National University of Singapore
School of Computing
Computing for Voluntary Welfare Organisations (CVWO) AY2020/21

Assignment: **Riding on Rails**

Issue date: 5th December 2020
Due date: **30th December 2020 (mid-assignment submission)**
25th January 2021 (final submission)

1 Overview

In this assignment, you will learn about the programming skills required for CVWO's summer projects and go through a website development cycle. The focus of this assignment is to promote proper code structure, coding techniques and familiarity with various development tools. You are expected to build up a good web development foundation through this assignment, and prepare yourself for a development cycle during your stint over the summer.

Let's begin...

Reminder:	Please read the entire assignment before starting.

Aside from web development foundations, this assignment mainly aims to familiarise you with Ruby on Rails, an open-source Ruby framework with strong MVC patterns. As CVWO uses Rails as its main framework, it is important to have a good understanding of this framework in order to work effectively on the projects.

Tip:	If you do not know what MVC is, it is strongly recommended that you do your own reading before attempting this assignment. During the course of CVWO and beyond, **independent learning is the key** to staying ahead.

2 What you need to know

> *"Babies are always more trouble than you thought – and more wonderful."*
>
> —Charles Osgood

The following sections are provided to aid you in your assignment. They include modern programming languages and technologies that are widely used in the industry.

2.1 Ruby on Rails

> *"Ruby on Rails® is an open-source web framework that's optimized for programmer happiness and sustainable productivity. It lets you write beautiful code by favoring convention over configuration."*
>
> —From the homepage of Rails

2.1.1 Introduction

Ruby on Rails is a free and open-source web framework. As the name suggests, this framework is written in Ruby. Notice that at this point, we have not explicitly asked you to learn the language. This is because the language is relatively easy to pick up, and you should be able to get a sufficient grip on the language by yourself.

Rails abstracts and simplifies common repetitive tasks by eliminating the need to write boilerplate code. There are also many auxiliary libraries that we can use to speed up the development process. Furthermore, it is also relatively easy to deploy as many hosting service providers support it. Hence, we have chosen to utilise Rails for its suitability in carrying our projects further.

Rails usually ships with a default web server (Puma) and database adapter (SQLite), which should be sufficient for this assignment. We do not impose any restrictions on the choice of either for the purpose of this assignment.

Tip:	Although not needed to complete the assignment as you could learn by trial and error, learning Ruby before attempting the assignment would make your life much easier. Unlike other frameworks such as PHP, Ruby is much easier to pick up although there are some gotchas that you may or may not run into during the assignment. There are many good online resources available.

Although having a more in-depth lesson on Rails would be appropriate here, staying true to one of Rails' major philosophies, DRY, we shall leave this task to a more established resource from the community, the Rails Guide http://guides.rubyonrails.org/. The following chapters are compulsory reading, and they would help greatly in the assignment given later.

- **Chapter 1**: Getting Started with Rails
  http://guides.rubyonrails.org/getting_started.html
- **Chapter 2**: Active Record Basics
  http://guides.rubyonrails.org/active_record_basics.html
- **Chapter 3**: Active Record Migrations
  http://guides.rubyonrails.org/active_record_migrations.html
- **Chapter 4**: Active Record Validations
  http://guides.rubyonrails.org/active_record_validations.html
- **Chapter 5**: Active Record Callbacks
  http://guides.rubyonrails.org/active_record_callbacks.html
- **Chapter 6**: Active Record Associations
  http://guides.rubyonrails.org/association_basics.html
- **Chapter 7**: Active Record Query Interface
  http://guides.rubyonrails.org/active_record_querying.html
- **Chapter 8**: Layouts and Rendering in Rails
  http://guides.rubyonrails.org/layouts_and_rendering.html
- **Chapter 9**: Form Helpers
  http://guides.rubyonrails.org/form_helpers.html
- **Chapter 10**: Action Controller Overview
  http://guides.rubyonrails.org/action_controller_overview.html
- **Chapter 11**: Rails Routing from the Outside In
  http://guides.rubyonrails.org/routing.html

2.2 Relational Database

Suggestion:	Although you will not likely be writing any SQL for this assignment, it is important to have at least a basic understanding of databases.

2.2.1 An overview of relational databases

Relational databases are databases that store data in tables, each containing a number of columns and rows. It is called ”relational” because tables are linked to other tables through foreign keys. For example, the Singapore Government's address database may use the NRIC/FIN of individuals to link addresses in the `addresses` table to names in the `names` table. In this case, the column containing NRIC/FIN in the `addresses` table is called the foreign key as it references the primary key in the `names` table.

In this assignment, we will be going through fundamental database concepts. Usually, an application will use one database, with several applications sharing the same database server. For instance, if you and a friend each had a blog, each blog needs one database, but the same server could have two databases defined, one for each blog.

At the highest level, a database contains a schema. A schema is a blueprint of your tables, containing their structures and relationships.

Tables can contain one or more columns, each defining an attribute that requires storage. (For example, a `students` table containing students' personal data might contain five columns: `matric_no`, `name`, `address`, `phone`, `date_of_birth`. In MySQL, columns have types: in this case, name is of type text (`varchar(255)`) and `date_of_birth` is of type `datetime`).

Keywords to look up: SQL, primary key, foreign key, Entity-Relationship Diagram. Try to get the overview before diving into details.

Actual information is stored as rows in a table. Each row must be uniquely identifiable; if two rows cannot be distinguished from each other, then there is no way to change the content or delete one of the rows. Therefore, it is common practice to dedicate one column that is guaranteed to be unique for each row. We call such a column a primary key. In this instance, the `matric_no` is probably a good choice to be the primary key as it uniquely identifies a row, since no two students can share a matric number. You will be prevented from inserting a new row when another row with the same primary key already exists in the table.

Relations indicate relationships between two tables. For example, the `home_faculties` table may contain two columns: `matric_no` and `faculty`, indicating a mapping from a student to her faculty. We can link this to the `students` table by using `matric_no`. As discussed earlier, `matric_no` in the `home_faculties` table is called the foreign key. The foreign key must map into the primary key column set of another table. Note also that students may become members of two faculties when doing double degrees.

What happens if you need to relate two tables together to retrieve an attribute? For instance, you might have a student in the double-degree programme and you need to retrieve his CAP for each of his faculties.

Other important concepts include *indexed columns* (where searching within the column is fast, at the expense of increased time required for modification), *unique keys* (which enforce uniqueness among values for non-primary key columns), and relation cascading (where deleting a row from a table can automatically update/delete all related entries in tables which reference this key).

After this section, you should be ready to produce a schema for your application. Consider how efficient your schema will be: How complex will it be to access the most commonly accessed data? Think about the number of queries and the number of tables accessed to complete a single user query. Your schema should be graphical,

This is called an Entity-Relationship Diagram.

with the table names, column names/types, primary keys, and relationships clearly indicated.

Aspiration:	Draw your application's database schema

2.3 HTML, CSS & JavaScript

2.3.1 HTML and CSS

HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are two of the core technologies for building Web pages. HTML provides the structure of the page, while CSS provides the (visual and aural) layout. Along with graphics and scripting, HTML and CSS are the basis of building Web pages and Web applications.

Below is a list of HTML tags you will frequently be using.

```
<a>,<body>,<br>,<button>,<div>,<em>,<form>,
<head>,<h1> - <h6>,<hr>,<html>,<img>,
<input>,<label>,<legend>,<li>,<option>,<p>,<script>,
<strong>,<style>,<table>,<tbody>,<td>,<textarea>,<tfoot>,<th>,
<thead>,<title>,<tr>,<ul>.
```

Please go through the tutorial from HTML Dog. You can experiment with various HTML tags on JSBin. We will not formally introduce CSS as it is better to learn from practice. Later sections will cover CSS briefly as the need arises.

2.3.2 JavaScript

JavaScript is a scripting language used to create and control dynamic website content, i.e. anything that moves, refreshes, or otherwise changes HTML or CSS elements on your screen without requiring you to manually reload a web page.

If you did not take CS1101S: Programming Methodology, Eloquent JavaScript is a good starting point. You are not expected to master JavaScript; however, you must demonstrate basic understanding. If you already know basic JavaScript syntax, you might like to start from Chapter 11 - Web programming: A crash course. Alternatively, you can refer to existing JavaScript examples on the web, and use the book to clarify any doubts.

Since JavaScript is used in React and React is an integral part of this project as well as CVWO projects, well-written JavaScript is very important for CVWO.

Suggestion: Modern browsers all come with a debugging inspector for HTML, CSS and JavaScript. You should decide on which browser you will be primarily debugging and testing with. However, note that there are compatibility differences between browsers, and you **must** test your application using different browsers. For CVWO, we usually recommend VWOs to use Firefox or Chrome; however, certain staff members still prefer to use Internet Explorer, in which case we try to ensure that they are using Internet Explorer 10 or above.

2.4 React

React is a JavaScript library for building user interfaces. It is the view layer for web applications.

At the heart of all React applications are components. A component is a self-contained module that renders some output. We can write interface elements like a button or an input field as a React component. Components are composable. A component might include one or more other components in its output.

Broadly speaking, to write React apps, we write React components that correspond to various interface elements. We then organise these components inside higher-level components which define the structure of our application.

For example, take a form. A form might consist of many interface elements, like input fields, labels, or buttons. Each element inside the form can be written as a React component. We would then write a higher-level component, the form component itself. The form component would specify the structure of the form and include each of these interface elements inside of it.

Importantly, each component in a React app abides by strict data management principles. Complex, interactive user interfaces often involve complex data and application state. The surface area of React is limited and aimed at giving us the tools to be able to anticipate how our application will look with a given set of circumstances. We dig into these principles later in the course.

You may refer to this React tutorial to help you understand how to use React.

2.5 Git

It might be your first time developing any piece of code which is non-trivial to write and maintain. As such, you might like to store copies of your work over time. This is called versioning. An additional function software engineering teams require is the ability to share code as it is being written. One such software used by software engineers (and CVWO) to solve both problems is Git. Git is however far more able than this and more information can be found on the Wikipedia article.

It will save you time and effort to use a versioning software since you will most likely run into bugs as you change (supposedly unrelated) pieces of code in your application.

2.6 Tools

Various tools can be used for web application development. In all cases, you should always find one that suits your working style. As such, there is no one answer to "which is the best IDE to use". Nonetheless, strictly speaking, any text editor will suffice. However, the following software is used by some of our CVWO members:

- Visual Studio Code
- RubyMine: Note that if you are a student, you are eligible for a free copy of the paid version of the app. See here.
- Sublime Text
- Vim
- AWS Cloud9 IDE

2.7 Best Practices

> *"If you give a hacker a new toy, the first thing he'll do is take it apart to figure out how it works."*
>
> —Jamie Zawinski

2.7.1 Abstraction and Modularisation

People who are new to web development usually end up with an unreadable mess. As such, as a project grows in size and scope, you may find it increasingly difficult to maintain and add new features. This is undesirable, especially for large projects.

The key insight here is to abstract common patterns away and reuse them where necessary.

For instance, you may decide to use the same headers and footers for every page in your app. These blocks can be abstracted into separate files and included where required instead of copying and pasting them everywhere. A logical structuring of your files helps increase its maintainability and encourages future code reuse. You may want to take a look at some websites and consider how you could incorporate abstraction.

2.7.2 Documenting Your Work

Programming is all about communicating computational processes. This means that your code should be clear and easily understood. Remember that code is read many more times than it is written!

Help others comprehend your code by including clear comments. However, having good comments does not excuse writing bad code. Consistently high quality of maintainable code is expected of you so that subsequent batches of students are able to work on your project. Some tips:

- Give sensible names to variables and functions so that their contents and goals are easily understood.
- Use standard algorithms where available. Mark modified algorithms as such.
- Use the standard library of the language you are using. JavaScript and Rails come with a large standard library that accomplishes many things. If the functionality you require is not in the standard library, find a well-maintained third-party library that accomplishes what you need. Write your own implementation only as a last resort.
- Don't over clutter your code with unnecessary comments. That means you don't have to comment every single line. Comments are excessive when it's blatantly obvious what the code does. A comment on top of each function describing what it does (often) makes your code more comprehensible than multiple in-line comments.

3 Assignment

3.1 Installation & Setup

Follow the installation and setup guide for React on Rails at the GitHub Repository below.

Installation Guide:	`https://github.com/CVWO/setup-guide`

There is a good chance that you will encounter technical problems during the installation process. If so, feel free to browse or post on the forums at this link, as other applicants might be facing the same issues.

Please **do not** contact us for technical issues. Debugging is a painful process that all programmers have to go through, but it is also a good learning opportunity.

3.2 Task

3.2.1 Task 1 - Managing Tasks

Implement a todo manager to keep track of all the tasks that you have to do. As a start, your todo manager should be able to perform basic CRUD operations. CRUD is an acronym for Create, Read, Update and Delete. They are the basic functionality that your application should have.

3.2.2 Task 2 - Categorising Tasks

Implement a tagging system to organise your tasks so that they are easy to search for. Since we are all busy people, your todo manager would likely fill up fast. Hence, it is important to be able to organise your tasks and search through them.

3.2.3 Optional Tasks

You are free to implement other features to improve the overall user experience, e.g. hosting. Extra credit might be awarded, depending on the completeness of your implementation. Refer to Section 4 for a non-exhaustive list of optional tasks.

If you are unsure of what else to add, consider the objective of this application (apart from pedagogy!). Think of various use cases that users might want to use your software for. What else might such a user want?

3.3 Requirements

3.3.1 Rails

Your backend code **must** be written using Ruby on Rails. For this assignment, you are allowed to use *any* gem (Ruby plugin) to make your life easier.

3.3.2 React

Your frontend code **must** be written in React. For styling, you may use existing UI libraries (e.g. Material UI, Semantic UI, etc.) or alternatively, design your web pages from scratch using CSS (Refer to Section 2).

3.3.3 Git

Versioning your assignment using git is **compulsory**, and submissions must be made either through GitHub or GitLab.

3.4 Submission

There are **two** submissions for this assignment – the mid-assignment submission (by 30th December 2020) and the final submission (by 25th January 2021).

To submit your assignment, create a Git repository (either on GitHub or GitLab) and email cvwo.assignment@gmail.com the link to your repository. We will clone and grade your code from there. Please use the same repository for both submissions.

You must push your deliverables to the repository by the given deadlines. Work-in-progress code is acceptable for the mid-assignment submission.

For the mid-assignment submission, please provide the following:

1. A `README` file containing your name and matriculation number.
2. A screenshot showing that you have successfully installed Rails. Just the default startup screen will do.
3. A short write-up on the basic use cases and your execution plan. You can also write down your suggestions and tell us the problems you are currently facing. This write-up must be no longer than 3 A4 sides, with a 12pt. Georgia or equivalent font face. Please submit in pdf format.

For the final submission, please provide the following:

1. A `README` file containing your name and matriculation number.
2. Source code: maintain the default file structure of the Rails application as much as possible.
3. A short write-up on what you feel about your accomplishments in this assignment as well as a short user manual. This write-up must be no longer than 3 A4 sides, with a 12pt. Georgia or equivalent font-face. Please submit in pdf format.
4. Proof of working application: It is highly desirable to have a working copy accessible from the Internet. Alternatively, submit a database dump with your source code.

3.5 Grading Scheme

We grade your assignments based on how much effort you have put into the assignment. The key point of this assignment is for you to demonstrate what you have learned in this process. The grading is flexible. This grade will be considered during the application process for CVWO.

We will also be running **plagiarism checks** on your submissions. We hope that this assignment will serve as a learning experience for everyone, so please do not copy code either from your peers or from past submissions.

4 Optionals

> *"For a long time it puzzled me how something so expensive, so leading edge, could be so useless, and then it occurred to me that a computer is a stupid machine with the ability to do incredibly smart things, while computer programmers are smart people with the ability to do incredibly stupid things. They are, in short, a perfect match."*
>
> —Bill Bryson

4.1 Cron

Difficulty: 2/5

Cron is a time-based job scheduler in Unix-like computer operating systems. It enables users to schedule jobs (commands or shell scripts) to run automatically at a certain time or date. It is commonly used to automate system maintenance or administration, though its general-purpose nature means that it can be used for other purposes, such as connecting to the Internet and downloading email.

Set a Cron Job under Linux. This example is to create a cron job to automatically backup the database.

Inside terminal:

```
> sudo crontab -e

// START THE TOOL TO SET THE CRON JOB

// EVERY DAY RUN THE BACKUP SCRIPT

0 0 * * * bash backup.sh
```

Inside `backup.sh`

```
cd /root/autobackups

mysqldump -u root -DatabasePassword DatabaseName > FileName
```

You are encouraged to find novel uses of Cron for your application. Backing up your database daily is merely one of the endless possibilities that Cron allows you to implement.

4.2 TypeScript

Difficulty: 3/5 — *Highly Recommended*

TypeScript is a superset of JavaScript that adds optional static typing.

Static typing greatly reduces the frequency of type-related errors. Its verbosity also improves the maintainability and scalability, especially for large projects.

We encourage adding TypeScript to your project. It may be easier to start with a TypeScript project than to introduce it later. This guide provides a good starting point for using TypeScript in your project.

4.3 RESTful APIs

Difficulty: 3/5

Representational State Transfer, REST in short, is an architecture for networked applications. Being one of the simplest architectures to deploy, it is a popular choice for many web services. It is a client-server architecture where a client initiates a request to the server to be processed, and receives a response with updated data. The Application Programming Interface or API defines the specifications of this communication. In the context of this assignment, the server will be our Ruby on Rails application, while the client will be our React application.

Requests and responses can take many forms - JSON is the most common format. JavaScript Object Notation or JSON is a text-based data-interchange format. It is already supported by all modern browsers and server-side scripting languages so you do not need to implement it yourself, but if you want to know more, you can visit this website. You may also want to read up on HTTP Messages, which will be how the JSON data are exchanged.

Note that this is just one of the many ways the React frontend can interface with the Ruby on Rails backend. If you wish to try this out, do check out this guide for Ruby on Rails and this guide for React.

4.4 Hosting

Difficulty: 4/5 — *Highly Recommended*

During the development phase, you will likely be running and hosting your application on your local machine, and therefore only you will be able to access it. Deployment means making the application available for use to the public. To do so, you need a server to build and run your application and to accept client requests. In addition, a domain name is necessary so that people can access your application through a URL.

For this assignment, you are encouraged to host your final product on **Heroku**. Heroku is a platform as a service (**PaaS**) which provides an easy-to-use platform to deploy a web application. After a proper setup, you can deploy your application simply by `git push heroku master`. This tutorial teaches you how to set up a Rails project on Heroku.

You need to register a Heroku account first, which requires your credit card information. For this assignment, the free tier should be more than sufficient.

If you developed the React frontend as a separate application from the Rails backend, e.g. from following a RESTful API architecture, you can also deploy the React application on **Netlify**. Netlify is similarly a PaaS that helps to deploy frontend web

applications quickly. You can use your GitHub account to register for a Netlify account. This guide runs through how you can deploy a React project on Netlify.

Alternatively, you can use other cloud services such as DigitalOcean or AWS Elastic Beanstalk if you wish.

4.5 Redux

Difficulty: 5/5

Redux is an open-source JavaScript library for managing application state and is commonly used with React.

Redux uses what is known as **Flux Architecture** to provide a ”single source of truth” for the state of an application. This is especially useful for large projects with many components, as the state of an application can quickly become decentralised and disorganised.

While it is likely not necessary given the scale of this assignment, it is a good exercise to improve your understanding of state management in React.

If you want to give it a go, this guide is a good place to get started.

4.6 Docker

Difficulty: 5/5

Docker is a tool designed to package software into standardised **Containers**.

Containers are similar to virtual machines, but unlike a virtual machine, they do not create a whole virtual operating system. Rather, they allow applications to use the same Linux kernel as the system that they're running on but in an isolated environment.

Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another. This could be from a developer's laptop to a test environment, from a staging environment into production, and perhaps from a physical machine in a data centre to a virtual machine in a private or public cloud.

As Docker requires some understanding of operating systems and networks, it can be challenging to grasp the concept behind it. Nevertheless, it is a useful skill to pick up if you want to become a more competent programmer. If you are up for the challenge, do check out this guide to get started.

The end:	Good luck and have fun!