Module 5- Computer Systems (2022-23)

Project

UNIVERSITY OF TWENTE.

Software Testing Document (STD) Template

Team ID: 29	**Project Name:** RaspberryPlant
Team members: Mayank Thakur, Chris Bosman, Louisa Hafferl, Meenakshi Girish Nair, Vithursika Vinasiththamby, Gyum Cho	**Mentor (s):** Priya Naguine and Radu Basarabá

Instructions:

1. Refer to the below table and complete all the sections with clarity.
2. Select those test strategies that are applicable to test your application.
3. Make sure to refer to the "Development-Security by Design Checklist" to see the possible vulnerabilities in your application.
4. Feel free to add features and test cases in the table that are essential to test your application.
5. You can use Selenium, SonarQube, and/or GitLab CI/CD to perform source code review, static and dynamic application testing, etc.

Test Strategy	Date (When did you perform the testing?)	Process/Function (Features to be tested)	Test Case	Step	Description	Status (Passed/Failed /Open)	Expected Results	Actual Result	Mitigation plan/Solutions	Review on the Mitigation plan (Passed/Failed)	Remarks on the Failed mitigation plan
Manual Testing		Web application	Logging in	1	User can enter credentials	**Passed**	User is led to the homepage	User is not led to homepage, but gets stuck on LogIn.	The credentials of the user are put in the authorization header, when sending the URL.	-	-
				2	The password should be at least 9 characters. As mentioned in the SBC checklist, the passwords are hashed in sha512	**Passed**	The password entered by user is at least 9 characters	Error message shown when password length is less than 9	-		
			Register page	1	User can enter credentials	**Failed**	User is registered in the database and will be led to the login page	No message is shown and user is not redirected to the login page	The register credentials of the user are put in the authorization header and send to the backend	Passed	-
			Homepage	1	User can add new plant	**Failed**	User can put required information about the plant.	User cannot define a new plant or add information about it	The user gets assigned two blank plants, which are further specified in the Plant page	Passed	-
				2	Plant will be added to the homepage	**Failed**	User can view their plants in the homepage	The plants are shown in the homepage, but there is no option to add new plants	The option to specify a new plant, next to the first plant is given in the plant page for the second plant.	Passed	-

			Plant page	1	User can see up to date information about the plants	**Failed**	User gets an overview of their plant	All the information about the plant is shown, since there is no connection to the data of the user	Implementing cookies should make sure that the user's plant data is showcased on the plant page	Passed	-
				2	User can click on vitals	**Passed**	User is led to a page with information about the vitals	User is redirected to a page with information about the vitals			
			Renaming pre existing plants	1	User can change the name of the plant by clicking on the edit button	**Passed**	User can rename their plant using the edit button	The plant names can be changed by the user			
			Getting notifications	1	User gets a notification when something important happens	**Open**	User receives email notification with information about the action to be taken	User does not get any notification (we have yet to implement this)	We changed the database and login to include user emails, and then created a new thread in the background which sends notifications.		
			Watering plants	1	If the amount of time specified by the user has elapsed, the plant should be watered	**Failed**	The user can water the plant right before and right after the watering time determined in the app. This means that the app can possibly overwater the plant.	The user watering the plant is independent of the watering times in the app. The app should water the plant at intervals from when it was last watered, and not at fixed times. This means that if the user waters the plant 10 minutes before the actual watering time, the app will not water the plant in 10 minutes, but wait another period before watering	We changed the implementation of the watering feature to keep track of the watering interval and the last time the plant was watered.	Passed	
API Testing		Web application	Login	1	Sending get request with correct username and password	**(partially) Passed**	Redirection URL to the user homepage	Redirection URL to the user homepage, however anyone can access this link	Redirection was not sufficient as anyone with the link can “login,” so we created cookies to manage sessions. This also makes it more secure while allowing more functionality	Passed	
			Register	1	Sending put request with new username and password	**Passed**	Redirection URL to login page	Redirection URL to login page			
			Logout	1	Sending a post request to logout URL, after having logged in	**(partially) Passed**	Nullification of sessionID value in user cookies. But user can still visit other pages (that they should not be able to access) and see stub HTML pages	Invalidation of session, inability to make any more get requests to other pages.	Ensure that the redirection URL is always sent, and in a format that can be used by the browser to actually redirect the user out of places they shouldn’t be	Passed	
			Getting plant and senor data	1	Send get request for required object with the user that it belongs to	**Passed**	A list of the object in JSON	A list of the object in JSON	-		

			Putting the watering schedule data into the database	1	Sending a put request with the watering schedule	**Open**	Status 200 and redirection to the settings page	Error: Need to implement the function	The function to send the put request needs to be created.	-	This is a pretty straightforward function to put data into the database, we just need to make a backend function for it.
Integration Testing		Web application, raspberryPi and sensors	Connecting to the database in python	1	Connecting to the database using psycopg2	**Failed**	The data is inserted into the database	The data send from the pi is seen in the right tables. However, with the sensor id being the primary key, we were not able to add multiple data to a single sensor	The database was modified and each senor table had valueid as the primary key, so that each individual sensor can input all the data it reads.	Passed	-
			Connecting to the database in java	1	Connecting to the database using JDBC	**Passed**	The data is successfully read and written from and to the database	The data is present in the database and is received from the database	-	-	-
			Receiving digital data from the sensor and adding it to the database	1	Using SQL queries to input the data into the right table	**Failed**	The correct data is put into the right table	The sensor reading that is printed on the console is put into the right table. However, the values are not what is expected	The sensors need to be calibrated before we send data to the database and some sensor data deviate a lot at the beginning. This needs to be fixed by grouping a specific number of data points and only adding it to the database if their standard deviation is lower than a specific value.	Passed	-
Security Testing (as mentioned in the Security by Design checklist)		Web application	Evaluating the database security	1	Trying to inject malicious input into the user input forms to get access to the database	**Failed**	Since prepared statements are used, an error will be shown when database queries are executed	Nothing happens	Prepared statements will be used for all SQL queries	Passed	-
				2	Trying to insert javascript code into the input boxes	**Open**	Since whitelisting is implemented, this will nullify the malicious code	N/A	An output sanitization class will read all the user input before processing (still needs to be implemented)	-	This class should now be too dificult to implement, and won't increase the server processing overhead by much.

Note: Refer to the following documentation on GitLab and SonarQube for clarity-

1. Source Code review with SonarQube: **https://docs.sonarqube.org/latest/**
2. GitLab integration with SonarQube: **https://docs.sonarqube.org/latest/analysis/gitlab-integration/**
3. SonarQube (Static Application Testing): **https://www.sonarqube.org/features/security/**
4. Gitlab (Static Application Testing): **https://docs.gitlab.com/ee/user/application_security/sast/**
5. GitLab (Dynamic Application Testing): **https://docs.gitlab.com/ee/user/application_security/dast/**

Prepared by: Dipti K. Sarmah (Project Coordinator)