Pir Mehr Ali Shah

**Arid Agriculture University, Rawalpindi**

*Office of the controller of Examinations*

**Final Exam (Theory)/ FALL 2020 (Paper Duration 24 hours)**

**To be filled by Teacher**

Course No.: SE-540 Course Title: Software Construction & Development

Total Marks: 20 Date of Exam: February 19, 2021

Degree: BS Software Engineering Semester: 5th Section: A (Morning/Evening)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Q. No.** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Marks  Obtained/  Total Marks |
| **Marks**  **Obtained** |  |  |  |  |  |  |  |  |  |  |  |

|  |
| --- |
| **Total Marks in Words:** |
| **Name of the teacher: Muhammad Azhar** |
| **Who taught the course: Signature of teacher / Examiner:** |

**To be filled by Student**

**Registration No.: 18-ARID-3002…….**

**Name: …………….AIMAN BIBI ………**

**Answer the following questions.**

**Q.No.1. (Marks: 6)**

**Briefly describe the concept of MVC architecture in terms of software construction and explain:**

* 1. **How MVC works for web-based applications**

**ANSWER:**

The Model View Controller **(MVC)** is an architectural pattern. It separates an application into three components:

* The model,
* The view
* The controller

**The Model**

It is a place where all data is stored related to the application in separate models. This is the place where data from controller and view is actually passed into, out of, and is manipulated. It is responsible to check the data against the stored information in database and then to supply it according to the sequence.

**The View**

It is the representation of the user-interface in a web-based application. Buttons, forms and other information visible to the user on the web, these are all part of View. View is implemented when controller call the view after having interaction with the appropriate model.

**The Controller**

The Controller plays a role of the traffic cop of the application in web environment. It handles incoming HTTP request. It is also directing traffic to where it should go and then identifying which view needs to load up and is interacting with the appropriate models.

This kind of web architecture allows us to identify and tackle application related issues effortlessly. The separate nature of MVC makes it easier to manage complexity of large applications easily.

* 1. **Importance of middleware and it’s usage**

**ANSWER:**

* Middleware provides services and capabilities to applications outside of what is offered by the operating system. It can encompass everything from web servers to authentication systems to messaging tools.
* It can support modern and popular runtimes for use cases.
* It helps developers to build applications more efficiently. It acts like the connective tissue between applications, data, and users.
* Middleware can help to make it cost-effective to develop and run applications at scale for organizations with multi-cloud environment.
* Some middleware services are accessed through [APIs](https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces), these APIs are sets of tools, definitions, and protocols that allow applications to communicate with each other.
* Some managements and services are handled by middleware such as; Data management, application services, messaging, authentication, and API management.
* This software lies between an operating system and the applications running on it.
* Middleware enables communication and data management for distributed applications.
  1. **Provide basic Implementation for CRUD operations in Laravel Framework and demonstrate MVC application lifecycle.**

**ANSWER:**

CRUD stands for Create, Read, Update and Delete which are operations needed in most data-driven apps that access and work with data from a database.

**Step 1: Download Laravel 5.8**

First of all we have to download laravel on our system. For this we have to go to our command prompt and go to the folder in which we want to download. After this we have to run "composer" command. After run "composer" command. You have to run following command.

composer create-project laravel/laravel=5.8 crude --prefer-dist

This command will make **crud** folder and download Laravel 5.8.

**Step 2: Mysql Database connection**

Now we have to make Mysql database connection with Laravel.   
We have to find **.env** file in Laravel folder. Open that file define mysql database configuration.

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.0

DB\_PORT=3306

DB\_DATABASE=testing

DB\_USERNAME=root

DB\_PASSWORD= -----(blank)

**Step 3: Migrate Table from Laravel to Mysql Database**

For this first we have to create migration file in our Laravel folder. For this we have to write following command in our command prompt.

Php artisan make:migration create\_crud\_table --create=crud  
  
This command will create migration file in **database/migrations** folder. In this file we have to define table column. Given below we can find migration file in which we have define table column.

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateCrudTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('cruds', function (Blueprint $table) {

$table->bigIncrements('id');

$table->string('first\_name');

$table->string('last\_name');

$table->string('image');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('cruds');

}

}

Now migrate this table definition from Laravel application to mysql database. For this we have write following command in command prompt. This command will make crud table in mysql database for perform CRUD operation from Laravel.

php artisan migrate

**Step 4: Create Model file in Laravel**

For creating model files we have to write following command in command prompt.

php artisan make:model Crud -m

This command will make **Crud.php** model file in **app** folder. In this file we have to define table column name.

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Crud extends Model

{

protected $fillable = [

'first\_name', 'last\_name', 'image'

];

}

### Step 5: Create Controllers in Laravel

For this we have to go to command prompt and write following command.

php artisan make:controller CrudsController –resource

This command will make **CrudsController.php** file in **app/Http/Controllers** folder.We have to just add code for do particular operation. Here below we can find CRUD controller file code.

<?php

namespace App\Http\Controllers;

use App\Crud;

use Illuminate\Http\Request;

class CrudsController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

$data = Crud::latest()->paginate(5);

return view('index', compact('data'))

->with('i', (request()->input('page', 1) - 1) \* 5);

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

return view('create');

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

$request->validate([

'first\_name' => 'required',

'last\_name' => 'required',

'image' => 'required|image|max:2048'

]);

$image = $request->file('image');

$new\_name = rand() . '.' . $image>getClientOriginalExtension();

$image->move(public\_path('images'), $new\_name);

$form\_data = array(

'first\_name' => $request->first\_name,

'last\_name' => $request->last\_name,

'image' => $new\_name

);

Crud::create($form\_data);

return redirect('crud')->with('success', 'Data Added successfully.');

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

$data = Crud::findOrFail($id);

return view('view', compact('data'));

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function edit($id)

{

$data = Crud::findOrFail($id);

return view('edit', compact('data'));

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

$image\_name = $request->hidden\_image;

$image = $request->file('image');

if($image != '')

{

$request->validate([

'first\_name' => 'required',

'last\_name' => 'required',

'image' => 'image|max:2048'

]);

$image\_name = rand() . '.' . $image->getClientOriginalExtension();

$image->move(public\_path('images'), $image\_name);

}

else

{

$request->validate([

'first\_name' => 'required',

'last\_name' => 'required'

]);

}

$form\_data = array(

'first\_name' => $request->first\_name,

'last\_name' => $request->last\_name,

'image' => $image\_name

);

Crud::whereId($id)->update($form\_data);

return redirect('crud')->with('success', 'Data is successfully updated');

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function destroy($id)

{

$data = Crud::findOrFail($id);

$data->delete();

return redirect('crud')->with('success', 'Data is successfully deleted');

}

}

**Index ()** In Laravel Crud controller index () method is a root method of crud controller; Under this method we will write code for display data from mysql database. In this code it will first fetch data from crud table and store under data variable, after this we want to make paginate link by using paginate () method in Laravel. For send data to view file, for this we have use view () method for send data to view file.   
**Create ()** This method is used for load **create.blade.php** file. In this file we can find form to insert new records and filling data. This insert data request will be send to store () method of CrudsController.php controller class.  
**Store ()** This method received add or insert new records request received from create () method. In this method, it will perform two operations. One is for upload image file by using move () method and second is for insert records into mysql table by using model class. After successfully insert data in mysql table from Laravel, page will redirect to index() method with success message.  
**Show ()** This method in Crud controller has been used for fetch single data details. For this it has been used findOrFail() method. After fetch details it will send to view.blade.php file.  
**Edit ()** This method main function is fetch single data from Mysql database and load into edit or update form to make changes.

**Update ()** this method received edit or update data request from edit (). This method has done two functions:

Upload of profile image with update

Edit mysql data in Laravel framework

**Delete ()**this method mainly used for remove single or multiple data from Mysql Database. This is last operation Crud Operation in Laravel.

### Step 6: Set Route in Laravel

For this we have to open to **routes/web.php** file. In this file we have to write following code.

Route::resource('crud','CrudsController');

### Step 7 - Set Data in View File in Laravel 5.8

This is the last step in Laravel Crud, in this step we have to set data in view file which has been store under **resources/views** folder.

Here below we can file all view file which has been used in Crud.

**resources/views/parent.blade.php**

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Laravel 5.8 Crud Tutorial</title>

<meta content='width=device-width, initial-scale=1, maximum-scale=1' name='viewport'/>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet" />

</head>

<body>

<div class="container">

<br />

<h3 align="center">Laravel 5.8 Crud Tutorial</h3>

<br />

@yield('main')

</div>

</body>

</html>

**resources/views/index.blade.php**

@extends('parent')

@section('main')

<table class="table table-bordered table-striped">

<tr>

<th width="10%">Image</th>

<th width="35%">First Name</th>

<th width="35%">Last Name</th>

<th width="30%">Action</th>

</tr>

@foreach($data as $row)

<tr>

<td><img src="{{ URL::to('/') }}/images/{{ $row->image }}" class="img-thumbnail" width="75" /></td>

<td>{{ $row->first\_name }}</td>

<td>{{ $row->last\_name }}</td>

<td>

</td>

</tr>

@endforeach

</table>

{!! $data->links() !!}

@endsection

**resources/views/create.blade.php**

@extends('parent')

@section('main')

@if($errors->any())

<div class="alert alert-danger">

<ul>

@foreach($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

</div>

@endif

<div align="right">

<a href="{{ route('crud.index') }}" class="btn btn-default">Back</a>

</div>

<form method="post" action="{{ route('crud.store') }}" enctype="multipart/form-data">

@csrf

<div class="form-group">

<label class="col-md-4 text-right">Enter First Name</label>

<div class="col-md-8">

<input type="text" name="first\_name" class="form-control input-lg" />

</div>

</div>

<br />

<br />

<br />

<div class="form-group">

<label class="col-md-4 text-right">Enter Last Name</label>

<div class="col-md-8">

<input type="text" name="last\_name" class="form-control input-lg" />

</div>

</div>

<br />

<br />

<br />

<div class="form-group">

<label class="col-md-4 text-right">Select Profile Image</label>

<div class="col-md-8">

<input type="file" name="image" />

</div>

</div>

<br /><br /><br />

<div class="form-group text-center">

<input type="submit" name="add" class="btn btn-primary input-lg" value="Add" />

</div>

</form>

@endsection

**resources/views/view.blade.php**

@extends('parent')

@section('main')

<div class="jumbotron text-center">

<div align="right">

<a href="{{ route('crud.index') }}" class="btn btn-default">Back</a>

</div>

<br />

<img src="{{ URL::to('/') }}/images/{{ $data->image }}" class="img-thumbnail" />

<h3>First Name - {{ $data->first\_name }} </h3>

<h3>Last Name - {{ $data->last\_name }}</h3>

</div>

@endsection

**resources/views/edit.blade.php**

@extends('parent')

@section('main')

@if ($errors->any())

<div class="alert alert-danger">

<ul>

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

</div>

@endif

<div align="right">

<a href="{{ route('crud.index') }}" class="btn btn-default">Back</a>

</div>

<br />

<form method="post" action="{{ route('crud.update', $data->id) }}" enctype="multipart/form-data">

@csrf

@method('PATCH')

<div class="form-group">

<label class="col-md-4 text-right">Enter First Name</label>

<div class="col-md-8">

<input type="text" name="first\_name" value="{{ $data->first\_name }}" class="form-control input-lg" />

</div>

</div>

<br />

<br />

<br />

<div class="form-group">

<label class="col-md-4 text-right">Enter Last Name</label>

<div class="col-md-8">

<input type="text" name="last\_name" value="{{ $data->last\_name }}" class="form-control input-lg" />

</div>

</div>

<br />

<br />

<br />

<div class="form-group">

<label class="col-md-4 text-right">Select Profile Image</label>

<div class="col-md-8">

<input type="file" name="image" />

<img src="{{ URL::to('/') }}/images/{{ $data->image }}" class="img-thumbnail" width="100" />

<input type="hidden" name="hidden\_image" value="{{ $data->image }}" />

</div>

</div>

<br /><br /><br />

<div class="form-group text-center">

<input type="submit" name="edit" class="btn btn-primary input-lg" value="Edit" />

</div>

</form>

@endsection

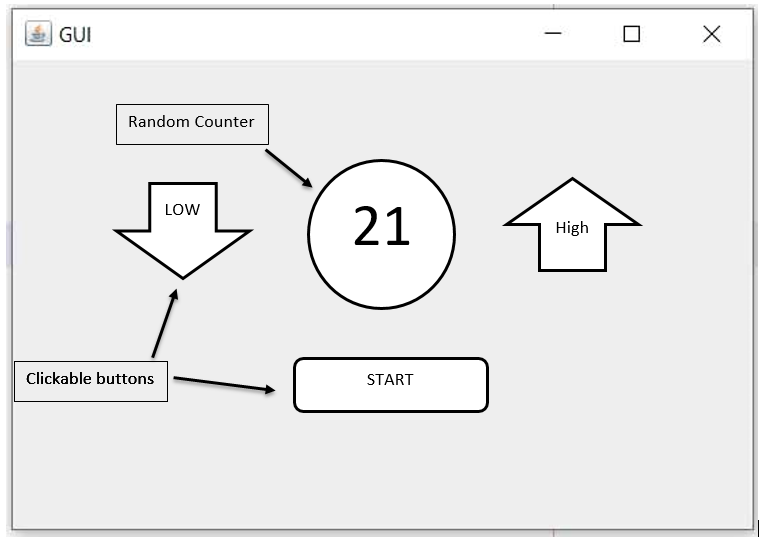
### Step 8: Run Laravel

### In last we have to run Laravel Crud application, for this we have to go to command prompt, and write following command.

php artisan serve

Above command will start Laravel. For running Laravel crud file, you have to write this url **http://127.0.0.1:8000/crud** in our browser for test code is working or not.

**Q.No.2. (Marks: 6)**

**As shown in Figure 1, create a random number game to make a guess if the upcoming random number will be higher / lower than the current one. To complete the implementation for Java GUI based desktop app, include following functionalities:**

**2.1 Provide a start/play button on start window. This will allow the users to start their game**

**2.2 Random numbers counter (Range 0 to 99) and high / low buttons to stop the counter**

**2.3 Keep generating random numbers until the user fails**

**2.4 Replace the numbers counter with win/failure label with a button either to start again or to close the game**

**2.5 Apply multi-threading concept to complete your requirements**

**ANSWER:**

**package** Exx1;

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JButton;

**import** javax.swing.JTextField;

**import** java.awt.event.ActionListener;

**import** java.util.Random;

**import** java.awt.event.ActionEvent;

**import** javax.swing.border.EmptyBorder;

**import** javax.swing.SwingConstants;

**public** **class** GuessNumberGame {

**private** JFrame frame;

**private** JTextField textField;

**public** **int** number1,number2;

JButton higher = **new** JButton("higher");

JButton lower = **new** JButton("lower");

JButton start = **new** JButton("start");

**private** **final** JButton playAgain = **new** JButton("Play Again");

**private** **final** JButton exit = **new** JButton("Exit");

**public** **void** guess() {

number2=number1;

random();

lower.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e){

**if**(number1>number2) {

textField.setText("YOU LOSE");

}**else** {

textField.setText("YOU WON");

}

End();

}

});

higher.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**if**(number1<number2) {

textField.setText("YOU LOSE");

}**else** {

textField.setText("YOU WON");

}

End();

}

});

}

**public** **static** **void** main(String[] args) {

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

Thread th1 = **new** Thread();

th1.run();

GuessNumberGame window = **new** GuessNumberGame();

window.frame.setVisible(**true**);

window.start.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) { **if**(!th1.isInterrupted()){

window.random();

window.textField.setText(Integer.*toString*(window.number1));

window.guess();

}

}

});

window.playAgain.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

window.textField.setText("");

window.number1 =0;

window.number2= 0;

window.lower.setVisible(**true**);

window.higher.setVisible(**true**);

window.start.setVisible(**true**);

window.playAgain.setVisible(**false**);

window.exit.setVisible(**false**);

window.start.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**if**(!th1.isInterrupted()){

window.random();

window.textField.setText(Integer.*toString*(window.number1));

window.guess();

}

}

});

}

});

window.exit.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

System.*exit*(1);

}

});

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

**public** **void** random() {

Random random=**new** Random();

number1=random.nextInt(100);

}

**public** **void** End() {

lower.setVisible(**false**);

higher.setVisible(**false**);

start.setVisible(**false**);

playAgain.setVisible(**true**);

exit.setVisible(**true**);

}

**public** GuessNumberGame() {

initialize();

}

**private** **void** initialize() {

frame = **new** JFrame();

frame.setBounds(100, 100, 450, 300);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.getContentPane().setLayout(**null**);

higher.setBounds(233, 98, 85, 21);

frame.getContentPane().add(higher);

lower.setBounds(115, 98, 85, 21);

frame.getContentPane().add(lower);

exit.setVisible(**false**);

playAgain.setVisible(**false**);

start.setBounds(173, 159, 85, 21);

frame.getContentPane().add(start);

textField = **new** JTextField();

textField.setHorizontalAlignment(SwingConstants.***CENTER***);

textField.setBorder(**new** EmptyBorder(0, 0, 0, 0));

textField.setEditable(**false**);

textField.setBounds(173, 68, 96, 19);

frame.getContentPane().add(textField);

textField.setColumns(10);

playAgain.setBounds(65, 159, 109, 21);

frame.getContentPane().add(playAgain);

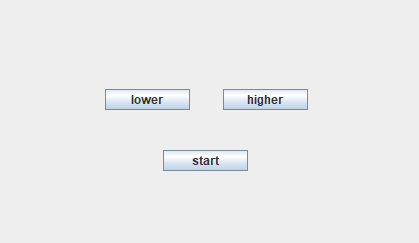
exit.setBounds(277, 159, 96, 21);

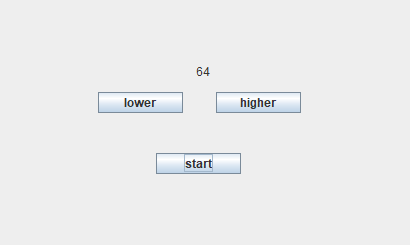
frame.getContentPane().add(exit);

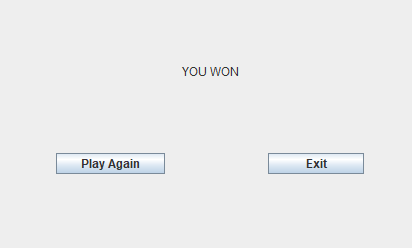
}

}

**Screen Shots:**





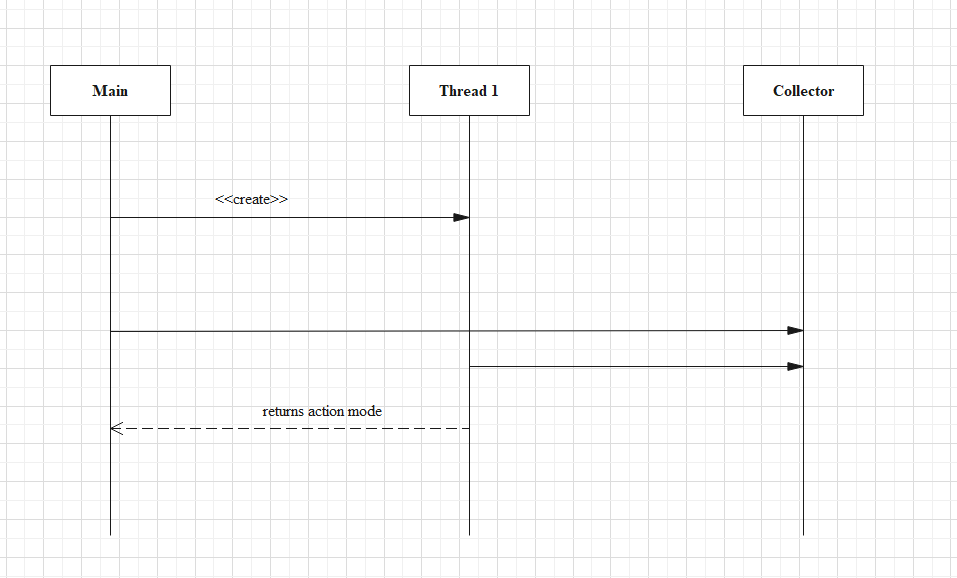


**Q.No.3. (Marks: 4)**

**Briefly describe the concept of multi-threading in java. From the solution of Q.No.2, provide an execution diagram to demonstrate the process of context switching between main thread and peer thread.**

**ANSWER:**

**Multithreading** in Java is a process of executing two or more threads simultaneously to maximum utilization of CPU. Multithreaded applications execute two or more threads run parallel. Hence, it is also known as Concurrency in Java. Each thread runs parallel to each other. Multiple threads don't allocate separate memory area, hence they save memory. Also, context switching between threads takes less time.

****

**Q.No.4. (Marks: 4)**

**There are some tools and technologies that help technical teams to manage changes to their source code over the time and keep track of every modification during the process of software construction. Keep in mind the concept of GitHub, provide a basic guideline about how and why you need to manage multiple branches and then merge to the original base origin. Make sure to cover the following key points:**

**4.1 Which tools and software will be used to prepare GitHub on your local system**

**4.2 How you can create a new branch of your original repository**

**4.3 Managing and switching between multiple branches using GitHub**

**4.4 Provide solution for both terminal as well as GUI based environment**

GitHub allows collaboration with any developer all over the world. An open source solutions enable potential developers to contribute and share the knowledge to benefit the global community.

GitHub is a Git repository hosting service, which provides a web-based graphical interface. It helps every team member to work together on the project from anywhere and makes it easy for them to collaborate.

We can use GitHub online as well as download GitHub Desktop.

**4.1 Which tools and software will be used to prepare GitHub on your local system**

**ANSWER:**

The **tools** and **software** that must be installed in the local systems are mentioned below.

* Check the version and install Git software on the local computer according to the operating system. Such as install **GitHub** for Windows OS.
* Get the Developer Certificate of Orginin (DCO) to license the code.
* Install **Code Management tools** such as Black Duck Hub, Dep-checker, and so on.
* Install **Issue** and **Bug checking** tools such as JIRA, GitHub Issues, GitLab, and so on.
* Install **Release** and **Archiving** management such as Docker Hub, GitHub release, and so on.
* Install the **project tacking health** software to get the proper tacking of projects such as GHCrawler, OSSTracker, and so on.
* If the developer wants to get **reviews on the code** then the developer can install some applications such as PullApprove, Sentinel, and so on.

The installation of GitHub Desktop is as simple as any other Windows application installation. All we need to do is:

* Open a browser.
* Visit [desktop.github.com](https://desktop.github.com/)
* Click Download for WIndows (64bit).
* When prompted, click Run.

Allow the installation to download and install.

Once the installation completes, GitHub Desktop will launch.

We use gitbash to use GitHub in our local system and we can install that software from the git source. installation of it will be easy

We can use git by GUI and command prompt to use git

Creating a new branch in GitHub for the original repository

the command syntax is --- git command command-name

**Uploading code to GitHub**

First, we'll need to create a GitHub account.

Once we're in, look for a + in the top corner. Click "New Repository" (the name for Git folders, "repo" for short). Give it a name - probably the same as the folder we created earlier where we've saved our commits. Then click "Create Repository". We can now copy the url you are redirected to and we can set the **origin** for our code.

There will be an authentication step at some point here - just follow the directions. Git is very good about giving us clear instructions on the next steps to take.

**4.2 How you can create a new branch of your original repository**

**ANSWER:**

## Creating a New Branch From GitHub Website

## Navigate to the main page of the GitHub repository for which you want to create a new branch.

## We’ll see the name of our current branch in the branch selector dropdown. To see all the branches in the repository, click on [NUMBER Branches]. In my case, we can see that we’re currently on Master branch and we have following branches.

**Viewing all branches in GitHub repository**

To create a new branch, click on the **branch selector dropdown** and start typing the name of the new branch in the text box.

Type the name of new GitHub branch in the search box

By default, this text box checks if there’s an existing branch with the name we just entered.

If not, we’ll be able to create a new branch by selecting “**Create Branch**”.

Click **create branch**

This will automatically create a new branch with the Master branch as our base branch.

However, to switch and edit code from the new branch we just created, we’ll still have to use GitHub’s desktop app or the terminal code.

## Using Command Line to Create New Branch in GitHub

Creating a new branch using the command line is nearly every developer’s favourite method. Because once we get a hang of it, it’s unbelievably easy to use.

First, make sure to cd into the local repository. Once we’re in the right folder, execute

$ git branch <branch-name>

This will create a new branch. But before we start making changes to our code, we will have to first switch to the new branch we just created. To do that, run

$ git checkout <branch-name>

Many developers, especially when they’re just getting started, forgegitt switching to the new branch. That’s why we can use this command that will create the new branch and immediately switch us to it:

$ git checkout -b <branch-name>

Once we’ve created a new branch and switched to it, we can start making changes in our code. However, all of these changes (including the new branch) is still only in our local machine.

To publish the new branch we created in GitHub and make it available for everyone in our team, run the following command:

$ git push -u <remote> <branch-name>

**4.3 Managing and switching between multiple branches using GitHub**

**ANSWER:**

command syntax -- git checkout branch-name

In GitHub Desktop, click **Current Branch**.

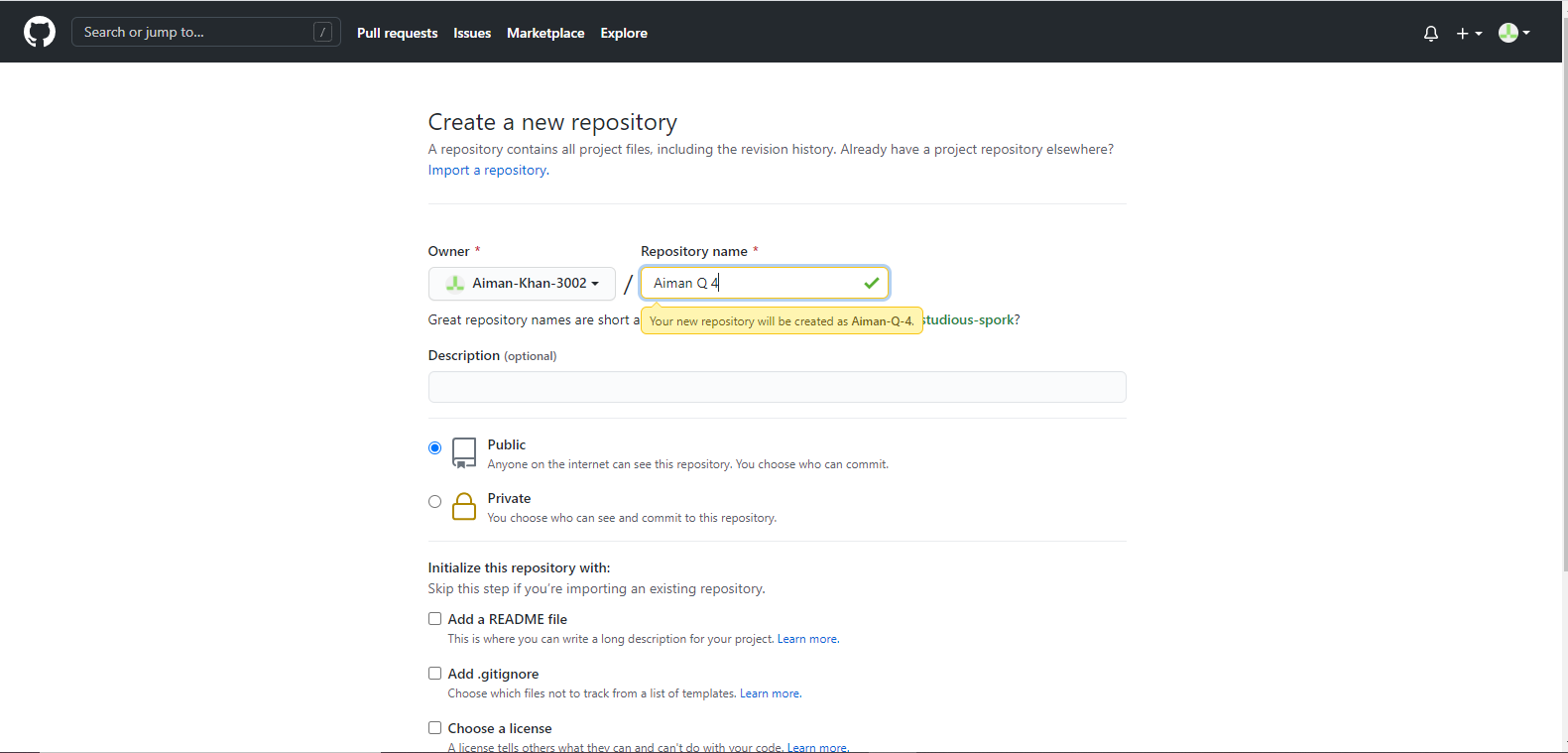
In the list of branches, click the branch we want to switch to.

If we have saved, uncommitted changes, choose **Leave my changes** or **Bring my changes**, then click **Switch Branch**.

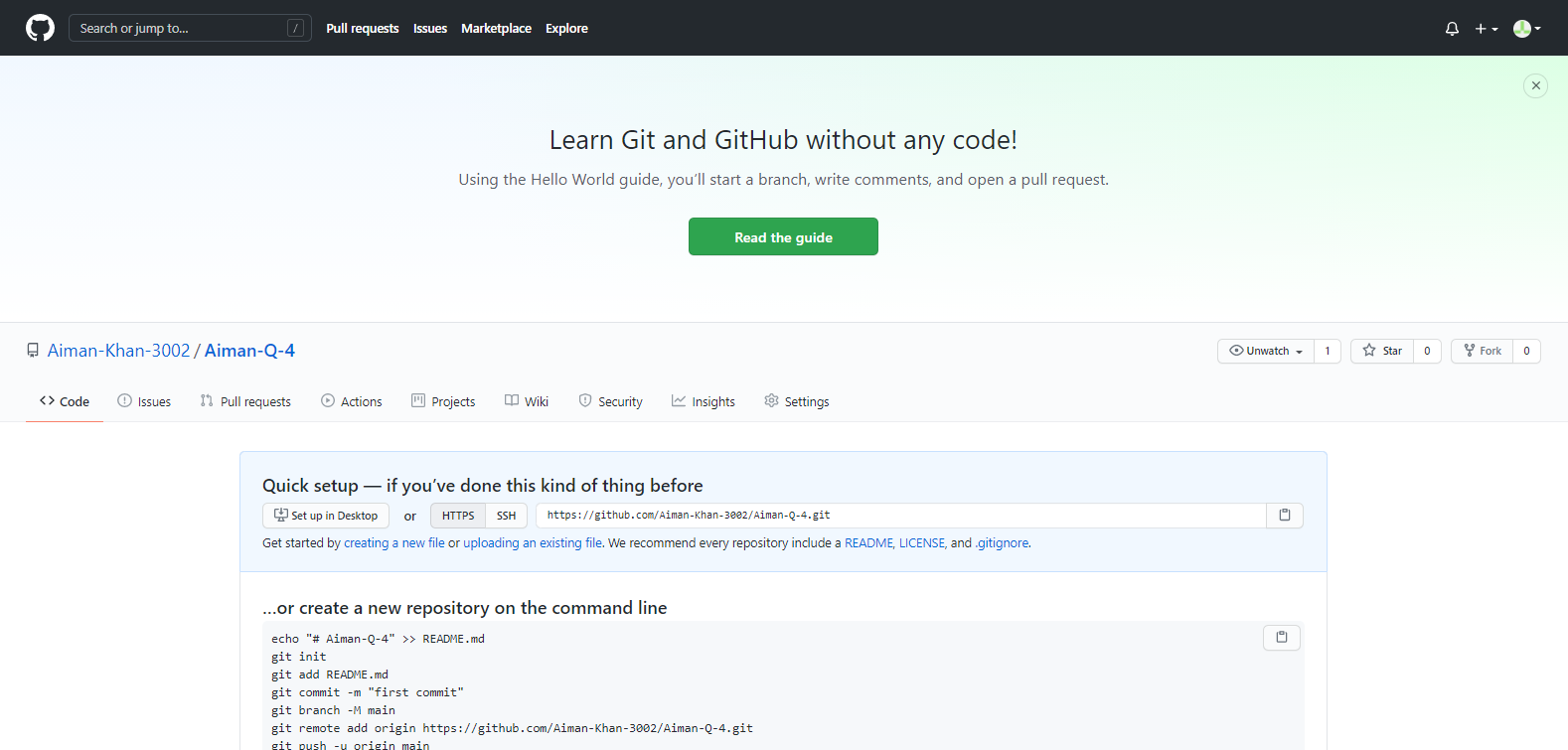
**4.4 Provide solution for both terminal as well as GUI based environment**

**ANSWER:**

Download GitHub or use online. Install GitBash as well. create a new repository as named as “ Aiman Q 4” in the figure below.



Here we can add files by clicking on “Add Files” option. But here I haven’t seen that option, so we may use the command.

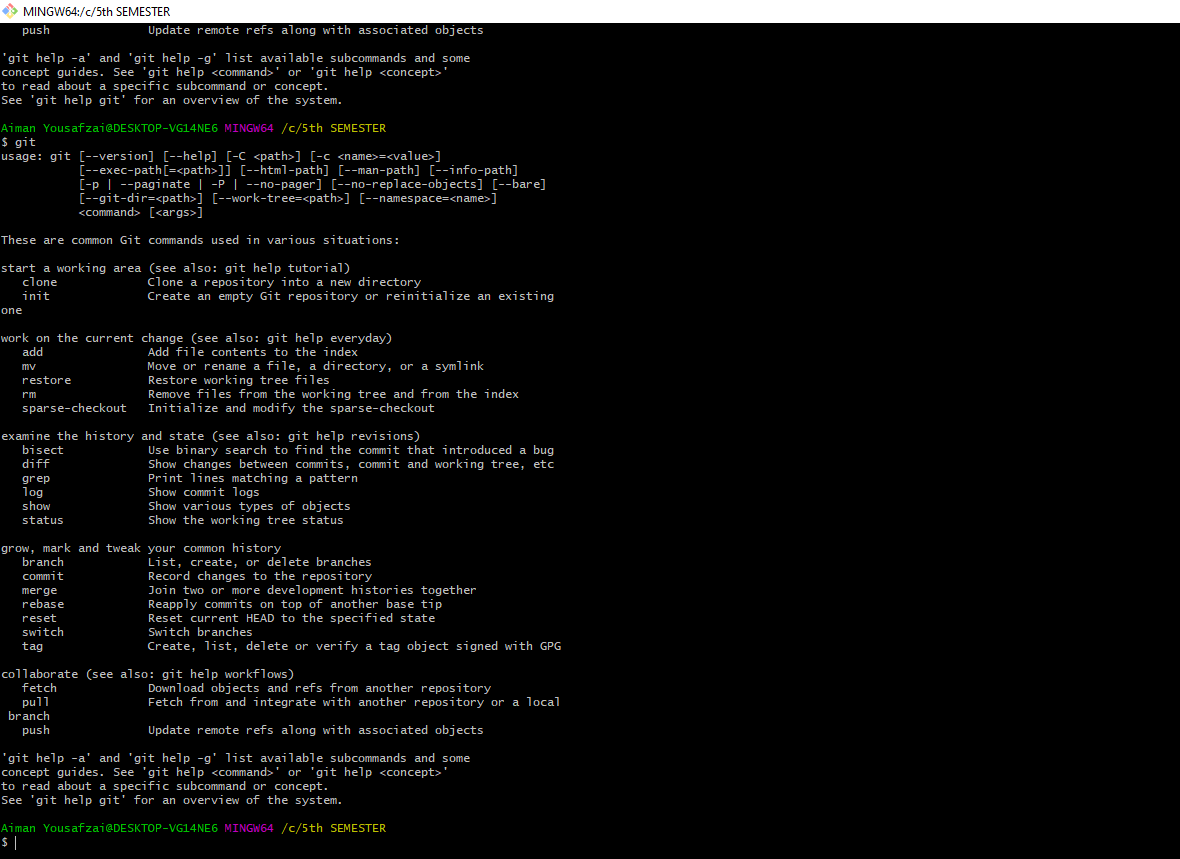


Open any file which have to add in “file manager”, right click on folder and select the “GitBash here” option and then we will be redirected to a new window and run sie commands here:

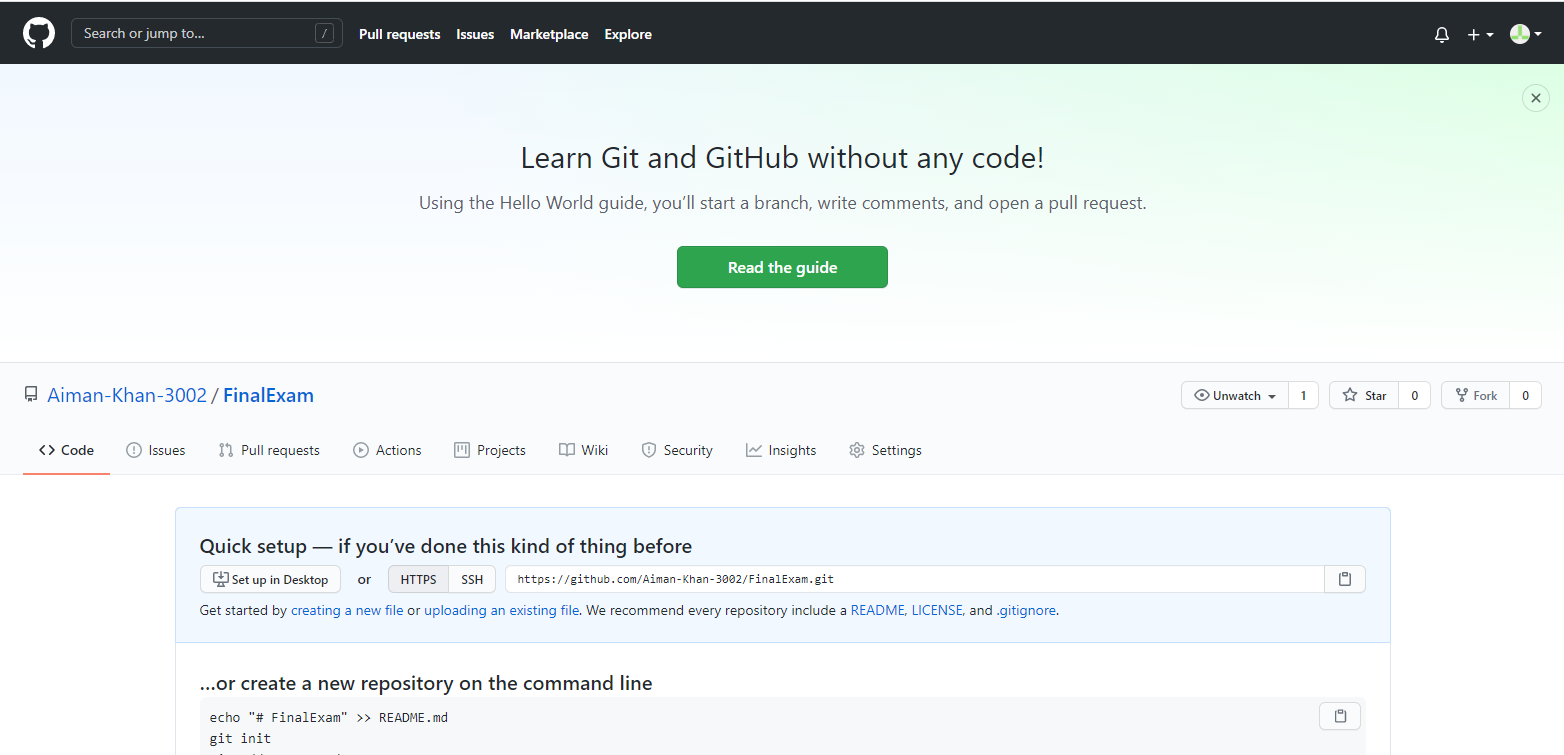
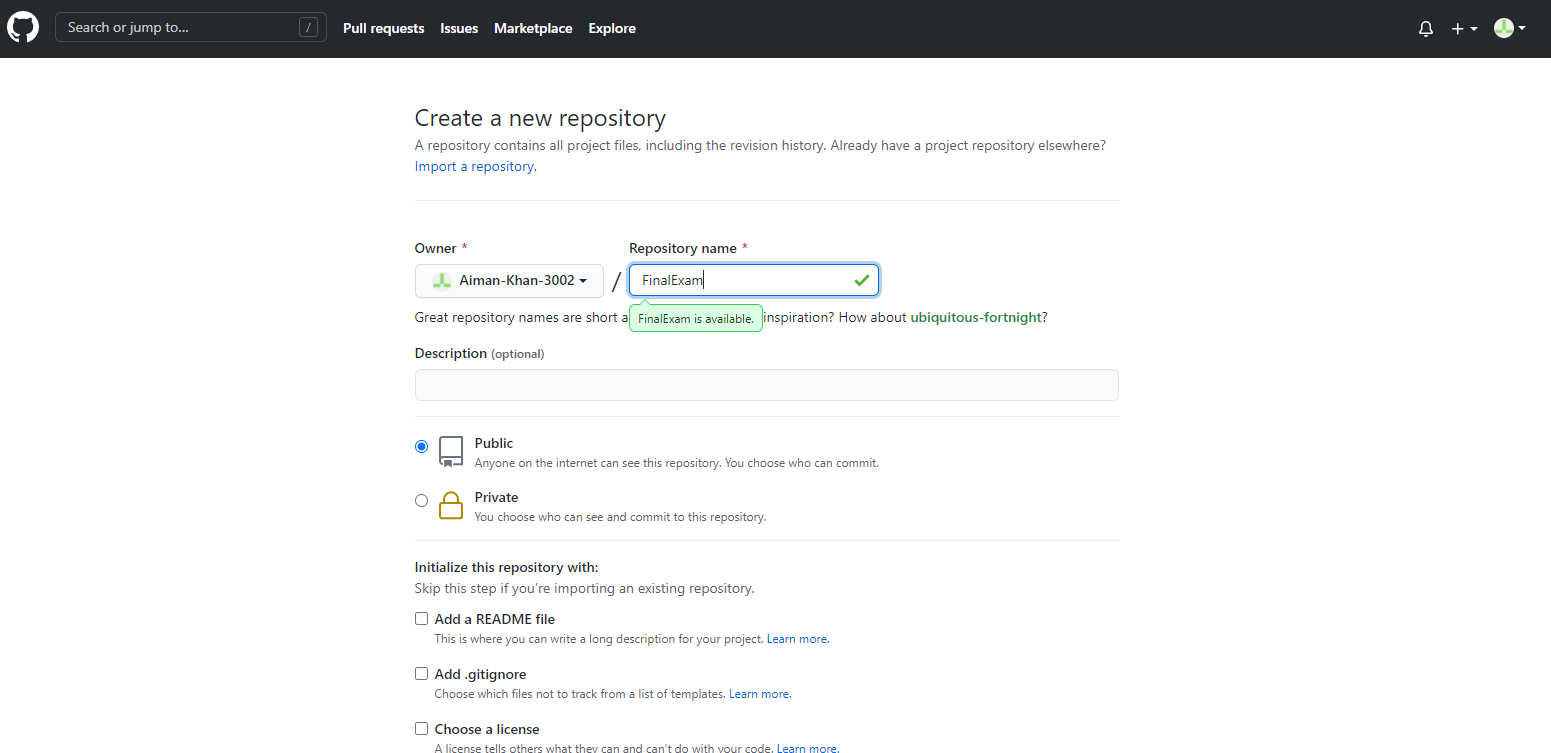
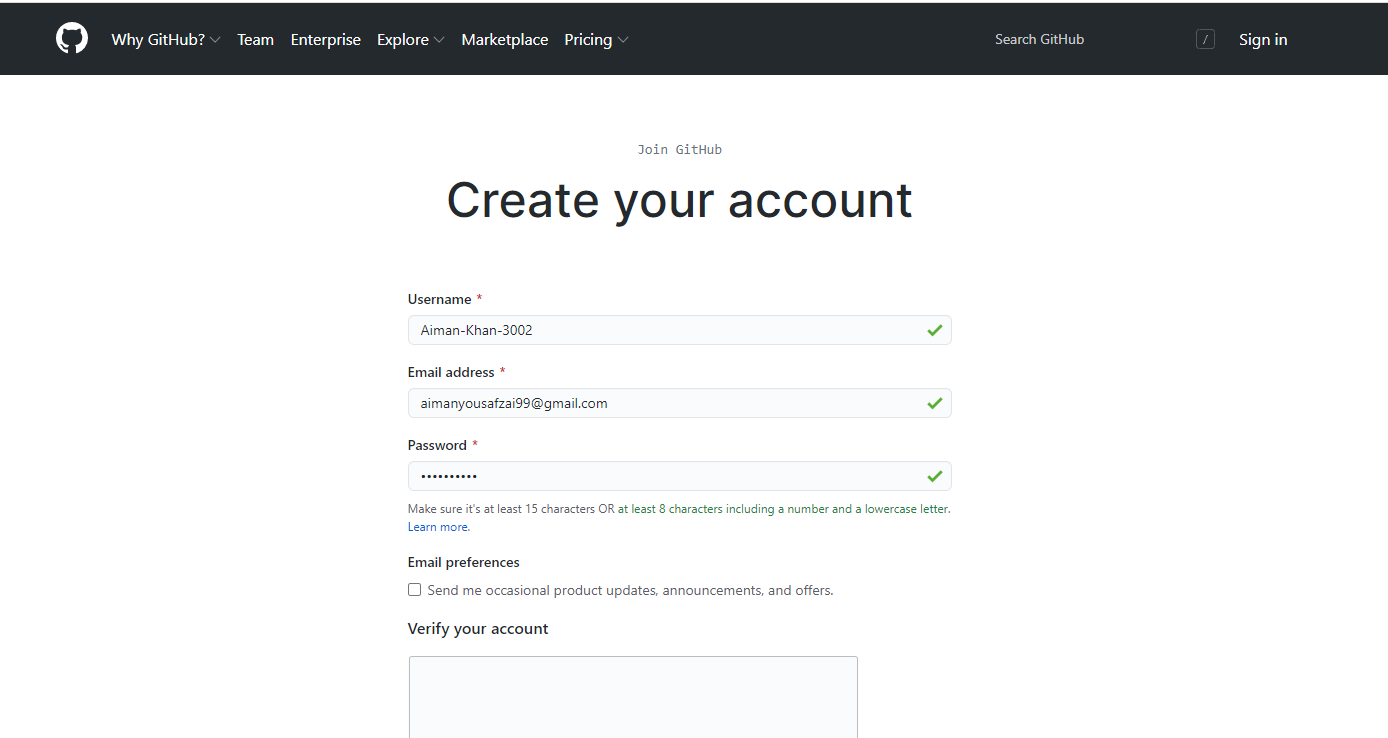
git branch <branch-name>

$ git checkout <branch-name>

$ git checkout -b <branch-name>



**Some other Shots:**



**Download GitHub:**

