Jarvis Emulator  
Test Plan  
COP 4331, Fall 2015

**Modification History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Who** | **Comment** |
| v0.0 | 10/3/2015 | Robin Schiro | Created document |
| v1.0 | 10/4/2015 | Jimmy Lam | Added my test cases |
| v1.1 | 10/5/2015 | Manuel Gonzalez | Added Speech Construction and Quality Assurance Test Cases |
| v1.2 | 10/5/2015 | Julian Rojas | Added Missing Test Cases |
| v1.3 | 10/6/2015 | Robin Schiro | Modified ref doc links |
| v1.4 | 10/8/2015 | Manuel Gonzalez | Modified test cases 10, 11, 12 |
| v1.5 | 10/8/2015 | Manuel Gonzalez | Removed test case 11 |
| V1.6 | 10/8/2015 | Julian Rojas | Modified Test Case 14 |

**Team Members:**

* Jimmy Lam
* Julian Rojas
* Manuel Gonzalez
* Robin Schiro

1. **Introduction**
   1. **Overall Objective for Software Test Activity**
      1. *Minimize Errors:* Tests will allow us to find bugs in the code, eliminate them, and verify that problems that had been created by those bugs no longer exist.
      2. *Improve Efficiency:*Because our application is heavily reliant on advanced algorithms, tests will permit us to hone in on algorithmic weaknesses to increase the speed and accuracy of the various modules in the application. These include facial recognition, face tracking, speech recognition, speech construction, and website data requests.
      3. *Maximize Usability:* User testing will contribute to the gradual improvement of modules that require user input. For example, testing will lead us through several iterations of our user interface until it is as intuitive as can be.
   2. **Reference Documents**
      1. [Concept of Operations](https://www.dropbox.com/s/uhnp6yg3z2rm68f/Concept%20of%20Operations.docx?dl=0)
      2. [Project Management Plan](https://www.dropbox.com/s/z3p8gxvhr7t3vlk/Project%20Management%20Plan.docx?dl=0)
      3. [Software Requirements Specification](https://www.dropbox.com/s/s8r4atqlh5720ma/Software%20Requirements%20Specification.docx?dl=0)
2. **Description of Test Environment**
   1. **Environment**
      1. Hardware
         1. Virtual Machine with the following specifications:
            1. CPU: Intel i5-3570K @ 3.40 GHz, 2 cores
            2. RAM: 2 GB
            3. Video Card: VMWare Virtual SVGA 3D Adapter
            4. Storage: 100 GB HDD
         2. Webcam
            1. Video Quality: 720p
            2. Snapshot Quality: 3 MP
      2. Software
         1. Windows 7 Operating System
         2. .NET Framework 4.5
         3. OpenCV
      3. Room
         1. The room in which the webcam is operating must be sufficiently lit for the webcam to record frames with at least 720p quality.
      4. The test environment contains the minimum specifications required to run the Jarvis Emulator. The software will operate well with any machine and webcam that possess at least these specifications.
   2. **Testers**
      1. All developers will serve as testers for this project. Each developer with possess a copy of the same Virtual Machine to run tests on.
      2. Optional testers will be our professor, Dr. Damla Turget, and our assigned TA, Amirreza Samiei.
3. **Stopping Criteria**
   1. **Discovery of Errors**
      1. On a periodic basis, we will execute all test cases (or at least all test cases that the software supports at the time of execution). When we find bugs and areas that could use improvement in the software, we will document our discoveries in a spreadsheet. Once the tester has gone through all test cases, he will place the spreadsheet in the project repository for the remaining developers to view. If the discoveries require more than one person to resolve, a meeting will be held to determine how tasks should be assigned.
   2. **No Errors Found**
      1. If no problems are discovered when all test cases have been run, the project will be considered “good enough to deliver”.
   3. **Definition of “Good Enough to Deliver”**
      1. “Good Enough” does not require that no known errors exist. With such a limited development time frame, there will undoubtedly be bugs in the product by the time we must deliver it. However, all issues that prevent any test cases from passing must be resolved by the time the project is completed. As such, the test cases will be designed to define the core features of the application. Near the end of the development cycle, time will be spent to improve the look and feel of the application, but cosmetics will not be a priority. Workarounds will only be acceptable after at least three attempts have been made to create a proper solution for a problem.
4. **Test Cases**

It is assumed that the application is already running in every test case.

|  |  |
| --- | --- |
| Facial Recognition | |
| **Objective:** Verify that the faces of different users are accurately recognized. | |
| **Test Conditions:** There are two or more trained users present (“trained user” is defined in SRS). | |
| **Description:** | **Expected Results:** |
| 1. Have one user walk into view of the webcam (within five feet, facing the camera). | The application should greet that user by name. |
| 1. Have the first user exit the room and a different user enter the room. This user should stand in a position similar to that described in Step 1. | The application should greet the new user by name. |

|  |  |
| --- | --- |
| Face Tracking | |
| **Objective:** Ensure that the application can track the position of a user’s face. | |
| **Test Conditions:** See Test Environment | |
| **Description:** | **Expected Results:** |
| 1. Click the ‘Enable Tracking’ button located in the ‘Video Feed’ tab of the application window. | A video feed coming from the webcam should appear inside the tab space. |
| 1. Toggle the ‘Display tracking borders’ option on. | You should see a square surrounding the face of each user in view of the webcam (as long as those users are facing the camera). |

|  |  |
| --- | --- |
| User Interface – Training – New User | |
| **Objective:** Verify that a new user can provide training data to the application. | |
| **Test Conditions:** The user has not already provided training data to the application. | |
| **Description:** | **Expected Results:** |
| 1. Go to the ‘Configuration’ tab and click the ‘New User’ button. | You should be redirected to the video feed tab. The feed should be visible, with ‘Display tracking borders’ enabled. |
| 1. Enter the name of the new User at the top of the ‘Video Feed’ tab. 2. Press the ‘Take Snapshot’ button as many times as you’d like, preferably with your face at various angles and with various expressions. | The training pictures should be stored in the folder specified within the ‘Configuration’ tab (default to the folder called ‘TrainingData’ in the same directory as the executable). The pictures for the user should be in a folder named after the user in the ‘TrainingData’ folder. |
| User Interface – Training – Existing User | |
| **Objective:** Verify that an existing user can provide additional training data to the application. | |
| **Test Conditions:** The user’s profile has already been created. | |
| **Description:** | **Expected Results:** |
| 1. On the ‘Configuration’ tab, select your name from the ‘Current User Profile’ dropdown menu. 2. Click the ‘Continue Training’ button. | You should be redirected to the video feed tab. The feed should be visible, with ‘Display tracking borders’ enabled. |
| 1. Press the ‘Take Snapshot’ button as many times as you’d like, preferably with your face at various angles and with various expressions. | The training pictures should be stored in the folder specified within the ‘Configuration’ tab (default to the folder called ‘TrainingData’ in the same directory as the executable). The pictures for the user should be in a folder named after the user in the ‘TrainingData’ folder. |

|  |  |
| --- | --- |
| User Interface – Configuration | |
| **Objective:** Verify that a user can customize his/her profile for the application. | |
| **Test Conditions:** Must be a trained user. | |
| **Description:** | **Expected Results:** |
| 1. On the ‘Configuration’ tab, select your name from the ‘Current User Profile’ dropdown menu. 2. In the ‘Applications’ area of this tab, input a string of words in the ‘Trigger Words’ field. In the corresponding ‘Application Path’ field, browse and select an application that you will interact with using your inputted trigger words. 3. In the ‘Websites’ area of this tab, input a string of words in the ‘Trigger Words’ field. In the corresponding ‘Website URL’ field, input the URL of the website that you will interact with using your inputted trigger words. 4. Click the ‘Save’ button and close the application. 5. Reopen the application, go to the ‘Configuration’ tab, and select your name from the ‘Current User Profile’ dropdown menu. | You should see the configuration settings that you had set before you had closed the application. |

|  |  |
| --- | --- |
| Voice Command | |
| **Objective:** Verify that Jarvis can recognize user voice commands. | |
| **Test Conditions:** The user is logged in and said the command clearly. | |
| **Description:** | **Expected Results:** |
| 1. A list of commands will be prepared and will be read clearly to Jarvis while using a microphone. | Jarvis should correctly output the words that the user said and respond accordingly. |

|  |  |
| --- | --- |
| Open other applications | |
| **Objective:** Verify that Jarvis can open other applications. | |
| **Test Conditions:** The user is logged in and clearly tells Jarvis to execute a valid command. | |
| **Description:** | **Expected Results:** |
| A list of windows applications will be made.  Jarvis will be instructed to open each application in a new window. | Jarvis should open up the application that the user called for in a new window. |

|  |  |
| --- | --- |
| Logging out | |
| **Objective:** Verify that Jarvis can log the user out of his or her computer. | |
| **Test Conditions:** The user is logged in and said the command clearly. | |
| **Description:** | **Expected Results:** |
| Jarvis will be told to log out of the computer. | Jarvis should be able to log the user out of his or her computer. |

|  |  |
| --- | --- |
| 1. Taking pictures of the user | |
| **Objective:** Verify that Jarvis can take a picture of the user for the user. | |
| **Test Conditions:** The user is logged in, positioned in front of the camera, and said the command clearly. | |
| **Description:** | **Expected Results:** |
| 1. A keyword for taking the picture will be said to Jarvis, and the user will pose for Jarvis. 2. The user will then check for the picture in the file that Jarvis saves it in. | Jarvis will take the picture and store it in a folder set by the user. |

|  |  |
| --- | --- |
| Notifying the user through speech | |
| **Objective:** Verify that Jarvis can correctly notify the user of system specific events. | |
| **Test Conditions:** The user is logged in. | |
| **Description:** | **Expected Results:** |
| 1. A list of fake events will be created (such as “System failure”, “System busy”, “Answer found”, etc.) 2. Each event will be triggered during runtime and with a user present. | Jarvis will verbally inform the user of the respective event once the event gets posted internally in the system. |

|  |  |
| --- | --- |
| Verbal response from user questions | |
| **Objective:** Verify that Jarvis can respond accordingly to preset user questions. | |
| **Test Conditions:** The user is logged in and said the question clearly | |
| **Description:** | **Expected Results:** |
| 1. The user will ask “How are you, Jarvis?” | 1. Jarvis should verbally respond with a general status of the system. |
| 1. The user will ask “Who am I, Jarvis?” | 1. Jarvis should state the name of the user, or say that it doesn’t know the user. |
| 1. The user will ask “How’s the day today, Jarvis?” | 1. Jarvis should verbally respond with the weather report from that day, or notify the user that he doesn’t have that information if the data is not available within 1 minute of the request. |
| 1. The user will ask “What model is my car?” | 1. Jarvis should respond that it doesn’t know this answer. |

|  |  |
| --- | --- |
| System response | |
| **Objective:** Verify that Jarvis’ average response time is below 5 seconds. | |
| **Test Conditions:** The user is logged in. | |
| **Description:** | **Expected Results:** |
| 1. Try a minimum of 10 randomly chosen events, commands or questions consecutively. | Jarvis should respond or notify the user accordingly within the 5 second threshold. |

|  |  |
| --- | --- |
| Memory usage | |
| **Objective:** Verify that Jarvis uses no more than 1GB of memory. | |
| **Test Conditions:** The user is logged in. | |
| **Description:** | **Expected Results:** |
| 1. Have the system working for several days without shutting it down. | When monitoring memory usage and storage they shouldn’t go over the threshold. |

|  |  |
| --- | --- |
| Jarvis acquires information from RSS Feed | |
| **Objective:** Verify that Jarvis can utilize weather forecast/news headlines data from a website. | |
| **Test Conditions:** Jarvis has already recognized the user. User says command clearly. | |
| **Description:** | **Expected Results:** |
| 1. The user will speak to Jarvis through the microphone. 2. Jarvis will then recognize the keyword and match it with the command associated with it. | Jarvis will then pull information from RSS Feed or website’s API, and either display it on the screen or speak it back to the user. |

|  |  |
| --- | --- |
| Detection of Active User – Multiple Users in View | |
| **Objective:** Verify that Jarvis can detect a user’s mouth movement. | |
| **Test Conditions:** All involved users are trained users. There are at least two faces in view of the camera. | |
| **Description:** | **Expected Results:** |
| 1. The user will speak to Jarvis. 2. Jarvis will detect multiple faces. 3. Jarvis will select an active user once it has noticed mouth movement. | If two or more users are present, Jarvis will try to recognize the active user depending on whether or not he/she is talking, hence mouth movement. |