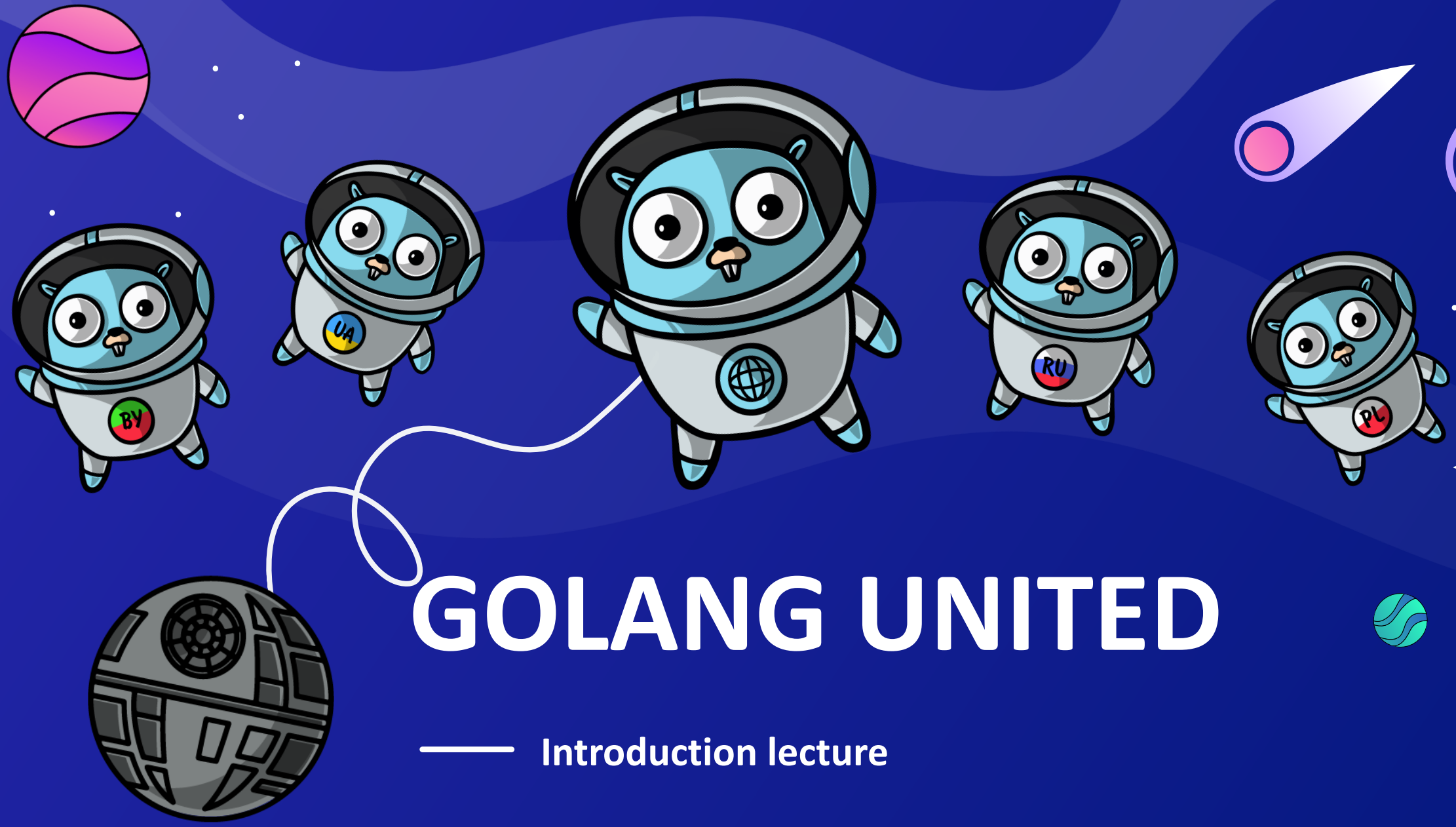


GOLANG UNITED

Introduction lecture

A bit of history

What is Golang, What it’s purpose?

Designed By

Golang United

Ken Thompson

- Designed and implemented the original **Unix OS**
- One of the creators and early developers of the **Plan 9 OS**
- Invented the **B programming language**
- contributions included his work on **regular expressions**
- Worked on **UTF-8**

Rob Pike

- member of the **Unix** team
- Involved in the creation of the **Plan 9**
- **Sam** and **Acme** text editors
- Co-author of **The Practice of Programming** and **The Unix Programming Environment** books.
- Co-creator of **UTF-8**

Robert Griesemer

- The Google's **V8** JavaScript engine
- The **Sawzall** language
- The **Java HotSpot** virtual machine
- The **Strongtalk** system.

Golang United

12 years of jorney

2007 Development started

Development started as internal project at Google

2009 Public release

Golang become open-source project

2012 v1.0 release

Golang released v1.0, language team promised compatibility of all 1.* versions

2015 1.5 release

In 1.4 and 1.5 version internals of the language was refactored, all language tooling was rewritten in Go.

2018 v1.11 release

Go modules release, this change improved dependency management a lot

2022 1.18 release

Generic types and fuzzing testing

Why Golang is so good?

Typing, Compilation, Concurrency, Standard library

Golang United

No legacy.

Golang is used a lot for cloud native development

Golang is simple and fast

Concurrent by design

Golang United

In real world Golang used a lot, for writing distributed systems.

Such companies as (Google, Lyft, Uber, Twitter and Facebook, Gitlab) already use Golang for their systems.

A lot of systems which drive current world such as Docker, Kubernetes and Google Cloud written at Golang as well.

Golang United

Huge standart library(https://pkg.go.dev/std):

- https://pkg.go.dev/compress@go1.17.6 — compression via different algorithms
- https://pkg.go.dev/crypto@go1.17.6 — supports many cryptographic algorithms
- https://pkg.go.dev/encoding@go1.17.6 — supports many encoding formats
- https://pkg.go.dev/net/http@go1.17.6 — built-in http server with http1.1/http2 support
- https://pkg.go.dev/html@go1.17.6 — built-in html template engine
- https://pkg.go.dev/text@go1.17.6 — built-in text template engine
- https://pkg.go.dev/database/sql@go1.17.6 — built-in sql library

The key differences of Golang

Type system, Interfaces, Concurrency

Golang United

Golang is explicit language

Code is the main documentation of itself

By default, all parameters passed by value not a reference

Using value type in most cases faster then use references

Code generation over generic programming (So far)

Data structures is simple, there is no such Collection framework (so far)

Golang United

Strong typing — every variable in golang has type

```
var d1 int64
var d2 int32
var d3 int
```

Type system

```
var (
    d1 int64
    d2 int32
)

fmt.Println(d1 == d2)
```

Compilator do the work for you

```
# github.com/burov/snipets/webserver
./main.go:22:17: invalid operation: d1 == d2 (mismatched types int64 and int32)
```

Golang United

Concurrency dramatically simple

```go
var wg sync.WaitGroup
message := []string{"Hello", "from", "Golang", "United", "Team"}
for _, str := range message {
    wg.Add( delta: 1)

    go func(s string) {
        fmt.Println(s)
        wg.Done()
    }(str)
}

wg.Wait()
```

Output:

```
Team
Hello
United
Golang
from
```

Golang United

An example of simple web-server

```go
package main

import (
    "fmt"
    "net/http"
)

func main() {

    http.HandleFunc( pattern: "/", func(w http.ResponseWriter, r *http.Request) {
        _, _ = fmt.Fprintf(w,  format: "Hello World")
    })

    host := "localhost:8080"
    fmt.Printf( format: "Listen and Serve on %q\n", host)
    if err := http.ListenAndServe(host,  handler: nil); err != nil {
        panic(err)
    }
}
```

% curl http://localhost:8080/
Hello World

Golang comparison

Golang United

Strong typing — every variable in golang has type

```java
package com.company;

public class Main {

    public static void main(String[] args) {
        Integer d1 = 0;
        long d2 = 0;

        System.out.println(d1 == d2);
    }
}
```

Output

true

Golang United

Strong typing — every variable in golang has type

```
var d1 int64
var d2 int32
var d3 int
```

Type system

```
var (
    d1 int64
    d2 int32
)

fmt.Println(d1 == d2)
```

Compilator do the work for you

```
# github.com/burov/snipets/webserver
./main.go:22:17: invalid operation: d1 == d2 (mismatched types int64 and int32)
```

Golang United

Java:

- Types system is very complex, some of the decisions have to be made, due to the initial design weakneses

Golang:

- Type system realy simple and explicit, there no hacks in it

Golang United

```go
package main

import "fmt"

type Printer interface {
    Print(s string)
}

type ConsolePrinter struct{}

func (c *ConsolePrinter) Print(s string) {
    fmt.Println(s)
}

func main() {
    var p Printer = &ConsolePrinter{}

    p.Print( s: "Hello World")
}
```

```java
package com.company;

interface Printer {
    public void print(String s);
}

class ConsolePrinter implements Printer {
    public void print(String s) {
        System.out.println(s);
    }
}

public class Main {

    public static void main(String[] args) {
        Printer printer = new ConsolePrinter();

        printer.print("Hello World");
    }
}
```

Golang United

Java:

- Interface should be defined by producer
- Interface should be explicitly implemented by Class

Golang:

- Interface might be defined by consumer or producer
- Interface implicitly implemented if type has all the methods defined in interface (Duck typing)

Golang United

Concurrency in Java

```java
package com.company;

public class Main {

    public static void main(String[] args) {
        var message = new String[]{"Hello", "from", "Golang", "United", "Team"};

        for (String msg : message) {
            new Thread(() -> {
                System.out.println(msg);
            }).start();
        }
    }
}
```

Output:

```
Team
Hello
United
Golang
from
```

Golang United

Concurrency dramatically simple

```go
var wg sync.WaitGroup
message := []string{"Hello", "from", "Golang", "United", "Team"}
for _, str := range message {
    wg.Add( delta: 1)

    go func(s string) {
        fmt.Println(s)
        wg.Done()
    }(str)
}

wg.Wait()
```

Output:

```
Team
Hello
United
Golang
from
```

Golang United

Java:

- Create system thread for each Thread object
- Static stack
- Communication by shared memory

Golang:

- Go routines is a local object, no system thread creation
- Dynamic stack, each go routine can have stack size up to a few gigabytes
- Communication by shared memory or built-in messaging (channels)

Golang United

An example of simple web-server in Java

```java
package org.example;

public class App {
    public static String getHello() { return "Hello world"; }
}
```

% curl http://localhost:8080/
Hello World

Golang United

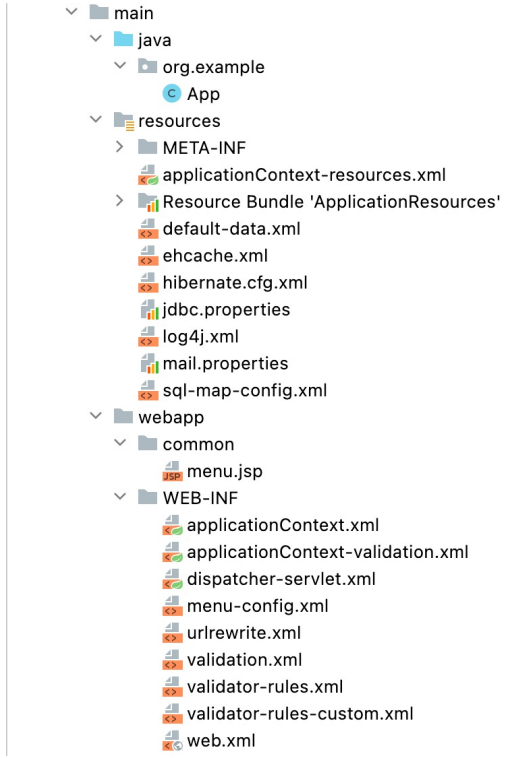
An example of simple web-server

```go
package main

import (
    "fmt"
    "net/http"
)

func main() {

    http.HandleFunc( pattern: "/", func(w http.ResponseWriter, r *http.Request) {
        _, _ = fmt.Fprintf(w,  format: "Hello World")
    })

    host := "localhost:8080"
    fmt.Printf( format: "Listen and Serve on %q\n", host)
    if err := http.ListenAndServe(host,  handler: nil); err != nil {
        panic(err)
    }
}
```

% curl http://localhost:8080/
Hello World

Golang United

Web project configuration in Java



Golang United

Golang:

- Built-in server
- Additional configuration isn’t required
- No external depedencies
- Build just with plain compiler

Java:

- Additional server required before servlets container
- Required configuration for frameworks
- Servlets containers and JavaEE libs required
- Gradle or Maven requied to build project

About the course

How the process looks like?

Golang United

What you will learn

0. Introduction lecture
1. Golang basic syntax
2. Golang memory model
3. Golang data structures
4. Golang byte, rune, string & error
5. Golang structures & methods & interfaces
6. Golang interfaces & OOP in Golang
7. STD packages io, context, os
8. Database & How to use in Golang
9. Web development with Golang
10. Concurrency in Golang

#First stage

#Individual phase

How you will learn

- **1** lecture in a week
- Homework each week
- Additional materials after each week
- Test after some lectures

What you will learn

- How to cooperate inside a team
- How to build scalable, distributed applications
- How to build CI/CD pipelines
- How to work in an agile environment
- Will work with such tools as Docker, Gitlab CI/CD

#Second stage

#Team work

Golang United

How to

How to get your place in **#second_stage** shuttle ?

- Complete all your homework, on **last lecture date + 2 weeks**
- Be prepared to speak about what you've learned

Development environment

Goland

- *Install Golang – https://golang.org/doc/install*
- *Install IDE – https://www.jetbrains.com/go/download*
- *Students license – https://www.jetbrains.com/student/*

Devlopment environment

Visual Studio Code

- *Install Golang - https://golang.org/doc/install*
- *Install VS Code - https://code.visualstudio.com/download*
- *Add support for Golang – https://code.visualstudio.com/docs/languages/go*

Devlopment environment

Vim

- *Install Golang - https://golang.org/doc/install*
- *Install Vim - https://www.vim.org/download.php*
- *Generate .vimrc file and download it https://vim-bootstrap.com or install go vim plugin https://github.com/fatih/vim-go (this is more for vim experts)*

Homework

Golang United

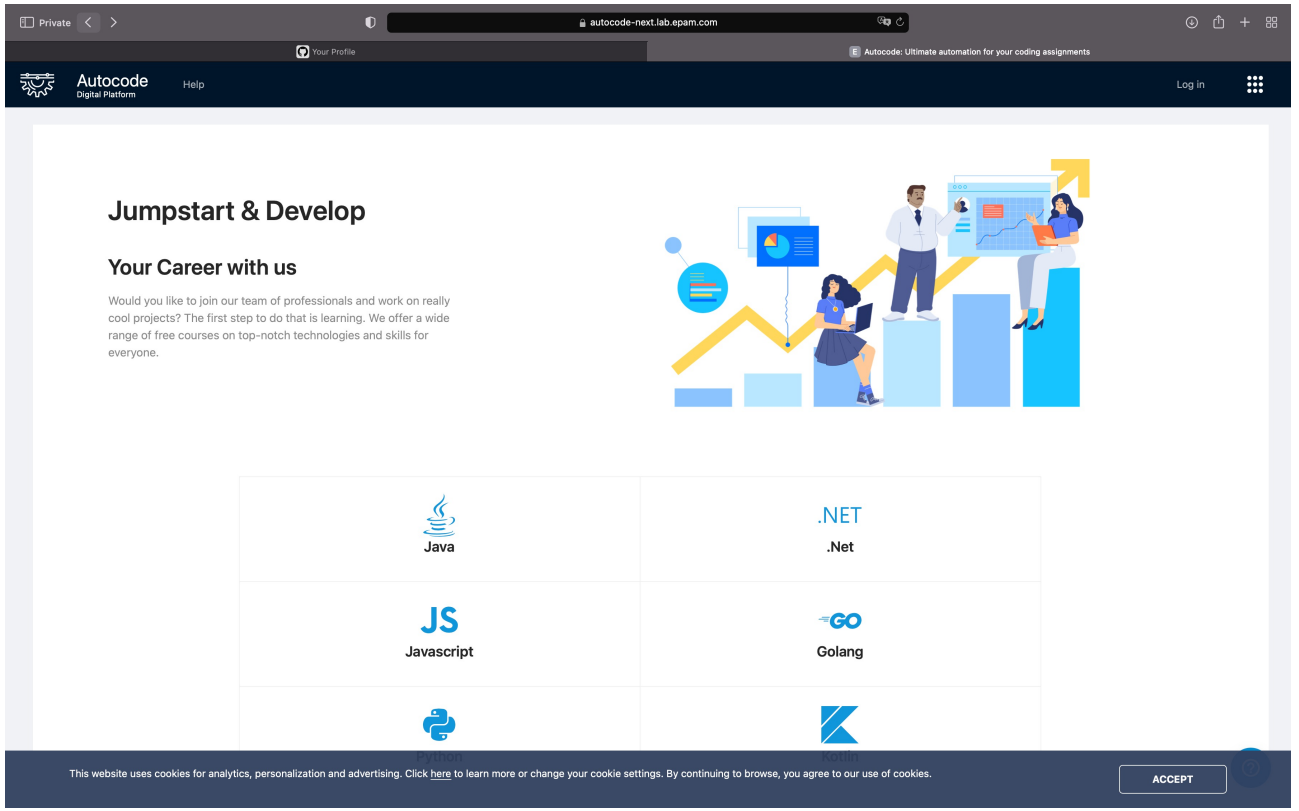
https://autocode-next.lab.epam.com

- Complete task for "Introduction module”
- Find instruction below

Golang United

Find platform for homework verification on

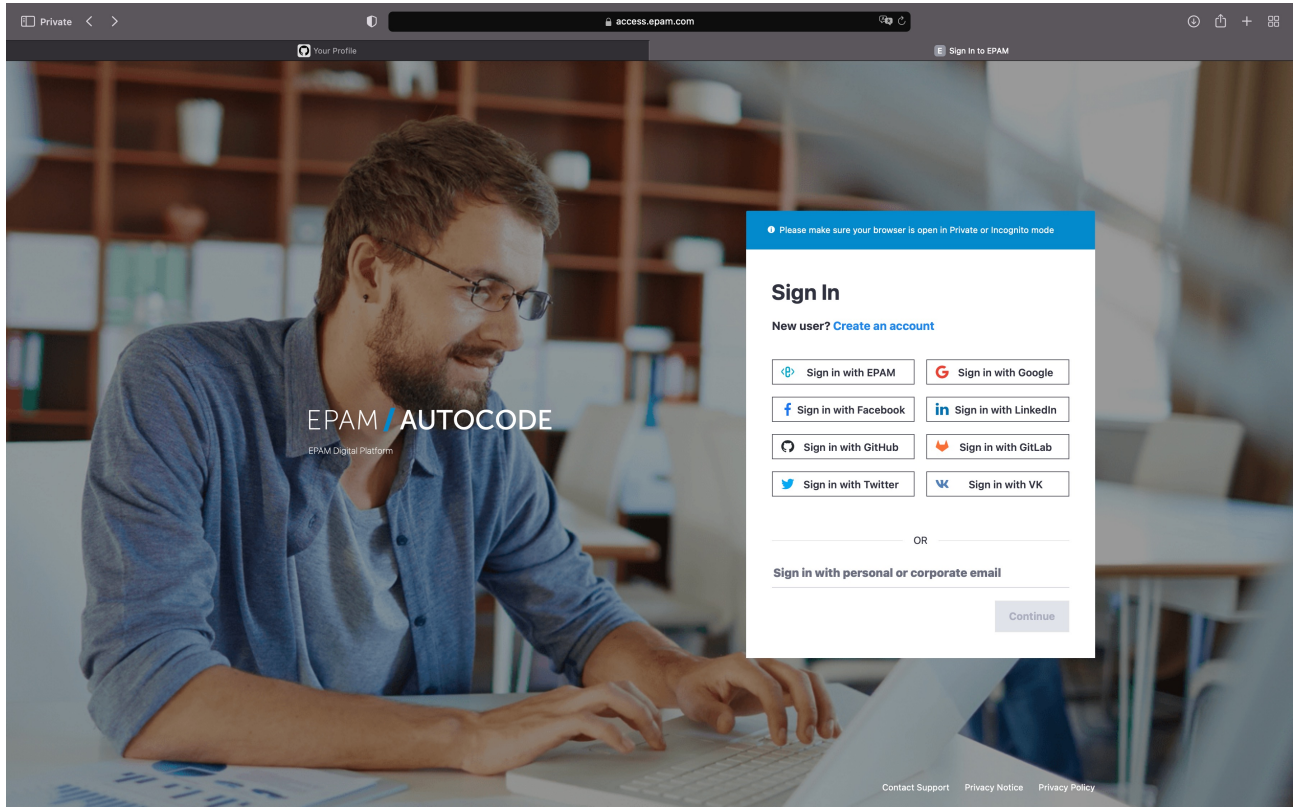
https://autocode-next.lab.epam.com



Golang United

Login to the platform

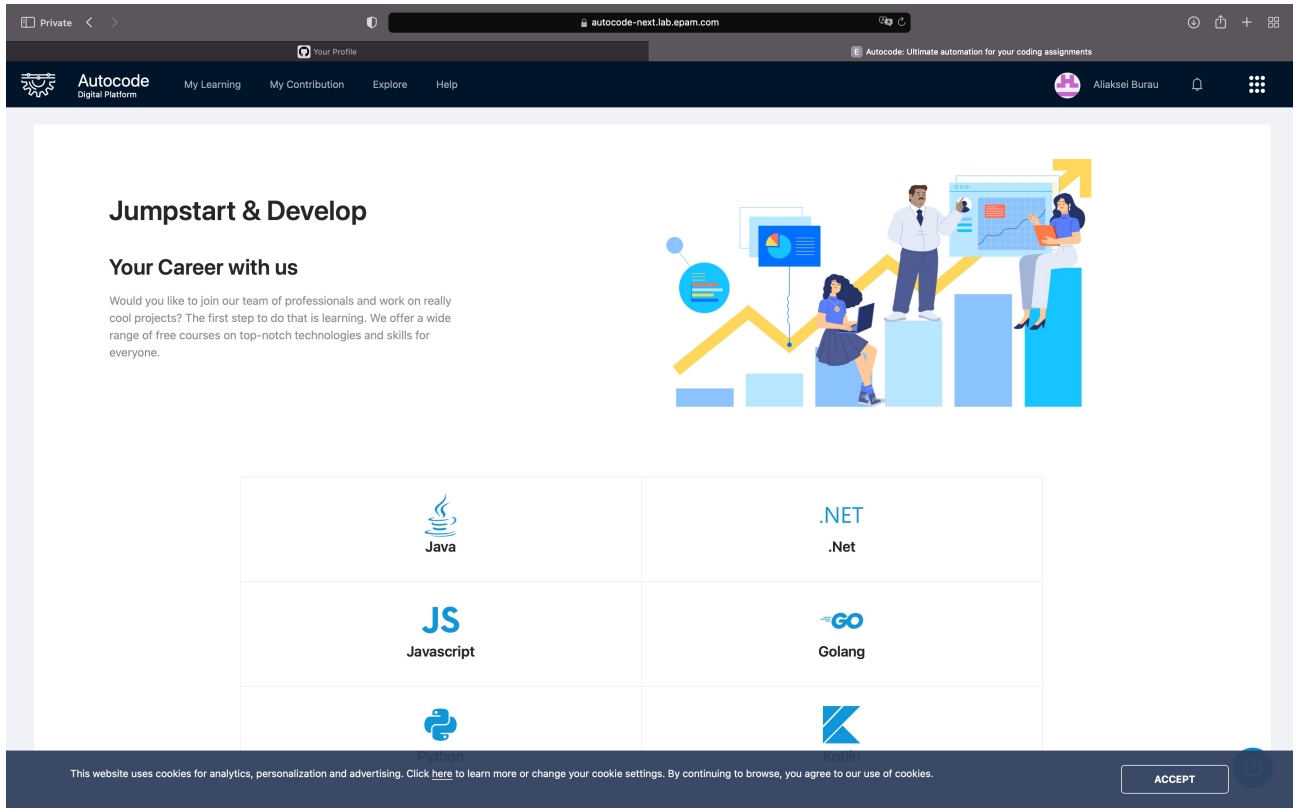
- Use **github** or **gitlab** account



Golang United

Main page

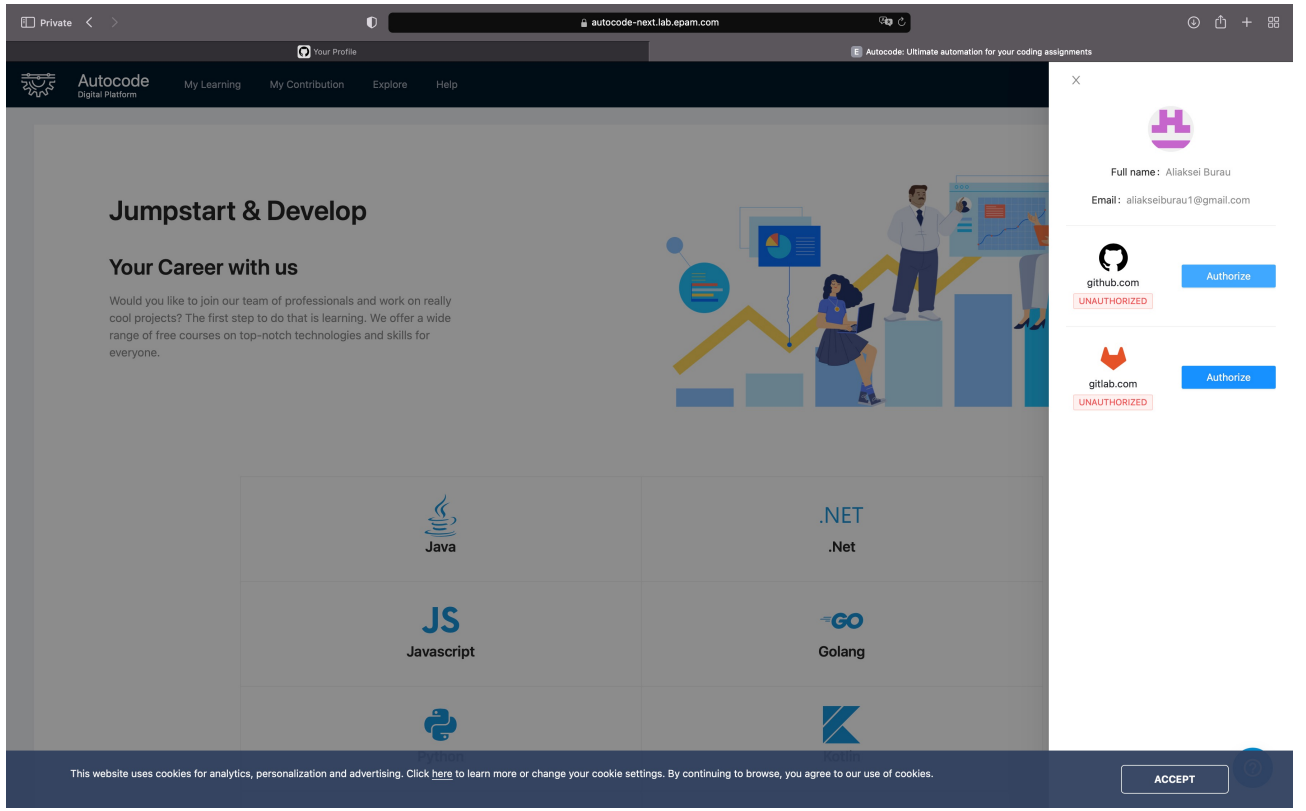
- Verify that you’re signed-in



Golang United

Authenticate platform to your github/gitlab account

- Go to ”My Profile” tab
- Give permissions to the **github** or **gitlab**
- Avoid using both

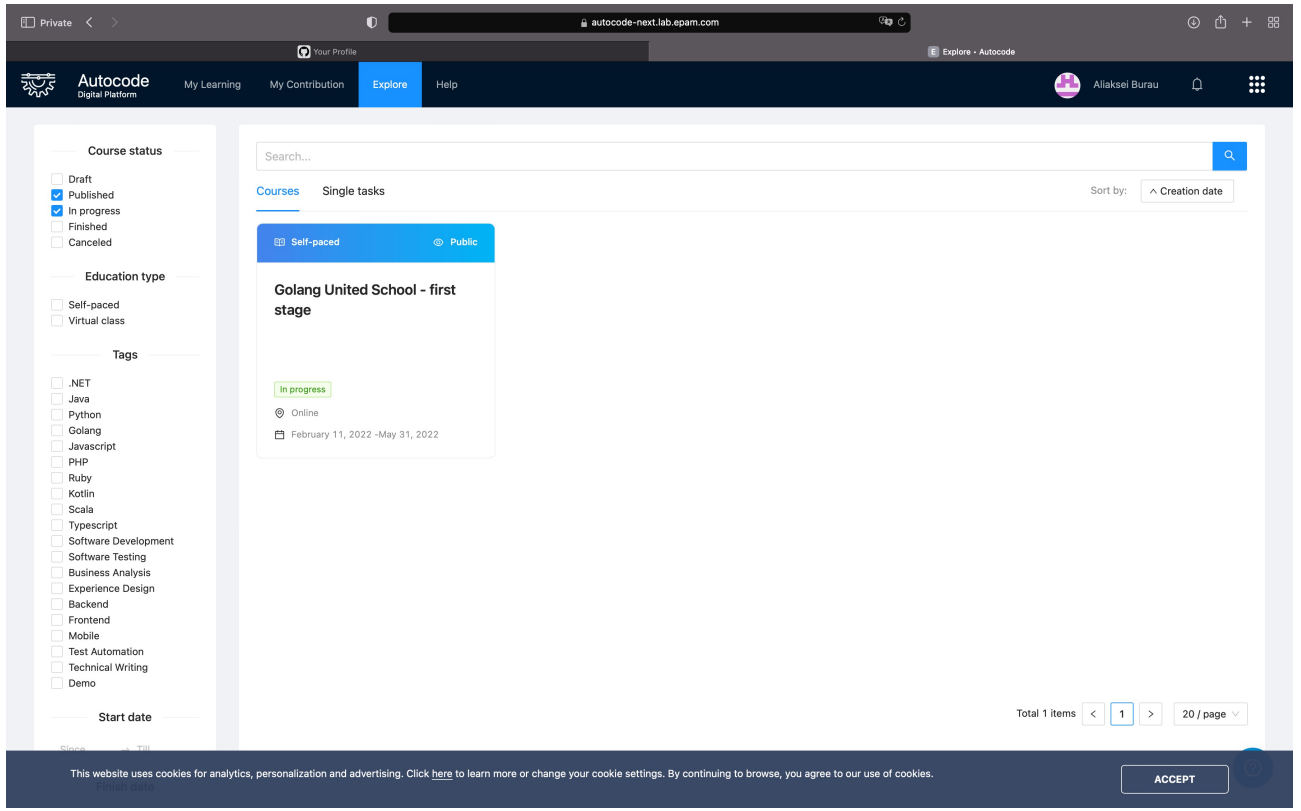


Golang United

Find the course

- Go to the **Explore** tab
- Find course with name

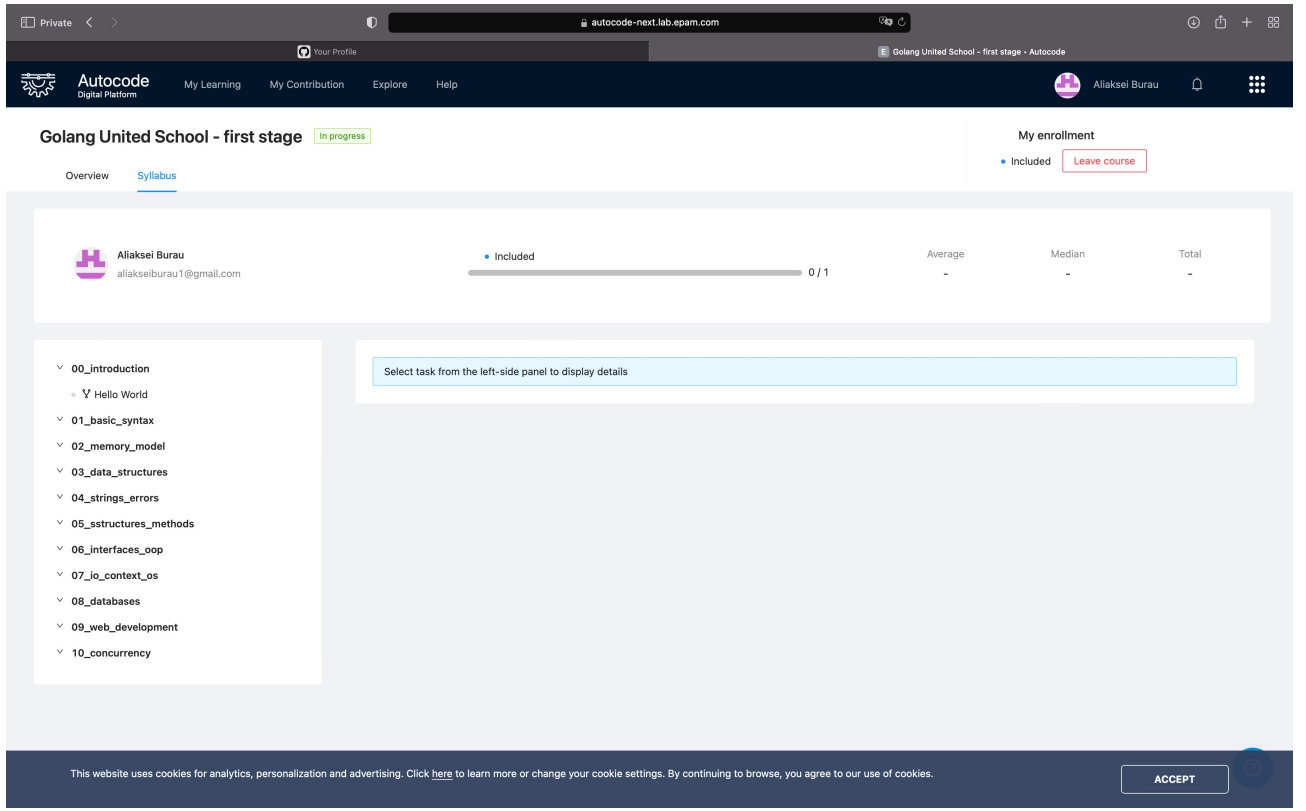
”**Golang United School - first stage**”



Golang United

Join the course

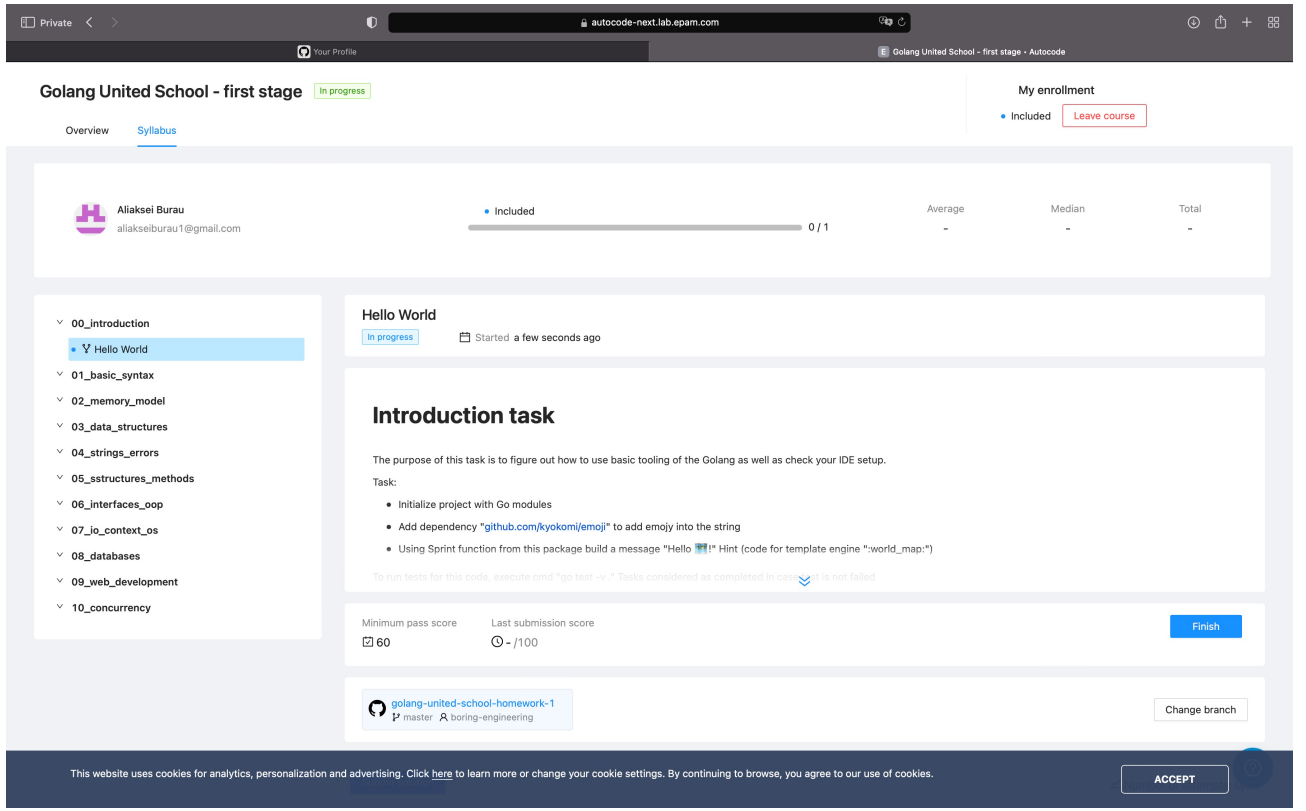
- Press **”enroll”** button



Golang United

Choose the task

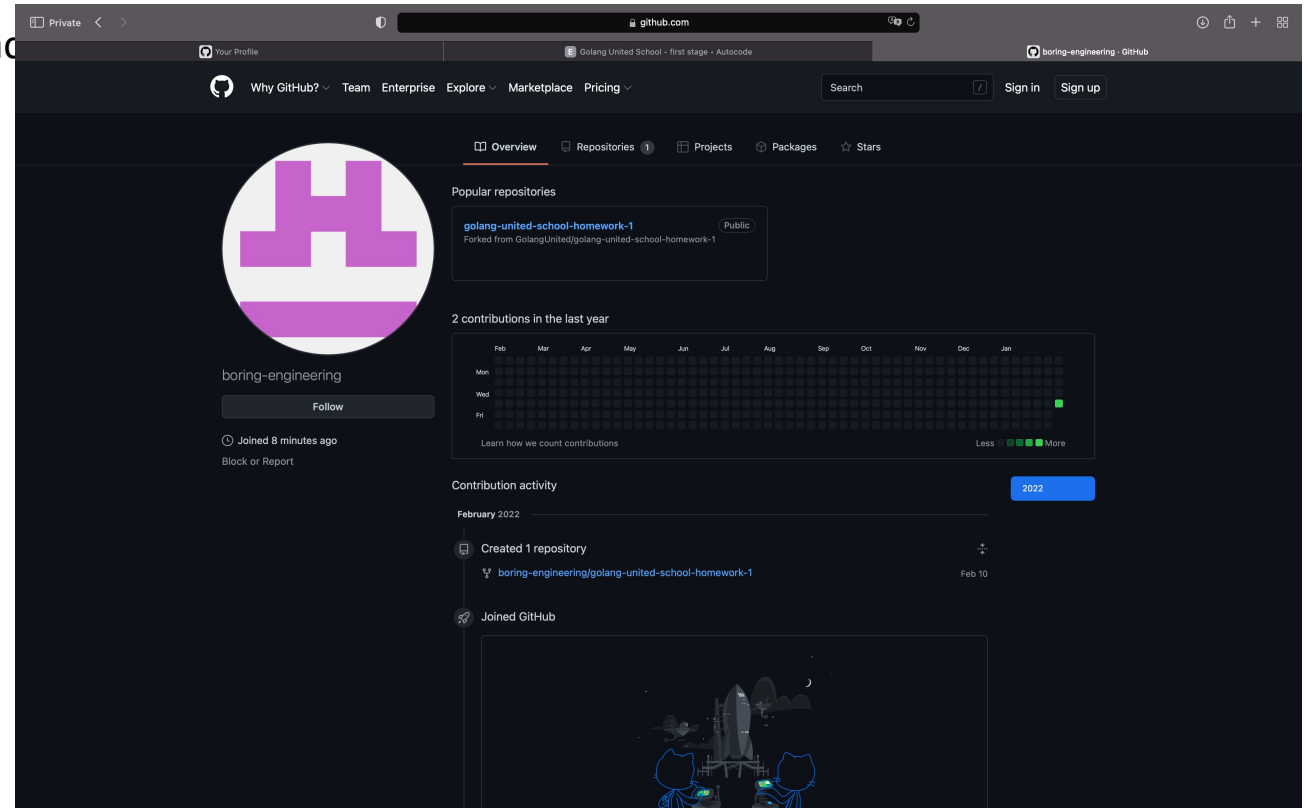
- Find task for lecture
- Press the **start** button



Golang United

Check that you have forked repo

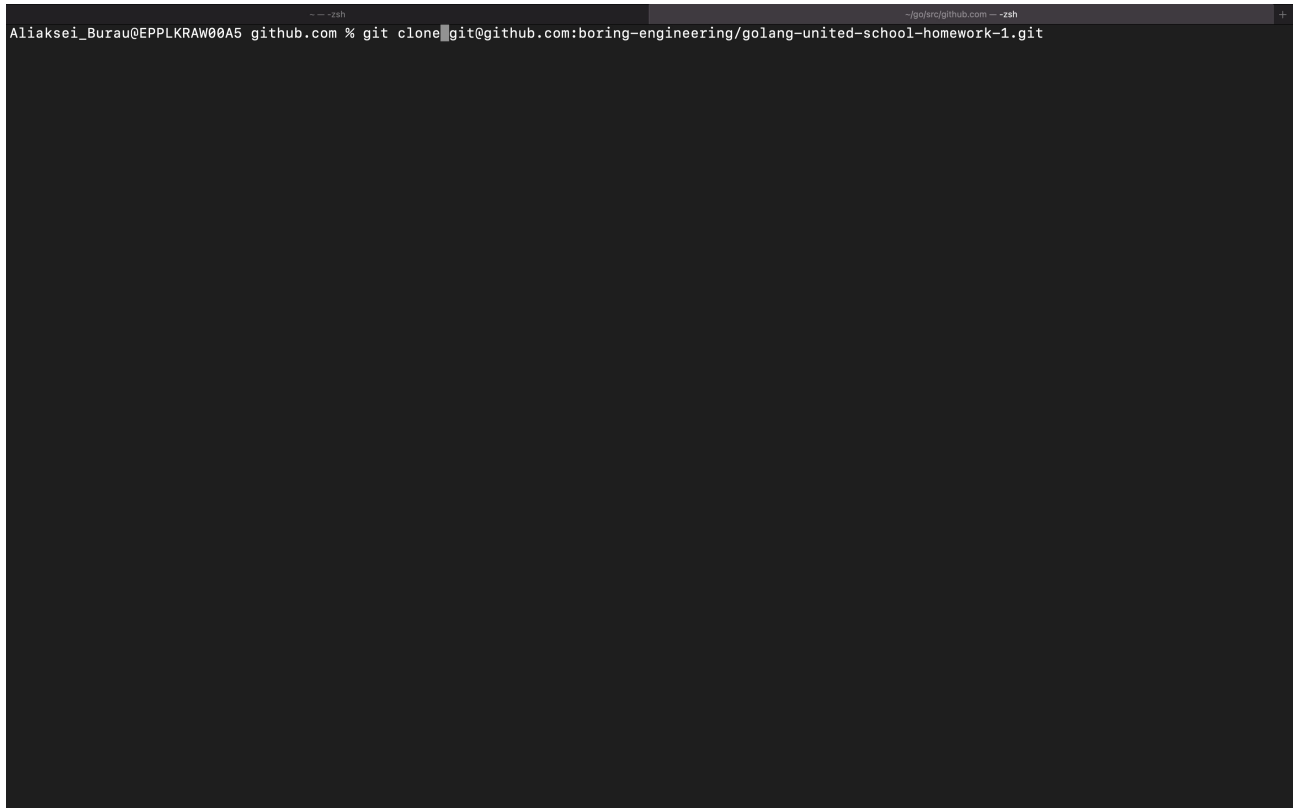
- After that you will see new repository on your ac



Golang United

Setup laptop

- Clone forked repo to your local machine
- You can use any text editor to write code

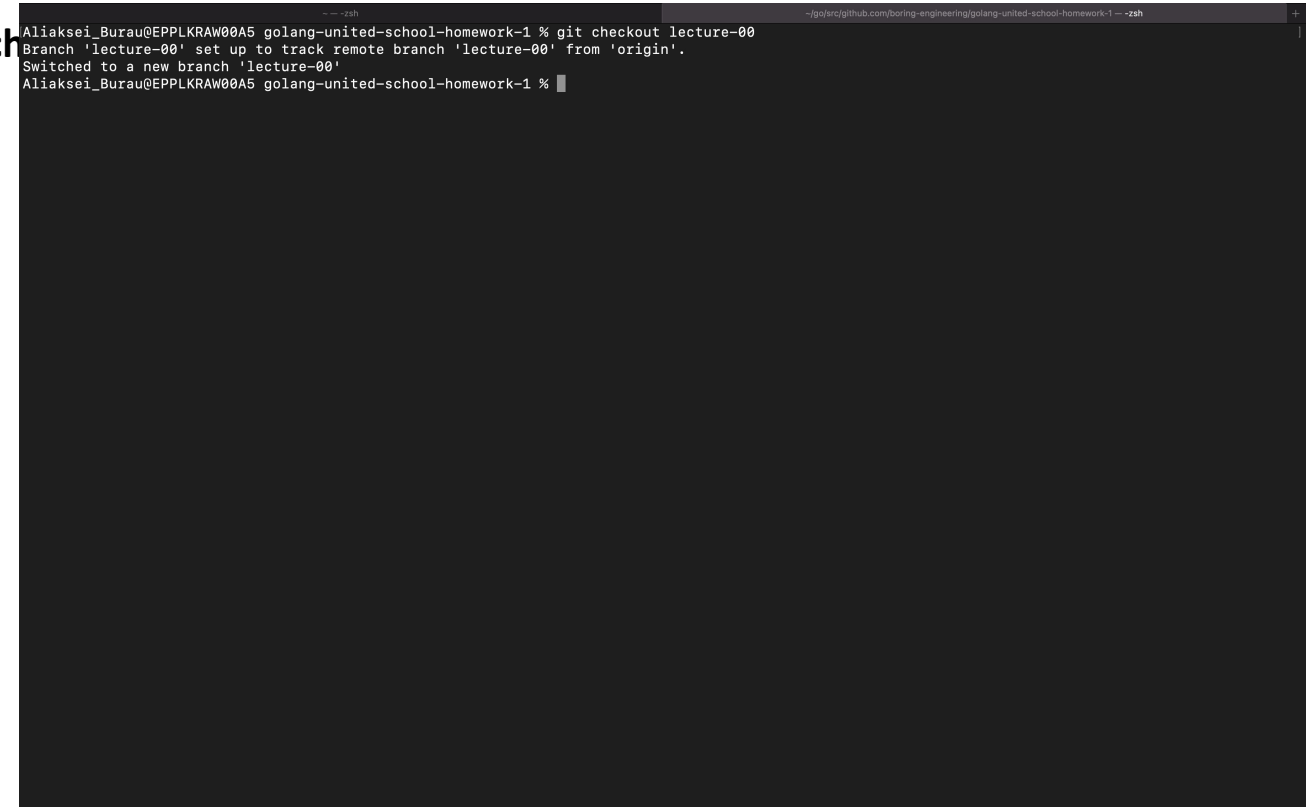


Golang United

Switch to branch with task

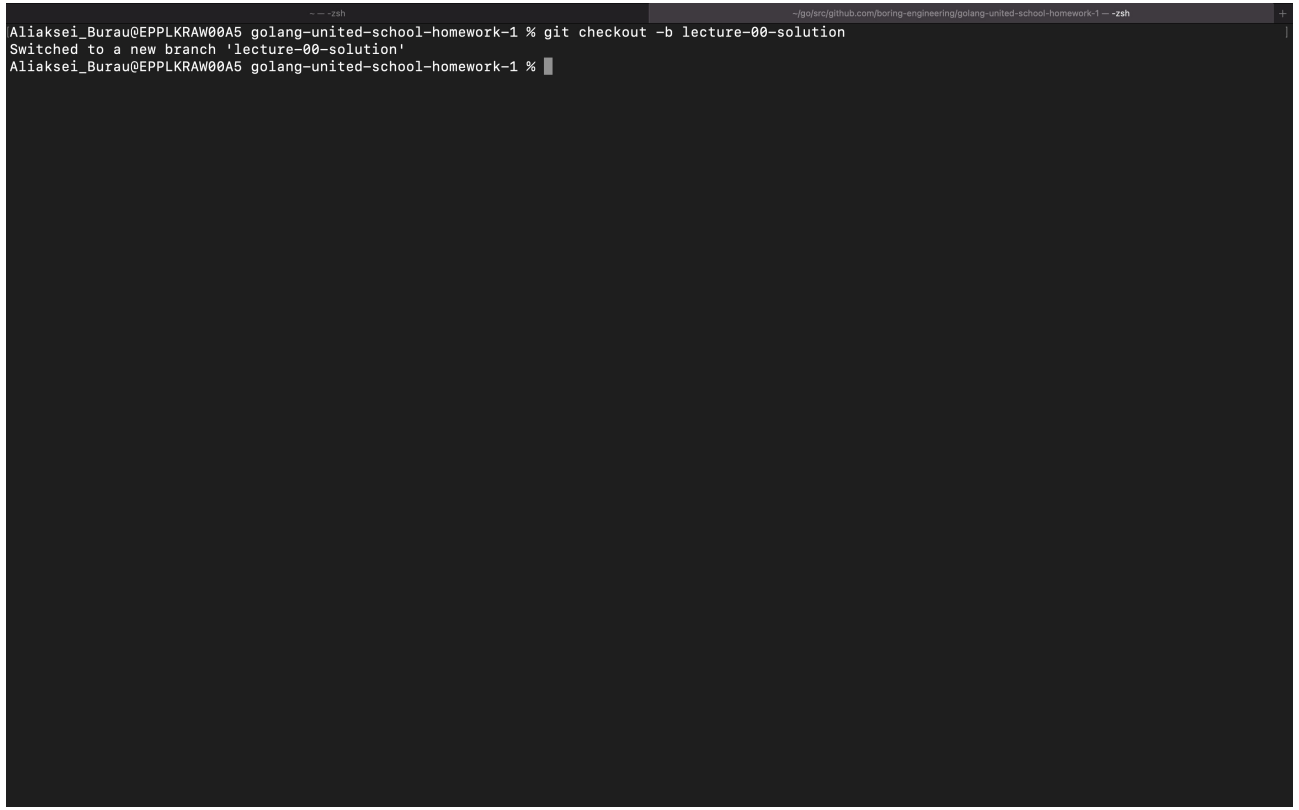
- Each lecture will be published to the new **branch**
- Find task for **”Introduction lecture”** in branch

“lecture-00”



Golang United

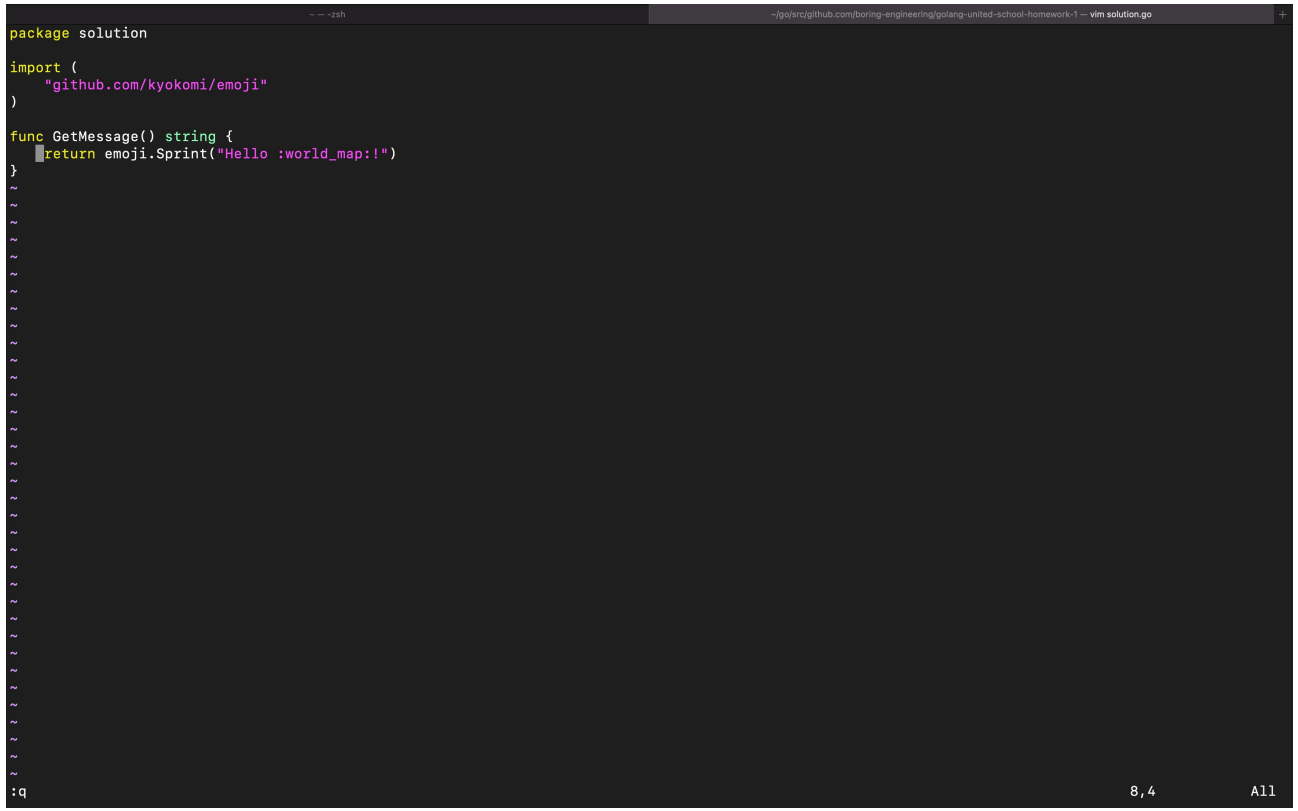
- To avoid conflicts please don’t push code into the same branch
- Create new one f.e. ”lecture-00-solution”



Golang United

Write code

- In **README.md** you can find instructions
- Complete task accounrding to instructions
- See screenshot for the hints

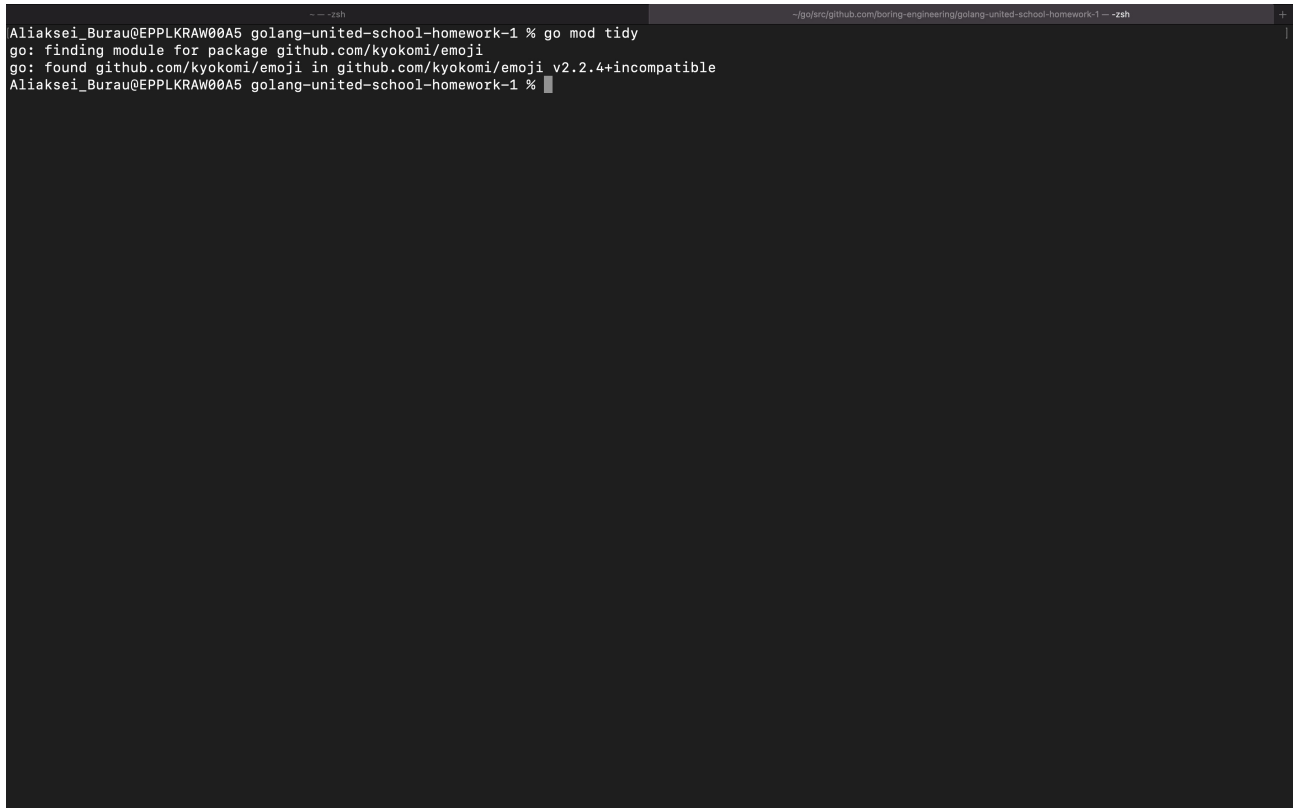


Golang United

Initialize repository

- Run **”go mod init”** to initialize **Go** project
- Run **“go mod tidy”** to fetch dependencies

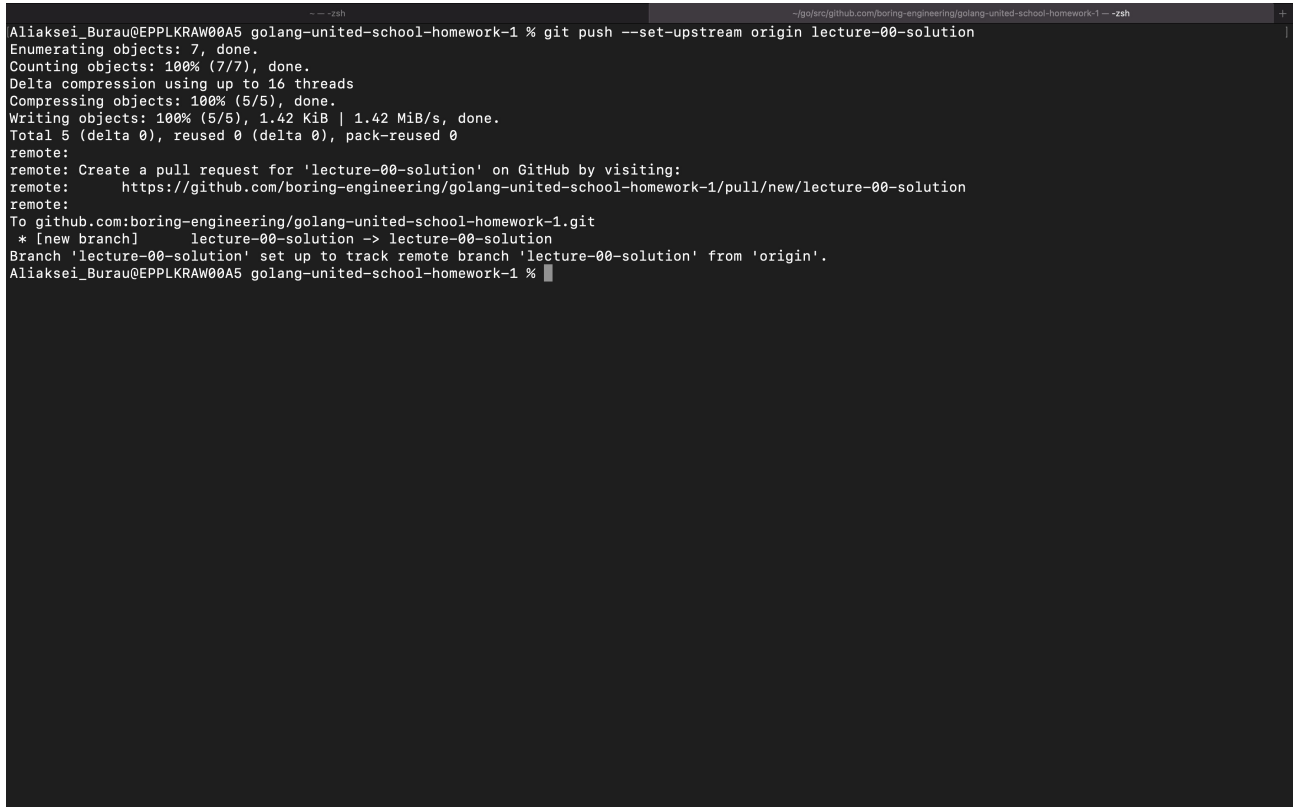
You should run this commands in repo



Golang United

Push the code

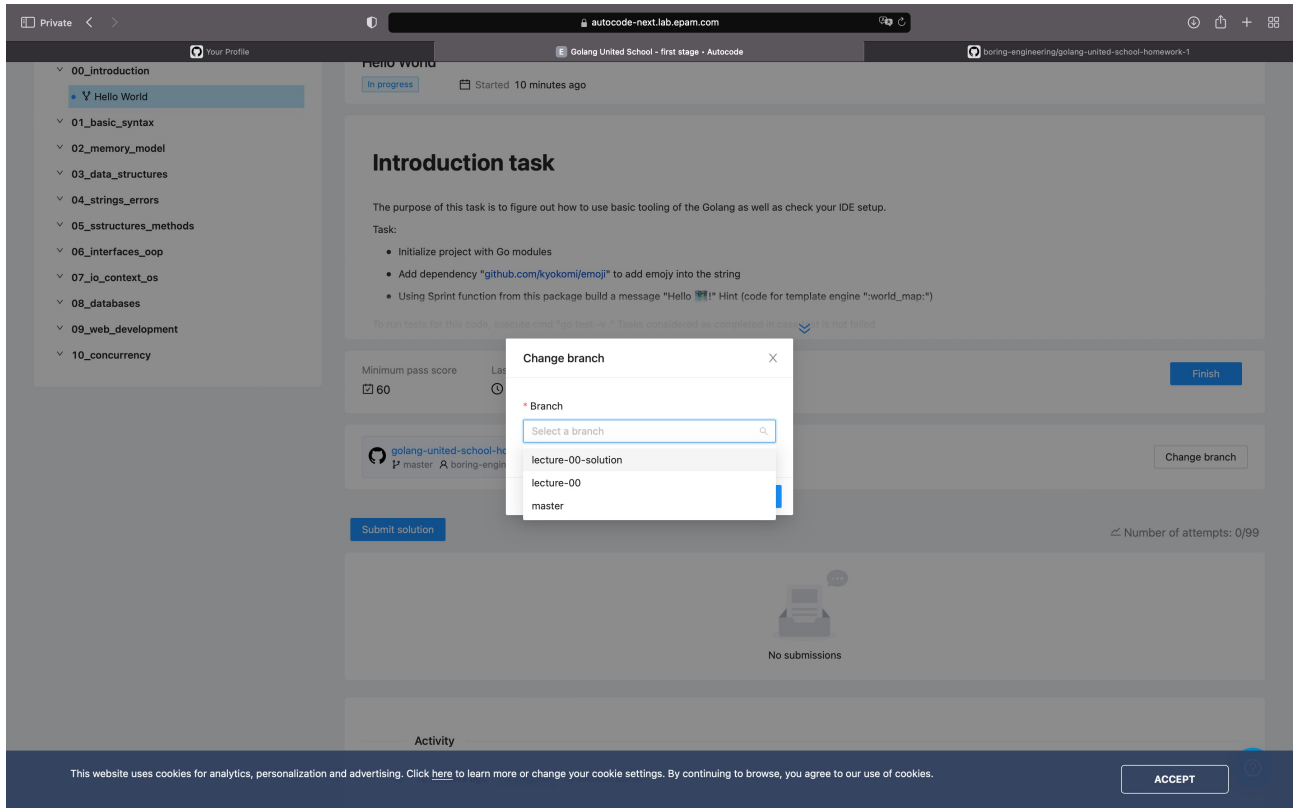
- Push new branch to the server



Golang United

Test submission

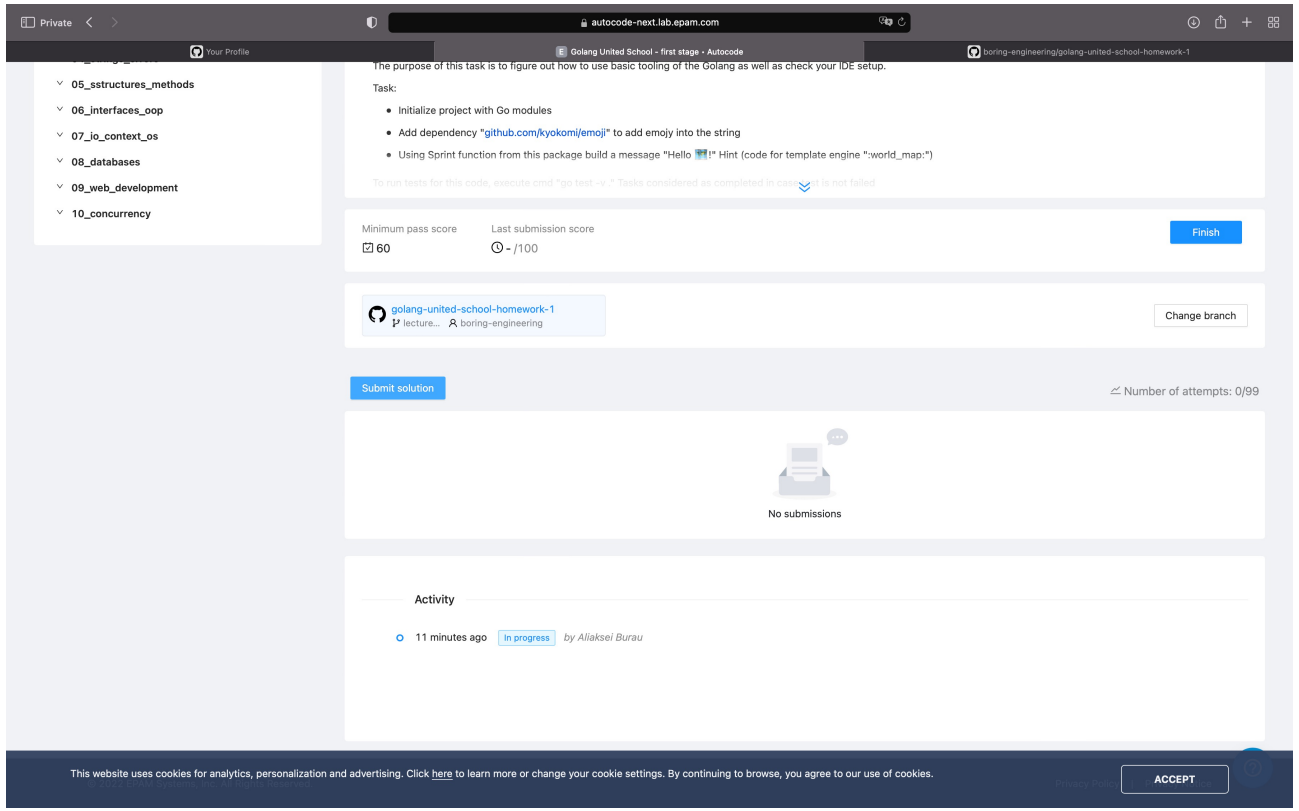
- Choose branch with code



Golang United

Test submission

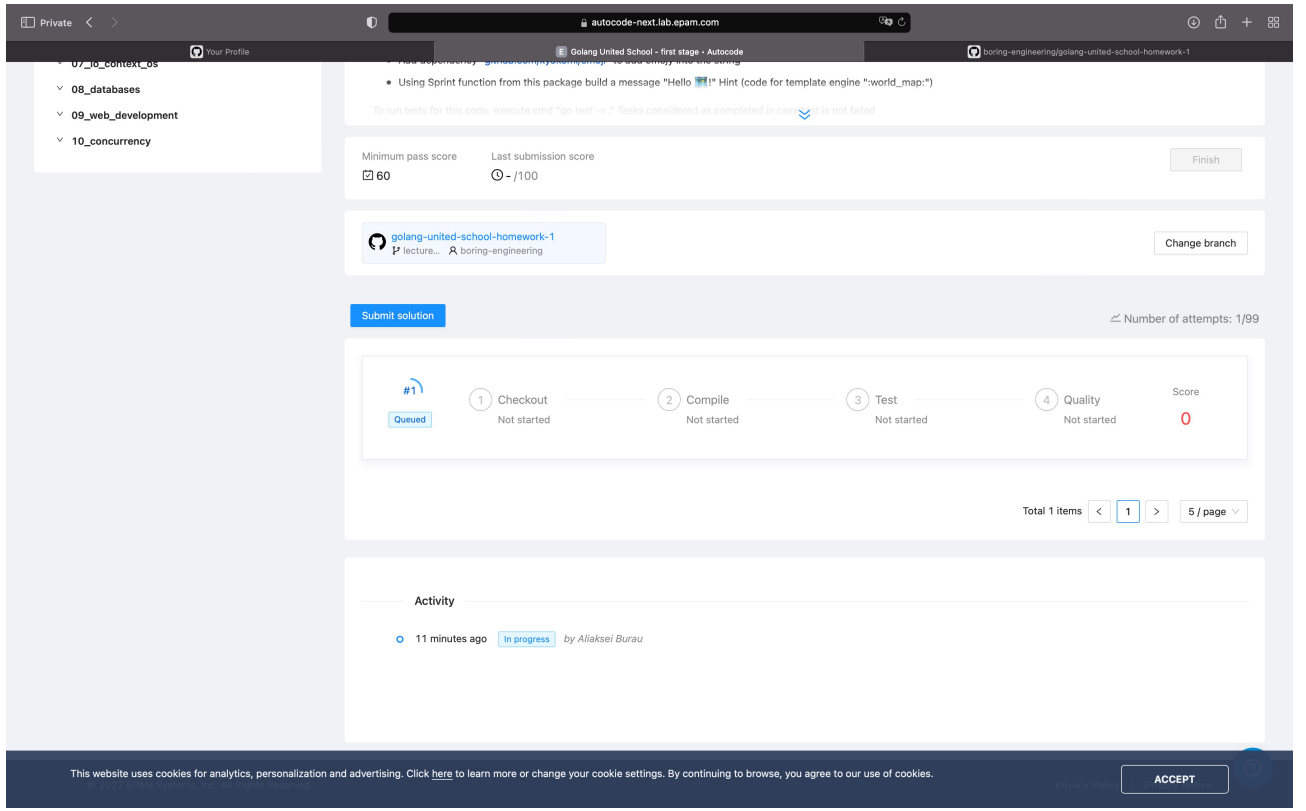
- Press button “submit solution”



Golang United

Wait for test result

- Press button “submit solution”



Golang United

Test submission

- In case you satisfied with mark, press “Finish”