Learning Outcome 3 Supporting Document

Tests Completed:

Using the test plans designed in LO2, the following test cases have been implemented:

- User Acceptance Testing (System Testing)
- HAR Algorithm Testing (Unit Testing)
  - LOOCV Algorithm
  - Testing on previously unseen data
- ValidatorTest (Unit Testing)
  - testValidPassword()
  - testValidEmail()
  - testValidMAC()
- SharedPreferencesTest (System Testing)
  - setSharedPreferencesTest()
  - editSharedPreferencesTest()
- DemandTest (System Testing)
  - testDemand()
- ModifyPasswordTest (Integration Testing)
  - modifyPasswordSuccessTest()
  - modifyPasswordFailShortPasswordTest()
  - modifyPasswordFailTest()
  - modifyPasswordEmptyCurrentPasswordTest()
  - modifyPasswordEmptyNewPasswordTest()
  - modifyPasswordEmptyFieldsTest()
- ModifyEmailTest (Integration Testing)
  - modifyEmailSuccessTest()
  - modifyEmailFailTest()
  - modifyEmailFailDuplicateEmailTest()
  - modifyEmailFailExistingEmailTest()
  - modifyEmailFailEmptyEmailTest()
  - modifyEmailFailEmptyPasswordTest()
  - modifyEmailFailEmptyFieldsTest()
- AuthenticationTest (Integration Testing)
  - logInSuccessTest()
  - logOutSuccessTest()
  - authenticationFailWrongEmailTest()
  - authenticationFailWrongPasswordTest()
  - authenticationFailEmptyEmailTest()
  - authenticationFailEmptyPasswordTest()
  - authenticationFailEmptyFieldsTest()
- AccountDeletionTest (Integration Testing)
  - deleteAccountSuccessTest()
  - deleteAccountFailWrongEmailTest()
  - deleteAccountFailWrongPasswordTest()
  - deleteAccountFailEmptyEmailTest()

    - deleteAccountFailEmptyPasswordTest()
    - deleteAccountFailEmptyFieldsTest()
- AccountCreationTest (Integration Testing)
    - createAccountSuccessTest()
    - createAccountFailTest()
    - createAccountFailEmptyNameTest()
    - createAccountFailEmptyEmailTest()
    - createAccountFailEmptyPasswordTest()
    - createAccountFailEmptyFieldsTest()

Planned Tests vs. Implemented Tests:

The implemented tests compare well with the test plans created in LO2 in terms of their functionality and quality coverage. The test plans perfectly outline the expected behaviour, operation and performance of each attribute that is to be tested, therefore providing a clear instruction while creating and implementing the tests.

Code coverage and yield are two important evaluation criteria to consider when testing a system. This is because they provide very important insights into the thoroughness and reliability of the testing process. Code coverage is used to describe the percentage of code that has been executed during testing, and serves an estimate of how much and how thoroughly the code has been tested. For the HARty system, all the implemented tests achieved a code coverage of 100%, indicating that the requirements have been well tested.

Yield is also another important evaluation criteria to consider as it represents the percentage of test cases that have passed, and serves as an estimate for if the software is functioning as expected. For the HARty system, all of the implemented tests achieved a yield of 100%, which further indicates that the requirements have been well tested.

The choice of using code coverage and yield as evaluation criteria is motivated by the need to build confidence in the HARty system and ensure that the system meets all the requirements. Using these evaluation criteria helps to determine the quality of the system and the testing process, and provides us a comprehensive understanding of the system's state of completion. In addition to this it also helps us identify areas that need improvement, so that when the system is completed end users are left with a high-quality system.

Chosen Testing Techniques:

Android:

The chosen testing techniques for testing in the Android application, are the Espresso and JUnit testing frameworks. The Android Orchestrator is used to automate the testing within HARry, as it helps improve test efficiency and management. Espresso is a UI testing framework specifically designed for the Android environment, that allows developers to test the user interface of the Android application by simulating user interactions and behaviour. On the other hand, JUnit is a Java unit testing framework that allows developers to write tests for individual components present in the codebase.

By combining the Espresso and JUnit testing framework, we are able to throughly test the Android application. These are appropriate testing techniques within the Android environment as they provide very comprehensive support for testing functional and quality requirements in Android applications.

HAR Algorithm:

The chosen testing techniques for testing the ML model are the LOOCV method, and testing on unseen data. The code for these techniques was executed in a Jupyter notebook environment, as it makes easy and quick to evaluate the results of the model.

Leave One Out Cross Validation (LOOCV) is a testing strategy used to ensure that every data point is used both in the training and validation sample. This is done to avoid introducing any biases in the model's evaluation. It also helps us obtain a more reliable estimate of the model's performance.

Testing the HAR algorithm on unseen data is important to evaluate the models' performance on data they have never encountered during training, as this allows us to simulate a real-world scenario where the model will be used and helps us avoiding overfitting the ML model.

By combining LOOCV and the method of testing on previously unseen data, we are able to get a reliable evaluation of the HAR algorithm and a clear understanding of the performance of limitations of the model.

System Testing:

The chosen testing technique for testing the system is the User Acceptance Testing (UAT) method. UAT focuses on verifying whether the system meets the end-users requirements and is ready to be released / delivered. It ensures that the final system is user-friendly, and high quality. In addition to this it allows end-users to essentially shape how the final system turns out, by allowing them to review the system after using it in a real-world setting. Therefore, by using this method we are able to determine the quality and performance of the system in the real world.

Results of Testing:

Android Test Results - The test results for this are too big to copy and paste here, so please view them at this file path "*/Software-Testing/Supporting-Documents/Test-Results/Test Results - Android.html*". Alternatively you can view my implemented test cases at this file path "*/Software-Testing/Test-Cases/*" or within the *androidTest* and and *test* folder within my source code.

LOOCV Results - These test results can be viewed at the file path */Software-Testing/Supporting-Documents/Test-Results/LOOCV-Essential-Features.ipynb*".

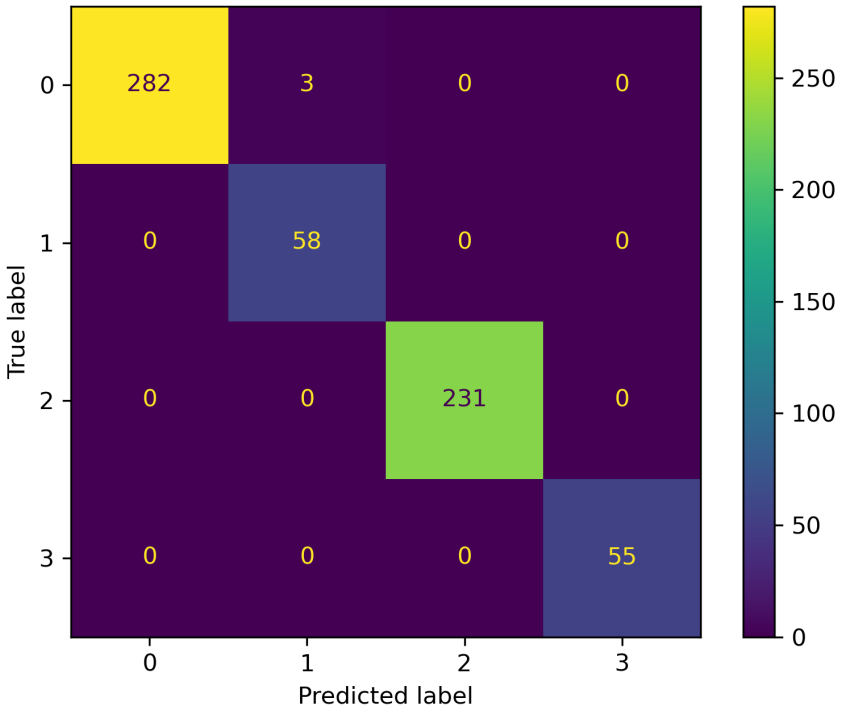
UAT Test Results - These test results can be viewed at the file path */Software-Testing/Supporting-Documents/Test-Results/UAT-Results.pdf*".

Essential Features Test Results on Unseen Data:

```
**************************************************************************
Classification report
**************************************************************************
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       285
           1       0.95      1.00      0.97        58
           2       1.00      1.00      1.00       231
           3       1.00      1.00      1.00        55

    accuracy                           1.00       629
   macro avg       0.99      1.00      0.99       629
weighted avg       1.00      1.00      1.00       629
```



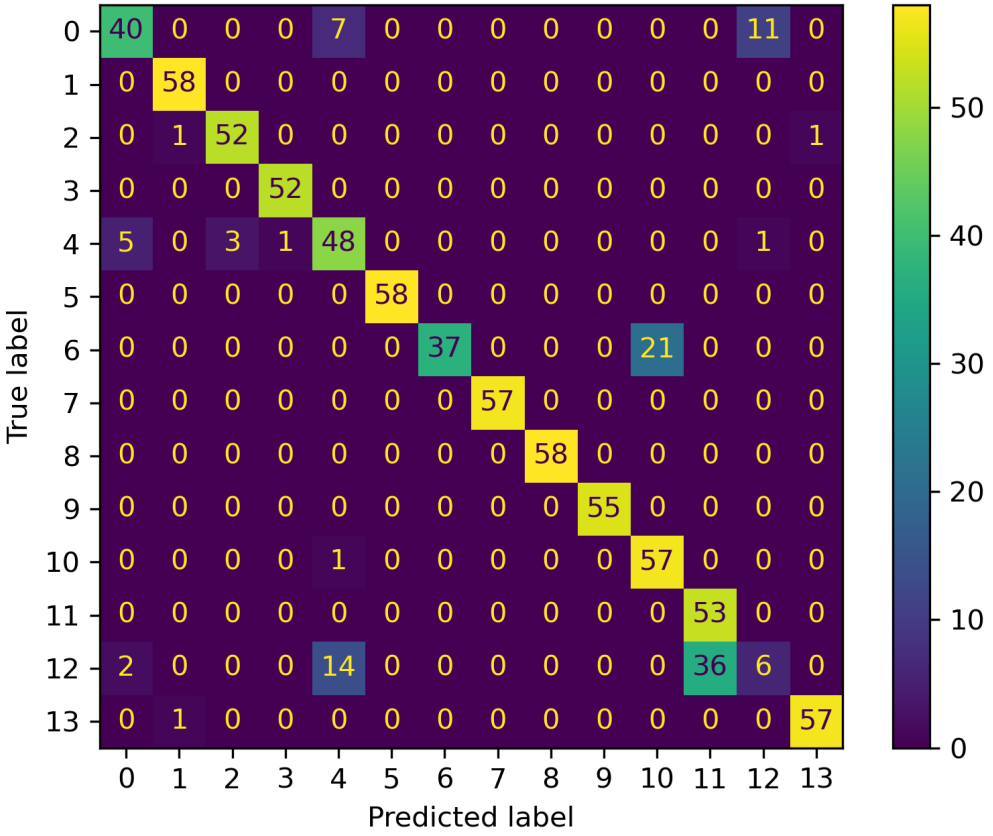
{‘Sitting/Standing’ = 0, 'Walking at normal speed' = 1, 'Lying Down' = 2, 'Running' = 3}

All Features Test Results on Unseen Data:

```
**********************************************************************************
Classification report
**********************************************************************************
              precision    recall  f1-score   support

           0       0.85      0.69      0.76        58
           1       0.97      1.00      0.98        58
           2       0.95      0.96      0.95        54
           3       0.98      1.00      0.99        52
           4       0.69      0.83      0.75        58
           5       1.00      1.00      1.00        58
           6       1.00      0.64      0.78        58
           7       1.00      1.00      1.00        57
           8       1.00      1.00      1.00        58
           9       1.00      1.00      1.00        55
          10       0.73      0.98      0.84        58
          11       0.60      1.00      0.75        53
          12       0.33      0.10      0.16        58
          13       0.98      0.98      0.98        58

    accuracy                           0.87       793
   macro avg       0.86      0.87      0.85       793
weighted avg       0.86      0.87      0.85       793
```



{‘Standing' : 0, ‘Movement' : 1, ‘Climbing stairs’ : 2, ‘Descending stairs’ : 3, ‘Desk work’ : 4, ‘Lying down left’ : 5, ‘Lying down on back’ : 6, ‘Lying down on stomach’ : 7, ‘Lying down right’ : 8, ‘Running’ : 9, ‘Sitting bent backward’ : 10, ‘Sitting bent forward’ : 11, ‘Sitting’ : 12, ‘Walking at normal speed’ : 13 }

These test results can also be viewed at the file path “*/Software-Testing/Supporting-Documents/Test-Results/*”.

Evaluation of Results:

The HARty Android application underwent a rigorous testing process, which involved 35 tests (System, Unit, and Integration) written using the test plans created in LO2. The results of the test are extremely positive, with the HARty Android application passing all 35 tests. In addition to these tests, we also used the Leave One Out Cross Validation method to measure the accuracy of the Essential Features HAR algorithm. The mean accuracy was calculated to be 97%, which is within our range of > 95%. Furthermore, the HAR algorithms were also tested on previously unseen data and was able to achieve 100% for the Essential Features HAR algorithm, and 87% accuracy for the All Features HAR algorithm (which is within our range of > 85%).

The User Acceptance Testing (UAT) was a great success, with users finding the HARty system easy to use and meeting all their requirements. The feedback from the UAT tests showed that there were not many improvements that needed to be made, which indicated that the system was well designed and user friendly.

To ensure compatibility, we ran the Android tests for our HARty application on 17 different Android devices with different combinations of OS versions. The results were good, and all compatible devices returned a yield of 100% while incompatible devices could not execute the tests. The Firebase Realtime Database service was also tested, and the results showed us that it was able to handle up to 6000 concurrent calls without the HARty system crashing, which is a positive sign regarding the reliability of the system.

Code coverage and yield was also analysed during the testing process. In the case of HARty, the high accuracy results and positive feedback from the UAT tests tell us that the code coverage and yield were high. This means that the testing process was effective. Therefore, we were able to test and for the following requirements:

- Usability
- Compatibility
- Reliability
- Small subset of Functionality Testing
- Small subset of Quality Testing

In conclusion, HARty's testing process was a huge success and has ensured that all the requirements tested were met. In addition to this, the high code coverage and yield indicate that the HARty system is a high quality and well designed system, that provides a good user experience.