SCHOOL OF COMPUTING, ENGINEERING AND BUILT ENVIRONMENT

Department of Computing

Web Platform Development (M3I324182)

Group Report

Name	Email	Id
Yonas Temesgen	ytilah200@caledonian.ac.uk	S1719046
Faouzi Jedidi	fjedid200@caledonian.ac.uk	S1719017
Heritier Muhire	hmuhir200@caledonian.ac.uk	S1719021

Table of contents

1.0 Introduction

This report aims to discuss the different aspects of designing and implementing the Milestone planner web application for the Web Platform Development 2 coursework. In this report, we will discuss and explain the link design within the application which will include the mapping of links to different functionalities and the logic behind the chosen link schema. Moreover, we will describe the persistence mechanism used in designing the database. We will explain the different functionalities an report on the testing that took place to ensure that all the functionalities are working properly. The final part of the report will be a discussion of the application's security and the different approaches used to ensure the best level of security possible.

2.0 Links

Our milestone planner web application was implemented using a structure that maintains a logical connection between the links. This linking structure makes user browsing way more natural since each link provides the functionality that the application provides for the request. We used the MVC architectural design pattern to develop the application since it helps for separation of concerns. Thus, when our application runs, it undergoes through the same process of three layers: Model, view, and controller.

Below is a detailed explanation of our link design within the application:

- Localhost: 8080: when the client-server connection runs successfully, the application runs on a localhost URL, sends a get request method, and then the server provides the response which contains the get index_milestones.jsp content, HTTP success code, which displays the content on the body.
- /UserServlet: on the signup page, it is recommended for a new user to signup. A user fills his personal information and then clicks on the signup button. When this button is clicked, it sends a request to the /UserServlet, which then sends the user credentials in H2User DAO, which then stores the user data in the database. After that, it sends a response to the home.jsp page.
- /LoginServlet: still on the signup page, when a user is already registered, is required to log in. When the login button is clicked, it will send a request to the LoginServlet, which then opens doPost(). Inside this method, it takes the request from the login.jsp and then goes to the h2person where it checks if the user inputs are already persisted and stored in the database. If the credentials are correct, the controller notifies the view, and then redirect the response to /home.jsp after making a successful login.

- /add_milestone: When the user is on the homepage, he can view all his milestones. If a user clicks on add milestone button, it gives the user a form and fills the information, and clicks add button. It sends a doPost HTTP request to the server, which delegates to the addMilestone servlet, and which then calls addMilestone(), and then add a milestone to the H2Milestone, and store it in the database. After that, it will send the response to the browser, and display a new milestone on the homepage.

- /Logout: A user can terminate a session from his dashboard. When a user clicks on a logout button, it terminates the session, and redirect the user to logout.jsp page.
- /ListServlet: By default, all milestones are already listed on each user homepage/dashboard. If a user chose to edit a milestone, he clicks on the edit button. It then sends a request in the /EditServlet, which loads the editMilestone(), and allows the user to edit a milestone. And once it is done, it adds the edited milestone in H2Milestone, stores it in the database. When it finishes saving it, the server sends a response to /ListServlet with an edited milestone added to the list.
- /sharedLink : on the user dashboard, when a user clicks on the share button, it sends a request to the sharedServlet, which then load the method doPost() that takes the request from share.jsp

as well as triggering the model which then load the milestone shared, and redirect it to a new page called share.jsp.

After stating the above URL used in our application, the following section explains some of the best practices to URL design structure that were applied in our application:

- Stable link: link never change when an application is launched, or a user is browsing. This enables quality features of the web platform such as accessibility and usability.
- Use of human-readable URL: we make sure our URLs are short, precise and straight to the point and easy to remember.
- Button: we set all our links to target =”_blank”: all our links open in the same window. The users use the browser’s back button to go back on the other page they were on.
- Used descriptive links: we make sure our links are short less than 15 characters but too descriptive to let the user understand enough our URL functionality.
- Formatting: we designed our links in a way that a link can’t be in all uppercase or lowercase characters. When a link is made up of two words, we used the camelcase. No underscore used when naming our links.

3.0 Persistence mechanism

When websites need to persist data, they typically store it in a database. While the persistence mechanism and the type of database may differ, database management systems (DBMS) offer similar services. DBMSs perform four functions. They are create, read, update, and delete, also known as CRUD.

There are many database management systems. For this project, we have used H2. It's a relational DBMS that is usually used with Java projects. It mainly has two modes. The first one allows data to be stored in a file. After the program running the web application is closed, the persisted data will not disappear. To achieve this, the class that acts as a data access object defines the location of the file. Then H2 uses the file to perform CRUD. It also encrypts the files so that no one can access it directly by opening the file. If the user manually deletes the file, it will create another one when the web application runs again. Compared to other DBMSs, H2 is easily portable.

On the other hand, the second mode stores the database in memory. The data will not be stored permanently. When the program closes, the data gets deleted. This provides flexible options for the developer to test out their databases.

H2 offers a web-based console. The user can access the interface through a browser. One drawback of this method is that the user cannot run the web application and access the web console simultaneously as there's only one instance of the database. Either the web console or the web application should not be running to utilize the database successfully.

The database that the web application interacts with has three tables namely user table, project, and milestone. Each table contains attributes which most of the time represent the attribute of a model class. The tables also have a primary key that uniquely identifies each row in a table. The relationship that exists between the tables is defined by the foreign keys. A foreign key is an attribute which is a primary key in another table.

Data Access Object

The data access objects sit between the database and the model. They provide functionalities without revealing the inner workings of how data is accessed from the database.
The web application has three data access objects or what is equivalent of that. To gain a deeper understanding, we will dive into one of them, H2 Milestone.

H2 Milestone has multiple methods that allow other classes to interact with a table called Milestone. As it interacts with the database, it first has to create a connection with it. It specifies the database and credential details. It also defines how it wants to store the database; in memory or file. Assuming it has connected successfully, some of its methods are discussed here.

Add milestone

This method first creates a prepared statement. A prepared statement helps improve the security of the system and allows an SQL query to repeatedly executed efficiently. It accepts a milestone object and uses it to get the information it needs. After doing that, it performs the necessary procedures like making sure the data that goes to the database confronts the data type of the database's attribute. It then executes the code responsible for inserting the data into the rows of a table.

N.B. Since most of the methods share similar steps between them, the focus will be on what is different.

Edit milestone

It accepts a milestone object and changes the row/s in the table based on the "where" condition of the SQL statement.

Remove milestone

This method also accepts a milestone object to performs its actions. It deletes a milestone from the database. Once the milestone is deleted, the user will not be able to access it.

4.0 Functionality and test report

The milestone planner applications consists of various functionalities. To get to the final version of the functionalities reported in this section, we first had to define the requirements and objectives of the application. We first defined the functional requirements for this application were as follow:

- The system shall allow the user to sign up.
- The system shall allow the user to login and logout.
- The system shall allow the user to add a milestone.
- The system shall allow the user to edit the existing milestones
- The system shall list all the incomplete milestone in the dashboard
- The system shall allow the user to delete milestones.
- The system shall allow the user to share the milestones with other users.
- The system shall store all the details in a database to be retrieved on demand.

To implement the different functionalities of the application, our team worked on each of the functionalities step by step until the final product. First we worked on the sign up function. On the start of the application the user will be directed to the homepage that includes the sign up form. As the user finishes filling the necessary details and clicks sign up, the details will be redirected by the sign up servlet to the database manager which stores the details in the database. Once signed up, the user can now login using his credentials by using the sign in which toggles a pop up window that the user will use to login. The user puts his credentials, the system will redirect them through the login servlet which will send the details to the database manager. The DBMS then check if the user credentials exist, if a match is found, the system initiate a session and the user will be redirected to the dashboard otherwise, he will be redirected to the sign up page to create an account.

Once the user is logged in, he will be allowed to add and manipulate the milestones. The user can add a milestone by clicking the “Add a Milestone“ which toggles a pop up form that allows the user to fill the necessary details for a milestone and click add. Then system redirects the details through the AddMilestoneServlet to the database manager which add the details to the database and assign that milestone to the user using the email. The servlet then redirects the user to the dashboard by calling the ListServlet that will display the details of the milestones in the user’s dashboard. The user then can edit the milestone by clicking the Edit button under the actions section. The system will show the user a form that he can fill. The details as well as the milestone id is sent to the EditServlet which, through the DBMS, updates the details of that milestone in the database. Once more, using the id, the user can delete a milestone by just clicking on the Delete button.

The user can also choose to share the list of milestones with another user. This process is simply possible by clicking the Share button. Once the user clicks share a shareable identifier will appear that he can use to share his milestone list with another user. Once the other user receives the identifier, he can insert it in the Insert Link on the navigator side of the dashboard and click Import. That will take the user to the shared dashboard where he can only view the list of the milestone without the ability to edit or delete or add.

To test the different functionalities of the application. We defined a set of objectives that the user should accomplish by using the application. The test descriptions is outlined in the table below:

Test Description	**Expected Results**	**Actual Results**
Sign Up	The user shall be able to sign up using the sign up form. The results should reflect on the database.	Works as Expected.
Login	The user can log in using the pre-signed-up credantials and be redirected to the dashboard. Or get redirected to the sign up page if the credentials don't exist.	Works as expected.
List Milestones	The user should find his own milestone once logged in.	Works as Expected
Add a Milestone	The user can add a milestone by clicking on the "Add a Milestone" button and filling the form. The results should reflect on the database and be listed in the dashboard	Works as Expected
Edit a Milestone	The user can edit an existing Milestone by choosing the milestone and click on the Edit button. The results should reflect on the database as well as on the list of milestone.	Works as expected. However when the user clicks on edit an empty form will appear instead of filling the form with that specific milestone.
Delete a Milestone	The user can delete a milestone by clicking Delete. The results should reflect on the database and the list on the dashboard.	Works as Expected
Share Milestones	Share milestones when share/Import is used. User can logout and another user can use the identifier to import the shared milestones.	Works as expected.
Logout	The session should be invalidated and the user redirected to the homepage.	Works as expected. However we can't stop the user from going back but he will only see an empty dashboard.

5.0 Application security

Web applications are complex to build and maintain; hence, they are always exposed to security risks and vulnerabilities. For example, a vulnerability might be a design flaw, URL parameter, a bug, or any other web app weaknesses that might guarantee an attacker the access to the site in an unintended way and make malicious attacks that can cause harm. Therefore, it is essential for any web app developer/designer to apply possible security measures to safeguard a web app; thus, avoid threats that can cause harm to stakeholders of the application at any time.

Below is a list of the actual security of the application that can be implemented in each webapp as well as the awareness of the possible security threats:

1. Enforce Use of complex password

It is important to enforce your users to use strong passwords. Attackers have access to a large quantity of users information and sophisticated tools that they use to crack passwords. To protect the webapp from such brute force attack, it is highly recommended to insist on good password practices such as a minimum number of 8 characters, made up of mixed format of characters such as uppercase letters, lowercase, special characters and numeric values.

2. Password hashing

It is a big mistake for a webapp that requires login information not to encrypt or not protect the user sensitive data such as passwords. This leaves it exposed to attackers to steal the info. User passwords should always travel or be stored as encrypted values and hashed. A preferable way of hashing would be using a cryptographic hash function called SHA-1. This method allows you to authenticate users by comparing the encrypted values, not plain. For a secure the user passwords more robust, it is crucial to use salt per each password, which then makes it expensive for hackers to compute. Therefore, if there is a forced attack to hack a user password, it becomes so hard that decrypting might not be possible to reveal the same password since every guess has to be hashed separately for every salted password.

3. Securing cookies

Cookies are important because they provide a way for the website to recognize and keep track of the user's browsing activity. So the cookies are often an interesting target for hackers. Therefore, securing sensitive cookies should be a taken into consideration when building a website. According to Infosec, "HttpOnly and secure flags can be used to make the cookies more secure. When a secure flag is used, then the cookie will only be sent over HTTPS, which is HTTP over SSL/TLS" (Dawid, 2018).

4. SQL injection

SQL injection refers to attacks that are used to steal information from the database. This attack occurs when an attacker uses a web form field or URL parameter to gain access to or manipulate your database. An attacker then injects a query to perform an operation such as deleting user

data, making changes to the database or retrieve information stored in the database. This attack can be prevented using a defensive mechanism such as prepared statement, and input data sanitation.

5. Protect against XSS attacks

cross-site scripting (XSS) attacks inject untrusted JavaScript into your pages, which then runs in the user's browser, and can change page content, or steal information to send back to the attacker. This XSS attack can also modify the webpage of an application to redirect its authorized users to scam sites since the users don't exactly know if it is malicious. The developer has to be cautious, and capable restrict users to inject malicious javascript content on the webpage.

6. Cross-site Request Forgery

This refers to attacks that are used to steal data from the website's front end. Such attacks are used to redirect users to a malicious website where attackers can steal data from targeted users. This is usually accomplished using "malicious scripts that are executed in client browsers as a result of user input, or client requests, or other expressions" (Danyetta, 2018). For example, this attack can happen through email phishing attempts, email attachments with embedded links or frequently visited sites by targeted users.

5. Use HTTPS

HTTPS is a protocol used to provide security over the Internet. It is highly advisable to use this secured and private for transmitting and receiving information across the internet. HTTPs is reliable for managing sensitive communication and providing data integrity for both the website and the user's information as well as their privacy. It should be enabled in every website over HTTP because it encrypts the connection to a site which then prevents the intruders from intercepting any of your data.

6. Two-factor user authentication

This refers to a security option that webapp uses which requires a user not only signing or accessing some information using his primary credentials (username & password), but also include something else that is connected to that user such as text message, email, phone call.

7. Use a secure and reliable host

When hosting your website, it is recommended to use a reputable and secured. Some of the hosting sites do not handle web security threats; which can put your site more vulnerable to attacks. Therefore, it is critical to select a host that is aware of risks and is devoted to keeping your website secure. Also, for mitigating circumstances, your host should be able to backup your data to a remote server so that your data can be restored anytime there is an attack that might cause harm. Finally, choose a host who uses updated technology and is ready to support you technically at anytime when necessary.

6.0 References

Codecademy. (2019). What is CRUD? Codecademy. [online] Available at: https://www.codecademy.com/articles/what-is-crud

AISHWARYA, Y., 2018. Learn The Difference Between Injection And Cross-Site Scripting Attacks!. Available from: https://www.securitycommunity.tcs.com/infosecsoapbox/articles/2018/02/13/learn-difference-between-injection-and-cross-site-scripting-attacks

DAWID,C., 2018. Securing Cookies with HttpOnly and secure Flags. -03-06T16:13:18+00:00, [viewed Apr 25,2019].Available from: https://resources.infosecinstitute.com/securing-cookies-httponly-secure-flags/

GERBER,R., COMPTON, T., 03, T.P.&DESIGN, 2.W., 2018. 9 security tips to protect your website from hackers.May 03, [viewed Apr 26, 2019]. Available from: https://www.creativebloq.com/web-design/website-security-tips-protect-your-site-7122853.

H2database.com. (2019). H2 Database Engine (redirect). [online] Available at: https://www.h2database.com/

JAIN, S. ,What every web app developer must know about security, 2018. -02-14T22:00:58.185Z [viewed Apr 26, 2019]. Available from: https://medium.com/walmartlabs/what-every-web-app-developer-must-know-about-security-3b4345353133.

W3schools.com. (2019). PHP Prepared Statements. [online] Available at: https://www.w3schools.com/php/php_mysql_prepared_statements.asp

8 Simple Ways to Improve your Website Security. [viewed Apr 25, 2019]. Available from: https://www.commonplaces.com/blog/8-simple-ways-to-improve-your-website-security/.