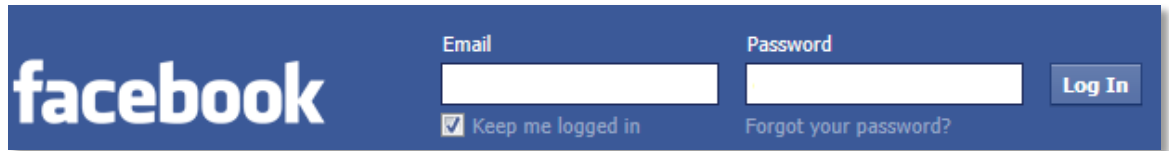


Creating a Facebook desktop application using Facebook SDK and FbGraphApiWrapper (.NET Framework 4, WinForms, Visual Studio)

1. Creating an application-account in Facebook

In order to create an application that communicates with Facebook and acts on behalf of your user, you must first create an application-entity in Facebook:

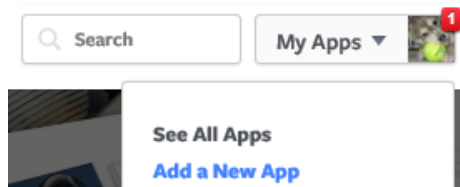
- a. You must have a Facebook account in order to create an application account which will be



created under your Facebook account.

- b. Create an Application Account:

- i. Go to <https://developers.facebook.com/apps>
- ii. Click the "+ Add a New App" button to create a new application account



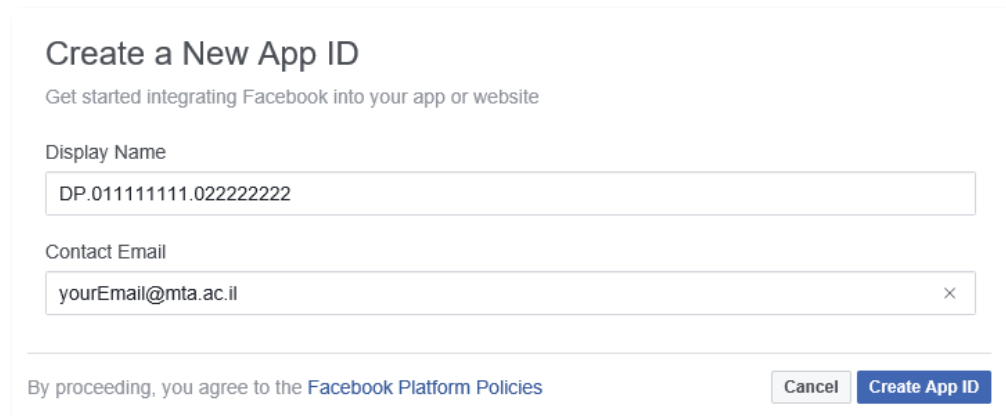
- iii. Give the application a name using the following format:

DP.011111111.022222222

replace 011111111 with the first student's ID number (9 digits!)

replace 022222222 with the second student's ID number (9 digits!)

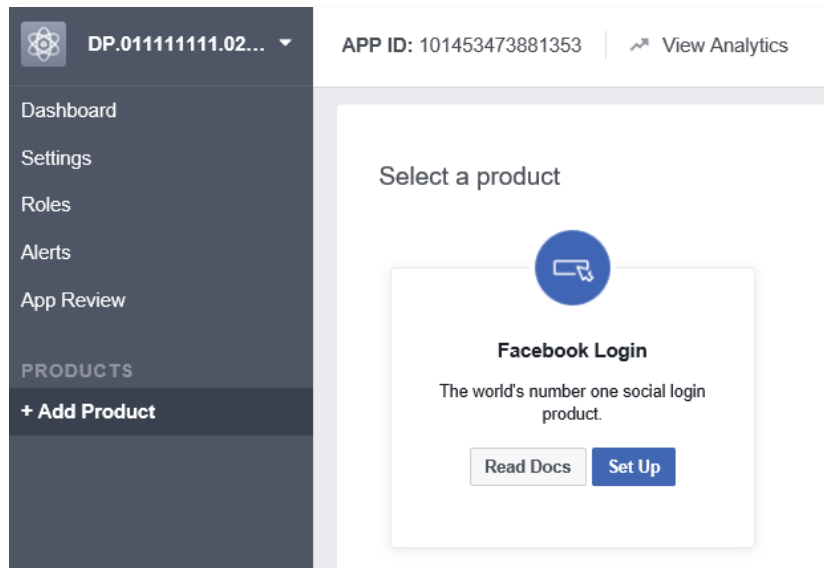
and fill in an email address!

A screenshot of the 'Create a New App ID' form on the Facebook Developers website. The form has a title 'Create a New App ID' and a subtitle 'Get started integrating Facebook into your app or website'. It contains two input fields: 'Display Name' with the value 'DP.011111111.022222222' and 'Contact Email' with the value 'yourEmail@mta.ac.il'. At the bottom, there is a checkbox for 'By proceeding, you agree to the Facebook Platform Policies' and two buttons: 'Cancel' and 'Create App ID'.

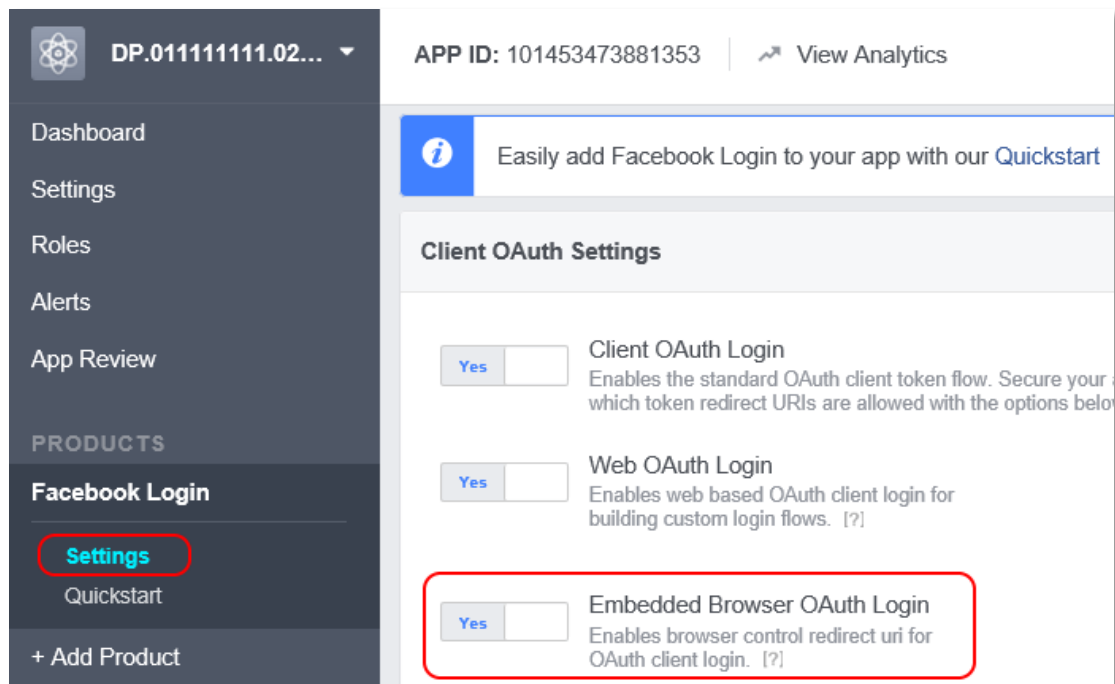
- iv. Enter the captcha and hit 'submit'



- c. Save your newly created **App ID** for later, and hit the '**Set Up**' button in the 'Facebook Login' section:

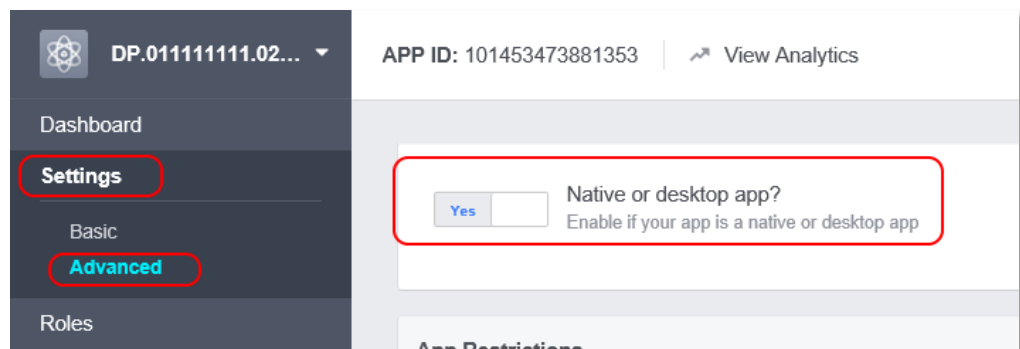


Click on 'settings' under 'Facebook Login' on the left panel, and turn on the 'Embedded browser OAuth Login' option (choose 'yes') (And then hit '**Save Changes**')



Go to 'Settings' tab (left menu):

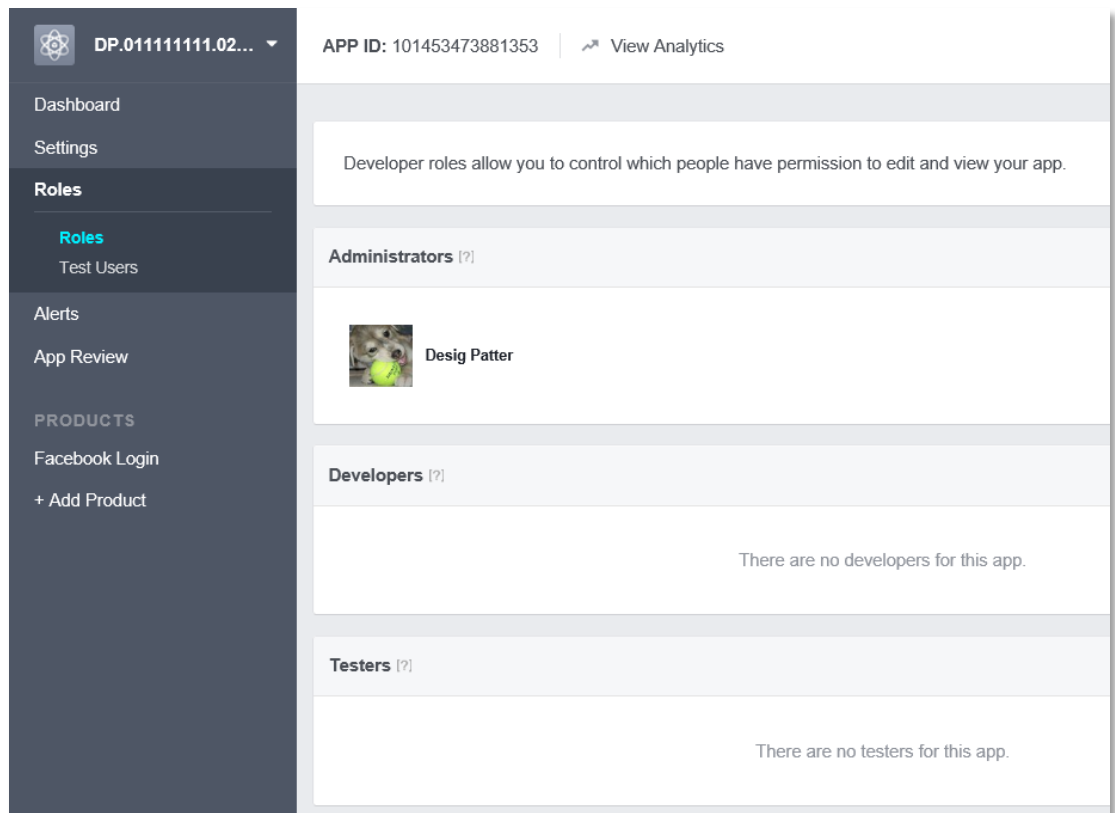
- i. In the '**Settings > Advanced**' section, select the **Native/Desktop** application type



- ii. Hit "**Save Changed**" and go to the 'Roles' tab.

"Roles" is where you can define the "Administrators", "Developers" and "Testers" of your app. Only these users can login through your app until you'll make it available for the general public use:

Add any facebook account to the Administrators/developers/testers list which you want to be able to test your app. **Specifically, add the <http://www.facebook.com/design.patterns> user as a tester** (you'll need to be 'friends' with desig patter in order to add him as tester).



The screenshot displays the Facebook App Developer console interface. On the left is a dark sidebar with navigation links: Dashboard, Settings, Roles (highlighted), Test Users, Alerts, App Review, PRODUCTS, Facebook Login, and + Add Product. The main content area shows the 'Roles' configuration for app DP.011111111.02... (App ID: 101453473881353). A message states: 'Developer roles allow you to control which people have permission to edit and view your app.' Below this, three sections are listed: 'Administrators [?]' containing one user 'Desig Patter' with a profile picture of a dog; 'Developers [?]' with the message 'There are no developers for this app.'; and 'Testers [?]' with the message 'There are no testers for this app.'

Creating a .NET 4 WinForms application using the FbGraphApiWrapper.dll assembly

- d. In Visual Studio, create a new **.NET Framework 4.0** WinForms project.
- e. From the reference folder of your project, add a reference to the .dll files included in the .zip file of the exercise (FbGraphApiWrapper.dll, Facebook.dll).
- f. Use the static login method
`LoginResult result = FacebookWrapper.FBService.Login("272862089537667",`
providing your AppID and the permissions required from your app's user to display a login form to your user.
If this is the first time your user (a facebook account owner) is using your app, he/she will be prompted to approve the permissions requested by your application.
For the list of permission, see this [link](#).
- g. The return value of the Login method (LoginResult) has a LoggedInUser property (of type FacebookWrapper.ObjectModel.User) which you should use in order to utilize your user's data and actions, in an object-oriented fashion, for example:
 - i. Data:
user.FirstName, user.LastName, user.Birthday, user.RelationshipStatus, etc.
 - ii. Relations to facebook objects:
user.Friends, user.FriendLists, user.Checkins, user.WallPosts, user.Events, user.Albums, user.Pokes, user.Videos, etc.
friend.FirstName, friend.LastName, friend.Albums, friend.Checkins, etc.
album.Photos, checkin.Comments, photo.Comments, photo.Tagged, photo.LikedBy, etc.
 - iii. Actions:
user.PostStatus(), user.PostPhoto(), user.CreateAlbum(), user.CreateFriendList(), etc.
album.UploadPhoto(), photo.Comment(), photo.Like(), status.Comment(), etc.
- h. If the user failed to login or simply closed/canceled the login dialog, the result object will indicate the error with the ErrorMessage property of the LoginResult object.
- i. The return value of the Login method (**LoginResult**) also has a AccessToken property which holds the AccessToken your app got in the Login process. You can save this accessToken for future use (save it to a file/DB) for connecting with facebook in regards to the logged-in user.
Use the static 'Connect' method, the AccessToken you got in the Login process, like such:
`LoginResult result = FacebookWrapper.FBService.Connect(theAccessToken);`
result.LoggedInUser will hold the User object with the logged in use data.

2. Resources:

- a. Visit <https://developers.facebook.com/docs/reference/api/> to understand more and get all the information about the Facebook Graph API
- b. Use the <https://developers.facebook.com/tools/explorer> application to browse data on facebook using the Graph API and understanding Jason
- c. The .zip file contains Class Diagrams of the object-oriented wrapper API (.png image files and .cd files which should be viewed in Visual Studio). They are also here on the next pages.
Use them to learn more about the structure of the API (note: These class diagrams are not complete)
- d. The **ReleaseNote - READ ME!!.txt** file contains interesting information regarding the changes made throughout the different versions of the API. You may find this information useful

