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 led to the homepage	-	-	
				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	**Passed**	User is registered in the database and will be led to the login page	Successfully registered message shown and redirected to login page	-	-	-
			Homepage	1	User can add new plant	**Passed**	User can put required information about the plant.	User can input information about their plant			
				2	Plant will be added to the homepage	**Passed**	User can view their plants in the homepage	The plants are shown in the homepage			

			Plant page	1	User can see up to date information about the plants	**Passed**	User gets an overview of their plant	All the information about the plant is shown			
				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	**Passed**	User receives email notification with information about the action to be taken	User gets notified about their plant condition			
API Testing		Web application	Login	1	Sending get request with correct username and password	**Passed**	Redirection URL to the user homepage	Redirection URL to the user homepage			
			Register	1	Sending put request with new username and password	**Passed**	Redirection URL to login page	Redirection URL to login page			
			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.	-	-
Integration Testing		Web application, raspberryPi and sensors	Connecting to the database in python	1	Connecting to the database using psycopg2	**Passed**	The data is inserted into the database	The data send from the pi is seen in the right tables	-	-	-
			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	**Passed**	The correct data is put into the right table	The sensor reading that is printed on the console is put into the right table	-	-	-

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	**Open**	Since prepared statements are used, an error will be shown when database queries are executed	N/A	Prepared statements will be used for all SQL queries	-	-
				2	Trying to insert javascript code into the input boxes	**Open**	Since whitelisting is implemented, this will nullify the malicious code	N/A	A class would read all the user input before processing	-	-

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)