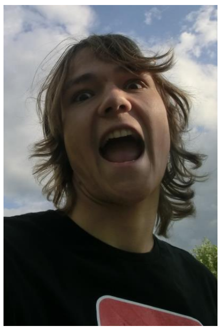
Contextproject Programming Life
Project Vision
Team 2

TU Delft


Ruben Bes
rbes
4227492

Mathijs Hoogland
mhhoogland
4237676

Jasper Denkers
jdenkers
4212584

Robbert van Staveren
rhvanstaveren
1527118

Willem Jan Glerum
wglerum
4141040

May 16, 2014

Abstract

This is the product vision for the Programming Live Contextproject, a second year course from Computer Science at TU Delft. We will develop a tool for genetic biologist to determine the diseases of a person by analysing DNA. This tool will visualize the mutations, trio data and gene interaction. The product will be intuitive to use and be able to handle existing data to compare the results to. The goal is to produce easier to use software then at hand and do this in a short fixed time frame.

Contents

1 Who is going to buy the product? Who is the target customer?

The product will be developed for free, for the use of clinical geneticists. These scientists use data from DNA to determine which diseases a person has, and how the mutations in their DNA affect other genes as well.

2 Which customer needs will the product address?

Our application is going to help our customers in visualizing DNA sequence variants and in linking SNP's with diseases. This is currently difficult because the human genome is extremely large and a lot is still not known. We are going to accept raw data, and are going to try to make it more understandable. We will do this by showing what protein is affected by a SNP, and by linking known diseases with certain SNP's. We are also going to visualize gene interactions.

3 Which product attributes are crucial to satisfy the selected needs, and therefore to the success of the product?

The product attributes that are crucial to satisfy the customer needs are an intuitive GUI for the user to use the application, an ability for the user to input data and a visual output of the results. There aren't any constraints on the so called non functional requirements; the applications could be a web service or a stand-alone desktop program. A file format for the used data is already available: VCF (Variant Call Format). The application should handle these files.

Furthermore, there should be existing information available (a reference sequence) so the application can compare input data and find annotations and report variants. Reeded input data should be mapped to the reference data to detect variants like SNPs, deletions, insertions, etcetera.

One of the main topic users would like to treat is which mutations cause which kind of diseases. The applications could help the users by using one of the five given methods: looking for known 'disease mutations', use family data of the persons whose data is used as input, filtering, looking at nearby SNPs or lastly, looking at nearby genes. The power of the application could be in expressing the results of these methods in a nice visual way.

In some of these methods it's needed to perform some calculations. The application should be able to execute these calculations fast. It would be possible to use a server-client model for the application where at client side, there's an interface for the user and where computations take place at server side.

4 How does the product compare against existing products, both from competitors and the same company? What are the products unique selling points?

There are no products to compare with from the same company, though several other product for DNA visualization have been made. Existing products like cBioPortal [1] and other software can either show relations between genes, or visualize specific genomic variants. For example Circos [2] visualizes data in a circular layout, useful for exploring relationships between objects.

The product will visualize and assist in the interpretation of genomic variants at different locations. The focus will be on trio data: genes of parents and children, and gene interactions: genes on different locations of the genome affecting the same attribute.

5 What is the target time frame and budget to develop and launch the product?

The time frame for this project is seven weeks, with a fixed deadline on the 26th of June. As this is a school project we should finish the product before the deadline as we cannot continue the project after this date. Furthermore there is not a real budget for this project, again because this is a school project. If we will succeed, each team member will receive 10 ECTS for this course.

6 MoSCoW method

This section describes the desired features of the application according to the MoSCoW method [3]. Categorized into four groups:

- **Must haves** describe requirements that must be satisfied in the final solution
- **Should haves** describe high-priority requirements that should be included if possible
- **Could haves** describe requirements that are considered desirable but not necessary.
- **Would haves** describe requirements that stakeholders have agreed will not be implemented in the solution, but could be added in the future.

6.1 Must haves

- Visualize trio data of father, mother and child
- Looking for known disease mutations
- Retrieving data from existing genetic databases

- Reading VCF (Variant Call Format) files
- Easy to use GUI for doctors.

6.2 Should haves

- Uploading VCF files to the server in the background

6.3 Could haves

- Exporting visualization data

6.4 Would haves

- Spread computational power over multiple threads, cores, or systems.
- Support for mobile web browsers.

7 User interace

As mentioned earlier, it is important that we will have an intuitive GUI. With this in mind, we tried to design our user interface as simple as possible. Please see figure 1 for a prototype of our Select Patient View. In this view, you can Add a new patient, or open a patient that the doctor previously added to the application.
In figure 2 you see the view that you get when you click on a patient or when the VCF file of a new patient has been analysed. At this stage, you can select a variant that you want to visualise.

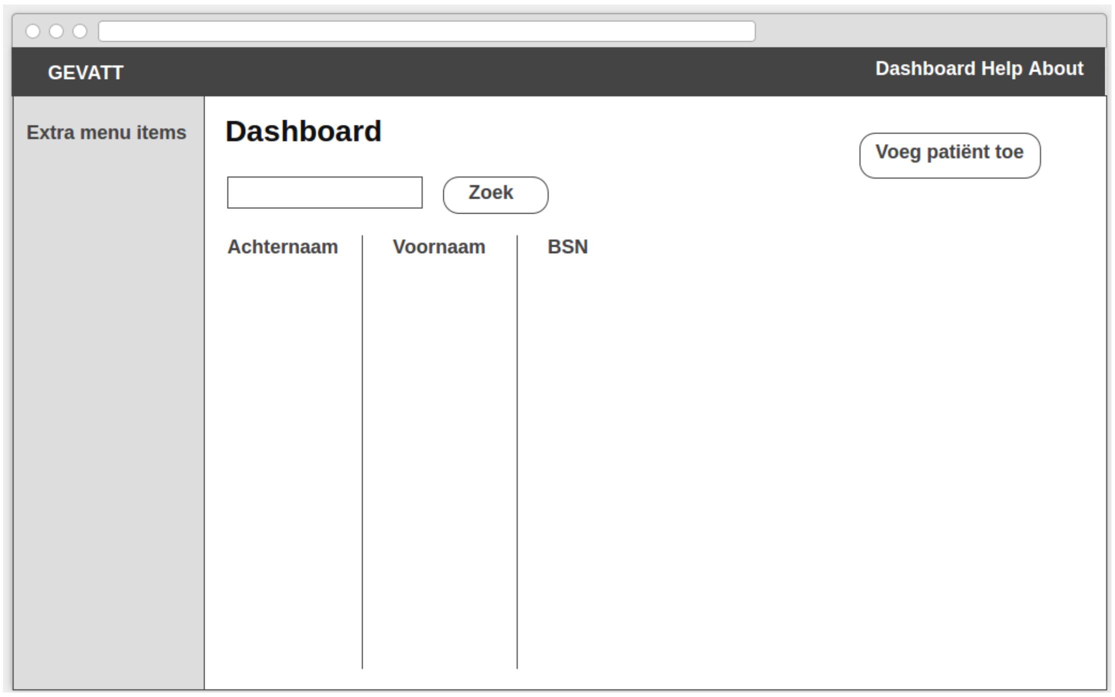


Figure 1: A prototype of the Select Patient View

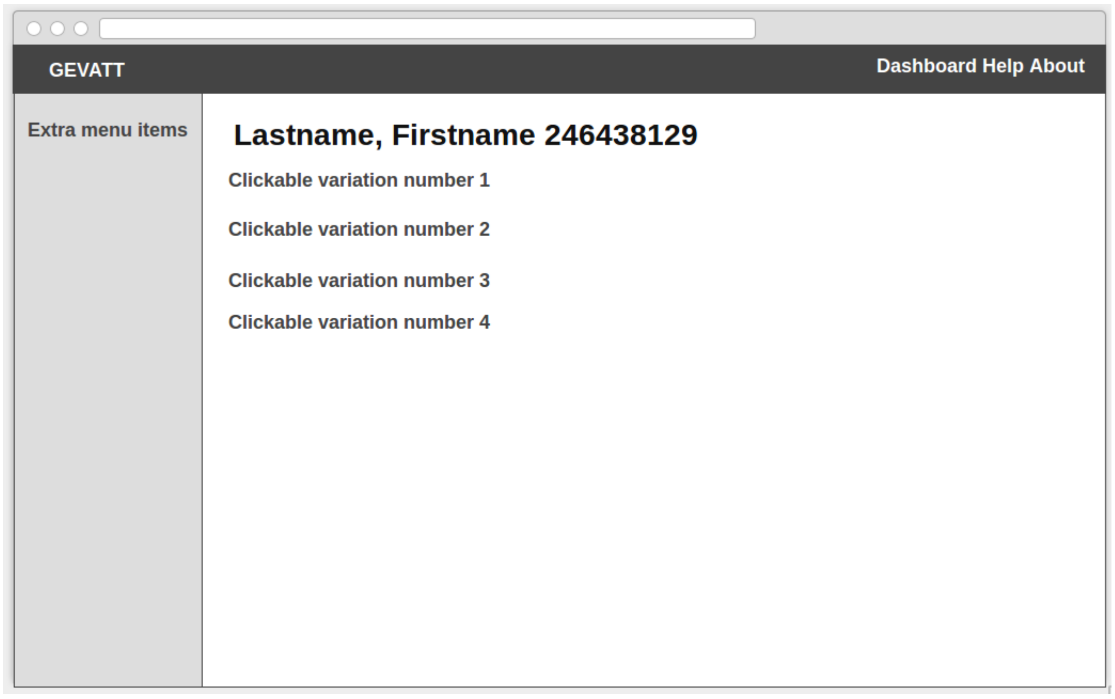

GEVATT
Dashboard Help About
Extra menu items
Lastname, Firstname 246438129
Clickable variation number 1
Clickable variation number 2
Clickable variation number 3
Clickable variation number 4

Figure 2: A prototype of the Select Variant View

In figure 3 you see the view that you get when you click on a variant in the Select Variant View. This variant shows what type of variant you selected, what protein the base pair is a building block for, and possible diseases that could be a result of this variant.

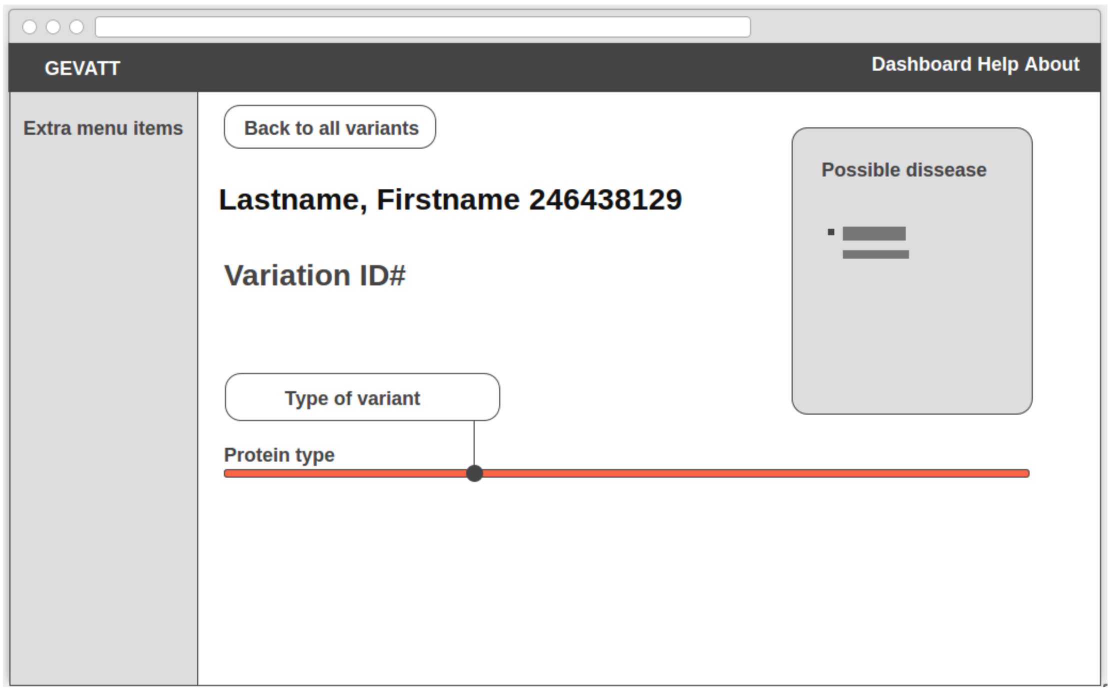


Figure 3: A prototype of the Lookup Variant View

8 Architecture

Our product will be a web application. This gives us a big advantage: we will be able to use the computational power of a server, and let our user use the program from a device that doesn't have a lot of computing power. Besides this advantage, we will also be able to let the user run our program without installing it on his or her device.

Our web application will be built in the play framework. This is an open source web application framework that is written in Scala and Java, and which follows the model-view-controller architectural pattern. This application will run on a server, and will serve the client a website. This website will be dynamic thanks to JavaScript and Ajax.

Since our web application needs to run queries from multiple servers, we are going to implement the possibility of executing queries parallel. This will result in a better performance.

9 Platforms

As mentioned in our previous section, our program will be built as a web application. Thanks to this, we will be able to run our program from any client that is able to run a website. In our design, however, we realized that our users will never want to use our application on a device like a mobile phone. With this in mind, we optimized the design for devices with a minimum width of 768 pixels, this is large enough for a tablet but too big for a mobile phone.

10 Requirements

The requirements for our application can be categorized into the following two groups:

- **Functional requirements** are requirements that describe what a system should do.
- **Non-functional requirements** are requirements that describe how a system should do it.

Since we are using the scrum development cycle, we expect that these requirements will be updated during the project.

10.1 Functional requirements

- Add a patient to our database
- Upload a VCF file that is linked to a patient
- Analyse a VCF file, look for known disease mutations
- Save the information of a certain patient
- Remove a patient

10.2 Non-functional requirements

- Run the application from a client that has at least a resolution with 768 pixels in width.
- When a new client (and thus a new VCF file) has been added, the program should be able to analyse this data and return a decent report within 10 minutes.

References

[1] Jianjiong Gao, B. Arman Aksoy, et al. cbioportal for cancer genomics. `http://www.cbioportal.org/public-portal/index.do`, 2014.

[2] Martin I Krzywinski, Jacqueline E Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J Jones, and Marco A Marra. Circos: An information aesthetic for comparative genomics. `http://genome.cshlp.org/content/early/2009/06/15/gr.092759.109.abstract`, 2009.

[3] Kevin Brennan et al. *A Guide to the Business Analysis Body of Knowledger*. Iiba, 2009.