For this tutorial we'll be briefly going over fragments as well as how to register users through Parse.

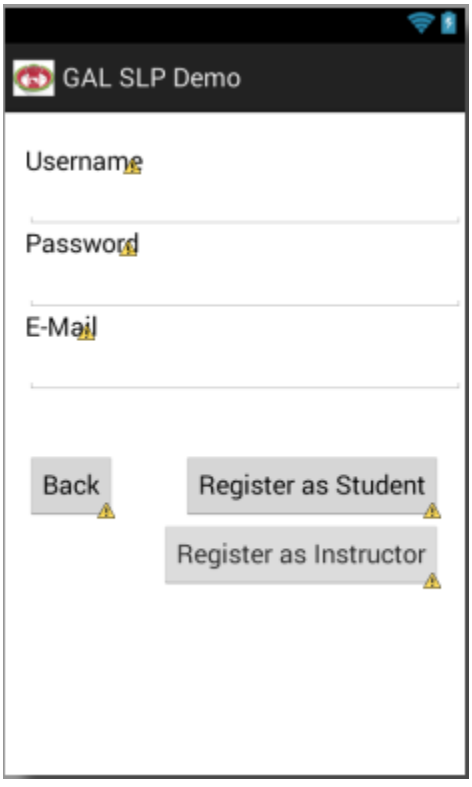
I use a fragment to do the registration screen so all you have to do to get to the login is to commit a new fragment which I do in the following code which is called when the register button is pressed.

```
public void register(View view) {
	FragmentTransaction transaction =
	getSupportFragmentManager().beginTransaction().replace(R.id.container, new
	Register());
	transaction.addToBackStack(null);
	transaction.commit();
}
```

On the registration page, to go back to the previous fragment just switch it out again

```
public void back(View view) {
	FragmentTransaction transaction =
	getSupportFragmentManager().beginTransaction().replace(R.id.container, new
	PlaceholderFragment());
	transaction.addToBackStack(null);
	transaction.commit();
}
```

My registration page looks something like this. You can turn on email verification in the settings on the webpage for your app.



Below is the code for a registration button. The first chunk finds the EditTexts. The next chunk gets the Strings from each EditText.

The third is where it gets interesting. It creates a new ParseUser and sets all of the parameters. In addition to the basic ones that Parse provides us, I also created a new field called level which defines what type of a user you are. 0 is for students, 1 is for instructors. This works because a ParseUser is a subclass of a ParseObject and so it inherits all of the functions of a parse object as well including the ability to create and define new fields.

After that we set our global variables to log the user in after a successful registration. If the registration is successful then it logs the user in based on the username and password they provided. If it isn't, it calls the login failed method because all the login failed method does is show a toast that you messed up somehow. You can also look at the ParseException to actually figure out what happened but I won't go over that in this lesson.

```
public void asStudent(View view) {
	//get the EditTexts
	EditText et_username = (EditText) findViewById(R.id.editText1);
	EditText et_password = (EditText) findViewById(R.id.editText2);
	EditText et_email = (EditText) findViewById(R.id.editText3);

	//get the text
	String username = et_username.getText().toString();
	String password = et_password.getText().toString();
	String email = et_email.getText().toString();

	//build the user object
	ParseUser user = new ParseUser();
	user.setUsername(username);
	user.setPassword(password);
	user.setEmail(email);
	user.put("Level",0); //differentiates between instructors and students

	// other fields can be set just like with ParseObject
	//user.put("phone", "650-253-0000");

	this.username = username;
	this.password = password;

	//sign up
	user.signUpInBackground(new SignUpCallback() {
		public void done(ParseException e) {
		    if (e == null) {
		      // Hooray! Let them use the app now.
		       registerSuccessful();
		    } else {
		      // Sign up didn't succeed. Look at the ParseException
		      // to figure out what went wrong
		       loginFailed();
		    }
		}
```

```
		});
	}

	public void registerSuccessful(){
		ParseUser.logInInBackground(this.username, this.password, new
LogInCallback() {
			public void done(ParseUser user, ParseException e) {
				if (user != null) {
					//The user has logged in
					loginSuccessful();
				} else {
					loginFailed();
				}
			}
		});
	}
```

Congratulations, you're done! In the next lesson I'll show you how to use the ParseObjects to populate a list. Namely a master detail view.