



## OPEN DATA KIT

# Project Plan : Improve the ODK-X suitcase application

## Student Information

- Name: Subham Raj
- Alternate names:  
Github :- shubhamr837
- Email: shubhamr837@gmail.com
- Telephone : +91 8757610034
- Time Zone: GMT+5:30



## Project

---

OPEN DATA KIT: Improve the ODK-X Suitcase application



## Project Abstract:

*ODK-X Suitcase* is a cross-platform tool that allows the user to upload, download, and update data on an ODK-X Cloud Endpoint from a personal computer . Users can either use the Graphical user interface or the command line interface to perform operations. Users can upload their data in CSV format and access them later. Data is uploaded in the Sync endpoint server. So, Sync endpoint server has to be configured to use the suitcase application .

To upload the tables users have to arrange data in a particular form with a specified `table_id`. This `table_id` is used later to download the table. Currently, users need to enter the exact `table_id` which they want to download else the application would show an error message stating that `table_id` doesn't exist. In this project, this problem will be solved by providing the user a list of existing `table_id` so that it is not necessary to remember it. Users might also want to select more than one `table_ids` for downloading more than one table at once. To allow the users to download multiple tables users need to be provided the download options which could be different for every table or same for all tables. There are 4

download options currently: download path, download attachments , apply scan formatting , download metadata.

Sync-Client ( Also a part of odk-x ) is a library used by the application to make http requests and return responses in json. In case of any error, responses are in html. So it is necessary to implement functionality that can show the appropriate error message in both the CLI and GUI.

Users are asked to login to the application every time they open it. This is inconvenient and the user must have a “remember me” option during login.

So, implementing a list of table\_id to select, allowing multiple downloads at once, improving error handling and saving user credentials are the important improvements that I will work on. Apart from this I will also work on improving the UI of the application and adding new features.



## Project Deliverables :

- 1 ) Present a list of existing table id to the user in the Download tab .
- 2 ) Add functionality to enable a user to select one or more database tables to download from the ODK-X Cloud Endpoint at once.
- 3 ) Save User Credentials.
- 4 ) Improve Error Handling.
- 5 ) other works.



## Present a list of existing table id to the user in the Download tab :



### Current implementation :

- 1 ) List of existing table ids are fetched from the server when the user logs in, uploads a table or resets the server.
- 2 ) User enters the table id to download in a text field. The path where the file will be downloaded is entered and other options are checked.
- 3 ) After the download button is clicked, the download path is verified and the list of table ids is searched for the entered table id.
- 4 ) If the table id is found to be present in the list the application starts downloading else it displays an error message stating that table id is invalid.

Users need to remember the exact table id for every download. This process is very inconvenient.

### Proposed implementation :

The list of existing table ids will be fetched the same way. This list will be used to implement a drop down of table ids. Users can directly select the table id from this list. To make it more convenient the list will be in alphabetical order.

To implement the dropdown JComboBox of java swing will be used. A listener will also be set for the dropdown which will perform necessary actions on selection.

Following code will implement the Drop down with the specified table ids in alphabetical order.

```
JComboBox<String> cb=new JComboBox<String>();
String[] tablesIds = (String[]) getAllTableId().toArray();
sort(tablesIds);
DefaultComboBoxModel<String> cbModel =new DefaultComboBoxModel<String>(tablesIds);
cb.setModel(cbModel);
cb.addActionListener (new TableIdDropDownActionListener(cbModel));
cb.setBounds( x: 50, y: 50, width: 90, height: 20);
this.add(cb);
```

Explanation for the code :

- The sort method sorts the array in alphabetical order.
- The DefaultComboBoxModel helps to dynamically modify items in the dropdown. It takes the list of strings to show in the dropdown in its constructor.
- getAllTableId() method from CloudEndpointInfo currently returns a Set<String> .So it will be type casted to String[].
- The selected id will be processed by the TableIdDropDownActionListener.

## Add functionality to enable a user to select one or more database tables to download from the ODK-X Cloud Endpoint at once :

Currently users have to enter one table\_id to download and wait before it finishes to enter a new table id to download.

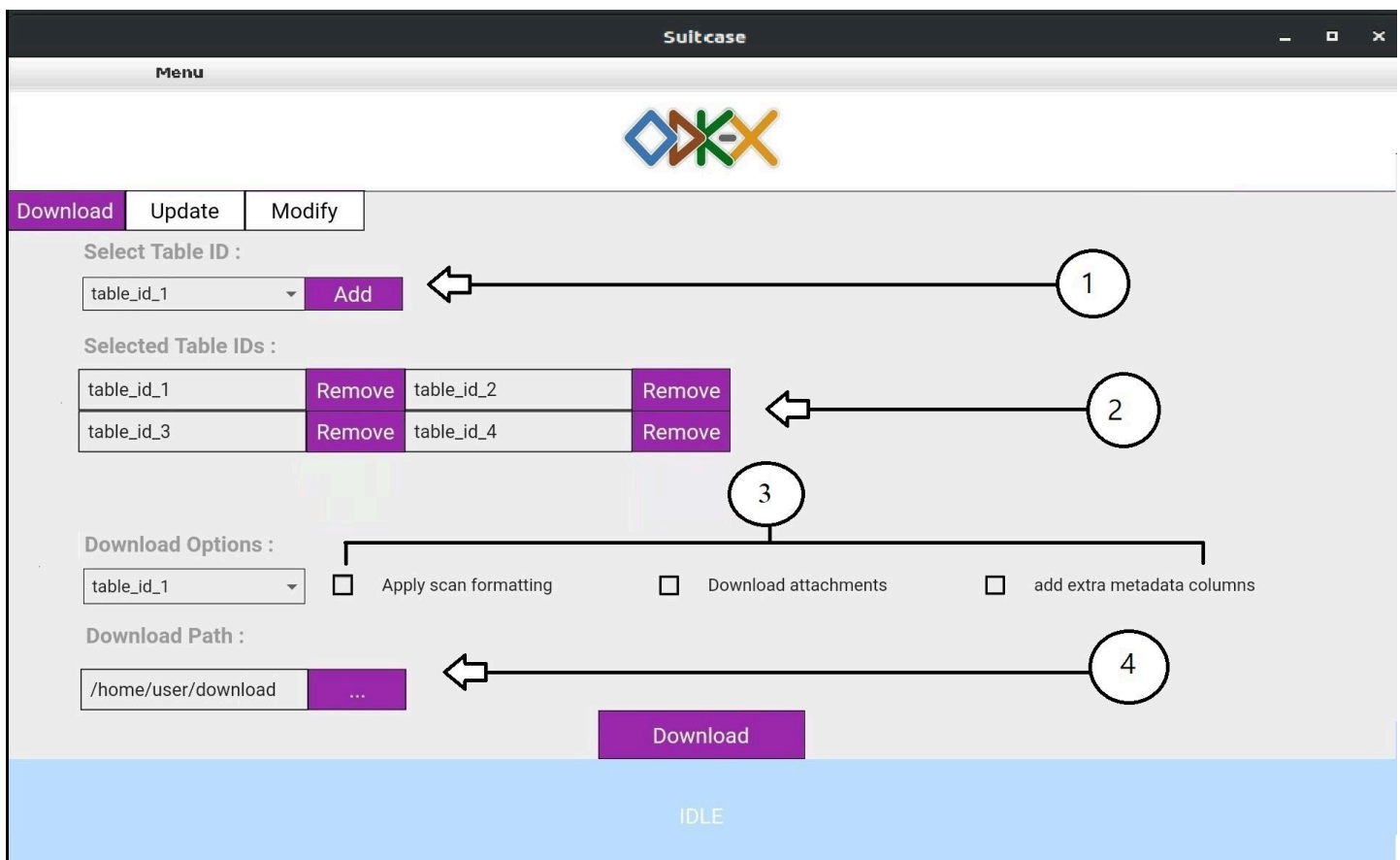
This project would aim to add the functionality to select multiple table\_ids and download it at once. The selected table ids will be added just below the drop down. With the option to remove. The selected table ids will be stored in a HashSet to ensure that no table id is added twice. Also the selected item will be made grey in the dropdown list using combo box renderer.

The screen will be structured as follows :

- On the top will be the dropdown to add table ids and the list of already added table ids.

- The download options (download path, download attachments, apply scan formatting, download metadata ) will have a drop down at the top which will contain the list of selected table ids. The drop down will also have an option “ All ” which will be selected by default. This will set the settings same for all the selected table ids.
- The download button and progress bar will be as it is in the current application.

Proposed Download page UI has 4 major components:



1. The dropdown to select table\_id.
2. Gridview of selected table\_id.
3. Download options for different tables.
4. Download path selection.

Following Objects will implement section 1 and 2:

### class **DropDownButtonPair** extends JPanel :

The **DropDownButtonPair** is the class to display the dropdown with an add button . After selection from the dropdown " add " button can be used to add it to the list of selected table\_ids.

Instance variables of this class :

```
JComboBox dropdown;  
AddRemoveButton button;  
TableIdDropDownActionListener listener;  
DefaultComboBoxModel defaultComboBoxModel;  
HashSet<String> added_table_ids;  
String currently_selected_item;
```

**JComboBox** is the dropdown which will display the table ids .

**AddRemoveButton** is the custom button which will display as an add button here .

**TableIdDropDownActionListener** is the listener for drop down value changes and will update the currently\_selected\_item accordingly .

**DefaultComboBoxModel** is the model provided by java swing to store the list of table ids and will be used by the JComboBox. Values in this model can be modified to change the elements of the dropdown dynamically.

**added\_table\_ids** is the HashSet to store the already added table ids .

**currently\_selected\_item** stores the value of the currently selected table\_id in drop down. Its value will be updated by **TableIdDropDownActionListener**.

### class **FieldButtonPair** extends JPanel :

This class contains the non editable text area to display the selected table ids. It contains a remove button on the right side of the text area to remove it.

Instance variables of this class :

```
JTextArea field;  
AddRemoveButton button;  
String table_id;
```

**AddRemoveButton** is a custom button. For this object it will act as a remove button with the functionality to remove the **FieldButtonPair** from the panel and the **table\_id** from the **HashSet** .

**class AddRemoveButton extends JButton:**

The **AddRemoveButton** class stores the **table\_id** of the field it is displaying which will be used to remove the **FieldButtonPair** from the gridview on selecting remove.

Instance variables of the class :

```
Action action  
String table_id
```

**Action** is an enum which will be either “ add “or “ remove “.

Code for the listener of **AddRemoveButton** :

```
@Override  
public void actionPerformed(ActionEvent actionEvent) {  
  
    AddRemoveButton source = (AddRemoveButton) actionEvent.getSource();  
  
    if (source.action == Action.ADD) {  
        if (!dropDownButtonPair.getSelected_table_ids().contains(dropDownButtonPair.getSelectedItem())) {  
            dropDownButtonPair.getSelected_table_ids().add(dropDownButtonPair.getSelectedItem());  
            addField(dropDownButtonPair.getSelectedItem());  
        }  
    } else if (source.action == Action.REMOVE) {  
        dropDownButtonPair.getSelected_table_ids().remove(source.getTableId());  
        removeField(source.getTableId());  
    }  
}
```

The action will be either add or remove .

**ActionEvent** will be type casted to get the **AddRemoveButton**.

For remove :



getTable\_id() method from **AddRemoveButton** will get the table\_id to remove the field. The table\_id will also be removed from the selected table ids HashSet in the **DropDownButtonPair** instance .

For add :

For already selected items the font color will be changed to grey in the dropdown using **ComboBox Renderer**. Still if the user clicks on an already selected item. Listener will check that the table\_id is not already selected in the HashSet of **DropDownButtonPair** instance.

After this the new table\_id will be added to the HashSet .

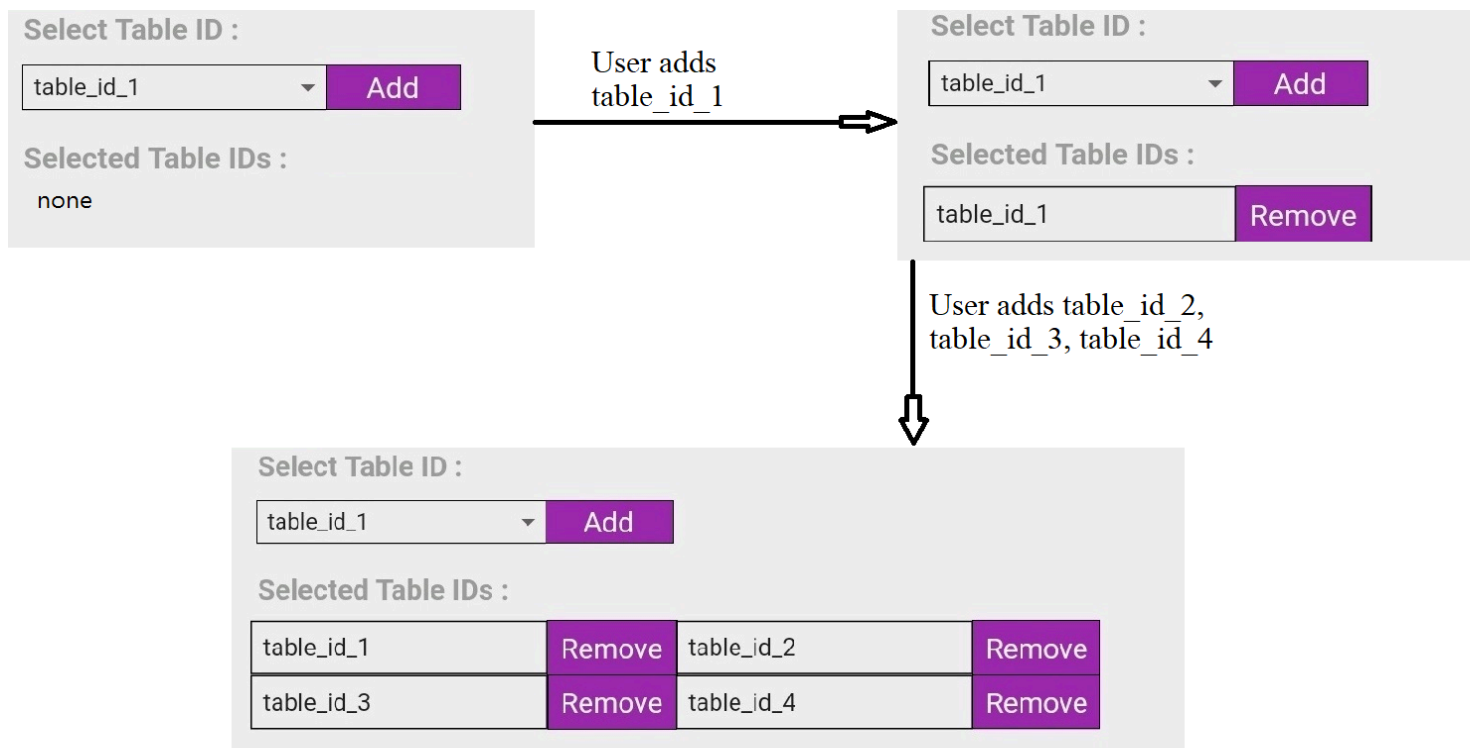
addNewField method in the code sample above will add the new field in the screen with remove option.

There are two ways of implementing the list of selected table\_ids in UI.

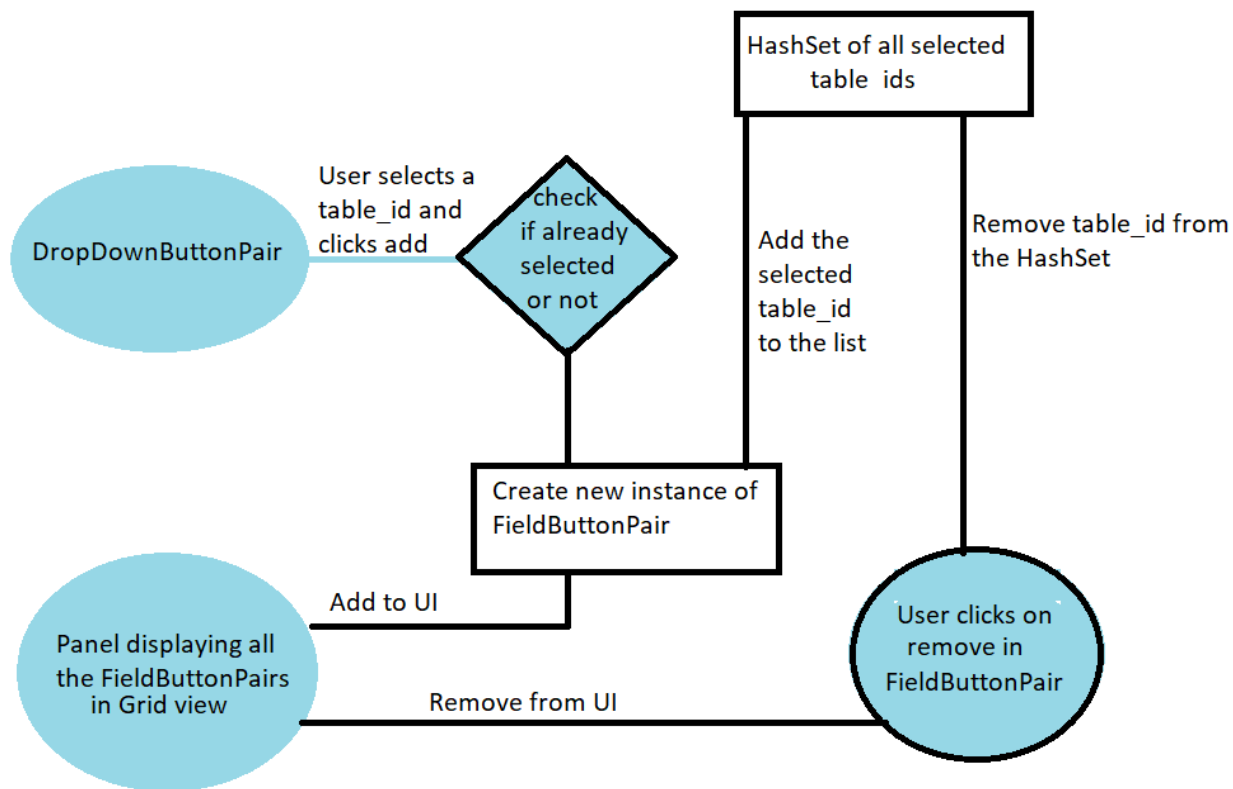
1 ) All the FieldButtonPair will be displayed in a 4\*3 grid view. This means 12 will be the max number of table\_ids that can be selected at once.

2 ) Use JScrollPane to make the grid view scrollable. Number of columns will be 2 and the number of rows won't have any constraint.

UI implementation for option 1:



Following Flow diagram shows the implementation :



### Download options section:

There are three download options for each table\_id "download attachments", "apply scan formatting" and "Extra metadata columns".

In the top of the download option section will be a drop down to select the table id.

After this user can set download options for each table\_id. There will also be an option "All" in the dropdown to set download options for all the table\_ids at once.

Currently the **CsvConfig** class in the package “org.opendatakit.suitcase.model” has the following instance variables :

boolean downloadAttachment;

boolean scanFormatting;

boolean extraMetadata;

Each table\_id will have a CsvConfig object for its download options.

### **Download Button and progress bar :**

This section of the current application will be the same for the UI part. It will contain the file chooser to specify the download path.

Two objects have to be initialized to run the download tasks.

1 ) AttachmentManager

2 ) ODKCsv

Both can be initialized using the variables in the csvconfig class and save path. After this the DownloadTask will be initialized for each table\_id.



### **Save user credentials :**



Currently the application initializes the sync-client from the credentials after the user is logged in. Once the application is closed the user is automatically logged out as credentials are not saved. This project would store the username, password, application id and cloud endpoint url to a properties file.

Code to write to properties file:

```

private static void saveUsernamePasswordToFile(Credentials credentials)
{
    try (OutputStream output = new FileOutputStream(propertiesFileName)) {

        Properties prop = new Properties();

        // set the properties value
        prop.setProperty("username", credentials.getUsername());
        prop.setProperty("password", credentials.getPassword());
        prop.setProperty("cloudEndpointUrl", credentials.getCloudEndpointUrl());
        prop.setProperty("appId", credentials.getAppID());

        // save properties to project root folder
        prop.store(output, comments: null);

    } catch (IOException io) {
        io.printStackTrace();
    }
}

```

The credentials will be secured in different ways for different operating systems .

- **Linux and mac**

Linux has a key management tool called gnome keyring .“org.netbeans.api.keyring“ library can be used to implement encryption in linux and mac.The library makes it easy to use gnome keyring from java. For mac the library uses Mac OS X Keychain, using the default login keychain.The library has the license for use.

Following code implements encryption and stores the password in a key value pair in linux and mac using the library “org.netbeans.api.keyring”.

```

private void encryptForLinux(String key,char[] password,String description)
{
    Keyring.save(key,password,description);
}

private char[] decryptForLinux(String key)
{
    return Keyring.read(key);
}

```

For both linux and mac the file permissions would be set to owner read and write only. This will be done using Posix file permissions. The following code will create the properties file with permissions and save the credentials.

```
public static void saveForLinux(Credentials credentials) {  
  
    createPropertiesFile(propertiesFileName);  
  
    Set<PosixFilePermission> perms = new HashSet<PosixFilePermission>();  
    //add owners permission  
    perms.add(PosixFilePermission.OWNER_READ);  
    perms.add(PosixFilePermission.OWNER_WRITE);  
    perms.add(PosixFilePermission.OWNER_EXECUTE);  
  
    try {  
        Files.setPosixFilePermissions(Paths.get(propertiesFileName), perms);  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
  
    saveUsernamePasswordToFile(credentials);  
}
```

- **Windows**

For windows the file will contain the password in an encrypted format .

The encryption will be done using the Data Protection API provided by windows .

The library windpapi4j (<https://github.com/peter-gergely-horvath/windpapi4j>) can be used to implement DPAPI encryption in Java. It has the license for use .

Following code would encrypt the password and store the credentials .

```

public static void saveForWindows(Credentials credentials) {
    createPropertiesFile(propfilename);

    try {
        WinDPAPI winDPAPI = WinDPAPI.newInstance(WinDPAPI.CryptProtectFlag.CRYPTPROTECT_UI_FORBIDDEN);

        byte[] clearTextBytes = credentials.getPassword().getBytes(StandardCharsets.UTF_8);

        byte[] cipherTextBytes = winDPAPI.protectData(clearTextBytes);

        String encryptedPassword = new String(cipherTextBytes, StandardCharsets.UTF_8);

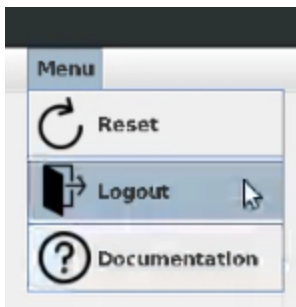
        credentials.setPassword(encryptedPassword);
        saveUsernamePasswordToFile(credentials);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

After this the user has to be logged in if all the credentials are present in the properties file and the server gives a proper response for the credentials.

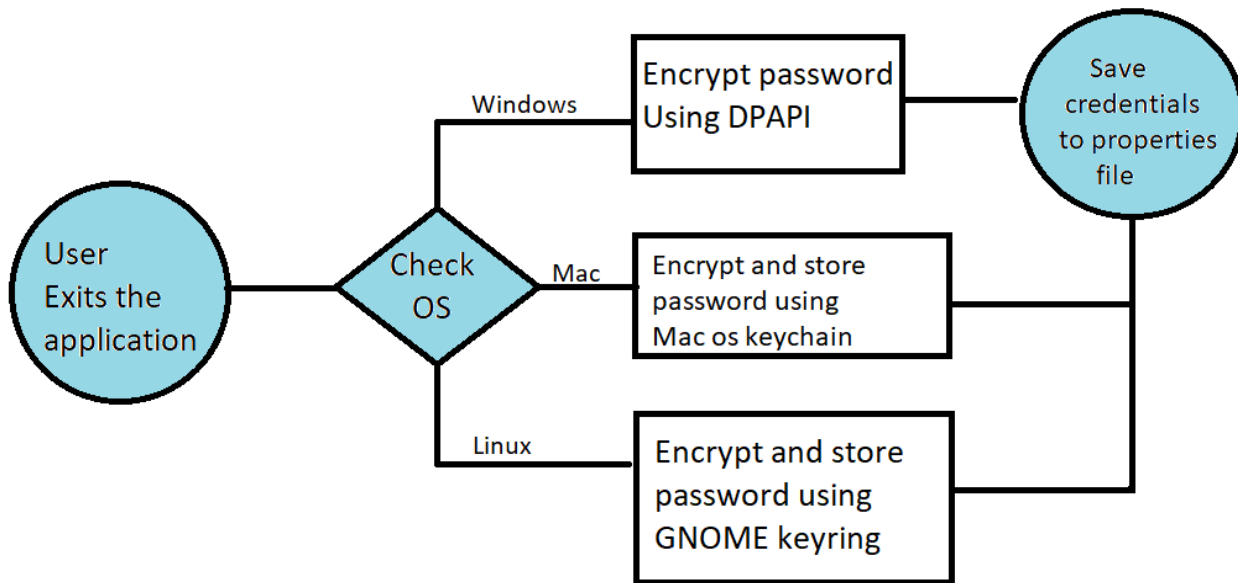
A log out button also has to be implemented. This will be implemented in a menu in the menu bar (top left corner of the application). The application would disable this button while any operation is being performed .

Proposed implementation of the menu in menu bar .

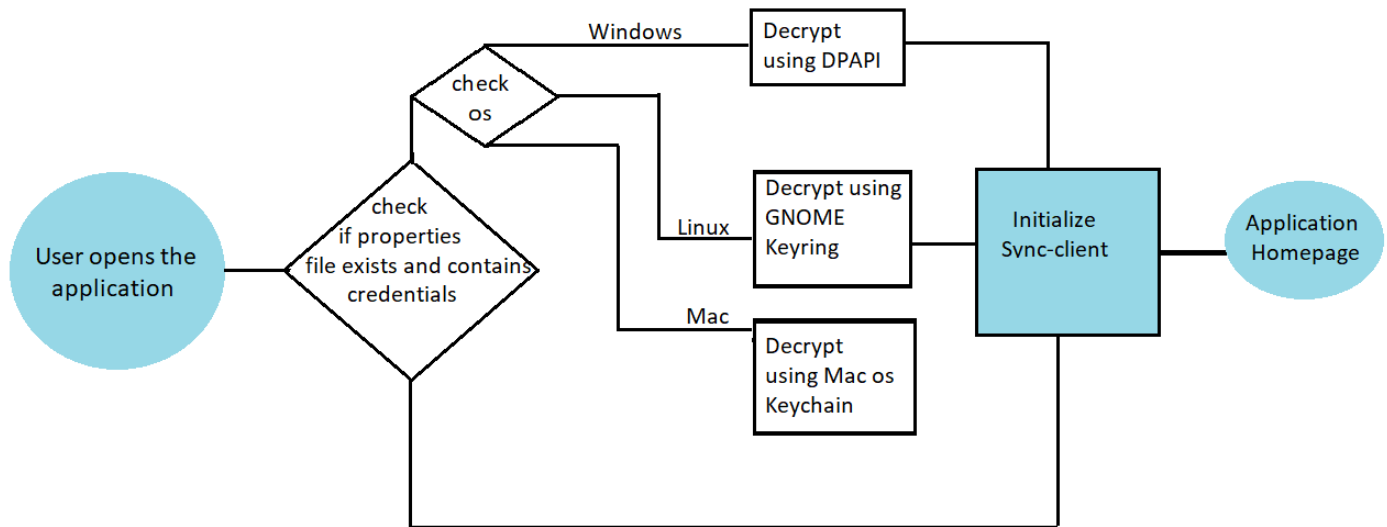


Following Flow Diagram describes the entire process.

**Encryption :**



### Decryption :



## Improve error handling so the error messages contained in the HTML response are propagated to the user-interface: -

Currently the Sync-client throws a `jsonException` when response is not in json. The Response is in html when any error occurs. This would require changes in both sync client library and suitcase.

Example : This is the html response from sync endpoint to the sync-client on providing wrong username when fetching tables .

```
<!doctype html>
<html lang="en">
  <head>
    <title>HTTP Status 401 - Unauthorized</title>
    <style type="text/css">body {font-family:Tahoma,Arial,sans-serif;} h1, h2, h3, b {color:white;background-color:#525D76;} h1 {font-size:22px;} h2 {font-size:16px;} h3 {font-size:14px;} p {font-size:12px;} a {color:black;} .line {height:1px;background-color:#525D76;border:none;}</style>
  </head>
  <body>
    <h1>HTTP Status 401 - Unauthorized</h1>
    <hr class="line" />
    <p><b>Type</b> Status Report</p>
    <p><b>Message</b> Unauthorized</p>
    <p><b>Description</b> The request has not been applied because it lacks valid authentication credentials for the target resource.</p>
    <hr class="line" />
    <h3>Apache Tomcat/8.5.64</h3>
  </body>
</html>
```

It has the following important parameters :

- Message
- Description
- Status code

Status code can be directly obtained from the `HttpServletResponse` response object in sync client .

Description and message are pretty much clear from the status code itself .

But still for other operations error responses from sync-endpoint will be analysed. If description and message are important it has to be provided to the user. For this Html can be parsed and description and message can be obtained.

A custom exception class can be created in the sync client which will store all this information .

This custom exception will be thrown whenever response status code is not equal to 200 .

Following code for exception can be used :



```

public class InvalidHttpResponseException extends Exception {
    private final ErrorCode code;

    public InvalidHttpResponseException(String message, Throwable cause, ErrorCode code) {
        super(message, cause);
        this.code = code;
    }

    public ErrorCode getCode() {
        return this.code;
    }
}

```

ErrorCode will be an enum which will have most common status codes and messages for the status codes. This will be thrown by the `HttpRequestExecute(HttpRequestBase request, String contentType, boolean excludeAcceptParams)` method . Currently in the sync client all the http requests are executed by this method .

### Displaying the Error Message to the user :

To display the error message a new error dialog will be created. It will display the error message to the user.

Currently the error message is same for both invalid login credentials or invalid cloud endpoint url.

As the status code will be different for both of them, different error messages will be defined.

Following code will display an error message with a custom icon and button.

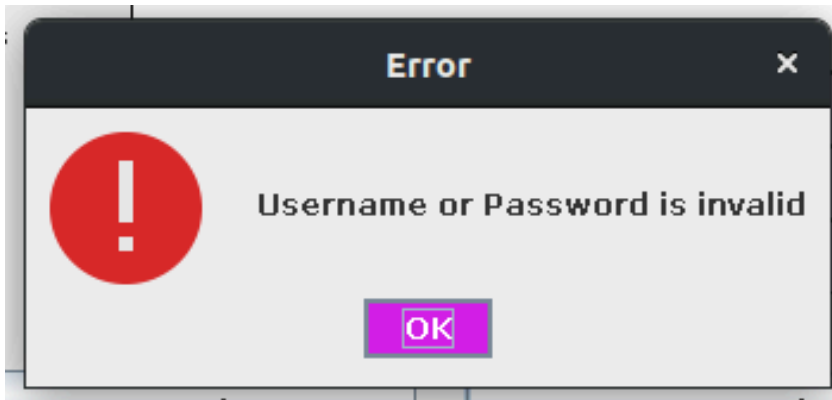
```

private void showErrorMessage(String title,
                             String description, String imagePath) {

    ImageIcon errorIcon = new ImageIcon(imagePath);
    JButton buttonToDispose = new JButton( text: "OK");
    buttonToDispose.setBackground(Color.decode(themeButtonColor));
    buttonToDispose.setSize(new Dimension( width: 40, height: 20));
    buttonToDispose.setForeground(Color.WHITE);
    buttonToDispose.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent actionEvent) {
            JOptionPane.getRootFrame().dispose();
        }
    });
    JButton[] buttonsFoDialog = {buttonToDispose};
    JOptionPane.showOptionDialog( parentComponent: null, description, title,
        JOptionPane.OK_OPTION,
        JOptionPane.INFORMATION_MESSAGE, errorIcon,
        buttonsFoDialog, buttonsFoDialog[0]);
}

```

The Error Dialog UI :



## Other works : -

### Improving the UI of the application :

Some improvements can be done in the UI of the application :

- **Changing the theme of the application .**

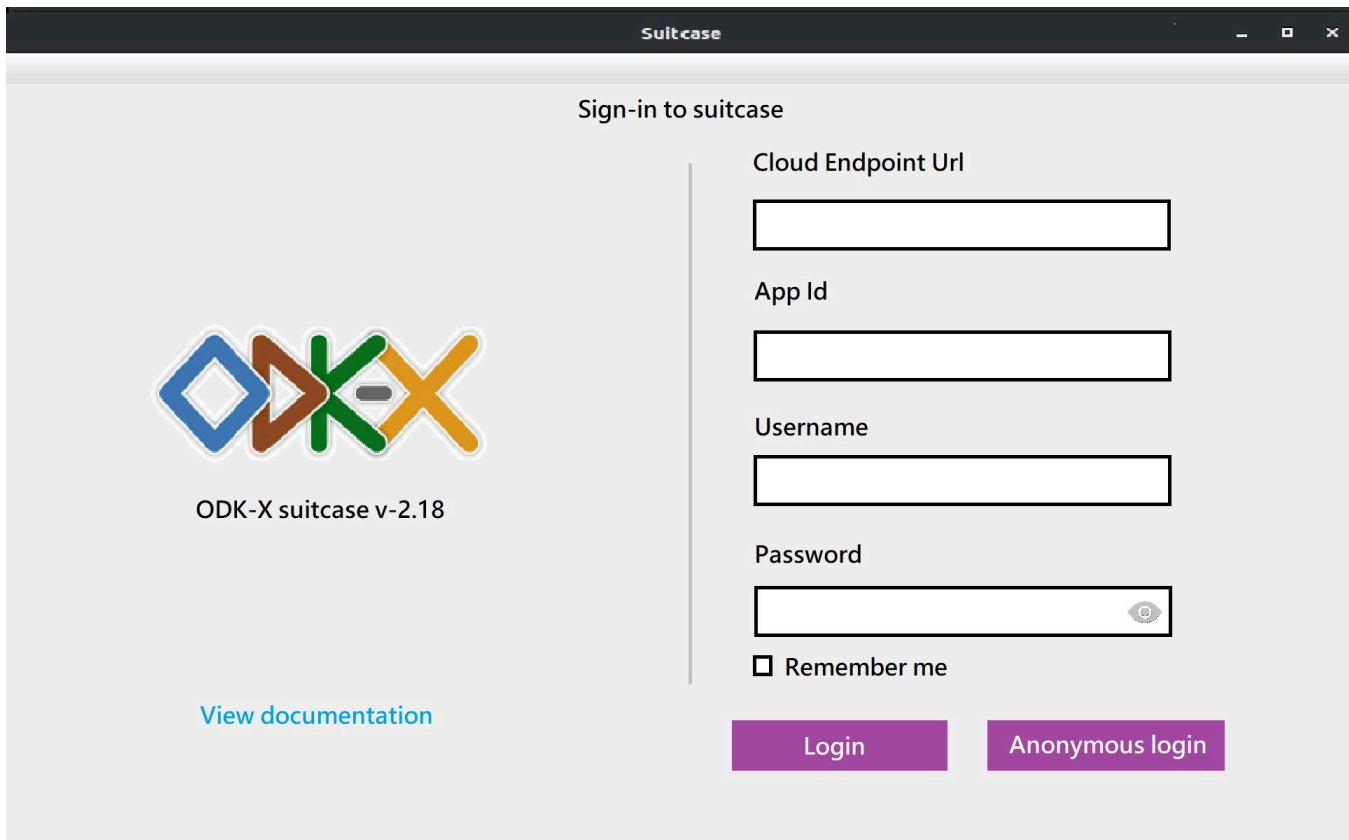
This involves :

- > Adding an odk-x logo to the application
- > Adding a background color to the application
- > Adding background color to buttons
- > Using custom fonts.

To improve the UI buttons can be modified to have a single color background. Currently buttons have background with transition color .

Proper padding can be added and the size of buttons can be reduced.

Proposed UI of Login page:



The image shows a proposed UI for the login page of ODK-X suitcase v-2.18. The window title is "Suitcase". The page has a light gray background. On the left, there is a logo consisting of four interlocking shapes in blue, orange, green, and yellow, with the text "ODK-X suitcase v-2.18" below it. A link "View documentation" is also present. On the right, the title "Sign-in to suitcase" is centered. Below it, there are four input fields: "Cloud Endpoint Url", "App Id", "Username", and "Password". The "Password" field has a toggle icon. Below the "Password" field is a checkbox labeled "Remember me". At the bottom right, there are two buttons: "Login" and "Anonymous login".

Suitcase

Sign-in to suitcase

Cloud Endpoint Url

App Id

Username

Password

☐ Remember me

[View documentation](#)

Login Anonymous login

- **Changing the title of the application.**

The application currently has the title “org.opendatakit.suitcase.Suitcase”. It can be changed to “Suitcase”.

- **Placing the reset button in a menu instead of the upload tab.**

Currently the reset button is placed in the upload tab. It can be moved to a menu in the top right corner of the application. This menu would also contain the log out option .

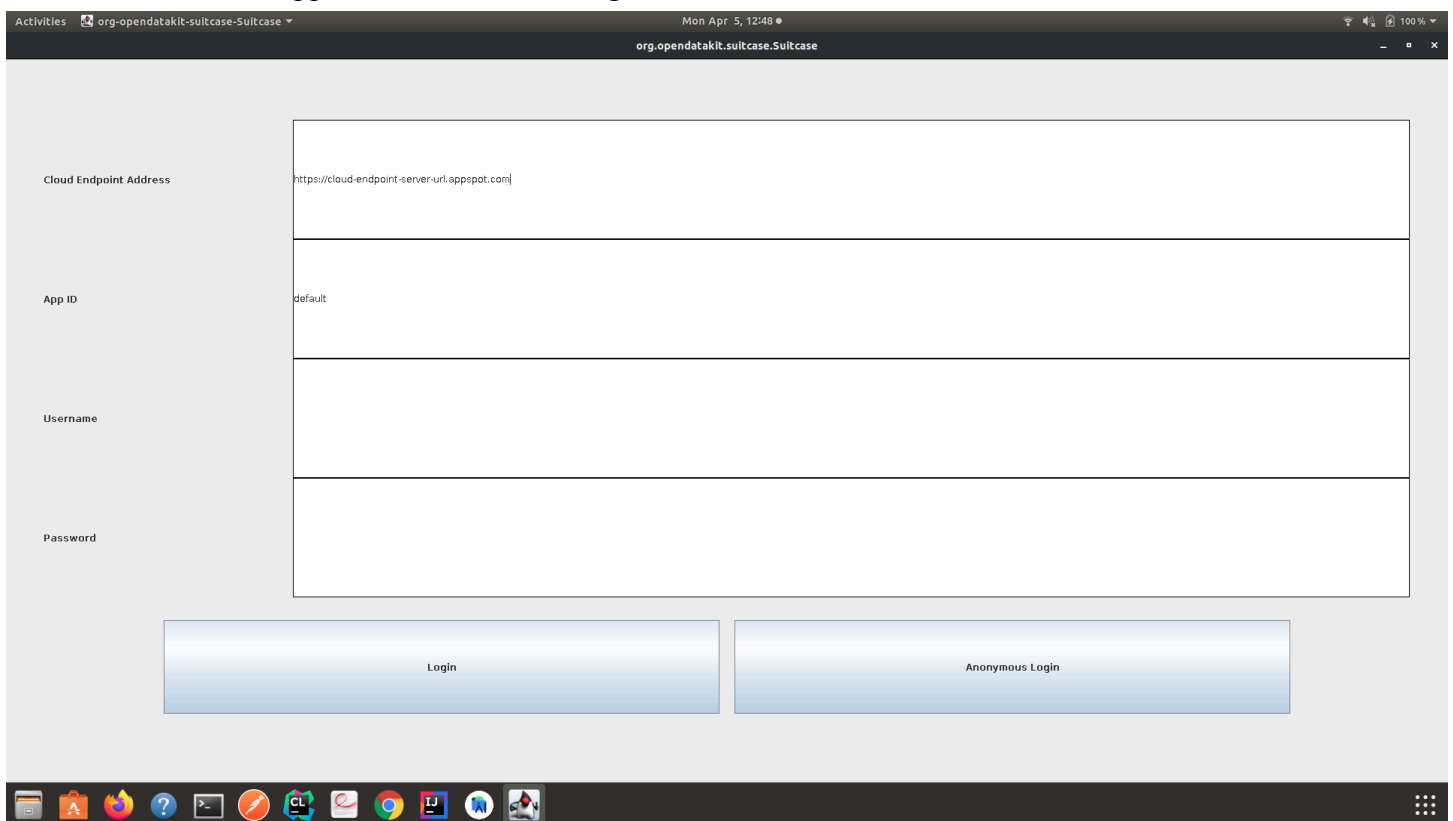
- **Adding icons to the buttons .**

Download , upload , reset , delete , logout and login buttons can have icons .

- **Making the components of the application responsive to the window size.**

Currently the components in the screen resize according to window size . On maximizing the window the components become too large and UI doesn't look good .

Current UI of application on maximizing :



The application also doesn't have a minimum window size setup , `setMinimumSize()` method in `JFrame` can be used to set a minimum size .

For large window size Possible solution is to set a maximum threshold of height and width till which the components will resize with the window after the size of the window exceeds this threshold extra padding will be added around the components.

The `addComponentListener()` method can be used to implement this for different components of the application. This method takes a `ComponentAdapter`. The Component adapter has a method which is called when the window is resized.

Following code implements this logic for the `buttonsPanel` in `loginPanel` using component adapter.

```
this.addComponentListener(new ComponentAdapter() {
    public void componentResized(ComponentEvent componentEvent) {
        if (componentEvent.getComponent().getSize().height < MAX_HEIGHT &&
            componentEvent.getComponent().getSize().width < MAX_WIDTH) {
            resizeComponent(componentEvent.getComponent().getHeight(), componentEvent.getComponent().getWidth());
        } else if (componentEvent.getComponent().getSize().height < MAX_HEIGHT &&
            componentEvent.getComponent().getSize().width > MAX_WIDTH) {
            int width = componentEvent.getComponent().getWidth();
            int LEFT_PADDING = DEFAULT_LEFT_PADDING + (width - MAX_WIDTH) / 2;
            int RIGHT_PADDING = DEFAULT_RIGHT_PADDING + (width - MAX_WIDTH) / 2;
            Border emptyBorder = BorderFactory.createEmptyBorder(DEFAULT_TOP_PADDING, LEFT_PADDING, DEFAULT_BOTTOM_PADDING, RIGHT_PADDING);
            buttonsPanel.setBorder(emptyBorder);
            resizeComponent(componentEvent.getComponent().getHeight(), MAX_WIDTH);
        } else if (componentEvent.getComponent().getSize().height > MAX_HEIGHT &&
            componentEvent.getComponent().getSize().width < MAX_WIDTH) {
            int height = componentEvent.getComponent().getHeight();
            int TOP_PADDING = DEFAULT_LEFT_PADDING + (height - MAX_HEIGHT) / 2;
            int BOTTOM_PADDING = DEFAULT_RIGHT_PADDING + (height - MAX_HEIGHT) / 2;
            Border emptyBorder = BorderFactory.createEmptyBorder(TOP_PADDING, DEFAULT_LEFT_PADDING, BOTTOM_PADDING, DEFAULT_RIGHT_PADDING);
            buttonsPanel.setBorder(emptyBorder);
            resizeComponent(MAX_HEIGHT, componentEvent.getComponent().getWidth());
        } else {
            int width = componentEvent.getComponent().getWidth();
            int LEFT_PADDING = DEFAULT_LEFT_PADDING + (width - MAX_WIDTH) / 2;
            int RIGHT_PADDING = DEFAULT_RIGHT_PADDING + (width - MAX_WIDTH) / 2;
            int height = componentEvent.getComponent().getHeight();
            int TOP_PADDING = DEFAULT_LEFT_PADDING + (height - MAX_HEIGHT) / 2;
            int BOTTOM_PADDING = DEFAULT_RIGHT_PADDING + (height - MAX_HEIGHT) / 2;
            Border emptyBorder = BorderFactory.createEmptyBorder(TOP_PADDING, LEFT_PADDING, BOTTOM_PADDING, RIGHT_PADDING);
            buttonsPanel.setBorder(emptyBorder);
            resizeComponent(MAX_HEIGHT, MAX_WIDTH);
        }
    }
});
```

`addComponentListener()` is used to add a listener to `loginPanel`. As the login panel increases in size, the `buttonsPanel` which contains “login” and “anonymous login” button will also increase till the `MAX_HEIGHT` and `MAX_WIDTH` thresholds are reached .

There are four cases for this :

- 1 ) Both height and width are under threshold .
- 2 ) width exceeds the threshold but height does not.
- 3 ) height exceeds the threshold but width does not.
- 4 ) Both exceed the threshold .

If size exceeds the threshold padding will be added to that dimension from both the sides .

The `resizeComponent()` method resizes the component according to the size of the parent of the `buttonsPanel` . It takes the input height and width of the parent .

```
public void resizeComponent(int height, int width){  
  
    int xCoordinatePositionLoginButton = width / 5;  
    int yCoordinatePositionLoginButton = height / 4 * 3;  
    buttonsPanel.setBounds(xCoordinatePositionLoginButton, yCoordinatePositionLoginButton,  
        width: width / 2,  
        height: height / 10);  
}
```

`xCoordinatePositionLoginButton` is the X Coordinate from the top left position of the frame .

It is set to  $\frac{1}{5}$  th of the current width. This means that the top left corner of the `buttonsPanel` will be at  $\frac{1}{5}$  th position of the current width in the screen. The Y coordinate position is set to  $\frac{3}{4}$  th of the current height. This means that the top left corner of the `buttonsPanel` will be at  $\frac{3}{4}$  of the current height in the screen.

In the `setBounds` method the third parameter is the width which is set to  $\frac{1}{2}$  of the current width of the screen . The fourth parameter is the height which is set to  $\frac{1}{10}$  of the screen height .

- **Adding an icon in the password text field to view/hide the text field .**

This will be implemented to view/hide the password while typing .

## Improving code structure of the application :

- **Replacing Hard coded string values .**

Currently all the labels and messages are hard coded in the application . A string resource file can be used to simplify this .

## Adding new features to the application :

- **Adding a delete table and update table option .**

A new tab name “Modify” can be added to the application. Modify table will have the option to delete and update. Sync-client already has a method for updating tables which is used by CLI of suitcase. For delete sync-client methods of reset can be modified to delete a table using table\_id.

Proposed UI of the page:

The image shows a proposed UI for a window titled "Suitcase". At the top is a dark grey menu bar with the text "Menu". Below the menu bar is a logo consisting of four colored squares (blue, orange, green, yellow) arranged in a cross pattern. The main content area has three tabs: "Download", "Update", and "Modify". The "Update" tab is selected and highlighted in purple. Under the "Update" tab, there are three sections: "Update" with a "Select Table ID" dropdown menu showing "table\_id\_1", a "Data version" input field with the value "2", and a "Data Path" input field with the value "/home/user/download" and a file explorer icon. To the right of these fields is a purple "Update" button. Below the "Update" section is a "Delete" section with a "Select Table ID" dropdown menu showing "table\_id\_1" and a red "Delete" button. At the bottom of the window is a light blue bar with the text "Idle".

- **Adding a link to Documentation in the login page**

Link to documentation will be added to the bottom of the login screen. Inside the application in the top right menu there will be a “Documentation” option with reset and logout. This option will directly open the Documentation in the default browser.

- **Add drag and drop for upload and update.**

Instead of opening the file chooser user can directly drag and drop the folder to upload/update. Test cases for this can be added using ComponentDragAndDrop of AssertJ swing.

## Writing test cases for the application :

Unit tests will be written using JUnit. AssertJ can be used to write test cases for the GUI of the application.

Unit tests will be written for the following:

- Download task
- Upload task
- Update task
- Reset task
- Delete task

## Why AssertJ Swing for GUI ?

Features of AssertJ swing that will be useful for testcase :

- Simulation of user interaction with GUI.
- Easy and reliable component lookup (by name , by type or by custom search ).
- Well written documentation.
- Ability to get screenshots for failed tests.
- Can be used with Junit. (Existing tests are using junit).

To implement this every component which has to be tested should be given a name using the setName() method . After this on each component can be easily searched and tested.

Test will be written for the following :

- **Button State while operation is being performed**

This will check if other buttons are disabled while any operation is performed.

- **Login Tests :**

To check login same test credentials will be used that are used for currently written tests.

Following code will implement a login test using AssertJ.

```
@Test
public void testLogin() {

    frameFixture.textBox( name: "CloudEndpointUrlTextFiled").enterText(System.getProperty("test.aggUrl"));
    frameFixture.textBox( name: "AppIdTextField").enterText(System.getProperty("test.appId"));
    frameFixture.textBox( name: "UsernameTextFiled").enterText(System.getProperty("test.userName"));
    frameFixture.textBox( name: "PasswordTextField").enterText(System.getProperty("test.password"));
    frameFixture.button( name: "LoginButton").click();
    frameFixture.button( name: "AnonymousLoginButton").requireDisabled();
    frameFixture.button( name: "LoginButton").requireDisabled();
    FrameFixture mainFrame = WindowFinder.findFrame( frameName: "mainFrame").withTimeout(loginTimeOut).using(robot);
}
```



After the four text fields are filled, the login button will be clicked. Both the login buttons will be tested if they are disabled or not just after login is clicked . New frameFixture will be initialized for the next screen . loginTimeout is the time for which frame fixture will wait to find the new frame .

- **Tests for display of error messages .**

Currently JOptionPane is used to display error messages or any other message in a dialog box. To test a message is displayed or not “frameFixture.optionPane().requireErrorMessage()” can be used .

Following error messages will be tested :

- > Download with incorrect path.
- > Upload with incorrect data .
- > Login with any field missing .
- > Anonymous login with any field missing .
- > Login with incorrect credentials .
- > Update with incorrect data .

Following code tests error message for login with empty username using AssertJ :

```
@Test
public void testLoginWithoutUsername() {

    frameFixture.textBox( name: "CloudEndpointUrlTextFiled").enterText(System.getProperty("test.aggUrl"));
    frameFixture.textBox( name: "AppIdTextField").enterText(System.getProperty("test.appId"));
    frameFixture.textBox( name: "UsernameTextFiled").enterText("");
    frameFixture.textBox( name: "PasswordTextField").enterText(System.getProperty("test.password"));
    frameFixture.button( name: "LoginButton").click();
    frameFixture.optionPane().requireErrorMessage().requireMessage("Username is required");

}
```

frameFixture is FrameFixture object of assertJ. It helps to control all the operations in the GUI.

- **Test Operations of download , upload , delete , reset .**

Gui Tests will be implemented for all these operations using AssertJ.

## **Documentation for the application :**

Documentation will be added for the following new features :

- Selecting and downloading multiple tables at once

- Deleting tables .
- Updating tables .

Changes in the existing documentation :

- Add screenshots in the documentation for upload and download. To explain all the options and text fields available on screen.
- Create video tutorials for upload and download .
- Add details for anonymous login feature .
- Add documentation for the reset button to refresh the table list .

To make the code easy to understand, details about all the functions and variables will be added as comments.

Apart from this video tutorials will be created for the ODK-X youtube channel for all the features.

To make the documentation of GUI features more clear screenshots will be included.

## Timeline :

<b>Community Bonding Period</b> May 17 - June 7	
<b>Week 1</b> 17th May - 23 May	<ul style="list-style-type: none"> <li>• Know all the community members</li> <li>• Discuss about the project deliverables.</li> <li>• Go through the documentation and find all the shortcomings in it.</li> <li>• Go through all the error responses in html by the sync-client.</li> </ul>
<b>Week 2</b> 24th May - 30 may	<ul style="list-style-type: none"> <li>• Understand the working of sync-client library.</li> <li>• Understand how CSV is parsed by suitcase and sync-client.</li> <li>• Go through all the html error messages for api requests.</li> </ul>
<b>Week 3</b> 31 May - 7 June	<ul style="list-style-type: none"> <li>• Go through the code base of suitcase.</li> <li>• Work on improving the UI of components ( buttons , text fields, panels ,etc ).</li> <li>• Discuss implementation of the project.</li> <li>• Discuss implementation of test cases.</li> </ul>

<b>Phase 1 (June 7 - July 16)</b>	
<b>Week 4</b> (June 7 - June 13)	<ul style="list-style-type: none"> <li>● Implementing table_id selection dropdown .</li> <li>● Implement Multiple table ID selection with “Add” and “remove” buttons</li> </ul>
<b>Week 5</b> (14 June - 20 June )	<ul style="list-style-type: none"> <li>● Implement the “download options” section for multiple downloads.</li> <li>● Improve the UI for download and upload tabs.</li> </ul>
<b>Week 6</b> (21 june - 27 june )	<ul style="list-style-type: none"> <li>● Add methods to encrypt and store password for different operating systems .</li> <li>● Add functionality to save credentials in properties file .</li> <li>● Add functionality to login users from credentials saved in the file .</li> </ul>
<b>Week 7</b> <b>28 june - 4 july</b>	<ul style="list-style-type: none"> <li>● Add logout option to the application.</li> <li>● Add the new tab for delete and update</li> <li>● Improve the UI of the Login Page and add remember me checkbox .</li> </ul>
<b>Week 8</b> <b>5 July - 12 July</b>	<ul style="list-style-type: none"> <li>● Writing Test cases for Login, Download,Update in GUI.</li> <li>● Write documentation for implemented features.</li> </ul>
<b>Phase 1 Evaluation</b> (July 12 - July 16)	<ul style="list-style-type: none"> <li>● Buffer Period</li> </ul>
<b>Work Period</b>	
<b>Week 9</b> (16 july -24 July )	<ul style="list-style-type: none"> <li>● Add a Custom exception to the sync-client library and improve error handling .</li> <li>● Add messages for different status codes .</li> <li>● Parse any important html responses.</li> <li>● Make a custom dialog box to display error messages.</li> </ul>
<b>Week 10</b> (26 July - 1 August )	<ul style="list-style-type: none"> <li>● Write test cases for all error messages.</li> <li>● Write test cases for new features (delete and update).</li> </ul>

<b>Week 11</b> <b>(2 August - 8 August)</b>	<ul style="list-style-type: none"> <li>● Add Icons and improve the UI .</li> <li>● Make components responsive to screen size .</li> <li>● Document the changes .</li> <li>● Documenting Scope for further improvements .</li> </ul>
<b>Week 12</b> <b>(9 August - 23 August )</b>	<ul style="list-style-type: none"> <li>● Fixing Bugs if found .</li> <li>● Improve code structure of application and add Comments in code for future contributors</li> <li>● Complete any other remaining work.</li> </ul>
<b>Final Evaluation</b> <b>(16 August - 23 August )</b>	<ul style="list-style-type: none"> <li>● Writing a blog post</li> <li>● Summarize</li> </ul>