**ReadMe Template Worksheet** 

Your ReadMes are the key deliverable that engineers will want to see as part of your job search. Each engineering team will look at different aspects of your ReadMe and repo. Some will go into the code itself and explore. Others will just want to see the showcase code snippets in the main ReadMe. Some will read the full thing to understand your approach, others will skim to specific sections.

It’s crucial that you cover all the different sections below to ensure that you’ve got the information for all engineers that check these out.

We regularly have employer partners discuss the importance of the ReadMes in what they’re looking for and why they interview the grads that they do - so don’t underestimate the importance of writing strong ReadMes!

For any pair or group project, you cannot share a ReadMe. These must be written independently to ensure that the engineers reading this understands **your** specific experience and approach.

It’s a good idea to start your ReadMes during the planning stage as this is the best way to get ahead and save time when it comes to finalising your first draft post-project. As you plan each aspect of your project, note down what you intend to do with screenshots of your plan and anything else you think would be useful, then when you execute this part in your code, you can adjust this part of your ReadMe as needed depending on how your process changed, or if it went as planned then you can leave it as it is.

**Make a copy of this document for each project you use throughout the course and fill in each section. Keep your copy in your Outcomes folder, so the team is able to add feedback into your docs.**

Once the content has been finalised by the Outcomes team, you can then upload these onto your GitHub repository later.

**Things To Consider:**

* That there are **no spelling mistakes in your ReadMe** - if you see a spelling error highlighted below, edit this.
  + Some engineers will reject applicants if their ReadMes are full of mistakes. From their perspective, if your ReadMes are full of mistakes, what is your code like…

* That your **technologies are capitalised correctly** - i.e JavaScript, jQuery, MongoDB

* That your **formatting is consistent** throughout - headers, indentation, full stops in bullets etc

* Any **hyperlink included works**

* That you **include images throughout** - code snippets, pictures of your planning stage, screenshots of the final project.
  + These can be still screenshots or gifs
  + This breaks up the text in your ReadMe and helps to keep the reader engaged

* That your ReadMe **sounds like you** - see this as an opportunity to showcase who you are to the engineering community and prospective employers.
  + Think back to the Personal Brand session and how employers want to **get a sense of who you are**. The content you write should sound as if you’re talking through your experience.

**ReadMe Sections**

**Description**

**Instructions**

*Here, give a short description of the project. It can be a couple of sentences where you discuss the point in time during the course that you completed it, the topic of the project and potentially the tech stack.*

**Insert your Description here:**

The project was designed to retrieve data from the Rick and Morty API and display it in a web format. The web application showcases characters from the show, and upon selecting a character from the list, it displays the episodes in which that character appears. Additionally, a search bar is provided to allow users to easily select their favourite character.

**Deployment link**

**Instructions**

*Here include the information on where the deployed project can be found. If login details are needed to access the full project, make sure you include them.*

*If you have not yet deployed your project, you can add this in later.*

**Insert your Deployment link here:**

[Ricky and Morty](https://showmewhatyougot.netlify.app)

**Getting Started/Code Installation**

**Instructions**

*Explain how the reader accesses your code. Include a step by step approach.*

**Insert your Getting Started/Code Installation here:**

npm i

npm i sass

npm bootstrap

npm install react-bootstrap bootstrap

**Timeframe & Working Team (Solo/Pair/Group)**

**Instructions**

*Share the timeframe given for the project and whether you worked independently, in a pair, or in a group.*

*If you worked in a pair or group, include the names of the people you collaborated with. As a bonus, you can also provide links to their GitHub repo.*

**Insert your Timeframe & Working Team here:**

This was a pair team effort by Antonio Climent and Gilberto Carnieli.

2 days were given to build this project.

**Technologies Used**

**Instructions**

*List every technology you used to complete the project. This can be in one long list, or broken down into categories (Back End, Front End, Development Tools).*

**Insert your Technologies Used here:**

* SASS
* REACT-ROUTER
* REACT
* GIT
* GITHUB
* INSOMNIA

**Brief**

**Instructions**

*Include the brief set by your instructional team here. This sets the context of the project you were working towards and mimics briefs you will be set later in your future roles.*

*This can either be in bullets or in a paragraph.*

**Insert your Brief here:**

**# ![](https://ga-dash.s3.amazonaws.com/production/assets/logo-9f88ae6c9c3871690e33280fcf557f33.png) Project #2: Reacathon**

​

**## Overview**

​

The second project is to **\*\*build a React application\*\*** that consumes a **\*\*public API\*\***.

​

**### Technical Requirements**

​

Your app must:

​

\* **\*\*Consume a public API\*\*** – this could be anything but it must make sense for your project.

\* **\*\*Have several components\*\***

\* **\*\*The app can have a router\*\*** - with several "pages", this is up to you and if it makes sense for your project.

\* **\*\*Include wireframes\*\*** - that you designed before building the app.

\* **\*\*Be deployed online\*\*** and accessible to the public (hosted on your public github, not GA github!)

​

---

​

**## Necessary Deliverables**

​

\* A **\*\*working application\*\***, hosted somewhere on the internet

\* A **\*\*link to your hosted working app\*\*** in the URL section of your Github repo

\* A **\*\*git repository hosted on Github\*\***, with a link to your hosted project, and frequent commits dating back to the *\_very beginning\_* of the project

\* **\*\*A `readme.md` file\*\*** with:

\* Explanations of the **\*\*technologies\*\*** used

\* A couple of paragraphs about the **\*\*general approach you took\*\***

\* **\*\*Installation instructions\*\*** for any dependencies

\* Link to your **\*\*wireframes\*\*** – sketches of major views / interfaces in your application

\* Descriptions of any **\*\*unsolved problems\*\*** or **\*\*major hurdles\*\*** your team had to overcome

​

---

​

**## Suggested Ways to Get Started**

​

\* **\*\*Don’t hesitate to write throwaway code\*\*** to solve short term problems.

\* **\*\*Read the docs for whatever technologies / frameworks / APIs you use\*\***.

\* **\*\*Write DRY code\*\***.

\* **\*\*Be consistent with your code style.\*\***

\* **\*\*Commit early, commit often.\*\*** Don’t be afraid to break something because you can always go back in time to a previous version.

\* **\*\*Keep user stories small and well-defined\*\***, and remember – user stories focus on what a user needs, not what development tasks need accomplishing.

\* **\*\*Write code another developer wouldn't have to ask you about\*\***. Do your naming conventions make sense? Would another developer be able to look at your app and understand what everything is?

\* **\*\*Make it all well-formatted.\*\*** Are you indenting, consistently? Can we find the start and end of every div, curly brace, etc?

\* **\*\*Comment your code.\*\*** Will someone understand what is going on in each block or function? Even if it's obvious, explaining the what & why means someone else can pick it up and get it.

\* **\*\*Write pseudocode before you write actual code.\*\*** Thinking through the logic of something helps.

​

---

​

**## Sign Off Reqs**

​

\* A working request in Insomnia from the API you want to use, showing that you can access the dataset you need.

\* A basic wireframe of your apps design + components

​

**## Useful Resources**

​

\* **\*\*[React]**(https://reactjs.org/)**\*\***

\* **\*\*[Some free APIs]**(https://apilist.fun/)**\*\***

\* **\*\*[Some more free APIs]**(https://github.com/public-apis/public-apis)**\*\***

\* **\*\*[Even more free APIs]**(https://dev.to/camerenisonfire/10-intriguing-public-rest-apis-for-your-next-project-2gbd)**\*\***

\* **\*\*[The best free API Finder]**(https://www.google.com)**\*\***

**Planning**

**Instructions**

*The planning stage is important, as all projects in your future roles will have detailed plans before any coding happens. It is a great experience to share with potential engineer employers, as this reflects real engineering team practices.*

*Start by explaining the initial steps you took in the project.*

* ***Did you do any sketches****? If so, discuss this and include images.*
* ***Any wireframes of the front end and UI****? You did? Then explain this and include images.*
* ***Any ERDs****? Same here, explain and include images.*
* ***Use a project management tool to plan the sprint****? If so, talk through this - what tool did you use? How you allocated tickets/responsibilities, sprint timeline etc. Also include screenshots of this.*
* ***Any pseudocode****?*
* ***If it was a group or pair project*** *- Discuss who was designated which tasks. This is very important, as engineers want to understand who owned the different code elements when looking at a group project.*

*For each project, review the above bullets and discuss every step you took in the planning stage, including the relevant images.*

*Not every project will include the above, but it’s important to discuss any of the bullets that you did implement.*

**Insert your Planning here:**

This was the wireframe that Antonio and I created before initiating our project. We were able to implement it as we had envisioned. Our basic wireframe outlined the various pages and routes we intended to incorporate.

A screenshot of a computer screen

Description automatically generated

Antonio handled coding and debugging, while I focused on researching potential issues and codes that would be necessary for the successful implementation of our project.

**Build/Code Process**

**Instructions**

*The Build/Code Process will be the longest section of your ReadMe and will be most insightful to the engineers that review them. This is where you will discuss the steps you took to code the project.*

*You want to see your ReadMes as a way to walk the engineers through your approach and problem solving from the start of the project through to the end.*

*You'll need to include a minimum of 3-4 code snippets, highlighting code you're particularly proud of and these code snippets will have descriptions on what you did, how and why to set the context of the snippet you include. These explanations are important for the engineers, as they will want to understand what you did and the reasoning behind the steps you took.*

*You don't need to document every single thing you coded, but walk them through the key sections of the project build.*

*For any group project, you will just focus on your contributions.*

*Some people will document the build/code process by discussing the key stages they worked on. Others will do a day by day guide. It’s entirely up to you how you structure this, as long as you discuss all the key things above.*

**Insert your Build/Code Process here:**

Day 1 – We began the day by creating the wireframe. We searched for key elements from the API that we would need and then proceeded to coding. The initial code we wrote focused on selecting data from all 42 pages, as obtained from the return in Insomnia.

import { Link, useLoaderData } from "react-router-dom"export default function Chars(){  
 /\* Data is an array of 42 pages containing characters each \*/  
 const chars = useLoaderData()  
 return (  
   <>  
     <h1>Characters</h1>  
     <div className="sections">  
       {/\* We map through the data \*/}  
       {chars && chars.map(page => {  
         const { results } = page  
         // We map each character  
         return results.map(char => {  
           const {id, name, image} = char  
           // And we render the component  
           return (  
             <Link key={id} to={`/characters/${id}`}>  
               <section>  
                 <h2>{name}</h2>  
                 <img className="charImg" src={image} alt="Character Image" />  
               </section>  
             </Link>  
           )  
         })  
       })  
       }  
     </div>  
   </>  
 )  
}

Day 2 – We completed the remaining sections of the code, which included implementing the characters, enabling navigation to new pages upon clicking, and defining the stylistic elements for our web. We also worked on the logic to extract information for each individual character.

The logic to get that single character

  try {  
    // all your fetching needs in one place!  
    const res = await fetch(`[https://rickandmortyapi.com/api/character/${extra}](https://rickandmortyapi.com/api/character/$%7Bextra%7D)`)  
    return res.json()  
  } catch (error) {  
    console.log(error)  
  }  
}

Once we had that piece of code running, it was time to code the "/episode/:epid" route. + you need to explain your approach

import { Link, useLoaderData } from "react-router-dom"export default function Episode(){  
  // episode is an array with data from the episode on the first position and an array of the data from the characters in the episode on the other  
  const ep = useLoaderData()  
  // ep[0] is the episode data itself  
  const {name, episode} = ep[0]  
  return (  
    <>  
      <h1>{episode} {name}</h1>  
        <h2 className="cast">Cast</h2>  
        <div className="episodeCont">  
          {/\* ep[1] is the data of the characters IN the episode \*/}  
        {ep[1].map(e => {  
          // destructuring avoiding variable 'name' to be repeated  
          const {id, name: charName, image} = e  
          return (  
            // rendering of the elements  
            <section key={id}>  
              <Link to={`/characters/${id}`}>  
                <h3>{charName}</h3>  
                <img className="charImg" src={image} alt={`Image of ${charName}`} />  
              </Link>  
            </section>  
            )  
        })}  
        </div>  
    </>  
  )  
}

In this section of the code, we load information for a single character using the base ID obtained from the API.

import { Link, useLoaderData } from "react-router-dom"export default function SingleChar(){  
  // We get a single character from the loader based on its id, then destructure it  
  const char = useLoaderData()  
  const {name, image, episode} = char  return (  
    <>  
      <h1>Characters</h1>  
      <div className="container">  
        <div className="char">  
          <h2>{name}</h2>  
          <img src={image} alt={`Image of ${name}`} />  
          <h3>Episode List</h3>  
          <div className="eps">  
            {/\* Episode consists of an array of strings each referring to the episode \*/}  
            {episode.map((ep, i) => {  
              // The episode string is a url with the last section of it containing the episode's unique id  
              // so we split it and the use num[num.length-1] to actually target the id  
              const num = ep.split('/')  
              return (  
                <Link to={`/episode/${num[num.length-1]}`} key={i}>Episode {num[num.length-1]}</Link>  
              )  
            })}  
          </div>  
        </div>  
      </div>  
    </>  
  )  
}

After completing all of the above, including a component for the Navbar that repeats on every page, we introduced an input on the homepage to enable users to search for any specific character they wish.

import { useState } from "react"  
import { useLoaderData, Link } from "react-router-dom"export default function Home(){  
  const [ search, setSearch] = useState('')  
  const data = useLoaderData()  
  // pattern gets updated each time the user writes on the  
  const pattern = new RegExp(search, 'i')  return (  
    <>  
    <h1>Search for Character</h1>  
    <div className="formCont">  
      <input type="text" name="charName" placeholder="Search Your Favourite Character" value={search} onChange={(e) => setSearch(e.target.value)}/>  
    </div>  
    <div className="sections">  
      {/\* Data is an array of 42 pages containing characters each so we map through it \*/}  
    {data.map(p => {  
      // we filter each page for the characters whose name match the search criteria and save them in a single array  
      const chars = p.results.filter(r => pattern.test(r.name))  
      // we map the array of matching characters to create each individual card  
      return chars.map(c => {  
        const {id, name, image} = c  
        return (  
          <Link key={id} to={`/characters/${id}`}>  
            <section>  
              <h2>{name}</h2>  
              <img className="charImg" src={image} alt="Character Image" />  
            </section>  
          </Link>  
        )  
      })  
    })}  
    </div>  
    </>  
  )  
}

Finally, we applied some styling. To be honest, the styling code is a bit scattered because we prioritised the logical aspects of the page. We incorporated a nice background picture and chose colours to resemble the aesthetics of the actual cartoon. We researched from the internet the colour palette from this cartoon so we will be able to use it.

**Challenges**

**Instructions**

*Challenges are great for showing your learning journey and problem solving, and this is a section that many engineers will check out. Every day of your engineering career you’ll encounter challenges, this is part of your growth and development. It’s the challenges you encounter that helps you become a stronger and more competent engineer.*

*Here you will detail any particular challenges you encountered as you were coding the project.*

*Questions to answer here:*

* *What technical challenges did you come across?*
* *Why were these challenges?*
* *What problem solving did you do to rectify them?*
* *Team dynamics/ Project management*
* *Tools/Tech you used*

**Insert your Challenges here:**

The most significant challenge we encountered was adapting to React. It was our first time using React for a project larger than a couple of documents, and navigating it without causing issues proved to be a learning curve. Debugging, in particular, was challenging.

During these two days of the project, I gained valuable insights from Antonio, who demonstrated a brilliant understanding of React. He patiently explained parts of the code that I initially found challenging.

The data structure from the API, while manageable, presented some difficulties. Retrieving all characters in a single request wasn't feasible since the API lacked an endpoint for it. Instead, we had to make individual requests for each page of characters (42 in total). Handling the promises from numerous requests added another layer of complexity.

**Wins**

**Instructions**

*The Wins section is your opportunity to highlight the aspects of your project you are most proud of. See this as your chance to showcase these parts of your projects to the engineers reading your ReadMes.*

*Things you could discuss here:*

* *Interesting problem solving you did*
* *Strong sections of code*
* *Collaboration with other team members*
* *Visual design of the project*

**Insert your Wins here:**

It was truly amazing to collaborate with Antonio; he proved to be an excellent partner for this project.

His calm and focused approach made problem-solving efficient, and he was always ready to assist in resolving any issues.

Each time our codes ran successfully, it felt like a victory. The project successfully showcased all the characters along with their respective episodes.

**Key Learnings/Takeaways**

**Instructions**

*This section is one of the other most important parts of your ReadMe from an engineers’ perspective and helps to differentiate each of you from your classmates and team members.*

*Engineers love to understand what you learn from each project and how it has shaped you as an engineer.*

*See this as your opportunity to show the engineers how your skills grew during each project sprint.*

*Things you could discuss here:*

* *What Technologies/Tools do you now feel more confident with? Tell them specifically what you learnt about these.*
* *What engineering processes did you become more comfortable with? Standups? Pair programming? Project management? Tell them what you learnt from these processes?*

**Insert your Key Learnings/Takeaways here:**

* Learned how to use REACT.
* Learned how to use Sass
* Learned how to use Router

**Bugs**

**Instructions**

*If you have any bugs in your project, it’s important that you flag them in your ReadMe. This helps the engineers reviewing your projects to understand that you are aware that there are issues - if you don’t flag these, then they won’t have that visibility that you know these problems are in your code and it can result in them not having a full understanding of your technical knowledge.*

*In either sentences or bullets, explain what the bugs are.*

*If you have no bugs, you can leave this section blank.*

**Insert your Bugs here:**

The most persistent bug that posed a challenge was the error caused by our initial struggle with handling promises. Once we figured out how to manage promises effectively, this issue was resolved. Additionally, we encountered bugs related to route misconfigurations and instances where we forgot to import essential files. While these were teachable moments, none of them were entirely unexpected, and we were able to address them systematically.

**Future Improvements**

**Instructions**

*It’s common to get to the end of your project and have ideas on what you would do if you have more time, as well as how you might improve it.*

*If you do, you should detail this here. It’s great to give that context on potential future improvements, to share your creative or technical ideas with the engineers reading your ReadMes.*

*In either sentences or bullets, explain what your future improvements would be.*

**Insert your Future Improvements here:**

What seems somewhat nonsensical at the moment is that the "All-characters" page has become redundant after developing the homepage. Given more time, we would eliminate all individual character pages in favour of an episode list.

However, the limitation lies in the API, which doesn't provide images or synopses for the episodes, offering only the character list, names, and episode numbers.

Despite these constraints, we believe there's still room for significant improvement and functionality in our project.