In this challenge and the next, we will be learning how to choose a suitable gem to set up a user management system. Let's get started!

What are gems?

A gem is a library for Ruby and Rails. It's like a collaborative effort from programming communities around the world where code are packaged into libraries that encapsulate common functionalities for apps. Gems are great because they make us more productive by not having to code everything from scratch. In fact, Rails is also a Gem! When you become more advanced, you can also make your own gems and open-source them for the world to use!

You will notice that when you create a Rails app, a list of default gems is included in the Gemfile. Take a look now at the default gems that have been installed in your rails app, what do you think these gems do?

Choosing a gem

Working with gems can speed up building your web app, but could also be a headache to customize down the line if you don't understand it. Be sure to understand if a gem meets your needs before deciding to incorporate it into your project! However, there are so many gems out there that sometimes, choosing a gem can be a problem. Before we use any gem, we should consider the following questions:

- **WHAT** is the gem we're going to use?
- **WHY** should we use this gem? Does it solve my problem / feature needs?
- **HOW** to use it?
- **WHEN** to use it?

For user management, there are 2 popular gems:

1 Devise

2 Clearance

By looking into these gems' documentations, which you should, we can deduce that:

Gem Type	Devise
WHAT	Heavy duty user management engine
WHY	For full-scale user management deployment
HOW	Include all dependency, then integrate
WHEN	For moderate and expert use

Now, ask among peers and everyone around. Which gem will you go for and why?

The end goal for this section is to think through and exchange opinions to make a final decision on choosing what gems to use.

Study Gem's Health

Between a gem that is currently supported by the community and a gem that was left out years ago, I'm pretty sure you'll choose the former.

Hence, while selecting a gem, always check for their last **commit** date and time. You can source it from the gem's repository header, example:

There are more factors to consider during gem selection, such as test supports, the line of codes being introduced, etc.

NOTE: Devise introduces about 3k lines of new code.

For now, figuring out “WHAT, WHY, HOW, and WHEN” and support timeline would be your main consideration factors.

For this tutorial, we’re going to use Clearance, since we’re building a lightweight app.

Create A Feature Branch

Recalling your git challenge, **a good developer NEVER works on master branch**. This is because the master branch is a lifeline for your app and your customers are using it. Breaking anything can be very nasty and chances are, you’re losing customers.

Another reason is to avoid unnecessary development conflicts when you work with a team of developer.

Hence, a good agile practice is by opening a feature branch to perform development. Not to worry, any new changes inside a feature branch will not affect the master branch.

1) Create a feature branch using the `git checkout` command:

```
$ git checkout -b feature/clearance
```

This way, you’re now ready to work on your feature branch.

2) You can use `git status` to check your current branch status. Notice `On branch ...`.

```
$ git status
On branch feature/clearance
Your branch is up-to-date with 'origin/feature/clearance'.


nothing to commit, working directory clean
$
```

Reading Gem’s Documentation and Integrate

Based on Clearance gem readme file, you’ll need to learn how to read the documentation. We are told to do these:

1. Add the Clearance gem.
2. Install the gem into our app.
3. Generate Clearance routes for customization.
4. Generate Clearance views for frontend customization.

TIPS:

1. Work in pairs among your peer and share your knowledge with your partner.
2. After you performed the above, how can you test the output?
3. Discuss with mentors when you and your partner are lost in the documentation.

Raising Pull Request and Merge to Master

Once you’re done with testing, it’s time to push to your remote feature branch.

At this stage, it’s sufficient enough to commit under single commit as: “install Clearance gem for user management”.

To do this, we do the following git sequence:

1) We’ll need to add the files using `git add .`.

```
$ git add .
```

2) Reviews the file that you’ve added into the system using `git status`:

```
$ git status
On branch feature/clearance
Your branch is up-to-date with 'origin/feature/clearance'.


Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)

modified:   Gemfile
modified:   Gemfile.lock
modified:   app/controllers/application_controller.rb
new file:   app/models/user.rb
new file:   app/views/clearance_mailer/change_password.html.erb
new file:   app/views/clearance_mailer/change_password.text.erb
modified:   app/views/layouts/application.html.erb
new file:   app/views/passwords/create.html.erb
new file:   app/views/passwords/edit.html.erb
new file:   app/views/passwords/new.html.erb
new file:   app/views/sessions/_form.html.erb
new file:   app/views/sessions/new.html.erb
new file:   app/views/users/_form.html.erb
new file:   app/views/users/new.html.erb
new file:   config/initializers/clearance.rb
new file:   config/locales/clearance.en.yml
modified:   config/routes.rb
new file:   db/migrate/20160203065546_create_users.rb
new file:   db/schema.rb
```

3) Then we use git commit to commit those changes.

```
$ git commit -m "install Clearance gem for user management"
```

4). Lastly, use git status to check which branch are you in. This is to make sure we are not pushing our changes into the wrong branch.

```
$ git status
On branch feature/clearance
Your branch is ahead of 'origin/feature/clearance' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working directory clean
$
```

5) After you have confirmed that you’re on the correct branch, you can push it to your feature branch using git push.

```
$ git push -u origin feature/clearance
```

Recall what the -u option means from what you have learnt previously.

6) All right, now we should proceed with raising pull request for merging feature branch to master branch. During this process, you can perform a code review session among your peers and mentors.

Github prepared a good documentation for you to kick start.

NOTE:

Among your peers, discuss about:

1. What and how can I perform a code review?
2. What can I do if I need to do some fixes?
3. How can I amend the fixes?
4. When to merge the pull request?