

RIVALYTICS

Midterm Report

By

Camilo Arias

300356827

For

CSIS 4495 - Applied Research Project - Section 002

Padmapriya Arasanipalai Kandhadai

VIDEO LINK: <https://youtu.be/orRBU9riH-4>

Winter 2025

Introduction

Marvel Rivals is a fast-paced *team-based* hero shooter that has very quickly gained a name for itself in the gaming community with its rich roster of Marvel characters and intricate strategies. Players must select heroes, synergize with teammates, and adapt to dynamic gameplay in a variety of playable maps. In each match, a total of 12 players will be divided into 2 teams of 6. Each hero has a unique set of abilities and powers which they use to eliminate the opposing team and capture the objectives to win the game.

Marvel Rivals is considered a live service game, which means that the game is constantly patched to achieve a balance. That is, to weaken characters who are overperforming, or strengthening those who are underperforming. As such, you'll need to constantly rethink how to strategize to optimize your performance and climb up the ranks.

Framing the Problem

Despite its growing popularity, Marvel Rivals is still a very recent game and thus there is no dedicated analytics platform to help players improve performance or refine strategies, leaving a significant gap in the community's needs. Competitive multiplayer games depend heavily on data analysis to optimize strategies, understand trends, and enhance player performance.

Tools like [Lolalytics](#) have successfully catered to the League of Legends community, providing comprehensive insights into win rates, hero matchups, and meta trends. However, no similar platform exists for Marvel Rivals. As such, I've developed a few key questions to guide the development of this project:

1. How can an analytics website deliver the necessary insights into hero performance and strategy optimization?
2. What are the most effective ways to present data for easy user comprehension?
3. What other tools can aid users to improve their performance in the game?

4. How can emerging trends and player feedback be integrated into the platform's functionality?

Existing Research

The success of platforms like Lolalytics and [Overbuff](#), which focus on League of Legends and Overwatch respectively, demonstrates the value of detailed analytics for competitive games. These tools emphasize win rates, pick rates, and team synergy to help players make informed decisions. However, common challenges include the lack of real-time updates, limited interactivity, and inadequate support for mobile devices. Rivalytics aims to address these limitations while tailoring the solution to Marvel Rivals' unique mechanics and gameplay.

Hypotheses and Benefits

A dedicated analytics platform for Marvel Rivals has the potential to significantly enhance player performance by providing valuable insights from game data. By presenting this data through visual dashboards and interactive tools, the platform is expected to improve player understanding of game mechanics, allowing for more informed decision-making during gameplay.

The platform will offer several key benefits: Players will have the opportunity to refine their gameplay and strategies using the insights provided, creating a more competitive environment where every advantage matters. Additionally, game developers can utilize the platform's data to gain valuable insights into the balancing of heroes and updating game mechanics, ensuring a fair and engaging experience for all players.

Summary of the Initially Proposed Project

My project, **titled Rivalytics**, aims to develop a comprehensive web-based platform designed to help *Marvel Rivals* players by providing in-depth hero statistics, matchup insights, and interactive tools for strategy development. The platform will aggregate game data, including win rates, pick rates, and ban rates. These features will be complemented by visual dashboards, including charts and graphs, to provide a clear representation of hero performance and popularity across different game scenarios. The platform will also serve as an educational resource, integrating video tutorials and written guides to help players refine their skills.

Regarding the website's technology stack, this includes React.js for the frontend, Node.js with Express.js for the backend, and MongoDB for efficient data storage. Chart.js will be used for data visualization, and Firebase will handle user authentication.

Changes to the Proposal

Most of the changes to the proposal are more about trimming the original scope of the project. As the semester has gone on and having seen some of the challenges that I've ran into, I've decided to bring down the scope of the project by removing the following features that were proposed initially:

- Custom Match Simulator
- Discussion Forums
- Team Builder

And while not a change, the only noticeable change so far regarding the project's timeline has been the implementation of the API for hero statistics, matchup data, and game trends.

I've also included a new feature in the website where every hero has a personal page to display not only their data, but also more information about the hero itself such as their biography, role, etc.

Other than that, most of the project remains unchanged, including the technology stack and the rest of the proposed timeline.

PROJECT PLANNING AND TIMELINE

From current period to end of term

Frontend Development (Weeks 8–11)

Milestones:

- Develop the user interface using React.js and Bootstrap.
- Integrate data visualizations.
- Implement user account features and community tools.
- Develop team builder feature and match simulation.
- Implement user guides and video embedding.
- Implement APIs for hero statistics, matchup data, and game trends.

Deliverables:

- Progress Reports 2 and 3

Testing (Week 12)

Milestones:

- Conduct testing with potential users.
- Ensure mobile responsiveness

Deliverables:

- Progress Reports 4 and 5

Final Report and Submission (Week 13)

Milestones:

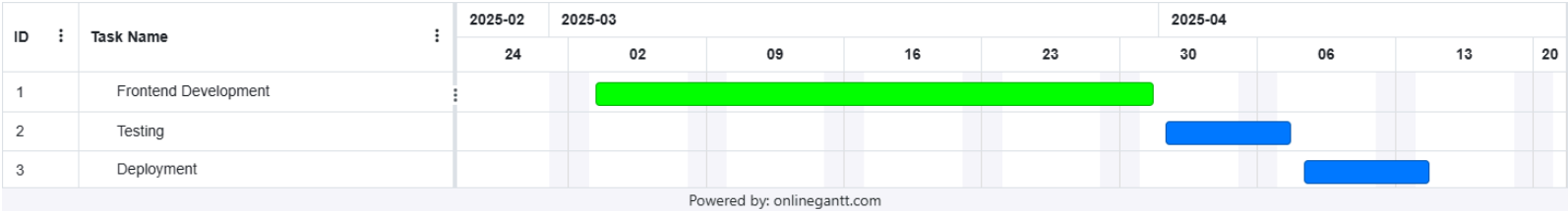
- Prepare final documentation.
- Submit fully working project.

Deliverables:

- Final report.

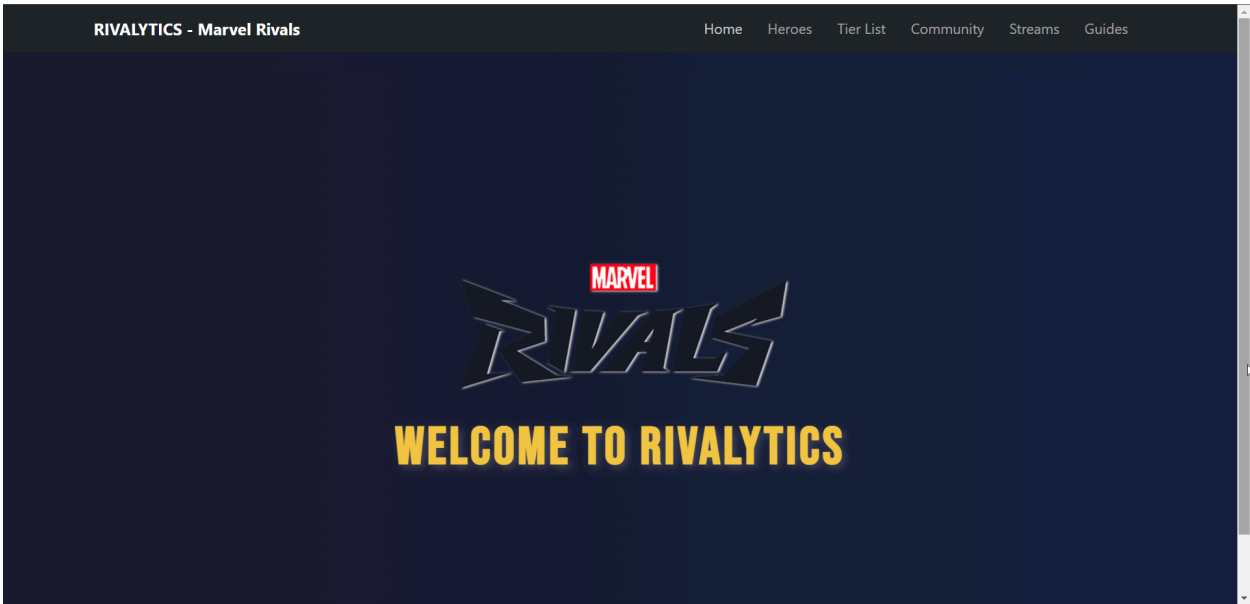
GANTT CHART

From current period to end of term



FEATURES IMPLEMENTED

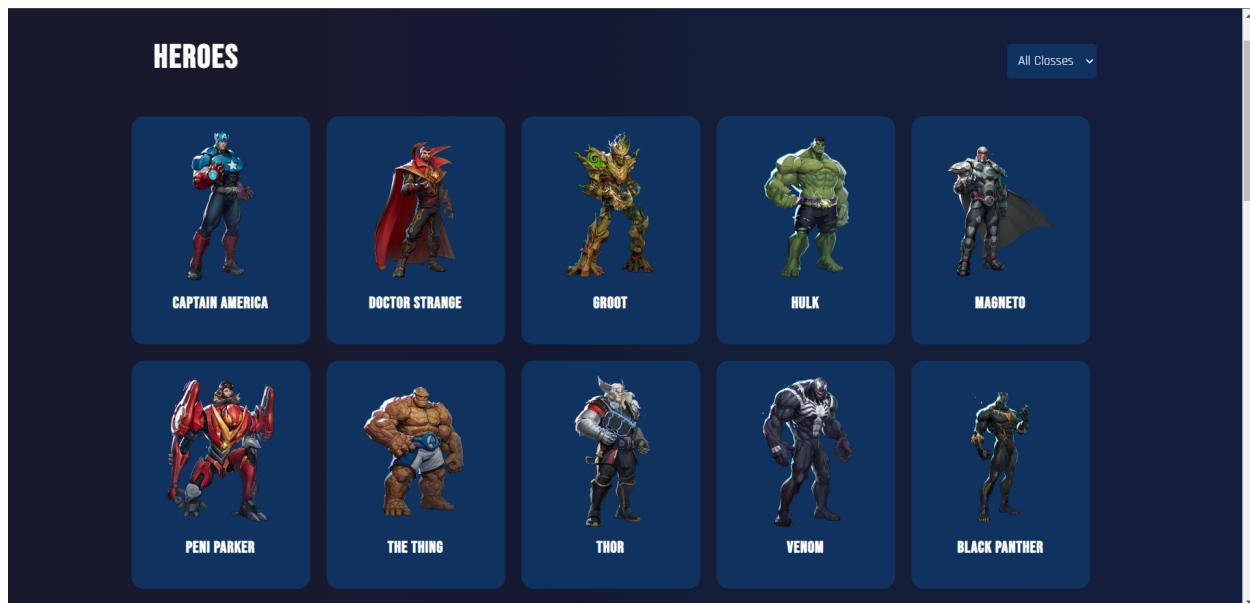
- **Home Page:** This is the landing page, featuring a basic welcome message and the logo of the game. Still unfinished as more will be added.



W25_4495_S2_CamiloA > Implementation > frontend > src > pages > JS HomePage.js > ...

```
1 import React from "react";
2 import "../HomePage.css";
3
4 const HomePage = () => {
5   return (
6     <div className="home-container">
7       <div className="home-content">
8         
9         <h1 className="home-title">Welcome to RIVALYTICS</h1>
10      </div>
11    </div>
12  );
13 };
14
15 export default HomePage;
16
```

- **Heroes Page:** This is the feature I want to show off in this report. This is a page containing all 37 heroes, their names, and their appearance in the game. They can be filtered by role (tank, duelist or support) so only the heroes from that role are displayed. The main feature of this page is that clicking on each hero will redirect the user to the hero's personal page.



JS HeroesPage.js X


W25_4495_S2_CamiloA > Implementation > frontend > src > pages > JS HeroesPage.js > HeroesPage > filteredHeroes.map() callback

```
1 import React, { useState, useEffect } from "react";
2 import axios from "axios";
3 import "./HeroesPage.css";
4 import { useNavigate } from "react-router-dom";
5
6 const HeroesPage = () => {
7   const [heroes, setHeroes] = useState([]);
8   const [filteredHeroes, setFilteredHeroes] = useState([]);
9   const [selectedRole, setSelectedRole] = useState("All");
10
11   useEffect(() => {
12     const fetchHeroes = async () => {
13       try {
14         const response = await axios.get("http://localhost:5000/api/heroes");
15         setHeroes(response.data);
16         setFilteredHeroes(response.data);
17       } catch (err) {
18         console.error("Error fetching heroes:", err);
19       }
20     };
21
22     fetchHeroes();
23   }, []);
24
25   const handleFilterChange = (role) => {
26     setSelectedRole(role);
27     if (role === "All") {
28       setFilteredHeroes(heroes);
29     } else {
30       setFilteredHeroes(heroes.filter(hero => hero.role === role));
31     }
32   };
33
34   const navigate = useNavigate();
35
36   const handleHeroClick = (heroId) => {
37     navigate(`/heroes/${heroId}`);
```

```
38   };
39
40
41   return (
42     <div className="heroes-page-container">
43       <div className="header-container">
44         <h1 className="heroes-title">Heroes</h1>
45         <select className="filter-dropdown" value={selectedRole} onChange={(e) => handleFilterChange(e.target.value)}>
46           <option value="All">All Classes</option>
47           <option value="Tank">Tanks</option>
48           <option value="Duelist">Duelists</option>
49           <option value="Support">Supports</option>
50         </select>
51       </div>
52
53       <div className="heroes-grid">
54         {filteredHeroes.map(hero => (
55           <div key={hero._id} className="hero-card" onClick={() => handleHeroClick(hero._id)}>
56             <img src={`/${images}/${hero.image}`} alt={hero.name} className="hero-image" />
57             <p className="hero-name-grid">{hero.name}</p>
58           </div>
59         ))}
60       </div>
61     </div>
62   );
63 };
64
65 export default HeroesPage;
```


- **Hero Details Page (For all 37 heroes):** While here I'm only showing off 1 hero, every hero has a page like this. Once you click on any given hero in the previous page, this page will display the hero's name, role, stats, biography and a small description as well.

[← Back to Heroes](#)



IRON MAN

DUELIST

Win Rate: 50.6%

Pick Rate: 12.4%

Ban Rate: 5.5%

Biography

Wounded by a weapon of his own design, billionaire inventor Tony Stark created a custom-made suit of armor to keep himself alive. Though his wounds have healed, his upgraded armor now saves the lives of countless others when Tony dons it to become the world's greatest fighting machine – the invincible Iron Man! When the legions of the night descended upon New York City, Iron Man and his fellow Avengers turned their tower into a stronghold. From there, Tony leads the effort to unravel the mysteries of Chronovium and to reverse the Timestream Entanglement.

Armed with superior intellect and a nanotech battlesuit of his own design, Tony Stark stands alongside gods as the Invincible Iron Man. His state of the art armor turns any battlefield into his personal playground, allowing him to steal the spotlight he so desperately desires.

JS HeroDetailsPage.js M X

W25_4495_S2_CamiloA > Implementation > frontend > src > pages > JS HeroDetailsPage.js > [🔍] HeroDetailsPage

```
1  import React, { useEffect, useState } from "react";
2  import { useParams, useNavigate } from "react-router-dom";
3  import axios from "axios";
4  import "../HeroDetailsPage.css";
5
6  const HeroDetailsPage = () => {
7    const { id } = useParams();
8    const navigate = useNavigate();
9    const [hero, setHero] = useState(null);
10   const [loading, setLoading] = useState(true);
11   const [error, setError] = useState(null);
12
13   useEffect(() => {
14     const fetchHeroDetails = async () => {
15       try {
16         const response = await axios.get(`http://localhost:5000/api/heroes/${id}`);
17         setHero(response.data);
18       } catch (err) {
19         setError("Failed to load hero details.");
20       } finally {
21         setLoading(false);
22       }
23     };
24
25     fetchHeroDetails();
26   }, [id]);
27
28   if (loading) return <p>Loading hero details...</p>;
29   if (error) return <p>{error}</p>;
30   if (!hero) return <p>Hero not found.</p>;
31
```

```

32   return (
33     <div className="hero-details-container">
34       <button onClick={() => navigate("/heroes")} className="back-button">← Back to Heroes</button>
35
36       <div className="hero-details-content">
37         <div className="hero-image-container">
38           <img src={` /images/${hero.detailImage}`} alt={hero.name} className="hero-detail-image" />
39         <p className="hero-description">{hero.description}</p>
40       </div>
41
42       <div className="hero-info">
43         <h1 className="hero-name">{hero.name}</h1>
44         <div className="hero-role">{hero.role}</div>
45         <div className="hero-stats">
46           <p><strong>Win Rate:</strong> {hero.winRate}%</p>
47           <p><strong>Pick Rate:</strong> {hero.pickRate}%</p>
48           <p><strong>Ban Rate:</strong> {hero.banRate}%</p>
49         </div>
50         <div className="hero-biography">
51           <h2>Biography</h2>
52           <p>{hero.biography}</p>
53         </div>
54       </div>
55     </div>
56   </div>
57 );
58 };
59
60 export default HeroDetailsPage;

```

WORK LOG

Date	Number of Hours	Description of Work Done
January 16, 2025	1	Coming up with ideas for the project
January 17, 2025	1	In-class consultation about 2 different project ideas
January 21, 2025	0.5	Deciding the idea for the project
January 23, 2025	1	Starting the project proposal
January 24, 2025	1	In-class project consultation and initial check-in
January 24, 2025	2	Defining the project's scope and features after consultation
January 25, 2025	3	Continuing project proposal writing
January 26, 2025	4	Finishing project proposal and submitting
February 7, 2025	2	Coming up with the possible endpoints
February 8, 2025	5	Setting up the backend and writing the project report

February 8, 2025	1	Committing to github repo
February 18, 2025	4	Finishing backend work
February 19, 2025	3	Starting the frontend
February 21, 2025	4	Creating the database and populating it
February 22, 2025	5	Continuing frontend work, including heroes page
February 23, 2025	4	Finishing heroes page and starting heroes detail page
February 24, 2025	7	Finishing heroes page, report and video