**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.

**Description**

Simple Scheduler will be an Android application that allows the user to manage their time. Users can download it from the Google Play Store. After opening the app the user will see a list of tasks that they have created. The user will have a few options from the main screen. One is to create a new category. Categories can be used to organize your tasks into separate groups. For example you may have a “School” category where you put assignments, and a “Shopping” category where you put a reminder to buy a birthday gift for your friend. Categories can of course be deleted. From the main screen you can filter the tasks by category showing only the tasks in the selected category. By default all tasks are shown.

The user will see a button used to create a new task. The task must be assigned a name, and can have an optional date and category. The user can also set a recurring date, for example every seven days, or the beginning of every month. If the user has enabled push/email notifications, then they can choose when to be reminded. For example, 30 minutes before the exact time of the task, or at 10AM on the day of the task. Tasks will be ordered first by date, then alphabetically.

Once the user finishes a task by checking it off, the task is sent to the task history list. The user swipes left from the main screen to see the Task History list. From here, finished tasks can be restored or deleted permanently. This will be done in one of two ways. Either individually selecting the task and selecting what to do with it. Or by selecting a checkbox by each task to decide what to do with multiple recent tasks at the same time.

There will also be an options button. This will switch to the options screen where the user can change features of the app. The user can sign in with Google to enable server synchronization and set up email notifications. Data will be stored locally if synchronization is not enabled. Other options include toggling push notifications and switching between light and dark modes. There will also be a “Help” button with a description of how to use the application.

The application will be written in Java using the NetBeans IDE. The interface will be developed with Android Studio. If the user chooses to use synchronization their data will be stored in a MySQL server, and we will use PHP to communicate with the database.

**Features**

* Features that will be completed by the end of the semester:
  + Simple Scheduler will be available free on the Google Play Store.
  + A task screen will be shown upon opening the app. Here the user can view their tasks sorted by date then alphabetically.
  + Tasks management will include editing, deleting, and a task history.
  + Tasks in history can be restored or permanently removed.
  + Tasks will have a name, and an optional category, and time.
  + Tasks can be set to be recurring and will automatically be set based on a time frame.
  + Users can sign in with Google to enable a server synchronization and save their data to their account.
  + Users can enable or disable push notifications as well as email notifications.
  + A settings menu where users can customize interface, change notifications, and a find help page.
* Features that will be completed if there is time:
  + A new screen to be able to view your tasks on a calendar as opposed to a list. It would be useful to see your time in a different way but as a purely cosmetic element it’s a low priority feature.
* Features that we would like to implement:
  + We think a companion desktop application/website would be a good functionality, however it is almost always more convenient to use this application on mobile.

**Technology**

* Android - The application will be developed for and used on Android phones.
* Android Studio - The tool for creating the interface.
* Java - The language used for the application.
* Netbeans - The IDE for Java development.
* PHP - Communication between the client and server.
* MySQL - Hosting the database.

**Server Information**

We will use MySQL as our database.

**Data Sources**

**Team Member Backgrounds**

|  |  |
| --- | --- |
| Jason Santana | * Java experience * No PHP or MySQL experience * No mobile development experience * Focus on back end code |
| Matthew Pinneo | * Java * No PHP or MySQL * No mobile development * Focus on User Interface |
| Matthew Spitzley | * Java experience * No PHP or MySQL experience * No experience with Android Studio * Focus on front end code |

**Dependencies, Limitations, Risks**

* Dependencies
  + Data synchronization will rely on a MySQL database.
  + Relies on a google login in order for synchronization to work.
* Limitations
  + No team members have experience in mobile development or database management. However learning new technologies is a great skill and there are plenty of resources and tutorials to help with the process.
* Risks
  + There is the possibility of data breaches which we plan to mitigate by using Google sign-in integration.

**Timeline**

The Table below depicts our schedule for the following weeks, beginning from the week of this proposal to the week of final presentations.

|  |  |
| --- | --- |
| Week 1 Jan 24 - 30: | Project Proposal |
| Week 2 Jan 31 - Feb 6: | Begin designing the application. As well as begin learning the experience we listed as missing (MySQL and Android) |
| Week 3 Feb 7 - 13: | Continue designing and learning. |
| Week 4 Feb 14 - 20: | Project update 1 - Initial Design |
| Week 5 Feb 21 - 27: | Begin making the application in our respective areas. |
| Week 6 Feb 28 - Mar 6: | Continue working on the application. |
| Week 7 Mar 7 - 13: | Continue working on the application. |
| Week 8 Mar 14 - 20: | Project update 2 - Technical issues |
| Week 9 Mar 21 - 27: | Finalize our individual work on the application. |
| Week 10 Mar 28 - Apr 3: | Ensure the separated work is working correctly when put together. |
| Week 11 Apr 4 - 10: | Repetitively test the application and debug in each area that needs it. |
| Week 12 Apr 11 - 17: | Project update 3 - Implementation and testing |
| Week 13 Apr 18 - 24: | Any final testing and debugging. |
| Week 14 Apr 25 - May 1: | Any final testing and debugging. |
| Week 15 May 2 - 8: | Final Presentation (May 5 8:00-10:00) |