AnyMemo-badpoem Release 3 Summary

Team members

Name	Github id	Number of story points and ideal hours that member was an **author** on.
Brendan McGarry	@brendanmcgarry	0 points
Paul Richard	@prich28	21 points
Dylan Fernandes	@dylanfernandes	29 points
Hiu Tung Lam (Emily)	@Ereimei	23 points
Adam Galati	@aagalati	18 points
Matthew Teolis	@MatthewTeolis	13 points
Olivier Nourry	@ONourry	21 points

Each group member is responsible for counting their own story points. It is the group leader's duty and responsibility to make sure they are accurate. Please keep in mind that we will check your GitHub stats (go to: "Graphs" on your GitHub project page, for example). Note, if your email and github id are not linked properly you will not be counted properly.

You will lose 1 mark if links below are not clickable.

Table of Content

Project summary (max one paragraph)

The AnyMemo application is principally a flashcard application where decks of cards with questions can be downloaded or created and used for studying. The application allows for making new decks, editing cards, studying and self-quizzing. The goal for this project was to add additional features onto the existing app. The major new features include adding tags to deck and achievement points for progress tracking.

Velocity and a list of user stories and non-story tasks for each iteration

(make sure the iteration is clickable link to the milestone on github)

Release 1

Total: 6 stories, 28 points, over 4 weeks

Sprint 1 (3 stories, 18 points(Pseudo Points))

#2: Reverse Engineer current Project [5 points] [Status: Done]

Each developers analyzed the code and created class diagram for AnyMemo.

#3: Decide on new features for app [5 points] [Status: Done]

Each developers came up with a list of feature and the group finalized the features to present the TA.

#4: Deploy Key Setup [8 points] [Status: Done]

Jenkins CI has been set up and connected to the GitHub repository.

Sprint 2, Release 1 (3 stories, 10 points)

#5: Tooltip when holding icon [2 points] [Status: Done]

A tooltip (pop up) is added to indicate the functionality of an icon when holding the icon on the device.

#6: Implement error page to replace stack track on app crash [3 points] [Status: Done]

A user-friendly error message is added to replace the stack track message when application crashes.

#9: Allow the user to add tags to decks [20 points] [Status: Pushed and Splitted]

This user story has been re-written and pushed to sprint 3 and 4 as user story #53 and #56 because of the underestimation of the tasks.

#10: Card editing: Combine “Edit” and “Detail” views [5 points] [Status: Done]

The access of edit functionality for card is combined to one single view. The “Edit” and the “Detail” options are merged and it leads to the same edit page. An “Advanced options” button is added to hide the advanced attributes and it gives user a warning if clicked.

Release 2

Total: 4 stories, 54 points, over 4 weeks

Sprint 3 (1 story, 20 points)

#53: Filter deck [13 points] [Status: Pushed]
This user story has been pushed to sprint 4 because it required the new central database set up from user story #56. Only half of the work was completed in this sprint.

#56: Add tags to decks [20 points] [Status: Done]
Tags feature has been added. A new central database is set up to store the tags. A user interface is created so the user can view all the tags belonging to the selected deck, can create new tag or existing tag to the deck and can delete the tag from the deck.

Sprint 4, Release 2 (3 stories, 34 points)

#52: Implement Achievement System [13 points] [Status: Done]
A user class, user statistics class and achievement point class are implemented to allow users to keep track of the their progress on decks and tags. They use the new central database and corresponding dao are created.

#53: Filter deck [13 points] [Status: Done]
Filter feature for tags has been added. The user can select the filter functionality in the header and can select existing tags to filter the decks. The existing code is updated when changes are made to decks and tags in the centralized memory object.

#106: Account page [8 points] [Status: Done]
Profile page feature has been added. A default account is created upon opening the application. The user can view the account information and edit his name.

Release 3

Total: 6 stories, 58 points, over 4 weeks

Sprint 5 (3 stories, 24 points)

#22: Rating Decks: Adding Ratings [8 points] [Status: Done]
Deck rating feature has been added. A new attribute "rating" is added to deck and the new central database. A user can long-press on the deck in "open" tag to update the deck's rating. The user can enter a score from 1 to 5 and the rating is displayed as star until the deck in "recent" tab.

#72: Daily points [8 points] [Status: Done]
Daily points feature has been added. Daily point class and dao are introduced to store the user's daily points to the new central database. If a user has a registered account, when he logs in, a custom toaster will indicate that he got 5 points. The custom toaster is also added to "easy" button of card single sided option for study session and to "remember" button of card single sided option for quiz session.

#104: Manage All Tags [5 points] [Status: Pending]
Tag management is added to allow user to edit all the existing tags. This feature includes editing the tag's name and deleting a tag.

#132: Account Register [8 points] [Status: Done]
Account registration feature has been added. This functionality uses implementation from user story #52. When a user opens the application, only if the user doesn't have a registered account, a prompt will ask to create an account with a username and name (they can't be empty field). A delete option is also added to the account page allowing user to delete his current account.

Sprint 6, Release 3 (3 stories, 34 points)

#71: Quiz with multiple choice option [13 points] [Status: Done]
Multiple choice option feature has been added. The user can select multiple choice option for card style in "study" and "quiz" session. If the option is selected, the system will provide four multiple choice as answers (one being correct and other three are randomly picked from all the other answers from the same deck). If the user selects the correct answer, then it will give a point to daily point, deck point and tag point accordingly. Those functionalities are implemented from user story #139. This user story also give points to single side card style.

#139: Point Statistics [13 points] [Status: Done]
This user story introduce the implementation of deckPoint, tagPoint, dailyPoint, achievementPoint and achievementTagPointJoin class and their corresponding dao. The detail of the data structure is explained here. The user can access his statistic page from his account page. The user can view his points sorted by date, by deck and by tag.

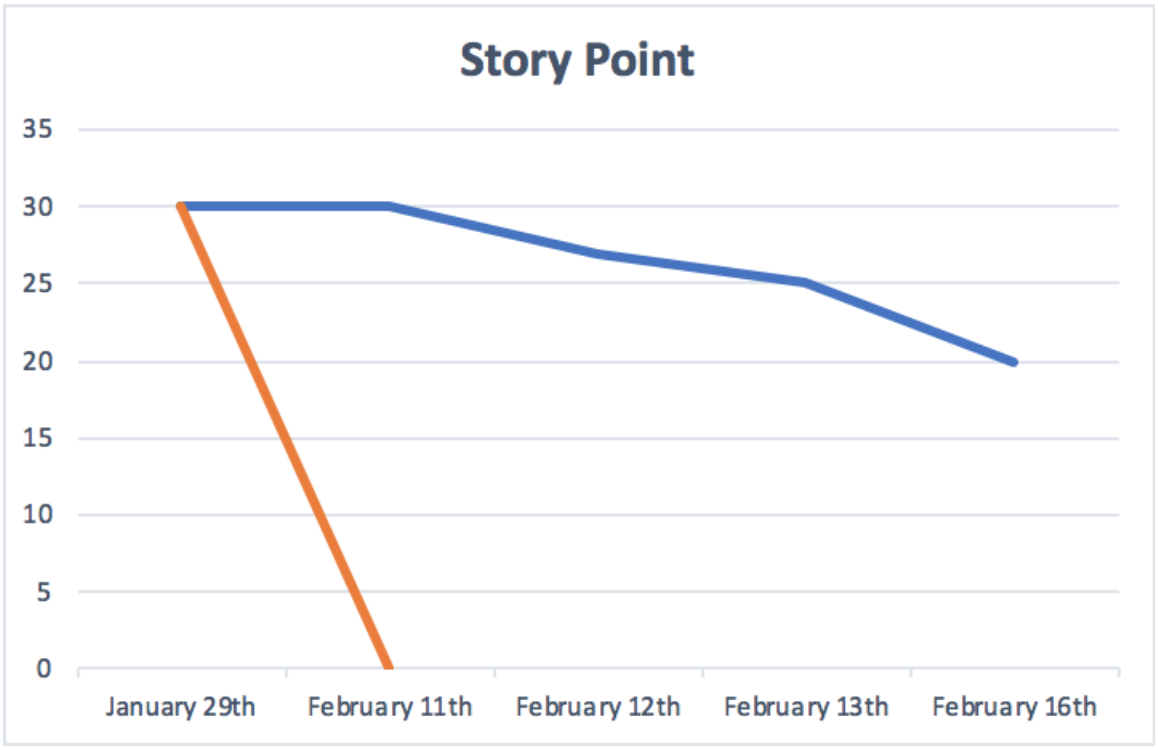
#184: Capture Image for QA [8 points] [Status: Done]
Camera feature has been added. When a user edit a question or an answer, he has the option to take a photo with his camera and upload it to the card.

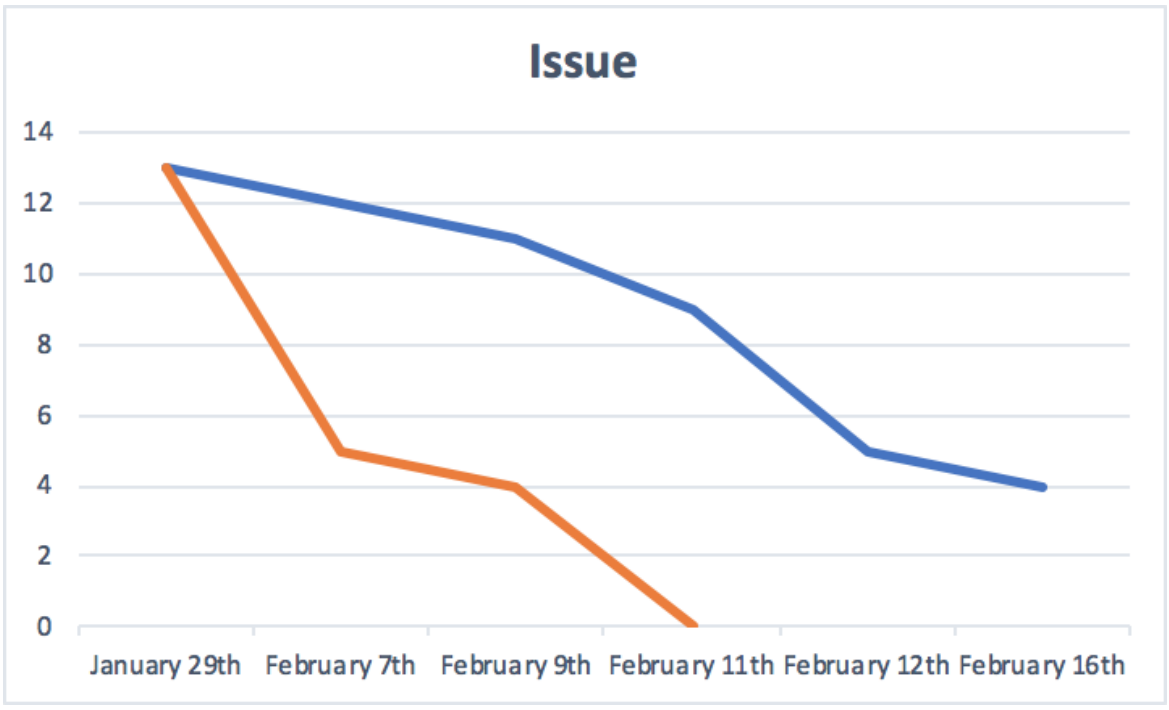
User stories for bug fixing:
#175: Central Database Fix [5 points] [Status: Done]
The new central database was never called to update the application's database when t he application is launched.

#200: Account Register Fix [2 points] [Status: Done]
Both user stories #72 and #132 from previous sprint needed to use a user account. After both stories are merged, the application creates a default user instead of asking for registration.

Burndown Charts

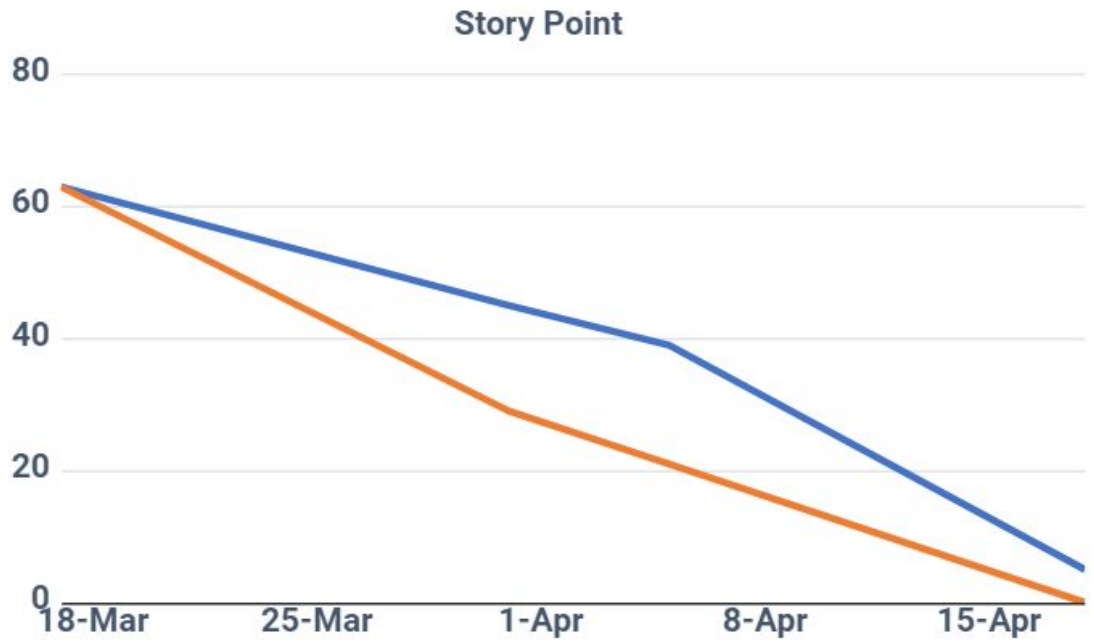
Release 1 burndown chart





Release 2 burndown chart

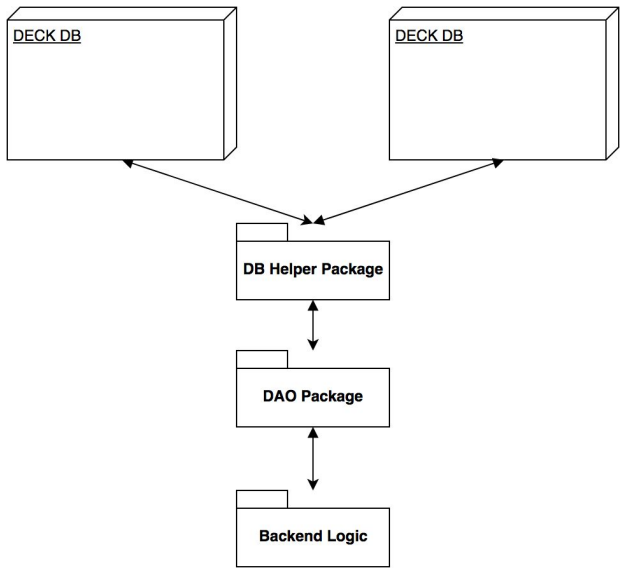
Release 3 burndown chart



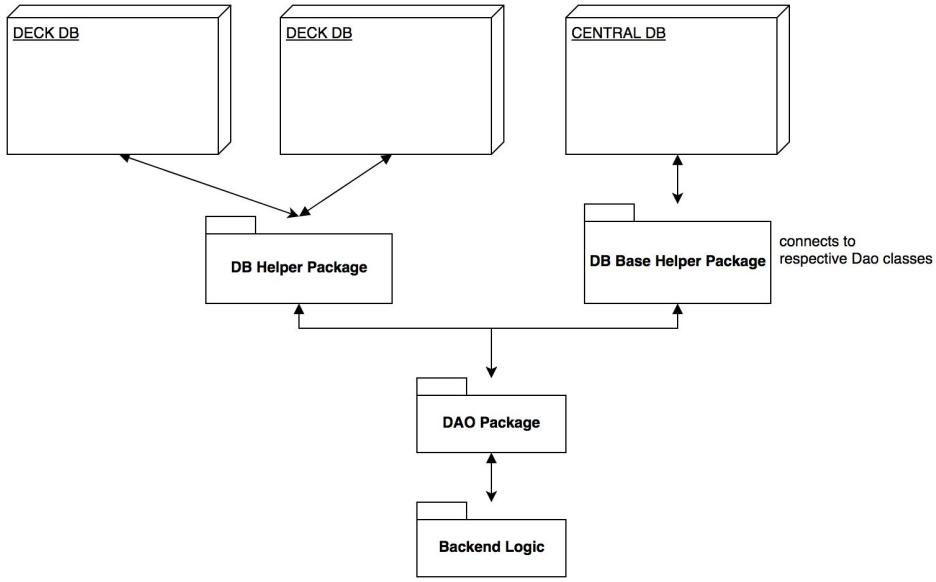
Overall Arch change

The largest architecture change in our project was altering the database. Originally each deck of cards was an individual database file. As we wanted to add tags and achievement points that were central to the functioning of the application, we needed to create a new database accessible. In order to not have a complete overhaul of the system, by making only one central database and migrating decks into objects (which would require refactoring much of the existing code), we kept the individual databases and also had a central database. The architecture for accessing the database was created. Tag is the only entity that exists in both the individual databases and the central database. The central database keeps track of all existing tags and the individual databases keep track of their own tags. The parallel helper package ensures the entities are speaking with the correct database.

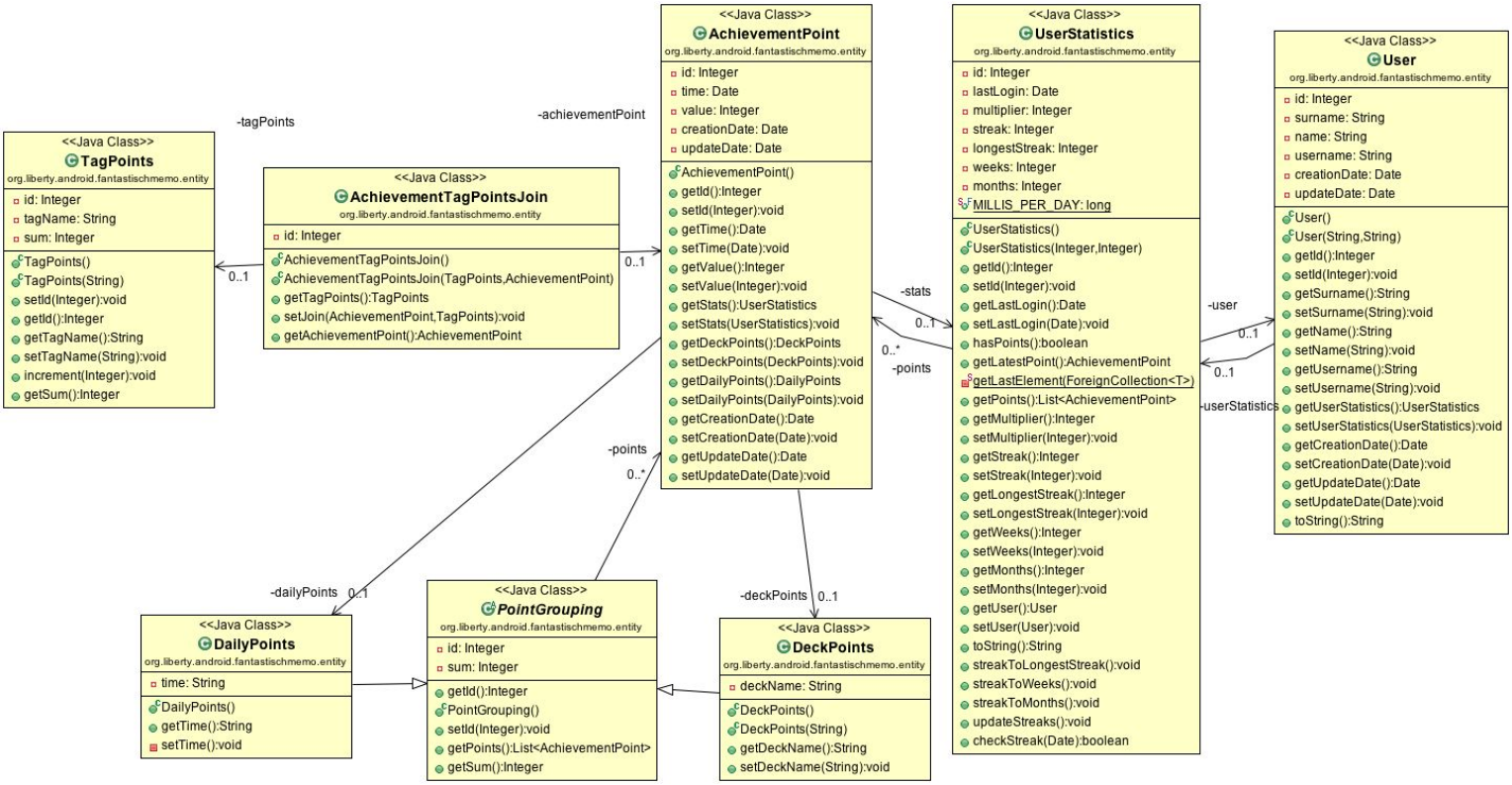
Before



After


DECK DB
DECK DB
CENTRAL DB
DB Helper Package
DB Base Helper Package
connects to
respective Dao classes
DAO Package
Backend Logic

Below is the class diagram for the achievement point system. It demonstrates the development from release two.



Name Conventions

Classes: Classes are mixed case with the first letter upper case. If the class consists of more than one word, every subsequent word follows the same convention (first letter capital). Classes names are nouns.

Interfaces: Interfaces are named like classes, mixed case with the first letter upper case. If the class consists of more than one word, every subsequent word follows the same convention (first letter capital). Interface names are nouns.

Methods/Functions: Methods and functions are mixed case with the first letter lower case. The name of a method or function is indicative of an action.

Constants: Constants are all uppercase letters. Each word is separated by an underscore ("_").

Variables: If one word, the variable is all lowercase. If more than one word, the variable is mixed case with the first word being lowercase and the subsequent words uppercase. Only iteration variables tracking a count are one letter.

Brackets and Spacing: All opening curly brackets for functions and classes are spaced one space away (on the same line) from rest of the statement. The closing bracket is placed on a seperate line after the last statement of the class or function. All indentations are a full tab from the previous indentation.

For example: Java naming conventions

Code

Key files: top **5** most important files (full path). We will also be randomly checking the code quality of files. Please let us know if there are parts of the system that are stubs or are a prototype so we grade these accordingly.

File path with clickable GitHub link	Purpose (1 line description)
app/src/main/java/org/liberty/android/fantastischmemo/entity/PointGrouping.java	Abstract class that defines the behaviour of classes used to filter point according to an object (deck or date).
app/src/main/java/org/liberty/android/fantastischmemo/entity/UserStatistics.java	Implemented achivementPoints so that a user can track his progress
app/src/main/java/org/liberty/android/fantastischmemo/dao/DeckDaoImpl.java	Added functions so the app can interact with the decks in the database. Also added a new deck rating to it.
app/src/main/java/org/liberty/android/fantastischmemo/ui/MultipleChoiceCardFragment.java	Implemented an entire new option to quiz user using a multiple choice system.
app/src/main/java/org/liberty/android/fantastischmemo/ui/GradeButtonsFragment.java	Added the point system to the previously existing options to take a quiz

Testing

Each story needs at least a tests before it is complete.
If some class/methods are missing unit tests, please describe why and how you are checking their quality. Please describe any unusually aspects of your testing approach.

List the **3** most important **unit test**s with links below.

Test File path with clickable GitHub link	What is it testing (1 line description)
app/src/androidTest/java/org/liberty/android/fantastischmemo/test/entity/AchievementPointTest.java	Updated from previous sprint to test getting and setting of new classes using mocks.
app/src/androidTest/java/org/liberty/android/fantastischmemo/test/db/DeckPointsDaoTest.java	Tests if interaction with ORM frameworks gives results expected from class methods.

app/src/androidTest/java/org/liberty/android/fantastischmemo/test/db/TagPointsDaoTest.java	Test for the join entity that creates a many to many relationship between TagPoints and AchievementPoints.

List the **3** most important **UI end to end (espresso) tests** with links below.

Test File path with clickable GitHub link	What is it testing (1 line description)
app/src/androidTest/java/org/liberty/android/fantastischmemo/test/ui/PointStatisticsUITest.java	Tests that UI components for displaying point statistics function correctly and assert that they appear at the right place.
app/src/androidTest/java/org/liberty/android/fantastischmemo/test/ui/AccountEditNameTest.java	Tests creating a new user and if the correct user information is displayed on the user's account page.
app/src/androidTest/java/org/liberty/android/fantastischmemo/test/ui/CreateDeleteAccountUITest.java	Tests to create a user using the registration popup and to delete the user in the account page.

Static Analysis

The static analysis tool that we are using is FindBugs. Since our codebase is mainly Java, the FindBugs tool analyses the Java programming language. In order to run the tool, we created a task in our build automation system. This was done by creating a new gradle script, called script-findbugs.gradle, that configured the FindBugs plugin to run on the whole project and generate an XML file at a specific destination. We then imported this gradle script into the main gradle script, by using the apply from keyword, in order for the new task to work.

In order to run the task, the following command would be executed: gradlew findbugs. This command would run the FindBugs static analysis on the entire project and generate the XML file. The gradle task was added to our Jenkins gradle task execution stage, where all the other tasks were being run to build the project. Once the task finished and the XML file was generated, it can be used by the Jenkins FindBugs plugin to generate the following report (an better/interactive report can be found by navigating to this link):

All Warnings	New this build	Fixed Warnings
434	3	3

Summary

Total	High Priority	Normal Priority	Low Priority
434	84	161	189

Details

Packages | Files | People | Categories | Types | Warnings | Origin | Details | New | Fixed | High | Normal | Low

Package	Total	Distribution
org.liberty.android.fantastischmemo	79	
org.liberty.android.fantastischmemo.common	5	
org.liberty.android.fantastischmemo.converter	41	
org.liberty.android.fantastischmemo.dao	2	
org.liberty.android.fantastischmemo.databinding	12	
org.liberty.android.fantastischmemo.downloader.anymemo	3	
org.liberty.android.fantastischmemo.downloader.common	6	
org.liberty.android.fantastischmemo.downloader.dropbox	2	
org.liberty.android.fantastischmemo.downloader.dropbox.entity	3	
org.liberty.android.fantastischmemo.downloader.google	38	
org.liberty.android.fantastischmemo.downloader.oauth	4	
org.liberty.android.fantastischmemo.downloader.quizlet	14	
org.liberty.android.fantastischmemo.entity	48	
org.liberty.android.fantastischmemo.modules	30	
org.liberty.android.fantastischmemo.provider	3	
org.liberty.android.fantastischmemo.queue	1	
org.liberty.android.fantastischmemo.receiver	2	
org.liberty.android.fantastischmemo.scheduler	1	
org.liberty.android.fantastischmemo.service	7	
org.liberty.android.fantastischmemo.service.cardplayer	2	
org.liberty.android.fantastischmemo.tts	9	
org.liberty.android.fantastischmemo.ui	101	
org.liberty.android.fantastischmemo.ui.loader	4	
org.liberty.android.fantastischmemo.utils	14	
org.liberty.android.fantastischmemo.widget	3	
Total	434	