Carsino

Group Members:
Name - e-mail - github username
Gabriel Hedström - blackdom20@gmail.com - gb22
Oliver Lidebo Kjellman - guskjeol@student.gu.se - SheolKjeol
Jonathan Nilsson - hellkor.1995@hotmail.com - Hellkor
Joakim Johansson - gusjohjodi@stdent.gu.se - jockkej
Olaide Ogunsola - gusogunsol@student.gu.se - olacuttie
Xiaolong Lin - linxl38@hotmail.com - tonylin123

Introduction

Carsino as we named our application is a phone casino application with a slightly different and interesting twist to it, different from any other Casino application that has ever been made or available in the market. Our application focuses on entertaining vehicle drivers as well as ensuring that their safety is key. As much as we want them to get entertained we also help them to stay focused on the road and our application helps to do this in a very unique way by integrating the application with the vehicle.

In a few steps we will explain give a quick explanation on how our Carsino application works.

The driver gets a few spins when the vehicle is in motion and these spins are controlled by speech. The driver starts the vehicle and the screen of the game immediately goes black, but he can still hear the sound in the background as he says spin. The slots spin and he hears a sound to signify whether he has won or lost. The phone screen also goes black when the hand brake is released.

When the vehicle is not in motion the screen stays the way it should be where you as the driver can see the display of the slots and see how they spin while you also hear the sounds. The driver can also check highscores and submit his or her name with their own score.

The natural evolution of our application would be to involve real money and add more games, but we decided to not do this during the school project, it would require us to venture into other areas than software engineering, we would need to set up bank accounts and gather knowledge about economics and probably even create a company, also we would need to take a long look at the laws regarding casinos and maybe even base it in another country.

Processes and Practises

Backlog/User stories

The backlog is a place where you keep the user-stories. We came up with our own user-stories to put in the backlog during the first sprint-meetings, trying to view our application from the perspective of a truck driver, we also made a few user stories from our perspective mostly including knowledge gathering, but far from every instance of research we had to do made it into the backlog. We hardly spent much time on this, but it still helped us knowing what to work on, so we would say it is definitely worth using.

The advantage of using user-stories is that you can more easily focus on making functionality for the application, in a more real scenario it would be easier to break-down input from the customer into user-stories to easier focus on the actual features the users want.

It is hard for us to see any disadvantages with using this, but maybe non-functionality can suffer if this approach to requirements is used, there is no focus on documentation either so it is probably not the best approach for safety-critical systems.

Backlog and user-stories are an integral part of scrum and will probably be used in many projects to come, unless all of our future projects are safety-critical or state-funded where documentation should be key.

Velocity

Velocity is a concept closely related with the backlog and user stories, we did not use any planning poker to assign the amount of points for the various stories, mainly because pivotals point system only allows for 0-3 points. We had to override the velocity all the time because the other courses got in the way and quickly reduced our velocity to zero. We can certainly see that velocity could be useful for watching your work speed and thereby estimate when things will get done. However in our case it was not very useful, we did most of the work in the last sprints, we failed to produce anything in the first sprint, and some other sprints had minimal progress, therefore the velocity was not our real velocity, but it was instead the velocity of our team working 4-20 hours a week on the project instead of the true velocity of 40+ hours we reached in the end.

We can understand that there is advantages to using a velocity method to predict releases, demo deadlines and overall planning. But it seems to have the disadvantage that if you are not focusing fully on the project it will give you a skewed perspective, and as such we could not really use the velocity to see the status of our project timescale.

However, we will use it together with the other scrum tools in future projects provided we can focus on the project better. We would probably use it even if we can not focus on a project 100% simply because it is integrated with scrum tools.

Pair Programming

We used some form of pair programming at every co-located meeting. We did not only pair at one computer though because we had to do a lot of knowledge gathering and doing this on two computers instead of one is much more effective, but having more than one person working on the same user story was very beneficial for us, because we were getting stuck a lot of the time, and having someone to talk to that were researching the same area was very good when we were stuck. In the very beginning we did not do any pair programming and it was much harder to get unstuck. We could sense the benefit as fast as we started working on the same tasks instead of having one person per task.

The advantage of this method is very apparent to us, some people might argue that it reduces the amount of tasks that could be worked on by 50%, but then again, for us this was not the case, we would rather have three stories worked on than six stories going nowhere. It also increases the quality of the code.

The disadvantages as stated above could be that the amount of tasks being worked is reduced, and when you program easier things and use classes that you know by heart, We think pair program loses a lot of its value. However it will still ensure that you do not sit and idle, as well as having two minds on the same code will probably reduce technical debt and ensure that the code quality is higher.

We will probably use this in future projects as long as we are delving into new areas, or if someone starts to lose their efficiency. If code quality is in focus then this is definitely the best approach.

We would not use pair-programming for simple applications where we already know most of the stuff.

Testing

We did not use any appropriate testing practices in our project, we did not write any test-cases in the early stages that could be applied later. We did not do any automated testing either, this is because no one of us knows how, and it feels like it would be to hard to learn during the relatively short development time we had after finishing the other courses. Getting test-case and automation skills would probably make sure that we would have nothing to test at all. The testing we did do was some kind of continuous ad-hoc testing, we all tested our parts during and after we had produced them, to see that they were working and find any apparent bugs and this worked good enough for us.

The advantages of this kind of testing is that it quickly tells you what main functions are working and what main functions are not working properly, it is also the easiest kind of testing, requiring no special knowledge and can be carried out by even the most novice of programmers.

The disadvantages of it is that it is not very helpful when finding bugs, for example this kind of testing will not find memory leaks and other errors tied to prolonged use, it will also be very hard finding bugs that result from obscure use-cases.

We will probably not use this “method” in any future projects, because finding bugs is usually important and it is hard to find any bugs without proper testing in place. Developing any kind of safety-critical application without a real testing practice or framework in place would be disasterous. It will probably be used in a lot of small assignment-type applications even in the future though.

Re-use

We did make use of some code from our last project when it comes to our database connection, it did save us some time so that we could focus on the real problems we had with connectivity, we also copied some lines of code from stackoverflow for a jellybean workaround in the voice recognition, it is attributed in the comments though.

Re-use saves a lot of time and if you have solved a problem before, it is great to just use the same method to solve it again instead of coming up with a whole new solution, There is no real disadvantage to reusing your old stuff, however copying others stuff might not always be a good idea because you could be breaking intellectual property laws if you were to sell your product. If we have something to reuse in the future, we probably will.

Group Structure

Somewhat early in the creation of the group, we did not specifically appoint a leader but Gabriel took on the leading role, which was great because he was good at organising everything and also mainly to act as the one who were to handle contact with the teachers and the supervisor.

Work Distribution

Work was usually distributed at the sprint meetings where each group member got assigned to user stories of high priority, we would start the week working on our own, or with the persons assigned to the same story, and if we during the end of the sprint still had a bit to go to fulfil the Definition of Done then the some of the other group members would pitch in. We would then give that task a bit more focus in order to try our best to finish it before the end of that sprint. We could sometimes have some troubles finishing the stories, but in the end we managed to get them all done when we worked together.
We also did quite some work from home if we already had assignments set for us, usually with great results but sometimes not the most productive.

Time Management

During the length of our project we started out a bit slowly, even when it was meant for us to put focus on the project. At first it was because we simply could not find the time, other courses were still not completely finished. So we were off to a slow start, even when the other courses were done we did not really put as much effort into it as the time allowed us to. But as time passed we put more and more effort into it, and when a few weeks had passed since the start of the project we had gotten into a quite nice pace. But in general if we could have had a better focus and motivation in the beginning it might have made the last weeks a bit less hectic as we did have to spend a bit more time than we usually had done up until that point.
We used pivotal tracker to keep track of our user stories and to keep a log of what we were supposed to do, who was supposed to do what and to keep track of when the deadline was for a particular task. These tasks are reviewed during our sprint meetings and we always have a retrospective view discussion afterwards. This really helped us to get a good overview of what had to be done and it felt rewarding as we were able to better see how we progressed.

Communication

Our communication primarily via the internet. We did handle this part quite well, whenever someone had an issue or simply was running late then that person would notify the rest of the group.

Tools

Communication tools

Facebook: Was our main communication platform. We also used it to share useful links for the project. Although the most common use of facebook for us was communicating deviations to our running schema.
Skype: We used skype many times to have teleconference for the project discussion. Skype was necessary for us to use, and the screen sharing was really handy to use for pair-programming on a group scale.

VCS

Github: Was our online service which we used to share and upload code fast. This was really useful, but the road to success here was a really bumpy one. It took quite some time for us to get a good grasp of how it worked and what to do and not to do. Also to get to know how it worked together with our IDE took some trial and error.

Documentation tools

Google drive: We used this as our platform for writing the project report, vision and etc, since it let everyone work on the same document at the same time. And everyone would have access to all the files.
Pivotal tracker: Pivotal tracker is a sprint planning tool that we used instead of a whiteboard for planning our sprints and assigning work tasks.

Database tools

PHPmyAdmin: This is a free database management tool written in PHP. We used it for storing the scores for the user.
http://www.freemysqlhosting.net: This was the website which we used to host our database on.

Programming tools

Android Studio: This was our IDE this project. It was quite useful but also confusing at times as both Android programming, this IDE and gradle were completely new to the entire group.

Sound

Audacity: This was the software that was used when recording and modifying sound for the application.

Result

In the end we got a great result. We were able to get our app working the way we had envisioned it. A more detailed explanation of the result of our application has been described in details in the introduction part of this report above.

Even though it was not everything we had written in our vision that was implemented. For instance we could not implement shutting down the application when the driver gets too distracted. We simply realized that this was not of high priority. So the decision to get the application up and running was really critical to us, with the a lot of focus on AGA. We focused more on getting AGA work because it was one of the major features of our application i.e. being able to integrate our application with the moving vehicle or in our case the simulator.

Main Page

The first thing that happens after an users opens the app is that the user gets presented with three choices; Start - High score - About

- **Start**
  Pressing start will start the game. As a start the game looks like an ordinary slot machine in which the user can tap the screen to get the wheels spinning and getting results. The user could also choose to use the voice activation for the game by saying any of the “key words”, such as spin och rulla.
  However if the phone is connected to a vehicle in motion the screen will turn black and the user will no longer be able to tap the screen to make the wheels spin instead the user will need to focus on the road, or playing the game by using the voice recognition.
- **High score**
  Even though we did not want to use real money for our app we still wanted the users in someway be able to compare their score with others. For this we created a high score page in which the top 10 results will be displayed.
- **About**
  We created a prompt in which we could add info about the game.

The Game

The game is a wheel based slot machine with different symbols on the wheels. The wheels can be spinned by tapping the screen och by voice commands. To fully get all the function of the app it should be connected to the car, in which case the screen will go black if the car is in motion, or if you would have failed to set the parking brake.
We did add sound for both when the wheels are spinning and also different sounds depending on how much you win, or if you loose. This is not only to make the app more interesting but it also fills a big function for when the screen is black, to give you information on what is going on on the screen.

Voice recognition

As one of our key requirements was to be able to control the app with your voice we naturally had to create a voice recognition that worked with our slot machine. As this was the first time for any of us to work with voice recognition we had some problems with continuous recognition, but in the end we could receive messages from the user and compare the result with a list of words that would make the wheel spin, if there was no match the app would do nothing and the voice recognition would wait for another voice input. It might have been better to go for a higher API because the voice recognition has been evolved in later api's to be better suited for continuous listening, however 5.0 lollipop is not really used on a lot of devices as of now, so we decided to take the harder path and develop for a wider audience.

Database

We realized quite early on that we did not really have much information that needed to be stored in a database, we could only come up with two attributes that we did need, both for our high score system, users name and users score. So we simply created a small database with only one table and two attributes. As we had earlier experience with databases this only took a small portion of our time.

Discussion

Main Driving Point

When we decided that we wanted to make a slot machine app that was voiced controlled we took a big risk, because we had done near to no research if it actually was a feasible thing to do. The two major things that could have ended really badly and that we had a lot of trouble with was the voice recognition and the slot machine visuals. The vision of our project could have still been reached even if we had failed the visual part, but it would have made the app "mhe", but if we had not fixed the voice recognition then the main function as a "safe to use while driving" app would have been thrown out the window.

Motivation

During our project there were a couple of days here and there that there was not a whole lot of work being done because of lack of motivation. This could probably been prevented with the use of a daily “Stand-up” meeting where one by one in the group tells the other members of what they have done the previous day and what work they plan on doing that day. This would probably have increased our motivation each day to actually complete the things that we had planned on doing each day.

Sounds

All the sounds that we used was not all taken from the internet, to be precise two of the sound was made by ourself with the human voice and some modifications with the help of some computer software. It was important to us that we legitimately got sounds that we had the permission to use in our application without any constraints. This is also why we made our own sound because there was none on the internet that suited us and thus the only way to satisfy our needs was to record and modify our own sound effects.

Future Projects...

If we were to do a similar project at some time in the future, then we probably would have implemented the daily “Stand-up” meeting, this would have insured that we actually did some progress each day of the project. Another point could be perhaps to plan ahead what was to be done or how long we should have worked each day, in order to make sure that we worked properly on those days that we worked from home as well.

Risk Management

We did not do any risk management prior to the project, it could have been wise to do so but we were simply too focused on other parts of the project. Luckily enough we did not run into any big enough accidents or mishaps that we could not easily solve so in hindsight we did not need any risk plan for this project. But you never know what could have happened beforehand.

Smalltalk

Even though we at times were certainly not as productive as we could have or should have been we had a really great time. Because when we did not code or did research we might

have instead been laughing or talking a bit. And if we look back at it then we really have nothing to regret as we did have a good time and we achieved what we set out to do.

Different Decisions We Made

We decided to make a game because we thought it would make the project more fun and intriguing for us, and that it would make us stand out from the other apps that was being made. Then of course we had to design the app so that it would be safe while driving, we did this by making it so that the user did not need to or feel encouraged to look at the screen to play the game. We did this by making the app turn the screen black while the car/truck is in motion, so that there would be no reason to look at your phone and instead the user would have other ways to operate the app without actually clicking the screen. The solution to this was naturally to create a way in which you could manipulate it with some sort of voice interface with a keyword and so we implemented the voice recognizer.

Problems and Issues

After the alpha where we had our first iteration of the Voice recognition working, we found out that it would be hard to make it listen constantly on our current API level, and then when we made it listen constantly there was more problems, including a beep every time it looped, the person working on it requested some help so we started pair programming and it went a lot smoother after that. Stuff like this could have been prevented by researching possible implementations of the features before making the decisions for the vision.

We did make some design documents during the first sprints, but this was very hard to follow due to how android works, it is not really possible to call methods of objects in a normal fashion, you need to switch between activities using intents and such. Therefor we could not really follow our initial design. The result of this was a highly coupled main class. Everything is connected very directly to this game class, in hindsight it feels like this is the only way this could have been done, because we did not have enough foreknowledge of how android works and thus could not have been prevented in any other way than to learn android before making the designs, or wasting precious time before the releases to change the design and refactor the code.

Database

We had several technical issues when we tried connecting to the database with our program. For example when we were running the highscore list, the app would crash. After the testing we finally figured out the problem, we could not implement net frame on the main thread using Android. Android have limited it to prevent the application from crashing. We used strict mode to test our code, and sometimes it worked temporarily.
We used a MySQL driver to connect from our program to the online database service. Some of our group members had quote a lot of experience with connecting Java to MySQL from last project. We even reused code from a previous project that also had a highscore system. Because this mean that project had a lot of similar code that we could reuse. Based on our

experience, reusing code can save time and resources. During the coding, it reduces our faults and construction of the database.