Nutri-Fitness Tracker Web Application

Josué J. Negrón Santiago
Master in Computer Science
Advisor: Jeffrey Duffany, Ph.D.
Electrical and Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico

Abstract *— There are several health programs that intend to help people to drop weight in a natural manner. Novovida Health & Fitness is such a program, which promotes a better eating habit and physical activities in order to stimulate weight loss. The problem is that many of these organizations lack the technology, or an information system, that will enable them to keep track of each user's progress within the program. Currently, many of these organizations keep track of their members improvements by using messy excel spreadsheets or even in a more oldy fashion way; paper and pencil. Nutri-Fitness Tracker is a web-application solution, that enables these organizations with a system that keeps track of their members improvements. Nutri-Fitness is a responsive web-application with an easy GUI that provides its members with a graphical representation of their improvements over time.*

Key Terms *— Fitness, Nutrition, Tracking, Web-Application.*

INTRODUCTION

Novovida Health & Fitness is a health-oriented business. Their primary focus is to help people drop weight naturally by adding a combination of nutrition, exercise and other techniques. Their main goal is to improve the overall well-being of the individuals that participate in their program.

Initially, when the individual joins the program, the first step is to fill up a questionnaire. The questionnaire is based on the PAR-Q. PAR-Q stands for Physical Activity Readiness Questionnaire. This questionnaire is a tool which helps the physical trainers become aware of possible sanitary or cardiovascular problems on the individual. With this information on hand then, the physical trainers can determine the intensity of the workouts the individual should follow.

Once a work plan is designed for the individual, Novovida keeps track of their members improvements. At a monthly basis, Novovida tracks the weight and body measurement (chest, hip & waist) for the individual.

So, I was approached by Novovida, with the expectations to come up with a technological solution that would help them standardize the recording of their member's improvements.

REQUIREMENTS

The Nutri-Fitness Tracker web application has the following requirements:

- Users should be able to authenticate using Google's sign-in API; with OAuth 2.0 protocol
- The system should store all information in a relational database
- The system should be role based; the application administrators should have access to specific content that the end-user shouldn't
- The system should calculate a "Health Overall" percentage
- The system should keep track of the user's physical improvements
- The system will provide the administrator with a section for recording the end-user's body measurements
- The system should provide a graphical representation of the end-user's improvements
- The system's Front-End should be "mobile first" design oriented

Google Sign-In

Google Sign-In is a secure authentication system that reduces the burden of login for users, by enabling them to sign in with their Google Account—the same account they already use with Gmail, Play, and other Google services.

Mobile First

Mobile-first design and layout (shown in Figure 1) are based on providing excellent mobile user-experience: fast download speeds, easy touchscreen navigation and so on. This approach was adopted because currently, 52.64% of the total traffic on the Internet is done via mobile phones, and by the end of the year experts from Zenith Media predict an increase of up to 75% [1].

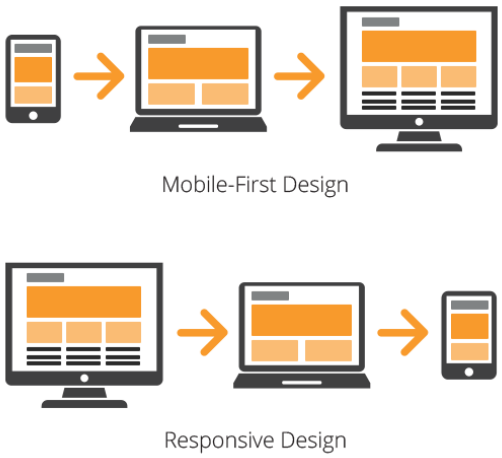

Figure 1
Mobile-First Diagram

SDLC Methodology

For this project, I decided to use the "Waterfall" development methodology. The Waterfall Model (shown in Figure 2), also referred to as a linear-sequential life cycle model, illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap [2]. For each phase of the waterfall methodology I did the following:

- **Requirement and Analysis:** In this initial phase, a set of meetings with the client were made in order to gather and document all the requirements needed for the system.
- **System Design:** Once all the requirements were gathered and analyzed, I proceeded to make a series of mockups. These mockups included variations on the front-end side of the application. For the creation of the mockups, I used an online tool called "moqups", which enabled me to create a series of templates in order to show to the client.
- **Implementation:** Once a design pattern was established, and the client agreed with the system design, I started the implementation process.
- **Integration and Testing:** Since the implementation process started, a series of tests were performed each time a phase within the application was completed.
- **Deployment:** Once a set of "stress tests" were made to the application in a test environment, the next step was to deploy the application into a working/production environment.

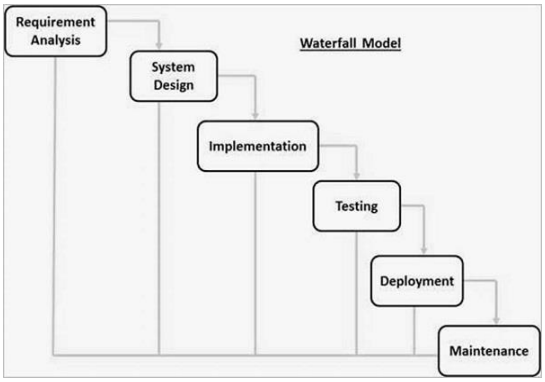

Figure 2
Waterfall Model Diagram

Web Application Development

In the software world, web application development is the creation of application programs that reside on remote servers and are delivered to the user's device over the internet. An end user can access a web application through a web browser. There are several tools and programming languages that can be used for the creation of a web application. For this project, I decided to use Microsoft tools that are well known in the software industry.

- **Database**: For the database, SQL Express was used in order to store the required information of the web application. SQL Express is a free version of Microsoft's primary relational

database management system (RDBMS) – the SQL Server. Essentially, the SQL Server is a database management system that can be used to store and access the information stored in many different databases. I started off by creating a local database called: FitnessTracker

- **Tables**: In the first phase of the development process, an analysis was made in order to determine the required information that would be stored in the database. With this information on hand, a series of relational tables were created; as shown in Figure 3.

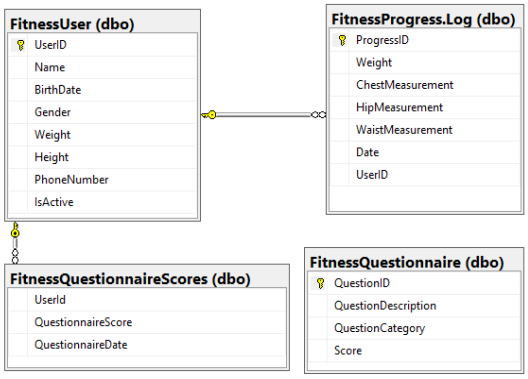

Figure 3
FitnessTracker Tables Diagram

- **Stored Procedures:** Once the tables were created, the next step was to create a series of stored procedures that will interact with the tables. Basically, a CRUD functionality was implemented for each table. CRUD stands for Create, Read, Update and Delete. The CRUD paradigm is common in constructing web applications, because it provides a memorable framework for reminding developers of how to construct full, usable models [3].
- **.Net Core Web API:** For the application to interact with the database, a middle component was needed. There's were the Web API comes into play. A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. A RESTful API -- also referred to as a RESTful web service -- is based on representational state transfer (REST) technology, an architectural style and approach to communications often used in web services development. For this web application I decided to use .Net Core Web API over WCF (Windows Communication Foundation) because REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST leverages less bandwidth, making it more suitable for internet usage.
- **Front-End & Back-End:** For the web application design pattern, I decided to implement ASP.NET MVC. MVC stands for Model-View-Controller Pattern. MVC is a design pattern used to decouple user-interface (view), data (model), and application logic (controller). This pattern helps to achieve separation of concerns. Using the MVC pattern for websites, requests are routed to a Controller which is responsible for working with the Model to perform actions and/or retrieve data. The Controller chooses the View to display and provides it with the Model. The View renders the final page, based on the data in the Model.

REPOSITORY

- **Git:** For a version controlling system, I decided to use Git. Git is a free repository system, which enabled me to backup, make commits, and make branches of my code. Git integrates with MS Visual Studio, making it easier for me to maintain my code backed-up and in a version-controlled environment.

DEPLOYMENT

- **Microsoft Azure:** For the deployment of the application, into a production environment, I decided to use Microsoft's Cloud Platform: Azure. Azure has a series of "Application Services", which enables to quickly build, deploy, and scale web apps. With Azure, I was able to host the web-application, Web API, and the database. Azure integrates with MS Visual Studio, which makes it very practical and simple to deploy each change. Figure 4 below

shows a diagram with some of App Services provided by Azure.

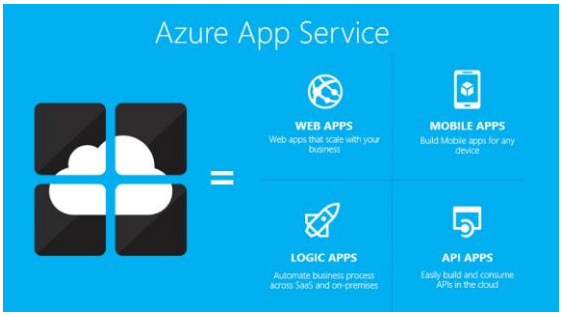

Figure 4
Azure App Service Diagram

APPLICATION LAYOUT

To start wrapping things up, this section will include a series of images showing a working copy of the application. The application is hosted at http://nutri-fitness.azurewebsites.net. Figure 5, shows how the web-application resembles the registration section.

Questionnaire

For the system to calculate an initial "Health Overall" percentage, each question has an assigned value. These values are provided by the nutritionist based on the PAR-Q method. Figure 6, shows how the web-application resembles the questionnaire section.

Modify

Once the user enters the required input, latter on these fields can be modifies. The system provides with several maintenance views (MVC), which each interacts with the Web API in order to maintain the application's data. Administrators of the application also have their maintenance views where they can handle the end-user's information.

Profile

Profile is the main page for the end-user. In this section, the system provides a graphical representation of the end-user's "Health Overall". In addition, the most current body measurements, for the end-user are shown. Figure 7, shows how the web-application resembles the profile section.

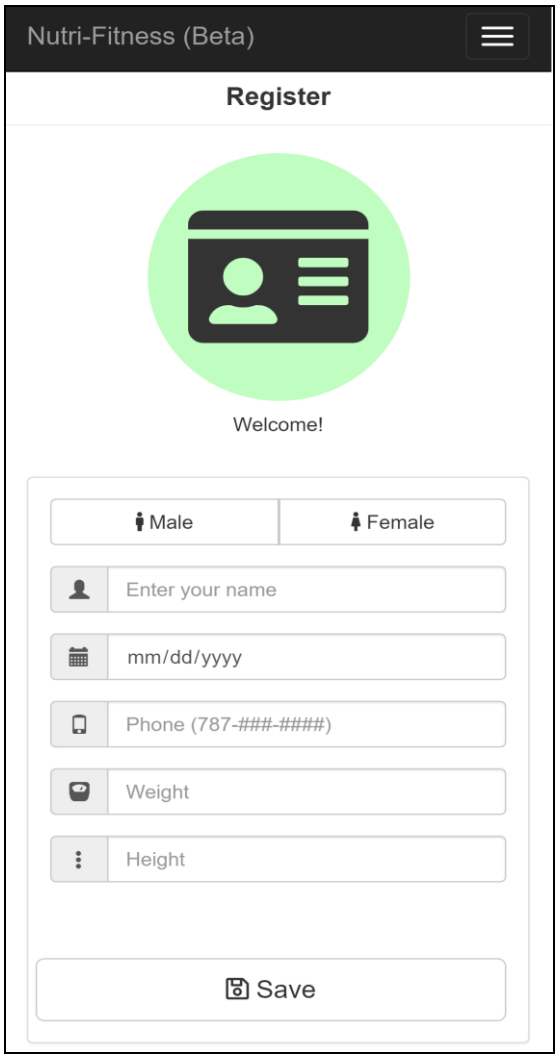

Figure 5
Nutri-Fitness Register Page

Progress

The Progress page shows end-users a graphical and detailed representation of their monthly progress. Figure 8, shows how the web-application resembles the progress section.

IMPLEMENTATION TOOLS

The following tools where used for the development of the Nutri-Fitness Tracker web-application:

- Visual Studio 2019 Community
- ASP.NET Web Application (.Net Framework)
- Model View Controller (MVC) Structure
- C# Programming Language

- .Net Core Web API
- SQL Express
- Git
- Azure App Services

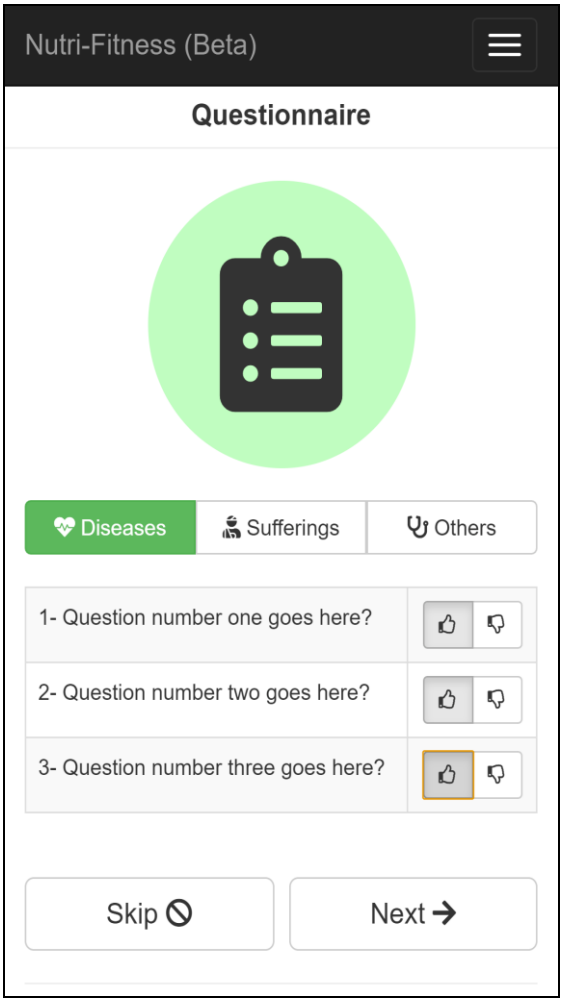

Figure 6
Nutri-Fitness Questionnaire Page

CONCLUSION

In this project, a complete SDLC (Software Development Life Cycle) of the Nutri-Fitness Tracker web-application is shown. The system meets the client's initial requirements for tracking end-user's progress; and further shows a graphical representation for the end-user's improvements over time.

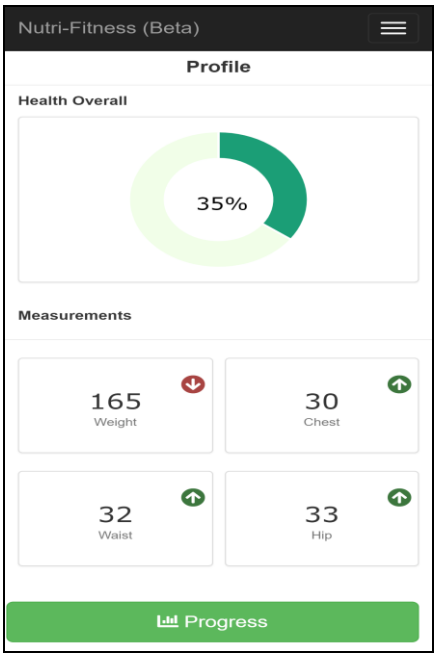

Figure 7
Nutri-Fitness Profile Page

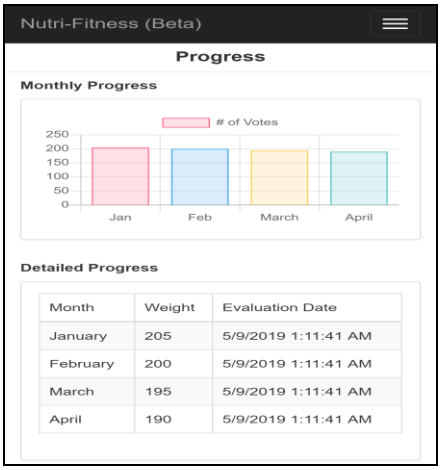

Figure 8
Nutri-Fitness Progress Page

REFERENCES

[1] D. Kaempf. (2018, March 28). *Mobile-First Web Design vs. Responsive Web Design*. [Online]. Available: https://darwindigital.com/mobile-first-versus-responsive-web-design/. [Accessed: May 1, 2019].

[2] Tutorialspoint.com (2017, May 15). *SDLC Waterfall Model* [Online]. Available: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm.

[3] Codecademy. (2018, October 11). *What is CRUD*? [Online]. Available: https://www.codecademy.com/articles/what-is-crud.