Prognomics Web Tools

Hakan Küçükel

2020 Laurea

Laurea University of Applied Sciences

Prognomics Web Tools

Hakan Kücükel
Business Information Technology
Bachelor's Thesis
March, 2020

Laurea University of Applied Sciences **Abstract**
Business Information Technology

Hakan Küçükel
Prognomics Web Tools
Year 2020 Number of pages 48

Prognomics website is developed for the Institute of Molecular Medicine of Finland. This website has a collection of tools that are going be used in cancer prognosis and research in the future. Prognomics Web Tools can calculate survival estimations from different data sources worldwide. These include Finnish, British, American and other European cancer databases. Project tasks include collecting data, planning, analysing, designing and developing mobile friendly website with specific graphs and charts to visualize survival estimations accurately. The website has been developed using Vue JS, Asp.Net Core and SQL Server. SQL queries are optimized for fast execution times and responsive data visualization. First working prototype was published under app.prognomics.net domain for user feed-back. This version already delivered most of the project goals with greater user experience than the earlier prototype.

Keywords: FIMM, Prognomics Web Tools, Cancer Survival Statistics, Web Application, Full-stack Development

Contents

1 Introduction

This thesis covers the development process of Prognomics Web Tools and focuses mostly on the technological aspects of handling the data and serving it in the required manner to make the user experience responsive and reliable. The academic justification is not looked into in the scope of this thesis but it will be available online for further reading.

1.1 Introduction to the Institute for Molecular Medicine Finland

FIMM is a research centre that focuses on human genomics and personalized medicine and is partnered with European Molecular Biology Laboratory (EMBL) as well as the Norwegian, Swedish and Danish centres for molecular medicine. (FIMM 2017)

The university of Helsinki's medical campus in Meilahti is also the home of FIMM. This is a fitting location as it is the leading place for academic medical research in Finland. In 2016 FIMM employed about 230 people of 29 different nationalities. (FIMM in a nutshell 2017)

FIMM was established in 2006 after the Ministry of Education set forth a motion to launch a research institute that would be a collaboration between the University of Helsinki and other biocentre universities and institutions.

At the beginning FIMM was a joint venture of the University of Helsinki, Helsinki and Uusimaa Hospital District (HUS) and the National Institute for Health and Welfare (THL). (Launch of FIMM 2014)

In 2027 FIMM joined the Helsinki Institute of Life Science (HiLIFE) as an independent operational unit. (FIMM in a nutshell 2017)

Financial support for the institute comes from various foundation, the ministry of education, the city of Helsinki and private companies and enterprises. A large portion of the funding comes from external sources; more than half of the institute's budget comes from external competitive grants and more than 10% from industrial collaboration. (What makes us special 2017)

The basic funding that comes from the University of Helsinki was 2.8 Million euros in 2016. Additional funding from the same source was 1.7 Million euros, together this accounts for about one fifth of the research centre's total funding. Other sources include Business Finland, the European Union, foundations and industrial collaborations. (FIMM in a nutshell 2017)

1.2 Background of Prognomics Web Tools

Comparing cancer characteristics for outcome prediction is highly important when personalizing treatment for each cancer patient. These survival models are usually based on generated data for survival outcome calculations but using actual survival data to calculate proper survival estimation is what adds great value to Prognomics project itself. (Linder, interview August 8th 2019)

The academic background for Prognomics web tools was laid out by Nina Linder MD, PhD in 2000. It was not until 2010 that actual development of the tools started due to the advances in technology that allowed this process to begin. (Linder, interview August 8th 2019)

It has been Dr Linder's goal in her research to promote the new decision support tools for precision medicine via Big Data and systems medicine and currently she's working on multiple projects to bring image-based diagnostics with special interest in artificial intelligence base methods for the analysis of digitized patient samples. (Linder 2017)

The earliest groundwork of what later evolved into Prognomics was the Finprog Study on a nationwide breast cancer database. This was put together in 1997 by a group of researchers from five universities in Finland because the developments in systemic adjuvant therapy in breast cancer. Creating an accurate prognostics model aimed to help assess the risk of recurrence and provide tools to decision making in treatment. (Finprog)

2 Project Planning and Analysis

Prognomics was developed to create a tool to differentiate aggressive cancer from indolent disease based on continuously updated database. This tool will be able to provide a prognostic estimate for even rare subgroups of patients. It will bring the estimations much closer to the individual level than any previously used methods.

2.1 Project Objectives

Project objective was to develop mobile friendly web site to visualize cancer survival estimations with specific graphs, charts and tables for research, education purpose and give an idea to health professionals and researchers about survival rate of specific cancer type with actual data from patients rather than generated survival estimation. Main goal of the project is to deliver accurate survival estimation models with great user experience.

2.2 Project Planning

Prognomics study was planned by Professor Dr Johan Lundin and Dr Nina Linder and a prototype also designed and developed for this purpose. During the first group meeting team discussed the current state of the project, available resources, project goals, what can be done and the timeline. Group was working on multiple projects same time so this made accurate timeline planning difficult.

Prognomics Web Tools' planning started with collecting all the available resources and setting up the development environment for older prototype so technical aspect of the current state can be understood easily. These resources were included in the older database, older codebase, new hosting provider information, design assets and so on.

2.3 Introduction to Current Problems

Original Finprog project developed using older web technologies such as Asp.net Web Forms and multipage application technics. Currently Microsoft recommends using newer technologies such as Asp.Net Razer Pages for Asp.net Web Forms replacement. (ASP.NET Web Forms)

Older dataset was limited to 2930 patients' information with fever variables. This made querying database and calculating survival estimation easier and faster. Current project dataset contains information about over 100 cancer sites with over 10 million patients' information. Considerable increase in data size created performance bottle-neck to query database and calculate survival estimation. Even though this dataset didn't contain any information related to personal identification data security was another thing to consider.

Newer prototype is also designed for larger displays such as computer monitors with older Html 4.0.1 standards which is lacks a lot of newer features such as web sockets, application cache, web storage and other graphics related APIs. It also used server-side rendered graphs and charts which resulted with higher network activity.

2.3.1 Web Applications Client/Server Architecture

In classic web application model web browser requests are responded by the server as static web resources or server side rendered html pages. This method works very fast with static content but dynamic content can be a really big issue with complex web apps. Multipage application's requests are all handled on the server side and this increases the amount of data transferred between server and client. (Figure 1) This approach also consumes more server resources and bandwidth when compared to single page applications. (Client-Server Overview 2020)

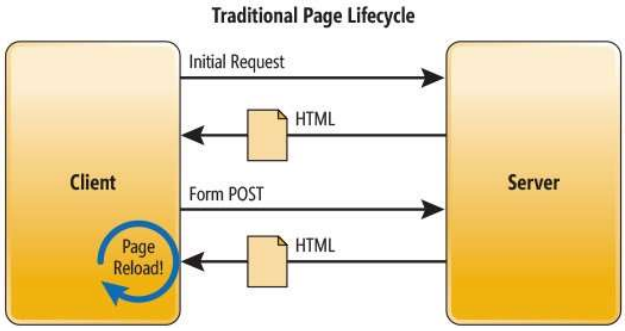


Figure 1: Traditional Page Lifecycle

2.3.2 Multipage Web Applications

Traditional web development technic which has been around since beginning of the web, Hypertext in Hypertext Transfer Protocol (HTTP) refers to linking related information by electronic connections which this technic is based on. Multipage applications have a lot of advantages such as search engine visibility and scalability which can be a really big problem with popular websites.(Hypertext) Multipage applications can be extremely fast with static content but when it comes to dynamic content rendering whole page can be slower to perform when you compare to partial rendering in single page applications. Development time and maintenance and updates can be slower related to the coupled nature of front and backend development architecture. (Sinha, R. 29 July 2019)

2.4 Possible Solutions

Using newer web technologies and technics such as Single Page Application or Progressive Web Apps were some of the possible solutions for frontend development. Rendering charts and graphs on the client-side using newer HTML5 standards would reduce amount of data transferred after each request to server. (HTML5)

In this stage replacing the current SQL Server Database with another database alternative wasn't considered but improving SQL queries for faster query time and reducing amount of data to transferred back to client were possible solutions for current problems.

Improving design with Mobile First Design approach would bring these tools to smaller display sizes such as mobile phones and tables. All the charts could be rendered on the client-side to give more responsive interactive user experience. (Mobile first 2012)

2.4.1 Single Page Applications

Single page application is web development technic where content of each page is generated dynamically on the client by JavaScript. This reduces both server load and data transfer

between client and server. (Figure 2) This is important if application is targeting mobile devices with limited bandwidth and resources. (Sinha, R. 29 July 2019) It gives much better user experience overall with faster user interaction and UI updates. This approach decouples backend and front-end development processes. AngularJS, React and Vue JS are currently the most popular JavaScript frameworks and libraries. (Lawson, K. 19 July 2018)

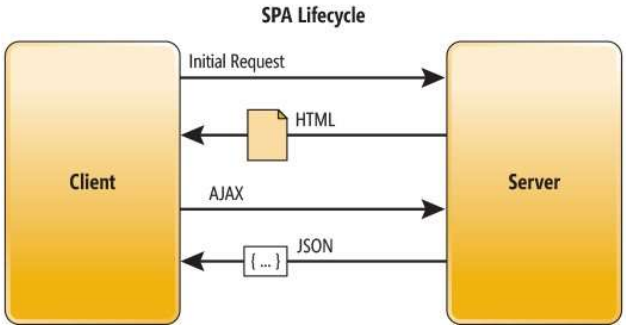


Figure 2: Spa Lifecycle

2.4.2 Mobile First Responsive Design

Smart phone market has kept growing since 2007 when first iPhone was released. Over 3 billion users in the world currently have a smartphone (Statista). Early smartphones were less powerful but also the cost of the mobile network was much higher than today's standards. Web applications needed to adapt themselves to this new trend to be able to reach more users and to be more efficient on mobile devices with limited resources and network capabilities and providing a better user experience on these small devices was another key point. By 2011 Google introduced their mobile first strategy. Mobile first required a new way of thinking in UX design, content delivery and planning on these small devices with limited screen space. Texts needed to be easy to read, other contents needed to be easily accessible and user interaction had to be simple and fluent. (Figure 3) In 2018 52,2% of web traffic worldwide was generated through mobile phones (Statista).

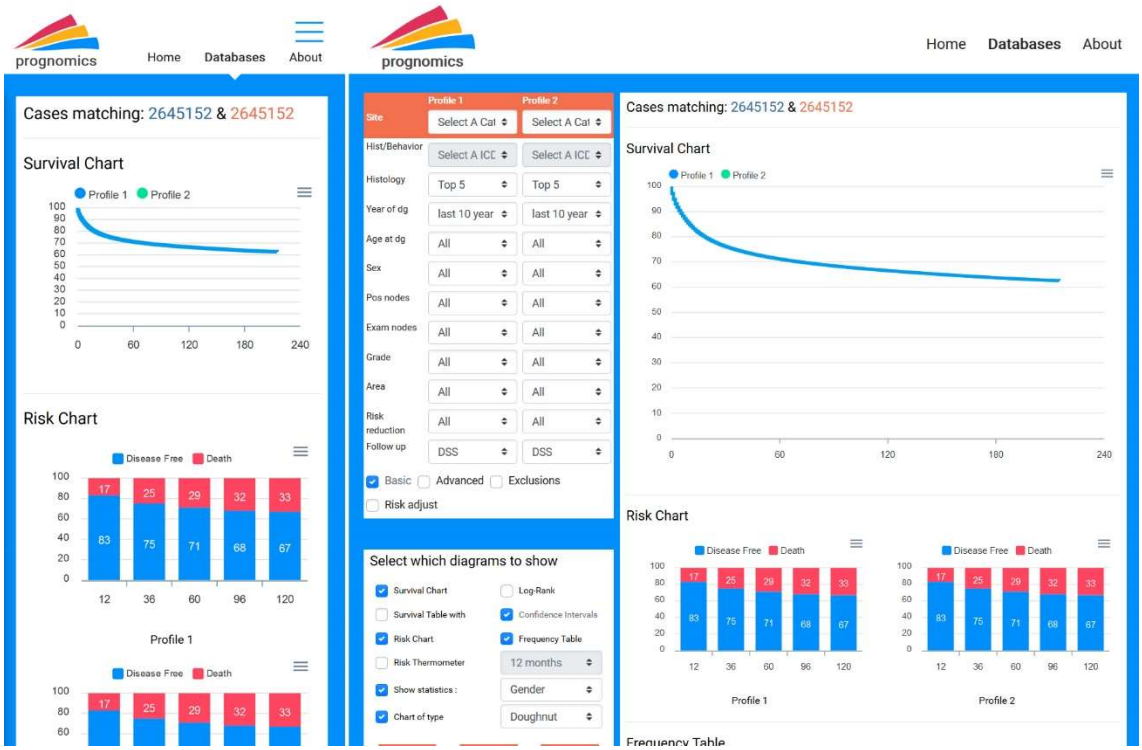


Figure 3: Prognomics Mobile First Responsive Design

2.5 Choosing Right Tools

Older prototype was deployed into new server after initial setup, also another local copy is created. This way user interaction, data transfer between server and client and query performances can be measured. Older prototype was designed with desktop first approach in mind and most of the UI elements were not fitting on small handheld devices. It was developed using older technologies such as Asp.Net Web Forms and one of the biggest design-flaw in Asp.Net Web Forms was embedding server-side view state information inside the current web page. (Asp.Net Web Forms) This view state data could reach couple of megabytes in complex web applications. View state data must be sent to server upon each request. This architectural behaviour of Asp.Net Web Forms was generating heavy traffic between server and client after each client-side update. (Figure 04)

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUENTM4MQ9kFgJmD2QWAmYPZBYCAgMPFgIeBWNsYXNzBQllbXBsb3llcnMWAgIBEBYCH
mltZ1dyYXBwZXIWBAIBDw8WAh8BBQQ0Mzk5ZGQCAw8PFgIeCEltYWdlVXJsBUAvZmlsZXMvc2tldGNoS
2QWAmYPDxYGHghDc3NDbGFzcwUUYnV0dG9uQSBNZW1iZXJMb2dvdXQfAQUTPHNwYW4+TG9nb3V0PC9zc
SBDaG9wIEhvdXNlZGQCAQ8PFgIfAQUENDM5OWRkAggPEA8WBh4NRGF0YVRleHRGaWVsZAUITGlzdE5hk
2xsZWdlIERpcGxvbWEbQ29tbWVyY2lhbCBCYXJ0ZW5kaW5nIENlcnQuE0ZpcnN0IEFpZC9DUFIgTHZsI
FN5c3RlbSBLbm93bGVkZ2UgLSBNSUNST1MfUE9TIFN5c3RlbSBLbm93bGVkZ2UgLSBTcXVpcnJlbA1Qc
mFsHFRvdXJpc20gQ2VydGlmaWVkIFN1cGVydmlzb3IRVW5pdmVyc2l0eSBEZWdyZWUFV0hNSVMMV1NFV
Tc0AzE5NQI2MwI1NwMxOTYDMTc5AzE3NQMxNzcDMjE0AzE1OAI2NAI2NQI2NgI2NwMxNTEUKwMbZ2dn2
BYSZg8PFgIfAQUEMzgyOGRkAgEPZBYEAgEPFgQeCmRhdGEtbXNnaWQFBDM4MjgeDGRhdGEtbXNncmNwc
C5qcGcnIGFsdD0nJyB3aWR0aD0nNjAnIC8+ZAIDDw8WAh8BBQ5JdmFuYSBXb3JrYWxvdGRkAgQPDxYCH
gJmDxUEATABMGRUZXN0aW5nIDEgMiAzPHA+LS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tPGJyPkNhcm5pc
w8WAh8BBQU4OTQ3M2RkAgIPZBYEAgEPFgIfA2hkAgIPFQIFMzY4MjlBPGltZyBzcmM9Jy9pbWFnZXMvZ
G91c2UtQnVzc2VyZGQCBg8PFgIfAQUbTWFyIDE4LCAyMDEzIDk6NTU6NTEgUE0gUFNUZGQCBw9kFgJmD
jo8L3A+DQo8cD48c3Ryb25nPkpvYiBUaXRsZTogPC9zdHJvbmc+IEJ1c3NlciA8L3A+DQo8cD48c3Ryb
mdldD0iX2JsYW5rIj5WaWV3IFNrZXRjaDwvYT48L3A+DQoNCjxwPjxzdHJvbmc+WWVhcnMgSG9zcGl0Y
Ww6IGthcm1pbmdAZ21haWwuY29tIDwvbGk+DQoJPC91bD4NCjwvcD4NCjwvcD4NCjwvcD4NCjwvcD4Pk8c3Ryb25nPkNvbW11b
255K0NodWQCAw8PFgIfAQUFODk0NzJkZAICD2QWBAIBDxYCHwNoZAICDxUCBTM2ODI5QTxpbWcgc3JjP
m9yZSBDaG9wIEhvdXNlLUJ1c3NlcmRkAgYPDxYCHwEFG01hciAxOCwgMjAxMyA5OjUyOjAzIFBNIFBTV
XBwbGljYXRpb246PC9wPg0KPHA+PHN0cm9uZz5Kb2IgVG10bGU6IDwvc3Ryb25nPiBCdXNzZXIgPC9wP
m9ic2Vla2VyLmFzcHg/aWQ9MzY4MjkiIHRhcmdldD0iX2JsYW5rIj5WaWV3IFNrZXRjaDwvYT48L3A+I
TogNDY0NTY3NjwvbGk+DQoJPGxpPlRvYWlsOiBrYXJtaW5nQGdtYWlsLmNvbSA8L2xpPg0KCTwvdWw+I
Q8WBB8RBQUzNjgyOR8SBQtBbnRob255K0NodWQCAw8PFgIfAQUFODk0NzFkZAICD2QWBAIBDxYCHwNo2
```


Figure 4: Asp.Net Web Forms View State

2.5.1 Frontend Development / Vue JS and Vue CLI

Newer SPA approach has been chosen to reduce amount of network activity and to increase user interactivity. One of the popular SPA JavaScript frameworks called Vue JS was selected for faster front-end development. Vue JS has several advantages against other popular framework such as small file size, easy to understand simple integration and great tooling etc. These advantages were the main reason to choose Vue JS over other popular frameworks. Adobe, Gitlab Netflix Xiaomi and Behance are some of the popular websites that are currently using Vue JS. (Vue JS 2019)

Vue Command Line Interface (Vue CLI) is standard tooling for Vue JS Application development. It's feature-rich, configurable and offers both command line and user interface to create and manage Vue JS project. Using Vue CLI simplifies configuring the project to develop, test and publish. (Figure 5)

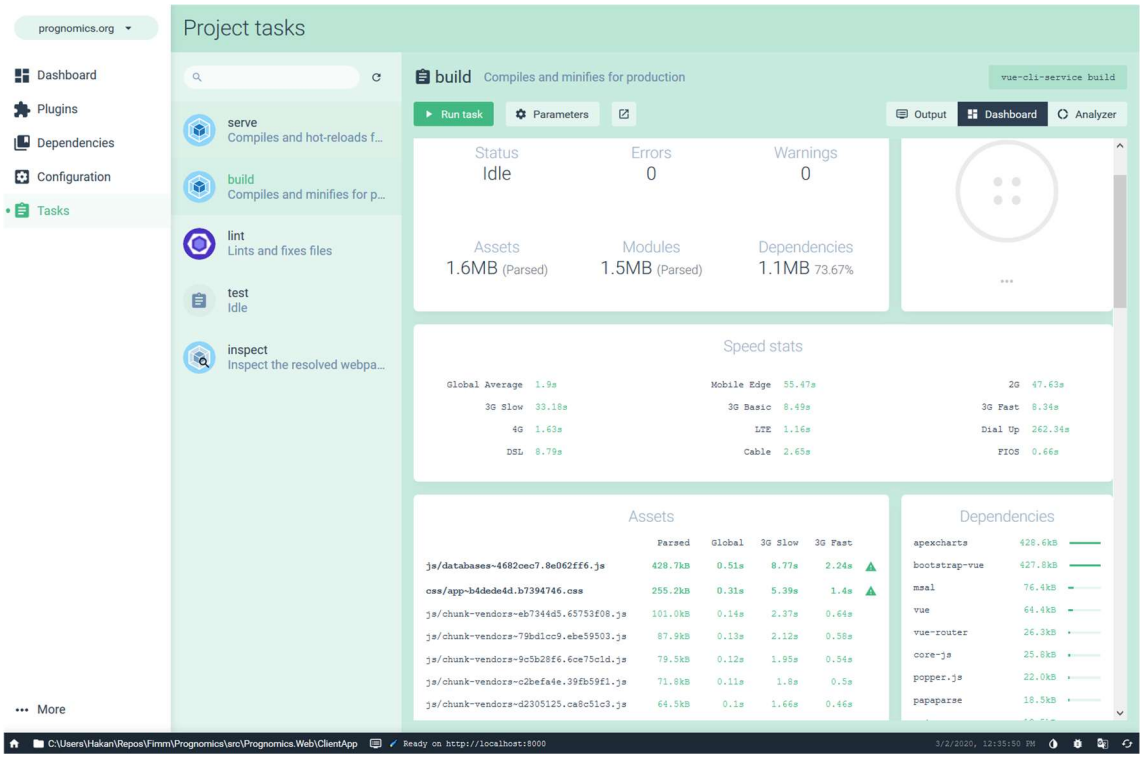


Figure 5: VUI CLI User Interface

2.5.2 Node JS and Node Package Manager

NodeJS is open-source JavaScript runtime environment built on top of Chrome's V8 JavaScript engine to execute JavaScript code outside of the web browsers. Modern JavaScript applications takes advantage of NodeJS for both front-end and back-end development, testing and deployment. (Node JS 2019) Node Package Manager is default package manager for NodeJS. NPM helps developer to reuse, share code and manage project dependencies. NPM currently is the biggest package registry available to developer community. NPM uses package.json file to manage project dependencies, building, testing definitions. (NPM 2019)

2.5.3 Backend Development and Asp.Net Core Web API

Application Programming Interface (API) is group of instructions, routines, procedures and tools for building and integrating different applications together. Asp.Net Core Web API open-source framework developed by Microsoft to build web services to consume from different clients such as mobile, desktop and browser applications. (Asp.Net Web Apps 2019)

Choosing Asp.Net Core Framework over alternatives was for better SQL Server integration and tooling. Newer Dotnet Core Framework also supports cross platform development. Prognomics is taking advantage of Asp.net Core cross platform compatibility to run on Red Hat OpenShift Container Platform.

2.5.4 Red Hat OpenShift Container Platform

Red Hat OpenShift security focused enterprise grade hybrid cloud and multi cloud orchestration platform build around Docker and Kubernetes. (Figure 6)

Red Hat OpenShift also focuses on developer productivity and comes with developer-friendly workflows including built-in continuous integration continuous delivery (CI/CD) tools that enable developers to build modern distributed applications. (OpenShift)

Prognomics hosting is provided by IT Centre for Science (CSC). CSC is non-profit state-owned company which provides services for culture and public administration, universities and national research system. CSC offers Red Hat OpenShift Container Platform as a service under Rahti Container Cloud. (CSC Blog 2019)

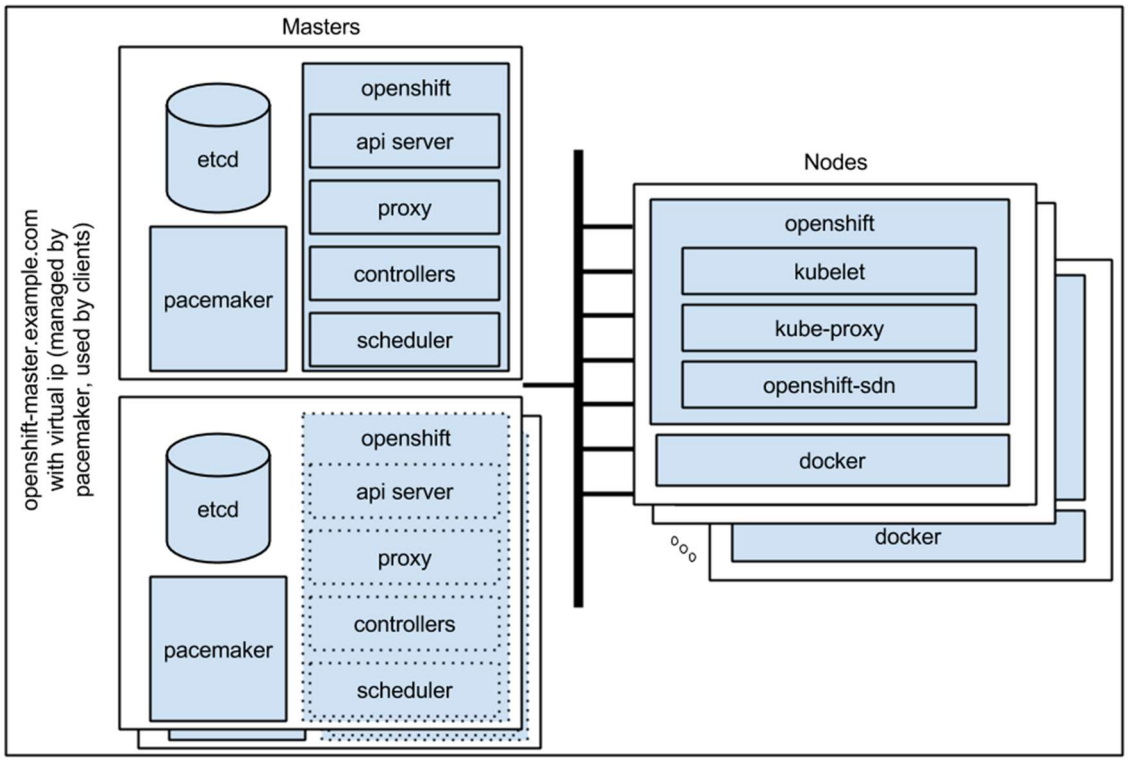


Figure 6: OpenShift Highly Available Cluster

2.5.5 Kubernetes

Kubernetes container orchestration platform was originally designed by Google to automate application deployments, scaling and managing containerized application and services. Modern cloud applications are heavily dependent on Docker Containers and managing and scaling thousands of containerized applications and services can be a big challenge. (Kubernetes Infrastructure) Kubernetes' main goal is to solve these problems modern applications are facing. All modern cloud infrastructures are built on top of the Docker and Kubernetes. (Figure 7)

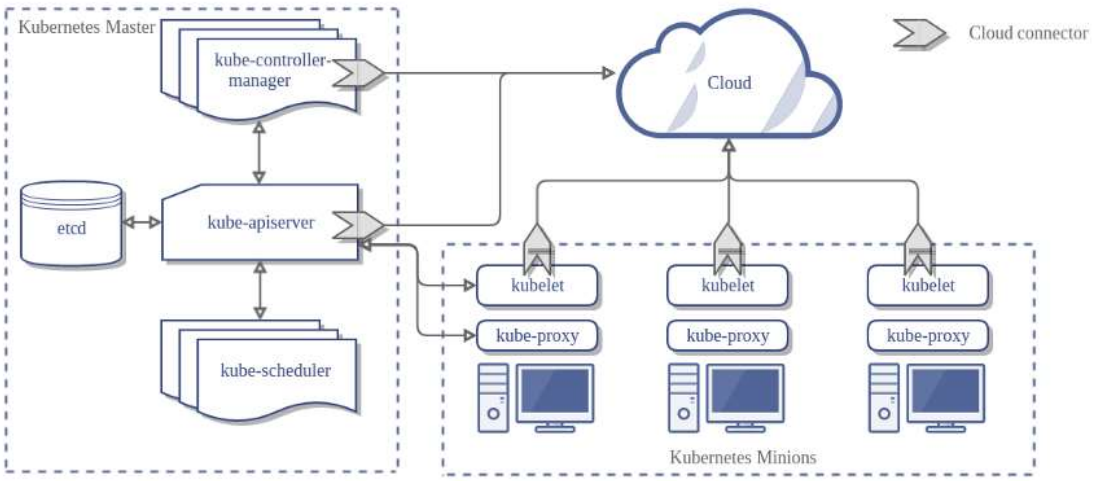


Figure 7: Kubernetes Cluster Architecture

2.5.6 Docker

Docker is an open-source project developed to solve problems that modern applications were facing. Docker containers include the application and all the dependencies and create isolated environment for each application. Docker requires fewer resources so it's faster and lighter than virtual machines and enables same hosting infrastructure to have higher density and lowers the cost per application. (What is a Container) Containerized applications are portable so re-deployment is much faster and easier than traditional application deployment methods. (Figure 8)

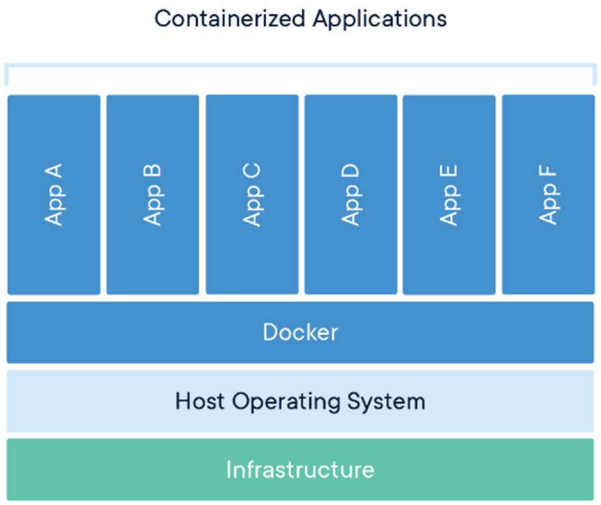


Figure 8: Docker Containerized Applications

2.5.7 Gitlab

GitLab is an open core company which develops software for the software development lifecycle and also offers free continuous integration and continuous deployment tools. (About Gitlab). Based on project requirements Gitlab has been chosen for source control and CI/CD platform for easy integration and deployment. (Figure 9). Gitlab was configured to build Docker Images after each push to master repository. When Docker build was successful new version of the application became available for deploying and testing on hosting environment.

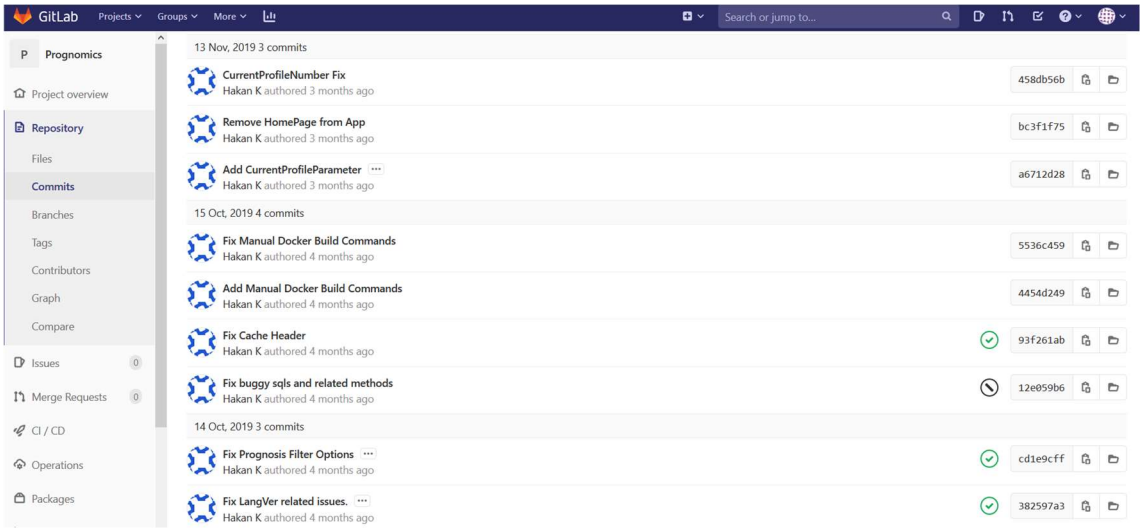


Figure 9: Gitlab Source Control

2.5.8 Visual Studio Code and Visual Studio 2019 Community Edition

Prognomics is using a lot of Microsoft products and technologies such as SQL Server for database, Asp.Net Core Web API for backend development and Azure Active Directory for user authentication and authorization. For these reasons Visual Studio 2019 Community Edition was selected as Integrated Development Environment (IDE) for Microsoft Technologies. This edition of Visual Studio is free for students and individual developers and small teams. Visual Studio Code was also used during development as alternative code editor. Visual Studio Code is a popular open-source cross platform editor.

2.5.9 SQL Server Management Studio

SQL Server Management Studio (SSMS) is a free tool developed by Microsoft for managing database servers, designing databases and queries. SSMS has a lot of additional features for analysis and reporting which makes easy to running queries and measuring query performances. (Figure 10)

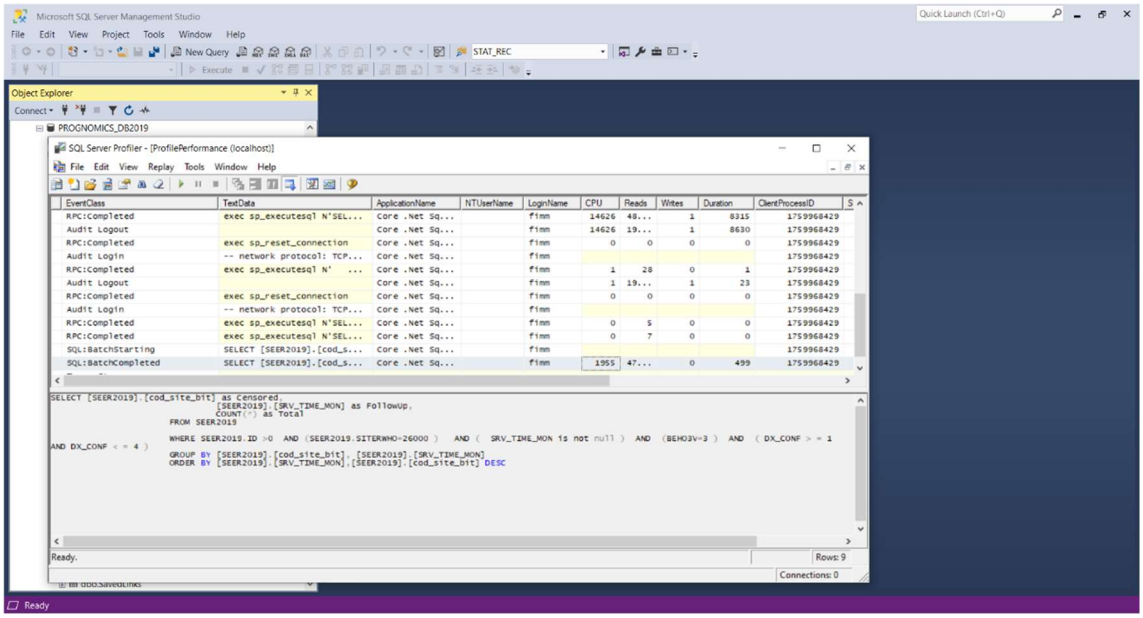


Figure 10: Sql Server Management Studio Profiler

3 Design and User Experience

Initial Prognomics website design followed traditional desktop focused guidelines and didn't offer great user experience on mobile devices. (Figure 11) Even though this prototype was using popular CSS framework called Bootstrap it didn't take advantage of responsive design approach offered by Bootstrap itself. Multipage application architecture was another reason for

slow user interface (UI) updates and poor user experience (UX). These issues were addressed and improved with updated new design applying mobile first approach and single page application architecture to overall user interaction.

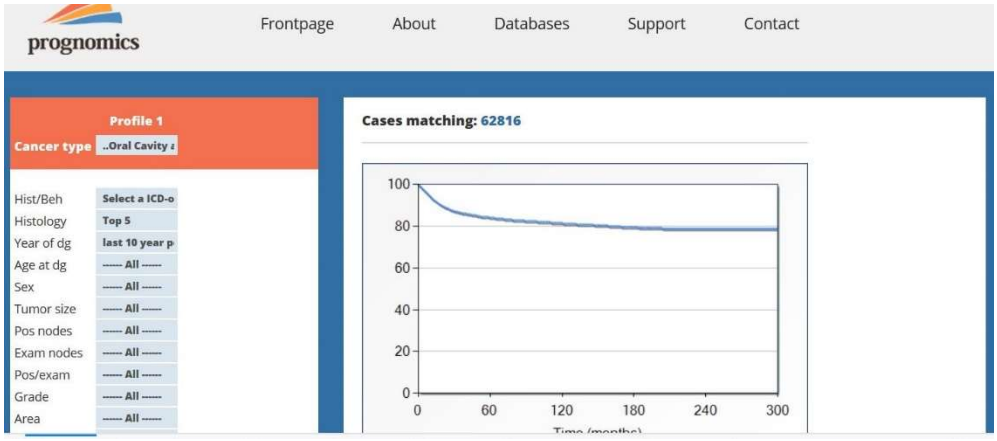


Figure 11: Prognomics Original Prototype

Updated new design followed the main design principals to create better user experience. Simplicity, functionality, faster UI updates and giving user feedback during interactions were the main goals. This design introduced side menu on the smaller devices to group filtering components under menu to use most of the display area for charts and tables. (Figure 3) Open-source Apex Charts have been used as client-side chart library. This library offered great flexibility and dynamic and faster UI updates. Logo and colour schemes are also updated to give a more modern look and feel. (Figure 12) Transition animations were added to give user feedback and bring attention to important UI updates and grab attention to changes.

Figure 12: Prognomics Current Design

4 Implementation

Prognomics Web Tools implementation was started with front-end application development with mock data and followed by back-end project development. During this process database queries were optimized for faster response times to improve user experience.

4.1.1 Data Collection

Prognomics cancer statistics database includes data from various countries. Each participating country has their data collection regulations about how collected data can be used in research.

Helsinki University holds its own ethics committee to assess non-medical research. Faculty of Medicine ethics committee conducts reviews, offers guides and assists in research related problem solving. These ethic committees follow Responsible Conduct of Research (RCR), The European Code of Conduct for Research Integrity, European Textbook on Ethics in Research guidelines. (Research Ethics 2019)

The FinProg database used as a basis for Prognomics has been collected from 2936 cases of breast cancer that have been diagnosed in the five University hospital regions in Finland between 1991 and 1992. They form about a half of all breast cancer diagnoses in Finland in this time. Clinicopathological data was collected directly from the records of the participating hospitals. (Lehtimäki 2013)

“Health care organizations have a statutory obligation to provide information on every case or strong suspicion of cancer detected in Finland to the Cancer Registry of the National Institute for Health and Welfare, which is maintained by the Finnish Cancer Society.” (Cancer Registry)

US National Cancer Institute organized The Surveillance, Epidemiology, and End Results (SEER) Program. SEER Research Data contains data between 1975-2016 for over 10 million cancer patients. (Table 2)

Table 1: SEER 18 Database

	All Cases **	**Malignant Cases**	**Malignant + In Situ Cases**
1975-2016 Data (Nov 2018 Submission)	10,450,709	9,428,053	10,146,162
1975-2015 Data (Nov 2017 Submission)	9,934,630	8,985,826	9,659,955
Increase from 2017 to 2018 Submission	516,079	442,227	486,207
Number of 2016 cases for SEER 18	497,931	430,095	473,44

SEER requires a signed Research Data Agreement form to access SEER Data which limits data usage to only statistical and analysis purposes to protect cancer patient identity. (Accessing the 1975-2016 SEER Data) (Figure 19)

Sample Agreement

It is of utmost importance to protect the identities of cancer patients. Every effort has been made to exclude identifying information on individual patients from the computer files. Certain demographic information - such as sex, race, etc. - has been included for research purposes. All research results must be presented or published in a manner that ensures that no individual can be identified. In addition, there must be no attempt either to identify individuals from any computer file or to link with a computer file containing patient identifiers.

In order for the Surveillance, Epidemiology, and End Results Program to provide access to its Research Data File to you, it is necessary that you agree to the following provisions.

1. I will not use - or permit others to use - the data in any way other than for statistical reporting and analysis for research purposes. I must notify the SEER Program if I discover that there has been any other use of the data.
2. I will not present or publish data in which an individual patient can be identified. I will not publish any information on an individual patient, including any information generated on an individual case by the case listing session of SEER*Stat. In addition, I will avoid publication of statistics for very small groups.
3. I will not attempt either to link - or permit others to link - the data with individual level records in another database. This includes attempts to link to any other SEER data.
4. I will not attempt to learn the identity of any patient whose cancer data is contained in the supplied file(s).
5. If I inadvertently discover the identity of any patient, then (a) I will make no use of this knowledge, (b) I will notify the SEER Program of the incident, and (c) I will inform no one else of the discovered identity.
6. I will not either release - or permit others to release - the data - in full or in part - to any person except with the written approval of the SEER Program. In particular, all members of a research team who have access to the data must sign this data-use agreement.
7. I will use appropriate safeguards to prevent use or disclosure of the information other than as provided for by this data-use agreement. If accessing the data from a centralized location on a time sharing computer system or LAN with SEER*Stat or another statistical package, I will not share my logon name or password with any other individuals. I will also not allow any other individuals to use my computer account after I have logged on with my logon name and password.
8. For all software provided by the SEER Program, I will not copy it, distribute it, reverse engineer it, profit from its sale or use, or incorporate it in any other software system.
9. I will cite the source of information in all publications. The appropriate citation is associated with the data file used. (Please see either Suggested Citations on the SEER*Stat Help menu or the Readme.txt associated with the ASCII text version of the SEER data.)

My signature indicates that I agree to comply with the above stated provisions.

Figure 13: Sample SEER Research Data Use Agreement

4.1.2 Front-end Development

Initial front-end project has been created using Vue CLI. Vue CLI gives options to configure the project for development, testing and publishing (Figure 5) Prognomics project takes advantage of popular developer tools such as Babel JS, ESLint, Jest Test Framework, Node Sass and Webpack. Front-end project also uses Vue Router, Vue Store, Bootstrap Vue and Apex Charts.

Reusable single file Vue components have vue file extension and these components use template, script and style tags for component visualization, application logic and styling. Vue component files are later processed with Babel and Node SASS to build proper HTML, CSS and JavaScript files that browsers can understand. (Figure 13)

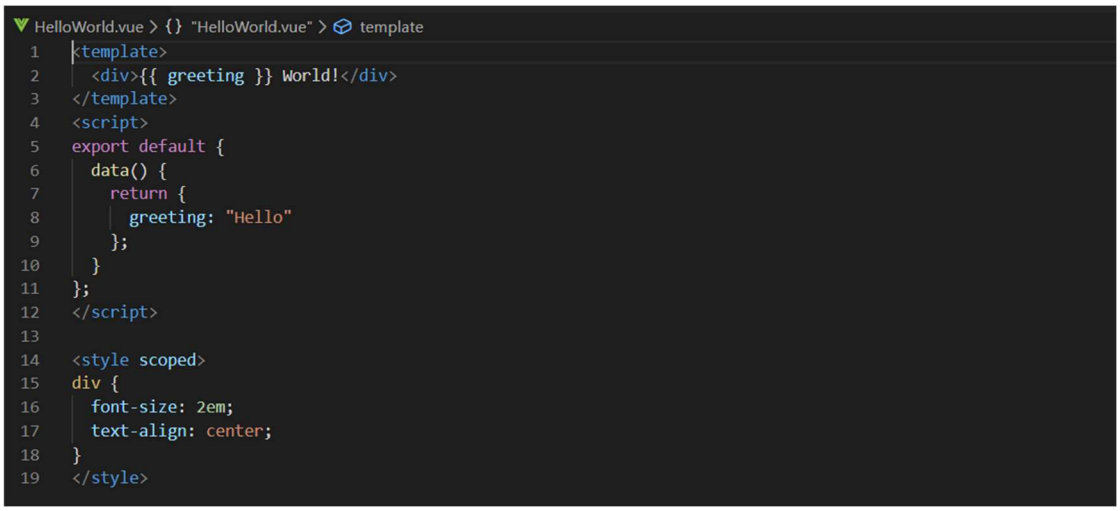


Figure 14: Vue Component Example

4.1.3 Prognomics Frontend Project Structure

Prognomics project structure followed Vue guidelines to navigate and find relative files easily during development process. (Figure 14)

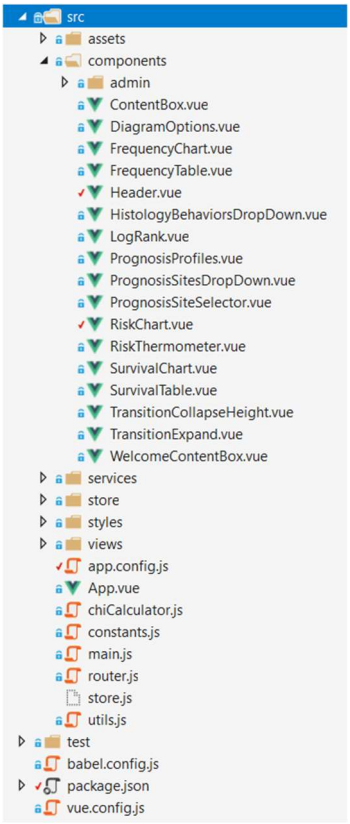


Figure 15: Prognomics Frontend Project Structure

Assets: All the static assets related to project are placed under this folder.

Components: All the reusable vue components are placed under this folder.

Services: All the application services like authentication and API services are placed under this folder.

Store: Vue Store is one of the most important features of Vue JS. Vue Store provides single source truth for all application and single state tree. Single state helps to track application state during each change and debug application when needed. Vue Store folder also contains getters to access state data, mutation to update state, actions which trigger asynchronous commit state changes.

Views/Pages: All main pages are placed under this folder

Tests: Project tests are placed under this folder

4.1.4 Prognomics Frontend Project Components

Prognomics web tools uses multiple components. Each component has single responsibility and vue guidelines followed during development process. (Figure 15)

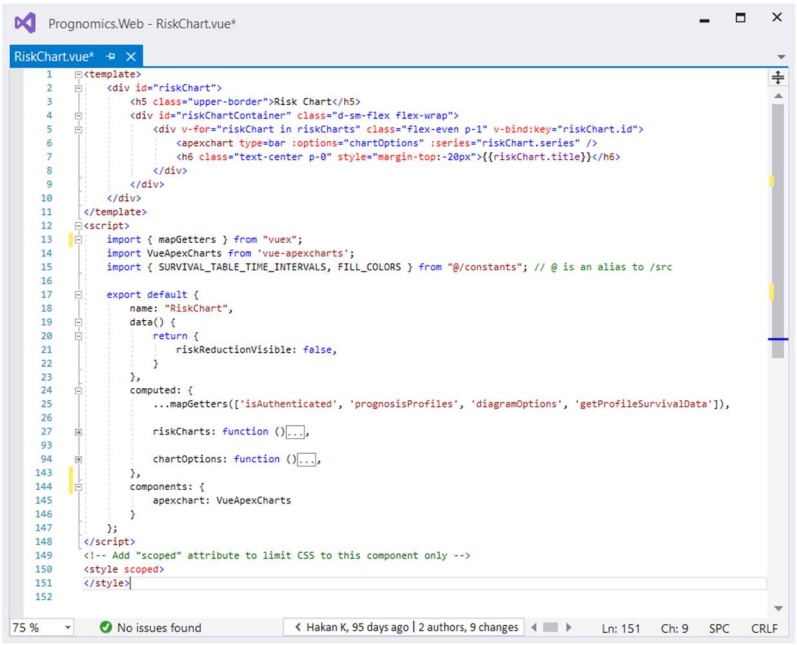


Figure 16: Prognomics Risk Chart Component

Vue components are used to visualize survival charts, risk charts, cancer sites, profiles filters and required tables like frequency and survival tables. Survival table shows statistics of given profile based on Kaplan-Meier estimator. (Table 1)

Table 2: Survival Table

Profiles	**12**	**36**	**60**	**96**	**120**
Cumulative Survival (Profile 1)	97.16%	91.69%	87.67%	83.49%	81.4%
Cumulative Survival (Profile 2)	98.67%	96.17%	94.19%	91.5%	89.76%
Number at Risk (Profile 1)	405,371	364,819	329,199	245,993	156,257
Number at Risk (Profile 2)	395,384	361,58	329,496	243,637	147,776

Survival curves are used to compare probability of survival over time between two populations. Survival curves are not just comparing mortality ratio but also used to compare occurrence of an event. (Goel 2010) Kaplan-Meier Estimator is a method used to calculate probability of each occurred event at the time. (Figure 16) It also takes censor information into account. (Park 2017) Prognomics implementation of Kaplan-Meier Estimator also calculates cumulative survival rate and risk reduction rate in same method. This method implemented with JavaScript on client-side to reduce amount of unnecessary traffic to server. (Figure 17)

These analysis results are also shown on the survival chart (Kaplan-Meier Survival Curves) component visually. (Figure 18)

$$P_i = \frac{S - D - C}{S - C}$$

$$P_n = P_i \times P_{i+1} \times P_{i+2} \ldots\ldots P_n$$

where:

P_i is the probability of surviving interval i

S is the number of people surviving up to the given interval

D is the number of people who die during the interval

C is the number of people censored during the interval

P_n is the probablity of surving through interval n

Figure 17: Formula for Kaplan-Meier estimator

```
export const KaplanMeierEstimater = (stats, riskReduction) => {
    const data = [];
    riskReduction = riskReduction && riskReduction >= 5 && riskReduction <= 80 ? 1 - riskReduction / 100 : 1;

    if (stats && stats.matchingCaseStats) {

        const numberOfPatients = stats.totalMatch;

        let numberAtRisk = 0;

        let followUpPatients = 0;

        let cumulativeSurvivalRate = 1.0;

        let riskReductionRate = 1.0;

        for (let i in stats.matchingCaseStats) {
            let match = stats.matchingCaseStats[i];

            numberAtRisk = numberOfPatients - followUpPatients;
            // EstimateKaplanMeier
            if (match.censored) {
                const deathRate = (match.total / numberAtRisk);
                cumulativeSurvivalRate *= (1 - deathRate);
                riskReductionRate *= (1 - deathRate * riskReduction);
            }

            followUpPatients += match.total;

            data.push({ followUp: match.followUp, probability: cumulativeSurvivalRate, riskReductionRate , numberAtRisk: numberAtRisk });
        }
    }
    return data;
}
```

Figure 18: Kaplan-Meier Estimator and Cumulative Survival Rate Implementation

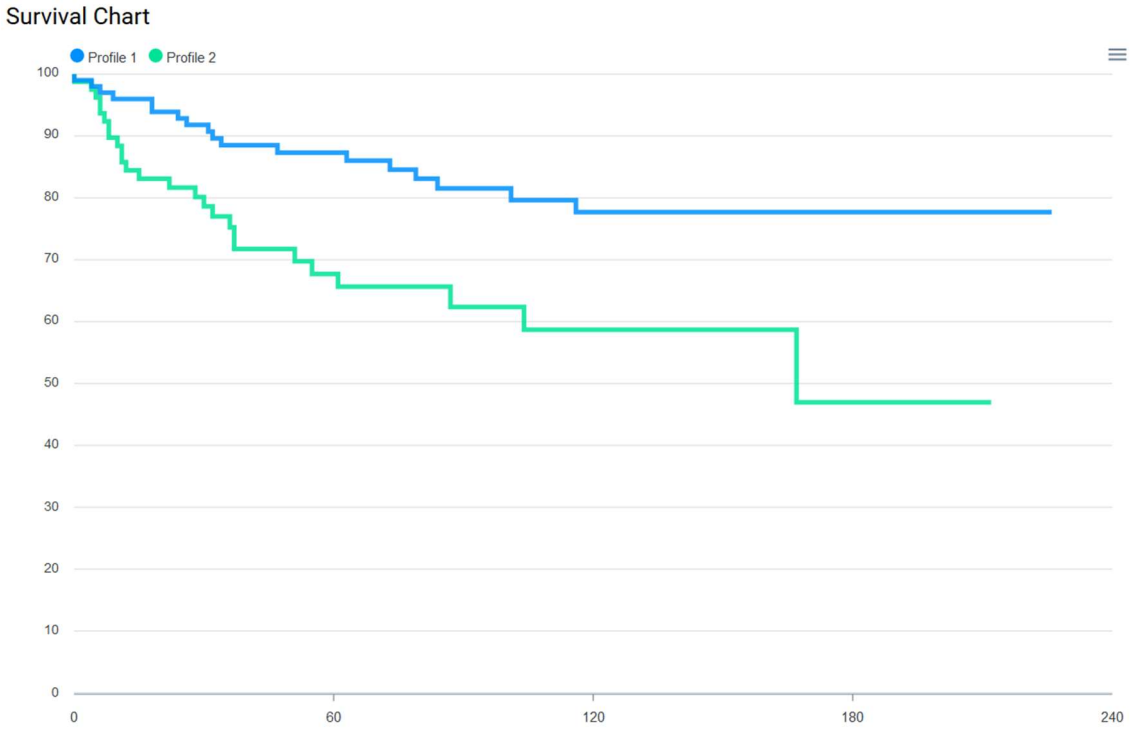


Figure 19: Survival Chart

4.1.5 Backend Development

Monolithic Asp.net Core Web API application has been created using dotnet cli. Monolithic applications are entirely self-contained applications. Biggest disadvantages are scalability and development time when application gets larger. Microservice architecture has been considered but this step required database changes and validating all the queries and results which would be time consuming. To reduce complexity in backend the application followed clean architecture guidelines. (Figure 19)

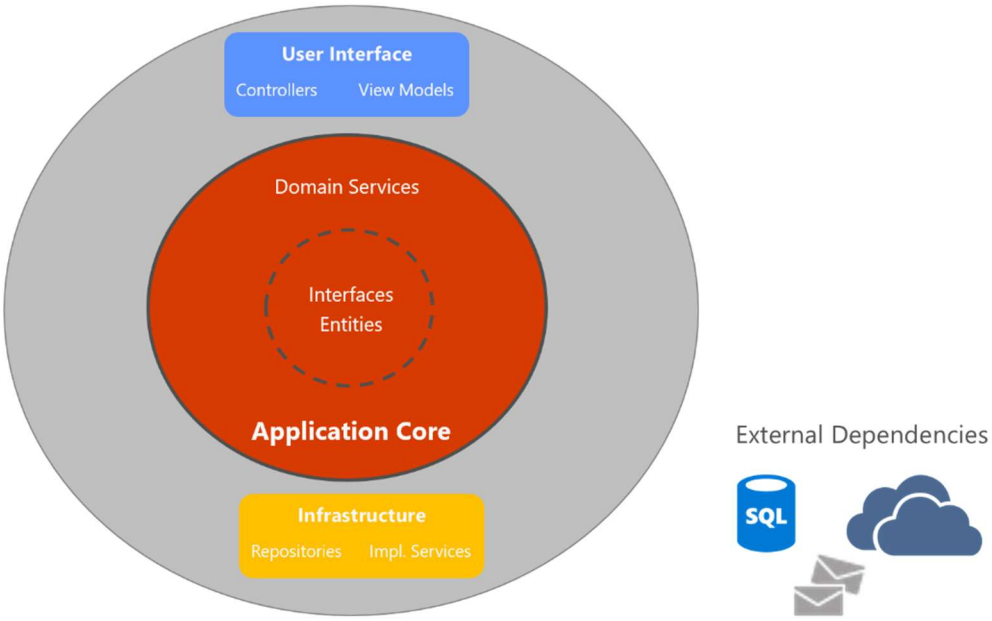


Figure 20: Clean Architecture Layers

Application code base is organized into smaller groups based on the responsibilities or concerns. This layered architecture has many advantages in organising and finding related code to improve the development process. Even though all the codebase was under single project relative layers were grouped within its own folders. (Figure 20)

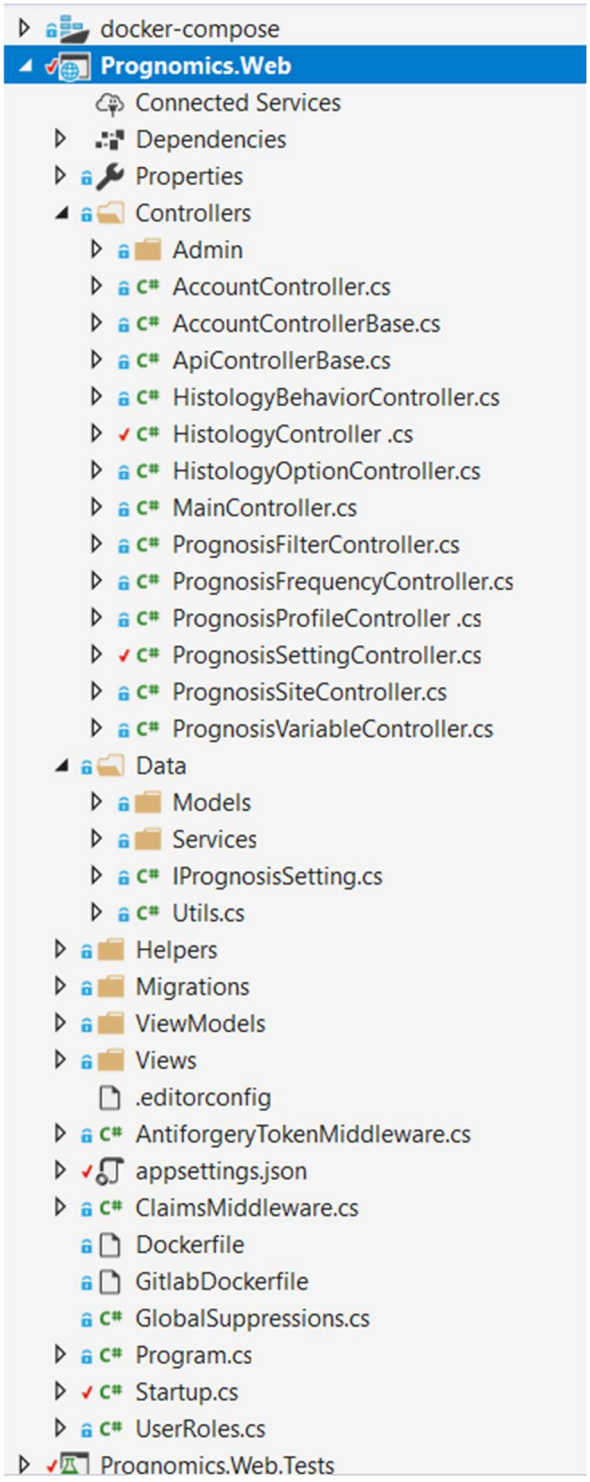


Figure 21: Prognomics Backend Project Structure

During development process entity models, data transfer objects, database context, data repositories and services have been created. Simple Object Mapper called Dapper has been selected for data access layer for faster SQL performance.

Plain Old CLR Objects (POCO) classes have been created to map existing SQL Server Database tables to code. This approach is known as Database First Approach. (Figure 21)

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Prognomics.Web.Data.Models.CM2014
{
    8 references | 0 changes | 0 authors, 0 changes
    public partial class Histology
    {
        [Column("ID")]
        0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public int Id { get; set; }
        [Column("site_recode")]
        [StringLength(255)]
        0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public string SiteRecode { get; set; }
        [Column("site_description")]
        [StringLength(255)]
        0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public string SiteDescription { get; set; }
        [Column("histology")]
        0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public short Histology1 { get; set; }
        [Column("histology_description")]
        [StringLength(255)]
        0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public string HistologyDescription { get; set; }
        [Column("hist_beh")]
        0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public int? HistBeh { get; set; }
        [Column("histology_behavior_description")]
        [StringLength(255)]
        0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public string HistologyBehaviorDescription { get; set; }
    }
```

Figure 22: Sample POCO Class

These POCO classes are used to query database and map data between business logic and to create, read, update and delete (CRUD) operations.

```
using System.Collections.Generic;
using Dapper;

namespace Prognomics.Web.Data.Services
{
    4 references | Hakan Kucukel, 159 days ago | 1 author, 2 changes
    public class HistologyService : DataService<Models.CM2014.Histology>
    {
        0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public HistologyService(System.Data.IDbConnection dbConnection) : base(dbConnection)
        {
        }

        29 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public override IEnumerable<Models.CM2014.Histology> All()
        {
            EnsureConnectionOpen();
            return DbConnection.Query<Models.CM2014.Histology>("Select * from Histologies");
        }

        32 references | Hakan Kucukel, 356 days ago | 1 author, 1 change | 0 exceptions
        public override Models.CM2014.Histology GetById(int id)
        {
            EnsureConnectionOpen();
            return DbConnection.QueryFirstOrDefault<Models.CM2014.Histology>("Select * From Histology where ID=@ID", new
              { ID = id });
        }

    }
}
```

Figure 23: Sample Data Service

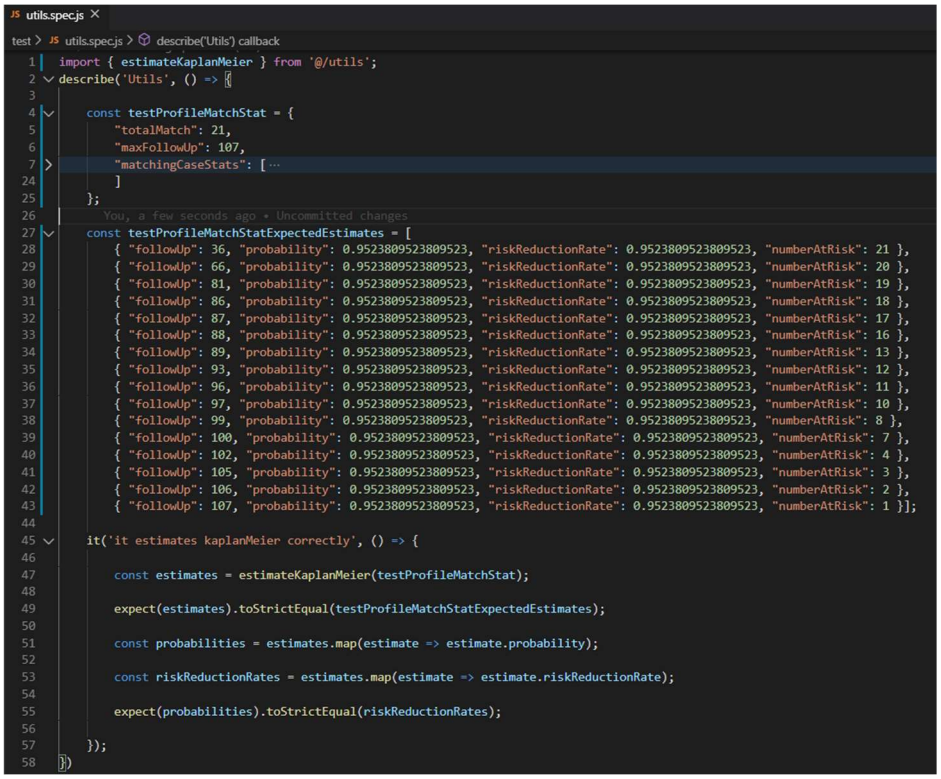
These services were injected to dependent API end points to be consumed by client application later on (Figure 22)

```
1   using System.Collections.Generic;
2   using System.Linq;
3   using Microsoft.AspNetCore.Mvc;
4   using Prognomics.Web.Data.Services;
5   namespace Prognomics.Web.Controllers
6   {
7       [Route("api/[controller]")]
8       [ApiController]
        1 reference | 0 changes | 0 authors, 0 changes | 0 requests
9       public class HistologyController : ApiControllerBase<HistologyService,Data.Models.CM2014.Histology>
10      {
            0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
11          public HistologyController(HistologyService histologyService) : base(histologyService)
12          {
13          }
14
15
16          // [Microsoft.AspNetCore.Authorization.Authorize]
17          [HttpGet]
            0 references | 0 changes | 0 authors, 0 changes | 0 requests | 0 exceptions
18          public ActionResult<IEnumerable<Data.Models.CM2014.Histology>> Get() => DataService.All().ToList();
19
20      }
21  }
22
```

Figure 24: Sample Web API End Point

5 Testing

Both frontend and backend applications have their own tests. Vue application is using open-source Jest JavaScript Testing Framework developed by Facebook. Jest is a well-documented zero config fast and safe testing framework with easy to understand API. (JestJS 2019) Vue project was configured to run Jest tests using Webpack and NPM. Current front-end tests use Jest's assertion API (Figure 25). These tests run locally before code is pushed in source control. Snapshot testing, end-to-end testing, having better test coverage and integrating with CI/CD pipeline are planned in the future.



```
utils.spec.js
test > utils.spec.js > describe('Utils') callback
import { estimateKaplanMeier } from '@/utils';
describe('Utils', () => {

    const testProfileMatchStat = {
        "totalMatch": 21,
        "maxFollowUp": 107,
        "matchingCaseStats": [···
        ]
    };
      You, a few seconds ago • Uncommitted changes
    const testProfileMatchStatExpectedEstimates = [
        { "followUp": 36, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 21 },
        { "followUp": 66, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 20 },
        { "followUp": 81, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 19 },
        { "followUp": 86, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 18 },
        { "followUp": 87, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 17 },
        { "followUp": 88, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 16 },
        { "followUp": 89, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 13 },
        { "followUp": 93, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 12 },
        { "followUp": 96, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 11 },
        { "followUp": 97, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 10 },
        { "followUp": 99, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 8 },
        { "followUp": 100, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 7 },
        { "followUp": 102, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 4 },
        { "followUp": 105, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 3 },
        { "followUp": 106, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 2 },
        { "followUp": 107, "probability": 0.9523809523809523, "riskReductionRate": 0.9523809523809523, "numberAtRisk": 1 }];

    it('it estimates kaplanMeier correctly', () => {

        const estimates = estimateKaplanMeier(testProfileMatchStat);

        expect(estimates).toStrictEqual(testProfileMatchStatExpectedEstimates);

        const probabilities = estimates.map(estimate => estimate.probability);

        const riskReductionRates = estimates.map(estimate => estimate.riskReductionRate);

        expect(probabilities).toStrictEqual(riskReductionRates);

    });
})
```

Figure 25: Sample Frontend Test

Asp.Net Core Web API project uses NUnit Test Framework. NUnit Testing Framework is popular, mature, open-source testing framework for .Net platforms. (NUnit 2019) Currently backend test project covers most important methods and better test coverage planned in the future. (Figure 26)

```
using NUnit.Framework;
using Prognomics.Web.ViewModels;
using System.Collections.Generic;

namespace Prognomics.Web.Tests.ViewModels
{
    0 references | Hakan Kucukel, 345 days ago | 1 author, 2 changes
    public class PrognomicsProfileTests
    {
        [Test]
        0 references | Hakan Kucukel, 345 days ago | 1 author, 2 changes | 0 exceptions
        public void PrognomicsProfileHashTest()
        {
            var profile1 = new PrognosisProfileRequest();
            var profile2 = new PrognosisProfileRequest();

            var profile3 = new PrognosisProfileRequest
            {
                CodedFilters = new Dictionary<string, object>(new List<KeyValuePair<string, object>>
                {
                    new KeyValuePair<string, object>("filterName1",1),
                    new KeyValuePair<string, object>("filterName2",new { }),
                    new KeyValuePair<string, object>("filterName3","testValue")
                }),
                Filters = new Dictionary<int, int>(new List<KeyValuePair<int, int>>
                {
                    new KeyValuePair<int, int>(1,1)
                })
            };
            Assert.AreEqual(profile1.GetHashCode(), profile2.GetHashCode());
            Assert.AreNotEqual(profile1.GetHashCode(), profile3.GetHashCode());
        }
    }
}
```

Figure 26: Sample NUnit Test

6 Deployment

Prognomics project takes advantage of Gitlab CI and OpenShift CICD pipelines. (Figure 27) Gitlab CI is configured to build new Docker Images after each source code commit. These images push to Gitlab Docker Registry after successful builds and become available for deploying to OpenShift Cloud Environment. (Figure 28) Automated script pulls newly build image and deploys to staging environment to review application further.

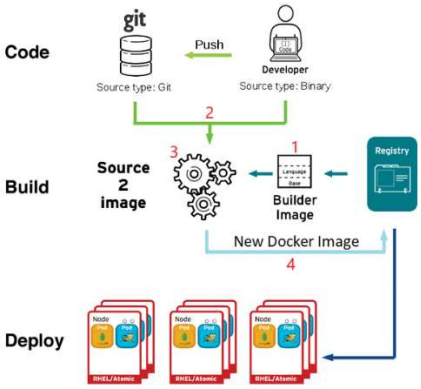


Figure 27: Prognomics CICD Pipeline Overview

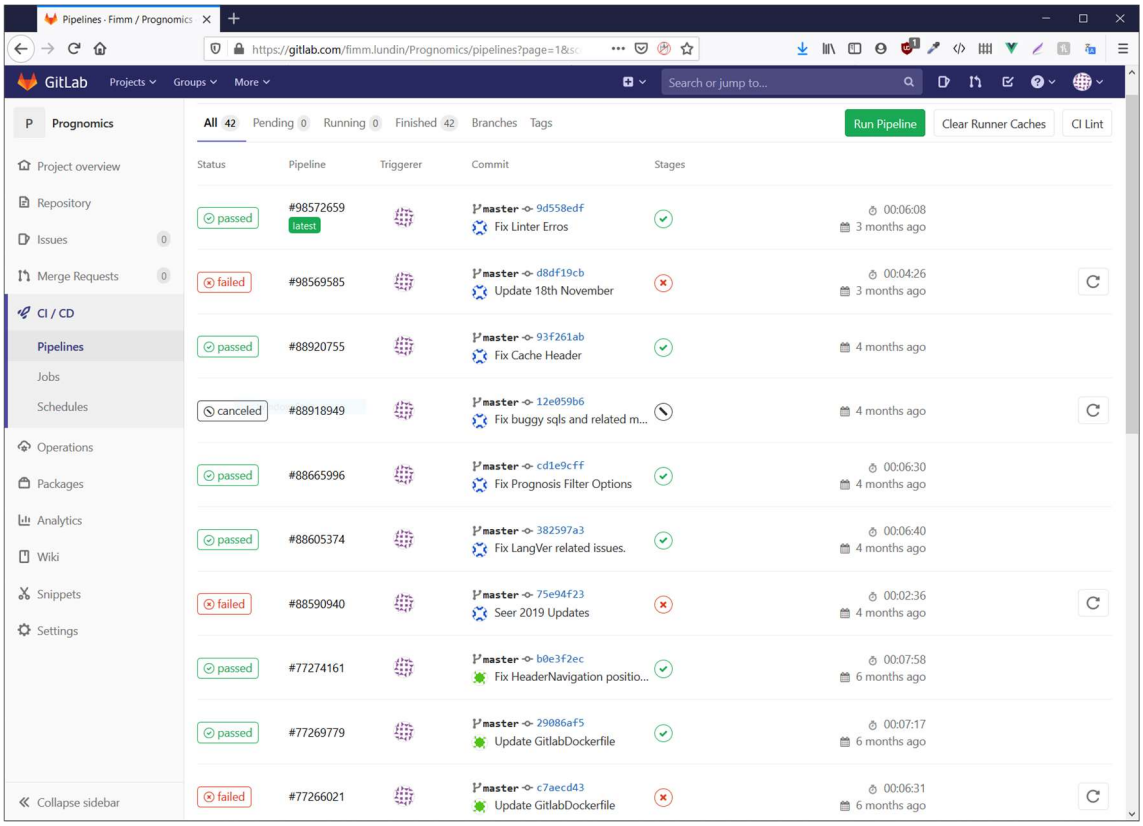


Figure 28: Gitlab CICD Pipeline

7 Methodology

Aim of Prognomics Web Tools is to deliver latest research findings and databases to global clientele of researchers using Applied Research Methodology. Applied research is designed to develop practical solutions for real world problems. It focuses on analysis and finding solutions for existing problems. (Kothari 2008)

Prognomics will provide a platform for researchers or its clientele to publish implementations of their findings based on the dataset of their choice. This dataset can be an existing one on Prognomics or they can bring their own and share with other researchers.

Based on the dynamic nature of the Prognomics Web Tools project Agile Methodology was the best fitting Software Development Life Cycle (SDLC) Model. Main reasons behind this decision was to increase communication and interaction between team members with different fields of expertise. To be able to get continuous feedback and readjusting the requirements based on the client reviews were also another important factor.

7.1 Agile Software Development Lifecycle

Software development was facing a big crisis in the early 90's known as "The Application Development Crisis". The estimated time for application development was about 3 years. This timeline was considerably well over 3 years in some of the important sectors. Traditional software development approaches were timeline based and final product was unveiled to the customer only when it was complete. During development time business needs and requirements were changing rapidly and software couldn't catch up with these changes. In 2001 group of professionals met in Snowbird, Utah and published the Agile Manifesto. (Figure 29). (To Agility and Beyond)

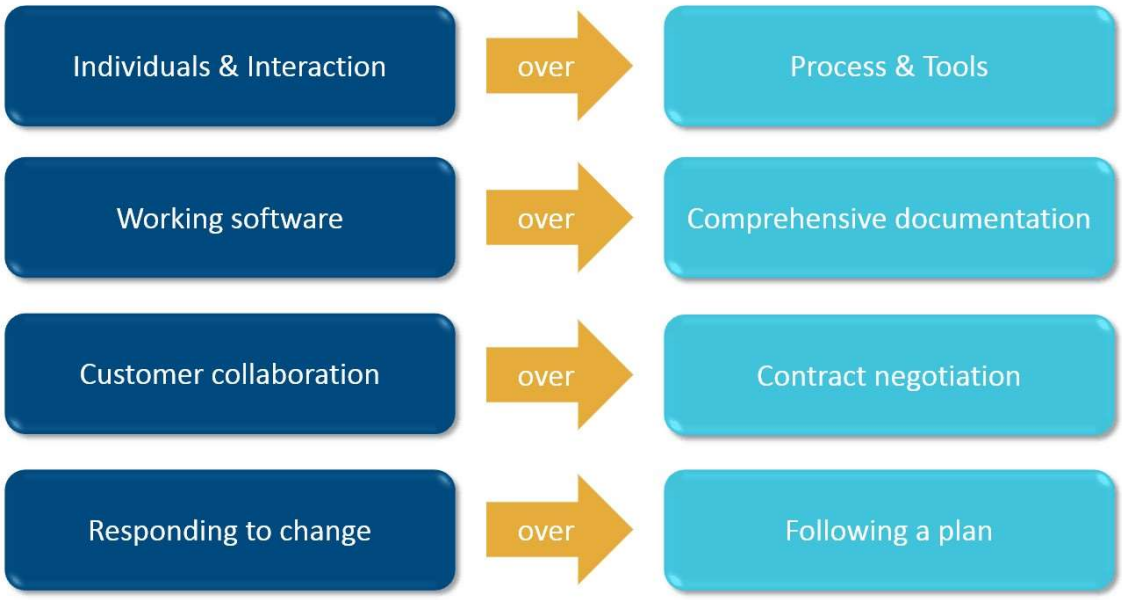


Figure 29: Agile Manifesto - The 4 Values

Agile Software Development Model was born to solve the issues of ineffective software development processes. Agile Model is an incremental development model where requirements and solutions evolve through joint effort and customer needs. (Figure 30)

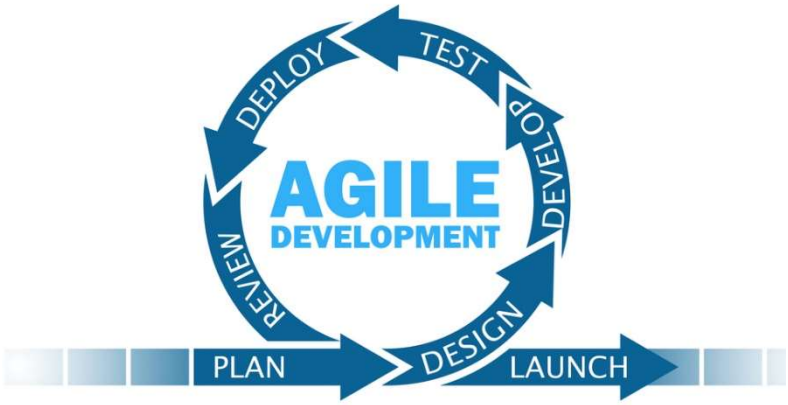


Figure 30: Agile Sprint

Development of Prognomics was started by FIMM Digital Diagnostics Group which is small research group with 8 members. Project was split into smaller achievable goals to deliver a working product and getting feedback to improve the development process. Project tasks were divided into small sub projects as mentioned before on the implementation section. Each iteration or sprint is started by reviewing earlier work and planning the next steps. After the design is approved development and testing phases are started and end result deployed for further user feedback. Slack channel has been created to collect user feedbacks fast and in an organized manner. (Figure 31)

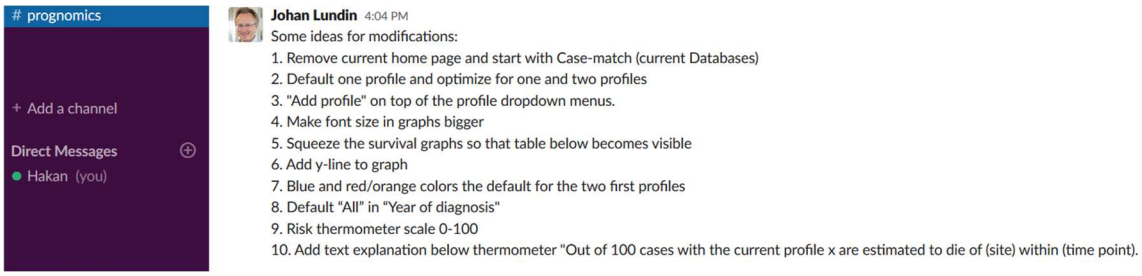


Figure 31: Prognomics Slack Channel

7.2 Analysis

Initially project requirements were analysed to find current problems in the existing prototype. Based on these results new development methodology and tool decisions were made as mentioned on project planning and analysis section.

During implementation of the project visual components are tested on different device sizes to improve overall look and feel of the application and to give best user experience. Network activities and response times are measured and slow components optimized to increase the responsiveness.

7.3 Results

The results are collected in 3 main categories based on the main goals of the project.
One of the intentions of Prognomics Web Tools was bringing all the world-wide available cancer survival data to researchers and healthcare professionals with mobile friendly, easily accessible website. Choosing right tools, design decisions and implementation played an important role. To achieve these goals, project was benefiting from open-source mobile friendly libraries such as Bootstrap and Apex charts. Additional custom controls were also designed and implemented with the same goal in mind. To provide this much information in a clean and accessible manner on the small devices was a big challenge. Results were promising and overall user feedbacks were positive. (Figure 32)

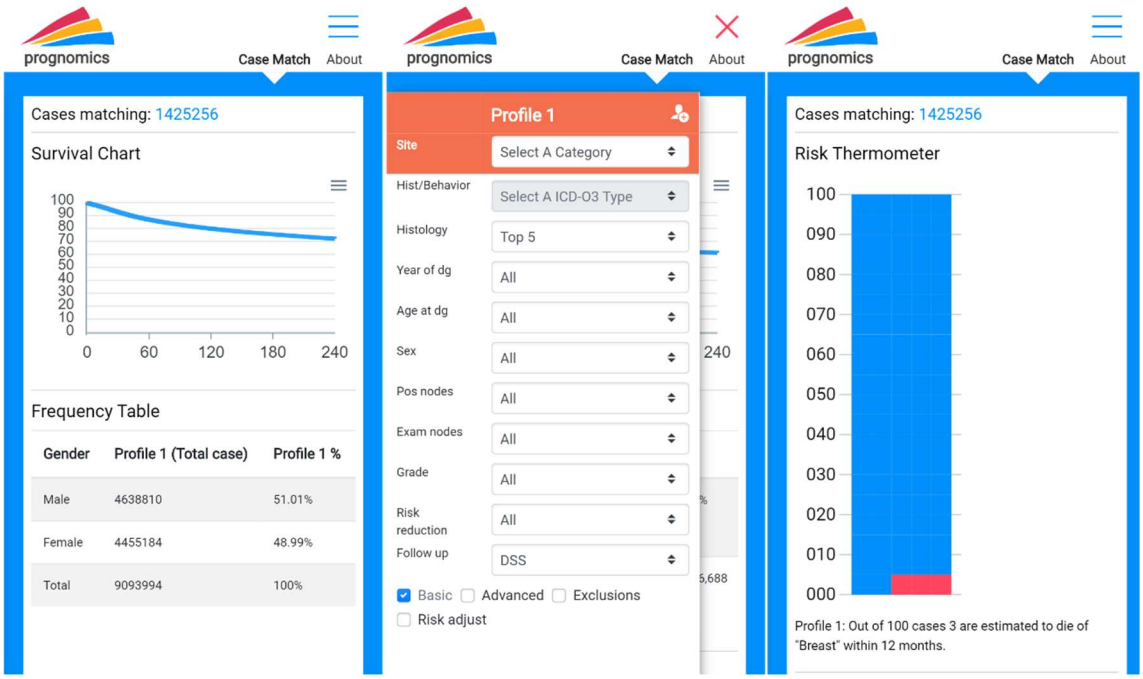


Figure 32: Prognomics Web Tools on IPhone 7/8

Responsiveness is important when it comes to UI design and implementation. Render times are also measured using built-in browser developer tools. Results were 5 frame per second minimum and 45 frame per second average. (Figure 33)

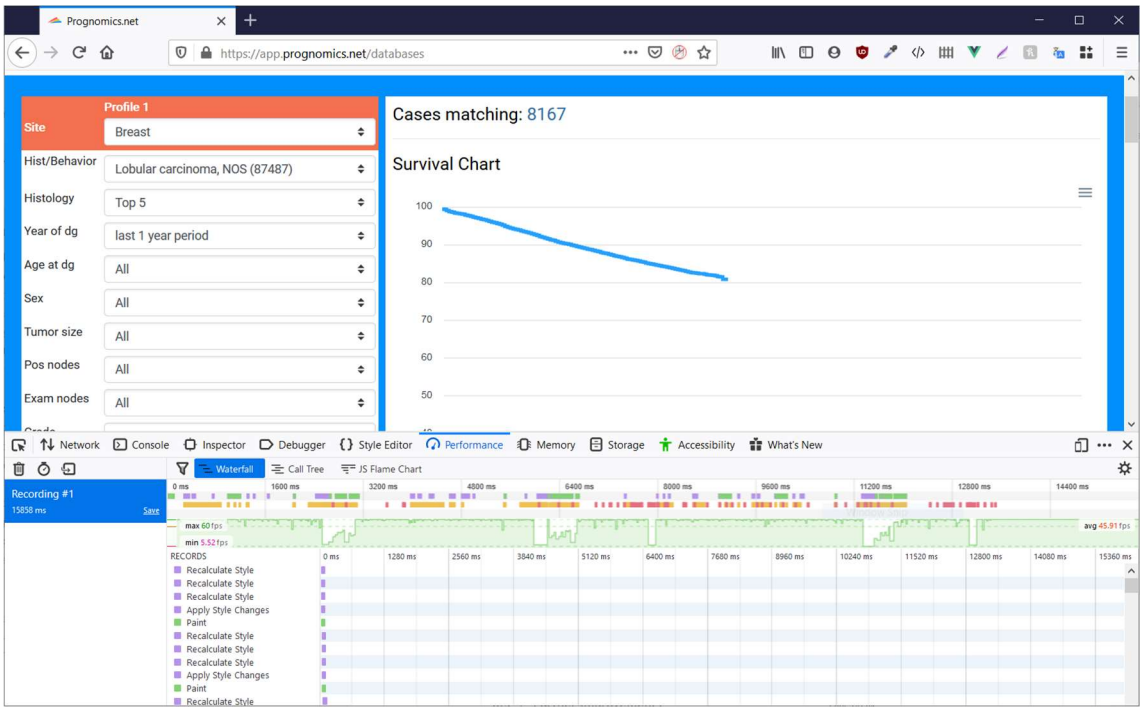


Figure 33: Prognomics Render Time

Network activities were measured and caching used on both server-side and client-side to improve performance. Multipage application was rendering UI components on the server side. This resulted in additional requests from the server and increased the network activity. (Figure 34)

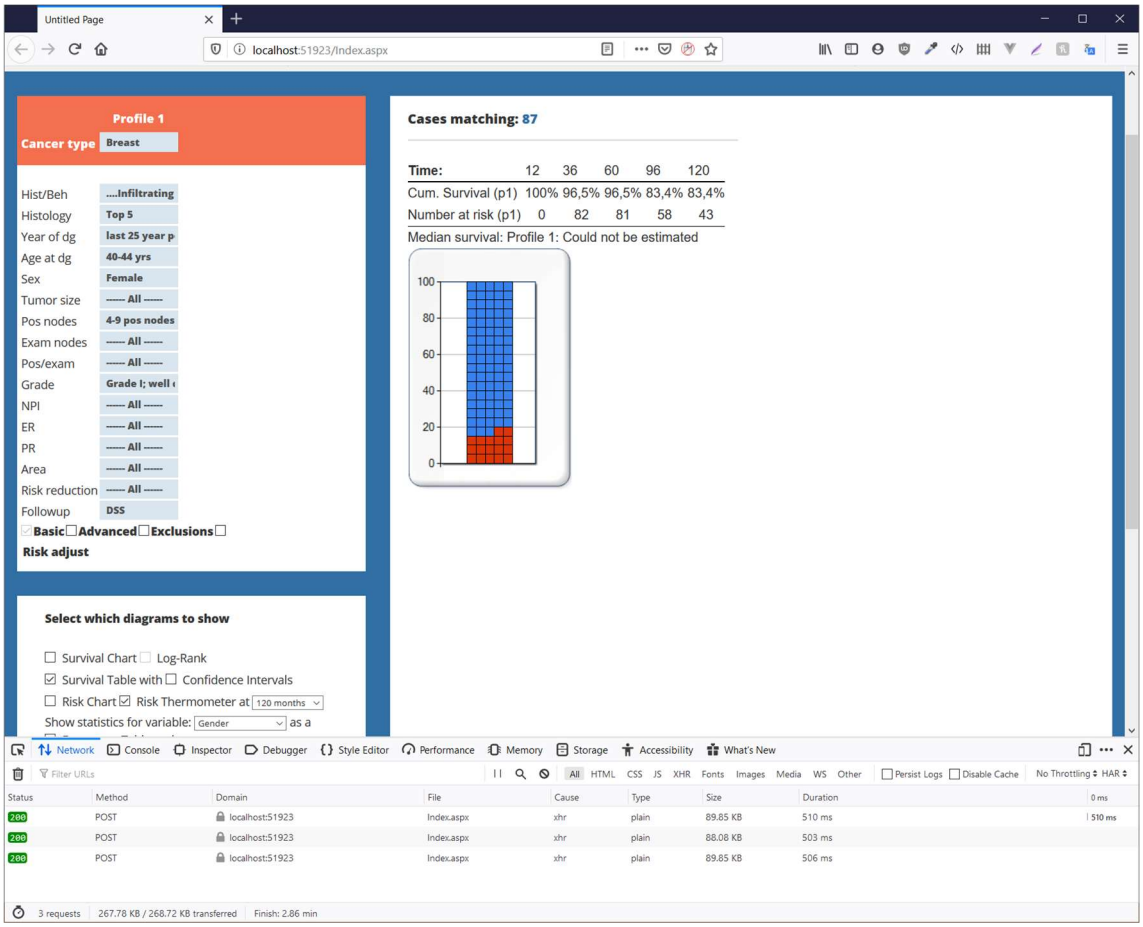


Figure 34: Prognomics Multipage Application

Using SPA architecture and rendering UI components on the client side, all network overhead issues are addressed and unnecessary round-trips to server removed. This also had a positive impact on the amount data transferred from server to client and it improved the overall performance. (Figure 35)

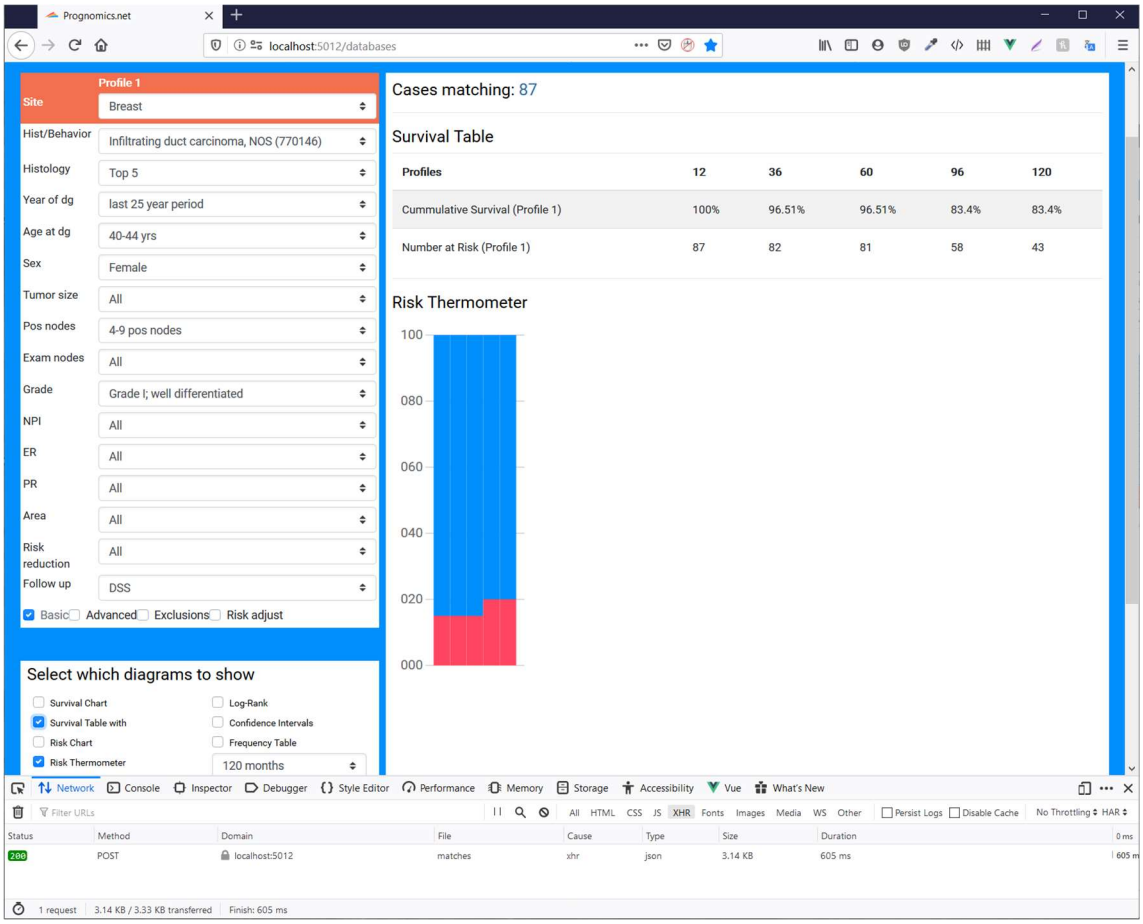


Figure 35: Prognomics Single Page Application

Improving SQL queries were another performance tweak needed to be taken into consideration. These measurements are collected using SQL Server Management Studio and included tools. Overall improvements were between 10-30 % in basic queries. Some of the more complex queries were performing poorly as shown in the Figure 36. Total execution time of the query was well over a second to be completed.

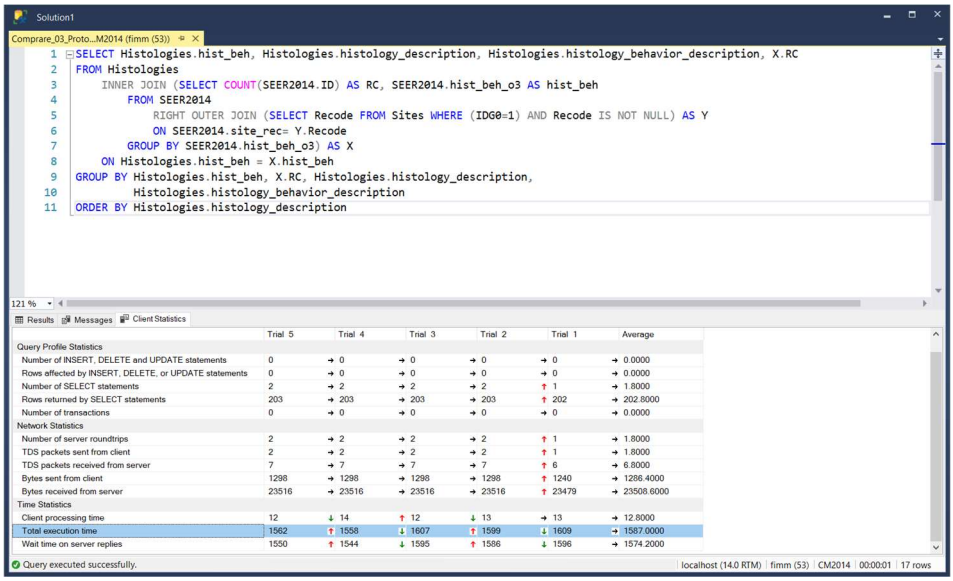


Figure 36: Prognomics Sql Query Generated by Prototype

These performance improvements have resulted in up to 5 times faster execution times as shown in Figure 37. With server-side caching, the server response times have been improved further. Additional improvements are done by using client-side session storage and caching.

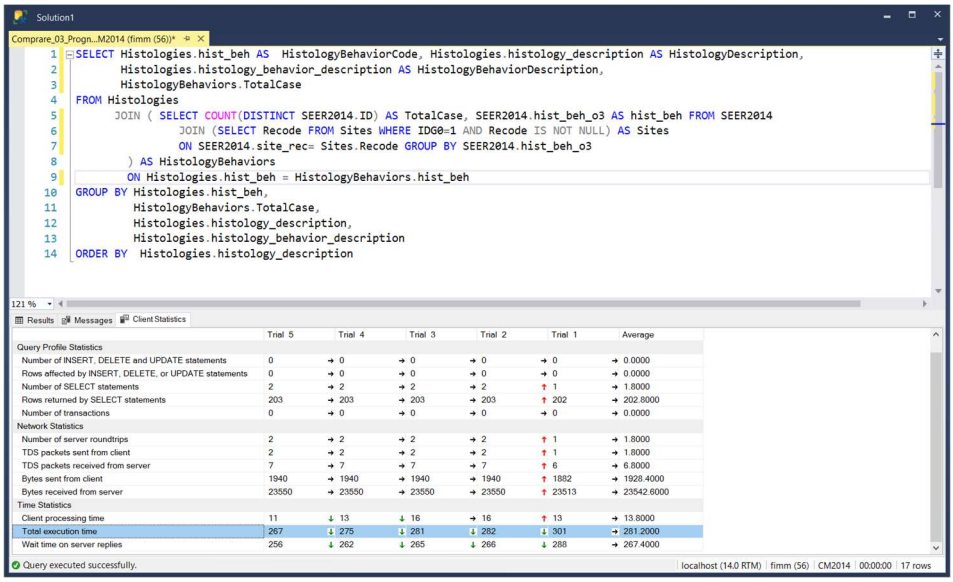


Figure 37: Prognomics Improved Sql Query Sample

8 Conclusion

With the usage of the mobile device growth world-wide in last decade, mobile friendly web-sites became more important to deliver the content to wider audience.

Accessibility being the main objective of this project all the required charts and tables were successfully implemented for both desktop and mobile devices. With faster and responsive UI updates overall user experience has improved. Most of the important components and methods are tested and validated to deliver correct results to users.

SQL queries are optimized for faster execution times. They are also validated to produce correct results based on the pre-defined models.

Prognomics Web Tools has successfully achieved its goals and delivered desired update required by client.

8.1 Met Objectives

The User Experience of Prognomics Web Tools was improved by updating the graphics and presentation of graphs and models. New layout also helped to create a new modern look.

Design was optimized for small devices to provide maximal usability to every end user regardless of their browser or device.

All the generated sql queries are tested for correct result outputs and optimized for better performances.

With Agile Model, user feedbacks are gathered during development process to apply required changes.

8.2 Further Improvements

Some of the improvements are already mentioned on introduction to current problems and solutions sections.

To improve UX even further, current client application can be updated to a progressive web application. Vue offers official support for the required changes.

Further improvements can be done by implementing in memory cache storage. Current database can be replaced with opensource alternative that has better scaling feature.

The backend application is using Monolithic approach and this can be updated to microservice architecture to improve performance and development processes.

References

Printed

Balaji, C. Murugaiyan, M. S. 2012. Waterfall vs V-Model Vs Agile: A Comparative Study on SDLC. Accessed October 2019. http://jitbm.com/Volume2No1/waterfall.pdf

Carlson, Curtis R. & Wilmot, William W., 2006. Innovation. The Five Disciplines for Creating What Customers Want. New York, Crown Business

Damato, B. Taktak, A. 2007.Outcome Prediction in Cancer

El-Bakry, H. Mastorakis, N. 2005. User Interface for Internet Applications https://www.researchgate.net/publication/228676160_User_interface_for_internet_applications

Goel, M. K., Khanna, P., & Kishore, J. 2010. Understanding survival analysis: Kaplan-Meier estimate. International journal of Ayurveda research, 1(4), 274–278. https://doi.org/10.4103/0974-7788.76794

Gregg, B. 2013. Systems Performance: Enterprise and the Cloud. Prentice Hall

Kothari, C.R. 2008. Research Methodology: Methods and Techniques. New Age International

Lehtimäki, T. 2013. Screen Detected Breast Cancer and Prognosis. https://pdfs.semanticscholar.org/0974/25a69796e64f6269e805e20a8cf61198b16f.pdf

Park, J. H., Kim, Y. S., Ryu, J. J., Shin, S. W., & Lee, J. Y. 2017. Cumulative survival rate and associated risk factors of Implantium implants: A 10-year retrospective clinical study. The journal of advanced prosthodontics, 9(3), 195–199. https://doi.org/10.4047/jap.2017.9.3.195

Smith, S. 2019. ASP.NET Core Architecture. Washington. Microsoft Developer Division

Stellman, A. Greene, J. 2015. Learning agile. Sebastopol, CA: O'Reilly.

Electronic

A Hands-On Guide to Mobile-First Responsive Design. Accessed September 2019. https://www.uxpin.com/studio/blog/a-hands-on-guide-to-mobile-first-design/

A Web-based System for Individualised Survival Estimation in Breast Cancer, 04 January 2003. Accessed 2019 September 2019. https://www.bmj.com/content/326/7379/29.full

About Gitlab. Accessed February 2020. https://about.gitlab.com/company

Accessing the 1975-2016 SEER Data. Accessed January 2020. https://seer.cancer.gov/data/access.html

Asp.Net Web Apps. 2019. Accessed December 2019. https://dotnet.microsoft.com/apps/aspnet/web-apps

Asp.Net Web Forms. Accessed December 2019. https://dotnet.microsoft.com/apps/aspnet/web-forms

CSC Blog. 15 August 2019. Introducing the Rahti Container Cloud. Accessed September 2019. https://www.csc.fi/web/blog/post/-/blogs/introducing-the-rahti-container-cloud

Cancer Registry. Cancer Information Notification. https://cancerregistry.fi/data-collection/

Client-Server Overview. January 2020. Accessed January 2020. https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview

Common Web Application Architectures. April 2019. Accessed September 2019. https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures

European Textbook on Ethics in Research. Accessed September 2019. http://ec.europa.eu/research/science-society/document_library/pdf_06/textbook-on-ethics-report_en.pdf

FIMM in a nutshell. 2017. Accessed September 2019. https://www.fimm.fi/en/fimm/about-fimm/fimm-nutshell

FIMM. 2017. Accessed September 2019. https://www.fimm.fi/node/20

FinProg. Accessed September 2019. http://www.finprog.org/objective2.asp

HTML 4.01 Specification. Accessed December 2019. https://www.w3.org/TR/html401/

HTML5. Accessed December 2019. https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5

HTTP. Accessed September 2019. https://www.w3.org/Protocols/HTTP/1.1/rfc2616bis/draft-lafon-rfc2616bis-02.html

Hypertext. Accessed September 2019 https://www.britannica.com/technology/hypertext

JestJS. 2019. Accessed September 2019. https://jestjs.io

Kubernetes Infrastructure. Accessed September 2019. https://docs.openshift.com/enterprise/3.0/architecture/infrastructure_components/kubernetes_infrastructure.html

Launch of FIMM. 2014. Accessed September 2019. https://www.fimm.fi/en/fimm/about-fimm/establishment

Lawson, K. 19 July 2018. What Is a Single Page Application and Why Do People Like Them So Much? 2018. Accessed September 2019. https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html

Mobile first: the origins and the current meaning. 5th March 2012. https://lazure2.wordpress.com/2014/02/18/mobile-first-the-origins-and-the-current-meaning/

NPM. 2019.. Accessed September 2019. https://www.npmjs.com

NUnit. 2019. Accessed September 2019. https://nunit.org

Nina Linder. 2017 Accessed September 2019. https://www.fimm.fi/en/nina-linder

Node JS. 2019. Accessed September 2019. https://nodejs.org/en/about

Number of smartphone users worldwide from 2016 to 2021. September 2019. Accessed January 2020. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

OpenShift. Accessed September 2019. https://docs.openshift.com/enterprise/3.0/architecture/infrastructure_components/kubernetes_infrastructure.html

Percentage of all global web pages served to mobile phones from 2009 to 2018. 22 July 2019. Accessed September 2019. https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/

Research Ethics. 2019. Accessed September 2019. https://www.helsinki.fi/en/research/research-environment/research-ethics

Single Page or Multi-Page Application: Which One is Better. 29 July 2019. Accessed September 2019. https://www.esds.co.in/blog/single-page-or-multi-page-application-which-one-is-better/#sthash.oaLd9GBd.Xe3fPvCu.dpbs

Sinha, R. 29 July 2019. Single Page or Multi-Page Application: Which One is Better? Accessed December 2019. https://www.esds.co.in/blog/single-page-or-multi-page-application-which-one-is-better/#sthash.oaLd9GBd.dpbs

To Agility and Beyond: The history—and legacy—of agile development. Accessed February 2020. https://techbeacon.com/app-dev-testing/agility-beyond-history-legacy-agile-development

Vue JS. 2019. The Progressive JavaScript Framework. Accessed September 2019. https://vuejs.org/

What is Docker. 31 August 2018. Accessed September 2019. https://docs.microsoft.com/en-us/dotnet/architecture/microservices/container-docker-introduction/docker-defined

What is NPM. Access September 2019. https://www.w3schools.com/whatis/whatis_npm.asp

What is a Container. Accessed September 2019. https://www.docker.com/resources/what-container

What makes us special. 2017. Accessed September 2019. https://www.fimm.fi/en/fimm/about-fimm/what-makes-us-special

Worldwide mobile app revenues in 2014 to 2023.1 August 2019. Accessed September 2019. https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast/

Unpublished

Linder, N. Interview August 8th 2019.

Figures

Tables

Abbreviations

AAD Azure Active Directory

API Application Programming Interface

CICD Continuous Integration and Continuous Deployment

CLI Command Line Interface

CRUD Create, Read, Update, Delete

CSC IT Centre for Science

EMBL European Molecular Biology Laboratory

FIMM Institute for Molecular Medicine Finland

HiLIFE Helsinki Institute of Life Science

IDE Integrated Development Environment

JS JavaScript

NPM Node Package Manager

POCO Plain Old CLR Objects

RCR Responsible Conduct of Research

SDLC Software Development Life Cycle

SEER The Surveillance, Epidemiology, and End Results

SPA Single Page Application

SQL Structured Query Language

SSMS SQL Server Management Studio

THL Terveyden ja Hyvinvoinnin Laitos (National Institute for Health and Welfare)

UI User Interface

UX User Experience