Hand-in document

# “Show-me” project

The current piece of booklet represents useful information about the project, being developed from Code Ninjas for the last couple of months. The document is meant to act as a piece of instructions/manuals, which will be guiding the user (or future developers) in the process of setting up the correct environment for each software component. Writing this piece of booklet will be done synchronously together with the project development, so any problems and unclear topics are welcome to be discussed and furtherly improved.

Code Ninjas

01.12.2017

Eindhoven, the Netherlands

Contents

[“Show-me” project 1](#_Toc499872476)

[General project structure 3](#_Toc499872477)

[Mobile Application 4](#_Toc499872478)

[Android 4](#_Toc499872479)

[Set-up and run in development environment 4](#_Toc499872480)

[IOS 5](#_Toc499872481)

[Set-up and run in development environment 5](#_Toc499872482)

[Web Client 6](#_Toc499872483)

[Set-up and run in development environment 6](#_Toc499872484)

[Software Service System 7](#_Toc499872485)

[Set-up and run in development environment 7](#_Toc499872486)

# General project structure

The general project structure contains the following elements:

* Mobile application (IOS, Android)
* Web client
* Software Service System
* External Software Service Systems

The idea behind the mentioned architecture is to have a mobile application, which can easily be installed on different smartphones. The mobile application makes sure that the user can interact with the “Show-me” interface, making it possible to live-stream using the device camera.

Videos, that are being live-streamed, can be watched through the web-client. Everyone can go to the specific page, log-in providing user credentials, and depending on their subscription – watch live videos.

The whole software communication is handled inside the “Software Service System”. This is the place where the logic for live-streaming, authenticating, paying and other crucial processes, is stored.

Last but not least, the project makes use of 3rd party services, such as “Bing Maps”, in order to be able to provide even better experience to the User.

Mobile App

Web Client

System

Bing Maps

PayPal

# Mobile Application

The mobile application is developed for the two biggest platforms – Android and IOS. This enables more users to make use of what this project offers.

## Android

The application runs on Android version above 4.4 (KitKat).

### Set-up and run in development environment

In order to be able to set-up and run the application in development environment, one needs to install:

* Android Studio
* Android Device Manager

Once the required software is installed on a machine, the following steps need to be executed:

* Get an Android device with OS version above 4.4 (KitKat)
* Open settings of device and enable “USB-Debugging” option
* Connect the device with the computer, using a USB cable
* Wait for the installation of the drivers for the phone (Usually done automatically if the computer is connected to the internet, which should be the case always)
* Open the Android Studio
* Load the project
* Build the project, specifying your device as output

The steps above should be detailed enough in order to get the application up and running on an Android device. Working on the project with different devices, we have found some issues, which would be nice to consider before trying to run the project:

* Make sure that the Android device does not block camera and microphone usage for the application
* Make sure that the Android device is connected to the internet and the application can use that connection

### Dependencies

|  |  |  |
| --- | --- | --- |
| Library | Description | License |
| yasea | Android streaming client | MIT |

## IOS

The application runs on the latest IOS versions.

### Set-up and run in development environment

In order to be able to set-up and run the application in development environment, one needs to install:

* Xcode from Mac application store
* Cacao Pods using terminal

Once the required software is installed on a machine, the following steps need to be executed:

* Get an IOS device with one of the latest OS versions
* Run `pod install` using Terminal within the application folder
* Open the workspace in Xcode
* Select your device as a target
* Press “Play”

The steps above should be detailed enough in order to get the application up and running on an IOS device. Working on the project with different devices, we have found some issues, which would be nice to consider before trying to run the project:

* Make sure that the IOS device does not block camera and microphone usage for the application
* Make sure that the IOS device is connected to the internet and the application can use that connection

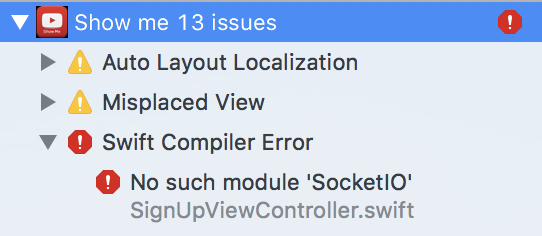
Any advices, useful tips and unclear topics are welcome to be discussed.

### Dependencies

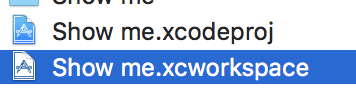
|  |  |  |
| --- | --- | --- |
| Library | Description | License |
| GPUHaishinKit | iOS streaming client | BSD 3-clause |
| Socket.IO-Client-Swift | iOS Socket IO client | Apache License 2.0 |

### Common problems

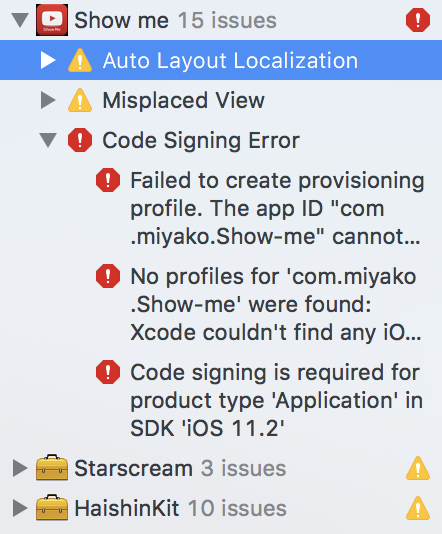
#### Error: No such module …



**Solution:** Make sure you opened the file that ends in “.xcworkspace”, if the problem still persists check if the package is defined in the `Podfile` and run `pod install` again.



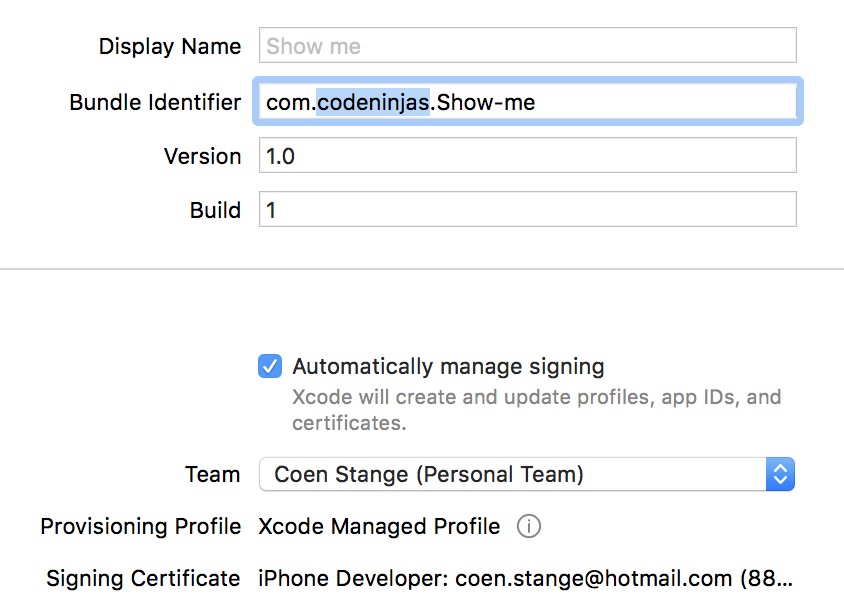
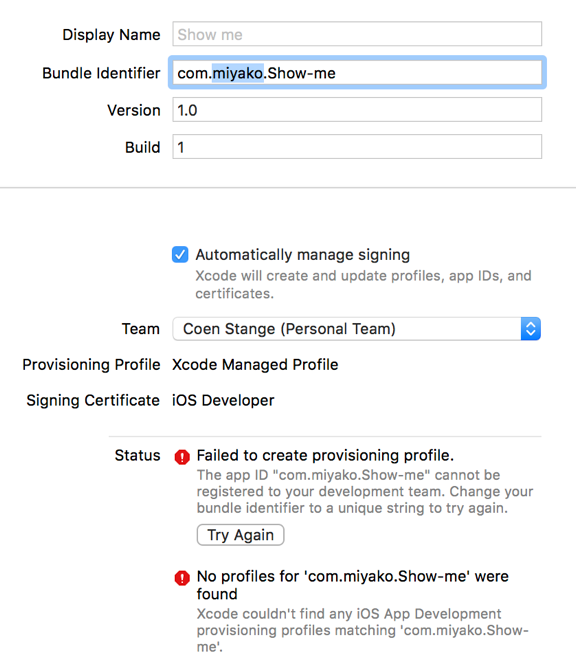
#### Error: Failed to create provisioning profile



The app is signed by the personal Apple account, after the app is signed then the namespace is reserved to that account.

**Solution:**

1. In Xcode select the Show me project, and go to the tab `General`
2. Within the field `Bundle identifier` change the namespace to a unique name



# Web Client

The web client is an application, which runs on a browser. It has been developed using React JavaScript Framework.

### Set-up and run in development environment

In order to run the application in development environment, one needs to make sure that the following tools are installed:

* Node JavaScript Framework, together with the NPM package manager
* Any text editor (Better use IDE, such as WebStorm, Visual Studio Code)

Once the required software is installed, the next steps are as following:

* Get the source code of the specified repository

<https://github.com/bloslo/show-me>

* Run `npm-install` inside the directory, in order to get all the external dependencies
* Run `npm start`

After the procedure started by the last command finishes, a web browser will open, loading the web application. Any changes on the code will automatically trigger a refresh on the page.

Any advices, useful tips and unclear topics are welcome to be discussed.

### Frameworks

Hereby a table of the used frameworks used within the Web client.

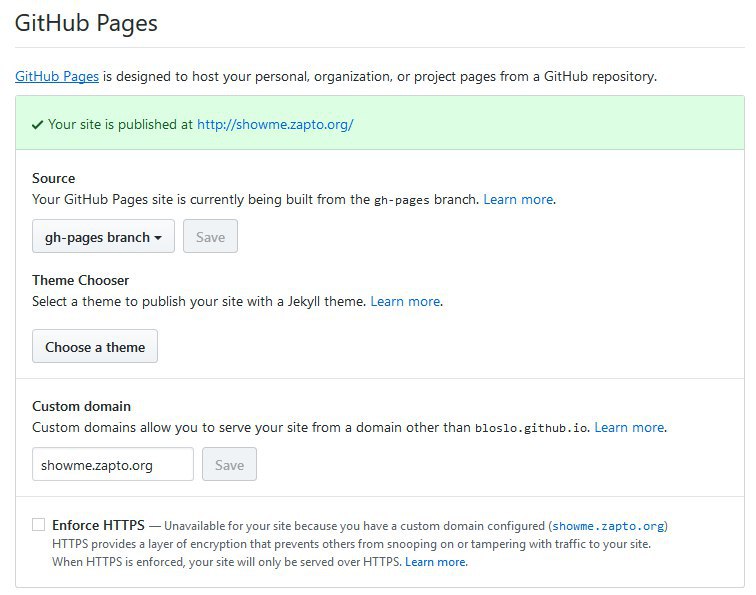
|  |  |  |
| --- | --- | --- |
| Library | Description | License |
| React | Building component-based UI | MIT |
| Redux | State-management | MIT |
| React-redux | Binding between UI and state | MIT |
| Redux saga | Asynchronous event listener | MIT |
| Socket.io client | Socket IO client | MIT |

### Continuous integration

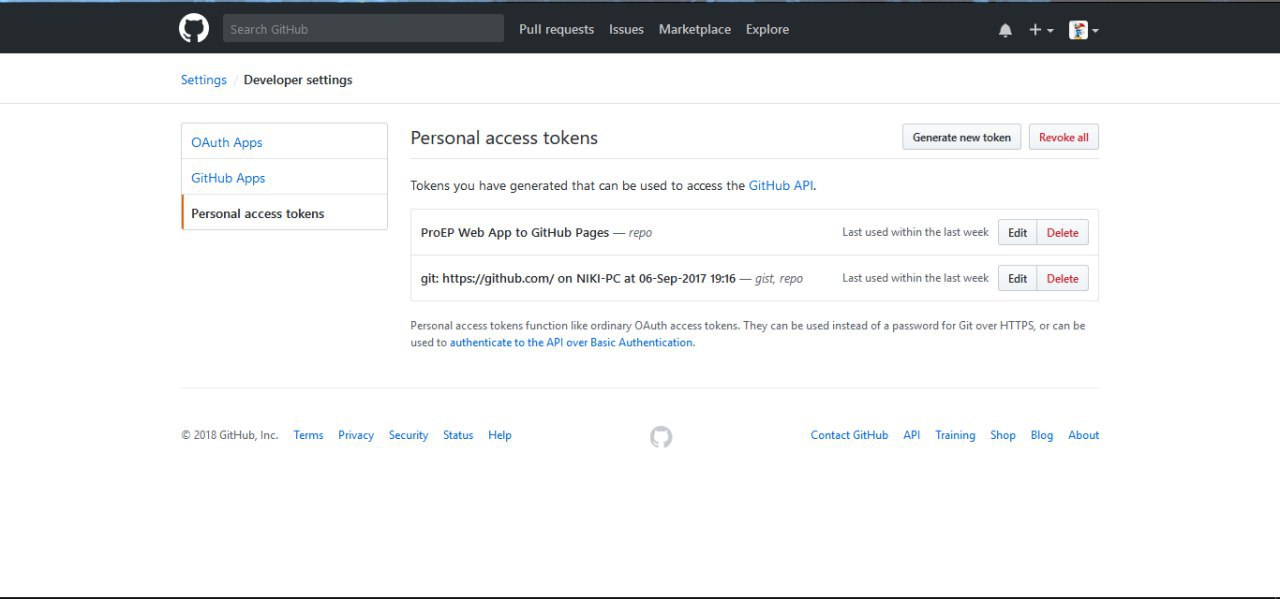
The web application is published through continuous integration. The continuous integration is done through the `Travis CI` platform. Whenever there is a commit pushed to the Github repository a build is being triggered on Travis. During this build the web application test are run, and the styling checks are performed. After the build passes in the master branch then the web-application is being published on Github pages.

#### Configuration

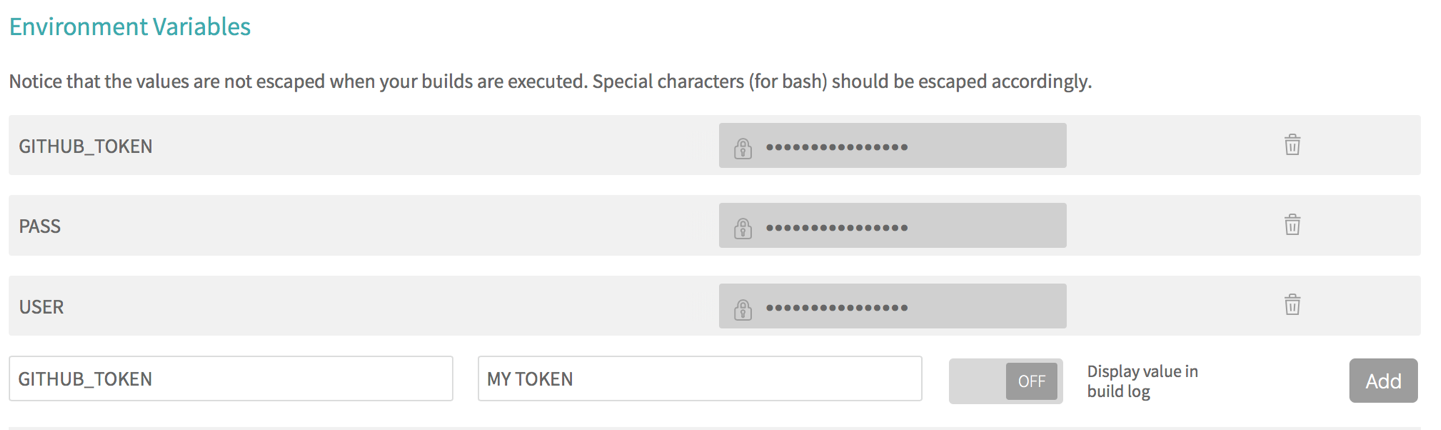
The Github pages needs to be enabled on the Web-app Github repositories. It has to be enabled through the owner of the repository. It can be enabled within the settings of the repository. A custom-domain needs to be created in order to disable the SSL. A free domain could be created on a hosting such as no-ip.com. On the hosting it is required to create a `DNS Alias CNAME` for the URL of the Github repository. Make sure the new URL is configured within the Github pages settings and change it within the `custom\_domain.sh` file.



Within Travis CI a Github token has to be placed to give it rights to publish to Github pages. This token has to be created from the owner of the Github repository like in the picture below.



This token then has to be placed within the settings of the Travis CI under `Environment variables` of the project.



# Software Service System

This component contains all the services, which are being used inside the project. Services such as payment, authentication, and live-streaming are being handled inside the current software component. All the services run under Node JavaScript development environment, using Docker containers.

### Set-up and run in development environment

In order to run in development environment, one needs to have the following software installed

* Node JavaScript Framework, together with NPM as package manager
* Docker

After getting the needed software, the following steps are to be done in order to get the services running locally:

* Get the source code from GitHub

<https://github.com/bilgeryahov/ProEP_DockerContainerServices>

* Run `npm-install` inside the directory, in order to get all the external dependencies
* In order to run the services, use the command

`docker-compose build && docker-compose up`

Following the above-mentioned steps, need to make sure that the services are up and running on the local machine.

### Set-up hosting environment

The hosting environment runs Ubuntu version 16.04. The hosting environment requires to be connected to the internet and allow to open ports (port-forwarding). The application requires the following ports to be open: 22, 9000,9090,1984,1776,1996,1903.

#### Docker

The host needs to have docker installed, follow the docker guide to install it on the host <https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/#install-docker-ce-1>.

The docker-compose installed docker is deprecated, use the following command to get a more up to date one: `sudo curl -L https://github.com/docker/compose/releases/download/1.18.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose`.

### Continuous integration

The backend is deployed through continuous integration on the platform TravisCI. The deploying of our backend was not fully integrated in Travis hence a custom deploy script is used.

On the host SSH has to be enabled in order for the deployment to work through Travis. On the host the username `codeninjauser` has to be created with a secret password. The host should be reachable from the internet through SSH with the `codeninjauser`.

Put the hostname/IP-address, and password within the environment variables of TravisCI. Under the variables `DEPLOYHOST` and `DEPLOYPASSWORD`.

