

# Application Deployment Tracking System

Team Members: Jackie Chow (CPE), Jeffrey Martinez (CPE)

Technical Directors: Daniel Jaquez, Gary Jutras



## PROJECT MOTIVATION

Successful web applications are built around the concept of continuous integration; the ability to modify existing features and add new features on-the-fly. Using a system that tracks the status of these changes is imperative to getting them in the hands of customers quicker. During a release of updated software there are several steps that require the input of many team members. This is because communicating what exactly needs to be accomplished in a coherent way is not trivial, which can result in delays on the completion of tasks. Delays in releasing have an impact on the ability of a company to react. Reaction time, given the right circumstances, can be costly. The main motivation for this project is to create a web application that would help to reduce the release cycle time of releasing updates to an application and therefore reduce the reaction time for the company.

## KEY ACCOMPLISHMENTS

### Research and Implementation of Test Driven Development:

To avoid having complex classes that involve using several overly complicated methods, following the Test Driven Development process will help reduce the amount of time fixing the code and prevent not being able to follow what the class does. The cycle will start with creating a test, run the tests so that it fails, and write code until it passes the tests. This also helps keep the classes simple and easier to manage.

### Research on Dependency Injection:

eMoney Advisor has certain coding standards that we needed to follow throughout developing the application. One of the biggest coding standards that we have been exposed to was dependency injection. Dependency injection is a technique whereby one object (or static method) supplies the dependencies of another object. In other words, dependency injection is like the middleman that separates two objects from each other in order to make them independent.

### Displayed Build Data using React and Pagination:

Using the information from the mock server, we needed to display that data onto a table. Since there was already code that gets the data from the server, using React, we then placed it all onto a table. The data that was displayed, included the status, build number, duration, start time, and end time. To make sure only 20 builds were being displayed at a time, a pagination class would keep track of what was displayed and only 20 builds would be shown at a time.

### Displaying build information as Gantt Chart:

After being able to display build information in the form of a table, the final step is the transfer that data onto a gantt chart. With the gantt chart, it would help visually show the overall status of a build, which can help project members plan according to what the current status of the build is. To do this, we used Google React Charts to display the primary build details.

### Attempted to Finish JumpCloud Login Configuration:

In the beginning of the semester, login functionality for our application seemed to work on one of our base applications but not on the other. The other base application is where we wanted to be able to login but was not possible. We then attempted to figure out what the problem was, but could not due to the fact that the SustainSys package library we were using seemed to be incomplete.

### Fetches Basic and Detailed Data From a Multiple Mock API Endpoint:

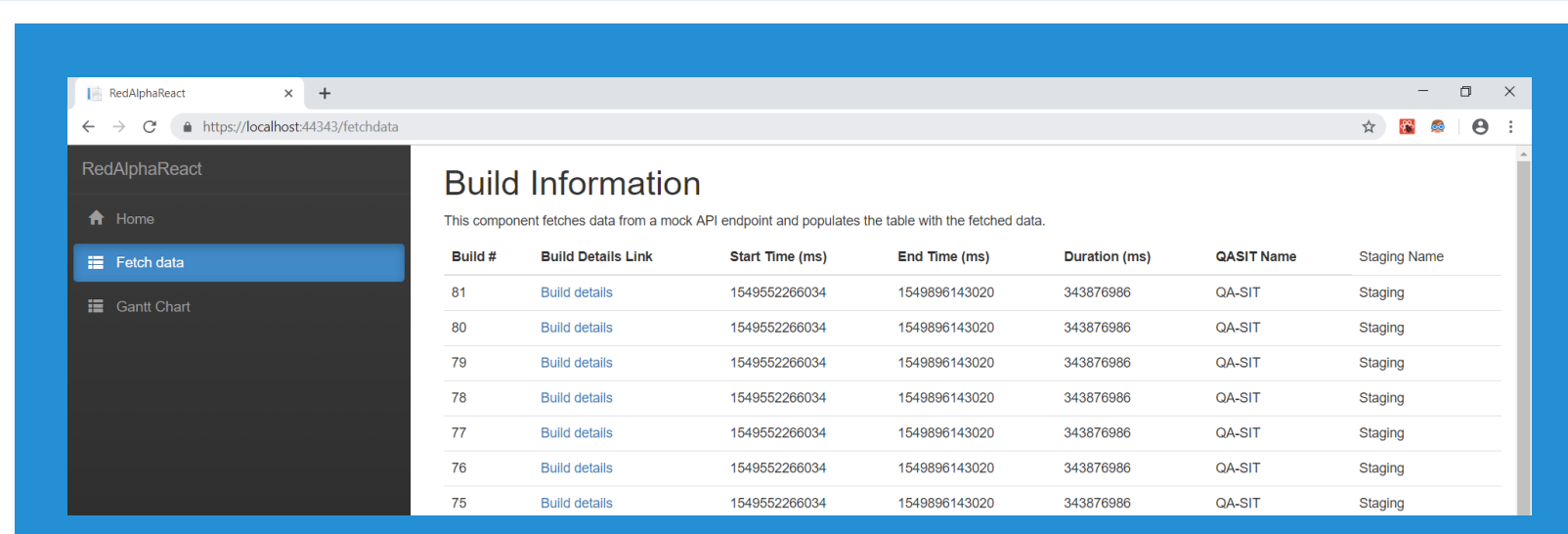
We implemented the retrieval of data that we wanted using RestSharp and mock API endpoints. These were created using Postman, an API development environment. Once the application receives the data, it is then deserialized so that we would be able to use it in our application. This was all done using Newtonsoft. Newtonsoft is a Json library that helps .NET work with JSON objects and JSON data.

### Update Existing Code to meet eMoney Advisor's Coding Standards:

The initial fetching code that fetched the data from the server was not up to par with eMoney's coding standards. To fix this, we had to go back and edit it in order to create something sustainable and scalable. In order to do that, we had to extract most things that were done into their own separate methods and follow the dependency injection technique.

### Wrote Appropriate Unit Tests:

In order to get a better sense of software development, we had to create unit tests for our code. Most companies use unit tests in order to create scalable projects and make it easier to detect if something was malfunctioning as soon as the new code is updated.



Build #	Build Details Link	Start Time (ms)	End Time (ms)	Duration (ms)	QA/ST Name	Staging Name
81	Build details	154850296034	1548506143020	343070996	QA-ST	Staging
80	Build details	154850296034	1548506143020	343070996	QA-ST	Staging
79	Build details	154850296034	1548506143020	343070996	QA-ST	Staging
78	Build details	154850296034	1548506143020	343070996	QA-ST	Staging
77	Build details	154850296034	1548506143020	343070996	QA-ST	Staging
76	Build details	154850296034	1548506143020	343070996	QA-ST	Staging
75	Build details	154850296034	1548506143020	343070996	QA-ST	Staging

Figure 4. Build information table. This table displays the build information that was retrieved from an eMoney server. The URL endpoints have been redacted.

## ANTICIPATED BEST OUTCOME

The best anticipated outcome is to develop a web application that displays build information. The data will be received from a server, which contains the build number, the status, start time, end time and the overall duration of the build. This information will be presented to the users, as a Gantt Chart. The chart illustrates the time schedule of each build and has it displayed as a type of bar chart. With the Gantt Chart highlighting the duration of each build, this will help project members plan accordingly depending on what has to be achieved.

## PROJECT OUTCOME

The Anticipated Best Outcome was achieved.

## FIGURES

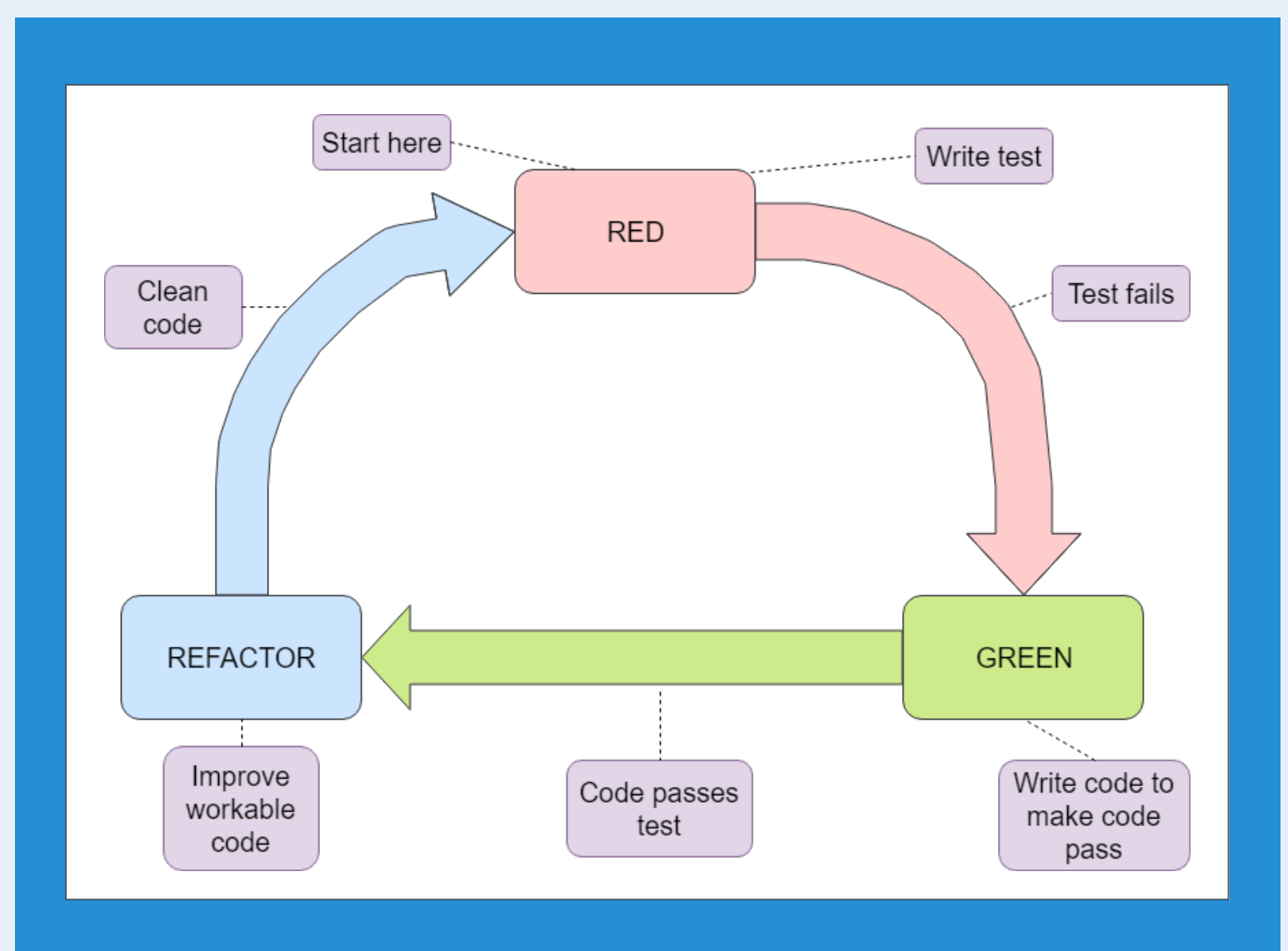


Figure 1. Test driven development (TDD) workflow. TDD is important to create scalable clean code.

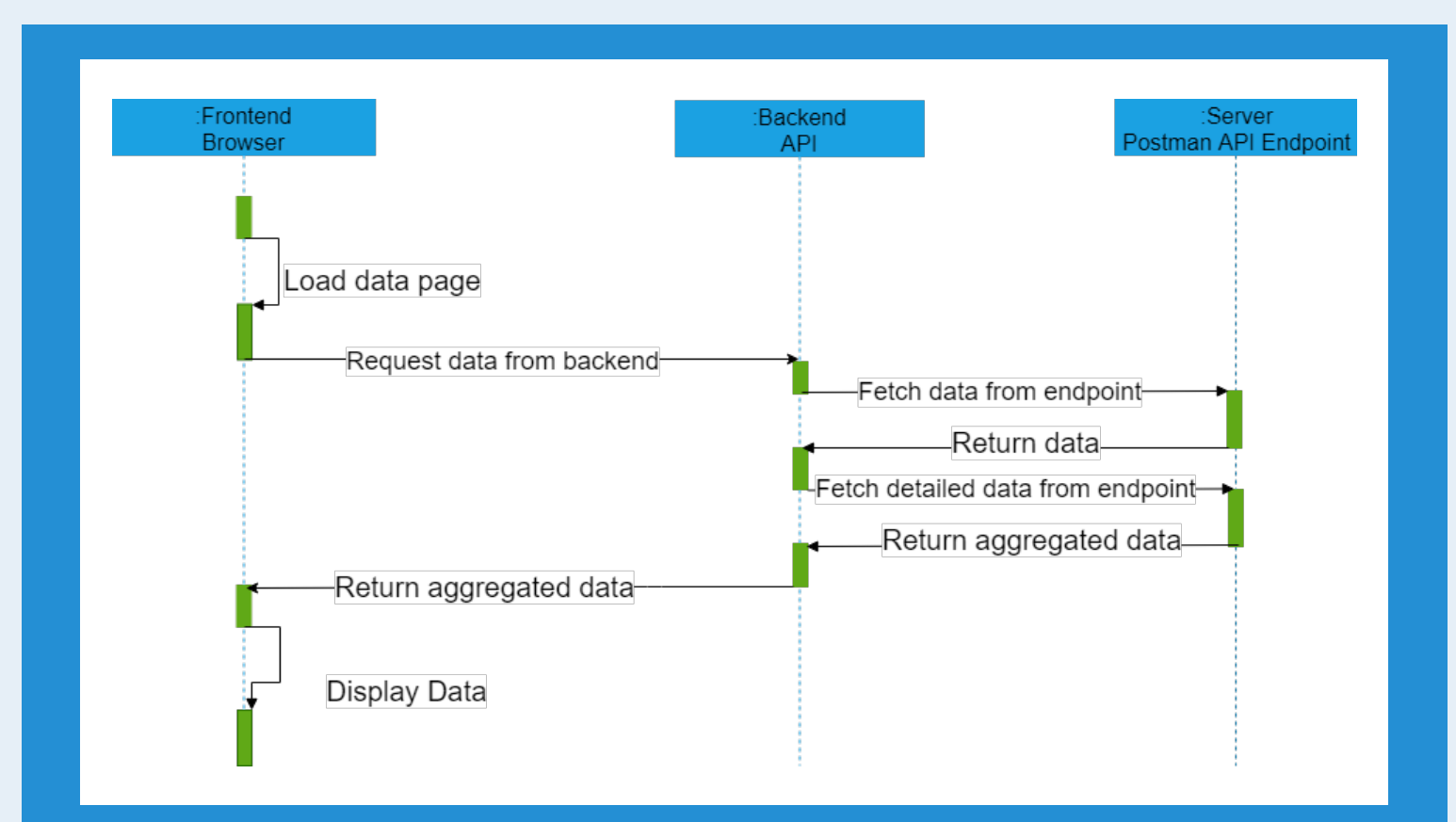


Figure 2. Developed application workflow. This diagram highlights what processes the application goes through in order to function.

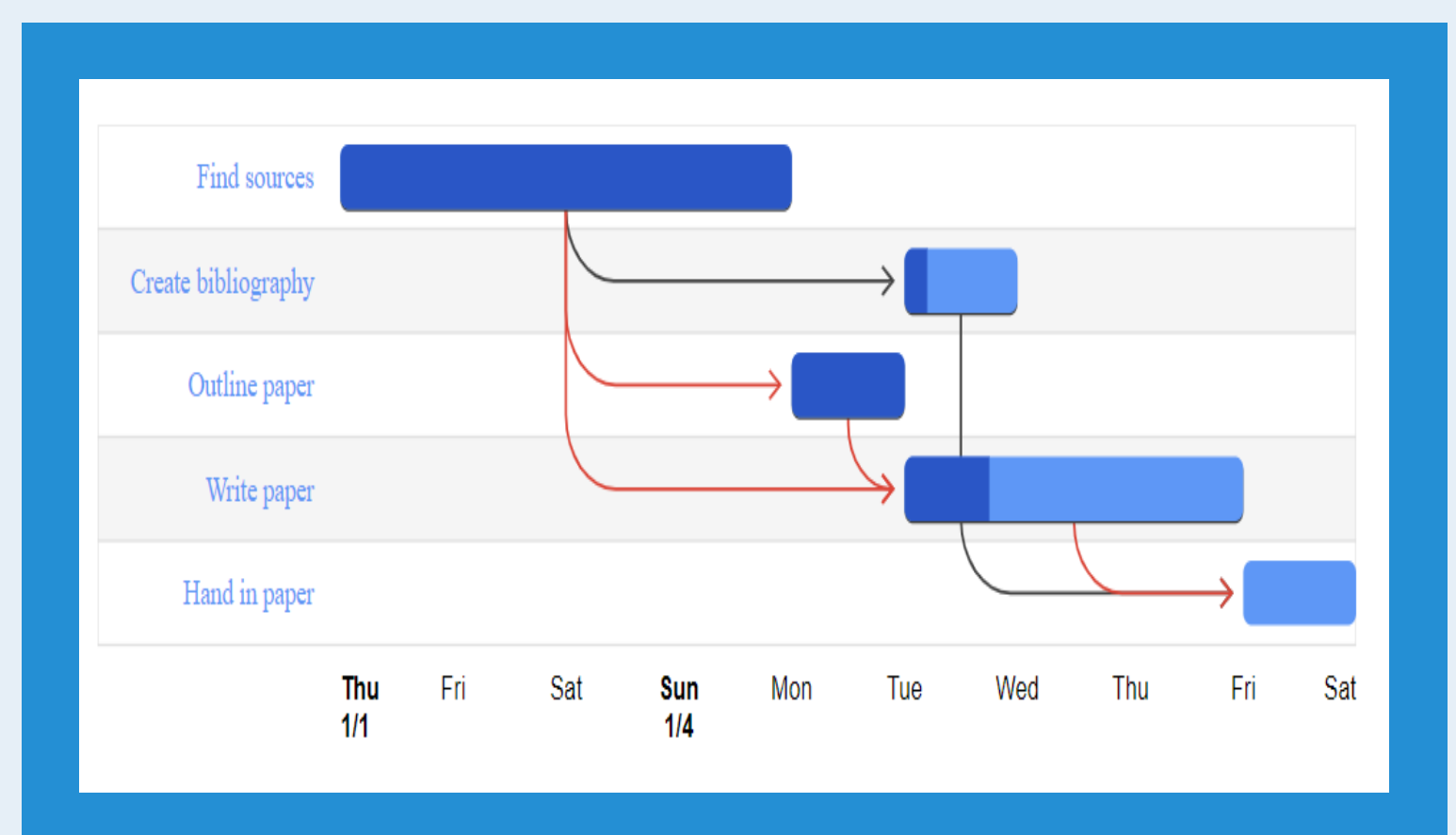


Figure 3. Gantt chart example. This example gantt chart is what our final gantt chart will look like. It displays time information in a clean and readable way.