UMKC
School of Computing and Engineering

HUMANOID

Project plan and First Increment

Team Number: 12

Sri Sai Narayana Ram Gopal Mangena
Achyuth Reddy Nalamadgu

UMKC School of Computing and Engineering

Project Objectives

Significance/Uniqueness:

The significance of this application is that it provides real life view of human anatomy and it also features the interactions with the application. This application helps in exploring more features of human anatomy in simple way using voice commands, gestures and gazes from the user based on deep learning concepts, through which we train the model. Our main aim is to provide the real life view of human anatomy with interaction features in Microsoft HoloLens.

System Features:

- Experience the spatial view of human body.
- Users can easily understand the complex human system.
- Can access instantly, just by wearing the head mounted device.
- Users can control the application by using gestures.
- Users can interact with application by voice commands.
- Users can also see the visualization of real time user data extracted

Approach

Data Source:

3D models: We will collect the 3D models of human body from the internet. Major source of 3D models is clara.io, which provides the real-time models for free. Also by using existing objects, we can develop complex models in Unity 3D.

Real-time data: Fitbit smart watch, which provides the data of an individual for the daily activity. Various activities of individual include number of steps, heart rate, number of floors he climbed, number of calories he burned, distance he traversed.

Tools:

- Unity 3D
- Microsoft Visual Studio

Expected Inputs/Outputs:

Input: Major source of input for this application is the real-time data that includes heartbeat, location, weight, sleep data.

Expected Output: Application displays real-time view of human anatomy along with the real-time data in the form of charts and graphs.

Algorithms:

Deep Learning: We will be using deep learning to train our application to respond to voice commands by user. After training and testing with separate sets, application would be able to respond to the voice commands from the users.

Related Work

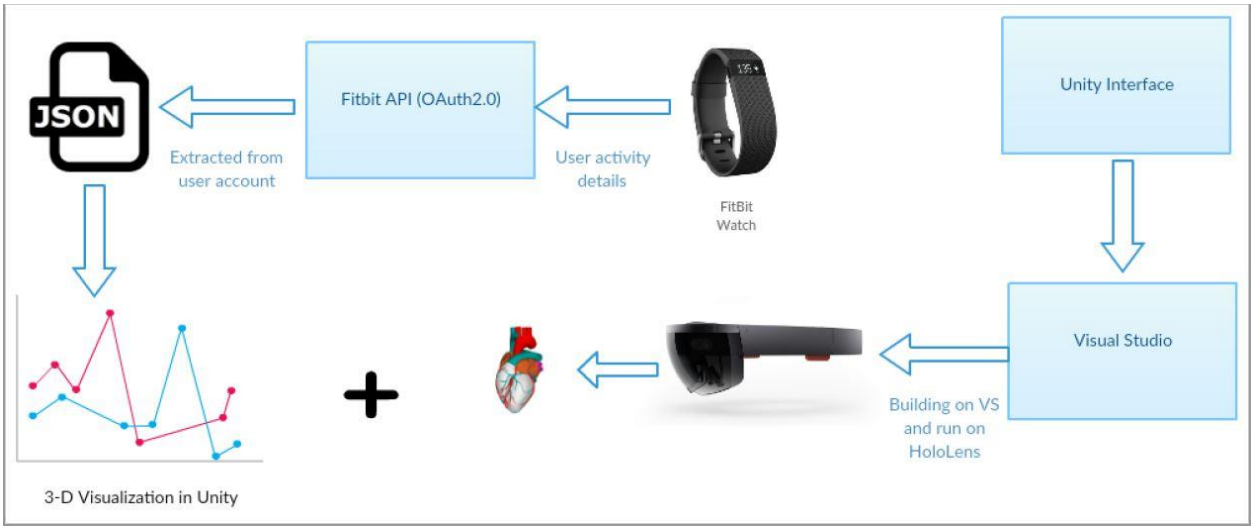
Open Source Projects:

- There are many applications in the field of medicine using Augmented Reality. Following are some of them.
    - AccuVein: Helps doctors to identify patients' veins.
    - VR Dentist: dental app for educational purposes.
    - Anatomy 4D: Visualizes detailed bone structures.
- By understanding the working of these applications, we want to develop an application that provides real time experience and interaction features to application.

Application Specification:

a. System Specification:

Architecture Diagram:



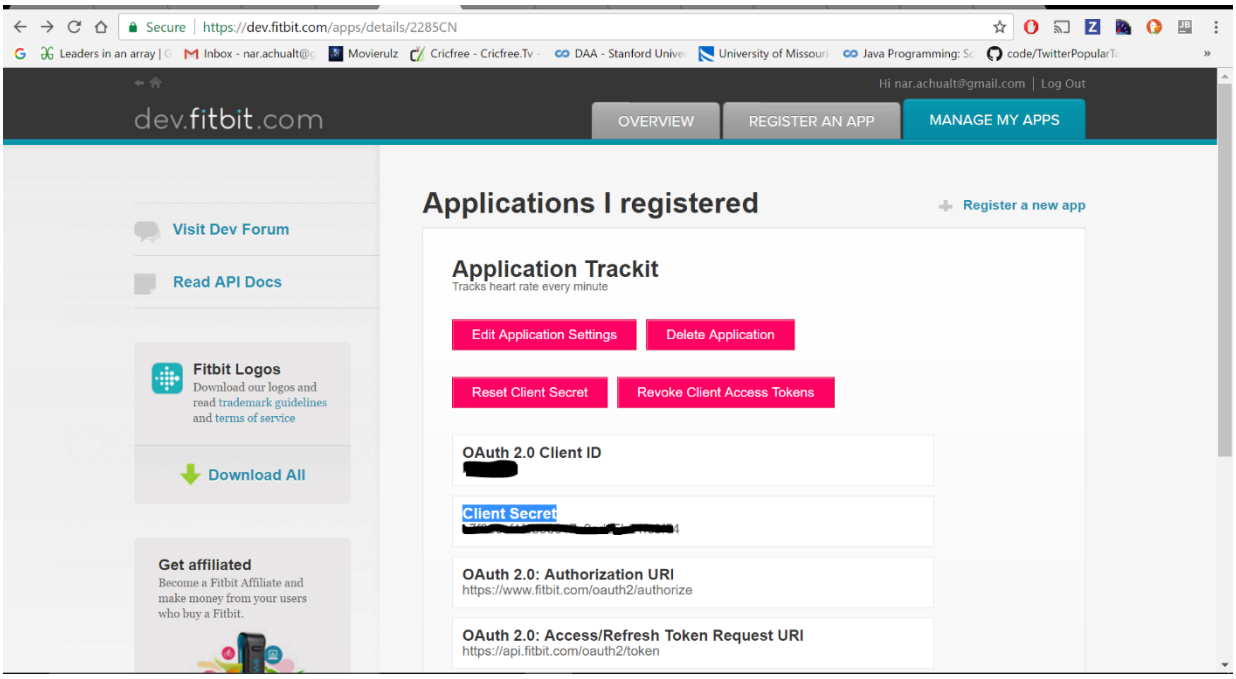
b. Existing applications/services used:

- Fitbit OAuth2.0 for authorization, to access Fitbit data.
- Fitbit API using developer account- https://dev.fitbit.com/apps/oauthinteractivetutorial
- Chrome browser for parsing the data.

Implementation

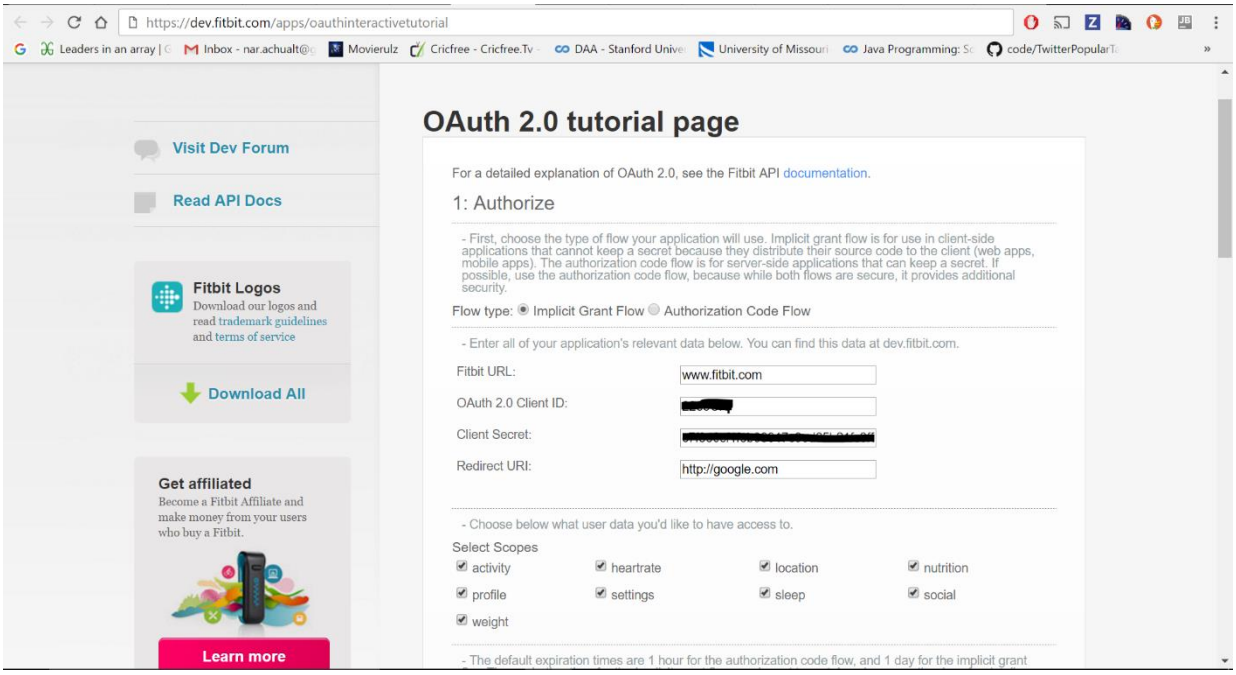
FitBit Data Extraction:

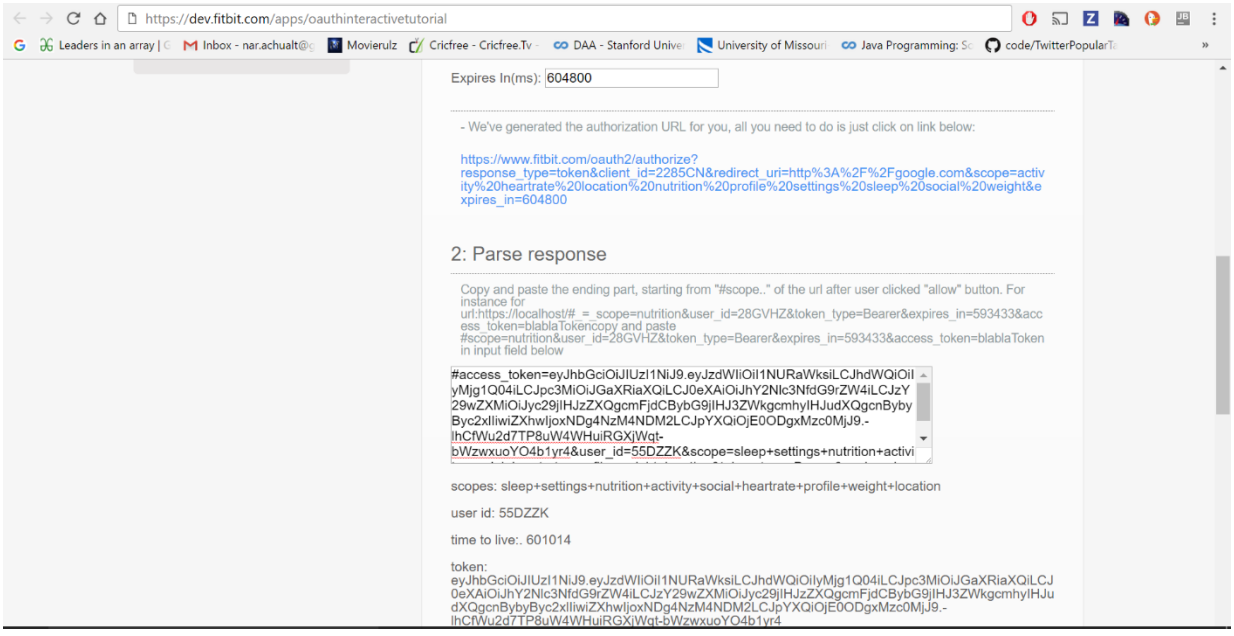
We register an application on Fitbit developer account, and provide the necessary information as required. Once we register we get the - OAuth 2.0 Client ID, Client Secret as shown below



Once we click on the OAuth 2.0 tutorial page we will see the screen below, fill up Client ID, secret key and redirected URI and select the required parametres for the data you want to extract. Here we extracted the data for heart rate.

Then we click on the url generated as shown in Img 3, then we parse the url in text box provided, next we proceed to make request through 'Send to Hurl.it' option as shown in Img 4. Then we give our crendentials and allow the application to access our account, then the JSON forrmat is displayed. Then we convert the JSON format data to CSV.


https://dev.fitbit.com/apps/oauthinteractivetutorial
Visit Dev Forum
Read API Docs
Fitbit Logos
Download our logos and read trademark guidelines and terms of service
Download All
Get affiliated
Become a Fitbit Affiliate and make money from your users who buy a Fitbit.
Learn more
OAuth 2.0 tutorial page
For a detailed explanation of OAuth 2.0, see the Fitbit API documentation.
1: Authorize
- First, choose the type of flow your application will use. Implicit grant flow is for use in client-side applications that cannot keep a secret because they distribute their source code to the client (web apps, mobile apps). The authorization code flow is for server-side applications that can keep a secret. If possible, use the authorization code flow, because while both flows are secure, it provides additional security.
Flow type: Implicit Grant Flow Authorization Code Flow
- Enter all of your application's relevant data below. You can find this data at dev.fitbit.com.
Fitbit URL: www.fitbit.com
OAuth 2.0 Client ID:
Client Secret:
Redirect URI: http://google.com
- Choose below what user data you'd like to have access to.
Select Scopes
activity heartrate location nutrition
profile settings sleep social
weight
- The default expiration times are 1 hour for the authorization code flow, and 1 day for the implicit grant


https://dev.fitbit.com/apps/oauthinteractivetutorial
Expires In(ms): 604800
- We've generated the authorization URL for you, all you need to do is just click on link below:
https://www.fitbit.com/oauth2/authorize?response_type=token&client_id=2285CN&redirect_uri=http%3A%2F%2Fgoogle.com&scope=activity%20heartrate%20location%20nutrition%20profile%20settings%20sleep%20social%20weight&expires_in=604800
2: Parse response
Copy and paste the ending part, starting from "#scope.." of the url after user clicked "allow" button. For instance for url:https://localhost/#_=_scope=nutrition&user_id=28GVHZ&token_type=Bearer&expires_in=593433&access_token=blablaTokencopy and paste #scope=nutrition&user_id=28GVHZ&token_type=Bearer&expires_in=593433&access_token=blablaToken in input field below
#access_token=eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI1NURaWksiLCJhdWQiOiIyMjg1Q04iLCJpc3MiOiJGaXRiaXQiLCJ0eXAiOiJhY2Nlc3NfdG9rZW4iLCJzY29wZXMiOiJyc29jIHJzZXQgcmFjdCBybG9jIHJ3ZWkgcmhyIHJudXQgcnBybyBvc2xliwiZXhwIjoxNDg4NzM4NDM2LCJpYXQiOjE0ODgxMzc0MjJ9.-lhCfWu2d7TP8uW4WHuiRGXjWqt-bWzwxuoYO4b1yr4&user_id=55DZZK&scope=sleep+settings+nutrition+activi
scopes: sleep+settings+nutrition+activity+social+heartrate+profile+weight+location
user id: 55DZZK
time to live:. 601014
token:
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI1NURaWksiLCJhdWQiOiIyMjg1Q04iLCJpc3MiOiJGaXRiaXQiLCJ0eXAiOiJhY2Nlc3NfdG9rZW4iLCJzY29wZXMiOiJyc29jIHJzZXQgcmFjdCBybG9jIHJ3ZWkgcmhyIHJudXQgcnBybyBvc2xliwiZXhwIjoxNDg4NzM4NDM2LCJpYXQiOjE0ODgxMzc0MjJ9.-lhCfWu2d7TP8uW4WHuiRGXjWqt-bWzwxuoYO4b1yr4

3 Make Request

Finally, when you have an access token, you can start making requests. If you had a token before, you don't need to go through steps 2-3, just paste your token below and make sure you enter your app data in step 1. We only support GET requests at the moment in this tutorial. But please feel free to check out other types of requests in the docs too on your own.

OAuth 2.0 Access Token: eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI1NURaWksiLCJhdWQiOiIyMjg1Q04iLCJpc

API endpoint URL: https://api.fitbit.com/1/user/-/activities/heart/date/today/1d/1sec/time/00:00/00:(

```
curl -i
     -H "Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI1NURaWksiLCJhdWQiOiIyMjg1Q(
     https://api.fitbit.com/1/user/-/activities/heart/date/today/1d/1sec/time/00:00/00:01.
```

Copy to clipboard Send to Hurl.it embedcurl.com by Runscope

X-Frame-Options: SAMEORIGIN

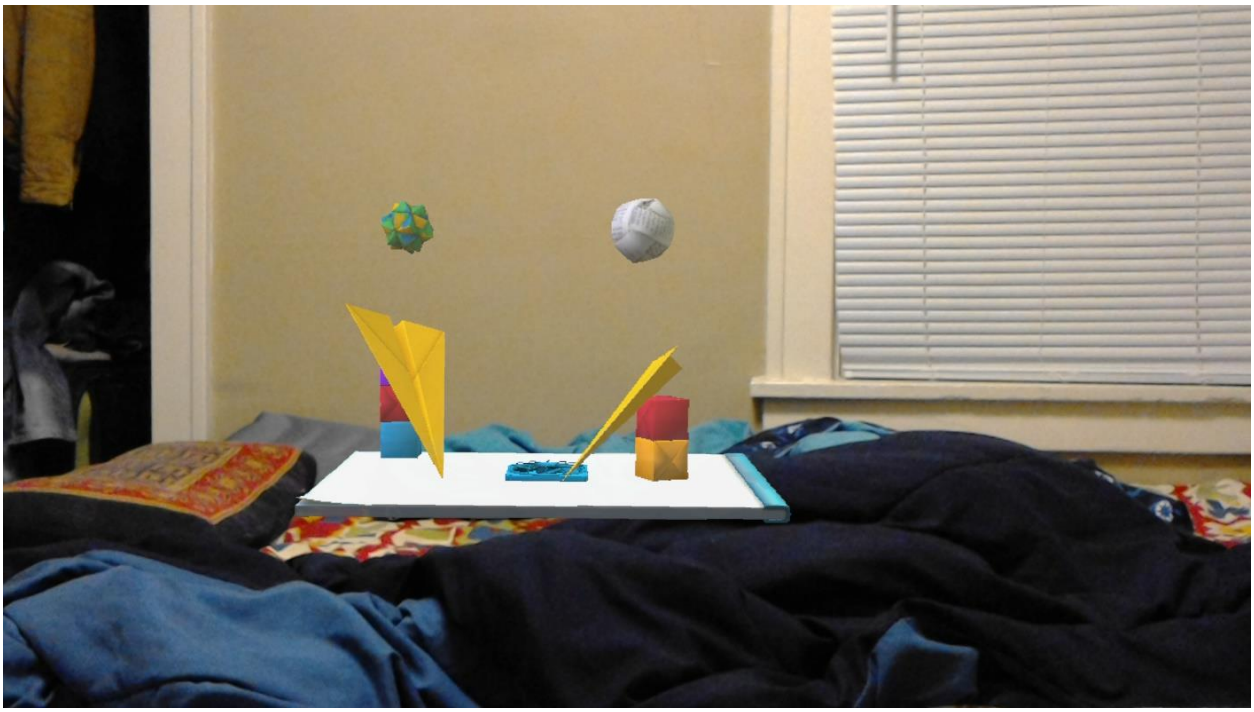
BODY view raw

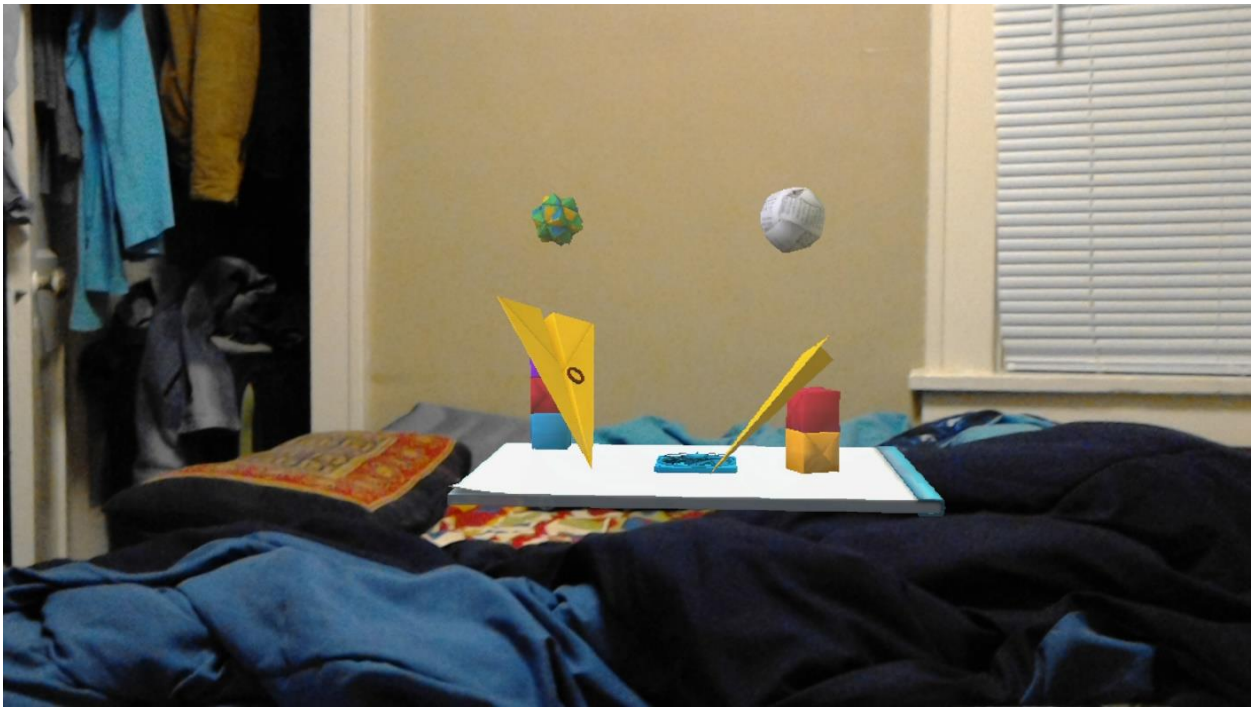

```
{
  "activities-heart": [
    {
      "customHeartRateZones": [],
      "dateTime": "today",
      "heartRateZones": [
        {
          "max": 98,
          "min": 30,
          "name": "Out of Range"
        },
        {
          "max": 137,
          "min": 98,
          "name": "Fat Burn"
        },
        {
          "max": 166,
          "min": 137,
          "name": "Cardio"
        },
        {
          "max": 220,
          "min": 166,
          "name": "Peak"
        }
```

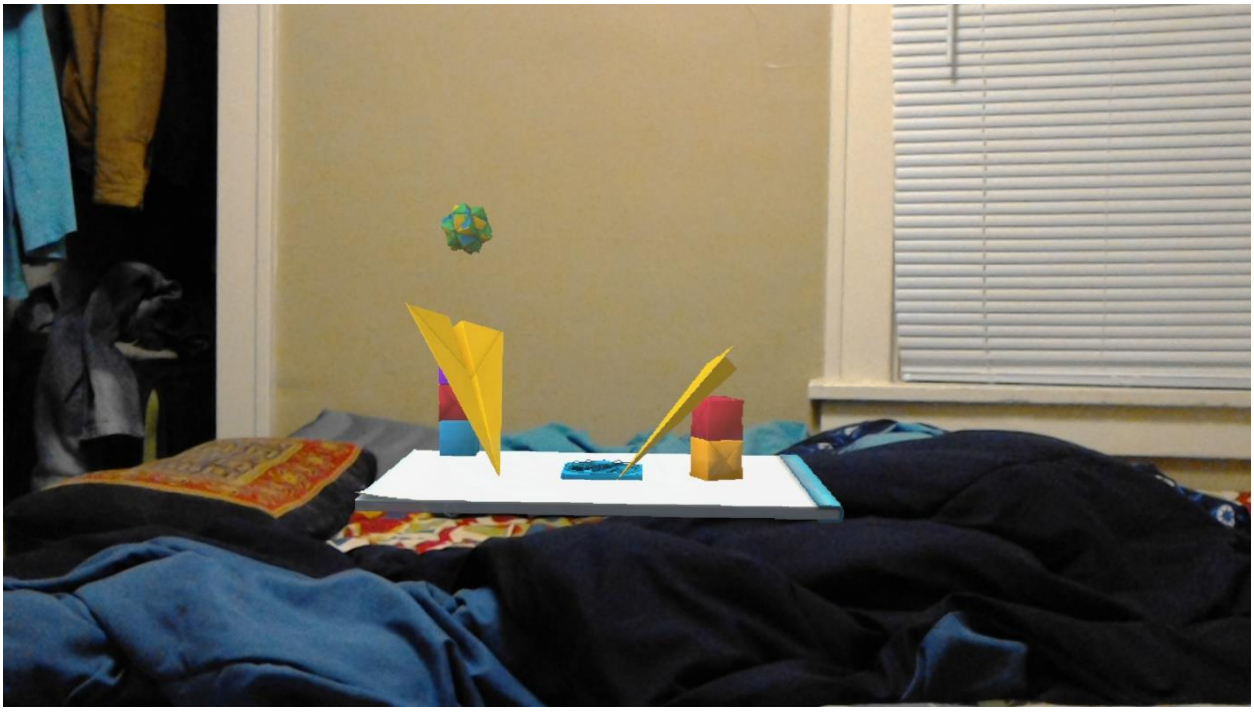

The extracted data is saved in csv file.

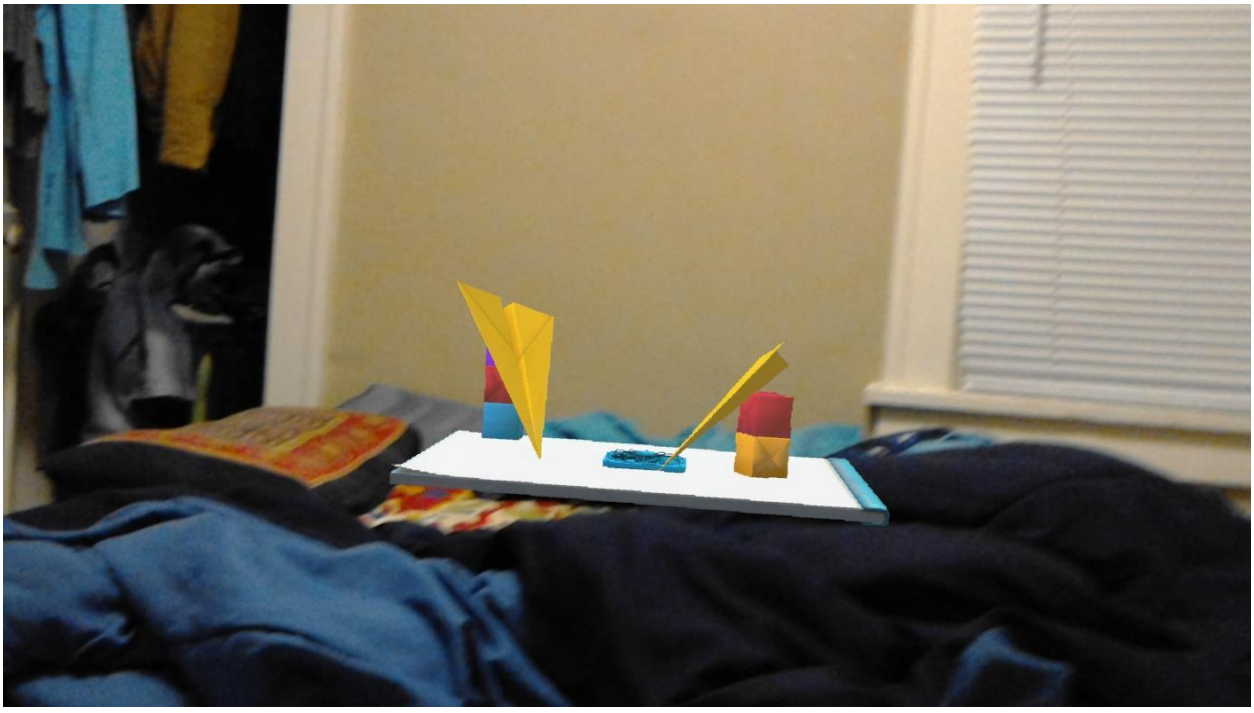
HoloLens:

We have implemented the basic origami with gaze, gestures, voice command and installed it to Microsoft HoloLens. Following screenshots corresponds to the implementation part of the application.

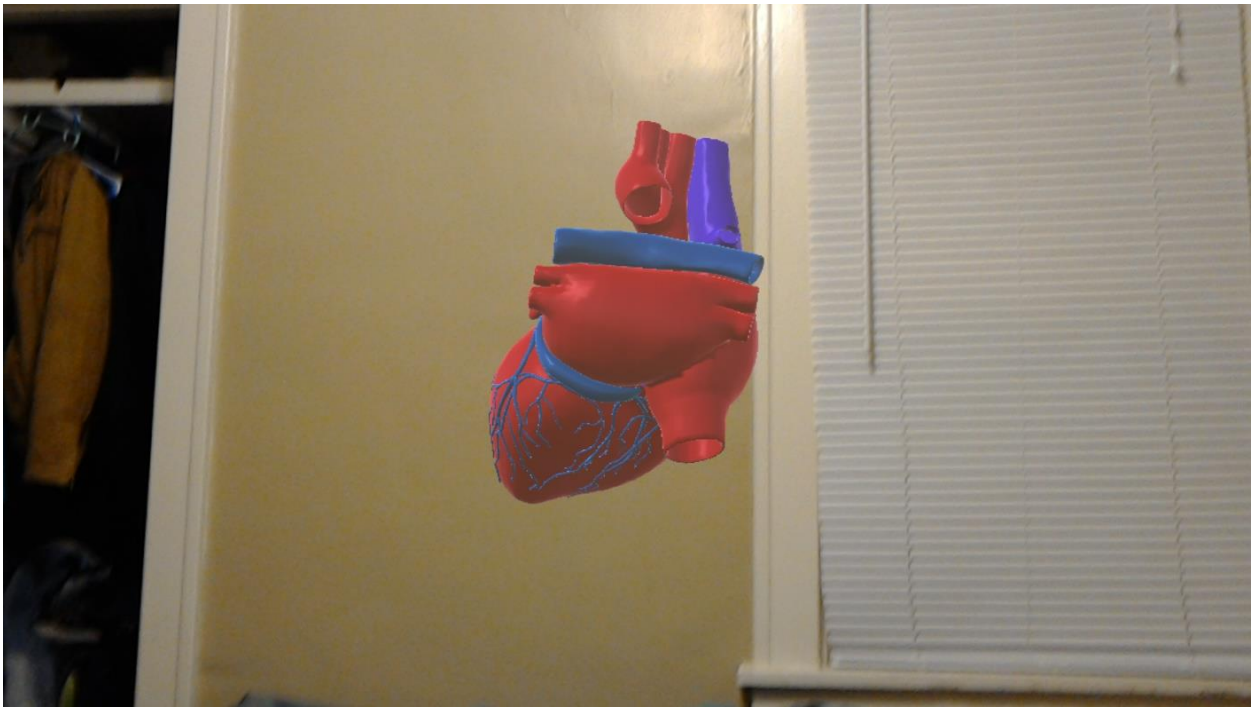








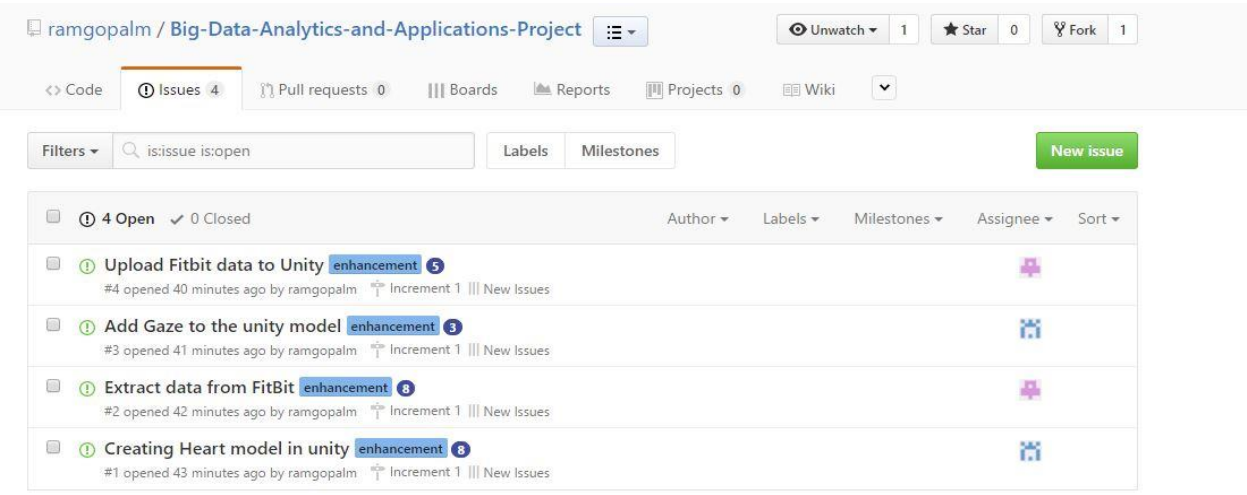
We have also installed a basic heart model to the HoloLens.

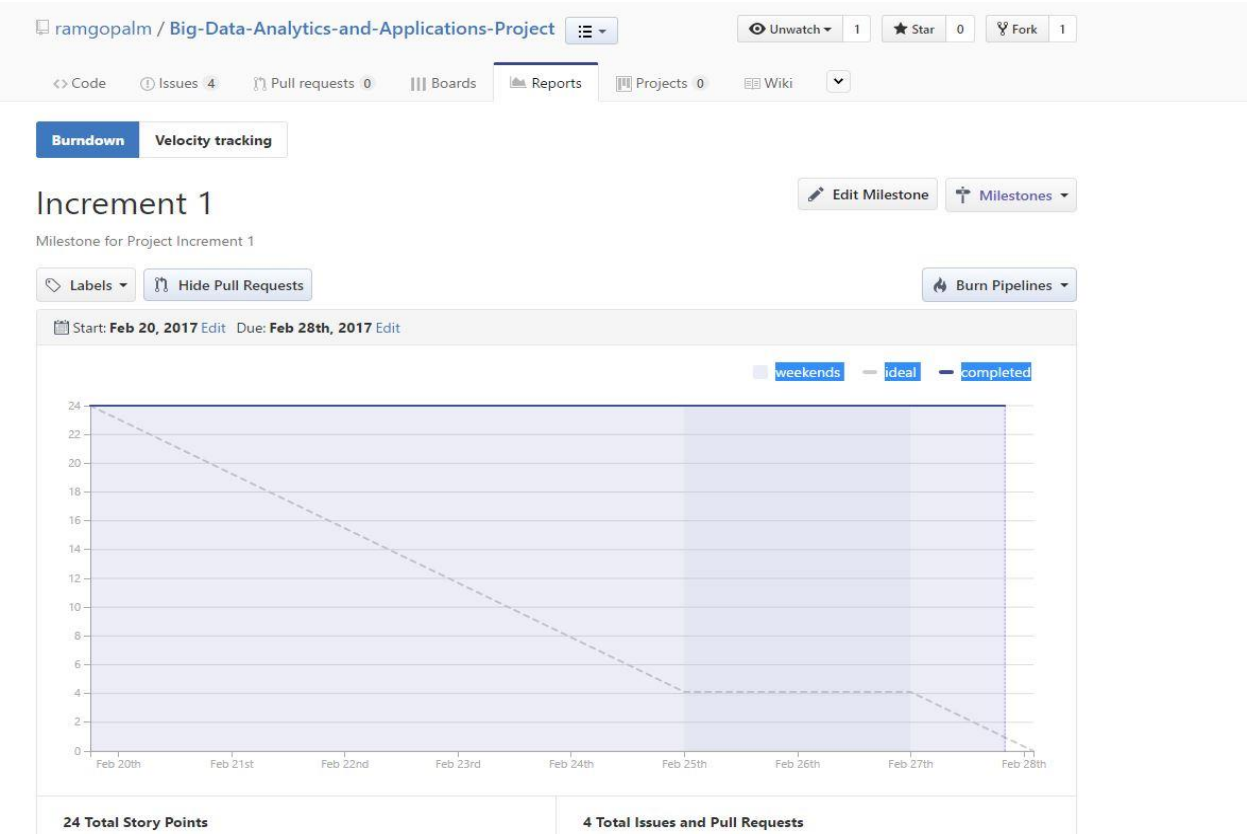


UMKC School of Computing and Engineering

Project Management

Project Timelines and Responsibilities:





Work Completed:

- Created Origami 3D model in Unity 3D
- Developed the basic Hologram model 'Origami' to the Microsoft HoloLens
- Added interactions to the Origami model by using c#.net
- Developed a basic human heart model and installed it to HoloLens
- Extracted the heart data from the Fitbit to excel sheet

Work to be completed:

- Need to develop complex human body model in unity 3D
- Develop interactions with the human model by using c#.net
- To display Fitbit data along with the Unity model in HoloLens

Issues:

- Creating human model in unity 3D.
- Adding interactions to the created model.

Contributions:

- Sri Sai Narayana Ram Gopal Mangena – 50%
- Achyuth Reddy Nalamadgu – 50%

Bibliography

- https://www.biodigital.com/developers
- https://www.eonreality.com/portfolio-items/virtual-anatomy-simulationv
- https://developer.microsoft.com/en-us/windows/holographic/holograms_101
- Github.com
- http://www.medpagetoday.com/practicemanagement/informationtechnology/59072
- http://anatomy4d.daqri.com/
- http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0091276
- http://www.acculvein.com/home/
- https://developer.microsoft.com/en-us/windows/holographic/documentation