Decision Tree Engine App

Project Report

2015 - 01 - 08

Group 3:

Valerio Lucantonio, Nathan Chape, Tamara Dancheva,
Anna Enbom, Eric Johansson, Niklas Sjöqvist

Table of contents

1. Background information

Decision Tree App Engine is the project assigned to our group, group 3, for the course *Software Engineering 2: Project Teamwork* (DVA313) at Mälardalen University. The purpose of this project is to create an application which allows a user to answer different surveys and to calculate the economic viability of a potential investment. After each survey the user will receive a summary with statistics on questions that "approve" an investment, how many that "disapprove" and how many that still are "uncertain". There will also be an administrator application which will be able to add, remove and modify surveys that will be available for the end user application.

We can decompose the application into 3 different units (Figure 1): end user application, administrator application and an online database that is common for the 2 applications. The administrator application is a web based application accessible by the admin from a web browser. The end user application is a mobile application implemented using javascript and deployed for IOS and Android.

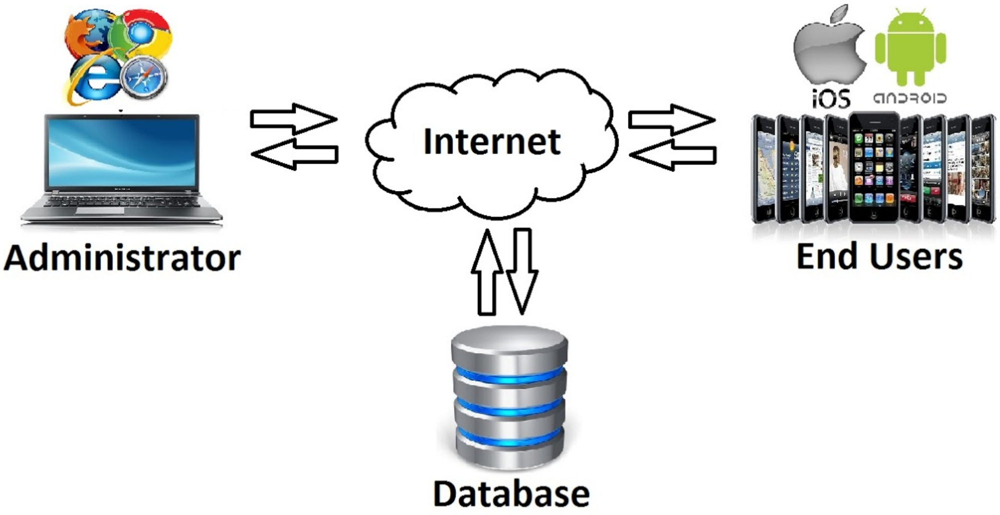


Figure 1: System Architecture

2. Project results

2.1 Produced results

Summarizing the results we can see that we covered more than 95% of functionalities we were supposed to implement. In particular at the beginning of the project we were supposed to produce only the engine that the administrator would use to create surveys. Since we were six people (so a good amount) we decided from the beginning to implement also the final user front end. As results we completed both the administrator application that allows to create these surveys and the end user application that allows final users to answer these surveys and get an evaluation if an industrial investment is profitable or not. Some side functionalities are missing (as you will see in next sections) since they were scheduled with lowest priority and we lost a member and we made optimistic estimation of the amount of work, but the core of the project is totally covered.

2.2 Deliverables and dates

Following here a table with all the produced document and softwares.

Deliverable	Date
Project Plan	20/10/2014
Design Description (first version)	04/12/2014
Design Description (last version)	08/01/2015
Product Prototype	04/12/2014
Final Product	08/01/2015
Test Specification (first version)	18/12/2014
Test Specification (last version)	08/01/2015
Project Report	08/01/2015

2.3 Acceptance test results

The administrator site acceptance test passed and it’s result can be seen in the “Test Specification” document under the title “Appendix 5.2”. The client found no difficulties with the implemented functionality but the button to add new answers to questions was in an undesirable position, this was noted and appropriate actions will be taken.

The end user application acceptance test couldn’t be performed due to an unknown database error. When the user tried to choose a language no buttons responded. Therefore the end user acceptance test could not be completed and is thereby failed. A solution for this problem is currently being worked on and when it’s completed the acceptance test will be attempted again.

The results of all tests performed during this project can be found in the “Test Specification” document.

2.4 Missing features, improvements and extensions

One of the requirements was that the administrator should be able to create nodes(questions) that accept user input and in the end a formula entered by the administrator is used to calculate if the expression results in a positive or negative outcome depending whether the result is above a known threshold. This functionalities was problematic since the logic for this tree greatly differs from the logic used for the regular nodes.

First of all, since there are additional values that have to be stored in the database (the formula, the threshold value, the name of the variables associated with the input nodes) it was necessary to create new tables in the database. Consequently, we needed to create different interface to update these tables in the administrator app and another way of interpreting and rendering this information in the end user application. This so called “economic tree” is one of a kind and once the user selects it, one chooses from a number of different calculations. Additionally there is just one economic tree in which the administrator is able to add new calculations.

Secondly, for the evaluation of the formula we need to use a library that parses and evaluates expressions which adds to the complexity of implementing this functionality and the amount of work needed to complete it.

Concerning this requirements, we have set up the database and we have created the models and the collections needed. So we have established the link between the two applications needed to make this functionality work. However the interface has to be yet implemented on both sides. According to our weekly backlog we should have finished this activity but the workload in the last three weeks proved to be heavier than expected so we did not have the time to finish it. Also the priority was set on implementing the main functionality first and then if there is still time left, start working on the other peripheral requirements. The implementation of this functionality requires a lot of work which is not proportional with its size and importance. That is why we gave this task a lower priority than fixing the bugs concerning our core logic.

Concerning the modularity of the application, there are some improvements that can be done. We followed the MVC architecture pattern, but since Backbone is a lightweight framework, this does not necessarily mean that the code can not be even more modular and the boundaries can not be even more defined.

One of the other improvements that can be done is to use hash maps when searching the json collection or find a more efficient way of retrieving objects from a collection (alternative way to a linear search). This will result in a more responsive application and is an especially necessary if the number of models in a collection is very big. This was not a high priority task since the customer said that the average number of questions is 15-20 per tree and there are currently 6 trees including the economic tree. Additionally concerning the addition of questions, the customer informed us that there will be a couple of changes or additions every one or two months.

The multilanguage support architecture can be improved upon by having one json packet as a file that is downloaded just once in the beginning since the number of questions is not big and this way we do not have to clutter the database and we can significantly simplify the retrieval of the information in the selected language. A big portion of the code is made up of iterating over the models and the array field type that we use in the database for every label(text) to enable this functionality.

The product can be further expanded by adding the economic tree functionality, adding a more visual representation of the tree in the administrator application (although it was not required from the customer) and a way for the user to review all answered questions from different trees (save the state permanently in a file locally on the device). Additionally add a clearer and simpler user-friendly way of reviewing this information and editing it (right now implemented with the previous/next buttons).

3. Project work

3.1 Organization and Routines

Since the project plan was written the organization of the team remains largely unchanged. The one notable exception being the loss of one of our team members, *Julieth Castellanos* in week 48. However, the impact on the actual organization of our team was minimal, instead it was felt more in the increased the workload.

During the first half of the course, tasks were divided between all the individual team members. Once implementation began, in the second half, the team was divided into three subgroups.

Task	**Members**
Admin App	Anna Enblom, Nathan Chape
End User App	Tamara Dancheva, Niklas Sjöqvist, Eric Johansson
Database/Inter-App Communication	Valerio Lucantonio*

**Due to the nature of the task Valerio worked with both teams during this period.*

By organizing ourselves in this way we were able to work in parallel allowing us to maximize our productivity more efficiently.

Our weekly routines typically involved 1 - 2 team meetings as well as the mandatory steering group meeting. During the team meetings we focused on what we, as a team, had accomplished the previous week as well as planning for the tasks that remained. Extra meetings were scheduled when needed, and once the team had been divided into subgroups these typically were only for a specific group. The following table gives an example of a typical week, once implementation had begun.

Monday	Team meeting (entire team), Steering group meeting
Tuesday	End User App Team meeting + Valerio
Wednesday	Individual work/Meeting

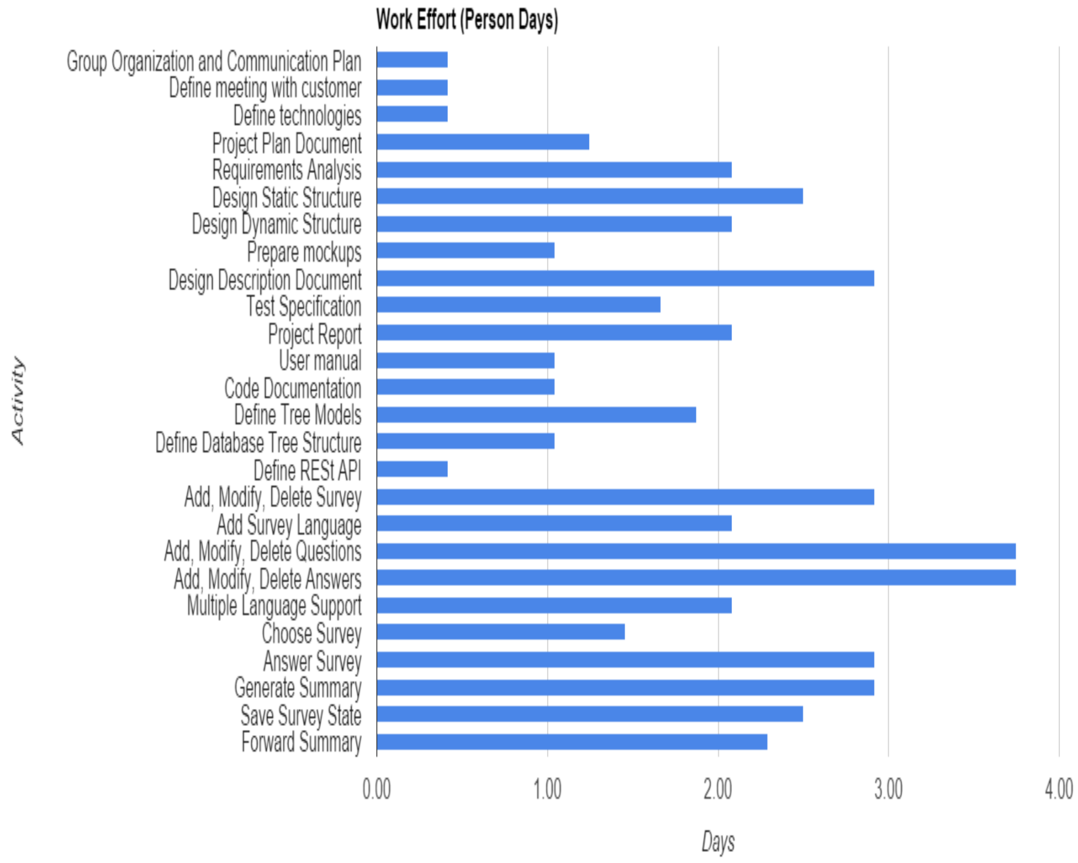
Thursday	Admin App Team meeting + Valerio
Friday	Individual work/Meeting

Prior to implementation, the week would look much like the above table minus the specific team meetings.

3.2 Total project effort

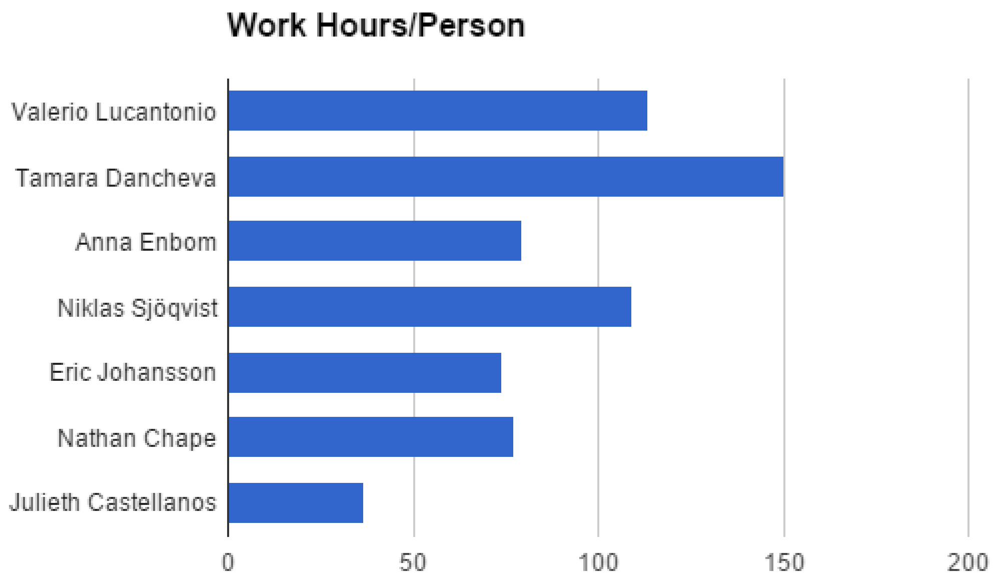
The following chart shows the different activities and the associated effort involved (in *person-days*).

Note: *this chart only shows the effort directly involved with the development and documentation of the product and does not take into account meeting and presentation times.*



Total work effort: 49 person-days

3.3 Worked hours per group member



Summary of the hours worked per team member.

3.4 Distribution of work and responsibilities

Role	Implementation Team	Member
Project Manager	Database/Inter-App Communication	Valerio Lucantonio
Client Liaison	Admin App (Head)	Anna Enblom
GitHub Supervisor	End User App (Head)	Tamara Dancheva
Document Supervisor	Admin App	Nathan Chape
Testing/Usability Supervisor	End User App	Niklas Sjöqvist
Testing/Usability Supervisor	End User App	Eric Johansson

This table gives an overview of the different roles and areas of responsibilities for each of the team members. The *implementation team* category indicates which part of the project implementation each team member was assigned.

3.5 Positive Experiences

During this project all of us have gained experience in using at least some new technique or tool, both for producing software (such as Cordova and Backbone.js) and for communication and sharing content (such as Asana and Github). During the initial discussions use we also discussed for example LibGDX and Java Spring, which was an opportunity to learn about new tools and learn from each other's previous experiences.

One way to organize this project work is to split the work into separate parts, i.e. some group members focuses on design, some on implementation and some on writing documents. We have instead chosen a distribution where all are participating in everything. We have all been taken part in meeting the customers, discussing the design, produce each text document, and implementation. This organization is more of a challenge since democracy and influence is more time-consuming, but mainly it helps us all learn more, for example by getting different perspectives and practicing to cooperate and compromise.

We all knew from beginning that software engineering projects in real life are different from ideal projects described in textbooks. This project has given a better understanding of the challenges that real life projects are facing, for example when the customers are not available when we need them and have conflicting requirements, and some new skills to deal with them.

3.5 Improvement Possibilities

The customers have been quite busy, in particular in the beginning and in the end of the project when we needed their feedback the most. In retrospect, we could have discussed their availability on our first meeting, and let them mark in a calendar when they most likely will have free time, so that we could have planned better. Optimally we could have decided on a regular customer meeting once every week, no matter if there were much to discuss or not, perhaps some weeks through Skype since the customers do not live in Västerås.

All of us are skilled and experienced in programming and software design. That means we have strong opinions on what solutions that are the best. That has led to quite harsh discussions sometimes. We also experienced misunderstandings, for example who should do what and if something we discussed was a decision or just a suggestion.
Communication issues are probably almost almost inevitable in projects like this. We managed to reduce the problem by addressing it. For future projects it is a good idea to write down important decisions and who objected them. It is also a good idea to involve the customers more since it is easier to accept a design decision you do not like if the customer likes it.