Introduction

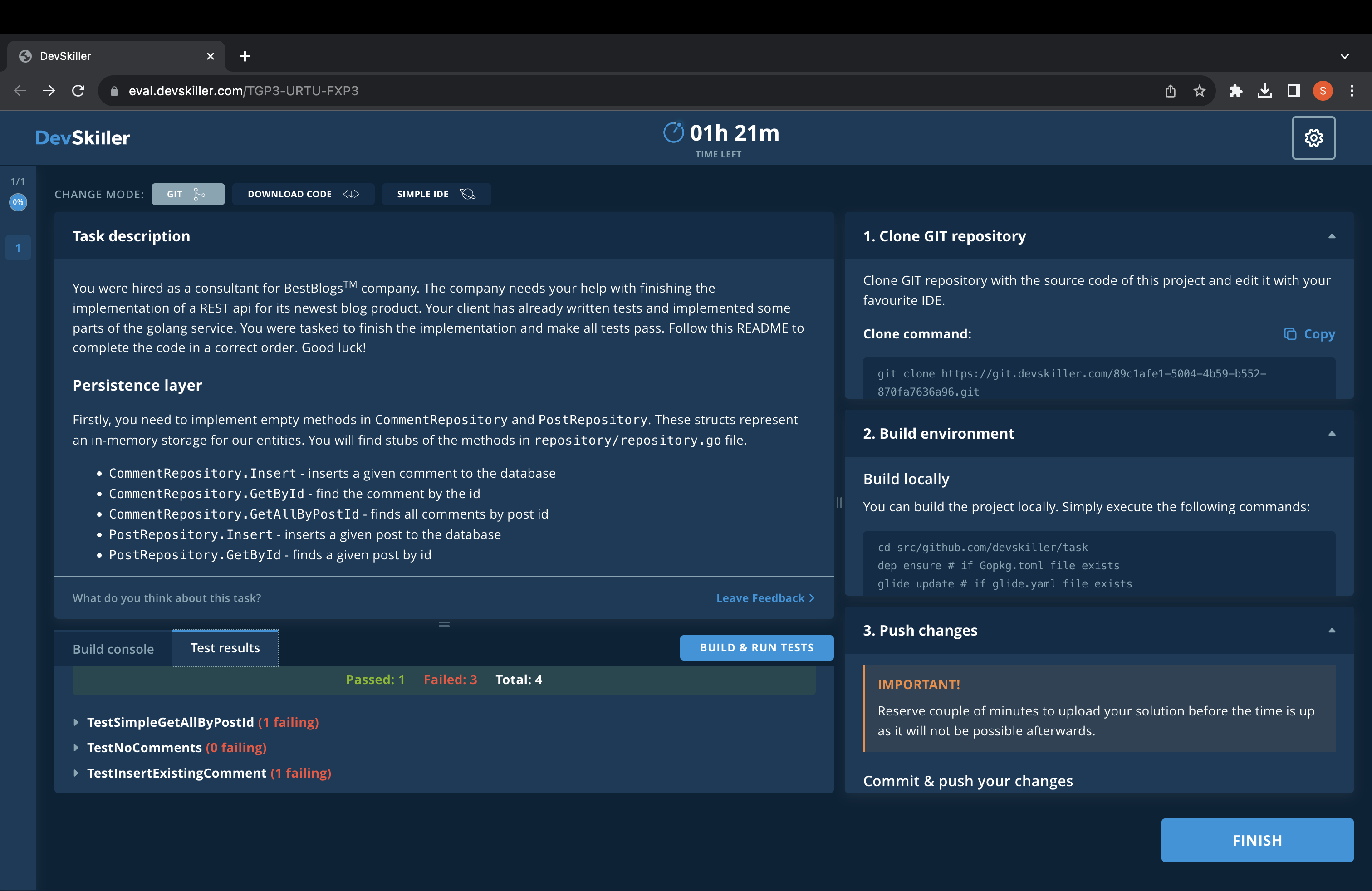
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated



## Introduction

You were hired as a consultant for BestBlogsTM company. The company needs your help with finishing the implementation of a REST api for its newest blog product. Your client has already written tests and implemented some parts of the golang service. You were tasked to finish the implementation and make all tests pass. Follow this README to complete the code in a correct order. Good luck!

### Persistence layer

Firstly, you need to implement empty methods in CommentRepository and PostRepository. These structs represent an in-memory storage for our entities. You will find stubs of the methods in repository/repository.go file.

* CommentRepository.Insert - inserts a given comment to the database
* CommentRepository.GetById - find the comment by the id
* CommentRepository.GetAllByPostId - finds all comments by post id
* PostRepository.Insert - inserts a given post to the database
* PostRepository.GetById - finds a given post by id

Detailed descriptions of desirable implementation may be found as comments in the methods' body.

### Web layer

Secondly, you need to implement a web layer for the blog API. This step involves implementing REST API handlers in service/rest.go. Each empty handler function you need to implement contains comments that form a specification of what needs to be done. As a reference, you can use the implementation of /api/post/post rest endpoint in service.rest.go The REST endpoints to implement are:

* /api/post/comment - deserializes JSON request payload as model.Comment and persists it into CommentRepository. Otherwise, appropriate error message and status code are returned.
* /api/get/post/[POST\_ID] - looks for a post with given id in the database and returns it. Otherwise, appropriate error message and status code are returned.
* /api/get/comments/[POST\_ID] - looks for all comments with given post id in the database and returns them. Otherwise, appropriate error message and status code are returned.

## Building and testing

#### Prerequisites:

1. make is installed on your system

#### Building

To build the binary run make command in the root directory of this repository.

#### Testing

To run all unit tests issue make test command in the root directory of this repository.

## Good luck!

Build console

gitlab.com/devskiller-tasks/rest-api-blog-golang [no test files]

? gitlab.com/devskiller-tasks/rest-api-blog-golang/bootstrap [no test files]

? gitlab.com/devskiller-tasks/rest-api-blog-golang/model [no test files]

=== RUN TestSimpleGetAllByPostId

--- FAIL: TestSimpleGetAllByPostId (0.00s)

repository\_test.go:20:

Error Trace: repository\_test.go:20

Error: elements differ

extra elements in list B:

([]interface {}) (len=1) {

(model.Comment) {

Id: (uint64) 1,

PostId: (uint64) 101,

Comment: (string) (len=8) "comment2",

Author: (string) (len=7) "author2",

CreationDate: (time.Time) {

wall: (uint64) 0,

ext: (int64) 62135606811,

loc: (\*time.Location)({

name: (string) "",

zone: ([]time.zone) <nil>,

tx: ([]time.zoneTrans) <nil>,

cacheStart: (int64) 0,

cacheEnd: (int64) 0,

cacheZone: (\*time.zone)(<nil>)

})

}

}

}

listA:

([]model.Comment) <nil>

listB:

([]model.Comment) (len=1) {

(model.Comment) {

Id: (uint64) 1,

PostId: (uint64) 101,

Comment: (string) (len=8) "comment2",

Author: (string) (len=7) "author2",

CreationDate: (time.Time) {

wall: (uint64) 0,

ext: (int64) 62135606811,

loc: (\*time.Location)({

name: (string) "",

zone: ([]time.zone) <nil>,

tx: ([]time.zoneTrans) <nil>,

cacheStart: (int64) 0,

cacheEnd: (int64) 0,

cacheZone: (\*time.zone)(<nil>)

})

}

}

}

Test: TestSimpleGetAllByPostId

=== RUN TestGetAllByPostId

--- FAIL: TestGetAllByPostId (0.00s)

repository\_test.go:31:

Error Trace: repository\_test.go:31

Error: elements differ

extra elements in list A:

([]interface {}) (len=2) {

(model.Comment) {

Id: (uint64) 1,

PostId: (uint64) 101,

Comment: (string) (len=8) "comment2",

Author: (string) (len=7) "author2",

CreationDate: (time.Time) {

wall: (uint64) 0,

ext: (int64) 62135606811,

loc: (\*time.Location)({

name: (string) "",

zone: ([]time.zone) <nil>,

tx: ([]time.zoneTrans) <nil>,

cacheStart: (int64) 0,

cacheEnd: (int64) 0,

cacheZone: (\*time.zone)(<nil>)

})

}

},

(model.Comment) {

Id: (uint64) 2,

PostId: (uint64) 101,

Comment: (string) (len=8) "comment2",

Author: (string) (len=7) "author2",

CreationDate: (time.Time) {

wall: (uint64) 0,

ext: (int64) 62135606811,

loc: (\*time.Location)({

name: (string) "",

zone: ([]time.zone) <nil>,

tx: ([]time.zoneTrans) <nil>,

cacheStart: (int64) 0,

cacheEnd: (int64) 0,

cacheZone: (\*time.zone)(<nil>)

})

}

}

}

listA:

([]model.Comment) (len=2) {

(model.Comment) {

Id: (uint64) 1,

PostId: (uint64) 101,

Comment: (string) (len=8) "comment2",

Author: (string) (len=7) "author2",

CreationDate: (time.Time) {

wall: (uint64) 0,

ext: (int64) 62135606811,

loc: (\*time.Location)({

name: (string) "",

zone: ([]time.zone) <nil>,

tx: ([]time.zoneTrans) <nil>,

cacheStart: (int64) 0,

cacheEnd: (int64) 0,

cacheZone: (\*time.zone)(<nil>)

})

}

},

(model.Comment) {

Id: (uint64) 2,

PostId: (uint64) 101,

Comment: (string) (len=8) "comment2",

Author: (string) (len=7) "author2",

CreationDate: (time.Time) {

wall: (uint64) 0,

ext: (int64) 62135606811,

loc: (\*time.Location)({

name: (string) "",

zone: ([]time.zone) <nil>,

tx: ([]time.zoneTrans) <nil>,

cacheStart: (int64) 0,

cacheEnd: (int64) 0,

cacheZone: (\*time.zone)(<nil>)

})

}

}

}

listB:

([]model.Comment) <nil>

Test: TestGetAllByPostId

=== RUN TestNoComments

--- PASS: TestNoComments (0.00s)

=== RUN TestInsertExistingComment

--- FAIL: TestInsertExistingComment (0.00s)

repository\_test.go:49:

Error Trace: repository\_test.go:49

Error: An error is expected but got nil.

Test: TestInsertExistingComment

Messages: test failed because of wrong error msg: <nil>

FAIL

FAIL gitlab.com/devskiller-tasks/rest-api-blog-golang/repository 0.005s

FAIL gitlab.com/devskiller-tasks/rest-api-blog-golang/service [build failed]

FAIL

-- Error output:

go: downloading github.com/stretchr/testify v1.7.0

go: extracting github.com/stretchr/testify v1.7.0

go: downloading github.com/davecgh/go-spew v1.1.0

go: downloading github.com/pmezard/go-difflib v1.0.0

go: downloading gopkg.in/yaml.v3 v3.0.0-20200313102051-9f266ea9e77c

go: extracting github.com/pmezard/go-difflib v1.0.0

go: extracting gopkg.in/yaml.v3 v3.0.0-20200313102051-9f266ea9e77c

go: extracting github.com/davecgh/go-spew v1.1.0

go: finding github.com/stretchr/testify v1.7.0

go: finding github.com/davecgh/go-spew v1.1.0

go: finding gopkg.in/yaml.v3 v3.0.0-20200313102051-9f266ea9e77c

go: finding github.com/pmezard/go-difflib v1.0.0

# gitlab.com/devskiller-tasks/rest-api-blog-golang/service [gitlab.com/devskiller-tasks/rest-api-blog-golang/service.test]

service/rest\_test.go:53:15: undefined: io.ReadAll

service/rest\_test.go:107:15: undefined: io.ReadAll

service/rest\_test.go:158:15: undefined: io.ReadAll

service/rest\_test.go:209:15: undefined: io.ReadAll

note: module requires Go 1.16

Test result

A screenshot of a computer

Description automatically generated

1. **Download code**

Download A blue and white box with white text

Description automatically generatedproject source code as a ZIP archive and edit it locally using your favourite IDE.

DOWNLOAD CODE

**IMPORTANT!**Publishing the task on public GIT hosting services such as GitHub, GitLab, or Bitbucket is prohibited!

**Read-only files**

The task author might have marked some of the project files as read-only. Any commits that contain changes to files matching the following patterns will be rejected!

\*\*/\*\_test.go

1. **Build environment**

A screenshot of a computer program

Description automatically generated

**Build locally**

You can build the project locally. Simply execute the following commands:

cd src/github.com/devskiller/task

dep ensure # if Gopkg.toml file exists

glide update # if glide.yaml file exists

go test $(go list ./... | grep -v /vendor/) -v

**Build console**

It is also possible to use our built-in console. Upload the source code and click **RUN TESTS** or **BUILD & RUN TESTS** button near the build console.

**IMPORTANT!**You can upload and build your solution as many times as you need. Just keep in mind that only the last uploaded version will be evaluated.

**3. Upload your solution**

**IMPORTANT!**Reserve couple of minutes to upload your solution before the time is up as it will not be possible afterwards.

Upload **ZIP** archive with your modified files. Each new upload will overwrite previous versions.

The uploaded ZIP archive must contain a devskiller.marker file in the same location as in downloaded ZIP. **Do not modify or delete this file from downloaded archive.**

File size must be **30 MB or less**. Remember to remove all build artifacts, external dependencies and libraries - leave only the source code.

**DRAG & DROP .ZIP FILE HEREORBROWSE FILES**

A screenshot of a computer

Description automatically generated

**1. Clone GIT repository**

Clone GIT repository with the source code of this project and edit it with your favourite IDE.

# Clone command:

Copygit clone https://git.devskiller.com/89c1afe1-5004-4b59-b552-870fa7636a96.git

**IMPORTANT!**Publishing the task on public GIT hosting services such as GitHub, GitLab, or Bitbucket is prohibited!

## Read-only files

The task author might have marked some of the project files as read-only. Any commits that contain changes to files matching the following patterns will be rejected!

\*\*/\*\_test.go

A blue screen with white text

Description automatically generated

**2. Build environment**

**Build locally**

You can build the project locally. Simply execute the following commands:

cd src/github.com/devskiller/task

dep ensure # if Gopkg.toml file exists

glide update # if glide.yaml file exists

go test $(go list ./... | grep -v /vendor/) -v

**Build console**

It is also possible to use our built-in console. Upload the source code and click **RUN TESTS** or **BUILD & RUN TESTS** button near the build console.

**IMPORTANT!**You can upload and build your solution as many times as you need. Just keep in mind that only the last uploaded version will be evaluated.

A screenshot of a computer program

Description automatically generated

**3. Push changes**

**IMPORTANT!**Reserve couple of minutes to upload your solution before the time is up as it will not be possible afterwards.

**Commit & push your changes**

git add .

git commit -m "Solution"

git push origin master

A screenshot of a computer screen

Description automatically generated