YouDanMu

Sprint 1

Incremental Testing and

Regression Testing Log

Team Member: Naiwei Zheng, Yucong Ma,

Yibo Gou, Ge Yan,

Jiaqi Zhu

1 Classification Of Components

1.1 Define Components

a. Incremental Testing Form

We use Top-down form of incremental testing since our modules are highly dependent to each other and our plan of implementation of project are top-down, meaning many of the sub-modules are not finished in Sprint 1, it is easier to design and implement for us to perform Top-down testing rather than bottom-up testing.

b. Modules

i. YDM

Main entrance of the program, the realization and initialization for all other modules, communicate with Browser Extension to listen events and assign action to each modules

- Input: Events from Video Providers
- Output: Events to signal other modules
- Parent Dependency: None
- Child Dependency: Danmaku Providers, Danmaku Timeline, Renderer, Video Providers, Danmaku, Browser Extension

ii. Danmaku Timeline

Store the timeline of Danmaku and host function that perform play, pause, speed change of the timeline. Communicate with Renderer as the data source of contents to be rendered

- Input: List of Danmaku fetched from Danmaku Providers
- Output: Animation frame update events
- Parent Dependency: YDM, Danmaku Providers
- Child Dependency: Danmaku

iii. Danmaku Providers

An interface to be implemented to be classes for loading and parsing the sources of danmaku.

- Input: Video identifier
- Output: List of Danmaku from the specific Danmaku source
- Parent Dependency: YDM
- Child Dependency: Danmaku

iv. Video Providers

An interface to be implemented to be classes that store the information of current video player states and events

- Input: Third party events (e.g. YouTube internal events)
- Output: Common YDM events
- Parent Dependency: YDM

- Child Dependency: Third party interface (e.g. YouTube API)

v. Browser Extension

The outward extension of the plugin to communicate with browser, hook up events listeners, inject javascript code and interact with DOM elements.

- Input: Events from browser
- Output: Common YDN events
- Parent Dependency: YDM
- Child Dependency: Browser API

vi. Danmaku

The class to be instantiated to store a single danmaku content, styles and state

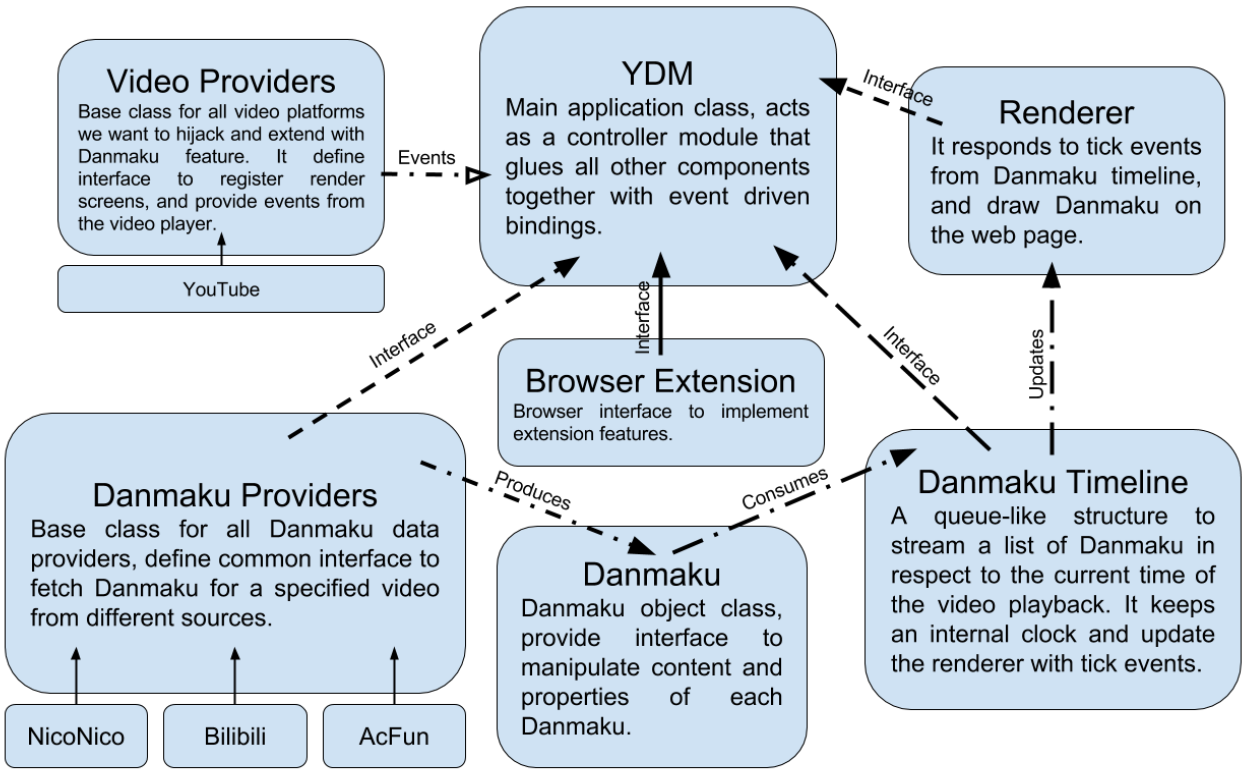
- Input: Danmaku content and properties
- Output: Danmaku object encapsulate the DOM element
- Parent Dependency: YDM
- Child Dependency: None

vii. Renderer

Convert our data to DOM elements that are rendered by the browser. React to resize and state change of the player.

- Input: Danmaku Timeline tick events
- Output: Draw Danmaku on the web page
- Parent Dependency: YDM, Danmaku Timeline
- Child Dependency: Danmaku

c. Diagram



1.2 Which form of incremental testing did you follow

For our incremental testing, we used top-down testing with the help of stubs. This is because most of the leaf-node modules in our project are depending on the environments, and should integrate with third party modules. Such as the Danmaku Providers classes, they invokes API from different websites but eventually returns a common data type to our other modules. The YouTube API hijack the YouTube API and listen to web page events but eventually provides a set of common events to other modules. With top-down testing, we can replace these leaf-node environment depending modules with dummy stubs that only responds to the common interface, without actually invoking their party interfaces. This way, we can free our testing environment from the actual production environment, for instance, we can test most of the modules merely under the command line, event without running a browser.

2 Incremental and Regression Testing

2.1 Automation

We do use automation to assist our incremental and regression testing. Since our source code is written in TypeScript, we want to benefits from the type checking and IDE static code analysis by writing our test cases also in TypeScript. Thus we need to compile our test scripts along with our source code. To do this we used Gulp.js, a stream-interfaced asynchronous JavaScript task runner; node-typescript, a package that stream the source files to the TypeScript compiler, and several other toolings.

We choose Jasmine.js as our test framework, with assistance with Chai.js as an environment independent assertion library, Mocha.js as the task runner in browser environment, and coverall.js as the code coverage reporter.

The testing strategy is to write a set of test cases for every single module. We refactored our modules into single files, and we can dynamically substitute other modules with dummy stubs with a static testing interface. This way we can just import a module, replaces other modules with stubs, then write assertion with set of test cases to conduct testing on a single module.

For regression testing, we wrote a Gulp job that watch on source code changes, with each change, it automatically rerun the compilation and testing process, and produce code coverage report. Thus every fix will be tested immediately and the automation process would efficiently assist the development iteration.

2.2 Defect log

* Severity: 1 – Workaround 2 – Important 3 – Critical

Module	YDM

Incremental Test Log

Defect No.	Description	Severity	How to correct
1	The plug-in sometimes will crush if user turn video into fullscreen.	3	Create a case in resize event function to handle this situation.
2	After the video reloaded, the event listener stop functioning	2	Listen more reload events rather than solely refresh event. Re-hook the event listeners each time the reload event happens
3	After switch video, Danmaku for previous video will still be played.	2	After switching video, identify the video.

Regression Test Log

Defect No.	Description	Severity	How to correct
1	After fixing reloading bug, sometimes the reload events triggered while the video is not reloaded	2	This is caused by internal crash and reload by the browser or video player. We guarded a limit of how many times the video can be reloaded, while the limit reach, the danmaku will not be reloaded and a prompt is out to the user.
2	After fixing the switch video bug. Sometimes the trial to identify video happens too often that affect the experience of the user.	2	Simplified the process of identifying video, the reload will only happens if the video is really changed

Module	danmakuProvider

Incremental Test Log

Defect No.	Description	Severity	How to correct
1	Some Danmaku cannot be downloaded due to the different requirement from different Danmaku sources(Login.etc)	1	Prompt user notification that Danmaku from certain Danmaku source cannot be loaded.
2	Some Website posted special Danmaku which written by script designed by website themselves.	1	Identify and filter those special Danmaku.

Regression Test Log

Defect No.	Description	Severity	How to correct
1	After adding identification to special danmaku, the renderer is not function to this new data	3	Fix the renderer to accept the new data type
2	The error message in the notification that used to prompt user that the danmaku cannot be download from specific source caused the parsing process to stop and the callback is never called after parser..	3	Added a return error message to the parser callback that specifically stated error why the danmaku cannot be downloaded

Module	Danmaku

Incremental Test Log

Defect No.	Description	Severity	How to correct
1	Different website has different properties for their own Danmaku which cannot be translate and recorded by our own format	1	Only parse the property that we have already identified.
2	Size ratio from Danmaku sources will cause the overlap	2	Limit the size ratio of Danmaku. When the ratio exceed certain value, ratio will be set as the value.

Regression Test Log

Defect No.	Description	Severity	How to correct
1	Size ratio are sometimes too small that cause display and calculation error to the renderer	1	Set a lower bound of the minimal size ratio

Module	Video Provider

Incremental Test Log

Defect No.	Description	Severity	How to correct
1	Danmaku start playing when advertisement is playing.	2	Identify current playing video is video content or ads.
2	When video playing speed exceed certain value(or negative number) our render will cause problem to the danmaku timeline and renderer	1	Limit the range of playing speed to values that accepted by most users.
3	Constantly rapidly hitting pause and play will cause inaccurate timeline	1	Sync the timeline with the video timeline each time the play event is triggered.

Regression Test Log

Defect No.	Description	Severity	How to correct
1	Setting the speed limit to caused calling the speed change function while the speed is not changed (while reach the lower or higher bound)	1	Added a statement to ensure the speed limit has really changed to call the speed change event.
2	If the timeline is synced to the time in the near past, it cause same danmaku being displayed twice	2	Adding a minimal difference between the timeline and the time to sync, if the limit is not reached, the timeline will not be updated. Also, if the timeline to synced to the time in the past, the renderer will clear the screen.

Module	Danmaku Timeline

Incremental Test Log

Defect No.	Description	Severity	How to correct
1	If seek time number exceed Danmaku max time will cause timeline to crash	3	Compare the max time. Conditionally seek time.
2	If Danmaku timeline loaded by user exceed video time, this will cause Danmaku continue playing after video ends.	2	Compare the video length with Danmaku timeline and delete the Danmaku which exceed the video length.
3	If Danmaku display is turned off, then be restarted, there will be no Danmaku show on the current page.	2	Notify users to refresh the pages which need to display Danmaku.

Regression Test Log

Defect No.	Description	Severity	How to correct
1	Because we cut the remaining timeline when the danmaku timeline exceed the video time. If the user modify the danmaku timeline the danmaku exceeded will not render.	2	We decided only skip rendering the danmaku instead of deleting it.

Module	Renderer

Incremental Test Log

Defect No.	Description	Severity	How to correct
1	Extremely large amount of Danmaku in the same screen will cause crush	3	Set a maximum number of Danmaku in the same screen.
2	Cannot calculate the available space	2	Update algorithm, each time we need allocate a new space, we recalculate the all the occupied space.
3	When a danmaku contains white space characters at the beginning or at the end of it. These character will be rendered and occupied screen spaces.	1	Before render the danmaku, we will trim the content of each danmaku.

Regression Test Log

Defect No.	Description	Severity	How to correct
1	Some Danmaku will not be display	1	Push those danmaku into a queue and will be displayed when there are some available space.
2	After we calculate for every Danmaku, delay will be generated by this algorithm	1	Optimize our algorithm.
3	After we use queue for Danmaku which cannot be displayed on time. Some old Danmaku will be displayed in inappropriate time	1	Set an expired time. When the Danmaku exceed valid time, this Danmaku will be set as invalid.

Module	Browser Extension

Incremental Test Log

Defect No.	Description	Severity	How to correct
1	If user run the extension during playing the video, the extension won't work.	1	Promote user to refresh the page.
2	The plugin waiting for message forever while extension cannot get the message from webpage.	2	Send timeout event to YDM and hang all operations
3	The event extension hooked has expired however the module still listening to the event.	2	Set a timeout to the event and handle it with YDM to hang all operations
4	When in one of the tabs where is plugin loaded is crashed, the module will also crashed.	2	Listen to processes.onExit event and unload all instances with YDM

Regression Test Log

Defect No.	Description	Severity	How to correct
1	Timeout event hang forever even after page reloaded	2	Add an exception to reload event to be detected after timeout happens