



TEAMMATES

12.02.2019

Ahmad Alsaif

V00790203

University of Victoria

SENG 480A

1.0 Introduction	1
2.0 Approach to Code Investigation	2
3.0 Quality Attribute Scenario:	2
3.1 Use Case Scenario	2
3.1.1 Test Driver Component Diagram	5
3.2 Growth Scenario	6

1.0 Introduction

TEAMMATES is a free web service that allows instructors and students to do peer evaluations and provide feedback to each other. TEAMMATES is a cloud-based service which runs on Google App Engine (GAE) and is used by many universities globally. In this report, I will explain two scenarios: a use case scenario and a growth scenario. For the use case scenario, I will show how Maintainability is supported by TEAMMATES' architectural design and provide some evidence. As for the growth scenario, I will show how data invasion can be manageable while they grow in number.

Before going through each scenario, I will first show the approach and steps I followed in order to gain a better understanding of the code.

2.0 Approach to Code Investigation

In this section I will explain the steps taken to understand the software architecture of TEAMMATES; and whether or not its architecture is designed in an organized and efficient way.

1. Launch and try TEAMMATES
2. Test TEAMMATES and how it works
 - a. Navigate through TEAMMATES
 - b. Watch series of tutorial videos on how the system works
3. Clone TEAMMATES project
 - a. Go through the packages
 - b. Understand library functions
 - c. Test wrong commands and how the system respond
 - i. See where the error output texts are located (e.g. in teammates/src/web/app/components/error-report.component.html)
 - d. Run some test files
4. Replicate the build environment.

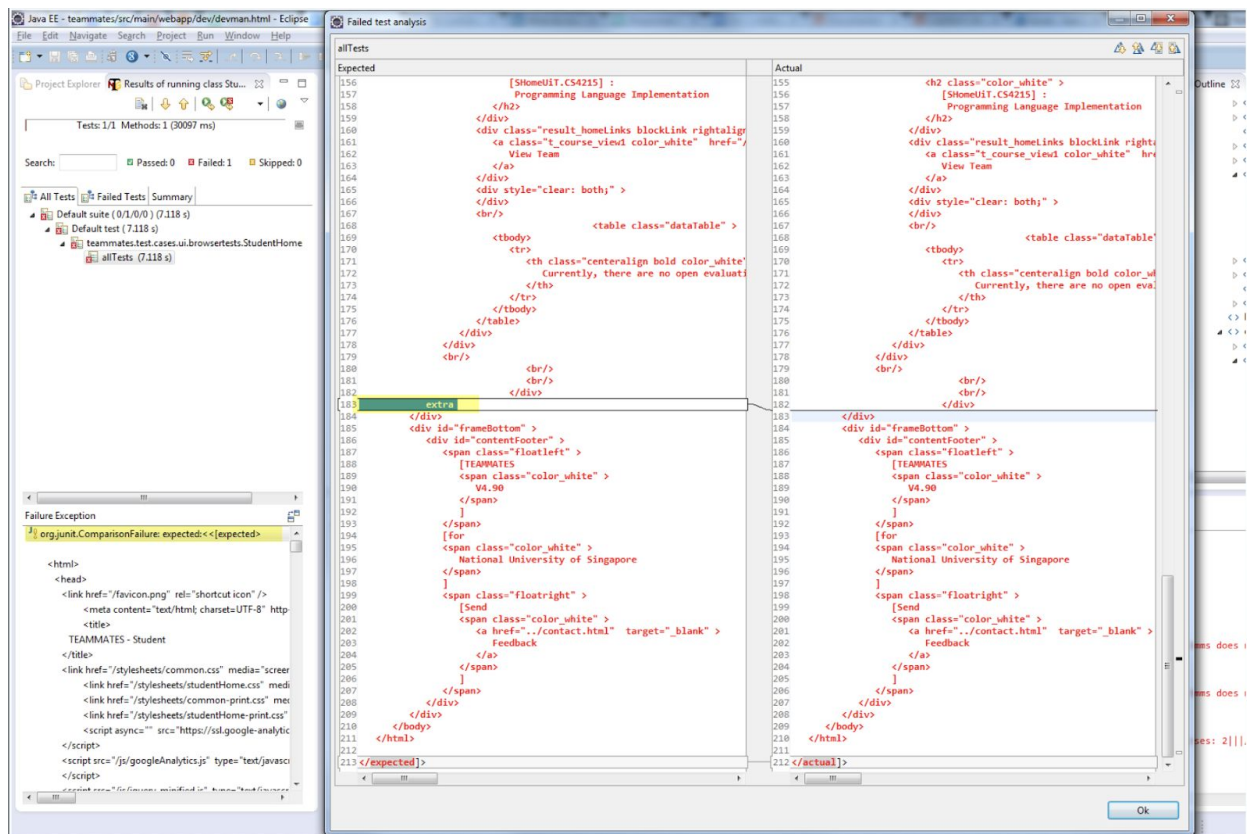
3.0 Quality Attribute Scenario:

3.1 Use Case Scenario

Aspect	Details
Scenario Name	Testing the Accuracy of the Search Engine
Business Goals	Highly maintainable code defended with tests
Quality Attribute	Maintainability
Stimulus	Test files being run on the data storage
Stimulus Source	Backend tester
Response	compares search results with expected output
Response Measure	Verifies whether or not the search results match with expected output

(Table 3.1.1: Use Case Scenario)

The table above is an example of a chosen scenario in which the stimulus is to test whether the search results are accurate or not. One of Teammates' goals is the use of fully automated regression testing. The aim of this use case scenario is to test the accuracy of the search engine. When testing the base components such as course id, instructor, student profile, etc, to see if they match with the database; it returns failure (figure 3.0). Which means the actual data is different from the expected data. A fragment of the test file used here is found in (figure 3.1) below.



(Figure 3.1.1: Test result)

```

140
141  /*
142   * Verifies that search results match with expected output.
143   * Compares the text for each comment as it is unique.
144   *
145   * @param actual the results from the search query.
146   * @param expected the expected results for the search query.
147   */
148   protected static void verifySearchResults(FeedbackResponseCommentSearchResultBundle actual,
149       FeedbackResponseCommentAttributes... expected) {
150       assertEquals(expected.length, actual.numberOfResults);
151       assertEquals(expected.length, actual.comments.size());
152       FeedbackResponseCommentAttributes.sortFeedbackResponseCommentsByCreationTime(Arrays.asList(expected));
153       FeedbackResponseCommentAttributes[] sortedComments = Arrays.asList(expected)
154           .toArray(new FeedbackResponseCommentAttributes[2]);
155
156       int[] i = new int[] { 0 };
157       actual.comments.forEach((key, comments) -> comments.forEach(comment -> {
158           assertEquals(sortedComments[i[0]].commentText, comment.commentText);
159           i[0]++;
160       }));
161   }

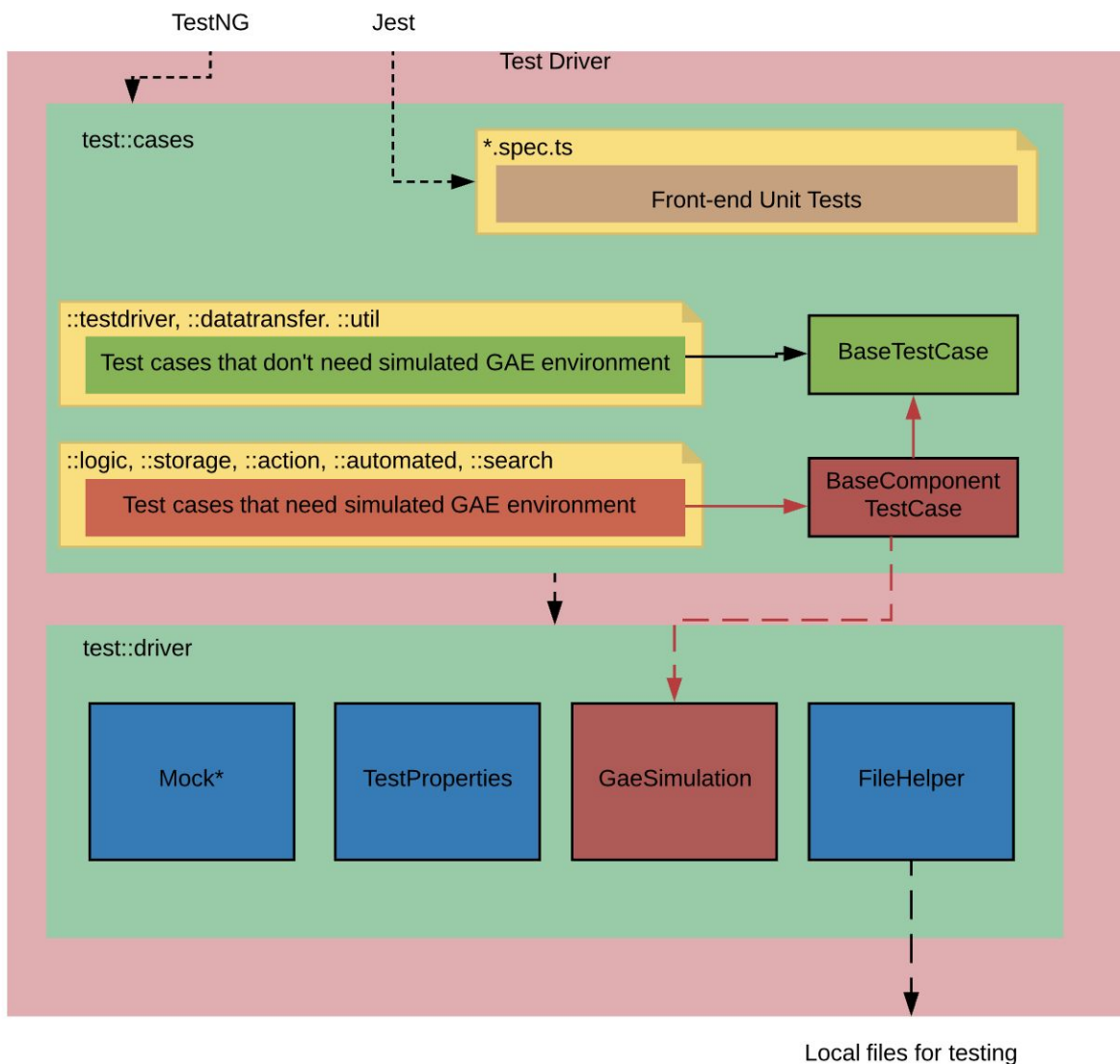
```

(Figure 3.1.2: Test file in
teammates/src/test/java/teammates/test/cases/BaseComponentTestCase.java)

3.1.1 Test Driver Component Diagram

The diagram below shows the test driver that is being used to test the UI website. It is a general diagram which explains how the test driver works. Within this diagram, I will show the flow of the test case scenario. The flow of the test case scenario is shown by the red arrows in (figure 3.2).

This component automates the testing of TEAMMATES.



(Graph 3.1.1: Test Driver Component)

3.2 Growth Scenario

Aspect	Details
Scenario Name	
Business Goals	Code should hold against high volume of data intrusions
Quality Attribute	Security
Stimulus	A high volume of requests from the web browser to the Web Api Servlet to access some data in the storage
Stimulus Source	Unauthorized users or developers
Response	Web API checks if users are unauthorized (e.g. not an admins), then refuses the action.
Response Measure	Web browser would print the following message " Method [name of the method] is not allowed for URL link] "(See figure 3.2.1)

(Table 3.2.1: Growth Scenario)

The above table shows a growth scenario where the stimulus source requests to access data that he/she are not allowed to access. In this scenario I will examine if the number of data intrusion increases extremely. How would the system respond? Would it crash or keep running?

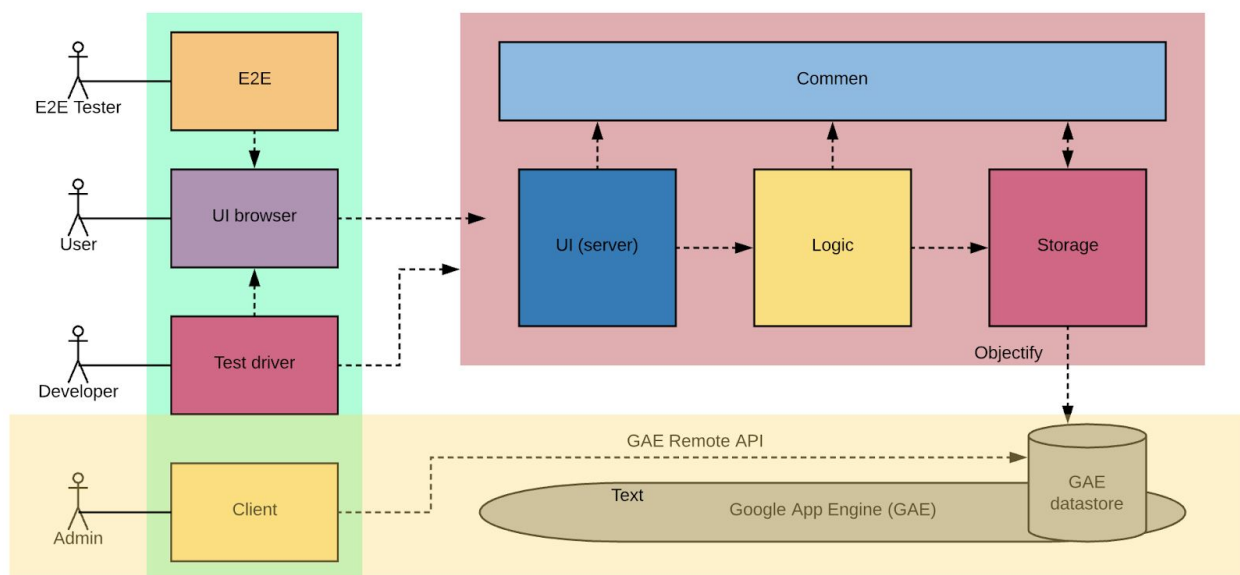
```

131
132 private Action getAction(String uri, String method) throws ActionMappingException {
133     if (!ACTION_MAPPINGS.containsKey(uri)) {
134         throw new ActionMappingException("Resource with URI " + uri + " is not found.", HttpStatus.SC_NOT_FOUND);
135     }
136
137     Class<? extends Action> controllerClass =
138         ACTION_MAPPINGS.getOrDefault(uri, new HashMap<>()).get(method);
139
140     if (controllerClass == null) {
141         throw new ActionMappingException("Method [" + method + "] is not allowed for URI " + uri + ".",
142             HttpStatus.SC_METHOD_NOT_ALLOWED);
143     }
144
145     try {
146         return controllerClass.newInstance();
147     } catch (Exception e) {
148         Assumption.fail("Could not create the action for " + uri + ": "
149             + TeammatesException.toStringWithStackTrace(e));
150         return null;
151     }
152 }
153
154 }


```

(Figure 3.2.1: message that shows to an unauthorized user)

TEAMMATES has a very good architecture for data security. All data is stored in a database within the Google App Engine; which is a cloud base. No developers, users or tests can access the database. We can see in the yellow highlighted section of the graph (graph 3.2.1) that the only allowed users to access the database would be the admin. So the number of data breach attempts would not matter since they can not access the data storage directly.



(Graph 3.2.1: Architecture of TEAMMATES)



Google App Engine is known for imposing various and data invasion restrictions on the application, e.g. each request to the app has to be served within 60 seconds. As a result, it helps to protect TEAMMATES data against data invasion regardless of the number of invaders. However, the development team should always keep up since TEAMMATES is running on Google App Engine cloud platform which is an emerging platform evolving rapidly.