Testing Document

Neolook Solutions
AI Switchboard

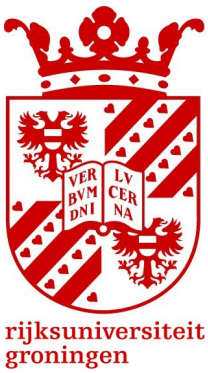
TA:

Ieva Randyte

Team:

Delia-Andreea Popa
Victor Andrei Didraga
Vasco Reynolds Brandao
Raluca-Alexia Neatu

University of Groningen



Contents

1 Introduction

Software testing is an important component of our project, as it ensures that all the specified requirements are met and that the system as a whole operates as intended. Thorough testing also facilitates the early identification and resolution of issues within the project, enabling corrections to be made before the product is completed. The aim of this document is to ensure that everything is working according to the needs of our client, Neolook Solutions, and that all their requirements have been met. In addition, we will explain the testing methods used to validate the functionality of our web application.

2 Acceptance Testing

Our web application underwent extensive acceptance testing in collaboration with our colleagues from Neolook Solutions. They were given the instructions listed below, along with tables to complete with their responses and feedback. Each of the acceptance tests were devised such that they can test the entirety of the user story which they are referring to. However, an acceptance test for US-5 was not included here, because the requirements associated with that user story were listed as a "Could" on the MoSCoW scale, and were not implemented in the final product.

2.1 US-1

- Given: The user has successfully logged into the web application.
- When: The user wants to conduct a model analysis on a piece of data.
- Then: Any of the available run-on-demand AI models can be run successfully on the given data.

The task is to follow the steps below and complete an analysis using one of the run-on-demand models.

1. Go to the Media page.
2. Choose a file.
3. Select the file type.
4. Click the “Upload” button.

5. Go to the Process Video page.
6. Choose the data format.
7. Choose a processing method.
8. Click on the file.

Criteria	**User Response**
Task difficulty	Easy
Issues Encountered	No
Passed?	Yes
Unsatisfied Requirements	None
Feedback	None

2.2 US-2

- Given: The user has successfully logged into the web application.
- When: The user wants to start a model analysis on a live feed of data.
- Then: The continuous AI models can be run successfully on the live data.

The task is to follow the steps below and run a continuous model on a live feed.

1. Add a camera from the Admin Panel.
2. Run the ffmpeg command in your terminal to start the stream.

Criteria	**User Response**
Task difficulty	Easy
Issues Encountered	None
Passed?	Yes
Unsatisfied Requirements	None

Feedback	Added camera by name. Simple & Straightforward.

2.3 US-3

- Given: The user has successfully logged into the web application.
- When: The user wants to run a model analysis and access the model output.
- Then: The output of the model can be successfully accessed and retrieved by the user after the model analysis is complete.

The task is to follow the steps below and download the desired model output.

1. Repeat the steps from US-1 to run the model analysis.
2. Go to the Media page.
3. If needed, filter the available files by the model output file type.
4. Click on the "Download File" button next to the file generated by the model.

Criteria	**User Response**
Task difficulty	Easy
Issues Encountered	None
Passed?	Yes
Unsatisfied Requirements	None
Feedback	Works well, I was able to download and watch the processed video.

2.4 US-4

- Given: The user has successfully logged into the web application.
- When: The user wants to be notified of any unexpected behavior detected on the live feed.

– Then: The notifications generated by the continuous models can be successfully accessed and viewed.

The task is to follow the steps below and access the notifications generated by the continuous model.

1. Follow the steps from US-2 to run the model.
2. Trigger the model by making a loud sound while the stream is running.
3. Go to the Notifications page. (The notifications can also be viewed in the database, and more details about them can be viewed in the log)

Criteria	**User Response**
Task difficulty	Easy
Issues Encountered	None
Passed?	Yes
Unsatisfied Requirements	No
Feedback	Make notification message a bit more specific (maybe which model triggered it)

2.5 US-6

– Given: The user has successfully logged into the web application.

– When: The user wants to access the model output of a given analysis.

– Then: The output of the model can be successfully accessed and retrieved by the user.

The task is to follow the steps below and download the desired model output.

1. Go to the Media page.
2. If needed, filter the available files by the desired model output file type.
3. Click on the "Download File" button next to the file that you want to download.

Criteria	User Response
Task difficulty	Easy
Issues Encountered	None
Passed?	Yes
Unsatisfied Requirements	No
Feedback	Minor addition: Filter should be able to go back into the state where all types are displayed.

2.6 US-7

- Given: The user has successfully logged into the web application.
- When: The user wants to upload a model output of a given analysis.
- Then: The output of the model can be successfully uploaded to the database.

The task is to follow the steps below and download the desired model output.

1. Go to the Media page.
2. Choose a file.
3. Select the file type.
4. Click the "Upload" button.

Criteria	User Response
Task difficulty	Easy
Issues Encountered	None
Passed?	Yes
Unsatisfied Requirements	None
Feedback	None

2.7 US-8

- Given: A new AI model is available.
- When: Old AI models are already included in the web application.
- Then: The new AI model is integrated in the web application.

The task is to assess the difficulty of the web application in regard to adding new AI models. There are no specific steps that need to be followed for this test, this aspect can only be assessed by inspecting the code.

Criteria	**User Response**
Task difficulty	Relatively easy
Issues Encountered	None
Passed?	Yes
Unsatisfied Requirements	No
Feedback	Project is structured in a way that promotes extendability/modularity at the code level. When adding a model, one must implement its corresponding type processor/interface. Once that's done, it can be simply added to the list of models to be used by the application.

3 Unit Testing

For unit testing, we created a unit test for every method that could be reasonably tested. To be more specific, because our project was developed in a web framework, there is code that is automatically added by Django, which did not need to be unit tested.

Below we attached a coverage report generated with the coverage package.

File	**statements**	**missing**	**excluded**	**coverage**
web_app/__init__.py	1	0	0	100%
web_app/admin.py	9	0	0	100%

web_app/ai_models/__init__.py	0	0	0	100%
web_app/ai_models/media_pipeline/__init__.py	0	0	0	100%
web_app/ai_models/media_pipeline/base_media_pipeline.py	48	48	0	0%
web_app/ai_models/media_pipeline/media_pipeline_analysis.py	11	5	0	55%
web_app/ai_models/media_pipeline/mediapipe_app.py	96	76	0	21%
web_app/ai_models/media_processing_interface/__init__.py	0	0	0	100%
web_app/ai_models/media_processing_interface/csv_processor.py	10	10	0	0%
web_app/ai_models/media_processing_interface/image_processor.py	10	10	0	0%
web_app/ai_models/media_processing_interface/json_processor.py	10	10	0	0%
web_app/ai_models/media_processing_interface/live_stream_processor.py	10	2	0	80%
web_app/ai_models/media_processing_interface/media_processor.py	9	2	0	78%
web_app/ai_models/media_processing_interface/txt_processor.py	10	10	0	0%
web_app/ai_models/media_processing_interface/video_processor.py	10	2	0	80%
web_app/ai_models/test_live_stream_ai_processing/__init__.py	0	0	0	100%
web_app/ai_models/test_live_stream_ai_processing/audio_analysis.py	24	14	0	42%
web_app/ai_models/test_video_ai_processing/__init__.py	0	0	0	100%
web_app/ai_models/test_video_ai_processing/video_analysis.py	42	32	0	24%

web_app/apps.py	21	6	0	71%
web_app/backends.py	19	4	0	79%
web_app/consumers.py	9	9	0	0%
web_app/forms.py	14	5	0	64%
web_app/migrations/0001_initial.py	5	0	0	100%
web_app/migrations/0002_alter_authtable_table.py	4	0	0	100%
web_app/migrations/0003_delete_authtable.py	4	0	0	100%
web_app/migrations/0004_initial.py	5	0	0	100%
web_app/migrations/0005_alter_csv_data_alter_image_data_alter_json_data_and_more.py	4	0	0	100%
web_app/migrations/0006_notification.py	6	0	0	100%
web_app/migrations/0007_notification_is_emergency.py	4	0	0	100%
web_app/migrations/0008_alter_notification_user.py	6	0	0	100%
web_app/migrations/0009_camera.py	4	0	0	100%
web_app/migrations/0010_remove_camera_ip.py	4	0	0	100%
web_app/migrations/__init__.py	0	0	0	100%
web_app/models.py	33	0	0	100%
web_app/notification_service.py	2	2	0	0%
web_app/tests.py	14	14	0	0%
web_app/tests_full/__init__.py	0	0	0	100%
web_app/tests_full/test_forms.py	13	0	0	100%
web_app/tests_full/test_media_view.py	380	0	0	100%
web_app/tests_full/test_models.py	49	4	0	92%

web_app/tests_full/test_notification_view.py	31	0	0	100%
web_app/tests_full/test_urls.py	46	0	0	100%
web_app/tests_full/test_user_login.py	18	0	0	100%
web_app/tests_full/test_views.py	141	0	0	100%
web_app/threads.py	19	12	0	37%
web_app/urls.py	3	0	0	100%
web_app/views.py	54	6	0	89%
web_app/viewslib/__init__.py	0	0	0	100%
web_app/viewslib/ai_processing_view.py	83	57	0	31%
web_app/viewslib/camera_manager_view.py	32	25	0	22%
web_app/viewslib/db_saver_view.py	28	20	0	29%
web_app/viewslib/frame_generator_view.py	52	34	0	35%
web_app/viewslib/live_feed_proccessing_view.py	14	8	0	43%
web_app/viewslib/media_view.py	121	19	0	84%
web_app/viewslib/notification_view.py	33	4	0	88%
web_app/viewslib/stream_renderer_view.py	11	0	0	100%
web_app/viewslib/user_login_view.py	22	2	0	91%
Total	**1608**	**452**	**0**	**72%**

3.1 Sonar Qube

Figure 1 illustrates the SonarQube report for our web application. The security issue is present as our web application is currently in an early security polishing stage, as it wasn't a necessary requirement that needed to be fulfilled. Most of the Reliability/Maintainability issues are targeted to addi-

tional function parameters that SonarQube labels as "misleading". Even so, the Django framework we used to develop the web application requires the presence of said function parameters. Among the issues mentioned, there are a couple of warnings regarding HTML/CSS sections, but these changes wouldn't have any impact on the quality of the code as they are purely design suggestions.

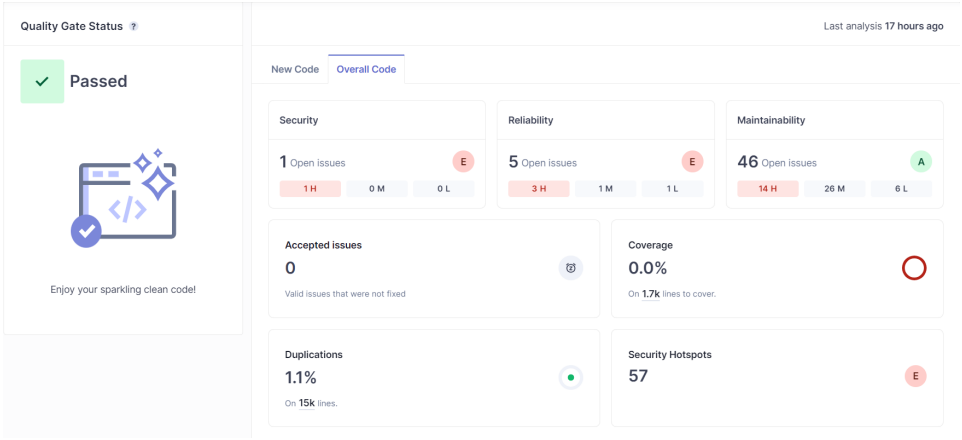


Figure 1: SonarQube report

4 Traceability Matrix

The traceability matrix shown in the table below links each requirement to its corresponding test, as well as specifying which module it relates to, which files are affected, and whether the test passes or not.

Requirement	**Files affected**	**Test**	**Passed**
M-RQ-F1	ai_processing_view.py	Acceptance test for **US-1**	Y
M-RQ-F2	views.py, media_view.py	test_upload_image_file	Y
M-RQ-F3	views.py, media_view.py	test_upload_video_file	Y
S-RQ-F4	views.py, media_view.py	test_upload_csv_file	Y
S-RQ-F5	views.py, media_view.py	test_upload_json_file	Y

S-RQ-F6	views.py, media_view.py	test_upload_text_file, test_invalid_file_type, test_invalid_file_extensions	Y
C-RQ-F7	ai_processing_view.py, media_view.py	Acceptance test for **US-1**	Y
C-RQ-F8	ai_processing_view.py, media_view.py	Acceptance test for **US-1**	Y
C-RQ-F9	ai_processing_view.py, media_view.py	Acceptance test for **US-1**	Y
C-RQ-F10	ai_processing_view.py, media_view.py	Acceptance test for **US-1**	Y
M-RQ-F11	db_saver_view.py	Acceptance test for **US-1**	Y
M-RQ-F12	camera_manager_view.py, frame_generator_view.py	Acceptance test for **US-2**	Y
M-RQ-F15	media_view.py	Acceptance test for **US-3**	Y
S-RQ-F17	media_view.py	test_download_text_file	Y
S-RQ-F18	media_view.py	test_download_csv_file	Y
S-RQ-F19	media_view.py	test_download_json_file	Y
S-RQ-F20	media_view.py	test_download_image_file	Y
S-RQ-F21	media_view.py	test_download_video_file	Y
M-RQ-F22	notification_view.py	test_emergency_notifications	Y
M-RQ-F23	db_saver_view.py, live_feed_processing_view.py	Acceptance test for **US-4**	Y
S-RQ-F26	media_view.py	Acceptance test for **US-6**	Y
S-RQ-F27	media_view.py	test_get_files_with_image, test_get_files_with_video, test_get_files_with_csv, test_get_files_with_json, test_get_files_with_text, test_get_files_with_all_file_types	Y
M-RQ-F28	media_view.py	Acceptance test for **US-7**	Y
M-RQ-F29	media_view.py	Acceptance test for **US-7**	Y

S-RQ-F30	media_view.py	test_delete_image_file, test_delete_video_file, test_delete_csv_file, test_delete_json_file, test_delete_text_file	Y
M-RQ-NF31	media_view.py	Acceptance test for **US-8**	Y

5 Temporary Document Notes

- colorcode acceptance/ unit tests in traceability matrix
- define passed terminology y/n
- center table text
- mention that reqs not implemented were skipped