Utkarsh

Task : Instagram backend api

**Descriptions:**

Models package: contains the structures of the user and post

User.go : contains the structure of users

Post.go : contains the structure of post

Controllers package: contains all the required functions on users and posts

User.go : contains all the functions for users

* Create a user
* Get the user using id
* Get all the users posts

post.go : contains all the functions for posts

* Create a post
* Get the post using id

**Codes**

main.go

*package* main

*import* (

"github.com/julienschmidt/httprouter"

"gopkg.in/mgo.v2"

"net/http"

"github.com/utkarsh1025/golang/controllers"

)

*func* *main*(){

    r := httprouter.New()

    uc := controllers.NewUserController(getSession())

    pc := controllers.NewPostController(getSession())

    r.GET("/post/users/:id",uc.GetUsersPosts)

    r.GET("/users/:id", uc.GetUser)

    r.POST("/users", uc.CreateUser)

    r.GET("/posts/:id", pc.GetPost)

    r.POST("/posts", pc.CreatePost)

    http.ListenAndServe("localhost:3333", r)

}

*func* *getSession*() \*mgo.Session{

    s, err := mgo.Dial("mongodb://localhost:27017")

*if* err != nil{

        panic(err)

    }

*return* s

}

go.mod

module github.com/utkarsh1025/golang

go 1.17

require (

    github.com/julienschmidt/httprouter v1.3.0 // *indirect*

    gopkg.in/mgo.v2 v2.0.0-20190816093944-a6b53ec6cb22 // *indirect*

)

go.sum

github.com/julienschmidt/httprouter v1.3.0 h1:U0609e9tgbseu3rBINet9P48AI/D3oJs4dN7jwJOQ1U=

github.com/julienschmidt/httprouter v1.3.0/go.mod h1:JR6WtHb+2LUe8TCKY3cZOxFyyO8IZAc4RVcycCCAKdM=

gopkg.in/mgo.v2 v2.0.0-20190816093944-a6b53ec6cb22 h1:VpOs+IwYnYBaFnrNAeB8UUWtL3vEUnzSCL1nVjPhqrw=

gopkg.in/mgo.v2 v2.0.0-20190816093944-a6b53ec6cb22/go.mod h1:yeKp02qBN3iKW1OzL3MGk2IdtZzaj7SFntXj72NppTA=

Models package-

User.go

*package* models

*import* "gopkg.in/mgo.v2/bson"

*type* User *struct* {

    Id       bson.ObjectId `json:"id" bson:"\_id"`

    Name     string        `json:"name" bson:"name"`

    Password string        `json:"password" bson:"password"`

    Email    string        `json:"email" bson:"email"`

}

Post.go

*package* models

*import* "gopkg.in/mgo.v2/bson"

*type* Post *struct* {

    Id       bson.ObjectId       `json:"id" bson:"\_id"`

    Caption  string              `json:"caption" bson:"caption"`

    Imageurl string              `json:"imageurl" bson:"imageurl"`

    Time     bson.MongoTimestamp        `json:"time" bson:"time"`

    Userid   string     `json:"userid" bson:"userid"`

}

controllers package-

User.go

*package* controllers

*import* (

    "encoding/json"

    "fmt"

    "net/http"

    "github.com/julienschmidt/httprouter"

    "github.com/utkarsh1025/golang/models"

    "gopkg.in/mgo.v2"

    "gopkg.in/mgo.v2/bson"

)

*type* UserController *struct* {

    session \*mgo.Session

}

*func* *NewUserController*(s \*mgo.Session) \*UserController {

*return* &UserController{s}

}

*func* (uc UserController) *GetUser*(w http.ResponseWriter, r \*http.Request, p httprouter.Params) {

    id := p.ByName("id")

*if* !bson.IsObjectIdHex(id) {

        w.WriteHeader(http.StatusNotFound)

    }

    oid := bson.ObjectIdHex(id)

    u := models.User{}

*if* err := uc.session.DB("mongo-golang").C("users").FindId(oid).One(&u); err != nil {

        w.WriteHeader(404)

*return*

    }

    x := u

    x.Password = "\*\*Hidden\*\*"

    uj, err := json.Marshal(x)

*if* err != nil {

        fmt.Println(err)

    }

    w.Header().Set("Content-Type", "application/json")

    w.WriteHeader(http.StatusOK)

    fmt.Fprintf(w, "%s\n", uj)

}

*func* (uc UserController) *CreateUser*(w http.ResponseWriter, r \*http.Request, \_ httprouter.Params) {

    u := models.User{}

    json.NewDecoder(r.Body).Decode(&u)

    u.Id = bson.NewObjectId()

    uc.session.DB("mongo-golang").C("users").Insert(u)

    uj, err := json.Marshal(u)

*if* err != nil {

        fmt.Println(err)

    }

    w.Header().Set("Content-Type", "application/json")

    w.WriteHeader(http.StatusCreated)

    fmt.Fprintf(w, "%s\n", uj)

}

*func* (uc UserController) *GetUsersPosts*(w http.ResponseWriter, r \*http.Request, p httprouter.Params) {

    id := p.ByName("id")

    ux := uc.session.DB("mongo-golang").C("Posts").Find(bson.M{"userid": id})

*var* episodesFiltered []bson.M

    ux.All(&episodesFiltered)

    uj, err := json.Marshal(episodesFiltered)

*if* err != nil {

        fmt.Println(err)

    }

    w.Header().Set("Content-Type", "application/json")

    w.WriteHeader(http.StatusOK)

    fmt.Fprintf(w, "%s\n", uj)

}

Post.go

*package* controllers

*import* (

    "encoding/json"

    "fmt"

    "net/http"

    "github.com/julienschmidt/httprouter"

    "github.com/utkarsh1025/golang/models"

    "gopkg.in/mgo.v2"

    "gopkg.in/mgo.v2/bson"

)

*type* PostController *struct* {

    session \*mgo.Session

}

*func* *NewPostController*(s \*mgo.Session) \*PostController {

*return* &PostController{s}

}

*func* (uc PostController) *GetPost*(w http.ResponseWriter, r \*http.Request, p httprouter.Params) {

    id := p.ByName("id")

*if* !bson.IsObjectIdHex(id) {

        w.WriteHeader(http.StatusNotFound)

    }

    oid := bson.ObjectIdHex(id)

    u := models.Post{}

*if* err := uc.session.DB("mongo-golang").C("Posts").FindId(oid).One(&u); err != nil {

        w.WriteHeader(404)

*return*

    }

    uj, err := json.Marshal(u)

*if* err != nil {

        fmt.Println(err)

    }

    w.Header().Set("Content-Type", "application/json")

    w.WriteHeader(http.StatusOK)

    fmt.Fprintf(w, "%s\n", uj)

}

*func* (uc PostController) *CreatePost*(w http.ResponseWriter, r \*http.Request, \_ httprouter.Params) {

    u := models.Post{}

    json.NewDecoder(r.Body).Decode(&u)

    u.Id = bson.NewObjectId()

    uc.session.DB("mongo-golang").C("Posts").Insert(u)

    uj, err := json.Marshal(u)

*if* err != nil {

        fmt.Println(err)

    }

    w.Header().Set("Content-Type", "application/json")

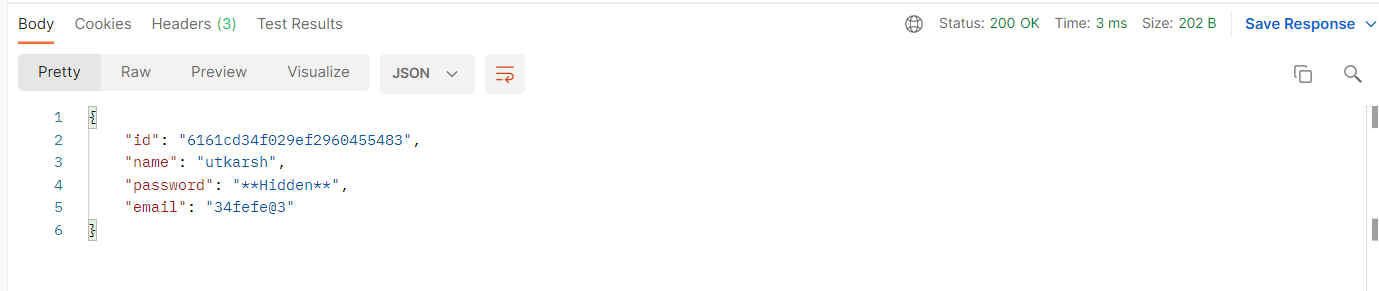
    w.WriteHeader(http.StatusCreated)

    fmt.Fprintf(w, "%s\n", uj)

}

**CONSTRAINTS:**

* Passwords should be securely stored such they can't be reverse engineered

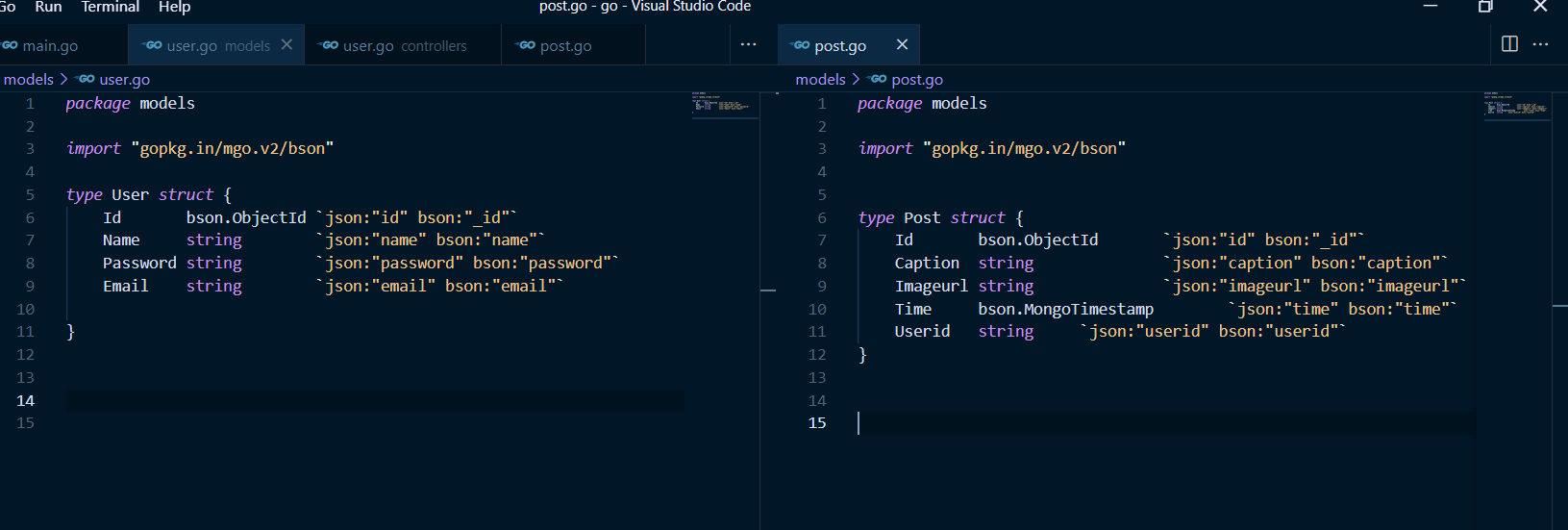


Users should have the following attributes

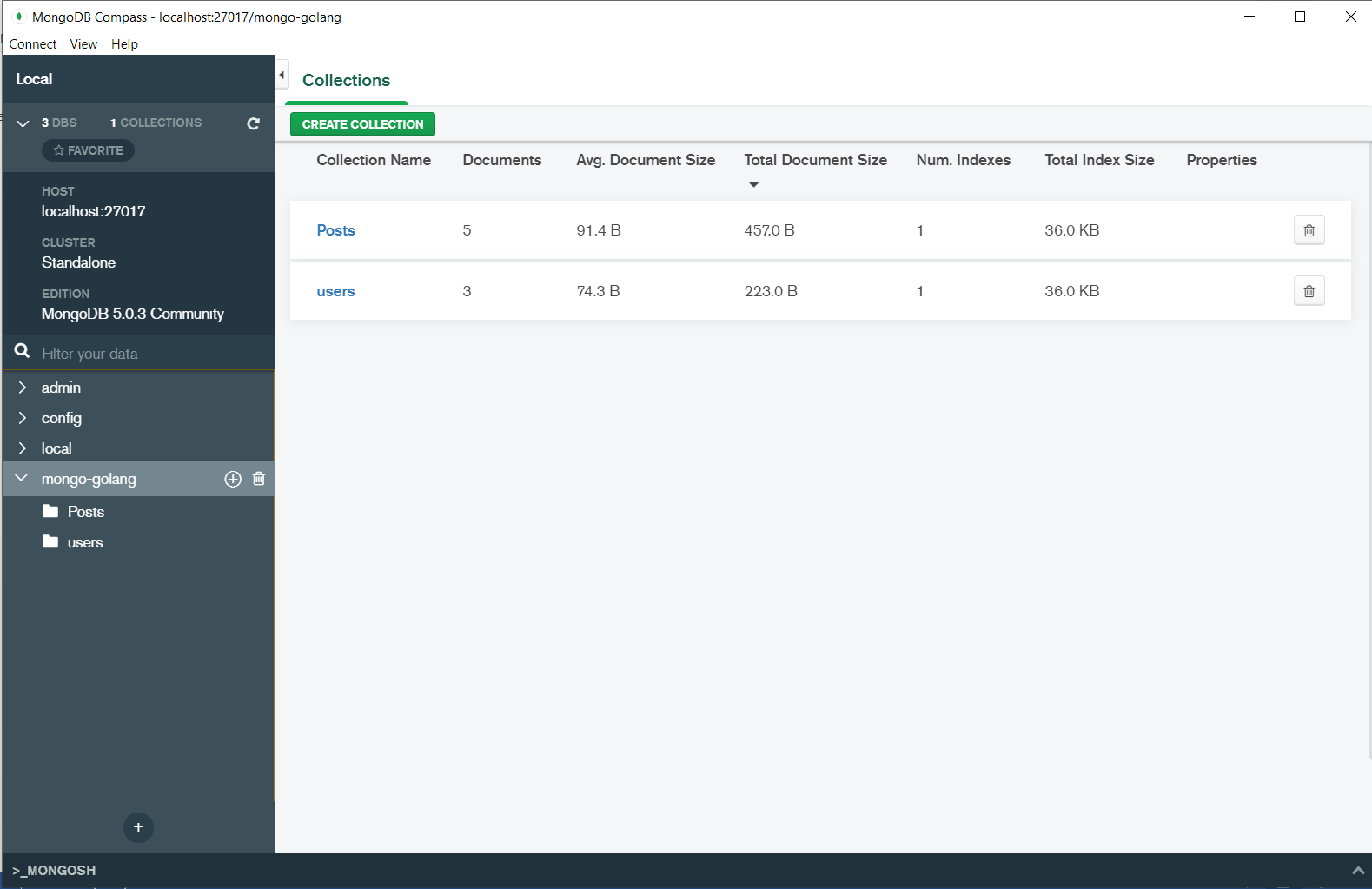
* Id
* Name
* Email
* Password

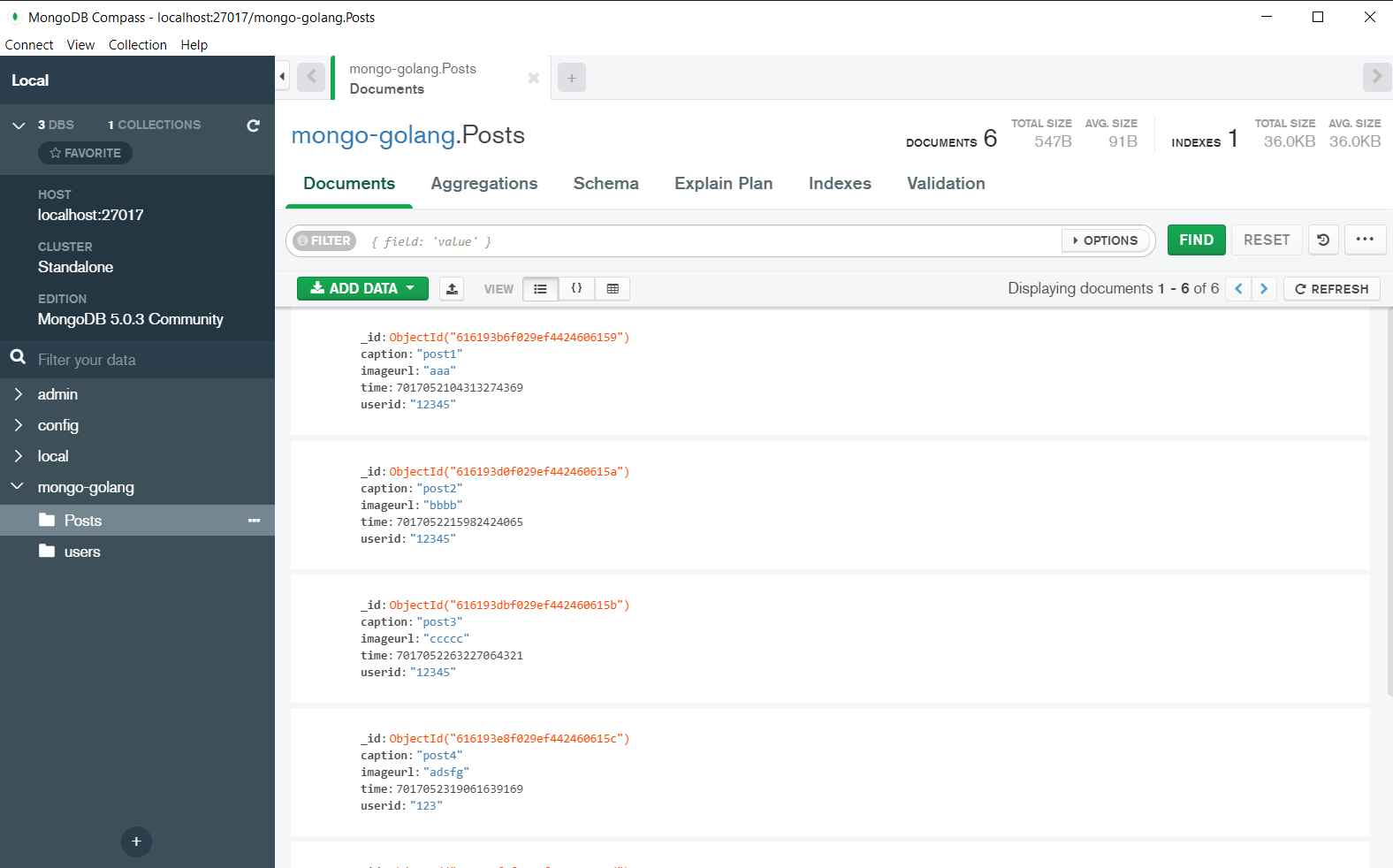
Posts should have the following Attributes. All fields are mandatory unless marked optional:

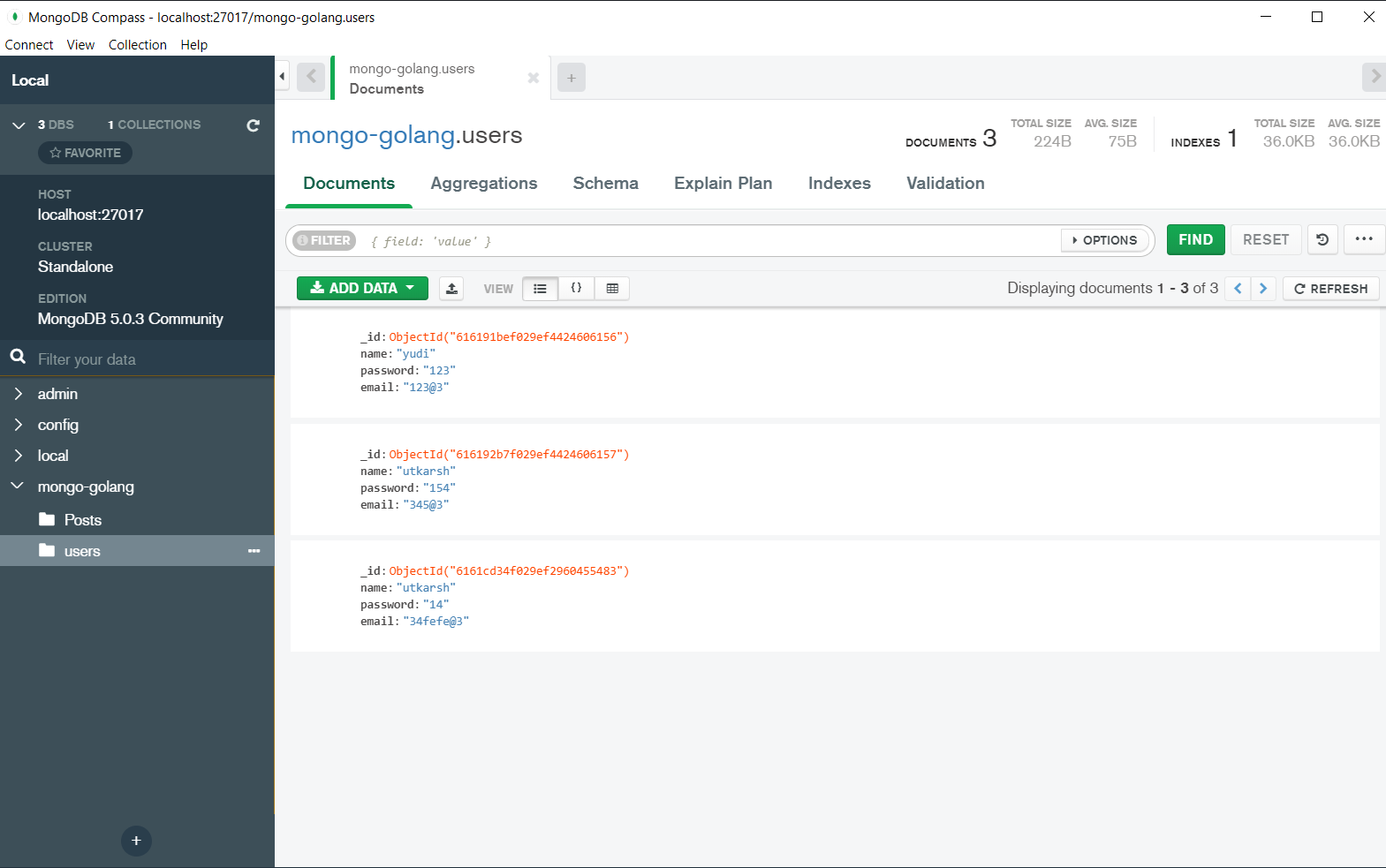
* Id
* Caption
* Image URL
* Posted Timestamp



**MONGODB :**

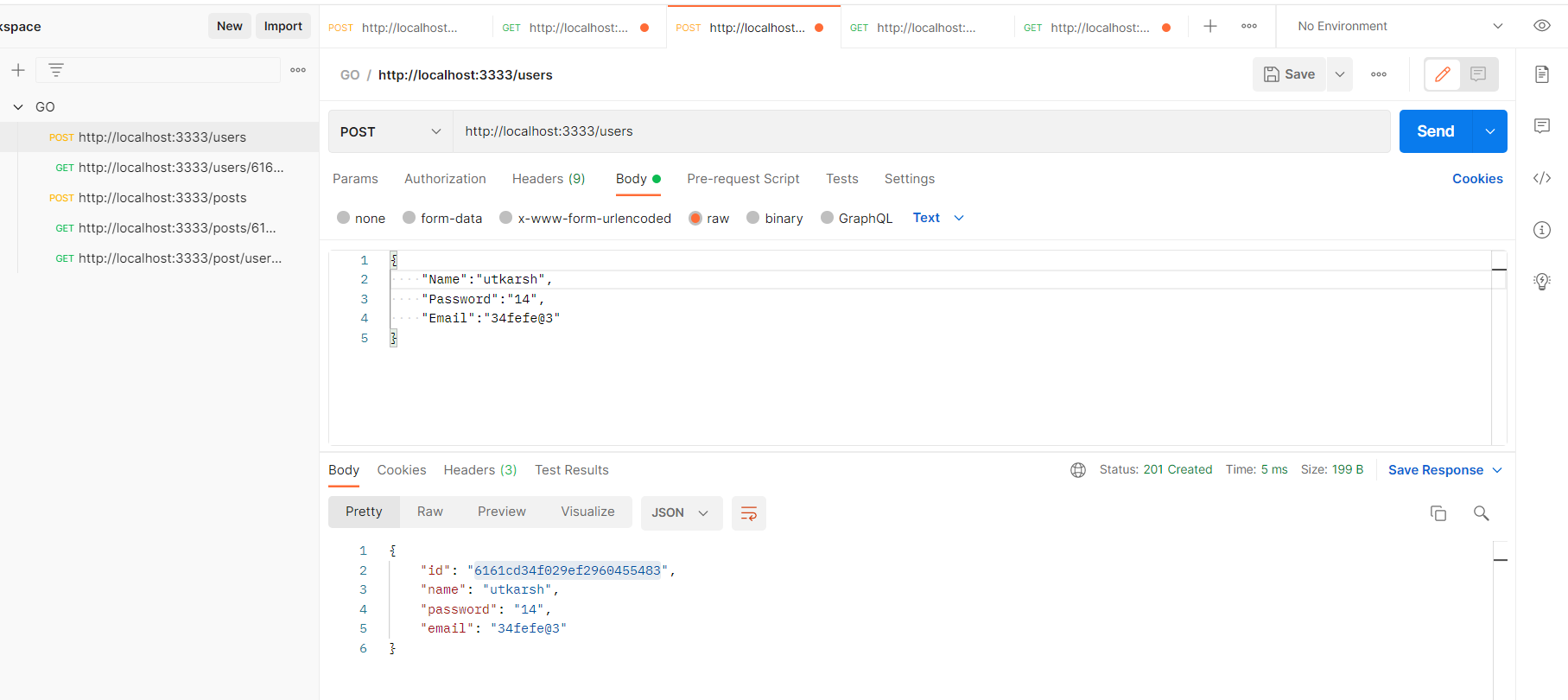




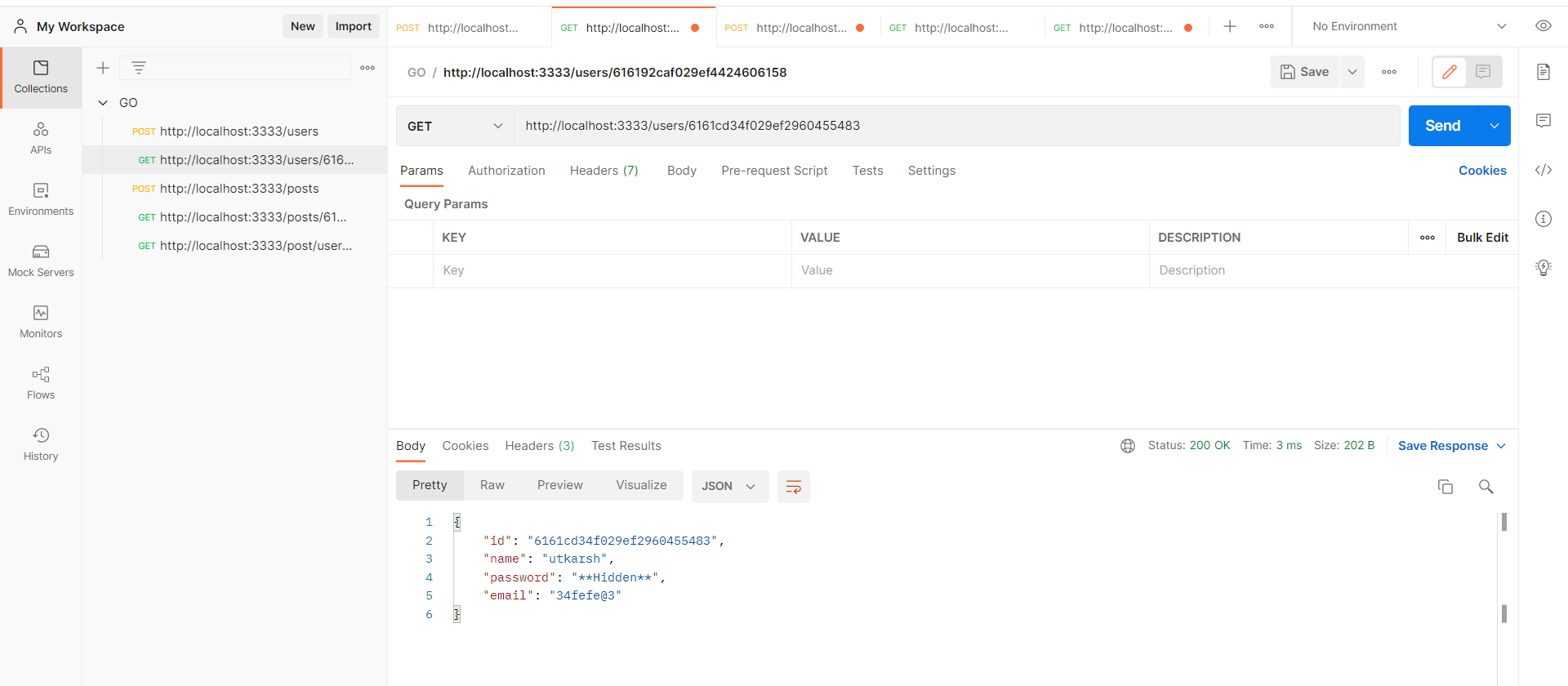


**TESTING and CONFIRMATION :**

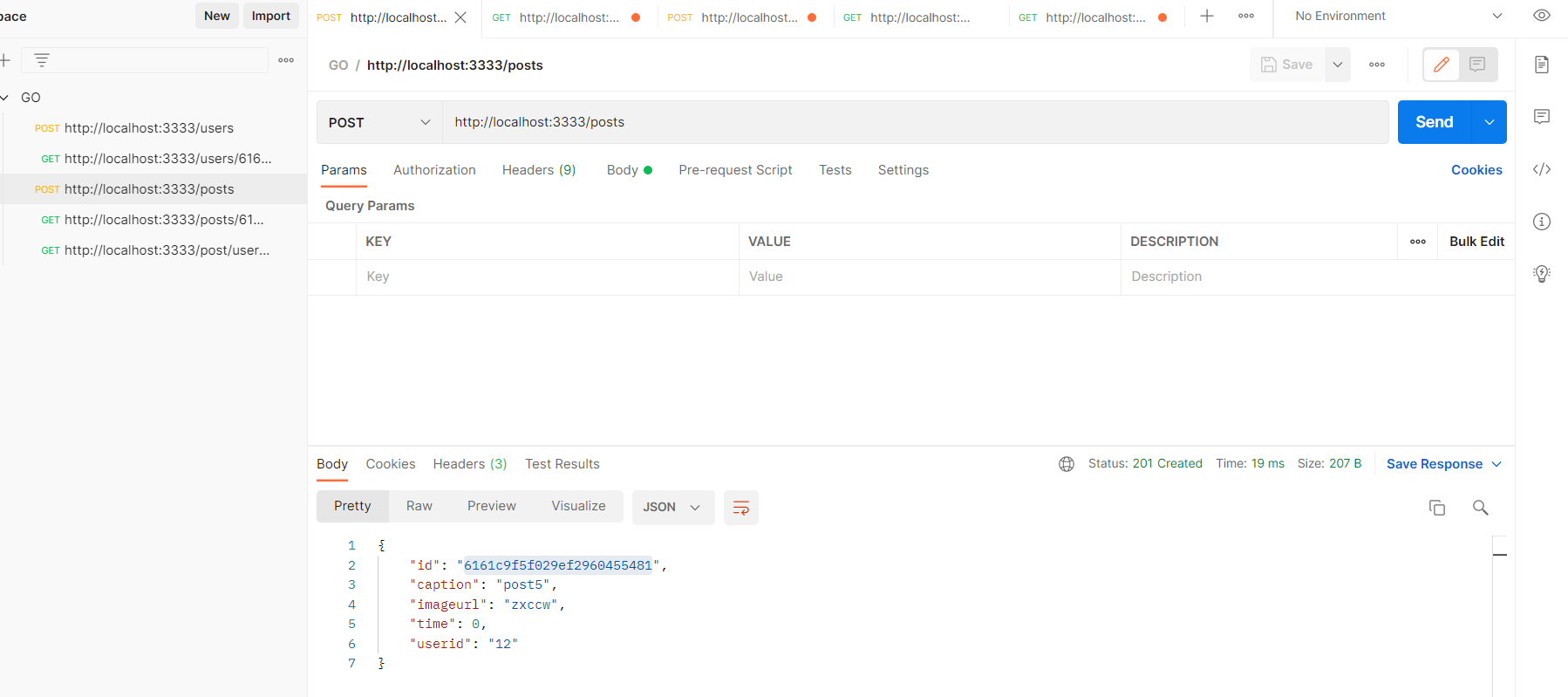
* Create an User
  + Should be a POST request
  + Use JSON request body
  + URL should be ‘/users'



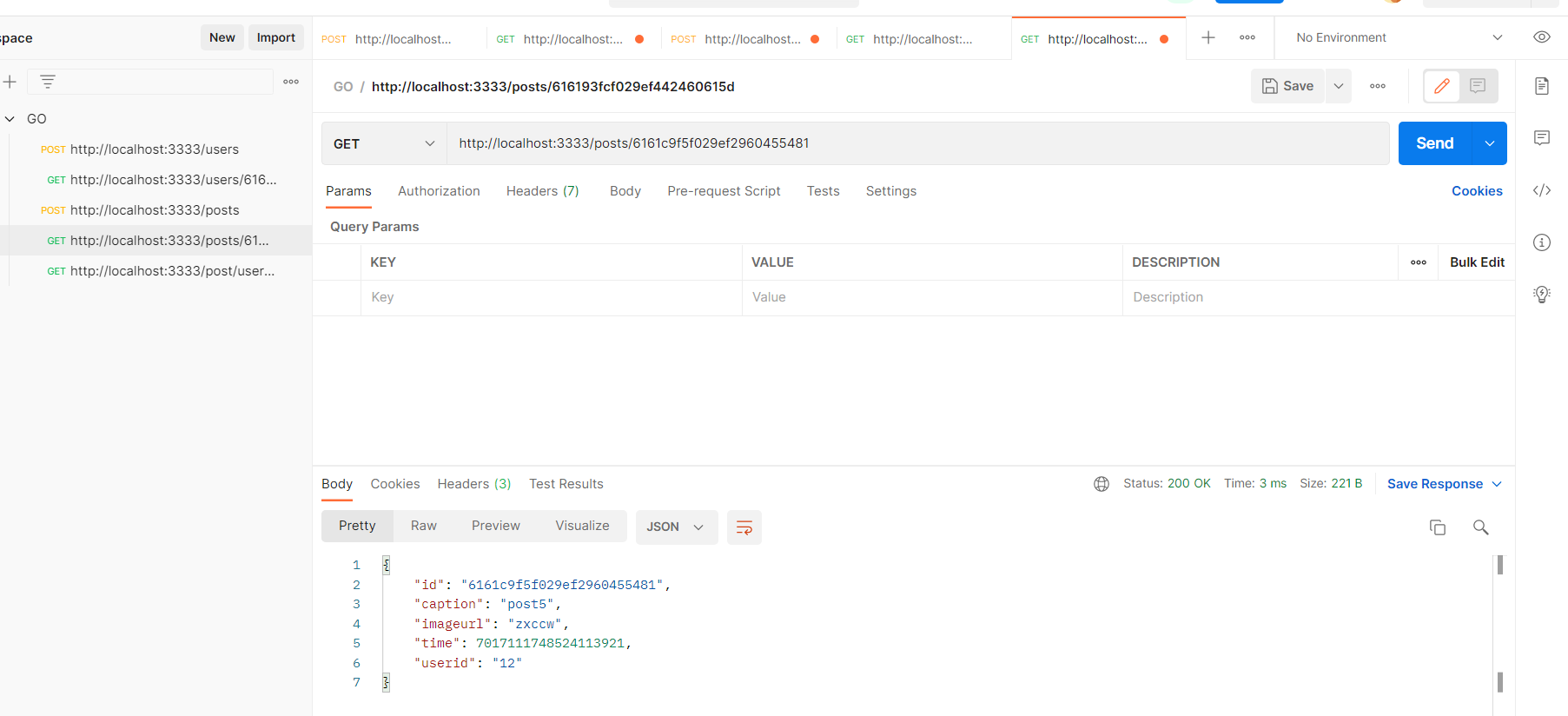
* Get a user using id
  + Should be a GET request
  + Id should be in the url parameter
  + URL should be ‘/users/<id here>’



* Create a Post
  + Should be a POST request
  + Use JSON request body
  + URL should be ‘/posts'



* Get a post using id
  + Should be a GET request
  + Id should be in the url parameter
  + URL should be ‘/posts/<id here>’



* List all posts of a user
  + Should be a GET request
  + URL should be ‘/posts/users/<Id here>'

