Software Requirements Specification (SRS) Document

ROBUS TEST PLUGIN DEVELOPMENT
(VINAY, SRIRAGH, SRIKAR)

Brief problem statement: -

Android app developers code and compile their apps in popular IDEs such as Android Studio, INTELLIZ, ECLIPSE. But for testing and bug detection IDEs are not providing any options there. So, the developers are choosing other software's such as robus test. Our motive is to design a plugin in IDEs which will be linked to robus test so that developers can directly test and bug their apps from ide's itself which makes their work simpler.

System requirements: -

- The system has must be implemented with any of the IDE's such as android studio or Intelliz.
- The system must be capable of running on any platform that supports the Java runtime environment.

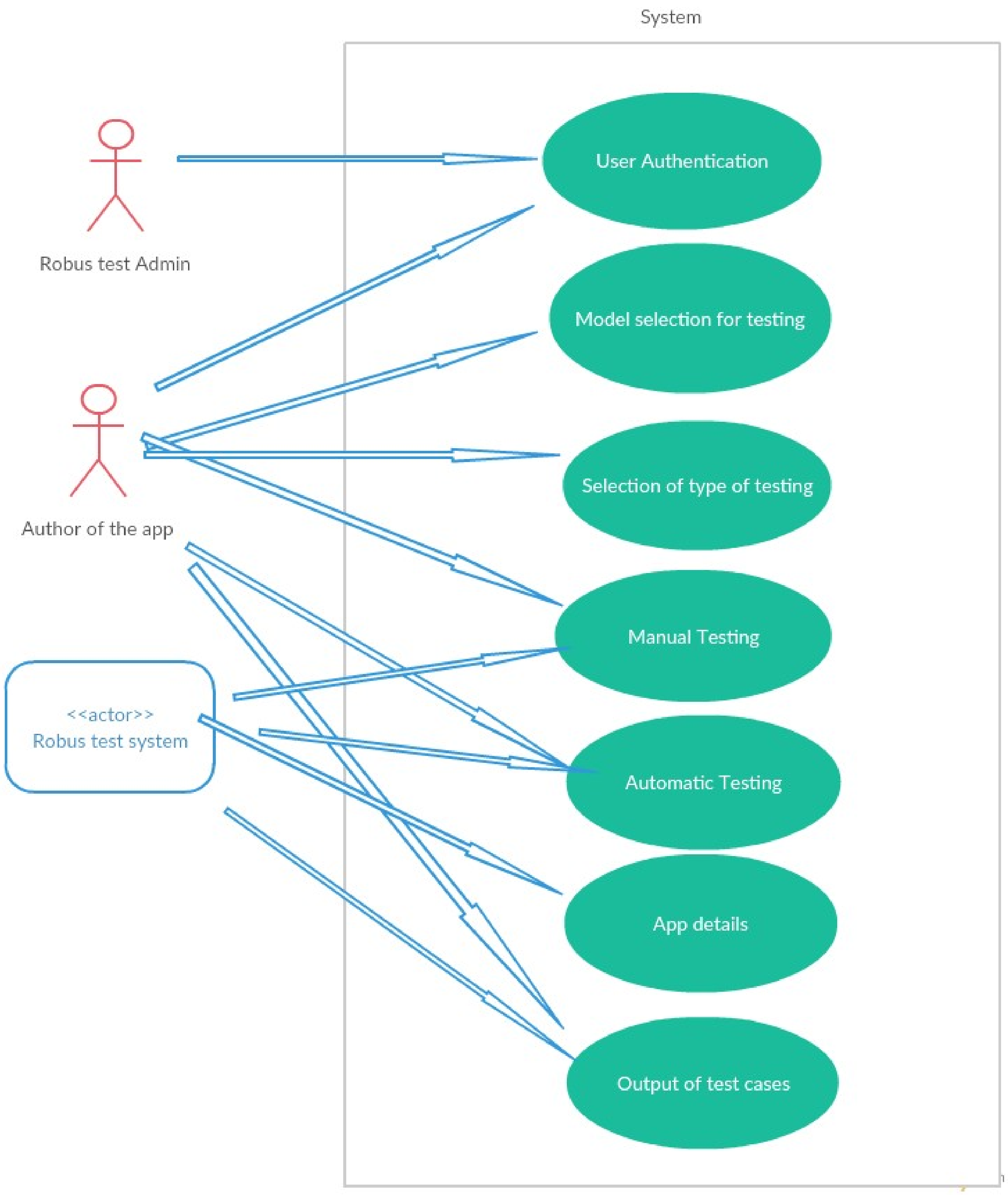
.

Users profile: -

The end users are basically the developers who are developing the apps by using various ide's such as android studio, eclipse and people employed for testing. The users just upload theirs apps and they can directly check the functioning of their app in any mobile phone's version.

Feature requirements (described using use cases)

No.	User Case Name	Description	Release
1.	User Authentication	User has to provide his/her credentials. Credentials are verified and appropriate error is displayed if credentials are wrong.	R1,R2
2.	Model avaliablility	The user is provided with UI to show the avaliable models to test on.	R1
3.	Selection of model to test and type of testing	The user has to choose between Manual testing or Automatic testing as well as the model he/she wishes to test on	R1,R2
1.	Manual Testing	The model being tested is displayed in working condition for manual testing by the user.	R2
4.	Automatic Testing	User provides the instructions to test the desired part of the app. The test case is run automatically and output is displayed.	R2
5.	App details	Details of the app being tested are displayed.	R2
6.	Output of test cases	Results of the test cases run are displayed along with some graphs showing details of the test case run.	R2



Use case diagram

Use case description

Use Case Number:	EU-0001
Use Case Name:	User Authentication
Overview:	User gains access to testing models.
Actors:	Author of the app.
Pre condition:	User has selected the authentication option.
Flow:	• System requests the user to provide credentials • User inputs his set of credentials • If the credentials can be determined to be authentic, System grants user the permission to use models available to test upon.
Post Condition:	System provides a list of models available for testing
Alternate Condition:	• If the credentials provided are wrong, then authentication fails • Error message is displayed for the user
Post Condition:	User is redirected to Authentication interface for another attempt

Usecase Number:	EU-0002
Usecase Name:	Model avaliability for testing
Overview:	User is provided with list of models avaliable for testing
Actors:	Author of the app
Pre condition:	User has successfully logged into RobusTest domain
Flow:	• System provides user with all the models avaliable for testing
Post condition:	User is now provided with types of testing on the model which he/she selects in the following step

UseCase Number:	EU-0003
UseCase Name:	Selection of type of testing and the model
Overview:	User is provided with two types of testing of which user has to select one and also select the model to test upon
Actors:	Author of the app
Pre Condition:	User has chosen one of the models avaliable for testing.
Flow:	• System provides the user with two option for testing namely, Manual and Automatic • User selects one of the ways of testing according to his/her needs
Post Condition:	Appropriate UI is displayed according to user's decision

Usecase Number:	EU-0004
Use case Name:	Manual Testing
Overview:	The model being tested is displayed in working condition for manual testing by the user
Actors:	Author of the app
Pre condition:	User has selected manual testing to be done on the model selected
Flow:	• System provides appropriate UI showing the model selected running the app • User is free to test any part or sub-part of the app without any previously written test cases
Post condition:	User left to test the app manually

Usecase Number:	EU-0005
Use case Name:	Automatic Testing
Overview:	User provides the instructions to test the desired part of the app. The test case is run automatically and output is displayed.
Actors:	Author of the app
Pre-condition:	User has selected Automatic testing to be done on the model selected
Flow:	• The test case to be run is predefined by the user • Test cases can be written so as to test certain parts of the app • Test case is run and the output is the result of the test case
Post condition:	The result of the test case is displayed

Usecase Number:	EU-0006
Use case Name:	App details
Overview:	Details of the app being tested are displayed
Pre condition:	Selection of type of testing is done
Flow:	• Details of the app that are important to the developer are displayed
Post condition:	The test case is run and output is displayed

Usecase Number:	EU-0007
Use case Name:	Output of test cases
Overview:	Results of the test cases run are displayed along with some graphs showing details of the test case run.
Pre condition;	User has selected the type of testing to be done
Flow:	• Displays whether the test case was run successfully or not • If not the appropriate errors are displayed
Post condition:	The User is provided to continue the testing or change the model or type of testing