# Wikipedia-Pen Pineapple Apple Pen Release 2 Summary

## Team members

| Name and Student id | GitHub id | Number of story points and ideal hours* that member was an author on. |
|---|---|---|
| **Amanda Wai, 27518714** (Team Leader) | amawai | [#20]: 2/8 points<br>[#22]: 1/13 points<br>[#41]: 1/2 points<br>[#45]: 2/3 points<br>[#54]: 1/13 points<br>**Total: 7 points**<br>**Ideal hours: 44** |
| Sophia Quach, 40012754 | sophiaquach16 | [#20]: 3/8 points<br>[#22]: 1/13 points<br>[#29]: 0.5/0.5 points<br>[#30]: 1/3 points<br>[#45]: 1/3 points<br>**Total: 6.5 points**<br>**Ideal hours: 40** |
| Jad Malek, 26345018 | jadmalek | [#22]: 2/13 points<br>[#28]: 0.5/0.5 points<br>[#30]: 2/3 points<br>[#41]: 1/2 points<br>[#57]: 1 point<br>**Total: 6.5 points**<br>**Ideal hours: 40** |
| Chen Jie Lu, 27754388 | SunXP | [#20]: 3/8 points<br>[#22]: 1/13 points<br>[#55]: 3/13 points<br>**Total: 7 points**<br>**Ideal hours: 44** |
| Maxim Nguyen, 27564171 | mnhn329 | [#22]: 3/13 points<br>[#54]: 3/13 points<br>**Total: 6 points**<br>**Ideal hours: 37** |
| Aman Bhandal, 27390858 | abhandal | [#22]: 3/13 points<br>[#56]: 3/13 points *(Not able to complete within this release and this task has been pushed into Sprint 6 of Release 3)*<br>**Total: 3 points**<br>**Ideal hours: 37** (presuming all 6 points were to be included in the calculation) |

| | | |
|---|---|---|
| Artem Mikhalitsin, 26349455 | artemmikhalitsin | [#22]: 2/13 points<br>[#24]: 1/1 point<br>[#27]: 1/1 point<br>[#53]: 2/2 points<br>**Total: 6 points**<br>**Ideal hours: 37** |

**Ideal hour estimate for all stories in each sprint was 140, and hence the total - 280 - will be divided accordingly based on the point assignment displayed below. Namely, number of story authored story points / total story points planned for Release 2 (45) * 280 hours = ideal hours for a developer.*

## Velocity

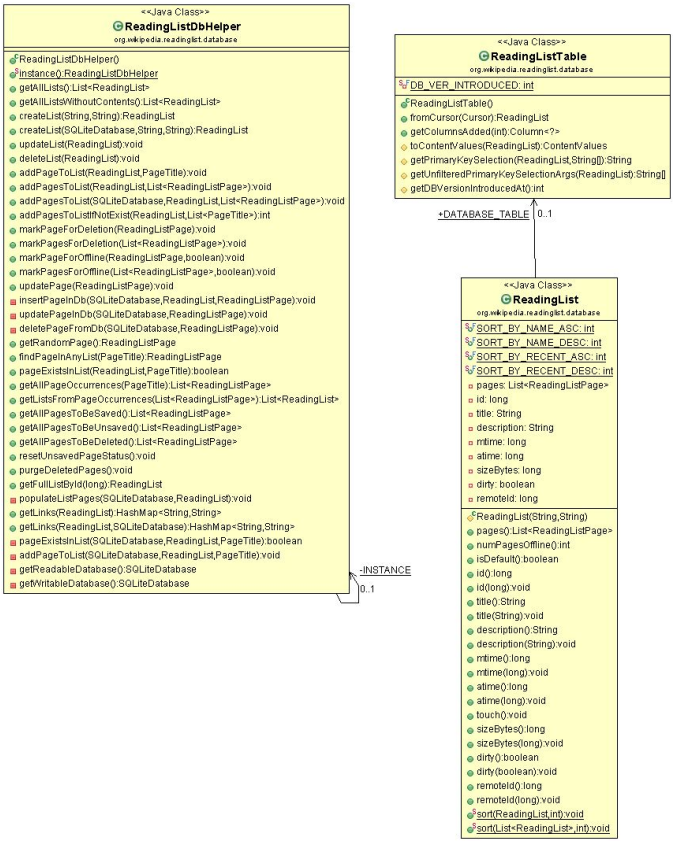
The commit for the second release has been tagged and can be accessed here.
Total: 11 stories, 42 points over 4 weeks
Iteration 3 (4 stories, 22.5 points)
Iteration 4, Release 2 (7 stories, 1 epic account for over Iterations 3, 4 and ultimately 6, 19.5 points)
*As described in the destination history task of the enhancement of trip planner user story, issues - and thus delays - encountered in implementing better trip integration workflow (#53) complicated the ability for a new search history database (and the other subtasks listed in the task in question) to be properly added into said workflow. While this task was initially considered to be additional functionality that would have elevated the look of the trip planner feature and improved user experience, it was still planned to be finished for release 2. Consequently, while a trip's destination history was not included in this release, it will be added later on in Sprint 6 (#70), along with a few other augmentations, to finalize and thus conclude the implementation of the trip planner feature.*

## Overall Arch and Class diagram

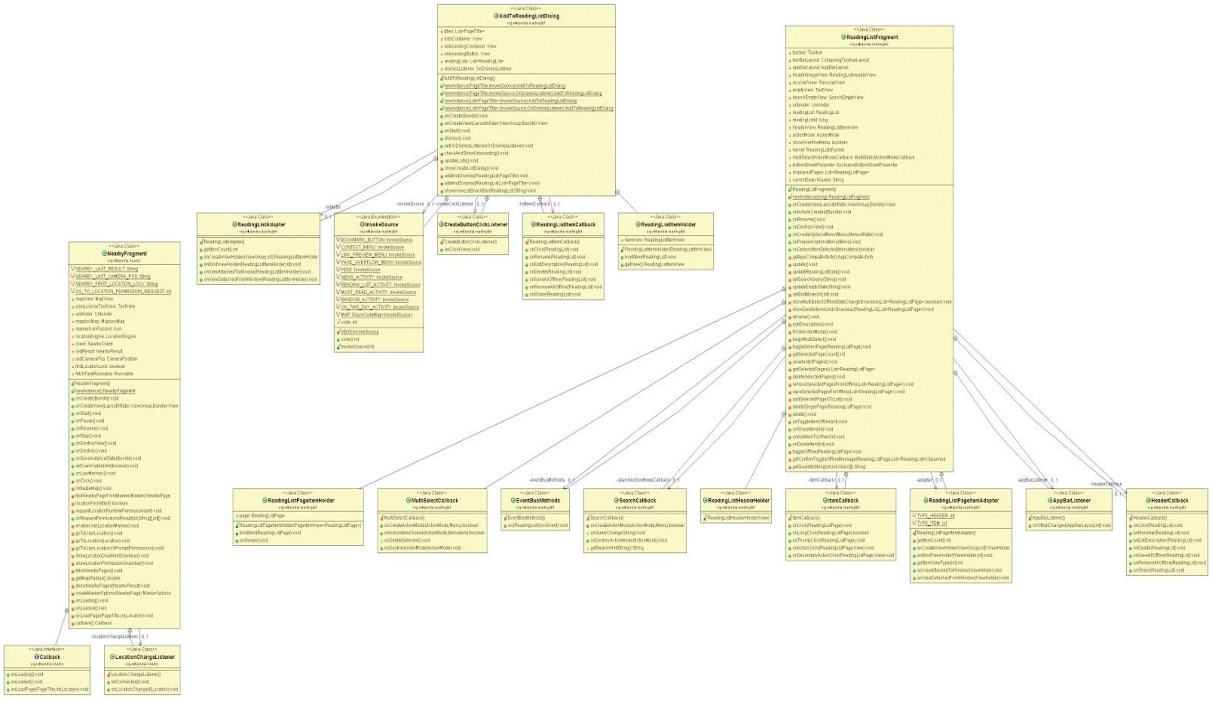
The diagrams recovered from the Wikipedia android application pertaining to the second (as well as the previous release) can be viewed here. Alterations made to the architecture over the course of Release 2 mainly include the addition of a travel package to the application's main source code, which consists of its own database table (and accompanying helper classes), fragments for the separate aspects of a trip planner, an integration controller for the trip planner, modifications made to the reading list and reading lists fragments, reading list database retrieval changes and adjustments applied to the reading list item view. All of these changes are thusly displayed below.

## *Share reading list feature*

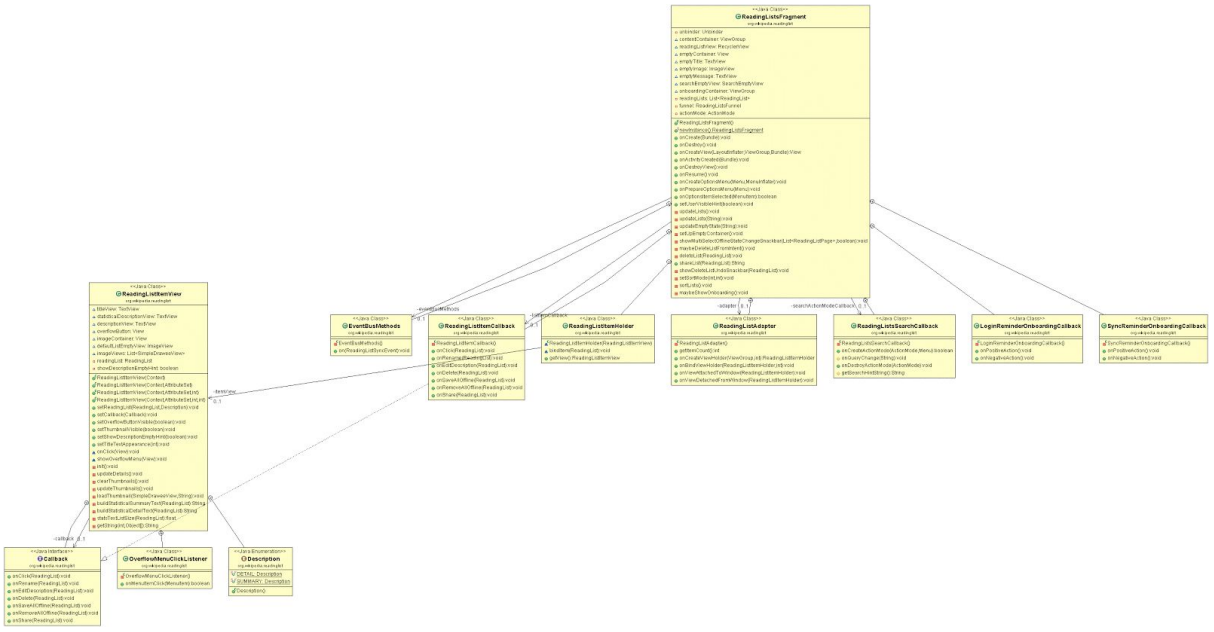
*Reading list database classes relevant to sharing implementation*




*Single reading list changes*

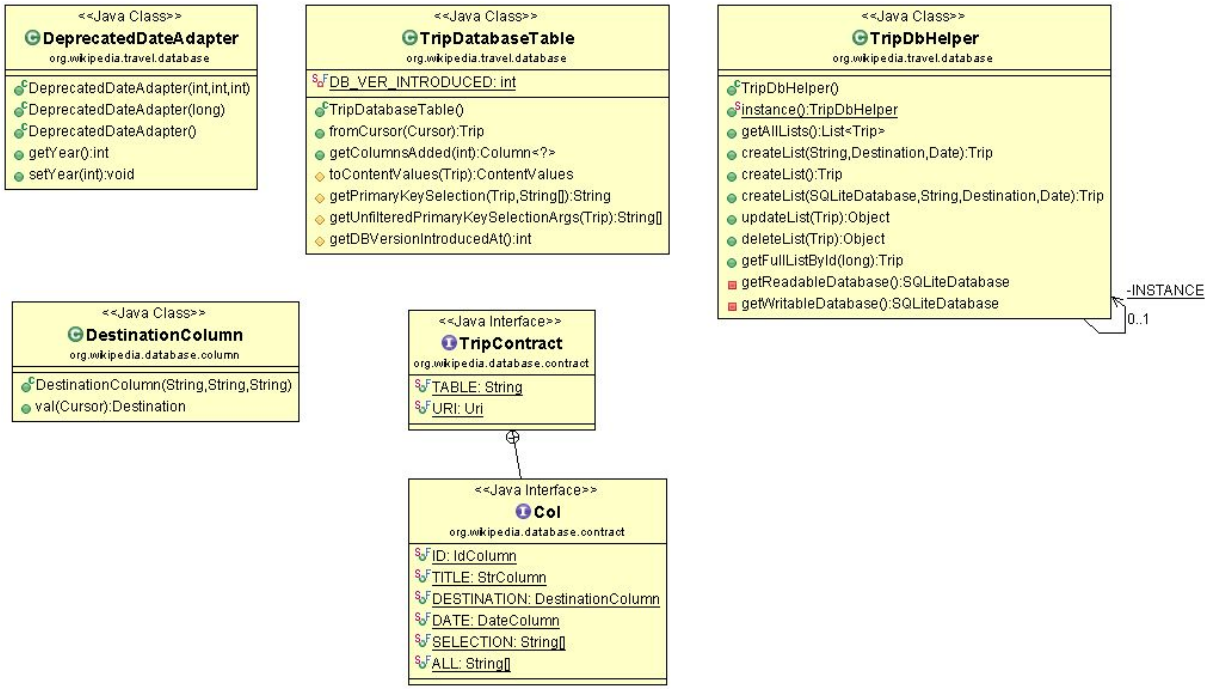


*Reading lists and reading list item view alterations*

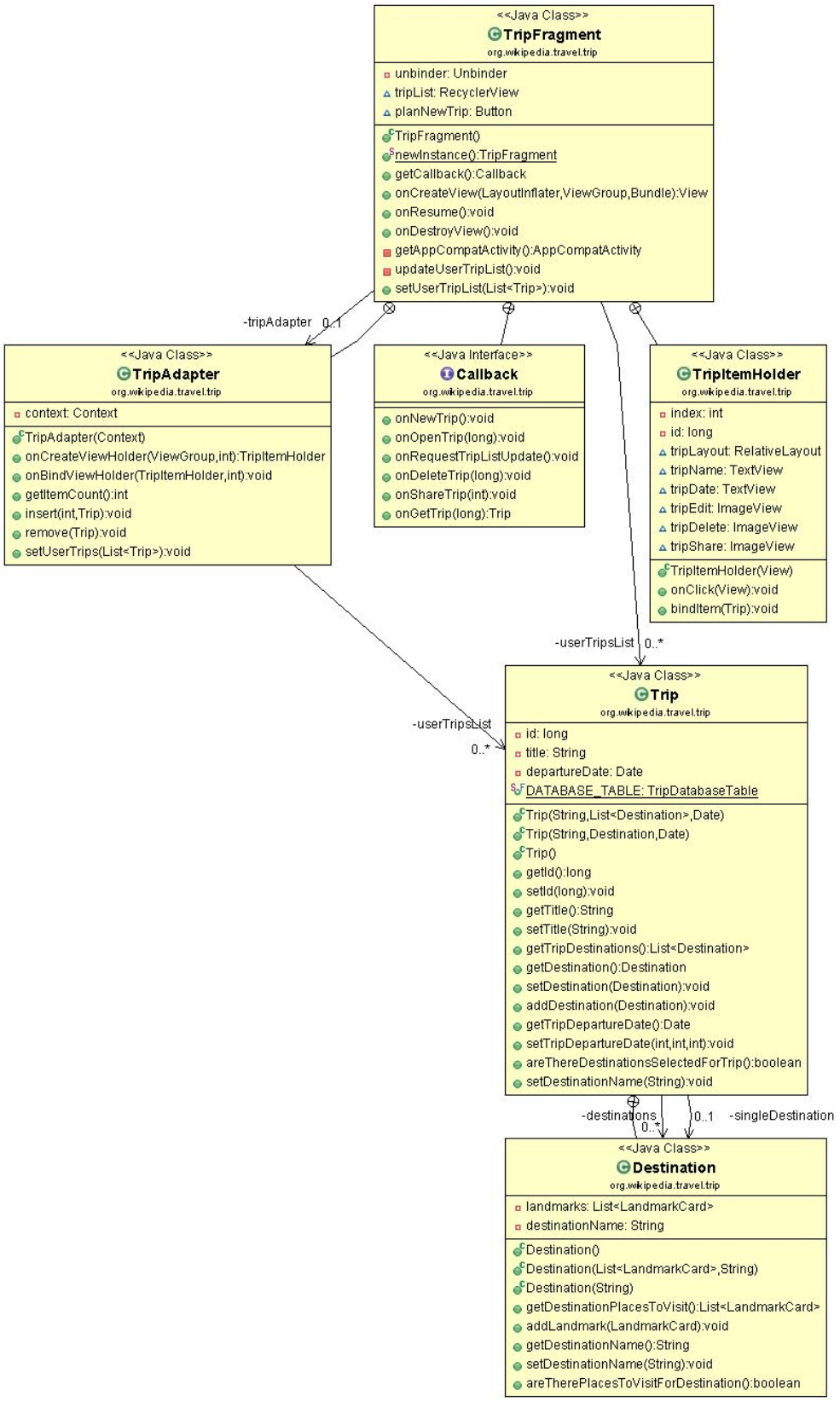


***Set up and enhancement of the trip planner feature***

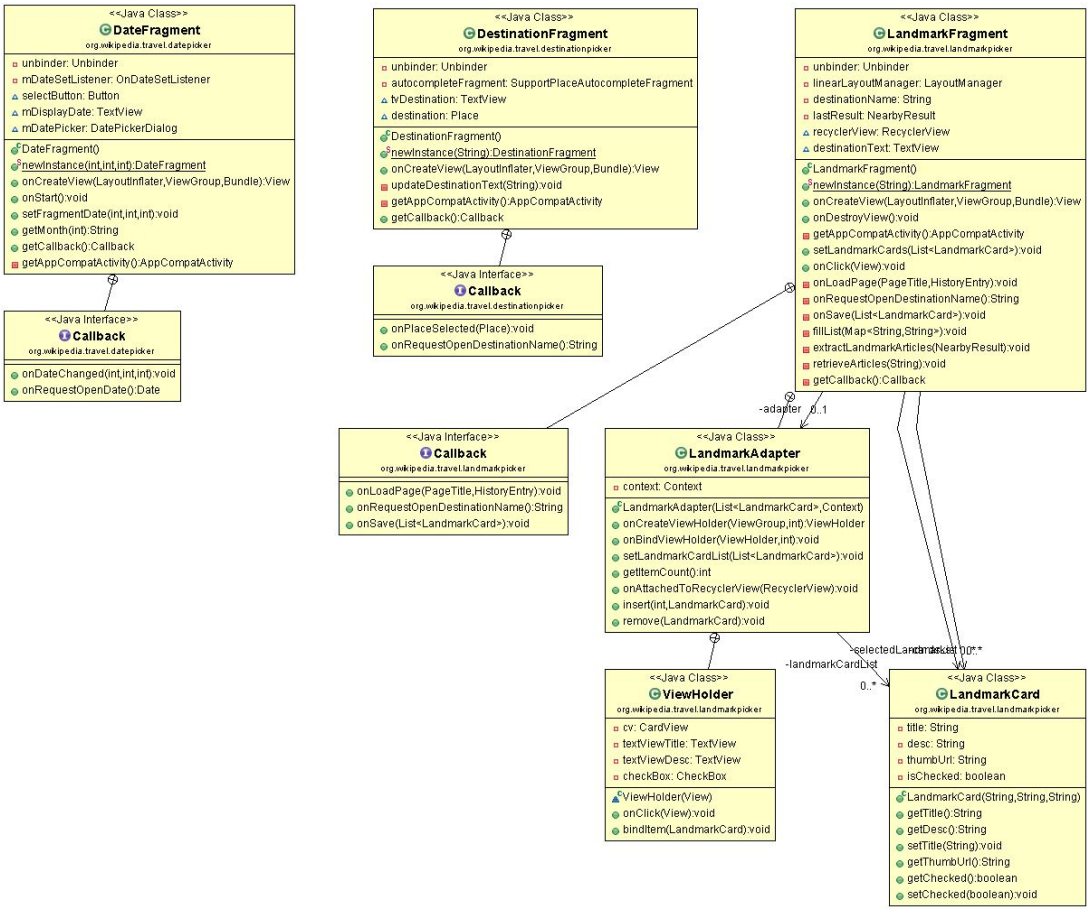
*Trip database*



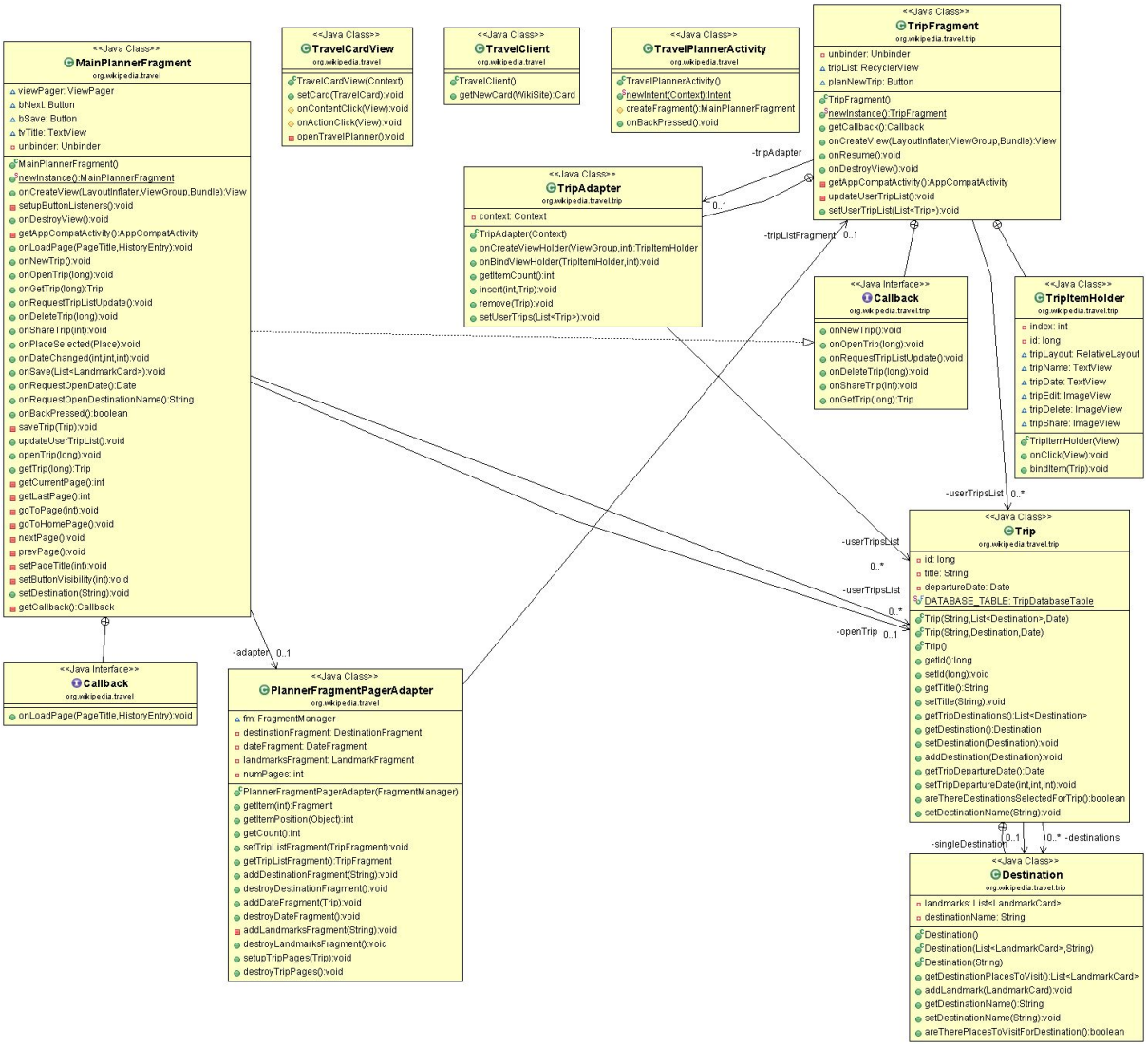
*Trip class and fragment*



*Date, destination and landmark picker implementations*



*Trip integration controller and its associations with trip*




## Plan up to next release

Total: 10 stories, 46 points over 4 weeks
Iteration 5 (5 stories, 25.5 points)
Iteration 6 (5 stories, 1 story part of the trip planner epic account to be completed in this release, 20.5 points)

## Infrastructure

List all libraries, frameworks, etc. You only need to **discuss changes** in your infrastructure.
The Google Places Library was added to the the project for the Trip Planning feature so that users can select a destination.
The testing framework UiAutomator was added to the project to allow for interaction between Android elements not accessible through Espresso.

## Name Conventions

The naming conventions used is: Prefixing the type, as in `type_foo_bar.xml`. Examples: `fragment_contact_details.xml`, `view_primary_button.xml`, `activity_main.xml`. Similarly, this convention applies to ids for graphical components, such as `feature_button_next`.

In addition, all file names should include the name of the feature to which it belongs.

Other coding conventions can be found in the development team's wiki.

## Code

***5** most important files (**full path**)*

| File path with clickable GitHub link | Purpose |
|---|---|
| android-wikipedia-390/blob/master/app/src/main/java/org/wikipedia/travel/MainPlannerFragment.java | The purpose of this file is to establish a central source for information for all Travel Planner fragments - data will be propagated upwards to be stored here. |
| android-wikipedia-390/blob/master/app/src/main/java/org/wikipedia/travel/trip/Trip.java | This purpose of this file is to model a Trip used in the travel planner. |
| android-wikipedia-390/blob/master/app/src/main/java/org/wikipedia/travel/trip/TripFragment.java | Thie fragment offers the option to create a new trip or view, edit and delete past trips. |
| android-wikipedia-390/blob/master/app/src/main/java/org/wikipedia/travel/destinationpicker/DestinationFragment.java | This fragment allows the user to select their trip destination using the Google Places Library. |
| android-wikipedia-390/blob/master/app/src/main/java/org/wikipedia/travel/landmarkpicker/LandmarkFragment.java | This fragment displays a list of landmarks (articles) associated to the destination selected by the user and is also the last step before a trip is saved to the database. |

## Testing

***3** most important **unit test**s with links*

| Test File path with clickable GitHub link | What is it testing |
|---|---|
| android-wikipedia-390/blob/master/app/src/test/java/org/wikipedia/travel/trip/TripTest.java | This tests the functionality of the Trip class (getters, setters) |
| android-wikipedia-390/blob/master/app/src/test/java/org/wikipedia/travel/database/TripDbHelperTest.java | This tests the functionality of the Trip table data operations |
| android-wikipedia-390/blob/master/app/src/test/java/org/wikipedia/travel/DeprecatedDateAdapterTest.java | This tests that the overridden date methods work properly |

***3** most important **UI end to end (espresso) tests** with links*
NOTE: When running these tests, if the blue "Welcome To Wikipedia" screen appears, "skip" must be pressed for these tests to run successfully.

| Test File path with clickable GitHub link | What is it testing |
|---|---|
| android-wikipedia-390/blob/master/app/src/androidTest/java/org/wikipedia/espresso/travel/LandmarkEspressoTest.java | Tests that once a destination is selected, related articles are displayed in the Landmark list |
| android-wikipedia-390/app/src/androidTest/java/org/wikipedia/espresso/travel/DatePickerEspressoTest.java | Tests if the date that is picked corresponds to the one the user selected. |
| android-wikipedia-390/app/src/androidTest/java/org/wikipedia/espresso/travel/NewTripButtonTest.java | Tests if the "New Trip" button exists upon loading the Trip Planner Activity. |

## Finished SHORT Story summaries

Points: 13, Priority: High, Risk: High, Iteration #3
**Story #22**: As a user, I would like to plan a trip using the Wikipedia application.
Feature: Trip Planning
This story involved setting up the trip planner feature that was to be enhanced in the downstream sprints. All of the team members were involved in both implementing and testing the relevant code. Signed off by the stakeholder Maxime Lamothe, the implementation involved setting up a main activity (that could render multiple fragments) for the trip planner, a trip class (to create trip objects with), database and its associated classes, as well as date, destination and landmark picker fragments.

Points: 13, Priority: High, Risk: High, Iteration #4
**Story #26**: As a user, I would like to plan a trip using the Wikipedia application.
Feature: Trip Planning
This story involved augmenting the trip planner feature that was established in the Set-Up trip planner user story. All of the team members were involved in both implementing and testing the relevant code, and mainly comprised of completing the trip planner integration controller, adding in the sharing functionality for a trip, and improving upon the landmark and destination code written in the previous sprint for this feature.
*As underscored in the Velocity section of this document as well as in the story's issue on Github, while this story was not completed, the remaining destination history task was moved to a finalization of trip planner user story that will be completed in Sprint 6 of Release 3, consequently concluding the implementation of the trip planner epic.*

Points: 2, Priority: High, Risk: Low, Iteration #4
**Story #41**: As a developer I would like to improve the current file structure of the travel package (implemented for #22) for better readability and consistency.
Feature: Trip Planning
This story involved refactoring the code written in the third iteration (specifically the code authored for the trip planner setup) to meet coding conventions established by this team and can be accessed here, and to improve the organization of all the pertinent code files. Jad and Amanda were responsible for the refactoring, and further information about the task separation can be observed in the issue for this user story.

Points: 3, Priority: High, Risk: Low, Iteration #4
**Story #30**: As a developer, I would like to be augment the current development environment to help with debugging builds by separating the current pipeline into build stages and improving the CI to enable notifications (for tests and builds).
Feature: Not associated to a feature in particular but is rather an improvement upon the development environment, Addition of travis build stages and email notifications for every build.
This story involved updating the travis config file to separate out the current continuous integration pipeline into stages and enforce the sending of notifications upon both travis build successes and failures. This was executed by Jad and Sophia - the former was responsible for the introduction of stages and the latter for email notifications revealing the culprit commit, a link to it and a link to the build itself (for more details, see the last section of this document).

Points: 3, Priority: High, Risk: Low, Iteration #4
**Story #45**: As a developer, I would like to add some espresso tests for fragments from #22, the trip planner set-up, and #26.
Feature: An improvement upon the current test coverage, Addition of end-to-end tests, this is for the Trip Planning feature implementation (as the reading list feature was covered mainly by unit tests).
This story involved writing espresso, end-end tests for the indicated feature's user stories that were carried out during this release. These tests were designed by Amanda and Sophia, and principally cover intents, trip, date, destination and landmark fragments that collectively contribute to the important functionality of the trip planner. They were particularly requested to be improved by the stakeholder after the third iteration and were addressed in the fourth iteration.

Points: 8, Priority: Medium, Risk: Low, Iteration #3
**Story #20**: As a user, I would like to share my reading list to external services.
Feature: Share reading list
The other main feature for this release besides the trip planner feature (set up and enhancement), this story involved adding in the ability for a user to share one of their reading lists, via an external service, with others. Amanda, Sophia and Roger were involved in both implementing and testing the relevant code. Signed off by the stakeholder Maxime Lamothe, the implementation involved the addition of a share button, creation of the lists and link to be shared and the core sharing functionality itself.

Points: 0, Priority: Medium , Risk: Low, Iteration #4
**Story #60**: As a developer, I would like to construct acceptance tests for each of the code and feature based user stories in Release 2.
Feature: Addresses both the Trip Planning and Share reading list feature implementations.
This story involved writing acceptance tests for each of the code and feature based user stories implemented during this release. These tests were designed by Jad and Sophia, and can be seen within the germane story and task issues (for particulars on the stories/tasks accounted for and the individuals responsible for making these tests, the issue link for this story contain all these details).

Points: 1, Priority: Medium , Risk: Low, Iteration #3
**Story #24**: As a developer, I would like a mockup of the features for the coming iteration.
Feature: Mockups are for both the Trip Planning and Share reading list features.
Mainly conducted for design and documentation purposes, this story involved constructing mock ups for the set up of the trip planner feature. Worked on by Artem Mikhalitsin, these mock ups were made and placed into the team's Github Feature Mockups wiki page.

Points: 1, Priority: Medium , Risk: Low, Iteration #4
**Story #27**: As a developer, I would like a mockup of the features for the coming iteration.
Feature: Mockups are for Trip Planning feature.
As with the story described prior, this story involved adjusting the mock ups from the previous iteration, Sprint 3, in order to account for the enhancements that were to be made to the trip planner feature. Worked on by Artem Mikhalitsin, these mock ups were improved and inserted into the team's Github Feature Mockups wiki page.

Points: 0.5, Priority: Low , Risk: Low, Iteration #3
**Story #28**: As a developer, I would like to create a burn down chart in order to keep track of the outstanding work left to do during the sprint.
Feature: Accounts for the velocity characterizing the team's work on both the Trip Planning and Share reading list features.
This story was purely for documentation purposes and involved constructing a burndown chart in order graphically represent the amount of work to do versus the time left remaining in the sprint in question. Filled in by all the team members, it was used to analyze the team's velocity and determine developer progress throughout the iteration.

Points: 0.5, Priority: Low , Risk: Low, Iteration #4
**Story #29**: As a developer, I would like to create a burndown chart in order to keep track of the outstanding work left to do during the sprint.
Feature: Accounts for the velocity characterizing the team's work on the Trip Planning feature.
Identical in nature to the previous story, but pertaining to sprint 4, this story was purely for documentation purposes and involved constructing a burndown chart in order to graphically represent the amount of work to do versus the time left remaining in the sprint in question. Filled in by all the team members, it was used to analyze the team's velocity and determine developer progress throughout the iteration.

## CI stages and notification

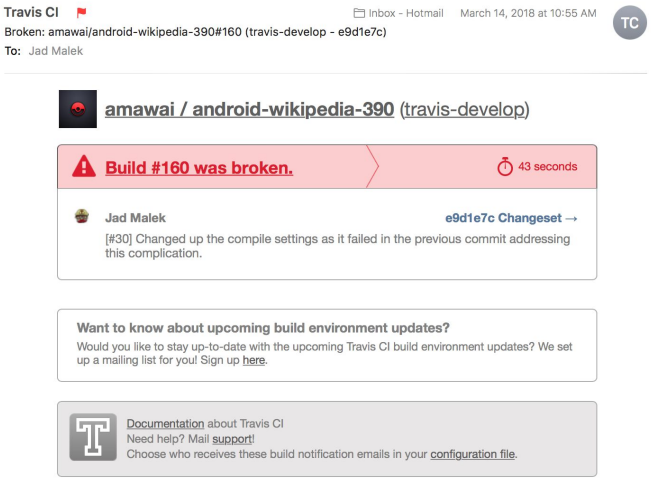
*Separation of CI pipeline into stages*

The link to this team's continuous integration environment is: https://travis-ci.com/amawai/android-wikipedia-390. As with Release 1, whenever new commits are made and pushed to a repository, or a pull request is made, Github notifies Travis and Travis runs specific instructions written in the project's .travis.yml file. During the course of this release, the pipeline was separated into compile, all tests, release one, release two, and deploy stages:

Compile stage - essentially builds all the application's dependencies
All tests stage - Unchanged from Release 1's CI environment but now established as its own stage in the pipeline
Release 1 tests stage - Builds all the applicable tests for the Release 1 features
Release 2 tests stage - Builds the applicable unit tests for this release's features. This will be enhanced in the subsequent release to include the UI tests for complete coverage of tests authored by Pen Pineapple Apple pen
Github releases stage - Builds to deploy when the build in question contains tagged commits (will adjusted in the next release to authenticate via an username and password as opposed to the current set up with the Github OAuth token)

Furthermore, a release three stage will be added within the next two sprints to account for the all tests constructed therein and static analysis on pull requests within the repository will be integrated within the pipeline as a whole.

*Example of Notification with Error*




The above image is an example of an email notification that was sent to all members of the development upon a failed build when attempting to commit to the travis-develop branch of Wikipedia repository. The notification system is set up so that emails are always sent out to the aforementioned recipients on both build passes and failures. The resulting email indicates the status of the build (with a link to inspect the build log), author of the commit, commit message and a link to the commit so that the recipients can view the commit and attempt to resolve the failure (particularly in the case where the travis build was unsuccessful). In this example, the commit - authored by Jad Malek -

addressed the separation of the continuous integration environment into the described pipeline. The developers can, subsequent to analyzing the error message provided by Travis within the build log, select the commit link within the email, inspect what has been altered and determine potential solutions to the failure of the Travis build. Upon doing this for the illustrated example failed build, it was deduced that the germane ./gradlew command could not be called in the manner shown on line 19 of the associated commit. Consequently, the inspecting developer realized that they had to specify the appropriate path from which to indicate this command, and the issue was thereafter resolved.