Project Planning

Programming Life: Group 3

Thunder Wolf Squadron

Contents

1. Introduction

The project that we will be working on is the creation of a genomic visualizer to be used by genetic researchers. The product will have, at an absolute minimum, an interactive design, a semantic zoom functionality, and the ability to parse and display a phylogenetic tree from given data. In addition, the user must be able to see individual mutations and annotations. Functionalities that could be added to the product, but are not compulsory include the implementation of query tools, meta-analysis, and other data integrations for the convenience of the user. While it is required that our goal be to have the minimal implementation, we are also looking to add all of the additional aspects listed above, as well as a unique style of visualization for the data. This unique visualization takes the form of a ribbon layout at the higher level of the semantic zoom.

2. Product

2.1 Epics

1. As the user, I must be able to interact with the user interface to navigate through the visualization and with the user interface so that I can easily understand the data that I had given to the product.
2. As the user, I must be able to zoom in and out on the visualization while the data that is displayed is dynamically changing to be only the most relevant for my current view so that I can have an easier time at navigating through and understanding the data.
3. As the user, I must be able to view a visual representation of a phylogeny so that I can give the program the data of the phylogeny and have it properly displayed on the screen.
4. As the user, I must be able to give the product my data and have it produce a visualization corresponding to the given data so that I can better understand the given data.

2.2 Release Schedule

Every Friday there will be a new release of the current working version of the product. This means that on Apr-29, May-6, May-13… up until Jun-17 we will be releasing a version of our product that is built up on the previous version.

On the same days as the releases, with the exception of May-6, we will be having a meeting with the teaching assistants as well as Thomas Abeel who is the acting representative for our customers. Meetings with the actual customers will be happening but do not have concrete dates yet.

In line with scrum methodology we will be treating each week as a sprint and therefore will be creating sprint plans and reflections for those sprints. These will be created and made

available on the same day as the release of the working version and after the meeting with the teaching assistants / customers.

2.3 Release Goals

The release goals are a basic idea of what we want to have completed and inside the release each week. In the event that an implementation does not get completed or fully integrated during the week it was scheduled for, it will be bumped to the following week and given the highest priority task for that week (barring any encroaching deadlines). Additionally, each week we will be trying to keep the code base clean and organized, though this is more of a persistent background task that everyone is responsible for, so it has been left out of the table.

4.1 : Apr-22	Basic zooming, Basic parsing with database visualization, Upper and lower level visualization.
4.2 : Apr-29	Semantic zooming, Basic user interface, Improved visualization
4.3 : May-6	Improved semantic zooming, Improved user interface, Clearer visualization
4.4 : May-13	Data input through user interface, Persistent database history
4.5 : May-20	Basic data analysis tools, Possible meta data analysis
4.6 : May-27	Improved data analysis tools, Basic meta data analysis
4.7 : Jun-3	Visualization customization through inputted user settings
4.8 : Jun-10	Data analysis customization through inputted user settings
4.9 : Jun-17	Final cleaning up of code base and polishing up of product

3. Product Backlog / MoSCoW

3.1 Must haves

These are the absolutely required features and implementations that were decided either by us or directly by the client.

- As the user, I want to be able to go from a ribbon view of the data to a graphical view of the data by zooming in on it so that I can better understand the data.
- As the user, I want to be able to view additional information about individual mutations so that I can better analyse the data.
- As the user, I want to be able to easily navigate through my data without much restriction so that I can use the visualization in the way that I want.
- As the user, I want to be able to view a phylogeny corresponding to the data that I have inputted so that I can better understand that data.

3.2 Should haves

These are the should haves of the project, as the name implies they are thing that we should include in the project, but will take a backseat to the must haves if it comes down to having to chose which features we are going to implement.

- As the user, I want to be able to view extra analytics on my data provided by the program so that I can better understand the data.
- As the user, I want to be able to quickly and efficiently input my data to the product so that I don't have to wait too long to see the visualization. (This essential means that the database and program are well optimized).

3.3 Could haves

These are the could haves, they represent things that we could possibly do, but, may not have time to do.

- As the user, I want to be able to view meta data analysis of my input data so that I can better analyse the data.
- As the user, I want to be able to perform queries on the data so that I can filter the data myself and have an easier time understanding the data.

3.4 Won't haves

The won't haves represent things that we know the client might like, or something that could in theory be implemented, but we are not going to implement it.

- As the user, I want the program to be coupled with an online resource so that I can use the product as a web app instead of locally.
- As the user, I want to be able to customize the way the database works so that I can choose different queries to create possibly different visualizations.

3.5 User stories of know how acquisition

The user stories of know how acquisition represent how the user is supposed to gain the knowledge of how to use our application. We made our program with the goal of being able to use it right out of the box and without any special background knowledge so these user stories will be short and simple.

- As the user, I want to be able to use the program without having any substantial background knowledge of computers so that I do not have to run into any trouble when first using the program.
- As the user, I want to be able to setup the program without anything special that I might have to do so that I can just start using it without worrying about how it works.

4. Definition of Done

This section will provide a definition of done for three different aspects of our project. For backlog items, the definition of done will be based on the implementation of the item. For sprints, it will be based on how fully the goals a particular sprint were met. Lastly, for releases, it will depend on how satisfied the customer is with the project. This means that the definition of done for releases can change after a release is made, but this means it will have an effect on the following release and not per say the current release.

4.1 Backlog Items

- All user stories for the corresponding item are fulfilled.
- Tests are run on the item and pass.
- The item is implemented while passing all unit tests.
- The code is well documented and has no conflicts with the current version in our source control.

4.2 Sprints

- All, or as many as possible, user story features are completed and integrated into the program.
- All testing functions are passing successfully.
- Relevant documentation is created or updated.
- No serious issues remain in the github issue list.

4.3 Releases

- The program is released on github.
- The program runs without errors.
- The customer is able to use the release.
- Relevant documentation is given with the release.

5. Glossary

SCRUM : an agile software development model based on multiple small teams working in an intensive and interdependent manner.

MoSCoW : a prioritization technique used in management, business analysis, project management, and software development to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement.