**Web Platform Development Group Report**

|  |  |
| --- | --- |
| **Group:** | AE |
| **Members:** | Shaun Cooper |
|  | Marc Millar |
|  | Amar Khaliq |
|  | Thomas Ferns |

I declare that all work submitted for this coursework is the work of Shaun Cooper, Marc Millar, Amar Khaliq and Thomas Ferns alone unless stated otherwise.

**Introduction**

This report explains the various decisions in design to implementation. Each section of the report will explain and expand on the specific area with describing why it was implemented that way. The sections which will be expanded upon are Link Mapping, Persistence Mechanism, Functionality & Testing and, Security.

Link Mapping is the design of the links throughout the application and how each page is tethered together. This is how the user will traverse the pages. It is also how the sharing projects function is supposed to work and how the URL is generated.

Persistence mechanism is the database that is used to hold the data for the underlying base of the system. This is needed for the rest of the functionality to work. This will explain the schema and the data access layer of the application

The functionality & Testing section goes on to explain the testing carried out of the application and if the functionality works as intended

Security of the application is vital to prevent important data being leaked which this section covers. The section explains the security decisions that were made and implemented into the application.

**Link Mapping**

When designing to link mapping, we didn’t thoroughly plan it out, however we did base our links to be as human readable and logical as possible, link mapping plays an important role especially within a web application; there are many default paths that users will try because they’re used universally throughout the internet. For example, the Login, Logout and Register page didn’t need much consideration, these paths are usually by default what a user will try to access when attempting to use any website. The other servlets however did required some planning, we decided the users list of projects would be fine under the path of Projects, mainly because it’s short sweet and self explanatory, a users projects are on their “Projects” path.

For the unique URL generation, it was very important that we provided the users a means of accessing their individually created projects to get and modify further information residing within them, the design for this was a randomly generated base64 encoded string, with a length of 10 characters. The reasoning for the base64 encoded string is detailed further within the security section of this document. Each unique project URL is simply placed after the path “Project”, we felt this we be sufficient as the users list of projects is on the plural version of the word, and an individual project should have the singular vocabulary. If a user enters an incorrect or unavailable project URL then they will receive an error message detailing that they’re entered an invalid URL, it was decided against using parameters within the URL because it didn’t abstract the parameter names from the users and also took up unnecessary spacing, an alternate implementation was decided upon where the 10 character URL would be put after in a sub-path of “Project”, therefore accessing an individual project would require going to a URL such as “Project/dEkA\_-3ka3”.

**Persistence**

During the initial design phase of the application it was vital that an effective persistence mechanism was implemented as it would be the basis of the application meaning without it the other functionality intended would not be able to function. The group decided on H2 being the best database to use as it is extremely flexible with its implementation as it can be implemented as a server, embedded or in memory. The H2 database was implemented by embedding it in the application itself. This is so it can easily be transferred between machines when working on the application and for the application when completed. The database when created loads an SQL file after making a connection to create the different tables of the database. The connection then closes and only opens if a request is made.

For the database schema it was important that the various tables required by the milestone system to function properly were interconnected in some way through the linking of primary keys and secondary keys. This is because each member of the system has to have their own projects and milestones specifically linked to them. This is done by the primary key in the user table which is automatically incremented for each user that registers with the system as it is then used in the project and milestone creation to ensure that only the user that created the projects and milestones can actually see and edit them. The design is a one to many design with a User’s unique ID being linked to the many project and milestones they may create.

The data access layer of the application is implemented through the various servlets which allows us to separate what the user sees and what the application is doing. This is important as it allows for various changes to be made to the Model layer or the View Layer of the Model View Controller architecture (MVC) without having to change the other two layers in any way. It is especially beneficial for team works as multiple developers can work in parallel on different components of the application. In the MVC architecture the servlets pass relevant information received from the user whether it is adding a milestone or project, or even registering themselves on the system and passes that information to the models. Once the models have given the results the servlets then pass that information on to the View layer which displays it on the screen so the user can see what their action has accomplished.

**Functionality & Testing**

Testing is an important aspect of developing any kind of software, it is important to ensure that all of the functionality implemented works as expected; whilst also noting the features that are not functioning to the highest standard. Testing ensures that the system created is robust, and handles unexpected user input in an effective manner, allowing the system to “fail gracefully”.

To ensure every feature was tested for, test cases were created to show each function within the site, these test cases were then used to create more in-depth tests on these features. The test cases table can be found in Appendix 1.

The test data can be found in Appendix 2, this shows all of the specific tests used to carry out the tests from the cases previously identified. To ensure the system is robust, all tests must have the following types of data tested:

* Normal - the expected input from a user
* Extreme - the extremes of the expected input/action
* Exceptional - incorrect data type/input from the user

These different levels of testing were carried out on each of the test cases, ensuring the system is able to handle different types of input.

These tests were carried out over the course of development and then again at the end when the final deliverable was complete. Overall these tests were effective in assessing the functionality of the site as all pages are tested ensuring they have validation and that the appropriate functions are called upon.

**Security**

Security was a big consideration when designing how projects should be accessed via URL, therefore measures were taken to ensure that various vulnerabilities were prevented by smart implementations. The decision to go with a randomly generated URL instead of a simple incrementing one stems from the disadvantages of incremental path generation on websites, if your uniquely generated paths are simply incrementing on the former then it’s very easy for anyone to gain access to a variety of different directories they otherwise wouldn’t be able to; for example if you seen a project was under the path of 55 you could try 56 and would most likely see someone else's project. Another big disadvantage of using incrementing paths is that attackers can use brute forcing to easily obtain all information residing within your website, incremental paths provide attacks with an easy method of iterating over pages, since they simply need to increment by 1 then check the new URL. Considering our user’s projects were to be unlisted, we decided to stay far away from simple incremental link mapping.

The eventual method of URL generation came down to random generation, by randomly generating a link it makes it extremely hard for attackers to iterate over the website and obtain all information residing within, also people won’t simple be able to add a number onto another URL and receive a result. The randomly generated string was a 10 character long base64 encoded string, this is very similar to youtube's URL design, however method of generation is probably different and theirs is 11 characters long. By having the string encoded in base64 it gave us a possible 64^10 unique URLs that could be generated, this is very future proof because it’s very unlikely we will create a quintillion projects within our lifetime, and even if we do, we could simply extend the generated URL by 1 character and obtain another seventy-two quintillion possibilities. Another benefit of this method is that if a user wants to iterate over every possible project, they would have to go through more than a quintillion different web pages, which is a lot more resource heavy and inefficient than looping through incrementally. Since there are so many possible URLs, we decided to leave them unlisted, it’s very unlikely that anyone will by chance input a used string and even if they sufficient implementation are implemented to prevent anyone but the creator from modifying their projects milestones and information.

Validation was also added thoroughly throughout the website as to prevent possible attacks when users are inputting details into the application, for the addition and modification of the projects and milestones, server side validation was utilised to ensure that users were only able to manipulate projects or milestones which they owned.

For inputting details related to projects and milestones, client-side validation was sufficient because if the user were to get an error it would be their own fault for messing with the client-side validation, however sufficient security measures were implemented in the server side as to ensure that security vulnerabilities were prevented.

When designing the data access layer, the use of a PreparedStatement was utilised with the insertion of parameters into a SQL statement, this was important because it provided a layer of security by sanitizing each parameter before insertion into the statement. Parameters were inserted into the statement via the methods PreparedStatement provided, beforehand a string would be instantiated with any parameters being replaced by question marks.

**Conclusion**

In conclusions, the project was successful in producing a Project planning application that fulfilled the requirements set. From the link mapping to the security decisions, each phase and section of the project while challenging was tackled and successfully implemented through effective teamwork and communication.

**Appendix**

Appendix 1

|  |  |
| --- | --- |
| Test No. | Purpose of Test |
| 1 | Ensure that the index page loads |
| 2 | Ensure that the user is only able to create valid accounts (an email address can only belong to one account, etc.). Check that users can log in |
| 3 | Ensure that logged in users are able to view, add, delete and share projects |
| 4 | Ensure that logged in users are able to add, remove and edit milestones from their projects |

Appendix 2

TEST DATA ON SEPARATE DOCUMENT