Snooze Lose

Project Report

Johan Karlsson
Akademin för Innovation, Design och Teknik
Mälardalens Högskola
Västerås, Västmanland, Sweden
jkn18010@student.mdh.se

Ludwig Johansson
Akademin för Innovation, Design och Teknik
Mälardalens Högskola
Västerås, Västmanland, Sweden
ljn18012@student.mdh.se

Alan Kurdmark
Akademin för Innovation, Design och Teknik
Mälardalens Högskola
Västerås, Västmanland, Sweden
ahn16010@student.mdh.se

Berat Karacalar
Akademin för Innovation, Design och Teknik
Mälardalens Högskola
Västerås, Västmanland, Sweden
bkr19001@student.mdh.se

Engincan Varan
Akademin för Innovation, Design och Teknik
Mälardalens Högskola
Västerås, Västmanland, Sweden
evn19004@student.mdh.se

ABSTRACT

[LJ] We have developed an Android app to help the user developer their self discipline and start getting up on time in the morning. This report will cover this process from start to finish. We will introduce the idea in greater detail, then talk about our plan to get from said idea to an (almost-) finished app. It will cover how we designed the app, how we implemented its features and which problems we ran into along the way. Finally, we will talk about what our future plans of the app are.

INTRODUCTION

[JK] The idea behind this project is to create an app that will help users stop snoozing their alarms in the morning. In order to discipline the user we have the idea of punishing them upon pressing the snooze button. The punishment in this case will be to charge the user a fee that will be donated to a charity. The amount of money charged is set by the user depending on how important they deem the alarm to be. Which charity the money goes to is also chosen by the user themselves.

[JK] In order to make this app a reality we need to implement some key features. One of them being the alarm part which is where the user will create and edit alarms. Another feature we require is a page where the user can select which charity they want to donate to. Last but not least we have to provide a way for the user to make a payment to their charity of choice.

METHODS

a) Project Start:

[EV trello and GitHub repository]

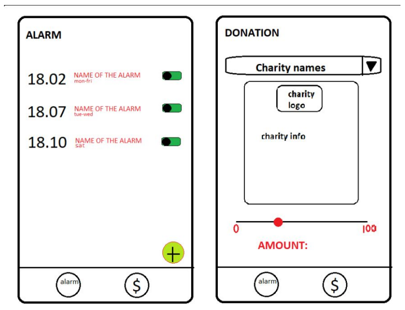
Before we even designing the project, we want to make sure that everyone knows what their responsibilities are. To achieve that, we created a trello, which is postcard website that you can organize. In the trello, we added 4 different categories which are new ideas, to do requirements, now implementing, finished. In each category, we had postcards that describes the requirements that the app needed such as alarm list page, swoosh etc… For every postcard, people assign themselves as the responsible and try to implement the requirement. In the end, we wanted to see every requirement in the finished category. Ideas part was the new features that we want to implement, such as real swish API or real charity database. We used trello to communicate between each other for the implementation of the app. Therefore, one can see what people are doing and what requirements we implemented and is going to be implemented.

Moreover, we wanted to create a GitHub repository (SnoozeLose GitHub Repository) so that everyone can work on the newest version of the app. We divided the apps into parts that one can work on, and with the help of GitHub, we were able to merge them quickly and without any mistakes. Also, since we are working with GitHub, our app had versions, which means that if something goes downside we were able to go back to the working version and start again.

In the end, we wanted to work systematically so that we are able to finish the app in time and without lack of requirements.

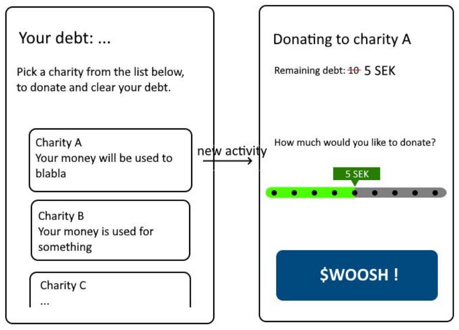
b) Design Phase:

[BK first sketches of the design, and the improvement]



[BK]Figure 1:First design of the project

Before we start to code, we first did the design and try to improve it. Above you can see our first sketch. At the beginning we thought there will be 2 pages; alarm and payment.

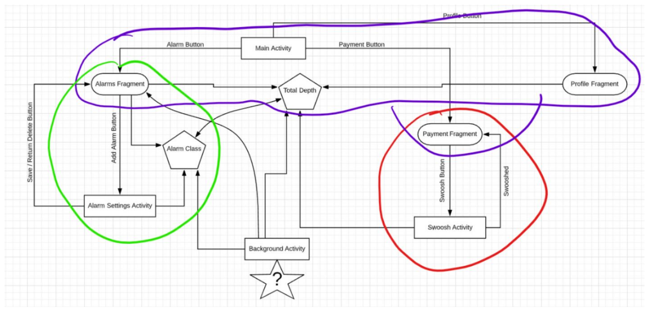

Your debt: ...
Pick a charity from the list below, to donate and clear your debt.
Charity A
Your money will be used to blabla
Charity B
Your money is used for something
Charity C
...
new activity
Donating to charity A
Remaining debt: ~~10~~ 5 SEK
How much would you like to donate?
5 SEK
$WOOSH !

[LJ] Figure 2: An improved design of the payment page (left side of the figure), and a second interface, to proceed with the payment, that's displayed after you select a charity (right side of the figure).

[LJ explaining the new payment interface, and motivating some decisions] This is a redone design of the payment page. The page where the user would donate to a charity, to clear the "debt" they have accumulated by snoozing. It displays this debt, instructions of clearing the debt, as well as a list of charities to choose from. As I didn't want to clutter this page, I split paying in to two different tasks. In the first task you select a charity that you like, and only then, is the second task displayed where you can put in how much you want to donate. Finally, the user would tap the "Swoosh" button, to start the Swoosh activity, where they could finalize the payment.

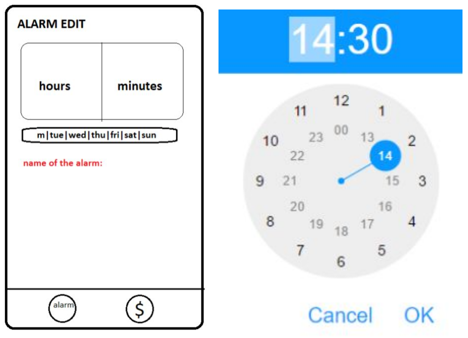
[BK first sketches of the design, and the improvement]

After that we decided to have 3 pages. 3rd page will be the profile page, where you can see the infos. We split the pages into some parts. Below you can see the flow chart and duties.



[BK] Figure 3: Flowchart, which person is doing what.

Some of us worked on the green part, some of us for the red and the rest worked on the purple part. When we were done with the sketches we tried to write the xml, design via android studio. Each of us tried to work for their pages. For example I and Johan worked for the green circle. I, personally, was thinking that design would be so easy to do it, but it wasn't. We did research about the tools, design, xml codes etc. Even we did a pretty good research, for some parts we couldn't do it so asked the others. I looked at phones alarm and see what is the necessities, or the design. So I decided to do the edit alarm page as you can see below. This was the first design of the alarm edit page. This was basically how I imagined but when it comes to design it, it was kind of hard to do it. I was thinking that when you select a date, it will automatically change the color of the selected date, but I couldn't do that so did some research and asked my friends. Also there is a time picker for android but it is totally different than mine. There was an available time picker so we used that .



[BK]Figure 4: First design of the edit alarm page, and time picker that we used in the project.

After I did "Alarm Edit" page I sent it to Engincan to add the fragment. And also he checked the things again if there was any mistake or something like that and fixed. After we did all the pages it was time to pick a good color harmony which is very important. So we picked black and orange because they are in harmony.

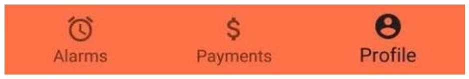
[BK]Figure 5: Here is the icon of our app. I googled "alarm app icon" and find a basic one. Just like a traditional app there is a basic alarm clock. It seems *minimalist that's why I chose that. After that I changed the colors. We used the same color in the app. Colors are conspicuous. That's the way the app seems like interesting. If you use colors in a correct way, you can sell it better.*

[EV] When we are done with the implementation of the app, we wanted to add multiple languages to our app so we created 3 different languages. First English as the default mode, then Swedish and Turkish are 2 different options available. We chose Swedish, since the audience for our app live in Sweden (swish users) and Turkish because we have 2 exchange students coming from Turkey.

c) Bottom Navigation Panel:

[EV bottom navigation panel and 3 fragments to merge with a sharedViewModel]

As you can see from the flowchart above, we had a Main Activity with 3 different fragments which nearly share the same data. That is why we want to implement the Main Activity in such a way that we can use the same class and variables between these 3 fragments. In order to achieve that, we used fragment container and and a viewModel. The container display the needed fragment and sharedViewModel organizes the data to show. SharedViewModel provided us a good way to communicate between fragments and we are able to achieve a more compact and efficient application. The viewModel consists of the statistics we want to show in our app such as total debt, current debt, how many times one snoozed, some graphs for future works.



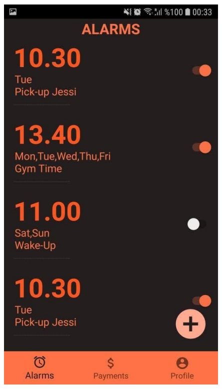
[EV]Figure 6: A screenshot of the bottom navigation panel.

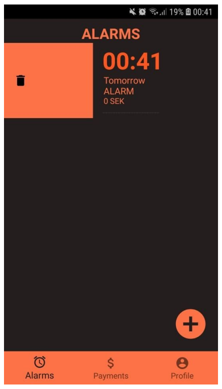
We created a bottom navigation panel with 3 options which are the fragments we used for the Main Activity. We think that, since it is traditional to have this navigation panel, many applications use these, we must use this type of UI to change the fragments. Then, we realized that people usually swipe left or right between the pages instead of actually clicking on the buttons of the navigation panel. Therefore, we added the swipe left or right functionality to our app so that we can swipe between pages.

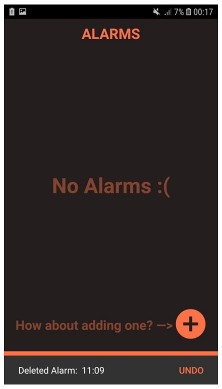
d) Alarm List Page:

[EV alarm list fragment]

Just like most the alarm apps, we want to implement our app in the traditional way to make people use our app. In the home page we have our alarm list which displays the alarms we have and a button to add more alarms which open the Add Alarm page, the same page as Edit Alarm page but the data is overwritten when saved. We used a recyclerview instead of listview because recyclerview is much more efficient and it has many advantages over listview such as detecting swipes and refreshing the list periodically. Just like a normal alarm app, we display each alarm with time, days to go on and the description. Moreover, we want to show the price of the alarm, since we can choose a cost to pay to the charities when we snooze the alarm. One can click to the alarm and edit it via Edit Alarm page, and also one can swipe left or right the alarm to delete it entirely, we knew this was a traditional habit for deleting so we added it. Also, in order to handle some human errors such as deleting an alarm accidentally, we added an undo button to recover the alarm when you delete an alarm. The aim of the page is to organize the alarm list. We added a textview, becomes visible when there is no alarms, to indicate that there is no alarms in the database and a small joke to show where to add the alarm.







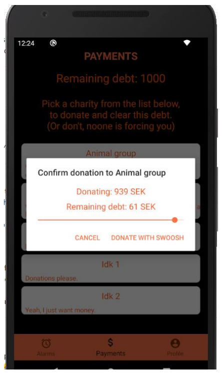
Figures 7-9: Alarm list, swiping to delete an alarm, recovering (undo) the latest deleted alarm and the view of the list when there is no alarm. Respectively.

[BK edit alarm page, time picker, days and creating a database to keep the selected alarms] In edit alarm page there were a time picker, days as buttons, amount,alarm label, vibration, sound , save and cancel buttons.

Days were created by buttons. We thought when we click the button it will change the color, which means we already clicked that button. That thing was kind of compelling. When we are done with the days, it was time to use time picker and print the selected values. We did a basic research and did the time picker part. So far we did the selected days and time picker. After that we did the alarmNames. We did the save and cancel button, but the most important part was to save the alarms. Therefore I created a database to keep the selected information. I opened an interface called AlarmDao and added insert update delete methods. With them also added deleteAllAlarms by using "delete from" statement, and getAllAlarms by using "select*from alarm_list order by time asc" statement. We used "asc" because we would like to sort it in ascending way. Also we used live data with array list (LiveData<List<Alarms>>) the reason was we could always change the alarms, hold the alarm data.

e) Payment Page:

[LJ discussing the implementation of this page] Implementing the payment page was fairly straight forward. The list of charities is a ListView. This makes it easy to add the charities programmatically, and also means there's no limit to how many charities we can add. If we add more charities than what fit the user's screen, they can simply scroll through them.
Every item in the list is a CardView. This gave them the desired appearance.


PAYMENTS

Remaining debt: 1000

Pick a charity from the list below, to donate and clear this debt. (Or don't, noone is forcing you)

Animal group

Confirm donation to Animal group

Donating: 939 SEK

Remaining debt: 61 SEK

Idk 1

Donations please.

Idk 2

Yeah, I just want money.

[LJ] Figure 10: A screenshot of the finished payment page. The user has selected the charity "Animal group"

Tapping a charity would display a dialog where they could adjust the payment amount, if they wanted to. The dialog also served as a confirmation, in case they accidentally tapped the wrong charity.
After confirmation, the Swoosh activity would start. Information, such as the charity's name and the payment amount, would be sent to the Swoosh activity as extras.

Swish vs. Swoosh:

[AK Discussing Swish and how Swoosh activity works] Swish is a swedish payment app that is secure and easy to use. It has an active user base of around seven million users to date. It was for these reasons that we wanted to incorporate Swish with our app. At this stage of development we did not know all the legal issues and all the steps that were required to use Swish's API. Swish has an open documentation and manual of how to use their API and everything seemed like it was going to be easy. Little did we know that you had to be a registered company with a company domain and you also had to sign their terms and conditions papers and have them review it for a minimum of two weeks. Even if we were to be a company with a domain, we would have time issues since a two week review of the papers would be too long of a waiting period for us. That is when we came up with an idea, Swoosh.

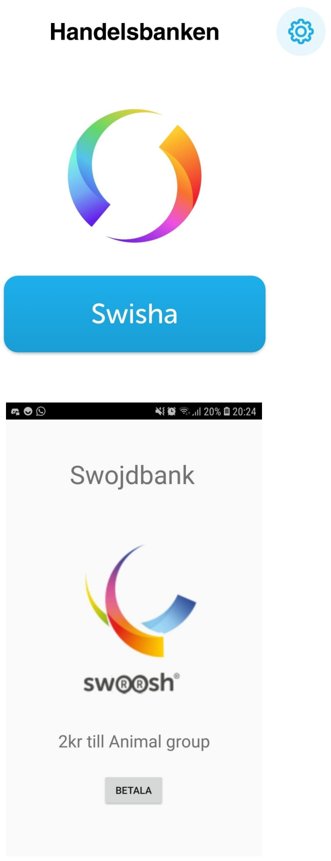
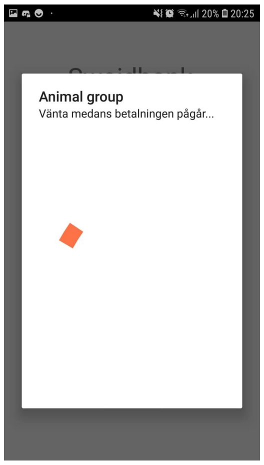


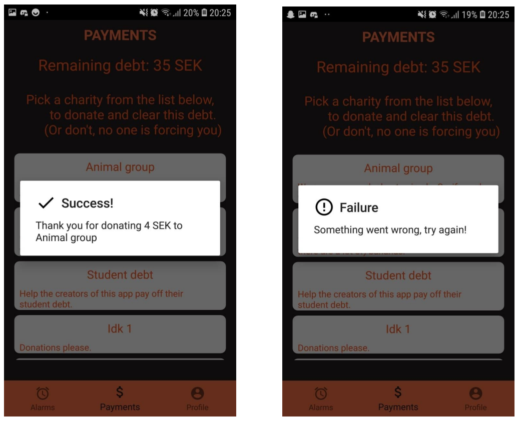
Figure 11: Screenshot to the left is the real swish app and the one to the right is our Swoosh activity. The design of the Swoosh activity is supposed to mimic the real Swish apps interface. The user has chosen to donate two kr to the charity "Animal group"

Swoosh is our substitute for Swish, it would work as a presentation version to demonstrate how the app would work if we had incorporated Swish. The way we designed the code would make it easy to replace the Swoosh activity with the code needed to make the real Swish work. Swoosh takes the extras from the previous payments fragment and shows the charity and the amount that the user chose from the payment fragment. The activity will open an AlertDialog if you press the pay button.



Animal group
Vänta medans betalningen pågår...

Figure 12: Screenshot of the AlertDialog that appears after pay has been pressed. It has a spinning circle loading bar that spins for two seconds to imitate the loading of a payment being made in the real Swish app. The payment will then be considered successful. If the user does not let the AlertDialog run for two seconds then the payment will be a failure.



Success!
Thank you for donating 4 SEK to Animal group

Failure
Something went wrong, try again!

PAYMENTS
Remaining debt: 35 SEK
Pick a charity from the list below, to donate and clear this debt. (Or don't, no one is forcing you)
Animal group
Student debt
Help the creators of this app pay off their student debt.
Idk 1
Donations please.
Alarms
Payments
Profile

Figure 12: If the payment has been considered successful it will show the Dialog to the left in the screenshot, showing the user how much he donated and to which charity they chose. If the payment was considered a failure it will show the screenshot to the right, indicating that something must have gone wrong in the Swoosh activity, either canceled by the user or some error.

As seen in Figure 12, the user gets response immediately after the Swoosh activity has done run its course. The Swoosh activity will send extras back to the payment fragment that tells the fragment if the payment was a success or not, which in turn creates the appropriate dialog.

f) Profile Page:

[EV profile page fragment]
This page is simpler than the other pages since it's only functionality is to show the personal data stored in the sharedViewModel such as total debt or debt. Since these data is stored in the phone not in any internet database, to be reliable and give customer to his/her privacy, we simply put some textviews in the fragment and change it accordingly we pull from the phone. We want to show the user some statistics to show them how much they improved. For the later versions we want to implement some monthly graphs to show how they are doing with the snoozing and helping charities. We had a thought that maybe we could implement a "light" mode for the app, which switches on/off from the profile page via a switch. Since the app is dark and orange, maybe we could add a functionality to change the theme of the app from night to light. However, we have decided that the light mode would be unnecessary since it is just an alarm app and having multiple themes won't do much.

g) Alarms:

The dummy alarm app:

[LJ how we made a prototype, to figure out alarms] Since we had never worked with scheduling of alarms before, there were a lot of things we had to figure out. At this point, we also did not know how alarms were created, deleted and edited in our app's code, since that was implemented by other members of the group. We decided to begin by creating a prototype, the "dummy alarm app", to learn how to work with Android's alarm API without first having to figure out the specific implementation details of our app.

Our goals with this prototype were simply:

- Activate an alarm (rings every other minute)
- Deactivate that alarm
- Open an activity every time the alarm rings
- Snooze button in that activity (ring again in 10 seconds)

In Android, the class AlarmManager (reference) is used to schedule alarms. It has methods, such as setRepeating(...), where you pass the time at which an alarm should ring, and the interval at which the alarm should repeat itself. This sounded perfect for us, but unfortunately, the API says it's inexact. Meaning it might not ring at the time we tell it to. It is also not very flexible, as it only allows for some predefined intervals.

Instead we ended up calling the method set(...), which lets you schedule an alarm at a specific, exact time. It will, however, only ring once. To get around this we made the alarm reschedule itself when it rang.

[JK] We faced other challenges as well when dealing with the scheduling of alarms. One of the problems we had was to make an activity open when the screen was locked. The solution to this ended up being setting different flags which gave us permission to open activities on locked screens.

[LJ closing the alarm correctly] We also had to figure out how to close the app when the user taps the "wake up" or "snooze" button, or leaves it by any other means. If the user receives a call, presses the home button on his phone, etc., the alarm activity would still live on in the recent apps list. Which is weird behaviour for an alarm. We solved this by overriding the activity's onStop() method (reference), which allows you to execute code as soon as the activity stops being visible on the screen. From there, we call the finishAndRemoveTask() method (reference).

Implementing the alarms:

[LJ] Now that we had a good understanding of how to work with alarms in Android, we began implementing it into the real app.

[JK] As we figured out, it's fairly simple to schedule an alarm with Android's API, however it is a little bit trickier when removing said alarm. In order to access the same alarm after the app has been killed we need to set a unique id for it. We did this using the requestcode parameter in the PendingIntent class. Since each alarm is assigned an auto incremented value we used it to create a system for giving each scheduled alarm a unique id. The system we came up with was to multiply the alarms own id by 7 and then add the integer value of whichever day it was set to ring. So for example if alarm 2 had monday and tuesday set then two alarms with the ids 16 (2*7 + 2) and 17 (2*7 + 3) would be scheduled. This is because monday and tuesday have the integer values of 2 and 3 respectively in javas calendar class. The reason we multiply by 7 is because each alarm can have up to 7 scheduled alarms, one for each day.

[JK] Alarms do not vibrate and make sound by themselves in Android. We had to implement those features ourselves. What we did was add checks in the activity that starts when an alarm is ringing. If the user had previously toggled vibration and sound on when creating the alarm we would start them using Android's Vibration and MediaPlayer classes. For the sound we used the devices default alarm ringtone which we could access by using Android's RingtoneManager class.

ACKNOWLEDGEMENTS

[AK] The future of this project looks positive. As we presented this project, the examiner really liked our idea and how our project was executed. He told us that this is something that has potential and that we should pursue this. Our plan right now is to become an organization so that we can incorporate the real Swish API so that real payments can be done. We need to do this since one of our main concerns have been that the users can not be certain that we are not just stealing their money instead of letting them donate it to their charities. If we work together with Swish then we will be able to let go of that concern since no money will be handled by us, Swish will handle that part. Swish has earned the goodwill of the people and when the Swish app says that the money will go to the charity of their choosing, the user will be sure that it is going to the right place.

Once Swish is integrated we plan to finally contact different charities to see if they want to work together with us. We will then switch out our dummy charities with the real ones and fetch their info and put it in the app. The last functionality to be added would be stats and graphs of how many times you have snoozed, your total debt and your monthly progression. That is a functionality that the examiner liked as well, people like seeing their progress so that they will know if they have improved or not. We would like to release this on Google Play someday.