Testing for TEAMMATES

https://github.com/TEAMMATES/teammates

SENG 275 Assignment 3

Andreas Anglin - V00851238
Geoffrey Harper - V00866723
Logan Macdonald - V00832812

Project Purpose

TEAMMATES is a free online tool for managing peer evaluations and feedback paths of students who are participating in a teachers class. The project is a web application that uses Java for its logic, UI and storage functionality. The database is the Google App Engine which uses objectified java code to maintain it. TEAMMATES uses Selenium and TestNG to help with automated testing. To get metrics on code coverage TEAMMATES implements codecov as online code coverage tool that is integrated into the project under to codecov.yml file. The Front End and Back End of the project in managed using gradle. TEAMMATES also uses the continuous integration application Travis CI in order to make sure build meet the company's standards of testing by running automated builds each time a pull request is committed.

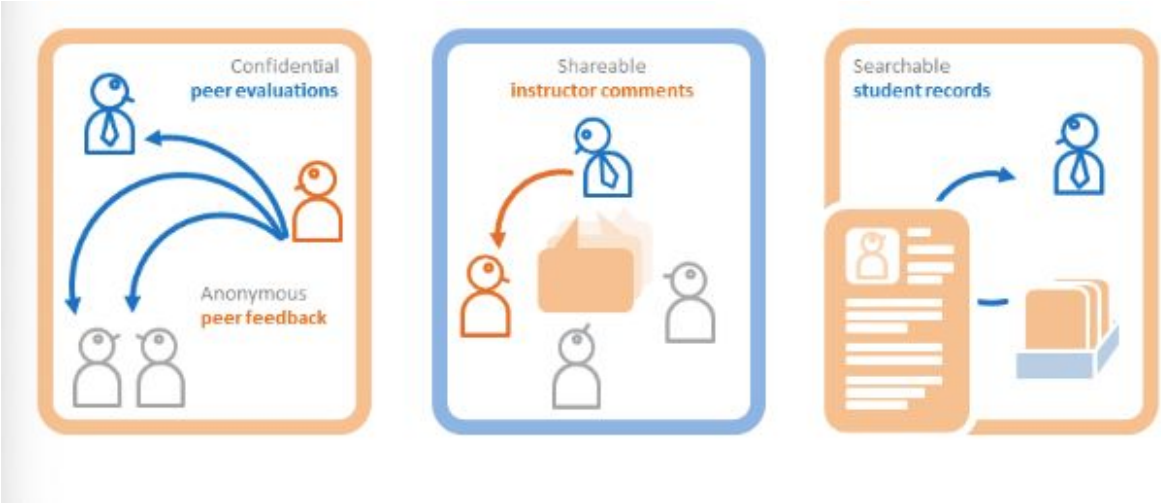


Figure 1: Diagram explaining TEAMMATES

Languages

Gradle is utilized to build the project. This project consists of a browser UI, written in HTML, CSS, and generated by an Angular framework. The program logic, server UI, and storage components make up the backend of the project and are all written in Java. A Google App Engine Datastore is also utilized as a database written in noSQL, which communicates with the Java storage backend component. Tests are written in Java with Jest for Unit tests and TestNG. Automated End to End testing of the web user interface

is done with Selenium, through Java. More automated testing is done with the utilization of a test driver with JSON formatted test data, which runs regression tests of Java code and HTTP responses/queries. An image from Teammates documentation is provided for a more in depth view of this project and its moving parts.

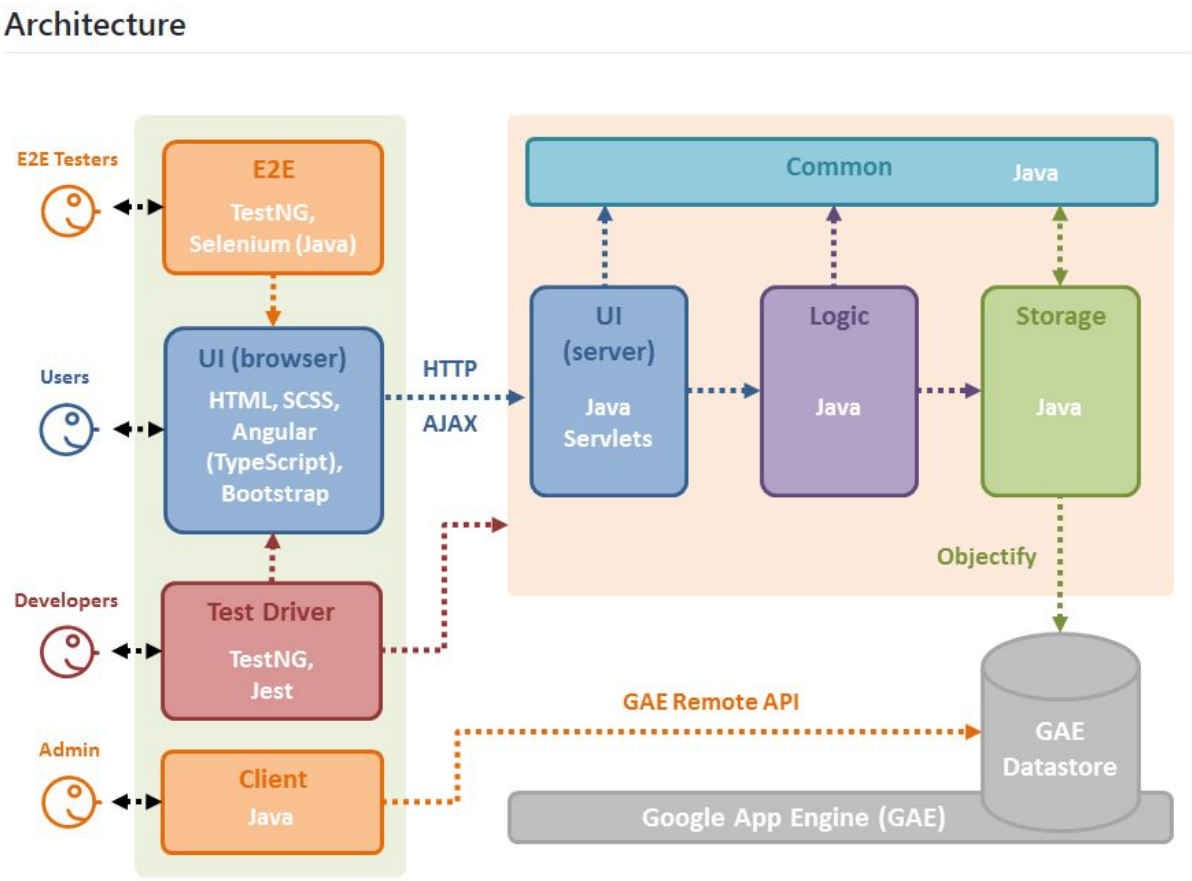


Figure 2. High level Architecture of TEAMMATES

We reviewed TEAMMATES on Windows 10 Pro Version 10.0.18362 Build 18326.

Bug Tracking

TEAMMATES tracks issues through GitHub, at https://github.com/TEAMMATES/teammates/issues. Currently there are 252 Open Issues and 5673 Closed.

This project has quite a regimented structure regarding the lifecycle of defects, from logging to closing. 'Issue Labels' are given to each defect for classification and an identity of importance. The following labels are defined as:

- Status
- Category (Type of work to be done)
- Priority (Importance, defined by maintainers)
- Difficulty Level
- Aspect (if non-functional, define it)
- Feature (functional aspect)
- Technology (Tech stack involved)
- Effort Estimate (Man hours required)

These labels help make the defect review a more efficient process. An in-depth diagram of the process, regarding how developers, reviewers, and team members interact follows.

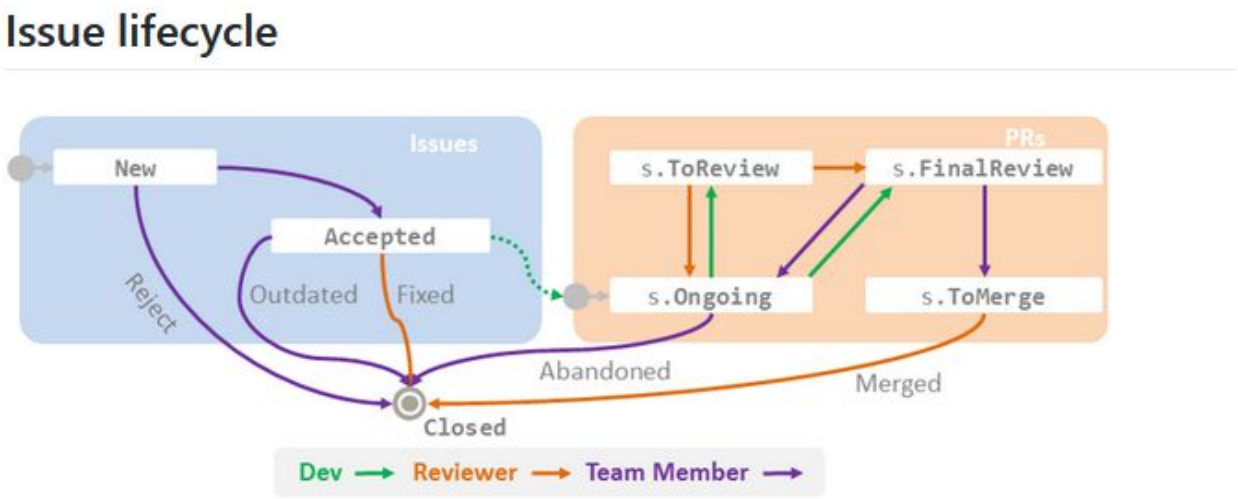


Figure 3. Flowchart for new issues in TEAMMATES

Note that any fixes must comply with TEAMMATES coding standards. Some notable findings follow:

- The code should be compliant with our coding standard, come with automated tests, and reviewed by (at least) a team member
- Where database operations are involved, the scalability and cost effectiveness of the implementation should be proven with quantitative evidence

Automated Tools

Tests are run using TestNG and Selenium implemented in Java. TestNG is inspired by tools like JUnit and NUnit with the goal to cover all categories of tests including unit,

functional, end-to-end, and integration. It was designed by Cédric Beust who found frustration with JUnit, specifically the requirement to set up static test classes if they require unique initialization and that JUnit creates individual test classes for each test case. Test NG includes the following additional functionality to JUnit:

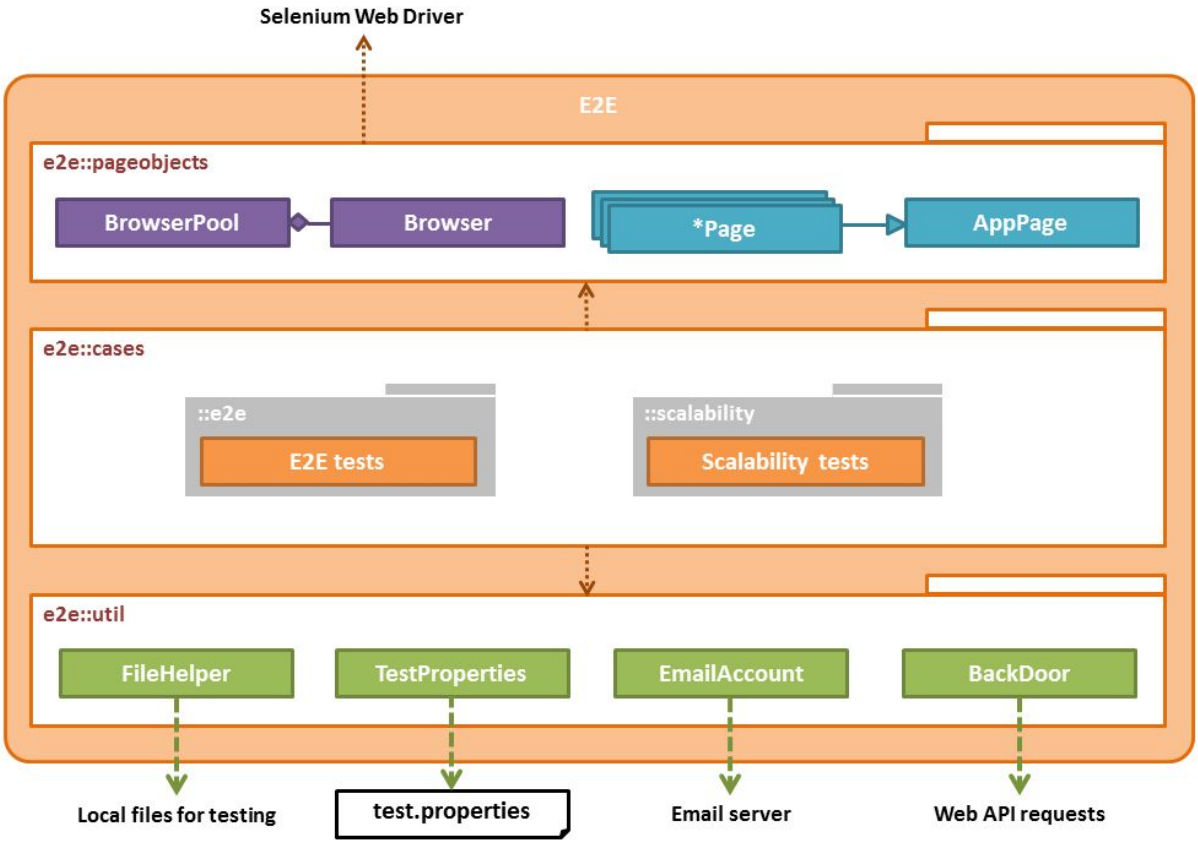
- Annotations.
- Run your tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc...).
- Test that your code is multithread safe.
- Flexible test configuration.
- Support for data-driven testing (with @DataProvider).
- Support for parameters.
- Powerful execution model (no more TestSuite).
- Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).
- Embeds BeanShell for further flexibility.
- Default JDK functions for runtime and logging (no dependencies).
- Dependent methods for application server testing.

End-to-end testing is done with Selenium webdriver, a tool for automating web browsers. It is primarily used for automated testing of web applications but can be used for any application such as web scraping. E2E tests are kept in /cases using helper functions from /util as needed. Each web page is abstracted with Java as they would appear in the Browser, and is kept in /pageobjects to be accessed by selenium.

Figure 4. Resources used for automated testing in TEAMMATES

Snapshot testing is performed to compare the expected look of web pages, emails, and CSV contents with those generated. TEAMMATES uses Jest for snapshot testing, a Javascript testing framework with built in support for snapshot testing. Jest needs to be run once first to collect expected results. Subsequent tests will then be compared to the previous snapshot.

Performance testing is done using JMeter along with TestNG and JMeter Java Api. JMeter is specifically designed to load and stress test web applications. It has the ability to simulate typical loads and simulated user behaviour and generate performance metrics for analysis. JMeter also offers the ability to record behaviour to create new tests efficiently.



Code Coverage

For code coverage codecov is used to obtain coverage statistics on the current build of the system. Codecov, – using the codecov.yml file – once synced to your git repository provides automatic coverage reports, reviews of coverage reports by folder and type of test (unittest,integration test, etc.).

TEAMMATES uses codecov to track line complexity and total code coverage of there /src folder. This allows TEAMMATES to get instant statistical information of the coverage percentages. At the moment the code coverage for TEAMMATES is 64.32% across all 600 files of the project. This level of coverage meets the minimum industry benchmark of having at least 50% code coverage. Looking at codevac 'SUNBURST COVERAGE' (https://codecov.io/gh/TEAMMATES/teammates) it is evident that the coverage is mainly in the logic and api side of the application. This is important for software the minimum software quality standard because it displays to the public that the majority of their testing is focused around important web and database logic. Low areas of coverage are mainly found in the UI. This is passable for a project about a

Student Feedback system since UI could be seen as the least important when compared to internal application logic and the database. However, because the app is about working with students submitting course feedback and teachers managing the evaluations, having minimum UI coverage seems unreasonable since it is vital for students and teachers to have a perfectly running UI since that is the only reason they interact with application.

Release Plan

There is no documentation regarding the standards of software release, however, from TEAMMATES first principle, they can be inferred. “We keep moving forward, always: We release frequently, in weekly time-boxed iterations. Every week, our product becomes better than the previous week.” Timeboxing is the division of a working schedule into smaller ‘timeboxes’ such that each has deliverables, a budget, certain goals, etc. Timeboxing helps to mitigate the risk of the unknowns in software development such as time and cost. In this case, it is fairly safe to assume that a new release usually occurs weekly, which may contain some defect fixes or new features. The new feature process is quite rigorous for this collective, and ensures that any new functionality is tested, efficient, and up to standards. Therefore, when a feature is ready for release, theoretically it should introduce little risk.

Interesting Findings

TEAMMATES is managed by a very small team with three core members. All other members are public contributions either addressing raised issues or submitting features. As there is no dedicated team for testing a majority of it is automated. Unit, Integration, End-to-end, snapshot, and performance testing are all done with automated tools. Testing falls on the contributor when either fixing issues or adding features.

Some static analysis is used for the purpose of code quality. Static analysis tools are mostly used, with personal discretion on what rules of each tool to follow. TEAMMATES outlines these guidelines for suppressing rules:

- The suppression should be as specific as possible, e.g. specific rule, specific scope (lines/methods/classes)
- The rule must be re-enabled after the suppression is no longer necessary
- The reason for violating the rule should be explained