



GRAND CANYON
UNIVERSITY™

iHeart Rate

Bryce Schmisser

CST-451 Capstone Project Requirements Document

Grand Canyon University

Instructor: Professor Mark Reha

Version 1.0

1 November 2020

Abstract

The project consists of two main applications, an Apple Watch app and a website. The combination of these two applications provide a user with a more informational and interactive interface to display the user's heart rate data. From the user's Apple Watch the application will send the heart rate information collected by it and send the data to the website. The website will be able to display the collected information such as the current heart rate and view past heart rate data entries. The website will also offer inputs to select certain period of time to display that on a table and chart for visual aid. Overall this project is to be able to get more use of the information collected on a person smart watch.

History and Signoff Sheet

Change Record

Date	Author	Revision Notes
24 - 31 October 2020	Bryce Schmisser	Initial draft version 1.0

Overall Instructor Feedback/Comments

Overall Instructor Feedback/Comments

Integrated Instructor Feedback into Project Documentation

☒ Yes ☐ No

Project Approval

☐ Professor Mark Reha

TABLE OF CONTENTS

FUNCTIONAL REQUIREMENTS	4
NON-FUNCTIONAL REQUIREMENTS	5
TECHNICAL REQUIREMENTS	6
LOGICAL SYSTEM DESIGN	7
USER INTERFACE DESIGN	8
REPORTS DESIGN	10

Functional Requirements

Use Cases

All user stories can be found in the excel document called Sprint_Back_log.xls. In the tab Functional (Web) it lists all the functional requirement that will be implemented in the web application. In the tab labeled Function (Watch) outlines all the functional requirement that will be in the final version of the apple watch application.

Out of Scope

The chart below outlines all the Function user stories that have been out of scope due to time management or not enough knowledge of how to implement the feature. More information of both these user stories can be found in the excel sheet Spring_Back_Log.xls in the tab called Functional Out of Scope.

Use Case or User Story	Approval Date	Justification
Live Feature		Struggling with figuring out how to implement this feature so until I know more, I am going to keep it out of scope
Exporting User Data		Taking the feature out of scope for now until I see how much time I have to complete all the other function features of the project

Non-Functional Requirements

Use Cases

All user stories can be found in the excel document called Sprint_Back_log.xls in the tab called Non-Functional. These outline all the non-function requirements that will be implemented and tested in the final project.

Out of Scope

The chart below outlines all the non-function user stories that have been out of scope due to time management or not enough knowledge of how to implement the feature.

Use Case or User Story	Approval Date	Justification
None at this time		

Technical Requirements

Use Cases

Listed in the chart below are all the technologies that will be used through the entirety of this project:

Technology or Tool	Version	Description
React	17.0.1	React will be used for the front end of the website application
Node JS	14.15.0	As the server for both ReactJS and Express JS
Express JS	12.16.3	Using the Express framework, the Rest API for the backend
Mongo DB (using Atlas)	4.4.0	Will be a cloud deployed database that will be used both for the web application and apple watch application
Visual Studio Code	1.49.1	VS Code will be used to edit the code for react and Node
Postman	7.36	Postman is an application is used to send HTTP Request in order to test the Rest API
MongoDB Compass	1.22.1	A graphical user interface uses for mongo DB
GitHub	2.22.2	A software used to backup and share all code easily
Jira	8.13	Project management tool used for to help keep the project on track
XCode	12.1	Used as an editor for XCode in creating the apple watch application
Swift	5.2	Used to create the front end and backend of the apple watch application
Docker	19.03.13	Docker will be used to containers the application as it will

Out of Scope

The chart below outlines all the technologies that have been out of scope due to its relevance to the project:

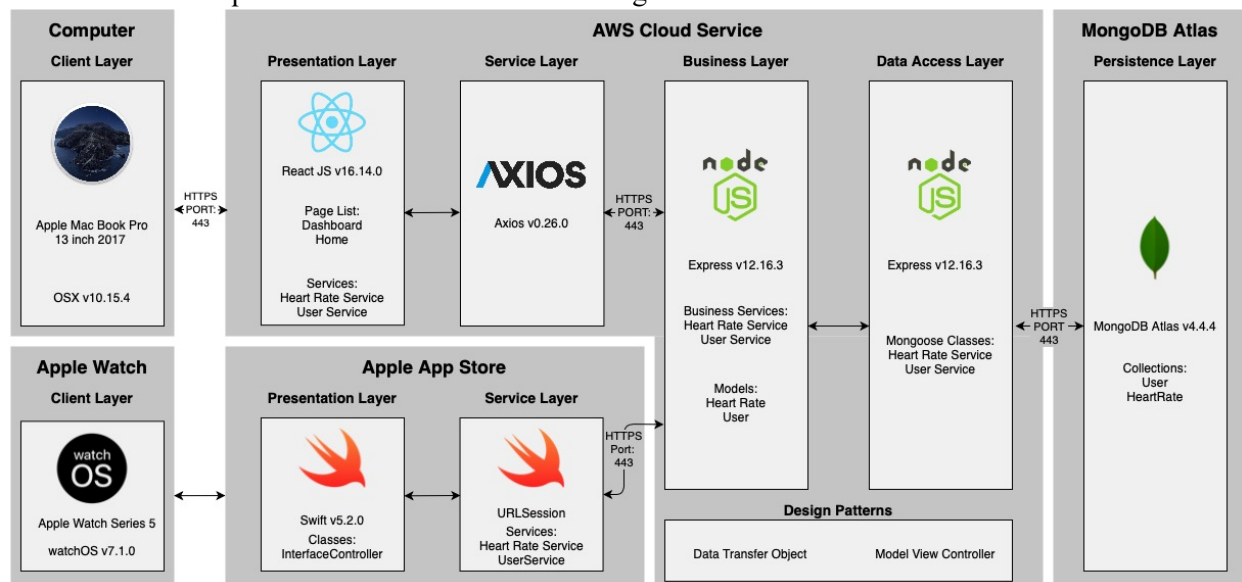
Technology or Tool	Approval Date	Justification
None at this time		

Logical System Design

The logical system design of the iHeart Rate project describes how the code will logically flow through the application. Starting from the client layer the user can interact with the project on two platforms: web browser and Apple Watch. An Apple MacBook Pro and Apple Watch Series 5 are listed as they are the two devices that the application is being developed and tested on.

The front end of the web application will be deployed using Vercel and the backend is deployed with Heroku. The user will interact with the front end of that application that is written with the React framework. React makes a use of components and services in order to display and retrieve information to and from the user. The components for this application include home, main and sign in to create all the views of the application. These components make use of the user and heart rate service in order to create REST API calls to the back end using the library of Axios. Axios is a JavaScript library that has the ability to make HTTPS request. These HTTPS requests are handled by the Express application for the back end of the application. The routes within the express application can pick out the data form the HTTPS request and create models from the data. Within Express there are two models that are implemented one for the user object and another for the heart rate object. These models make it easy to send the object data between both business and data services as the Express application uses the DAO design pattern. The business services either, user business service or heart rate business service are created in order to transfer data from the routes to the data services. There are only two data services as there are only two object models that the application deals with. The data services connect the backend of the application to the MongoDB Database. The database that is running on the cloud using Mongodb Atlas which contains two collections: user and heart rate in order to hold documents of the object models.

The application has a similar path of data and code logic however the beginnings of the process are different. As the Apple Watch app cannot be deployed on a cloud platform it must be hosted from the Apple App Store. The front end and service layer code for the apple watch app will be done in Swift. Swift is a responsible for setting up the front end of the application and making the REST API calls to the back end. With the addition of the URL Session library provided by Swift the creation of HTTP requests is just as easy as Axios. The HTTP request is then sent to the Express application in AWS just as Axios did with the request from React. After the request reaches the Express application, the data follows the same flow as the request from React did to reach MongoDB.



User Interface Design

Web Application

The first page, *Figure 2-1*, that will be displayed to the user will be the home page where the iHeartRate Applications are described. There will be a little section on why the app was created as well as how the app works with the apple watch. This page will also contain the ‘Sign in with Apple’ button that will lead them to the user’s dashboard.

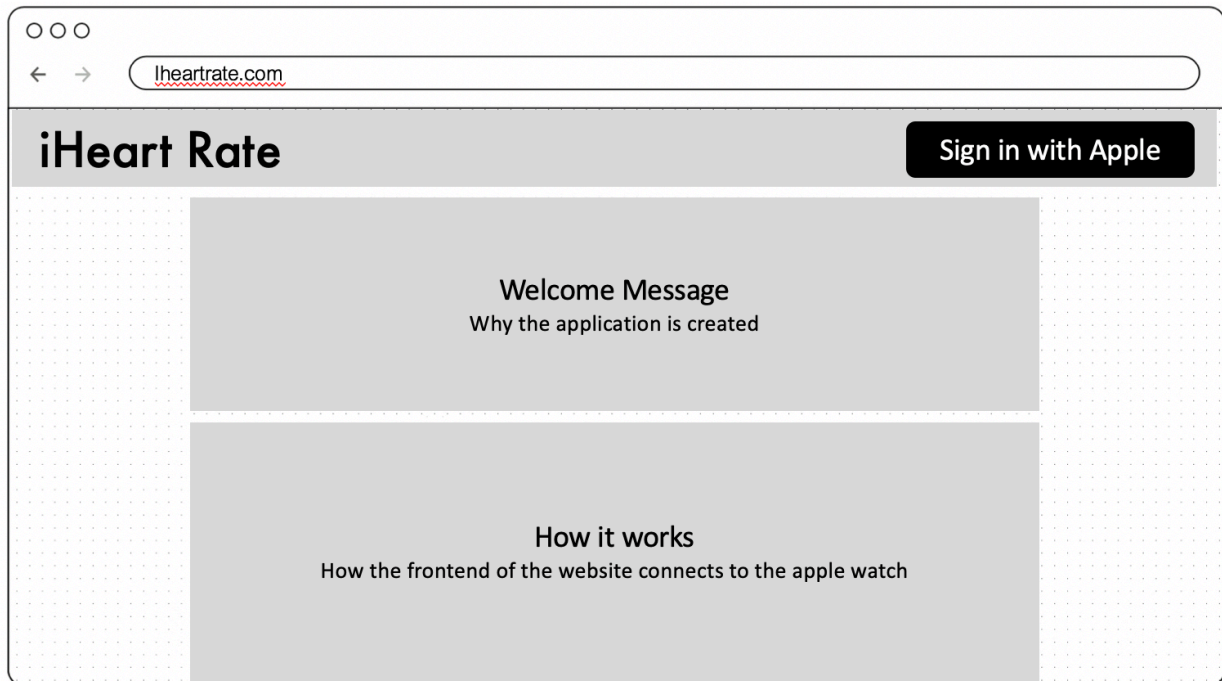


Figure 2-1. Home Page of Web application

The dashboard page shown in *Figure 2-2* that the user can access within the web application is the main page displaying all the users heart rate information. This is where the user will spend most their time as it is the page where the information is actually provided to the user. The page is set up with a side menu that displays a welcome message, their own image, name, and the contents of the page. The right panel of the page will contain a graph, a paginated table, the average heart rate, and the last recorded heartrate. Lastly in the upper left and corner there will be a button in order to select dates. Once pressed it will give a popup to the user that will allow them to enter certain dates and time in order to grab heart rate information with the selected time span.

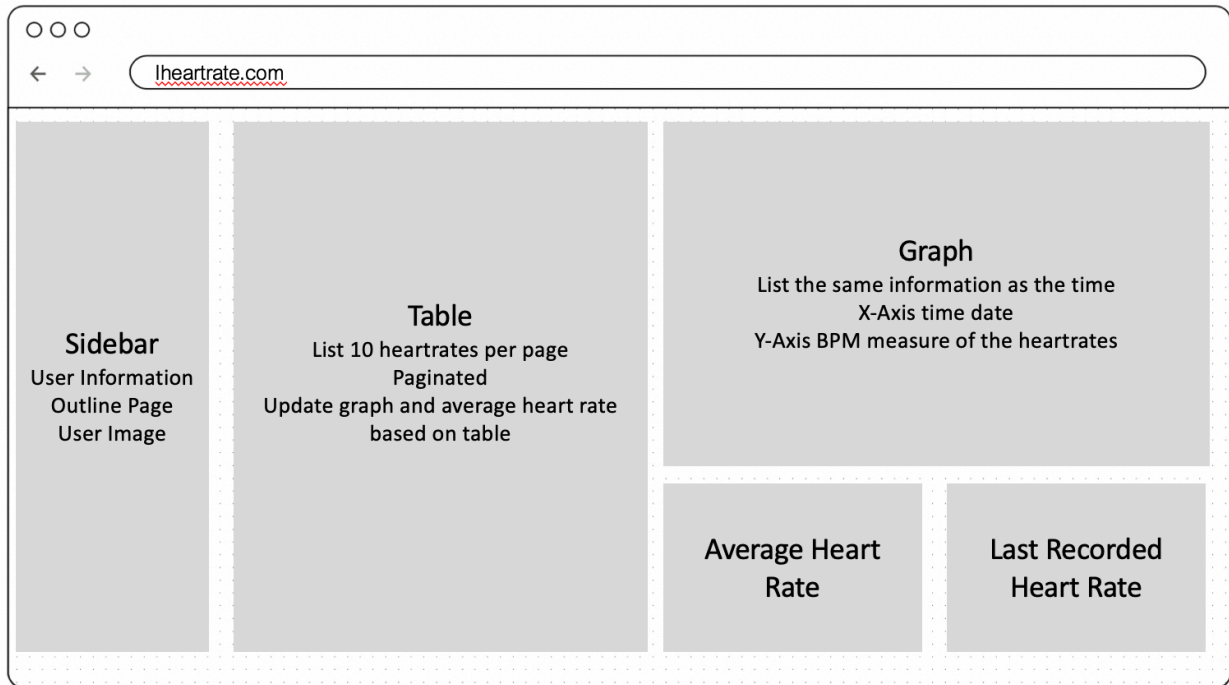


Figure 2-2. Dashboard Pages

Apple Watch Application

Just as the website did the Apple Watch application will only contain two pages, Figure 2-3. The first page being a welcome page will display a welcome message and the Sign in Apple Button. The second page will contain a singular button that will change to start if the application is off but stop if the application is recording.

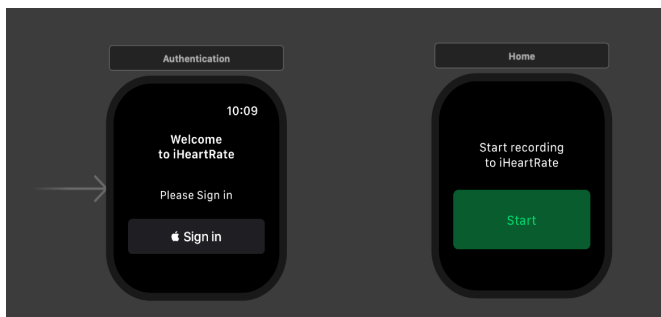


Figure 2-3. Apple Watch Application Wire Frame

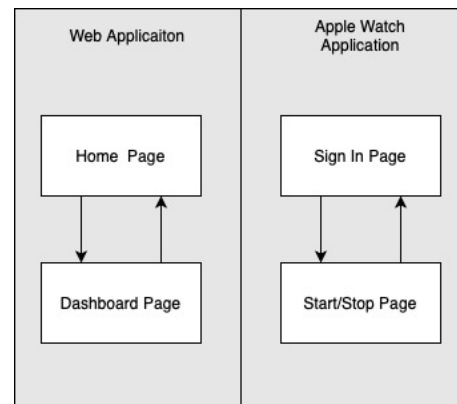
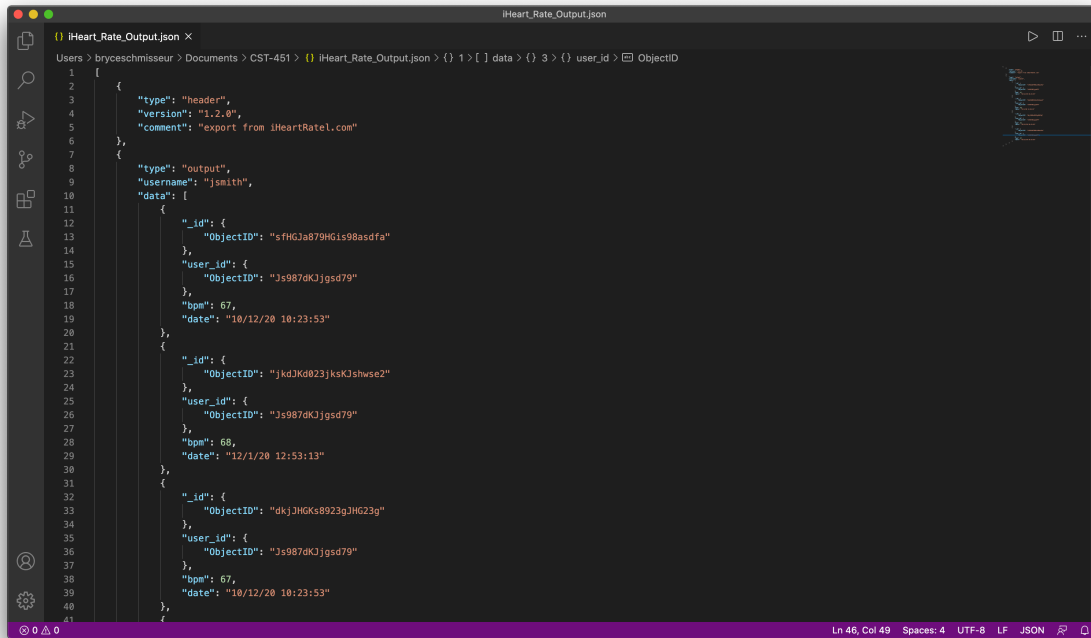


Figure 2-4. Sitemap

Reports Design

User Output Log

The document below will be a file that the user will be able to export from the web application of iHeart Rate. This file will contain all the heart rate information that has been collected from their apple watch as seen in seen in the example. This file will be formatted in JSON in order to make it easier to read for the user. This will also give the user the ability to upload this information to any other application that will accept it.



```
1 {
2   {
3     "type": "header",
4     "version": "1.2.0",
5     "comment": "export from iHeartRateL.com"
6   },
7   {
8     "type": "output",
9     "username": "jsmith",
10    "data": [
11      {
12        "_id": {
13          "ObjectID": "sfHGJa879HGis98asdfa"
14        },
15        "user_id": {
16          "ObjectID": "Js987dKJjgsd79"
17        },
18        "bpm": 67,
19        "date": "10/12/20 10:23:53"
20      },
21      {
22        "_id": {
23          "ObjectID": "jkdJKd023jksKJshwse2"
24        },
25        "user_id": {
26          "ObjectID": "Js987dKJjgsd79"
27        },
28        "bpm": 68,
29        "date": "12/1/20 12:53:13"
30      },
31      {
32        "_id": {
33          "ObjectID": "dkjJHGKs8923gJHG23g"
34        },
35        "user_id": {
36          "ObjectID": "Js987dKJjgsd79"
37        },
38        "bpm": 67,
39        "date": "10/12/20 10:23:53"
40      },
41    ]
42  }
43 }
```

Figure 3-1. Sample Output of the user's information