# BAB VI

**Object Oriented Programming I**

## Tujuan

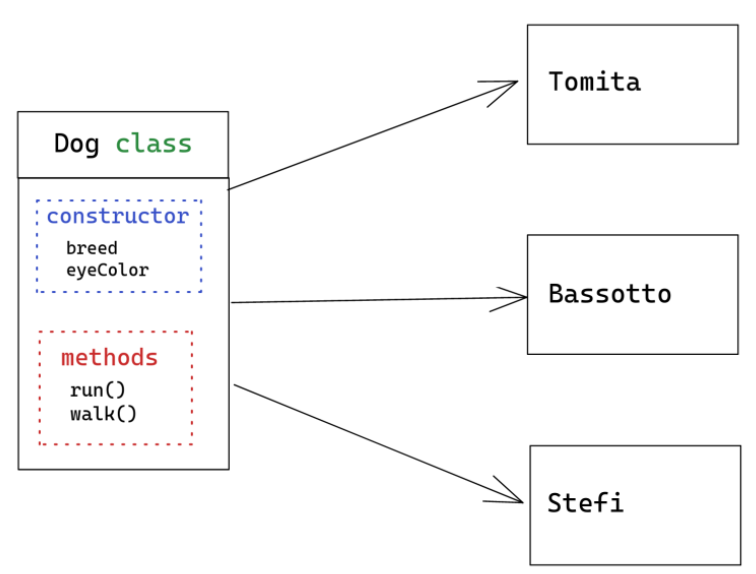
1. Praktikan dapat memahami konsep Pemrograman Berorientasi Objek
2. Praktikan mampu memahami penggunaan *Class* dan *Constructor*
3. Praktikan mampu mengimplementasikan penggunaan *Class* dan  *Constructor*

## Dasar Teori

### OOP (*Object Oriented Programming*)

Pemrograman berorientasi objek (*Object Oriented Programming* atau disingkat OOP) adalah paradigma pemrograman yang berorientasikan kepada objek yang merupakan suatu metode dalam pembuatan program, dengan tujuan untuk menyelesaikan kompleksnya berbagai masalah program yang terus meningkat. Objek adalah entitas yang memiliki atribut, dan perilaku (*behaviour*). Ide dasar pada OOP adalah menyelesaikan suatu masalah bukan menggunakan perintah program tetapi dengan membuat *object-object* yang saling berinteraksi. Salah satu konsep yang sering digunakan dalam OOP adalah *Class* & *Constructor*.

### *Class*



*Class* adalah cetak biru (*blueprint*) rancangan dari objek. Kita bisa membuat banyak objek dari satu macam *class*. *Class* mendefinisikan dan mengelompokkan sebuah tipe dari objek. Di dalam *class* kita dapat mendeklarasikan variabel dan menciptakan objek (*instance*). Sebuah *class* mempunyai anggota yang terdiri dari *atribut* dan *method*. *Attribute* adalah semua isian (*field)* identitas yang kita berikan pada suatu *class*. Misalkan kita membuat suatu *class* manusia, maka beberapa contoh isian (*field)* atribut yang dimiliki adalah nama dan umur. *Method* dapat kita artikan sebagai semua fungsi maupun prosedur yang merupakan perilaku (*behaviour*) dari suatu *class*.

Berikut bentuk umum dari penggunaan *class*:

|  |
| --- |
| [modifier] class <nama\_class> {  <class body>  } |

Keterangan :

1. *Access* *Mode* *Modifier* adalah kata kunci untuk menentukan level atau tingkat akses member dan *class*. *Modifier* ini hampir sama dengan enkapsulasi pada sebuah *method/function*. Namun jarak jangkauan aksesnya berada pada *package*.

Umumnya ada 3 jenis Modifier:

1. Kosong / *default* / *not specified* : dapat diakses oleh *class* lain tapi dalam satu package
2. *Public* : dapat dipakai di *class* manapun atau *package* lain.
3. *Private* : Tidak bisa dipakai oleh *class* manapun
4. <*class*\_name> bisa diganti dengan nama *class* yang diinginkan
5. <*class*\_body> ini biasanya berisi variabel , *method* , *function*, *constructor* maupun data dari *class* lain

Adapun cara untuk mengakses *class* di *class* yang lain (*object*) dapat menggunakan format berikut.

|  |
| --- |
| <nama\_kelas> <nama\_objek> = new <nama\_kelas>(<parameter>); |

Keterangan :

1. <nama\_kelas> merupakan nama dari *class* lain yang mempunyai modifier *public* maupun *default*
2. <nama\_objek> merupakan variabel bebas yang mewakili *class* yang akan dipakai
3. <parameter> berhubungan dengan sebuah *constructor* yang sifatnya opsional

### *Constructor*

Tipe khusus *method* yang digunakan untuk menginstansiasi atau menciptakan atribut-atribut yang nantinya akan dimiliki sebuah objek. Nama *constructor* = nama kelas. *Constructor* TIDAK BISA mengembalikan nilai. Tanpa membuat *constructor* secara eksplisit-pun, Java akan menambahkan *constructor* *default* secara implisit. Tetapi jika kita sudah mendefinisikan minimal sebuah *constructor*, maka Java tidak akan menambah *constructor* default. *Constructor* *default* tidak punya parameter. *Constructor* bisa digunakan untuk membangun suatu objek, langsung mengeset atribut atributnya. *Constructor* seperti ini harus memiliki parameter masukkan untuk mengeset nilai atribut. *Access* *Modifier* *constructor* sebaiknya adalah *public*, karena *constructor* akan diakses di luar kelasnya.

Cara panggil *constructor* adalah dengan menambah *keyword* “new”. *Keyword* “new” dalam deklarasi ini artinya kita mengalokasikan pada memory sekian blok memori untuk menampung objek yang baru kita buat.

|  |
| --- |
| <modifier> <nama\_class> (<parameter>) {  <body\_constructor>  } |

Keterangan :

1. <modifier> bisa menggunakan salah satu dari 3 *modifer* yang ada
2. <nama\_*class*> merupakan bagian nama sebuah *class* dipakai karena anti saat dipanggil dari *class* lain akan langsung otomatis dijalankan
3. <parameter> disini menyesuaikan dari objek *class* tersebut dipakai oleh *class* lain contoh :

Main Class

|  |
| --- |
| userService dataLogin = new userService(username,password); |

Constructor Class userService

|  |
| --- |
| class userService {  public userService(String username, String Password) {}  } |

### *Inheritance*

Prinsip *inheritance* pada OOP adalah di mana kita dapat membentuk *class* baru yang “mewarisi” atau memiliki bagian-bagian dari *class* yang sudah ada sebelumnya. Konsep ini menggunakan sistem hirarki atau bertingkat.

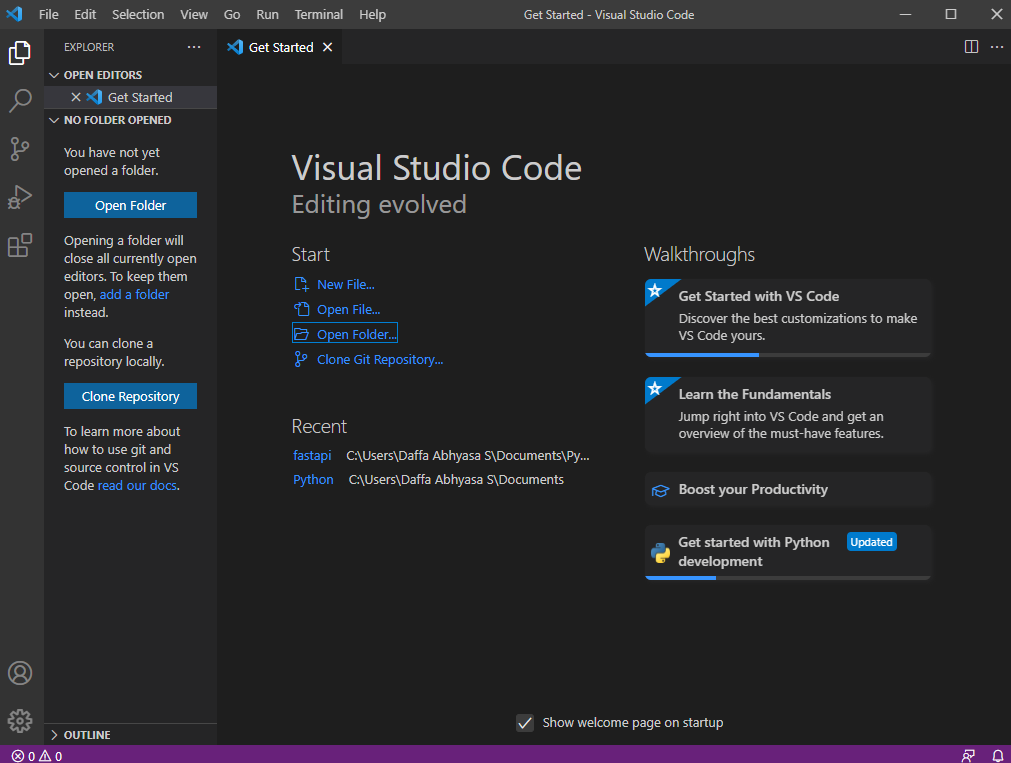
Contoh dari *inheritance* (Python):

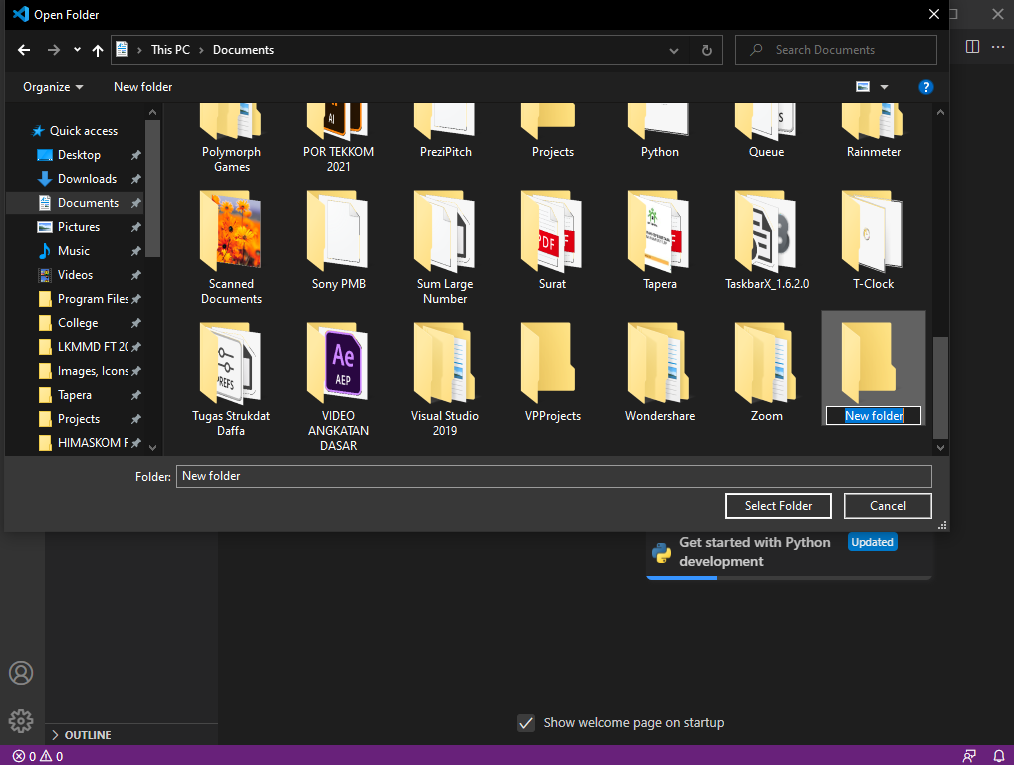
Disini *class* *guest* mewarisi *constructor* serta *method* yang ada di *class* userService sebagai *class parent* dari *class* *gues*.

|  |
| --- |
| from userService import userService    class guest(userService):  def \_\_init\_\_(self, email, password):  super().\_\_init\_\_(email, password)  self.data = {  "nama3kelompokxx@gmail.com" : {  "email" : "nama3kelompokxx@gmail.com",  "password" : "12345",  "role" : "guest"  }  } |

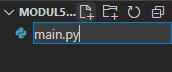
## Percobaan

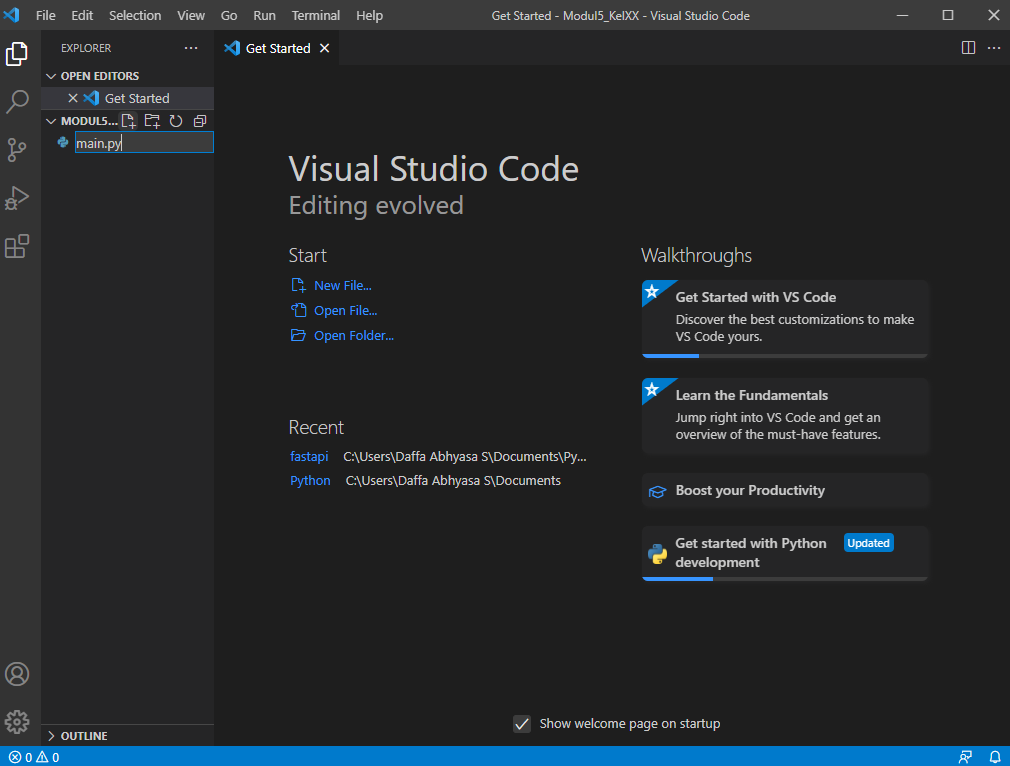
### Python

1. Buka Visual Studio Code, lalu klik *Open Folder* 
2. Buat *folder* baru (klik kanan. lalu klik *new*, lalu klik *folder*) di *folder* Documents atau *folder* lainnya (sesuai preferensi anda). Beri nama *folder* Modul5\_KelXX. (Ganti huruf XX dengan nomor kelompok anda sendiri)

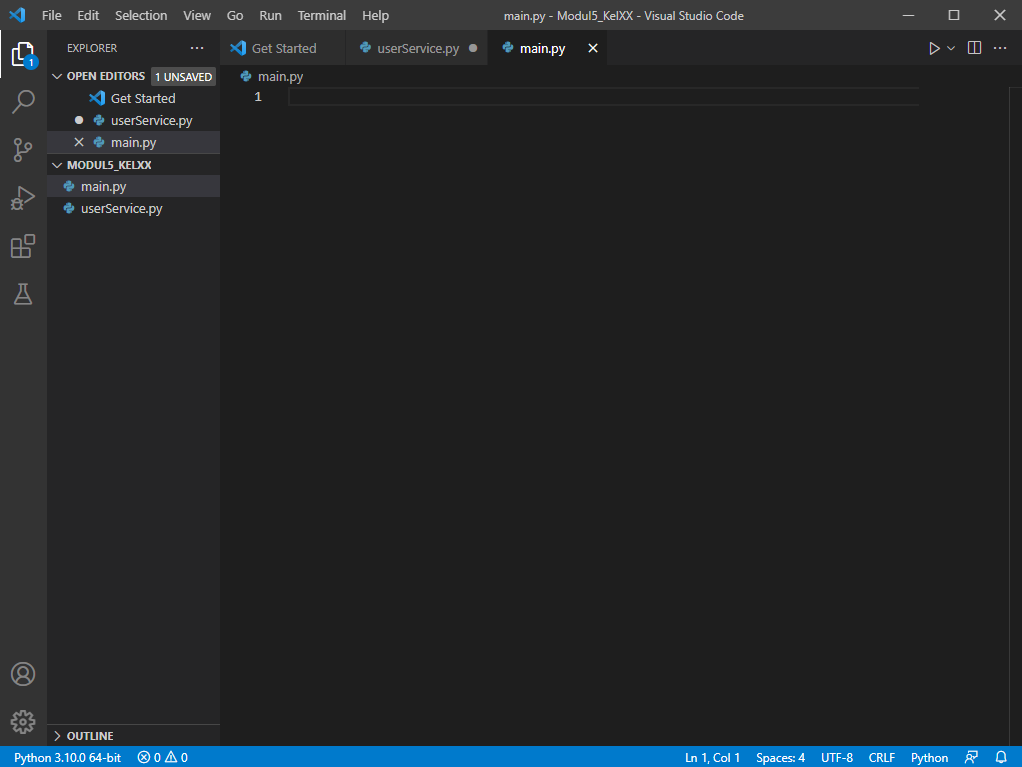


1. Klik *icon new file* disamping nama *folder* lalu beri nama *file* main.py



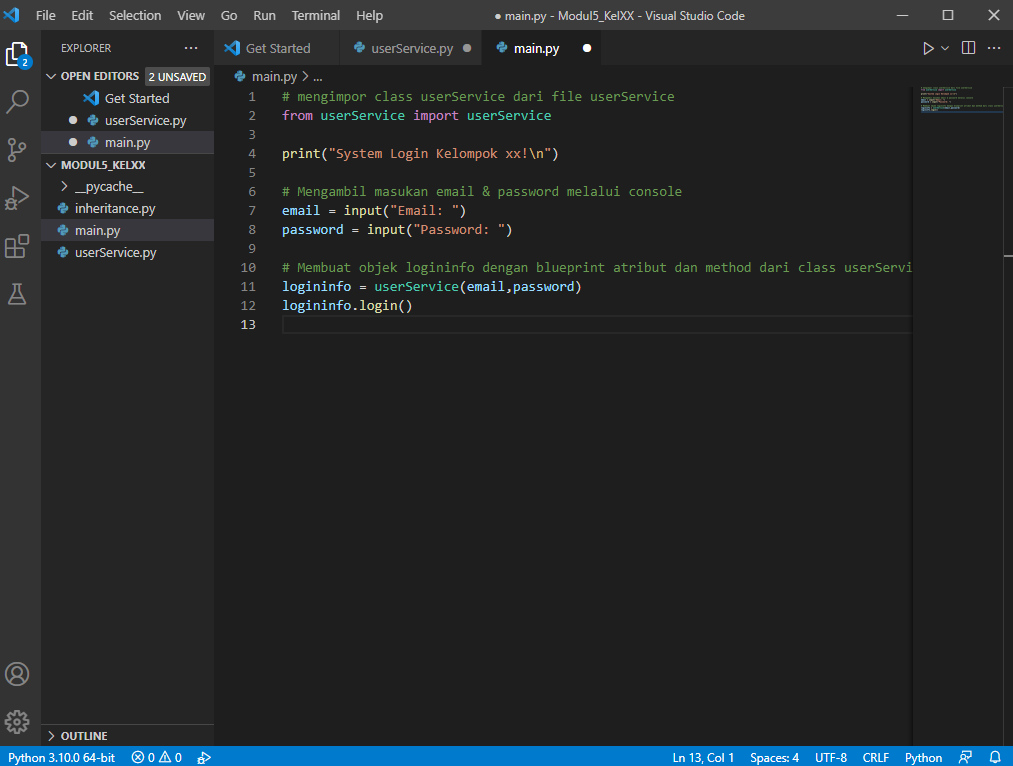


1. Lakukan hal yang sama untuk membuat *file* baru kedua dan beri nama *file* userService.py

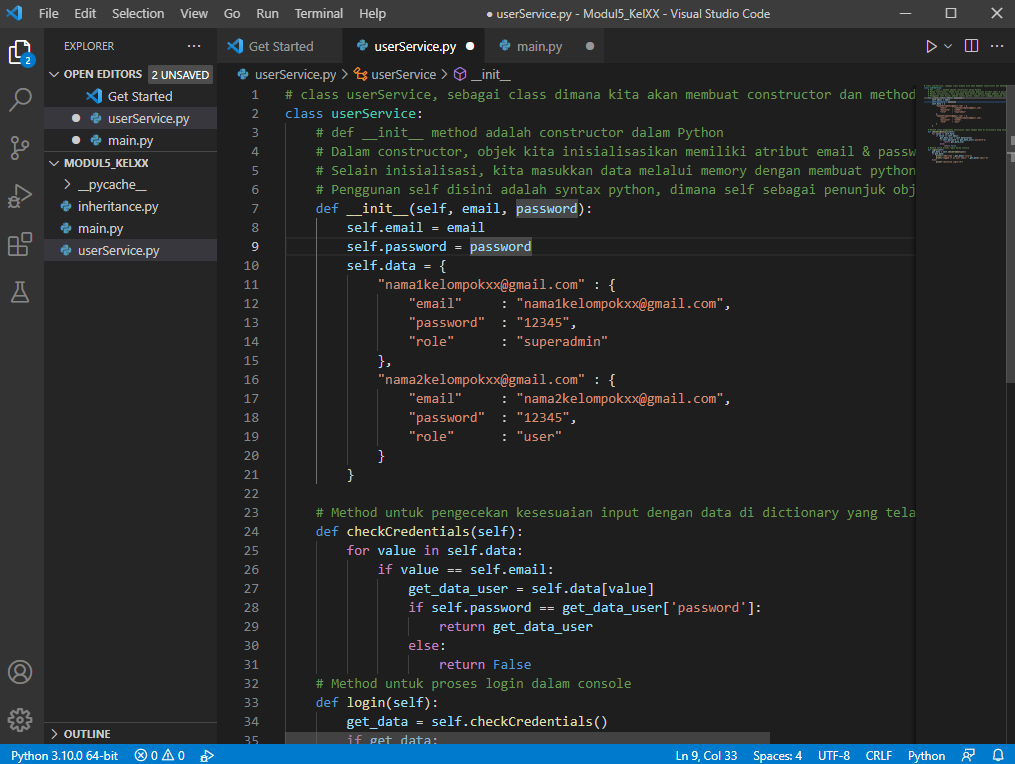


1. Tambahkan kode berikut di *file* main.py

|  |
| --- |
| # mengimpor class userService dari file userService  from userService import userService    print("System Login Kelompok xx!\n")    # Mengambil masukan email & password melalui console  email = input("Email: ")  password = input("Password: ")    # Membuat objek logininfo dengan blueprint atribut dan method dari class userService  logininfo = userService(email,password)  logininfo.login() |

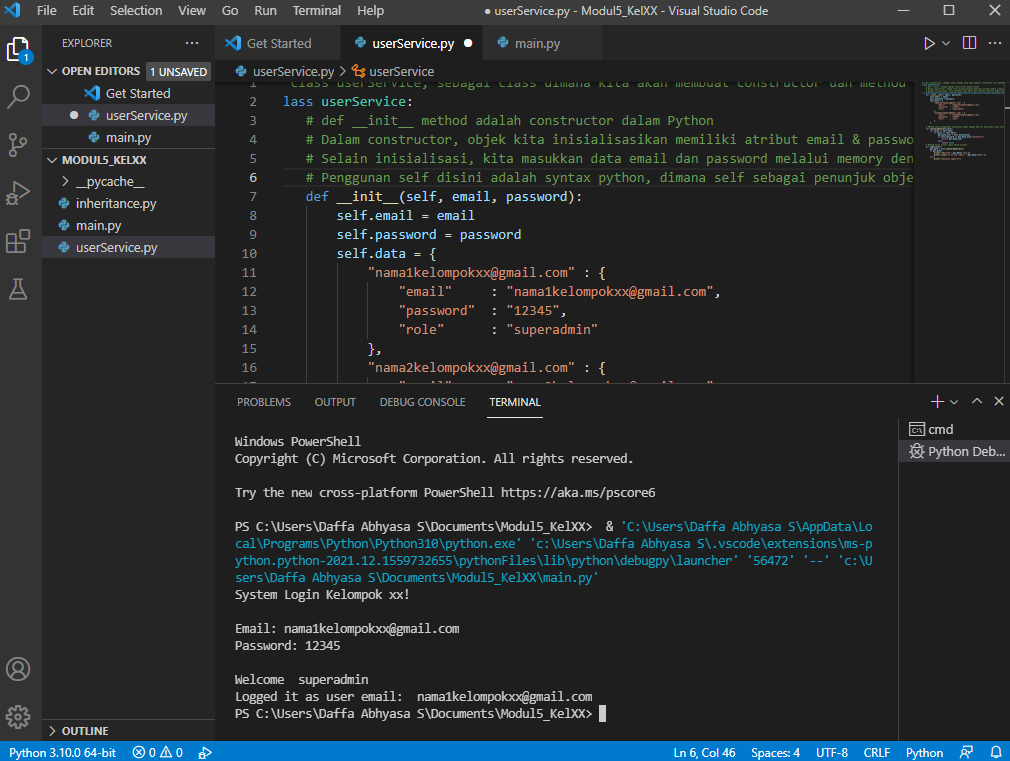


1. Tambahkan kode berikut di *file* userService.py. Ganti "nama1kelompokxx@gmail.com” dengan nama anggota kelompok dan nomor kelompok anda Anda juga dapat mengganti password apabila anda ingin.

****

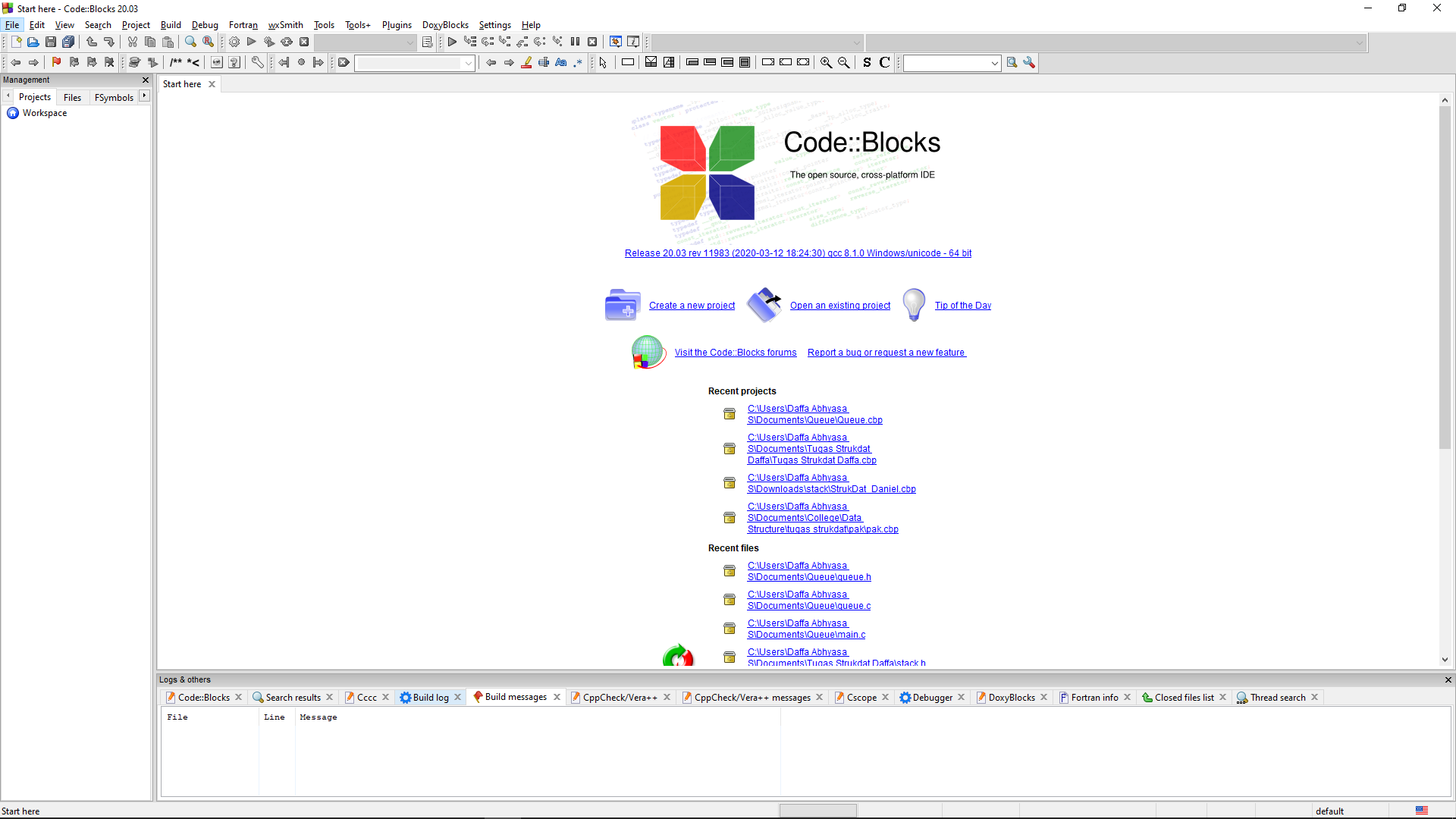
|  |
| --- |
| # class userService, sebagai class dimana kita akan membuat *constructor* dan method untuk objek user  class userService:  # def \_\_init\_\_ method adalah *constructor* dalam Python  # Dalam *constructor*, objek kita inisialisasikan memiliki atribut email & password  # Selain inisialisasi, kita masukkan data melalui memory dengan membuat python dictionary self.data  # Penggunan self disini adalah syntax python, dimana self sebagai penunjuk objek yang sedang dipakai  def \_\_init\_\_(self, email, password):  self.email = email  self.password = password  self.data = {  "nama1kelompokxx@gmail.com" : {  "email" : "nama1kelompokxx@gmail.com",  "password" : "12345",  "role" : "superadmin"  },  "nama2kelompokxx@gmail.com" : {  "email" : "nama2kelompokxx@gmail.com",  "password" : "12345",  "role" : "user"  }  }    # Method untuk pengecekan kesesuaian input dengan data di dictionary yang telah kita buat  def checkCredentials(self):  for value in self.data:  if value == self.email:  get\_data\_user = self.data[value]  if self.password == get\_data\_user['password']:  return get\_data\_user  else:  return False  # Method untuk proses login dalam console  def login(self):  get\_data = self.checkCredentials()  if get\_data:  print("\nWelcome ",get\_data['role'])  print("Logged it as user email: ",get\_data['email'])  else:  print("\nInvalid Login!\n") |

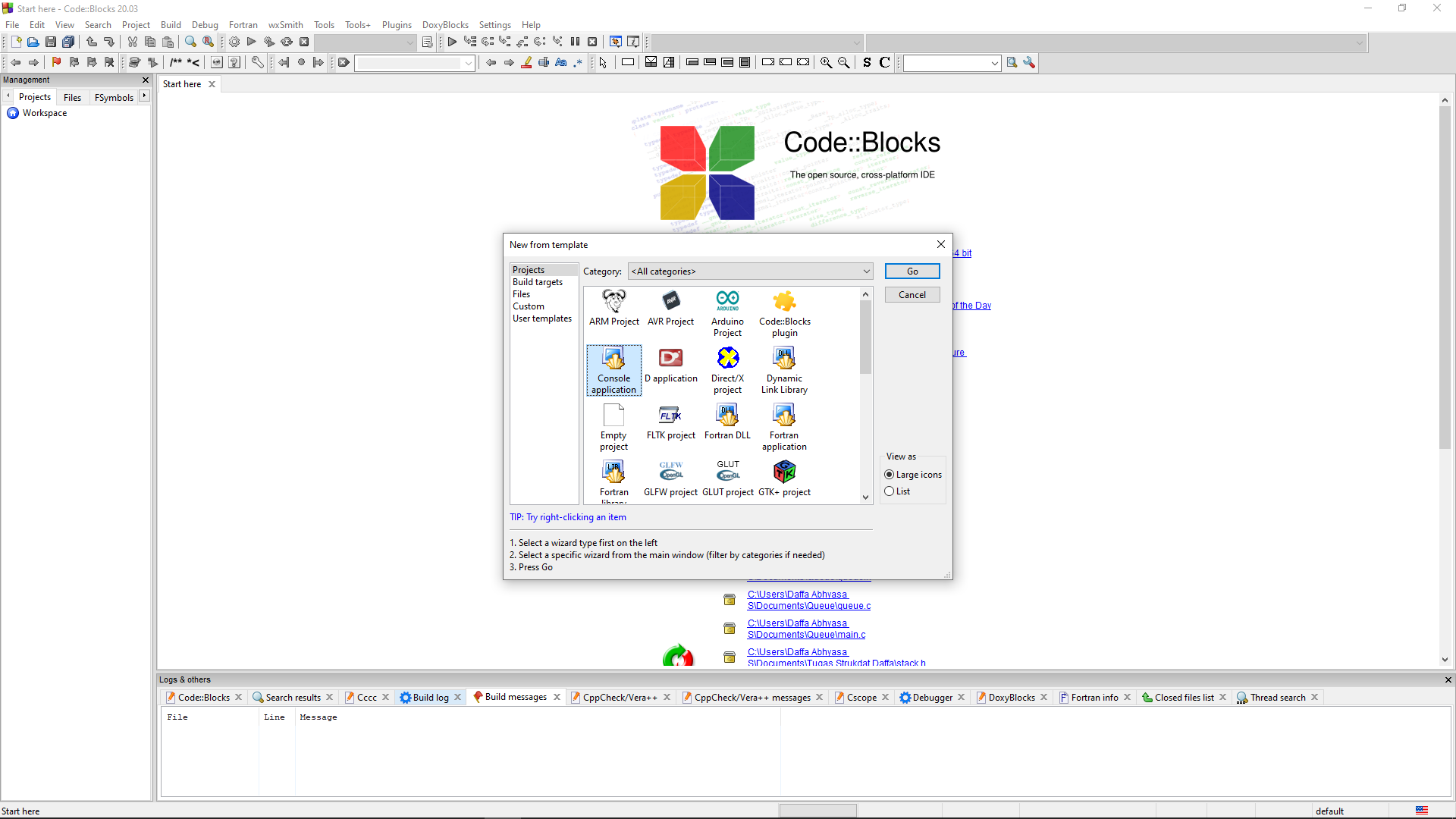
1. Jalankan aplikasi dengan mengklik Run lalu Start Debugging. Masukkan *email* dan *password* sesuai dengan yang telah anda ganti. Lihat keluaran yang muncul

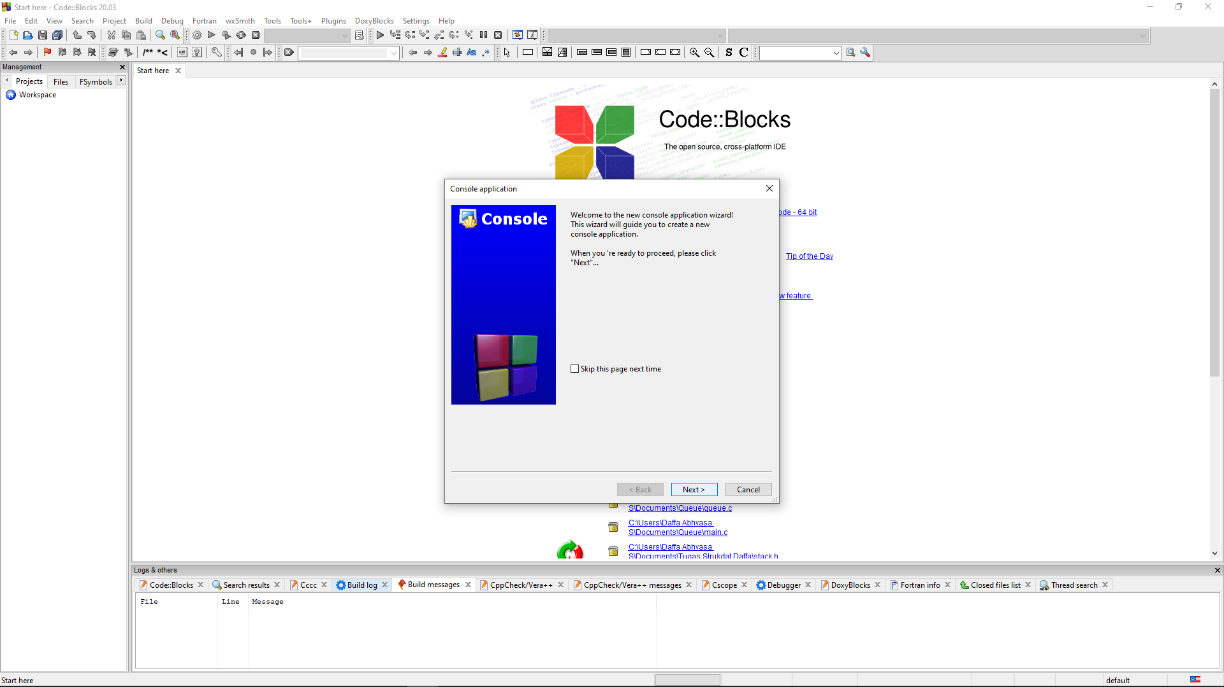
****

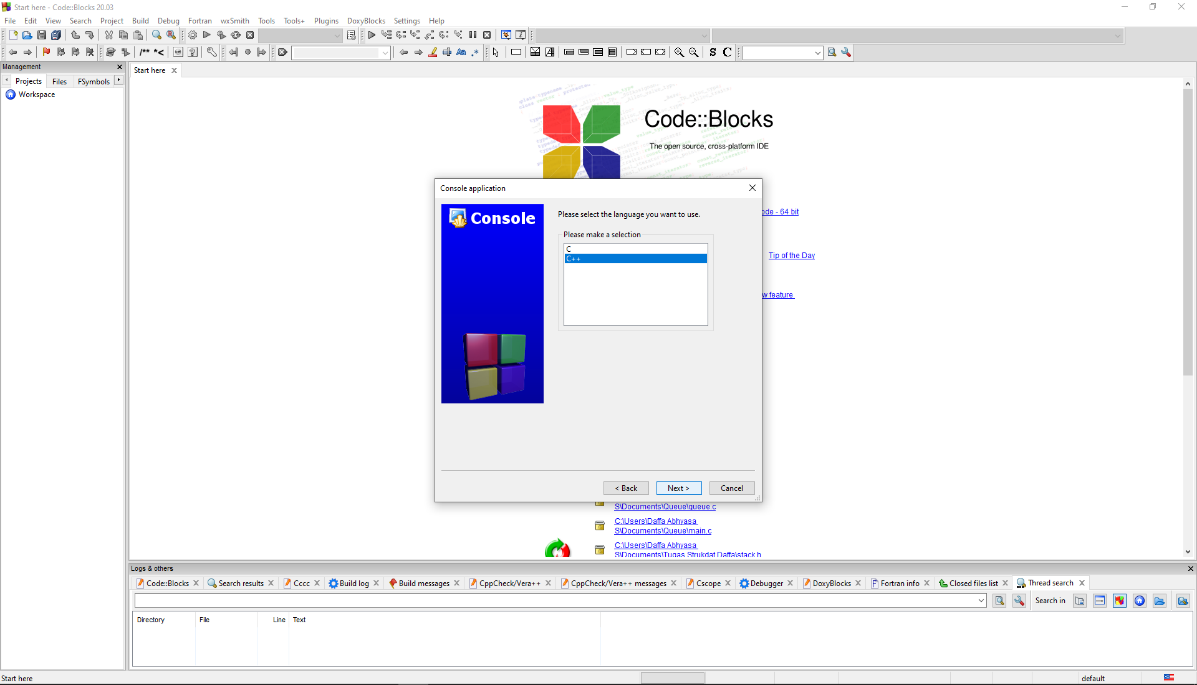
### C++

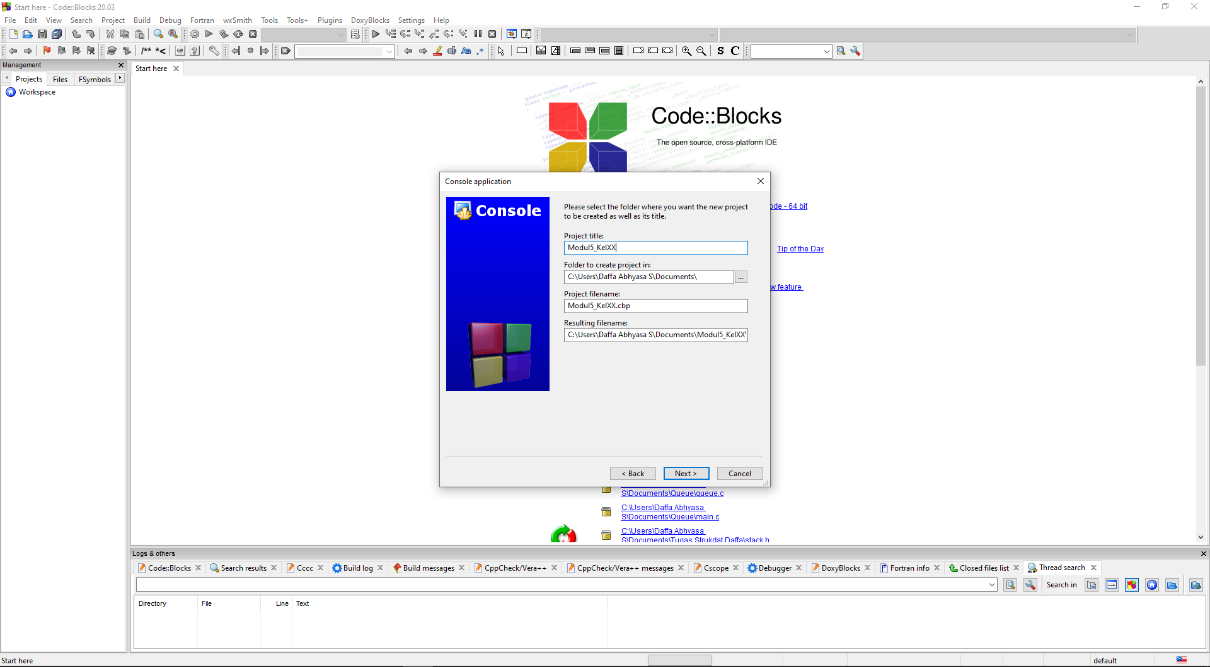
1. Buka Codeblocks , klik “*Create a new project*”.



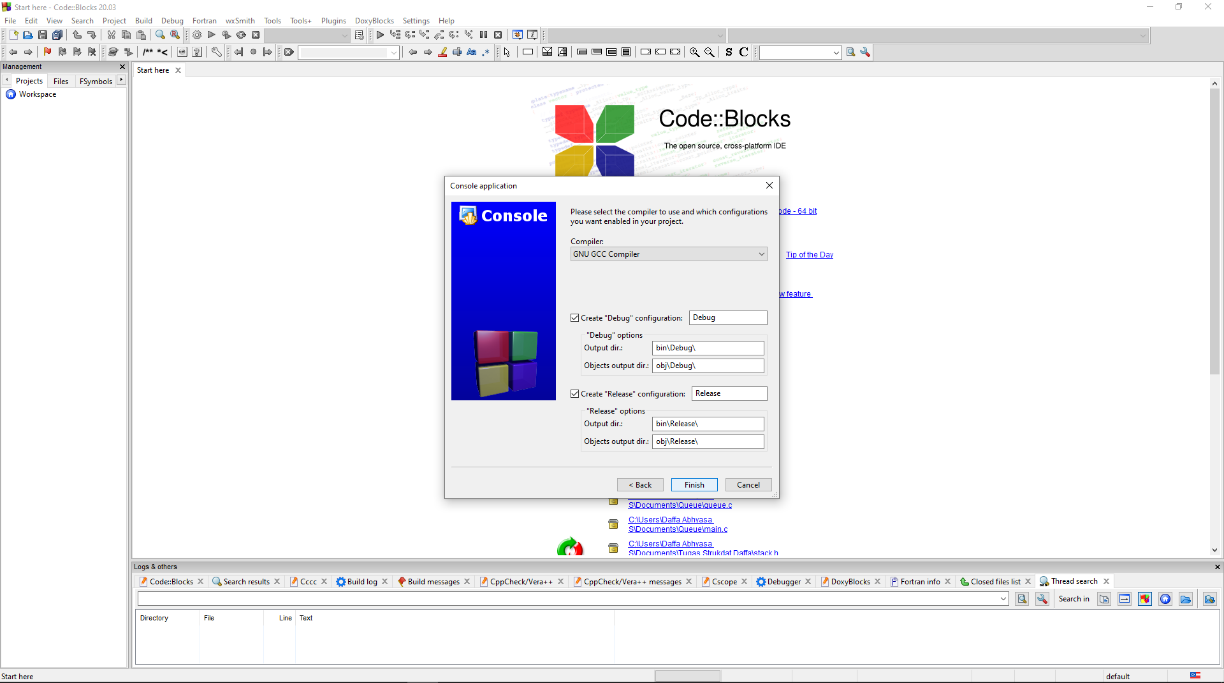
1. Pilih *Console Application*, klik *“Go”*.****
2. Klik “Next”



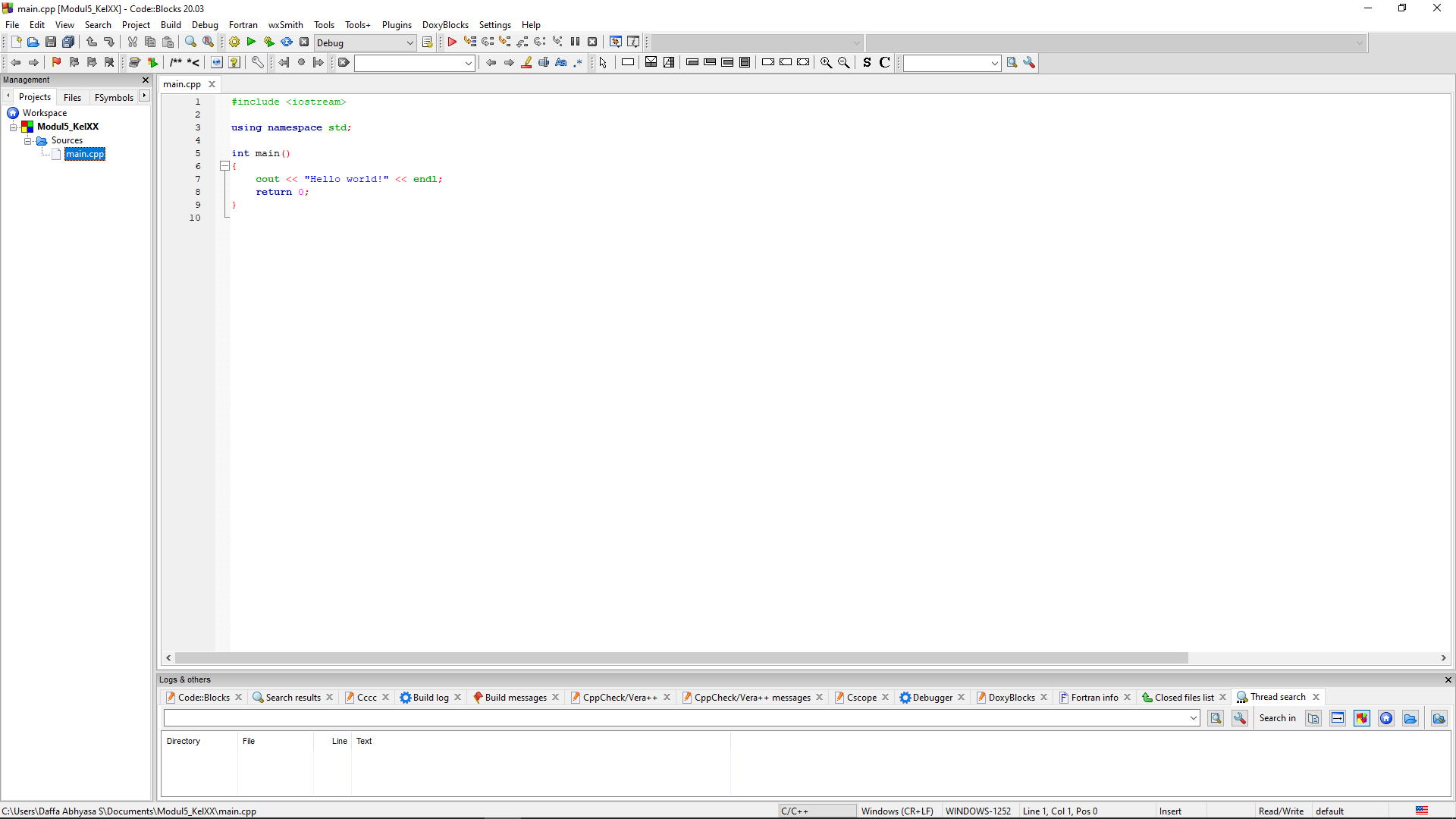
1. Pilih C++ , lalu klik “Next”
2. Ubah *project name* dengan nama “Modul5\_Kelxx”, ganti xx sesuai dengan kelompok. Pilih tempat penyimpanan projek sesuai yang diinginkan.



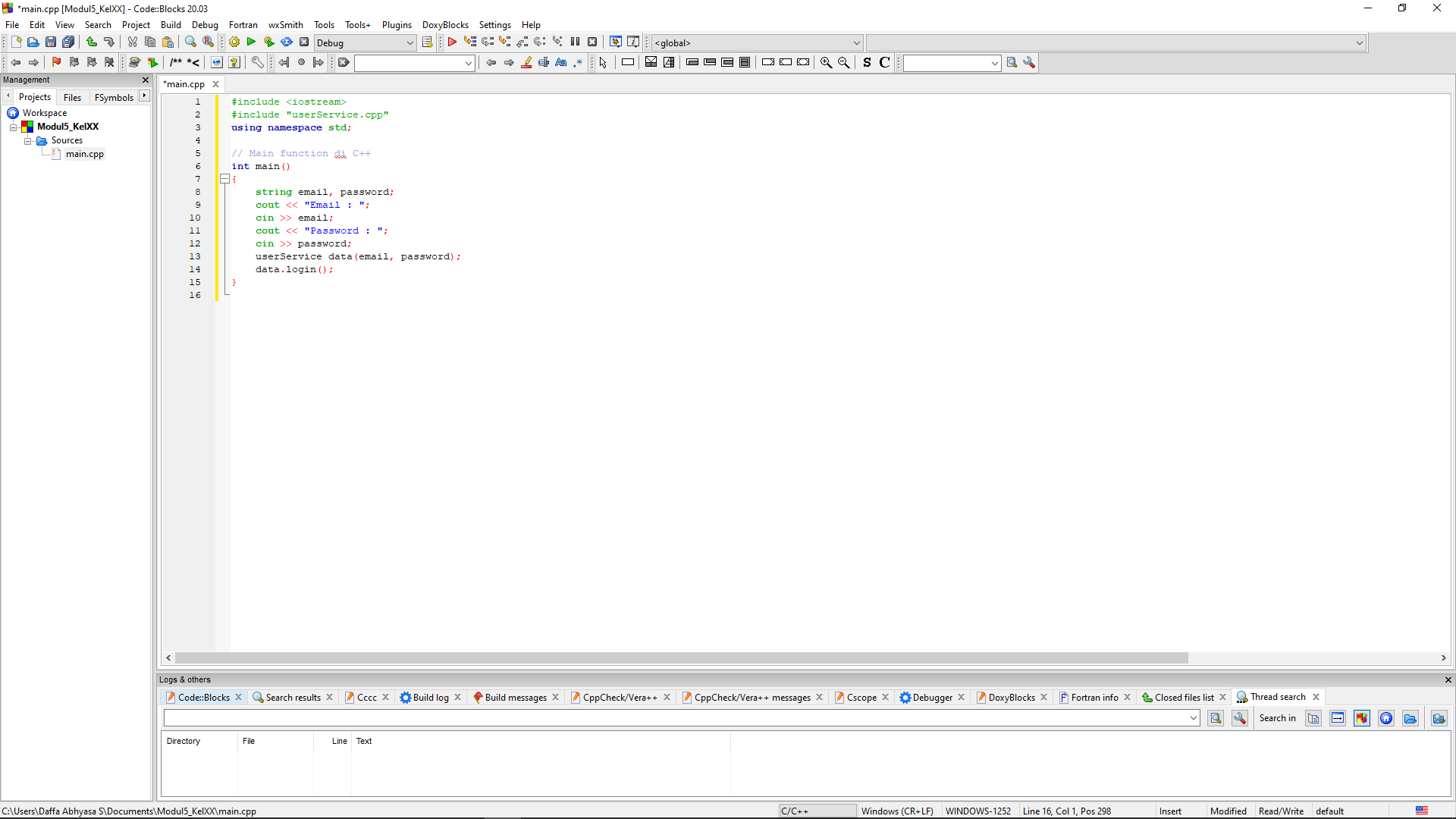
1. Pilih *compiler* yang akan digunakan, lalu klik *“Finish”*



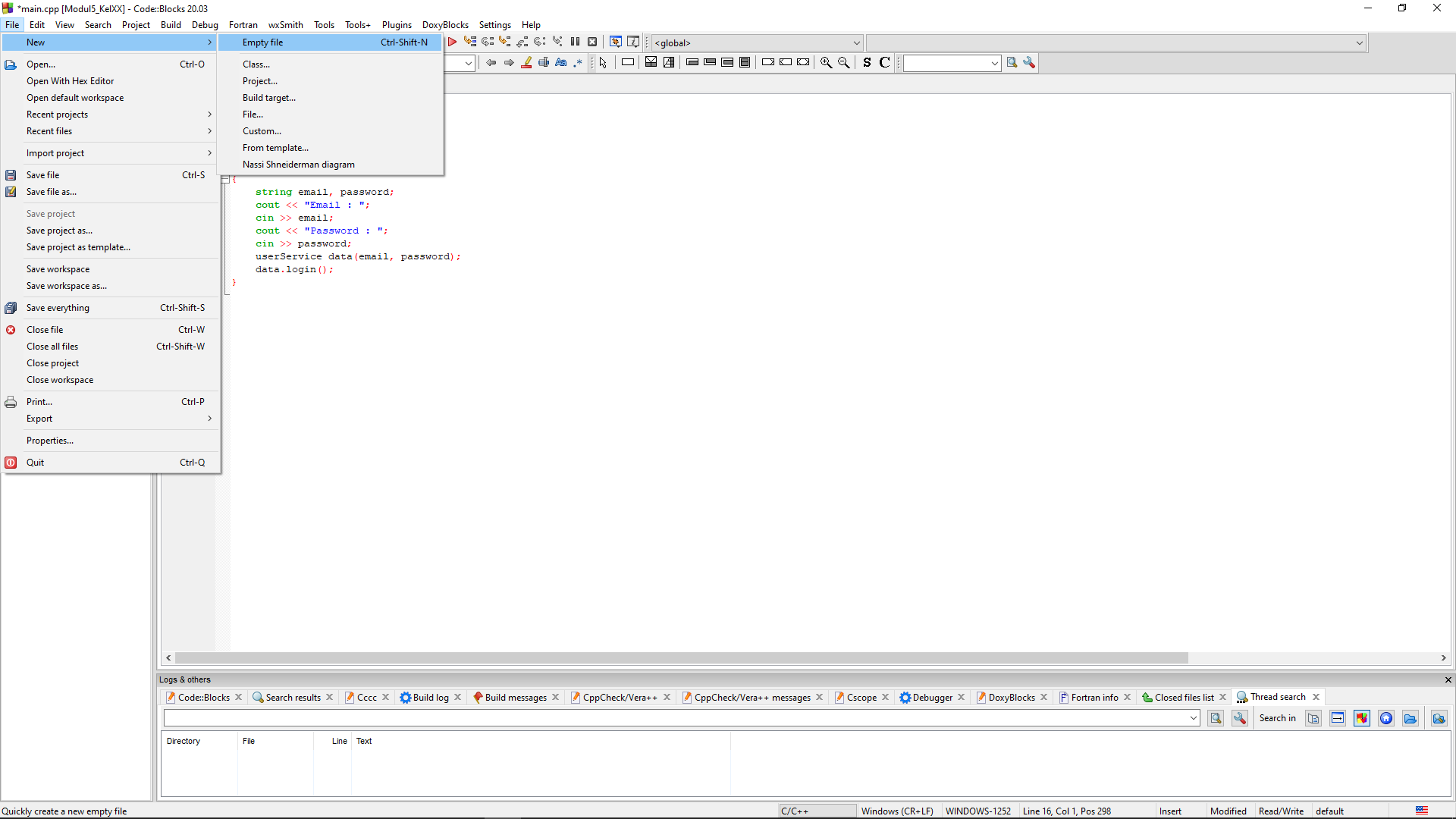
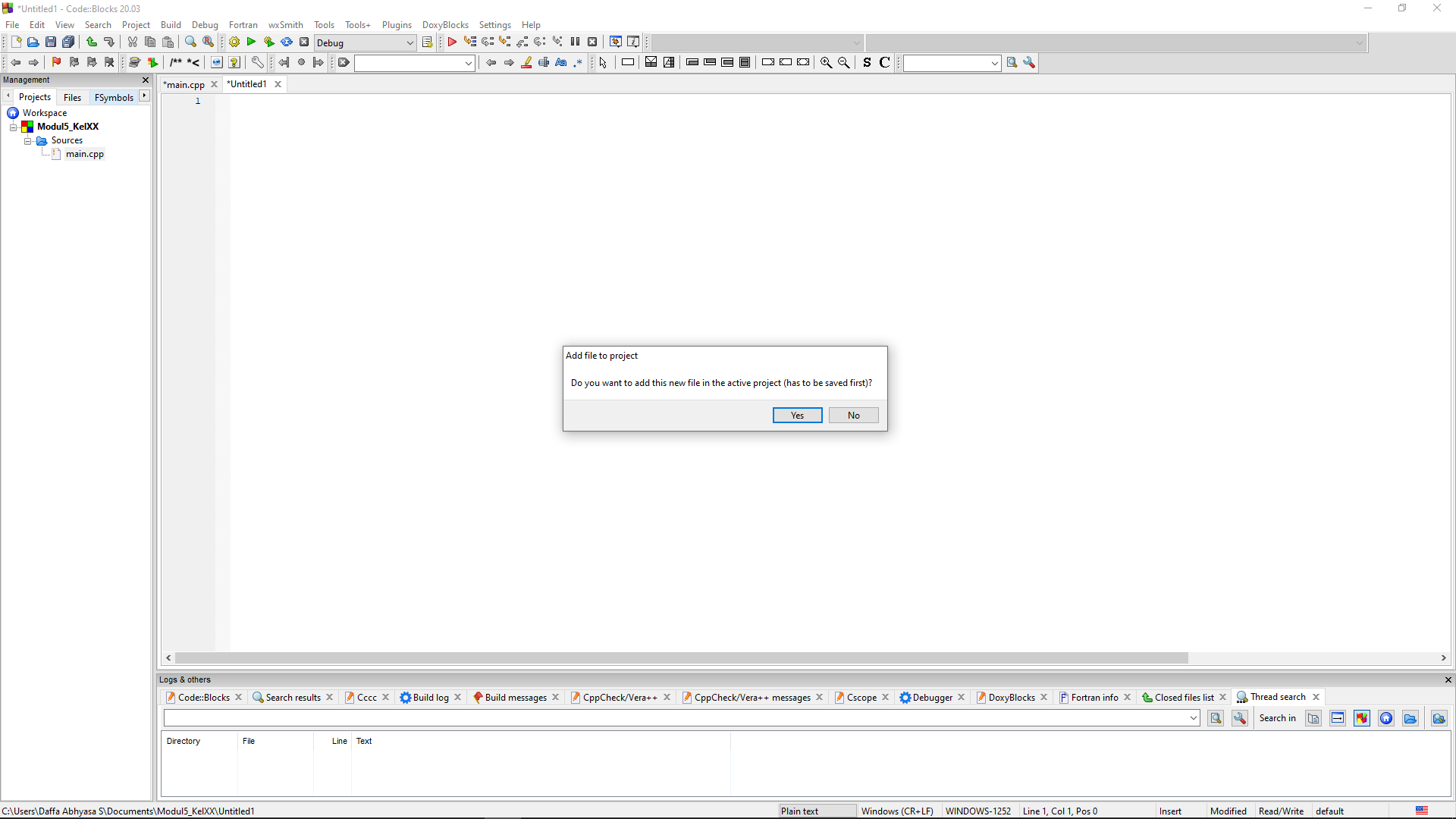
1. Klik tanda tambah disamping *folder* *“Sources”* di tab *Management*. Pilih *file* main.cpp

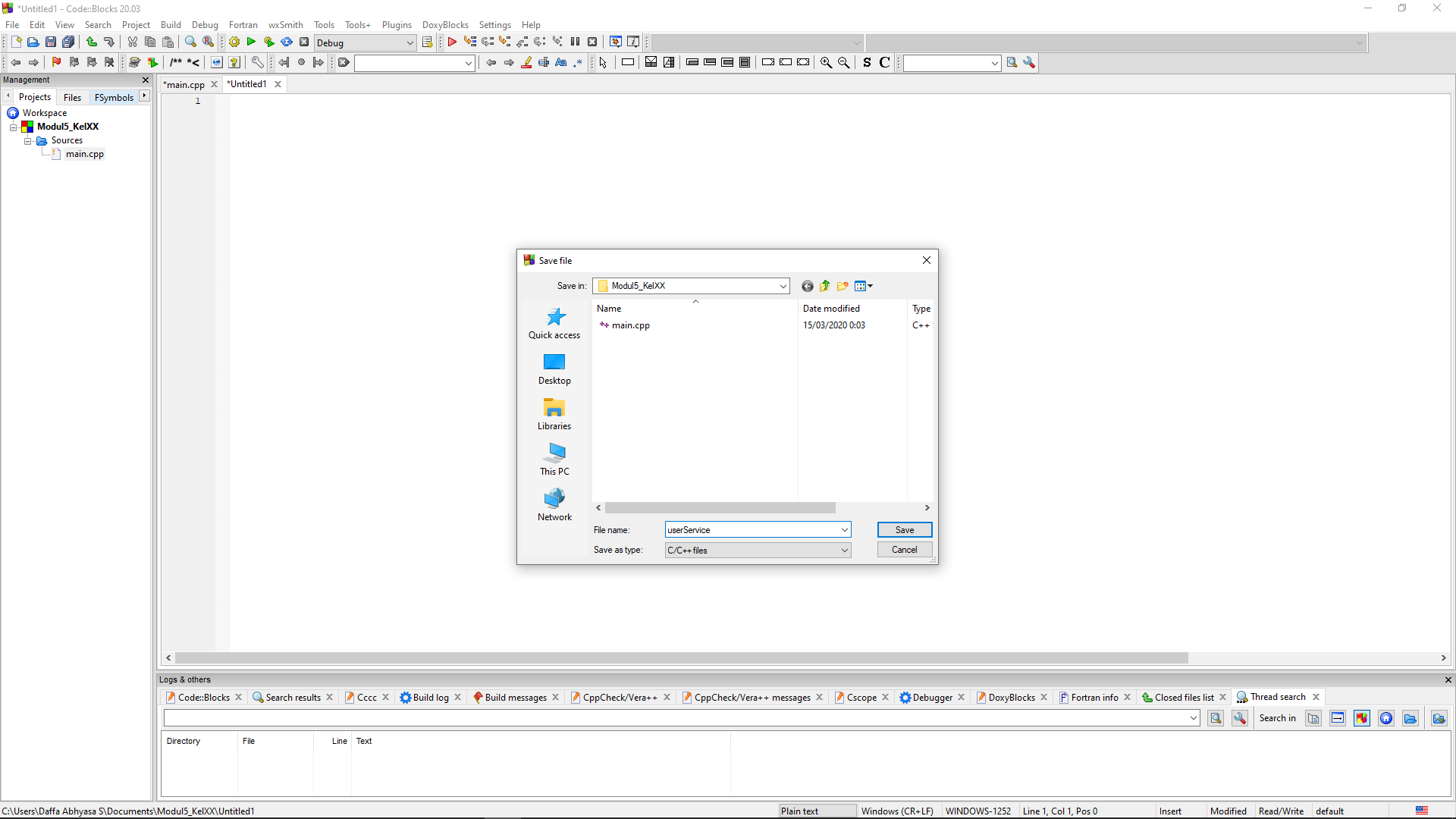
****

1. Tambahkan kode berikut pada main.cpp.

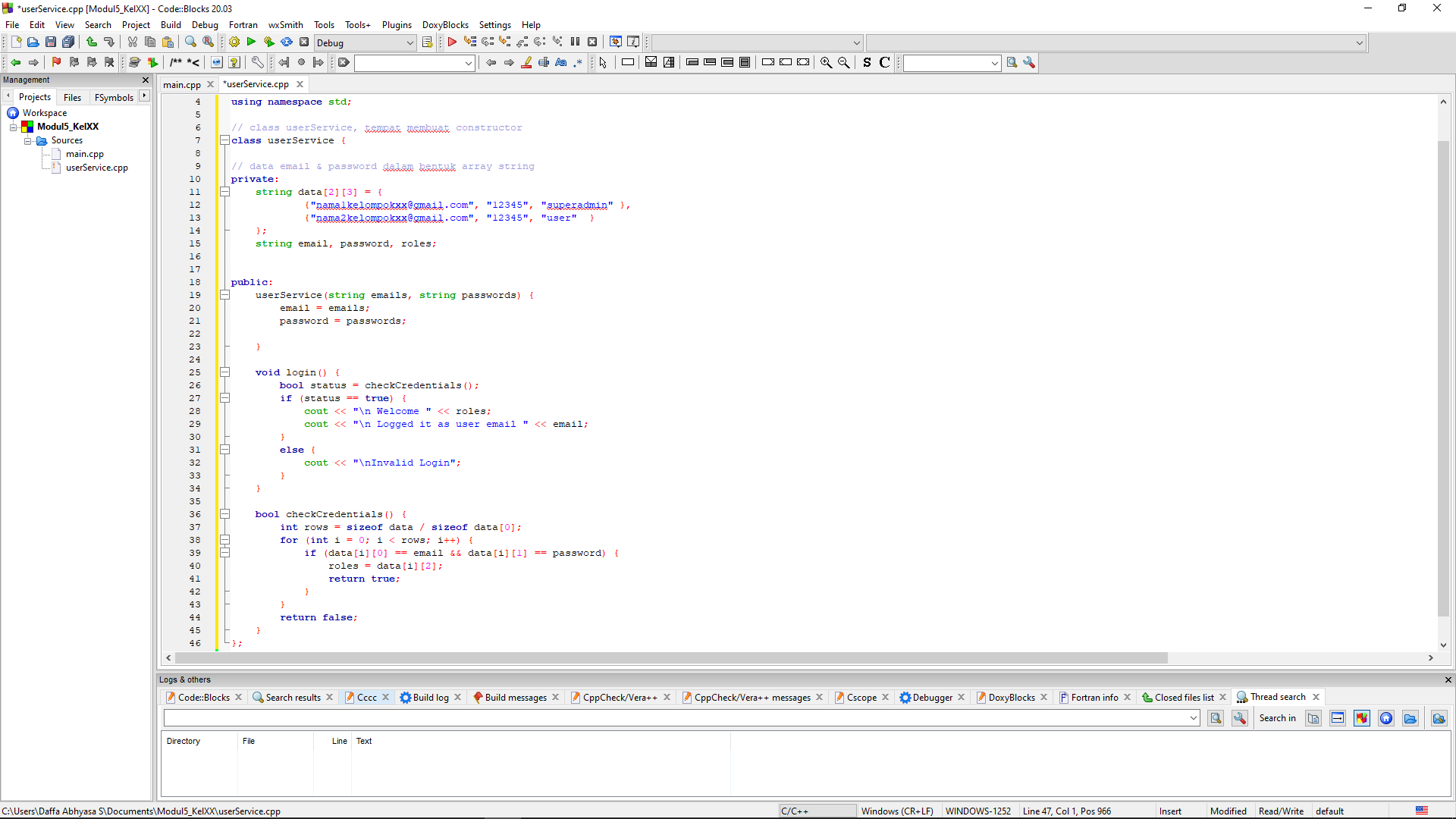
****

|  |
| --- |
| #include <iostream>  #include "userService.cpp"  using namespace std;    // Main function di C++  int main()  {  string email, password;  cout << "Email : ";  cin >> email;  cout << "Password : ";  cin >> password;  userService data(email, password);  data.login();  } |

1. Klik *“File”*, kemudian *“New”*, lalu pilih *“Empty File”*
2. Klik *“Yes”*
3. Beri nama *file* userService.cpp. Klik *“Save”*

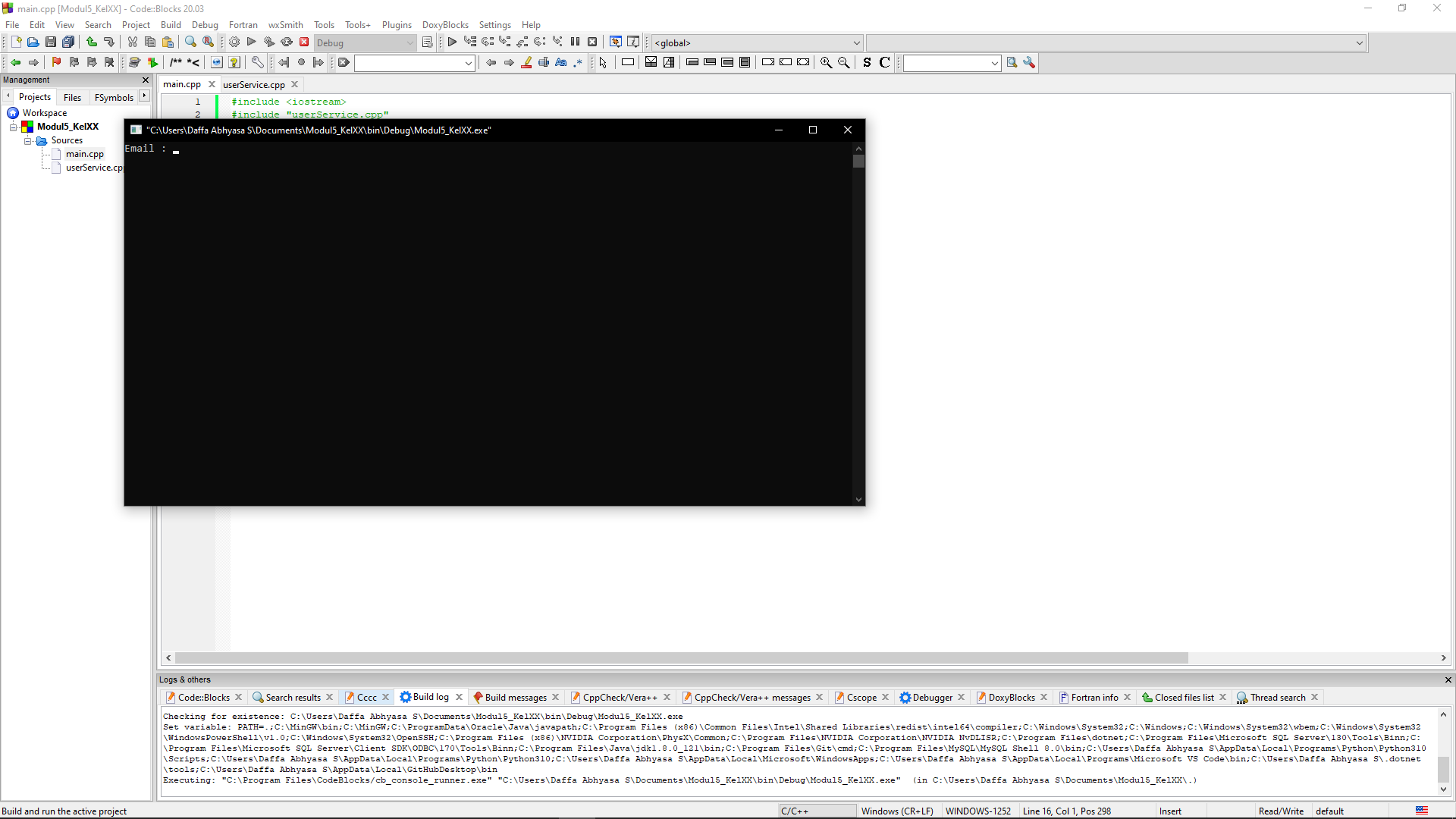
****

1. Tambahkan *code* berikut ke *file* userService.cpp. Ubah alamat *email* sesuai dengan anggota kelompok!

****

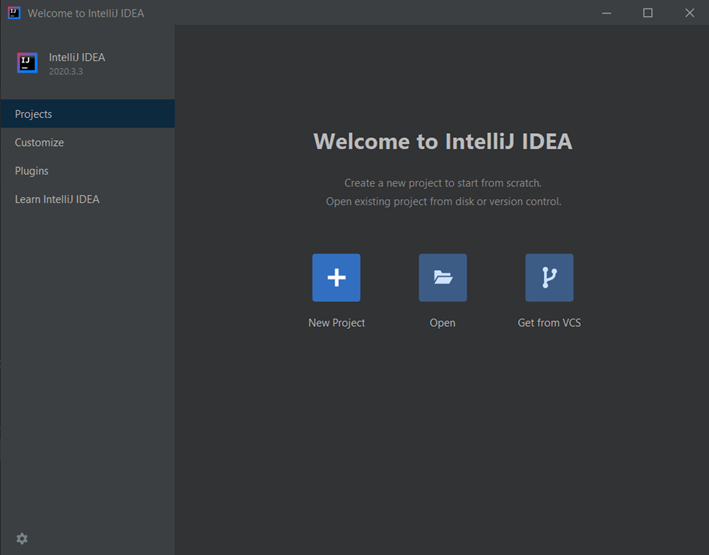
|  |
| --- |
| // header  #include <iostream>  #include <string>  using namespace std;    // *class* userService, tempat membuat *constructor*  *class* userService {    // data email & password dalam bentuk array string  private:  string data[2][3] = {  {"nama1kelompokxx@gmail.com", "12345", "superadmin" },  {"nama2kelompokxx@gmail.com", "12345", "user" }  };  string email, password, roles;      public:  userService(string emails, string passwords) {  email = emails;  password = passwords;    }    void login() {  bool status = checkCredentials();  if (status == true) {  cout << "\n Welcome " << roles;  cout << "\n Logged it as user email " << email;  }  else {  cout << "\nInvalid Login";  }  }    bool checkCredentials() {  int rows = sizeof data / sizeof data[0];  for (int i = 0; i < rows; i++) {  if (data[i][0] == email && data[i][1] == password) {  roles = data[i][2];  return true;  }  }  return false;  }  }; |

1. Jalankan program dengan mengklik *“Build and run”* pada *Toolbar*. Perhatikan keluaran pada konsol!

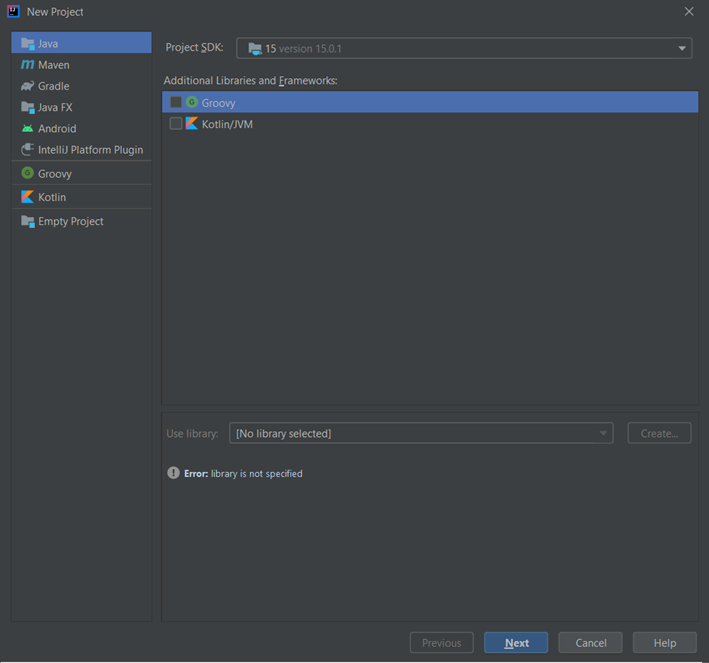
****

### Java

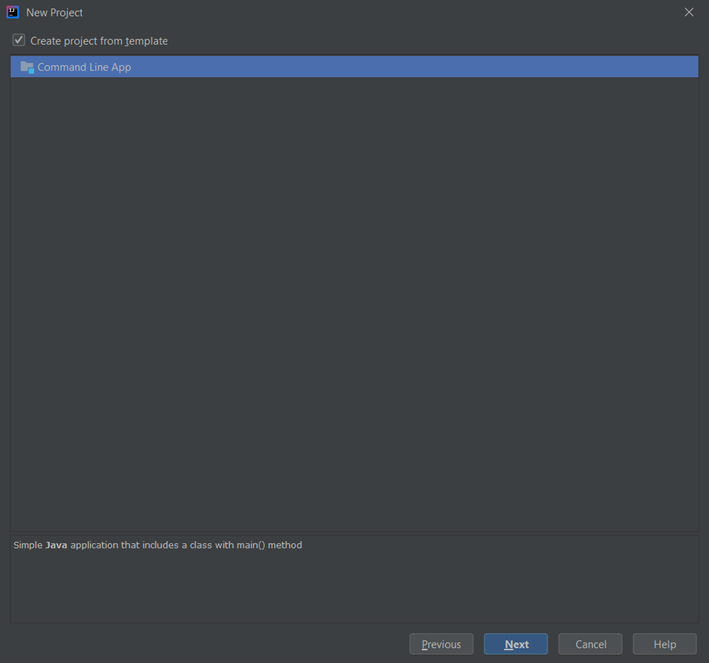
1. Buka IntelliJ IDEA, kemudian buat *project* baru.



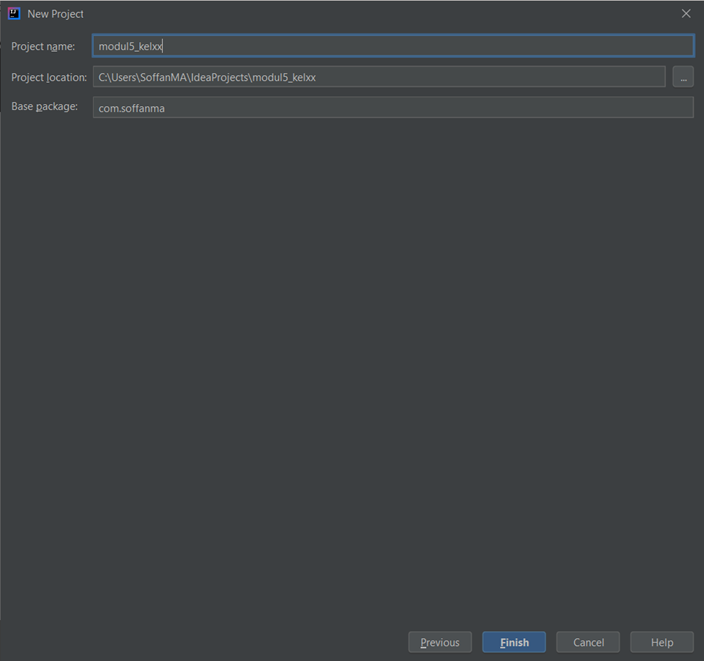
1. Pilih Java, kemudian klik *Next*.



1. Centang *“Create project from template”*, kemudian klik *Next*.



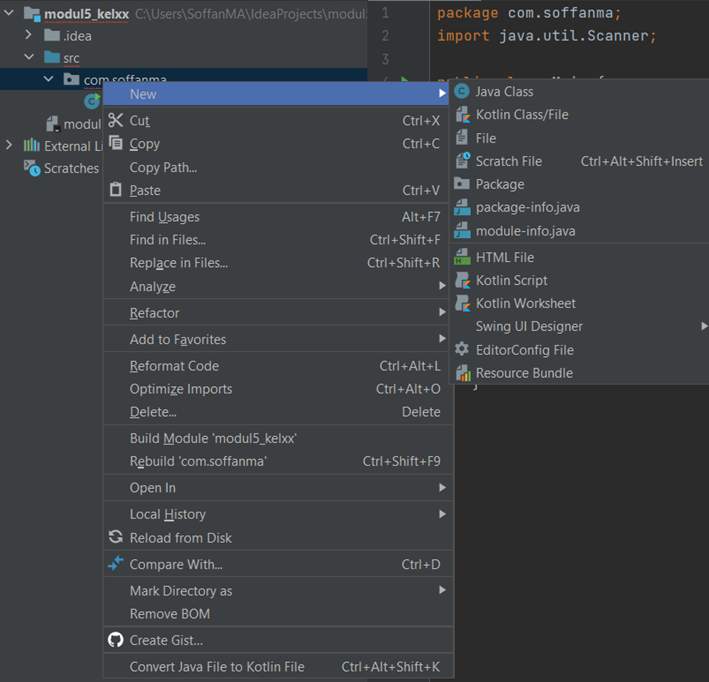
1. Beri nama *project* modul5\_kelxx, kemudian klik *Finish.*



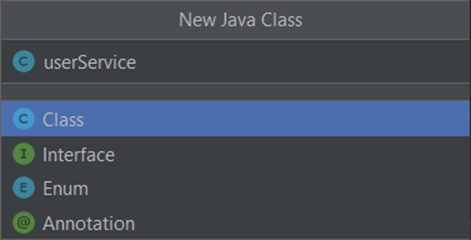
1. Tambahkan *source code* berikut pada *file* Main.java.

|  |
| --- |
|  |

1. Buat sebuah *class* baru dengan klik kanan pada *folder* *package -> New -> Java Class*.



1. Beri nama *class* baru dengan nama “userService”.



1. Tambahkan *source code* berikut pada userService.java. Ubah data sesuai kelompok.

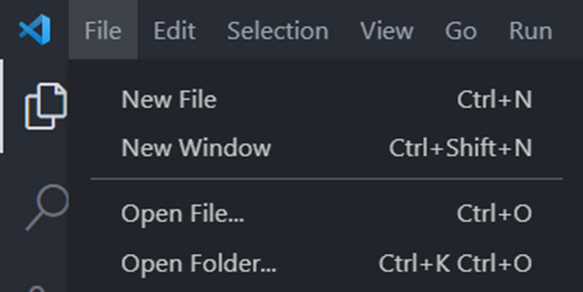
|  |
| --- |
| package com.company; //sesuaikan nama package  public *class* userService {  private String[][] data = new String[2][3];  private String email, password, roles = "";  //ini namanya *constructor* yang akan dijalankan setiap *class* diinisialisasikan  public userService(String emails, String passwords) {  email = emails;  password = passwords;  String[][] data = {  {"nama1@kelompokXX.com", "12345", "supedAdmin"},  {"nama2@kelompokXX.com", "4567", "user"}  };  this.data = data;  }  //method bernama checkCredential  private boolean checkCredential() {  for(int i = 0; i < data.length; i++ ) {  if(data[i][0].equals(email)) {  if(data[i][1].equals(password)) {  roles = data[i][2];  return true;  }  }  }  return false;  }  //method bernama login  public void login() {  boolean status = checkCredential();  if(status == true) {  System.out.println("\nWelcome " + roles);  System.out.println("Logged it as user email " + email);  } else {  System.out.println("\nInvalid Login ");  }  }  } |

1. Jalankan aplikasi dengan menekan tombol *start* pada kanan atas.

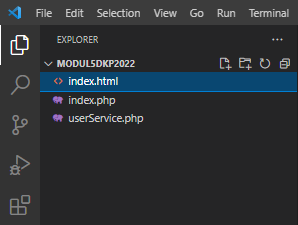


### PHP

1. Buat sebuah *folder* baru bernama Modul5\_KelXX pada direktori C:/xampp/htdocs.
2. Buka Visual Studio Code kemudian buka *folder* yang telah dibuat.



1. Buatlah 3 *file* baru seperti pada gambar.



1. Buka *file* index.html kemudian masukan *source code* berikut.

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Modul 5</title>  </head>  <body>  <form action="./index.php" method="POST">  <h1>Sistem Login Kelompok xx</h1><br>  <input type="email" name="email" placeholder="email">  <input type="password" name="password" placeholder="password">  <button type="submit">Submit</button>  </form>  </body>  </html> |

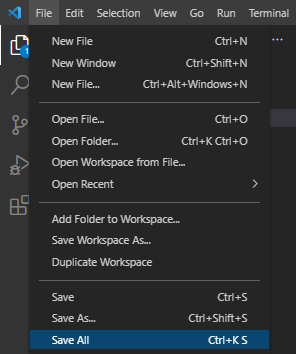
1. Buka *file* index.php kemudian masukan *source code* berikut.

|  |
| --- |
| <?php  include("userService.php");  $user = new userService($\_POST['email'], $\_POST['password']);  if($get\_user = $user->login()) {  echo 'Selamat Datang '.$user->getRole();  echo ', Masuk Sebagai email: '.$get\_user;  } else {  echo 'Invalid Login';  } |

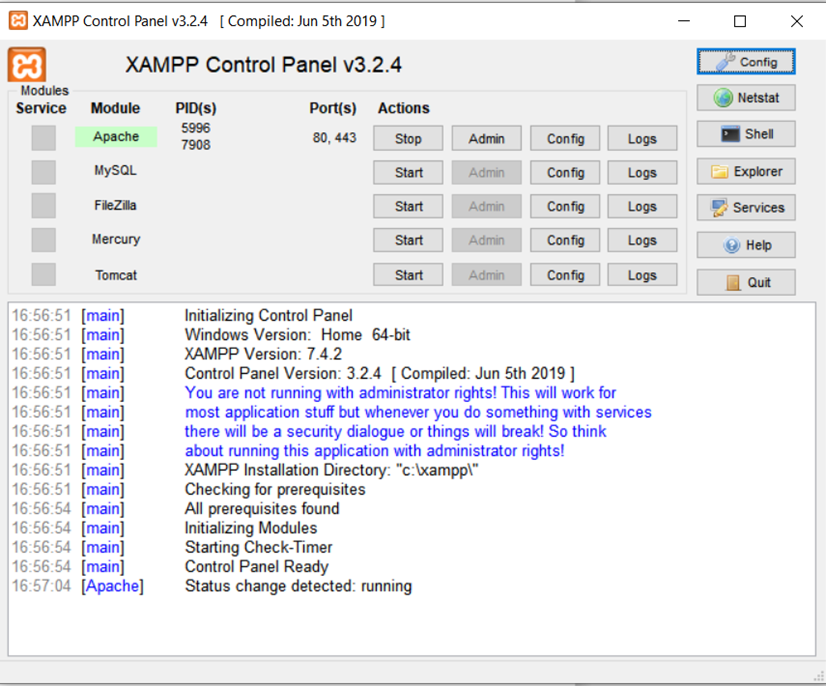
1. Buka *file* userService.php kemudian masukan *source code* berikut. Ubah isi *dummy data* sesuai kelompok kalian.

|  |
| --- |
| <?php  class userService  {  protected $email; // using protected so they can be accessed  protected $password; // and overidden if necessary  protected $dataUser; // dummy data  protected $getRole; // stores the role data  public function \_\_construct($email, $password)  {  $this->\_dataUser = [  (object) [  'email' => "nama1@kelompokXX.com",  'password' => "12345",  'role' => "Super Admin"  ],  (object) [  'email' => "nama2@kelompokXX.com",  'password' => "4567",  'role' => "User"  ]  ];  $this->email = $email;  $this->password = $password;  }  public function login()  {  $user = $this->checkCredentials();  if($user) {  $this->getRole = $user->role;  return $get\_data = $user->email;  } else {  return false;  }  }  protected function checkCredentials()  {  foreach($this->\_dataUser as $key => $value) {  if($value->email == $this->email && $value->password == $this->password) {  return $value;  }  }  return false;  }  public function getRole()  {  return $this->getRole;  }  } |

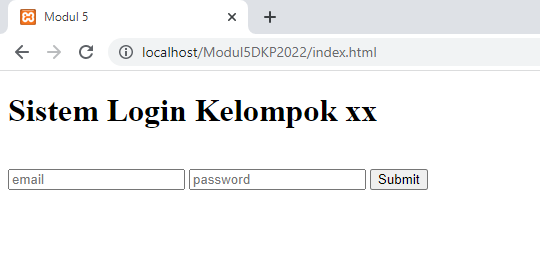
1. Klik *File* -> *Save All.*



1. Buka XAMPP kemudian klik *start* pada modul Apache.



1. Buka *browser* kemudian masukan *url* berikut localhost/nama\_folder/index.html. Ganti nama\_*folder* dengan nama *folder* yang dibuat sebelumnya misal “localhost/modul5\_kel01/index.html”.



1. Uji dengan memasukkan *email* dan *password* yang terdapat dalam *dummy data* maupun yang tidak terdapat dalam *dummy data*.