Portfolio Work

For the portfolio you will be creating a REST API for a Job Advertising system.

Part of the work has been completed during the learning sessions, and includes the Browse and Read components of the API for Regions, Subregions, Countries, States and Cities.

Also, there is a basic API developed for User Management.

At this time, no security has been added to the system.

Before securing the API we require the core API features to be developed and tested.

Before Commencing

Before commencing you will need to do the following:

- Create a new empty private GitHUb (or equivalent) repository
  - Ensure that the repository does NOT have a ReadMe.md, .gitignore, License or any other files.
  - Details are shown in the Version Control Requirements section.
- Create a new GitHub Project
  - The style will be a basic "Kanban".
  - Each feature and sub-feature should be added as issues (see Portfolio Planning & Documentation).
- Create a new Laravel application
  - Ensure the application is named the same as your repository name.
  - Update the default `.gitignore` as required to make sure that backup files, IDE files and other items are not included in the repository.
  - Duplicate the `.env` file and name the copy `.env.dev`, add this to your version control.

Portfolio Planning & Documentation

Before starting code, you should plan the required steps from this project as GitHub (or equivalent) issues.

Before starting this go through the notes on GitHub Project Management.

Once completed the notes, you will:

- Create a Kanban board to track your work.
- Create an automation that adds new issues to the project.
- Plan the portfolio tasks by adding GitHub issues.
- Move all tasks to the To Do column.
- Determine the next three tasks to be completed and move into the Up Next column.

Version Control Requirements

All code must be version controlled and placed into a **PRIVATE** repository on GitHub or a similar remote system. Access to this repository **must** be given to the assessors and lecturers.

You must create a new *empty* **private** repository on GitHub (or equivalent) for this work. The repository will use the naming structure:

```
XXX-SaaS-BED-Portfolio
```

Replace XXX with your initials.

If you are repeating the cluster, then you must use the following format for your repository name:

```
XXX-SaaS-BED-Portfolio-YY-SN
```

Where `YY` is the two digit year and `N` is the semester of study.

Commit Messages

Commit messages **must** use the conventional commit style. This is outlined below:

Type of commit	Prefix	Example	Notes
Start of project	init	init: Start of Project	
Feature work	feat	feat: Add User create method	
Feature with identifier	feat(...)	feat(user): Add create method	**preferred**
Bug fix	fix(...)	fix(user): Fix issue #1234	
Documentation	docs(...)	docs(user): Update Scribe documentation	**preferred**

Other conventional commit message types are available, and you are directed to Conventional Commits for more guidance and examples.

Note that conventional commits allow for multiple line comments. For example:

```
feat(user): Update browse API

- Add pagination to user API
- Add example use to API docs

close #1234
```

You may also link commits to your issues, and automatically close them by using the following keywords and syntax:

- `fix #xxx`
- `close #xxx`
- `resolve #xxx`

We suggest using `close` when completing a new (sub-)feature and `resolve` when completing a bug-fix.

Further useful details:

- Conventional Commits Cheatsheet (github.com)
- Closing Issues via Commit Messages - The GitHub Blog
- Linking a pull request to an issue - GitHub Docs
- Quickstart for GitHub Issues - GitHub Docs

Requirements for ALL Features

- All data MUST be validated
- Correct HTTP Responses will be given (200 OK, 201 Created, 404 Not Found, 403 Forbidden, etc)
- A standardised response structure will be used:

  ```
  {
    "success": true|false,
    "message": "Some form of message",
    "data": {
      ...
    }
  }
  ```

Features

You are required to develop the following initial feature sets:

- Companies
- Positions

There are pre-requisite features that include, but may not be limited to:

- Regions
- Sub-Regions
- Countries
- Cities

More details will be given in the following sections, and may include additional work to previously implemented features.

In the Positions and Companies features, you may use foreign keys to link to the relevant models such as country and city.

You may alternatively copy the required full text data from the relevant model into the Positions and Companies, but be aware that any changes to the city or country would then not be updated in the relevant models.

Companies

The companies feature will allow a client to administer companies.

This includes the ability to:

- browse
- read
- add
- edit
- "soft delete"
- "soft undo"

At this time, companies may be edited, added and deleted by any user.

The company model will have the following fields:

- (company) name
- city
- state
- country

- logo (image)

Use the country, state, city and company name as a combined unique key so that a company with multiple locations can advertise a position for a particular city.

Key relationships include, but may not be limited to:

- A company may have one or more positions related to it.

Remember that soft deletes will be required.

Positions (Listings)

The positions feature will allow a client to manage positions. This includes:

- browse
- read
- add
- edit
- "soft delete"
- "soft undo"

At this time, positions may be edited, added and deleted by any user.

The positions model will include, but may not be limited to the following fields:

- advertising start date
- advertising end date
- position title
- position description
- position keywords
- minimum salary
- maximum salary
- salary currency (default AUD)
- company (including city, state and country)
- benefits
- requirements
- position type (permanent, contract, part-time, casual)

Key relationships include, but may not be limited to:

- A position belongs to a company.
- A position belongs to a user (client).

Remember that soft deletes will be required.

Applications (FUTURE FEATURE)

The applications feature will allow an applicant will be able to apply for a position.

This feature is *not* to be implemented at this time.

Users

The users feature allows for users of the system to be managed.

This includes:

- browse
- read
- add
- edit
- "soft delete"
- "soft undo"

The users model should include:

- user nickname
- given name
- family name
- email address
- company
- user type (client, staff, or applicant)
- status (active, unconfirmed, suspended, banned, unknown, suspended)

During "online" registration the client would be asked if they will be a client (posting positions) or an applicant (looking for work).

A user cannot be both client and applicant.

Key relationships include, but may not be limited to:

- A client may have zero of more positions advertised.
- A client belongs to one company
- An applicant belongs to no company
- An applicant may have zero or more applications (FUTURE FEATURE)

Remember that soft deletes will be required.

Roles and Permissions

Roles and permissions are NOT to be implemented at this time.

You do not need to create an API for Roles and Permissions as these are tied into the security of the application and the API.