32386120 | Alec Ng
32136111 | Justin Lane
45909116 | Aviral Garg
31580129 | Stephen Hu
90294133 | Yue (Mark) Chen
15991152 | Anh Duc (Andrew) Bui
54952130 | Tushar Kalra

CodePal - Testing and Validation

Introduction

CodePal is a web based application which allows its users to write compilable code in its embedded IDE while watching Youtube tutorials and searching StackOverFlow posts all in the same browser window. The application is going to allow users to switch between these three core functionalities with ease and also allow them to resize and restructure their relative positions. The main focus for testing the application goes into running Unit tests to see if each of these three functionalities are running their specific tasks correctly while also making sure the UI is responding without lag to users new restructuring and resizing attempts by getting feedback from the users. Another area of focus is on making sure that the user authentication is working correctly as the users have the option to save their code if they choose to.

Verification Strategy

In order to ensure the the software meets the user's real need, verification will be done on a regular basis. We would get feedback from users who have technical backgrounds and also those who do not. Verification of the user interface will be done on a weekly schedule where the design and selected theme is presented to users for approval. Users are provided with GUI mockups as well as prototypes of the software itself to use and feedback is received. Software specifications and requirements will be reviewed with the users to ensure that everything is correct. Changes to the specifications or requirements are made during these reviews, prior to development to ensure an efficient development process.

Non-Functional Testing and Results

The non-functional requirements are mainly focussed on performance, usability, and security. For testing non-functional requirements, users will be given access to CodePal hosted on Github Pages. Users will be given the inputs mentioned below and their feedback will be compared to the expected outputs.

To test for performance, the system will be placed under different workloads and its efficiency will be compared each time to ensure that the system performs at the quickly under different circumstances.

To test for security requirements, calls to the API will be made with the correct and incorrect access tokens to ensure privileged API calls are restricted to the appropriate users.
To test for usability, different first-time users will be given access with different combinations of features and feedback would help us point out usability issues to specific components of the system.

Test #	Requirement	Input	Expected Output
1	**Performance:** A fast and fluid UI with maximum features	Multiple components being used simultaneously (video playing, compiling code in the code editor)	The components run without delay (no lag)
2	**Security Requirements**: Privileged calls to the API should be restricted to the respective roles	Logged-in user attempts to view another user's private snippet	The private snippet created by the other user should not be accessible by the current user
3	**Usability**: An interactive UI/UX so that users can use the product more effectively.	Access to CodePal website	They are easily able to accomplish basic tasks (changing layout, compiling code) the first time they encounter the design.

Functional Testing Strategy

Unit testing will be the crux of our testing strategy. Every feature should be accompanied by a set of unit tests validating the correct function of expected inputs, boundary cases, and exceptions. Test cases are created based on a pairwise testing approach for large inputs, and additional test cases will target the business requirements of the feature and the given scenario.

These test suites also as a regression suite, as they will be run and be required to pass before introducing any additional code commits. This ensures the original functionality is not broken with new releases.

For more involved interactions between services, integration test suites will also be made to ensure the correct functionality between our upstream dependencies, including Facebook Login, Ace Editor, and HackerEarth.

For bug tracking/handling, we will be using a simple spreadsheet split into modules (YouTube, Code Editor, Backend, etc.) On the spreadsheet there is indication of: Date of entry, bug description, severity, and status for comments on the progression towards fixing said bug. Any member of the team can add or resolve bugs in the database.

Severity will be split into the categories of

- Severe - this bug is interrupting core functionality
- Warning - non-critical bug that may be ignored for the time being, but should be fixed before delivery
- Shippable - trivial effect on functionality, may not be worth the time spent fixing

Adequacy Criterion

At a basic level, every scenario should have a corresponding test case. The edge cases, and exceptions relating to that scenario should be covered as well.

The code coverage targets we aim to meet is 65% branch coverage, and 70% line coverage. Given our resources, we believe these are adequate testing targets to instill confidence in the functionality of our system in terms of the conditions tested, and the comprehensive core codebase coverage.

Test Cases and Results

API Snippet Tests

Test #	Requirement	Input	Expected Output
1	Logged-in user should be able to create a snippet	Logged-in user clicks on "Create Snippet"	A new snippet is created for the logged-in user
2	Logged-in user should be able to view a snippet they have created	Logged-in user clicks on the title link to a previously-created snippet	The logged-in user should be presented with a read-write view of the snippet they created
3	Logged-in user should be able to edit a snippet they have created	Logged-in user clicks on "Update Snippet" on a previously-created snippet	The logged-in user's existing snippet should have its contents updated with the changes they made to the snippet
4	Logged-in user should be able to delete a snippet they have created	Logged-in user clicks on "Delete Snippet" on a previously-created snippet	The logged-in user's snippet should be removed

Code Editor Tests

Test #	Requirement	Input	Expected Output

5	Syntax Highlighting is enabled for each supported language	User switches input language for current tab from select element	Code in editor has syntax highlighted, showing errors and suggestions according to language chosen
6	Ace Editor theme is changed on demand	User switches theme for all tabs from select element	Editor's theme changes according to theme chosen
7	Current tab contents are able to be downloaded locally	User inputs some code in the editor, types a name in the "filename" input and presses "Download"	File with extension matching the chosen language is saved locally onto the user's machine
8	Current tab is saved as a code snippet	User types a name in the "filename" input and presses "Save"	Snippet is saved to db via REST API, and user is faced with success message. Tab name changes from 'untitled' to the chosen name
9	Given a valid program, code is compiled and executed	User types in a compilable, runnable program in the language they have chosen and presses "Run"	The output shows the time taken to compile and run, and any stdout that results in running the program
10	Given a program with a syntax error, the output shows the error	User types in a non-compilable or non-runnable program that includes using a non-declared variable, and presses "run"	The output shows the time taken to compile and run, and shows the line number and specific cause of error that caused the failure
11	Given a program that runs for more than five seconds, shows a timeout error	User types in a compilable, runnable program that has a while loop that never terminates, and presses "run"	After 5 seconds, the output shows the time taken to compile and run, and shows a timeout error
12	Given an unexpected API issue, show a generic error message	User types in any sort or code in the editor, is not connected to the internet, and presses "run"	The output shows a generic error asking the user to try again.
13	Given a program with a programmatic error (e.g. seg fault, trying to access as private method), show the cause of the error	User types in a compilable, runnable program that contains a language specific programmatic error. For this case, a C program that accesses an index of an array that is out of bounds, and presses "run"	The output shows the time taken to compile and run, and shows the line number and failure cause.
14	Create a new tab	User presses the '+' button on top of the code editor	A new tab is created named 'untitled' with no contents, and the language for this tab is the same as the previous tab the user was on

15	Switch between existing tabs	User creates two tabs using the new tab button. User is on tab 2, and presses tab 1.	Focus switches to tab 1 and the contents are switches from showing tab 2 code to tab 1 code
16	Delete current tab	User has at least 2 tabs open and presses the delete ‘X’ button on one of them	If there is code present in the deleted tab and it is not saved, display a warning message. When deleting the tab, it is removed from the UI and last created tab is brought into focus
17	Modify ace editor settings	User presses the ‘custom settings’ button, inputs a custom tab size and font size and presses ‘save’	Changes are propagated to the code editor, and if the user is logged in, the settings are saved to the db via REST API
18	Search and replace for a phrase in the editor	User presses ‘Ctrl+F’ and types in a phrase that exists in the code editor	The line with the phrase is highlighted. For each successive search press, other phrases that exist are subsequently highlighted
19	Logged in user has all saved defaults restored upon page load	User logs in from the front page	If the user had previously saved saved settings, they are automatically restored.

Youtube Tests

Test #	Requirement	Input	Expected Output
20	Resizing does not cause the component to reload	User changes the size of the youtube component	The video continues to play and does not reload
21	Altering the layout does not cause the component to reload	User click and drags the component to another location	The video continues to play and does not reload
22	Search for videos	User searches for a topic in the search bar	A list of videos is displayed for the user to select from

StackOverflow Tests

Test #	Requirement	Input	Expected Output
23	Search for questions	User inputs keywords, tags or queries	Return up to 15 questions displayed on the pane
24	Search for new questions	User inputs new keywords, tags or queries	The old result will be discarded and return other new 15 questions displayed on the pane
25	Close StackOverflow pane	User chooses to close the pane	The pane should disappear
26	Select returned questions	User clicks on the question that he/she wants to view	A new tab should be opened and directed to the original StackOverflow page