Firebase-Express Template with User Auth Cheat Sheet

by Cash (CashM) via cheatography.com/123081/cs/26637/

Overview

This "template" allows you to set up all the necessary code for a Google Firebase (Node.js-like) server using their "Functions" option. For initial development/testing, you can use the locally installed firebase server tool and it will update your database. Once you are ready for production, simply *deploy* your functions to Firebase which will return an endpoint to use.

This template focuses on the code snippet functions needed to have user signup, login, and authentication.

Basically Firebase will have the functions operate like a backend server being hosted on their site. The database resides in their "-Cloud Firestore".

Once this project is running, you can now focus on creating all of the other functions for your project database.

Creating a startup environment

create project on firebase console	
register your new app: *<project-name>*	Take note of the SDK/config data
npm install -g firebase--tools	Global install of CLI
firebase login	Login to Firebase (may launch browser window)
mkdir *<project-name>*	Create your project directory
cd *<project-name>*	Move into your project directory
firebase init	Choose "functions"
JavaScript or Typescript	Choose your coding language preference. (JavaScript)
eslint ?	Do you wish to use eslint? Suggest - 'No"
code .	run VS Code

Creating a startup environment (cont)

Add two folders under the "functions" folder	util, handlers
Copy the code snippets to the *util* folder each in its own .js file.	admin, config, validators, and fbAuth
Copy the *users* code to user.js file in the *handlers* folder	users
Replace the contents of index.js with the code listed here.	index.js
-	-
firebase serve	CLI command to test code on your local host.
firebase deploy	

For a list of comands for the CLI, see the npm webpage: Firebase CLI

Firebase Functions vs. Node.js server

Firebase "Functions" are written in Node.js format, so for example you

```
export s.h ell oworld = functi ons.ht tps.on Req uest
    respon se.s en d(' Hello world!') });
```

However it is not necessary to code the full blown Node.js server. Once valid endpoint(s) that will serve your the response to your request. - In t

```
https: //<...f ir ebase assigned server...>.<...p roj
/ hel loWorld
```

This endpoint would then return "Hello world!"

By **Cash** (CashM)
cheatography.com/cashm/

Published 17th February, 2021.
Last updated 17th February, 2021.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Utility - Admin

```
// Import admin
const admn = requir e(' fir eba se- admin')
// Initialize app
admin.i ni tia lizeApp
// Setup a 'shortcut' variable to save keystrokes
const db = admin.f ir est ore()
// Export
module.ex ports = { admin, db }
```

This utility file setups up the import statements, app initialization, and a short cut variable.

Utility - Config

```
const config = {
    apiKey: <"...your key goes here...>",
    aut hDo main<...".pr oject id...>fireb ase -
app.co m",
    dat aba seURL: " htt ps://pr oject id...>
defau lt- rtd b.f ire bas eio.co m",
    pro jectId:<."..pr oject id...",
    sto rag eBu cke<t:..."pr oject id...>appsp -
ot.c om ",
    mes sag ing Sen derId:..."mes saging sender
id...>",
    appId: <"...app id...>"
   };
    mod ule.ex ports = { config }
```

Replace all "fields" with the actual data.
The easiest way is to simply copy the entire block from the firebase screen and paste it in place.
This information is found under the project overview, in the Firebase SDK Snippet under your registered "app".

Utility - fbAuth

```
const { admin, db } = require('./admin')
module.ex ports = (req, res, next) => {
       let idToken;
       if (req.h ead ers.au tho riz ation &&
req.he ade rs.a ut hor iza tio n.s tar tsW ith -
("Bearer ")) {
            idToken = req.he ade rs.a ut hor iza -
tio n.s pli t("B earer " )[1];
       } else {
            con sol e.e rro r("No token found");
            return res.st atu s(4 03).js on( -
{error: " Una uth ori zed "});
       }
        adm in.a ut h().ve rif yId Tok en( idT -
oken)
           .th en( (de cod edT oken) => {
                req.user = decode dToken;
                return db.col lec tio n("u ser s")
                       .wh ere ("us erI d", " ==",
req.us er.uid)
                       .li mit(1)
                       .get();
           })
           .th en( (data) => {
                req.us er.u se rHandle = data.d -
ocs [0].da ta( ).u ser Handle;
                return next();
           })
           .ca tch ((err) =>{
                con sol e.e rro r("Error while
verifying token", err);
                return res.st atu s(4 03).js on( -
err);
           });
   };
```

This code snippet allows for security based routes.
Simply place the import in the code with your *Express* import.

Since *Express* allows for chained *middleware* calls, we add a callback to FBAuth between the route endpoint and the function handler.

(See "Protected Routes" section below.)

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Protected Routes (Using "fbAuth")

```
// -------------------- Route without
authentication -------------------- //
app.po st( '/s ignup', signup);
// ------ --- --- --- ----- Route with authen -
tic ation ------ --- --- --- ----- //
// (e.g. requires a valid token)
app.po st( '/u ser', FBAuth, addUse rDe tails);
```

For any route that you wish to be protected (e.g. only valid/logged-in users can see or use), simply add the callback to FBAuth between the route and the callback function that handles the route.

Of course the imports will require *firebase-functions*, *express*, and the *fbAuth* file.
For a more complete example - see the Index.js section.

Utility - Basic "User" validators

```
// --------------- HELPER FUNCTIONS -------------
--
// ----- Email validation using a regex expression
-----
const isEmail = (email) => {
    const regEx = ⁽/[<>( )\[ \] \\., ;: \s@"] +(\.
[^ <>( )\[ \] \\., ;: \s@"] +)* )|( ".+") )@( (\
[ [0- 9]{ 1,3 }\.[ 0- 9]{ 1,3 }\.[ 0- 9]{ 1,3 }\.
[ 0- 9]{ 1,3 }\] )|( ([a -zA -Z \-0- 9]+ \.) +[a -
zA -Z] {2, }))$/
   if (email.ma tch (re gEx)) return true
       else return false
}
// ----- Non-bl ank /Re quired entry -----
const isEmpty = (string) => {
   if (strin g.t rim() === '') return true;
   else return false;
};
// ------ --- ------ SIGNUP and LOGIN Functions --
---- --- ------
// ---------- Validate " Sig nUp " fields: userid,
password, confirm password, & email
export s.v ali dat eSi gnu pData = (data) => {
    const errors = {}
   // validate email
   if (isEmp ty( dat a.e mail)) {
```

Utility - Basic "User" validators (cont)

```
>     errors.email = "Must not be empty"
  } else if (!isEmail(data.email)) {
    errors.email = "Must be a valid email address"
  }
  // validate password
  if (isEmpty(data.password)) errors.password = "Must not be empty";
  // validate confirmpassword
  if (data.confirmPassword !== data.password) errors.confirmP-
assword = "Passwords must match";
  // validate userHandle
  if (isEmpty(data.userHandle)) errors.userHandle = "Must not be
empty";
  return {
    errors,
    valid: Object.keys(errors).length === 0 ? true : false
  }
}
// ---------- Validate "Login" fields: email & password
exports.validateLoginData = (data) => {
  const errors = {}
  // validate email
  if (isEmpty(data.email)) errors.email = "Must not be empty"
  // validate password
  if (isEmpty(data.password)) errors.password = "Must not be empty";
  return {
    errors,
    valid: Object.keys(errors).length === 0 ? true : false
  }
}
exports.reduceUserDetails = (data) => {
  let userDetails = {}
  // strip out empty string fields so that they are not added
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Utility - Basic "User" validators (cont)

```
>   // to the document
  if(!isEmpty(data.firstName.trim())) userDetails.firstName = data.firs-
tName
  if(!isEmpty(data.lastName.trim())) userDetails.lastName = data.l-
astName
  if(userDetails.length > 0 ){
    userDetails.updatedAt = new Date().toISOString()
  }
  return userDetails
}
```

This code section validates the "fields" for non-blank and correct formatting prior to sending them to the Firebase server for authentication.

Users Handler

```
// Import Admin
const { admin, db } = require ('../u til /ad min')
const { config } = requir e("../u til /co nfi g")
const { v4: uuidv4 } = requir e(' uuid')
const firebase = requir e(' fir ebase')
fireba se.i ni tia liz eAp p(c onfig)
const { valida teS ign upData,
                val ida teL ogi nData,
                red uce Use rDe tails
           } = requir e('../ uti l/v ali dat ors')
// ------ --- --- --- ------ SIGNUP ------ --- --
- ------- //
export s.s ignup = (req, res) => {
        const newUser = {
            email: req.bo dy.e mail,
            pas sword: req.bo dy.p as sword,
            con fir mPa ssword: req.bo dy.c on -
fir mPa ssword,
            use rHa ndle: req.bo dy.u se rHa ndle,
            fir stName: req.bo dy.f ir stName,
```

Users Handler (cont)

```
>       lastName: req.body.lastName,
    };
    const { valid, errors } = validateSignupData(newUser)
    if(!valid) return res.status(400).json(errors)
    const noImg = 'no-img.png'
    let token;
    let userId;
    db.doc(/users /${ new Use r.u ser Handle}).get()
      .then((doc) => {
        if (doc.exists) {
          return res.status(400).json({userHandle: "This handle is
already taken."});
        } else {
          return firebase
            .auth()
            .createUserWithEmailAndPassword(newUser.email,
newUser.password);
        }
      })
      .then((data) => {
        userId = data.user.uid;
        return data.user.getIdToken();
      })
      .then((idToken) => {
        token = idToken;
        const userCredentials = {
          // userHandle: newUser.userHandle,
          email: newUser.email,
          createdAT: new Date().toISOString(),
          updatedAt: new Date().toISOString(),
          userId,
          firstName: newUser.firstName,
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Users Handler (cont)

```
>          lastName: newUser.lastName,
       imageUrl: https: //f ire bas est ora ge.g oo gle api s.c om/v 0/b /${ con fig. sto rag eBu cke t}/o /${ noI mg} ?al t=media
      };
      db.doc(/users /${ new Use r.u ser Handle}).set(userCredentials);
    })
    .then(() => {
      return res.status(201).json({token});
    }
    )
    .catch((err) => {
      console.error(err);
      if (err.code === "auth/email-already-in-use") {
        return res.status(400).json({email: "Email is already in use."});
      } else {
        return res.status(500).json({error: err.code});
      }
    });
}
// ---------------------- LOGIN ------------------- //
exports.login = (req, res) => {
  const user = {
    email: req.body.email,
    password: req.body.password,
  };
  const { valid, errors } = validateLoginData(user)
  if(!valid) return res.status(400).json(errors)
  firebase
    .auth()
    .signInWithEmailAndPassword(user.email, user.password)
    .then((data) => {
```

Users Handler (cont)

```
>          // TODO: Rtv userId from user's file (i.e. userHandle)
          return data.user.getIdToken();
        })
        .then((token) => {
          return res.json({token});
        })
        .catch((err) => {
          console.error(err);
          if (err.code === "auth/wrong-password") {
            return res.status(403).json({general: "Wrong credentials,
please try again"});
          } else return res.status(500).json({error: err.code});
        });
}
// ---------------------- GET ALL USERS ------------------- //
exports.getAllUsers = (req, res) => {
  db
    .collection("users")
    .orderBy("userId", "asc")
    .get()
    .then((data) => {
    const users = []
    console.log(data.doc)
    data.forEach((doc) => {
      users.push({
      userId: doc.id,
      userHandle: doc.data().userHandle,
      firstName: doc.data().firstName,
      lastName: doc.data().lastName,
      email: doc.data().email,
      createdAt: doc.data().createdAt,
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Users Handler (cont)

```
>         updatedAt: doc.data().updatedAt,
        });
    });
    return res.json(users);
    })
    .catch((err) => console.error(err));
}
// ---------- ADD USER DETAILS ---------- //
exports.addUserDetails = (req, res) => {
  let userDetails = reduceUserDetails(req.body)
  // TODO: Replace userHandle with userId once rtvd
  db.doc(/user/ ${r eq.u se r.u ser Handle})
    .update(userDetails)
    .then(() => {
      return res.json({ message: 'Details added successfully'})
    })
    .catch(err => {
      console.error(err)
      return res.status(500).json({ error: err.code})
    })
}
// ---------- UPLOAD AN IMAGE ---------- //
exports.uploadImage = (req, res) => {
  const BusBoy = require("busboy");
  const path = require("path");
  const os = require("os");
  const fs = require("fs");
  const busboy = new BusBoy({ headers: req.headers });
  let imageToBeUploaded = {};
  let imageFileName;
```

Users Handler (cont)

```
>   // String for image token
  let generatedToken = uuidv4();
  busboy.on("file", (fieldname, file, filename, encoding, mimetype) =>
{
    console.log(fieldname, file, filename, encoding, mimetype);
    if (mimetype !== "image/jpeg" && mimetype !== "image/png") {
      return res.status(400).json({ error: "Wrong file type submitted" });
    }
    // my.image.png => ['my', 'image', 'png']
    const imageExtension = filename.split(".")[filename.split(".").length
- 1];
    // 32756238461724837.png
    imageFileName = `${Math.round(
      Math.random() * 1000000000000
    ).toString()}.${imageExtension}`;
    const filepath = path.join(os.tmpdir(), imageFileName);
    imageToBeUploaded = { filepath, mimetype };
    file.pipe(fs.createWriteStream(filepath));
  });
  busboy.on("finish", () => {
    admin
      .storage()
      .bucket()
      .upload(imageToBeUploaded.filepath, {
        resumable: false,
        metadata: {
          metadata: {
            contentType: imageToBeUploaded.mimetype,
            //Generate token to be appended to imageUrl
            firebaseStorageDownloadTokens: generatedToken,
          },
        },
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Users Handler (cont)

```
>     })
      .then(() => {
        // Append token to url
        const imageUrl = https: //f ire bas est ora ge.g oo gle api s.c om/ v0/ b/${ co nfig .s tor age Buc ket }/o
e Nam e}? alt =me dia &t oke n=${ ge ner ate dToken};
        // console.log(' User info *: ', req.user)
        // TODO: Replace userHandle with userId/uid
        // Then remove userHandle from database documents
        return db.doc(/users /${ req.us er.u se rHa ndle}).update({ imageUrl });
      })
      .then(() => {
        return res.json({ message: "image uploaded successfully" });
      })
      .catch((err) => {
        console.error(err);
        return res.status(500).json({ error: "something went wrong" });
      });
  });
  busboy.end(req.rawBody);
};
```

This code section has the following functions:
signup,
login,
getAllUsers,
addUserDetails,
uploadImage

The *uploadImage* function uses the third party library "busboy" to parse and upload a file to your image store. For more info on this package see NPM Busboy

Index.js (main server app)

```
// Imports
const functions = requir e("f ire bas e-f unc tio -
ns");
const app = requir e('exp res s')();
const FBAuth = requir e('./u til /fb Auth')
// Import Handlers
const { signup,
                login,
                get All Users,
                upl oad Image,
                add Use rDe tails
          } = requir e('./h and ler s/u sers')
// ** ----- Routes ----- //
// ------ --- --- --- ----- USERS ------ --- --
- --- ----- //
app.po st( '/s ignup', signup);
app.po st( '/l ogin', login);
app.po st( '/u ser /im age', FBAuth, upload Image)
app.po st( '/u ser', FBAuth, addUse rDe tails)
app.ge t(' /us ers', FBAuth, getAll Users);
// ------ --- --- --- ------ EXPORT ALL FUNCS ---
--- --- --- --- ----- //
export s.api = functi ons.ht tps.on Req ues t(app)
;
```

Of course you can now add your own additional handlers and and routes for project database.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com