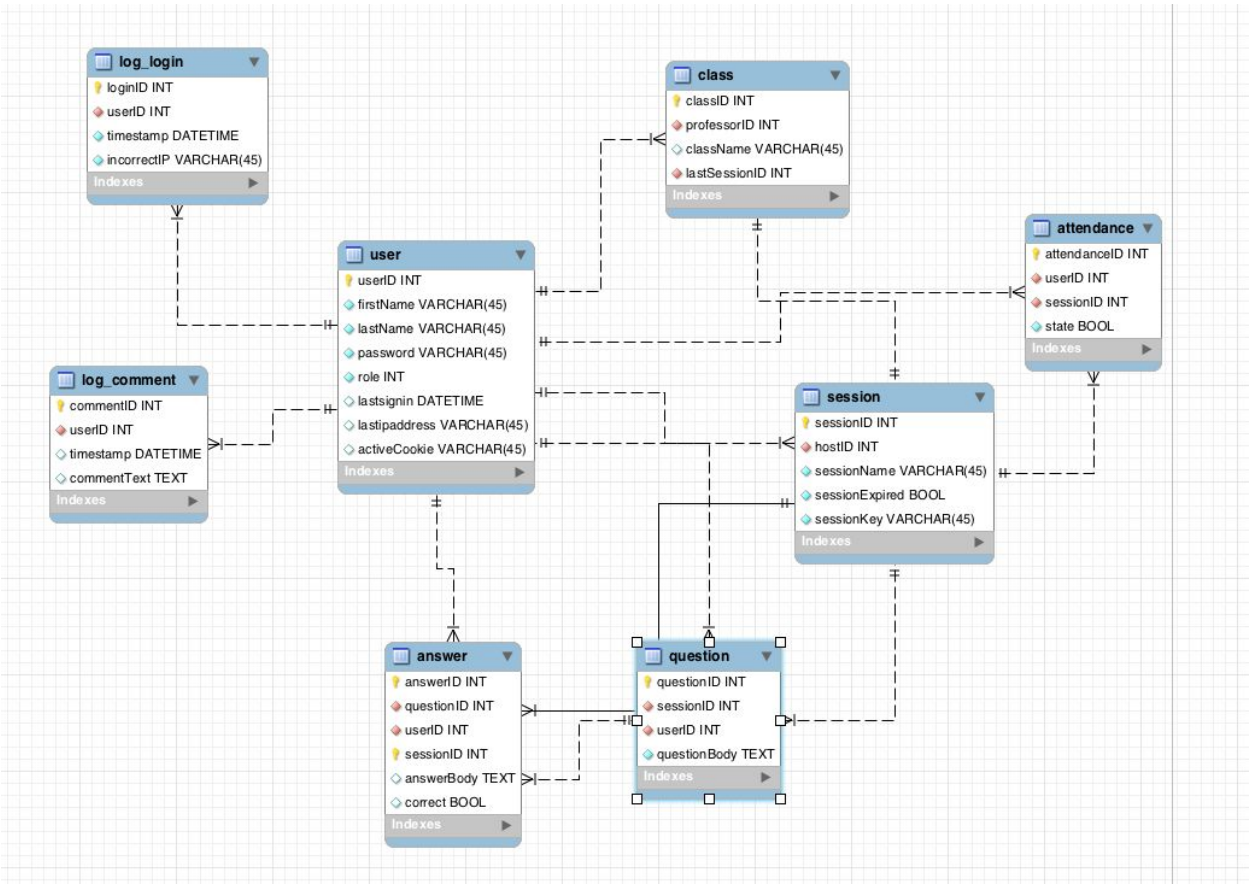
- An HTML text box at the bottom of the history window.

- Upon clicking the post button a JS script will send the information to the database to update the history table with a new entry.
- Once there is confirmation of success, the script will update the history window with the new post.

3.2.5.2.4 Channel Aspect

- Channels will be constructed of the following database objects
    - History table
        - Columns
            - userID (String)
            - dateTime (String)
            - commentText (String)
            - commentID (Primary Key) (INT)
                - Sequential starting from 0
        - History Table is loaded whenever the user switches to the given channel
            - Loading criteria
                - Load 30 days of content
                - If 30 days exceeds 2mb, cutoff at 2mb of data

3.2.6 Database Design



3.2.7 Repository

3.2.5.1 Languages:

- mySQL, PHP, JS, HTML/CSS

3.2.5.2 Aspects

3.2.5.2.1 Database Aspect

- See 3.2.6.

3.2.5.2.2 Viewer Aspect

- File tree representation
    - Folders and files are displayed as Boxes
    - In any given folder, the files and subfolders are displayed.
    - Clicking and dragging the folders/File objects into a folder will
        - Reorganize the database with the new file structure
        - Eliminate the folder/file to be dropped from the graphical representation of the previous parent folder.
    - Folder Object
        - Clicking on the folder will refresh the page with the contents of that folder displayed.
    - File Object
        - Clicking on the file object will present the user with a menu of two options
            - 1. Download to the user's hard drive
            - 2. Open in the code collaboration tool

3.2.5.2.3 Upload Aspect

- Upload Button
- Upload process
    - Can choose either file or folder upload
    - Once upload button has been pressed, the OS's finder will open, and the user can search for a file/folder
    - The chosen file gets encrypted using SSL and sent to the server
    - The location to upload to in the server is determined by which folder the user is currently viewing in the website
        - Needs to have some marker there
    - This is sent with the file to the server in a PHP script to generate the SQL query that will insert the file in the correct spot
    - Once the file is inserted, the HTML is updated for the page and a new object is created.
    - The page refreshes with the new object in place.

3.3 States and Modes of Operation

The following list contains the states. Operator can operate in these states.

1. The first state is pre-login. The website homepage will redirect to a login page where students and faculty can login to the secure pages.
2. The second state is post-login, which defaults to a homepage with a navigation dashboard.

3.4 CAPABILITY REQUIREMENTS (USE CASES)

This paragraph identifies a required capability and itemizes the requirements associated with the capability. The requirements specify required behavior of the software system and includes applicable parameters, such as response times, throughput times, other timing constraints, sequencing, accuracy, capacities (how much/how many), priorities, continuous operation requirements, and allowable deviations based on operating conditions. The requirements include, as applicable, required behavior under unexpected, un-allowed, or "out of bounds" conditions, requirements for error handling, and any provisions to be incorporated into the software system to provide continuity of operations in the event of emergencies.

Use Cases traditionally form part of the Object-Oriented style of design and development. There are a number of benefits associated with using Use Cases to document and analyse the requirements:

· Improves the ability to validate the Component as Use Cases describe *how* a Component should behave rather than *what* it should do.

· Ensures completeness of requirements by promoting the consideration of all conditions, success and failure scenarios.

· Specific technique available to transition from requirements to design.

· Development of test cases is easier as all scenarios are included in each use case, which in turn improves the traceability between use cases and test cases.

· Can be used to drive the identification of other project aspects, such as GUI designs, project planning and build planning.

The procedure has the following steps

1. Understand the domain, behaviour and context of the component

a. Gather “building codes” and quality attributes (using 7 software qualities)

2. Identify Use Cases - Understand the behaviour of the Component by identifying the major Actors and Use Cases required to satisfy the Actors’ goals

a. Identify the Primary Actors - These are the people and external systems that have goals to be achieved through direct interactions with the Component. Define the goals that each Primary Actor needs to achieve through their interactions with the Component. The goals are similar to

a purpose, however should be worded as individual sentences so that success or failure of that goal can be easily determined

b. Identify the Use Cases - Identify a Use Case for each goal of the Primary Actors. Only one Use Case should be used to address each goal, however each Use Case may satisfy a number of goals across various Actors. If multiple Use Cases are identified for one goal, one of two options can be taken: (a) revise the focus of the goal and raise it to a higher level so that one Use Case will satisfy the goal, (b) break the goal into a number of smaller goals that can be satisfied by one Use Case

c. Develop the Use Case Diagram showing the relationship between the identified Primary Actors and Use Cases

d. Expand, update, and finalize Use Case if needed

3.4.1 *Professor Creates Class* Use Case

The Professor Creates Class Use Case presents the process of a professor creating a class to publish to the students.

3.4.1.1 *Create Class* Activity

3.4.1.1.1 Requirement

1. Initial Form: A form that gathers basic information to create the class.
2. Assignment Creation: Has a header, a description, and a submission area.
3. Calendar: A calendar that has class events and assignments on it.
4. Announcements Creation: Header and a description to convey info to the class.
5. Grading Rubric: A spreadsheet with the grade breakdown and calculation
6. Upload Button: A button that allows for uploading documents into the class repository.
7. Repository: A repository with documents for the class as well as code.
8. Save/Publish: Allows to save things above that they were working on.

3.4.1.1.2 Description

The Professor will create a class and publish it for the students.

3.4.1.1.3 Preconditions

Professor must be logged in to the website.

3.4.1.1.4 Post-conditions

Class page has been created and is viewable by the students.

3.4.1.1.5 Flow

1. He will go to the CSCI website and hit the login button in the upper right hand corner. This will generate a pop-up box or a new page with which he input his username and password to login. It will then revert him back to the main page. (CSCI website/Login button/Login page/Username and password authentication and encryption)

2. After logging in, He will go to the class section/tab of the website and clicks the button new class. Then a form will appear that needs to be filled out to define what the class is. (CSCI website/class page/create class button/create class form)
3. Once the professor has filled out this form he will hit create. This will create a new class with a title on the class page of the website. Students and professors will then be able to access the class by clicking on it from the class page. (new class form/class page/accessing the new class)
4. Once the professor does this he will have several option on the right hand side of the page to create assignments, announcements, edit the grading rubric, upload files, take attendance, and edit the class calendar. (New class page/Buttons for the above options)
5. He will then begin by creating a welcome announcement for the front page. (Announcements buttons/announcement form/save/publish)
6. After this he will set up the grading rubric. (Grading rubric button/Grading rubric form with default values/save/publish)
7. Then he will input the initial assignments which will populate to the class calendar. (Add assignment button/assignment form/publish/calendar population)
8. Lastly he will upload the course syllabus to the file repository. (Upload button/upload pop-up/publish)
9. Through the process he will have the ability to save his work or publish it. One option will be to save it until a later time and the other option puts it out for everyone to see. (Save/publish/repository to hold all the partial forms/system to auto save every so often.)

Alternate Flow:

1. Alternate flow 1: If the Professor's user times out while creating the class the system will save the inputted data and then go back to the homepage or the login page.
2. Alternate flow 2: Professor wants to import a previous iteration of the class.
3. Alternate flow 3: The site/internet crashes while he is inputting the data for his class.
4. Alternate flow 4: A new class gets saved over and old class.
5. Alternate flow 5: Error arises when the assignments can sync with the class calendar.

3.4.2 *Code Collaboration* Use Case

The Code Collaboration Use Case presents the process by which a student and TA would use the tool.

3.4.2.1 *Code Collaboration* Activity

3.4.2.1.1 Requirement

1. Integrated codemirror API into the webpage.
2. Private sessioning
    a. Must be logged in to view
    b. Secure connection
    c. By invitation only
3. Student can only invite TA's
    a. TA's can invite anyone (student, TA, or Professor).

3.4.2.1.2 Description

A student starts a collaboration session with a TA, edits, and saves code.

3.4.2.1.3 Preconditions

User logs into secure portal.

3.4.2.1.4 Post-conditions

The file has been downloaded or committed to github.

3.4.2.1.5 Flows

Basic Flow:

1. The student will log on to the csci webpage during the assigned TA's hours (or off-hours, if the TA is ok with it), and sign into his specific csci account that he has already set up (if not, he'll see a form to create a new account, and he'll follow those steps).
2. After signing in, a whole new set of previously hidden site features will be visible (most likely hosted on a second web page that can only be accessed through signing in, so that it's invisible to those not signed in). One of the features will be that of starting a session with the code-sharing that we will implement.
3. The student will send a request to the TA (or vice versa, if we want to lock it down to only TAs) in order to start a session within the code collaboration.
4. Both students will then see the same web page and be able to type different things to each other.

Alternate Flow:

If only the TA is able to start a session, it would follow the above procedures (both students signing into their accounts), but the student will have to alert the TA through a

different means, and have the TA send the request to collaborate. Aside from that difference, they all seem to be equal.

3.4.2.1.6 Requirements

3.4.3 Virtual Classroom Use Case

3.4.3.1 *Virtual Classroom* Activity

3.4.3.1.1 Requirement

1. Connection to Canvas to pull class list or mysql database holding class information and roster
2. UI for web interface to display classes and a button to enter a session
3. Generation of attendance code and validation of that code using php to check a mysql database
4. HTML Modal box to allow for input from student and displaying of question from teacher
5. HTML button to submit the answer and store into a mysql database
6. Whiteboard or Bash Terminal UI and implementation
7. Mysql Table storing questions and answers from students
8. HTML button to close the session
9. JS code to send to server when student is on page, so when student leaves server notes it down in the attendance table.
10. Same page as entering but the button is able to tell if the class is in session and return appropriate button using php
11. If class is in session will mark the user as tardy in the attendance table

3.4.3.1.2 Description

The virtual classroom is a webpage where the professor is able to take attendance, liveshare notes, and monitor class activity.

3.4.3.1.3 Preconditions

The student must be logged in.

3.4.3.1.4 Post-conditions

The professor will terminate the session, ending the connection between students and the server.

3.4.3.1.5 Flow

1. Student heads to the new CS website and logs in with their credentials (NetID or Email/Password)
2. Student is able select from a list of classes that they are currently enrolled in.

3. Student is then able to enter the virtual classroom and waits for teacher to give attendance code.
4. Student gets attendance code from the teacher and then enters it and is able to view the board.
5. When teacher places a code or terminal question on the board, student is able to click it and enter their own answer.
6. Once the student has entered an answer they are able to submit and await teacher instruction.
7. At the end of the class student logs out or closes browser and the session ends.

3.4.4 Code Repository Use Case

3.4.4.1 *Code Repository* Activity

3.4.4.2.1 Requirement

1. Integrates with mySQL database
2. User Interface
    a. File tree representation
        i. File icons
            1. Pop up menu with options to:
                a. Open in code sharing window
                b. Download
                    i. Interface with onboard hard drive
    b. Upload Button
        i. Interface with onboard hard drive

3.4.3.3.2 Description

The code repository is a partitioned file storage system, designed to allow students to store their files in the cloud. It will be accessed via a dedicated page, and can upload and download to the user's system.

3.4.3.3.3 Preconditions

The software system components have instantiated, registered, and all attributes are set to valid values.
The software system is in a state specified by the software system States and Transitions Diagram in which it will need to open or create files.

3.4.3.3.4 Post-conditions

The user closes the

3.4.3.3.5 Flow

1. I navigate to the repository page, which redirects to the login page

    a. (File Tree Representation)
2. I log in and it then redirects to the repository page
    a. (Login page, file tree representation)
3. I see the general partition and its contents displayed.
    a. (file tree representation)
4. I navigate by clicking on the folder that I want to put the files in.
    a. (File Tree representation)
5. I click upload
    a. (Upload Button)
6. I choose, from the pop up window, the file that I want to upload and click upload.
    a. (Interface with onboard hard drive)
7. The file appears in the folder online and I am able to view it in a text editor by clicking on it.
    a. (File Tree Representation, Pop-up menu, interface with code editor)

3.4.5 Forum Use Case

3.4.5.1 *Forum* Activity

3.4.5.1.1 Requirement

1. Left side menu of available forums
2. Viewing box for previous comments in the channel
    a. Previous comments will have the username of the posting user on them, as well as a date/time stamp
3. Text entry box for creating comment
4. HTML button for submitting the comment

3.4.5.1.2 Description

The Forum is a multi-channel communication platform, allowing students to communicate in a secure location about group projects or knowledge sharing.

3.4.5.1.3 Preconditions

User must be logged in.

3.4.5.1.4 Post-conditions

User logs out.

3.4.5.1.5 Flow

I as a student want to seek help by asking a question on the CSCI 105 forum

1. I navigate to the forum page and select from the menu on the left hand side of the screen the csci 105 forum.

2. I see the history of the conversation and scroll through it to see if there is an answer to my question.
3. Not seeing an answer, I click on the text entry field and begin to type my question.
4. When finished, I press the enter key or hit the send button to post my message in the forum.
5. I see that my classmate is over generous and has pasted his code solution into the forum, however, as soon as it is pasted, Dr. Lin deletes the comment.
6. Frustrated, I decide to create a new channel by selecting the option from the left side menu. A small window opens in browser and I enter the forum name, and hit create.

3.4.6 Website Login Use Case

3.4.6.1 *Students and Professor Logging-In Activity*

3.4.6.1.1 Requirements

1. Homepage: The homepage must be able to accept login information and validate it to a user stored within the database
2. Navigation Bar: Has a layout of all the pages the user has privileges to access

3.4.6.1.2 Description

The user is looking to login to their account on the Biola CSCI web portal

3.4.6.1.3 Precondition:

User exists within the CSCI department and has a login username and password

3.4.6.1.4 Post-conditions

User can login into the CSCI web portal and gain access to their account

3.4.6.1.4 Flows

Basic Flow:

1. The user types in the url into the address bar of their primary internet browser and the browser loads the page. (CSCI Homepage)
2. The user clicks the login button within the navigation bar (Navigation Bar)
3. The user then enters in Login username into the text field labeled username (CSCI Homepage)
4. The user enters in Login password into the text field labeled password (CSCI Homepage).
5. The user clicks the login button to submit their information (CSCI Homepage)
6. The website validates the information entered into the username and password fields (CSCI Homepage)
7. The website loads with the user's relevant information – i.e. class that they are a part of their repo and user status

8. The navigation bar at the top of the screen now displays a web pages the user can access

Alternate Flows:

1. Alternate Flow 1: The user gets to the website by accessing an external link which may lead to the homepage
2. Alternate Flow 2: The user's internet connection drops before the website loads in which they cannot access the website until connection is restored and they refresh the page
3. Alternate Flow 3: The user's internet connection drops after only part of the website has loaded in which parts of the homepage are visible but maybe not all.

4 INTERFACE REQUIREMENTS

This paragraph specifies the requirements, if any, imposed on interfaces internal and external to the software system.

4.1 External Interface Requirements

This paragraph specifies the requirements, if any, imposed on data external to the software system. Included are requirements, if any, on databases. Review the Use Case Diagram to identify those entities (Actors or other Components) outside the boundary of the Component. The external interfaces are derived from the interactions between these entities. All interfaces with Actors will require User Interfaces and interfaces with other Components will require Component Interfaces

The requirements imposed on the external use cases include

- User form interaction for a professor to create a class environment
- Assignment creation for a professor within the class environment
- Announcement creation for a professor within the class environment
- Grading for a professor within the class environment
- Uploading files for a professor to interact within the classroom creation environment
- Displaying File within code collaboration environment
- Pulling class listing and roster to display attendance in virtual classroom environment
- Pulling of forum information to display within forums environment
- Pulling folders and files within users repository to display

4.2 Internal Interface Requirements

This paragraph specifies the requirements, if any, imposed on data internal to the software system. Included are requirements, if any, on databases and data files to be included in the software system.

- Validating Student/TA status within Code Collaboration environment to determine use eligibility
- Verifying student login credentials
- Verifying student information within the the virtual classroom environment
- Store the information for the creation of a class environment
- Store files within active users repository
- Creation of new forum
- Creation of new post within forum

5 NON-FUNCTIONAL Requirements

This section consists of non-functional requirements.

5.1 Security and Safety Requirements

All software delivered is to be tested and meet the following requirements for security and safety.

- Secured against SQL Injection
    - Countermeasures implemented in all php calls to database
- Secured against Cross-Site Scripting - XSS
    - Implement built-in htmlentities() function that php offers
- Systems in place to prevent brute force attacks
    - Countermeasures implemented to prevent multiple login attempts
- Secure against file inclusion vulnerability
    - Implement all PHP to check for unvalidated user-input
- Implement logging of all users including information on
    - IP Address
    - Last logged on
    - Incorrect login attempts (IP and DateTime)
- Logging of errors
    - Most PHP and server side error logging will be handled by the web server (apache or nginx)

5.2 Quality Requirements

All software delivered will meet the following quality and testing requirements.

- The website will be developed using an Object Oriented Design paradigm where applicable, and according to W3C standards.

- All PHP code delivered will be error and warning free, confirmed through web server logs
- All features will be tested prior to delivery by external non developers.
- All HTML and CSS will be run through and W3C HTML CSS validator to ensure that html code meets requirements and will return no errors.
- Javascript coding according to JavaScript: Google style guide
- The entire site will have a consistent design theme, utilizing the same color schemes and layout
- The website should be able to run on major desktop web browsers without error (Chrome, Safari, Firefox)

6 Requirements Traceability AND VERIFICATION

Traceability from each requirement in this specification to use cases it addresses.

This section also defines a set of verification methods and specifies for each requirement the method(s) to be used to ensure that the requirement has been met. A table may be used to present this information, or each requirement may be annotated with the method(s) to be used. Qualification methods may include:

Analysis:
The processing of accumulated data obtained from other qualification methods. Examples are reduction, interpretation, or extrapolation of test results.

Inspection:
The visual examination of software system code, documentation, etc according to coding standards of project and of W3C.

Demonstration:
The operation of the software system, or a part of the software system, that relies on observable functional operation not requiring the use of instrumentation, special test equipment, or subsequent analysis.

Test:
The operation of the software system, or a part of the software system, using instrumentation or other special test equipment to collect data for later analysis. Special test equipment would include W3C HTML CSS validator.

Use Case	Requirement #	Validation Method (A, I, T, or D)
Classroom Creation	1	I, D, T
	2	I, D, T
	3	I, D, T
	4	I, D, T
	5	I, D, T
	6	I, D, T
	7	I, D, T
	8	I, D, T
Code Collaboration	1	I, D, T
	2	I, D, T
	3	I, D, T
Virtual Classroom	1	I, T
	2	I, D, T
	3	I, T
	4	I, T
	5	I, D,T
	6	I, D, T
	7	I, D, T
	8	I, D
	9	I, D, T
	10	I, D, T
	11	I, D, T
	12	I, T
Code Repository	1	I, T
	2	I, D, T
	3	I, D, T
Forum	1	I, D, T
	2	I, D, T
	3	I, D, T
	4	I, D, T

7 NOTES

7.1 Assumptions

1. The current CSCI website is old and is in need of improvement

7.2 Abbreviations and Acronyms

UML	Unified Markup Language
HTML	Hyper Text Markup Language
UI	User Interface
CSCI	Computer Science

7.3 Definitions

The following requirements and/or design definitions have been made.

Actor	An external entity (either human or machine) that interacts with the Component
Actor Descriptions	A description of the Actor which defines their goals and their relationship with the Component
Component	The system or part of the system that is within the scope of the analysis activities
Component Interface	A type of external interface that specifically addresses the interfaces between two Components
Pre-condition	The set of conditions that must be met prior to the

	scenario being carried out. Events to satisfy the pre-conditions should not be present within the scenario
Post Condition	The set of conditions that must be met in order for the scenario to be considered complete. The events within each scenario must allow for the post-conditions to be completed
Use Case	A set of Scenarios that describe the way in which an Actor may interact with the Component in order to achieve their goal
User Case Diagram	A pictorial view that shows the relationship between Use Cases. Useful for assess the complexity of the behaviors of a Component
User Interface	A type of external interface that specifically addresses the interfaces a Component and a human Actor