Shawn Salm

Professor Chen-Hsiang (Jones) Yu

COMP7200 – Mobile Application Development

20 October 2017

Firetime – Final Project Report

Problem Description

Tracking the actual total amount of time that one spends performing a given activity during the day can be a daunting task. This is especially true for activities that are performed off and on throughout the day. Tracking the accumulation of the total amount of time spent on an activity over the course of a day, a week, and/or a month can be difficult and time consuming. One example of this is an office worker tracking the total amount of time they spend standing at their desk. Many offices today feature desks that can be raised up and brought down allowing the employee to either stand or sit at their desk while working as a way of improving their health. Other activity examples that one might want to track how much time they use performing the activities include reading, walking, a student doing homework or assignments, and cooking. One might also choose to track how they burn their time such as by watching TV or playing video games.

Analysis of the Problem

There are two main tasks of this problem that causes it to be difficult to manage. First, recording the start and stop times, either on paper or entering them into a computer can be a chore. Second, the accumulation and reporting these time periods for a day, a week, or a month can be tricky. An easy to use Android application that will let the users enter in activities and track the amount of time they spend performing a given activity with a simple toggle button is a solution to the first task. A push on a button featured beside the activity name will start tracking the time. Another push of the button will then stop it. The program will automatically record these times so that the total amount of time spent for the current day, week, or month can be displayed for each activity. The target user is anyone ranging from age 5 to 100 that are looking to track the amount of time they spend on given activities are the target users. Most likely however, adults with busy

lives who want to track their time will be the core customers. Having an easy way to track the amount of time one spends during the day on different activities can help them to know whether they should increase the amount of time performing an activity that is good for them or help one to reduce the amount of time they spend doing unproductive activities.

Design

The design of Firetime is an MVC Android application that uses the built in SQLite database to track the user's time for given activities that are arranged into categories. Built in user interface components are used and therefore, no third-party libraries were needed for this first release version. The Model part of the application is split into two layers, a data access layer (DAL) and a business layer. The DAL is split into four logical pieces: a static class with the database schema, a helper class that extends SQLiteOpenHelper to create the database and handle versioning, repository classes to execute the SQL statements, and model classes that represent the tables in Java. The business layer is split into two areas, domain models and services. Domain models represent the objects of the application in a real-world fashion. For example, the domain model for an activity contains a start and stop methods for tracking time. The service objects make an easy to use layer for performing actions such as retrieving activities and history records and is the API for the application.

Model	Business Layer	Services
		Domain Objects
	Data Access Layer	Repositories
		Database Helper Classes
View	User Interface Layer	Layout files
		Menu Files
		Values files such as strings.xml and colors.xml
Controller	Application Layer	Activities
		Fragments
		Adapters

Figure 1 – The layers that make up the MVC pattern in Firetime.

The general flow of the application is for a user to interact with the View or User Interface Layer by way of layout files, menus, etc. These views produce events that the Controller or Application Layer handles. Firetime being an Android application, these classes are usually Activities, Fragments, or Adapters. Finally, the controllers make calls to the Model or Business Layer by way of services and domain models which in turn utilize the repositories and database helper classes to access the SQLite database. Below is a detailed description of the model part of Firetime's MVC design.

Data Access Layer (DAL)

Firetime's SQLite database schema contains three tables. One table to hold categories (tbl_activity_category) which are used to arrange activities. The activities table (tbl_activity) holds information about the user's activities and can have many time history records which are stored in the activity history (tbl_activity_history) table. The fields and relationships between the tables can be seen in the ER diagram below.

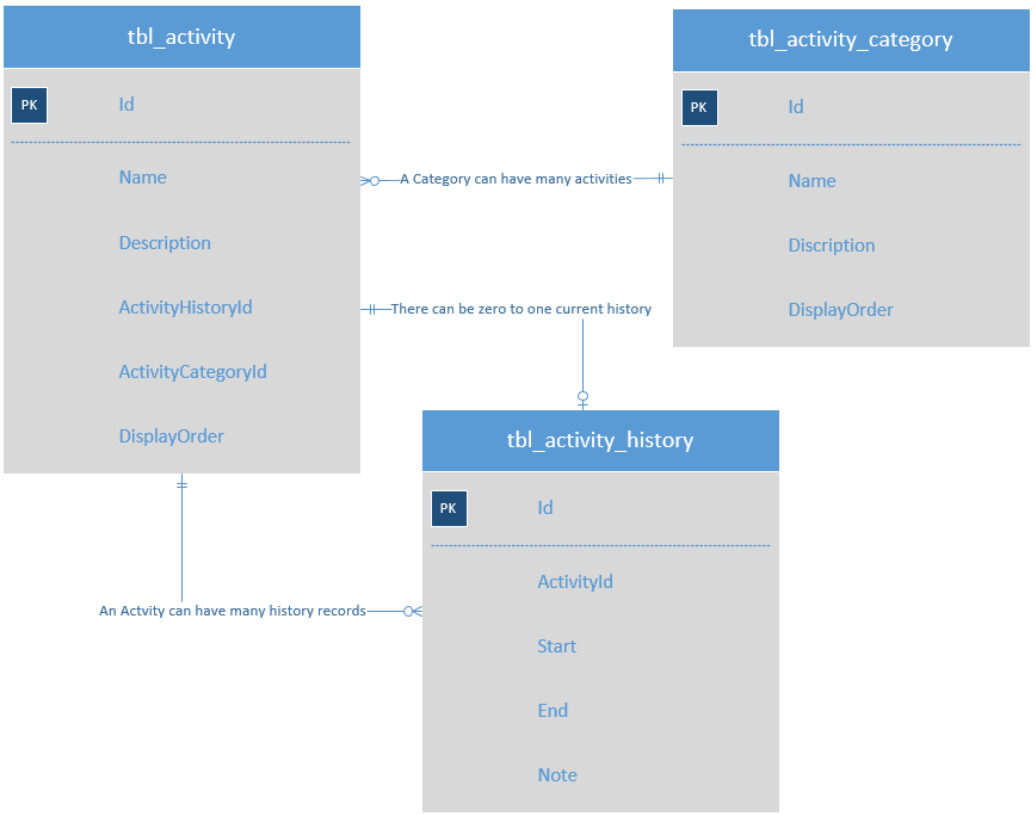


Figure 2 – Firetime's database schema.

The code design of the DAL for Firetime can be seen below in the UML class diagram. Model classes such the Activity class and the ActivityHistory class represent the data stored in the SQLite database tables. The FiretimeDatabaseHelper class manages the creation, updates, and access to the SQLite database. The FiretimeDBSchema class contains the definitions of the tables. Finally, the repository classes such as the ActivityRepository class and the ActivityHistoryRepository class contain and execute the SQL statements needed to retrieve the data from the SQLite database and return the model classes.

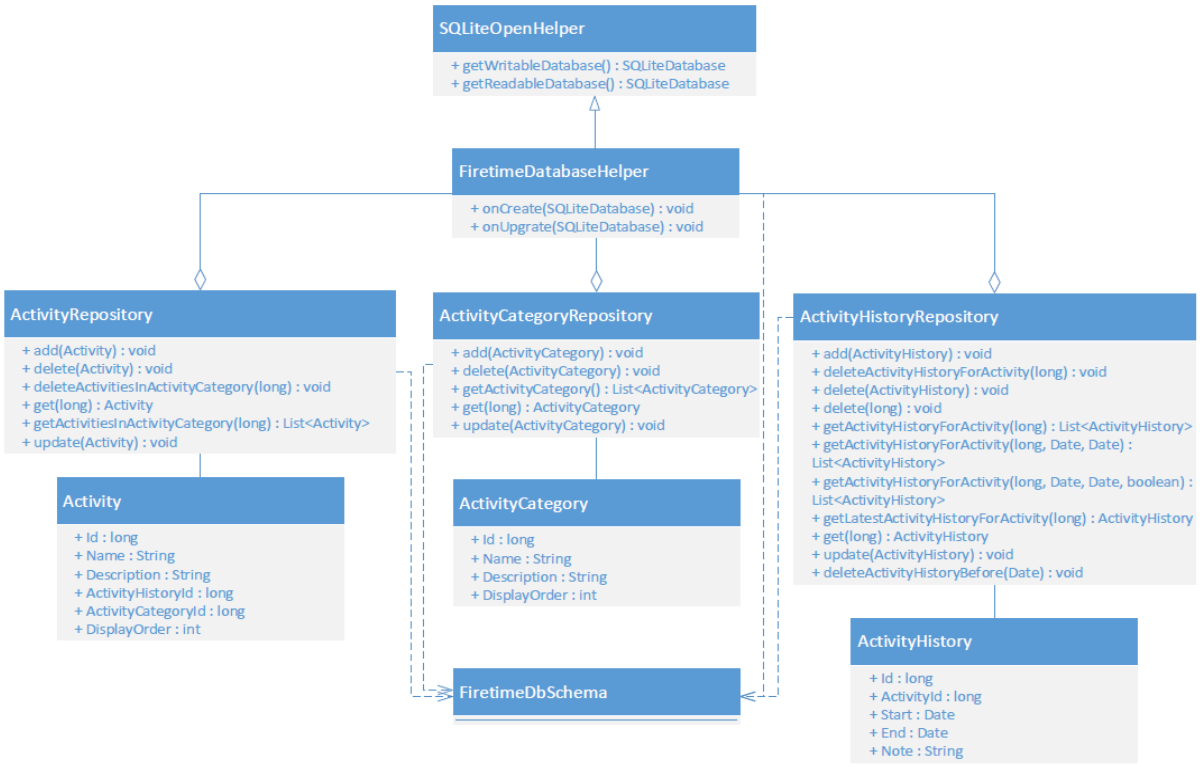


Figure 3 – UML class diagram of Firetime's data access layer

Business Layer (Domain Models)

Domain models represent the application from a business perspective. Firetime contains three domain models: ActivityDomainModel class, which represents an activity that the user wants to start tracking time on, ActivityHistoryDomainModel which represents a single period of time that the user performed the activity, and ActivityCategoryDomainModel which represents the

categories in which the user can organize their activities into. The UML class diagram for Firetime's domain models is featured below. Note that the class ActivityDomainModel contains start() and end() methods for starting and stopping tracking time for a given activity.

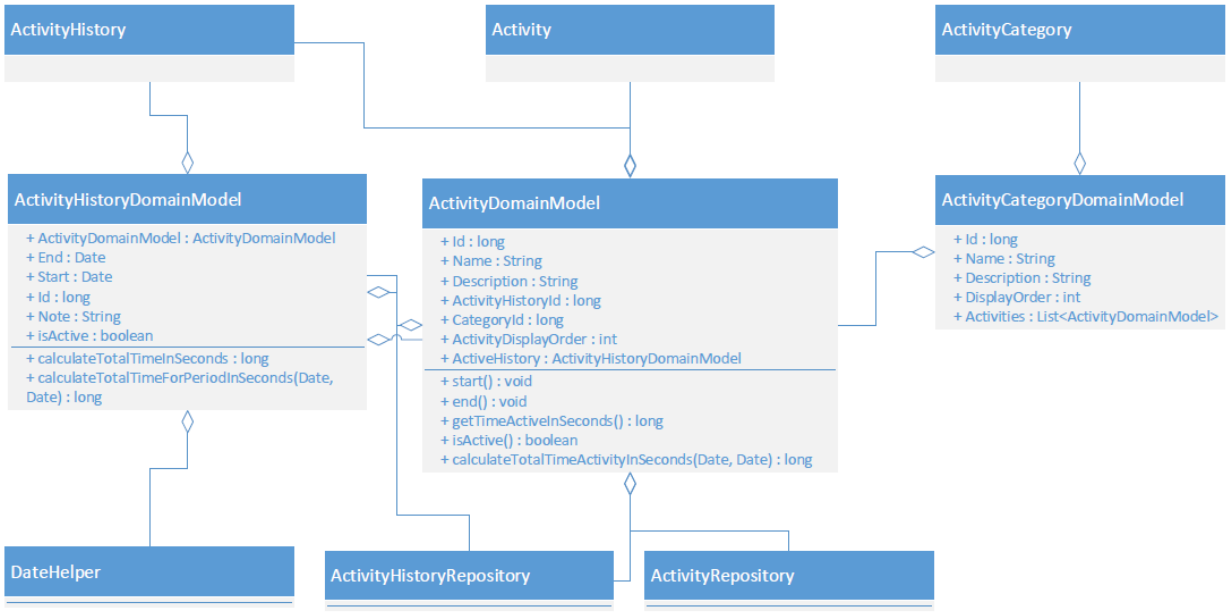


Figure 4 – Firetime's domain models UML class diagram.

Business Layer (Services)

Finally, the service classes of Firetime's model layer form the API (application programming interface) in which the controllers utilize to access functionality within Firetime. The services, such as ActivityService, ActivityHistoryService, and ActivityCategoryService classes provide the controllers methods that return, save, and delete domain objects and know how to convert DAL models from and to Domain Models. The ActivityHistoryService, for example, provides methods for getting activity history records over a given period of time. It can also be used to delete all the history records from a given date back, which could be used in the future to clean up old data. Below is a UML class diagram for Firetime's services that shows how they related and utilize the repository classes as well as return domain models.

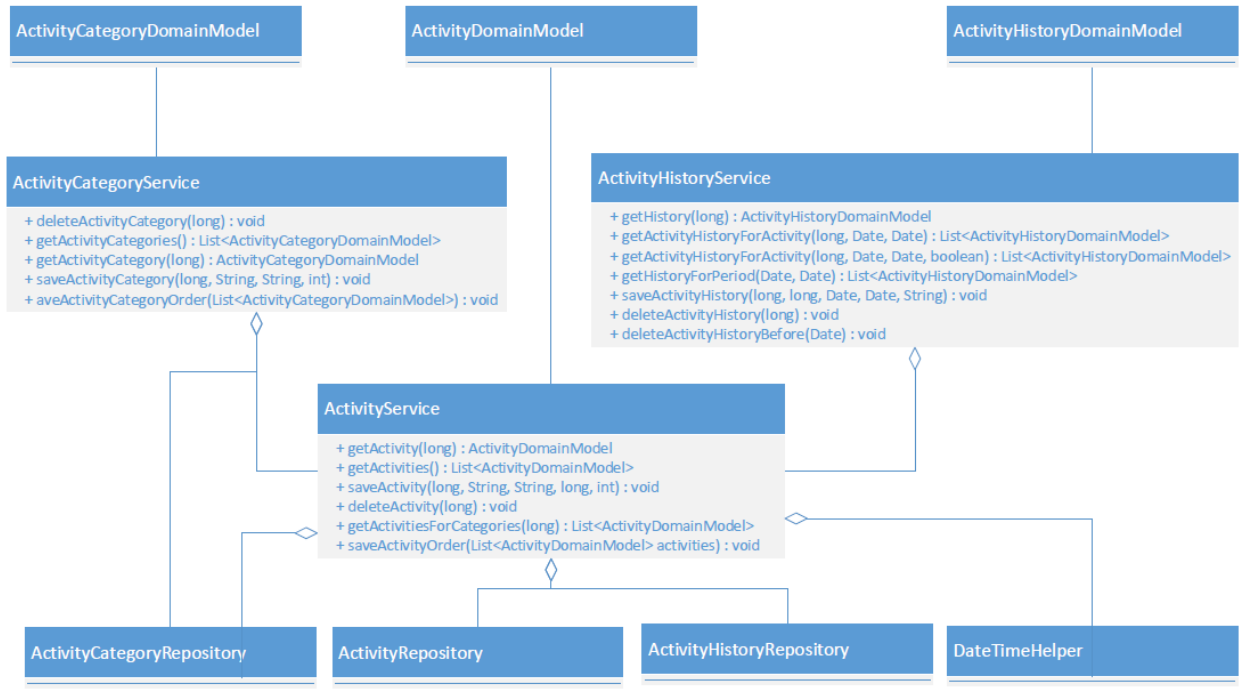


Figure 5 – Firetime's services UML class diagram.

A few benefits from this design are listed below:

- The Database Access Layer can easily be changed from a local database to a networked database without changing any of the business, application, and user interface layers. Usually done using interfaces and dependency injection.
- Unit testing each layer makes code more SOLID.
- Separation of concerns allows the maintenance of the application to be simplified.
- Application layer (controllers) have a simple, easy to understand API.

Application's Sequence Flow

One of the benefits of the service layer is that it hides the complexity of the model from the rest of the program. Below is example code for how a controller such as a fragment in Android could use the ActivityCategoryService in order to get the list of categories and their activities domain models from the database.

```
ActivityCategoryService service = new ActivityCategoryService(getActivity());
mActivityCategoryDomainModels = service.getActivityCategories();
```

Code Example 1 – Getting categories with all their activities.

The UML sequence diagram below shows all that happens when the CurrentActivitiesFragment object passes the list of ActivityDomainModel objects to the CurrentActivitiesAdapter object in order to bind the data to a list control on the main screen of Firetime.

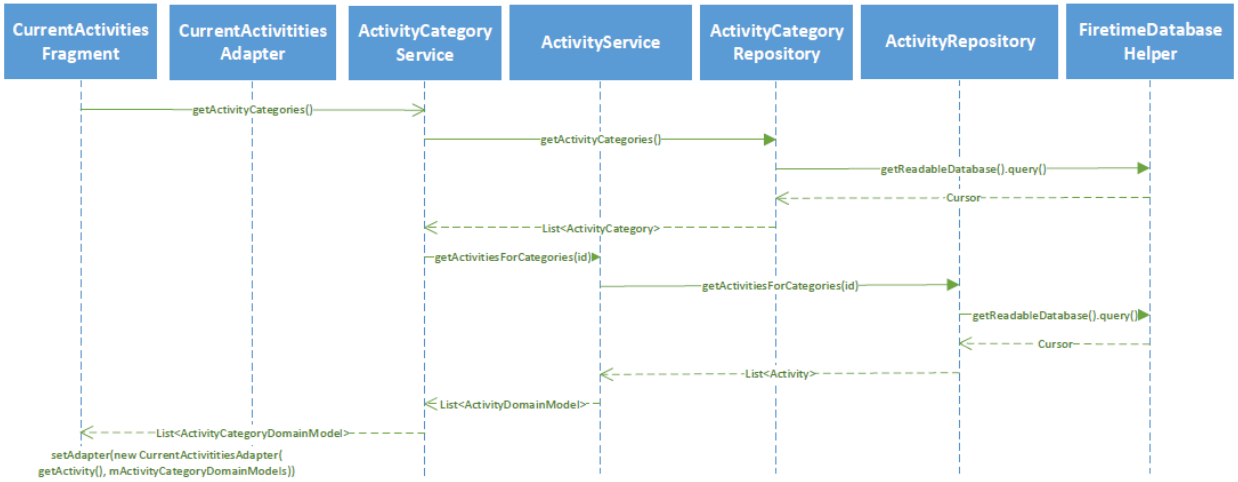


Figure 6 –UML sequence diagram for passing the categories and activities to an adapter.

Another example of how the business layer of Firetime hides the complexity from the controllers can be seen when an activity button is pushed causing the application to either start to stop tracking time for a given activity.

```
if (activityDomainModel.isActive()) {
    try {
        activityDomainModel.end();
        timer.setBackgroundColor(ContextCompat.getColor(mContext, R.color.darkred));
    } catch (Exception e) {
        e.printStackTrace();
    }
} else {
    try {
        activityDomainModel.start();
        timer.setBackgroundColor(ContextCompat.getColor(mContext, R.color.darkgreen));
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Code Example 2 – Domain model is used to start and stop tracking time for an activity.

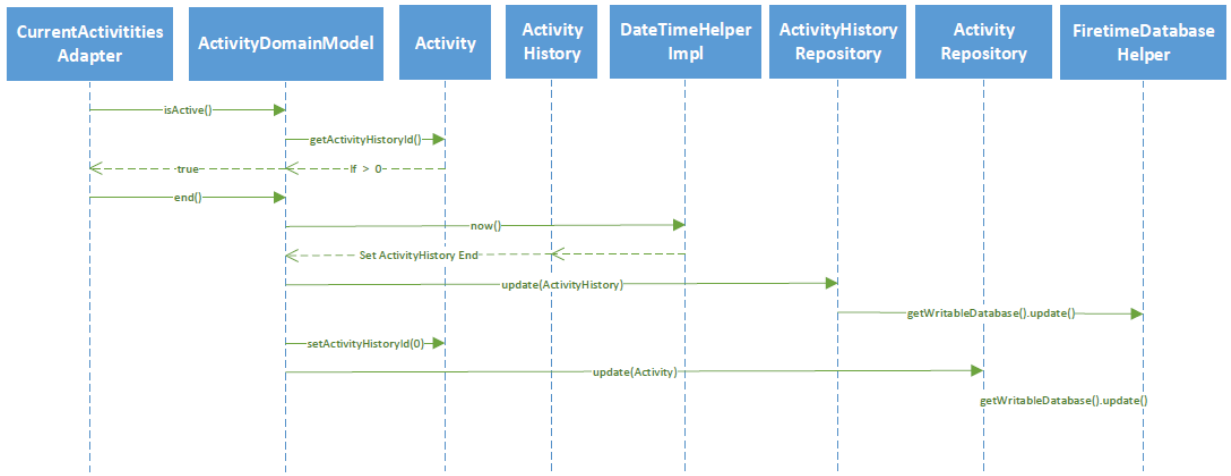


Figure 7 –UML sequence diagram for stopping tracking time for an activity.

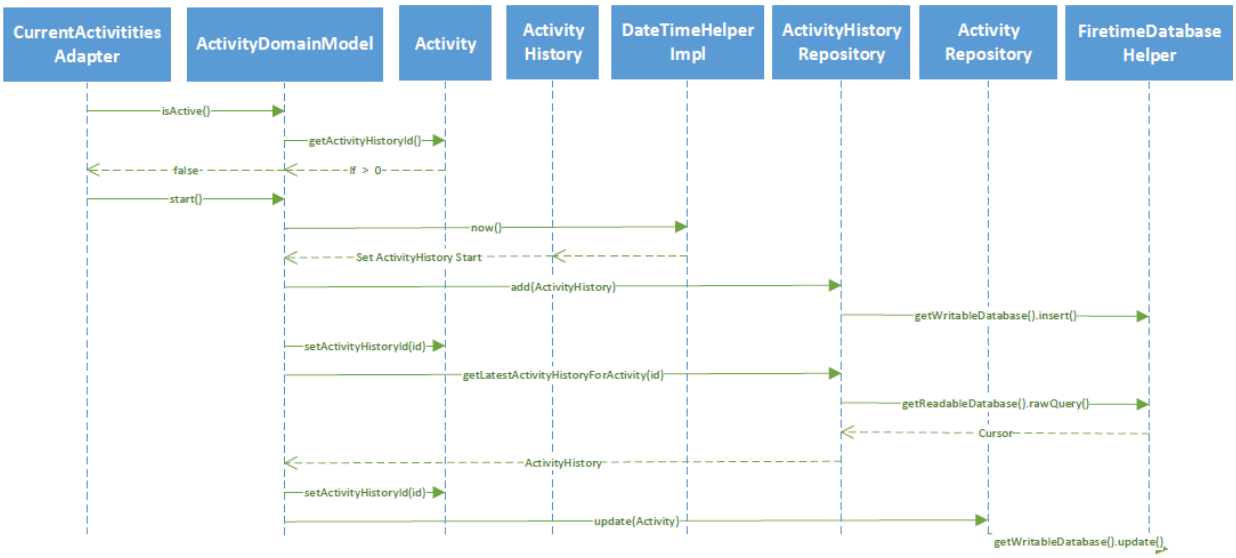


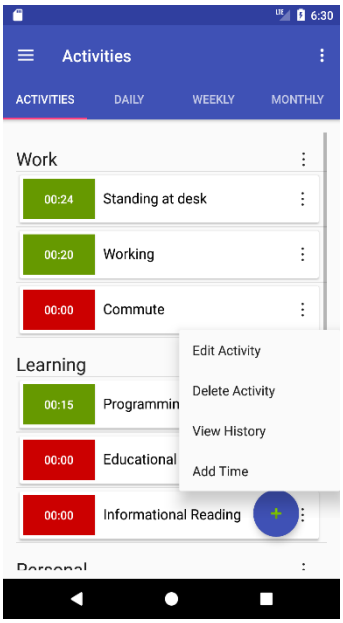
Figure 8 –UML sequence diagram for starting tracking time for an activity.

Implementation

The data access layer and the business layer classes have been implemented with unit tests. The core functionality of the UI displaying the amount of time for activities as well as the ability to start and stop time uses built-in Android components such as the ViewPager, ExpandableListView, FloatingActionButton, and NavigationView. The navigation items are made up of the drawer, toolbar menu, and popup menus. Firetime runs on Android version 8 (Oreo) and above and uses version 1.8 of the Java language.

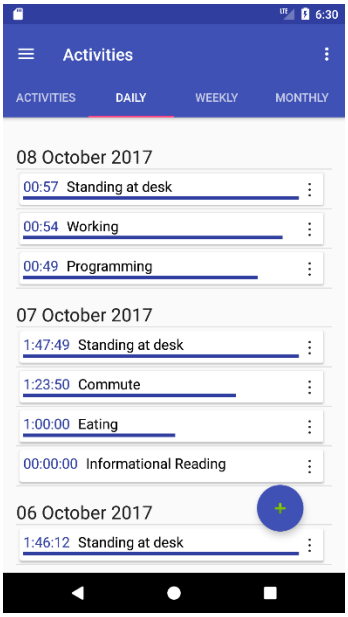
Evaluation

The main functionality of Firetime version one is to track the user's time on given activities. Below is a screenshot of the first screen users see when starting Firetime. The buttons to the left of the activities such as "Standing at desk" or "Working" control when the application starts and stops tracking time turning green when it is and red when it is not. The activities are organized by categories such as "Work" or "Learning" which can collapse or expand to hide or show their activities. Both the categories and activities have popup menus to the right of them. These menus allow the user to perform additional functionality such as to edit, delete, and order activities, or add time manually if one forgets to use the timer buttons.

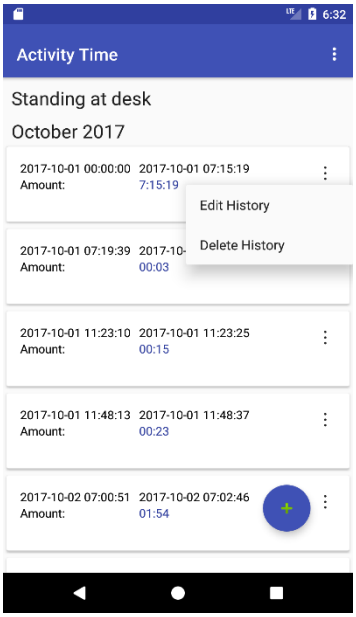


Screenshot 1 – The main activities screen.

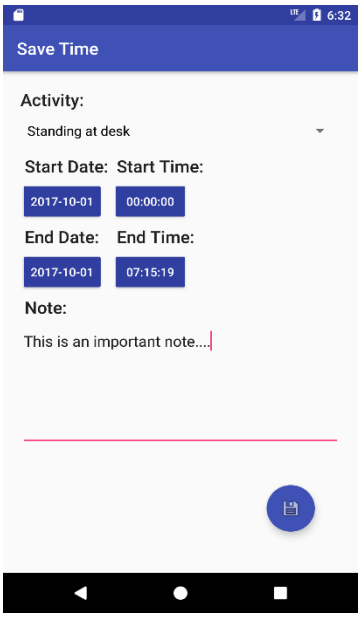
The users can also view their history of time spent on activities grouped by daily, weekly, and monthly time periods by tapping on the "DAILY", "WEEKLY", or "MONTHLY" tabs. For example, the daily screen shows the past seven days activities with a bar displaying relative the other activities' time in the section or same day. A popup menu is also available that allows the users to view the details of each day, week, or month as well as edit or delete them.



Screenshot 2 – The DAILY tab showing the past seven days of activities.



Screenshot 3 – History details for the "Standing at desk" activity.



Screenshot 4 – Adding time manually to the "Standing at desk" activity.

Instructions

Activities are any action, event, or task that you wish to track the amount of time you spend doing on a daily, weekly, or monthly basis. You may add, edit, delete, or reorder an activity. To add an activity, tap the add button at the bottom right corner of the activities view. To edit or delete an activity tap on an activity's menu. You can also reorder the activities within a category tapping on the category's menu. When you start an activity, you create a history record. The primary way to create activity history records is by pushing the timer button to start and stop the activity on the activities view. You can also manually add, edit, and delete a history record. Finally, categories are currently simply a way of organizing activities on the activities view. You can add, edit, delete, and reorder categories by tapping on the categories menu item. Note that you can only delete a category if it does not contain any activities.