Web Platform Development 2

M3I324182-18-B: 18/19 B

Web-based Milestone Planner Application

Group 4 Report

Student Name	Matric Number	Email
Fatuma Ingabire	S1719023	fingab200@caledonian.ac.uk
Debbie Adejumo	S1719011	dadeju200@caledonian.ac.uk
Elizabeth Michael Akpan	S1719014	eakpan200@caledonian.ac.uk

06th May 2019

We, Fatuma Ingabire, Debbie Adejumo, and Elizabeth Michael Akpan, declare that all the work submitted for this coursework is our work alone unless stated otherwise.

Introduction

This report outlines the design choices and justification of methods used in the implementation of a web-based milestone planner application. It details:

1. The various link schemas and functionalities that are supported by the application
2. The data persistence mechanism of the application
3. The functional requirements of the application and test report showing the current performance of the application
4. A security appraisal of the web application

The milestone planner web application is an online productivity tool that allows users to keep track of key project milestones. Users of this web app will be able to login and logout at their convenience, and interactively use their milestone dashboard to add, delete, edit, and track incomplete milestones for specified projects.

Logic of Link Schema

Table 1: *Application functionality and link schema mapping*

Link	Application Functionality
webApp.login.do	This link facilitates the login functionality. The doGet() method returns the login.jsp page which contains the client facing code. The doPost() method sends the login credentials provided by the user, i.e. username and password, and the response redirects the user to the webApp.listMilestone.do link, i.e. the milestone dashboard
webApp.listMilestone.do	This link enables milestones to be viewed on the dashboard by the end user/client. The doGet() request retrieves the milestones in the MilestoneDB array and sends the response to the dashboard.jsp page where each milestone in the array is listed as a card.
webApp.deleteMilestone.do	This link calls the function that removes a milestone from the milestone database and hence, the dashboard. The doGet() method sends a request to call the

	deleteMilestone() method for the milestone with the specified details, and returns the milestone dashboard with the updated milestone array using the webApp.listMilestone.do link
webApp.addMilestone.do	This link calls the function that adds a milestone to the milestone database and hence, the dashboard. The doPost() method send the milestone details entered by the user to the addMilestone() method. In turn, the response is an updated milestone dashboard with the added milestone and this is done through the response redirect to the webApp.listMilestone.do link
webApp.updateMilestone.do	This link calls the function that updates a milestone in the database and hence, the dashboard. The doGet() method sends a request to call the updateMilestone() method with the updated details, and returns the milestone dashboard with the updated milestone array using the webApp.listMilestone.do link
webApp.logout.do	This link ends the session for a user and redirects the user to the login page. The doPost() method in the LogoutServlet invalidates the session and forwards the request to the login page.

Persistence Mechanism and Database Schema

H2, a light-weight Java relational database management system, was used in this web application to persist user data and milestone details. It was chosen for this project as it can embedded in the web application. The Java API JDBC was used to establish a connection with the database.
For the web application, a single database called milestonedb was used. The database contains two tables namely; milestonelist and userInfo. Table 1 below summarizes the schemas of the tables.

***Table 2**: milestonelist and userInfo schemas*

Table Name	**Columns**	**Column Constraints**
milestonelist	**milestoneId** *A unique id that is randomly generated by the application*	INT NOT NULL PRIMARY KEY
	milestoneName *The name of the milestone*	VARCHAR Maximum character size - 30

	project *The name of the project where the milestone belongs*	VARCHAR Maximum character size - 30
	description *The description of the milestone.*	VARCHAR Maximum character size - 255
	isComplete *Keeps track of the completion status of the project*	VARCHAR Maximum character size - 30
	completionDate *The date when the milestone was completed*	VARCHAR Maximum character size - 30
userInfo	**id** *A unique id that is generated by the database*	INT NOT NULL PRIMARY KEY
	name *The username of the user*	VARCHAR Maximum character size - 30
	hashPassword *The hashed password of the user*	VARCHAR Maximum character size - 255

To persist the data obtained from the client side, a data access object (DAO), MilestoneDB.java, was implemented. The DAO enabled the creation of the database and its tables (milestonelist and validatelist). Furthermore, the DAO had methods such as updateMilestone(), confirmUser() and deleteMilestone() that performed CRUD operations in the database.

The process of persisting data starts from the client side where data is sent using an HTTP request. The relevant servlet processes the HTTP request and then, using the DAO's services, sends commands to the database. The database processes the commands and returns the results back to the servlets which send them to the client using HTTP response.

Functionality of Application

The web-based milestone planner application is an online productivity tool that allows users to keep track of key project milestones. Users of this web app will be able to login and logout at their convenience, and interactively use their milestone dashboard to add, delete, edit, and track incomplete milestones for specified projects. The main functionality of the milestone application are the following:

1. **Registration:** All first-time users are required to register in order to gain access to the application

2. **Secure login and logout:** All users are required to login with a unique username and password in order to access their milestone dashboard
3. **Display all milestones:** All milestones added by the user would be visible on their dashboard as milestone cards. Each milestone will card will have the milestone name, project name, milestone description, milestone due date, and action buttons to mark as complete, edit, delete, and/or share
4. **Add a milestone:** All users will be able to add a new milestone to their milestone dashboard with all required details entered and saved
5. **Edit a milestone:** Users would be able to edit any and all details of incomplete milestones on their dashboard.
6. **Delete a milestone:** Users will be able to delete milestones from the dashboard
7. **Mark as complete:** Users should be able to mark a milestone as complete upon completion. This action saves the completion date of the milestone and removes it from the milestone dashboard
8. **List incomplete milestones:** The milestone dashboard will display all incomplete milestones. A milestone is only complete once the completion checkbox has been clicked.
9. **Share milestone link:** A user will be able to share the link to a specific milestone externally. Clicking the share icon will provide the user with a shareable milestone link.

Test Report

Table 2: *Expected and actual results of testing the milestone application*

Test	Expected Result	Actual Result
Register a new user to the milestone app	SUCCESS Upon entry of the password and username, the user should be redirected to the login page.	SUCCESS The user is redirected to the login page and the registered name and password are safely stored in the database.
Log in to an existing account with the right username and password	SUCCESS The user should be redirected to the milestone planner dashboard and any milestone cards that were saved in previous sessions are displayed on the dashboard.	SUCCESS The milestone planner dashboard is displayed with previously saved milestone cards.
Login to the milestone app with an incorrect username/password	ERROR An error message should be displayed prompting the user to enter a valid username and	ERROR An error message is displayed prompting the user to enter a valid username and password

	password	
Select the add milestone icon	SUCCESS An add milestone modal should be displayed with input fields for the user to enter milestone details such as name, project name, description, and the due date	SUCCESS A modal is displayed with input fields for the user to enter milestone details such as name, project name, description, and the due date
Save new milestone	SUCCESS A new milestone card should be added to the dashboard and should contain the milestone details entered by the user in the *Add new milestone* modal	SUCCESS A new milestone card is added to the dashboard containing the milestone details entered by the user in the modal
Add a milestone without filling in all required fields	ERROR A tooltip should be displayed on every empty input field prompting the user to enter the required milestone details	ERROR A tooltip is displayed, prompting the user to fill in missing values.
Select the edit milestone icon	SUCCESS A modal should be displayed with a prefilled form containing existing details of the selected milestone	SUCCESS Details of the milestone are displayed in an edit form on a modal
Save edited changes to a milestone	SUCCESS Once changes have been made to a milestone and saved, the original milestone card should be updated to reflect the changes made	SUCCESS The original milestone card is updated to reflect changes made by the user.
Delete a milestone using the delete icon	SUCCESS The milestone card and all milestone details on the card should be removed from the milestone dashboard	SUCCESS The milestone card and all milestone details on the card are removed from the milestone dashboard
Mark a milestone as complete	SUCCESS The application should save the completion date of the milestone and the completed milestone should be removed from the milestone dashboard	SUCCESS The completed milestone is removed from the dashboard and the completion date is saved on the milestone application.

Share a milestone	SUCCESS	SUCCESS
Logout of a milestone account	SUCCESS The user's milestone dashboard should be closed and the user should be redirected to the login page for a new login session	SUCCESS The user's milestone dashboard should be closed and the user should be redirected to the login page for a new login session

Application Security

The MileStone Planner is an application that manages and helps a user track productivity and it is important that the application is secure. The privacy of users was a priority in the process of developing the application. To ensure user information is secure and user accounts are not easily penetrated, the team decided and implemented the following:

User Authentication

This is a process of confirming the identity of a user. In the Milestone Application, the information collected and stored in the database as a form of identifying a user is username and password information. To authenticate a user, the level of authentication used is a 1-factor authentication. The steps involved in a 1-factor authentication are registration and login.

Storing User Password

According to the 7Safe Breach report (UK Security Breach Investigations Report, 2010), one of the main targets for stolen data is sensitive company information. Information stored in the MileStone Planner falls into this category. Project details and progress can both be considered confidential information to an individual and corporations. A threat the application faces is one by hackers. Black-hat hackers are usually fishing for compromised applications to access by getting a user's password. To combat this threat, it is pertinent that the passwords are safely stored in the database. To ensure secure storage of the passwords in the database even if the password file is compromised, the team decided on storing the passwords as hashes.

Hash Algorithm(PBKDF2)

The hashing algorithm chosen for the application is the PBKDF2 algorithm. The reason for choosing the PBKDF2 algorithm over others like MD5 and SHA-512 is because it is a slower algorithm and is great for combating a commonly used strategy by hackers - brute force. Another reason is that the configuration

strength is very strong as compared to MD5 and SHA-512. Other hash algorithms that can perform at the standard of the PBKDF2 are BCrypt and SCrypt.

- **How the Hash works:** A hash is a one-way cryptographic function that turns any amount of data by generating a non-reversible fixed-length String. Unlike encryption, the hash cannot be decrypted. The hash algorithm works for the 2-step process of the team's chosen 1-factor authentication.
    1. **Hashing Registration Details**
       For a user to register, the RegisterServlet collects registration information from the user. An instance of the HashPassword algorithm is created and the password collected is parsed to the createHash method of the instance. The password is hashed. The hash is collected and stored to the database alongside the username. Upon registration, there is a redirection to the login page.
    2. **Comparing Login Hash**
       To login, an instance of the HashPassword and parses the password collected to the createHash method of the instance. The hashed password is compared to the existing hash in the database and if the match is correct, the user is redirected to the dashboard.
- **Possible Attack and How Salting can Prevent Attack:** Attacks a hacker can conduct on the hashed password include: guessing the password from similar hashes, brute force and using lookup tables. To combat this, the password was salted. The salt randomizes the hash so, if 2 users have the same password, the resulting hashes will be very different. Salting ensures every single user's password has a unique hash.
- **Implementation of the Hashing Algorithm and Salting**: As recommended in the resources on BlackBoard, the hashing algorithm was not implemented from scratch. The algorithm used was based on the example presented in the resources on GCULearn in the HashPassword Class. The HashPassword class contains methods that compute the hash, compare the hashes and create the hash.

Implementation of a Password Policy

In addition to hashing the user password, a password policy was created as an extra layer of security. This goal of the password policy is to ensure a user cannot choose a password that can be easily guessed by hackers. The password policy does the following:

- Sets a minimum and maximum value of the length a password field can take both on the registration and login pages.
- Sets a title to give users context on password requirements.
- Sets acceptable values and gives users warnings when the values they enter are not acceptable values in the system.
- Sets character requirements and constraints for the password field on both registration and login pages.

Filtering and Session Management

- The Web filter ensures the user has a session before they are able to access the dashboard. Before implementing the filter, the user was able to access the dashboard using the URL but, filtering requests that the user logs in before they can view dashboard or interact with the dashboard.
- Sessions are implemented to track a user logging in to the system and ends when a user logs out of the system.

Constraints and Vulnerabilities of the Application Security

The following are constraints and vulnerabilities of the application's security:

- Sending and getting requests in the system is not track for other functionalities of the application.
- The database is not secure. It is not safe from SQL injections.
- Adding an additional layer of security e.g a firewall would have catered for attacks such as cross-site scripting(XSS) and SQL injections
- The password policy is not fully-fledged out. For instance, in the event that users do not remember their passwords, there is no plan set up to recover the password.
- A multi-factor authentication would have been a more secure authentication process and prevented automated attacks.
- There is no limit to the number of times logging in is possible in the event of an invalid password.

Bibliography

UK Security Breach Investigations Report. (2010). [online] 7Safe. Available at: https://www.7safe.com/about-7Safe/news/details/2010/01/01/uk-security-breach-investigations-report-released [Accessed 25 Apr. 2019].