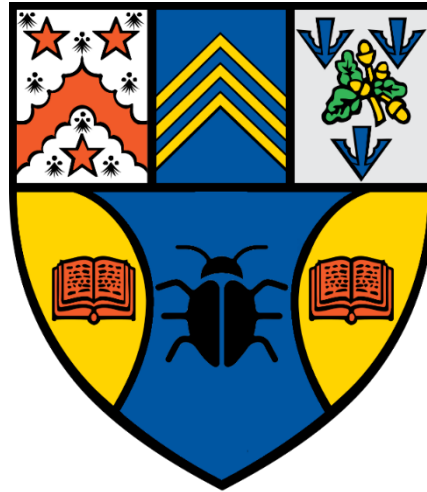


Project Documentation



Abertay
University

CMP311: Professional Project Development and Delivery

EH14: The Robots

Team Members: Connor Claypool, Gordon Deacon, Jack
Fairbairn, Euan Garden, Joseph Sweeney

Contents

.....	0
Project Documentation.....	0
User Guide	3
User Guide – Standard User.....	3
View Programs	3
Registration/Login.....	5
Submit a Report	6
Verify a report.....	8
Your Statistics.....	10
Leader board	10
Activity Stats.....	10
Your Awards.....	11
Reputation	11
Badges	11
Edit Details/Delete Account.....	12
2FA/Browser Sessions.....	13
User Guide - Vendor	14
Become a Vendor.....	14
Vendor Dashboard	15
User Guide - Administrator	17
Admin Dashboard	17
Managing Users	18
Developer Guide	21
Directory Structure	21
Deployment	21
Models	21
Report	21
Verification Batch.....	22
Verification Assignment.....	22
Verification Submission.....	22
Tip.....	22
User Metric Cache Entry	23

SkillTag	23
VendorRequest	23
Services	23
Difficulty calculators.....	24
Report Metrics	24
Tip Generators	24
User Metrics.....	25
Verification Assigners.....	25
Verification Evaluators.....	25
Gamify.....	25
Gamify Config.....	25
Badges	26
GiveReporterPointsListener	26
GiveVerifierPointsListener	26
Testing.....	27

User Guide

User Guide – Standard User

View Programs

You can view each program on the landing page of the website (Figure 1: View Programs), and each program can be viewed individually to see the full description of the program (). Each program has a button to join the selected programs. Another button is present to contact the vendor of the program.

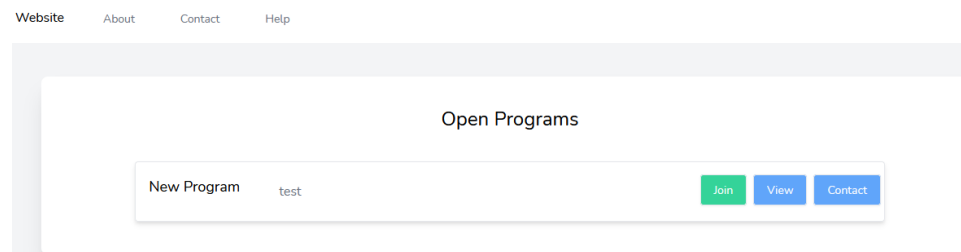


Figure 1: View Programs

As can be seen, each program is listed on the landing page of the web application.

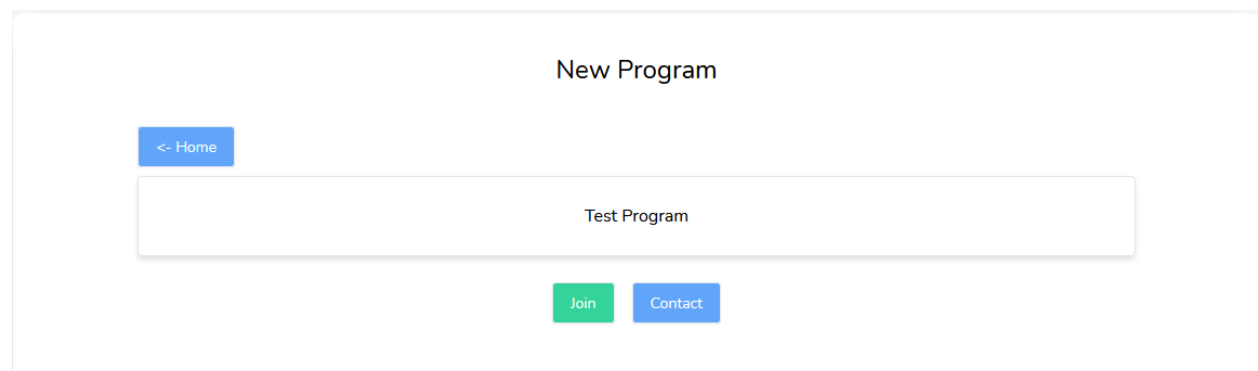


Figure 2: Program Description

Each program can be viewed individually to view the description, and the user can contact the vendor.

Contact Vendor

[< Home](#)

VENDOR:

Please select one

FULL NAME:

Enter your name...

EMAIL:

MESSAGE:

Enter your message...

Please fill in the form on the left to contact the vendor of the selected program.

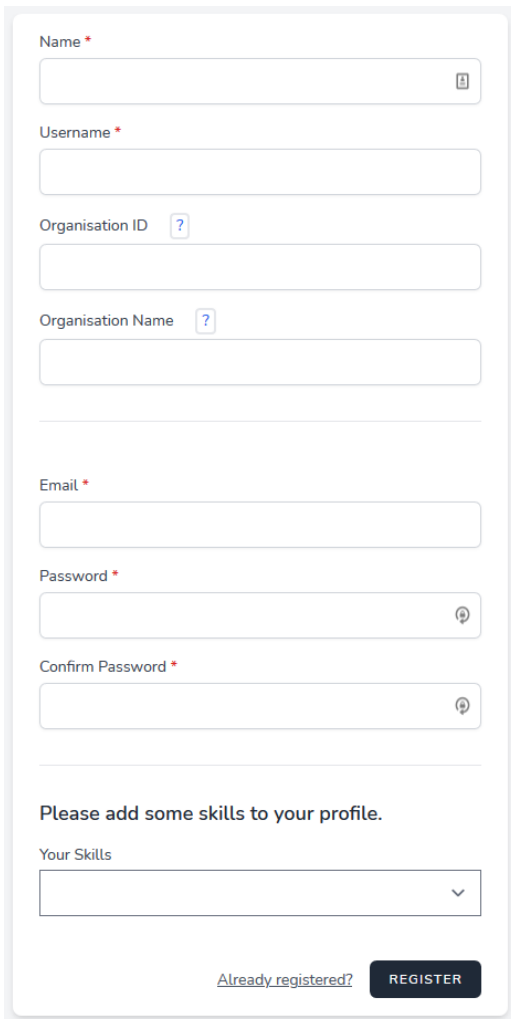
Once the Vendor has seen your message they will be in touch as soon as they can.

If you want to contact the site admins, please, [click here](#)

Submit

Figure 3: Contact Vendor

Registration/Login

A registration form with the following fields: Name (required), Username (required), Organisation ID (optional), Organisation Name (optional), Email (required), Password (required), and Confirm Password (required). Below these fields is a section titled "Please add some skills to your profile." with a "Your Skills" dropdown menu. At the bottom, there is a link "Already registered?" and a "REGISTER" button.

Name *

Username *

Organisation ID ?

Organisation Name ?

Email *

Password *

Confirm Password *

Please add some skills to your profile.

Your Skills

[Already registered?](#) **REGISTER**

Figure 4: Registration Form

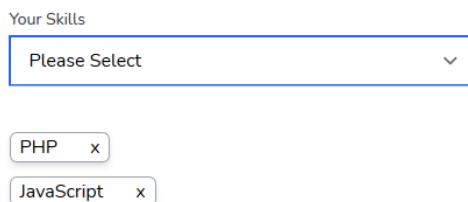
The registration form is very straightforward to use. Firstly, give your name and set a unique username. You may optionally give an organisation ID such as your student number or employee number. You may also optionally give your organisation name such as the name of your university or employer.

You must give a contact email address for your account, as well as set a strong user password. Your password must contain:

- An upper-case letter
- A lower-case letter
- A number
- A special character (eg. ! ? #)

Then ensure you type the correct password again to confirm your password.

Please add some skills to your profile.

A section for adding skill tags. It starts with a "Your Skills" dropdown menu showing "Please Select". Below it are two skill tags: "PHP" and "JavaScript", each with a small 'x' icon to remove it.

Your Skills

Please Select

PHP x

JavaScript x

Figure 5: Skill Tags

The final input is to give your profile some skill tags. This is optional and indicates areas of technology that you are comfortable with. Select using the dropdown and click on your applicable skills from the selection and add as many as you like. You can remove these by clicking the cross for that skill tag as displayed in Figure 5.

Once you are happy with your details, register and you will automatically be logged in. In future, you will log in to the site through the login page, which takes your email and password.

Submit a Report

To create a report, navigate to the Reports tab once logged in. Simply fill in the form displayed on this page to create a report. Firstly, select the relevant program and enter a title, as shown in Figure 6.

Website Report Verify Connor C ▾

Submit a Vulnerability Report

First, select which program your report applies to:
Be sure to double-check this is set correctly so your report is delivered to the correct vendor!

Program 52935 ▾

Give your report a short title:

A report

Figure 6: Vulnerability Reporting Form

Next, detail the steps involved in exploiting the given vulnerability, as shown in Figure 7. Steps can be written in Markdown, and the markdown preview for each step can be toggled using the Toggle Preview button located above each step. The Add Step button, situated at the bottom of the procedure area, can be used to add a new step, and the relevant Remove Step button can be used to remove a step.

Please detail the steps required to exploit this vulnerability:

Include enough information so that other users can reproduce your work, and split your procedure into manageable steps

Step 1

TOGGLE PREVIEW

REMOVE STEP

First, connect with ``nc 192.168.0.1 3456``

Step 2

TOGGLE PREVIEW

REMOVE STEP

Then run ``ls -la``

ADD STEP

Figure 7: Procedure Details in the Vulnerability Reporting Form

Once the procedure is complete, use the sliders below to indicate the given characteristics of the vulnerability, as shown in Figure 8. Finally, submit the report using the Submit Report button at the bottom of the form.

Use the sliders below to indicate some of the high-level characteristics of this vulnerability:

It's important to be accurate here, since the users verifying your report will also be assessing these characteristics

How severe is this vulnerability, on a scale of 1 to 5?

Severity: 2



How complex is this vulnerability to exploit, on a scale of 1 to 5?

Complexity: 4



How reliable is the procedure for exploiting this vulnerability, on a scale of 1 to 5?

Reliability: 3



Once you're ready, click here to submit your report!

SUBMIT REPORT

Figure 8: Vulnerability Metrics and Submit Button in the Reporting Form

Verify a report

To verify a report, navigate to the Verify tab. On the left is a list of pending and completed verifications. When selecting a complete verification, the procedure is shown, as shown in Figure 9. When selecting a pending verification, however, the verification form is displayed, as shown in Figure 10.

The screenshot shows the 'Verify' tab selected in the top navigation bar. The page title is 'Verify a Vulnerability Report'. On the left, there is a list of reports: 'A report' with status 'Complete'. On the right, under the heading 'Here is the procedure for this report:', there are two steps: 'Step 1' with the instruction 'First, connect with nc 192.168.0.1 3456' and 'Step 2' with the instruction 'Then run ls -la'.

Figure 9: A completed vulnerability verification form

The screenshot shows the 'Verify' tab selected in the top navigation bar. The page title is 'Verify a Vulnerability Report'. On the left, there is a list of reports: 'Another Report' with status 'Pending' and 'A report' with status 'Complete'. On the right, under the heading 'First, follow the steps below to see if you can reproduce this vulnerability report:', there are two steps: 'Step 1' with the instruction 'First, connect with nc 192.168.0.1 3456' and 'Step 2' with the instruction 'Then run ls -la'. At the bottom, there is a question 'Were you able to successfully verify this vulnerability report?' followed by an unchecked checkbox.

Figure 10: Vulnerability Verification Form

To complete a verification, first work through the procedure and indicate using the checkbox whether the vulnerability could be verified. If this checkbox is checked, the first set of sliders will appear, as shown in Figure 11; use these to rate the given characteristics of the vulnerability. Next, use the second set, shown in Figure 12, to rate the given characteristics of the report itself. Finally, use the Submit button to submit the report. Alternatively, the verification assignment can be cancelled using the Cancel button.

Great! Use the sliders below to indicate your assessment of some of the high-level characteristics of this vulnerability:

<p>How severe is this vulnerability, on a scale of 1 to 5?</p> <p>Severity: 1</p> <p><input type="range"/></p>	<p>How complex is this vulnerability to exploit, on a scale of 1 to 5?</p> <p>Complexity: 1</p> <p><input type="range"/></p>
<p>How reliable is the procedure for exploiting this vulnerability, on a scale of 1 to 5?</p> <p>Reliability: 1</p> <p><input type="range"/></p>	

Figure 11: Vulnerability Metrics in the Verification Form

Finally, rate the following aspects of the report itself:

<p>On a scale of 1 to 5, how would you rate the overall quality of this report?</p> <p>Quality: 1</p> <p><input type="range"/></p>	<p>On a scale of 1 to 5, how would you rate the level of detail included in this report?</p> <p>Detail: 1</p> <p><input type="range"/></p>
---	---

Once you're ready, click here to submit your verification!

SUBMIT

If verifying this report is beyond your skill set, you can also click below to cancel this assignment:

CANCEL ASSIGNMENT

Figure 12: Report Metrics, Submit Button and Cancel Button in the Verification Form

Your Statistics

Leader board

After completing reports and verifications on the web application reputation will be earned and given to the you. If you are performing well by reporting and verifying reports on a regular basis you may reach the top 10 of the users on the web application for reputation (XP). If so, you will be able to view yourself on your dashboard leader board with your username and reputation present. Figure 13: Leaderboard.

Leaderboard (Top 10)			
Rank	Username	XP	Awards
1	Reporter	1000	1000
2	Verifier1	100	100
3	Verifier2	100	100
4	Verifier3	100	100
5	diego59	0	0
6	dss	0	0
7	zbernier	0	0
8	veda92	0	0
9	ramona49	0	0
10	douglas.hessel	0	0

Figure 13: Leaderboard

Activity Stats

After completing reports and verifications on the web application, you will increase your number of completed reports and verifications. The activity graph shows your number of completed reports and verifications in the month you completed them allowing to track your activity on the web application. See After completing reports and verifications on the web application reputation will be earned and given to the you. If you are performing well by reporting and verifying reports on a regular basis you may reach the top 10 of the users on the web application for reputation (XP). If so, you will be able to view yourself on your dashboard leader board with your username and reputation present. See Figure 14: Activity.

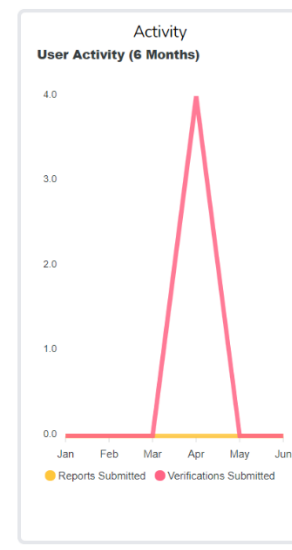


Figure 14: Activity

Your Awards

Reputation

On completion of the last verification for the report submitted, all users who took part on the reporting and verification process including yourself will be given points for taking part in the process as a reward. Each verifier will gain the same amount of reputation as the other. Reputation Gained will appear on your dashboard when signed in. Reputation is assigned automatically and can be viewed at the top righthand of your user dashboard, this can be seen under Figure 15: User's Reputation.

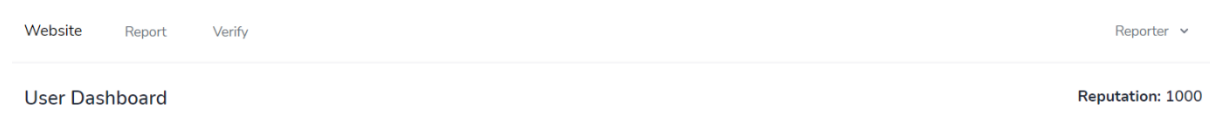


Figure 15: User's Reputation

Badges

On completion of the last verification for the report submitted, all users who took part on the reporting and verification process will have their stats checked. If you have reached a milestone for earning a badge once the verification process is finished the badge or badges will be assigned to your account.

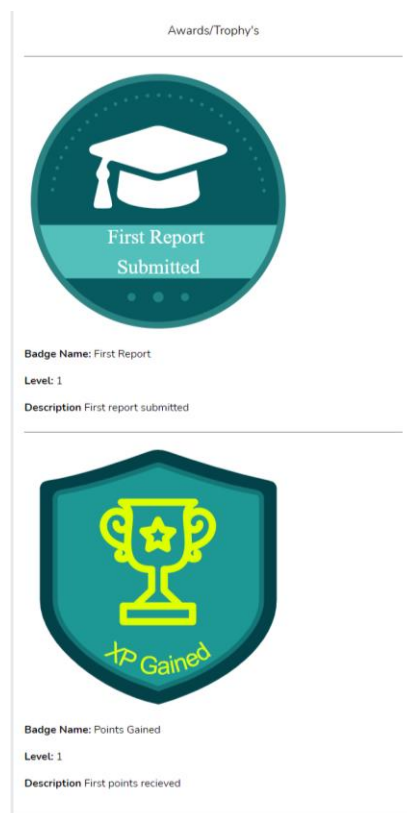


Figure 16 – User badges

Any badges that are gained by partaking in reporting and verifying will appear on your dashboard. The badges that appear in the dashboard as seen in **Error! Reference source not found.** will have information such as:

- Unique badge icon
- Badge name
- Badge level
- Badge description

The badges for each award will have a unique icon. The icon will change with each level of the badge. Each badge will have a description that informs you of how you achieved that badge.

Badges can be awarded for:

- Number of verifications completed
- Number of reports submitted
- Amount of total reputation gained

As you progress, the badges that are more challenging may be unlocked for consistent partaking in reporting and verifying.

Edit Details/Delete Account

If you want to update your details, you must navigate to your user profile. This is found on the navigation bar on the right-hand side, where your username is displayed. Click this and a dropdown will appear showing 'profile'.

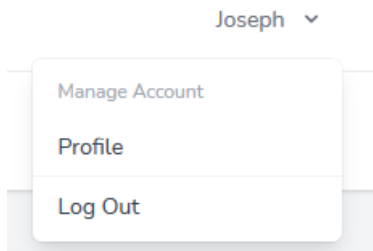


Figure 17: Profile Dropdown

Make any changes you wish to make, and click save.

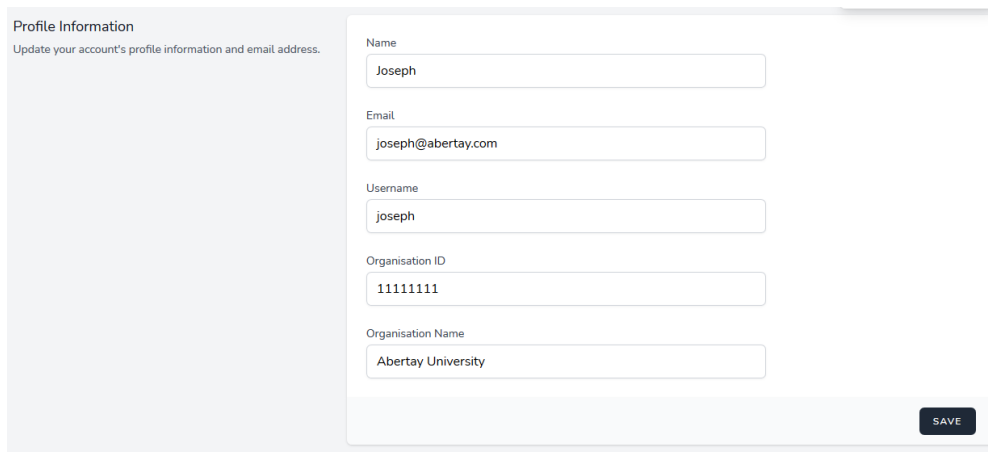
A screenshot of the 'Edit Profile' form. On the left, a sidebar titled 'Profile Information' contains the text 'Update your account's profile information and email address.' The main form area has several input fields: 'Name' (containing 'Joseph'), 'Email' (containing 'joseph@abertay.com'), 'Username' (containing 'joseph'), 'Organisation ID' (containing '11111111'), and 'Organisation Name' (containing 'Abertay University'). A 'SAVE' button is located at the bottom right of the form.

Figure 18: Edit Profile

If you want to delete your account, this can be found at the bottom of the profile information page. You will receive a warning message to confirm that you wish to delete your account.

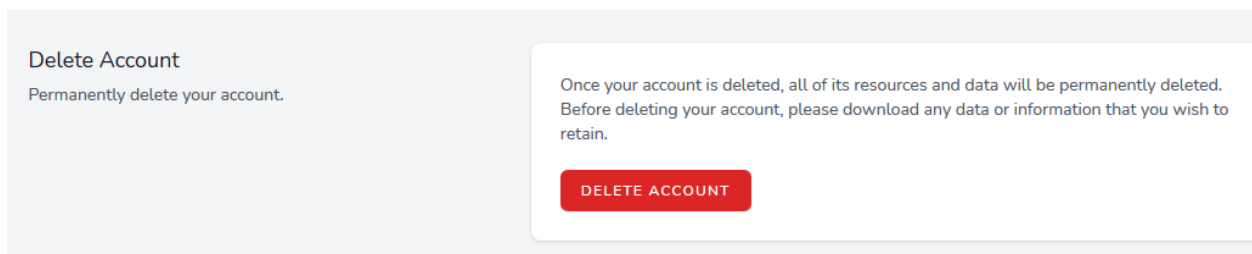
A screenshot of the 'Delete Account' confirmation page. On the left, the title 'Delete Account' is followed by the text 'Permanently delete your account.' On the right, a white box contains a warning: 'Once your account is deleted, all of its resources and data will be permanently deleted. Before deleting your account, please download any data or information that you wish to retain.' Below this warning is a red button labeled 'DELETE ACCOUNT'.

Figure 19: Delete Account

2FA/Browser Sessions

2FA support as well as browser session logout functionality can also be found under profile information.

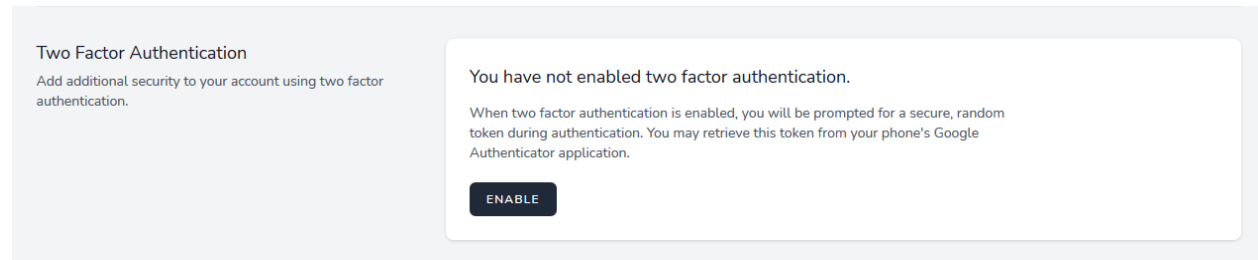


Figure 20: 2FA

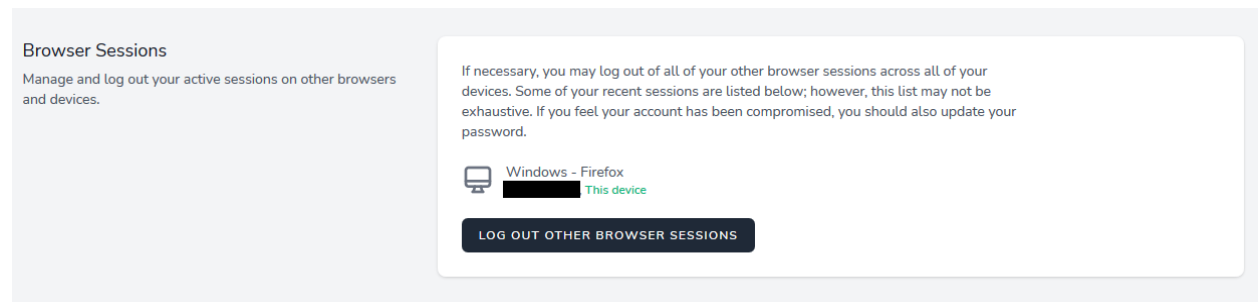


Figure 21: Browser Sessions

User Guide - Vendor

Become a Vendor

Vendor status allows you to create your own bug bounty programs. If you want to become a vendor, the first step for this is navigating to your profile information (See Edit Details). Figure 22 shows the button to become a vendor and pressing this will show the dialog box displayed in Figure 23. Remember, if you want to apply for vendor its best to use an organisation name.

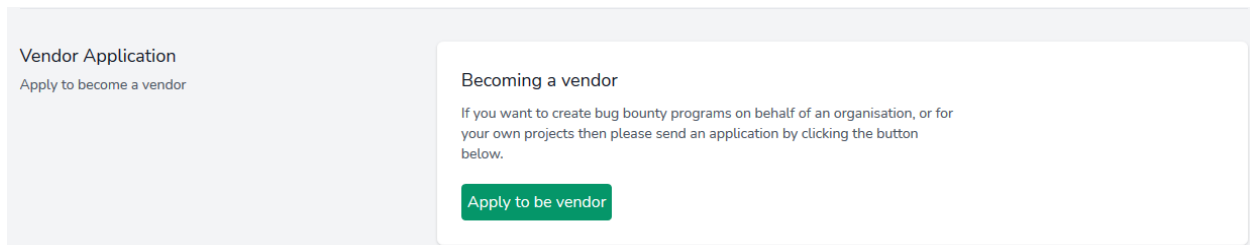


Figure 22: Application for Vendor

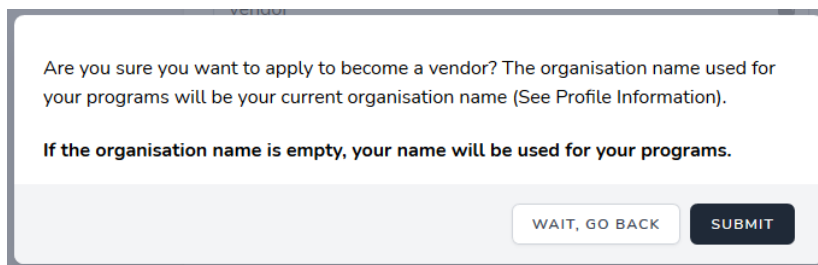


Figure 23: Alert Message

An administrator can approve your request to become a vendor. If approved, you will see a new item appear on your dashboard navigation as shown in Figure 24.

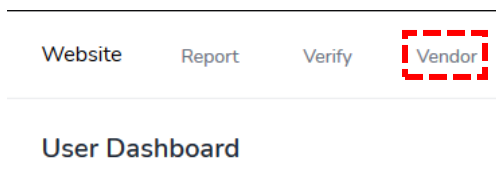
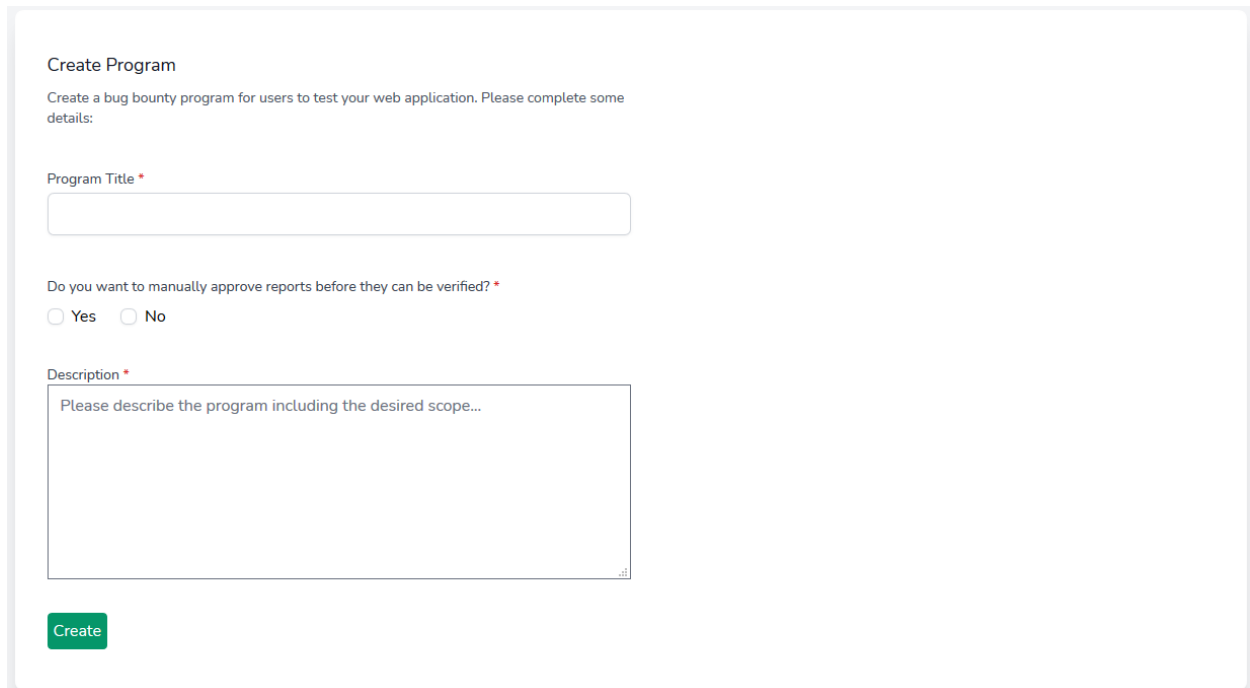


Figure 24: navigation update for vendor

You can then navigate to the vendor dashboard.

Vendor Dashboard

The vendor dashboard allows you to have control of your own bug bounty programs for other users to submit and verify reports for. To start off as a vendor, create a bug bounty program using the create program form, shown in Figure 25.



The form is titled "Create Program" and includes a sub-header: "Create a bug bounty program for users to test your web application. Please complete some details:". It contains three main input sections: a "Program Title" text field, a "Do you want to manually approve reports before they can be verified?" section with "Yes" and "No" radio buttons, and a "Description" text area with a placeholder "Please describe the program including the desired scope...". A green "Create" button is located at the bottom left of the form.

Create Program

Create a bug bounty program for users to test your web application. Please complete some details:

Program Title *

Do you want to manually approve reports before they can be verified? *

☐ Yes ☐ No

Description *

Please describe the program including the desired scope...

Create

Figure 25: Create a program

Firstly, give your program a unique title. The second part of this form asks whether you want to manually approve reports before they can be verified. This allows you to have more control over the program, ensuring that you're happy with the quality of a report and that the report is within the scope you have assigned. Finally, give your program a description and make sure to include the scope of the bug bounty. This helps to ensure that your product is tested in the way that you want.

Once you have created a program, you can also edit or delete the program to reflect any changes you need to make. You can also delete a program. This is shown in Figure 26.

Your Programs

New Program

Delete

Program Title *

New Program

Do you want to manually approve reports before they can be verified? *

☐ Yes

☐ No

Description *

This is a brand new program

Update

Figure 26: Your programs

If your bug bounty program is set for manual approval of reports before verification, make sure you check the vendor dashboard regularly. Figure 27 shows the different aspects of the report display.

Submitted Reports

REPORT TITLE	DATE SUBMITTED	APPROVED FOR VERIFICATION	VERIFIED STATUS	Actions
This is a new report	2021-04-24T22:16:32.000000Z	Approve	pending	View Report

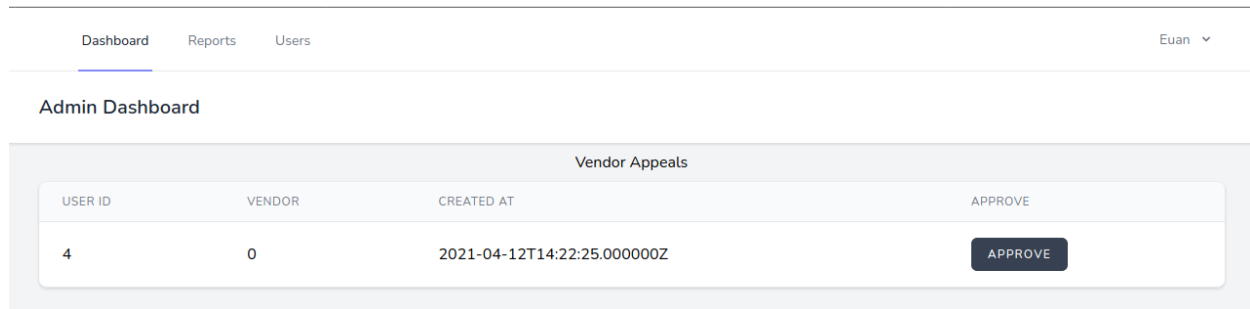
Figure 27: Reports view

In the example shown, this report requires verification so first view the report, by clicking the **view report** button in the **Actions** column. When satisfied with the report content, click **Approve** in the **Approved for verification** column. This will allow verifiers to approve the submitted report.

User Guide - Administrator

Admin Dashboard

The admin dashboard includes the functionality to approve users to become a vendor user type. Once the request has been made, the request will appear on the admin dashboard in a tabular view. To approve the user so that the user becomes a vendor, the approve button should be pressed. The vendor request can be seen in Figure 28.



Vendor Appeals			
USER ID	VENDOR	CREATED AT	APPROVE
4	0	2021-04-12T14:22:25.000000Z	APPROVE

Figure 28: Vendor Request

Managing Users

The users page becomes visible when on an administrator account. Here the users can be managed through the following methods:

- Create
- Update (edit)
- Delete

To create a user, simply click the create user button above the users table on the users tab as seen in Figure 29.

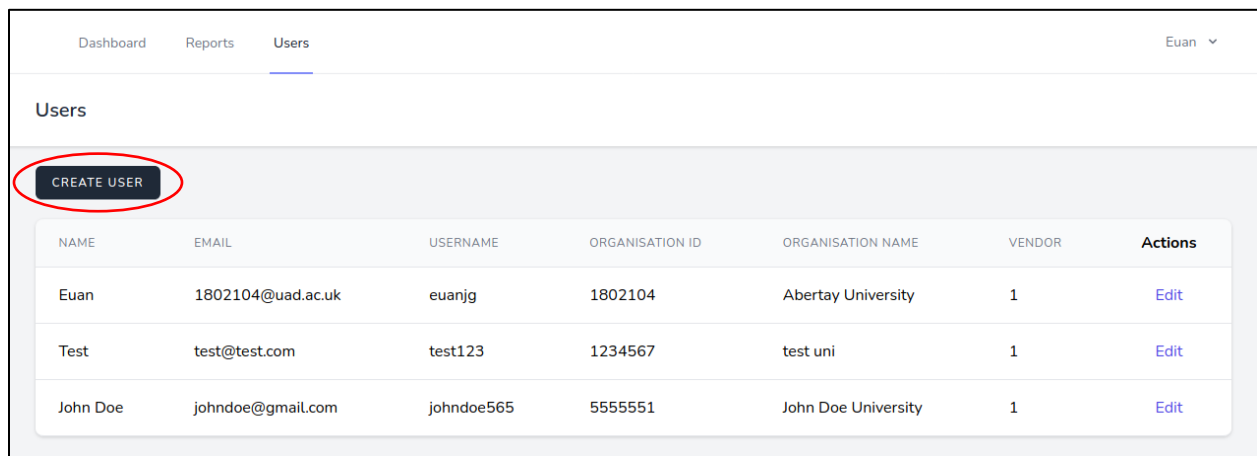
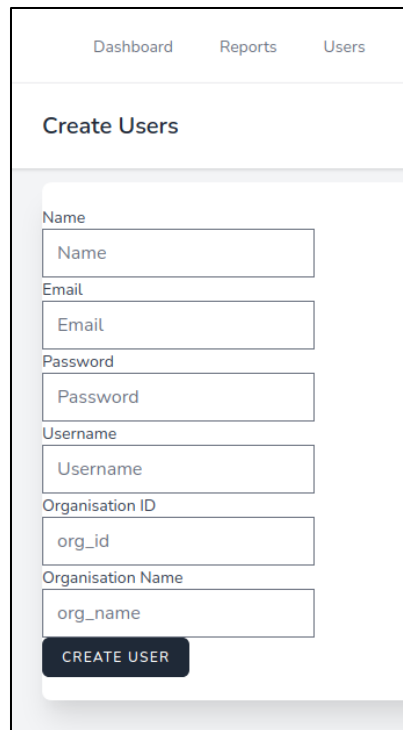


Figure 29: Create User Button

This will redirect the user to the create user form where the following details are required: Name, Email, Password, Username, Organisation ID and Organisation Name. Once these fields have been filled out, the create user button can be pressed and the user will be added to the database. The form to fill in the user details can be seen below in Figure 30.

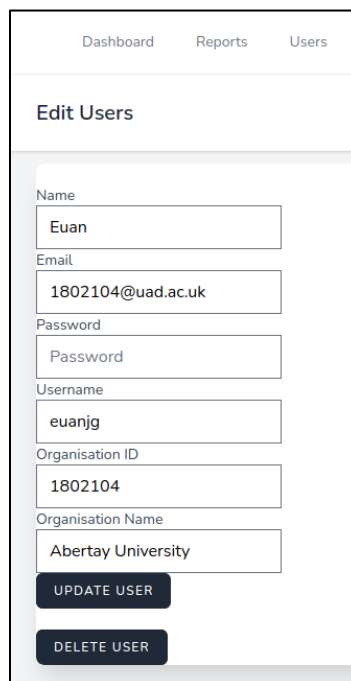


The 'Create Users' form is located within a navigation bar containing 'Dashboard', 'Reports', and 'Users' links. The form itself has a title 'Create Users' and a light gray background. It contains seven text input fields, each with a label above it: 'Name', 'Email', 'Password', 'Username', 'Organisation ID', and 'Organisation Name'. Below the 'Organisation Name' field is a dark blue button labeled 'CREATE USER'.

Field Label	Field Value
Name	Name
Email	Email
Password	Password
Username	Username
Organisation ID	org_id
Organisation Name	org_name
Action	CREATE USER

Figure 30: Create User Form

To edit a user's information, the edit option under 'Actions' column can be clicked, and this will link the user to the edit users form. Here, the user's information can be changed and updated accordingly once the data has been entered and the update user button has been pressed. The edit users form can be seen below in Figure 31.

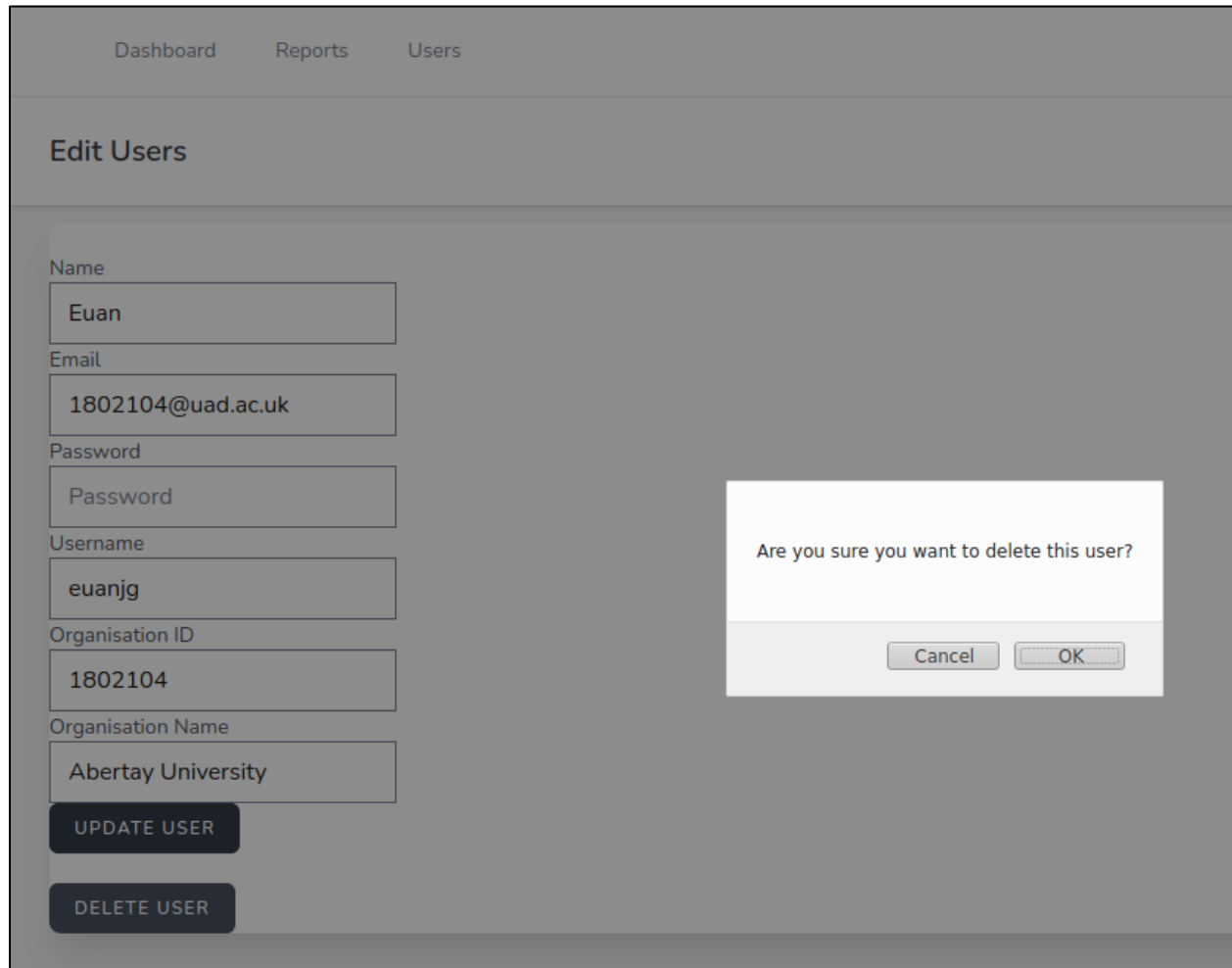


The 'Edit Users' form is located within a navigation bar containing 'Dashboard', 'Reports', and 'Users' links. The form has a title 'Edit Users' and a light gray background. It contains seven text input fields, each with a label above it: 'Name', 'Email', 'Password', 'Username', 'Organisation ID', and 'Organisation Name'. Below the 'Organisation Name' field are two dark blue buttons: 'UPDATE USER' and 'DELETE USER'.

Field Label	Field Value
Name	Euan
Email	1802104@uad.ac.uk
Password	Password
Username	euanjg
Organisation ID	1802104
Organisation Name	Abertay University
Action 1	UPDATE USER
Action 2	DELETE USER

Figure 31: Edit User Form

The final method to managing users is a delete function. This allows the admin to delete the selected user from the database. This can be accessed once the edit option is clicked and the delete user button is pressed once the user is navigated to the edit user form. A pop-up is shown once this button is clicked to display a final warning before the user is deleted. An example of a user being deleted is shown in Figure 32 below.



The screenshot shows a web application interface with a top navigation bar containing 'Dashboard', 'Reports', and 'Users'. Below this is a header section titled 'Edit Users'. The main content area contains a form with the following fields and values:

- Name: Euan
- Email: 1802104@uad.ac.uk
- Password: Password
- Username: euanjg
- Organisation ID: 1802104
- Organisation Name: Abertay University

At the bottom of the form are two buttons: 'UPDATE USER' and 'DELETE USER'. A confirmation dialog box is displayed over the form, asking 'Are you sure you want to delete this user?' with 'Cancel' and 'OK' buttons.

Figure 32: Delete User Option

Developer Guide

This application was built using the Laravel PHP framework, the Laravel Jetstream starter kit with Inertia JS, Vue JS and Tailwind CSS. The documentation for each of these frameworks can be found below:

- Laravel 8: <https://laravel.com/docs/8.x>
- Laravel Jetstream: <https://jetstream.laravel.com/1.x/introduction.html>
- Inertia JS : <https://inertiajs.com/>
- VueJS 3: <https://v3.vuejs.org/>
- Tailwind CSS: <https://tailwindcss.com/docs>

Directory Structure

The application backend is organized in the same way as a standard Laravel application; see the Laravel documentation at <https://laravel.com/docs/8.x/structure> for details. The frontend components of the application are stored under 'resources/js'.

Deployment

The application can be deployed similarly to any standard Laravel application; see the Laravel documentation at <https://laravel.com/docs/8.x/deployment> for details. The most important factors are that the database connection and queue driver are correctly set in the application's '.env' file. Additionally, to enable background jobs to work correctly, Supervisor can be used on a Linux server to keep the 'queue:work' process running. See the Laravel documentation at <https://laravel.com/docs/8.x/configuration> for details on configuring the application using its '.env' file, and at <https://laravel.com/docs/8.x/queues> for configuring background jobs.

Models

This section describes the database models used in the application; each model corresponds to a database table. In addition to the fields listed for each model below, every table includes an 'id' field for the primary key, plus 'created_at' and 'updated_at' timestamps.

Program

The 'Program' model corresponds to the 'programs' table and is used to store bug bounty program details. Its fields are as follows:

- 'Title' – The program's title
- 'Description' – A longer description of the program
- 'VendorID' – The user ID of the vendor managing this program
- 'Exclusive' – Whether this is a public program or not
- 'vendorApproval' – Whether vendor approval is required before reports are assigned to verifiers

Report

The 'Report' model corresponds to the 'reports' table and is used to store submitted vulnerability reports. Its fields are as follows:

- 'program_id' – A foreign key referencing the program the report is for
- 'title' – The report's title
- 'procedure' – A JSON array containing the steps in the report's procedure

- 'metrics' – A JSON object containing each of the submitted vulnerability metrics
- 'creator_id' – A foreign key referencing the user who created the report
- 'status' – An enum whose value is either 'pending', 'verified' or 'rejected'

Verification Batch

The 'VerificationBatch' model corresponds to the 'verification_batches' table and is used to group one set of verifiers who have been assigned to verify a given report. Its fields are as follows:

- 'report_id' – A foreign key referencing the report to be verified
- 'status' – An enum whose value is either 'pending', 'accepted' or 'reassigned'. This status relates to the evaluation of the verifications themselves, and not the report.
- 'completed_at' – The time at which the final verification in the batch was completed. Null until the batch is complete.
- 'voted_vulnerability_metrics' – A JSON object containing the average voted value for each of the vulnerability metric values submitted by the verifiers. Null until the batch is complete.
- 'voted_procedure_metrics' – A JSON object containing the average voted value for each of the procedure metric values submitted by the verifiers. Null until the batch is complete.

Verification Assignment

The 'VerificationAssignment' model corresponds to the 'verification_assignments' table and is used to store the individual verification assignments in a verification batch. Its fields are as follows:

- 'verification_batch_id' – A foreign key referencing the verification batch this assignment is part of
- 'assignee_id' – A foreign key referencing the user to whom this verification is assigned.
- 'status' – An enum whose value is either 'pending', 'complete' or 'cancelled'
- 'actioned_at' – The time at which this assignment was either completed or cancelled.

Verification Submission

The 'VerificationSubmission' model corresponds to the 'verification_submissions' table and is used to store submitted verification forms. Its fields are as follows:

- 'verification_assignment_id' – A foreign key referencing the corresponding verification assignment.
- 'verifiable' – A Boolean value indicating whether the user was able to verify the report.
- 'vulnerability_metrics' – A JSON object containing the values submitted for each vulnerability metric.
- 'procedure_metrics' – A JSON object containing the values submitted for each procedure metric.

Tip

The 'Tip' model corresponds to the 'tips' table and is used to store tips generated by the application. Its fields are as follows:

- 'user_id' – A foreign key referencing the user this tip is for.
- 'type' – The class name of the tip generator class which created this tip
- 'location_tag' – A string indicating where to display this tip. This can be used to select and pass only the relevant tips to a given view.

- 'content' – The content of the tip itself.
- 'read_at' – The time at which the tip was displayed to the user. Null until the tip has been read.

User Metric Cache Entry

The 'UserMetricCacheEntry' model corresponds to the 'user_metric_cache' table and is used to store the last calculated value for user metrics. This table is only populated when the relevant event occurs, so will most likely not contain the value of every metric for every user. Its fields are as follows:

- 'user_id' – A foreign key referencing the relevant user
- 'metric' – The class name of the relevant 'UserMetric' class
- 'value' – The value of the metric

SkillTag

The 'SkillTag' model corresponds to the 'skill_tags' table. This is used to store the self-evaluated skills that a user can assign themselves on registering or when editing their profile. Currently this is not used for particular actions for that user but allows for future implementation of 'recommended programs' for that user.

Its fields are as follows:

- tag_id – a unique integer acting as primary key to uniquely identify this skill tag. Note that in this case, tag_id replaces 'generic' id although this will likely be altered in future iterations.
- tag_name – the name that will be displayed to the user for the skill tag i.e. "JavaScript"

VendorRequest

The 'VendorRequest' model corresponds to the 'vendor_requests' table. This is used to store user requests to become a vendor, for admin to verify.

Its fields are as follows:

- user_id – the user requesting to become a vendor
- approved – Boolean indicating whether approved or not

Services

Much of the business logic of the application is contained in so-called Services, which can refer to either an abstract base class representing a specific service type, or a concrete implementation of one of these.

Within the 'app/Services' directory, several subdirectories can be found, each of which contains an abstract base class for that service type as well as its implementations. The concrete implementation or implementations which are actually used by the application for each service are specified in the config file 'config/bugbounty.php'.

This config file is processed by the bug bounty service provider in 'app/Providers/bugbounty.php', which registers each selected concrete implementation to ensure that when some code requests a given base

service class, the correct implementation is provided. The registration and requests occur via Laravel's service container and service provider system, which enable automatic dependency injection and the dynamic binding of concrete classes to abstract base classes or interfaces. As the full details of Laravel's service container and service provider system are beyond the scope of this document, see Laravel's documentation at <https://laravel.com/docs/8.x/container> and <https://laravel.com/docs/8.x/providers> for details.

Each of the individual service types are covered in more detail below.

Difficulty calculators

Difficulty calculators are service classes used to assess the difficulty of verifying a report, to aid in the assignment of reports to verifiers. The main location to use these would be within verification assignment code. To create a difficulty calculator, create a subclass of 'DifficultyCalculator' and implement the 'calculateDifficulty' method, which takes a 'Report' model as a parameter and returns a numeric value. The active difficulty calculator implementation is specified in the config file 'bugbounty.php', as discussed above, in the key 'difficultyCalculator'.

Report Metrics

Report metrics are split into two categories: vulnerability metrics and procedure metrics. Vulnerability metrics are indicated by the reporter in the reporting form as well as by the verifier in the verification form, and describe the characteristics of a vulnerability, such as complexity or severity. Procedure metrics are only indicated by the verifier in the verification form, and represent aspects of the report itself, such as quality or detail.

To create a report metric, create a subclass of the relevant intermediary class, 'ProcedureMetric' or 'VulnerabilityMetric', and set the relevant fields:

- 'name', the name of the metric.
- 'title', the title of the metric to be displayed on the relevant form.
- 'extra', a longer subtitle to be displayed on the relevant form.
- 'min', the minimum value of the metric.
- 'max', the maximum value of the metric.

The list of active vulnerability metrics and that of active procedure metrics, those used in the relevant forms, are set using the config fields 'vulnerabilityMetrics' and 'procedureMetrics', respectively.

Tip Generators

Tip generators are used to create customized tips when a relevant event occurs. To implement a tip generator, create a subclass of 'TipGenerator' and create handler methods for each relevant event. Each of these handler methods should receive an instance of the given event as their only parameter and return an array of 'Tip' models to be inserted into the database. The names of these handler methods do not matter, as they do not override any base class methods. To register these handler methods, override the base class property 'eventHandlers', an associative array where the keys are the event classes and the values are the names of the corresponding handler methods. Finally, to set the list of active tip generators, use the 'tips' config key.

User Metrics

User metrics are used to record a specific characteristic of each user, such as their level of activity or their average report quality. To implement a user metric, create a subclass of the 'UserMetric' class and implement the 'compute' method, which takes an array of user IDs as its only argument and returns an associative array where the keys are the user IDs and the values are the metric values. Next, methods to determine the users for which the metric should be updated at a given event should be created. These methods should take the relevant event as their only argument and return an array of user IDs. These methods should then be registered by overriding the base class property 'getUserMethods'. This field is an associative array where the keys are the event classes and the values are the names of the corresponding methods which calculate which users' the metric should be updated for. The list of active user metrics can be set in the config key 'userMetrics'.

Verification Assigners

Verification assigners are used to choose which users should be assigned to verify a given report. To implement a verification assigner, create a subclass of 'VerificationAssigner' and implement the 'selectVerifiers' method, which takes an Eloquent query builder as its argument and returns a list of 'User' models. The query builder argument will select every eligible user, i.e. users who did not submit the given report, have never previously been assigned to it and are not the program vendor. This method can use this query builder to further filter, prioritize and select a given number of eligible users to assign the report to. To set the active verification assigner, use the config key 'verificationAssigner'.

Verification Evaluators

Verification evaluators are used to check the results of each verification for a report before either marking that report as accepted or rejected, or re-assigning it to new verifiers if there was a problem with the verification itself. To implement a verification evaluator, create a subclass of 'VerificationEvaluator' and implement the 'evaluateVerifications' and 'evaluateReport' methods, which both take a 'VerificationBatch' model as an argument and return true or false. The former of these methods determines whether to trust the verifications themselves, and thus whether to re-assign the report to new verifiers, and also synthesizes the results of the verifiers submitted report metrics. This method should return 'true' if the verifications are to be accepted. If this is the case, the latter method is then used to determine whether the report is marked as verified or rejected and should return 'true' if the report is accepted. To set the currently active verification evaluator, use the config key 'verificationEvaluator'.

Gamify

The folder gamify in the 'app' folder contains both Badges and Points folders for the gamification elements of the web application. All badges and points created will appear in their respective folders. The command to create a new badge is 'php artisan gamify:badge BadgeName' and the command to create a new point is 'php artisan gamify:point PointName'.

Gamify Config

Gamify is a config file created from the vendor package qcod/aravel-gamify. The config file dictates the model who will receive reputation, the channels in which user reputation will be broadcast to, models used for both badges and reputation and the levels a badge can be.

Badges

New badges created will appear in the Badges folder. The badges' name, description level and icon can all be set within the badge class. The Badges default name will be the pretty version of the badge class name. The badge name can be set by '\$name' within the class or changed by 'getName()' to name the badge dynamically. Badge icon can be set similarly to name with '\$icon' or by 'getIcon()' to change the icon dynamically. Badge level can also be set with '\$level' or 'getLevel()', the default badge level will be 1 as configured in the 'config/gamify.php'. The qualifier function 'qualifier(user)' checks against the user if they have the correct number of points, reports and verified reports. Cache must be cleared whenever any changes have been made or if badge is created or removed. The command for clearing cache is 'php artisan cache:forget gamify.badges.all'.

GiveReporterPointsListener

This listener has access to the VerificationBatchComplete Event. Once the event is run after all verifiers have completed their verification the listener will give points to the reporter. Points given can be changed within the listener class.

GiveVerifierPointsListener

This listener has access to the VerificationBatchComplete Event. Once the event is run after all verifiers have completed their verification the listener will give points to each verifier for the verifier batch. Points given can be changed within the listener class.

Testing

Table 1: Automated test results

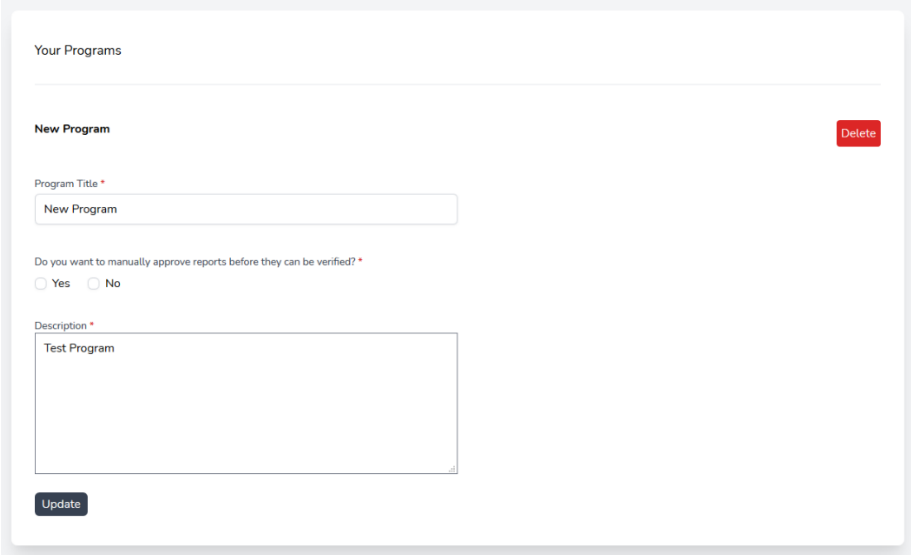
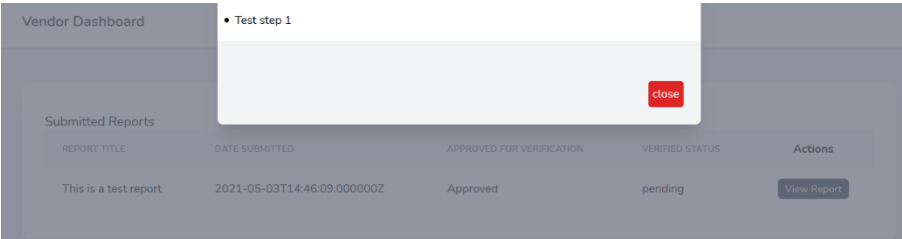
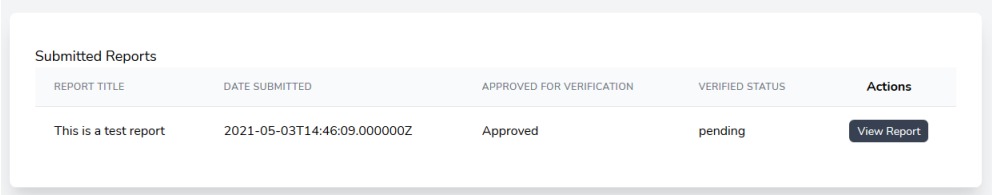

Test	Description	Passing
Report can be created	Tests that users can successfully submit a report form and have it inserted into the database.	Yes
Report is assigned to verifiers	Tests that the correct verification batch and verification assignments are created once a report is submitted.	Yes
Tips are generated when report submitted	Tests that the correct tips are generated when a report is submitted.	Yes
Tips are generated when verification submitted	Tests that the correct tips are generated when a verification is submitted.	Yes
Tips are generated when verification batch is completed	Tests that the correct tips are generated when a verification batch has been completed.	Yes
Metrics are updated when report submitted	Tests that the correct metrics are updated when a report is submitted.	Yes
Metrics are updated when verification submitted	Tests that the correct metrics are updated when a verification is submitted.	Yes
Metrics are updated when verification batch is completed	Tests that the correct metrics are updated when a verification batch is completed.	Yes
Users can verify a report	Tests that users can submit a report form and have this entered into the database and processed correctly.	Yes


Table 2: Manual test results


Test	Description	Passing	Screenshot
Users can be created	Tests that users can be created once the form is filled out and is added to the users table	Yes	
Users can be updated	Tests that user's information can be edited and is updated in the users table	Yes	
Users can be deleted	Tests that users can be deleted and is removed from the users table	Yes	

Vendors can be approved	Tests that the approve vendor request method is successful and the user can become a vendor	Yes	<div><div>Vendor Appeals</div><table><thead><tr><th>USER ID</th><th>VENDOR</th><th>CREATED AT</th><th>APPROVE</th></tr></thead><tbody><tr><td>4</td><td>0</td><td>2021-04-12T14:22:25.000000Z</td><td><div>APPROVE</div></td></tr></tbody></table><div></div><div>↓</div><div><div>Vendor Appeals</div><table><thead><tr><th>USER ID</th><th>VENDOR</th><th>CREATED AT</th><th>APPROVE</th></tr></thead><tbody><tr><td>4</td><td>1</td><td>2021-04-12T14:22:25.000000Z</td><td><div>APPROVE</div></td></tr></tbody></table></div></div>	USER ID	VENDOR	CREATED AT	APPROVE	4	0	2021-04-12T14:22:25.000000Z	<div>APPROVE</div>	USER ID	VENDOR	CREATED AT	APPROVE	4	1	2021-04-12T14:22:25.000000Z	<div>APPROVE</div>
USER ID	VENDOR	CREATED AT	APPROVE																
4	0	2021-04-12T14:22:25.000000Z	<div>APPROVE</div>																
USER ID	VENDOR	CREATED AT	APPROVE																
4	1	2021-04-12T14:22:25.000000Z	<div>APPROVE</div>																
User can log in	Tests that a user can use their credentials to successfully login	Yes	<div><div>Email</div><div>joseph@abertay.com</div><div>Password</div><div>••••••••</div><div><input type="checkbox"/> Remember me</div><div>Forgot your password?</div><div>LOG IN</div><div>joseph ▾</div></div>																
User can register	Tests that a user can successfully register using their personal details	Yes	<div><div>Name *</div><div>Registered</div><div>Username *</div><div>register</div><div>Organisation ID ?</div><div>43214321</div><div>Organisation Name ?</div><div>Abertay University</div><div>Email *</div><div>register@abertay.com</div><div>Password *</div><div>••••••••</div><div>Confirm Password *</div><div>••••••••</div><div>Please add some skills to your profile.</div><div>Your Skills</div><div></div><div>Already registered?</div><div>REGISTER</div><div>Registered ▾</div></div>																
Register password strength	Tests that a user must give a strong password	Yes	<div><div>Whoops! Something went wrong.</div><div><ul style="list-style-type: none">The password must be at least 10 characters and contain at least one uppercase character, one number, and one special character.</div></div>																

Register password match	Tests that confirm password and password must match	Yes	<div>Whoops! Something went wrong.</div> <div><div>validation.confirmed</div></div>
Required fields	Tests that register form required fields work as intended	Yes	N/A
User edit	Tests that a user can edit account details	Yes	<div><div><div>Name</div><div>joseph</div></div><div><div>Email</div><div>joseph@abertay.com</div></div><div><div>Username</div><div>joseph</div></div><div><div>Organisation ID</div><div>11111111</div></div><div><div>Organisation Name</div><div>Abertay University</div></div></div> <div><div><div>Name</div><div>joseph-bugs</div></div><div><div>Email</div><div>joseph@abertaybugbounty.com</div></div><div><div>Username</div><div>joseph</div></div><div><div>Organisation ID</div><div>11111111</div></div><div><div>Organisation Name</div><div>Abertay University</div></div><div>Saved.</div></div>
Vendor Application	Tests that a user can request to become a vendor	Yes	N/A
Change password	Tests that a user can change their password	Yes	N/A

Vendor create program	Tests that a vendor can create a program	Yes	
Vendor Edit/Delete program	Tests that a vendor can delete or edit their programs	Yes	N/A
Vendor view report	Tests that a vendor can view submitted reports for their programs	Yes	
Vendor Approve for verification	Tests that a vendor can approve reports for verification where the program requires it	Yes	
Gamification Reporter Receives Reputation	Test that a reporter gains points when a verification batch is finished	Yes	

			<div>Reporter ▾</div> <div>Reputation: 1000</div>
Gamification Reporter Receives Badges	Test that a reporter gains points when a verification batch is finished	Yes	<div>Awards/Trophy's</div> <div><div>Awards/Trophy's</div><div><p>First Report Submitted</p></div><div>Badge Name: First Report</div><div>Level: 1</div><div>Description First report submitted</div></div>
Gamification Verifier Receives Reputation	Test that a verifier gains points when a verification batch is finished	Yes	<div>Verifier 1 ▾</div> <div>Reputation: 0</div> <div>Verifier 1 ▾</div> <div>Reputation: 100</div>
Gamification Verifier Receives Badges	Test that a verifier gains points when a verification batch is finished	Yes	<div>Awards/Trophy's</div> <div></div>

			<div><div>Awards/Trophy's</div><div></div><div><div>Badge Name: Points Gained</div><div>Level: 1</div><div>Description First points recieved</div></div></div>	
--	--	--	---	--