CPEN321: Software Engineering

Test Plan Document

2nd December 2016

Group: Cyann

Chen Chen (3281 6143)
Howard Zhou (4898 6137)
Justin Toh (2954 9136)
Luvian Wang (1693 9134)
Xi Chu (1735 3137)
Yufei Qiao (4885 7130)

Table of Contents

1. Introduction

Cyann is intended to be an learning management system that provides quality user experience. To ensure users can get or post question and comment the backend API endpoints needs to be thoroughly tested.

2. Verification Strategy

We will conduct hallway usability test with random population of students by presenting them a functional product to get feedback of user experience and how the UX compare to other LMS currently on the market.

3. Non-Functional Testing and Results

Refer to the entries highlighted in RED in section 6.

4. Functional Testing Strategy

4.1 Testing Strategy (Frontend)

Since there is no need to test UI fragment individually(they tend to change with software update and maybe represented differently on different OS and devices). The frontend developers will be working on the test together as a team after the UI implementation is completed to ensure quality UX by performing hallway usability testing.

To ensure the accuracy of UI fragment(each component does what it intends to do), the frontend team will be using Appium for its UI automation feature.

Our testing approach consists of simulating onPress event using Appium's automation feature and print the retrieved data in JSON format and compare them against the expected outcome.

4.2 Testing Strategy (Backend)

Every developer is in-charged of writing manual tests for the features/task that they are assigned since each of them would be familiar with the code that they write.

- We will be using **Postman** to write manual tests for our API endpoints since it has an intuitive GUI & a low learning curve for the subteam.
- These tests will then be executed using **Postman's** built-in test runner.

To make sure the each feature/API endpoint is interacting together correctly as intended, the backend subteam will work together to write integration tests.

- These test will be written using **mocha, expect.js & supertest**
- We use a BDD (Behaviour Driven Development) approach in our test cases so that we can test that our backend system meets our requirements & use cases.

Our main testing approach would be:

- Make API calls to our backend server followed by asserting that the JSON responses match our expected values.
- Use **Istanbul** to measure our test coverage.
- Meet at least **80% overall test coverage** by our release (demo day).
- The manual tests will (ideally) have to pass before every commit.
- Integration tests will (ideally) have to pass before merging any feature branches to the MASTER branch.

A feature is only considered as done when it meet all the written specs, passes all the tests & merged into MASTER branch.

4.3 System testing strategy

Systems level would consist of performing tests on every use cases via our app being deployed on a mobile device & the backend API server running live on a cloud. This is to simulate a real user interacting with our product so that we can catch all the potential bugs that a user might encounter.

System testing will be done at the end of every sprint, making sure that all bugs are caught and resolved before moving on the the next sprint.

4.4 Bug tracking strategy

Our bugs will be actively tracked using Github's issue tracker so that the team can easily access it and edit them in real-time. We will consider migrating over to a formal bug-tracking tool once we have completed our backlog task ahead of time.

Bugs will be categorized into 3 classes:

- **A: Affects the app's basic functionality & renders the app not usable.**
    - To be fixed ASAP.
    - Eg. Backend's API route not working.
    - Eg. List isn't being rendered on app.
- **B: Doesn't affects the app's basic functionality, app is usable but user experience is greatly affected.**
    - To be fixed by the end of the current sprint.
    - Eg. API Calls taking > 1sec due to slow DB queries.
    - Eg. App's font size is too large, causing text to be chopped off at the end.
- **C: Doesn't affects the app's basic functionality, can be resolved at a later date**.
    - To be fixed by the product's release date.
    - Eg. Title field is out of alignment by 1px.
    - Eg. Button color inconsistent with Hi-fi mockup.
    - Eg. Keys in API's JSON response contains spelling mistake.

5. Adequacy Criterion

5.1 Adequacy criteria for the frontend system

- Each UI component will be tested for alignment on different devices (iOS/Android with different screen size)
- Ensure that each UI component will perform its intended usage, for example: pop to a new scene / calling API endpoint.

5.2 Adequacy criteria for the backend system

- Make sure that at least one test exists for each API endpoint.
- Based on how our backend system is designed, integration tests on API endpoints would be sufficient to detect all possible bugs that unit tests can detect.
- Once an API endpoint manages to pass all it’s integration tests, we assume that all methods called during that process is functioning as intended.

6. Test Cases and Results

6.1 Frontend Tests(*verification*) (Last Updated on 2nd December)

Test #	View	Scenario	Expectation	Result
1	preLogin.js	Facebook never logged in and no jwt stored locally	Transition to login.js	P
2		Facebook logged in and jwt stored locally	Transition to courseList.js	P
3		Facebook not logged in and jwt stored locally	Transition to login.js	P
4		Facebook logged in and no jwt stored locally	Would never happen	-
5	courseList.js	During componentDidMount()	/api/users/my/courseData called	P
6		No course registered	Present user with searchbar	P
7		Course(s) registered	Present user with course card view	P
8		Plus icon touched	Present user with searchbar	P
9		Users icon touched	Present user with a list of user registed in the course	P
10		Close icon touched	dropCourse modal appears	P
11		“Yes” button on dropCourse modal touched	/api/courses/removeUser/:courseId called, modal closed	P
12		“No” button on dropCourse modal touched	Modal closes	P
13		“Course Card” touched	Transition to course.js	P
14		During componentWillMount	/api/courses/:courseId/posts called	P
15		During componentDidmount	/api/users/my/posts called	P
16		During componentDidmount	/api/users/my/	P
17		During componentDidmount	/api/users/my/comments	P

18	Course.js	Icon on tabbar touched	Transition to corresponding scene	P
19		Swipe right action performed	Transition to next scene	P
20		Search Actionbutton touched	searchModal appears	P
21		A post's title is touched	Post content appears below it	P
22		A post's content is touched	/api/courses/:courseId/posts/:postId called, transition to viewQuestion.js	P
23		Question title textbox touched	Keyboard appears	P
24		Question content textbox touched	Keyboard appears	P
25		Ask button touched	/api/courses/:courseId/posts called	P
26		Ask button touched while question title and question content are null	titleBodyEmptyModal appears	P
27		Readings icon touched	/api/:courseId/files/:type called	P
28		Assignment icon touched	/api/:courseId/files/:type called	P
29		File button touched	/api/:courseId/files/:type/download, transition to fileView.js	P
30		"Past post" text touched	A list of user's past post appears	P
31		"Past comment" text touched	A list of user's past comment appears	P
32		"Credits" text touched	Credits modal appears	P
33		"Log out" text touched	Facebook Logged out, transition to preLogin.js	P
34	fileView.js	"Back" button pressed	Transition back to Course.js	P
35	viewQuestion.js	During componentWillMount	A list of comments rendered	P
36		A comment preview touched	Whole comment rendered in text area	
37		"Write" button pressed	Keyboard appears	P
38		"Thumb up" icon pressed	api/courses/:courseId/posts/:postId/comments/:commentId/upvote called	P
39		"Back" button pressed	Transition to course.js	P
40	login.js	"Log in" button pressed	Facebook logged in window appears	P

6.2 Backend Tests (Last Updated on 1st December)

Original Spreadsheet:
https://docs.google.com/spreadsheets/d/1qWzdspZrvEtCupBfAei_bQMUa_PA3VnyO9o2_xTClBo/edit?usp=sharing

#	API Route	Test Description (BDD)	P/F	Notes
1	[GET] /api/honor/:userId	should return UNAUTHORIZED error (missing JWT)	P	
2		should return the user's honor points	P	
3		should NOT return the user's honor points (invalid :userId)	P	
4		should NOT return the user's honor points if user is an instructor	P	
5	[POST] /api/users/register	should return a JWT Token (new user)	P	
6		should return a JWT Token (existing user)	P	
7		should NOT return a JWT Token ("email" param missing)	P	
8		should NOT return a JWT Token ("socialToken" param missing)	P	
9		should NOT return a JWT Token ("profileImg" param missing)	P	
10		should NOT return a JWT Token (invalid socialToken)	P	
11		should NOT return a JWT Token (expired socialToken)	P	
12	[GET] /api/users/my	should return UNAUTHORIZED error (missing JWT)	P	
13		should return user's profile data (userType == student)	P	
14		should return user's profile data (userType == instructor)	P	
15	[GET] /api/users/my/posts	should return UNAUTHORIZED error (missing JWT)	P	
16		should return user's list of created posts (user has created a post)	P	

17		should return user's list of created posts (user has never created a post)	P	
18	[GET] /api/users/my/comments	should return UNAUTHORIZED error (missing JWT)	P	
19		should return user's list of created comments (user has created a comment)	P	
20		should return user's list of created comments (user has never created a comment)	P	
21	[GET] /api/users/my/courseData	should return UNAUTHORIZED error (missing JWT)	P	
22		should return user's list of registered courses (user has joined a course as a student)	P	
23		should return user's list of registered courses (user has not joined a course as a student)	P	
24	[GET] /api/courses	should return UNAUTHORIZED error (missing JWT)	P	
25		should return a list of all courses (40	P	
26	[GET] /api/courses/:courseId	should return UNAUTHORIZED error (missing JWT)	P	
27		should return course data	P	
28		should return course data (DB doesn't have entry corresponding to :courseId)	P	
29	[GET] /api/courses/users/:courseId	should return UNAUTHORIZED error (missing JWT)	P	
30		should return list of users registered to course	P	
31	[POST] /api/courses/	should return UNAUTHORIZED error (missing JWT)	P	
32		should not allow non-instructors to create a course	P	
33		should allow an instructor to create a new course	P	
34	[PUT] /api/courses/addUser/:courseId	should return UNAUTHORIZED error (missing JWT)	P	
35		should register user into the course (56	P	
36		should not add registered users into the course again (i.e. double entry)	P	
37	[PUT] /api/courses/rem	should return UNAUTHORIZED error (missing JWT)	P	

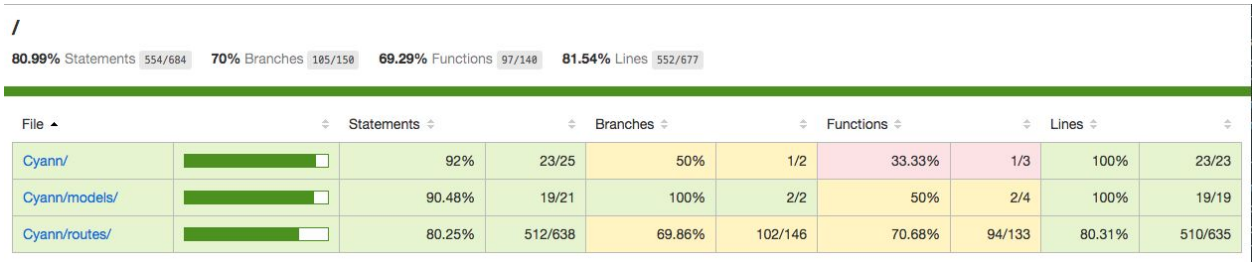
38	oveUser/:courseId	should deregister user from the course	P	
39	[PUT] /api/courses/:courseId	should return UNAUTHORIZED error (missing JWT)	P	
40		should not allow non-instructors to modify a course	P	
41		should allow an instructor to modify a course	P	
42	[GET] /api/courses/:courseId/posts	should return UNAUTHORIZED error (missing JWT)	P	
43		should return a list of all courses	P	
44	[GET] /api/courses/:courseId/posts/:postId	should return UNAUTHORIZED error (missing JWT)	P	
45		should return post data (DB doesn't have entry corresponding to :postId)	P	
46		should return post data	P	
47	[POST] /api/courses/:courseId/posts	should return UNAUTHORIZED error (missing JWT)	P	
48		should create a post	P	
49		should create a post & send an email if an instructor created the post	P	
50		should NOT create a post ("title" param missing)	P	
51		should NOT create a post ("content" param missing)	P	
52	[PUT] /api/courses/:courseId/posts/:postId	should return UNAUTHORIZED error (missing JWT)	P	
53		should edit a post if user is the post's author	P	
54		should NOT edit a post if user is not an instructor and the post's author	P	
55		should NOT edit a post if user is an instructor but not the post's author	P	
56		should NOT edit a post ("title" param missing)	P	
57		should NOT edit a post ("content" param missing)	P	
58	[DELETE] /api/courses/:courseId/posts/:postId	should return UNAUTHORIZED error (missing JWT)	P	
59		should delete a post if user is the post's author	P	
60		should delete a post if user is an instructor (40	P	

61		should NOT delete a post if user neither an instructor nor the post's author	P	
62	[GET] /api/courses/:courseId/posts/:postId/comments	should return UNAUTHORIZED error (missing JWT)	P	
63		should return a list of all comments	P	
64	[GET] /api/courses/:courseId/posts/:postId/comments/:commentId	should return UNAUTHORIZED error (missing JWT)	P	
65		should return a comment	P	
66		should return a comment (DB doesn't have entry corresponding to :commentId)	P	
67	[POST] /api/courses/:courseId/posts/:postId/comments	should return UNAUTHORIZED error (missing JWT)	P	
68		should create a comment	P	
69		should NOT edit a comment ("content" param missing)	P	
70	[PUT] /api/courses/:courseId/posts/:postId/comments/:commentId	should return UNAUTHORIZED error (missing JWT)	P	
71		should edit a comment if user is the comment's author	P	
72		should NOT edit a comment if user is not an instructor but the comment's author	P	
73		should NOT edit a comment if user is an instructor but not the comment's author	P	
74		should NOT edit a comment if user is a TA but not the comment's author	P	
75		should NOT edit a comment ("content" param missing)	P	
76	[DELETE] /api/courses/:courseId/posts/:postId/comments/:commentId	should return UNAUTHORIZED error (missing JWT)	P	
77		should delete a comment if user is the comment's author	P	
78		should delete a comment if user is an instructor	P	
79		should NOT delete a comment if user neither an instructor nor the comment's author	P	
80		should NOT delete a comment if user neither an instructor nor the comment's author but a TA	P	
81	[PUT] /api/courses/:courseId/posts/:po	should return UNAUTHORIZED error (missing JWT)	P	

82	stId/comments/:commentId/setAsAnswer	should set a comment as answer if user is an instructor	P	
83		should set a comment as answer if user is a TA	P	
84		should NOT set a comment as answer if user is the comment's author	P	
85		should NOT set a comment as answer if user is not the comment's author	P	
86		should STAY set if a comment is already set as answer	P	
87	[PUT] /api/courses/:courseId/posts/:postId/comments/:commentId/unsetAsAnswer	should return UNAUTHORIZED error (missing JWT)	P	
88		should unset a comment as answer if user is an instructor	P	
89		should unset a comment as answer if user is a TA	P	
90		should NOT unset a comment as answer if user is the comment's author	P	
91		should NOT unset a comment as answer if user is not the comment's author	P	
92		should STAY unset if a comment is previously not set as answer	P	
93	[PUT] /api/courses/:courseId/posts/:postId/comments/:commentId/upvote	should return UNAUTHORIZED error (missing JWT)	P	
94		should upvote a comment if user is an instructor	P	
95		should upvote a comment if user is a TA	P	
96		should upvote a comment if user is not the comment's author	P	
97		should NOT upvote a comment if user is the comment's author	P	
98		should NOT upvote a comment if user has already voted	P	
99	[PUT] /api/courses/:courseId/posts/:postId/comments/:commentId/resetVote	should return UNAUTHORIZED error (missing JWT)	P	
100		should reset the votes of a comment if user is an instructor	P	
101		should reset the votes of a comment if user is a TA	P	
102		should reset the votes of a comment if user is not the comment's author	P	
103		should NOT reset the votes of a comment if user is the comment's author	P	

104		should NOT reset the votes of a comment if user has not voted yet	P	

6.3 Backend Test Coverage (Last Updated on 1st December)



/

80.99% Statements 554/684 **70%** Branches 105/150 **69.29%** Functions 97/140 **81.54%** Lines 552/677

File		Statements		Branches		Functions		Lines	
Cyann/		92%	23/25	50%	1/2	33.33%	1/3	100%	23/23
Cyann/models/		90.48%	19/21	100%	2/2	50%	2/4	100%	19/19
Cyann/routes/		80.25%	512/638	69.86%	102/146	70.68%	94/133	80.31%	510/635

6.4 System/Acceptance Tests(*validation*) (Last Updated on 2nd December)

A	B	C	D
First Name (or Anonymous)	Date of Test	Feedback	
Alicia	Nov 25, 2016	1. Keyboard will mistakenly pop up when I swipe from question posting to file display.	
		2. In file reading mode, "back" button should be at the top left corner, consistent with other scenarios.	
		3. There is a "x" botton not working.	
John	Nov 25, 2016	1. Search button should be placed at the bottom of the page as a bar and can swipe up.	
		2. On the first page, put "+" button on the top right corner of each course, instead of a "menu" button.	
		3. Display animation for each page not just for one time.	
Juliana	Nov 25, 2016	1. Not sure which tab I'm on	
		2. Not sure if looking at past post and past comment	
		3. Past comments doesn't show title	
		4. Navigating is a little confusing	
		5. Button should just be a plus sign or repositioned	
Fred	Nov 25, 2016	1. Make course cards bigger	
		2. Backbutton	
		3. Search bar on main page	
		4. New post notification(backend)	
		5. Unread tag(backend)	
	The app has made adjustments accordingly		
	The feedback has not been considered yet		

Our acceptance/user validation test was performed November 25th with six test users, while four of them recorded their suggestion. While some of the suggestions requires changing our database scheme, therefore not taken into consideration, we still managed to change most of the suggestion related to our UI.

Please see a list of use case tested below, for a complete user flow for each use case, please refer to our requirement document located under **docs/**

#	Description	Tested(Y/N)	Live up to user's expectation (Y/N)	Note
1	User Authentication(WEB)	N	-	Unable to perform as webend isn't finished with implementation
2	User Authentication(Mobile)	Y	Y	
3	User creates a course	N	-	Unable to perform as action requires instructor privilege
4	User updates a course	N	-	Unable to perform as action requires instructor privilege
5	User registers a course	Y	Y	
6	User creates a post	Y	Y	
7	User edits a post	Y	Y	
8	User deletes a post	Y	Y	
9	User creates a comment on post	Y	Y	
10	User edits a comment	Y	Y	
11	User deletes a comment	Y	Y	
12	User sets a comment as answer	N	-	Unable to perform as action requires instructor privilege
13	User unsets a comment as answer	N	-	Unable to perform as action requires instructor privilege
14	User upvotes a comment	Y	Y	
15	User removes vote on a comment	Y	Y	
16	User searches for posts	N	-	Unable to perform as the functionality isn't implemented on the frontend
17	User uploads a file	N	-	Unable to perform as action requires instructor privilege
18	User reads a file	Y	Y	
19	User deletes an uploaded file	N	-	Unable to perform as action requires instructor privilege
20	User views honor points leaderboard	Y	Y	
21	User views another user's profile	N	-	Unable to perform as the functionality isn't implemented on the frontend
22	User contacts other user for help	Y	Y	