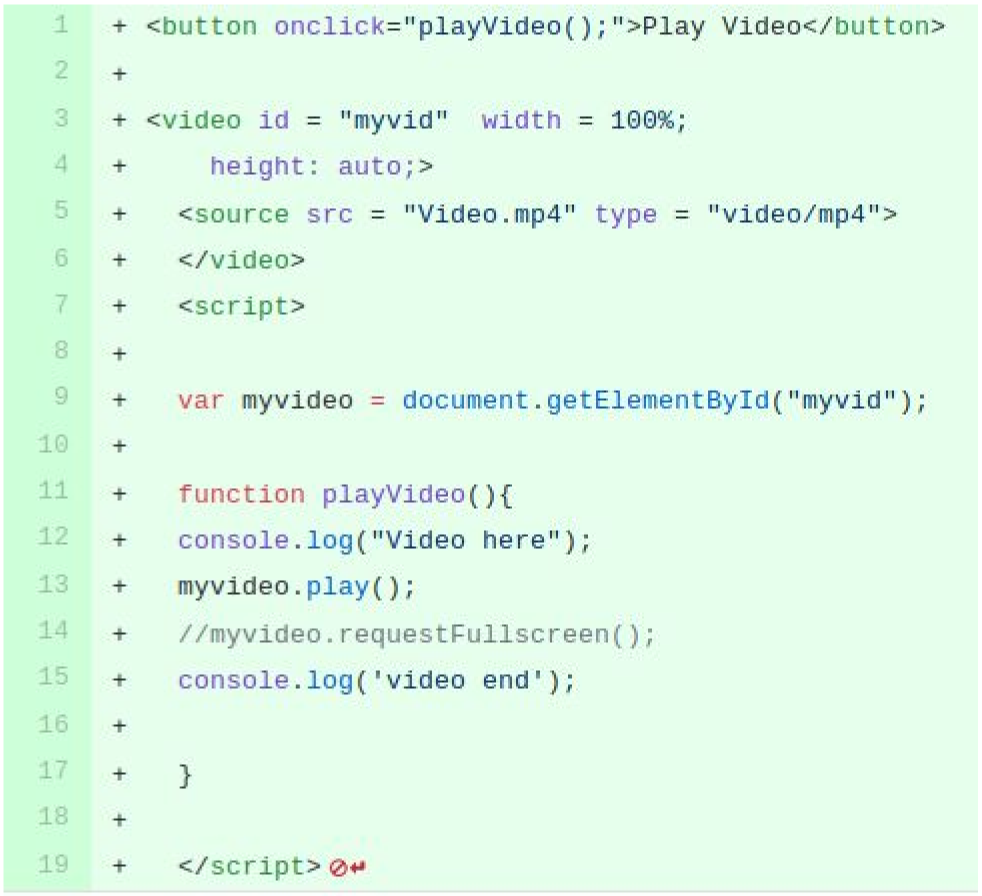
1.)

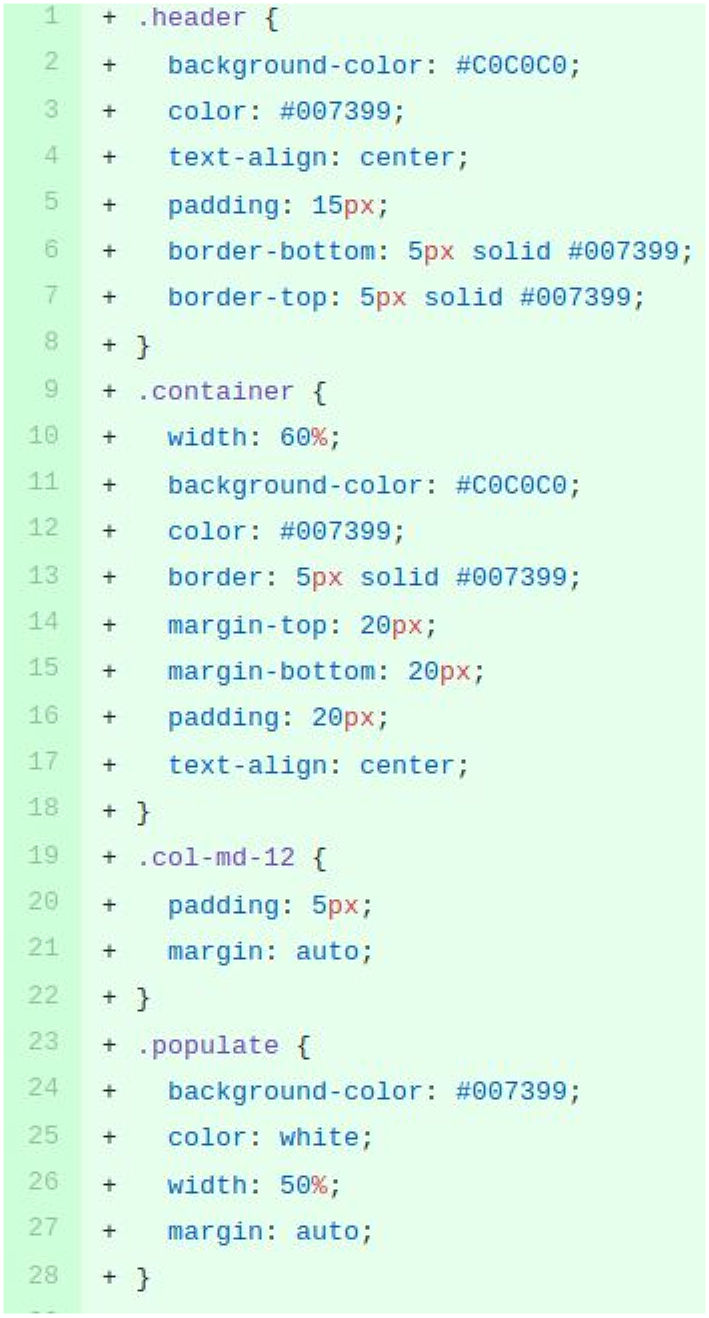
We implemented all the key features, which are the eye movement test, visual acuity, and pulmonary movement. This was done by connecting webgazer to a button action which are configured to each different test type. Applied an extensive 2 minute video for eye movement when the eye movement button is pressed where the user would focus on a multicolored moving ball across the screen. Updated the patient log in, and added a social login for the chat system for the doctor. Made it so we redirected to a new page when administering the eye visión tests which would compartmentalize our code as well as the website. Our original plan was to include a messaging component as well as allowing patients to connect with their doctors. We implemented the social media aspect of this mobile application by using html, php, apache web server. We created a database that is being hosted in a raspberry pi.

2.)



```
+ <button onclick="playVideo();">Play Video</button>
+
+ <video id = "myvid"  width = 100%;
+     height: auto;>
+   <source src = "Video.mp4" type = "video/mp4">
+   </video>
+   <script>
+
+   var myvideo = document.getElementById("myvid");
+
+   function playVideo(){
+   console.log("Video here");
+   myvideo.play();
+   //myvideo.requestFullscreen();
+   console.log('video end');
+
+   }
+
+   </script>
```

This will give an onclick to play the video for the eye movement test and also will allow webgazer to open while the video is being played. This also made the video element and allowed it so that the video width was the correct side of the screen so that it can be formatted for all different sized screens.



```
+ .header {
+   background-color: #C0C0C0;
+   color: #007399;
+   text-align: center;
+   padding: 15px;
+   border-bottom: 5px solid #007399;
+   border-top: 5px solid #007399;
+ }
+ .container {
+   width: 60%;
+   background-color: #C0C0C0;
+   color: #007399;
+   border: 5px solid #007399;
+   margin-top: 20px;
+   margin-bottom: 20px;
+   padding: 20px;
+   text-align: center;
+ }
+ .col-md-12 {
+   padding: 5px;
+   margin: auto;
+ }
+ .populate {
+   background-color: #007399;
+   color: white;
+   width: 50%;
+   margin: auto;
+ }
```

This is the css code added to reformat the patient page in order to make it look more presentable and easy to administer the test.

From the first revision, we completely changed the format of the application. We transitioned from a android/IOS native app to a web app. Our whole application was revised and the new iteration shows a dramatic change in the application architecture. The change was to accommodate the webgazer api which had compatibility issues with the ionic framework we originally had planned to use. Also changing how the exam is going to be administered. Instead of having characters played in the background as a video, we will only have one exam have a simple video, and the others will have the basic test that where always given, but through the web instead of a doctor's office. This will still eliminate the need to go to the clinic for these tests, while still being able to modernize how the exams are administered.

For our first iteration we defined functions for a messaging component. As the following pice of code shows, this was implemented using typescript. This function would generate a message to the front end. There was also a tag that included the time. The idea was to use this function to connect to a chat service offered by firebase.

```
sendMessage(){
   this.messages.push({
     user: 'Doctor',
     createdAt: new Date().getTime(),
     msg: this.newMsg
   });
   this.newMsg = '';
   setTimeout(()=>{
     this.content.scrollToBottom(200);
   })
 }
```

For our second iteration we implemented the messaging component and added an extra feature called “Note To Self” .
From the first iteration our plan was to create charts that would indicate the progress of the user, however due to time constraint and missing dependencies this was dropped and replaced.

```
 this.doughnutChart = new Chart(this.doughnutCanvas.nativeElement, {
     type: "doughnut",
     data: {
       labels: ["Visual Acuity", "Eye Movement", "Pupiliary
Reflex"],
       datasets: [
         {
           label: "# of Votes",
           data: [100, 100, 100],
           backgroundColor: [
             "rgba(255, 99, 132, 0.2)",
             "rgba(54, 162, 235, 0.2)",
             "rgba(255, 206, 86, 0.2)",
             "rgba(75, 192, 192, 0.2)",
             "rgba(153, 102, 255, 0.2)",
             "rgba(255, 159, 64, 0.2)"
           ],
           hoverBackgroundColor: ["#FF6384", "#36A2EB", "#FFCE56",
"#FF6384", "#36A2EB", "#FFCE56"]
         }
       ]
     }
   });
 }
```

The graphs and statistics graphics were implemented for the doctor. Similarly the chart.js module was used to generate tables. The tables are used for the statistics of the patient, once the patient has completed an examination the charts are generated.

```
<li class="nav-item">
       <a class="nav-link" href="charts.html">
         <i class="fas fa-fw fa-chart-area"></i>
         <span>Charts</span></a>
     </li>
     <li class="nav-item">
       <a class="nav-link" href="tables.html">
         <i class="fas fa-fw fa-table"></i>
         <span>Patients</span></a>
     </li>
   </ul>

   <div id="content-wrapper">
```

The following is the creation of a database in mysql.
Another feature that was implemented was the user profile.

For iteation 1 we had created dummy data, this data was passed to the front end by using data binding .

```
messages = [
  {
    user: 'Doctor',
    createdAt: 15554090856000,
    msg: ' Hello, dear patient?'
  },
  {
    user: 'Patient',
    createdAt: 155540900956000,
    msg: ' I dont feel so good?'
  },
  {
    user: 'Doctor',
    createdAt: 15540910560000,
    msg: ' Make sure to complete exams?'
  }
]
```

We created databases using php.

```
<?php
   require_once 'functions.php';
   createTable('patientss',
             'user VARCHAR(16),
             pass VARCHAR(16),
             INDEX(user(6))');
   createTable('messages',
             'id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
             auth VARCHAR(16),
             recip VARCHAR(16),
             pm CHAR(1),
             time INT UNSIGNED,
             message VARCHAR(4096),
             INDEX(auth(6)),
             INDEX(recip(6))');
   createTable('doctors',
             'user VARCHAR(16),
             doctor VARCHAR(16),
             INDEX(user(6)),
             INDEX(doctor(6))');
    createTable('profiles',
             'user VARCHAR(16),
             text VARCHAR(4096),
             INDEX(user(6))');
?>
```

Originally there was no method to check whether the user entered the correct data, that is the user was able to randomly use a username and password to log in. This can be a security issue since we are using mysql. We made sure the data passed was correct using the following function. The user was also given feedback when choosing a username so that no two users have the same identification.

```
 if (isset($_POST['user']))
 {
   $user   = sanitizeString($_POST['user']);
   $result = queryMysql("SELECT * FROM members WHERE user='$user'");
```

The first iteration created a setting page but did not implement the functionality for updating a profile. The following code allows user to add a description and use a photo for identification.

```
<ion-tab-button tab="notifications">
         <ion-icon name="person"></ion-icon>
         <ion-label>Account</ion-label>
       </ion-tab-button>
```

We created the ability for the user to upload images and

```
 $saveto = "$user.jpg";
   move_uploaded_file($_FILES['image']['tmp_name'], $saveto);
   $typeok = TRUE;

   switch($_FILES['image']['type'])
   {
```

```php
    case "image/gif":   $src = imagecreatefromgif($saveto); break;
    case "image/jpeg":  // Both regular and progressive jpegs
    case "image/pjpeg": $src = imagecreatefromjpeg($saveto); break;
    case "image/png":   $src = imagecreatefrompng($saveto); break;
    default:            $typeok = FALSE; break;
  }

  if ($typeok)
  {
    list($w, $h) = getimagesize($saveto);

    $max = 100;
    $tw  = $w;
    $th  = $h;

    if ($w > $h && $max < $w)
    {
      $th = $max / $w * $h;
      $tw = $max;
    }
    elseif ($h > $w && $max < $h)
    {
      $tw = $max / $h * $w;
      $th = $max;
    }
    elseif ($max < $w)
    {
      $tw = $th = $max;
    }

    $tmp = imagecreatetruecolor($tw, $th);
    imagecopyresampled($tmp, $src, 0, 0, 0, 0, $tw, $th, $w, $h);
    imageconvolution($tmp, array(array(-1, -1, -1),
      array(-1, 16, -1), array(-1, -1, -1)), 8, 0);
    imagejpeg($tmp, $saveto);
    imagedestroy($tmp);
    imagedestroy($src);
```

```
   }



 for ($j = 0 ; $j < $num ; ++$j)
 {
   $row            = $result->fetch_array(MYSQLI_ASSOC);
   $followers[$j] = $row['doctor'];
 }

 $result = queryMysql("SELECT * FROM doctors WHERE
doctors='$view'");
 $num    = $result->num_rows;

 for ($j = 0 ; $j < $num ; ++$j)
 {
     $row            = $result->fetch_array(MYSQLI_ASSOC);
     $following[$j] = $row['user'];
 }
```

We created a social loging so that users can connect with doctors, originally patient could message doctors in the first iteration, however this did not take into account that a patient could have multiple doctors, therefore we solved this problem by implementing a social aspect.

```
$mutual    = array_intersect($followers, $following);
 $followers = array_diff($followers, $mutual);
 $following = array_diff($following, $mutual);
 $doctors    = FALSE;
```

In the first iteration there was no security check of what type of data was typed. We implemented the following as a safeguard.

```
function sanitizeString($var)
 {
   global $connection;
   $var = strip_tags($var);
   $var = htmlentities($var);
   if (get_magic_quotes_gpc())
     $var = stripslashes($var);
   return $connection->real_escape_string($var);
 }
```

In the first iteration our login was not persistent, therefore users could not be identified we solved this by creating a database using mysql to store the user name and their data. We implemented mysql by creating a database named valleychildrens and a user with privilges.

```
 $dbhost = 'localhost';
 $dbname = 'valleychildrens';
 $dbuser = 'vch';
 $dbpass = 'cse120';
```

Originally creating a new user was not implemented in the first iteration, so we used mysql to create a table that would store new users and created a signup page.

```
$error = $user = $pass = "";
 if (isset($_SESSION['user'])) destroySession();

 if (isset($_POST['user']))
 {
   $user = sanitizeString($_POST['user']);
   $pass = sanitizeString($_POST['pass']);

   if ($user == "" || $pass == "")
     $error = 'Not all fields were entered<br><br>';
   else
   {
     $result = queryMysql("SELECT * FROM members WHERE
user='$user'");

     if ($result->num_rows)
       $error = 'That username already exists<br><br>';
     else
     {
       queryMysql("INSERT INTO members VALUES('$user', '$pass')");
       die('<h4>Account created</h4>Please Log
in.</div></body></html>');
     }
   }
}
```