WPD2 Coursework Report

Project Title: Milestones System

Date: 24/04/2018

Team A

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Student ID** | **Email** |
| William Thomson | S1426481 | WTHOMS204@caledonian.ac.uk |
| Alex McBride | S1715224 | AMCBRI206@caledonian.ac.uk |
| Yakoob Hayat | S1714096 | YHAYAT200@caledonian.ac.uk |
| Mie Tanaka | S1716578 | MTANAK200@caledonian.ac.uk |

“I declare that all work submitted for this coursework is the work of Alex McBride, William Thomson, Mie Tanaka and Yakoob Hayat alone unless stated otherwise.”

Table of Contents

[1 Introduction 3](#_Toc512746626)

[2 Architectural Overview 3](#_Toc512746627)

[3 Application URLs 4](#_Toc512746628)

[3.1 URL Format 4](#_Toc512746629)

[3.2 Mapping of Links to Functionality 5](#_Toc512746630)

[4 Persistence 6](#_Toc512746631)

[4.1 Database Schema 6](#_Toc512746632)

[4.2 Java Classes 7](#_Toc512746633)

[5 Share Project Functionality 9](#_Toc512746634)

[6 Security Overview 9](#_Toc512746635)

[7 Test Reports 11](#_Toc512746636)

[7.1 Unit Tests 11](#_Toc512746637)

[7.2 Integration Tests 13](#_Toc512746638)

[7.2.1 Test Login / Register 13](#_Toc512746639)

[7.2.2 Test Projects 21](#_Toc512746640)

[7.2.3 Test Milestones 29](#_Toc512746641)

[7.2.4 Test Share Project 39](#_Toc512746642)

[7.3 Summary of Test Results 46](#_Toc512746643)

[8 References 47](#_Toc512746644)

[9 Table of Figures 48](#_Toc512746645)

# Introduction

This report covers the design and development of a web application to track milestones in a project (see Figure 1). It was created as part of the coursework for the Web Platform Development 2 module at Glasgow Caledonian University. The application was to be developed using Java, fulfilling the brief below.

“The features of a milestone (object) can be designed by you but should at least consist of:

* A description of the milestone
* An intended due date and
* The actual completion dates.
* Milestones can be removed from the list.
* Milestones can be edited.
* A milestone (list) can be shared with friends using a link.

Any application features not specified can be designed by you.”

The source code repository can be found at: <https://bitbucket.org/wpd2gcu/milestones>

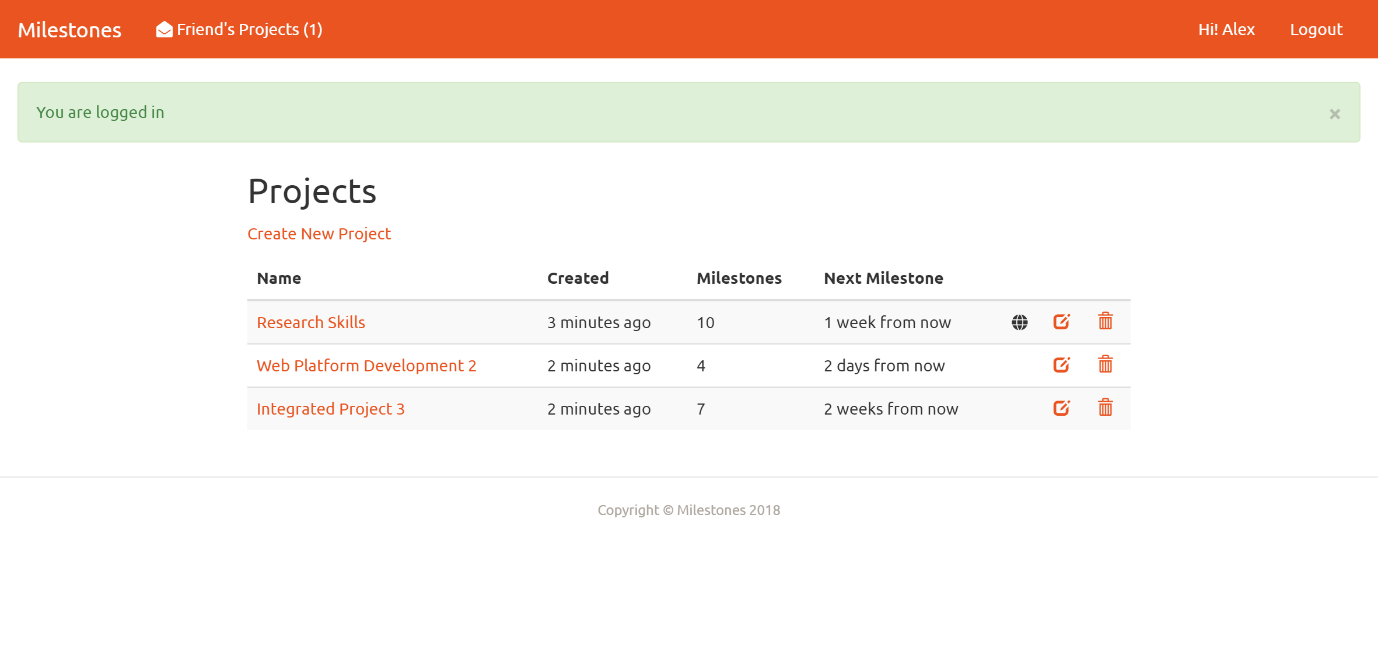


Figure - The Completed Application

# Architectural Overview

For the architecture of our application we decided to implement the site using a form of Model-View-Controller, but in our case more precisely Model-Template-Servlet. We split the application up into three main components – (1) the model containing the business logic and entity classes, (2) templates containing the HTML and views, and (3) servlets handling input from the user (see Figure 2).

Splitting an architecture up into three sections is a tried and tested pattern for creating applications. It increases separation of concerns and cohesion, while reducing coupling between components. It allows different components to be reused, for instance the model can be repurposed to support a phone app API in addition to a web site, and so long as the interface remains the same the servlets and templates would not need to be modified. The pattern also permits developers to work on different parts of the application at the same time without stepping on each other’s toes, which can also have the effect of reducing potential merge conflicts while using source control.

For handling dependencies, we used a simple Inversion of Control (IoC) container, which had the chief responsibility of making testing simpler. It enabled us to inject mock versions of dependencies such as the DatabaseService and EmailService for use when testing, so test functionality could be isolated as much as possible.

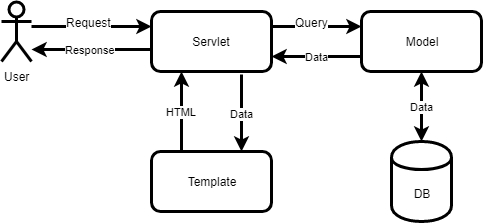


Figure - Architectural Overview

We also used view models in places. A view model wraps the model object and can provide functionality needed for the view that does not belong in the model. It can allow you to transform data into a needed format, combine data from multiple sources, as well as to restrict the parts of the model the view has access to.

# Application URLs

## URL Format

When designing the URLs, we tried to give each resource its own link, and used the following format: ‘**scheme://domain/path’.** The use of query strings, such as ‘path?foo=bar’ were kept to a minimum. The path component of the URLs typically takes the format ‘/entity/action/{id}’, with the id an integer that points to a row in the database. There are several actions shared by each entity, for example: create, delete, details, and edit. Therefore, the URL to view a project with an ID of 4 would be: ‘/projects/details/4’. This is a format we tried to maintain throughout the site.

The URL scheme is easily extendable, as any string can be included after the action. Most often this is an integer, and there is a special method in BaseServlet called getRouteId() for returning an integer value included in this manner, but this is optional and any format can be followed. Areas of the site that display personal data, for instance the edit account page, show no identifying data in the URL.

One issue with our chosen scheme is use of database IDs to identify entities, which are easily guessable. If is a project has an ID of 3456, you can guess there must be one of 3455 as well. As the sharing URL is public and can be viewed by anonymous users, it would be trivial to create a script that walked all available database IDs to scrape project data. A solution would be to generate a cryptographic identifier, a random GUID (globally unique identifier) or similar, that was associated with the row and used to fetch it. However, this would hurt the length and readability of URLs. If handled correctly exposing database IDs in URLs is not a security risk, however it can be good practice to avoid it.

If the resource requested is not found, a 404 error is generated, using a custom error page.

## Mapping of Links to Functionality

This table shows a list of resources, their description, and which servlet contains the functionality.

|  |  |  |  |
| --- | --- | --- | --- |
| Resource | Type | Description | Servlet |
| /projects | GET | Display welcome page or list of projects if logged in | ProjectIndexServlet |
| /projects/create | GET | Display create project form | ProjectCreateServlet |
| /projects/create | POST | Create a new project | ProjectCreateServlet |
| /projects/details/5 | GET | Display the details of a single project | ProjectDetailsServlet |
| /projects/edit/5 | GET | Display the edit form for a single project | ProjectUpdateServlet |
| /projects/edit/5 | POST | Edit a single project | ProjectUpdateServlet |
| /projects/delete/5 | GET | Display delete form for a project | ProjectDeleteServlet |
| /projects/delete/5 | POST | Delete a single project | ProjectDeleteServlet |
| /milestones/create/5 | GET | Display the create milestone form for the project | MilestoneCreateServlet |
| /milestones/create/5 | POST | Create a new milestone for the project | MilestoneCreateServlet |
| /milestones/edit/7 | GET | Display the edit form for a single milestone | MilestoneEditServlet |
| /milestones/edit/7 | POST | Edit a single milestone | MilestoneEditServlet |
| /milestones/delete/7 | GET | Display the delete milestone form | MilestoneDeleteServlet |
| /milestones/delete/7 | POST | Deletes a milestone from the project | MilestoneDeleteServlet |
| /api/autocomplete?term=name | GET | JSON API action for searching for user names | ApiAutocompleteServlet |
| /api/share-project | POST | JSON API action for sharing a project | ApiShareProjectServlet |
| /api/unshare-project | POST | JSON API action to stop sharing a project | ApiUnshareProjectServlet |
| /api/make-public | POST | JSON API action to make a project public | ApiMakePublicServlet |
| /users/account | GET | Displays logged in user detail | UserAccountServlet |
| /users/account | POST | Saved updated logged in user detail | UserAccountServlet |
| /users/delete | GET | Displays delete confirmation message | UserDeleteServlet |
| /users/delete | POST | Deletes the logged in user detail from users table | UserDeleteServlet |
| /users/login | GET | Displays blank log in form | UserLoginServlet |
| /users/login?token='43a82b29-54aa-46d0-a40e-dc128a626340 | GET | Displays blank login form and if token is valid, create user using the data stored in temp user. | UserLoginServlet |
| /users/login | POST | Check if username and password match, if true navigates user to project page. If not returns current form back | UserLoginServlet |
| /users/logout | GET | Displays logout confirmation message | UserLogoutServlet |
| /users/logout | POST | Remove user detail from session, navigates user back to login screen | UserLogoutServlet |
| /users/pw\_reset\_email\_sent | GET | Displays rest password sent message | UserPwResetEmailSentServlet |
| /users/pw\_reset\_email | GET | Displays email input form | UserPwResetEmailServlet |
| /users/pw\_reset\_email | POST | Send reset token link to user, and navigate user to password sent message page | UserPwResetEmailServlet |
| /users/pw\_reset?token=012fc669-1a4c-4d21-be45-409ff6edb0d0 | GET | Displays blank password input form stores token in session if token attached | UserPwResetServlet |
| /users/pw\_reset | POST | Saves the input password to token if token exist, else shows password reset fail message. | UserPwResetServlet |
| /users/register | GET | Displays blank register form | UserRegisterServlet |
| /users/register | POST | Saves data in tempUser and send verification email | UserRegisterServlet |
| /not-found | GET | Custom 404 error | NotFoundServlet |
| /server-error | GET | Custom 500 error | ServerErrorServlet |
| /unauthorised | GET | Custom 401 error - redirects to login screen | UnauthorizedServlet |
| /share/5 | GET | Loads the specified shared project | ViewSharedLinkServlet |

# Persistence

## Database Schema

To persist data, we used the Java H2 database engine, a lightweight relational database. H2 was simple to integrate into the application, has support for different platforms, good security, and fast performance (H2, No date). The database schema we developed is straightforward and comprises five tables: users, temp\_users, projects, milestones and shared\_projects (see ERD in Figure 1).

The users table stores data about the user, including username, email, joined date, and password hash. In addition, it contains columns used by the login system, resetToken for storing the password reset token emailed to the user, and loginCount for tracking how many failed login attempts the user has made.

The users table has a one-to-many relationship with projects, through the userId foreign key. The projects table contains information about a project, including name, created date, and username of the creator. The username is mirrored here as in the users table, so that it does not need to be queried for every time a project is displayed. If the username is changed, then every project the user has created needs to be updated as well.

Projects have a one-to-many relationship with milestones, through the projectId foreign key. The milestone contains information about a milestone, including name, due date, the actual completion date, and a count of the milestones that are contained in the project.

The shared\_projects table sits at the intersection between projects and users, and stores which projects have been shared with which users. The composite key is made up of the userId and projectId foreign keys. The table contains information about when the project was shared and if the user has viewed it yet.

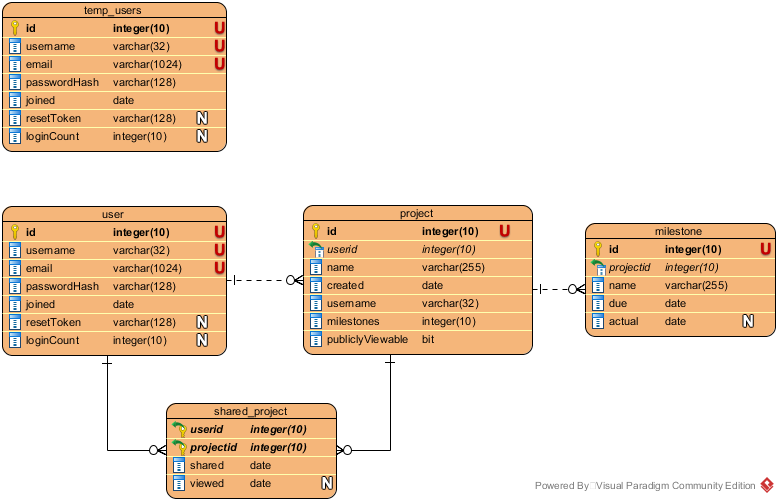


Figure - Entity Relationship Diagram

Lastly, there is temp\_users, which was added to store users which had registered but not verified their emails. This table mirrors data stored in the user table, which is fine in a small learning project such as this but would likely cause problems in a production application. The addition of a second table means that the user schema and class are now doubled, so any changes made to one must be made to the other, doubling the work needed. Of course, it’s inevitable that a developer will forgot to update one of the classes, therefore introducing a bug. A better long-term solution would need to be found that did not require the addition of this secondary table, but that was not possible in the timeframe allowed.

## Java Classes

Database tables are mapped onto Java classes in a straightforward way. We followed the Active Record design pattern, where classes are responsible for persisting themselves to storage (Fowler, No date). This was the simplest and most obvious way to implement the persistence layer. There are some issues however, namely that it does not follow the Single Responsibility Principle (SRP). SRP states that classes should only have one reason to change, yet our entity models have two (Ellis-Jones & James, 2017). Firstly, the data we need to store about the user could change, for instance if we decided to add a field, but also the way that data needs to be persisted could change, if we decided to change database engine. This is a clear flouting of the SRP. To create more robust system the database access code could be refactored into separate classes, meaning change in one would not affect change in another, however this was not possible during our tight development timeframe. For a complete class diagram of the model component of the application see Figure 3.

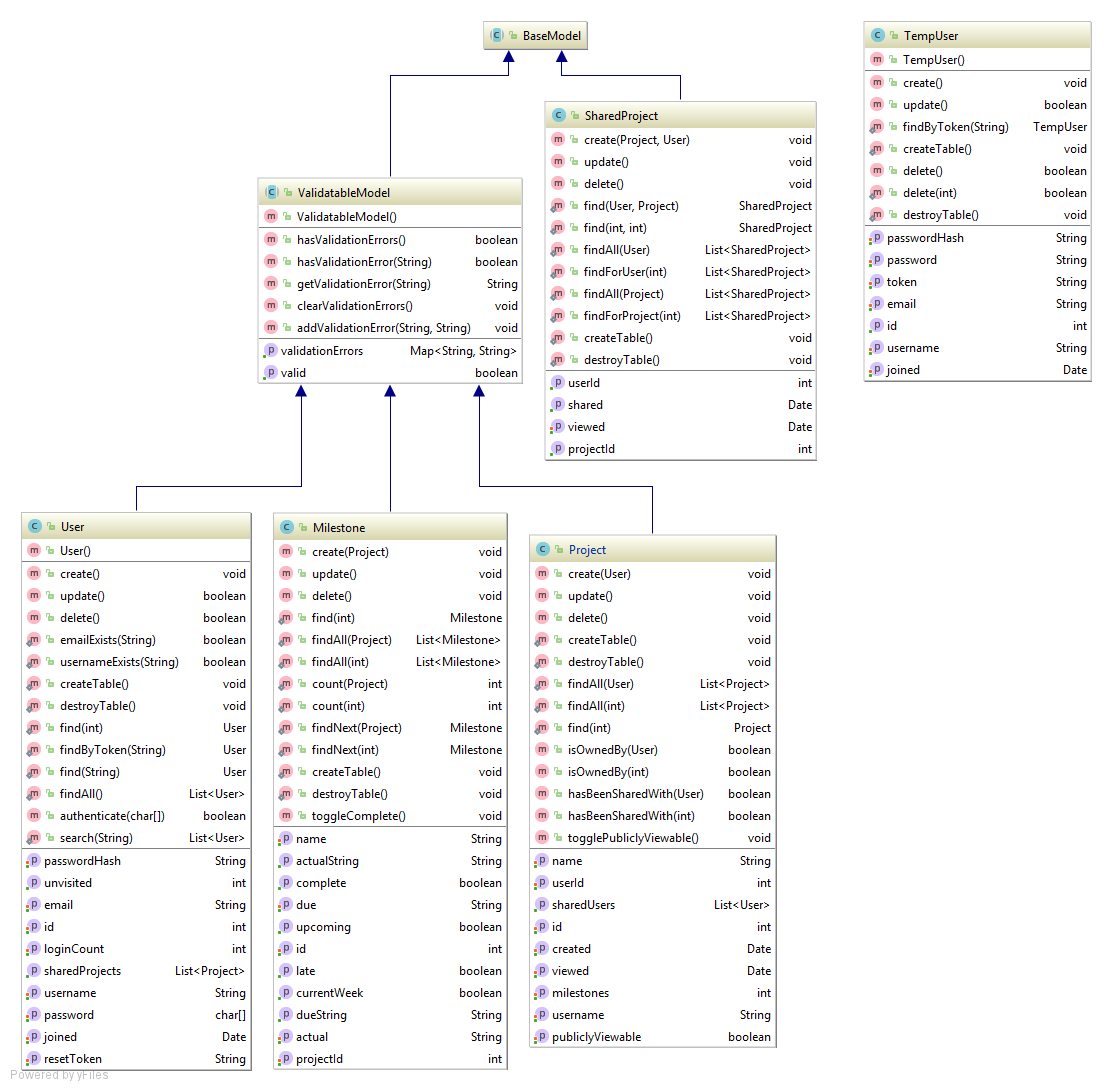


Figure - Class Diagram for Model

In adhering to the Active Record pattern, model classes were also responsible for validating themselves, and so an inheritable ValidatableModel abstract class was introduced. Each model class implements a method called validate(), that is called when the model is being validated. The child class validates its own attributes and, if needed, adds validation errors to the parent. If an error is detected, validation fails. All model classes inherit from BaseModel, a class that contains code for retrieving a database connection. We did not use a script to create the database, instead each model is responsible for creating and destroying its own table.

To simplify development, we created seed data that would be inserted into the database each time the server started up. This data contains some users, projects, milestones, and some shared projects. This is controlled in the class util/StartupContextListener by the RESET\_DATABASE\_ON\_STARTUP attribute. Setting this to false will switch off the behaviour and stop the database for being reset every time you run the app. The startup context listener is an event handler that is attached to the server startup event in web.xml, and handles initializing and registering services for the application. Seed data is defined inside of the H2DatabaseService class. The following users are available to use for testing purposes:

|  |  |  |
| --- | --- | --- |
| Name | Email | Password |
| Alex | [alex@email.com](mailto:alex@email.com) | Password1 |
| Steve | [steve@email.com](mailto:steve@email.com) | Password1 |
| Phil | [phil@email.com](mailto:phil@email.com) | Password1 |
| Jeff | [jeff@email.com](mailto:jeff@email.com) | Password1 |

# Share Project Functionality

We implemented the share project functionality in two ways. Firstly, to share a project the user needs to click the share button in the project details page, to display the share project modal (see Figure 3). This dialog box allows the user to make the project public, so it can be viewed by anyone, as well as to share the project with a specific user.

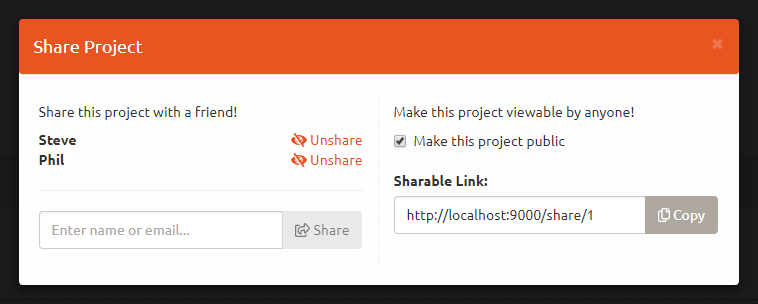


Figure - Share Project Modal Dialog

To make the project public, the user enables the checkbox, which displays the shareable link inside of a text input element. The copy button allows the link to be easily copy it to the clipboard. The checkbox can be toggled off, removing public access rights from the project. If a user lands on a public project, the page is displayed in read-only mode where no input is allowed.

To share a project with a specific member the user must enter their username or email address in the text input control. This is an autocomplete textbox that searches the database for users with matching details. Once selected the user can share the project with them by clicking the Share button. When that user next loads a page they are shown a notification that a project has been shared with them (see Figure 4). The shared project is again viewable in read-only mode.

Figure - The Friend’s Projects button, showing one new notification

# Security Overview

The application is an informal one to help students track university projects, as such security was not the primary concern, but it was still held in mind during development and good security practices were followed. The practices are outlined in this section of the document.

The site uses the Java HttpSession class to store session data and stores the session ID in a cookie in the browser. If cookies are not available, the site reverts to URL rewriting. This is done by wrapping every URL inside of an encodeUrl() call. To retrieve the user’s session either the cookie or the query string is used, depending on which is available. The session is primarily used to store a User object for the currently logged in user, as well as some other data such as flash messages (temp messages that pop down from the top of the page), and the synchronisation token (more on that later). The session only lasts while the browser is active, and if the user closes the browser the session is lost.

One of the first security issues we considered was SQL injection. This kind of attack permits hackers to sneak SQL statements into user input and have them executed against the database. To counteract this, we employ SQL parameterisation using Java’s PreparedStatement class. The class enables the site to compile a SQL statement into an object that can be formatted with input parameters without the risk of injection, which largely mitigates the issue. The compiled statements can also be executed multiple times to improve performance.

The site also takes steps to prevent cross-site-scripting (XSS) attacks. XSS is where the attacker places <script> tags, or other HTML elements inside of user input. When the input is displayed on the page the hidden code is executed. This is typically used to insert JavaScript files which can then hijack the user’s session and gain unauthorised access to data. To counteract this, we escape all user input before outputting it to the page, using the Apache StringEscapeUtils package. A downside is that this method requires every input string to be correctly escaped, which could obviously lead to holes if one is missed. Some web frameworks automatically escape all output by default, which has a performance impact, but means that it’s not possible for missed input to sneak through. A technique such as this should be considered for future projects. It only takes a single piece of input to be unescaped for the whole site to be vulnerable to attack, which is a scary thought.

Another security measure we protect against is Cross-Site Request Forgery (CSRF) attacks. This allows attackers to perform unwanted actions on a web page by tricking users into submitting forms, typically using social engineering methods (Owasp, 2018). This allows the attacker to perform state changing operations such as inserting or editing data. To protect against this, we chose to implement the “synchroniser token” method (Wichers, et al., 2018). A cryptographically secure token is generated and stored in the user’s session and placed inside of a hidden input inside the HTML form. When the form is submitted the token is checked against the session. If a mismatch is detected an error is thrown. With this method the server can be sure the request to modify data has come from a form it generated. The token is generated using a simple Java UUID (globally unique identifier) and is generated once per session.

For storing the user’s password, we used salting and hashing. It is a common error to store passwords in the database as clear text. If the database is hacked or stolen, the hacker will have direct access to all member’s passwords. This will let the attacker gain access to the milestones site, but as most people use passwords across sites a successful hacker may be able to impersonate the user in multiple places across the web. The most basic technique is to hash the password, using a cryptographic function to convert it into a random looking string. The hash is stored in the database and even if stolen the hacker will not be able to see the original password. To authenticate the user their inputted password is hashed again and compared against the one in the database to make it is correct. This technique is not perfect however, if a hacker gains access to the database they can compare the hashed passwords to tables of precomputed values, called a lookup attack. As values are precomputed this is quick as it only involves string comparison. To prevent this, a cryptographically secure random string, known as a salt, is combined with the password before it is hashed (Defuse Security, 2017). The salt is stored alongside the password hash, either in its own database column, or concatenated to the start of the hash. This means the hacker can no long do a simple lookup and needs to generate the password hashes themselves, by including the salt, which vastly increases the amount of computational power needed to crack the security. In addition, purposely slow hashing algorithms are often chosen for hashing passwords, making hackers lives even harder. This is one of the few times that a programmer will choose a slow algorithm over a fast one!

We also employ an authorisation system within the serlvets, to check sure that the user is viewing data that they have access to. The BaseServlet parent class contains a method called authorize(), which determines if the current user has permission to view that servlet. The current user’s User object is stored in the session, no data is contained in the session then authorize sets the response to the 401 unauthorized HTTP status. A custom error handler defined in web.xml is then able to catch the error and redirect the user to the login page. There are also overridden methods for checking that the current user has the privilege to view a project or milestone. This prevents hackers from getting access to various areas of the site, such as by manually editing the ID in the URL to view or edit someone else’s project.

While we have done much to handle potential security issues in the site, there are some things that could be improved. One would be HTTPS, which is a version of HTTP secured using Secure Socket Layer (SSL). HTTPS encrypts data sent between the browser and the server, so that it cannot be sniffed. This is crucially important in the days of public Wi-Fi, where it’s sometimes possible for hackers to view transmitted data. If the data is encrypted first, then the hacker will have no way to know what they are looking at. If time allowed we could have enable SSL for the site and experimented with implementing this feature, however that was not possible in the time allowed.

# Test Reports

There are many unit tests in the project, covering mainly the model and util packages. During development of these components the Test-Driven Development (TDD) methodology was followed. TDD emphasises creating unit tests first, and then writing the code needed to make the tests pass. This means that code is designed with testing in mind, also that it has high unit test coverage. It was up to individual team members to decide if they wanted to implement unit tests for the code they wrote.

For testing we used an in-memory H2 database, which let us more easily test the database persistence code. Before each test the database is reinitialized and seeded with data, meaning that each test is separate, and side-effects are minimised. For dependencies, we followed the Inversion of Control (IoC) pattern. The IoC pattern uses a lightweight container to allow developers to separate the creation of a dependency from its use (Fowler, 2004). We evaluated using third party IoC containers, or Dependency Injection (DI) frameworks, but then realised it was simpler to implement our own. IoC allows test versions of services to be substituted for ones needed in production.

## Unit Tests

This table contains a list of all unit tests, the category of the test, and that they have all passed.

|  |  |  |
| --- | --- | --- |
| Name | Suite | Passed |
| testGetConnection | BaseModelTests | Yes |
| testFindAll | MilestoneTests | Yes |
| testCount | MilestoneTests | Yes |
| testCreateTable | MilestoneTests | Yes |
| testFind | MilestoneTests | Yes |
| testValidate | MilestoneTests | Yes |
| testUpdateNoActual | MilestoneTests | Yes |
| testFindNext | MilestoneTests | Yes |
| testCreate | MilestoneTests | Yes |
| testDelete | MilestoneTests | Yes |
| testInvalid | MilestoneTests | Yes |
| testUpdate | MilestoneTests | Yes |
| testIsComplete | MilestoneTests | Yes |
| testIsLate | MilestoneTests | Yes |
| testIsCurrentWeek | MilestoneTests | Yes |
| testIsUpcoming | MilestoneTests | Yes |
| testGetSharedUsers | ProjectTests | Yes |
| testFind | ProjectTests | Yes |
| testIsOwnedBy | ProjectTests | Yes |
| testValidate | ProjectTests | Yes |
| testLoadAll | ProjectTests | Yes |
| testCreate | ProjectTests | Yes |
| testDelete | ProjectTests | Yes |
| testInvalid | ProjectTests | Yes |
| testUpdate | ProjectTests | Yes |
| testIsReadOnly | ProjectTests | Yes |
| testFindAll | SharedProjectTests | Yes |
| testFind | SharedProjectTests | Yes |
| testCreate | SharedProjectTests | Yes |
| testDelete | SharedProjectTests | Yes |
| testRenameUser | UserTests | Yes |
| testValidateUpdateWithNoPassword | UserTests | Yes |
| testFindAll | UserTests | Yes |
| testFind | UserTests | Yes |
| testValidate | UserTests | Yes |
| testUsernameExists | UserTests | Yes |
| testInvalidUserExists | UserTests | Yes |
| testEmailExists | UserTests | Yes |
| testUpdatePassword | UserTests | Yes |
| testCreate | UserTests | Yes |
| testDelete | UserTests | Yes |
| testUpdate | UserTests | Yes |
| testAuthorize | UserTests | Yes |
| hasValidationErrorTest | ValidatableModelTests | Yes |
| getValidationErrorsTest | ValidatableModelTests | Yes |
| testSeed | DatabaseServiceTests | Yes |
| testInitialize | DatabaseServiceTests | Yes |
| testConnect | DatabaseServiceTests | Yes |
| testDestroy | DatabaseServiceTests | Yes |
| testGenerateToken | AntiForgeryHelperTests | Yes |
| testCheckTokenInvalid | AntiForgeryHelperTests | Yes |
| testGenerateTokenCalledMultipleTimes | AntiForgeryHelperTests | Yes |
| testCheckTokenValid | AntiForgeryHelperTests | Yes |
| testMessage | FlashHelperTests | Yes |
| testClearMessages | FlashHelperTests | Yes |
| testGetMessages | FlashHelperTests | Yes |
| testMessageOverload | FlashHelperTests | Yes |
| testHasMessages | FlashHelperTests | Yes |
| testGet | IoCTests | Yes |
| testRegisterInstance | IoCTests | Yes |
| testUnregisterInstance | IoCTests | Yes |
| testGetInstance | IoCTests | Yes |
| testGetEmail | UserManagerTests | Yes |
| testGetUserIdUserNull | UserManagerTests | Yes |
| testLogin | UserManagerTests | Yes |
| testGenerateEmailToken | UserManagerTests | Yes |
| testGetUser | UserManagerTests | Yes |
| testIsLoggedIn | UserManagerTests | Yes |
| testGetUsername | UserManagerTests | Yes |
| testLogout | UserManagerTests | Yes |
| testEmail | ValidationHelperTests | Yes |
| testRequired | ValidationHelperTests | Yes |
| testLength | ValidationHelperTests | Yes |
| testPassword | ValidationHelperTests | Yes |
| testPast | ValidationHelperTests | Yes |
| testCreate | TempUserTest | Yes |
| testFindByToken | TempUserTest | Yes |
| testDelete | TempUserTest | Yes |
| testDeleteWithTime | TempUserTest | Yes |
| testSendEmailUsingGMailSMTP | EmailServiceTest | Yes |

## Integration Tests

This section of the report documents the integration tests performed.

### Test Login / Register

**Test log Register**

|  |  |  |  |
| --- | --- | --- | --- |
| 1.1 User can create a new user account (**users/register**) | | | |
| Test number | Input | Expected | Result |
| 1.1.1 | Click “Register” link on nav bar or welcome page  Click register without inputting any | Shows validation required error | As expected |
|  | | | |
| 1.1.2 to 1.1.4 | Input “mietest”, [mietta25@gmail.com](mailto:mietta25@gmail.com),  Password1, Password1 | Opens up register page | As expected |
|  | | | |
| 1.1.5 | Redirected to authentication email sent page, tempUser data stored | Opens up email sent page | As expected |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 1.1.6 | User recieves email,  Clicks sent link | On click of user sent email, user navigated to login page.  User account created. | As expected |
|  | | | |

**Test log login**

|  |  |  |  |
| --- | --- | --- | --- |
| 1.2 User can login (**users/login**) | | | |
| Test number | Input | Expected | Result |
| 1.2.1 | Click “login” | Opens Login page | As expected |
| 1.2.2 | click login in without input any | Shows required field validation error. | As expected |
| 1.2.3 to 1.2.6 | Input email [mietta25@gmail.com](mailto:mietta25@gmail.com)  Input password “Password1”  Click “Login” | Redirected to projects page if successfully authenticated | As expected |
| 1.1.2 1.2.3 to 1.2.4 | | | |
|  | | | |

**Test log logout**

|  |  |  |  |
| --- | --- | --- | --- |
| 1.3 User can logout (**users/logout**) | | | |
| Test number | Input | Expected | Result |
| 1.3.1 | Click “logout” | Opens Logout confirmation page | As expected |
|  | | | |
| 1.3.2 | click logout | Navigate user to Login page | As expected |
|  | | | |

**Test log Reset password**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1.4 User can reset password (**users/pw\_reset\_email & users/pw\_reset**) | | | | | |
| Test number | Input | Expected | | | Result |
| 1.4 | Click “forgot password” from login page | User navigated to password reset email page | | | As expected |
| 1.4.1 | Click “send” without input any email | Shows validation error if not input | | | As expected |
|  | | | | | |
|  | Input [any@gamil.com](mailto:any@gamil.com)  Click “send password reset email” | Show no valid email if input email does not exist in database | | | As expected |
|  | | | | | |
| 1.4.2 | Input email [mietta25@gmail.com](mailto:mietta25@gmail.com) | Redirected to email sent page. | | | As expected |
|  | | | | | |
| 1.4.3 | User opens email account | User find password reset email | | | As expected |
| 1.4.4 | User clicks the link sent | Navigated to password reset page | | | As expected |
|  | | | | | |
| 1.4.4.1  . | Click “reset password” without input | Validation error if no input | | | As expected |
| Input “Password2” and “Password1” and save | Password do not match error shows | | | As expected |
| Tick Show password | Password is visible | | | As expected |
|  | | | | | |
| 1.4.4.2 | Input “Password2” and save | Password rest confirmation | | | As expected |
|  | | | | | |
| 1.4.4.3 | Input [mietta25@gmail.com](mailto:mietta25@gmail.com)  “Pasword2” | | User can log back with new passwor | As expected | |
|  | | | | | |

**Test log Edit user**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1.5 User can edit their own account (**users/account**) | | | | | |
| Test number | | Input | Expected | | Result |
| 1.5.1 | | Click username “Hi! testmie” | User can view user account detail | | As expected |
| 1.5.2 | | Delete username and password | validation error if no input | | As expected |
| 1.5.2 to 1.5.5 | | Input “testmie2”  [mietta25@gmail.com2](mailto:mietta25@gmail.com2)  Password3 | User navigated back to Project page where username updated to “testmie2” | | As expected |
|  | | | | | |
|  | | | | | |
|  | Logout and log back in with [mietta25@gamil.com2](mailto:mietta25@gamil.com2) and Password 3 | | Navigate user to project page | As expected | |
|  | | | | | |

**Test log Delete user**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1.6 User can logout ( **users/logout**) | | | | | | |
| Test number | | Input | | Expected | | Result |
| 1.6.1 | | Click “testmie2” | | Opens Logout confirmation page | | As expected |
|  | | | | | | |
|  | | click “delete this account” | | Navigate user to delete confirmation page | | As expected |
|  | | | | | | |
| 1.6.2 | Click “Delete” | | User get naviagate back to log in page and can see “user account delted” message | | As expected | |
|  | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Input [mietta25@gmail.com2](mailto:mietta25@gmail.com2)  “Password3” | Validation error, Email or password are incorrect | As expected. |
|  | | | |

### Test Projects

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Id | | Test Case | | | | | | | Tests details | | | | | | | Expected Results | Actual Result |
| 2.1 | | Access the project index page without logging in | | | | | | | Paste the following link into the url bar:  **http://localhost:9000/projects** | | | | | | | The user is directed to the Login page | As Expected |
|  | | | | | | | | | | | | | | | | | |
| Id | Test Case | | | | | | | | | Tests details | | | | | | Expected Results |  |
| 2.2 | Access the Project index page that list all project for the user | | | | | | | | | Log in using following details:  Email: [yakoob.hayat@gmail.com](mailto:yakoob.hayat@gmail.com)  Password: GLIderzz123 | | | | | | Project for this account are displayed | As Expected |
|  | | | | | | | | | | | | | | | | | |
| Id | | Test Case | | | | | | | | | Tests details | | | | | Expected Results |  |
| 2.3 | | Create a new project | | | | | | | | | Click on the **Create new project**  Enter Title as **F1 Score calculator**  Click on **Add Project** | | | | | The User is directed to the milestone view for that page | As Expected |
|  | | | | | | | | | | | | | | | | | |
| Id | | | Test Case | | | | | | | | | Tests details | | | | Expected Results |  |
| 2.4 | | | Title Length | | | | | | | | | Click on the **Create new project**  Enter Title as **Subcarinal Social polarised magnetic refuge number**  Click on **Add Project** | | | | A message is displayed asking the user to re enter the title as it it too long | As Expected |
|  | | | | | | | | | | | | | | | | | |
| Id | | | | | Test Case | | Tests details | | | | | | | Expected Results | | |  |
| 2.5 | | | | | Edit Title | | Click on the Edit icon on the **F1 Score Calculator**  Enter the new Title as **F1 Score Sim**  Click on **Update** | | | | | | | Project is updated and the new title is shown in the project index page | | | As Expected |
|  | | | | | | | | | | | | | | | | | |
| Id | | | | Test Case | | | | Tests details | | | | | | | Expected Results | |  |
| 2.6 | | | | Delete Project | | | | Click on the Bin Icon on the **F1 Score Sim**  Click on **Delete**  on the project details page | | | | | | | The User is directed back to the Project index and the Project; **F1 score Sim** is no longer in the project list | | As Expected |
|  | | | | | | | | | | | | | | | | | |
| Id | | | | | | Test Case | | | | | | | Tests details | | | Expected Results |  |
| 2.7 | | | | | | View Project detail/Milestone view | | | | | | | Click on Project name: **Contrast Creator** | | | Milestones view is displayed | As Expected |
| **ID-07.1** | | | | | | | | | | | | | | | | | |

### Test Milestones

**Creating a Milestone**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3. Milestone Functionality | | | | | | | | | | | |
| Test number | | Input | | | | Expected | | | | | Result |
| 3.1 | | Navigate to “research skills” project and add a milestone called “valid milestone” with a due date within the next week. | | | | Milestone should display in the “current week” tab on the projects/details view. | | | | | As expected |
|  | | | | | | | | | | | |
| Test number | | | | Input | | | Expected | | Result | | |
| 3.1.1 | | | | Add Milestone title with 251 characters. | | | Should be prompted to reduce length of title. | | As expected | | |
|  | | | | | | | | | | | |
| Test number | Input | | | | | Expected | | | | Result | |
| 3.1.2 | Navigate to “research skills” project and add a “late milestone” with a due date of yesterday (27/04/2018). | | | | | Milestone should display in the “late” tab on the projects/details view. | | | | As expected | |
|  | | | | | | | | | | | |
| Test number | | | Input | | Expected | | | Result | | | |
| 3.1.3 | | | Navigate to “research skills” project and add an “upcoming milestone” with a due date 06/05/2018. | | Milestone should display in the “upcoming” tab on the projects/details view. | | | As expected | | | |
|  | | | | | | | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | Input | | | Expected | | Result |
| 3.1.4 | Navigate to “research skills” project and add a milestone without a title with a due date of 06/05/2018. | | | Milestone should not be created, and user should be prompted to input a title. | | As expected |
|  | | | | | | |
| Test number | | Input | Expected | | Result | |
| 3.1.5 | | Navigate to “research skills” project and create a milestone without a due date. | Milestone should not be created, and user should be prompted to input a due date. | | As expected | |
|  | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test number | Input | Expected | Result |
| 3.1.6 | Add a milestone with a due date exactly 6 days and 23 hours and 59 minutes after current date. | Milestone should be displayed in the current week tab. | As expected |
|  | | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test number | Input | | | | Expected | | | | Result |
| 3.1.7 | Create a milestone that will be displayed in the “upcoming” tab. The milestone will be exactly one week and one minute in the future. | | | | Milestone should be displayed in the “upcoming” tab. | | | | As expected |
|  | | | | | | | | | |
| Test number | | | Input | Expected | | | Result | | |
| 3.1.8 | | | Wait one minute after test 3.1.7 and check that this milestone moves to the current week tab. | Milestone should be displayed in the current week tab and not displayed in upcoming tab. | | | As expected | | |
|  | | | | | | | | | |
| Test number | | Input | | | | Expected | | Result | |
| 3.1.9 | | Create a milestone that will be displayed in the “current week” tab. The milestone will be set one minute in the future. | | | | Milestone should be displayed in the “current week” tab. | | As expected | |
|  | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test number | Input | Expected | Result |
| 3.1.10 | Wait one minute after test 3.1.9 and check that this milestone moves to the late tab. | Milestone should be displayed in the late tab and not displayed in the current week tab. | As expected |
|  | | | |

**Edit Milestones**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | | Input | Expected | | | Result |
| 3.2 | | Navigate to “research skills” project and edit “Milestone 1”.  Title: “Mile 1”  Due Date: “07/09/2018” | Milestone should be modified, and user should be notified of change. The milestone should now be in “upcoming”. | | | As expected |
|  | | | | | | |
| Test number | Input | | | Expected | Result | |
| 3.2.1 | Navigate to “research skills” project and edit “Milestone 1”.  Title: *none*  Due Date: “07/09/2018” | | | Milestone should not be modified, and user should be prompted to add a title | As expected | |
|  | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test number | Input | Expected | Result |
| 3.2.2 | Navigate to “research skills” project and edit “Mile 1”.  Actual: “26/04/2018” | Milestone should be marked as done and displayed in the “done” tab of the projects/details view. | As expected |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test number | Input | Expected | Result |
| 3.2.3 | Navigate to “research skills” project and edit “Milestone 1”s actual date. Make actual date’s value in the future.  Actual: “29/04/2018” | System should disable all future date values for “actual date“ so user cant mark a milestone as complete on a future date. | As expected |
|  | | | |

**Milestone deletion**

|  |  |  |  |
| --- | --- | --- | --- |
| Test number | Input | Expected | Result |
| 3.3 | Navigate to “research skills” project and delete “Mile 1”. | User should be redirected to the milestone delete view. Where they will confirm deletion. The milestone will no longer be displayed on the system. | As expected |
|  | | | |

**Milestone Complete**

|  |  |  |  |
| --- | --- | --- | --- |
| Test number | Input | Expected | Result |
| 3.4 | Navigate to “research skills” project mark “milestone 6” as complete. | The milestone should be moved to “done” tab. And completion date should be set to the time/date that the “mark as done” button was selected. | As expected |
| Before  After | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test number | Input | Expected | Result |
| 3.4.1 | Navigate to “research skills” project mark “milestone 6” as incomplete. | The milestone should be moved back to “current week” tab. And completion date should be set to null. | As expected |
| Before  After | | | |

### Test Share Project

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4. Sharing | | | | | | |
| Test number | | Input | Expected | | Result | |
| 4.1 | | Click share project button in project details | Opens Share Project modal | | As expected | |
|  | | | | | | |
| 4.2 | | Click close button on modal | Modal dialog closes | | As expected | |
|  | | | | | | |
| 4.3 | Check make project public checkbox | | | Copy link input and confirm message displayed | | As expected |
|  | | | | | | |
| 4.4 | Click Copy button next to input. | | | URL copied to clipboard, confirmation displayed | | As expected |
|  | | | | | | |
| 4.5 | Enter name ‘Alex’ in friend share input | | | Autocomplete for ‘Alex’ is shown | | As expected |
|  | | | | | | |
| 4.6 | Enter name ‘alex@email.com’ in friend share input | | | Autocomplete for ‘Alex’ is shown | | As expected |
|  | | | | | | |
| 4.7 | Enter ‘Alex and click Share | | | Message you cannot share with yourself displayed | | As expected |
|  | | | | | | |
| 4.8 | Enter ‘Steve’ and click Share | | | Steve added to list | | As expected |
|  | | | | | | |
| 4.8 | Enter ‘Steve’ again and click Share | | | Message saying already shared with Steve is displayed | | As expected |
|  | | | | | | |
| 4.9 | Click ‘Unshare’ next to Steve | | | Steve removed from list | | As expected |
|  | | | | | | |
| 4.10 | Share with Steve and login with his details: [steve@email.com](mailto:steve@email.com) Password1. | | | Notification displayed in navbar | | As expected |
|  | | | | | | |
| 4.11 | Click Friend’s Projects button | | | Modal is displayed | | As expected |
|  | | | | | | |
| 4.12 | Click close on modal dialog | | | Modal is closed | | As expected |
|  | | | | | | |
| 4.13 | Click view shared project in Friend’s Projects modal | | | Project is displayed in read-only mode | | As expected |
|  | | | | | | |

## Summary of Test Results

The tests were completed successfully and confirm that all the functionality needed by the brief has been implemented. All the tests completed as expected.

# References

Defuse Security, 2017. *Salted Password Hashing - Doing it Right.* [Online]   
Available at: https://crackstation.net/hashing-security.htm  
[Accessed 24 04 2018].

Ellis-Jones & James, 2017. *Think you understand the Single Responsibility Principle?.* [Online]   
Available at: https://hackernoon.com/you-dont-understand-the-single-responsibility-principle-abfdd005b137  
[Accessed 24 04 2018].

Fowler, M., 2004. *Inversion of Control Containers and the Dependency Injection pattern.* [Online]   
Available at: https://martinfowler.com/articles/injection.html  
[Accessed 24 04 2018].

Fowler, M., No date. *Active Record.* [Online]   
Available at: https://www.martinfowler.com/eaaCatalog/activeRecord.html  
[Accessed 24 04 2018].

H2, No date. *History and Roadmap.* [Online]   
Available at: http://www.h2database.com/html/history.html  
[Accessed 24 04 2018].

Owasp, 2018. *Cross-Site Request Forgery (CSRF).* [Online]   
Available at: https://www.owasp.org/index.php/Cross-Site\_Request\_Forgery\_(CSRF)  
[Accessed 23 04 2018].

Wichers, D., Petefish, P., Sheridan, E. & Righetto, D., 2018. *Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet.* [Online]   
Available at: https://www.owasp.org/index.php/Cross-Site\_Request\_Forgery\_(CSRF)\_Prevention\_Cheat\_Sheet  
[Accessed 24 04 2018].

# Table of Figures

[Figure 1 - The Completed Application 3](#_Toc512746548)

[Figure 2 - Architectural Overview 4](#_Toc512746549)

[Figure 3 - Entity Relationship Diagram 7](#_Toc512746550)

[Figure 4 - Class Diagram for Model 8](#_Toc512746551)

[Figure 5 - Share Project Modal Dialog 9](#_Toc512746552)

[Figure 6 - The Friend’s Projects button, showing one new notification 9](https://caledonianac-my.sharepoint.com/personal/amcbri206_caledonian_ac_uk/Documents/WPD2%20Coursework%20Report.docx#_Toc512746553)