|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Assessment # and title** | | | | SaaS-AT2-Pt2 | | Portfolio Part 2 – Front-End Development using MVC | | | | | | |
| **Lecturer name** | | | | *Student to fill this section out* | | | | | | | | |
| **Student name** | | | | *Student to fill this section out* | | | | | | | | |
| **Student ID number** | | | | *Student to fill this section out* | | | | | | | | |
| **Telephone contact number** | | | | *Student to fill this section out* | | | | | | | | |
| **Email** | | | | *Student to fill this section out* | | | | | | | | |
| By completing and submitting this signed form to my lecturer, I am stating that:   1. The attached submission is completely my own work 2. I have correctly cited all sources of information used in this work (if required) 3. I have kept a copy of this assessment (where practicable) 4. I understand a copy of my assessment will be kept by the NMTAFE for their records 5. I understand my assessment may be selected for use in the NMTAFE’s validation and audit process to ensure student assessment meets requirements | | | | | | | | | | | | |
| **Student Signature** | | *Student to fill this section out* | | | | | | | **Date** | | *Student to fill this section out* | |
| Assessors please note: Where verbal clarification has been sought from a student to gather additional assessment evidence from an assessment item, question/s and response/s must be recorded, signed, and dated by the assessor, against the relevant assessment item/s. | | | | | | | | | | | | |
| NB: Feedback will be given via Blackboard when possible. | | | | | | | | | | | | |
| **Submission 1** | | Result | Satisfactory / Not Yet Satisfactory | | | | | | | Date | |  |
| *To satisfy requirements for this assessment, you need to complete the following:* | | | Feedback to student… | | | | | | | | | |
| **Submission 2** | | Result | Satisfactory / Not Yet Satisfactory | | | | | | | Date | |  |
| *To satisfy requirements for this assessment, you need to complete the following:* | | | Feedback to student… | | | | | | | | | |
| **Student Feedback** | | | Feedback from student… | | | | | | | | | |
| Lecturer Signature | | |  | | | | | Student Signature | | | |  |
| **Assessment type (þ):** | | | | | | | | | | | | |
|  | Questioning (Oral/Written) | | | |  | | 3rd Party Report | | | | | |
|  | Practical Demonstration | | | |  | | Other – Project/Portfolio (*please specify below):* | | | | | |
|  |  | | | |  | | | | | | | |

Contents

[Required Resources 4](#_Toc164775630)

[General Details & Requirements 7](#_Toc164775631)

[Scenario 10](#_Toc164775632)

[Assessment Steps 11](#_Toc164775633)

[00 Setting Up 12](#_Toc164775634)

[01 Creating Laravel Project 14](#_Toc164775635)

[01 Evidence 19](#_Toc164775636)

[02 Templates 20](#_Toc164775637)

[02 Evidence 22](#_Toc164775638)

[03 Static Pages 23](#_Toc164775639)

[03 Evidence 25](#_Toc164775640)

[04 User BREAD/CRUD 26](#_Toc164775641)

[04 Evidence: Screenshots 34](#_Toc164775642)

[04 Evidence: Checkpoint 35](#_Toc164775643)

[05 Listings BREAD/CRUD 36](#_Toc164775644)

[06 User Registration Verification Email 37](#_Toc164775645)

[06 Verification Evidence 38](#_Toc164775646)

[07 Roles and Permissions 39](#_Toc164775647)

[08 Demonstration and Code Review 40](#_Toc164775648)

[99 Submission of Work 41](#_Toc164775649)

[Appendix A: Development Folder Structure 42](#_Toc164775650)

[PC Users: 42](#_Toc164775651)

[Mac/Linux Users: 42](#_Toc164775652)

[PC/Mac/Linux Users: 42](#_Toc164775653)

[Appendix B: Code Style Guidelines 43](#_Toc164775654)

[PHP Code (General) 43](#_Toc164775655)

[Applications Built with Laravel 43](#_Toc164775656)

[HTML Code 43](#_Toc164775657)

[JavaScript 43](#_Toc164775658)

[Python Code 43](#_Toc164775659)

[JSON Code 43](#_Toc164775660)

[Appendix C: Commenting Code 44](#_Toc164775661)

[Appendix D: Database Tables and Fields 45](#_Toc164775662)

[Appendix T: Test Data 46](#_Toc164775663)

[Appendix Z: Example Code 51](#_Toc164775664)

|  |
| --- |
| Required Resources |
| The base requirements this assessment task, are listed below. They are listed as common (for both PC and Mac), and for the individual operating systems.  Alternatives to the ones given are available, but there will be very limited support for these systems. Lecturers may provide such support at their discretion and outside of class time.  ***IMPORTANT: Some items may not be required for use in this assessment task.*** |
| Common: |
| * Access to Office 365 & Microsoft Word * PhpStorm   + You may use a similar IDE such as VS Code, but support will be limited. * Git - Version Control System   + Part of Laragon, or by installing via the Git-SCM installer * MailPit   + A self-contained mail system that provides an excellent GUI for testing the sending of mail from applications without leaving your development machine.   + You will need to first install MailPit and then update your .bash\_aliases file. Details explaining how to do this this are available on the SQuASH Help-Desk FAQs.   + Mailpit. (n.d.). *Mailpit - email & SMTP testing tool*. [online] Available at: <https://mailpit.axllent.org> [Accessed 23 Apr. 2024]. * Web Browsers (Must have **at least** TWO different rendering engines)   + Chromium based: Edge, Chrome, (one of)   + Firefox   + Safari (MacOS/iOS only) * Command Line Terminal Interface   + See the following sections for PC/MAC/Linux * \*AMP Stack   + See the following section on LAMP/WAMP/MAMP stacks for the provision of PHP, Web Server, Database Servers and other supporting systems/applications. |
| PC: |
| Web Services (plus Database Services)   * Laragon ([Laragon - portable, isolated, fast & powerful development environment](https://laragon.org/))   + An easily updateable, and extensible WAMP stack that comes with MySQL, NodeJS, PHP, Apache, Git and NginX by default, and the simple ability to add Memcached, Redis, PostgreSQL, MongoDB, and other services.   Command Line Interface Terminal   * Windows Terminal ([Windows Terminal - Microsoft Apps](https://apps.microsoft.com/detail/9n0dx20hk701?launch=true&mode=full&hl=en-us&gl=au&ocid=bingwebsearch))   + The Microsoft equivalent to the iTerm2 option for MacOS.   + A very nicely presented terminal, with some cool features.   + We recommend using the Git Bash CLI with this to provide an environment that is very similar to many hosting provider systems that use Linux and thus Bash or equivalent CLI. Information on setting this up is found on the [SQuASH Help Desk Knowledge Base](https://help.screencraft.net.au/hc/2680392001/65/add-git-bash-to-microsoft-terminal). |
| Mac: |
| Command Line Interface Terminal   * iTerm2 ([iTerm2 - macOS Terminal Replacement](https://iterm2.com/index.html))   + Very customisable terminal application with much more friendly output than the default MacOS terminal.   Web Services   * Laravel Herd ([Laravel Herd](https://herd.laravel.com/))   + This is a relatively new introduction for MacOS, NginX, PHP. It also supports some other services in the Pro version.   Database Services   * DBngin ([DBngin | All-in-One Database Version Management Tool](https://dbngin.com/))   + We suggest partnering the free version of Herd with DBngin for PostgreSQL, MySQL and some other database services. |
| Alternatives: |
| Alternatives are available for the above “stacks”.  These include Docker Desktop, XAMPP, WAMPP, MAMP, AMPSS and many more.  Support for the use of alternative stacks to those indicated is exceptionally limited, and at the discretion of the lecturer(s).  It is important to remember that we directly support Laragon (Windows) and by appointment Herd and DBngin (Mac).  Herd has just been released for Windows ([Laravel Herd](https://herd.laravel.com/windows)), but students will still require database services and other items installed separately for development when using the Free version. |
| General Details & Requirements |
| Assessment Due Date |
| This assessment is split into components that have several due dates.  This item is due:   * Session 15 at 18:00 (6:00PM) on the day of the scheduled lecture   Refer to Blackboard for most accurate dates, which may alter due to unforeseen circumstances.  We also will endeavour to update these document(s) at the same time.  *It is advantageous to you to attempt to meet the deadline provided  as failure to do so will be deemed an attempt.* |
| General Instructions |
| Read the WHOLE assessment before commencing the portfolio.  Note any items that are not clear and seek clarification from the assessor/lecturer before commencing.  Follow the steps listed in this assessment item in the order given.  You **will be required** to demonstrate and answer questions about your code, solutions, and other aspects of working with MVC systems during or just after the assessment.  Final submission of the documentation, code, and associated evidence is at the end of the portfolio. |
| Important |
| We presume that all assessment work is completed on a PC with the software as specified. This should reduce the issues encountered.  Whilst other applications and operating systems may be used, lecturers and assessors are unable to give extensive support to ensure your environment is working as required.  Any support is at the discretion of the lecturer and under normal circumstances by appointment outside of lecturing time. |
| Answering Questions |
| When a step includes a question, you must attempt to answer it.  Space will be provided for your answer.  All answers must be in complete sentences.  Answer lengths are expected to be between the limits provided.  The answer space will expand with the content you type or images you paste.  Resize images to fit the space provided, ensuring they are still legible.  *If required, make sure to add any code you’ve written in a separate file to your submission.  DO NOT put long pieces of code (over 15 lines) in this document.* |
| Sources of Information |
| In industry, it is good practice to keep track of where information was obtained.  This is especially true if it is a written document, or even code.  You will include the APA6 (or APA7) reference/bibliography with each answer, also embed bibliography references within any code that uses any external or provided resources.  For example:   * au.rs-online.com. (n.d.). *RS Components | Electronic and Electrical Supplies | Australia*. [online] Available at: <https://au.rs-online.com/>. * McCallum, N. (2020). *How to Write C++ Comments*. [online] www.nickmccullum.com. Available at: https://www.nickmccullum.com/how-to-write-cpp-comments/ [Accessed 21 Nov. 2023]. * bootcamp.laravel.com. (n.d.). *Laravel Bootcamp - Learn the PHP Framework for Web Artisans*. [online] Available at: http://bootcamp.laravel.com [Accessed 23 Apr. 2024]. |
| Code Storage |
| We require that you create a private GIT repository on GitHub and use this to store a copy of your work. Further information is given in the assessment steps.  In addition, you should regularly compress and copy your code to your TAFE provided cloud storage (OneDrive within your college Office365).  Do NOT try copying uncompressed folder structures to OneDrive as this will take hours to complete. |
| Code Style |
| Appendix A: Code Style Guidelines contains the for details on the code style(s) to use for this assessment. |
| Scenario |
| You are currently working as an intern for a small Perth-based start-up company called *Incredibly Obvious Technologies*.  The company have given you the task of creating a demonstration web based back-end administration interface and code for a small application. There will also be a “general public” facing basic web front‑end for the application.  In your investigations they have laid out several questions and tasks to be performed. These tasks and questions are outlined in the remainder of this document.  Because you are working for a company who have Web, IoT and Software development as their primary industries, they require their staff to be able to work both in a ‘code’ environment, ‘WYSIWYG’ environment, and at the command line interface (CLI).  You are also required to apply version control practices during your development and testing cycles. |

This area intentionally left blank.

|  |
| --- |
| Assessment Steps |
| Complete the steps that follow, in the order given.  Make sure you carefully complete each step before progressing to the next.  Ensure any evidence is added to this document BEFORE going onto the next step. |

This area intentionally left blank.

|  |
| --- |
| 00 Setting Up These steps provide the setting up of the portfolio of work and must be completed before attempting any work. Complete the Front Page Make sure you have filled out the front page of this document. Read the Assessment Completely Familiarise yourself with the content and document your progress in this assessment. This means, **READ** the **WHOLE** assessment **AT LEAST ONCE** before starting any work.  Make notes on the requirements of this assessment, as details appear as you progress through and are not given in one location.  This is very similar to how details emerge during the continuous development interviews and meetings with a client.  At any stage during this assignment, you may consult the stakeholder(s) or their representative(s). Download Provided Files You must download the required files from Blackboard.  These files include, but may not be limited to:   * ReadMe.md * Secrets.md * Resources.zip (or Resources.7z)   Once downloaded, uncompress the files into a safe place, but DO NOT move the files until directed to do so in the instructions that follow. Enabled SQLite & PDO SQLite Extensions  * Open Laragon * Right Click in the window * Hover on PHP, then the Extensions option, and click on sqlite3. * Repeat and click on the pdo\_sqlite extension.  Run MS Terminal Open the Microsoft Terminal with the Bash CLI available for commands to be executed[[1]](#footnote-2).  Split the terminal into three parts vertically using ALT+SHIFT+-  Click in the middle section and then split this into half using ALT+SHIFT++ Verify Availability of Composer, PHP, Node and NPM Run each of the following commands and make sure no errors are given:  Check PHP Available (should be v8.2 or later)  php --version  Check Composer Available (should be v2.4 or later)  composer --version  Check NodeJS Available (should be at least node v18)  node --version  Check NPM Available (dependant on the version of node)  npm –version  If any of these are not available, then check with your assessor/lecturer. Create a New EMPTY Repository on GitHub In a browser open <https://github.com> and log into your personal (or TAFE) GitHub account.  Once logged in, create a new Empty Repository and name the repository SaaS-FED-POR-Pt2-XXX, where XXX are your initials.  You will need this shortly.  **IMPORTANT:**  **Ensure you DO NOT create any files as part of the Repository as this will lead to issues when you push to this remote.** |
| 01 Creating Laravel Project Follow these instructions carefully to set up the project ready for the rest of the portfolio.  For all stages of this assessment, you will require:   * Laragon (or equivalent) * Git * MS Terminal (or equivalent) * 2 Browsers (Chrome and Firefox / Chrome and Safari)   For non-Windows users, your environment will be different. Windows Users: (Git) Bash and Terminal To install (Git) Bash in the MS Windows Terminal please follow the instructions on the ScreenCraft SQuASH Knowledge base:   * <https://help.screencraft.net.au/hc/2680392001/65/add-git-bash-to-microsoft-terminal?category_id=35>  Global Settings Configure git as required by following the information in:   * <https://github.com/AdyGCode/Intro-to-Version-Control/blob/main/docs/05-global-settings.md>.  Verify the Global Settings Make sure your global settings are correct for User Email and Name:  git config --global --list  You should have user.name and user.email, plus the init.defaultBranch setting configured. Updating/Installing the Laravel Installer Install (or update) the Laravel Installer script by completing the following:   1. Open your Bash terminal. 2. Make sure you are in your Source/Repos folder 3. Execute the following command:   composer global require laravel/installer  Once completed we are now ready to create the Laravel Project. Creating a new Laravel Project In the same terminal run:  laravel new  At each prompt use the following details, replacing the XXX with YOUR initials (e.g. Adrian Gould would use ag for their initials). Settings for the Laravel Project  | Step | Details | | --- | --- | | Name of project: | saas-at2-pt2-XXX  NB: Take a screenshot of this step and include in the Evidence area under Evidence 01a.  It should be similar to: | | Starter Kit: | breeze | | Breeze stack: | blade | | Dark Mode: | yes | | Testing framework: | Pest | | Git repository: | Yes | | Database driver | SQLite OR MySQL/MariaDB |   Note:  If you use MySQL you will need to make sure any database users/database are created. These instructions are not provided. Editing the Environment Settings Open PhpStorm (or equivalent) and locate and open the project folder.  Once open, locate and open the .env file and make the following changes:   1. Change the Application Name to: “Workopia | v2 | XXX” where XXX are your initials. 2. Change the Mail Host to 127.0.0.1 and the Mail Port to 2525 3. Change the Mail From address to XXX@example.com where XXX are your initials. 4. Change the Application URL to http://saas-at2-pt2-XXX.test where XXX are your initials. 5. Alter the Time Zone to Australia/Perth. 6. Alter the App Locale to be en\_AU.  Move ReadMe.md In the terminal, rename the README.md file to ReadMe-Laravel.md using the Bash CLI command:  mv README.md ReadMe-Laravel.md  Download the provided ReadMe.md skeleton from BlackBoard and place in the root of your project.  Read through the ReadMe.md file. You will need to complete this before submitting for assessment. You will be able to make some changes to the file before progressing (eg your name, repository name, and so on). Update .gitignore From BlackBoard, download the gitignore.zip (or gitignore.7z) and uncompress the file.  Move .gitignore file into the root of your project, replacing the one that was automatically created. Checking New Project works Open the Laragon window and click STOP to halt all running Laragon services.  Once all have stopped, click START ALL to restart the services.  You may be prompted for your administrator credentials.   * At TAFE click NO, with BYOD fill out the details and click YES.   Open your two Web Browsers and head to: http://saas-at2-pt2-XXX.test (replace XXX with your initials). Evidence Take a screenshot of one of the browsers, making sure you include the Web Address and include in the Evidence 01b area. Add and Commit New Files to the Repository Make sure you are in your project folder in the CLI (cd ~/Source/Repos/saas-at2-pt2-XXX) before continuing.  In the Command line execute the following commands:  Check the repository status  git status  Add the New ReadMe file to staging  git add ReadMe.md  Add the updated .gitignore file to staging  git add .gitignore  Add the remaining files to staging  git add .  Commit the initial stage of the portfolio  git commit -m “init: Initialise the Workopia Version 2 – Laravel 11 Project”  Add the upstream location to your remote list  git remote add origin https://github.com/YOURGITHUBUSERNAME/SaaS-FED-POR-Pt2-XXX.git  Check the remote is correctly listed  git remote -v  Push the commits on the main branch to the remote (origin)  git push --set-upstream origin main  Answer any questions/prompts as needed to push the project to your private repository. Verify Push Executed Correctly Go to github.com and refresh your repository page to ensure the files have been uploaded.  Take a Screenshot of the repository home page and add to the Evidence area under Evidence 01c. Run MailPit If you do not have MailPit running, see the FAQs on the SQuASH Help Desk for details on how to install and then use it[[2]](#footnote-3). We also recommend adding a command line alias to make it easier to run MailPit from within Laragon.  We recommend that you split the middle of your three pane Terminal window vertically – one side for MailPit and the other for the Tailwind CSS watcher (see later).  Make sure you are in the project folder (saas-at2-pt2-XXX) BEFORE running:  mailpit --smtp 0.0.0.0:2525  See the instructions for more options and details.  Screen shot of the terminal layout shown here for reference:  A screenshot of a computer  Description automatically generated Invite your lecturer(s)/Assessor(s) to your private repository Follow the instructions in the FAQs on the SQuASH Help Desk and invite your assessor/lecturer to your repository.   * <https://help.screencraft.net.au/hc/2680392001/71/invite-collaborators-to-your-private-git-repository>   Provide Evidence of the invitation being sent using a screenshot. |
| 01 EvidenceEvidence 01a: Laravel Project Name (Screenshot)Evidence 01b: Initial Project Page & URL (Screenshot)Evidence 01c: GitHub Repository Page (Screenshot)Evidence 01d: Collaboration Request Sent (Screenshot) |
| 02 Templates We provide the basic templates for you to use.  Download the workopia-views-and-templates.zip (or workopia-views-and-templates.7z) file from BlackBoard.  Uncompress the files into a folder such as workopia-laravel-templates.  Open Windows File Explorer (WIN+E), using this window, locate and open this “workopia laravel templates“ folder as you will need this in subsequent steps.  If you wish, use WIN+🡪 to put this window on the Right.  Open a second Windows File Explorer (WIN+E), using this window, locate and open your Source/Repos/saas-atr2-pt2-XXX folder. Use WIN+🡨 to put this window on the Left.  Duplicate the project’s existing resources folder and rename it to “z-resources-backup”.  Navigate into the existing resources/views folder in the project.  Copy the components, auth, layouts folders from the portfolio-workopia-resources folder into your resources/views folder, replacing the current files and folders.  Copy the dashboard.blade.php file into the views folder, replacing the existing file.  Copy the welcome.blade.php file to the same folder and overwrite the existing file.  IMPORTANT:  You may get error that some routes are missing, to fix this, change any route(“…”) to be route(“home”) for this step.  Once you have copied the templates into the correct locations, go back to your browser and refresh the page.  Make sure you see the new default guest homepage, which should look like the one shown below, but without the login section:  (login page showing header and footer of Workopia site)  If you get error pages, please check the views for any route(‘…’) and replace them with route(“home”), as explained above. Check In and Commit All Files Run the following commands:  git add .  git commit -m “feat(templates): Add basic layout templates”  git push  Take a Screenshot of the output of the commit and include in the 02 Evidence area as needed. Evidence Take a screenshot of the new guest homepage, including your browser address bar, and add to the 02 Evidence area. |
| 02 EvidenceScreenshot of Git Commit and PushScreenshot of Updated Home (Guest Welcome) Screen |
| 03 Static Pages In this portfolio we are not working with multiple branches. If you wish to do so then please feel free to do so, and make sure you use “pull requests” to merge completed features back into the main branch. You should also update the local main branch to include the new code as needed. Setting up Initial Static Pages Create a views/pages folder in the resources area.  Move the welcome.blade.php file into this new folder. Add Contact Us, About and Pricing Pages Create three new view files in the same folder, contact-us.blade.php, about.blade.php and pricing.blade.php.  Add the code below to each page:   |  |  | | --- | --- | | <x-guest-layout>  <section>  <h3 *class*=“text-center text-3xl mb-4   border border-gray-300 font-bold p-3  bg-gray-200 dark:bg-gray-500 text-black   dark:text-white/80 rounded-lg”>  PAGE NAME  </h3>  <p *class*=“text-center text-3xl my-8”>  Place holder for static pages FILENAME.  </p>  </section> </x-guest-layout> | Pages Folder with  static page files |   Replace the text PAGE NAME in each file with the appropriate text of “Contact Us”, “About Workopia” and “Pricing”. Also replace FILENAME with the required page’s filename. Create a Static Pages Controller Use the command:  php artisan make:controller StaticPages  Locate and open the new StaticPages class file. Make Welcome Method in StaticPages Controller Create a new public function called welcome that returns the welcome view.  Update the web routes so that the ‘/’ home route calls the StaticPages class to render the welcome method that was just created.  Route::get(‘/’, [StaticPages::*class*, ‘welcome’])->name(‘home’); Add About, Contact Us and Pricing Methods & Routes Create the required routes and methods for the about, contact us and pricing static pages. Test Static Page Routes Test the home route by visiting the appropriate URL, such as http://saas-atr2-pt2-XXX.test.  Test the welcome route by visiting <http://saas-atr2-pt2-XXX.test/welcome>.  Test the pricing, about, and contact us routes.  Add Evidence to the 03 Evidence section as indicated (Pricing and Contact pages, with the browser’s Address bar). Check In and Commit All Files Run the following commands:  git add .  git commit -m “feat(static): Create static page controller and views”  git push  Take a Screenshot of the output of the commit and include in the evidence area as needed. |
| 03 EvidenceScreenshot of updated Pricing page with Address BarScreenshot of updated Contact Us page with Address BarScreenshot of Git Add, Commit and Push |
| 04 User BREAD/CRUD In this step, you will be creating the administration BREAD/CRUD for the Users (table) of the system.  You will be creating views and the appropriate controller methods for Browse, Read, Edit, Add and Delete.  The requirements for this are:   * Ensure that a User is logged in before showing the Users area. * Allow any **logged in** user to Browse (10 users per page) the users. * Allow any **logged in** user to Read a user and view any selected User’s details. * Allow any **logged in** user to Edit any selected user. * Allow any **logged in** user to Add a NEW user. * Allow any **logged in** user to Delete any selected user.   Remember that to create a new controller you must use the singular version of the table (that is, User) and Pascal case the controller’s name, again using the singular version of the table name (UserController).  The following command is to be used, and it is a one-line command, to create the UserController.  php artisan make:controller UserController --model=User --resource --requests --pest  A slightly less wordy version is:  php artisan make:controller UserController -m User -rR --pest  Once created you will need to complete the following tasks. Update your Web Routes File We need the routes to each of the ‘User Admin’ pages to be added.  These routes are:   | HTTP VERB | URI | Route Name |  | Method Called | | --- | --- | --- | --- | --- | | GET|HEAD | users | users.index | › | UserController@index | | POST | users | users.store | › | UserController@store | | GET|HEAD | users/create | users.create | › | UserController@create | | GET|HEAD | users/{user} | users.show | › | UserController@show | | PUT|PATCH | users/{user} | users.update | › | UserController@update | | DELETE | users/{user} | users.destroy | › | UserController@destroy | | GET|HEAD | users/{user}/edit | users.edit | › | UserController@edit | | GET|HEAD | users/{user}/edit | users.edit | › | UserController@edit |   We will also ensure the user is logged in before they access the User Administration. Note that these default routes do not include any soft delete (trash/recycle bin) routes.  To do so, update the web routes in the routes/web.php file, and add the following code to generate the routes for each of the pages:  Route::middleware([‘auth’,])->group(*function* () {  Route::resource(‘users’, UserController::*class*);  });  You can verify the routes have been created by using the following CLI command:  php artisan route:list Update the Navigation Layout **Important**:  Do *not* update the navigation-guest layout.  Also, the routes have been defaulted to ‘home’ or ‘welcome’ to stop any rendering errors until such time as you update the navigation to point at the relevant page(s) in future steps.  Any link that points to the “Users” now needs to be updated to say route(“users.index”).  Example:  <x-nav-link *:href*=“route(‘users.index’)”   *:active*=“request()->routeIs(‘users.\*’)”   *class*=“group”>  {{ \_\_(‘Users’) }} </x-nav-link>  **Note:**  users.\* will let any route that starts with users to be recognised.  For example users.index and users.destroy. Create Update User Migration to Add ‘Login At’ and ‘Logout At’ Run the command:  php artisan make:migration update\_users\_table\_with\_login\_data  To this new file (check your database/migrations folder), update the up and down methods using the following code:  *public function* up(): *void* {  Schema::table('users', *function* (Blueprint $table) {  $table->dateTime('login\_at')->nullable();  $table->dateTime('logout\_at')->nullable();  }); }  */\*\*  \* Reverse the migrations.  \*/ public function* down(): *void* {  Schema::dropColumns('users', ['login\_at', 'logout\_at']); } Create an Update Users with City and State Migration Run the “artisan make:migration” command to create a new migration that updates the users table with city and state. Suggest using the name ‘update\_users\_with\_city\_and\_state’ for the migration.  See the details in Appendix D: Database Tables and Fields for field sizes and types for the migration. Remember that you need both UP and DOWN methods.  **Note:**  You may put the city and state for the user in the main users table, or the profile table.  For this assessment, it is not important as to the final location of this data, but that the methodology is correctly applied. Create the Users Views Folder Using any suitable method, create a new resources/views/users folder. For example, to use the command line you would execute:  mkdir resources/views/users Create Browse Users Page (index) In your editor, navigate to the resources/views/users folder and add a new PHP file named index.blade.php.  Base the layout of the page on the Laravel Bootcamp tutorial code:   * <https://github.com/AdyGCode/laravel11-bootcamp-2024-s1>  Create the User Browse (index) Method Open the UserController class and edit the index method by including this:  $users = User::*all*(); *return* view('users.index', compact(['users']));  Test the Users page shows all the users.  Take a screenshot and add to the 04 Evidence area. Required Changes after the Users Page is Tested Update the index method to use paginate(10) instead of all().  Update the view to show the pagination controls in the footer of the table, by adding a <tfoot> … </tfoot> section before the </table>. Inside this tfoot add <tr><td> … </td></tr>. Finally inside this <td> … </td> add {{ $users->links() }}.  Test the updated page and take a screenshot and add to the 04 Evidence area. Check In and Commit All Files Note:  If you press enter before the second " on the commit message it allows you to add more lines of information.  To end the message, close the " as normal.  Run the following commands:  git add .  git commit -m “feat(users): Add users feature  - create UserController - create Users index view  - create Users index method without pagination  - update Users index view to include pagination - update Users index method to include pagination ”  git push  Take a Screenshot of the output of the commit and include in the 04 Evidence area as needed. Soft Delete Migration Because we want the Users to be recoverable, we add soft deletions to the Users table.  First, create a new migration to add soft deletes:  artisan make:migration update\_users\_table\_with\_soft\_deletes  In the new migration file modify the stub code to add and remove the soft delete field (deleted\_at) from the “users” table.  *return new class extends* Migration {  */\*\*  \* Run the migrations.  \*/  public function* up(): *void* {  Schema::table('users', *function* (Blueprint $table) {  $table->softDeletes();  });  }   */\*\*  \* Reverse the migrations.  \*/  public function* down(): *void* {  Schema::table('users', *function* (Blueprint $table) {  $table->dropSoftDeletes();  });  } };  Run the migration to update the table (NB: Not a fresh migration).  php artisan migrate Create Read, Add, and Edit User Pages In your editor, navigate to the resources/views/users folder and add new PHP files named show.blade.php, create.blade.php and edit.blade.php.  Edit these pages and create the required layouts, basing the layout of the page on the Laravel Bootcamp tutorial code:   * <https://github.com/AdyGCode/laravel11-bootcamp-2024-s1>  Create Delete User Confirmation Page Even though we are using soft deletes for the users, we will add a second level of ‘caution’ by adding a confirm the delete page. Create Delete Confirmation Blade File In your editor, navigate to the resources/views/users folder and duplicate your show page, making the required changes so the buttons are CONFIRM and CANCEL only.  Also make sure you do not have the “ADD USER” button on the page. Add Delete Method to the UserController Navigate to the end of the UserController file, and just before the destroy method add:  */\*\*  \* Show form to confirm deletion of user resource from storage.  \*/ public function* delete(User $user) {  *return* view('users.delete', $user); }  Make sure that you have added the above code BEFORE the last close curly-bracket (}) in the file. Create “Delete” Route This route displays the delete confirmation page with the selected user.  Open the web routes file and add and update the following in the file:  Route::middleware('auth')->group(*function* () {  Route::resource('users', UserController::*class*);  Route::get('users/{user}/delete',[UserController::*class*, 'delete'])->name('user.delete'); }); Add the ‘destroy’ code to the destroy method of the User Controller Add the following to the destroy method of the User Controller:  $user->delete(); *return* redirect(route('users.index')); Handling The Trash Due to using Soft Deletes you will need some way of managing these ‘deleted’ users.  The steps you will need to complete are based on the code from:   * <https://github.com/AdyGCode/laravel11-bootcamp-2024-s1> * Presentations in Session 12   Some of the steps are given in full, others require you to fill in the blanks.  **Remember:**  You will need to add the SoftDeletes to the Users model to allow for the soft deletes to occur. Check in the session 12 presentation for more information. Add the Trash Routes Add the routes to handle the ‘recycle bin’ or “trash”.  Add routes for showing all ‘trashed’ users, recovering a user from ‘trash’, emptying the ‘trash’, removing a single user from ‘trash’, and recovering all users from ‘trash’. Showing all users in the trash: Route::get('users/trash',[UserController::*class*, 'trash'])->name('users.trash'); Recover a user from trash: Route::patch('users/trash/{id}/recover',[UserController::*class*, recoverOne])->name('users.recover-one'); Emptying the trash Route::delete('users/trash/empty',[UserController::*class*, 'emptyAll'])->name('users.empty-all');  Note:  It is important to note the correct HTTP methods are used in the routes.  **Patch** (or put) – as you are updating the user from being in trash to being restored to the user list  **Delete** – as the trashed users are being permanently removed.  This applies also to single user delete and all user recovery.  Complete the additional routes for remove a single user from trash (emptyOne method with empty-one alias) and recover all (recoverAll method with recover-all alias) users from trash. Create “List Users in Trash” Method The list users in trash method is a public function named trash(). It returns the View type.  */\*\*  \* Return view showing all users in the trash.  \*/ public function* trash(): View {  $users = User::onlyTrashed()->latest()->get();  *return* view('users.trash', compact(['users'])); } Create “Restore User from Trash” Method Add the “Recover One” user method to the Users Controller, as given below:  */\*\*  \* Restore user from the trash.  \*  \** ***@param*** *$id  \** ***@return*** *RedirectResponse  \*/ public function* recoverOne($id): RedirectResponse {  $user = User::onlyTrashed()->find($id);  $user->restore();  *return* redirect(route('users.trash')); } Create an “Empty All Trashed Users” Method Add the following to create the empty trash method:  */\*\*  \* Permanently remove all users that are in the trash  \*  \** ***@return*** *RedirectResponse  \*/ public function emptyAll*(): RedirectResponse {  $users = User::onlyTrashed()->get();  $trashCount = $users->count();  *foreach* ($users *as* $user) {  $user->forceDelete(); *// This restores the soft-deleted user* }  *return* redirect(route('users.trash')); } Recover All Trashed Users Method Add a Recover All method to enable the recovery of all users in the Trash:  */\*\*  \* Restore all users in the trash to system  \*  \** ***@return*** *RedirectResponse  \*/ public function* recoverAll(): RedirectResponse {  $users = User::onlyTrashed()->get();  $trashCount = $users->count();   *foreach* ($users *as* $user) {  $user->restore(); *// This restores the soft-deleted user* }  *return* redirect(route('users.trash')); } Add a Permanently Remove One User from Trash Method Determine how to remove a single user from the recycle bin permanently.  Create the required code to implement this feature. Test Routes and Pages for User Admin Feature Complete testing of the new functionality.  Make sure that EACH function from the User Admin feature works as expected:   * Browse users (retrieve all, paginated). * Read a single user (show). * Add new user (create). * Edit an existing user (update). * Delete a user (delete) into recycle bin.   Also test the Recycle bin functionality:   * Delete at least four users. * Browse deleted users. * Recover a single user. * Permanently remove a single user. * Recover all users. * Permanently remove all trashed users.   Refresh your database after testing using:  php artisan migrate:fresh --seed Update ReadMe.md Update the ReadMe document with any additional details that are now evident from the development so far. |
| 04 Evidence: ScreenshotsScreenshot of Users page BEFORE paginationScreenshot of Users page AFTER pagination navigation addedScreenshot of the Version Control status, commit and push actions |
| 04 Evidence: Checkpoint During class time you will demonstrate to the lecturer/assessor your working code.  You will also be expected to open the code and show sections that may include, but are not limited to:   * Routes * User Model * User Controller * User Views * Validation  CHECKPOINT 1: Evidence You will be required to Demonstrate your working code to the lecturer/assessor before completing any further stages of this assessment.  This is to ensure you are competing the work as expected.  Examples of what you will be required to show may include, but not limited to:   * Registration of user. * Logout and Login. * Browse all users (with pagination) and show/read details of one user. * Edit a user’s basic details (NOT profile). * Delete user when using browse page and the read/show page. * Show list of deleted users in “trash”. * Recover a deleted user from “trash”. * Empty the “trash” of Deleted Users. |
| 05 Listings BREAD/CRUD In this stage you will create the required views, methods, requests, and other required items for the Job Listings.  You are expected to:   * Create the Listings model, Migration, and Seed files. * Create the Controller, Request and associated files. * Create the required Routes in the web routes file.   The above are to satisfy the following features:   * Create and test a Browse view and method that shows listings in a table, 10 at a time. * Create and test a Read view and method that shows any selected Listing in a detailed page. * Create and test an Add view and methods that allows a User to add a new Listing. * Create and test an Edit view and methods that allows a logged in User to edit an existing listing. * Create and test a Delete view and methods that allow for a listing to be MARKED as deleted (soft delete). * Create and test Job Listings trash operations including restore a deleted listing, permanently remove a soft deleted job listing, restore all deleted job listings and permanently remove all soft deleted job listings.   The delete method will use soft deletes that allow for a deleted listing to be restored.  You will need to create the required views for the above:   * listings/index.blade.php * listings/show.blade.php * listings/create.blade.php * listings/edit.blade.php * listings/delete.blade.php * listings/trash.blade.php   To begin this process, use the following to create the stubs for Listings:  php artisan make:model Listing --controller --factory --migration  --resource --pest --policy --requests --seed  Use the data from Appendix D: Database Tables and Fields to assist in defining the migration for the listings table. Version Control Make sure you have added, committed and pushed any feature changes to git and GitHub.  We suggest adding and committing each feature of the Listings in turn… suggested commit messages could be:  feat(listings): Create listings stubs  feat(listings): add browse listings with pagination  feat(listings): add create new listing  feat(listings): add edit listing  feat(listings): add delete listing into listing trash Possible other features/improvements It is possible to enhance the project by ensuring the same features that are part of the Non Laravel implementation of Workopia are also implemented.  For example - Search  feat(listings): add search function to listings browse Update ReadMe.md Update the ReadMe document with any additional details that are now evident from the development so far. |
| 06 User Registration Verification Email At this point you will now implement the requirement for the user to verify themselves before they are able to use the system.  More details at: <https://laravel.com/docs/11.x/verification>.  Edit the user model and update the class line to read:  *class* User *extends* Authenticatable *implements* MustVerifyEmail  If any other steps are required, ensure you have implemented them before continuing. Identify Unverified User Identify an unverified user from the seed data.  Note which user you are using in the 06 Verification Evidence section. Test Verification Log in as an unverified user.  Take a screenshot of the page displayed, add to the 06 Verification Evidence section. Click the (Re)Send Verification Email Make sure that MailPit is running and then open MailPit web interface.  Click the ‘resend verification’ to send the email.  Take screenshot of received Email, add to 06 Verification Evidence section. Click the Verification link to Complete Verification Take a screenshot of the newly verified User and add to 06 Verification Evidence section. Update ReadMe.md Update the ReadMe document with any additional details that are now evident from the development so far. Version Control Make sure you have added, committed and pushed any feature changes to git and GitHub.  Suggested message for the commit:  feat(email\_confirm): Add User confirmation email |
| 06 Verification Evidence06a Which Unverified User are you testing with?06a Unverified User Logged-in Screen06c Mail Pit Email received on Clicking (Re)Send Verification Email06d User now Verified After using Verification Link |
| 07 Roles and Permissions In this stage we will be adding Roles and Permissions to the Administration interface. Once the User Interface for the Roles and Permissions is created you will use it to allocate permissions to the various roles. Roles & Permissions Admin Area The Roles and Permissions area must only be accessed by an Administrator. Roles The Roles you will be implementing are:   |  |  |  | | --- | --- | --- | | Administrator | Staff | Client |  Permissions for Each Role The table below shows the Permissions for each Role, and conditions that also need to be applied (e.g. A client may only delete their OWN listings).   |  |  |  |  | | --- | --- | --- | --- | | Permission | Administrator | Staff | Client | | User-Browse | All | All | No | | User-Show | Any | Any | Their profile only | | User-Edit | Any | Any client  Their own profile | Their own profile | | User-Add | New staff member  New Administrator | New client | Only via register | | User-Delete | Any (not themselves) | Any Client only | Only themselves | | User-Trash-Recover (one) | Any |  |  | | User-Trash-Remove (one) | Any |  |  | | User-Trash-Empty (all) | Any |  |  | | User-Trash-Restore (all) | Any |  |  | | Listing-Browse | All | All | All | | Listing-Show | Any | All | All | | Listing-Edit | Any | Any | Own only | | Listing-Add | No | No | Own only | | Listing-Delete | Any | Any | Own only | | Listing-Trash-Recover (one) | Any | Any | Own only | | Listing -Trash-Remove (one) | Any | Any | Own only | | Listing -Trash-Empty (all) | Yes | No | Own only | | Listing -Trash-Restore (all) | Yes | No | Own only | | Roles & Permissions | Yes | No | No |  Update ReadMe.md Update the ReadMe document with any additional details that are now evident from the development so far. Version Control Make sure you have added, committed and pushed any feature changes to git and GitHub.  Suggested message for the commit:  feat(roles\_perms): Add roles and permissions admin interface  Create roles and permissions controller  Create roles and permissions admin view  Create methods to admin roles for users |
| 08 Demonstration and Code Review In this evidence section you will be expected to demonstrate the portfolio to the lecturer/assessor.  You will be required to demonstrate the following:   * User Login * User Logout   All the following operations require user to be verified and logged in.   * Ordinary User:   + Job Listings: Add, Edit, Delete, Recover OWN job listings, Empty their ‘trashed listings’. * Staff User:   + Job Listings: Add, Edit, Delete, Recover own listings, empty own trashed listings.   + Others Job Listings: Edit, Delete job listings for Ordinary user.   + Users: Add, Edit and Delete ORDINARY users. * Administrator User:   + Users: Add, Edit, Delete, Restore from trashed users, or Empty trashed users   + Users: Assign and remove roles and/or permissions to any user (For example, make a user a staff member)   You will also be asked a series of questions about your code, methods used, and underlying knowledge about Laravel, MVC, development and other aspects of connecting databases with web sites. |
| 99 Submission of Work To submit the portfolio, do the following:   * Ensure you have completed the ReadMe.md markdown file as required in Appendix Z: Example Code ReadMe. * Compress your WHOLE project folder as a ZIP or 7Z file. * Save this document with your answers. * Open Blackboard, locate and open the assessment: AT2 Portfolio Task 2. * Click the Submit button. * Upload this **completed assessment document** separately from the compressed code. * Upload the **compressed code** to the same submission. * Upload the completed Secrets.md file as defined in Appendix Z: Example Code Secrets.md. * Click Submit.   All answer documents MUST be submitted in Microsoft Office 365’s Word format. |

# Appendix A: Development Folder Structure

We recommend that all work for this and following portfolios are kept in a suitable folder structure.

For SaaS, this folder is in a folder inside you user profile.

* For PC users this is: C:\Users\XXXXXXXX\Source\Repos
* For Mac and Linux users this is: ~/Source/Repos

We then recommend that you keep this cluster’s work in a SaaS folder.

## PC Users:

Create the base folder structure using the following steps:

* Open a terminal application such as Windows Terminal with the Laragon Git Bash CLI.
* Enter the following commands pressing ENTER at the end of each line:
  + cd %USERPROFILE%
  + mkdir –p Source\Repos
  + cd Source\Repos

## Mac/Linux Users:

Create the base folder structure using these commands:

* cd ~
* mkdir –p Source/Repos
* cd Source/Repos

## PC/Mac/Linux Users:

If you wish to use a SaaS folder for the work, then use the commands:

* mkdir SaaS
* cd SaaS

This will create and place you in the SaaS folder.

Note: If using Laragon, then you would point the “root” folder to the SaaS folder.

# Appendix B: Code Style Guidelines

The following guidelines should be applied to your code as it is developed.

Many may be applied via the use of PhpStorm, PyCharm or similar plugins and code formatting.

## PHP Code (General)

Please refer to the PHP PSRs:

* <https://www.php-fig.org/psr/psr-1/>
* <https://www.php-fig.org/psr/psr-12/>
* <https://www.php-fig.org/psr/psr-4/>

## Applications Built with Laravel

Laravel code must be at least compliant with PHP Standards. Additional standards are shown in the following articles:

* <https://dev.to/lathindu1/laravel-best-practice-coding-standards-part-01-304l>
* <https://dev.to/lathindu1/laravel-best-practice-coding-standards-part-02-a40>

## HTML Code

Please refer to the Google Style guide:

* <https://google.github.io/styleguide/htmlcssguide.html>

## JavaScript

Refer to the Google JS Style guide:

* <https://google.github.io/styleguide/jsguide.html>

## Python Code

Your code will follow the PEP 8 standard.

## JSON Code

JSON should be formatted in an appropriate manner.

# Appendix C: Commenting Code

Here are some guideline rules for commenting code:

**Rule 1**: Comments should not duplicate the code.

**Rule 2**: Good comments do not excuse unclear code.

**Rule 3**: If you can’t write a clear comment, there may be a problem with the code.

**Rule 4**: Comments should dispel confusion, not cause it.

**Rule 5**: Explain unidiomatic code in comments.

**Rule 6**: Provide links to the original source of copied code.

**Rule 7**: Include links to external references where they will be most helpful.

**Rule 8**: Add comments when fixing bugs.

**Rule 9**: Use comments to mark incomplete implementations.

*Readability Counts   
- Zen of Python*

# Appendix D: Database Tables and Fields

To assist you in developing this portfolio the following are the various attributes required for the migrations.

YOU MUST NOT edit any previous or provided migrations. Any changes to the database fields MUST be done via update migrations that you create. For example, you may see these in action in Create New User Migration to Add Login at and Logout at Fields.

| Table | Field Name | Type | Size | Key? | Notes |
| --- | --- | --- | --- | --- | --- |
| Users | Id | Unsigned big integer auto-increment |  | PK |  |
| Name | String |  |  | Required |
| Email | String |  | Unique | Required |
| Email verified at | Date Time |  |  | Required |
| Password | String |  |  | Required |
| Remember token | String |  |  | Nullable |
| Created at | Date Time |  |  | *Remember that timestamps in Laravel creates both of these fields.* |
| Updated at | Date Time |  |  |
| Deleted at | Date Time |  |  | Non Standard, Update Migration required |
| Login at | Date Time |  |  | Non Standard, Update Migration required |
| Logout at | Date Time |  |  | Non Standard, Update Migration required |
| Profile | City | String | 128 |  | Non Standard, Update Migration nullable |
| State | String | 128 |  | Non Standard, Update Migration nullable |
| Listings | Id | Unsigned big integer auto-increment |  | PK |  |
| User id | Unsigned big integer |  | FK | Default 0 |
| Title | String |  |  | Required |
| Description | Text |  |  | Required |
| Salary | String | 45 |  | Required |
| Tags | String |  |  | Nullable |
| Company | String |  |  | Required |
| Address | String |  |  | Required |
| City | String | 128 |  | Required |
| State | String | 128 |  | Required |
| Phone | String | 45 |  | Nullable |
| Email | String |  |  | Required |
| Requirements | Text |  |  | Required |
| Benefits | Text |  |  | Required |
| Created at | Date Time |  |  | *Remember that timestamps in Laravel creates both of these fields.* |
| Updated at | Date Time |  |  |

# Appendix T: Test Data

The following is the test data to use when developing and testing this portfolio.

### Test Users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | eMail | Password | City | State |
| Terry Bull | Terry.Bull@example.com | Password7 | Wagga Wagga | NSW |
| Flo Chart | Flo.Chart@example.com | Password6 | Alice Springs | NT |
| Isolde House | Isolde.House@example.com | Password5 | Canberra | ACT |
| Russell Cattle | Russell.Cattle@example.com | Password4 | Launceston | TAS |
| Lyle Bull | Lyle.Bull@example.com | Password3 | Hobart | TAS |
| Jaqueline Hyde | Jaqueline.Hyde@example.com | Password2 | Gold Coast | QLD |
| Freida Livery | Freida.Livery@example.com | Password1 | Adelaide | SA |

### Test Listing

We are providing a restricted set of test listings for use this assessment:

|  |  |
| --- | --- |
| Test Position | 1 |
| User ID | Field Data |
| Title | Data Scientist |
| Description | Exciting opportunity to join our team in leveraging data to drive insights and innovation. |
| Salary | $90,000 - $110,000 AUD annually |
| Tags | Data Science, Analytics, Innovation |
| Company | Insightful Solutions |
| Address | 789 King Street |
| City | Sydney |
| State | New South Wales |
| Phone | (02) 9876 5432 |
| Email | careers@insightfulsolutions.com.au |
| Requirements | Bachelor's degree in Computer Science, Mathematics, or related field; 3+ years of experience in data science or analytics; Proficiency in Python, R, and SQL; Strong analytical and problem-solving skills; Excellent communication and teamwork abilities |
| Benefits | Competitive salary; Flexible work arrangements; Health and wellness benefits; Ongoing learning and development opportunities |
| Created at | 2024-04-24 09:30:00 |

|  |  |
| --- | --- |
| Test Position | 2 |
| User ID | Field Data |
| Title | Mining Engineer |
| Description | Exciting opportunity to work in the mining industry in Western Australia, contributing to the extraction and processing of valuable resources. |
| Salary | $100,000 - $120,000 AUD annually |
| Tags | Mining, Engineering, Resources |
| Company | Outback Mining Co. |
| Address | 456 Desert Highway |
| City | Kalgoorlie |
| State | Western Australia |
| Phone | (08) 7654 3210 |
| Email | careers@outbackmining.com.au |
| Requirements | Bachelor's degree in Mining Engineering or related field; 3+ years of experience in the mining industry; Knowledge of mine planning and scheduling software; Strong safety focus; Excellent teamwork and communication skills |
| Benefits | Competitive salary; Performance-based bonuses; Opportunities for career progression; Employee assistance program; Health and wellness initiatives |
| Created at | 2024-04-26 09:45:00 |
|  |  |

|  |  |
| --- | --- |
| Test Position | 3 |
| User ID | Field Data |
| Title | Civil Engineer |
| Description | Exciting opportunity to be part of major infrastructure projects in Western Australia. |
| Salary | $80,000 - $100,000 AUD annually |
| Tags | Civil Engineering, Infrastructure, Construction |
| Company | West Coast Engineering |
| Address | 321 Harbour Road |
| City | Perth |
| State | Western Australia |
| Phone | (08) 8765 4321 |
| Email | jobs@westcoastengineering.com.au |
| Requirements | Bachelor's degree in Civil Engineering or related field; 5+ years of experience in civil engineering; Knowledge of relevant software such as AutoCAD and Civil 3D; Strong problem-solving abilities; Excellent project management skills |
| Benefits | Competitive salary; Opportunities for career advancement; Supportive work environment; Health and wellness programs; Employee training and development |
| Created at | 2024-04-25 10:00:00 |

|  |  |
| --- | --- |
| Test Position | 4 |
| User ID | Field Data |
| Title | Data Scientist |
| Description | Exciting opportunity to join our team in leveraging data to drive insights and innovation. |
| Salary | $90,000 - $110,000 AUD annually |
| Tags | Data Science, Analytics, Innovation |
| Company | Insightful Solutions |
| Address | 789 King Street |
| City | Sydney |
| State | New South Wales |
| Phone | (02) 9876 5432 |
| Email | careers@insightfulsolutions.com.au |
| Requirements | Bachelor's degree in Computer Science, Mathematics, or related field; 3+ years of experience in data science or analytics; Proficiency in Python, R, and SQL; Strong analytical and problem-solving skills; Excellent communication and teamwork abilities |
| Benefits | Competitive salary; Flexible work arrangements; Health and wellness benefits; Ongoing learning and development opportunities |
| Created at | 2024-04-24 09:30:00 |

|  |  |
| --- | --- |
| Test Position | 5 |
| User ID | Field Data |
| Title | Software Developer |
| Description | Exciting opportunity to be part of a dynamic software development team and contribute to the creation of innovative solutions. |
| Salary | $85,000 - $100,000 AUD annually |
| Tags | Software Development, Programming |
| Company | Tech Innovations Pty Ltd |
| Address | 789 Innovation Drive |
| City | Canberra |
| State | Australian Capital Territory |
| Phone | (02) 9876 5432 |
| Email | careers@techinnovations.com.au |
| Requirements | Bachelor's degree in Computer Science or related field; 3+ years of experience in software development; Proficiency in Java, Python, and SQL; Strong problem-solving skills; Excellent teamwork and communication abilities |
| Benefits | Competitive salary; Flexible work arrangements; Opportunities for career growth; Health and wellness benefits; Professional development programs |
| Created at | 2024-05-01 10:30:00 |

|  |  |
| --- | --- |
| Test Position | 6 |
| User ID | Field Data |
| Title | Registered Nurse |
| Description | Rewarding opportunity to join our healthcare team and provide compassionate care to patients. |
| Salary | $65,000 - $80,000 AUD annually |
| Tags | Healthcare, Nursing, Registered Nurse |
| Company | Compassionate Care Hospital |
| Address | 456 Care Avenue |
| City | Adelaide |
| State | South Australia |
| Phone | (08) 7654 3210 |
| Email | jobs@compassionatecarehospital.com.au |
| Requirements | Bachelor's degree in Nursing; Current registration as a Registered Nurse with AHPRA; Experience in acute care preferred; Strong clinical assessment and decision-making skills; Compassionate and patient-centered approach |
| Benefits | Competitive salary; Shift allowances; Professional development opportunities; Supportive work environment; Employee assistance program |
| Created at | 2024-04-30 09:45:00 |

|  |  |
| --- | --- |
| Test Position | 7 |
| User ID | Field Data |
| Title | Senior Financial Analyst |
| Description | Exciting opportunity to join our finance team and drive financial analysis and reporting processes. |
| Salary | $90,000 - $110,000 AUD annually |
| Tags | Finance, Analyst, Reporting |
| Company | Capital Insight Group |
| Address | 123 Finance Street |
| City | Brisbane |
| State | Queensland |
| Phone | (07) 8765 4321 |
| Email | careers@capitalinsightgroup.com.au |
| Requirements | Bachelor's degree in Finance, Accounting, or related field; 5+ years of experience in financial analysis; Advanced Excel skills; Strong analytical and problem-solving abilities; Excellent communication and presentation skills |
| Benefits | Competitive salary; Performance-based bonuses; Opportunities for career advancement; Health and wellness benefits; Flexible work arrangements |
| Created at | 2024-04-29 10:00:00 |

# Appendix Z: Example Code

The example code is provided as a download from Blackboard.

The files include, but are not limited to:

* ReadMe.md starter
* Secrets.md starter
* .gitignore starter
* resources/views folders and files

Here is a list of the files and folders as uncompressed from the file SaaS-POR-Part-2-Base-Files-2024-S1.zip (or SaaS-POR-Part-2-Base-Files-2024-S1.7z):

### ReadMe.md

This ReadMe.md file is based on one by:

*Professional README Guide*. (n.d.). Coding-Boot-Camp.github.io. Retrieved April 15, 2024, from https://coding-boot-camp.github.io/full-stack/github/professional-readme-guide

You must complete the ReadMe.md as part of the assessment process.

### Secrets.md

The Secrets.md markdown file is based on the following provided code, and you must complete the information within as part of the assessment process.

# Secrets for Assessment  
  
*>* Name: YOUR NAME <STUDENTID@tafe.wa.edu.au>  
>   
> Campus: CAMPUS NAME  
  
## Database details:  
  
| Item | Value | Notes |  
|-------------------|-----------|--------------------------------------------|  
| Database Type | | SQLlite, MySQL, MariaDB, PostgreSQL, etc |  
| Host name | | localhost, 127.0.0.1, hostname.domain, etc |  
| Host port | | 3306, 27017, 6379, 11211, 5432, etc |  
| Database Name | | |  
| Database Username | | |  
| Database Password | | |

1. Setting up Git Bash in MS Terminal: <https://help.screencraft.net.au/hc/2680392001/65/add-git-bash-to-microsoft-terminal> [↑](#footnote-ref-2)
2. Install and Run Mailpit <https://help.screencraft.net.au/hc/2680392001/69/install-and-run-mailpit> and <https://help.screencraft.net.au/hc/2680392001/66/add-bash-command-line-aliases-for-git?category_id=35> [↑](#footnote-ref-3)