


NEW YORK INSTITUTE OF TECHNOLOGY

Reinvent the Future.

New York Institute of Technology

Department of Computer Science

CSCI 426 - Project

Fall 2020

Prof. Houwei Cao

Project Name: Heroes Assemble

Name: Nuran Ghoneim, Nadia Hossain, Riddhi Makwana, and Shipra Priyadarshini

Email: nghoneim@nyit.edu, nhossa02@nyit.edu, rmakwana@nyit.edu, and spriyada@nyit.edu

Table of Contents

Abstract	3
<hr/>	
1.0 Project Background	3
1.1 Problem we are addressing	
<hr/>	
1.2 Motivation	
<hr/>	
2.0 Related Work	4
2.1 Existing Applications	
<hr/>	
2.2 Review of Related Work	
<hr/>	
2.3 How it relates to our project	
<hr/>	
3.0 Data Description	5
3.1 Data used	
<hr/>	
3.1 Method used to collect the data	
<hr/>	
3.2 Issues encountered during data collection	
<hr/>	
3.3 Interesting examples from the data	
<hr/>	
4.0 Approach	8
4.1 Detailed description of the approach	
<hr/>	
5.0 Experiments and Results	9
5.1 Evaluation of methodology	
<hr/>	
5.2 Experiments & Challenges	
<hr/>	
5.3 Presentation of Results	
<hr/>	
5.4 Discussion of Results	
<hr/>	
6.0 Conclusions	16
6.1 Main contributions of the project	
<hr/>	
6.2 What worked and what did not	
<hr/>	
6.3 Considerations for future work	
<hr/>	
7.0 Individual Contribution of Each Team Member and Learning Experience	17

7.1 Nadia Hossain	
7.2 Nuran Ghoneim	
7.3 Riddhi Makwana	
7.4 Shipra Priyadarshini	

8.0 Reference	18
----------------------	-----------

Abstract

This report explains the process of creating a website for superheroes. The website consists of a search page, profile, and quiz. The quiz allows users to find out which superhero they might be. The report will explain the motive, difficulties, data, and code of the project.

1.0 Project Background

Problem & Motivation

There is a massive fanbase dedicated to superheroes. Some people only like comics by certain publishers while others love all heroes from all universes. While there are websites dedicated to each universe and have all the comics, movies, and television shows relating to their characters, there isn't really a platform where you can find any hero from any universe. Take DC and Marvel, for instance, they each have their own sites and all their characters listed with their backstories, abilities, and more. However, everything is separated due to them essentially being owned and run by different publishers, and the fact that they are from different worlds.

Since there are so many fans out there who love superheroes— us included — we thought it would be a great idea to gather all these heroes onto one platform. It's a great way for people who do not know as much about this topic to get an introduction through one source, rather than visiting multiple websites. For example, if someone is trying to learn about superheroes, it could be very overwhelming due to the amount of information there is out there. Having all the characters on one platform will make the process a lot easier and enjoyable. Our platform also provides the opportunity for people to learn about heroes they might not have heard of. Not all brands of comics have websites and not all of them are very well known. So this increases the exposure of these lesser-known heroes in a very simple and accessible way.

2.0 Related Work

Existing Applications & Review

Some applications that currently exist are DC Comics and Marvel. They each have their own websites where they have all their characters, movies, shows, merchandise, and more. While these both have amazing characters, they are owned by different organizations— DC Comics is owned by Warner Bros while Marvel is owned by Disney. Another existing application is called Superhero Database. This website has all superheroes listed out along with their biographies, and what movies some of these characters have been in. However, a lot of their sections such as history, power, and equipment are empty. So it does not hold much information on the characters, and because it is more of a statistical website, it does not have the pictures of each character. Comic Vine is another existing application that is created by a fan. It has the creator's favorite characters on the home page, and a section with all the characters combined. However, this website primarily has DC and Marvel characters, and we want to include all the superheroes. So while there are a few websites out there, there isn't a website that combines all the characters together into one platform regardless of the publisher and includes these characters' histories. While there are comics that do not have their own websites, there are a few that do, such as Dark Horse Comics. This is one example of comics and heroes that might not be as well known but are definitely out there.

How They Relate

As mentioned above, these websites are specifically directed towards superhero fanatics. Just like our website, they have the characters, their histories, and more. While our platform is very simplistic in its structure, it does have the main features that these applications do. Having a list of all the characters and their backgrounds is the main and most important feature on all these platforms. Of course, they have added features depending on how much they have to offer. For instance, on the DC and Marvel websites, they have shows and movie sections, compared to Dark Horse where they have just comic books. So while all these applications have everything users would need and more, they are all separated by publisher. That is where we come in, combining all the characters onto one platform.

3.0 Data Description

Datasets

We have used 6 tables to insert superhero information. The tables used are:

Appearances: Which has general superhero characteristics like gender, height, weight, etc.

Biography: Which has detailed information on superhero like place of birth, full name, aliases etc.

Connection: which is mainly the groups to which the superhero is connected.

Power stats: which has power level of the superhero on a total of 100 like intelligence, power, speed etc.

Work: which has occupation level details on the superheroes.

Lastly, the superhero table which contains images in form of URL for each.

All these superhero tables are connected using superheroID as the primary key. Then we created a new table called superhero_master, which consists of all the superhero data. Creating the new table was a simpler solution to display all the information instead of calling each table separately. We used a public API URL called superheroAPI.com to retrieve the superhero information. In order to use this URL we needed to create an access token which was generated using FB login. And the response received was in json format.

Examples

So here we have the dataset example for Iron Man. As you can see it is in json format. Which has the iron man information required to insert the data.


```

{
  "response": "success",
  "id": "732",
  "name": "Ironman",
  "powerstats": {
    "intelligence": "100",
    "strength": "85",
    "speed": "58",
    "durability": "85",
    "power": "100",
    "combat": "64"
  },
  "biography": {
    "full-name": "Tony Stark",
    "alter-egos": "No alter egos found.",
    "aliases": [
      "Iron Knight",
      "Hogan Potts",
      "Spare Parts Man",
      "Cobalt Man II",
      "Crimson Dynamo",
      "Ironman"
    ],
    "place-of-birth": "Long Island, New York",
    "first-appearance": "Tales of Suspense #39 (March, 1963)",
    "publisher": "Marvel Comics",
    "alignment": "good"
  },
  "appearance": {
    "gender": "Male",
    "race": "Human",
    "height": [
      "6'6",
      "198 cm"
    ],
    "weight": [
      "425 lb",
      "191 kg"
    ],
    "eye-color": "Blue",
    "hair-color": "Black"
  },
  "work": {
    "occupation": "Inventor, Industrialist; former United States Secretary of Defense",
    "base": "Seattle, Washington"
  },
  "connections": {
    "group-affiliation": "Avengers, Illuminati, Stark Resilient; formerly S.H.I.E.L.D., leader of Stark Enterprises, the Pro-Registration Superhero Unit, New Avengers, Mighty Avengers, Hellfire Club, Force Works, Avengers West Coast, United States Department of Defense.",
    "relatives": "Howard Anthony Stark (father, deceased), Maria Stark (mother, deceased), Morgan Stark (cousin), Isaac Stark (ancestor)"
  },
  "image": {
    "url": "https://www.superherodb.com/pictures2/portraits/10/100/85.jpg"
  }
}

```

Displaying Iron Man in the Website:

[Home](#)
[Find A Hero](#)
[Which hero are you?](#)



Iron Man

Alter Ego: No alter egos found.

Place of Birth: Long Island, New York

First Appearance: Tales of Suspense #39 (March, 1963)

Publisher: Marvel Comics

Gender: Male

Race: Human

Height: 6'6

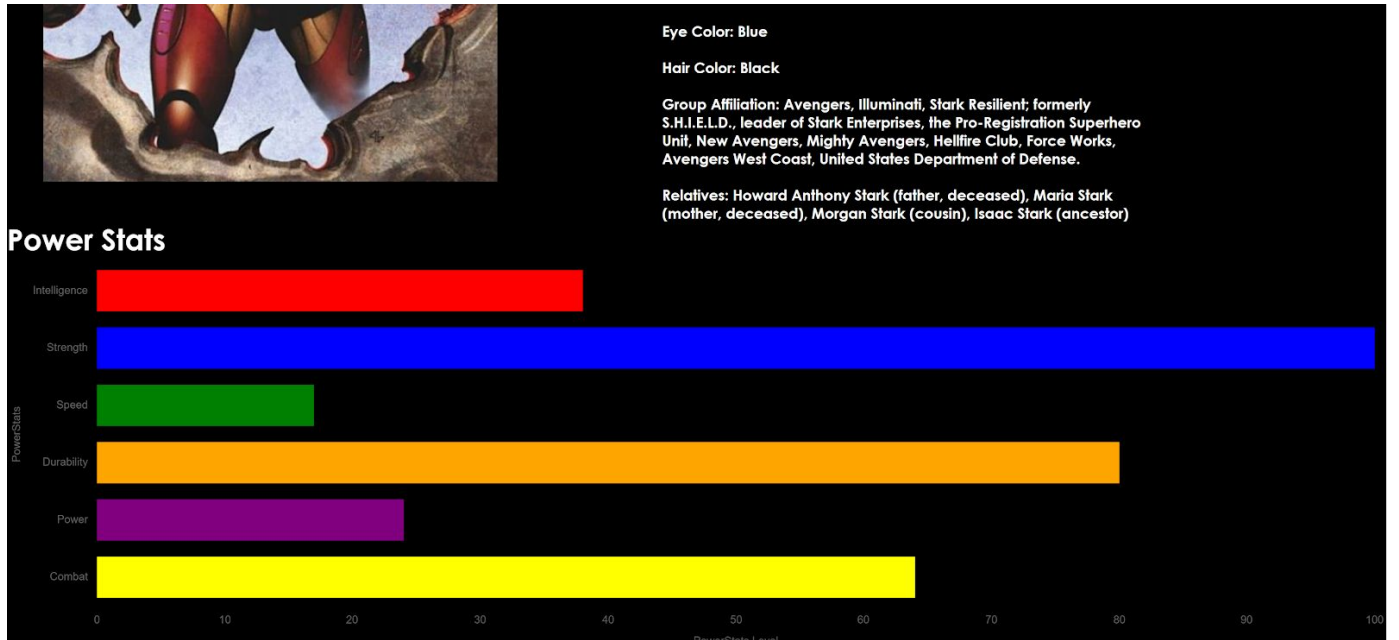
Weight: 425 lb

Eye Color: Blue

Hair Color: Black

Group Affiliation: Avengers, Illuminati, Stark Resilient; formerly S.H.I.E.L.D., leader of Stark Enterprises, the Pro-Registration Superhero Unit, New Avengers, Mighty Avengers, Hellfire Club, Force Works, Avengers West Coast, United States Department of Defense.

Relatives: Howard Anthony Stark (father, deceased), Maria Stark



The name of the hero is displayed in the center, the image of the superhero is displayed to the top left, characteristics are displayed on the right and the powers are displayed on the bottom.

Display of the database:

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> appearance		732	InnoDB	utf8mb4_0900_ai_ci	80.0 KiB	-
<input type="checkbox"/> biography		732	InnoDB	utf8mb4_0900_ai_ci	176.0 KiB	-
<input type="checkbox"/> connections		732	InnoDB	utf8mb4_0900_ai_ci	192.0 KiB	-
<input type="checkbox"/> powerstats		732	InnoDB	utf8mb4_0900_ai_ci	64.0 KiB	-
<input type="checkbox"/> superhero		732	InnoDB	utf8mb4_0900_ai_ci	144.0 KiB	-
<input type="checkbox"/> superhero_info		732	InnoDB	utf8mb4_0900_ai_ci	208.0 KiB	-
<input type="checkbox"/> superhero_master		732	InnoDB	utf8mb4_0900_ai_ci	352.0 KiB	-
<input type="checkbox"/> superhero_v		~0	View	---	-	-
<input type="checkbox"/> work		732	InnoDB	utf8mb4_0900_ai_ci	96.0 KiB	-
9 tables	Sum	~5,856	InnoDB	utf8mb4_0900_ai_ci	1.3 MiB	0 B

☐ Check all With selected:

Data dictionary

Create table

superhero_info is combined data from the table biography and superhero

4.0 Approach

Our project had two concepts, one was the personality quiz and the other was the section where users could view all the superheroes and all the information about them. First, we needed to find a superhero database from where we could extract data from. We were able to find a website and retrieve the API for us to use. Then we worked to prepare the data to be implemented into the website later on. For the website, we wanted to focus on a couple of things: one being that the user could type in a name of a character and it would search for all the characters that go by that name and display them, the second part we wanted to focus on is having a page with all the superheroes listed out with the pictures and names. When clicked on, the user would be redirected to an informative page regarding that specific character. We wanted to include a personality questionnaire, to make our website more inviting and fun. Users would just need to answer a couple of questions and then find out which superhero they are. Some technologies used were J2EE, HTML, CSS, and jQuery for the frontend and Java and MySQL for the backend. To connect the front end and back end we used the software XAMPP which provided Apache, MySQL and Tomcat. Additionally, some libraries that we used were Java Restful Web Services, MySQL Connectors, jQuery Connectivity, and Charts.

5.0 Experiments and Results

Methodology

When it came to the quiz, we ran multiple tests to ensure that users would get the answers we intended on giving them. We picked a range of answers and did multiple tests to display our findings. We had to test out a couple of different approaches to build the quiz before landing on the most realistic method. We essentially created a point-based algorithm, so depending on the user's answers that is how they were assigned which superhero they were. In terms of displaying the superheroes, we wanted to make an UI where users would find all information and photos of the superheroes all on one page. Other alternative methods would have been too complicated to navigate, hence why we made everything on one page.

Creating the table.superhero_master: which had all the data

```
select
superhero_info.SuperHeroId ,
superhero_info.FullName ,
superhero_info.AlterEgos ,
superhero_info.Aliases ,
superhero_info.PlaceOfBirth ,
superhero_info.FirstAppearance ,
superhero_info.Publisher ,
superhero_info.Alignment ,
superhero_info.name ,
superhero_info.Image ,
appearance.SuperHeroId ,
appearance.Gender ,
appearance.Race ,
appearance.Height ,
appearance.Weight ,
appearance.EyeColor ,
appearance.HairColor ,
connections.SuperHeroId ,
connections.GroupAffiliation ,
connections.Relatives
FROM connections
INNER JOIN superhero_info ON superhero_info.SuperHeroId = connections.SuperHeroId
INNER JOIN appearance ON appearance.SuperHeroId = connections.SuperHeroId
ORDER BY appearance.SuperHeroId
```

Getting each variable from the tables and combining them into one master table

Search page:

```
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  <title>FindAHero</title>
</head>

<body>
<header>
  <div class = "main">
    <ul>
      <li> <a href="index.html">Home</a></li>
      <li class= "active"> <a href="searchhero.jsp"> Find A Hero </a></li>
      <li> <a href="superHeroQuiz2.jsp"> Which hero are you? </a></li>
    </ul>
  </div>
  <div class ="title">
    <h1> Find a Hero</h1>
  </div>
  <form class="example" action="superhero_test2.jsp">
    <input type="text" placeholder="Search..." name="search" color=black>
    <button onclick="myFunction()">Click Me</button>
  </form>
</header>
</body>

<script>
function myFunction() {
  var x = document.getElementById("myDIV");
  if (x.style.display === "none") {
    x.style.display = "block";
  } else {
    x.style.display = "none";
  }
}
</script>
```

The search bar is created with HTML and Javascript. In the head tags the webpage is being titled, displayed and styled with HTML commands. In the header there is a main class which is creating a button on the top right corner to make navigation easier. There is another class called title that labels the page and finally a class called example that defines the search bar. All of that is then followed by a script that connects to the search bar to help retrieve the data. A concept of parameters is used to connect both the pages that are the search page and display hero page. The name of the particular superhero is used as a parameter which is then used in a SQL query to get the ID of the superhero from the superhero table and this ID is then used to query the remaining tables in the database to display the superhero characteristics.

Selected Hero:

```

28
29 <div class="display">
30   <%
31     try {
32       Class.forName("com.mysql.jdbc.Driver");
33       Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/superhero", "root", "123456");
34       Statement st = connection.createStatement();
35       ResultSet rs;
36       String superHeroId = request.getParameter("sHeroId");
37       rs = st.executeQuery("select * from superhero.superhero_master where superHeroId = "+superHeroId+"");
38       while (rs.next()) {
39         String name = rs.getString("Name");
40         String alterEgos = rs.getString("AlterEgos");
41         String placeOfBirth = rs.getString("PlaceOfBirth");
42         String firstAppearance = rs.getString("FirstAppearance");
43         String publisher = rs.getString("Publisher");
44         String aliases = rs.getString("Aliases");
45         String imageUrl = rs.getString("Image");
46         String userid=rs.getString("Name");
47         int sHeroId = rs.getInt("SuperHeroId");
48         String gender = rs.getString("Gender");
49         String race = rs.getString("Race");
50         String height = rs.getString("Height");
51         String weight = rs.getString("Weight");
52         String eyeColor = rs.getString("EyeColor");
53         String hairColor = rs.getString("HairColor");
54         String groupAffiliation = rs.getString("GroupAffiliation");
55         String relatives = rs.getString("Relatives");
56       }
57     } catch (Exception e) {
58       e.printStackTrace();
59     }
60   }
61 %>

```

Once a hero is selected from the search bar the user is brought to another page that connects to the database by variable connection, then defines all information in the table which will then be displayed. The superhero ID is used to query the table to display the characteristics of that particular superhero. This superheroId is used in the *where* condition of the SQL query. And the corresponding result set from the query is used to get each of the particular characteristics of the superhero like name, alterEgos, placeOfBirth, firstAppearance, etc.

Stat Power Statistic Graph:

```

</div>
<div id="graphDiv">
  <p> <h1> Power Stats</h1></p>
<canvas id="mycanvas" style=" left:10px; width:350px; height:100px;background-color: black;"></canvas>
</div>
</body>
<script src="https://code.jquery.com/jquery-3.3.1.js"></script>
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js@2.7.0/Chart.bundle.js"></script>
<script style="color:white">
  $(document).ready(function(){
    bargraph();
  })
  var bargraph = function(){
    $.ajax({
      url : "superHeroPowerStats.jsp?search=<%request.getParameter("search");%>",
      dataType: 'json',
      type: "GET",
      success : function(data){
        var str = "dataset";
        var powerstatsLabel = [];
        var dataset1 = [];
        console.log(data.powerstats);
        for(var i in data.powerstats) {
          dataset1.push(data.powerstats[i].Intelligence);
          dataset1.push(data.powerstats[i].Strength);
          dataset1.push(data.powerstats[i].Speed);
          dataset1.push(data.powerstats[i].Durability);
          dataset1.push(data.powerstats[i].Power);
          dataset1.push(data.powerstats[i].Combat);
        }
        powerstatsLabel.push("Intelligence");
        powerstatsLabel.push("Strength");
        powerstatsLabel.push("Speed");
        powerstatsLabel.push("Durability");
        powerstatsLabel.push("Power");
        powerstatsLabel.push("Combat");

        var chartdata = {
          labels : powerstatsLabel,
          datasets: [
            {
              fill : false,
              lineTension : 0,
              //backgroundColor: "rgba(130,224,170,0.75)",
              backgroundColor: ["red", "blue", "green", "orange", "purple", "yellow"],
              data: dataset1
            }
          ]
        };
      }
    });
  }

```

Displays all powers of the hero in a bar graph using AJAX and jQuery. The powerstats of a superhero is displayed using chart.js and this chart.js uses a dataset in the form of JSON file. The JSON dataset is created using the SQL query and resultSet and is loaded onto a different page namely superHeroPowerStats.jsp. And through AJAX and jQuery calls, this dataset is loaded to a variable. The x-axis and y-axis of the graphs are also predefined in this graph as an object notation of the chart.js mechanism.

Quiz:

```

</header>
<body onload="init()">
  <div id="pgMainMenu">
    <h1 id="headerTitle">Which SuperHero Am I?</h1>

    <p id="options">
      <button id="bStartQuiz" name="startQuiz" onclick="startQuiz()">Start Quiz</button>

    </p>
  </div>
  <div id="pgQuiz">
    <p id="question">Question</p> <br>
    <p>
      <input id="OptAnswer1" type="radio" name="answer" value="1"><label
        for="OptAnswer1"><span id="answerPos1">Answer1</span></label>
    </p>

    <p>
      <input id="OptAnswer2" type="radio" name="answer" value="2"><label
        for="OptAnswer2"><span id="answerPos2">Answer2</span></label>
    </p>

    <p>
      <input id="OptAnswer3" type="radio" name="answer" value="3"><label
        for="OptAnswer3"><span id="answerPos3">Answer3</span></label>
    </p>

    <p>
      <input id="OptAnswer4" type="radio" name="answer" value="4"><label
        for="OptAnswer4"><span id="answerPos4">Answer4</span></label>
    </p>
    <button id="bSubmitAnswer" onclick="checkAnswer()">Submit
      Answer</button>
    <button id="bResetQuiz" onclick="resetQuiz()">Cancel Quiz</button>
  </div>
  <div id="QuizResults">
    <h3>
      <span>Quiz Results</span>
    </h3>
    <p id="ResultMessage">Congratulations you did it!</p>
    <p id="QuizScore">Final Score: 0</p>
    <button id="bResetQuiz" onclick="resetQuiz()">Main Menu</button>
  </div>
</body>

```

The quiz uses a set of 10 predefined questions and answers. The answers are then assigned to a point value. For example consider question 1, “If you were on your way to your day job and saw a bank robbery, what would you do?”, the options assigned with their respective point values were, “Put on my disguise and solve the issue” with point value of 1, “I would mind my own business” with a point value of 2, “I would call the police.” with a point value of 3 and lastly “I don’t have a job” with a point value of 4. These points at the end of the quiz are summed and then a corresponding superhero is displayed based on the points nearing that particular superhero.

Quiz Continued:

```
function checkAnswer() {
    var answer = document.getElementsByName("answer");
    var selectedAnswer = 0;

    for (var i = 0; i < answer.length; i++) {
        if (answer[i].checked)
            selectedAnswer = answer[i].value;
    }
    if (selectedAnswer == "") {
        alert("You haven't chosen an answer");
        return false;
    } else {
        quizScore += parseInt(selectedAnswer);
    }
    quiz[currentQuiz].asked = "true";
    nextQuestion();
}

function nextQuestion() {
    document.getElementById("OptAnswer1").checked = false;
    document.getElementById("OptAnswer2").checked = false;
    document.getElementById("OptAnswer3").checked = false;
    document.getElementById("OptAnswer4").checked = false;
    var numberAsked = 0;
    for (var i = 9; i >= 0; i--) {
        if (quiz[i].asked == "true") {
            numberAsked++;
        } else {
            currentQuiz = i;
        }
    }
    if (numberAsked == 10) {
        ResultsInfo();
    } else {
        document.getElementById("question").innerHTML = quiz[currentQuiz].questiontext;
        document.getElementById("answerPos1").innerHTML = quiz[currentQuiz].answer1;
        document.getElementById("answerPos2").innerHTML = quiz[currentQuiz].answer2;
        document.getElementById("answerPos3").innerHTML = quiz[currentQuiz].answer3;
        document.getElementById("answerPos4").innerHTML = quiz[currentQuiz].answer4;
        document.getElementById("answerPos5").innerHTML = quiz[currentQuiz].answer5;
        document.getElementById("answerPos6").innerHTML = quiz[currentQuiz].answer6;
        document.getElementById("answerPos7").innerHTML = quiz[currentQuiz].answer7;
        document.getElementById("answerPos8").innerHTML = quiz[currentQuiz].answer8;
        document.getElementById("answerPos9").innerHTML = quiz[currentQuiz].answer9;
        document.getElementById("answerPos10").innerHTML = quiz[currentQuiz].answer10;
    }
}
```

Many functions were used to define the quiz because of its complexity. One function was to clear all the options before the question was asked, one function was to check if all questions were answered, one to calculate the total points for that particular quiz session and finally one to display the superhero.

Experiments & Challenges

Some challenges we faced were transferring code between each other due to server resources and the amount of data we were working with. We also had some trouble with MySQL on some of our computers because certain systems would not allow access due to software updates and such. We were also worried about how our data was going to be affected because when the API was retrieved, there were a few missing values.

To test the data, we searched various superheroes to make sure they were displayed. Then whatever data appeared for the superhero we searched, we would look at our database, and see if that matched. We also searched various snippets of superhero names, to ensure that the string matching works. String matching was one of the more important aspects of this project because it would be annoying if users had to search the entire name of the superhero to display everything, but rather they can search a small snippet of the superhero's name.

Results & Discussion

We were able to complete everything we set out to do. The homepage has a search engine for users to search for any character they want. It is a partial string search, so users do not need to enter the full name of the character in order to get results. Additionally, if the user is not familiar with superheroes and would like to learn more, rather than searching them up, we have all of them listed out with their pictures and their name. They can scroll through and pick whichever character they want and will be redirected to another page with more information on that selected character. We also completed the personality quiz for users to take and find out what superhero they are.

6.0 Conclusions

The main contributions of the project were retrieving the superhero API and formatting it the way we needed to. We also had to create a website and a user and provide users with the functionality of searching for characters and ensuring a partial string would work. Having the quiz was an added bonus for users to have another feature to explore. We were fortunate enough to be able to accomplish everything we proposed we would do. Users can search for heroes, they can view all the heroes listed out and click on whichever one they want to learn more about, and they can take a personality quiz. One thing we did not have time to complete was including all the characters as options on the quiz. There are hundreds of characters so we had to pick a handful to work with for the time being. For future implementations, we would like to incorporate a login system so users can save their favorite characters in a favorites section. We would also like to fully complete the quiz, so it takes all the heroes into account, as well as making an android app so users have the option to use our platform on the go.

7.0 Individual Contribution of Each Team Member and Learning Experience

7.1 Nadia Hossain: I worked on the front end UI and the quiz. The frontend was styling so there were not that many challenges. The quiz, however, was particularly challenging to work on because we tried so many different ways to code a quiz. We first tried to retrieve the data from MySQL and then use javascript to code a quiz. However, it was not particularly an easy thing to do because there were so many superheroes and it would be very time-consuming to create so many factors that would lead to each of the 800 superheroes. To get the quiz working for the time being, we only implemented it using 5 superheroes at the moment and for future implementation, we would try to accompany all 800.

7.2 Nuran Ghoneim: I worked on the backend and helped with the quiz. I worked with Shipra to help format the data and prepare it for it to be used for the website. I also worked with Nadia by helping develop the questions for the quiz and the answers, as well as assigning which character to which answer. I worked through the debugging of the code and created the point based logic system for the quiz. Nadia is not as familiar with superhero personalities as I am, so I was behind the logic of the personalities.

7.3 Riddhi Makwana: I worked on the frontend, primarily focusing on creating the layout of the website and designing it. I then incorporated the main features and functionalities that we needed such as the search engine. Additionally, I worked on adding tables to MySQL for the information to display on the frontend. I ensured that the UI was user friendly and easy to navigate.

7.4 Shipra Priyadarshini: I worked on the designing the schema based on the response we got from the superhero API used. I worked on the authentication and key used to retrieve the superhero data from the superhero API. I worked on the parsing and loading of the superhero data onto the MySQL database using Java, J2EE, JDBC and Restful API. Additionally, I worked with Riddhi and Nuran on getting the connection established between the frontend and the backend which was used to display the data. In this project, I learned the various authentication factors used to parse a Restful API and means of parsing them so that it can not only be loaded into the database but also works in an efficient way.

8.0 References

“SuperHero API Documentation.” *SuperHero API*, superheroapi.com/.

Comic Vine. comicvine.gamespot.com/.

DC, www.dccomics.com/.

“Rick and Morty!” *Dark Horse Comics*, www.darkhorse.com/.

Superhero Database, www.superherodb.com/.

“The Official Site for Marvel Movies, Characters, Comics, TV.” *Marvel Entertainment*, www.marvel.com/.