Milestone 6

**Team 7 -Hotshots**

**Teammates Architectural Documentation**

**Table of Contents**

[1.0 Purpose 3](#_Toc4873007)

[2.0 About teammates system 3](#_Toc4873008)

[3.0 Stakeholders 4](#_Toc4873009)

[4.0 Business Goals 6](#_Toc4873010)

[5.0 Architecturally Significant Requirements (ASR) 7](#_Toc4873011)

[6.0 Utility Tree 8](#_Toc4873012)

[7.0 Quality Attribute Scenarios (QAS) 9](#_Toc4873013)

[7.1 Scenarios Tables 9](#_Toc4873014)

[7.2 Complete List of Scenarios 11](#_Toc4873015)

[8.0 Module view 12](#_Toc4873016)

[8.1 Primary Presentation 12](#_Toc4873017)

[8.2 Element Catalog 12](#_Toc4873018)

[8.3 Behaviour Diagram 15](#_Toc4873019)

[8.4 Context Diagram 15](#_Toc4873020)

[8.5 Rationale 15](#_Toc4873021)

[9.0 Component and Connector View (C&C View) 17](#_Toc4873022)

[9.1 Primary Presentation 17](#_Toc4873023)

[9.2 Element catalog 18](#_Toc4873024)

[9.3 Interface Context Diagram 19](#_Toc4873025)

[9.3.1 Objectify Interface 19](#_Toc4873026)

[9.3.1.1 SEI Element Interface Specifications 20](#_Toc4873027)

[9.3.2 Data Transfer Interface 21](#_Toc4873028)

[9.3.2.2 SEI Element Interface Specifications 21](#_Toc4873029)

[9.4 Behaviour Documentation 22](#_Toc4873030)

[9.5 Variability guide 22](#_Toc4873031)

[9.6 Rationale 23](#_Toc4873032)

[10.0 Code Quality and Technical Debt 24](#_Toc4873033)

[10.0 Code Quality Report 24](#_Toc4873034)

[10.1 Introduction 24](#_Toc4873035)

[10.2 Review Summary 24](#_Toc4873036)

[10.3 Finding Summary 24](#_Toc4873037)

[10.3.1A Dodgy Code 25](#_Toc4873038)

[10.3.1B Internationalization Issues: 25](#_Toc4873039)

[10.3.1C Code Vulnerability Warnings: 25](#_Toc4873040)

[10.4 Technical Debt report 26](#_Toc4873041)

[10.4.1 Using Objectify 26](#_Toc4873042)

[10.4.1A Technical Debt 26](#_Toc4873043)

[10.4.1B Recommendation 27](#_Toc4873044)

[10.4.2 Search Engine Optimization (SEO) Friendliness 30](#_Toc4873045)

[10.4.2A Technical Debt 30](#_Toc4873046)

[10.4.2B Recommendation 30](#_Toc4873047)

[References 31](#_Toc4873048)

# 

# s1.0 Purpose

# 2.0 About teammates system

Teammates strives to provide anonymous peer evaluation and feedback, while also allowing an instructor to share their own comments. Ethics and privacy are an important aspect for any software may it be open source or licensed. “Teammates” is hosted on Google App Engine and thereby uses the Google servers for storing and managing the data. This means that Teammates uses the same security mechanisms that Google uses to protect the data. The communication between the software components is performed using the inbound bandwidth within the Google Cloud Servers and thereby the data transfer between different software components is securely transported. Understanding the current goals of Teammates, it is clear that the consumer data hosted within the application is not being used for research purposes, none of the developers have access to the consumer data in production. Only the system administrators have access to consumer data for troubleshooting and performance monitoring purposes.

# 3.0 Stakeholders

Shown in Table 1.0 is the list of stakeholders for TEAMMATES

**Table 1.0: List of Stakeholder**

|  |  |  |
| --- | --- | --- |
| **Stake Holder** | **Actor(s)** | **Interest in the Architecture** |
| **Acquirer** | Developers, End-users | Oversee the procurement of the Teammates product. As Teammates is open source software anyone can download the source code from the github repository. The Teammates website does have terms of use agreement that Teammates is to be used at your own discretion and risk and you will be solely responsible for any damages that may arise from such use. As they are protected under the GNU General Public License 2.0, any distributed works must be made available under the same license |
| **Analyst** | System/Product/Business Analyst | Responsible for analyzing the architecture to make sure it meets certain critical quality attribute requirements. Simplicity, Performance, and Flexibility Quality Attributes. |
| **Architect** | Software Architect | It is the responsibility of this stackholder to ensure that elicitated requirements from the other stackholders are well designed and documented. The products produced by this stackholder provides a means of reasoning about the system functionalities |
| **Communicator** | Project Mentor | Responsible for assuring performance to standards and processes to provide confidence in a product’s suitability. |
| **Business Manager** | Project Lead | Although TeamMates is a not for profit business it still needs funding and has business operations costs. The person in charge of dealing with finances would be interested in the architecture to estimate the cost of new development costs in regards to developers, hardware needs and other aspects of the software development life cycle |
| **DevOps/Deployment** | DevOps Manager, DevOps Engineer, Conformance Checkers, Project Mentor | These stakeholders include analyzers and conformance checkers and they are interested in the architecture to ensure developers are developing the software in the way the architecture intended. In a project as big as TeamMates conformance by developers will likely be an issue. These type of stakeholders also involve deployment managers. Deployer is interested in the architecture because build systems, configurations, frameworks and hardware requirements, etc needed to deploy the system are often decided upon with direct correlation with the architecture. Being a web application, deployment is going to be an ongoing task and will likely need to adapt to changes in the software architecture. |
| **Database Administrator** | Database architect, Database Designer, Database Administrator | Most of the software components rely on database for communicating with each other. The architecture uses Google Cloud Datastore. He/She makes sure that the architecture of the software is well designed and optimised for effective communication among several software components. |
| **Designer** | Frontend Developer, UI/UX Engineer | The team of designers build a frontend wireframe architecture and try to make the User Interface as intuitive and user friendly as possible. The frontend and backend of the software architecture go hand in hand, in order to build a useful and smart user interface, it is important for these stakeholders to understand most of the software architecture. |
| **Developers** | Software Engineers, Software Developers | These stakeholders write the code to implement the results solution requirements |
| **Project Manager** | Project Manager/Project Mentor/Project Lead | Project manager manages and plans the execution of the project, build comprehensive plans and make sure the project aligns to the business goals |
| **Supplier** | Google Cloud Platform | Teammates architecture is largely built upon Google Cloud Infrastructure. Therefore, the infrastructure supplier for the architecture for Teammates is Google Cloud Platform |
| **Tester** | Software Tester | Anyone can write tests for Teammates. They have hundreds of TODO tasks and issues. If testers know the architecture, they will be able to write more tests and better tests. This will detect more bugs and improve the overall performance |
| **User** | Instructors, Students | Some users may be interested in the architecture because they have there own ideas of ways to improve Teammates. The users are the teachers and students. They know best what they want Teammates to be. Knowing the architecture will allow them to better describe their ideas in a software development context |

# 4.0 Business Goals

TEAMMATES is a peer feedback/evaluation tool for student teams while allowing the instructor the ability to oversee and share his/her own comments. It aims to provide a powerful, simplistic, and flexible experience. It has the following business goals

1. A custom communication portal for university teachers and students that tailors specifically to their needs and desires.
   * TeamMates wants to get more user experiences from both teachers and students and make adjustments to the feedback. This will help make a better product and increase our user base.
2. Enhances smooth partnerships among the invaluable contributors.
   * Giving contributors recognition has been and will remain a business goal for TeamMates. \* Maintaining contributor relations and gaining more contributors is a goal of TeamMates.
3. Creating a webservice with an emphasis on usability.
   * Usability is perhaps the most vital non-functional attribute for TeamMates. Teachers and students are busy people and supplying a fast and easily understood service is vital to improve peer to peer advertising and user adoption. Creating user surveys to test usability levels and getting improvements is a major goal for TeamMates at this stage of development.
4. A webservice which is aesthetically appealing.
   * The feelings and emotions users experience when using a web application is very important to a web application service in acquiring users and maintaining users. [13] Improving the overall feelings and visual appeal of the front end UI is a major business goal for TeamMates going forward.
5. A product which although open source is secure and trustworthy.
   * There is ongoing discussion regarding security implications of OSS vs CSS particularly with web applications containing personal information. [14] TeamMates wishes to ensure users that they have made organizational commitment to make TeamMates a leader in OSS security. TeamMates will also be transparent in its security practices and concerns. [15]
6. Front end should be modifiable and easy for outside developers to commit updates.
   * To get front end developers to work on TeamMates the environment should be easy to set up and instructions should be well documented. It is a goal of TeamMates to get more front end commits at this stage of development.
7. Front-end and back-end Migration
   * TeamMates wishes to make front-end and back-end development as independent as possible. This includes migrating to Angular 7 as the front end framework and changing the middleware architecture from SSR-based to AJAX call back-end. [16]

# 5.0 Architecturally Significant Requirements (ASR)

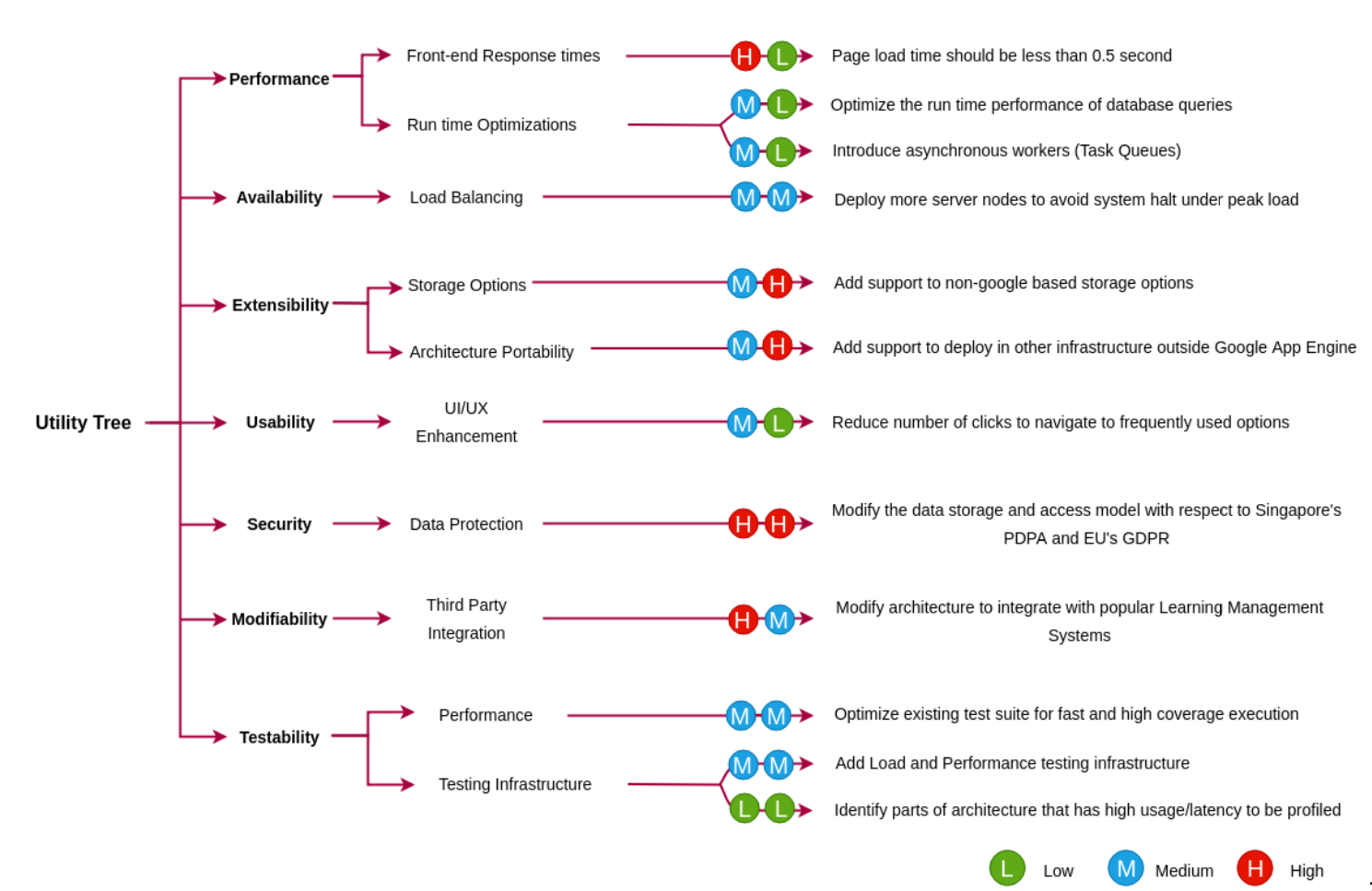
Shown in Table 2.0 is the ASR for TEAMMATES

**Table 2.0: Architecturally Significant Requirements (ASR)**

|  |  |  |
| --- | --- | --- |
| **Quality Attributes** | **Attribute Refinement** | **ASR** |
| Performance | Front-End Response Time | A user creates an evaluation for a group member while the system is under peak load. The front end should respond in less than 0.5 seconds. |
| Performance | Page Loading Time | Whenever page is loading, Server should not timeout or freeze. |
| Usability | New user proficiency | Users of Teammates should be proficient with basic features within 1 hour of usage time and advanced features within 4 hours of usage time |
| Usability | Streamless Display Adjustment | The display should react and adjust to all sizes of media without glitches or bugs |
| Usability | Browser Compatibility | Teammates must be able to run on all modern web browsers |
| Usability | User Interface Enhancement | Users should be able to navigate through basic functionality in a reduced number of clicks |
| Availability | Outright Uptime | Users account should be accessible at all time |
| Modifiability | Improving product structure especially during integration | Developers should be able to improve the existing code and architecture with ease. |
| Extensibility | Non-Google based storage options | The software architecture must be able to use other cloud storage options |
| Extensibility | Deploying in infrastructure other than Google App Engine | The architecture should be flexible enough to deploy on different infrastructure other than Google App Engine. |
| Security | Data Protection | Meeting current international data protection levels |
| Security | Data Privacy | Whenever evaluation is being done, team members' evaluations should not be revealed to one another within the team or another team |

# 6.0 Utility Tree

Figure 1.0 shows the Utility Tree



**Figure 1.0: Utility Tree**

# 7.0 Quality Attribute Scenarios (QAS)

## 7.1 Scenarios Tables

Table 3.0 to 5.0 shows the different QAS that formed the focal point for TEAMMATES

**Table 3.0: Scenario 1**

|  |  |
| --- | --- |
| **Aspect** | **Details** |
| Scenario Name | Confidentiality, Integrity and availability during data exchange within teammates should be provided |
| Business Goals | Ensured Data trustworthiness. |
| Quality Attributes | Security |
| Stimulus | Data being revealed to unauthorized users. Denial of Service(DoS) attack could be activated. |
| Stimulus Source | Human being, it may be internal or external being |
| Artifact | The target can either be the data exchange within the system or the application services. |
| Environment | Affect the operational system that is online. |
| Response | Data exchange with the system and the external services should be encrypted. File size received by the system should be monitored in order to avoid DoS |
| Response Measure | How many unauthorized evaluations was reveal and how often does the system became unavailable due to DoS attack? |

**Table 4.0: Scenario 2**

|  |  |
| --- | --- |
| **Aspect** | **Details** |
| Scenario Name | User defined UI issue resolution |
| Business Goals | Creating a webservice with an emphasis on usability and A webservice which is aesthetically appealing |
| Quality Attributes | Usability, Modifiability |
| Stimulus | **External:** General User |
| Stimulus Source | Negative feedback was received from users regarding a particular aspect of the UI. Issue could be functionality, error, glitch, display, etc. |
| Artifact | Front end java code, HTML, TypeScript and CSS |
| Environment | Runtime, Device display, Browser |
| Response | Analyze the negative feedback and formulate redesign plans. Formulate cost/benefit analysis and decide on plan. Execute plan and modify code/files. Test front end for bowser bugs and media size issues etc. If no problems re-evaluate feedback. Preferably from same source of original negative feedback. Rate the improvement. If unacceptable recycle the process. |
| Response Measure | UI improvements will range in response time: **Small** = 1 page / 2-3 elements = 6 hours, **Medium** = 2-3 pages / 4 - 6 elements = 1-2 days, **Large** = 3-5 pages / 7 - 10 elements = 3-5 days |

**Table 5.0: Scenario 3**

|  |  |
| --- | --- |
| **Aspect** | **Details** |
| Scenario Name | Resolving an issue by introducing new functionality. |
| Business Goals | A product which, although open source, is secure and trustworthy. |
| Quality Attributes | Modifiability |
| Stimulus | Wishes to submit a fix to an existing issue |
| Stimulus Source | Any open-source developer |
| Response | Developer pulls the master github branch. Developer works on own branch and fixes the issue. Developer submits a pull request for review. |
| Response Measure | Code is reviewed by Continuous Integration tools to check for build errors, integration errors, run errors. Core team member assigned to review pull request and final review. If approved code is merged to master Teammates git branch. |

## 7.2 Complete List of Scenarios

Having detailed the scenarios that was focused on in section 7.1, this section focus on listing of other scenarios as shown in Table 7.2

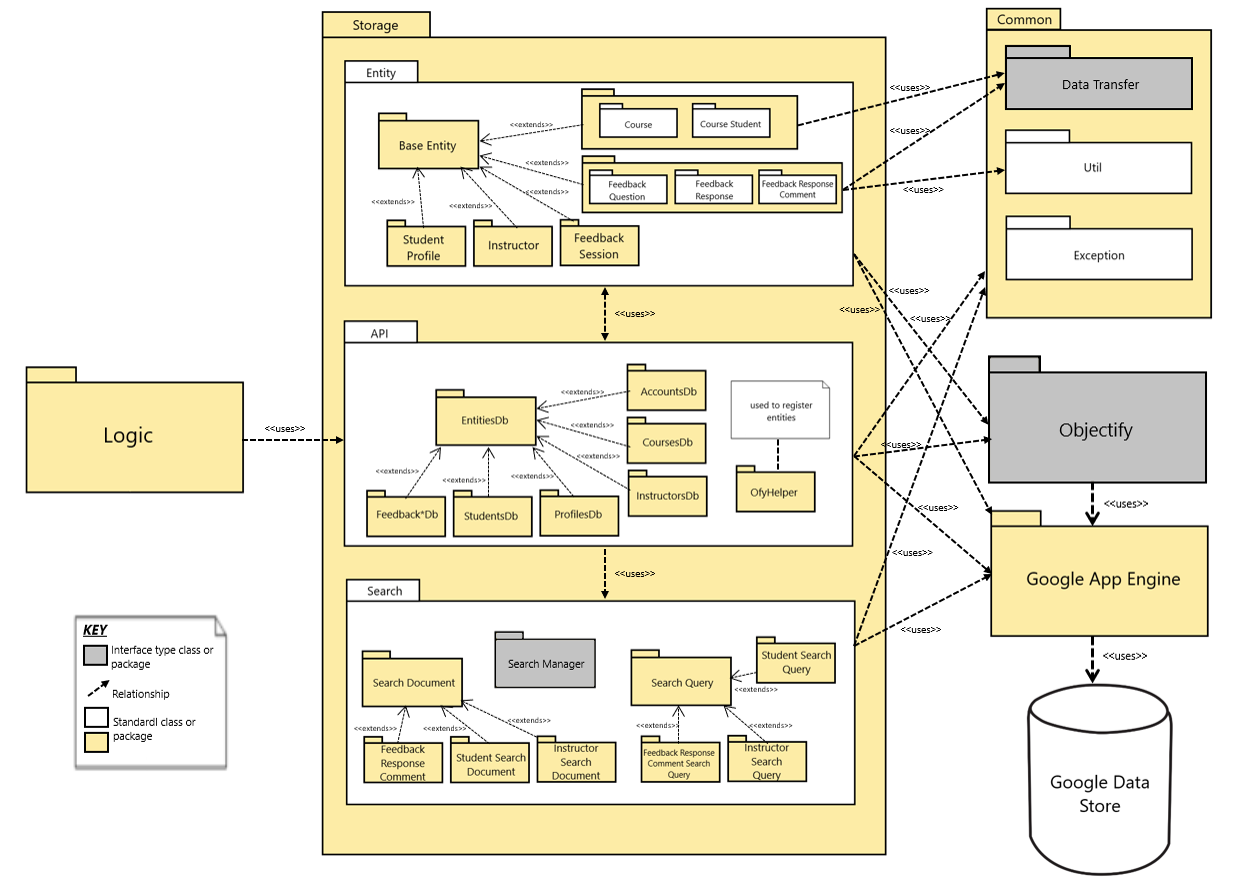
**Table 6.0: Other Scenarios**

|  |  |
| --- | --- |
| Quality | Scenario |
| Availability | Users entering a URL or clicking a button and the service was available for the user |
| Availability | Users browsing through all the pages of the software and pages are not broken |
| Security | Users private information are not displayed within the URL at runtime |
| Security | Confidentiality, Integrity and availability during data exchange within teammates should be provided |
| Usability | User defined UI issue resolution |
| Usability | Users create an anonymous evaluation and he/she finds the process of creating the evaluation easy |
| Modifiability | Resolving an issue by introducing new functionality. |

# 8.0 Module view

## 8.1 Primary Presentation

Shown in Figure 2.0 is the model view for TEAMMATES with key consideration on storage

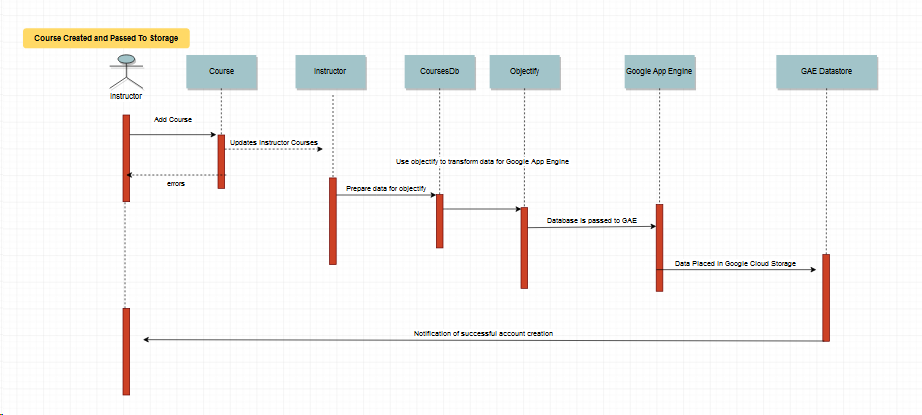


**Figure 2.0: Model View**

## 8.2 Element Catalog

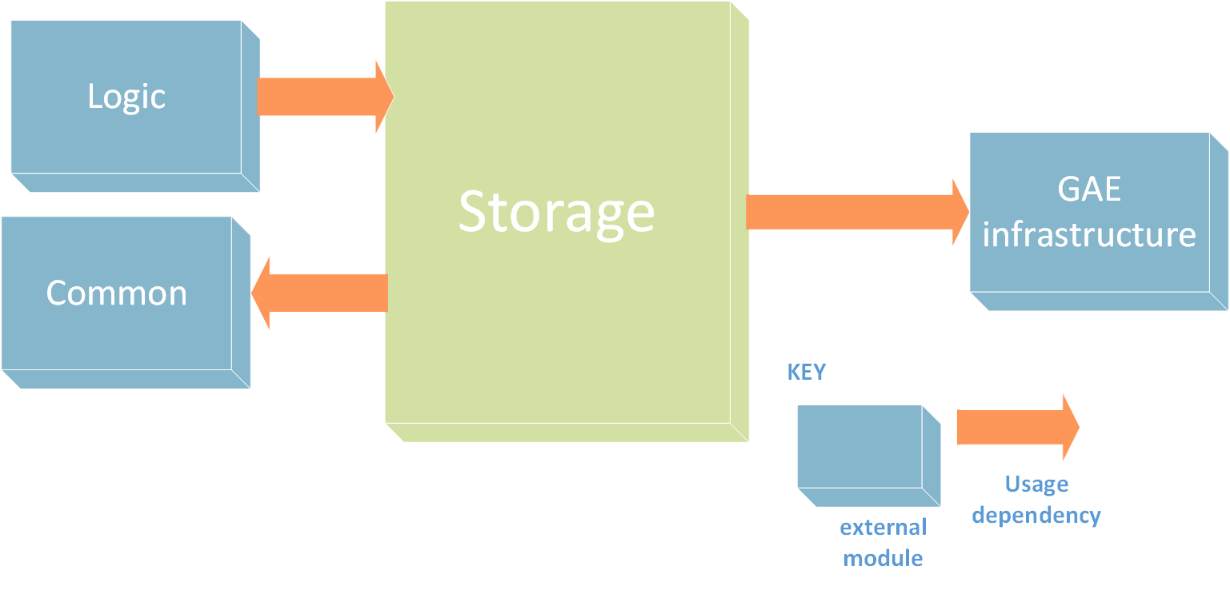
|  |
| --- |
| **Storage** This module contains all the storage logic including the workflow, internal API’s used for communication, interaction with internal search engine, and interaction with Google Cloud Datastore database. |
| **Common** This package contains the common utility classes that are used by the storage module. It also contains the data transfer logic acting as a api between frontend and backend communication and exception handling logic. |
| **Data Transfer** This module contains the data transfer logic between the frontend and the storage module. |
| **Util** This package contains the utility classes used by the storage module such as data structure for storing array of objects like ArrayLists. |
| **Exception** This package is the exception handling module. It contains the logic of handling all the run time exceptions without halting the workflow of storage module. |
| **Entity** Entity module represents a record in the database for every data object. |
| **Base Entity** This module acts as a common base class for several data objects which can be inherited to reuse common attributes of the data. |
| **Course** Represents a course entity. This module acts as a data object and has course specific attributes used to create new entries for courses in the database. |
| **Course Student** This package is a data object which has “Student Course” specific attributes i.e. the courses enrolled by the students. It is used to create new entries for student courses in the database. |
| **Feedback Question** This module represents the data structure for feedback question. It is used to create new entries for feedback questions in the database. |
| **Feedback Response** This module defines the structure for feedback response data object. It is used to create new entries for feedback responses in the database. |
| **Feedback Response Comment** This module represents the data object specific to the feedback response specifications. It creates new entries for comments on feedback response in the database and can be generated by student or instructor. It can be represented as: generator -> [comments about] -> feedback response |
| **Student Profile** This module represents profile details for the student entities associated with an Account entity. It is used to create new entries for student profiles in the database. |
| **Instructor** The instructor module represents the instructor for the specific course. It defines the data structure for storage of instructor data object and is used to create new entries for assignment of instructor to a particular course. |
| **Feedback Session** This module represents an instructor created feedback session. It maps the data structure used to represent feedback session in the database and is used to create new database entries for the same. |
| **API** This package provides an internal api for the application to communicate with the datastore. It provides functionalities for performing database operations like CRUD (Create, Read, Update, Delete). |
| **Entities DB** This module acts as a base class for all classes performing CRUD operations against the datastore. |
| Feedback DB Handles the CRUD operations for all the feedback related operations like feedback session, feedback question, feedback response and feedback response comments. |
| **Students DB**, **Profiles DB**, **AccountsDB**, **CoursesDB**, **InstructorsDB** These modules handle the CRUD operations for students, profiles, accounts, courses and instructor database. |
| **Ofyhelper** This module initializes the Objectify service at the startup of the application and register entity classes in the objectify service. Objectify is the main underlying interface used to communicate with the Google Cloud Datastore. |
| **Search** This package defines the logic for the document storage for indexing and/or searching the records in the database. |
| **Search Document** This module defines how we store the document for indexing and searching. |
| **Search Manager** This module manages the documents and indexes in the datastore for the use of search functions. |
| **Search Query** This module defines the structure and method used by the application to query the database. |
| **Student Search Query** This module represents a data structure that define how we query for students in the database. |
| **Student Search Document** This module defines the structure to represent how the students are stored in the database. |
| **Instructor Search Document** This module defines the structure to represent how the instructors are stored in the database. |
| **Feedback Response Comment Search Query** This module defines a data structure that defines how we query for feedback response comments. |
| **Instructor Search Query** This module defines a data structure that defines how we query for instructors. |
| **Objectify** Objectify is a Java data access API for Google Cloud Datastore which is used as an interface between the application storage logic and the underlying datastore. |
| **Google App Engine** Google App Engine (GAE) is a platform as a service offering on the Google Cloud Platform which has internal libraries like Objectify to establish connection and perform database transactions with Google Cloud Datastore. |
| **Google Cloud Datastore** Google Cloud Datastore is a fully managed database service offered by Google which is an underlying database technology for Teammates. |

## 8.3 Behaviour Diagram



**Figure 3.0: Behavioral Diagram**

## 8.4 Context Diagram



**Figure 4.0: context diagram**

## 8.5 Rationale

"Teammates" is a web-based application product which is entirely hosted on Google App Engine Standard Environment. Google App Engine is a platform as a service offering from Google Cloud Platform. Given this, the entire storage infrastructure of the Teammates is built upon Google Cloud Datastore. In order to perform database operations, Teammates uses Objectify which is a Java data access API for Google Cloud Datastore which is used as an interface between the application storage logic and the underlying datastore. One of the Quality Attribute Scenario of the Teammates is ‘Extensibility’ which states that the application must be flexible to host the database on different platforms outside Google Cloud Datastore and therefore, the storage architecture of Teammates is designed to be adaptable to other data storage options. The migration of the storage system is aimed to be achieved in less than 1 hour.

Teammates design aims for high level of portability to make use of several data storage options outside Google Cloud Datastore. The data storage infrastructure of Teammates is built around 4 core modules:

1. The **common module** is the facilitator for frequently used utility classes necessary for data structuring in Java, in-app data transfer APIs and exception handling mechanism to prevent run time breakdowns.
2. Secondly, the **entity module** represents the entire database schema of the Teammates. This module being isolated from rest of the other modules ensures that there is a scope of integrating with other data storage options since the modules that perform the CRUD, search and indexing operations are designed separately.
3. Third, the CRUD operations on the database are taken care by the **API module** in the Teammates architecture. This module acts as an internal API between the application logic and the underlying datastore. Since objectify is the interface between the storage logic and the google cloud datastore, the API module makes use of ofyhelper classes to use Objectify.
4. Finally, the **search module** serves as a information search service in the Teammates and it is also designed to be a separate module. The search module also uses API module in order to perform database operations.

The storage architecture of Teammates is aimed to be designed in a flexible manner even if it is entirely dependent on the Objectify for interfacing with the underlying datastore. Although the architecture currently uses Objectify version 5 which only supports interfacing with Google Cloud Datastore, but the Objectify version 6 extends its support to storage services outside google cloud datastore. It provides built-in facilities to help migrate schema changes forward. Objectify also surfaces all native datastore features, including queries, transactions, asynchronous operations, partial indexes and batch operations.

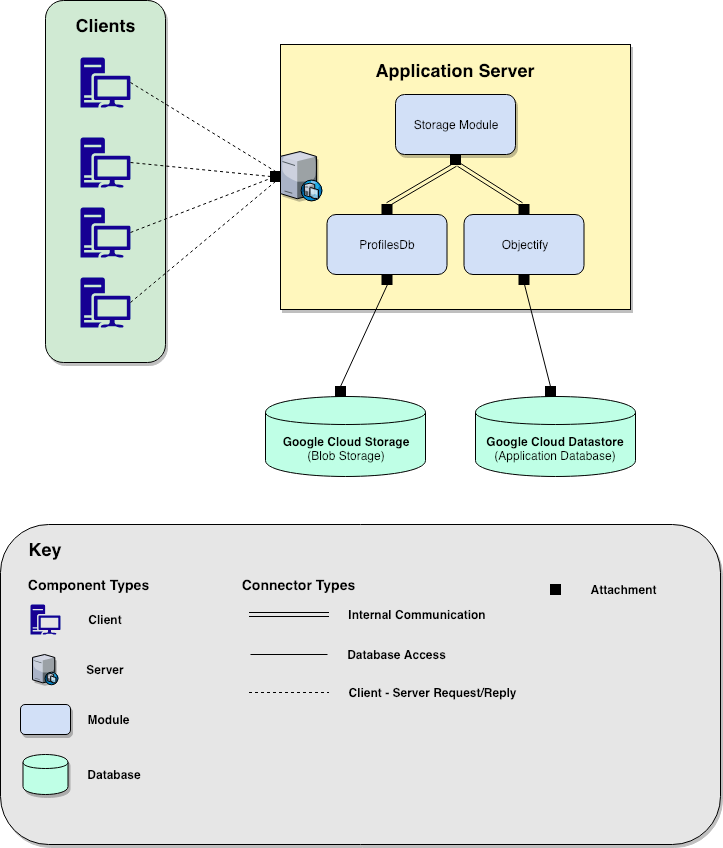
Another important consideration is that the Google Cloud Datastore uses unstructured No-SQL data storage system. This gives rise to the concern of data storage portability into structured databases. Looking at the architecture of the Teammates and the database schema, it is clear that even if the underlying datastore is is unstructured, the schema of Teammates is in a structured form. This makes the architecture more adjustable for migration to other SQL or No-SQL type of databases.

Modifiability is another important Quality Attribute Scenario for the given primary presentation view. While extending Teammates to other data storage options, some modifications in the existing storage architecture become essential. One of the possible modifications would be migrating from Objectify version 5 to version 6. This may not be a tedious process since the modules are strategically isolated from each other and only the API module will be affected.

# 9.0 Component and Connector View (C&C View)

## 9.1 Primary Presentation

**QAS**  
Extensibility (The software architecture must be portable to use other storage options outside google cloud datastore)



**Figure 5.0:** C&C View

**Description**

The above component and connector view capture one of our Quality Attribute Scenario (QAS) “Extensibility” which states that the application must be portable to use different storage options outside the Google environment. Analyzing the run-time behavior of the storage architecture of Teammates, it is known that they use two different storage options, first, Google Cloud Storage for storing the student profile data with blob data (images) and the second, Google Cloud Datastore for general purpose database operations.

## 9.2 Element catalog

Elements and their properties are described below:

**Client:**

All the interactions with the application are initiated by the client terminals. A client can be any device (mobile, tablet, desktop or laptop) making a request to the Teammates application through URL.

**Application Server:**

All kinds of requests from clients are handled by the application server. When database operations are requested by the clients, the storage module of the application server activates.

**Storage Module:**

Whenever database operations are requested, the storage module makes a decision if the requested data operation contains blob and is about student profile database, or if it is a general database operation request. For the former, it activates the objectify sub-module and for the latter, ProfilesDb sub-module is activated.

**ProfilesDb:**

All kinds of student profile data or blob operations are carried out through ProfilesDb component. It acts as a bridge between Google Cloud Storage and the application storage module. The ProfileDB is just one of the many objects defined within Teammate to access the storage module. Others includes, AccountDb, CourseDb, FeedbackQuestionDb e.t.c

**Objectify:**

The general database operations except the operations related to student profile data and blob data are carried out through Objectify. It acts as a bridge between Google Cloud Datastore and the application storage module.

**Google Cloud Storage:**

It is a student profile data storage and also blob storage for storing student images for the Teammates application.

**Google Cloud Datastore:**

This component is a main application data storage for the Teammates. It is an unstructured data store and is not used for storing blob data and student profiles data.

**Variability Guide**

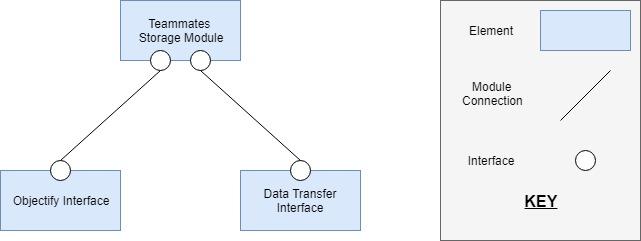
*(Changes by Karan: Changed objectify v6 to v5 and v7 to v6)*

**Application Server -** To allow for extensibility internal storage operations should be independent of external google cloud built-in storage functions. This is an area of future improvement for Teammates.

**Objectify -** Dependency on the Objectify 5 interface puts tight constraints on data storage options and the applications extensibility. Objectify forces Teammates to use the google cloud data store. Objectify 6lessens these constraints and would allow Teammates to use a number of other storage options. Upgrading is a likely future change.

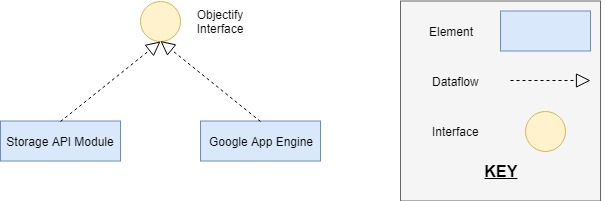
## 9.3 Interface Context Diagram

According to SEI, an interface is a boundary across which two elements meet and interact or communicate with each other. The key interfaces used by teammates that impact extensibility quality attribute are Objectify and Data Transfer interface.



**Figure 6.0:** Interface context diagram

## 9.3.1 Objectify Interface

****

**Figure 7.0:** Objectify Interface

Objectify is the convenient interface used by teammates to communicate with Google Data Store through Google App Engine (GAE). It helps to persist, retrieve, delete and query own typed objects with GAE. As our collection of elements in Storage API Module is interacting with GAE via Objectify, we are considering a particular example of CourseDB element to define Element Interface Specifications.

### 9.3.1.1 SEI Element Interface Specifications

1.       Interface Identity: CourseDB Interface

2.       Resource Provided:

a.       Resource Syntax: The method is defined named getCourseEntity and takes courseId as a parameter. It returns load().id(courseId).now() .

b.       Resource Semantics: The resource performs the functionality of fetching the course details providing the CourseId parameter.

c.       Error Handling: DatastoreTimeoutException, ConcurrentModificationException, or DatastoreFailureException are thrown by Objectify if something goes wrong.

3.       Data Types: The data types of the variables are as follow:

a.       courseId: String

b.       courseName: String

c.       createdAt: DateTime

d.       deletedAt: DateTime

e.       timezone: ZoneId

4.       Error Handling: In case anything goes wrong, the exceptions are thrown. Some of the exceptions that are handled are DateTimeException, EntityDoesNotExistException, InvalidParametersException.

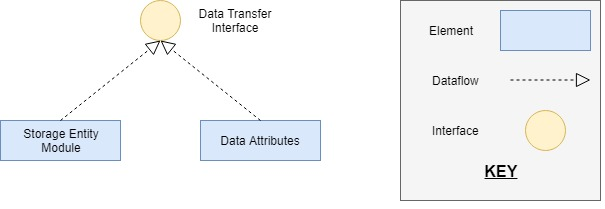
5.       Variability: The interface can be modified to scale up the transactions handling if the peak load is exceeded.

6.       Quality Attributes: The interface can impact the performance while querying the data from the data store when the transactions(queries) are performed.

7.       Rationale: The interface is designed to improve the performance of the transactions performed on the data store. It also provides human friendly query interface and easy-to understand transaction model.

8.       Usage Guide: The interface can be used by calling the methods to perform the query. The method call getCourseEntity should return the response with course details if the courseId is passed correctly. Similarly, the other methods (e.g. updateCourse) can be called to perform the transactions accordingly.

## 9.3.2 Data Transfer Interface

****

**Figure 8.0:** Data Transfer Interface

Data Transfer interface is another important interface used by Teammates storage module that can impact extensibility. This interface helps in creating the data transfer objects that are used for querying the data in GAE using Objectify. These data transfer objects are called as Data Attributes. There are a lot of elements interacting through this interface and for the Element interface specifications, we consider an example of CourseAttributes element.

### 9.3.2.2 SEI Element Interface Specifications

1.       Interface Identity: CourseAttribute Interface

2.       Resource Provided:

a.       Resource Syntax: The method is defined named Builder and takes courseId, name and timeZone as a parameter. This method also perform some validation checks on these parameters passed and then create the CourseAttribute.

b.       Resource Semantics: The resource performs the validation checks on the data required for the course and encapsulate them to the data object that can be transferred for the querying.

c.       Error Handling: The resource performs the validity checks on parameter values. The null value exceptions are handled.

3.       Data Types: The data types of the variables are as follow:

a.       courseId: String

b.       courseName: String

c.       timezone: ZoneId

4.       Error Handling: The data objects generated by interface are the Java objects and if anything goes wrong, it is handled by Java automatically.

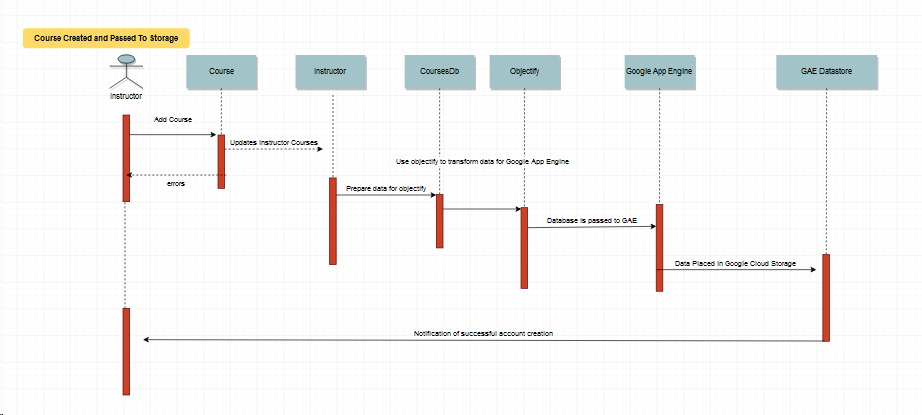
5.       Variability: The memory of the object generated by an interface is managed by Java Virtual Machine (JVM). If a lot of memory is utilized, JVM’s heap size can be increased.

6.       Quality Attributes: The interface ensures the quality attribute of Robustness, since it generates the Java objects and Java supports the strong type checking mechanism.

7.       Rationale: The interface is designed for the encapsulation of the data into the objects and also perform the validity checks on the data. It is very developer friendly to use the interface and generate the entity attributes.

8.       Usage Guide: The interface can be used by passing the values of the required parameters in the builder method and it will return the entity attributes object. Here, the builder method takes courseId, name and timeZone as a parameter input and returns the CourseAttributes object.

## 9.4 Behaviour Documentation



**Figure 9.0:** Behavioural Documentation

The sequence diagram above exemplifies creating a course being created and being sent to storage on the GAE database. Teammates depends primarily on web applets in order to get the code across as seen in the figure above. However, behind the scenes a number of java classes are called in order to service the web applet. Once a course is added, an instructor account needs to be updated with the correct permissions to be associated with the account. After the instructor permissions, the account is added to a course database. From here the entire database is passed to objectify in order to transform the information into data that can be read by the google app engine which can then be passed to the google datastore. The teammates storage operations are focused on CRUD operation (create, read, update, delete). Essentially before the data itself reaches objectify these operations are performed internally to ensure that the correct data is being uploaded to the google data store. In order to maintain consistency, add/delete operations wait until data persists in the google data store before returning. An assumption here is made that enough time has elapsed to “persist”, meaning that the data has propagated through all of the google data stores before returning. Certain errors could arise due to this design decision, and as such is under review as transactional control isn’t foolproof.

## 9.5 Variability guide

Objectify is the convenient interface used by teammates to communicate with Google Data Store through Google App Engine (GAE). It helps to persist, retrieve, delete and query own typed objects with GAE. As our collection of elements in Storage API Module is interacting with GAE via Objectify, we are considering a particular example of CourseDB element to define Element Interface Specifications.

## 9.6 Rationale

The component and connector view of the teammates highlights the "Extensibility" Quality Attribute Scenario for the Teammates application. The view mainly depicts the data storage and retrieval operations of the Teammates during the run-time. It is observed that the Teammates uses two different types of data storage infrastructures. One is Google Cloud Storage which is used for blob data storage and storing the student profile data which holds the student profile picture as the blob data. The second is the Google Cloud Datastore which is used for all the data storage and retrieval operations for the Teammates except blob and student profile data storage. This is because Google Cloud Datastore is not optimized for storing blob data. One of the reasons for choosing two separate data storage infrastructures is the cost of operation during the runtime. Google cloud datastore is relatively a cheaper offering from Google and Google Cloud Storage being a relational and transaction friendly database has a higher cost of operation.

Given such constraints, the QAS of making the application portable to support other data storage options outside Google Cloud Platform is a valid concern. Our component and connector view grips on the run-time behavior of the storage logic of the teammates which clearly documents the storage logic which is the main region of focus with respect to the "Extensibility" QAS.

# 10.0 Code Quality and Technical Debt

## 10.0 Code Quality Report

## 10.1 Introduction

A Code Quality Review is a task with the goal of identifying types of weaknesses that exist within a given code base. The task involves both manual and automated review of the underlying source code and identifies specific issues that may be representative of broader classes of weakness inherent in the code. For Milestone 5, we have performed an automated code review of the project “Teammates” using FindBugs code review tool. A Code Review does not attempt to identify every issue in the code but instead attempts to identify types of risk within the code such that mitigation strategies can be devised. It should be noted that while the review process will be as thorough as possible in finding and reporting code weaknesses, it is not guaranteed to always find every possible weakness. If no issues are found, the review does not implicitly certify that the application is 100-percent “hack proof.” A Code Review is a strong part of an overall risk mitigation program to improve the quality of the application.

## 10.2 Review Summary

The secure code review of the Teammates application was completed on March 23, 2019 by our team “project-team-7”. The review was performed on code obtained from Github repo of Teammates bundled under directory name “teammates.git”. A meeting among our team members was held on March 23, 2019, at which time information about the code structure was presented along with high-level overviews of how things like data validation, performance, and data encoding were implemented in the code. The actual review involved an automated investigation of the Java code using the FindBugs tool. The specific findings after using the FindBugs tool are presented in the subsequent sections.

## 10.3 Finding Summary

This section provides a summary of the findings resulting from this code review.

The summary of the analysis is shown below:

**Table 7.0: Metrics**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Metric | Total | Density |
| 1 | High Priority Warnings | 4 | 0.13 |
| 2 | Medium Priority Warnings | 80 | 2.60 |
| 3 | Total Warnings | 84 | 2.73 |

#### 

**Table 8.0: Warnings Summary**

|  |  |  |
| --- | --- | --- |
| No | Warning Type | Number |
| 1 | Bad practice Warnings | 4 |
| 2 | Correctness Warnings | 6 |
| 3 | Internationalization Warnings | 3 |
| 4 | Malicious code vulnerability Warnings | 3 |
| 5 | Performance Warnings | 4 |
| 6 | Dodgy code Warnings | 64 |
| 7 | Total | 84 |

For the Teammates application, we found several issues but we would like to discuss the three major findings uncovered after FindBugs analysis. According to FindBugs, some of them have high impact and some have a medium impact on the overall code quality of Teammates.

### 10.3.1A Dodgy Code

* A dodgy code can be understood as a piece of code which has never been used or has no impact on the overall application execution.
* After the automated investigation using FindBugs, it was found that there were 64 places in the code which had circular dependencies, unused declaration and initialization of variables and one of the places had the loading of Null value into a function.
* The analysis is presented in the FindBugs report.

### 10.3.1B Internationalization Issues:

* While encoding data from string to bytes and vice versa, it is found there is reliance on default encoding and decoding without specifying a charset name.
* Given that the application is being used globally and future scope of supporting multiple languages, there is a scope of carefully considering such internationalization issues.
* These issues are presented in detail in the FindBugs report.

### 10.3.1C Code Vulnerability Warnings:

* With respect to data security considerations, some malicious code vulnerability issues were detected. Though they are not many, circumventing them may not be a good decision in the long run.
* There are 3 places in the code where the internal object representations are exposed. If some untrusted code accesses these instances and introduced unchecked changes to the mutable object would lead to the compromised security or other important properties.
* The findings are documented in the Find Bugs report.

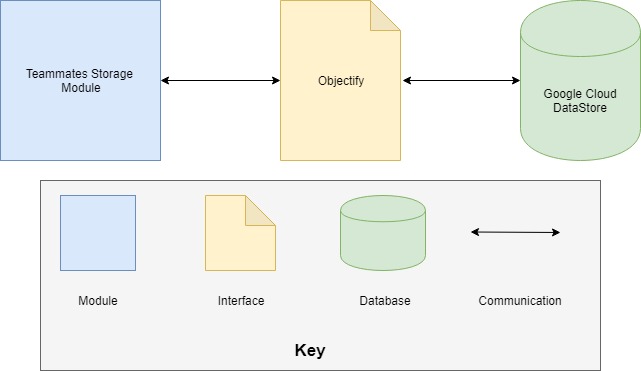
Although we discussed the top three results of the code review, there are several other findings which were uncovered. Bad coding practice was one of them which the FindBugs reported and most of them deal with performing string operations. The reason behind this is the differences in the use of methods in several Java versions. The second type of warning was Correctness warning which reported certain scenarios in the code that returned unwritten variables and passing null parameters to a non-null method parameter. The final one was the performance warning. There are a few fields in the code which are declared and initialized but never used anywhere. Such occurrences are advised to be removed from all the classes.

## 10.4 Technical Debt report

## 10.4.1 Using Objectify

### 10.4.1A Technical Debt

Objectify is identified as the key potential debt considering the Extensibility quality attribute scenario. It is an interface used by teammates to communicate with Google Data Store through Google App Engine (GAE). The technical debt of this comes as result of being locked in to the Google App Engine (Environment). We have focused on the quality attribute scenario of extensibility which states the ability to adapt the storage platform on data storage options outside of the Google Cloud Datastore. The following pictorial representation illustrates high dependency on objectify for routine database transactions.



**Figure 10.0: Representation of dependency on Objectify**

Currently, the teammates is using Google cloud service but the extensibility scenario focuses on adding support for other cloud service providers. Whereas, objectify is specifically designed for Google Cloud Datastore and tightly coupled with GAE. Based on the architecture design decisions made for using objectify, it is now hard to add a support to other cloud service providers.

### 10.4.1B Recommendation

These can be rectified if Teammates would upgrade from objectify version 5, to objectify version 6+. “Objectify v6+ uses the Cloud Datastore API and can be used from anywhere - GAE Standard, GAE Flex, GCE, or outside Google Cloud entirely” [1]. This directly addresses the extensibility issues of multiple data storage options within or outside of the Google Cloud Datastore as the new API being used would not rely on the Google App Engine Datastore, but switch to the Cloud Datastore API.

The Teammates development has commented on their own challenges with the Google App Engine as follows: Platform: TEAMMATES is running on Google App Engine cloud platform, which adds the following challenges.

* It is an emerging platform evolving rapidly. We have to keep up.
* It imposes various restrictions on the application, e.g. each request to the app has to be served within 60 seconds.
* It charges us based on usage. We have to optimize usage.

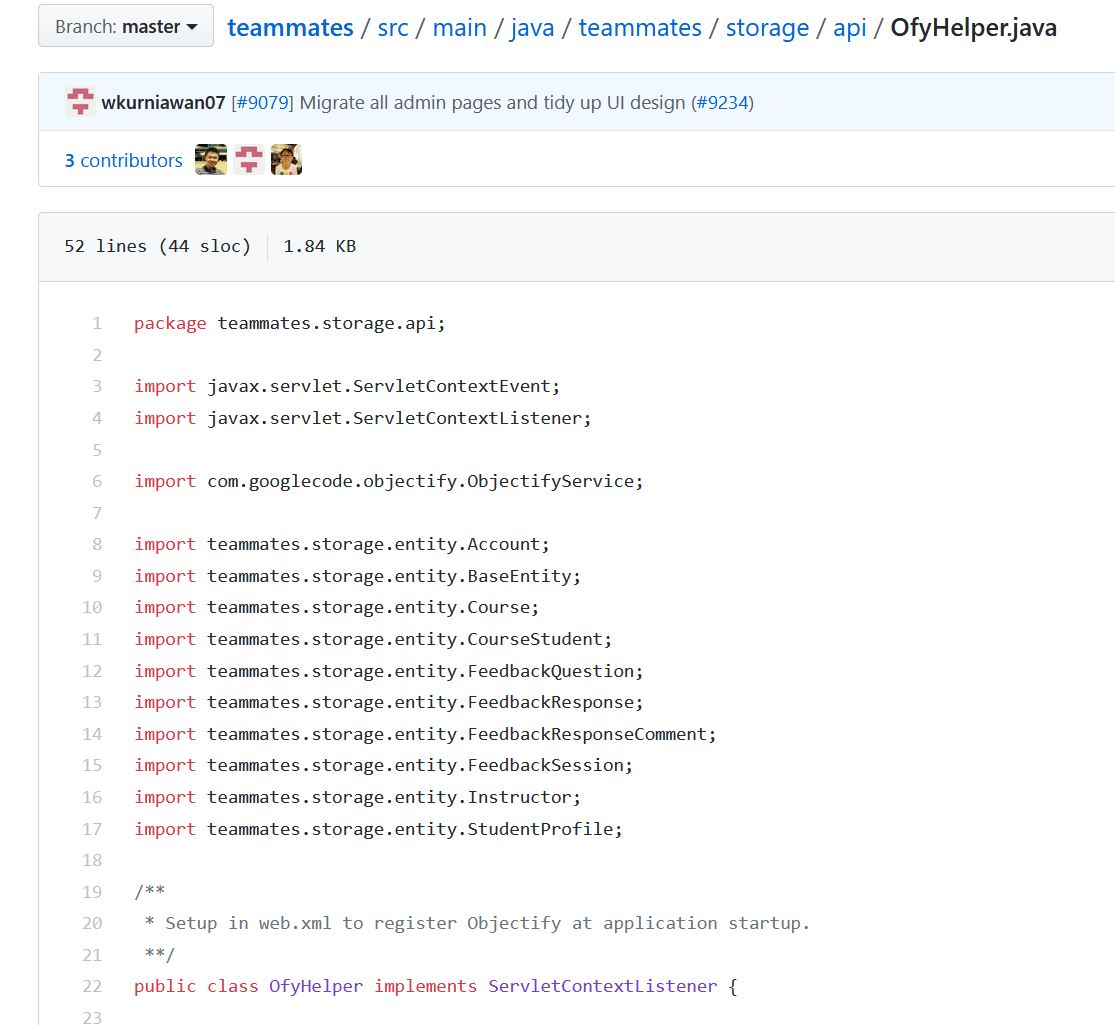
Within Teammates own development comments they state that they have to keep up with the developmental changes of the Google App Engine. However when doing research Google has stated that they are "in the process of replacing this proprietary interface with standards-based (REST and GRPC) interfaces to cloud services. This is the new "Google Cloud SDK" which can be used from both outside and inside GAE Standard. This required a rewrite of Objectify, which is v6" [5].

Following the link to "<https://github.com/objectify/objectify/wiki/UpgradeVersion5ToVersion6>" highlights some of the challenges of upgrading from v5 to v6+ for objectify.

When analyzing the codebase of teammates we see that there are some major codebase changes that would have to occur if they intend to keep up with the Google App Engine cloud platform. This would be a massive undertaking however here are some of the technical debts that could be focused on

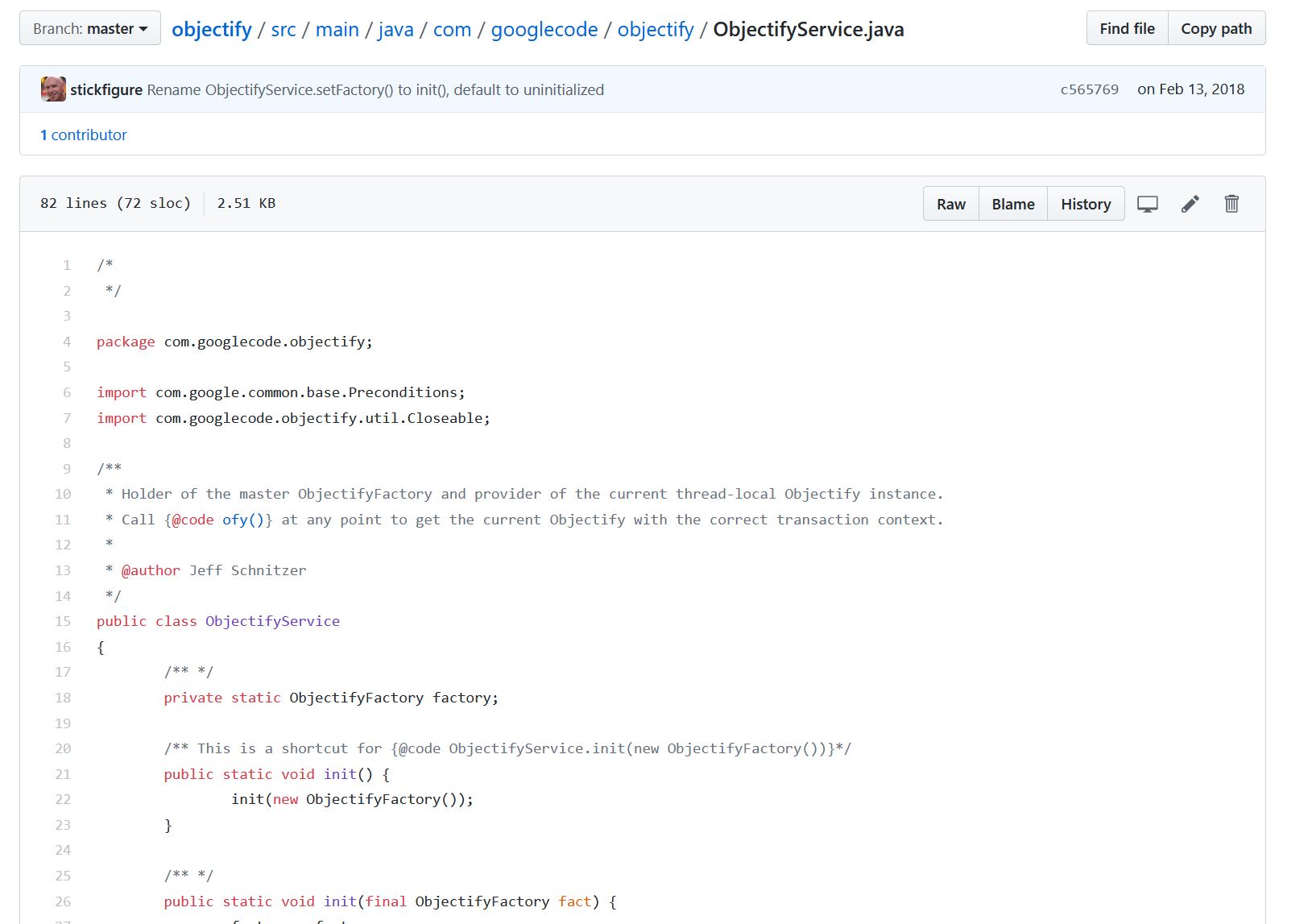
* "You will need to call ObjectifyService.init(), possibly passing in a custom ObjectifyFactory instance. The previous behavior of starting out with a default ObjectifyFactory incurs a connection cost and may fail in environments which require a custom DatastoreService" [4].

As evidenced in the ofyhelper.java file snapshot below we see that an instance to the ObjectifyService has already been implemented under the line (import com.googlecode.ObjectifyService;)



**Figure 11.0: Representation of dependency on Objectify**

If we analyze this further we can look into the ObjectifyService code and find the following snapshot. We can see that the ObjectifyService.java file is the holder of the master ObjectifyFactory and provider of the current thread-local Objectify instance.



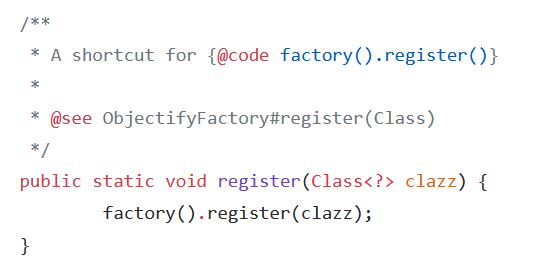
**Figure 12.0: Representation of dependency on Objectify**

As a result the recommendation would be to create a custom ObjectifyFactory instance within the ofyhelper.java file. And then as we see in the image below each instance of the storage classes are registered with the Google App Engine. This practice is limited to version 5 however.



**Figure 13.0: Representation of dependency on Objectify**

Our recommendation would be to change to the new version of the ObjectifyFactory instances by using the code below. Here we can register each of the classes inside of an ObjectifyFactory instance and then register and pass the ObjectifyFactory in order to be recognized by the new Google Cloud SDK.



**Figure 14.0: Representation of dependency on Objectify**

## 10.4.2 Search Engine Optimization (SEO) Friendliness

### 10.4.2A Technical Debt

Teammates adopted the Single Page Application (SPA) architecture in designing the frontend. The decision of implementing SPA architecture using Angular version 7 has caused a challenge in terms of friendliness on SEO. Angular v7 is usually rendered on the client side which prevents web crawlers and search engines from seeing the complete structure and contents of the website [6]. Being visible to the public and acquiring users through search engines is important to Teammates business goals. With the growing use of angular, the solution has been developed to solve this named Angular Universal. But based on the teammate’s frontend design, there is the problem for using Angular Universal with teammates. The trade-off here is that Universal can run only in Node.js and it requires webpage files and web API’s to run on different servers.

### 10.4.2B Recommendation

To solve this problem, it is required to separate the Webpages from the web API’s and make it run on different servers. Once completed, the Angular Universal can be implemented which uses server side rendering (SSR). This will allow web crawlers to detect the site by executing javascript code along with producing complete html at the server. Implementing Angular universal is a tradeoff though as it creates another dependency on outside technology and SSR is not ideal and defeats some of the purpose of using a SPA.

# References