Module 5- Computer Systems (2021-22)

Project

UNIVERSITY OF TWENTE.

Software Testing Document (STD) Template

Team ID: 22	**Project Name:** RC Racing System
Team members: Liran Neta, Kristiyan Velikov, Laurens Neinders, Kağan Gülsüm, Rick Pluimers, Alex Petrov	**Mentor (s):** Alex Mo Mohammed Assaad

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/Security Testing	During Sprint 2	Authentication	Logging In with google	1	The Google button on the login page should redirect to Google oauth screen	**Failed**	Successful redirect to the Google Oauth page	Redirected to the error page. Error message: redirect_uri_mismatch	Check and replace the wrong redirect uri with the correct in the google api console	**Passed**	…
Manual/Security Testing	During Sprint 2	Authentication	Logging In with google	2	Access the main page for the race	**Passed**	After login with google account the user should be redirected to the main page of the web app	…	…	…	…

Manual/Security Testing	During Sprint 2	Authentication	Logging in for the first time	1	After successful login for the with google/oauth the user should be redirected to a page where he should write a username	**Failed**	The user should be redirected to a page where he should create a new username.	User was redirected to the main page of the web app without having an account in the database.	Fix back-end call logic and database checks.	**Passed**	...
Manual/Security Testing	During Sprint 3	...	Logging in for the first time	2	Trying to enter a name that already exists	**Failed**	A pop out box that says that the username is already taken should come out	The user successfully accessed the main page even though he entered a username that already exists. The username was not saved in the database.	Fix back-end call logic and exception handling	**Passed**	...
Manual/Security Testing	During Sprint 2	Authentication	Accessing pages without being logged-in	1	Trying to access web pages of the app without being authenticated	**Passed**	User was redirected to the main page.	...	...	**...**	...
Manual/Security Testing	During Sprint 4	Session expiration	Being inactive for more than 30 minutes	1	Trying to access a page of our web-app after 30 or more minutes of inactivity	**Passed**	User's session was successfully invalidated after 30 or more minutes of inactivity. He was redirected to the login page.	...	...	**...**	...
Manual/Security Testing	During Sprint 3	Authentication (Preventing XSS)	Inserting a inappropriate username	1	Trying to insert a malicious script in the field for a username when creating a new account	**Passed**	User was not able to insert a malicious username to the database. Pop up box was fired.	...	...	**...**	...
Manual Testing	During Sprint 4	Analyzing page loading time	Check when accessing the page if everything loads on time	1	Some calls from the front end to the back end take time. The processing of data also takes time. During this period the DOM in the front end is not correctly displayed.	**Failed**	The DOM in the front end should be displayed correctly at all time	While waiting for the REST API call to the back end to complete, the data is incorrectly loaded	Hide the processing of the data with loading animation	**Passed**	...
Manual/Security Testing	During Sprint 2	Using HTTPS (Preventing MitMA)	Accessing the page with http	1	When a user tries to access our page with HTTP it will be redirected to the HTTPS	**Passed**	The user was successfully redirected to the https	...	...	**...**	...

							variant of our web page				
Manual	During Sprint 3	Starting a race	Trying to start a normal race	1	Clicking on the race button while in the race page and completing a lap. User also only starts a race on behalf of itself. (Privileges assigned)	**Passed**	After completing a lap a pop out with the time of the lap of the user was displayed and the database was updated correctly with the new race containing his username	...	...	**...**	...
Manual	During Sprint 3	Starting a race	Trying to start a race after more than 30 minutes of inactivity	2	Clicking on the race button while in the race page after the user has been inactive for more than 30 minutes	**Passed**	The user was redirected to the login page since his session was expired	...	...	**...**	...
Manual	During Spring 3	Starting a race	Trying to start a race while there is someone already racing	3	Clicking on the race button while in the race page while there is an ongoing race.	**Passed**	A pop out stating that the user should wait because there is an outgoing race was displayed	...	...	**...**	...
Manual	During Sprint 3	Starting a race	Trying to start a race when there is no socket connection between the back-end and the raspberry pi	4	Clicking on the race button while in the race page when there is no connection to the Raspberry pi	**Passed**	A pop out stating that there was an error should be displayed	...	...	**...**	...
***API Testing**	During Sprint 4	User details correctness (Privileges assigned)	Getting the details as a player (user netaliran23@gmail.com)	1	Using Postman, check if we get correct details of the current user: https://localhost:8443/rest/player	**Passed**	{"username":"liranneta23","email":"netaliran23@gmail.com","name":"liran neta","wins":0,"losses":0}	{"username":"liranneta23","email":"netaliran23@gmail.com","name":"liran neta","wins":0,"losses":0}	...	...	...
***API Testing**	During Sprint 4	Self-results correctness	Getting my races (as the user netaliran23@gmail.com)	1	Using Postman, check if we get correct races of the current user https://localhost:8443/rest/myraces	**Passed**	[]	[]	...	...	...

***API Testing**	During Sprint 4	Leaderboards correctness	Getting all the races	1	Using Postman, check if we receive correctly all the races https://localhost:8443/rest/allraces	**Passed**	All the races as they are currently in the DB	All the races as they are currently in the DB	...	...	...
Unit Testing	During Sprint 3,4	DB functionality	Get all races	1	A method in the database class should get all races.	**Passed**	The number of output races is correct.	...	...	...	...
		DB functionality	Get all reces for user/no user	1	A method in the database class should get all races for a scpecific user. No races should be outputted if the user does not exist.	**Passed**	The races for the specified user return as output when the user exists. No races are returned when the user does not exist.	...	...	...	...
		DB functionality	Inserting new race	1	A method in the database class should insert a new race based on a user, race time and sector times.	**Passed**	A new race row exists in the DB with the correct data, relating to the correct user.	...	...	...	...
		DB functionality	Get username from email	1	A method in the database class should get a username based on an email.	**Passed**	The correct name is returned.	...	...	...	...
		DB functionality	Get all registered usernames	1	A method in the database class should get all registered usernames.	**Passed**	All DB usernames are returned.	...	...	...	...
		DB functionality	Check player registered	1	A method in the database class should check if a given user is registered by his email.	**Passed**	Return true if the email exists, false of the email does not exist.	...	...	...	...
		DB functionality	Inserting new user	1	A method in the database class should insert a new user.	**Passed**	A new row is inserted in the player table.	...	...	...	...
		DB functionality	Inserting a new user / existing	1	A method in the database class should not insert a	**Passed**	No new user will be inserted.	...	...	...	...

					new user given existing credentials.						
		DB functionality	Get all challenges	1	A method in the database class should get all the challenges that were done in total for a user. Also a method should get all pending and all sent challenges.	**Passed**	All challenges with set flag isfinished for that user are returned. Also if true flag is set, returns all pending challenges, if not, returns all sent challenges .	...	...	...	...
		DB functionality	Check if a user is friends with another user	1	A method in the database class should check whether two players are friends or not	**Passed**	True if friends, false if not.	...	...	...	...
		DB functionality	Check if user is in a challenge with another user	1	A method in the database class should check whether there is an ongoing challenge between two players.	**Passed**	True if in a challenge, false if not.	...	...	...	...
		DB functionality	Requesting a challenge	1	A method in the database class should request a challenge only to an existing user.	**Passed**	A new challenge is inserted in the challenges table with challenger, his race, challengee and isfinished flag set to false.	...	...	...	...
		DB functionality	Responding to a challenge	1	A method in the database class should respond to a challenge only if the challenge was meant for that user.	**Passed**	Set isfinished flag to true and update the race of the challengee. Nothing is updated if the id doesnt correspond to that challengee.	...	...	...	...

		DB functionality	Deleting a request	1	A method in the database class should delete a challenge request only if that request is pending for the specified user.	**Passed**	Delete a row if the correct user is specified.	...	...	...	...
		DB functionality	Get all friends with their wins and losses	1	A method in the database class should get all friends for a particular user and also their wins and losses.	**Passed**	All friends and their correct wins and loses.	...	...	...	...
		DB functionality	Sending a friend request	1	A method in the database class should create a new friendship between two users that does not already exist.	**Passed**	A new friendship row is inserted in the DB with the two users and the valid flag set to false.	...	...	...	...
		DB functionality	Responding to a friend request	1	A method in the database class should respond to a friend request by updating the flag to true.	**Passed**	The valid flag for that friendship is set to true.	...	...	...	...
		DB functionality	Getting all requests	1	A method in the database class should get all pending or all sent friend requests.	**Passed**	All pending requests with friend2 thsi user and valid flag set to false. Or all requests with friend1 this user and valid flag set to false	...	...	...	...
		DB functionality	Deleting a friendship	1	A method in the database class should delete a friendship and all its associated challenges.	**Passed**	The row of that friendship is deleted. All challenges that have friend1 and friend2 as challenger and challengee are deleted.	...	...	...	...

Security Testing		DB security with prepared statements (Preventing SQL Injection)	Trying to inject a Javascript code	1	A method that takes external input should not allow the javascript code to execute	**Passed**	Nothing is executed after injection.	...	...	...	...
		DB security with prepared statements	Trying to inject a SQL code	1	A method that takes external input should not allow the sql code to execute	**Passed**	No data is altered or accessed after injection	...	...	...	...
Static application testing and source code review.	22-10-2021	source code review	using sonarqube and sonarcloud	1	Using sonarqube to get a review on the source code whenever a commit to the master branch is done.	**failed**	Commiting to master will result in a sonarqube test report	Gitlab CI/CD connection problems with the sonarqube test server caused it to not work. Using sonarcloud instead doesn't work for out special version of gitlab.com (namely gitlab.utwente.nl)	making a local runner instead of using a shared one so the localhost test server would be found. using sonarcloud instead of sonarqube, this didn't work either.	**failed**	this is already talked about in the other columns
		Static application security testing	using gitlab CI/CD	2	Using the build-in gitlab CI/CD SAST feature to do Static security testing.	**failed**	commiting to master will result in the pipeline doing the SAST test cases.	For some reason, when SAST is added, the pipeline does not run anymore whenever a commit to master is done.	looking online if others have had similar experiences.	**failed**	This might be a problem specific to gitlab.utwente.nl as there is nothing to be found about this online.
Dynamic testing using gitlab CI/CD.	22-10-2021	security testing in the deployed environments.	CSP header	1	Tests whether the CSP header is set to improve security against e.g. Cross-site scripting	**open**	CSP is set	The CSP header is not set.	Setting the CSP header in the HTML of the webpages.	open	We have not yet implemented this.
Manual Testing	Throughout Spring 3	IR sensors	To see if the sensor detect movement	1	Simply use a moving object like your hand to check if the sensor sends back data to the small python script written for testing purposes	**Passed**	We should receive data once something blocks the IR sensor	The sensor was successful at detecting a moving hand or a moving screwdriver.	Buying a new higher quality sensor.	Passed	
	...	...	To see if the sensors detect the car moving	2	Use the sensors on the real track to see if it detects the car	**Failed**	The sensors should send back a correct signal to the python script once a car passes through	The sensors were not reliable in sensing the car and sometimes did not send any meaningful value back.	...	The new sensors proved to be better in detecting the car. (Passed)	
	...	...	To test the new quality sensors	3	Use the new sensors with a real	**Open**	The sensors can detect the car passing through	Tests are ongoing	Paint the car so that the IR sensor can see it easier.	Open	

			with the track and the car		application of the track and a race						
	10-02-2021	The challenge system	Test the back-end endpoints for sending and accepting a challenge	1	Use Postman to send requests to the respective endpoints in both correct and dirty manners.	**Passed**	The implementation should edit the database in a correct fashion when valid requests come in, and protect the database when invalid requests come in.	The implementation behaves as expected and catches illegal actions and renders them useless.	Debug the code and go through the logic again.	Passed	

*For the API Testing, we are using Postman,in such a way that we first login into the web application, and then use the Cookie in our API requests. The requests are kept neatly and arranged based on sections (requests related to players, other which are related to races, etc)

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)