**Simple Scheduler**

Jason Santana, Matthew Pinneo, and Matthew Spitzley

**Abstract**

Simple Scheduler is an Android app for time management. Users can schedule and manage their plans quickly and easily, and never miss a due date with customizable reminders. Users can choose to synchronize their data by signing in with Google.

## **Revised Requirements**

### **Tools and Technologies**

1. Android Studio

This is our primary IDE for development. The front end Java code as well as the interface will be developed in Android Studio

1. Android

It will run on Android version 9 or newer.

Will not have a specific interface for tablets.

1. Java

Java is the main language of the front end application.

1. Visual Studio Code

IDE we will use for PHP and SQL.

1. MySQL

The open source database we will use.

1. PHP

Our language for communication with the MySQL database.

### **Requirements List**

1. Task List Screen
   1. A button that will show a listbox of categories
      1. Displays all of the existing categories as well as an “All” and “Edit” option
         1. Selecting a category will cause the Task List screen to only display the tasks of that category.
         2. Selecting “All” will display all tasks, and selecting “Edit” will display the category edit screen.
   2. A list of tasks will be displayed for the user. Users will be able to read the details of the task: name, time, and date.
      1. When a task is selected it can be edited or marked as complete
      2. Tasks are sorted first by date, then alphabetically.
         1. Dates will automatically be updated based on what time zone they are in.
      3. Each task falls under a header based on the time.
         1. Headers include: Late, No Date, Today, This Week, Next Week, Next Month, and Later.
         2. Late tasks have a color separated date.
   3. A settings button that leads to the settings screen
   4. A new task button that leads to the new task screen
2. Category Edit Screen
   1. Show a list of categories sorted alphabetically.
   2. Selecting a category allows the user to change its name or delete it.
      1. If a category is deleted, the existing tasks with that category will be changed to “no category”
   3. Selecting the add button allows the user to create a new category.
      1. This lets the user name the new category then it is added to the list of categories.
3. New Task Screen
   1. This screen incorporates many text boxes and checkboxes for the user to manually enter information in about their task
      1. Name textbox which will have the text “New Task” by default. This text box will not be allowed to be empty.
         1. An empty text box will result in the application telling the user they cannot make the task until it has a name.
      2. Which category this task falls under
         1. If no category is selected then when the task is made is it marked as “no category”
      3. Date to be completed, this field is optional.
         1. When filling this out the user can choose from a calendar for which day the task is.
         2. The other way to fill this out is to manually enter in a date.
            1. The way the user enters the date manually will depend on how they chose to read dates in the settings.
      4. Time to be completed, this is optional.
         1. This will be filled out by a scroller for the hour and minute of the day.
      5. Once a date is selected, options for recurring and notifications will be displayed.
         1. A setting for whether this task is recurring
            1. If checked, selects when the task should reoccur
            2. Tasks can recur every day, weekdays (Monday through Friday), weekly, monthly, or yearly.
         2. A checkbox for whether to get email notifications.
            1. If checked, allow the user to choose when the notification occurs.
         3. A checkbox for whether to get push notifications.
            1. If checked, allow the user to choose when the notification occurs.
4. Edit Task Screen
   1. Tapping a task outside of the completion checkbox allows the user to edit the task offering the exact same options as the new task screen.
5. Calendar Screen
   1. Displays a calendar of the month. In each day that has a task, it shows the name of the task.
   2. Undated tasks are displayed separately from the calendar.
      1. Selecting the undated will show the undated tasks in the same way selecting a day shows the day's tasks.
   3. The category listbox is accessible in the calendar screen as well.
      1. Selecting a category will then display only tasks of that category.
   4. The settings menu is also accessible from the calendar screen.
   5. If the user selects a day with no task it will bring them to the new task screen with the day already filled in for that selected day.
   6. If the user selects a day with a task already in it, a screen will display the tasks for that day.
      1. A new task button will appear with the list.
   7. An option to display the calendar in a yearly view.
      1. This view will not show specific tasks.
      2. Days with tasks will be highlighted.
      3. Each month can be selected to view the more detailed monthly view.
6. History Screen
   1. Accessed by swiping left from the starting screen, the history screen will show completed tasks.
      1. They will be sorted by date, then alphabetically.
   2. Tapping a task on the history screen will “select” it.
   3. A select all button will select every task.
   4. An edit button to select multiple, specific tasks.
   5. Once at least one task is selected, the user can delete the selected task(s) permanently, or restore it as an uncompleted task.
7. Settings Screen
   1. Accessible from the history, task list, and calendar screens.
   2. A “Help” button which will display a brief description of Simple Scheduler’s features.
   3. An option to toggle between 12 hour and 24 hour time.
   4. The ability to change how the user wants to read dates.
      1. Month/Day/Year (2/8/2021)
      2. Day/Month/Year (8/2/2021)
      3. AbbreviatedMonth. Day Year (Feb. 8 2021)
      4. Day AbbreviatedMonth. Year (8 Feb. 2021)
      5. Month Day Year (February 8th 2021)
      6. Day Month Year (8th of February 2021)
   5. An option to change the screen seen on startup to the task list screen or the calendar screen.
   6. An option to select a default time for when a notification occurs.
      1. By default the notification will be 1 hour prior to the event occurring.
      2. If the task is created with less than an hour until it occurs it will immediately send out the notification.
   7. An option to toggle late notifications, which will notify the user they have missed a deadline.
      1. Users can choose these to be push, email, or both.
   8. An option to toggle all push notifications (this will remove them from the task add and edit screens).
   9. An option to sign in with a Google account.
      1. A button for a google sign in where the user will enter their email and password, then be connected to google.
   10. If the user is signed in, an option to synchronize their tasks to the database.
   11. If the user is signed in, an option to toggle all email notifications using the email from the Google account (this will remove them from the task add and edit screens).
8. Database
   1. Category table
      1. category
   2. Task table
      1. task\_name
      2. datetime
      3. Category (Category table FK)
      4. recur
      5. email\_notifications
      6. push\_notifications
      7. Completed
   3. The Sign in will not allow a user to have multiple devices on the same account.
   4. If the user is not connected through their google email and tries to run the app in two different instances each device will only act on what is locally stored.

## 

## **Design Description**

Our Simple Scheduler program will be made in Android studio with Java. As seen in the class diagram the very base of our program relies on the Task class. What each task will need is a String name, a Category category, a ZonedDateTime time to be due, a String recur, a ZonedDateTime email, a ZonedDateTime push, and a boolean complete. By default the name will be “New Task,” the category will be another class that is used by the task class. The ZonedDateTime will be using the java package java.time, which should handle time zone changes according to the phone's internal clock. The email and push will be ZonedDateTime because these are to let the class know when the user would like their notifications to be sent out. Finally the complete boolean is to know whether or not the task is already done.

The functions in the Task class will be as follows: getName, setName, getCategory, setCategory, getTime, setTime, getRecur, setRecur, getEmail, setEmail, getPush, setPush, isComplete, setComplete. The getName function will return the task name as a String. The setName function will take a String input and replace the Task’s name. The getCategory function will return the Category object assigned to this Task. The setCategory function will take a Category object to set the Task’s category. The getTime function will return a ZonedDateTime which is the due date of the Task. The setTime function will take a ZonedDateTime to set the Task’s time. The getRecur function will return a String which will correspond to one of 5 options the user has in the application: daily, weekdays (Monday through Friday), weekly, monthly, or yearly. The setRecur function will take a String and set when or if the task will recur. The getEmail function will return a ZonedDateTime that is the time when the user will receive an email notification for the given Task. The setEmail function will take a ZonedDateTime and set when the email notification will occur. The getPush and setPush functions will act the same as the email functions, but will return and take ZonedDateTime fields which gives the task a time for a push notification. The isComplete function will return a boolean that tells if the task has been completed. The setComplete will take a boolean and change the completion status of the task. The completion status of the task will be used to separate the tasks to be displayed on the list and calendar screen and the tasks to be displayed in history.

The next class is the TaskList class, which will contain an ArrayList of all tasks the user has made with taskList. This class has the functions sort, getTasks, getHistory, add, and remove. The sort function will take the taskList ArrayList and sort it from closest date to the furthest date. If tasks have the same date they will be sorted alphabetically. The getTasks function will return the list of non-completed tasks to be displayed on the Task List Screen, as well as the Calendar Screen. The getHistory function will return the list of completed tasks to be displayed on the history screen. The add function will add a new Task object to the taskList. The remove function takes a Task object and will remove that object from the taskList.

The Category class is a simple class that just has a String name and the getter and setter functions for this name. The CategoryList class will use this class and serves the same function as the TaskList class. This class has an ArrayList of all categories. It has 3 functions: add, sort, and remove. The add function takes a Category object and adds it to the categoryList. The sort function sorts the categoryList alphabetically. The remove function takes a Category object and removes the object from the categoryList.

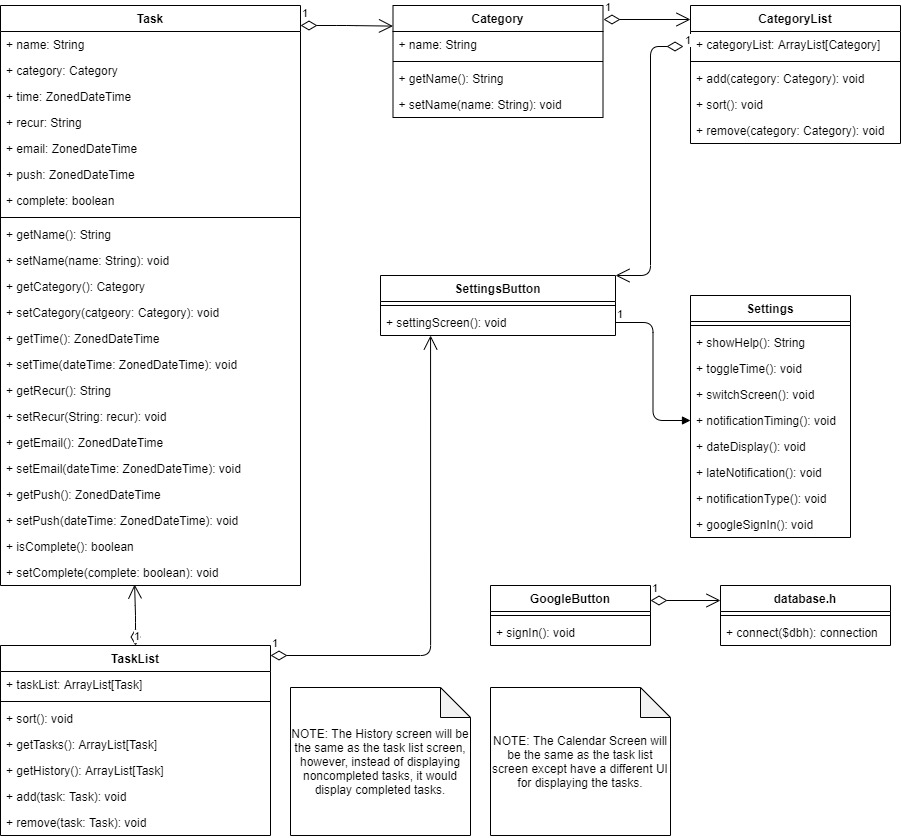
All screens will include the settings button, which on selection will take the user to the settings screen. This class has the functions: showHelp, toggleTime, switchScreen, notificationTiming, dateDisplay, lateNotification, notificationType, and finally the Google sign in button. The showHelp function will help the user navigate the application, and toggleTime will change the hour scheme to either 12 or 24 hour timing. switchScreen will change what the main screen is when the app is opened to either the task list screen or the calendar screen. The notificationTiming function lets the user choose their default time for when they want a notification sent to them, the default is set as one hour before a task is due. The dateDisplay will let the user change how they want their dates to be displayed: Month/Day/Year, Day/Month/Year, AbbreviatedMonth. Day Year, Day AbbreviatedMonth. Year, Month Day Year, or Day Month Year. The lateNotification function enables the user to turn on or off notifications for when a task was not completed on time. The notificationType function lets the user choose what type of notifications to receive by default, so they can be invisible in the new task screen. Finally the Google sign in button will allow the user to sign in through Google for synchronization to the MySQL database. The user’s settings changes will be saved in SharedPreferences.

The database will have a two table structure. It will only be used to store the user’s task information, while their Google information will be stored locally in SharedPreferences. The first table is a lookup table “Category” with a single field “category” (char(50)). The next table, “Task”, will store each of the users tasks. The only required field is the primary key “task\_name” (char(200)). This reflects our program as the task name is the only required entry for a task. Next we use “category” (char(50)) which is a foreign key connecting to the lookup table. There is a one (task) to many (category) relationship because several tasks can have the same category. The next field is “time” (dateTime) which will store the due date of the task. The program will handle time zone issues so we need only store a time in the database. Next we have the “recur” char(10)) field, which will correspond to one of 5 options the user has in the application: daily, weekdays (Monday through Friday), weekly, monthly, or yearly. Next the “email\_notification” (dateTime) field will store the time when the user has scheduled their email notification. The “push\_notification” (dateTime) field will be the same but for a push notification. Finally, the “completed” (boolean) field tells if a task has been completed. This allows us easier sorting of the tasks for the separate history and task lists. A task is not removed until it is deleted from history so this field will simplify our display and sorting.

Database synchronization will occur on 8 separate events. On startup, on opening the add task screen, on closing the add task screen, on opening the task edit screen, on closing the task edit screen, on completing a task, on restoring a task, and finally on deleting a task. This covers every time the user changes task information, or the data we will be storing. Settings changes as well as the user’s Google ID will be stored locally in SharedPreferences. Until the user signs in, this data will be stored locally in a SQLite database. Upon sign in, it will be synchronized to the MySQL server.

## **Appendix**

## **Class Diagram**



## 

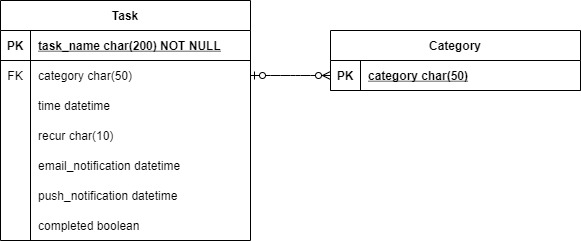
## **Message Documentation**

Messages will occur on 8 events. On startup, on opening the add task screen, on closing the add task screen, on opening the task edit screen, on closing the task edit screen, on completing a task, on restoring a task, and finally on deleting a task. Task information will involve the following fields:

* *Category* table
  + Category char(50)
    - Primary Key
* *Task* table
  + task\_name char(200) NOT NULL
    - Primary Key
  + category char(50)
    - Foreign Key
  + time datetime
  + recur char(10)
  + email\_notification datetime
  + push\_notifications datetime
  + completed boolean

## 

## **Storage Documentation**

Task data will be stored locally in a SQLite database. The tables and database can be created in Android Studio using the android.database.sqlite library. Upon sign in a single PHP script will begin synchronizing the SQLite database with the MySQL database.****