

Real-Time Sign Language Transcription

Final Year Project

## PHASE – II

Submitted by

**Hamza Tariq 70113214**

**Hussnain Tariq 70110706**

**Hammad Akhtar 70111155**

**Project Supervisor Sir Sayed Zishan Ali**

**BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING DEPARTMENT OF SOFTWARE ENGINEERING, UNIVERSITY OF LAHORE**

## MAY 30, 2024

**Table of Content**

[Chapter 5: Testing](#_Toc167058949)

[5.1. Test Case Specifications 25](#_Toc167058950)

[5.2. Black Box Test Cases 26](#_Toc167058951)

[5.2.1. Equivalence Partitions (EP) 26](#_Toc167058952)

[5.2.2. Boundary Value Analysis 26](#_Toc167058953)

[5.2.3. Decision Table Testing 26](#_Toc167058954)

[5.2.4. State transition Testing 27](#_Toc167058955)

[Use Case Testing 27](#_Toc167058956)

[5.3. White Box Test Cases 27](#_Toc167058957)

[5.3.1. Cyclometric complexity 27](#_Toc167058958)

[5.4. Performance testing 27](#_Toc167058959)

[5.5. Stress Testing 27](#_Toc167058960)

[5.6. System Testing 28](#_Toc167058961)

[5.7. Regression Testing 28](#_Toc167058962)

[5.7.1 Selecting Regression Tests 28](#_Toc167058963)

[5.7.2. Regression Testing Steps 28](#_Toc167058964)

[Chapter 6: Tools and Techniques 29](#_Toc167058965)

[Chapter 7: Summary and Conclusion 29](#_Toc167058966)

[Chapter 8: User Manual 29](#_Toc167058967)

## 

## Test Case Specification

## Login

|  |  |
| --- | --- |
| Positive Test Case | |
| ID | LOGIN\_POSITIVE |
| Priority | High |
| Description | To verify user authentication to system. |
| Reference | Functional Requirement reference |
| Users | Administrator. |
| Pre-requisites | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. |
| Steps | 1. Open the web link to system. 2. Enter login id 3. Enter Password. 4. Press Login. |
| Input | Login id and password |
| Expected result | Successfully enters the system and main home page opens. |
| Status | Tested, passed. |

|  |  |
| --- | --- |
| Negative Test Case | |
| ID | TC\_LOGIN\_FAILURE |
| Priority | High |
| Description | To verify user authentication to system. |
| Reference | Functional Requirement reference |
| Users | Administrator. |
| Pre-requisites | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. |
| Steps | 1. Open the web link to system. 2. Enter login id. 3. Enter Password. 4. Press Login. |
| Input | Incorrect Login id or password or deactivated credentials. |
| Expected result | Does not allows access to system features and notifies the error. |
| Status | Tested, passed. |

## Signup

|  |  |
| --- | --- |
| Positive Test Case | |
| **ID** | LOGIN\_POSITIVE |
| **Priority** | High |
| **Description** | To verify user authentication to system. |
| **Reference** | Functional Requirement reference |
| **Users** | New users |
| **Pre-requisites** | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. |
| **Steps** | 1. Open the application 2. Navigate to the registration/signup page 3. Enter valid username/email, password, full name, address, and cell number 4. Click on the "Register" button |
| **Input** | Valid username/email, valid password, valid full name |
| **Expected result** | Successfully enters the system and main home page opens. |
| **Status** | Tested, passed. |

|  |  |
| --- | --- |
| Negative Test Case | |
| **ID** | TC\_LOGIN\_FAILURE |
| **Priority** | High |
| **Description** | To verify user authentication to system. |
| **Reference** | Functional Requirement reference |
| **Users** | New users. |
| **Pre-requisites** | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. |
| **Steps** | 1. Open the application 2. Navigate to the registration/signup page 3. Enter valid username/email, password, full name, address, and cell number 4. Click on the "Register" button |
| **Input** | Invalid/incomplete username/email, invalid/incomplete password, invalid/incomplete full name, invalid/incomplete address, invalid/incomplete cell number  . |
| **Expected result** | Does not allows access to system features and notifies the error. |
| **Status** | Tested, passed. |

## Uploading Video

|  |  |
| --- | --- |
| **Positive Test Case** | |
| **ID** | TC\_UPLOAD \_POSITIVE |
| **Priority** | High |
| **Description** | Verify that the user can upload a video file (mp4) |
| **Reference** | Functional Requirement reference |
| **Users** | Users |
| **Pre-requisites** | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. |
| **Steps** | 1. Open the application 2. Navigate to the Upload Video Section 3. Select Valid mp4 format file from storage 4. Click on the "Upload" button |
| **Input** | Valid MP4 file |
| **Expected result** | The video is successfully uploaded and displayed on the page, “Get Transcript” button appears |
| **Status** | Tested, passed. |

|  |  |
| --- | --- |
| **Negative Test Case** | |
| **ID** | TC\_UPLOAD \_FAILURE |
| **Priority** | High |
| **Description** | Verify that the user can upload a video file (mp4) |
| **Reference** | Functional Requirement reference |
| **Users** | Users |
| **Pre-requisites** | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. |
| **Steps** | 1. Open the application 2. Navigate to the Upload Video Section 3. Select invalid/corrupted mp4 format file from storage or any other file format. 4. Click on the "Upload" button |
| **Input** | Corrupted MP4 file, or Non-MP4 files |
| **Expected result** | Nothing uploads, no “Get Transcript” |
| **Status** | Tested, passed. |

## Camera

|  |  |
| --- | --- |
| Positive Test Case | |
| **ID** | TC\_CAMERA\_POSITIVE |
| **Priority** | High |
| **Description** | To verify camera availability |
| **Reference** | Functional Requirement reference |
| **Users** | Users |
| **Pre-requisites** | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. 4. User’s device has a camera of webcam privileges attached |
| **Steps** | 1. Open the application 2. Navigate to the Real Time Transcription Button |
| **Input** | Permission to use camera |
| **Expected result** | Successfully enters the real time transcription screen |
| **Status** | Tested, passed. |

|  |  |
| --- | --- |
| Negative Test Case | |
| **ID** | TC\_CAMERA \_FAILURE |
| **Priority** | High |
| **Description** | To verify camera availability |
| **Reference** | Functional Requirement reference |
| **Users** | Users. |
| **Pre-requisites** | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. 4. User’s device has a camera of webcam privileges attached |
| **Steps** | 1. Open the application 2. Navigate to the Real Time Transcription Button |
| **Input** | Does not allow camera permissions |
| **Expected result** | Does not allows access to real time transcription screen and notifies the error. |
| **Status** | Tested, passed. |

|  |  |
| --- | --- |
| Negative Test Case | |
| **ID** | TC\_CAMERA \_FAILURE\_02 |
| **Priority** | High |
| **Description** | To verify camera availability |
| **Reference** | Functional Requirement reference |
| **Users** | Users. |
| **Pre-requisites** | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. 4. User’s device does not have a camera of webcam privileges attached |
| **Steps** | 1. Open the application |
| **Input** | .- |
| **Expected result** | No Real Time Transcription Button Visible |
| **Status** | Tested, passed. |

## Downloading Transcripts

|  |  |
| --- | --- |
| Positive Test Case | |
| **ID** | TC\_TRACRIPTS \_POSITIVE |
| **Priority** | High |
| **Description** | To verify the system generates a transcript. |
| **Reference** | Functional Requirement reference |
| **Users** | Users |
| **Pre-requisites** | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. 4. A valid video has been procced successfully or a real time session has been ended |
| **Steps** | 1. Click on “Get Transcript” button if visible |
| **Input** | - |
| **Expected result** | Successfully downloads a .txt transcript and opens an editable dialog |
| **Status** | Tested, passed. |

|  |  |
| --- | --- |
| Negative Test Case | |
| **ID** | TC\_TRACRIPTS \_POSITIVE |
| **Priority** | High |
| **Description** | To verify the system generates a transcript. |
| **Reference** | Functional Requirement reference |
| **Users** | Users |
| **Pre-requisites** | 1. System is online. 2. User must have active login credentials provided by system administrator. 3. User has internet access. 4. A valid format video has been procced successfully or a real time session has been ended |
| **Steps** | 1. If the feed could not a understood for the whole length of the video or session, the system will notify user with error. |
| **Input** | valid format video or real time session with no sign-language |
| **Expected result** | If the feed could not a understood for the whole length of the video or session, the system will notify user with error. |
| **Status** | Tested, passed. |

## Black Box Testing

## Black Box Testing, alternatively termed Behavioral Testing, is a software testing approach wherein the tester is unaware of the internal structure, design, or implementation details of the item under examination. The tests conducted within this methodology can pertain to either functional or non-functional aspects, although they typically focus on functionality.

## Equivalence Partitioning (EP)

## Equivalence Partitioning (EP) is a highly employed technique aimed at reducing the volume of necessary test cases for system evaluation. This method is widely utilized to categorize inputs into equivalence classes, thereby streamlining the testing process.

## Login

|  |  |  |
| --- | --- | --- |
| Inputs | Valid Partitions | Invalid Partitions |
| Username/Email | Compulsory field. Case insensitive. Contains alphabets [A-Z, a-z], numeric keys [0-9], special keys [., \_]. Must contain @ sign and domain suffix (e.g., .com). | Empty field. Doesn’t contain @ or domain suffix. Contains invalid characters. |
| Password | Compulsory field. Contains more than 8 characters. May include symbols, alphabets [a-z, A-Z], and digits [0-9]. | Empty field. Contains less than 5 characters. Contains invalid characters (based on specific system rules). |

## Signup

|  |  |  |
| --- | --- | --- |
| Inputs | Valid Partitions | Invalid Partitions |
| Full Name | Compulsory field. Contains more than 1 character. Contains alphabets [A-Z, a-z] and spaces. | Empty field. Contains only numeric or special characters. |
| Email | Compulsory field. Case sensitive. Valid email address. Alphabets [A-Z, a-z], numeric keys [0-9], special keys [., \_]. Contains @ sign and .com. | Empty field. Doesn’t contain @ or .com. |
| Password | Compulsory field. Contains more than 8 characters. May include symbols, alphabets [a-z, A-Z], and digits [0-9]. | Empty field. Contains less than 8 characters. |
| Confirm Password | Compulsory field. Contains more than 8 characters. May include symbols, alphabets [a-z, A-Z], and digits [0-9]. | Empty field. Contains less than 8 characters. |

## Boundary Value Analysis (Character Count)

## Login

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Inputs | Minimum boundary | | Maximum boundary | |
| Min - 1 | Min | Max | Max + 1 |
| Email | 11 | 12 | 254 | 255 |
| Password | 7 | 8 | 254 | 255 |

## Signup

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Inputs | Minimum boundary | | Maximum boundary | |
| Min - 1 | Min | Max | Max + 1 |
| Email | 11 | 12 | 254 | 255 |
| Password | 7 | 8 | 254 | 255 |
| Name | 3 | 4 | 254 | 255 |

## Decision Table Testing (DT)

## Decision table is a testing method, which aims to ensure that each one of the possible branches from each decision point is executed at least once and thereby ensuring that all reachable code is executed.

## Signup

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Inputs | Values | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 |
| Name | T/F | T | F | T | T | F |
| Email | T/F | T | T | F | T | F |
| Password | T/F | T | T | T | F | F |
| Status | T/F | Signup | Error | Error | Error | Error |

## Login

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Inputs | Values | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 |
| Email | T/F | T | T | F | T | F |
| Password | T/F | T | T | T | F | F |
| Status | T/F | Login | Error | Error | Error | Error |

## Video Upload

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Values | Case 1 | Case 2 |
| Video | Valid/Invalid | Valid | Invalid |
| Status | Uploaded/Error | Uploaded | Error |

## State Transition Testing

## State transition testing is a methodical approach used to validate the behavior of a system when it undergoes changes in state based on various events or inputs. This testing technique ensures that all possible states and transitions are covered, helping to identify any unexpected behaviors or errors in the system's workflow.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| State | Name | Events | New User | Existing User | Login | Signup | Video Upload | View Transcript | Logout |
| S1 | Start | Open App | S1 | S1 | - | - | - | - | - |
| S2 | User Registration | Submit Registration | S2 | - | - | S2 | - | - | - |
| S3 | User Login | Submit Login | - | S3 | S3 | - | - | - | - |
| S4 | Video Upload | Upload Video | - | - | S4 | - | S4 | - | - |
| S5 | Request Transcript | Request Transcription | - | - | - | - | S5 | S5 | - |
| S6 | View Transcript | View Transcription | - | - | - | - | - | S6 | - |
| S7 | Logout | Logout | - | - | - | - | - | - | S7 |

## Use Case Testing

## Signup

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_001 | |
| **Use Case Name** | Registration/Sign up | |
| **Description** | This use case involves the process of creating a new account within the Sign Language Transcription System, allowing users to access and utilize the system's features. | |
| **Primary Actor** | User | |
| **Secondary Actor** | None | |
| **Pre-Condition** | The user must have access to the system's registration interface. | |
| **Post-Condition** | A new user account is successfully created, and the user gains access to the system. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
|  | * The user accesses the registration interface. * The user enters the required information, such as username, email, and password. * The user submits the registration form. * The system notifies the user of successful account creation. | * The system displays the account creation form. * The system validates the entered information. * The system processes the registration request and creates a new user account. |
| **Alternate Flow** | If the entered information is incomplete or fails validation:   * The system notifies the user of the validation error. * The user corrects the information and resubmits the form. * Steps 4 to 8 are repeated. | |

## Login

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_002 | |
| **Use Case Name** | Login | |
| **Description** | User can login to the system | |
| **Primary Actor** | User | |
| **Secondary Actor** | None | |
| **Pre-Condition** | The user must have access to the system's registration interface. | |
| **Post-Condition** | A new user account is successfully created, and the user gains access to the system. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
|  | * The user accesses the login interface. * The user enters their username/email and password. * The user submits the login form. | * The system displays the login form. * The system validates the login credentials. * The system processes the login request and grants access to the user. |
| **Alternate Flow** | If the entered credentials are incorrect:   * The system notifies the user of the authentication failure. * The user retries the login with correct credentials. * Steps 4 to 6 are repeated. | |

## Real Time Transcription

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | UC\_004 | |
| **Use Case Name** | Perform Sign Language Translation | |
| **Description** | This use case involves the system translating live Sign Language gestures into natural language in real-time. | |
| **Primary Actor** | User (Sign Language User) | |
| **Secondary Actor** | None | |
| **Pre-Condition** | The user must be logged into the system, and the device must have access to a camera. | |
| **Post-Condition** | The system successfully transcribes the Sign Language gestures into natural language. | |
| **Basic Workflow** | **Actor Action** | **System Action** |
|  | * The user selects the "Real- Time Transcription” option. * The user performs Sign Language gestures in front of the camera. | * The system activates the camera for live translation. * The system processes the live video feed, extracting key features. * The system applies Natural Language Processing (NLP) models to translate gestures into text. |
| **Alternate Flow** | If the system encounters difficulty in recognizing gestures:   * The system may prompt the user to adjust lighting or perform clearer gestures. * Steps 3 to 5 are repeated until successful translation. | |

## White Box Testing

## White testing is a testing technique that examines the program structure and derives test data from the program logic/code. The other name of glass box testing is clear testing, open box testing, logic drives testing or path driven testing or structural testing

## Cyclomatic Complexity

## Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measure the number the linearly-independent paths through a program module. Lower the program’s Cyclomatic complexity, lower the risk to modify and easier to understand

## 

## Sign-up

|  |  |
| --- | --- |
| **Cyclomatic Complexity** | |
| Cyclomatic Complexity (M) | E - N + 2 |
| Number of States (N) | 6 |
| Number of Transitions (E) | 6 |
| Cyclomatic Complexity (M) | 6 - 6 + 2 = 2 |

## Login

|  |  |
| --- | --- |
| **Cyclomatic Complexity** | |
| Cyclomatic Complexity (M) | E - N + 2 |
| Number of States (N) | 6 |
| Number of Transitions (E) | 6 |
| Cyclomatic Complexity (M) | 6 - 6 + 2 = 2 |

## Video Transcription

|  |  |
| --- | --- |
| **Cyclomatic Complexity** | |
| Cyclomatic Complexity (M) | E - N + 2 |
| Number of States (N) | 8 |
| Number of Transitions (E) | 8 |
| Cyclomatic Complexity (M) | 8 - 8 + 2 = 2 |

## 

## Real Time Transcription

## 

## 

|  |  |
| --- | --- |
| **Cyclomatic Complexity** | |
| Cyclomatic Complexity (M) | E - N + 2 |
| Number of States (N) | 6 |
| Number of Transitions (E) | 5 |
| Cyclomatic Complexity (M) | 5 - 6 + 2 = 1 |

## Performance Testing

## Performance Testing is a detailed assessment of the software program’s execution speed, reaction to user inputs, ability to provide the expected results, the manner in which it optimizes the resources, and its capacity to manage growing loads without a significant decline in performance. While functional testing is aimed at detecting errors in the software’s features and capabilities, the purpose of performance testing is to identify performance issues that may affect the usability of the software and its ability to perform optimally under regular usage conditions. undefined

## Speed: This aspect evaluates the ability of the software application to respond with speed to the actions of the users or system events. It ensures that the users do not have to wait for a long time or encounter any form of delay when using the software.

## Scalability: This also measures how well the software can handle the load of increasing number of users or transactions and still perform optimally. It assists in defining the maximum number of users that a given software can accommodate before its efficiency starts to decline.

## Stability: This dimension measures the ability of the software to perform in a steady manner when subjected to different loads and usage patterns. It aims at establishing any vulnerability or irregularity that may arise when the software is under different loads or usage.

## Stress Testing

## Stress testing is a specific type of software testing aimed at determining the stability of a software system in conditions that it was not designed to handle, or in other words, testing the software beyond its normal limit. Still, it is important to note that stress testing is most important for applications that are vital or heavily used but can be applied to any software system.

## Stress testing is particularly focused on the examination of the system stability, its capability to work in under adverse conditions, such as high loads, resource depletion, and other kinds of failures. While testing is usually concerned with ensuring that a piece of software behaves as required under normal conditions, stress testing is more interested in finding the system’s flaws or the ways it malfunctions when stressed to its limits.

## Key objectives of stress testing include:

## Availability: Determining the system’s accessibility and its ability to perform during peak usage or when there are competing demands on resources, so that users can access and use the system.

## Robustness: Determining the software’s capability to remain stable and performant when under pressure or in a hazardous environment and not freeze up or suffer through catastrophic malfunctions.

## Error Handling: Verifying the capacity of the system to alert the user or administrator and prevent or correct the errors or exceptions that may occur during periods of high stress on the application, while also preventing the loss or corruption of data.

## System Testing

## System Testing is a broad type of testing that checks the total effective working and compatibility of a software product as a component of the entire computer based system. This differs from unit testing or integration testing that are done on a component or a subsystem level, system testing aims at testing the software within its complete environment, including the black-box testing.

## System testing is carried out with one principal goal, and that is to validate that the software under test complies with the requirements and functions in the intended environment. Due to the nature of contact of many software systems with other software components, hardware devices, or other systems, et cetera, ensures that tested system integrates well and works in harmony with the other systems within the broader system architecture.

## Key aspects of system testing include:

## Completeness: Ensuring that the whole of the software product and any component, element or part of it performs its specified function and is compliant to the defined system requirement.

## Integration: Verification of various forms of software conjunction and information exchange at the interfaces between the modules, parts or sub-systems.

## Interfacing: Verification of the interfaces and communication interfaces used by the software in its interaction with other systems, databases or any hardware interfaces.

## Regression Testing

## Regression testing can be described as an important quality assurance engineering method which checks if recent changes or modifications to a software code base result in the creation of a new undesirable effect or if it has affected other desirable components or aspects of the software in a negative manner. It entails rerunning a selected set of a previously run test cases to check on the stability of a software after integrating change code into it.

## The main goal of regression testing is to minimize the probability of encountering regression defects – they can occur because of code entries or deletions, bugs, or other changes made to the system. Thus, regression testing enables to preserve the stability, reliability, and the essential similarity of the software product in the consequent releases or iterations while retesting the influenced portions of the software.

## Key principles of regression testing include:

## Verification: Ensuring that new code changes or update do not impact the capability of the previous functionalities of the software or avert any kind of hindrance, loss in overall performance.

## Risk Mitigation: Assessing and managing possible risks which may occur when making changes to the code; the goal is to maintain code purity and usability over the course of the software production.

## Efficiency: Improving the efficiency of execution of regression tests highlighting the important test cases, utilizing the automated tools, and avoiding repetition to gain faster results and more effective release processes.