Tampere University Unit of Computing Sciences
COMP.SE.610 Software Engineering Project 1
COMP.SE.620 Software Engineering Project 2

Project M

Mobile MESH

Test Plan

Roope Mantere
Neea Tienhaara
Diyaz Yakubov
Katja Heiskanen
Elsa Huovila
Vertti Nuotio
Jahid Sagor

Version history

Version	Date	Author	Description
0.1	15.03.2023	Diyaz	First draft
0.2	28.03.2023	Diyaz	Adding more details, defining test cases
0.3	31.03.2023	Diyaz	Adding more details, polishing

Contents

1 Introduction 4
1.1 Purpose and scope of document 4
1.2 Product and environment 4
1.3 Project constraints related to testing 4
1.4 Definitions, abbreviations and acronyms 4
2 Quality Assurance Process 6
2.1 Manual Quality Assurance Process 6
2.2 Automated Quality Assurance Process 6
3 Testing process 6
3.1 General approach 6
3.2 Definition of done and goals 7
3.3 Testing roles 7
3.4 Test schedule 7
3.5 Test documentation 8
4 Test cases 8
4.1 Unit testing 8
4.2 Integration testing 8
4.3 System testing 8
4.4 Special testing 9
4.5 Acceptance testing 9
4.6 Xyz (of your choice..) 6
5 Adopted Tools 11
6 Open issues 7
APPENDIX A […Z] 11

1 Introduction

1.1 Purpose and scope of document

This document outlines the test plan for the project titled Mobile MESH Network. The purpose of this document is to define the testing approach, process, and procedures to ensure the delivery of a high-quality* product.

1.2 Product and environment

The objective of the project is to develop an application that facilitates self-organizing MESH network among scouts. The application will utilize WLAN and/or Bluetooth technology to establish a network between mobile devices. The app runs on a mobile device, and the goal is that the application can be used in a context where there is no GSM connection available.

The intended user group for this application are scouts. Scouts will be moving in remote environments, where there is no existing internet network, and need to share information. The application is intended to work as a platform for information sharing and reporting, including standard reports via text and mission-specific tools for exchanging structured information, such as geolocations.

The application will also include built-in trust mechanisms to authenticate new nodes, ensuring the security and reliability of the network. The user interface should be designed to be intuitive and user-friendly, allowing the users to easily navigate the app's features.

1.3 Project constraints related to testing

It will depend on the success of the last few sprints how much of the app we can use for testing. The most relevant constraints for the project development are the technical abilities of group members, as well as the time limitations inside a course that also requires extensive documentation about the testing process.

The implementation of automated e2e tests is out of the scope of this course, it requires better infrastructure, expertise and more time. Moreover, there are some technical limitations, such as inability to run several emulators on one machine, that are impossible to overcome with current budget and time constraints.

A big testing constraint specific to this project is the requirement of running at least two devices in close proximity to form a MESH network connection and having to export the project into those phones. It means remote testing is likely out of the question and we need to find participants that are able to join us live.

There were no strict limitations from the customer other than a requirement for passing acceptance tests.

2 Quality Assurance Process

2.1 Manual Quality Assurance Process

The manual QA process will involve a team of [number of team members] testers who will perform functional and non-functional testing on the product. The testers will report defects using a defect tracking system and will retest the defects after they have been fixed.

2.2 Automated Quality Assurance Process

The automated QA process will involve running unit-tests in Kotlin with a help of JUnit testing framework. The automated tests will be executed using GitHub Actions CICD tool, where its job runners will be triggered by new changes (commits).

3 Testing process

The testing approach will follow an agile methodology. Testing will be done in parallel with development, and defects will be reported and resolved promptly.

3.1 General approach

The general strategy for testing can be outlined as follows:
Unit Testing:

- Unit tests will be developed using Kotlin and will be run using Android Studio.
- The unit testing framework used will be JUnit.
- All unit tests will be automated and will cover a minimum code coverage of 30%.

System Testing:

- System testing will be performed manually and will cover all functional and non-functional requirements.
- The testing will be done using Android Studio's emulator and on actual Android devices.
- Any issues discovered during testing will be documented in GitHub Issues for further analysis and resolution.

Acceptance Testing:

- Customer acceptance tests will be defined by the customer and will be performed manually.
- The acceptance criteria defined by the customer will be used to validate the product against the customer's requirements.
- Any issues discovered during acceptance testing will be documented in GitHub Issues for further analysis and resolution.

Usability Assessment:

- A basic usability assessment will be performed by the development team.
- The usability assessment will focus on the user interface and user experience of the application.
- Any issues discovered during the assessment will be documented in GitHub Issues for further analysis and resolution.

Overall, testing will be conducted in an iterative manner, with each phase building on the previous one. The development team will use GitHub for issue tracking and collaboration, and Android Studio will be used as the primary development and testing tool. The team will aim to ensure that the product meets all functional and non-functional requirements, as well as the customer's acceptance criteria.

3.2 Definition of done and goals

The definition of done for each testing phase will be as follows:

- Unit testing: All unit tests pass, and code coverage is at least 30%.
- System testing: The product meets all functional and non-functional requirements.
- Special testing: Any specialized testing required for the product is successfully completed.
- Acceptance testing: The product meets the acceptance criteria defined by the customer.

3.3 Testing roles

The team will perform manual and automated testing. The development team will also be responsible for writing and executing unit tests. Here is the full list of the roles of personnel related to testing, including other parties:

- Customer: responsible for defining acceptance criteria and participating in acceptance testing.
- End-users: responsible for using the application and providing feedback to improve its usability and functionality.
- Developers: responsible for unit testing and ensuring their code meets quality standards.
- Project Manager: responsible for overseeing the project and ensuring that testing aligns with the project goals and objectives.
- Stakeholders: responsible for providing input and feedback on the testing process and results.

3.4 Test schedule

The following test schedule will be followed:

- Unit testing: March - May
- System testing: April

- Special testing: April
- Acceptance testing: May

3.5 Test documentation

Test documentation will include test plans, test cases, and defect reports. Test logs and diaries will be used to document the results of testing activities, including any defects that can be found during testing. These logs will include the date and time of the test, the tester responsible, the actual results, and any defects found.

4 Test cases

Tests cases were chosen based on the team's capabilities and time constraints. We opted out the System Level testing and tests automation. The only possible tests automation would be unit testing that would be incorporated into the GitHub pipeline.

4.1 Unit testing

Unit testing Unit tests will be written by the development team using Kotlin and JUnit testing framework. Test cases will cover all critical and non-critical code paths.

4.2 System testing

System tests will be performed by the testing team to ensure that the software meets all the functional and non-functional requirements specified in the requirements documentation.:

- Verify that the app can establish a MESH network between two devices using WLAN or Bluetooth technology.
- Verify that the app can authenticate new nodes joining the network and maintain the security and reliability of the network.
- Verify that the app can share information and reports, including geolocation information, between devices on the network.
- Verify that the app can handle multiple devices joining and leaving the network without affecting the network's overall functionality.
- Verify that the app can work in a context where there is no GSM connection available.

Due to a lack of time the aforementioned tests would be performed manually.

4.3 Usability testing

Our goal is to recruit a few (1-3) people outside of the project crew to test the functionalities and the usability of the application. Since the application is a

quite generic chatting application despite the intended user group being scouts or military personnel, and since the time frame for testing the application is rather slim, we are not paying close attention to if the recruited person is an actual end-user or not. Also, due to the nature of the application, it might be better to have the system tests in person, which also limits the recruiting process a bit. Still, we acknowledge the fact that testing with the actual end-users would be a better option.

The test task would include the user successfully forming a network connection with another phone on their own and sending a message to them.

4.4 Special testing

Special testing Any specialized testing required for the product will be performed by the testing team. Test cases will be designed based on the specific requirements.

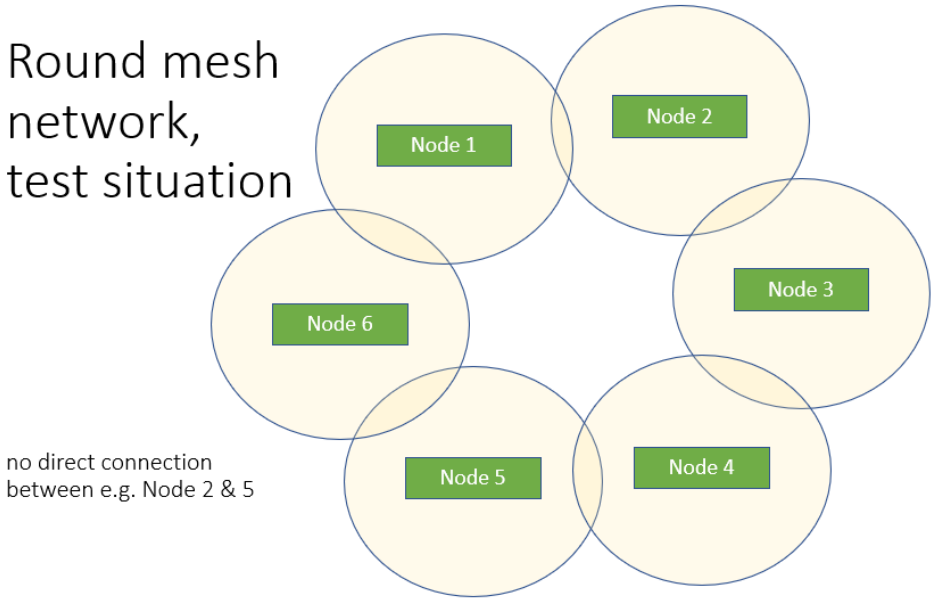
- Test the app's ability to handle unexpected network disruptions, such as a device being turned off or out of range.
- Test the app's ability to handle data being transferred between/through devices on the network.

4.5 Acceptance testing

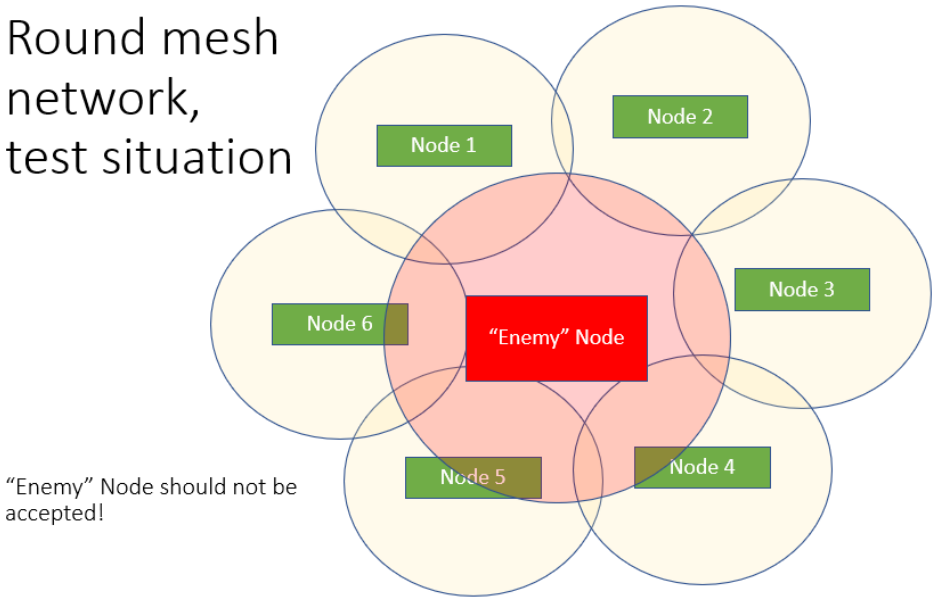
Acceptance testing Acceptance tests will be performed by the customer to ensure that the product meets the acceptance criteria.
Tests will be done in Tampere, either near Hervanta Campus or near Solita's office at Peltokatu 26 (there we could have for example a network where nodes are close to Tori and opposite side of railroad (there is a bridge close to Solita's office)
First: Close GSM data connection so that only WLAN (and maybe but not required) BT data transfer is allowed.

- Network Connectivity Test: This test will verify that devices can connect to each other and communicate over the mesh network. It will involve setting up a test network with multiple devices and verifying that each device can communicate with other devices in the network.
    - Set up the network, 2 devices, then 3rd & 4th & 5th or maybe more devices. 5 is needed at minimum.
    - The nodes should be kind of a ring, so that there is no connection from the opposite side of ring, but the message should travel

through nodes.



- Node Discovery & Routing Tests: These tests will verify that devices can discover other devices on the mesh network and verify that the mesh network can route data between devices.
  - Send messages from device A to device N, send message from device B to all devices.
  - Close the WLAN&BT from a device that is between two devices.
  - Send some new messages – they should be routed in a new route, or the sender should get information that there is no route between nodes.
- Security Test: One device tries to join the network, but it will not be accepted, no messages should go through it. Note: we don't test if the signals can be listened, in real world application the radio network would be encrypted.



- Usability Test: The application should be easy to adopt, we'll have one user that hasn't seen this before and (s)he will try to use the app.

- Compatibility Test: The application will be used in at least 2 different Android phones.

5 Adopted Tools

At present, the project exploits Junit test framework for unit tests and the GitHub actions as an CICD tool for automation.

APPENDIX A […Z]