

## SWE2005 – SOFTWARE TESTING

GVision

# Talk like a pro

**REVIEW 1**

## Submitted by:

Soutik Mandal – 21BEE1036

# Modules found for GVision :

## Module 1: User Interface (4\*4 Matrix Keypad)

This module will handle user input via the 4\*4 matrix keypad. The keypad will have braille numbers and alphabets for accessibility. The user will be able to make and receive calls and send and receive SMS messages.

## Module 2: Communication (GSM Module)

This module will handle communication with the mobile network using a GSM module. It will enable sending and receiving SMS messages and making and receiving phone calls.

## Module 3: Audio (Speaker and Microphone)

This module will handle audio input and output using a speaker and a microphone. It will enable playing audio and recording voice calls.

## Module 4: Power (Battery and Charging)

This module will handle power management, including battery charging and monitoring. It will ensure that the mobile phone can operate for a reasonable amount of time.

**Module 5: CODE**

# Requirements and respective test cases found for each module :

## The user interface should be easy to navigate, with clear instructions and prompts for each action.

* + 1. Test Case 1: Clear instructions and prompts
* Test Objective: To verify that the user interface provides clear instructions and prompts for each action.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Settings" menu.
      3. Select "Display & Sound" from the menu.
      4. Adjust the volume by using the volume buttons.
      5. Verify that the phone provides clear instructions and prompts on the screen for each action taken, such as "Volume increased by 10%".
    1. Test Case 2: Easy navigation
* Test Objective: To verify that the user interface is easy to navigate.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Contacts" menu.
      3. Select a contact from the list.
      4. Edit the contact's information.
      5. Navigate back to the "Contacts" menu.
      6. Verify that the user interface is easy to navigate, with clear labels and intuitive button placement.
    1. Test Case 3: User feedback
* Test Objective: To verify that the user interface provides feedback to the user.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Messaging" app.
      3. Compose a new message.
      4. Send the message.
      5. Verify that the phone provides feedback to the user, such as a confirmation message that the message was sent, or an error message if the message failed to send.

## The keypad should be responsive and able to register user input accurately and quickly.

* + 1. Test Case 1: Accuracy of Keypad
* Test Objective: To verify that the keypad accurately registers user input.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Phone" app.
      3. Dial a phone number using the keypad.
      4. Verify that each digit entered is accurately displayed on the screen.
      5. Verify that the phone number dialled is correct.
    1. Test Case 2: Responsiveness of Keypad
* Test Objective: To verify that the keypad is responsive and quickly registers user input.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Messaging" app.
      3. Compose a new message.
      4. Type a message using the keypad.
      5. Verify that the phone responds quickly and accurately to each keypress, with no noticeable delay or lag.
    1. Test Case 3: Usability of Keypad
* Test Objective: To verify that the keypad is easy to use and comfortable for users.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Notes" app.
      3. Create a new note.
      4. Type a long message using the keypad.
      5. Verify that the keypad is easy to use and comfortable for users, with

no difficulty in typing long messages or experiencing any discomfort while using the keypad for an extended period.

## The keypad should have braille numbers and alphabets for accessibility.

* + 1. Test Case 1: Braille numbers
* Test Objective: To verify that the keypad has braille numbers for accessibility.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Phone" app.
      3. Dial a phone number using the keypad.
      4. Verify that the keypad has braille numbers displayed next to the regular numbers, and that they are accurately displayed and legible for users who are visually impaired.
    1. Test Case 2: Braille alphabets
* Test Objective: To verify that the keypad has braille alphabets for accessibility.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Messaging" app.
      3. Compose a new message.
      4. Use the keypad to type a message.
      5. Verify that the keypad has braille alphabets displayed next to the regular alphabets, and that they are accurately displayed and legible for users who are visually impaired.
    1. Test Case 3: User Experience
* Test Objective: To verify that the user experience of using the braille keypad is satisfactory.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Notes" app.
      3. Create a new note.
      4. Use the keypad to type a long message using the braille numbers and alphabets.
      5. Verify that the user experience of using the braille keypad is satisfactory, with no difficulty in typing long messages, and that the keypad is comfortable for users who are visually impaired.

## The phone should be able to make and receive calls, with the ability to dial numbers or select contacts from a phone book.

* + 1. Test Case 1: Making a call by dialing a number
* Test Objective: To verify that the phone can make calls by dialing a number.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Phone" app.
      3. Dial a phone number using the keypad.
      4. Press the call button.
      5. Verify that the phone successfully makes the call, and that the call quality is satisfactory.
    1. Test Case 2: Making a call by selecting a contact from the phone book
* Test Objective: To verify that the phone can make calls by selecting a contact from the phone book.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Phone" app.
      3. Select the phone book option.
      4. Choose a contact from the list.
      5. Press the call button.
      6. Verify that the phone successfully makes the call to the selected contact, and that the call quality is satisfactory.
    1. Test Case 3: Receiving a call
* Test Objective: To verify that the phone can receive calls.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Wait for an incoming call.
      3. Answer the call.
      4. Verify that the call quality is satisfactory.
      5. End the call.
      6. Verify that the call was ended successfully.

## The phone should be able to send and receive SMS messages, with the ability to create and store drafts and view message history.

* + 1. Test Case 1: Sending an SMS message
* Test Objective: To verify that the phone can send SMS messages.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Messaging" app.
      3. Compose a new message.
      4. Enter the recipient's phone number or select a contact from the phone book.
      5. Type the message.
      6. Press the send button.
      7. Verify that the message is sent successfully, and that the recipient receives the message.
    1. Test Case 2: Receiving an SMS message
* Test Objective: To verify that the phone can receive SMS messages.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Wait for an incoming SMS message.
      3. Read the message.
      4. Verify that the message content is displayed correctly.
      5. Reply to the message if necessary.
      6. Verify that the reply is sent successfully.
    1. Test Case 3: Drafting and storing an SMS message
* Test Objective: To verify that the phone can create and store draft SMS messages.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the "Messaging" app.
      3. Compose a new message, but do not send it.
      4. Save the message as a draft.
      5. Navigate away from the messaging app.
      6. Return to the messaging app and open the draft.
      7. Verify that the draft is displayed correctly, with the correct recipient and message content.
      8. Make any necessary changes to the draft.
      9. Send the message.
      10. Verify that the message is sent successfully, and that the recipient receives the message.

## The user interface should have a clear display screen that shows important information such as battery life, signal strength, and time.

* + 1. Test Case 1: Displaying battery life information
* Test Objective: To verify that the phone's display screen accurately shows battery life information.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Verify that the phone's battery is fully charged.
      3. Use the phone normally for a period of time.
      4. Check the display screen to verify that the battery life information is accurate.
      5. Drain the battery to a low level.
      6. Check the display screen again to verify that the battery life information is accurate.
    1. Test Case 2: Displaying signal strength information
* Test Objective: To verify that the phone's display screen accurately shows signal strength information.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Verify that the phone has a strong signal.
      3. Move to an area with weaker signal strength.
      4. Check the display screen to verify that the signal strength

information is accurate.

* + - 1. Move back to an area with strong signal strength.
      2. Check the display screen again to verify that the signal strength information is accurate.
    1. Test Case 3: Displaying time information
* Test Objective: To verify that the phone's display screen accurately shows time information.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Verify that the phone's clock is synchronized with the correct time.
      3. Use the phone normally for a period of time.
      4. Check the display screen to verify that the time information is accurate.
      5. Change the phone's time settings to an incorrect time.
      6. Check the display screen again to verify that the time information is accurate after the time settings have been corrected.

## The phone should have a speaker and microphone that allow for clear communication during phone calls.

* + 1. Test Case 1: Testing speaker quality
* Test Objective: To verify that the phone's speaker allows for clear communication during phone calls.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Place a call to another phone.
      3. Speak normally into the phone's microphone and listen to the other phone's response through the phone's speaker.
      4. Verify that the speaker's sound quality is clear and loud enough to hear the other person's voice without difficulty.
      5. Increase the volume to the maximum level and verify that the sound quality remains clear.
    1. Test Case 2: Testing microphone quality
* Test Objective: To verify that the phone's microphone allows for clear communication during phone calls.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Receive a call from another phone.
      3. Speak normally into the phone's microphone and listen to the other person's response through the other phone's speaker.
      4. Verify that the microphone's sound quality is clear and loud enough for the other person to hear without difficulty.
      5. Move the phone around while speaking and verify that the microphone's sound quality remains clear.
    1. Test Case 3: Testing call clarity in noisy environments
* Test Objective: To verify that the phone's speaker and microphone allow

for clear communication during phone calls in noisy environments.

* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Place a call to another phone while in a noisy environment, such as a busy street or a crowded room.
      3. Speak normally into the phone's microphone and listen to the other person's response through the phone's speaker.
      4. Verify that the speaker's sound quality is clear and loud enough to hear the other person's voice over the background noise.
      5. Verify that the microphone's sound quality is clear enough for the other person to hear your voice over the background noise.

## The user interface should have a way to adjust volume and other audio settings.

* + 1. Test Case 1: Testing volume adjustment
* Test Objective: To verify that the user interface allows the user to adjust the volume of the phone's audio output.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Play an audio file, such as a music track or a recorded voice message.
      3. Attempt to adjust the volume using the volume buttons on the phone.
      4. Verify that the volume of the audio output changes accordingly and that the user can adjust the volume to a comfortable level.
    1. Test Case 2: Testing audio settings
* Test Objective: To verify that the user interface allows the user to adjust other audio settings, such as bass, treble, and balance.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Play an audio file, such as a music track or a recorded voice message.
      3. Navigate to the audio settings menu on the phone.
      4. Attempt to adjust the bass, treble, and balance settings using the phone's controls.
      5. Verify that the audio settings change accordingly and that the user can adjust the settings to their desired level.
    1. Test Case 3: Testing headphone jack output
* Test Objective: To verify that the phone's headphone jack output is functional and can be adjusted using the user interface.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Connect a pair of headphones to the phone's headphone jack.
      3. Play an audio file, such as a music track or a recorded voice message.
      4. Attempt to adjust the volume and other audio settings using the phone's controls while the headphones are plugged in.
      5. Verify that the audio output is routed to the headphones and that the user can adjust the volume and audio settings while the headphones are plugged in.

## The phone should have a way to lock and unlock the screen to prevent accidental button presses.

* + 1. Test Case 1: Testing screen lock feature
* Test Objective: To verify that the phone's screen can be locked to prevent accidental button presses.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Navigate to the settings menu on the phone.
      3. Locate the screen lock feature and activate it.
      4. Attempt to press any button on the phone to see if the screen is locked and no buttons are responsive.
      5. Verify that the screen is locked and no buttons are responsive.
    1. Test Case 2: Testing screen unlock feature
* Test Objective: To verify that the phone's screen can be unlocked when the user needs to use it.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Activate the screen lock feature as described in Test Case 1.
      3. Attempt to unlock the screen by entering the correct code or pattern.
      4. Verify that the screen is unlocked and that the user can access the phone's features.
    1. Test Case 3: Testing accidental button press prevention
* Test Objective: To verify that the screen lock feature effectively prevents accidental button presses.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Activate the screen lock feature as described in Test Case 1.
      3. Attempt to press any button on the phone while the screen is locked.
      4. Verify that no buttons are responsive and that the screen remains locked.
      5. Attempt to unlock the screen and verify that the phone's features are accessible once again.

## The user interface should have an easy-to-use power management system that allows the user to turn the phone on and off, as well as charge the battery.

* + 1. Test Case 1: Testing power on/off feature
* Test Objective: To verify that the phone can be turned on and off using the

power management system.

* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Verify that the phone is powered on and operational.
      3. Locate the power management system on the phone.
      4. Turn off the phone using the power management system.
      5. Verify that the phone has powered off and is no longer operational.
      6. Turn the phone back on using the power management system.
      7. Verify that the phone has powered on and is operational once again.
    1. Test Case 2: Testing battery charging feature
* Test Objective: To verify that the battery can be charged using the power management system.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Verify that the phone's battery is not fully charged.
      3. Connect the phone to a power source using the charging cable.
      4. Verify that the phone's battery is charging and the battery icon on the display screen shows the charging progress.
      5. Disconnect the phone from the power source once the battery is fully charged.
      6. Verify that the battery icon on the display screen shows that the battery is fully charged.
    1. Test Case 3: Testing low battery warning feature
* Test Objective: To verify that the phone provides a low battery warning to the user.
* Test Steps:
  + - 1. Turn on the Arduino-based mobile phone.
      2. Allow the phone's battery to drain to a low level.
      3. Verify that the phone displays a low battery warning on the display screen and provides an audible alert.
      4. Connect the phone to a power source using the charging cable.
      5. Verify that the low battery warning is no longer displayed once the battery starts charging.

## The GSM module should be compatible with the mobile network in the user's region.

* + 1. Test case 1: Verify that the GSM module can connect to the mobile network in the user's region.
* Test steps:
  + - 1. Insert a SIM card from a mobile network operator in the user's region.
      2. Power on the Arduino based mobile phone.
      3. Verify that the GSM module connects to the mobile network successfully.
      4. Check the signal strength of the mobile network and ensure it is

strong enough for making calls and sending SMS messages.

* Expected result: The Arduino based mobile phone should be able to connect to the mobile network in the user's region without any issues.
  + 1. Test case 2: Verify that the GSM module can make and receive calls on the mobile network.
* Test steps:
  + - 1. Insert a SIM card from a mobile network operator in the user's region.
      2. Power on the Arduino based mobile phone.
      3. Dial a phone number and make a call.
      4. Verify that the call is connected and the audio quality is good.
      5. End the call.
      6. Receive a call and verify that the incoming call is displayed on the screen.
      7. Answer the call and verify that the audio quality is good.
      8. End the call.
* Expected result: The Arduino based mobile phone should be able to make and receive calls on the mobile network without any issues.
  + 1. Test case 3: Verify that the GSM module can send and receive SMS messages on the mobile network.
* Test steps:
  + - 1. Insert a SIM card from a mobile network operator in the user's region.
      2. Power on the Arduino based mobile phone.
      3. Compose an SMS message and send it to a phone number.
      4. Verify that the SMS message is sent successfully and received by the recipient.
      5. Receive an SMS message and verify that the message is displayed on the screen.
* Expected result: The Arduino based mobile phone should be able to send and receive SMS messages on the mobile network without any issues.

## The communication system should be able to send and receive SMS messages, with the ability to create and store drafts and view message history.

* + 1. Test case 1: Verify that the communication system can send and receive SMS messages.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Compose an SMS message and send it to a phone number.
      3. Verify that the SMS message is sent successfully and received by the recipient.
      4. Receive an SMS message and verify that the message is displayed on the screen.
* Expected result: The communication system of the Arduino based mobile

phone should be able to send and receive SMS messages without any issues.

* + 1. Test case 2: Verify that the communication system can create and store drafts of SMS messages.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Compose an SMS message but do not send it.
      3. Save the message as a draft.
      4. Open the drafts folder and verify that the draft is stored there.
      5. Open the draft and verify that the message content is correct.
* Expected result: The communication system of the Arduino based mobile phone should be able to create and store drafts of SMS messages without any issues.
  + 1. Test case 3: Verify that the communication system can view message history.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Open the message history folder.
      3. Verify that all previously sent and received SMS messages are displayed there.
      4. Select a message from the history and verify that the message content is correct.
* Expected result: The communication system of the Arduino based mobile phone should be able to view message history without any issues, displaying all previously sent and received SMS messages.

## The communication system should be able to make and receive phone calls, with the ability to dial numbers or select contacts from a phone book.

* + 1. Test case 1: Verify that the communication system can make and receive phone calls.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Dial a phone number and make a call.
      3. Verify that the call is connected and the audio quality is good.
      4. End the call.
      5. Receive a call and verify that the incoming call is displayed on the screen.
      6. Answer the call and verify that the audio quality is good.
      7. End the call.
* Expected result: The communication system of the Arduino based mobile phone should be able to make and receive phone calls without any issues.
  + 1. Test case 2: Verify that the communication system can dial phone numbers.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Open the dialer app.
      3. Enter a phone number and dial it.
      4. Verify that the call is connected and the audio quality is good.
      5. End the call.
* Expected result: The communication system of the Arduino based mobile phone should be able to dial phone numbers without any issues.
  + 1. Test case 3: Verify that the communication system can select contacts from a phone book.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Open the phone book app.
      3. Select a contact from the list.
      4. Dial the contact's phone number.
      5. Verify that the call is connected and the audio quality is good.
      6. End the call.
* Expected result: The communication system of the Arduino based mobile phone should be able to select contacts from a phone book and dial their phone numbers without any issues.

## The GSM module should have a strong and reliable signal to ensure clear communication.

* + 1. Test case 1: Verify that the GSM module has a strong and reliable signal in areas with good network coverage.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Verify that the signal strength indicator on the screen shows a strong signal.
      3. Make a call and verify that the audio quality is good and the call does not drop.
* Expected result: The GSM module of the Arduino based mobile phone should have a strong and reliable signal in areas with good network coverage, ensuring clear communication.
  + 1. Test case 2: Verify that the GSM module can maintain a stable signal in areas with weak network coverage.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Move to an area with weak network coverage.
      3. Verify that the signal strength indicator on the screen shows a weak signal.
      4. Make a call and verify that the audio quality is still good and the call does not drop.
* Expected result: The GSM module of the Arduino based mobile phone should be able to maintain a stable signal in areas with weak network coverage, ensuring clear communication.
  + 1. Test case 3: Verify that the GSM module can quickly regain a strong signal after losing it.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Move to an area with no network coverage.
      3. Verify that the signal strength indicator on the screen shows no signal.
      4. Move back to an area with good network coverage.
      5. Verify that the signal strength indicator on the screen shows a strong signal.
      6. Make a call and verify that the audio quality is good and the call does not drop.
* Expected result: The GSM module of the Arduino based mobile phone should be able to quickly regain a strong signal after losing it, ensuring clear communication.

## The communication system should have a clear and easy-to-use interface that shows important information such as battery life, signal strength, and time.

* + 1. Test case 1: Verify that the interface displays battery life accurately.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Verify that the battery indicator on the screen shows full battery life.
      3. Use the phone for a period of time.
      4. Verify that the battery indicator on the screen shows an accurate representation of the remaining battery life.
* Expected result: The interface of the Arduino based mobile phone should display battery life accurately and update in real-time as the battery is consumed.
  + 1. Test case 2: Verify that the interface displays signal strength accurately.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Verify that the signal strength indicator on the screen shows a strong signal.
      3. Move to an area with weak network coverage.
      4. Verify that the signal strength indicator on the screen accurately reflects the weak signal.
* Expected result: The interface of the Arduino based mobile phone should display signal strength accurately and update in real-time as the signal changes.
  + 1. Test case 3: Verify that the interface displays the time and date accurately.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Verify that the interface displays the correct time and date.
      3. Wait for a period of time.
      4. Verify that the time and date displayed on the interface have updated accurately.
* Expected result: The interface of the Arduino based mobile phone should display the time and date accurately and update in real-time. The interface should also be easy to read and understand, with important information prominently displayed.

## The system should have a way to adjust volume and other audio settings to ensure clear communication during phone calls.

* + 1. Test case 1: Verify that the volume can be adjusted during a phone call.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make a phone call.
      3. While on the call, adjust the volume using the volume buttons.
      4. Verify that the audio level changes accordingly.
* Expected result: The system of the Arduino based mobile phone should allow the user to adjust the volume during a phone call to ensure clear communication.
  + 1. Test case 2: Verify that audio settings can be adjusted before making a phone call.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Open the settings menu.
      3. Navigate to the audio settings.
      4. Adjust the audio settings as desired.
      5. Make a phone call.
      6. Verify that the audio settings are applied and the audio quality is good.
* Expected result: The system of the Arduino based mobile phone should allow the user to adjust audio settings before making a phone call to ensure clear communication.
  + 1. Test case 3: Verify that audio settings are saved and applied for future phone calls.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Open the settings menu.
      3. Navigate to the audio settings.
      4. Adjust the audio settings as desired.
      5. Make a phone call.
      6. Hang up the phone.
      7. Make another phone call.
      8. Verify that the audio settings are applied and the audio quality is good.
* Expected result: The system of the Arduino based mobile phone should

save the audio settings adjusted by the user and apply them for future phone calls to ensure clear communication.

## The communication system should have a way to switch between the phone's speaker and a connected headset or earphones.

* + 1. Test case 1: Verify that the phone's speaker works during a phone call.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make a phone call.
      3. During the phone call, switch the audio output to the phone's speaker.
      4. Verify that the audio comes out of the speaker and is clear.
* Expected result: The communication system of the Arduino based mobile phone should allow the user to switch between the phone's speaker and other audio output devices and the speaker should work correctly.
  + 1. Test case 2: Verify that a headset can be connected to the phone.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Connect a compatible headset to the phone's audio jack or via Bluetooth.
      3. Make a phone call.
      4. During the phone call, switch the audio output to the connected headset.
      5. Verify that the audio comes out of the headset and is clear.
* Expected result: The communication system of the Arduino based mobile phone should allow the user to connect a compatible headset and switch between the headset and other audio output devices, and the headset should work correctly.
  + 1. Test case 3: Verify that the phone remembers the audio output setting for future use.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Connect a compatible headset to the phone's audio jack or via Bluetooth.
      3. Make a phone call and switch the audio output to the headset.
      4. End the phone call and disconnect the headset.
      5. Make another phone call.
      6. Verify that the phone remembers the audio output setting from the previous phone call and applies it to the current call.
* Expected result: The communication system of the Arduino based mobile phone should remember the audio output setting from the previous phone call and apply it to the current call if the same audio output device is used again.

## The system should be able to receive and display caller ID

**information for incoming calls.**

* + 1. Test case 1: Verify that the caller ID is displayed for an incoming call.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make sure that the SIM card is inserted and the phone is connected to the mobile network.
      3. Call the Arduino based mobile phone from another phone and verify that the caller ID information is displayed on the phone's screen.
* Expected result: The communication system of the Arduino based mobile phone should be able to receive and display the caller ID information for incoming calls.
  + 1. Test case 2: Verify that the caller ID information is displayed correctly.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make sure that the SIM card is inserted and the phone is connected to the mobile network.
      3. Call the Arduino based mobile phone from another phone with a known phone number.
      4. Verify that the caller ID information displayed on the phone's screen matches the known phone number.
* Expected result: The communication system of the Arduino based mobile phone should display the correct caller ID information for incoming calls.
  + 1. Test case 3: Verify that the phone remembers the caller ID information for future use.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make sure that the SIM card is inserted and the phone is connected to the mobile network.
      3. Receive an incoming call and verify that the caller ID information is displayed on the phone's screen.
      4. End the call and wait for some time.
      5. Make another call from the same phone number as before.
      6. Verify that the caller ID information displayed on the phone's screen is the same as before.
* Expected result: The communication system of the Arduino based mobile phone should remember the caller ID information from previous calls and display it for future calls from the same phone number.

## The GSM module should be able to handle call forwarding, call waiting, and other standard phone features.

* + 1. Test case 1: Verify that call forwarding works as expected.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make sure that the SIM card is inserted and the phone is connected to the mobile network.
      3. Enable call forwarding on the phone by dialing the appropriate code or accessing the phone's settings.
      4. Make a call to the Arduino based mobile phone from another phone and verify that the call is forwarded to the designated number.
* Expected result: The communication system of the Arduino based mobile phone should be able to handle call forwarding and forward incoming calls to the designated number.
  + 1. Test case 2: Verify that call waiting works as expected.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make sure that the SIM card is inserted and the phone is connected to the mobile network.
      3. Make a call to the Arduino based mobile phone from another phone.
      4. While the call is in progress, make another call to the same phone.
      5. Verify that the Arduino based mobile phone alerts the user about the second call and provides options to either end the first call and answer the second call or put the first call on hold and answer the second call.
* Expected result: The communication system of the Arduino based mobile phone should be able to handle call waiting and alert the user about incoming calls while on another call.
  + 1. Test case 3: Verify that other standard phone features work as expected.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make sure that the SIM card is inserted and the phone is connected to the mobile network.
      3. Access the phone's settings or dial the appropriate code to access the list of standard phone features.
      4. Verify that the phone features such as call barring, call blocking, call waiting, call forwarding, and conference calling are available and can be configured as expected.
* Expected result: The communication system of the Arduino based mobile phone should be able to handle standard phone features and provide options to configure these features as expected.

## The communication system should have a way to block unwanted calls and SMS messages.

* + 1. Test case 1: Verify that the phone can block specific phone numbers.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make sure that the SIM card is inserted and the phone is connected to the mobile network.
      3. Access the phone's settings or dial the appropriate code to access the call blocking feature.
      4. Add a specific phone number to the list of blocked numbers.
      5. Ask someone to call the Arduino based mobile phone from the blocked number and verify that the call is blocked and not received by the phone.
* Expected result: The communication system of the Arduino based mobile phone should be able to block specific phone numbers as configured.
  + 1. Test case 2: Verify that the phone can block SMS messages from specific phone numbers or keywords.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make sure that the SIM card is inserted and the phone is connected to the mobile network.
      3. Access the phone's settings or dial the appropriate code to access the SMS blocking feature.
      4. Add a specific phone number or keyword to the list of blocked messages.
      5. Ask someone to send an SMS message to the Arduino based mobile phone from the blocked number or with the blocked keyword and verify that the message is blocked and not received by the phone.
* Expected result: The communication system of the Arduino based mobile phone should be able to block SMS messages from specific phone numbers or keywords as configured.
  + 1. Test case 3: Verify that the phone can unblock previously blocked phone numbers and SMS messages.
* Test steps:
  + - 1. Power on the Arduino based mobile phone.
      2. Make sure that the SIM card is inserted and the phone is connected to the mobile network.
      3. Access the phone's settings or dial the appropriate code to access the call or SMS blocking feature.
      4. Remove a previously blocked phone number or keyword from the list of blocked numbers or messages.
      5. Ask someone to call or send an SMS message to the Arduino based mobile phone from the previously blocked number or with the previously blocked keyword and verify that the call or message is received by the phone.
* Expected result: The communication system of the Arduino based mobile phone should be able to unblock previously blocked phone numbers and SMS messages as configured.

## The audio system should have a high-quality speaker that produces clear and loud sound.

* + 1. Test case: Sound clarity Input: Audio file with a mix of low and high- frequency sounds. Output: Verify the sound produced by the high-quality speaker is clear and there is no distortion.
    2. Test case: Volume Input: Audio file with low volume output: Verify that the speaker produces loud sound and is able to play the audio file at a comfortable listening level.
    3. Test case: Bass response Input: Audio file with bass-heavy music. Output: Verify that the speaker produces a good bass response and there is no loss of clarity in the audio output.

## The audio system should have a sensitive and accurate microphone that captures clear voice recordings during phone calls.

* + 1. Test case: Voice clarity Input: Make a phone call and record the voice at different distances. Output: Verify that the microphone is sensitive enough to capture clear voice recordings, even at a distance.
    2. Test case: Background noise cancellation Input: Make a phone call in a noisy environment. Output: Verify that the microphone is able to filter out background noise and capture clear voice recordings.
    3. Test case: Accuracy Input: Record a phone call and play it back. Output: Verify that the voice recording is accurate and there are no issues with the audio quality.

## The system should be able to adjust volume and other audio settings to ensure optimal sound quality.

* + 1. Test case: Volume adjustment Input: Audio file with a range of volume levels. Output: Verify that the system is able to adjust the volume level to ensure optimal sound quality without any distortion or clipping.
    2. Test case: Audio settings adjustment Input: Play an audio file with different audio settings. Output: Verify that the system is able to adjust the audio settings such as bass, treble, and equalization to ensure optimal sound quality.
    3. Test case: Dynamic adjustment Input: Play an audio file that changes dynamically in volume and other audio settings. Output: Verify that the system is able to dynamically adjust the volume and other audio settings to ensure optimal sound quality at all times.

## The speaker should be able to play audio from the phone's media library, such as music and videos.

* + 1. Test case: Audio file compatibility Input: Play an audio file from the phone's media library. Output: Verify that the speaker is able to play audio files in different formats such as MP3, WAV, and FLAC.
    2. Test case: Playback control Input: Play an audio file and pause, resume, and stop the playback. Output: Verify that the speaker responds correctly to playback control commands from the phone's media player.
    3. Test case: Playlist support Input: Create a playlist in the phone's media library and play it through the speaker. Output: Verify that the speaker is able to play audio files in a playlist in the correct order without any issues.

## The microphone should be able to record voice calls and other audio input with good quality.

* + 1. Test case: Voice call recording Input: Make a phone call and record the conversation. Output: Verify that the microphone is able to record voice calls with good quality and clarity.
    2. Test case: Audio input recording Input: Record audio input from different sources such as the phone's built-in microphone, an external microphone, or a headset. Output: Verify that the microphone is able to record audio input with good quality and clarity, regardless of the source.
    3. Test case: Noise reduction Input: Record audio in a noisy environment. Output: Verify that the microphone is able to filter out background noise and record audio input with good quality and clarity.

## The audio system should be able to switch between the phone's speaker and a connected headset or earphones.

* + 1. Test case: Speaker output Input: Play an audio file through the phone's speaker. Output: Verify that the audio system is able to play audio through the phone's speaker without any issues.
    2. Test case: Headset output Input: Connect a headset or earphones to the phone and play an audio file. Output: Verify that the audio system is able to play audio through the headset or earphones without any issues.
    3. Test case: Output switching Input: Play an audio file through the phone's speaker and then connect a headset or earphones while the audio is playing. Output: Verify that the audio system is able to switch the audio output from the phone's speaker to the connected headset or earphones seamlessly without any interruption in the audio playback.

## The system should be able to receive and play sound notifications for incoming calls and SMS messages.

* + 1. Test case: Call notification sound Input: Receive an incoming call and check the notification sound. Output: Verify that the system is able to receive and play a sound notification for an incoming call.
    2. Test case: SMS notification sound Input: Receive an SMS message and check the notification sound. Output: Verify that the system is able to receive and play a sound notification for an incoming SMS message.
    3. Test case: Sound volume Input: Receive an incoming call or SMS message while playing music or other audio. Output: Verify that the sound notification is played at an appropriate volume that is loud enough to be heard but not too loud as to interfere with the currently playing audio.

## The audio system should have a noise-cancellation feature that filters out background noise during phone calls.

* + 1. Test case: Quiet environment Input: Make a phone call in a quiet environment. Output: Verify that the audio system is not filtering out any sound and the conversation is clear.
    2. Test case: Noisy environment Input: Make a phone call in a noisy environment, such as a busy street or a construction site. Output: Verify that the audio system is able to filter out background noise and the conversation is clear.
    3. Test case: Background noise test Input: Play a recorded audio file with background noise and make a phone call while the audio is playing. Output: Verify that the audio system is able to filter out the background noise from the audio file and the conversation is clear.

## The system should be able to record and save voice memos or notes.

* + 1. Test case: Record and playback Input: Record a voice memo and play it back. Output: Verify that the system is able to record and play back voice memos or notes.
    2. Test case: Save and retrieve Input: Save a voice memo and retrieve it later. Output: Verify that the system is able to save and retrieve voice memos or notes without any loss of data or quality.
    3. Test case: File format Input: Record a voice memo and check the file format. Output: Verify that the system is able to save voice memos or notes in a common file format such as MP3 or WAV for easy sharing and playback on other devices.

## The audio system should have a clear and easy-to-use interface that shows important information such as battery life, volume level, and audio source.

* + 1. Test case: Interface display Input: Turn on the phone and check the audio system interface. Output: Verify that the audio system interface is clear and easy-to-use, and displays important information such as battery life, volume level, and audio source.
    2. Test case: Volume adjustment Input: Adjust the volume level using the audio system interface. Output: Verify that the audio system interface accurately displays the current volume level and allows for easy adjustment.
    3. Test case: Audio source selection Input: Select a different audio source such as a connected headset or earphones using the audio system interface. Output: Verify that the audio system interface accurately displays the selected audio source and allows for easy switching between sources.

## The power module should be able to monitor the battery level and provide accurate battery status information to the user.

Test Case 1: Input:

* + - Connect the Arduino based mobile phone to a fully charged battery. Output:
    - The power module should display the battery level as 100% and provide accurate battery status information to the user.
    - Verify that the displayed battery level is consistent with the actual battery level.

Test Case 2: Input:

* + - Connect the Arduino based mobile phone to a partially charged battery. Output:
    - The power module should display the battery level as a percentage of the remaining charge and provide accurate battery status information to the user.
    - Verify that the displayed battery level is consistent with the actual battery level.

Test Case 3: Input:

* + - Connect the Arduino based mobile phone to a low battery. Output:
    - The power module should display a warning message indicating that the battery level is low and provide accurate battery status information to the user.
    - Verify that the displayed battery level is consistent with the actual battery level.
    - Verify that the warning message is displayed when the battery level reaches a certain threshold.

## The module should be able to charge the battery efficiently and quickly.

Test Case 1: Input:

* + - Connect the Arduino based mobile phone to a dead battery.
    - Start the charging process.

Output:

* + - The power module should start charging the battery.
    - Verify that the battery is charging at an efficient and quick rate.
    - Verify that the battery is not overheating during the charging process.

Test Case 2: Input:

* + - Connect the Arduino based mobile phone to a partially charged battery.
    - Start the charging process. Output:
    - The power module should start charging the battery.
    - Verify that the battery is charging at an efficient and quick rate.
    - Verify that the battery is not overheating during the charging process.

Test Case 3: Input:

* + - Connect the Arduino based mobile phone to a fully charged battery.
    - Start the charging process. Output:
    - The power module should not start charging the battery as it is already fully charged.
    - Verify that the power module detects the fully charged state of the battery.
    - Verify that the power module does not waste energy by attempting to charge a fully charged battery.

## The power module should have overcharge protection to prevent damage to the battery.

Test Case 1: Input:

* + - Connect the Arduino based mobile phone to a charger.
    - Fully charge the battery and keep the phone connected to the charger. Output:
    - The power module should detect that the battery is fully charged and stop charging it.
    - Verify that the power module provides overcharge protection to prevent damage to the battery.
    - Verify that the battery does not overheat or become damaged due to overcharging.

Test Case 2: Input:

* + - Connect the Arduino based mobile phone to a charger.
    - Partially charge the battery and keep the phone connected to the charger. Output:
    - The power module should detect that the battery is partially charged and start charging it.
    - Verify that the power module provides overcharge protection to prevent damage to the battery.
    - Verify that the battery does not overheat or become damaged due to

overcharging.

Test Case 3: Input:

* + - Connect the Arduino based mobile phone to a charger.
    - Disconnect the charger before the battery is fully charged.
    - Reconnect the charger after a few minutes. Output:
    - The power module should detect that the battery is partially charged and start charging it again.
    - Verify that the power module provides overcharge protection to prevent damage to the battery.
    - Verify that the battery does not overheat or become damaged due to overcharging.

## The module should have over-discharge protection to prevent damage to the battery and ensure a longer lifespan.

Test Case 1: Input:

* + - Use the Arduino based mobile phone until the battery level drops to a very low level.
    - Keep using the phone until the battery level reaches the critical level and the phone turns off.

Output:

* + - The power module should prevent the battery from discharging below a safe level to prevent damage to the battery and ensure a longer lifespan.
    - Verify that the phone shuts down before the battery level drops to a critical level.

Test Case 2: Input:

* + - Connect the Arduino based mobile phone to a charger.
    - Start the charging process.
    - Disconnect the charger before the battery is fully charged. Output:
    - The power module should detect that the battery level is low and prevent the battery from discharging below a safe level to prevent damage to the battery and ensure a longer lifespan.
    - Verify that the power module provides over-discharge protection to prevent damage to the battery.

Test Case 3: Input:

* + - Use the Arduino based mobile phone until the battery level drops to a low level.
    - Keep using the phone until the battery level reaches the critical level and the phone turns off.
    - Connect the phone to a charger.

Output:

* + - The power module should detect that the battery level is low and prevent the battery from discharging below a safe level to prevent damage to the battery and ensure a longer lifespan.
    - Verify that the power module provides over-discharge protection to prevent damage to the battery.
    - Verify that the phone starts charging once connected to the charger.

## The power module should be able to communicate with the phone's software to provide accurate battery level information to the user.

Test Case 1: Input:

* + - Connect the Arduino based mobile phone to a computer via USB cable.
    - Check the battery level on the computer's operating system. Output:
    - The power module should communicate with the phone's software and provide accurate battery level information to the user.
    - Verify that the displayed battery level on the computer is consistent with the actual battery level of the phone.

Test Case 2: Input:

* + - Install a battery monitoring app on the Arduino based mobile phone.
    - Check the battery level displayed by the app. Output:
    - The power module should communicate with the phone's software and provide accurate battery level information to the user.
    - Verify that the displayed battery level on the app is consistent with the actual battery level of the phone.

Test Case 3: Input:

* + - Use the Arduino based mobile phone until the battery level drops to a low level.
    - Connect the phone to a charger.
    - Monitor the battery level displayed by the phone's software while the phone is charging.

Output:

* + - The power module should communicate with the phone's software and provide accurate battery level information to the user.
    - Verify that the displayed battery level on the phone's software is consistent with the actual battery level of the phone.
    - Verify that the displayed battery level increases as the phone charges.

## The module should have a power-saving mode that helps to extend the battery life when the phone is not in use.

Test Case 1: Input:

* + - Turn on the power-saving mode on the Arduino based mobile phone.
    - Let the phone idle for a period of time. Output:
    - The power module should reduce power consumption when the phone is not in use to extend the battery life.
    - Verify that the battery level does not decrease significantly while the phone is idle with power-saving mode turned on.

Test Case 2: Input:

* + - Turn on the power-saving mode on the Arduino based mobile phone.
    - Use the phone normally for a period of time. Output:
    - The power module should adjust power consumption to save energy and extend the battery life.
    - Verify that the battery level does not decrease as rapidly as it would without power-saving mode turned on.

Test Case 3: Input:

* + - Turn on the power-saving mode on the Arduino based mobile phone.
    - Use the phone heavily, running multiple apps and services. Output:
    - The power module should adjust power consumption to save energy and extend the battery life.
    - Verify that the battery level does not decrease as rapidly as it would without power-saving mode turned on, even under heavy use.

## The module should have a low-battery warning that alerts the user when the battery level is low.

Test Case 1: Input:

* + - Use the Arduino based mobile phone until the battery level drops to a low level.

Output:

* + - The power module should detect the low battery level and send a warning signal to the phone's software.
    - Verify that the phone displays a low battery warning message to the user.

Test Case 2: Input:

* + - Connect the Arduino based mobile phone to a charger and start the charging process.
    - Disconnect the charger before the battery is fully charged. Output:
    - The power module should detect that the battery level is low and send a warning signal to the phone's software.
    - Verify that the phone displays a low battery warning message to the user.

Test Case 3: Input:

* + - Use the Arduino based mobile phone until the battery level drops to a low level.
    - Keep using the phone until the battery level reaches a critical level and the phone turns off.
    - Connect the phone to a charger and start the charging process. Output:
    - The power module should detect that the battery level is low and send a warning signal to the phone's software.
    - Verify that the phone displays a low battery warning message to the user.
    - Verify that the phone starts charging once connected to the charger.

## The power module should be able to charge the phone while it is in use.

Test Case 1: Input:

* + - Use the Arduino based mobile phone normally while it is connected to a charger.

Output:

* + - The power module should be able to charge the phone's battery while it is in use.
    - Verify that the battery level of the phone increases while the phone is being used and connected to the charger.

Test Case 2: Input:

* + - Use the Arduino based mobile phone heavily, running multiple apps and services, while it is connected to a charger.

Output:

* + - The power module should be able to charge the phone's battery while it is in use, even under heavy use.
    - Verify that the battery level of the phone increases while the phone is being used heavily and connected to the charger.

Test Case 3: Input:

* + - Use the Arduino based mobile phone to perform a resource-intensive task, such as playing a high-performance game, while it is connected to a charger.

Output:

* + - The power module should be able to charge the phone's battery while it is in use, even during resource-intensive tasks.
    - Verify that the battery level of the phone increases while the phone is being used to perform a resource-intensive task and connected to the charger.

## The module should be able to support multiple charging methods, including USB charging and wireless charging.

Test Case 1: Input:

* + - Connect the Arduino based mobile phone to a computer via USB cable for charging.

Output:

* + - The power module should detect the USB charging method and charge the phone's battery.
    - Verify that the battery level of the phone increases while it is connected to the computer via USB cable.

Test Case 2: Input:

* + - Place the Arduino based mobile phone on a wireless charger for charging. Output:
    - The power module should detect the wireless charging method and charge the phone's battery.
    - Verify that the battery level of the phone increases while it is placed on the wireless charger.

Test Case 3: Input:

* + - Connect the Arduino based mobile phone to a power bank via USB cable for charging.
    - Place the power bank on a wireless charger for charging. Output:
    - The power module should detect both USB and wireless charging methods and charge the phone's battery.
    - Verify that the battery level of the phone increases while it is connected to the power bank via USB cable and placed on the wireless charger.

## The power module should have a clear and easy-to-use interface that shows important information such as battery level, charging status, and estimated battery life remaining.

Test Case 1: Input:

* + - Turn on the Arduino based mobile phone and access the power module interface.

Output:

* + - The power module interface should be clear and easy-to-use.
    - Verify that the interface displays important information such as battery level, charging status, and estimated battery life remaining.

Test Case 2: Input:

* + - Connect the Arduino based mobile phone to a charger.
    - Access the power module interface. Output:
    - The power module interface should display the charging status, such as "Charging" or "Fully Charged".
    - Verify that the interface updates in real-time to show the progress of the

charging process.

Test Case 3: Input:

* + - Use the Arduino based mobile phone until the battery level is low.
    - Access the power module interface. Output:
    - The power module interface should display the estimated battery life remaining.
    - Verify that the interface provides an accurate estimate of the remaining battery life based on the current usage of the phone.

## The Arduino code should be able to detect input from the 4\*4 matrix keypad and convert it into characters that can be used to make calls and send SMS messages.

* + 1. Test Case 1: Verify keypad input detection
* Input: Press various keys on the 4\*4 matrix keypad
* Expected output: Arduino code should detect the input from the keypad and convert it into characters. Verify if the output matches the expected output for each key press.
  + 1. Test Case 2: Verify call making functionality
* Input: Enter a phone number using the keypad and initiate a call
* Expected output: Arduino code should convert the keypad input into the corresponding phone number and initiate the call. Verify that the call is made to the correct number.
  + 1. Test Case 3: Verify SMS sending functionality
* Input: Enter a phone number and message using the keypad and send an SMS
* Expected output: Arduino code should convert the keypad input into the corresponding phone number and message, and send the SMS. Verify that the SMS is sent to the correct number with the correct message.

## The code should be able to recognize braille numbers and alphabets and convert them into standard alphanumeric characters.

* + 1. Test Case 1: Verify braille input recognition
* Input: Enter various braille numbers and alphabets using the braille input mechanism
* Expected output: Arduino code should recognize the braille input and convert it into the corresponding standard alphanumeric characters. Verify if the output matches the expected output for each input.
  + 1. Test Case 2: Verify alphanumeric output generation
* Input: Enter various braille numbers and alphabets using the braille input mechanism
* Expected output: Arduino code should convert the braille input into the corresponding standard alphanumeric characters. Verify that the output

generated is a standard alphanumeric character that is expected based on the braille input.

* + 1. Test Case 3: Verify full sentence translation
* Input: Enter a full sentence using the braille input mechanism
* Expected output: Arduino code should convert the entire sentence into standard alphanumeric characters. Verify that the output generated is a meaningful sentence with correct spelling and grammar.

## The code should be able to handle incoming and outgoing calls and SMS messages through the GSM module.

* + 1. Test Case 1: Verify incoming call handling
* Input: Simulate an incoming call using a test phone and send the call to the Arduino based mobile phone
* Expected output: Arduino code should detect the incoming call and allow the user to answer or reject it. Verify that the user is able to answer or reject the call correctly.
  + 1. Test Case 2: Verify outgoing call functionality
* Input: Enter a phone number using the keypad and initiate a call
* Expected output: Arduino code should initiate the call using the GSM module. Verify that the call is made to the correct number and the user is able to hear the audio.
  + 1. Test Case 3: Verify SMS sending and receiving functionality
* Input: Send an SMS message to the Arduino based mobile phone and verify the received message. Send an SMS message from the Arduino based mobile phone and verify that it is sent correctly.
* Expected output: Arduino code should detect incoming SMS messages and notify the user. Verify that the received message is displayed correctly. Arduino code should also send SMS messages using the GSM module. Verify that the sent message is delivered correctly to the recipient.

## The code should be optimized for low power consumption to ensure longer battery life.

* + 1. Test Case 1: Verify low power mode
* Input: Keep the Arduino based mobile phone idle for an extended period of time
* Expected output: Arduino code should put the device into a low power mode after a certain period of inactivity. Verify that the device consumes less power and battery life is extended during this low power mode.
  + 1. Test Case 2: Verify power consumption during normal use
* Input: Use the Arduino based mobile phone to make calls, send SMS messages, and use other features for an extended period of time
* Expected output: Arduino code should optimize power consumption during normal use to ensure longer battery life. Verify that the device is consuming less power than expected during normal use and battery life is extended.
  + 1. Test Case 3: Verify power consumption during charging
* Input: Charge the Arduino based mobile phone and measure the power consumption during charging
* Expected output: Arduino code should optimize power consumption during charging to ensure faster charging and longer battery life. Verify that the device is consuming less power during charging than expected and the battery is charged faster.

## The code should have a responsive and intuitive user interface that is easy to use for a blind person.

* + 1. Test Case 1: Verify keypad input recognition for visually challenged users
* Input: Simulate a visually challenged user and enter various numbers and alphabets using the keypad
* Expected output: Arduino code should recognize the keypad input and provide appropriate feedback to the user. Verify that the user is able to enter numbers and alphabets accurately and quickly.
  + 1. Test Case 2: Verify voice feedback for visually challenged users
* Input: Simulate a visually challenged user and use various features of the Arduino based mobile phone
* Expected output: Arduino code should provide voice feedback to the user for various features, such as making a call, sending an SMS, and checking the battery level. Verify that the voice feedback is clear and easy to understand.
  + 1. Test Case 3: Verify ease of use for visually challenged users
* Input: Provide the Arduino based mobile phone to a visually challenged user and observe the usage
* Expected output: Arduino code should provide an intuitive user interface that is easy to use for visually challenged users. Verify that the user is able to use various features of the Arduino based mobile phone without any difficulty or confusion.

## The code should be able to read out incoming SMS messages and provide voice prompts to the user for sending and composing SMS messages.

* + 1. Test Case 1: Verify incoming SMS reading functionality
* Input: Simulate an incoming SMS message and send it to the Arduino based mobile phone
* Expected output: Arduino code should detect the incoming SMS message and read it out loud to the user. Verify that the message is read accurately and clearly.
  + 1. Test Case 2: Verify voice prompts for sending SMS messages
* Input: Use the Arduino based mobile phone to send an SMS message
* Expected output: Arduino code should provide voice prompts to the user for composing and sending the SMS message. Verify that the prompts are clear and easy to understand, and that the user is able to compose and send the message correctly.
  + 1. Test Case 3: Verify voice prompts for composing SMS messages
* Input: Use the Arduino based mobile phone to compose an SMS message
* Expected output: Arduino code should provide voice prompts to the user for composing the SMS message, such as reading back the message, editing the message, and adding recipients. Verify that the prompts are clear and easy to understand, and that the user is able to compose the message correctly.

## The code should be able to provide haptic feedback on the hand glove to help the user navigate through the menu system and to provide feedback on incoming calls and SMS messages.

* + 1. Test Case 1: Verify haptic feedback for menu navigation
* Input: Navigate through the menu system using the hand glove
* Expected output: Arduino code should provide haptic feedback on the hand glove to help the user navigate through the menu system. Verify that the feedback is clear and easy to understand, and that the user is able to navigate through the menu system correctly.
  + 1. Test Case 2: Verify haptic feedback for incoming calls
* Input: Simulate an incoming call and receive it using the hand glove
* Expected output: Arduino code should provide haptic feedback on the hand glove to indicate the incoming call. Verify that the feedback is clear and easy to understand, and that the user is able to answer the call correctly.
  + 1. Test Case 3: Verify haptic feedback for incoming SMS messages
* Input: Simulate an incoming SMS message and receive it using the hand glove
* Expected output: Arduino code should provide haptic feedback on the hand glove to indicate the incoming SMS message. Verify that the feedback is clear and easy to understand, and that the user is able to read the message correctly.

## The code should have a feature to enable the user to manage their contact list using the keypad.

* + 1. Test Case 1: Verify contact list management feature
* Input: Use the keypad to manage the contact list, such as adding, deleting, and editing contacts
* Expected output: Arduino code should provide a user-friendly interface on the keypad for managing the contact list. Verify that the user is able to add, delete, and edit contacts accurately and quickly.
  + 1. Test Case 2: Verify contact list retrieval feature
* Input: Use the keypad to retrieve a contact from the contact list
* Expected output: Arduino code should provide a user-friendly interface on the keypad for retrieving a contact from the contact list. Verify that the user is able to retrieve the contact accurately and quickly.
  + 1. Test Case 3: Verify contact list backup feature
* Input: Use the keypad to back up the contact list
* Expected output: Arduino code should provide a user-friendly interface on the keypad for backing up the contact list. Verify that the backup process is smooth and reliable, and that the user is able to restore the contact list if needed.

## The code should have an automatic voice recognition feature that can recognize specific voice commands from the user, such as "call" or "send SMS".

* + 1. Test Case 1: Verify voice recognition for calling feature
* Input: Speak "call" followed by a contact name to initiate a call
* Expected output: Arduino code should recognize the voice command "call" and initiate the call to the specified contact. Verify that the call is initiated correctly and accurately.
  + 1. Test Case 2: Verify voice recognition for sending SMS feature
* Input: Speak "send SMS" followed by a contact name and a message to send an SMS message
* Expected output: Arduino code should recognize the voice command "send SMS" and compose and send the SMS message to the specified contact. Verify that the SMS message is composed and sent correctly and accurately.
  + 1. Test Case 3: Verify voice recognition accuracy
* Input: Speak various voice commands to test the voice recognition accuracy
* Expected output: Arduino code should accurately recognize the voice commands and perform the corresponding actions, such as calling, sending SMS messages, and navigating through the menu system. Verify that the voice recognition accuracy is high and the system is easy to use.

## The code should be able to provide clear and audible notifications for incoming calls and SMS messages, allowing the user to respond promptly.

* + 1. Test Case 1: Verify audible notification for incoming calls
* Input: Simulate an incoming call and verify the audible notification
* Expected output: Arduino code should provide a clear and audible notification for incoming calls, allowing the user to respond promptly. Verify that the notification is clear and easy to understand, and that the user is able to respond to the call promptly.
  + 1. Test Case 2: Verify audible notification for incoming SMS messages
* Input: Simulate an incoming SMS message and verify the audible notification
* Expected output: Arduino code should provide a clear and audible notification for incoming SMS messages, allowing the user to respond promptly. Verify that the notification is clear and easy to understand, and that the user is able to read and respond to the message promptly.
  + 1. Test Case 3: Verify notification volume and clarity
* Input: Adjust the notification volume and test the clarity
* Expected output: Arduino code should provide a notification that is clear and audible, even in noisy environments. Verify that the volume is adjustable and the clarity is maintained at different volume levels.

# As a result we found, 5 modules, 50 requirements and 150 test cases for our project GVision i.e. an Arduino based wearable mobile phone for visually challenged people.