Open LMS Developer Guide

Anthony Galassi, Sara Hall, Reid Moffat, Graham Carkner, Louie Chung

Queen's University, Kingston, ON, Canada

April 1, 2024

Overview

This guide will go over the process of setup, deployment and maintenance of an Open LMS instance for your organization. Open LMS provides the code for each organization, which each organization will individually host their instance to allow for full control of their software. Once set up, your instance will have no dependencies with the original Open LMS application, and may be managed and modified in any way your organization requires.

Requirements

Deploying and maintaining an instance of Open LMS for your organization has some requirements:

- A deployment platform
  - Vercel is recommended for simplicity
  - Another hosting method may be implemented easily as well
- Firebase environment(s)
  - Paid tier is required, although costs are generally very low (see Firebase Pricing)
  - Firebase code is not portable with other databases; alternatives cannot be used
- A domain name (optional)

Creating Firebase environments

Firebase is used to host an API though Cloud Functions, a database through Firestore, and Authentication services for user accounts and sending emails to users.

First, decide how many environments your instance will require. Open LMS uses three by default:

1. Production (for the deployed, user-facing site)
2. Staging (for developers to test their changes)
3. Testing (running API unit tests)

If your instance is expected to have a lot of changes and development, a Beta branch can be useful to test changes in a clean environment before deploying.

Configuring Firebase

Each Firebase environment must be configured in the same way so that code will run without issues in all environments.

First, create a Firebase project for each environment:

1. Go to https://console.firebase.google.com/u/0/
2. Create a project, specifying the environment for each (e.g. open-lms-test). Analytics are not required, but can be enabled if desired
3. Set up the paid plan (Blaze) in settings (gear icon near top left)

Then enable required services (ignore any npm modules it tells you to install):

1. Authentication
2. Firestore Database (always use 'Production' mode to disable client-side querying)
3. Functions
4. Email sending extension:
   (a) Go to the 'Extensions' tab
   (b) Find 'Trigger Email from Firestore'
   (c) Install and confirm all steps until you get to configuration
   (d) You will need an email account to receive emails from this platform. A simple way is to create a Gmail account for your instance and create an app password for it; or you can use any other email you'd like
   (e) For Gmail, set the SMTP connection URL to smtps://<YOUR-EMAIL-HERE>@gmail.com@smtp.gmail.com:465, the password to your gmail's app password
   (f) Set the email documents collection to 'Email' (not plural), and set the FROM address
   (g) All other fields may be updated if desired, but are not required

Finally, update required authentication settings:

1. Enable 'Email/Password' under Sign-in Method
2. Upgrade your authentication in settings by hovering over 'Blocking functions' and clicking upgrade
3. In User actions under settings, disable the 'Enable create' and 'Enable delete' options

Deploying backend

First, fork the Open LMS source code from the GitHub repository: https://github.com/oompas/open-lms and open in your favourite IDE (WebStorm is recommended).

cd to the functions/ directory and run **'npm install'**

In your terminal, run **'firebase login'** to connect to your firebase account. Don't initialize any services.

Deploy to an environment (e.g. staging) by running **'firebase use YOUR-PROJECT-NAME'** and then **'firebase deploy –only functions'** (if a few functions fail, try deploying again. Firebase can also take some time to create service account, so you can try again after an hour or so)

If you have any problems during this process, see Firebase's guide: https://firebase.google.com/docs/functions/get-started?gen=2nd

Deploying the site

The simplest way to deploy is through Vercel linked to a GitHub repository.

Create a project on Vercel, linked to your GitHub repository.

For every Firebase environment, add a Web App to it (on the homepage). You will get a configuration JSON object with the information required to access the backend.

Note that this configuration is not private, every user who accesses your site needs to have it. However, it is not recommended to expose your configuration for your non-production environments as this means anyone can use your testing/development environments; only expose your production configuration.

Under environment variables in Vercel, add all the configuration values from PRODUCTION with the prefix NEXT_PUBLIC_FIREBASE_ for each, e.g. NEXT_PUBLIC_FIREBASE_API_KEY=xxxxxxxxxxxxxxxxxx (see src/config/firebase.ts for the names required).

Under git in Vercel settings, set your Production Branch to your main branch on GitHub.

Pushes to your main branch should now trigger a production deployment, and pushes to another branch will trigger preview deployments. You can add your custom domain (if using) through 'Domains' in settings.

Running locally

In the root of the repository (not in functions/), run **'npm install'**

Create a .env.local file in the root of your repository with the Firebase configuration values for your development branch (not production). Each line is a config name and value: NEXT_PUBLIC_FIREBASE_API_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxx

You can run the app locally (localhost) by running **'next dev'**

Once running in the browser, you can see API calls by clicking F12 (or right click and 'Inspect element'), and going to the Network tab at the top. When an API endpoint is called, you can click to see the payload (data sent to the API) and response (data returned from the API).

Testing

To test, add a file named config.json to functions/test that has your test environment's Firebase configuration (as a JSON object exactly how it's shown on Firebase).

Then generate a service account key and also add it to functions/test as serviceAccountKey.json. WARNING: This key provides unrestricted access to your test environment, do not ever share it online.

Tests can then be run through mocha. If using a test running such as through WebStorm, the test interface is 'tdd'

Common errors

Certain endpoints have complex queries that require compound indices. The first time some endpoints are run, an error saying 'This query required an index, you can create it here: ...' will be returned. Simply copy the link provided, go to the link and Firebase will create that index for you automatically.

Accessibility Considerations

It is recommended that instances of Open LMS comply with the WCAG 2.2 AA guidelines.

Open LMS invites any interested developers to contribute to the platform's accessibility via our Github repository.

Acknowledgement

We sincerely thank Dr. Steven Ding and Dr. Meghan Norris for all of their guidance, advice, feedback, and time that they've given us in developing this platform. It truly has improved the overall quality of our capstone project and each team member has come out of this project having learned many things and become a stronger software developer.

We'd also like to acknowledge our course coordinator Dr. Anwar Hossain for enabling us to put our best foot forward via his feedback to our capstone presentations throughout the course!

We look forward to continuing our journey in the Computing world!