**Lecture 1:** Git Intro

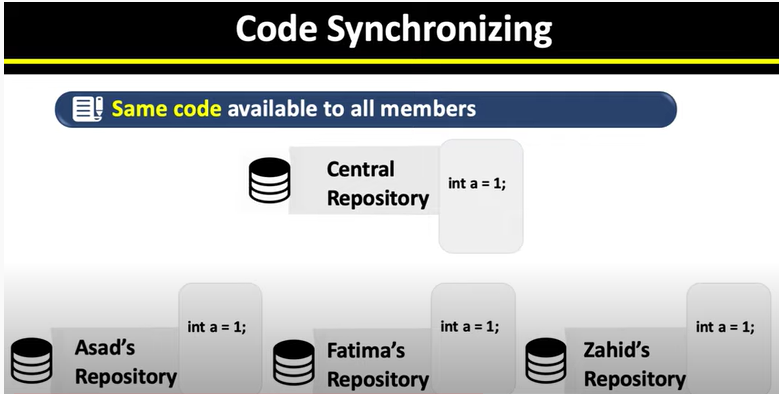
VCS: Version Control System is like keep tracking of all changes to code.

For example, if we write some code like

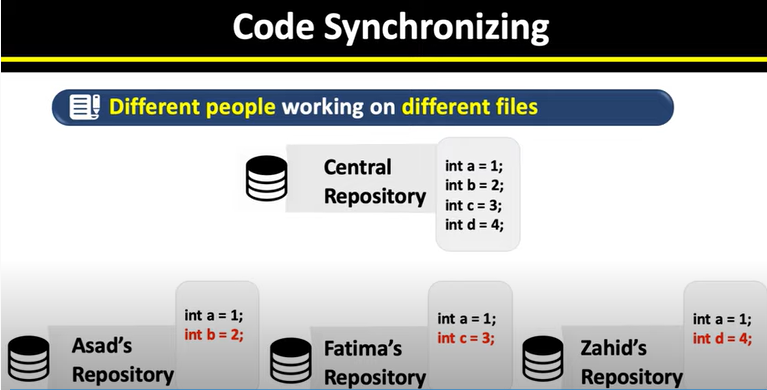
**File:1**

1. **Created file-1**
2. **Write some code:**
3. **Int x=12;**
4. **Int b=x+1;**
5. **After that, next time we require some change in the value of x let suppose we wang to make it 12 so we’ll do some changes like  
   (i) int x=~~12~~ 13;  
   (ii) int b=x+1**
6. **Now we want to keep track of change we just did in the code. Therefore, we require a *version control system.***
7. **Best way to keep track of all changes is to make an account on GitHub.**

**CODE SYNCHRONIZING BETWEEN DIFFERENT GROUPS:**

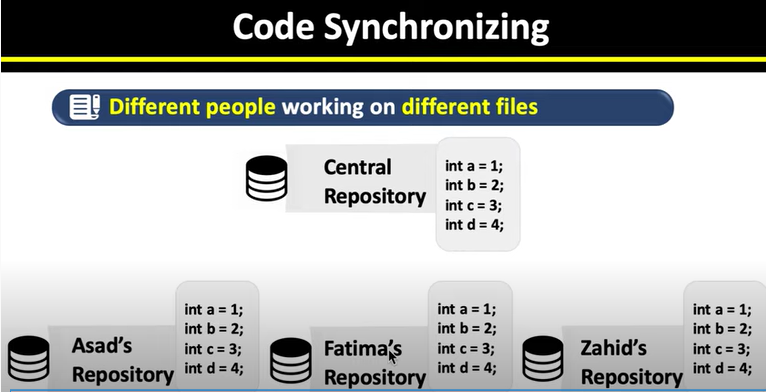


As you can see above there are three group members each have their own code.  
Also, we have a central repository.   
In VCS, we put all our code’s int this central repository so that everyone will have the same updated version of the same code.



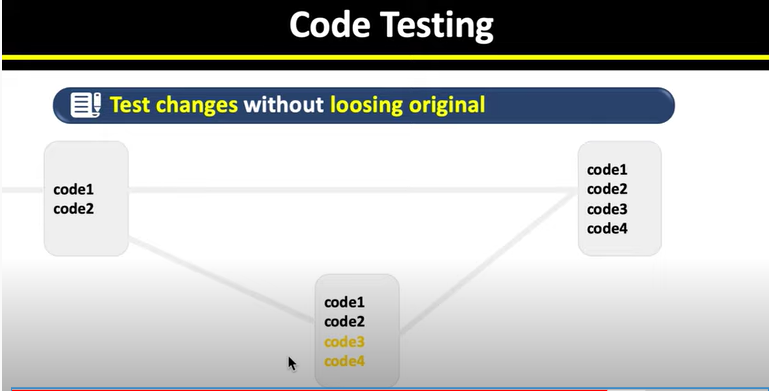
**As we can see, here we have all the updated code in out central repository.**

**Now everyone will have the same code/updated code in their systems.**



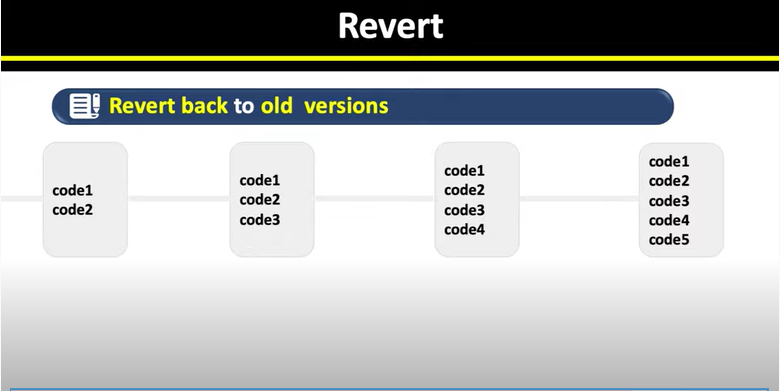
**Everyone now have the same code/updated code in their own systems.**

**TEST CHANGES WITHOUT LOSING ORIGINAL**



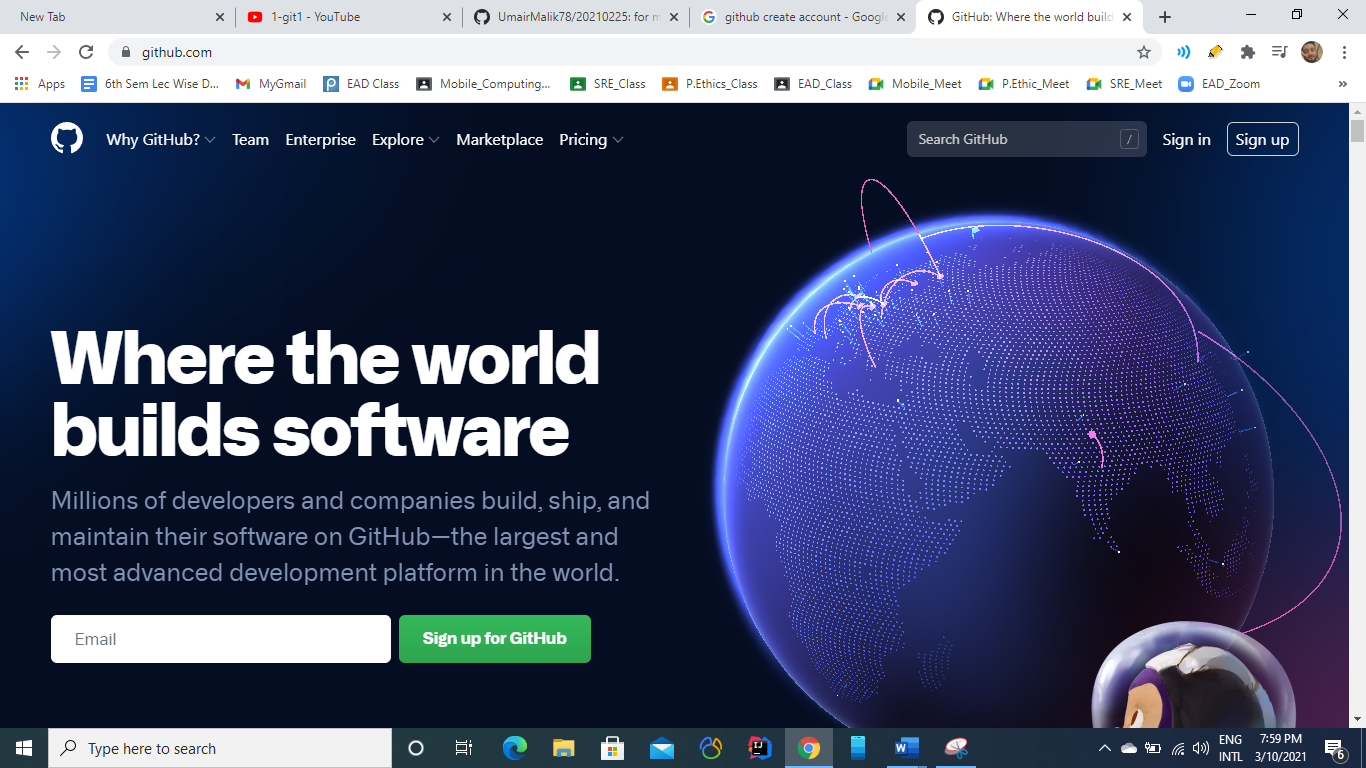
**Until we are not sure about the new added code, we can keep our previous code.  
After that, when we have tested our new code, we can add it to the original as shown in above figure.**

**REVERT BACK TO OLD VERSIONS FACILITY:**

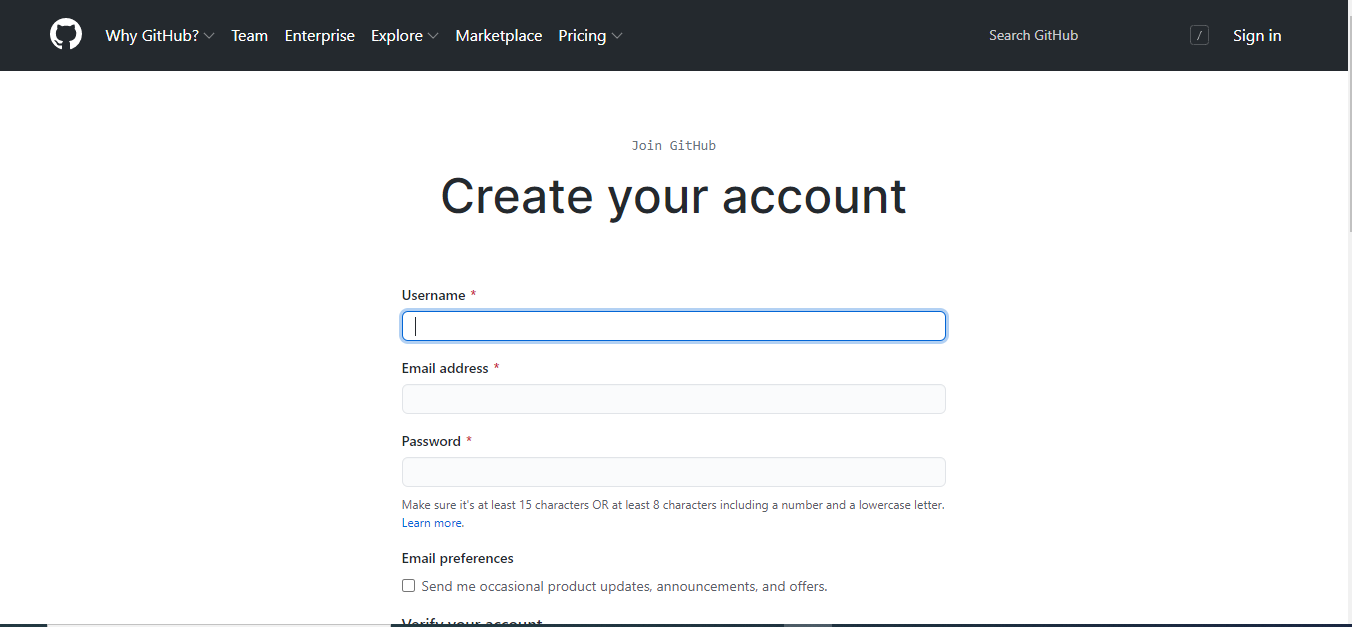


**We can also look back at our code or code history and can find revisions, upgradations, deletion etc easily.**

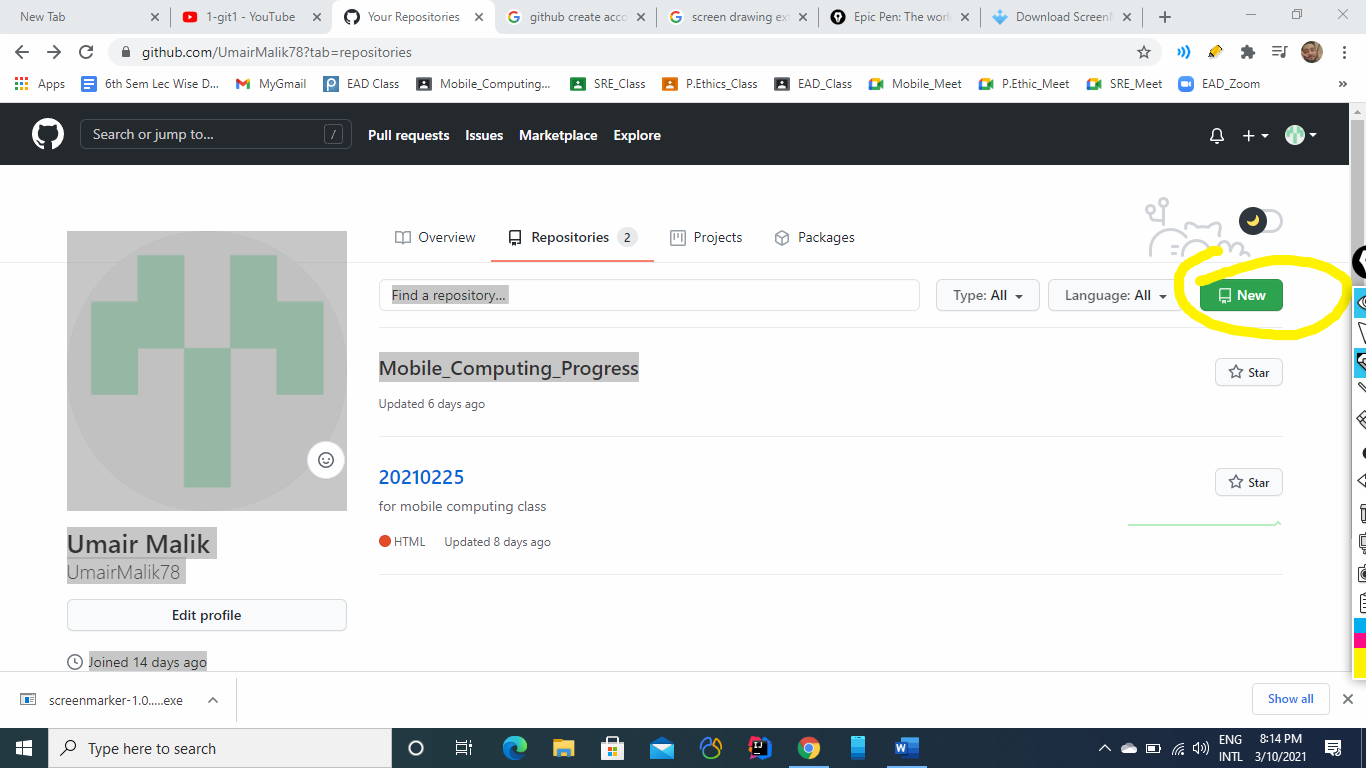
**CREATING ACCOUNT ON GitHub.com**



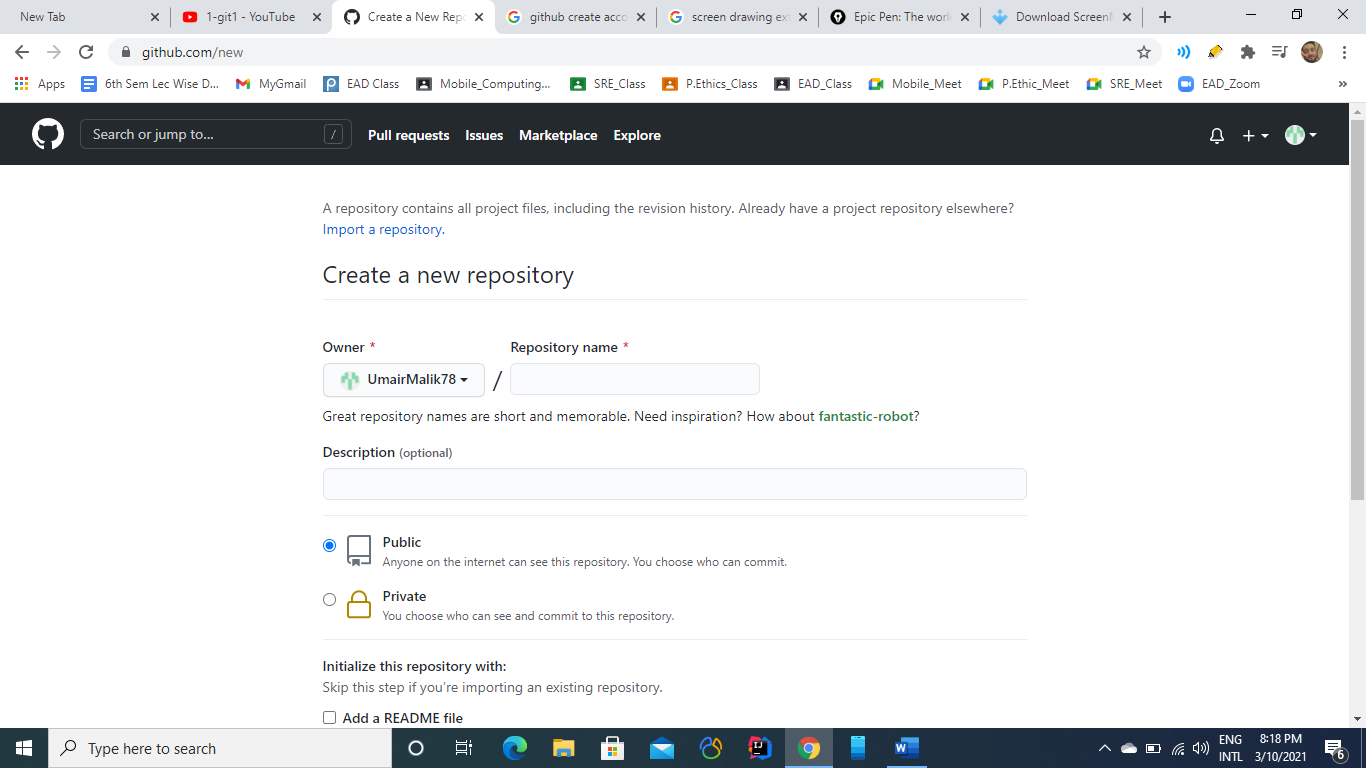
From there select ***sign up for GitHub.***

You’ll find a new page like this. 

Fill out all necessary fields with respective data and Select **Create Account** button.



From this screen creeate on **new button**  to create a new Repository.

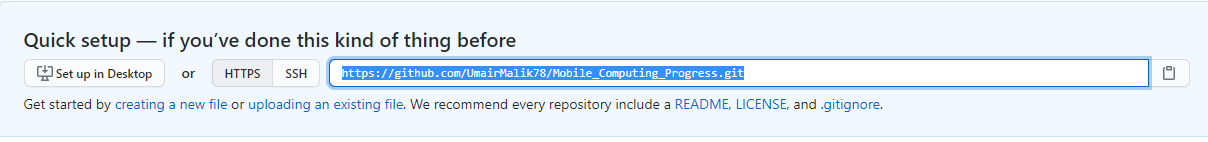


From this menu, Wrtie a new Repository name which weill be of couurse unique wuth reference to you id.

You can also make your repositry as ***Private*** or ***Public***:

**Private**: If you make private, then noone elese can make change to it

**Public:** In this case, everone will be able to see and make changes to your work.



This is the URL of your new Repository, which we’ll use ;ater to work with it.

**Lecture No 2**

**Git Clone:**

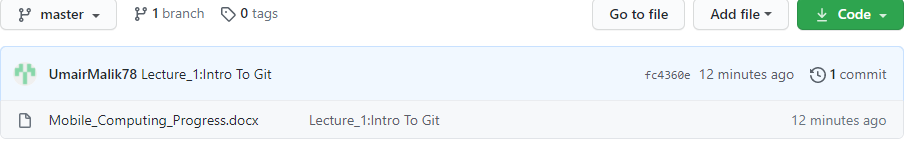
1. Git Clone means to cone the central repository into our ocal repositroy
2. For this we use command  
   ***git clone [url]***

Where URL is the url of our repository.

1. When we clone a repos, a folder/repos is created at our system locally.

**How to add files to central repository:**

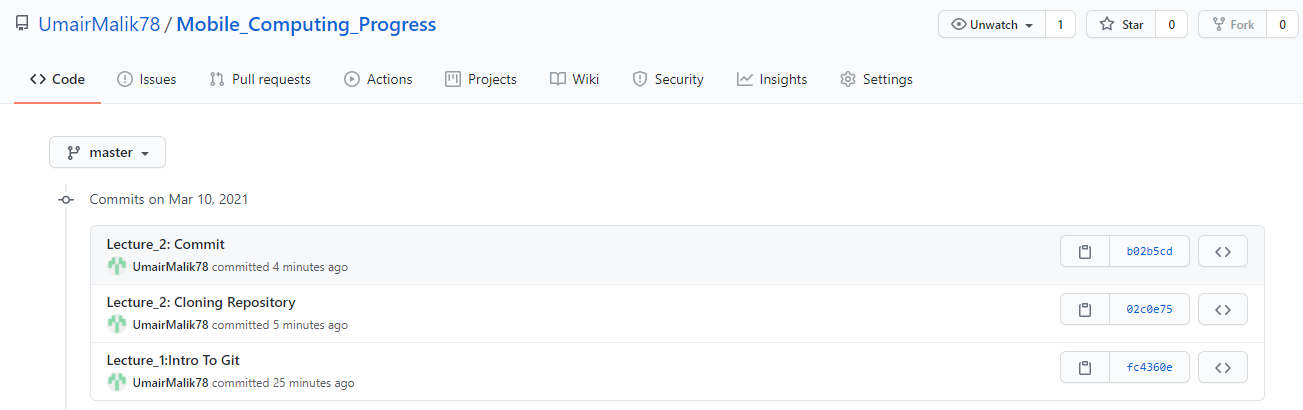
1. **Place file into the local repository:**
2. Create File into the local repository we created with the help of ***git clone.***
3. Let suppose we create a file named as **Mobile\_Computing\_Progress.docs**
4. **git add *filename***(i) Add file in the local repository.  
   (ii) git add **Mobile\_Computing\_Progress.docs**
5. **git commit**  
   (i) It is used to make changes with message to track changes.  
   (ii) git commit -m “First Commit”
6. **git Push**  
   (i) It is used to finally add your work on the server.  
   (ii) git Push



After above steps, we’ll see our file located on the server of GitHub.com

Suppose we want to make changes to file.

1. We first add new changes onto the file
2. After that we’ll add file again using ***git add filename.***
3. Then we’ll make commit.
4. At the end we’ll run push command to make newly file available on the server.



As we can see new commit has been added to the server.

To see changes made online we’ll use

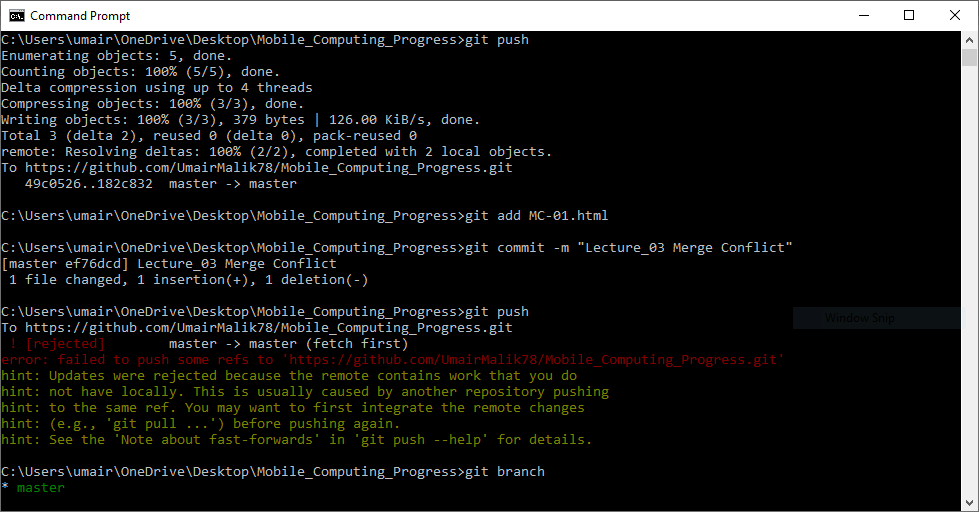
**Git Pull:**

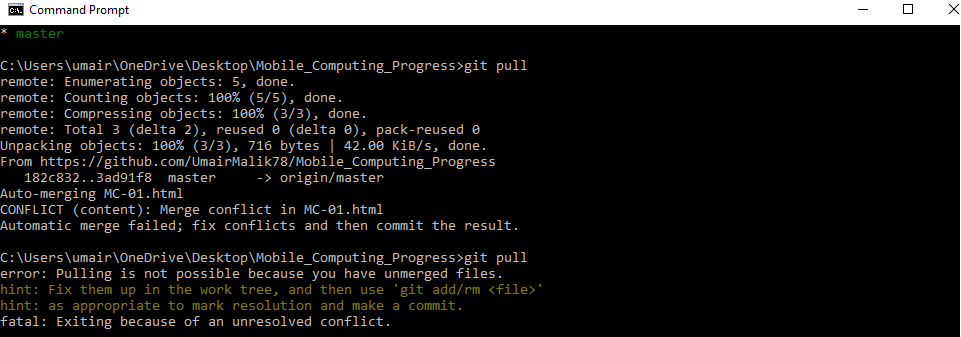
This command is used to get changes from GitHub to our local repository.

**Lecture No 3**

**Merge Conflicts:**

Merge conflict occur when we try to update or change the same line of code at two different points i.e., locallay as well as on main repo.  
Git become confused that which should be save and which should not



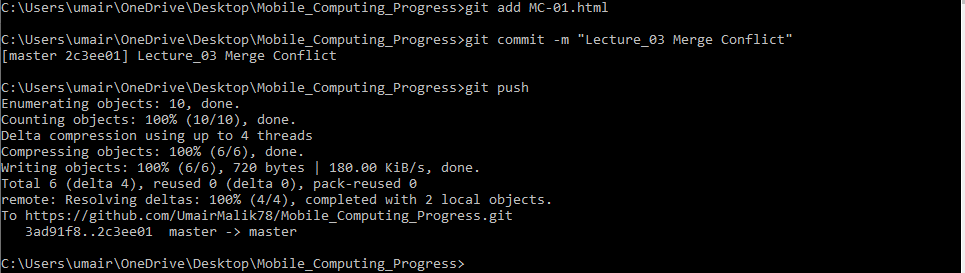
When we try to push our code to main repo it gives us error to first pull the changes made at web  
So we try to pull them first.  


On Pulling, we get CONFILCT ERROR.

**Because same line changed locally is changed at web so we have to make choice which we want to retain.**

Suppose we want to retain changes made at web so we select that incoming changes.

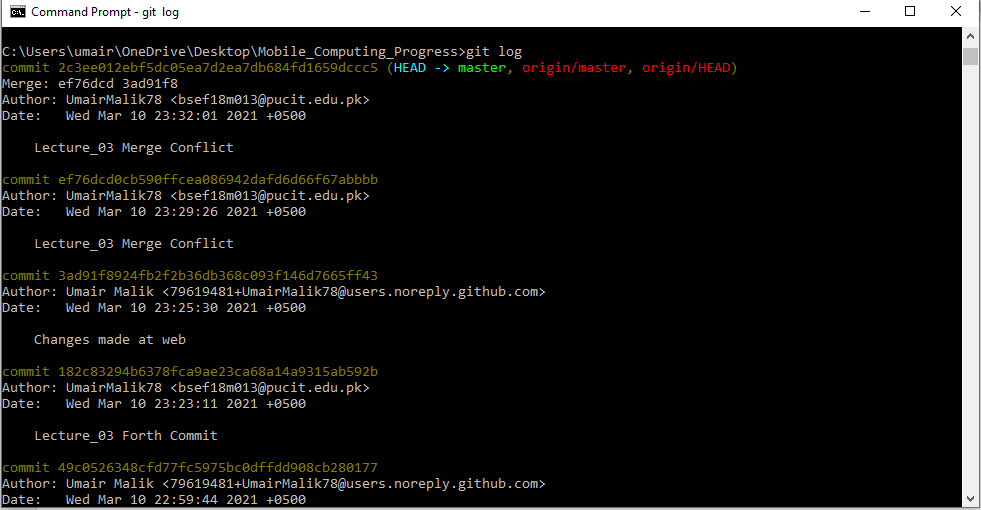
Now when we try to push our code, it will be successfully pushed into the Main Repo or Server.



**So in this way, we get the conflict resolved.**

**Git Log:**

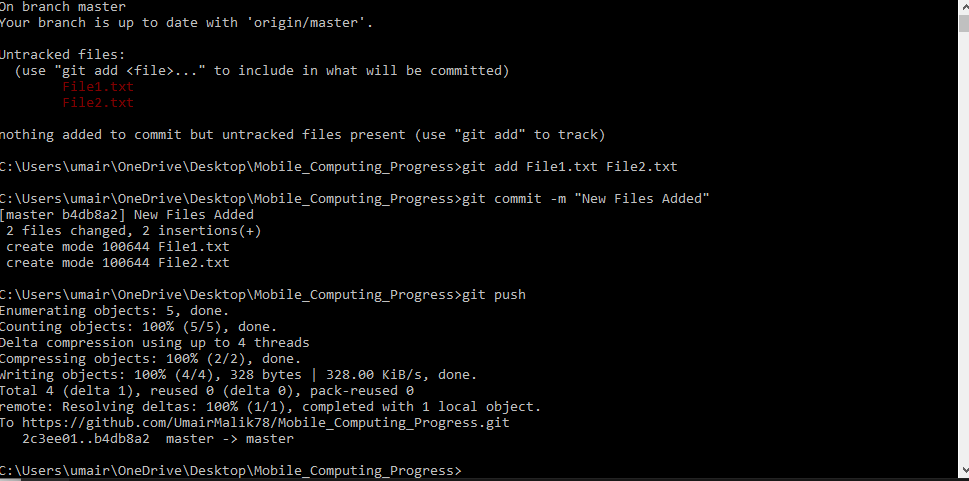
Git Log is used to get the details of commits we make in our repository so far.



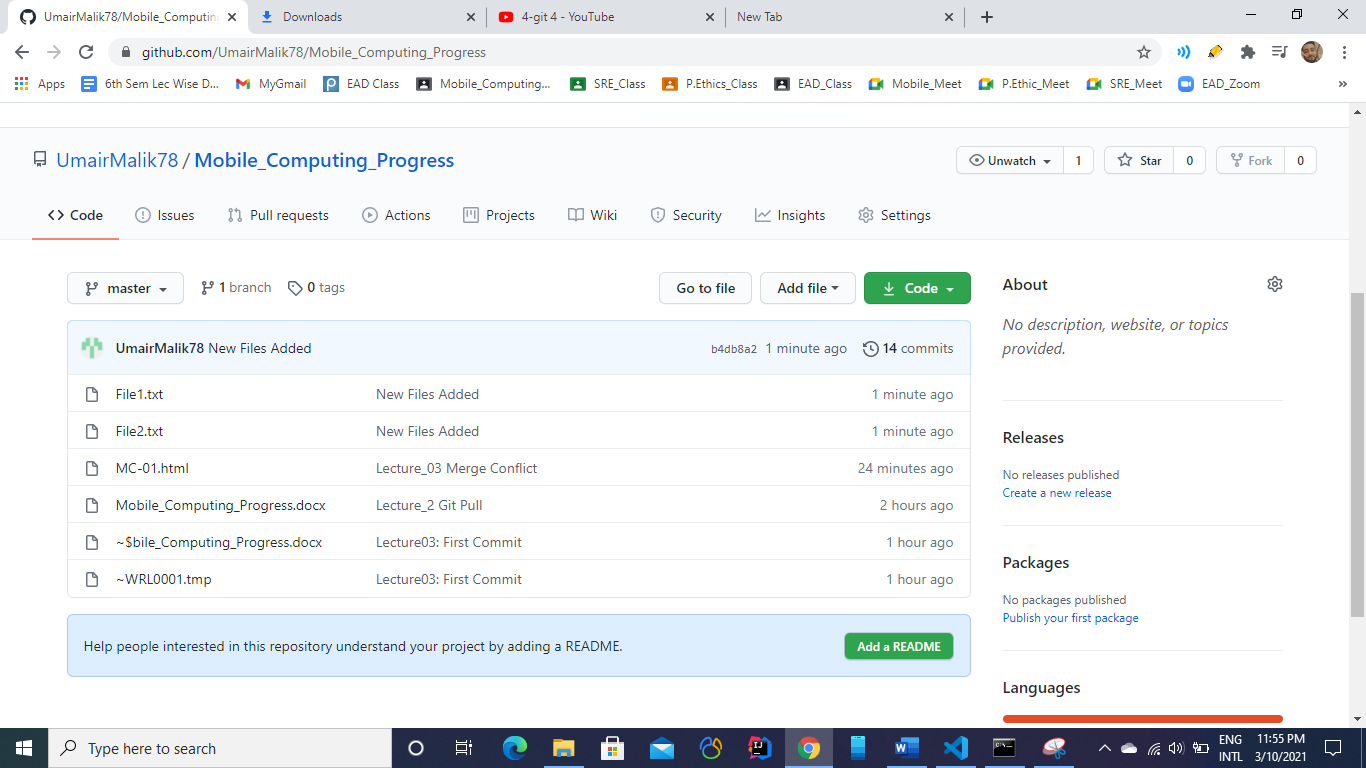
**Lecture No 4: Branching**

**1. REMOVING FILES USING GIT:**

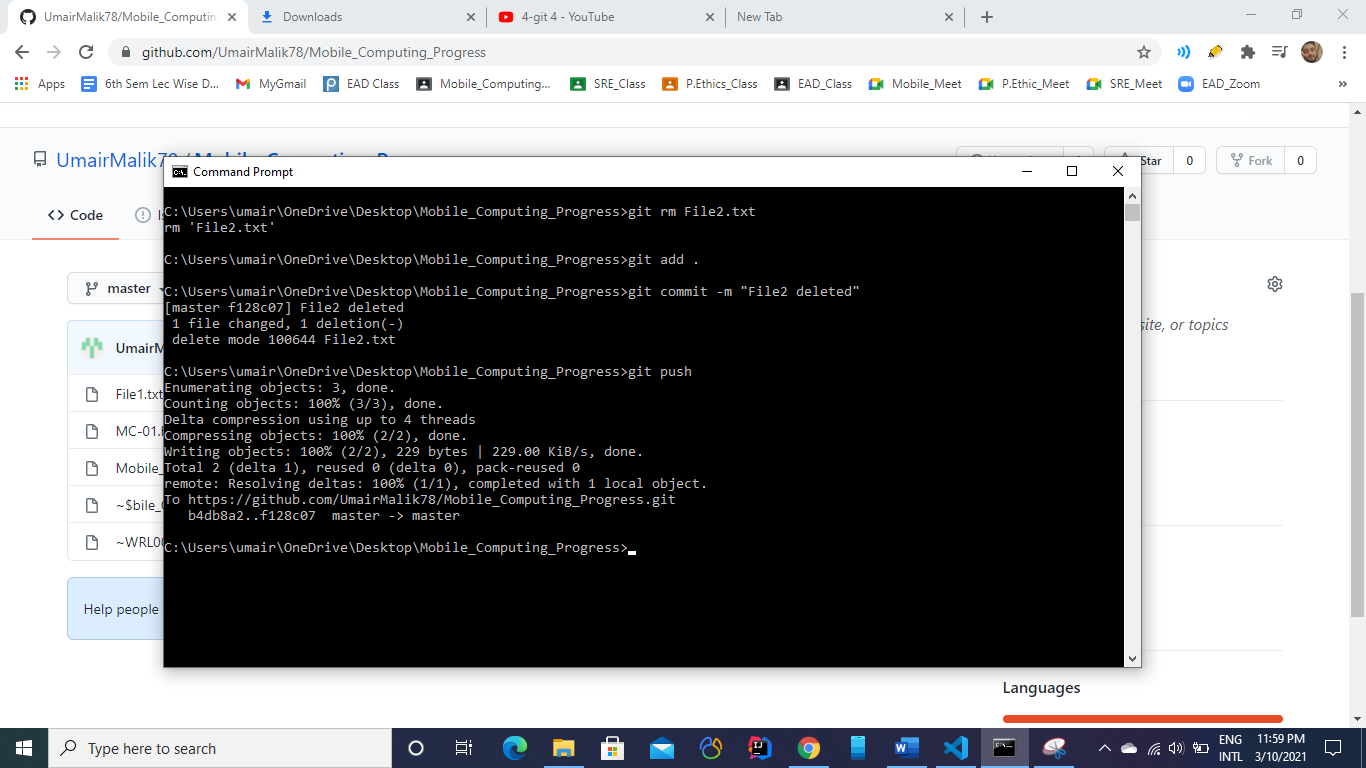
1. First we’ll add teo new text files into our repository using add commit and push method.



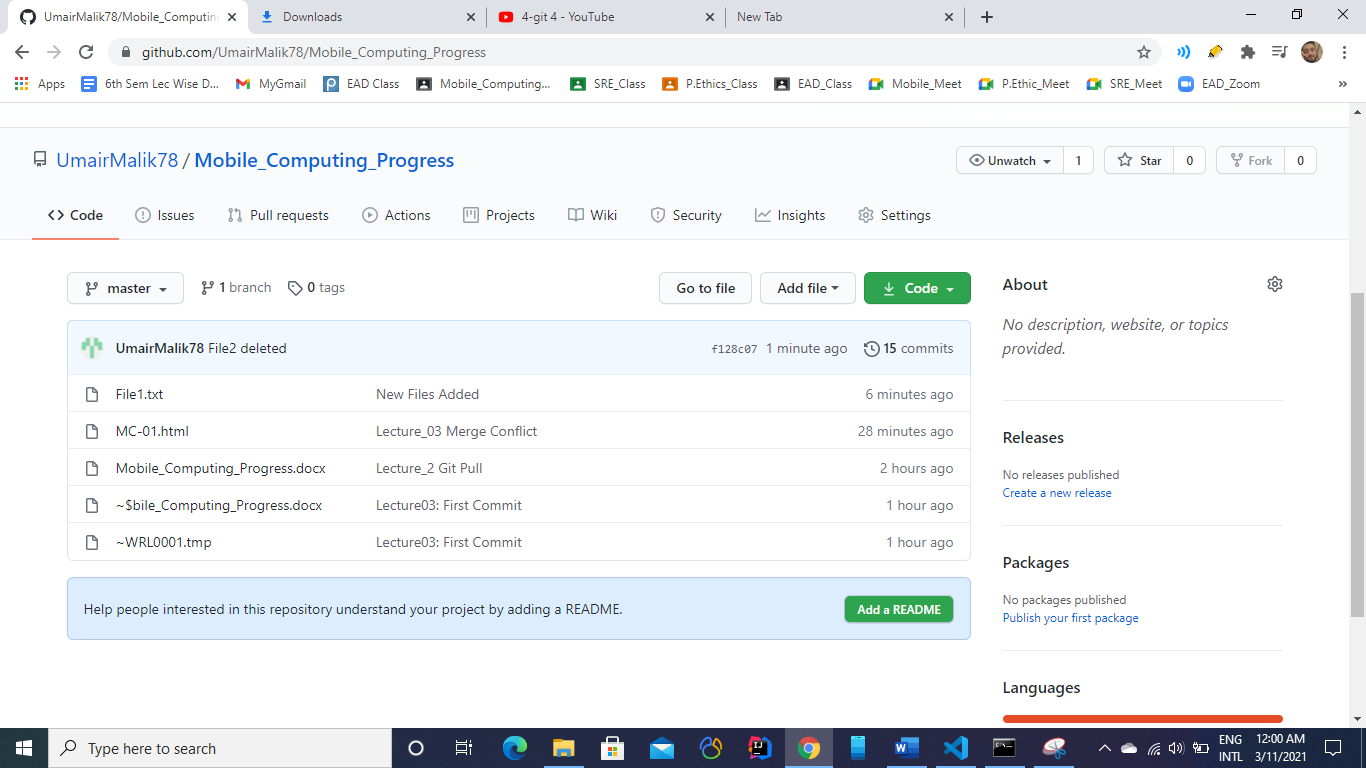
1. Now this will be available on server also as you can see.



1. Now we’ll delete file 2 using git
2. First we’ll remove file from our local repo.



1. Now we’ll see changes at the server.



As we can see file 2 has been removed successfully.

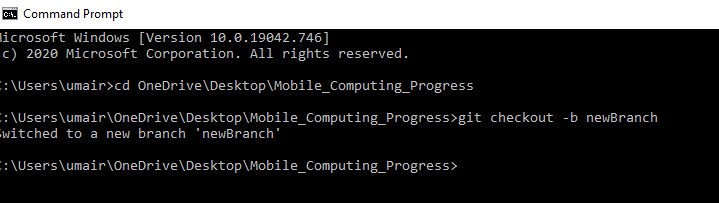
**2. BRANCHING:**

**Different comands for branching:**

1. **Git branch**
2. **Git check**
3. **Git merge**

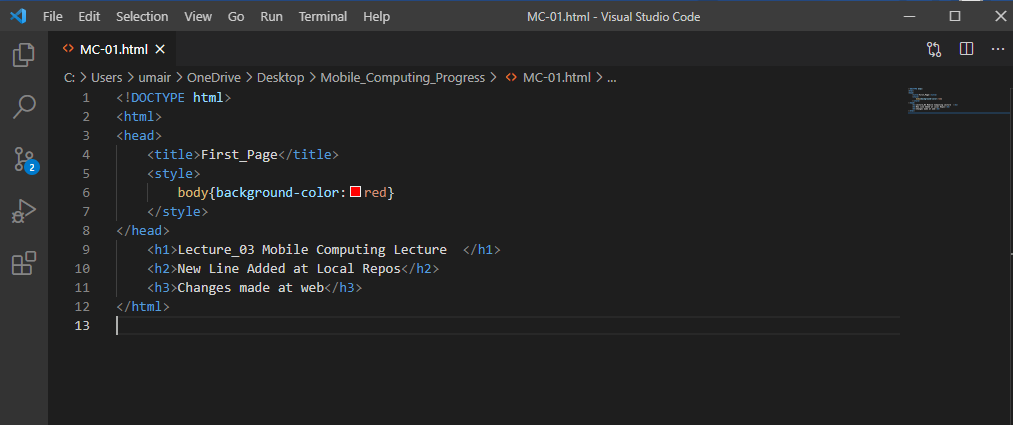
**Create a Branch:**

1. New branch is made with the following command  
   git checkout -b *branchName*

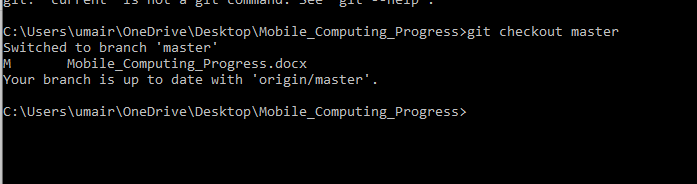


1. Now all of our work is saving in newBranch.
2. As soon as, we switchd to master branch, previous code will be appear,i.e., code we write in the previous branch.

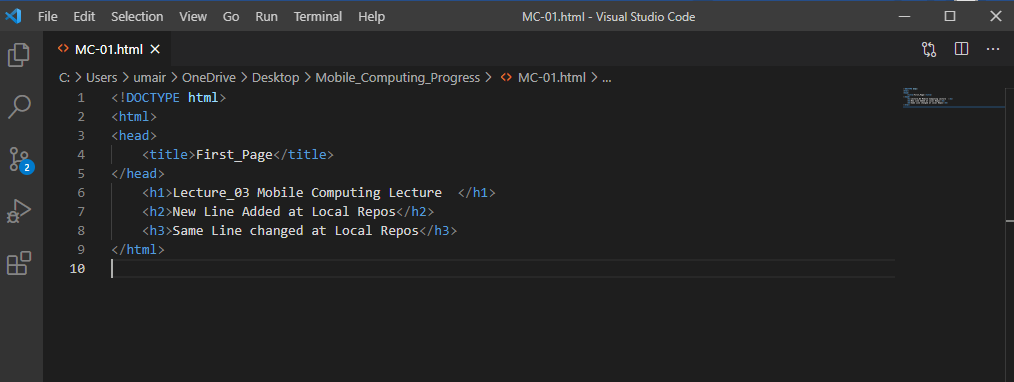
Suppose we are on ***NewBranch*** and we write some piece of code in MC-01.html



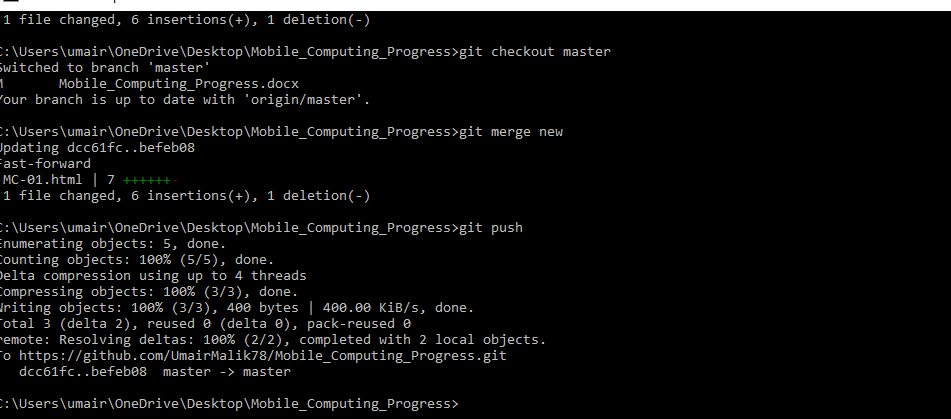
Now we switched to the Master branch:



As soon as we shift to it, previous code will be shown in IDE.



After when we have tested our new additon/new code then we can merge them both with the help of following command  
 ***git merge branchName***



When we have merged our newBranch code with master, then we have no futher need it so we’ll delete it

***DELETING BRANCH:***

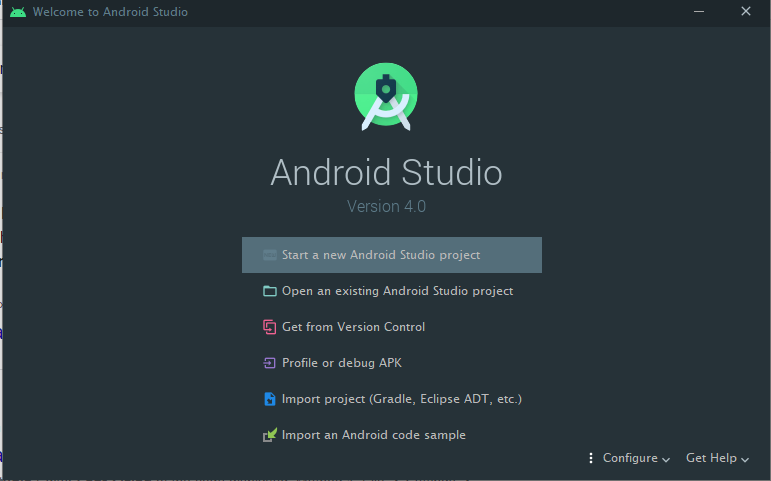
To delete a branch simply type:

***git branch -D “branchName”***

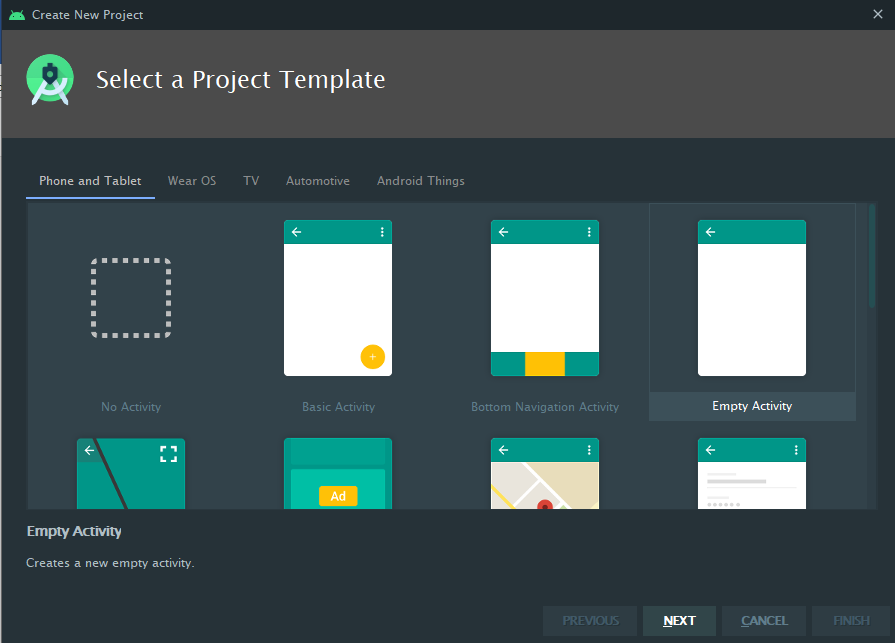
**Lecture No 5: Android Studio**

**First Program: Hellow Worlld!!**

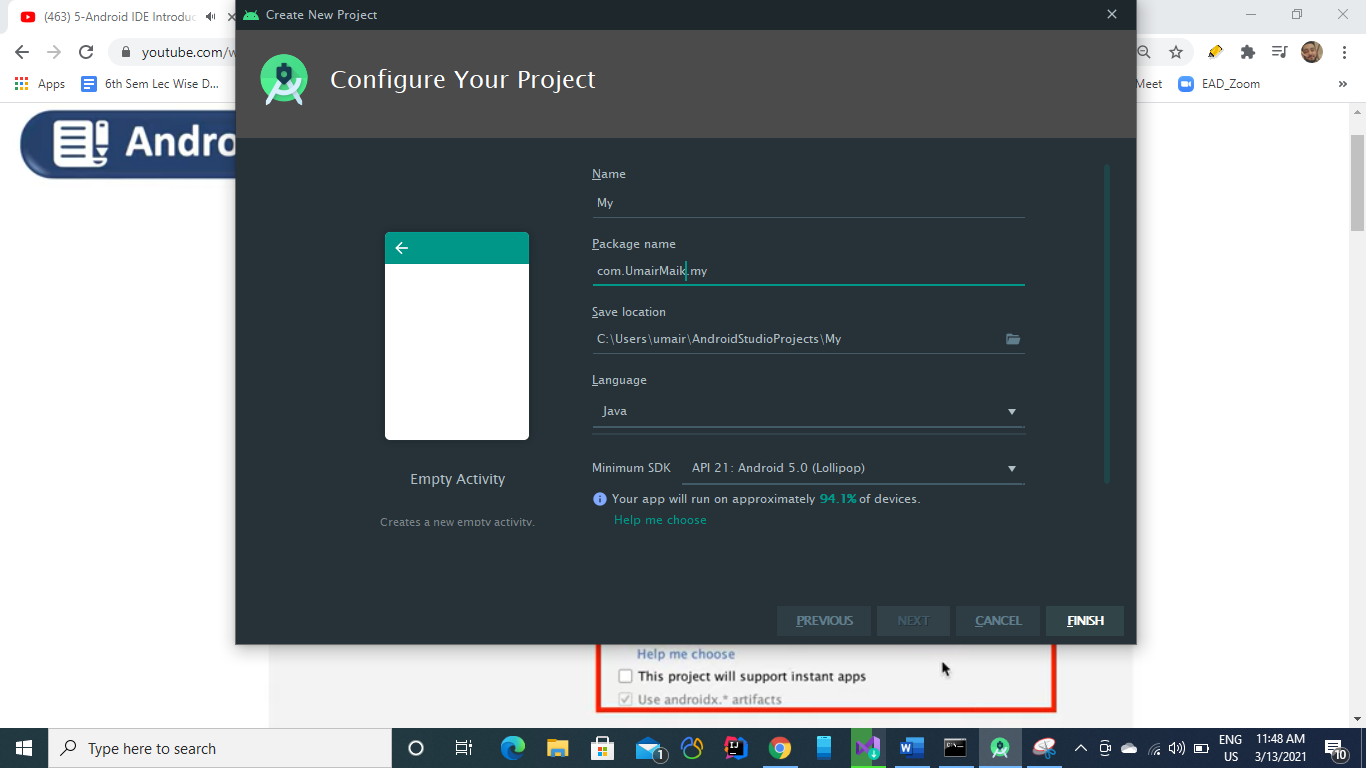
1. Open Android Studio on your laptop

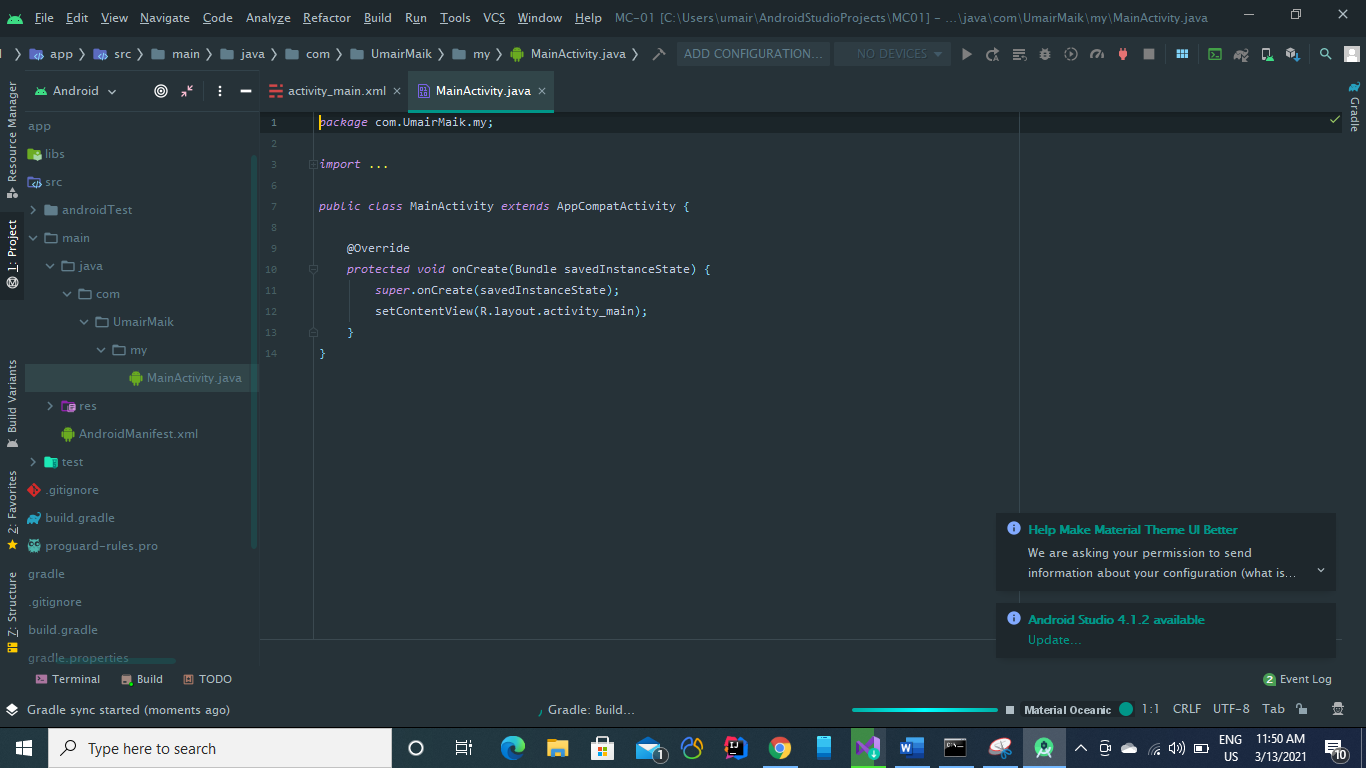


1. Select Start a new Android Studio Project and choose Empty Activity.



1. Fill the required fields.

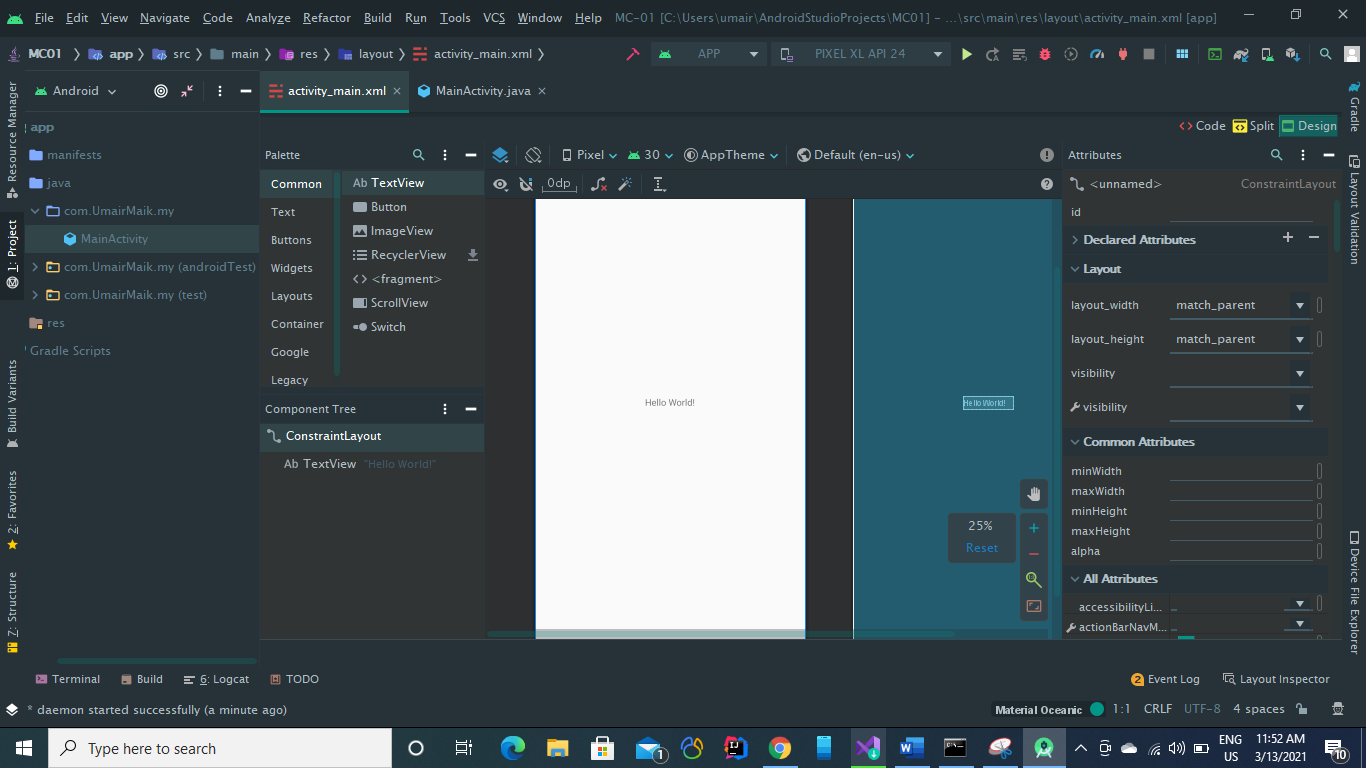




You have two tabs

**Main Activity:** All code will be paste here including variabes logics functions.

**Activity\_main.xml:**

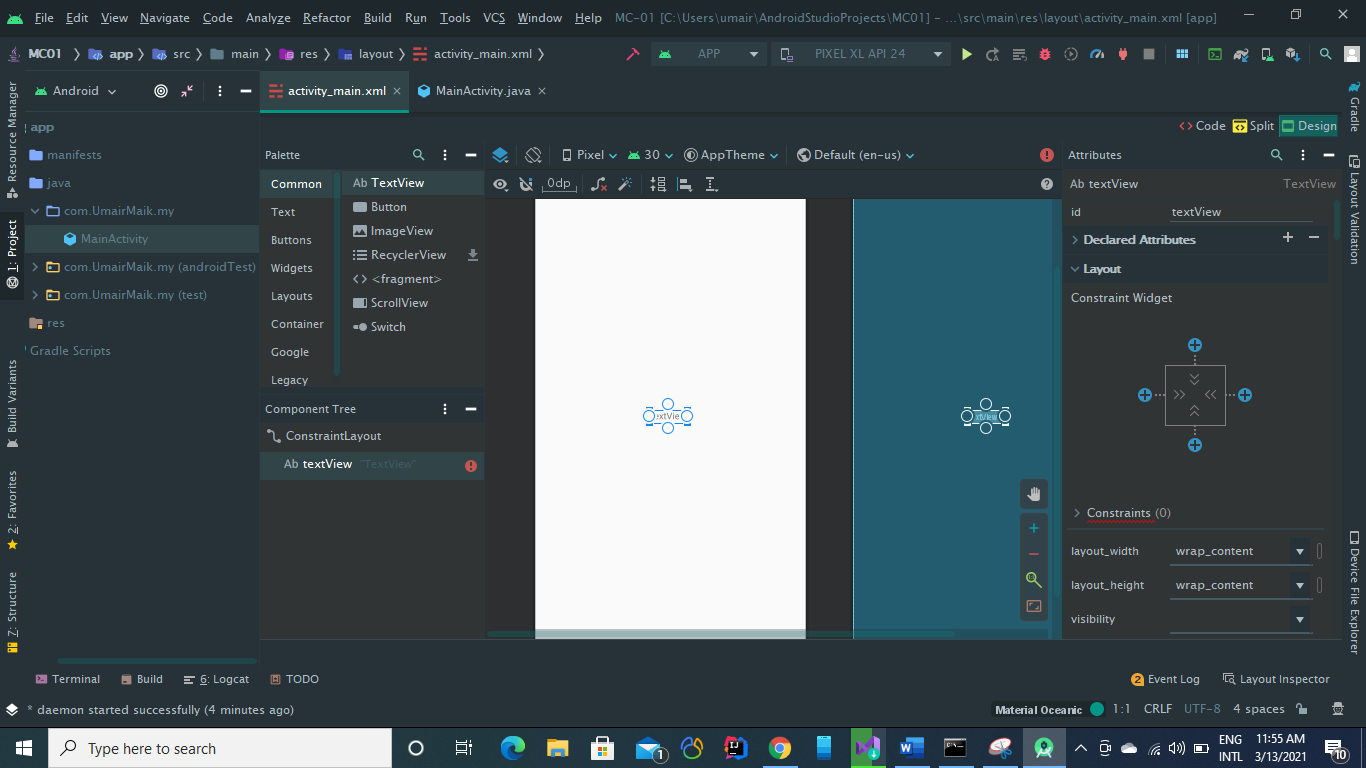
It is for GUI.   


From activity\_main.xml we can add buttons, text fields inpiut fields or different GUI options in our application

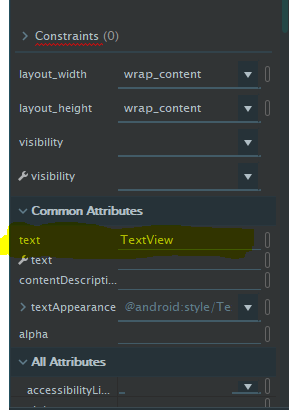
1. Adjust margins
2. Apply constrains etc

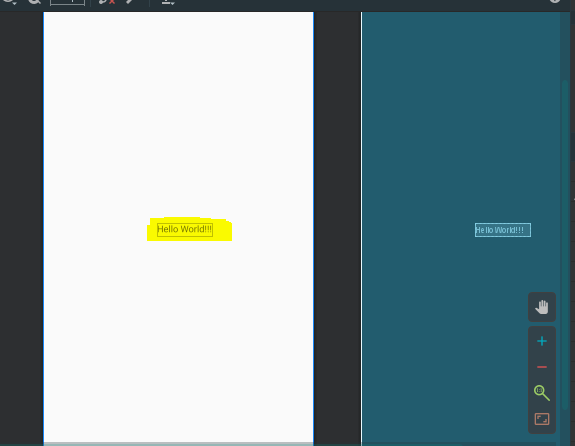
**Writing Hello World in our first app:**

1. Select textview option from Pallete tab and drag a textView into the white screen available in activity\_main.xml

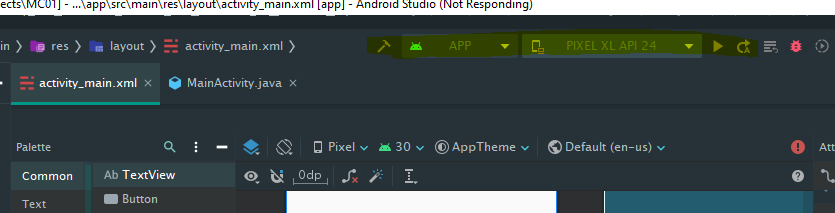


1. TO add text “Hello World!!!” see at right window and under Common Attribute set new text.



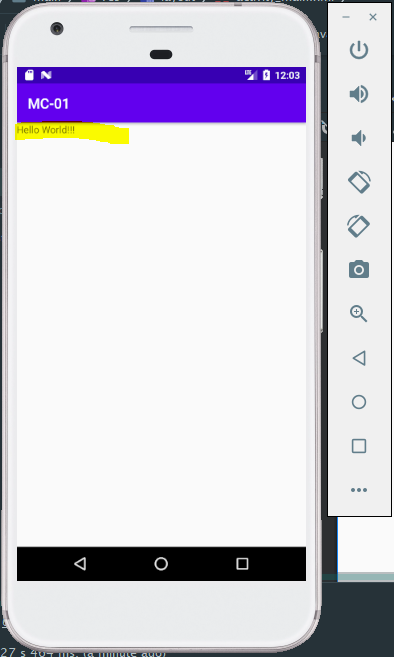
Text will be now shown in white scrren text box  


1. To see thr above scrren or to run the app , select your android virtual device first and then select click on Run.



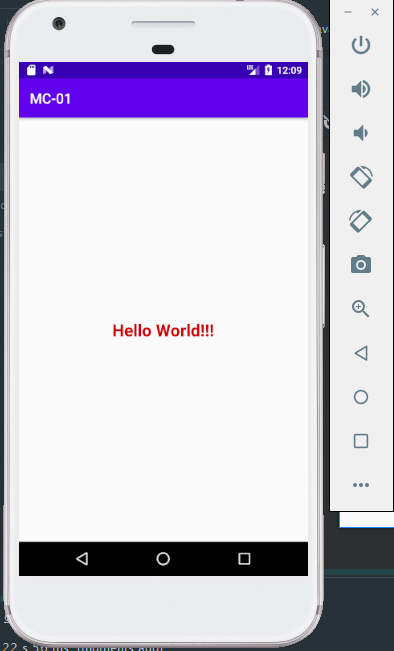
**ANDROID VIRTUAL DEVICES**

1. You can download your android virtual devices from from ***AVD Manager***. Under **Tool>>AVD Manager**.



So this is our first Hellow Workd Application

* We can also set
* contraints
* positions
* text color
* text size etc  
  **From Blue print screenand Attributes tab.**



After some formatting.

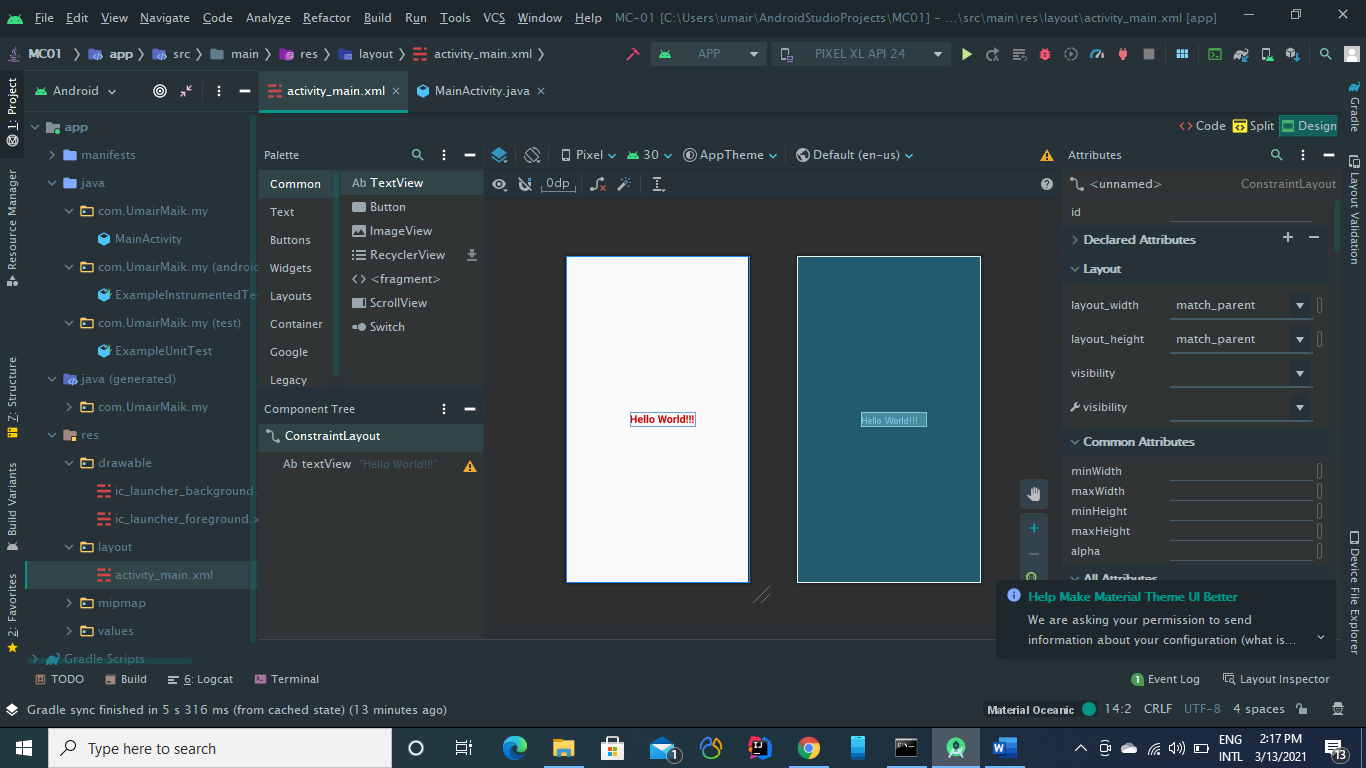
**Resource>>Values**

In this folder, we define values globally to use in the entire project.

You can manipulate from both options

Either in the text view as well as in the design view.

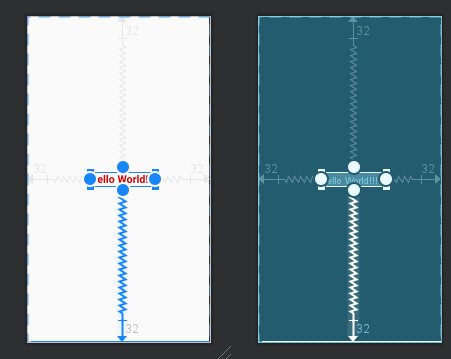
**For example, we can first anipulate text using deign view:**



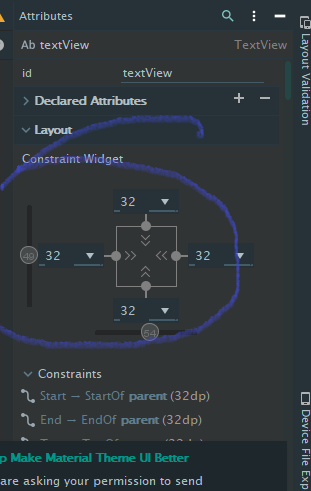
**Changes we made in the design view will also appear in the code view/text view as you can see below**



We can set constraints using the design window by just with use of mouse drage options.

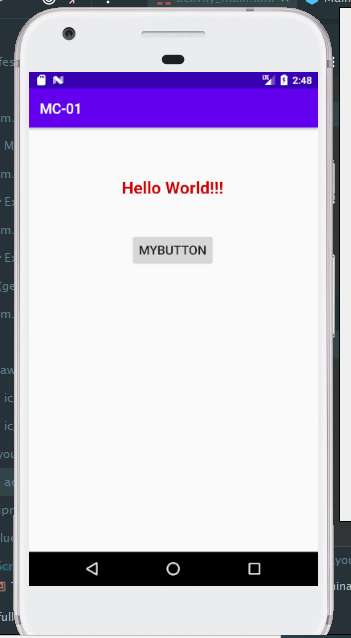


We can also set margins to the boxes or options from the right side of design window.

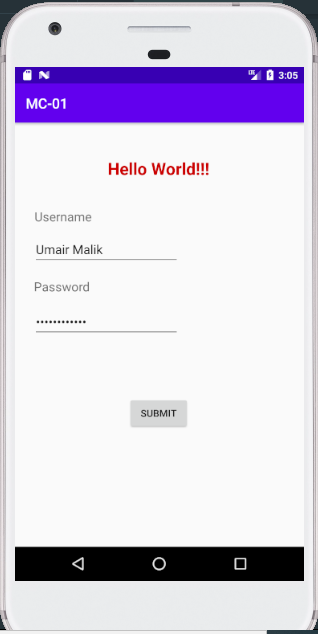


**Different Screens Using Android Studio Development**

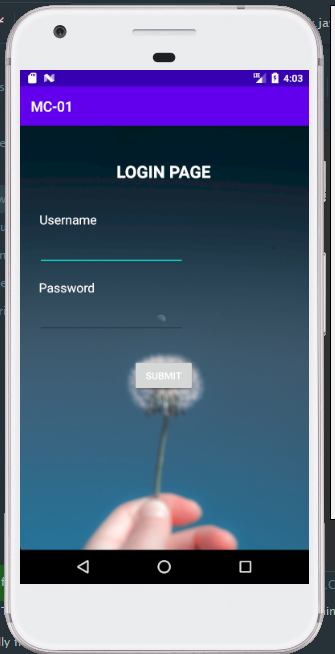
**Screen 1:**



**Screen 2:**



**Screen 3:**

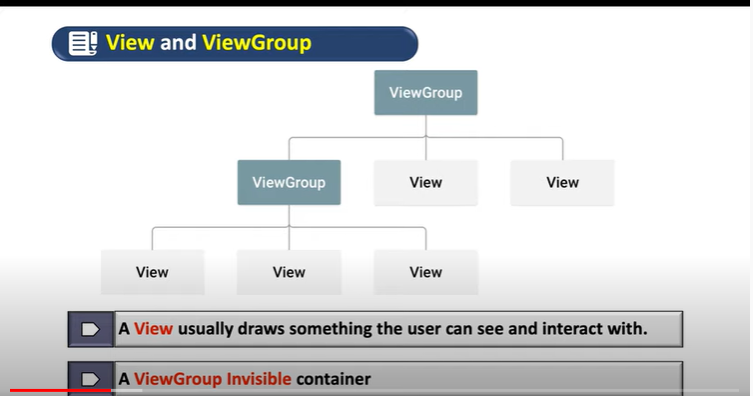


**Lecture\_05 Layouts in Android Studio**

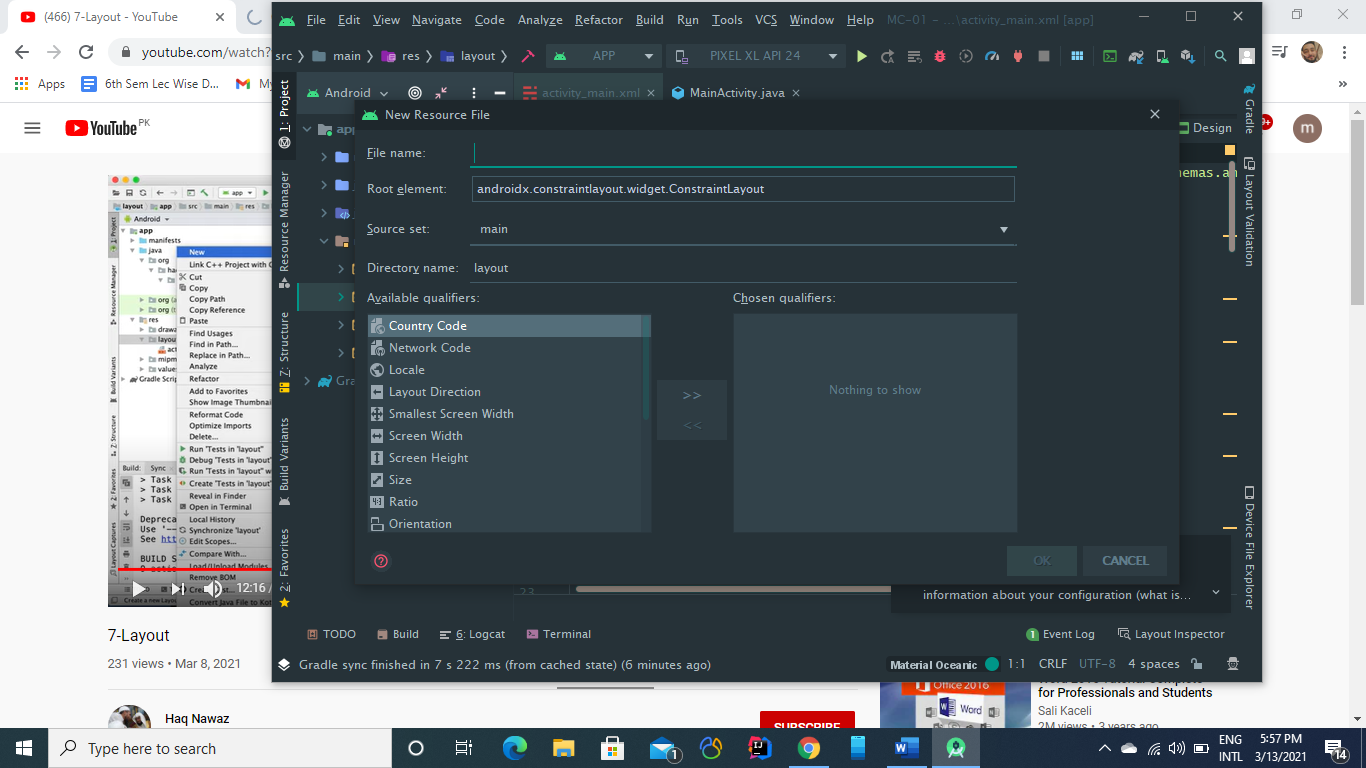
**What is View?**

A view is anything on the screen through whcich user can intercat with and can see. It is usually called as “***Widgets”.***

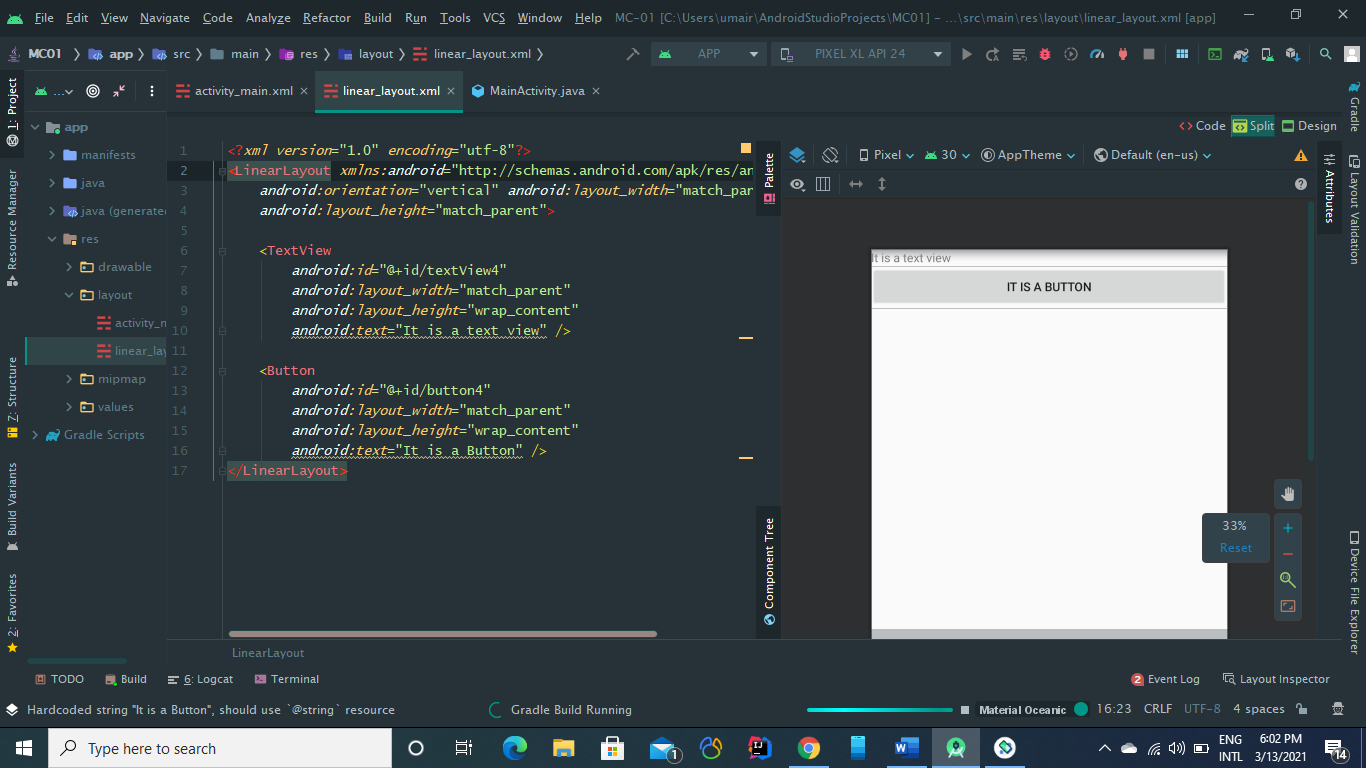
**What is ViewGroup?**

A viewGroup is like a container which contains different views in it. It is ususally called as “***Layouts***”. 

**Adding New Linear Layout:**

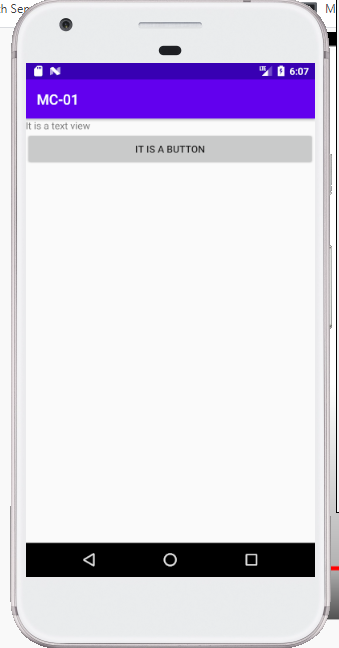
1. Go to res>>Layout folder
2. Right click on folder and select add new layout Resource File.
3. A new window will appear lke this
4. 

Now we’ll add some widgets like a button and a text view in the layout.



We also have to load the layout in the main\_activity.java which tells the compiler from where it should start.

By default, It is set to ***activity\_main.xml.***

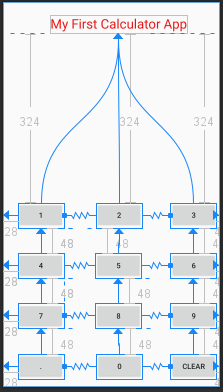


As we cann se, in this layout all elements come in sequential order or in linear form.

**Constraint Layout:**

Suppose we want to add a textview and a button and want them to align then we’ll use our contraint view in this regard.

Some Contraint Layout applied in the image below

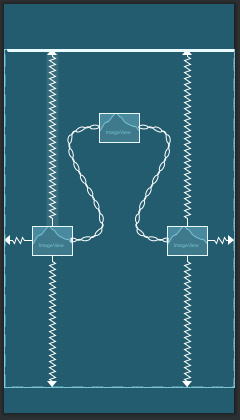


Result of above contraints is shown here:



We can also use chains

Chains are used in constraints to make the distance from one view to another fixed.  
In chains Biasness doesn’t work as you can see in the image below.



No matter in which orientation we go, the distance from right and left view, will remain same.