Multimidia Youtube

1. Introduction

•A brief introduction to your project and the expected features:

My project is a multimedia system, in which it is allowed to post youtube links. Once posted a link, some logged in user can comment on the videos. Any user can view the videos posted.

2. Design and Implementation

2.1 The REST API Specification

•Give the details of your REST API, the various REST API end points and the operations to be supported on these end points:

// ************** START routes **********************

//isloged

```
app.use('/isloged',require('./server/routes/isloged.js'))

//logout
app.use('/logout',require('./server/routes/logout.js'))

//signup
app.use('/signup',require('./server/routes/signup.js'))

//login
app.use('/login',require('./server/routes/login.js'))

//video
app.use('/api_videos',require('./server/routes/videos.js'))

//error messages
app.use('/error',require('./server/routes/error.js'))



//sucess messages
app.use('/sucess',require('./server/routes/sucess.js'))

// **************** END routes **********************
```

//error router

```
router.get('/signup', function(req,res,next){
   res.json({error:'signup'})
} )

router.get('/login', function(req,res,next){
   res.json({error:'login'})
} )
```

//islogged

```
router.get('/' , isLogged , function(req, res , next){
   res.json([{message: true , user: req.user}])
}); //check if user is logged
```

//loggin router

```
router.post('/', passport.authenticate('local-login', {
      successRedirect : '/sucess/login', // redirect to the secure profile section
      failureRedirect : '/error/login', // redirect back to the signup page if there is an error
      failureFlash : true // allow flash messages
   })
)
```

//logout router

```
router.get('/', function(req, res) {
   req.logout();
   res.json([{message:'sucess'}]);
});
```

//signup

```
router.post('/', passport.authenticate('local-signup' ,
   {
      successRedirect : '/sucess/signup', // redirect to the secure profile section
      failureRedirect : '/error/signup', // redirect back to the signup page if there is an error
      failureFlash : true // allow flash messages
   })
)
```

//sucess

```
router.get('/signup', function(req,res,next){
   res.json({sucess:'signup'})
} )
```

```
router.get('/login', function(req,res,next){
    res.json({sucess:'login'})
} )
```

//the video and comment router:

```
//get all
router.get('/', function(req, res , next) {
    Video.find({}, function(err, videos) {
        if (!err){
            res.json(videos);
        }
        else {next(err)}
    })
    .populate('_author') //populate creator with user info
    .exec(function (err, video) {
        if (err) return handleError(err);
            console.log('The _author is %s', video._author);
            // prints "The creator is Aaron"
        }
    );
});

//get specific video
router.get('/:id', function(req, res , next) {
    var id = req.params.id;

    Video.findOne({_id:id}, function(err, video) {
        if (!err){
            res.json([video]);
        }
        else {next(err)}
    })
    .populate('_author') //populate creator with user info
    .exec(function (err, video) {
        if (err) return handleError(err);
            console.log('The _author is %s', video._author);
            // prints "The creator is Aaron"
        }
    );

});
```

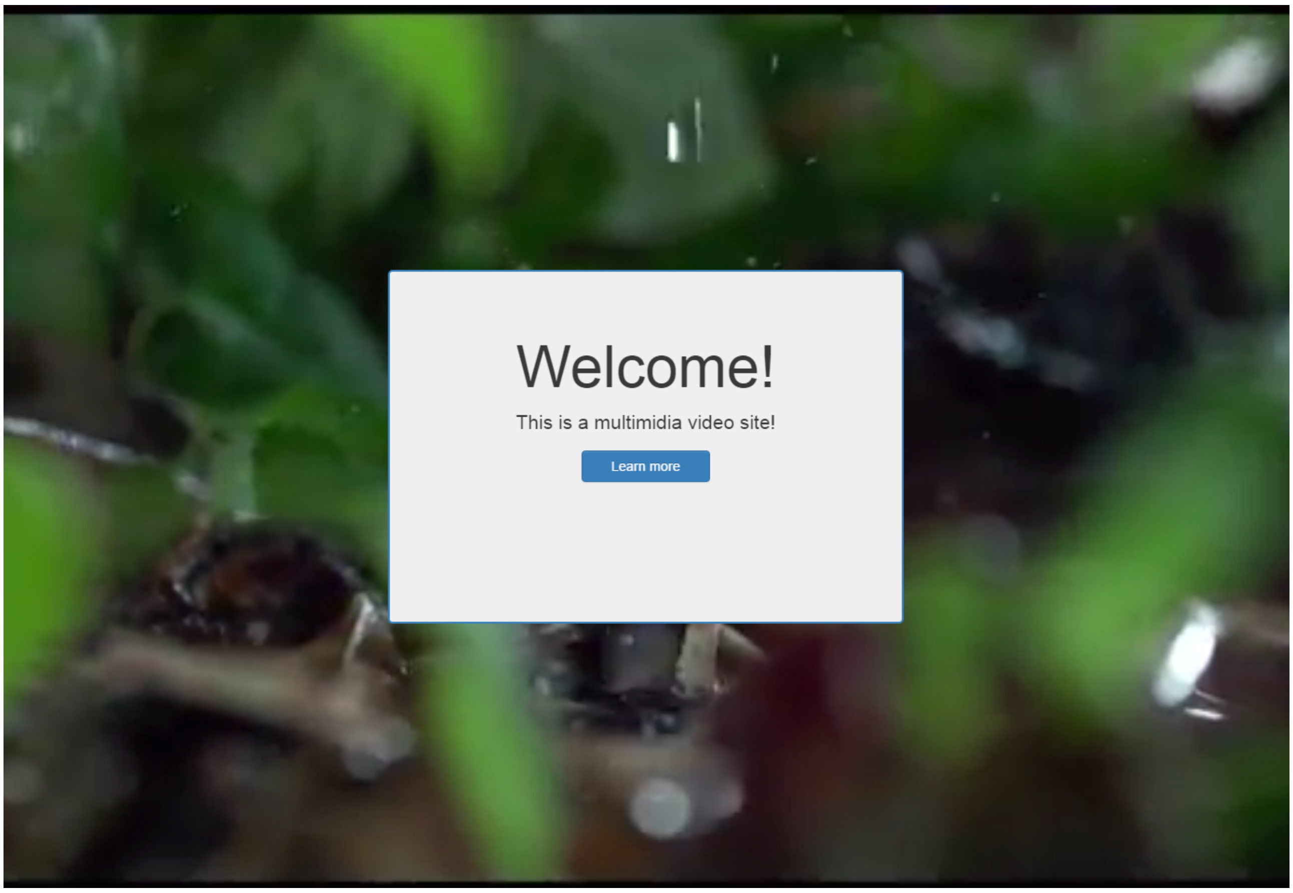
```
//get comments
router.get('/:idVideo/comments', function(req, res , next) {
    var id = req.params.idVideo;

    Comment.find({_video:id}, function(err, comments) {
        if (!err){
            res.json(comments);
        }
        else {next(err)}
    })
    .populate('_creator') //populate creator with user info
    .exec(function (err, comment) {
        if (err) return handleError(err);
            console.log('The creator is %s', comment._creator);
            // prints "The creator is Aaron"
        });
    });

//post comments
router.post('/:idVideo/comments', function(req, res , next) {
    var id = req.params.idVideo;
    if (!req.body) return res.sendStatus(400);

    Video.findOne({_id:id}, function(err, video) {

        if(err) next(err);

        var CommentInstance = new Comment(req.body);

        CommentInstance.save(function(err, comment) {
            if(err) next(err);

            res.json(comment);
        });

    });


});
```

```
//delete comments
router.delete('/:idVideo/comments/:idComment', function(req, res , next) {
    var idComment = req.params.idComment;

    Comment.remove({_id: idComment} , function(err){
        if(err) res.json({message: "Error: " + err})

        res.json({message: "sucess"})
    })

});



//update comments
router.put('/:idVideo/comments/:idComment', function(req, res , next) {
    var idComment = req.params.idComment;

    Comment.findOneAndUpdate({_id:idComment}, req.body, function (err, comment) {
        if(err) res.json({message: "Error: " + err});

        res.send({message: "sucess"});
    });

});

//save new videos if is logged
router.post('/', isloged ,function(req, res , next) {
    if (!req.body) return res.sendStatus(400)

    var VideoInstance = new Video(req.body);

    VideoInstance.save(function(err,video){
        if(err) next(err);

        res.json(video);
    });
});
```

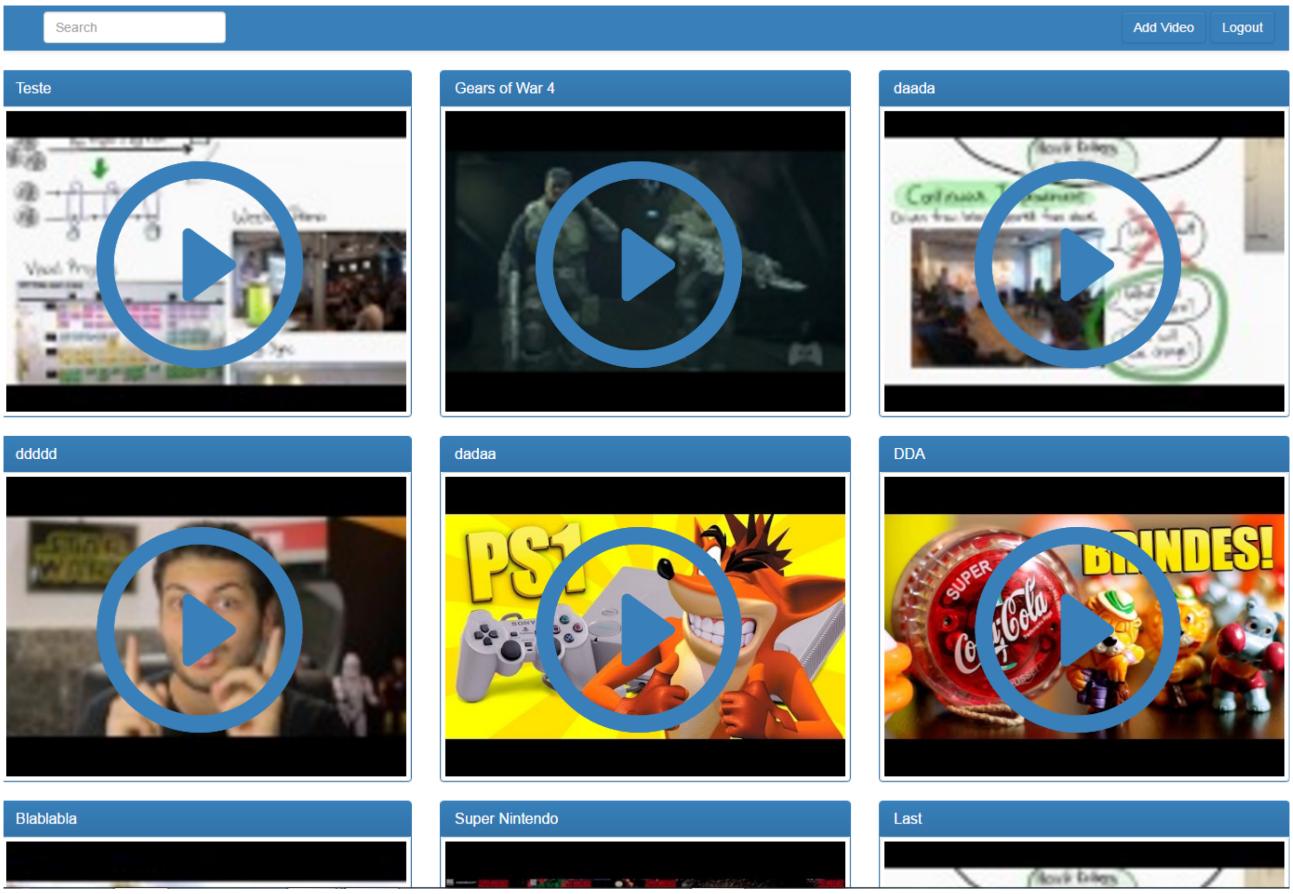
2.2 Front-end Architecture Design

•Give some details of the architecture and structure of your front-end, both web application and hybrid mobile application, in a format that you consider suitable, You may choose to use any formal languages or structure diagrams to express the details.

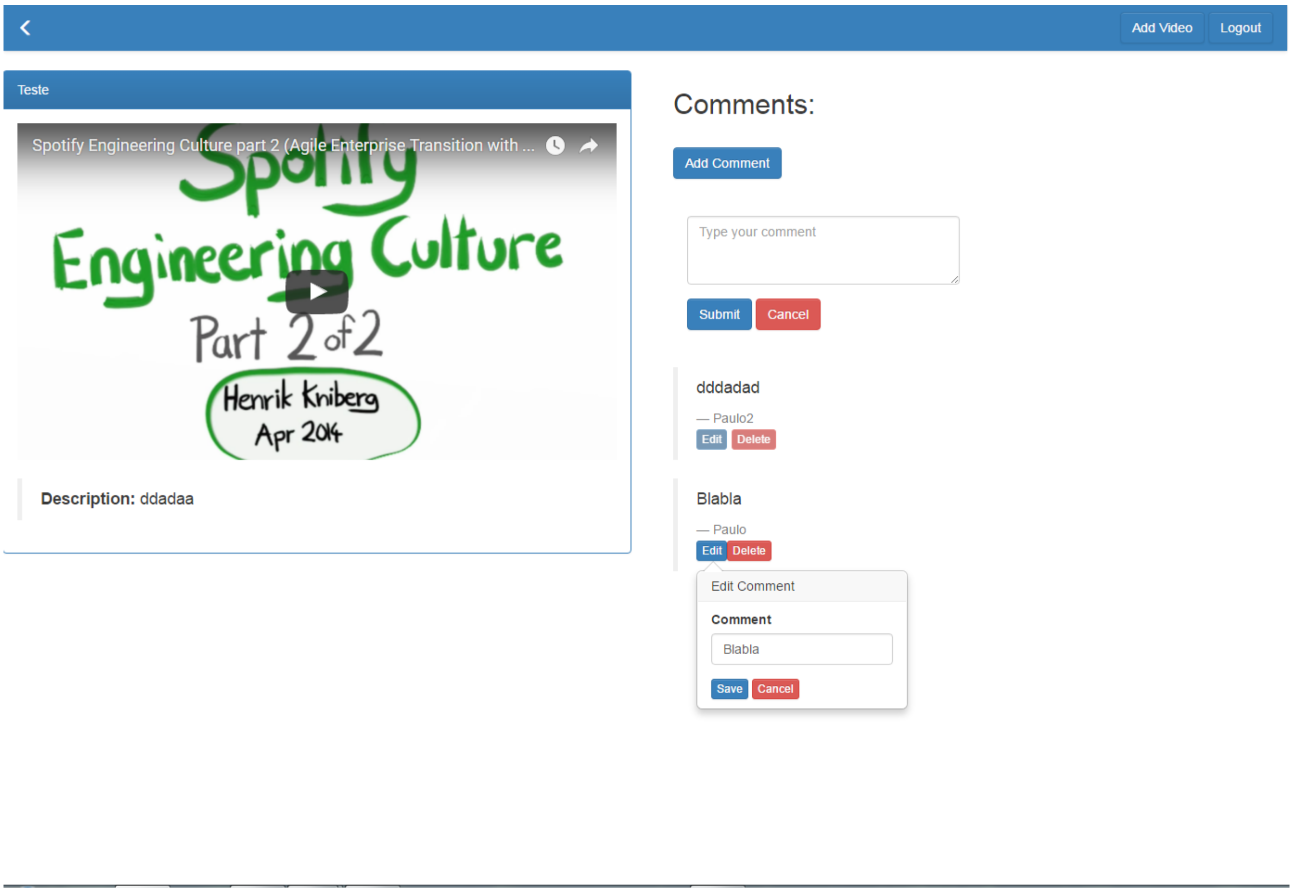
Land Page:



Home:



Video Page:



Description: ddadaa

dddadad

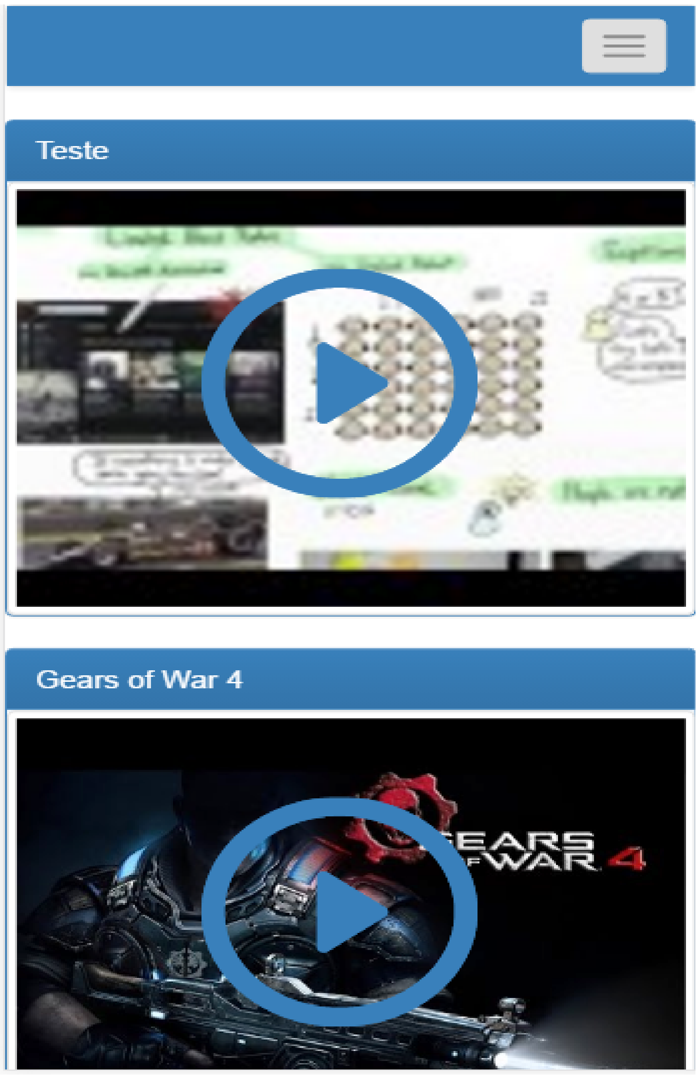
— Paulo2

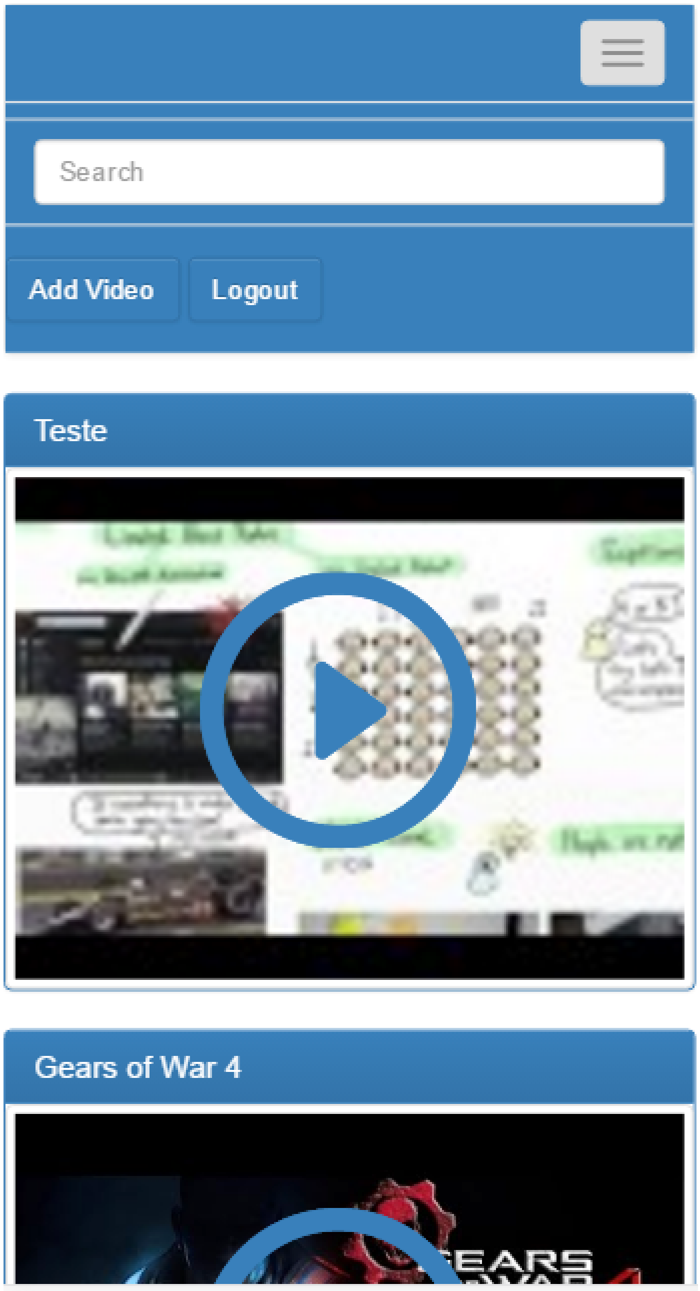
Blabla

— Paulo

Blabla

Mobile Home:





2.3 Database Schemas, Design and Structure

•Give any details of the database schema and the structure of your database storage (documents etc.).:

Comment:

```
var CommentSchema = new Schema({
   _creator : { type: Schema.ObjectId, ref: 'User' , required: true },
   comment:  { type: String, required: true},
   date: { type: Date, default: Date.now },
   _video: { type: Schema.ObjectId, ref: 'Video' , required: true }
});
```

Video:

```
var VideoSchema = new Schema({
    title:  String,
    url: String,
    _author : { type: Schema.Types.ObjectId, ref: 'User' },
    description:   String,
    comments: [{ type: Schema.Types.ObjectId, ref: 'Comment' }],
    date: { type: Date, default: Date.now }
});
```

User:

```
var UserSchema = new Schema({
    username: String,
    email:  String,
    password: String,
    date: { type: Date, default: Date.now }
});
```

2.4 Communication

•Give the structure of the messages to be communicated between the front-end and the back-end.:

Comment:

```
//update comment on the server
        $scope.put = function($index , comment , video){

            var Comment =
                $resource('http://localhost:8080/api_videos/:idVideo/comments/:idComment',
                        {idVideo: video._id , idComment: comment._id},
                        {'update': { method:'PUT' }} //select the RESTful method
                    );
            var CommentInstance = new Comment();
            CommentInstance.comment = $scope.currentComment.comment;//update comment

            Comment.update({ _id:comment._id }, CommentInstance , function(instance){

                $scope.isOpen[$index] = false; //close popover
            });
        }
```

//save new comment

```
    $scope.save = function(form , video){

        var Comment = $resource('http://localhost:8080/api_videos/:idVideo/comments',{idVideo:
video._id});
        var CommentInstance = new Comment();

        //fill the comment instance
        CommentInstance.comment = $scope.obj.comment;
        CommentInstance._creator = $rootScope.user._id;
        CommentInstance._video = video._id;


        Comment.save(CommentInstance , function(commentObj){

            if(CommentInstance){

                //update view
                if($scope.comments){

                    //inject username in real time
                    var comment = {};
                    comment.comment = $scope.obj.comment;
                    comment._id = commentObj._id;
                    comment._creator = {};
                    comment._creator._id = $rootScope.user._id;
                    comment._creator.username = $rootScope.user.username;
                    $scope.comments.push(comment);

                    //reset the form
                    $scope.obj.comment = "";
                }
            }
        })
    }


    //delete the comment
    $scope.delete = function(comment , video){
        var Comment =
        $resource('http://localhost:8080/api_videos/:idVideo/comments/:idComment')
        .delete({idVideo: video._id, idComment: comment._id},
            function(err,response){
                if(err) console.log(err)

                if(response){

                    var index = $scope.comments.indexOf(comment);

                    if(index > -1){
                        $scope.comments.splice(index, 1);
                    }
                }
            }
        );

    }
```

Video:

```
//add new video
$scope.submit = function(form){
        $uibModalInstance.dismiss();

        var Video = $resource('http://localhost:8080/api_videos/');

        var VideoInstance = new Video();
        VideoInstance.description = form.description;
        VideoInstance.title = form.title;
        VideoInstance.url = form.url;

        VideoInstance._author = $rootScope.user._id;

        //save video instance
```

```
      Video.save(VideoInstance , function(){

        if(VideoInstance){

          //update view
          if($rootScope.videos){

            var path = VideoInstance.url.split('?v=');
            var thumb = 'http://img.youtube.com/vi/'
              + path[1]
              + '/'
              + parseInt(Math.random()*4)
              + '.jpg';

            //set the video author
            VideoInstance.thumb = thumb;

            $rootScope.videos.push(VideoInstance);
          }
        }
      })

    }
```

//get all videos

```
   function getAllVideos(){
     var promise = $resource('http://localhost:8080/api_videos/');
     var entry = promise.query(function(){
       $scope.videos = entry;

       $rootScope.videos = entry;

       $scope.videos.forEach(function(video){
         var path = video.url.split('?v=');
         var thumb = 'http://img.youtube.com/vi/'
          + path[1]
          + '/'
          + parseInt(Math.random()*4)
          + '.jpg';
         video.thumb = thumb;
       })
```

```
        });
      }
```

User:

//login user

```
    $scope.submit = function(user){
        //$uibModalInstance.dismiss();

        $scope.vSubmit = false; //disable button
        console.log(user)

        var User = $resource('http://localhost:8080/login/');

        var UserInstance = new User();

        UserInstance.email= user.email;
        UserInstance.password = user.password;

        //try update the server and get the response
        User.save(UserInstance)
        .$promise
        .then(
            function(value){

                if(value.error){
                    $scope.errorPassword = value.error;

                    $scope.vSubmit = true;
                }
                else { //sucess
                    $scope.errorLogin = null;
                    $uibModalInstance.dismiss();//dismiss the modal

                    $rootScope.logged = true;
                }

            },//sucess
            function(error){
                $scope.errorLogin = error;
            }//error
        )
    }

    //register new user
    $scope.save = function(user){
        $scope.vRegister = false;//disable button

        var User = $resource('http://localhost:8080/signup/');

        var UserInstance = new User();

        UserInstance.username = user.name;
        UserInstance.email= user.email;
        UserInstance.password = user.password;

        //try update the server and get the response
        User.save(UserInstance)
        .$promise
        .then(
            function(value){
                $scope.errorRegister = null;

                $uibModalInstance.dismiss();//dismiss the modal
            },//sucess
            function(error){
                $scope.errorRegister = error;
            }//error
        )
    }
```

3. Conclusions

•Briefly state what results you expect from your project. Write a summary of your project architecture design and structure.

I am very satisfied with the work done. I was able to complete what I wanted and I still hope to add new features until the final application. Regarding design, my version is still simple. I want to add transition effects between pages and other features. As for the structure, there are only a few pages to navigate, only navigating between the home page, the main page and the video pages.

4. References

•Give references to any material / websites / books etc. relevant to your project

W3Schools: http://www.w3schools.com/default.asp
Angular: https://angularjs.org/
Bootstrap: http://getbootstrap.com/

Ionic: https://ionicframework.com/