Go School(Ngee Ann Polytechnic)

Project
Go-Live
API-T-DROID
(API-TESTING TOOL)

Chan Jiunn Hwa

26th April 2021

Version 0.90(Alpha)

Table of Contents

1 Executive Summary

This project(API-T-DROID) is a web-based API microservice providing API testing services.

API-T-DROID(suggested pronounciation is Epic-Driod, here after using the homonym EPIC-DROID), or in short EPIC.

EPIC mission is to be ‘An Adroit Users’ Epics API Testing Tool’. This means, as the rhyming suggests, it is an

- Automated API Test Tool
- That runs Test Cases to Epics
- Adroitly

Generally, with the growth of microservices, rise of DevOps and Agile methodology, it means that testing departments needs to keep in pace, and cannot remain as that ‘other deparment’ but needs to integrate and automate.

With test automation, the organization benefits the quality process through speed, wider and deeper test coverage, consistency, significant cost savings and productivity gains, all resulting in faster time to market with a better delivery and product, as personally attested while developing this tool.

Having a good testing tool, allows developers, QAs and other stakeholders to scale productivity and quality; it enables Capability Maturation - that is “Towards A Continuous Testing Model Organization.”

2 Overview

This documents provides a record of the various phases and activities, the rationale, considerations and artifacts produced. It provides a closure of the ‘current state’, a capture while it is still live.

API-T-DROID culminates upon the earlier Go lessons and exercises right from the the console ShopList app of Assignment 1. Standing upon the artifacts and reuse functions accumulated since, this project aims to mature the codebase with each iteration.

Since the 1st assignment till date, this project has always exercised a Process and Plan approach to building out the Product. So far, this disciplined ‘overall waterfall’ type of process has yielded consistent, on time, and above minimum-stated-requirements deliverable. Also, Process and Product feature density has gained progressively even with the same constant mandays of <10 days of these assignments.

The project believes in incorporating a SDLC framework in development, because each time we build an App, we are also concomitantly building and exercising out the Developmental Process.

2.1 Objective

Having experienced common issues and difficulties of Manual Testing, the objective of EPIC is to deliver a performant, and cost effective automated testing application.

2.2 Scope

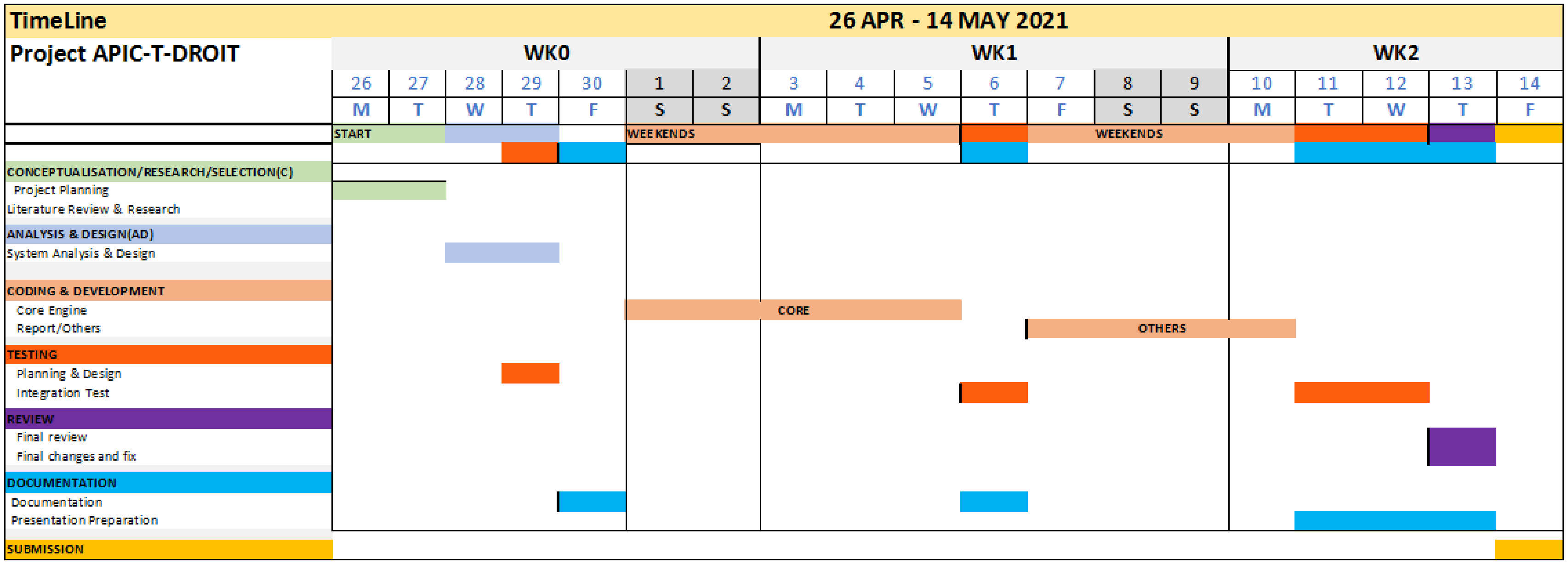
Due to time constraints, and given the understanding of the fact that the developer does not come from a Test/QA background, the challenge and scope is to develop both a useful and usable application that is baseline complete.

3 Project Plan

A Project Plan helps ensure that work is expended with a disciplined approach and effort is planned and budgeted for each lifecycle activity.

3.1 Schedule

The proposed schedule:



3.2 Table of Activities

CONCEPTUALISATION/RESEARCH/SELECTION(C)	
Project Planning	**A. Creation of Project Plan** Mapping and time-boxing the activities, as well as listing out the desired features are the first activities. The intention is to ensure proportionate and balanced effort. From experience, consistent output comes from proportionate budgeting, and coding usually does not occupy more than 50% of the lifecycle for a balanced plan.
Literature Review & Research	**B. Background Research** Just as in thesis writing or patent filing, it is always required to have a literature review or prior art.
ANALYSIS & DESIGN(AD)	
Functions	**C. Functional Analysis** Based on the background research, features are listed, though it is estimated not all can be implemented. The outputs are database design, application and security architecture, and the logical feature list. **D. Feasibility Analysis** Scoping and feasibility has to be estimated.
CODING	
Sprint	**E. Code Methodology** The project uses a 'sprint' type of methodology for the coding sub-phase, but overall planning is still a 'waterfall' design. Each major feature will have a budgeted sprint cycle of up to 1 manday, while minor features will take 0.25 manday.

	F. Risk Management To manage timeline risk, each feature(f1,…) is allowed a sprint cycle of about 1 day. Any deepening of features can be revisited in later sweeps, or put up as future features. Bottlenecks are required to be resolved within the day, as buffered up issues takes an exponential risk to panicking.
TESTING	
Planning & Design Integration Test	**G. Test Case Design and Execution** Unit, Regression and Integration are included. As much as possible, though not explicitly required, Security Test Cases/Pen Test will be injected too.
REVIEW	
Final review Final changes and fix	**H. Review & Fix** The last 3 days will be code freeze, and will only allow reviews and bug fixes.
DOCUMENTATION	
Planning Document Structure Setup Record Final Review	**I. Documentation** Documentation is continuous, but the main cycle starts some time after coding, after coding has peaked. Final rounds will be done alongside the Review and Fix cycle, collated before submission.

SUBMISSION	**J. Zip and Upload** By 2359 14th May 2021(FRI), consolidate and put into submission folder: ● Writeup ■ Add this document in doc format. ■ Add pdf version as backup, in case the word doc format runs. ● Presentation slides ● Video of demo ■ test screen shots ■ videos ● Source code and all necessary files to setup and run the prototype ■ source files, executable, test data, scripts and any other artifacts. ■ Executable and sql scripts of Test Target Server(gocrudapi) ● Zipped and upload<YourName_ProjectName.zip>

3.3 Deliverables

In this project span of about 3 weeks, the following deliverables have been produced:

- Application - a Web Application for testing API.
- Documentation - User Guide as a self-contained html page.

3.4 Product Demonstration

It was stated In the project proposal, the tentative idea was to demonstrate bugs discovery of some public API(subject to permissions) such as:

- Shopee
- Other public API(to be researched)

However, due to the fact that the surface area of such APIs are large and time has to be taken to research, scope, ask and obtain and APIKEY of theirs, and primarily as well that a quick sampling of some outside APIs(EventBrite, Shopee, etc) shows variances to the textbook way of structuring API endpoints, thus rendering this activity non-conclusive, and risk of side effects.

Therefore, as a baseline, EPIC will demonstrate against the CRUD Course Catalog microservice of previous assignment.

4 System Analysis

Functional analysis helps to provide an overall coherent release, and also allows roadmap planning of future features of strategic value, as they will be rooted in the same feature tree.

The list provides the desirable logical feature set, but not all the features listed here may be able to be released on current submission.

4.1 Functional Features - Non-Testing Services

In order to provide a scaffold to hold the test functions, a minimal set of UI and related services are provided.

4.1.1 Web UI

The system shall provide a Web interface to manage logins, and signups.

4.1.2 Logging

The system shall provide mechanism to do system logging. Items of interest includes request path, responses, job duration.

4.2 Functional Features - Test Execution Services

Due to the time constraint, and that the focus is on the ‘Core Engine’ - the Test Execution Services, the project shall provide Create, Read and Delete functionality, as Update may mean multiple joins due to the nesting(although a Delete Old and Add New is equivalent to Update).

As the cases, epics, and jobs defintion are still a work-in-progress, and thus evolving, effort will be better spend on the ‘Core Engine’, rather than ‘Auxillary Services’, as CRD is sufficient to exercise all ‘Core Engine’ services. Also, the final datastore could be also NoSQL database.

4.2.1 Test EndPoint Management

The system shall provide an interface for user to manage endpoints.

4.2.2 Test Case Management

The system shall provide an interface for user to manage test cases.

4.2.3 Test Epic Management

The system shall provide an interface for user to manage test epics.

4.2.4 Test Job Management

The system shall provide an interface for user to manage test jobs.

4.2.5 Test Execution

The system shall provide an interface for user to select the Test Job for running. The test execution shall execute the actions defined, and check the response for status, response times, and other defined pass critieria.

4.2.6 Test Reporting

The system shall provide an interface for the user to see the test response and result for the test runned.

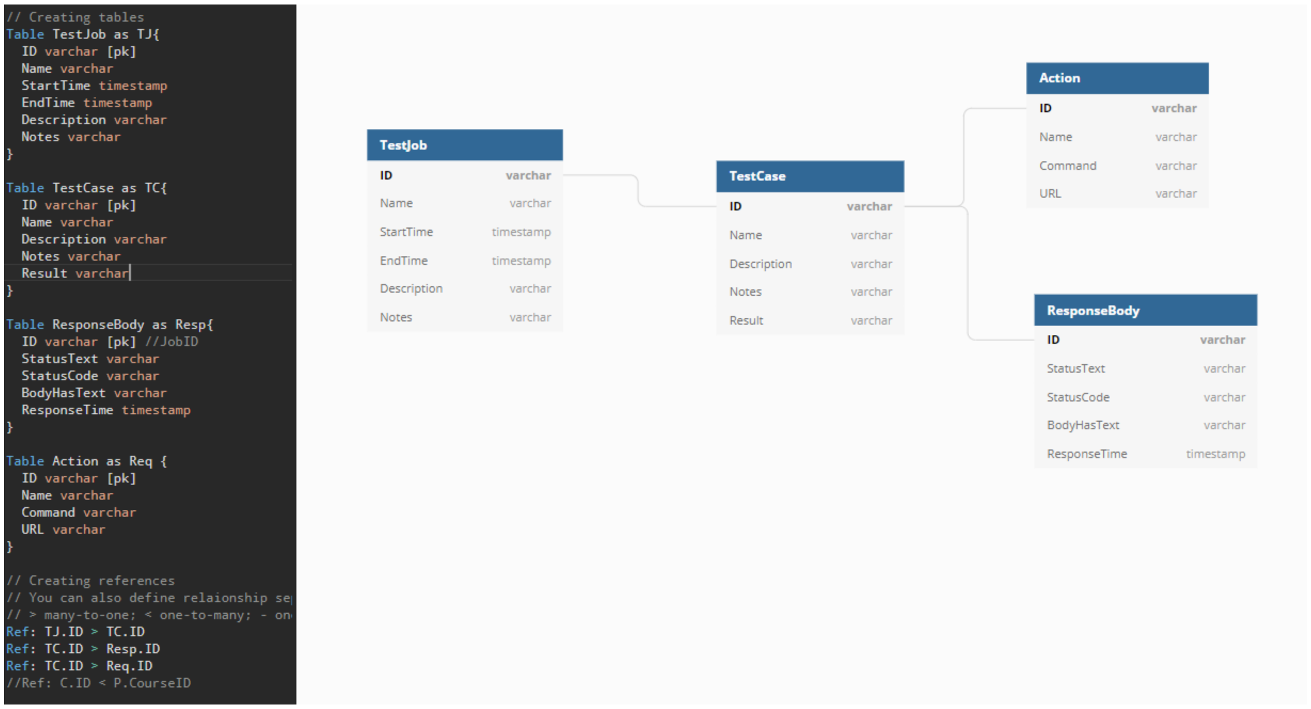
4.2.7 Tests Addition

The system shall provide a scaffold to allow additions of other test functionalities. Such test functionalities can include such as:

- Load Test, Security Test(Injections), Expired Links, Fuzzing
- Any Other

4.3 System Architecture Design

4.3.1 Database Schema

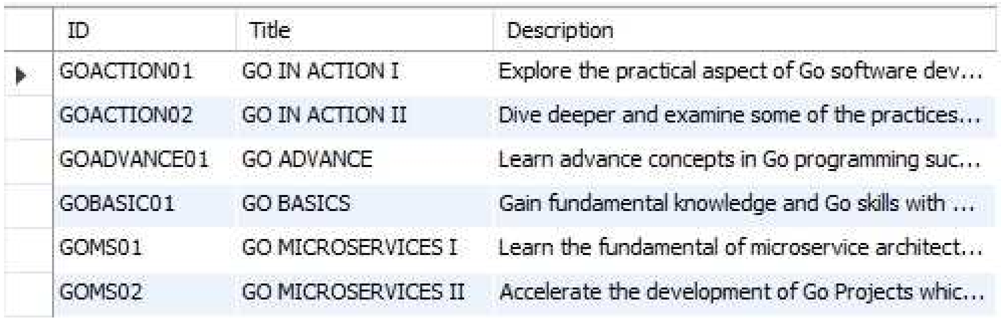


```
// Creating tables
Table TestJob as TJ{
  ID varchar [pk]
  Name varchar
  StartTime timestamp
  EndTime timestamp
  Description varchar
  Notes varchar
}

Table TestCase as TC{
  ID varchar [pk]
  Name varchar
  Description varchar
  Notes varchar
  Result varchar
}

Table ResponseBody as Resp{
  ID varchar [pk] //JobID
  StatusText varchar
  StatusCode varchar
  BodyHasText varchar
  ResponseTime timestamp
}

Table Action as Req {
  ID varchar [pk]
  Name varchar
  Command varchar
  URL varchar
}

// Creating references
// You can also define relaionship se
// > many-to-one; < one-to-many; - on
Ref: TJ.ID > TC.ID
Ref: TC.ID > Resp.ID
Ref: TC.ID > Req.ID
//Ref: C.ID < P.CourseID
```

4.3.1.1 Mock Data

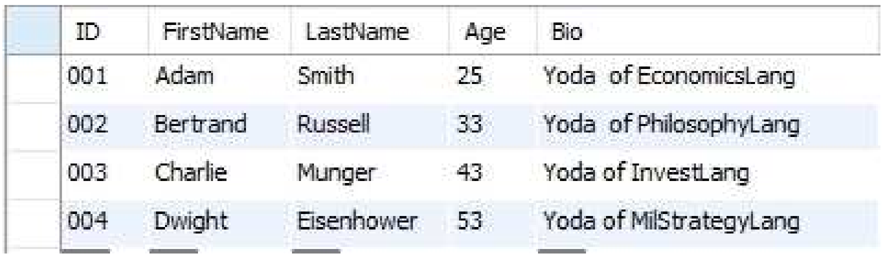
The mock data used by the test target server(gocrudapi) are shown below.

Courses:



ID	Title	Description
GOACTION01	GO IN ACTION I	Explore the practical aspect of Go software dev…
GOACTION02	GO IN ACTION II	Dive deeper and examine some of the practices…
GOADVANCE01	GO ADVANCE	Learn advance concepts in Go programming suc…
GOBASIC01	GO BASICS	Gain fundamental knowledge and Go skills with …
GOMS01	GO MICROSERVICES I	Learn the fundamental of microservice architect…
GOMS02	GO MICROSERVICES II	Accelerate the development of Go Projects whic…

Trainers:

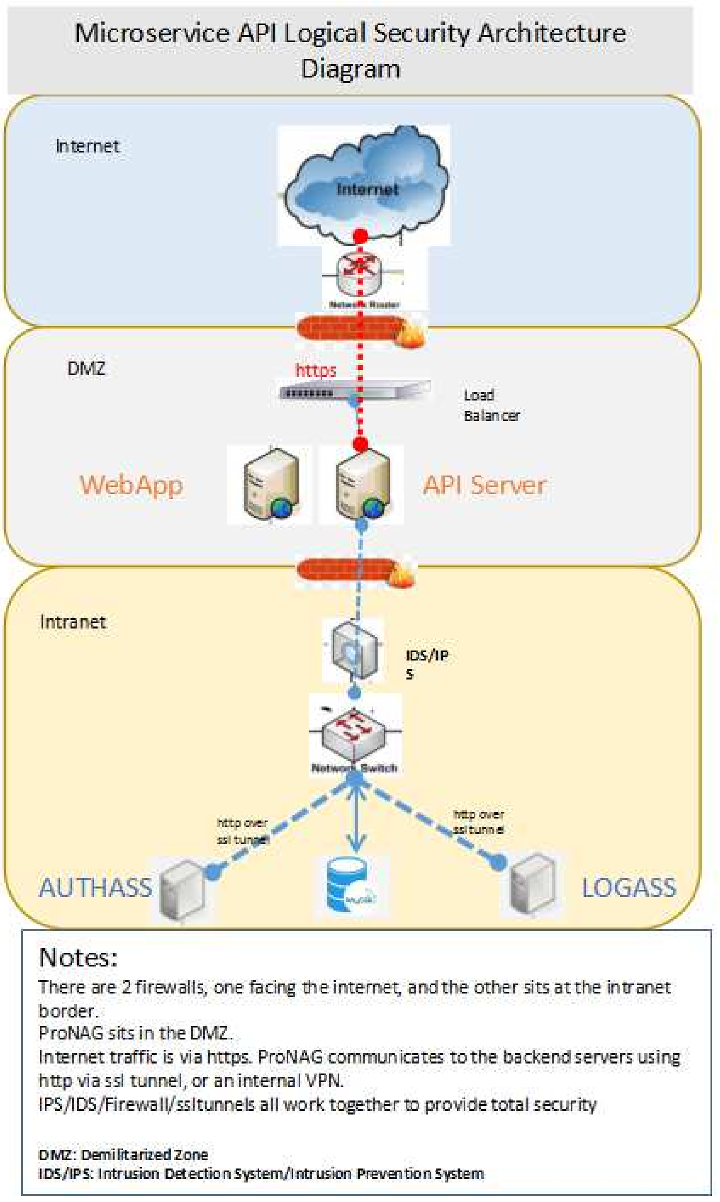


ID	FirstName	LastName	Age	Bio
001	Adam	Smith	25	Yoda of EconomicsLang
002	Bertrand	Russell	33	Yoda of PhilosophyLang
003	Charlie	Munger	43	Yoda of InvestLang
004	Dwight	Eisenhower	53	Yoda of MilStrategyLang

4.3.2 Security Architecture Diagram

The current version is a technology preview version, more focused on the core engine. For further development towards an ‘entreprise version’ with teams, projects, and organization setup, the below would rerpresent the deployment architecture.

To begin with, all Internet facing traffic shall use https. However, Intranet, and internal traffic can use http but communicated through a ssl tunnel. Besides using a ssl tunnel, it is possible to have internal VPN. Many switches nowadays can layer multiple secure private networks(VPN) within an intranet, or over an internet. Additionally, deployed servers need to be os patched and hardened, unused ports lockdown, and enabled only with minimal relevant priviledges. Physically, the actual archtecture implementation can differ as devices nowadays have combined functionality. For example, some load balancers provide IDS/IPS features, and vice versa.



Notes:
There are 2 firewalls, one facing the internet, and the other sits at the intranet border.
ProNAG sits in the DMZ.
Internet traffic is via https. ProNAG communicates to the backend servers using http via ssl tunnel, or an internal VPN.
IPS/IDS/Firewall/ssltunnels all work together to provide total security

DMZ: Demilitarized Zone
IDS/IPS: Intrusion Detection System/Intrusion Prevention System

WebApp(not in current scope)
Front end to handle API signups, and other account services.

API Server
The course catalog service that is current scope.

LOGASS(Logging-As-A-Service)
a centralized backend service to aggregate distributed logging.

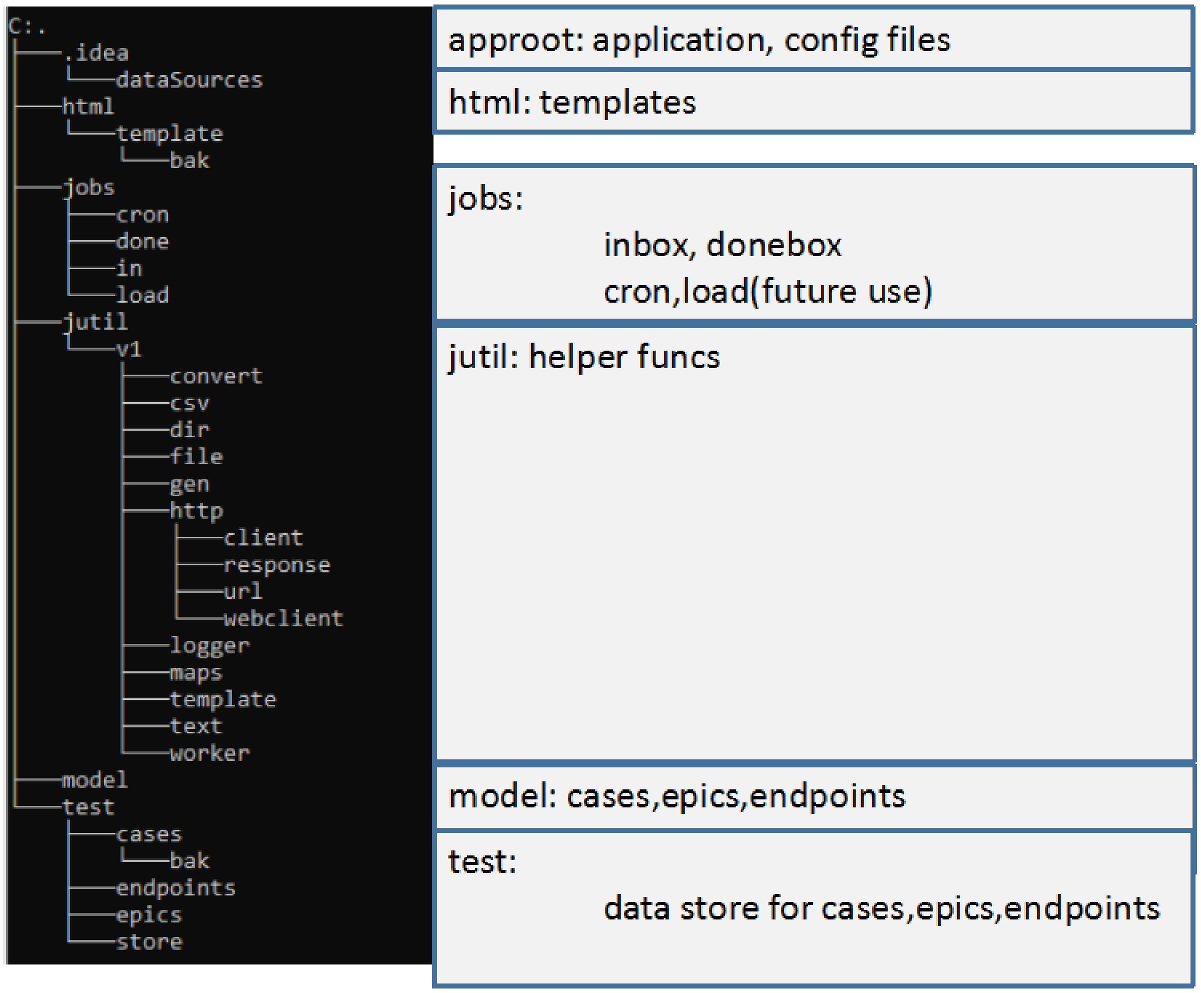
Auth(Auth-As-A-Service)
a centralized backend providing Auth services.

5 System Design & Development

This section describes the actual development of the features, walking through the design and the key code sections.

5.1 Project Schema

The following maps out the structure of the application files, and the work flow directories.



```
C:.
├───.idea
│   └───dataSources
├───html
│   └───template
│       └───bak
├───jobs
│   ├───cron
│   ├───done
│   ├───in
│   └───load
├───jutil
│   └───v1
│       ├───convert
│       ├───csv
│       ├───dir
│       ├───file
│       ├───gen
│       ├───http
│       │   ├───client
│       │   ├───response
│       │   ├───url
│       │   └───webclient
│       ├───logger
│       ├───maps
│       ├───template
│       ├───text
│       └───worker
├───model
└───test
    ├───cases
    │   └───bak
    ├───endpoints
    ├───epics
    └───store
```

approot: application, config files

html: templates

jobs:
inbox, donebox
cron,load(future use)

jutil: helper funcs

model: cases,epics,endpoints

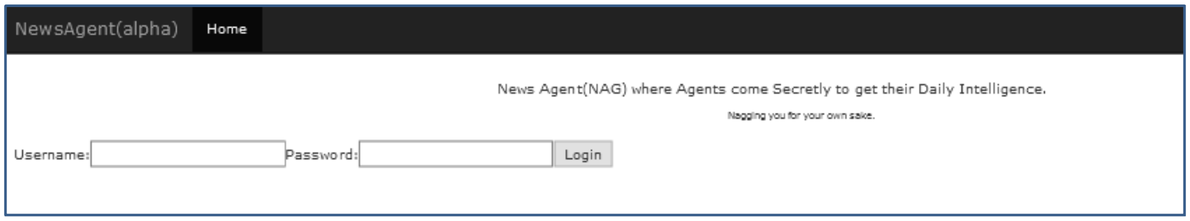
test:
data store for cases,epics,endpoints

5.2 Functional Features - Non-Testing Services

In order to provide a scaffold to hold the test functions, a minimal set of UI and related services are provided.

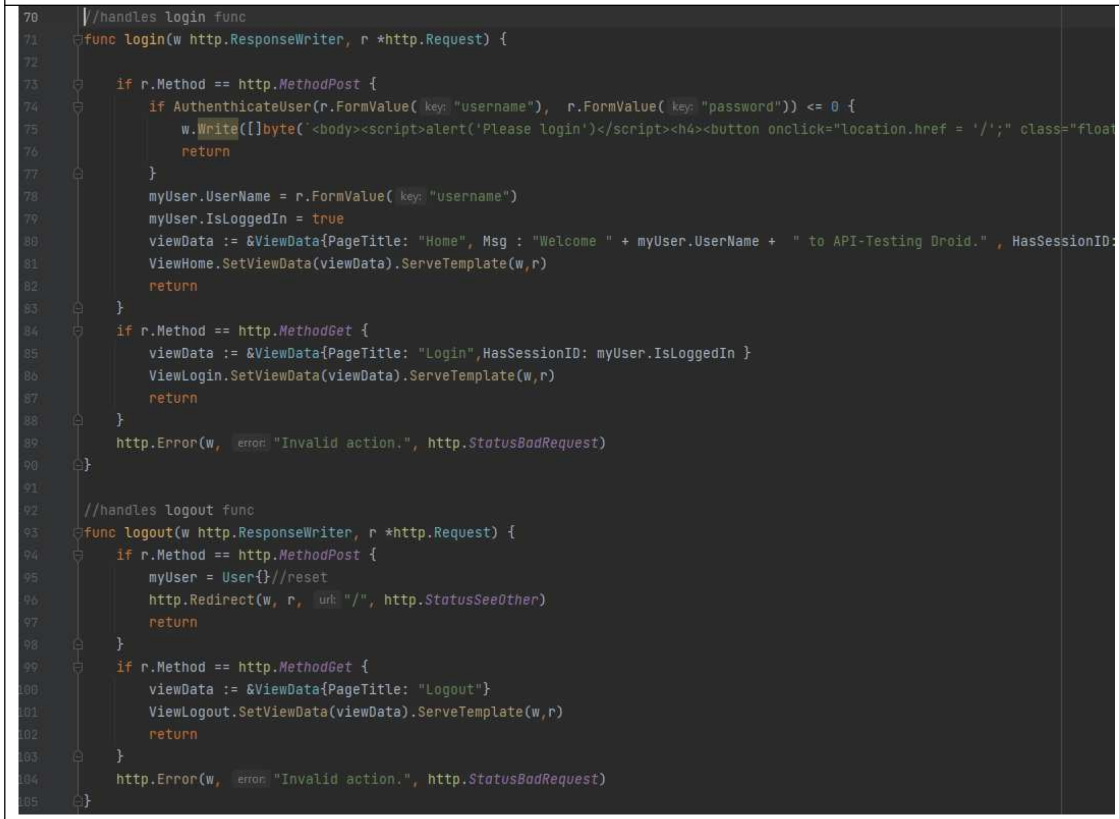
5.2.1 Web UI

The system shall provide a Web interface to manage logins..



News Agent(NAG) where Agents come Secretly to get their Daily Intelligence.

Nagging you for your own sake.



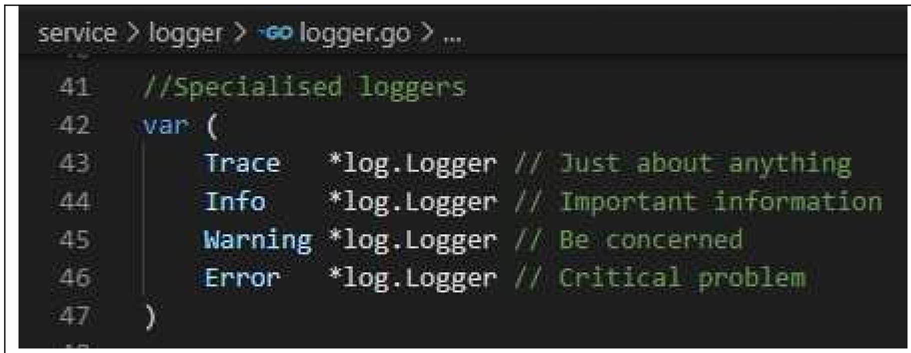
```go
//handles login func
func login(w http.ResponseWriter, r *http.Request) {

    if r.Method == http.MethodPost {
        if AuthenthicateUser(r.FormValue( key: "username"),  r.FormValue( key: "password")) <= 0 {
            w.Write([]byte(`<body><script>alert('Please login')</script><h4><button onclick="location.href = '/';" class="float
            return
        }
        myUser.UserName = r.FormValue( key: "username")
        myUser.IsLoggedIn = true
        viewData := &ViewData{PageTitle: "Home", Msg : "Welcome " + myUser.UserName +  " to API-Testing Droid." , HasSessionID:
        ViewHome.SetViewData(viewData).ServeTemplate(w,r)
        return
    }
    if r.Method == http.MethodGet {
        viewData := &ViewData{PageTitle: "Login",HasSessionID: myUser.IsLoggedIn }
        ViewLogin.SetViewData(viewData).ServeTemplate(w,r)
        return
    }
    http.Error(w,  error: "Invalid action.", http.StatusBadRequest)
}

//handles logout func
func logout(w http.ResponseWriter, r *http.Request) {
    if r.Method == http.MethodPost {
        myUser = User{}//reset
        http.Redirect(w, r,  url: "/", http.StatusSeeOther)
        return
    }
    if r.Method == http.MethodGet {
        viewData := &ViewData{PageTitle: "Logout"}
        ViewLogout.SetViewData(viewData).ServeTemplate(w,r)
        return
    }
    http.Error(w,  error: "Invalid action.", http.StatusBadRequest)
}
```

Main.go

5.2.2 Logging

File logging is provided, with different levels such as Trace, Info, Warning, Error.

Log files are rotated daily.



service > logger > logger.go > ...

```go
//Specialised loggers
var (
    Trace   *log.Logger // Just about anything
    Info    *log.Logger // Important information
    Warning *log.Logger // Be concerned
    Error   *log.Logger // Critical problem
)
```

/service/logger/loger.go

5.3 Functional Features - Test Execution Services

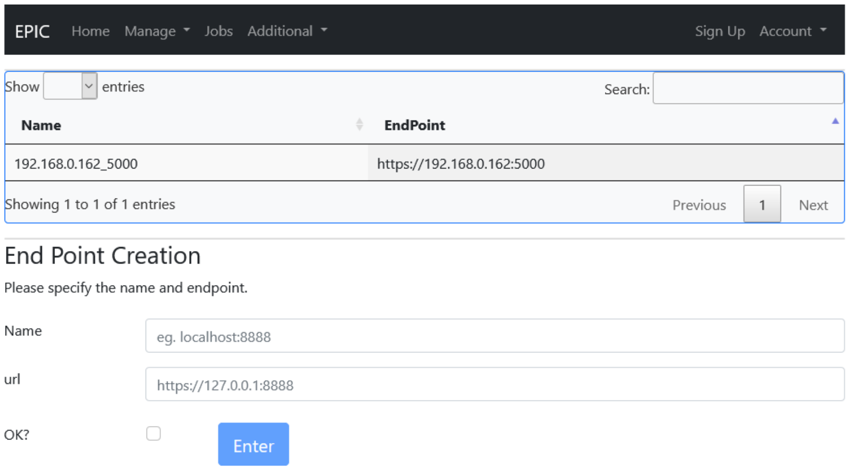
The system shall provide a scaffold for the user to manage Test Cases Creation and Management.

5.3.1 Test EndPoint Management

The system shall provide an interface for user to manage endpoints.

An endpoint consisting of name and the url is created, when user confirms and hit Enter.

apiEndPoint is to populate the data table.



Name	EndPoint
192.168.0.162_5000	https://192.168.0.162:5000

Showing 1 to 1 of 1 entries

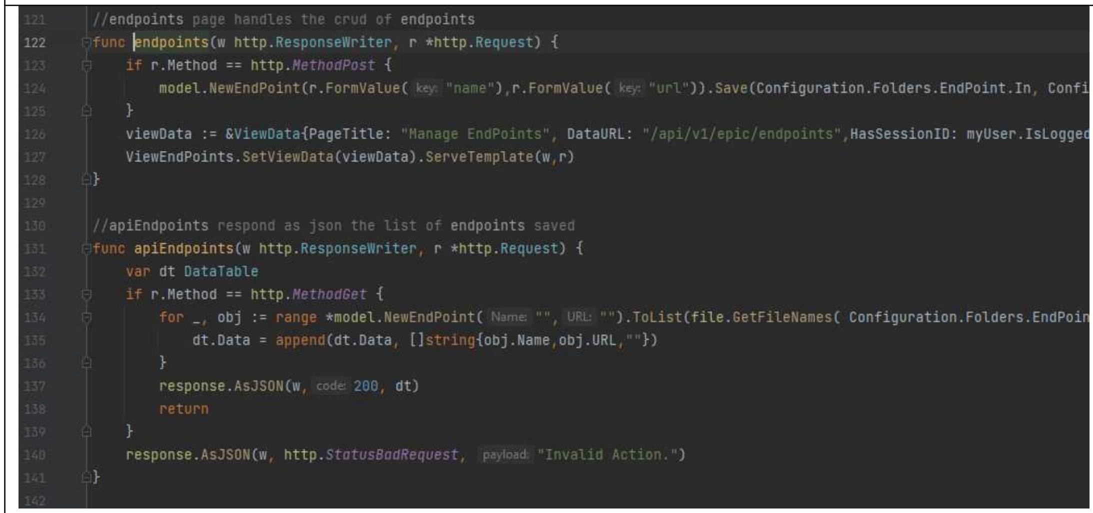
End Point Creation

Please specify the name and endpoint.

Name: eg. localhost:8888

url: https://127.0.0.1:8888

OK? ☐



```go
//endpoints page handles the crud of endpoints
func endpoints(w http.ResponseWriter, r *http.Request) {
    if r.Method == http.MethodPost {
        model.NewEndPoint(r.FormValue( key: "name"),r.FormValue( key: "url")).Save(Configuration.Folders.EndPoint.In, Confi
    }
    viewData := &ViewData{PageTitle: "Manage EndPoints", DataURL: "/api/v1/epic/endpoints",HasSessionID: myUser.IsLogged
    ViewEndPoints.SetViewData(viewData).ServeTemplate(w,r)
}

//apiEndpoints respond as json the list of endpoints saved
func apiEndpoints(w http.ResponseWriter, r *http.Request) {
    var dt DataTable
    if r.Method == http.MethodGet {
        for _, obj := range *model.NewEndPoint( Name: "", URL: "").ToList(file.GetFileNames( Configuration.Folders.EndPoin
            dt.Data = append(dt.Data, []string{obj.Name,obj.URL,""})
        }
        response.AsJSON(w, code: 200, dt)
        return
    }
    response.AsJSON(w, http.StatusBadRequest, payload: "Invalid Action.")
}
```

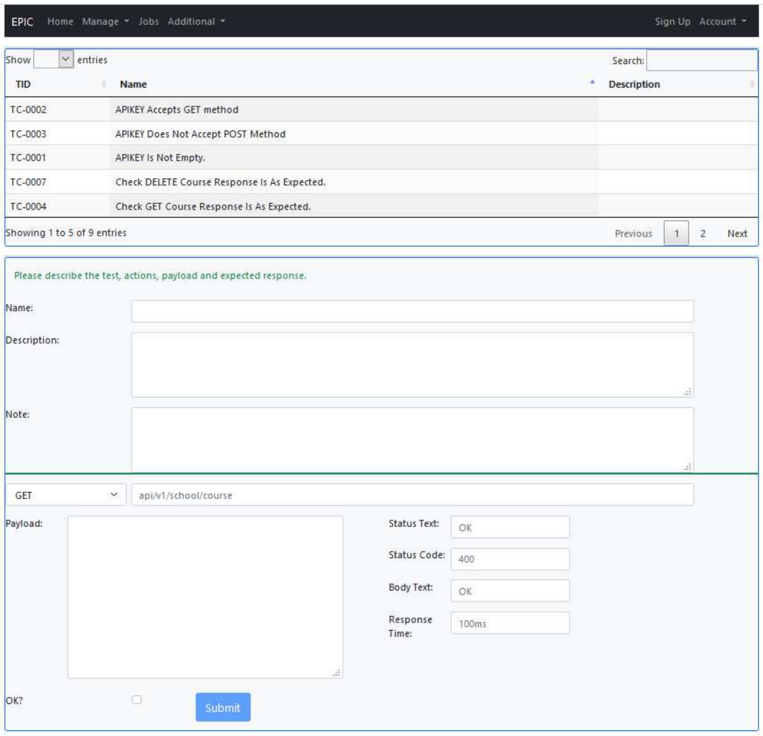
server.go

5.3.2 Test Case Management

The system shall provide an interface for user to manage test cases.

A test case consisting of name, description, notes, action, payload and response is created, when user confirms and hit Enter.

apiCases is to populate the data table.



Show entries Search:

TID	Name	Description
TC-0002	APIKEY Accepts GET method	
TC-0003	APIKEY Does Not Accept POST Method	
TC-0001	APIKEY Is Not Empty.	
TC-0007	Check DELETE Course Response Is As Expected.	
TC-0004	Check GET Course Response Is As Expected.	

Showing 1 to 5 of 9 entries Previous 1 2 Next

Please describe the test, actions, payload and expected response.

Name:

Description:

Note:

GET api/v1/school/course

Payload:

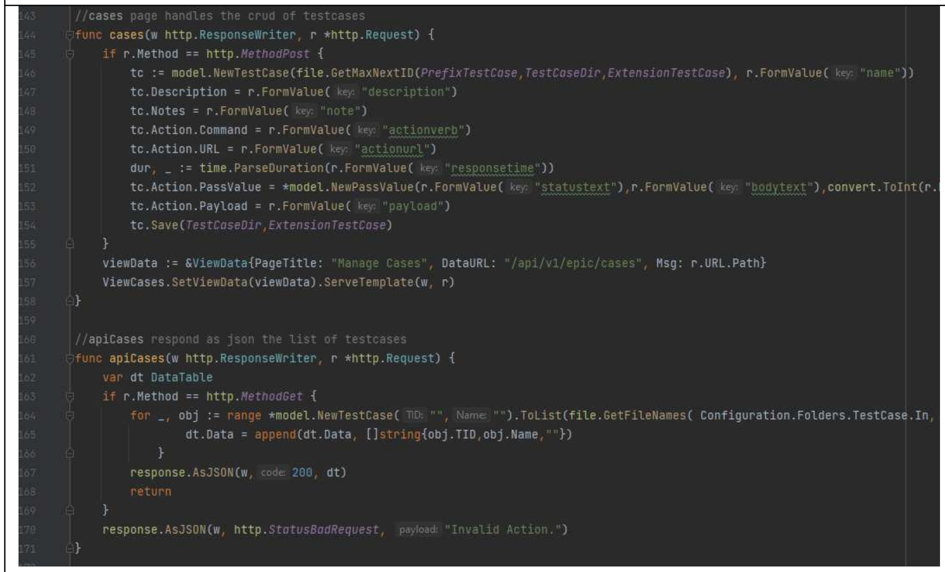
Status Text: OK

Status Code: 400

Body Text: OK

Response Time: 100ms

OK?



```go
//cases page handles the crud of testcases
func cases(w http.ResponseWriter, r *http.Request) {
    if r.Method == http.MethodPost {
        tc := model.NewTestCase(file.GetMaxNextID(PrefixTestCase,TestCaseDir,ExtensionTestCase), r.FormValue( key: "name"))
        tc.Description = r.FormValue( key: "description")
        tc.Notes = r.FormValue( key: "note")
        tc.Action.Command = r.FormValue( key: "actionverb")
        tc.Action.URL = r.FormValue( key: "actionurl")
        dur, _ := time.ParseDuration(r.FormValue( key: "responsetime"))
        tc.Action.PassValue = *model.NewPassValue(r.FormValue( key: "statustext"),r.FormValue( key: "bodytext"),convert.ToInt(r.
        tc.Action.Payload = r.FormValue( key: "payload")
        tc.Save(TestCaseDir,ExtensionTestCase)
    }
    viewData := &ViewData{PageTitle: "Manage Cases", DataURL: "/api/v1/epic/cases", Msg: r.URL.Path}
    ViewCases.SetViewData(viewData).ServeTemplate(w, r)
}

//apiCases respond as json the list of testcases
func apiCases(w http.ResponseWriter, r *http.Request) {
    var dt DataTable
    if r.Method == http.MethodGet {
        for _, obj := range *model.NewTestCase( TID: "", Name: "").ToList(file.GetFileNames( Configuration.Folders.TestCase.In,
                dt.Data = append(dt.Data, []string{obj.TID,obj.Name,""})
            }
        response.AsJSON(w, code: 200, dt)
        return
    }
    response.AsJSON(w, http.StatusBadRequest,  payload: "Invalid Action.")
}
```

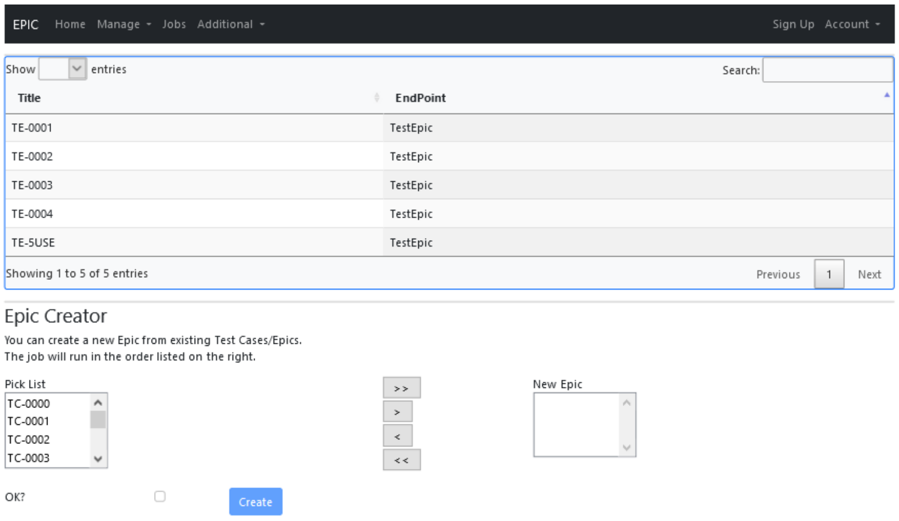
server.go

5.3.3 Test Epic Management

The system shall provide an interface for user to manage test epics.

A test epic is created when test cases(TC*) are selected into the right box, and user confirms and hit Enter.

apiEpics is to populate the data table.



Show entries Search:

Title	EndPoint
TE-0001	TestEpic
TE-0002	TestEpic
TE-0003	TestEpic
TE-0004	TestEpic
TE-5USE	TestEpic

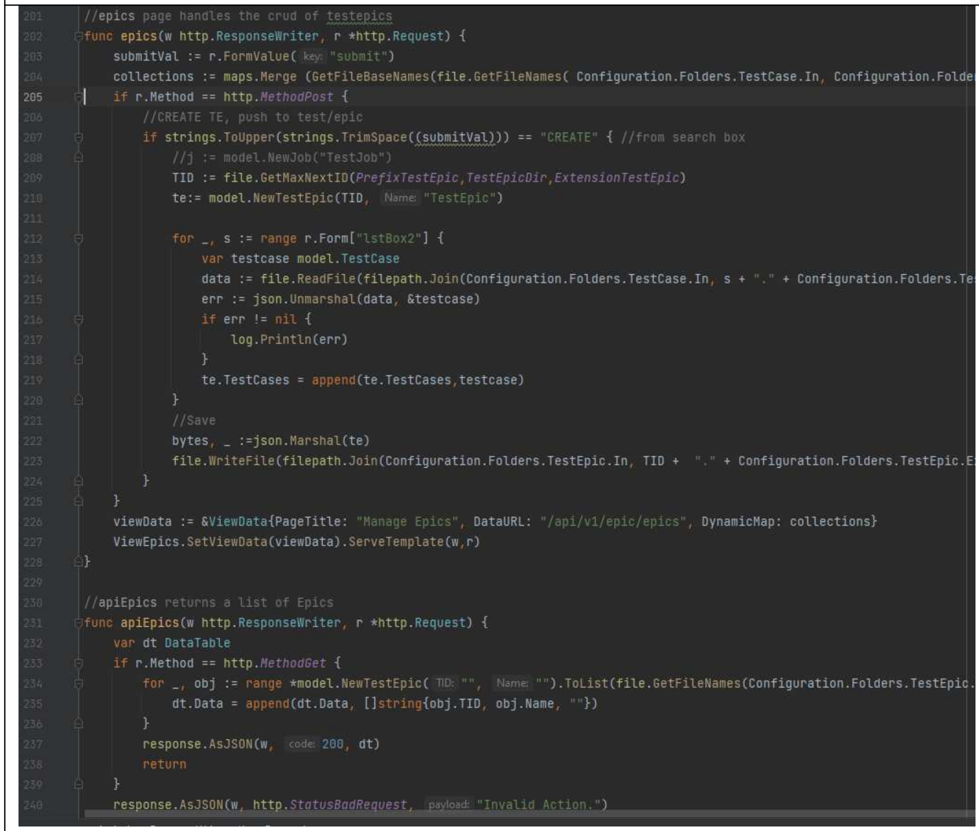
Showing 1 to 5 of 5 entries Previous 1 Next

Epic Creator

You can create a new Epic from existing Test Cases/Epics.
The job will run in the order listed on the right.

Pick List: TC-0000, TC-0001, TC-0002, TC-0003

New Epic


```go
//epics page handles the crud of testepics
func epics(w http.ResponseWriter, r *http.Request) {
    submitVal := r.FormValue( key: "submit")
    collections := maps.Merge (GetFileBaseNames(file.GetFileNames( Configuration.Folders.TestCase.In, Configuration.Folde
    if r.Method == http.MethodPost {
        //CREATE TE, push to test/epic
        if strings.ToUpper(strings.TrimSpace((submitVal))) == "CREATE" { //from search box
            //j := model.NewJob("TestJob")
            TID := file.GetMaxNextID(PrefixTestEpic,TestEpicDir,ExtensionTestEpic)
            te:= model.NewTestEpic(TID,  Name: "TestEpic")

            for _, s := range r.Form["lstBox2"] {
                var testcase model.TestCase
                data := file.ReadFile(filepath.Join(Configuration.Folders.TestCase.In, s + "." + Configuration.Folders.Te
                err := json.Unmarshal(data, &testcase)
                if err != nil {
                    log.Println(err)
                }
                te.TestCases = append(te.TestCases,testcase)
            }
            //Save
            bytes, _ :=json.Marshal(te)
            file.WriteFile(filepath.Join(Configuration.Folders.TestEpic.In, TID +  "." + Configuration.Folders.TestEpic.E
        }
    }
    viewData := &ViewData{PageTitle: "Manage Epics", DataURL: "/api/v1/epic/epics", DynamicMap: collections}
    ViewEpics.SetViewData(viewData).ServeTemplate(w,r)
}

//apiEpics returns a list of Epics
func apiEpics(w http.ResponseWriter, r *http.Request) {
    var dt DataTable
    if r.Method == http.MethodGet {
        for _, obj := range *model.NewTestEpic( TID: "",  Name: "").ToList(file.GetFileNames(Configuration.Folders.TestEpic.
            dt.Data = append(dt.Data, []string{obj.TID, obj.Name, ""})
        }
        response.AsJSON(w,  code: 200, dt)
        return
    }
    response.AsJSON(w, http.StatusBadRequest,  payload: "Invalid Action.")
```

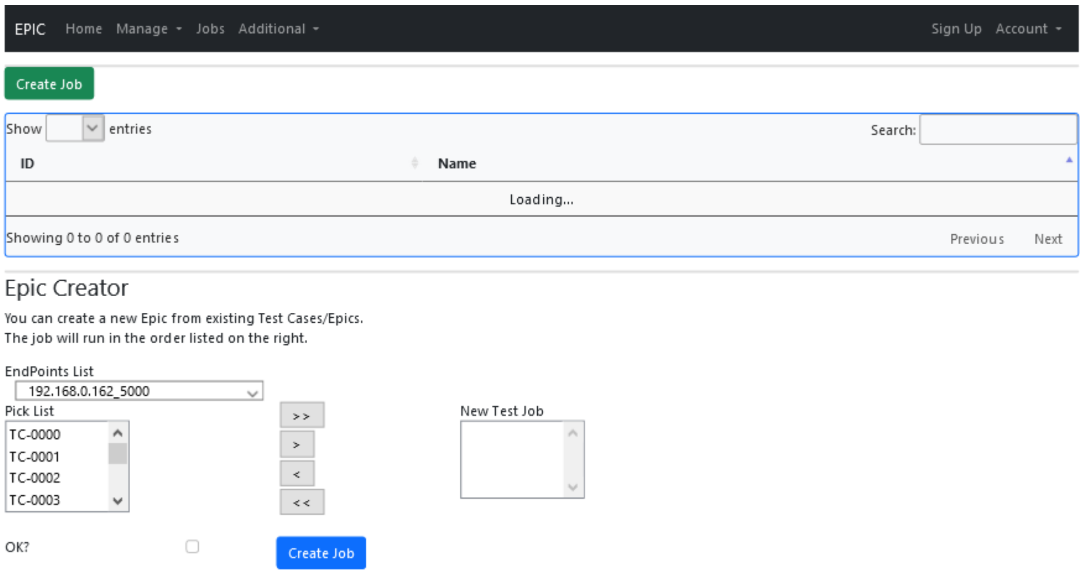
server.go

5.3.4 Test Job Management

The system shall provide an interface for user to manage test jobs.

A test job is created when test cases(TC*) and/or test epics(TE*) are selected into the right box, and user confirms and hit Enter.

apiJobs is to populate the data table.



Show entries Search:

ID	Name
Loading...	

Showing 0 to 0 of 0 entries Previous Next

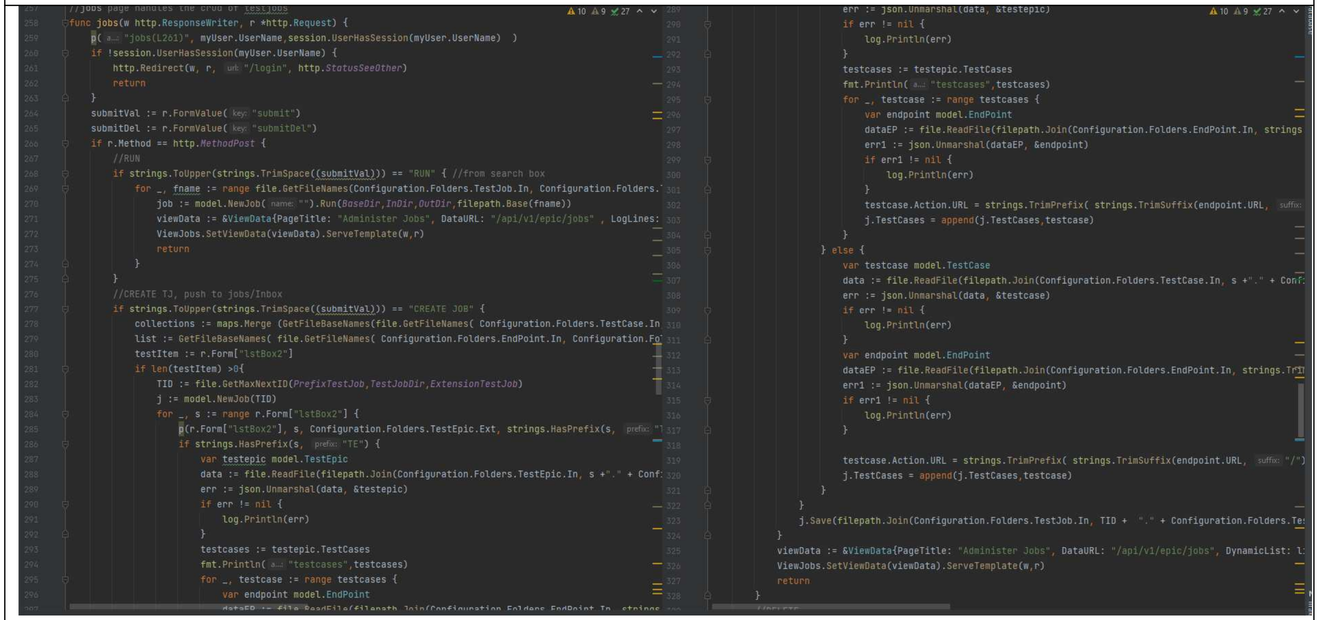
Epic Creator

You can create a new Epic from existing Test Cases/Epics.
The job will run in the order listed on the right.

EndPoints List: 192.168.0.162_5000

Pick List: TC-0000, TC-0001, TC-0002, TC-0003

New Test Job



```go
//jobs page handles the crud of testjobs
func jobs(w http.ResponseWriter, r *http.Request) {
    p( a...: "jobs(L261)", myUser.UserName,session.UserHasSession(myUser.UserName)  )
    if !session.UserHasSession(myUser.UserName) {
        http.Redirect(w, r,  url: "/login", http.StatusSeeOther)
        return
    }
    submitVal := r.FormValue( key: "submit")
    submitDel := r.FormValue( key: "submitDel")
    if r.Method == http.MethodPost {
        //RUN
        if strings.ToUpper(strings.TrimSpace((submitVal))) == "RUN" { //from search box
            for _, fname := range file.GetFileNames(Configuration.Folders.TestJob.In, Configuration.Folders.
                job := model.NewJob( name: "").Run(BaseDir,InDir,OutDir,filepath.Base(fname))
                viewData := &ViewData{PageTitle: "Administer Jobs", DataURL: "/api/v1/epic/jobs" , LogLines:
                ViewJobs.SetViewData(viewData).ServeTemplate(w,r)
                return
            }
        }
        //CREATE TJ, push to jobs/Inbox
        if strings.ToUpper(strings.TrimSpace((submitVal))) == "CREATE JOB" {
            collections := maps.Merge (GetFileBaseNames(file.GetFileNames( Configuration.Folders.TestCase.In
            list := GetFileBaseNames( file.GetFileNames( Configuration.Folders.EndPoint.In, Configuration.Fo
            testItem := r.Form["lstBox2"]
            if len(testItem) >0{
                TID := file.GetMaxNextID(PrefixTestJob,TestJobDir,ExtensionTestJob)
                j := model.NewJob(TID)
                for _, s := range r.Form["lstBox2"] {
                    p(r.Form["lstBox2"], s, Configuration.Folders.TestEpic.Ext, strings.HasPrefix(s,  prefix: "
                    if strings.HasPrefix(s,  prefix: "TE") {
                        var testepic model.TestEpic
                        data := file.ReadFile(filepath.Join(Configuration.Folders.TestEpic.In, s +"." + Conf
                        err := json.Unmarshal(data, &testepic)
                        if err != nil {
                            log.Println(err)
                        }
                        testcases := testepic.TestCases
                        fmt.Println( a...: "testcases",testcases)
                        for _, testcase := range testcases {
                            var endpoint model.EndPoint
                            dataEP := file.ReadFile(filepath.Join(Configuration.Folders.EndPoint.In, strings
```

```go
                        err := json.Unmarshal(data, &testepic)
                        if err != nil {
                            log.Println(err)
                        }
                        testcases := testepic.TestCases
                        fmt.Println( a...: "testcases",testcases)
                        for _, testcase := range testcases {
                            var endpoint model.EndPoint
                            dataEP := file.ReadFile(filepath.Join(Configuration.Folders.EndPoint.In, strings
                            err1 := json.Unmarshal(dataEP, &endpoint)
                            if err1 != nil {
                                log.Println(err)
                            }
                            testcase.Action.URL = strings.TrimPrefix( strings.TrimSuffix(endpoint.URL,  suffix:
                            j.TestCases = append(j.TestCases,testcase)
                        }
                    } else {
                        var testcase model.TestCase
                        data := file.ReadFile(filepath.Join(Configuration.Folders.TestCase.In, s +"." + Conf
                        err := json.Unmarshal(data, &testcase)
                        if err != nil {
                            log.Println(err)
                        }
                        var endpoint model.EndPoint
                        dataEP := file.ReadFile(filepath.Join(Configuration.Folders.EndPoint.In, strings.Tri
                        err1 := json.Unmarshal(dataEP, &endpoint)
                        if err1 != nil {
                            log.Println(err)
                        }

                        testcase.Action.URL = strings.TrimPrefix( strings.TrimSuffix(endpoint.URL,  suffix: "/")
                        j.TestCases = append(j.TestCases,testcase)
                    }
                }
                j.Save(filepath.Join(Configuration.Folders.TestJob.In, TID +  "." + Configuration.Folders.Te
            }
            viewData := &ViewData{PageTitle: "Administer Jobs", DataURL: "/api/v1/epic/jobs", DynamicList: l
            ViewJobs.SetViewData(viewData).ServeTemplate(w,r)
            return
        }
        //DELETE
```

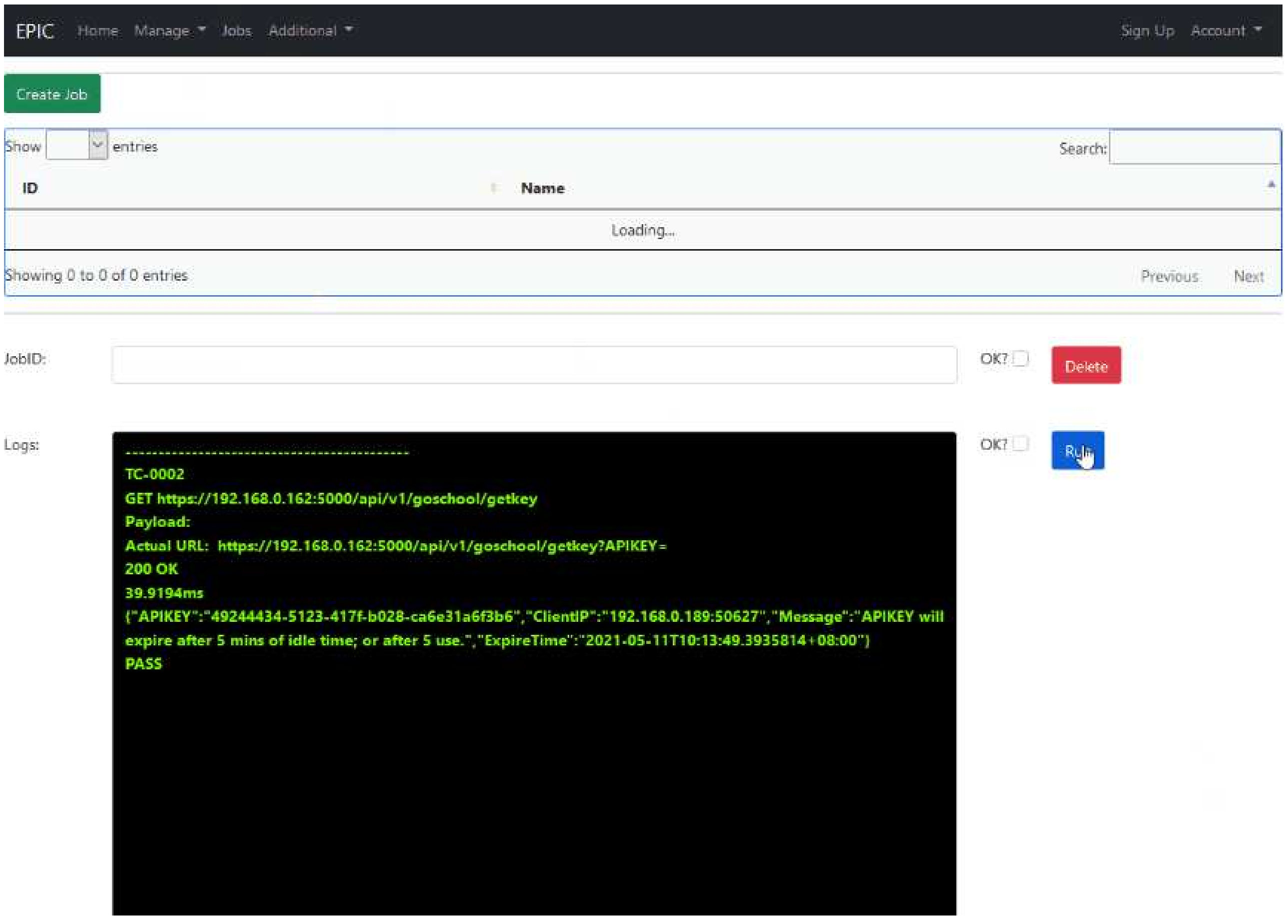
server.go

5.3.5 Test Execution

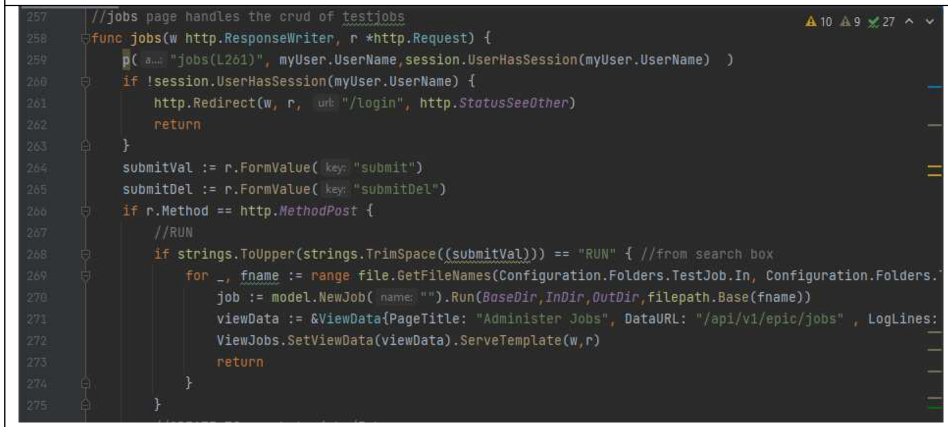
The system shall provide an interface for user to select Test Job for running. The test execution shall execute the actions, and check the response for status, response times, and other defined pass critieria.

A test job is can be executed when use click to select a job from the table, checks confirms and hit Run.

apiJobs is to populate the data table.



```
---------------------------------------------
TC-0002
GET https://192.168.0.162:5000/api/v1/goschool/getkey
Payload:
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/getkey?APIKEY=
200 OK
39.9194ms
{"APIKEY":"49244434-5123-417f-b028-ca6e31a6f3b6","ClientIP":"192.168.0.189:50627","Message":"APIKEY will
expire after 5 mins of idle time; or after 5 use.","ExpireTime":"2021-05-11T10:13:49.3935814+08:00"}
PASS
```



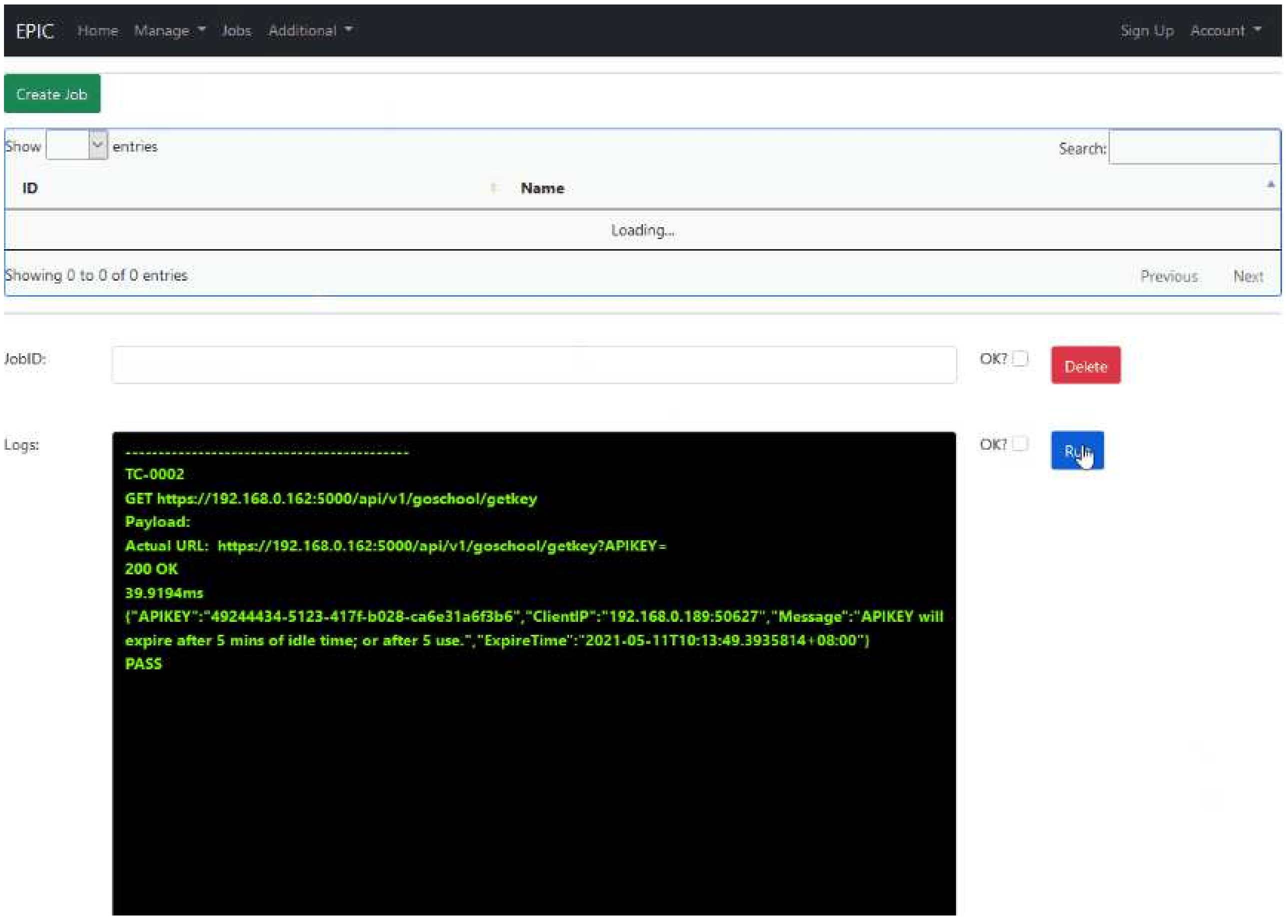
```go
//jobs page handles the crud of testjobs
func jobs(w http.ResponseWriter, r *http.Request) {
    p( a... "jobs(L261)", myUser.UserName,session.UserHasSession(myUser.UserName)  )
    if !session.UserHasSession(myUser.UserName) {
        http.Redirect(w, r,  url: "/login", http.StatusSeeOther)
        return
    }
    submitVal := r.FormValue( key: "submit")
    submitDel := r.FormValue( key: "submitDel")
    if r.Method == http.MethodPost {
        //RUN
        if strings.ToUpper(strings.TrimSpace((submitVal))) == "RUN" { //from search box
            for _, fname := range file.GetFileNames(Configuration.Folders.TestJob.In, Configuration.Folders.
                job := model.NewJob( name: "").Run(BaseDir,InDir,OutDir,filepath.Base(fname))
                viewData := &ViewData{PageTitle: "Administer Jobs", DataURL: "/api/v1/epic/jobs" , LogLines:
                ViewJobs.SetViewData(viewData).ServeTemplate(w,r)
                return
            }
        }
```

server.go

5.3.6 Test Reporting

The system shall provide an interface for the user to see the test response and result for the test runned.

Currently, the result is shown in the below panel when a test job is run.



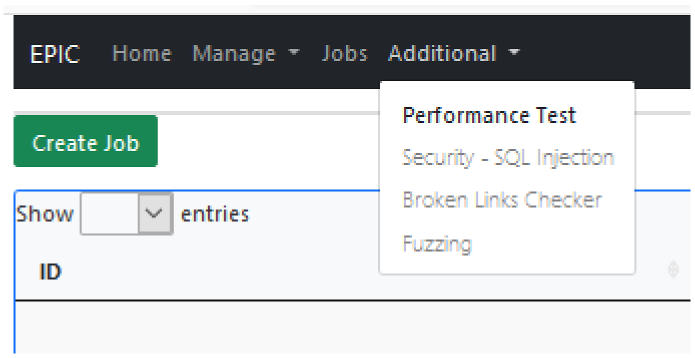
```
--------------------------------------------
TC-0002
GET https://192.168.0.162:5000/api/v1/goschool/getkey
Payload:
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/getkey?APIKEY=
200 OK
39.9194ms
{"APIKEY":"49244434-5123-417f-b028-ca6e31a6f3b6","ClientIP":"192.168.0.189:50627","Message":"APIKEY will
expire after 5 mins of idle time; or after 5 use.","ExpireTime":"2021-05-11T10:13:49.3935814+08:00"}
PASS
```

5.3.7 Tests Addition

The system shall provide a scaffold to allow additions of other test functionalities. Such test functionalities can include such as:

- Load Test
- Security Test(Injections)
- Expired Links
- Fuzzing

The Additional Menu allows sub-menus to be added.



5.3.8 Authentication Practices

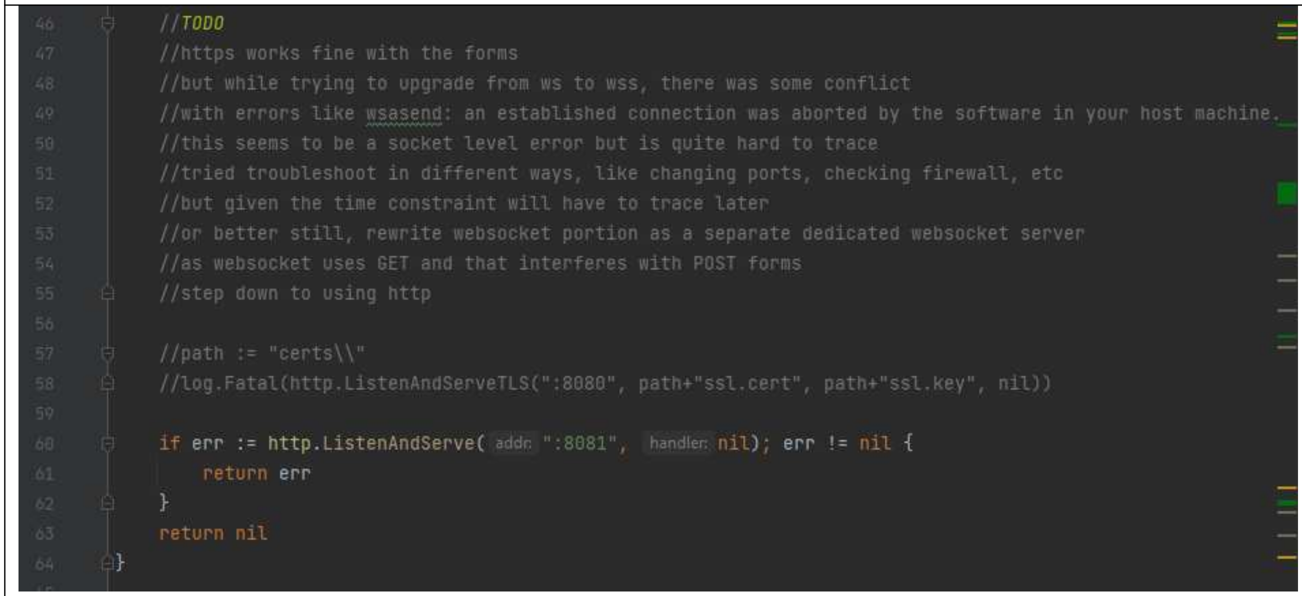
In this version, a simple user authentication is achieved via the function, with users information stored in SQLite. In previous assignmets the database used was mySQL, this switch is to explore the capabilities and compare of SQLite.

5.3.9 Communications Security

As the API is internet facing, https is used to protect against Man-In-The-Middle-Attack (MITM)attacks where the attacker is able to intercept and read the non-encrypt traffic.

In production, the self-signed certifcate is replaced with an actual signed certificate.

Currently, because of an issue with wss, it is stepped down to http.

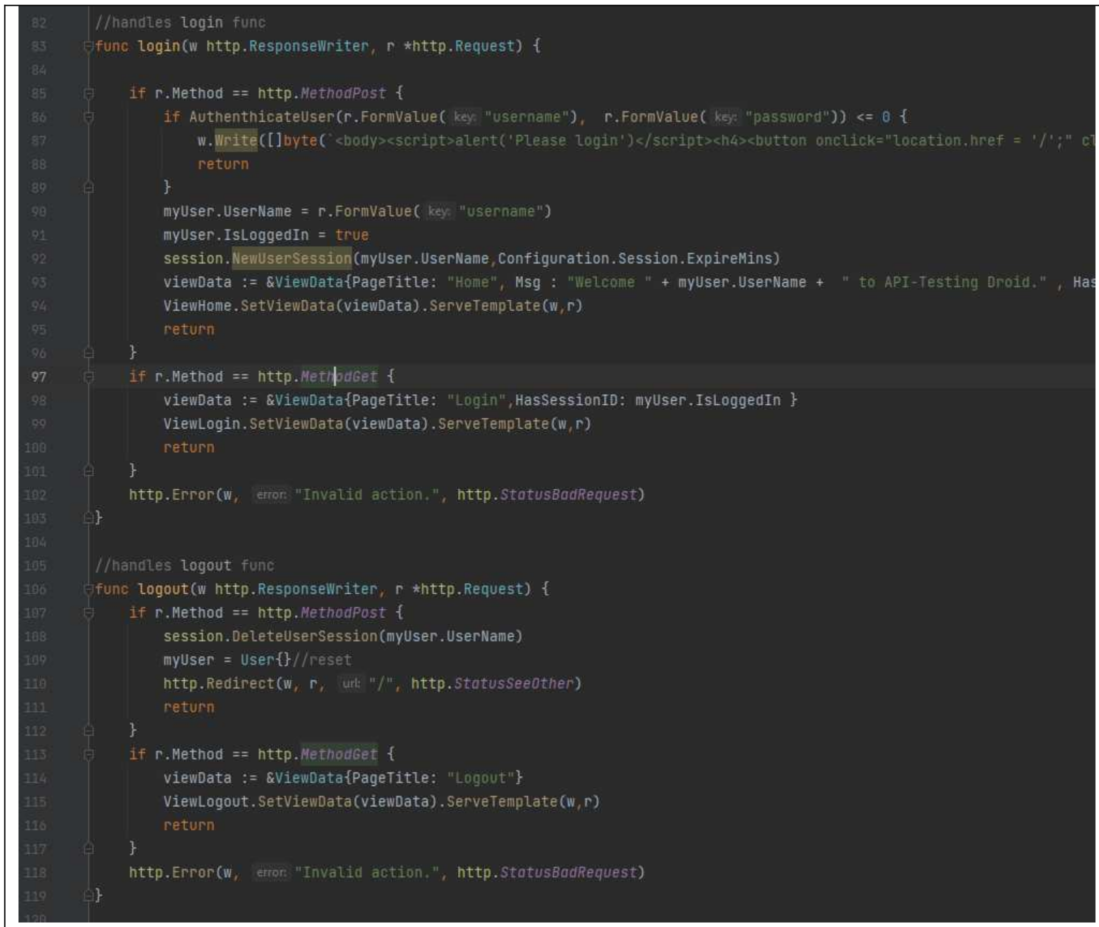


```
    //TODO
    //https works fine with the forms
    //but while trying to upgrade from ws to wss, there was some conflict
    //with errors like wsasend: an established connection was aborted by the software in your host machine.
    //this seems to be a socket level error but is quite hard to trace
    //tried troubleshoot in different ways, like changing ports, checking firewall, etc
    //but given the time constraint will have to trace later
    //or better still, rewrite websocket portion as a separate dedicated websocket server
    //as websocket uses GET and that interferes with POST forms
    //step down to using http

    //path := "certs\\"
    //log.Fatal(http.ListenAndServeTLS(":8080", path+"ssl.cert", path+"ssl.key", nil))

    if err := http.ListenAndServe( addr: ":8081",  handler: nil); err != nil {
        return err
    }
    return nil
}
```

server.go

5.3.10 Session Management

- Once login is provided a session is provided with expiry defined as Configuration.Session.ExpireMins, which is configurable via config.json
- Concurrent login is disallowed for the same username as the session is mapped to the username.
- A goroutine also checks evey minute to remove stale sessions.

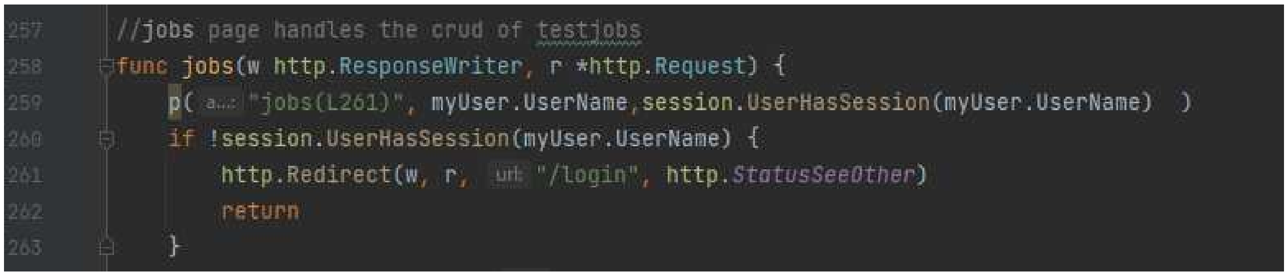


```go
//handles login func
func login(w http.ResponseWriter, r *http.Request) {

    if r.Method == http.MethodPost {
        if AuthenthicateUser(r.FormValue("username"),  r.FormValue("password")) <= 0 {
            w.Write([]byte(`<body><script>alert('Please login')</script><h4><button onclick="location.href = '/';" cl
            return
        }
        myUser.UserName = r.FormValue("username")
        myUser.IsLoggedIn = true
        session.NewUserSession(myUser.UserName,Configuration.Session.ExpireMins)
        viewData := &ViewData{PageTitle: "Home", Msg : "Welcome " + myUser.UserName +  " to API-Testing Droid." , Has
        ViewHome.SetViewData(viewData).ServeTemplate(w,r)
        return
    }
    if r.Method == http.MethodGet {
        viewData := &ViewData{PageTitle: "Login",HasSessionID: myUser.IsLoggedIn }
        ViewLogin.SetViewData(viewData).ServeTemplate(w,r)
        return
    }
    http.Error(w, "Invalid action.", http.StatusBadRequest)
}

//handles logout func
func logout(w http.ResponseWriter, r *http.Request) {
    if r.Method == http.MethodPost {
        session.DeleteUserSession(myUser.UserName)
        myUser = User{}//reset
        http.Redirect(w, r, "/", http.StatusSeeOther)
        return
    }
    if r.Method == http.MethodGet {
        viewData := &ViewData{PageTitle: "Logout"}
        ViewLogout.SetViewData(viewData).ServeTemplate(w,r)
        return
    }
    http.Error(w, "Invalid action.", http.StatusBadRequest)
}
```

Server.go **Jutil/v1/session/session.go**

For those Protected Pages, which is currently defined as those job execution pages,(Job Menu and LoadTest Menu), session control is applied.

Protection can be applied by including these lines on the required func.



```go
//jobs page handles the crud of testjobs
func jobs(w http.ResponseWriter, r *http.Request) {
    p("jobs(L261)", myUser.UserName,session.UserHasSession(myUser.UserName)  )
    if !session.UserHasSession(myUser.UserName) {
        http.Redirect(w, r, "/login", http.StatusSeeOther)
        return
    }
```

5.4 Coding Best Practice(Idiomatic Go)

This section describes the Idioms the project has(tried to) use.

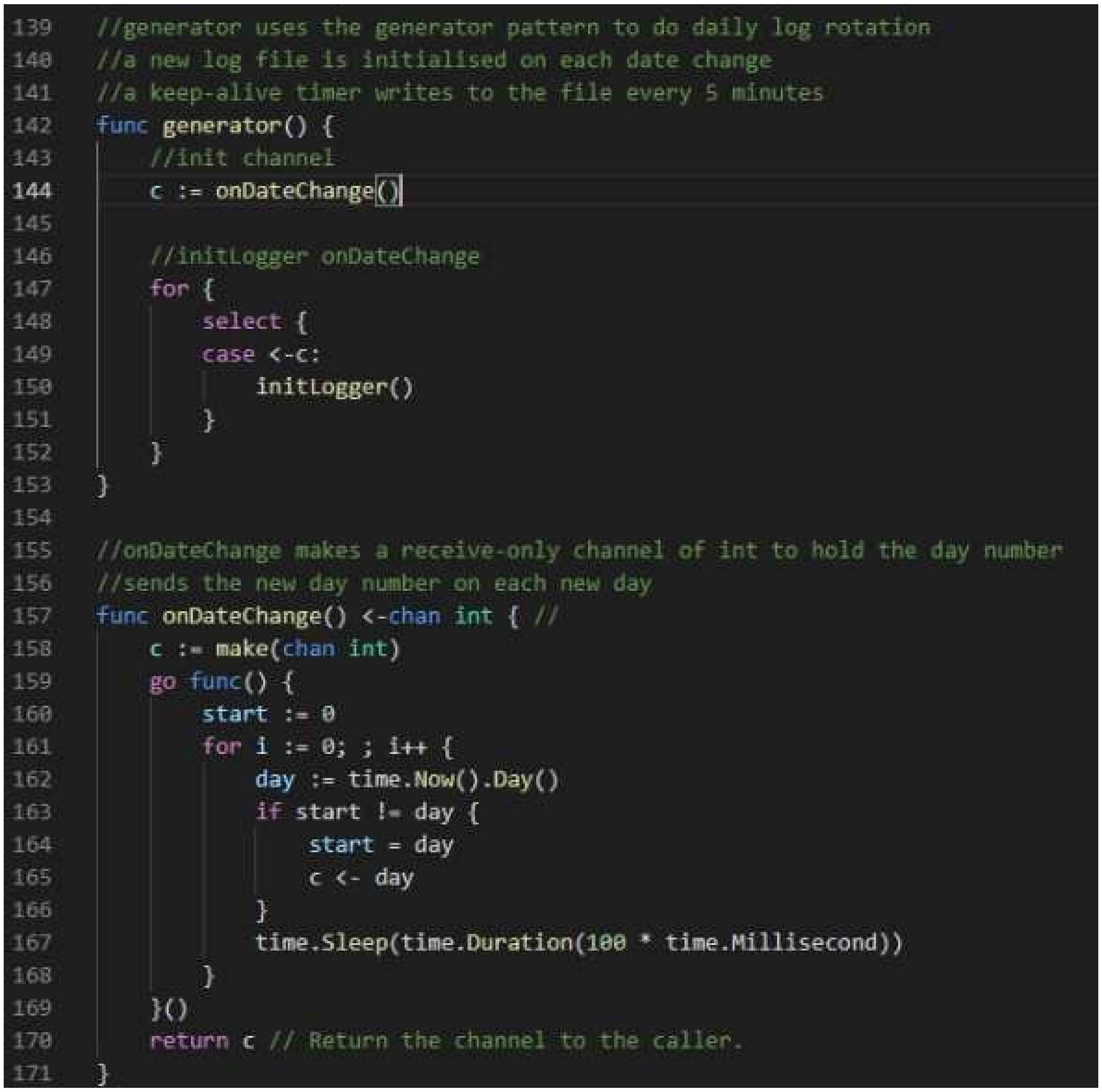
5.4.1 Generator Pattern

In the logging service, a generator pattern is employed to handle log rotation. The log files are rotated on each new day. A separate goroutine is used to initialize and point to the new log file.

It is the same algorithm used in LOGASS(Logging-As-A-Service) introduced in Assignment Go-In-Action 2.

The onDateChange checks every 100 millisecs to see if it is a new day, and will generate an integer(DayNum) on each new day.

The receiver than runs initLogger.

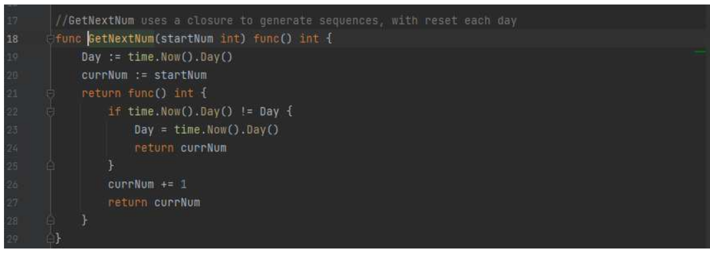


```go
//generator uses the generator pattern to do daily log rotation
//a new log file is initialised on each date change
//a keep-alive timer writes to the file every 5 minutes
func generator() {
    //init channel
    c := onDateChange()

    //initLogger onDateChange
    for {
        select {
        case <-c:
            initLogger()
        }
    }
}

//onDateChange makes a receive-only channel of int to hold the day number
//sends the new day number on each new day
func onDateChange() <-chan int { //
    c := make(chan int)
    go func() {
        start := 0
        for i := 0; ; i++ {
            day := time.Now().Day()
            if start != day {
                start = day
                c <- day
            }
            time.Sleep(time.Duration(100 * time.Millisecond))
        }
    }()
    return c // Return the channel to the caller.
}
```

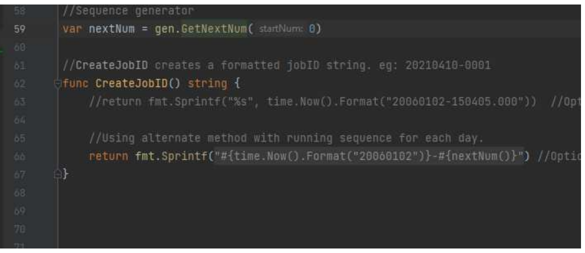
/jutil/v1/logger/logger.go

5.4.2 Closure Pattern

The func GetNextNum uses a closure to generate sequences, with reset each day. The job file are named in the pattern of 20210410-0001.json as shown.



```go
//GetNextNum uses a closure to generate sequences, with reset each day
func GetNextNum(startNum int) func() int {
    Day := time.Now().Day()
    currNum := startNum
    return func() int {
        if time.Now().Day() != Day {
            Day = time.Now().Day()
            return currNum
        }
        currNum += 1
        return currNum
    }
}
```

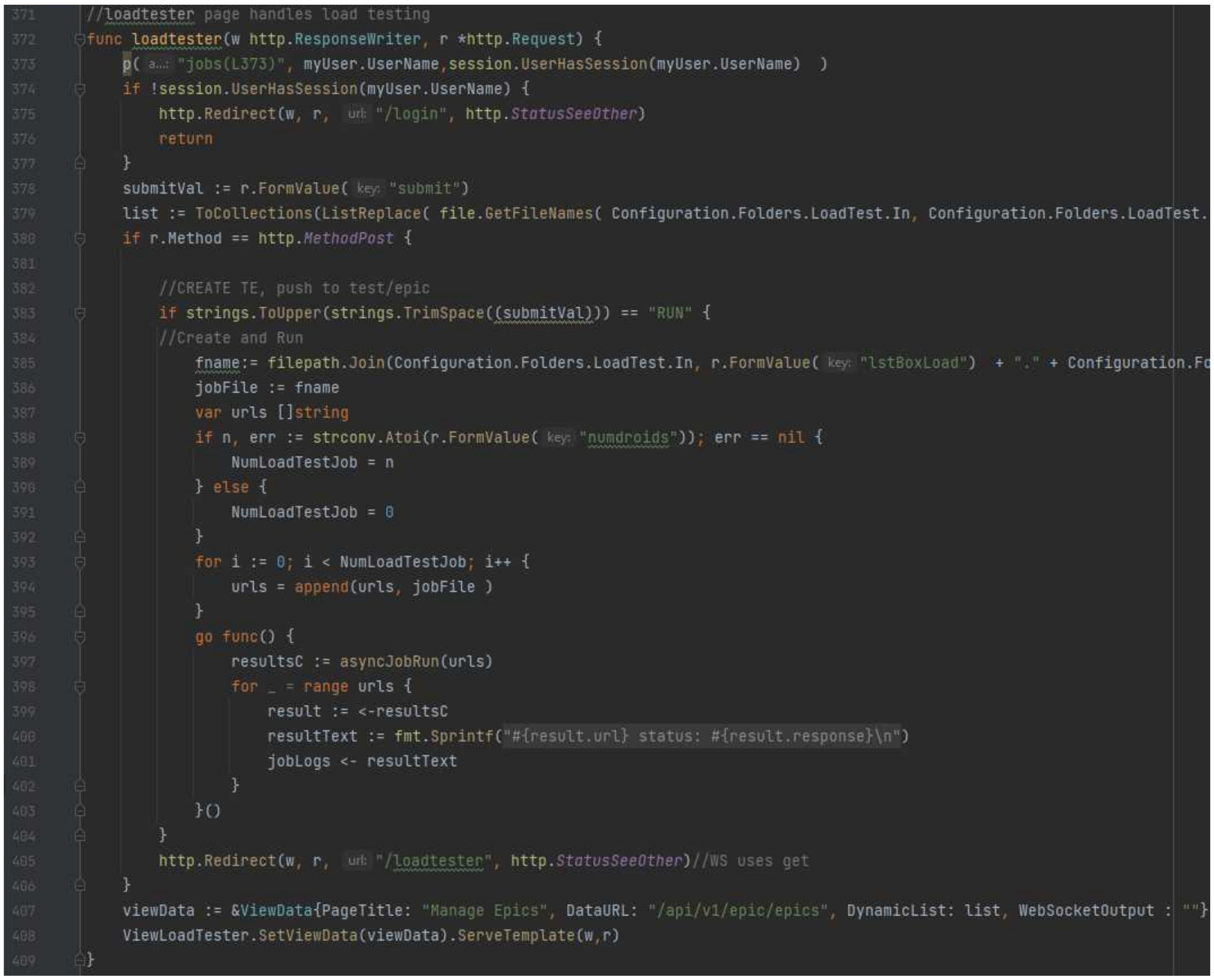


```go
//Sequence generator
var nextNum = gen.GetNextNum( startNum: 0)

//CreateJobID creates a formatted jobID string. eg: 20210410-0001
func CreateJobID() string {
    //return fmt.Sprintf("%s", time.Now().Format("20060102-150405.000"))  //Opt

    //Using alternate method with running sequence for each day.
    return fmt.Sprintf("#{time.Now().Format("20060102")}-#{nextNum()}") //Optic
}
```

jutil/v1/gen/gen.go **Model/jobs.go**

5.4.3 Go Routine Fan Out Pattern with Channel Aggregation of Response

In the Load Test function, goroutine was use to fan out and execute the N loads.
The response is then read off from the result channel.
This is then pushed to the websocket handler(wsHandler) to write out to browser.



```go
//loadtester page handles load testing
func loadtester(w http.ResponseWriter, r *http.Request) {
    p( a...: "jobs(L373)", myUser.UserName,session.UserHasSession(myUser.UserName)  )
    if !session.UserHasSession(myUser.UserName) {
        http.Redirect(w, r,  url: "/login", http.StatusSeeOther)
        return
    }
    submitVal := r.FormValue( key: "submit")
    list := ToCollections(ListReplace( file.GetFileNames( Configuration.Folders.LoadTest.In, Configuration.Folders.LoadTest.
    if r.Method == http.MethodPost {

        //CREATE TE, push to test/epic
        if strings.ToUpper(strings.TrimSpace((submitVal))) == "RUN" {
        //Create and Run
            fname:= filepath.Join(Configuration.Folders.LoadTest.In, r.FormValue( key: "lstBoxLoad")  + "." + Configuration.Fo
            jobFile := fname
            var urls []string
            if n, err := strconv.Atoi(r.FormValue( key: "numdroids")); err == nil {
                NumLoadTestJob = n
            } else {
                NumLoadTestJob = 0
            }
            for i := 0; i < NumLoadTestJob; i++ {
                urls = append(urls, jobFile )
            }
            go func() {
                resultsC := asyncJobRun(urls)
                for _ = range urls {
                    result := <-resultsC
                    resultText := fmt.Sprintf("#{result.url} status: #{result.response}\n")
                    jobLogs <- resultText
                }
            }()
        }
        http.Redirect(w, r,  url: "/loadtester", http.StatusSeeOther)//WS uses get
    }
    viewData := &ViewData{PageTitle: "Manage Epics", DataURL: "/api/v1/epic/epics", DynamicList: list, WebSocketOutput : ""}
    ViewLoadTester.SetViewData(viewData).ServeTemplate(w,r)
}
```

Server.go

6 SYSTEM TESTING

6.1 Unit Test

Each of the functions have had gone through repeated unit tests through the sprint cycle and bugs were corrected. It will not be documented here due to sheer number.

6.2 Integrated Test Cases

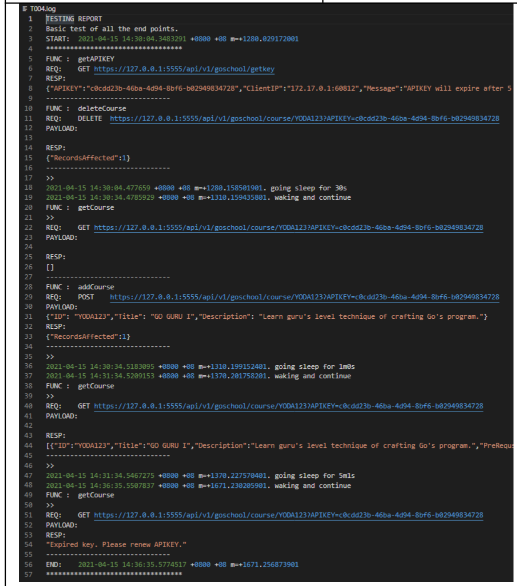
For this project, the testing strategy will be an ‘equivalence test’, a statistical sampling of the output based on the previous assignment and this, since they are both calling the same backend REST API.

Test data(request json payload) is in Appendix 11.1

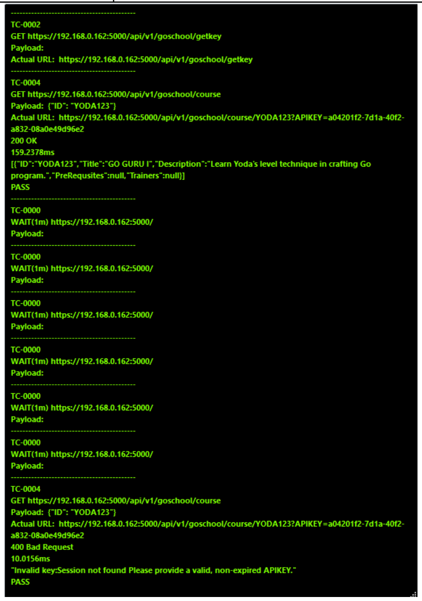
A Test Guide in html is provided separately on how to set up. Sample of some previous test runs are also included.

6.2.1 T004 Verify API Expires After 5 mins Of Idle Time.

Date	2021-04-15 11:32:11 AM	
Test Case ID	T004	
Test Scenario	Verify API Expires After 5 mins Of Idle Time is valid.	
Test Steps	Using any REST client, execute the following actions or its equivalent, in the following sequence: getAPIKEY,deleteCourse,Wait(30s),getCourse,addCourse,Wait(1m),getCourse,Wait(5m1s),getCourse	
Test Data	Refer to addcourse.req,updatecourse.req,deletecourse.req	
Expected Results	**Actual Results**	**Pass/Fail**
Verify the presence of the following message string. "Expired key. Please renew APIKEY."	As Expected.	Pass



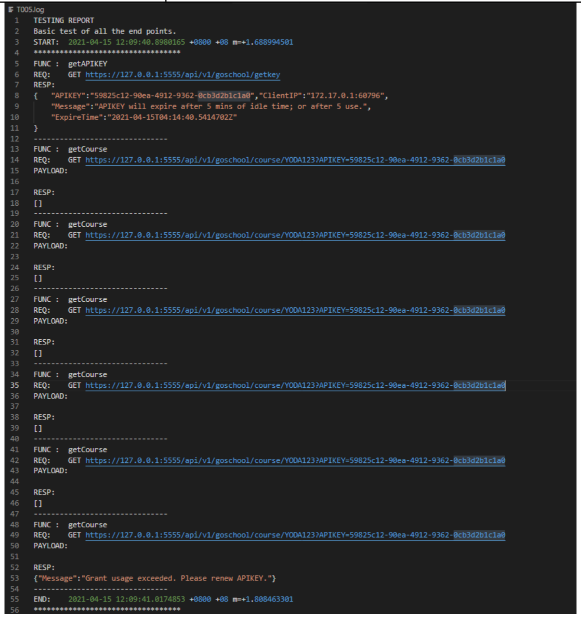
```
T004.log
TESTING REPORT
Basic test of all the end points.
START:  2021-04-15 14:30:04.3483291 +0800 +08 m=+1280.029172001
**********************************
FUNC :  getAPIKEY
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/getkey
RESP:
{"APIKEY":"c0cdd23b-46ba-4d94-8bf6-b02949834728","ClientIP":"172.17.0.1:60812","Message":"APIKEY will expire after 5
-------------------------------
FUNC :  deleteCourse
REQ:    DELETE  https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=c0cdd23b-46ba-4d94-8bf6-b02949834728
PAYLOAD:

RESP:
{"RecordsAffected":1}
-------------------------------
>>
2021-04-15 14:30:04.477659 +0800 +08 m=+1280.158501901. going sleep for 30s
2021-04-15 14:30:34.4785929 +0800 +08 m=+1310.159435801. waking and continue
FUNC :  getCourse
>>
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=c0cdd23b-46ba-4d94-8bf6-b02949834728
PAYLOAD:

RESP:
[]
-------------------------------
FUNC :  addCourse
REQ:    POST    https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=c0cdd23b-46ba-4d94-8bf6-b02949834728
PAYLOAD:
{"ID": "YODA123","Title": "GO GURU I","Description": "Learn guru's level technique of crafting Go's program."}
RESP:
{"RecordsAffected":1}
-------------------------------
>>
2021-04-15 14:30:34.5183095 +0800 +08 m=+1310.199152401. going sleep for 1m0s
2021-04-15 14:31:34.5209153 +0800 +08 m=+1370.201758201. waking and continue
FUNC :  getCourse
>>
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=c0cdd23b-46ba-4d94-8bf6-b02949834728
PAYLOAD:

RESP:
[{"ID":"YODA123","Title":"GO GURU I","Description":"Learn guru's level technique of crafting Go's program.","PreRequs
-------------------------------
>>
2021-04-15 14:31:34.5467275 +0800 +08 m=+1370.227570401. going sleep for 5m1s
2021-04-15 14:36:35.5507837 +0800 +08 m=+1671.230205901. waking and continue
FUNC :  getCourse
>>
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=c0cdd23b-46ba-4d94-8bf6-b02949834728
PAYLOAD:
RESP:
"Expired key. Please renew APIKEY."
-------------------------------
END:    2021-04-15 14:36:35.5774517 +0800 +08 m=+1671.256873901
**********************************
```



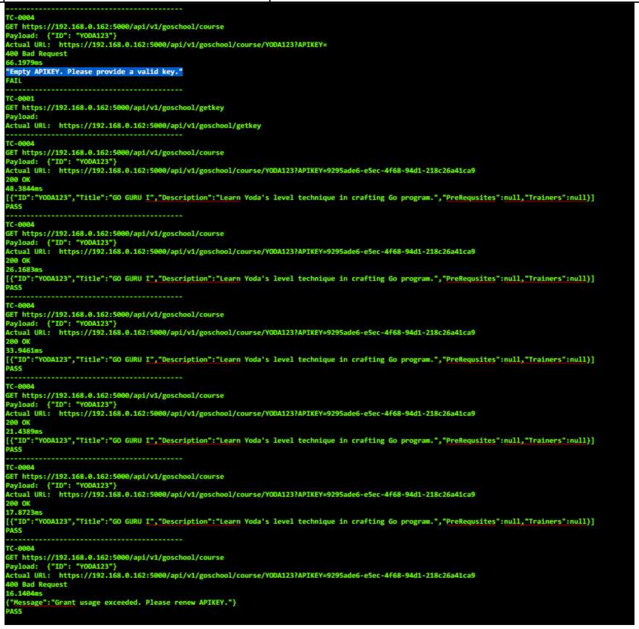
```
-----------------------------------------
TC-0002
GET https://192.168.0.162:5000/api/v1/goschool/getkey
Payload:
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/getkey
-----------------------------------------
TC-0004
GET https://192.168.0.162:5000/api/v1/goschool/course
Payload:  {"ID": "YODA123"}
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/course/YODA123?APIKEY=a04201f2-7d1a-40f2-
a832-08a0e49d96e2
200 OK
159.2378ms
[{"ID":"YODA123","Title":"GO GURU I","Description":"Learn Yoda's level technique in crafting Go
program.","PreRequsites":null,"Trainers":null}]
PASS
-----------------------------------------
TC-0000
WAIT(1m) https://192.168.0.162:5000/
Payload:
-----------------------------------------
TC-0000
WAIT(1m) https://192.168.0.162:5000/
Payload:
-----------------------------------------
TC-0000
WAIT(1m) https://192.168.0.162:5000/
Payload:
-----------------------------------------
TC-0000
WAIT(1m) https://192.168.0.162:5000/
Payload:
-----------------------------------------
TC-0000
WAIT(1m) https://192.168.0.162:5000/
Payload:
-----------------------------------------
TC-0000
WAIT(1m) https://192.168.0.162:5000/
Payload:
-----------------------------------------
TC-0004
GET https://192.168.0.162:5000/api/v1/goschool/course
Payload:  {"ID": "YODA123"}
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/course/YODA123?APIKEY=a04201f2-7d1a-40f2-
a832-08a0e49d96e2
400 Bad Request
10.0156ms
"Invalid key:Session not found Please provide a valid, non-expired APIKEY."
PASS
```

6.2.2 T005 Verify API Expires After 5 Use

Date	2021-05-14 01:04:47 PM	
Test Case ID	T005	
Test Scenario	Verify API Expires After 5 Use is valid	
Test Steps	Using any REST client, execute the following actions or its equivalent, in the following sequence: getAPIKEY,getCourse,getCourse,getCourse,getCourse,getCourse,getCourse.	
Test Data	Refer to addcourse.req,updatecourse.req,deletecourse.req	
Expected Results	**Actual Results**	**Pass/Fail**
Verify the presence of the following message string. "Grant usage exceeded. Please renew APIKEY."	As Expected.	Pass



```
TESTING REPORT
Basic test of all the end points.
START:  2021-04-15 12:09:40.8980165 +0800 +08 m=+1.688994501
**********************************
FUNC :  getAPIKEY
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/getkey
RESP:
{   "APIKEY":"59825c12-90ea-4912-9362-0cb3d2b1c1a0","ClientIP":"172.17.0.1:60796",
    "Message":"APIKEY will expire after 5 mins of idle time; or after 5 use.",
    "ExpireTime":"2021-04-15T04:14:40.5414702Z"
}
-------------------------------
FUNC :  getCourse
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=59825c12-90ea-4912-9362-0cb3d2b1c1a0
PAYLOAD:

RESP:
[]
-------------------------------
FUNC :  getCourse
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=59825c12-90ea-4912-9362-0cb3d2b1c1a0
PAYLOAD:

RESP:
[]
-------------------------------
FUNC :  getCourse
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=59825c12-90ea-4912-9362-0cb3d2b1c1a0
PAYLOAD:

RESP:
[]
-------------------------------
FUNC :  getCourse
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=59825c12-90ea-4912-9362-0cb3d2b1c1a0
PAYLOAD:

RESP:
[]
-------------------------------
FUNC :  getCourse
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=59825c12-90ea-4912-9362-0cb3d2b1c1a0
PAYLOAD:

RESP:
[]
-------------------------------
FUNC :  getCourse
REQ:    GET https://127.0.0.1:5555/api/v1/goschool/course/YODA123?APIKEY=59825c12-90ea-4912-9362-0cb3d2b1c1a0
PAYLOAD:

RESP:
{"Message":"Grant usage exceeded. Please renew APIKEY."}
-------------------------------
END:    2021-04-15 12:09:41.0174853 +0800 +08 m=+1.808463301
**********************************
```



```
-------------------------------------------
TC-0004
GET https://192.168.0.162:5000/api/v1/goschool/course
Payload:  {"ID": "YODA123"}
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/course/YODA123?APIKEY=
400 Bad Request
66.1979ms
"Empty APIKEY. Please provide a valid key."
FAIL
-------------------------------------------
TC-0001
GET https://192.168.0.162:5000/api/v1/goschool/getkey
Payload:
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/getkey
-------------------------------------------
TC-0004
GET https://192.168.0.162:5000/api/v1/goschool/course
Payload:  {"ID": "YODA123"}
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/course/YODA123?APIKEY=9295ade6-e5ec-4f68-94d1-218c26a41ca9
200 OK
48.3844ms
[{"ID":"YODA123","Title":"GO GURU I","Description":"Learn Yoda's level technique in crafting Go program.","PreRequsites":null,"Trainers":null}]
PASS
-------------------------------------------
TC-0004
GET https://192.168.0.162:5000/api/v1/goschool/course
Payload:  {"ID": "YODA123"}
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/course/YODA123?APIKEY=9295ade6-e5ec-4f68-94d1-218c26a41ca9
200 OK
26.1683ms
[{"ID":"YODA123","Title":"GO GURU I","Description":"Learn Yoda's level technique in crafting Go program.","PreRequsites":null,"Trainers":null}]
PASS
-------------------------------------------
TC-0004
GET https://192.168.0.162:5000/api/v1/goschool/course
Payload:  {"ID": "YODA123"}
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/course/YODA123?APIKEY=9295ade6-e5ec-4f68-94d1-218c26a41ca9
200 OK
33.9461ms
[{"ID":"YODA123","Title":"GO GURU I","Description":"Learn Yoda's level technique in crafting Go program.","PreRequsites":null,"Trainers":null}]
PASS
-------------------------------------------
TC-0004
GET https://192.168.0.162:5000/api/v1/goschool/course
Payload:  {"ID": "YODA123"}
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/course/YODA123?APIKEY=9295ade6-e5ec-4f68-94d1-218c26a41ca9
200 OK
21.4389ms
[{"ID":"YODA123","Title":"GO GURU I","Description":"Learn Yoda's level technique in crafting Go program.","PreRequsites":null,"Trainers":null}]
PASS
-------------------------------------------
TC-0004
GET https://192.168.0.162:5000/api/v1/goschool/course
Payload:  {"ID": "YODA123"}
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/course/YODA123?APIKEY=9295ade6-e5ec-4f68-94d1-218c26a41ca9
200 OK
17.8723ms
[{"ID":"YODA123","Title":"GO GURU I","Description":"Learn Yoda's level technique in crafting Go program.","PreRequsites":null,"Trainers":null}]
PASS
-------------------------------------------
TC-0004
GET https://192.168.0.162:5000/api/v1/goschool/course
Payload:  {"ID": "YODA123"}
Actual URL:  https://192.168.0.162:5000/api/v1/goschool/course/YODA123?APIKEY=9295ade6-e5ec-4f68-94d1-218c26a41ca9
400 Bad Request
16.1404ms
{"Message":"Grant usage exceeded. Please renew APIKEY."}
PASS
```

7 SYSTEM DEPLOYMENT PLAN

7.1 Demo Day Staging

Because of need to have all the parts up and running throughout the duration till demo days, the system will be on ‘bare metal’, and not dockerized.

8 Product Limitations

This are some of the critical issues encountered that needs to be handled.

8.1 Requires Target Server To Follow Textbook REST API URL Conventions

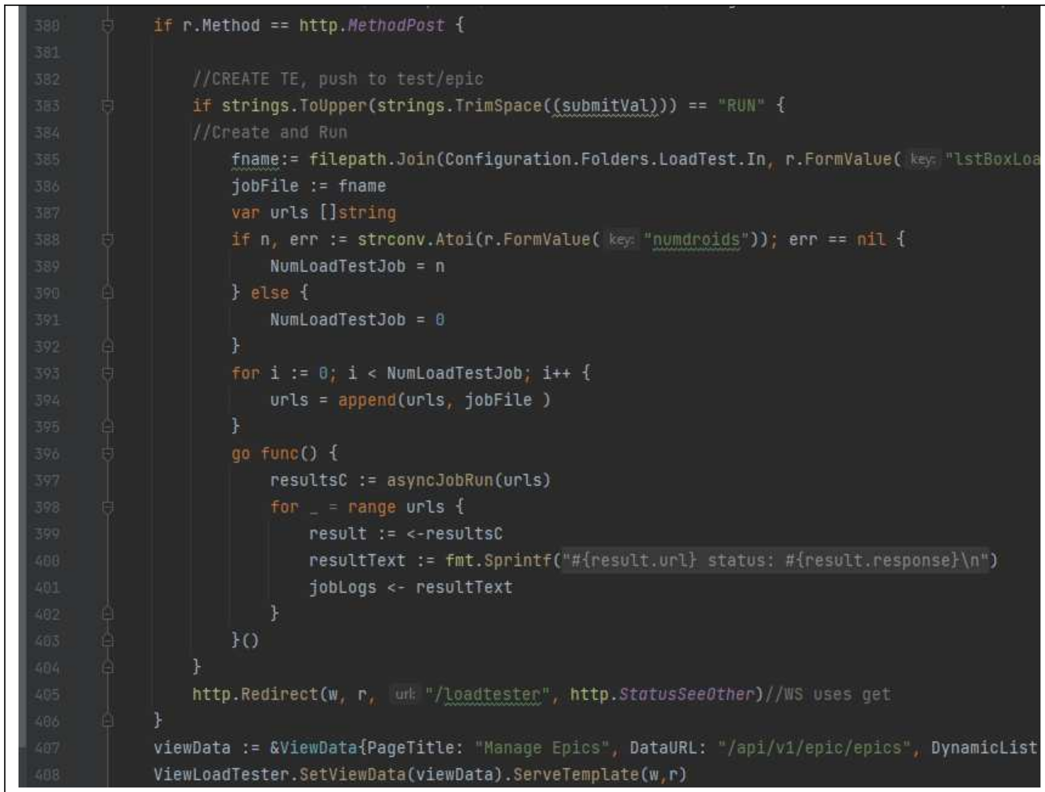
While doing this project, it is found to have at least 3 different ways to attach an APIKEY, and this creates complication as they need to be handled, and could break the execution if it is of an unknown convention

AsParam: https://192.168.0.162:5000/api/v1/goschool/course/YODA123?APIKEY=307d17c9-d778-4a51-b38c-414555160238
AsURL: https://mocki.io/v1/e750d778-4861-498e-b00e-213314f799dd
As Header: GET /something HTTP/1.1 X-API-Key: abcdef12345

8.2 WebSocket

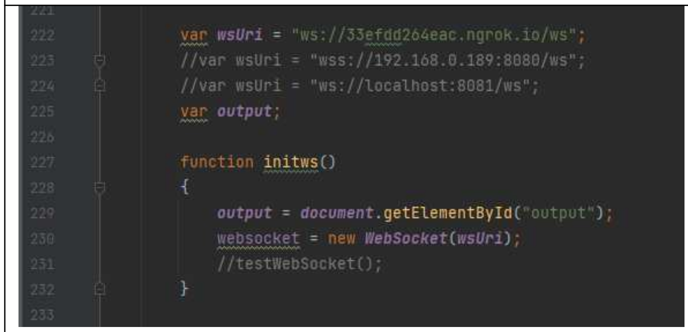
8.2.1 Conflict in Http Verbs

Using the built-in template, the web forms produced have actions of POST and GET. If we submit a form, it is a POST action for the handler, but this is a conflict with Websocket as it uses GET(after 1 day of debugging, first time trying websocket). A workaround of quickly returning the form, while injecting an ajax script solves this but it has side effects(such as the form may take too long to return and by then the socket is already closed). One alternative is to re-architech it such that there is a dedicated websocket server/service.



```go
if r.Method == http.MethodPost {

    //CREATE TE, push to test/epic
    if strings.ToUpper(strings.TrimSpace((submitVal))) == "RUN" {
    //Create and Run
        fname:= filepath.Join(Configuration.Folders.LoadTest.In, r.FormValue( key: "lstBoxLoa
        jobFile := fname
        var urls []string
        if n, err := strconv.Atoi(r.FormValue( key: "numdroids")); err == nil {
            NumLoadTestJob = n
        } else {
            NumLoadTestJob = 0
        }
        for i := 0; i < NumLoadTestJob; i++ {
            urls = append(urls, jobFile )
        }
        go func() {
            resultsC := asyncJobRun(urls)
            for _ = range urls {
                result := <-resultsC
                resultText := fmt.Sprintf("#{result.url} status: #{result.response}\n")
                jobLogs <- resultText
            }
        }()
    }
    http.Redirect(w, r, url: "/loadtester", http.StatusSeeOther)//WS uses get
}
viewData := &ViewData{PageTitle: "Manage Epics", DataURL: "/api/v1/epic/epics", DynamicList
ViewLoadTester.SetViewData(viewData).ServeTemplate(w,r)
```

Server.go



```javascript
    var wsUri = "ws://33efdd264eac.ngrok.io/ws";
    //var wsUri = "wss://192.168.0.189:8080/ws";
    //var wsUri = "ws://localhost:8081/ws";
    var output;

    function initws()
    {
        output = document.getElementById("output");
        websocket = new WebSocket(wsUri);
        //testWebSocket();
    }
```

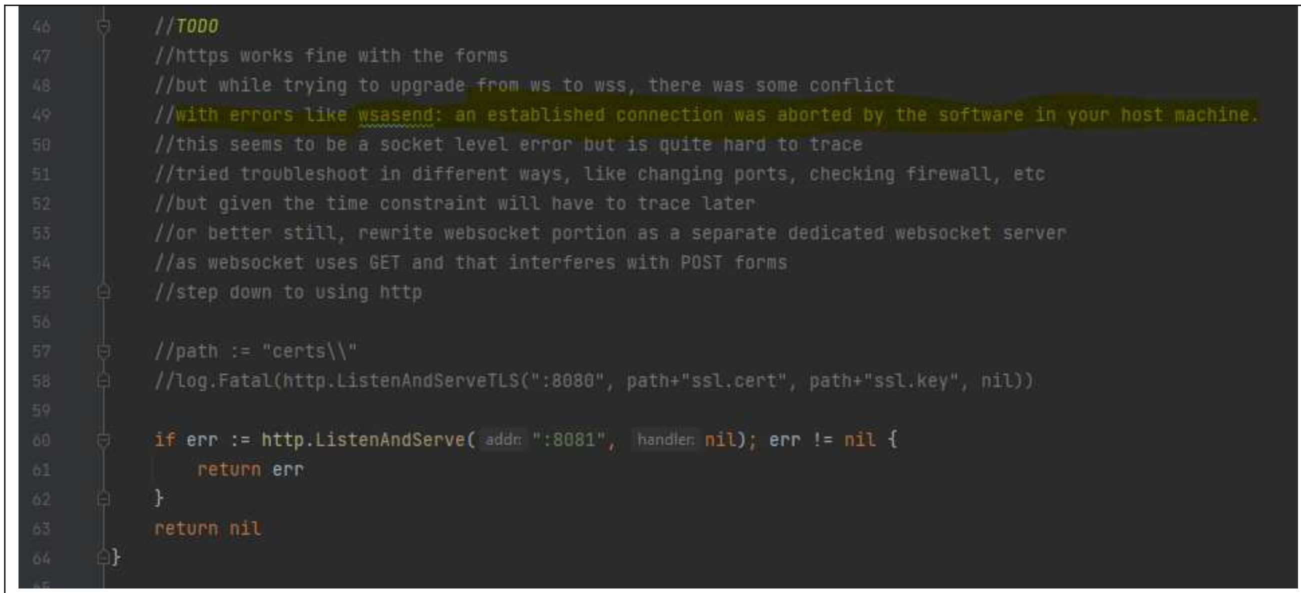
Loadtester.gohtml

8.2.2 Wsasend Socket Error

While trying to upgrade from ws to wss, the below error was encountered.

So again, it could be due to some Other force closing the socket which after investigation, and at this point, is not the top suspect. It could possibly be a POST and GET conflict somewhere deep in the way the calls are handled by the standard library, or it could be a coding bug of mine.

So again one alternative is to re-architech it such that there is a dedicated websocket server/service.



```go
	//TODO
	//https works fine with the forms
	//but while trying to upgrade from ws to wss, there was some conflict
	//with errors like wsasend: an established connection was aborted by the software in your host machine.
	//this seems to be a socket level error but is quite hard to trace
	//tried troubleshoot in different ways, like changing ports, checking firewall, etc
	//but given the time constraint will have to trace later
	//or better still, rewrite websocket portion as a separate dedicated websocket server
	//as websocket uses GET and that interferes with POST forms
	//step down to using http

	//path := "certs\\"
	//log.Fatal(http.ListenAndServeTLS(":8080", path+"ssl.cert", path+"ssl.key", nil))

	if err := http.ListenAndServe( addr: ":8081",  handler: nil); err != nil {
		return err
	}
	return nil
}
```

Server.go

9 Product Roadmap

This morning was basically a technology preview of the system. To go to market, or to be a viable product that is accepted by Users, much more work needs to be done. Besides the functional enhancement, security features can be added in accordance to a roadmap.

The below is documented to allow further future research.

9.1 UI

Not just being cosmetical, a well-designed UI improves user productivity and reduces mistakes.

9.1.1 Colors

From a pre-Alpha release, the colors are good enough, but a CSS or design specialist will be needed if it is to go to market.

9.1.2 Process Flows

The flow is generally acceptable now as it is quite intuitive, and the usage is quite standardised across the screens, such that if a user understands one screen, the rest are similar. But with additions or enhancements, and with lessons learned, a further sweep can be done to see if it can be made smoother, or better.

9.1.3 Templating

Generally, if the business domain is fixed as per an organization, Test Cases can be saved and reused through templating. Such exemplary test cases can be copied, and details amended as per other relevant scenarios. Through templating, and reuse the Organization can build up Knowledge as well as Standards.

9.2 Response Parsing

Currently, only a simple parsing is done on the response status, and body.

9.2.1 Regex/Custom Func

The system can be greatly enhanced if the response can be passed to some custom callback function or Regex to do deeper parsing. For most normal use case, this is not really necessary, but it is a nice to have feature in terms of marketing the product.

To achieve this, there are several ways but one could be the use of GopherJS which is tightly integrated with Golang. Another idea is to pass the response body to a ‘Parsing-As-A-Service’ microservice.

9.3 Message Format

It is understood that in order that the system can be long-lived, it has to have a standard, or universal or extensible message format. I would argue that this is perhaps even more important than the core execution engine. With a universal and extensible message(job,test defintion) new changes can be added by changing the format, instead of constantly changing the engine.

9.3.1 Universal/Extensible

A mental model would be to start to think in terms of XML, FIX, etc as an example, but not in terms of their verbosity. Eventually, in a distributed setup of ‘Test Droids’, the system as a whole is basically exchanging messages.

9.4 Language

The project has some words for actions such as Wait(Duration) that is different from the CRUD actions. It is considering whether to lift such actions out and put them into a different category.

9.5 Entreprise Versioning

The version shown is more of a desktop, personal use version. To cater for eventual entreprise version, the following can be done.

9.5.1 Orgs,Teams,Projects,Users

The system will have to consider how to incorporate functions to cater to the Organization, Teams such as Dev or QA, who may have 1 to many projects, and the different classes of Users.

9.6 Lifecycle Integration

A testing tool is a companion to development, and as such it needs to be in sync with their cycles.

9.6.1 SDLC

Developers may interface with this system at various point of their development cycle. The system can for instance help provide Mocking Services to simulate the final API with Dummy data, etc.

9.6.2 QA

For a Teams view, it could perhaps be an important feature that at Project Inception, QAs can locked in skeletal test cases(with perhaps a title and some header details), and work out the details as the project moves along. By such, the visibility of having these test cases can be helpful to both Dev and QA, and allow better alignment of requirements amongst all stakeholders.

9.7 Distributed Model

This morning, the system shown a Load Test using a 100 go-routines. However, even if we run 10,000 go-routines, the test effect is not the same as perhaps running 10 Droids with 1000 goroutines each.

9.7.1 Actor Model

The Distributed Droid Model, using the Actor Model is to scale to a true concurrent testing tool It will also have benefit of being de-coupled, and be more resilent. This should be the final API-T-Droid.

9.8 Platforms

Though this may remain purely an API testing tool, but it would be good to allow options to integrate testing to the web or mobile front. In that situation, it would be a truly end-to-end testing tool.

9.8.1 Web

Many web fronts run ajax or websockets calls to backend APIs, and for the remaining web forms applications, Selenium for example can be used to simulate clicks. Another way is to perhaps capture or intercept http calls through for instance Fiddler to achieve stimulated front end actions for testing.

9.8.2 Mobile

From what I know, a lot of mobile apps are acutally calling backend APIs, so it means it is possible to integrate to a full end-to-end testing tool.

10 Data Dictionary

10.1 Courses

This is defined in Model/Course/Course.go

ID	Type	Description
ID	string	Primary Key. The ID for the course.
Title	string	Required. The course title.
Description	string	A short summary of the course.

10.2 Trainer

This is defined in Model/Course/Course.go

ID	Type	Description
ID	string	Primary Key. The Employee ID of the trainer.
FirstName	string	First name of trainer.
LasttName	string	Last name of trainer.
Age	int	Trainer’s age.

11 REFERENCES

1. Introduction to API Testing https://docs.katalon.com/katalon-studio/docs/introduction_api_testing.html

2. Global API Testing Market to Grow to Over $1 Billion by 2022. https://www.devopsdigest.com/global-api-testing-market-to-grow-to-over-1-billion-by-2022

3. STATE OF TESTING REPORT 2020. https://www.practitest.com/assets/pdf/PT_repo4_20-20.pdf

4.

5. 17 Best Functional & API Automation Testing Tools In 2021 https://theqalead.com/tools/automation-testing-tools/

6. Top 10 API Testing Tools in 2020. https://medium.com/@alicealdaine/top-10-api-testing-tools-rest-soap-services-5395cb03cfa9

7. The 10 API Testing Tools You Can't Live Without in 2021 https://www.testim.io/blog/the-9-api-testing-tools-you-cant-live-without-in-2019-2/

8. What Are Automation Testing Tools? 9 Types & Examples https://theqalead.com/topics/what-are-automation-testing-tools/

9. Top 3 API Testing Tool Comparison https://huddle.eurostarsoftwaretesting.com/soapui-vs-postman-katalon-studio-top-3-api-testing-tool-comparison/

10. Introducing Redpoint's Software Testing Landscape https://medium.com/memory-leak/introducing-redpoints-software-testing-landscape-3c5615f7eeae

11. Global Software Testing Services Market 2019-2023 | 12% CAGR Projection Over the Next Five Years. https://www.businesswire.com/news/home/20191115005456/en/Global-Software-Testing-Services-Market-2019-2023-12-CAGR-Projection-Over-the-Next-Five-Years-Technavio

12. Why IT Companies should switch to Automation Testing? https://www.appventurez.com/blog/why-should-companies-switch-to-automation-testing/

12 APPENDIX

This section covers any other miscellaneous addendums.

12.1 Client Test Data

Test data(request json payload) used.

getcourse.req:

```
{
        "ID": "YODA123",
}
```

Addcourse.req:

```
{
"ID": "YODA123",
"Title": "GO GURU I",
"Description": "Learn guru's level technique of crafting Go's program."
}
```

Updatecourse.req:

```
{
        "ID": "YODA123",
        "Title": "GO GURU I",
        "Description": "Learn Yoda's level technique in crafting Go program."
}
```

deletecourse.req:

```
{
        "ID": "YODA123",
}
```

12.2 User Guide

User Guide.html is provided as a separate file for user convenience.

The html page embeds notes and demo videos(animated gifs).

12.3 Test Guide

Test Guide.html is provided as a separate file for user convenience.

The html page embeds notes and demo videos(animated gifs) to demonstrate the steps on how to test the application as well as some recordings of previous tests.