Code Workflow of Adding an Admin User

1. When starting the application, the login screen is displayed. The screen is realized by 'MLALoginActvity.java', is invoked from App -> src -> main -> java ->paril -> ui -> activity. The sequence of actions in 'MLALoginActivity.java' are below:
    a. The rendering of the screen is done by the invocation of 'onCreate' method that sets the view of login.
    b. After the view is set and user inputs (login credentials) are provided, a click on 'login' button invokes 'onClick' method inside 'onCreate' that, among other work, performs the authentication. The logic of the authentication is defined in the action class ('MLALoginAPI') in the same activity. An action class defines pre-action, post-action, and action logic in the 'onPreExecute', 'onPostExecute', and 'doInBackground' methods respectively. When the 'MLALoginAPI' object is executed: (1) 'onPreExecute' displays a progress dialog. (2) 'doInBackground' creates a client endpoint and invokes the 'authenticate' method that is mapped to the 'api/Register/GetRegisterAuth' Web service. The mapping is defined in 'APIInterface.java' under the webservice folder. The Web service, in turn, performs authentication on the server side. (3) 'onPostExecute' creates an intent that starts a new activity (defined in 'MLAHomeActivity.java' under the same code folder) – navigating to the home screen of the app.

    Note: Intent is a messaging mechanism of Android, it cab be used to start a new activity or service from the existing one. An intent includes a to-be-started activity or service as well as the parameters. In this case, three parameters are passed to 'MLAHomeActivity': (1) userId, (2) userName and (3) userType.

2. The sequence of actions in 'MLAHomeActivity.java' are below:
    a. The rendering of the screen is done by the invocation of 'onCreate' method that sets the view of home screen of the user of a certain type (e.g. Admin, Instructor, student).
    b. When pulling up the left panel, method 'parentClicking' is invoked to display the menus.
    c. If the user clicks on 'Home' menu on the left panel of the home screen, 'onHomeClick' method is invoked to render a screen fragment rendered by 'MLAHomeFragment.java' under ui->fragment.
    d. To add an admin, user clicks on 'Admin' on the menu which navigates to another screen rendered by 'MLAAdminViewFragment' under ui->fragment.

3. The sequence of actions in 'MLAAdminViewFragment' are below:
    a. The rendering of the screen is done by the invocation of 'onCreateView' method that sets the view to display a list of the existing admin users and a '+' button.
    b. 'onResume' is called immediately before the activity starts to interact with the user. The logic for fetching the existing list of admins is defined in the action class

('MLAGetAllAdminAPI') in the same activity. When the 'MLAGetAllAdminAPI' object is executed: (1) 'onPreExecute' displays a progress dialog. (2) 'doInBackground' creates a client endpoint and invokes the 'getAdminUsers' method that is mapped to the 'api/Admin/GetAllAdmin' Web service as defined in 'APIInterface.java'. The Web service retrieves all existing admins on the server side encapsulated in model object 'MLAAdminDetails'. (3) 'onPostExecute' uses an adapter object of 'MLAAdminAdapter' to facilitate the presentation of the list of the admins for user interaction. Specifically, when a selected item on the list is clicked for 'edit/view' or 'delete', starts activity 'MLAViewAdminActivity' to edit/view the admin details, or executes the 'MLAPostAdminRmvAPI' object that triggers 'removeAdmin' mapped to Web service 'api/DeleteAdmin/PostAdminRmv'.

Note: In Android, Adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to an adapter view then view can takes the data from the adapter view and shows the data on different views like as ListView, GridView, Spinner, etc.

c. To add a new admin, the user clicks on '+' button which navigates to another screen rendered by 'MLAViewAdminActivity' under ui->activity.

Note: On click of '+' button, two parameters are passed to 'MLAViewAdminActivity': (1) To allow user to add a new admin user, and (2) Do not allow user to edit information of any existing admins.

4. The sequence of actions in 'MLAViewAdminActivity' are below:
    a. The rendering of the screen is done by the invocation of 'OnCreate' method that sets the view to add a new admin user.
    b. After user inputs are provided for the new admin, click on the save menu option that triggers 'onOptionsItemSelected' to: (1) validate the input fields (e.g. UserName, Password, EmailId, Telephone, etc.), and (2) create an object of action class 'MLAAddAdminAPI' defined in 'MLAViewAdminActivity.java', to add new admin. On execution of this object, 'doInBackground' method is invoked to perform a task, while 'onPreExecute' and 'onPostExecute' methods are invoked to do necessary preprocessing and postprocessing work respectively.
    c. In this case, 'doPreExecute' prepares a client request to be sent to the server.
    d. 'doInBackground' method calls 'addAdmin' method that is mapped to a Web service 'api/Register/PostAddAdmin'. The mapping is specified in the 'APIInterface.java' file under the Web service folder. The Web service returns an object of type 'MLAAdminDetails' that is a model object representing a specific type of application data, i.e. Admin User Details.
    e. 'onPostExecute' method is invoked to check the status. If HTTP status code equals to 202, the execution is successful. Else, error is returned to the user.

The overall workflow of adding an admin user is illustrated in the following figure:

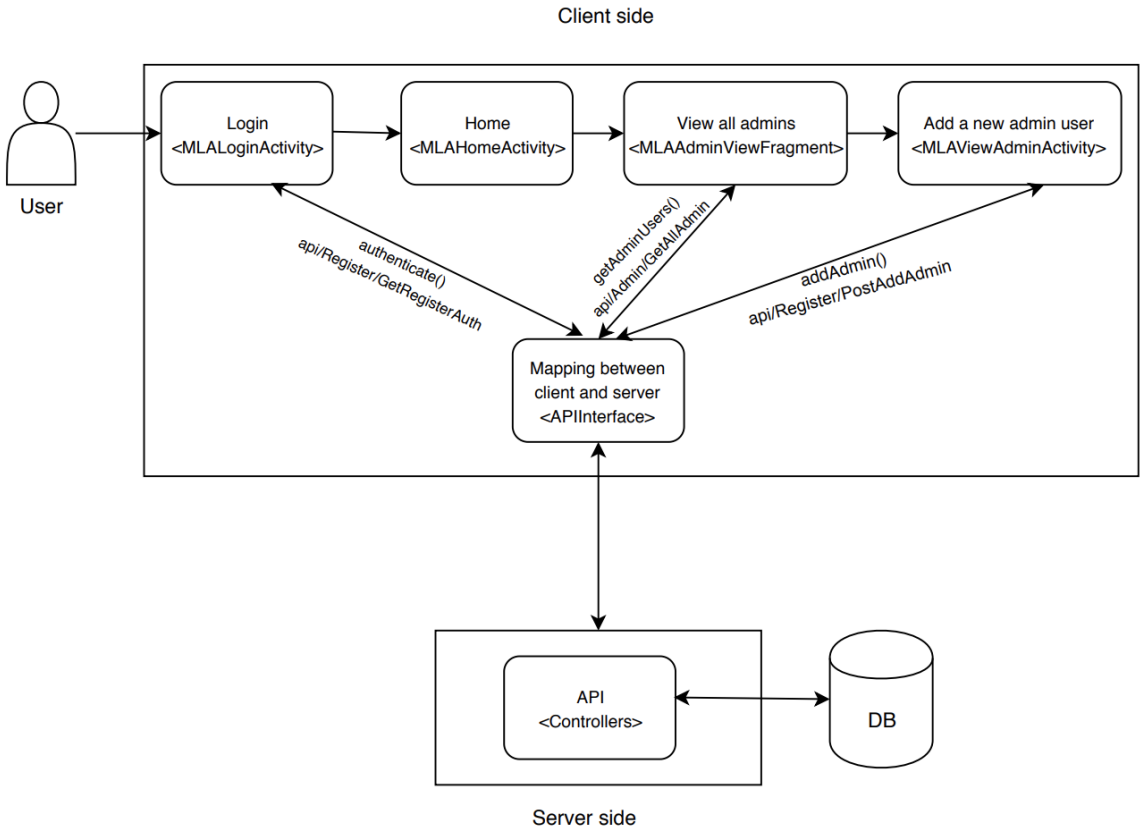

Figure: The Workflow of Adding an Admin User