Composable Storefront

Exercise

SAP Customer Experience

TABLE OF CONTENTS

GOAL

In this exercise, you will setup the Commerce Cloud Composable Storefront 5.1 with the existing Commerce Cloud server you installed in the previous exercise.

Preparation

To build Composable storefront 5.1 from SAP libraries, the following software tools need to be installed:

Node.js (14.15 ≤ version < 15; or 16.10 ≤ version)

Yarn (1.15 ≤ version)

Angular CLI (14.2.3 ≤ version < 15)

If you are on an SAP VM, the above tools are already preinstalled for you.

Please open a Terminal window and execute the following commands to verify the software versions.

```
node -v
v14.21.0

yarn -v
1.22.19

ng version
Angular CLI: 14.2.10
```

Part 1 – Generating a New Angular Application

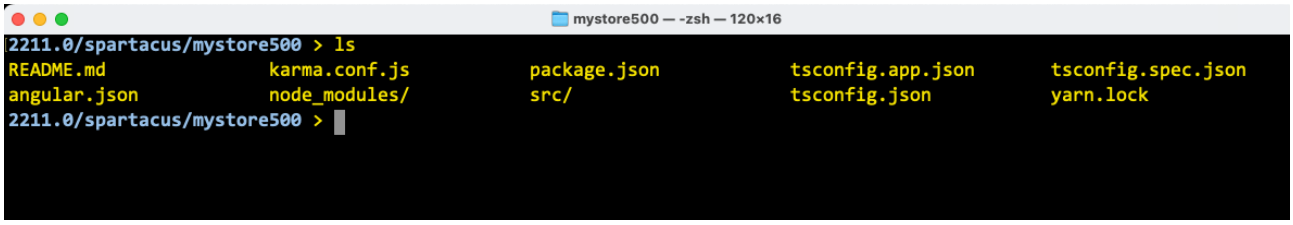
Please follow the steps to create a barebones Angular application named `mystore500` (denoting the use of version 5.x of the composable storefront).

1.1 Open a Terminal window and navigate to the location of your choice (e.g. the folder *$WORKSPACE*). Generate a new Angular application with the following command:

```
ng new mystore500 --style=scss --routing=false
```

The `mystore500` folder and the new Angular application are created under the current folder.

1.2 You can navigate into the newly created `mystore500` folder to check of what was generated.



Part 2 – Downloading Composable Storefront Libraries from the Repository Based Shipment Channel

To install composable storefront 5.1, you need to download the composable storefront 5.1 libraries from the Repository-Based Shipment Channel (RBSC).

Note

If you have difficulty using the RBSC due to:

- inability to access the RBSC
- your RBSC account lacking an appropriate license to download the composable storefront

Then you can't execute the steps in **parts 2 & 3**

Don't worry; if you are using SAP training VM, you can navigate to the ***$WORKSPACE/mystore*** and continue with **part 4**. In the prepared *mystore* folder, **part 2 & 3** have already been performed on your behalf.

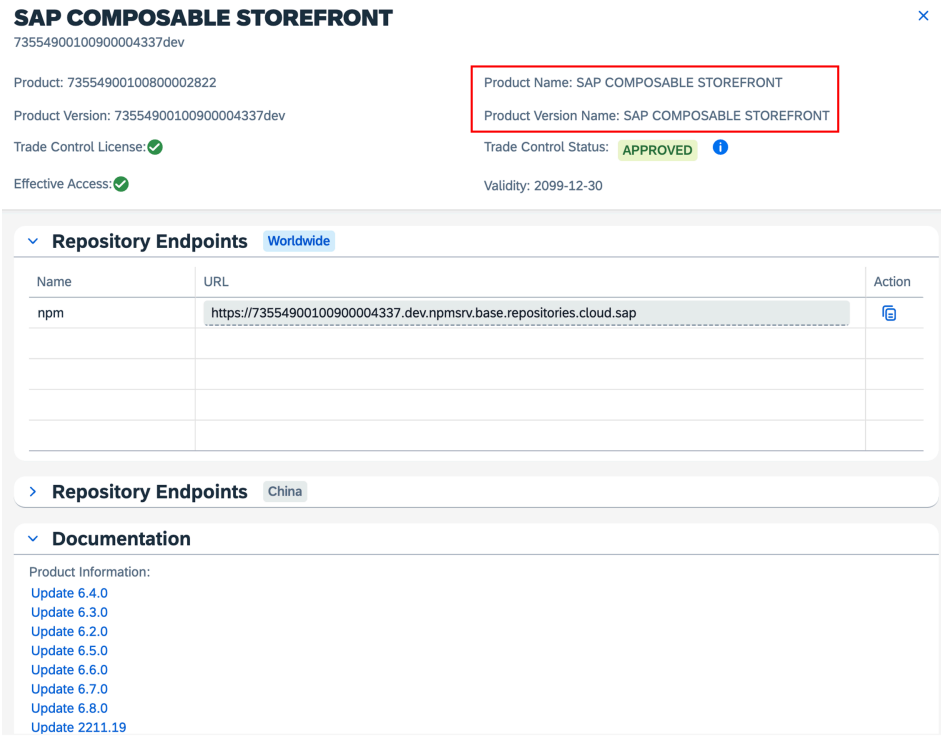
However, we recommend you read through the steps of **parts 2 & 3** so that you understand the process.

2.1 Create an S-user for RBSC with the necessary licenses to download the composable storefront libraries.

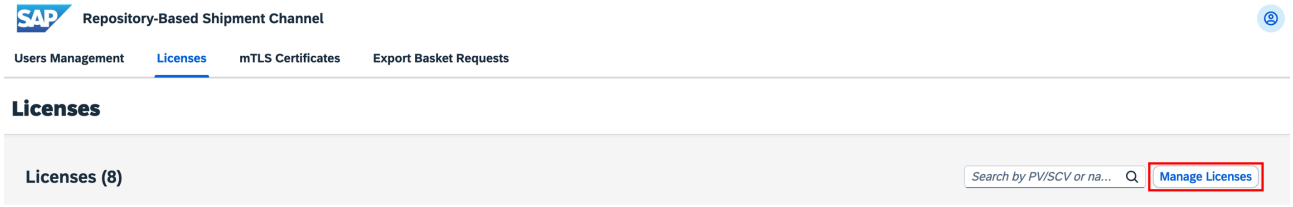
2.2 Log into the RBSC repository with your S-user account at the following web address.

`https://ui.repositories.cloud.sap/www/webapp/users/`

2.3 Click on the "Licenses" tab to check if you have the necessary licenses. For the exercises, you'll need the SAP COMPOSABLE STOREFRONT license. Refer to the screenshot below for details on the SAP COMPOSABLE STOREFRONT license.

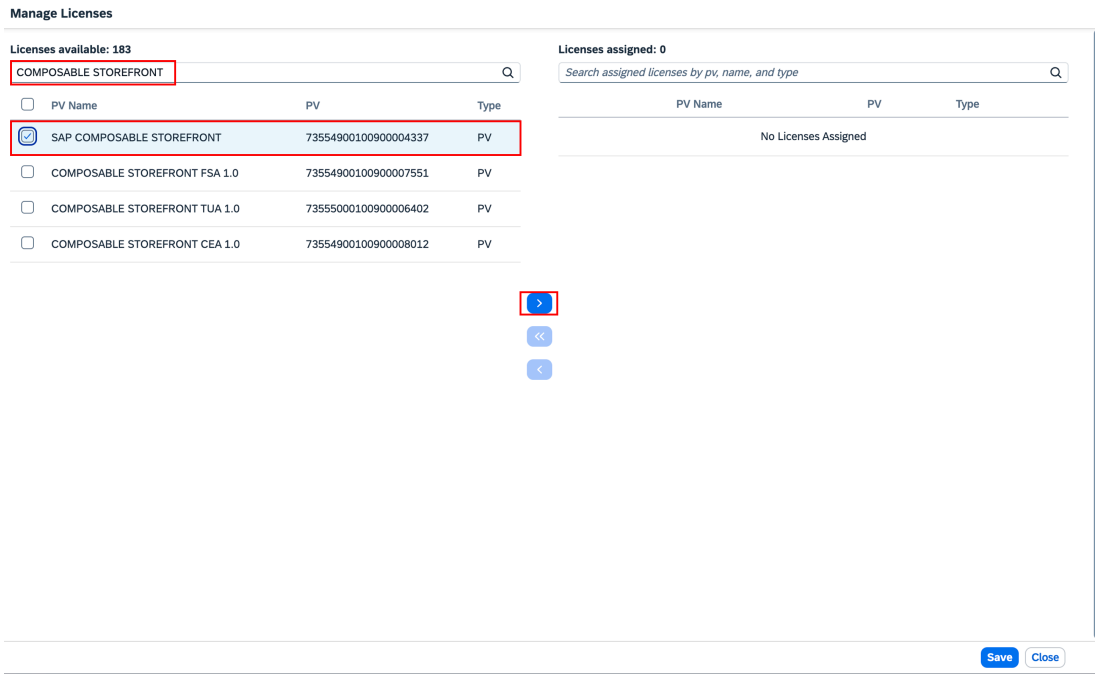


If you don’t see the SAP COMPOSABLE STOREFRONT licenses, click “Manage Licenses” to assign the required licenses.

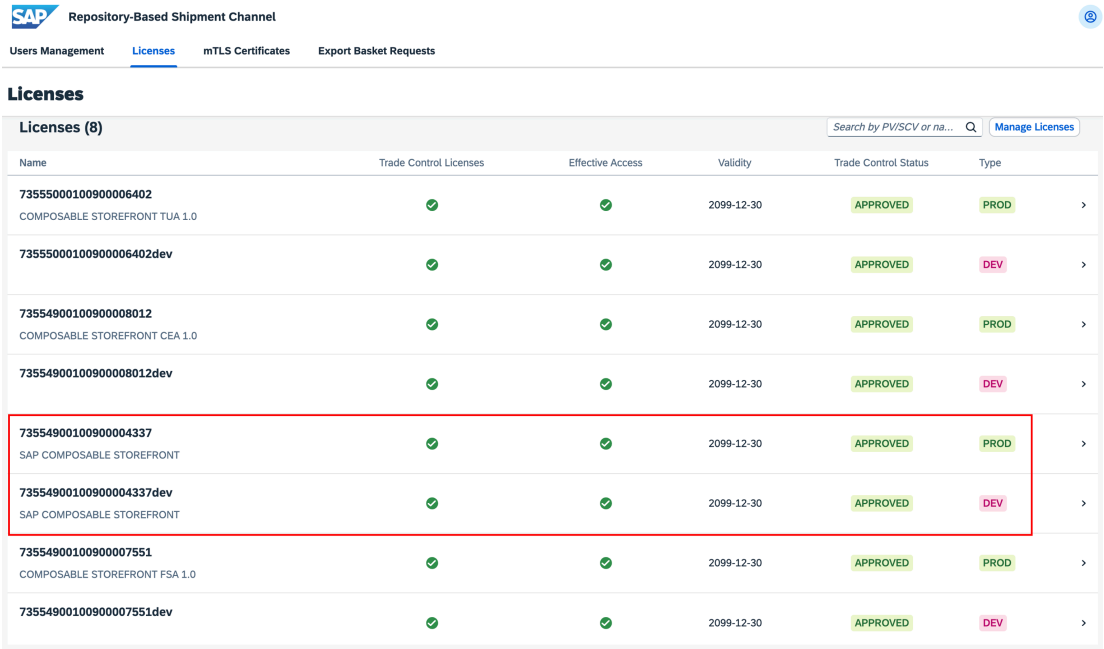


In the Manage Licenses popup, type "COMPOSABLE STOREFRONT" into the search box on the left side and search for available licenses. For the exercises, you only need the SAP COMPOSABLE STOREFRONT licenses. However, if you wish to build other industry accelerator storefronts—such as the Financial Service Accelerator (FSA), Travel and Utility Accelerator (TUA), or Citizen Engagement Accelerator (CEA)—you can assign those licenses as well.

Select the desired licenses, click the '>' button in the middle to assign them, and then click Save.

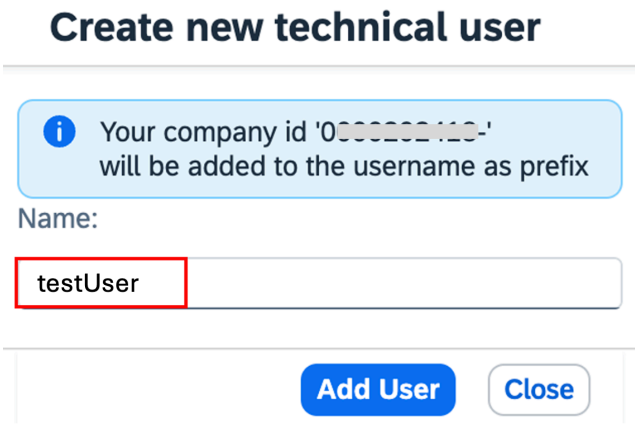


Now, navigate to the Licenses tab to view all assigned licenses.

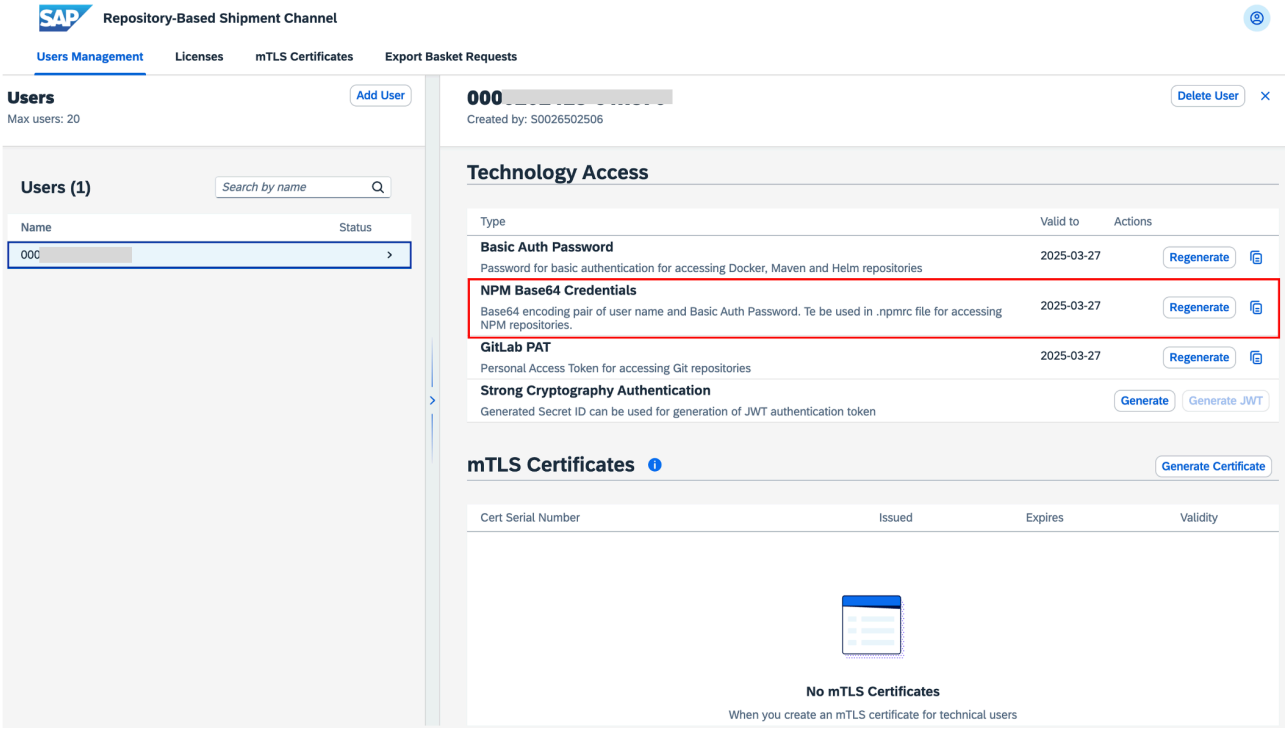


Next, you'll need a technical user to generate the access tokens.

2.4 If a user is not visible in the User Management tab, click on "Add User" to create a technical user.



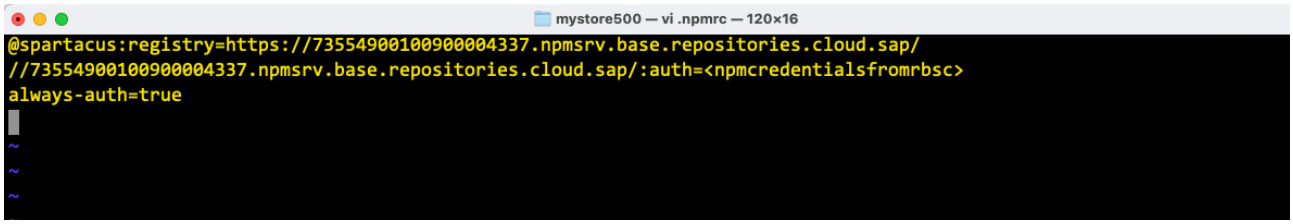
Now that you have added a user to the repository, select the new user in the User Management tab. On the right side, you will see various types of credentials or tokens generated for the user. For the exercises, you will only need the NPM Base64 Credentials to download the composable storefront libraries for your Angular projects.



2.5 At the root of the `mystore500` Angular application you created in the previous steps, create a `.npmrc` file with the following content:

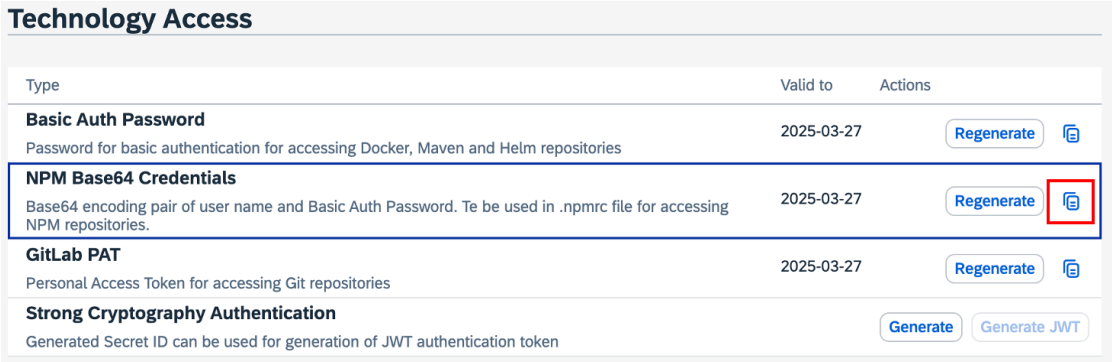
```
@spartacus:registry=https://73554900100900004337.npmsrv.base.repositories.cloud.sap/
//73554900100900004337.npmsrv.base.repositories.cloud.sap/:auth=<npmcredentialsfromrbsc>
always-auth=true
```

Use a plain text editor to create the file and copy the content. You can also do so directly in the Terminal window using `vi` or `nano`.



> **Note**
>
> The two slashes (//) at the start of the second line are required.

2.6 In the User Management tab of the RBSC repository, select the technical user and click on the copy button on the far right side to copy the generated NPM Base64 Credentials.



2.7 In the `.npmrc` file you just created, replace `<npmcredentialsfromrbsc>` with the NPM Base64 Credentials you copied from the RBSC repository, and save the `.npmrc` file.

Part 3 – Setting up composable storefront project

> **Note**
>
> If you didn't perform the part 2 steps due to:
>
> - inability to access the RBSC
> - your RBSC account lacking an appropriate license to download the composable storefront
>
> Then you **can't** execute the steps in **part 3**
>
> Don't worry; if you are using SAP training VM, you can navigate to the ***$WORKSPACE/mystore*** and continue with **part 4**. In the prepared *mystore* folder, **part 2 & 3** have already been performed on your behalf.
>
> **However**, we recommend you read through the steps of **part 3** so that you understand the process.

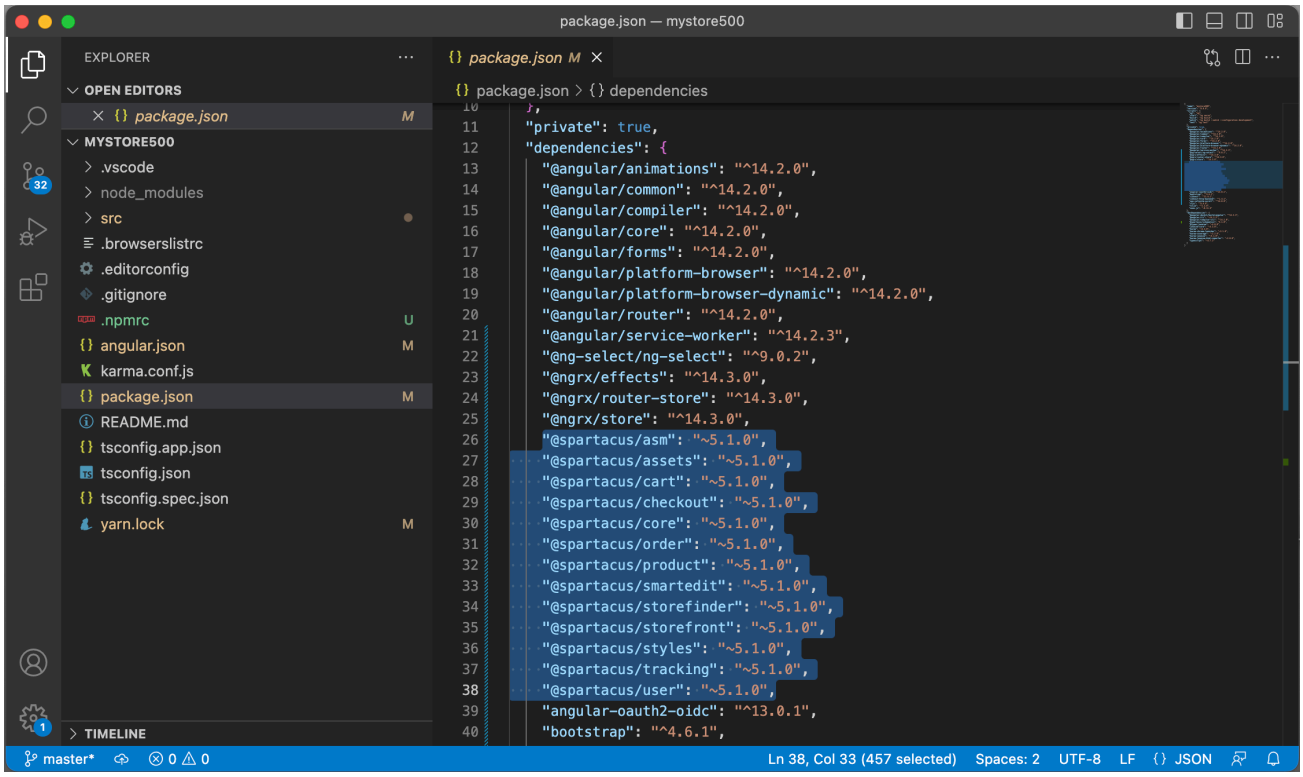
The easiest way to start or configure a new composable storefront project is to use the composable storefront schematics tool to quickly set up your Angular application.

3.1 Make sure you are in the `mystore/mystore500` folder, then execute the following command in the Terminal window:

```
ng add @spartacus/schematics@5.1.0 --no-interactive
```

The schematics tool adds the core composable storefront files and configurations needed to work with the SAP Commerce Cloud sample stores.
We use the --no-interactive flag to bypass the schematics prompts and install the composable storefront with a predefined set of features. Otherwise, you'll be asked to choose which features you would like to set up.

To verify what versions of composable storefront libraries were installed, open the `mystore500` application in Visual Studio Code (on SAP VM, make sure you are in the mystore500/mystore folder, and type `code .` which starts VS code and opens the current folder), open the `package.json` file at the root folder, and look for the string @spartacus.



Note

The composable storefront does not support B2C and B2B storefronts running together in a single storefront application. If you install any of the B2B features, the B2C storefront will load but it will not work properly.

If you select a feature that is for B2B storefronts, the schematics automatically add any required B2B configurations that are missing.

If you install any of the following features, your composable storefront will automatically become a B2B storefront:

- Organization - Administration
- Organization - Order Approval
- Product - Bulk Pricing

- Product Configurator - CPQ Configurator

3.1 (Make sure you are in the `mystore/mystore500` folder.) Install the Angular application dependencies required by the composable storefront with the following command:

```
yarn install
```

3.3 Verify the Base URL and other configurations.

Open the `src\app\spartacus\spartacus-configuration.module.ts` file, and check for any changes you want to make for your setup. Specifically, please check the following configurations (and add them if they are not present):

3.3a `baseUrl`: Points to your SAP Commerce Cloud server, by default it's `https://localhost:9002`.

3.3b `prefix`: Defines the prefix for OCC API calls. The default for composable storefront libraries 5.1 is `/occ/v2/`; this entry is not added by schematics.

3.3c `features.level`: Defines the compatibility level.

3.3d `context`: Defines the site context such as base site, language, and currency.

A sample configuration is as follows:

```
providers: [provideConfig(layoutConfig),
provideConfig(mediaConfig), ...defaultCmsContentProviders, provideConfig(<OccConfig>{
    backend: {
      occ: {
        baseUrl: 'https://localhost:9002',
        prefix: '/occ/v2/',
      }
    },
  }), provideConfig(<SiteContextConfig>{
    context: {
      currency: ['USD','JPY'],
      language: ['en'],
      baseSite: ['electronics-spa'],
      urlParameters: ['currency', 'language', 'baseSite'],
    },
  }), provideConfig(<I18nConfig>{
    i18n: {
      resources: translations,
      chunks: translationChunksConfig,
      fallbackLang: 'en'
    },
  }), provideConfig(<FeaturesConfig>{
    features: {
      level: '5.1'
    }
  })]
```

Part 4 – SmartEdit Integration(optional)

Note

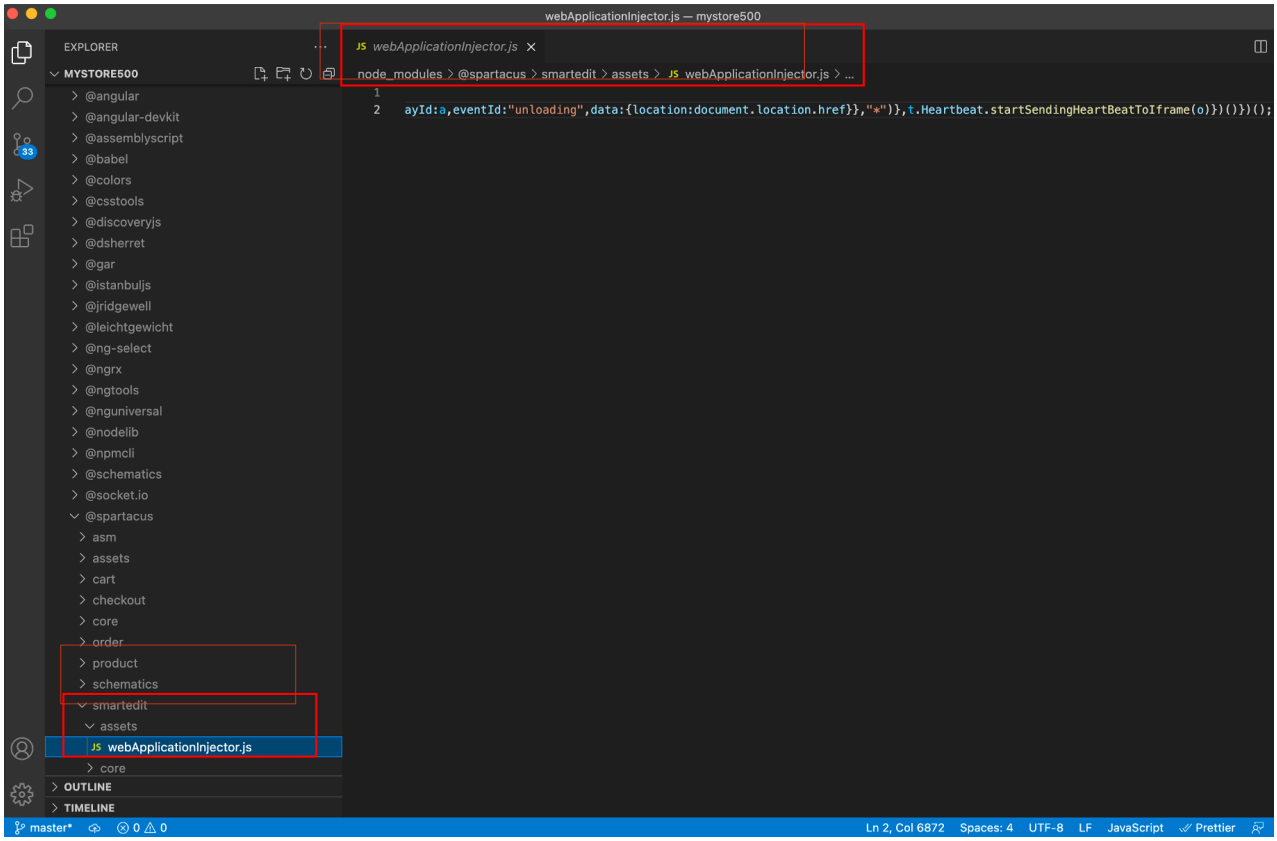
If you didn't perform parts 2 and 3, then **use the folder prepared for you, *$WORKSPACE/mystore***, instead of the ***mystore500*** folder.

If you haven't yet started Visual Studio Code, navigate to the `mystore` folder (or `mystore500` if you performed parts 2 & 3 yourself), and type `code .` to start VS Code and open the current folder.

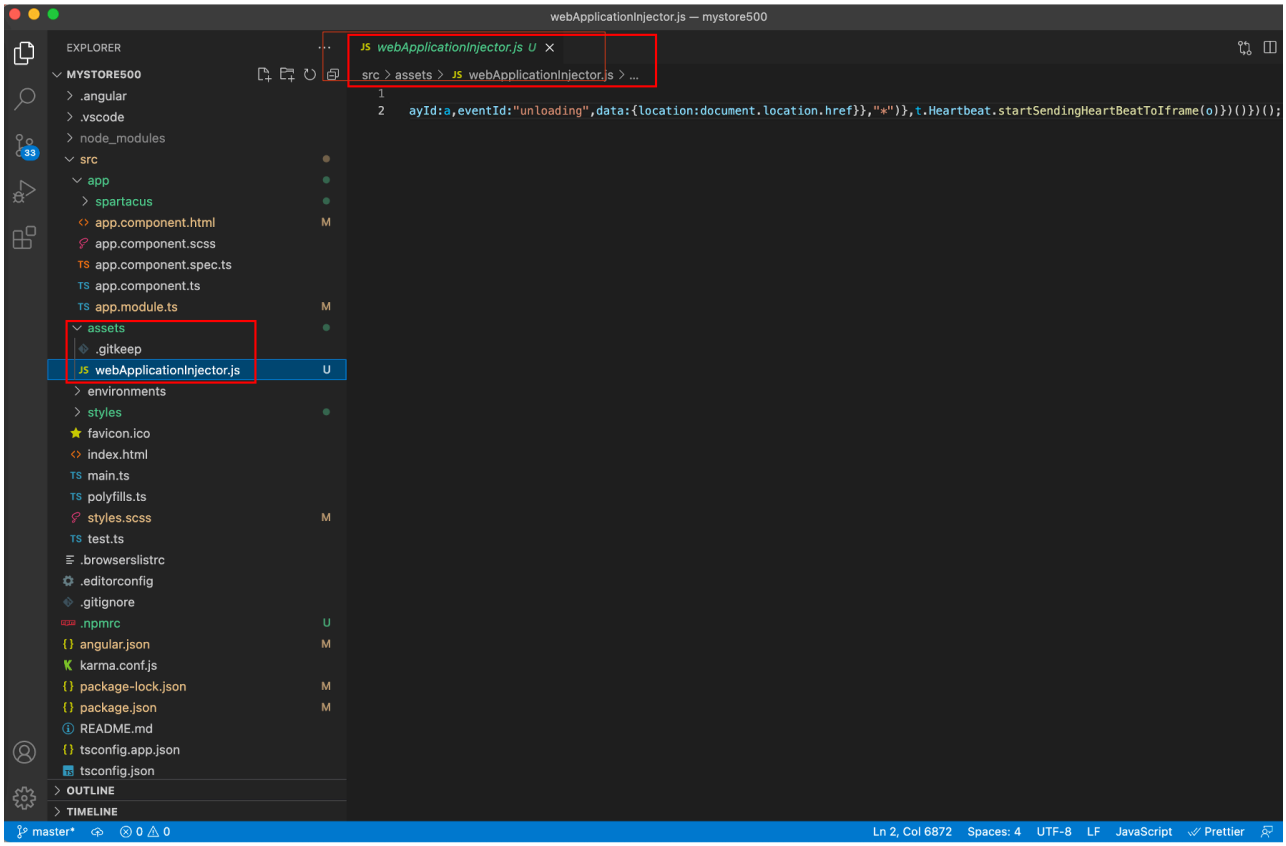
With the Visual Studio Code started and the `mystore500` (or `mystore`) folder open, let's continue.

Your composable storefront application needs to use the `webApplicationInjector.js` file shipped with the version of SAP Commerce Cloud you are using. In the composable storefront's smartedit 5.1 library `@spartacus/smartedit`, the `webApplicationInjector.js` file has been updated to work with SAP Commerce Cloud 2211.

Please copy the `webApplicationInjector.js` file from the `@spartacus/smartedit/assets` library under the `node_modules` folder (it is a very long single line, you see only the end here) …



…and paste it into the `src/assets` folder of the `mystore500` (or `mystore`) folder.


webApplicationInjector.js — mystore500
EXPLORER
webApplicationInjector.js U
src > assets > webApplicationInjector.js > ...
1
2 ayId:a,eventId:"unloading",data:{location:document.location.href}},"*")},t.Heartbeat.startSendingHeartBeatToIframe(o)})()})();
MYSTORE500
.angular
.vscode
node_modules
src
app
spartacus
app.component.html
app.component.scss
app.component.spec.ts
app.component.ts
app.module.ts
assets
.gitkeep
webApplicationInjector.js
environments
styles
favicon.ico
index.html
main.ts
polyfills.ts
styles.scss
test.ts
.browserslistrc
.editorconfig
.gitignore
.npmrc
angular.json
karma.conf.js
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
OUTLINE
TIMELINE
master*
0 0
Ln 2, Col 6872
Spaces: 4
UTF-8
LF
JavaScript
Prettier

Part 5 – Starting Your Composable Storefront Application

Still in the `mystore/mystore500` folder, start your composable storefront application with the following command in the Terminal window:

```
yarn start --ssl
```

Your composable storefront application will be compiled and then started in the Angular Live Development Server. The `--ssl` switch allows the storefront application to be served over the secure HTTPS protocol.

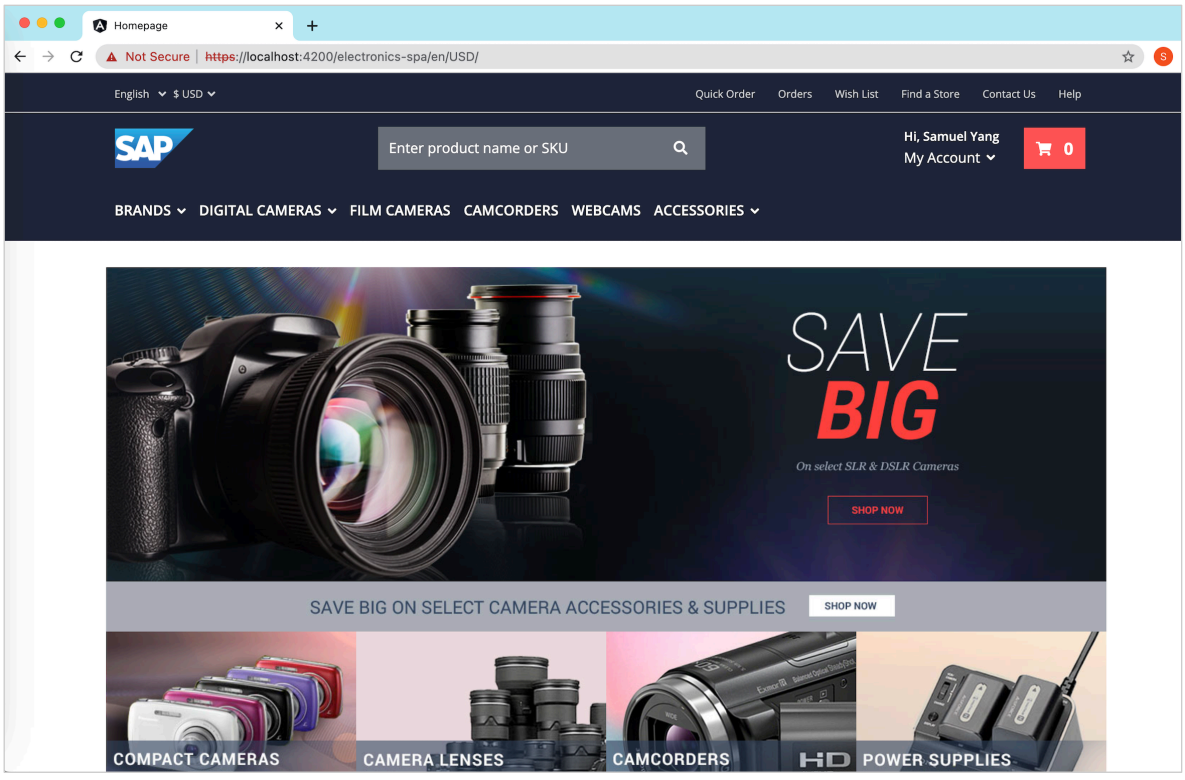
Congratulations! You've built your first composable storefront.

Verify

Visiting the Composable Storefront

Before verifying the composable storefront, you need to make sure the commerce cloud server is up and running. If not, please navigate to `$WORKSPACE/hybris/bin/platform` and execute `./hybrisserver.sh`.

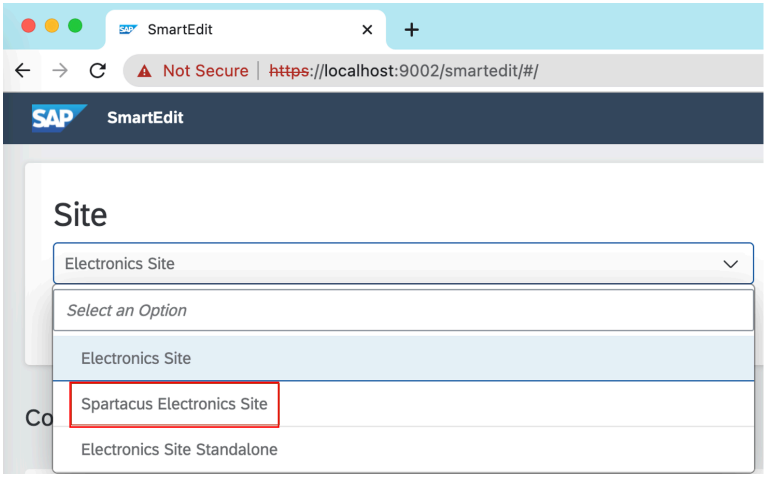
To verify your solution, navigate to https://localhost:4200 in your browser (Make sure you type “https” instead of “http”). If you installed Electronics sample data and the Spartacus sample data extensions, the composable storefront for Electronics should appear.



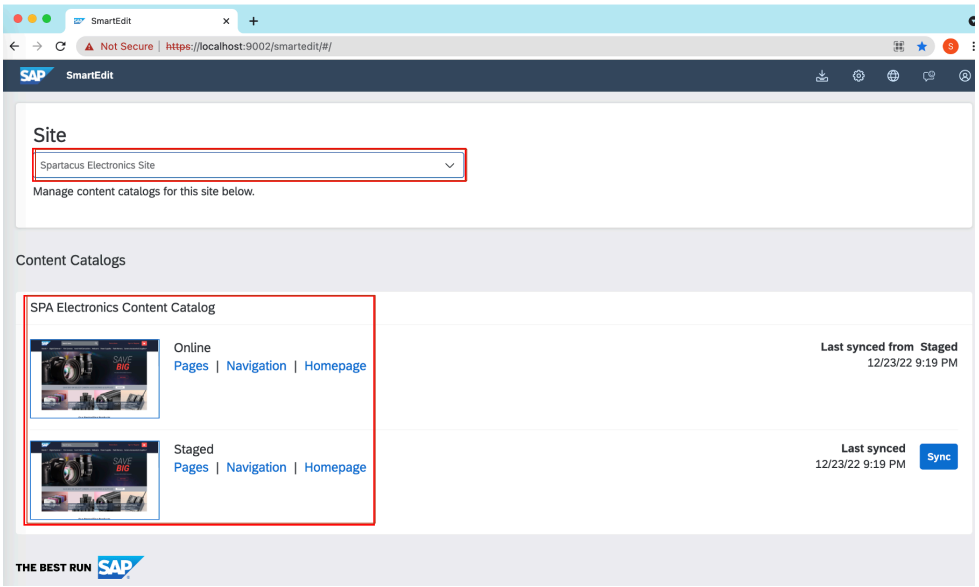
Using SmartEdit

If you followed the steps in **Part 4**, your composable storefront should be integrated with SmartEdit, and you can manage the storefront’s WCMS components in Smartedit, just like you managed the old Accelerator storefront.

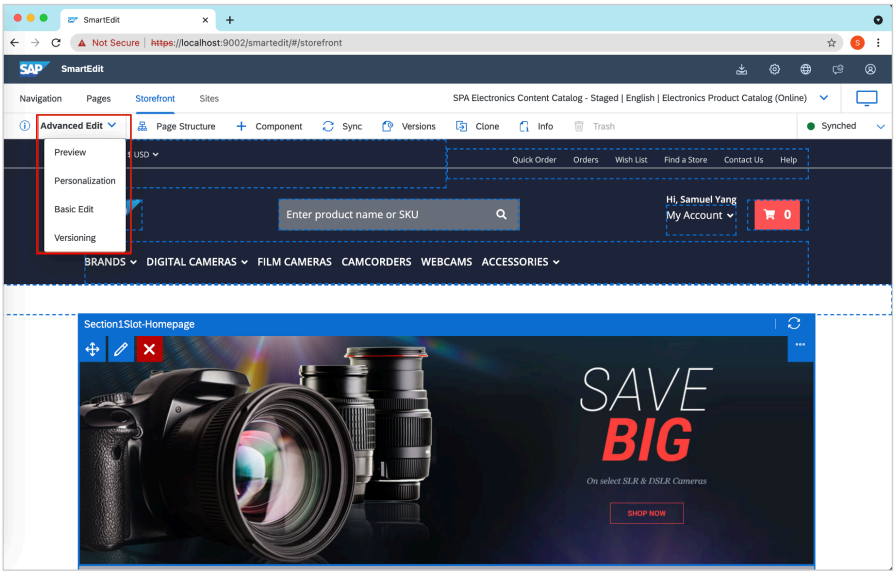
Navigate to https://localhost:9002/smartedit in your browser and login with *admin*|*nimda*.
Select “Spartacus Electronics Site” in the Site dropdown.



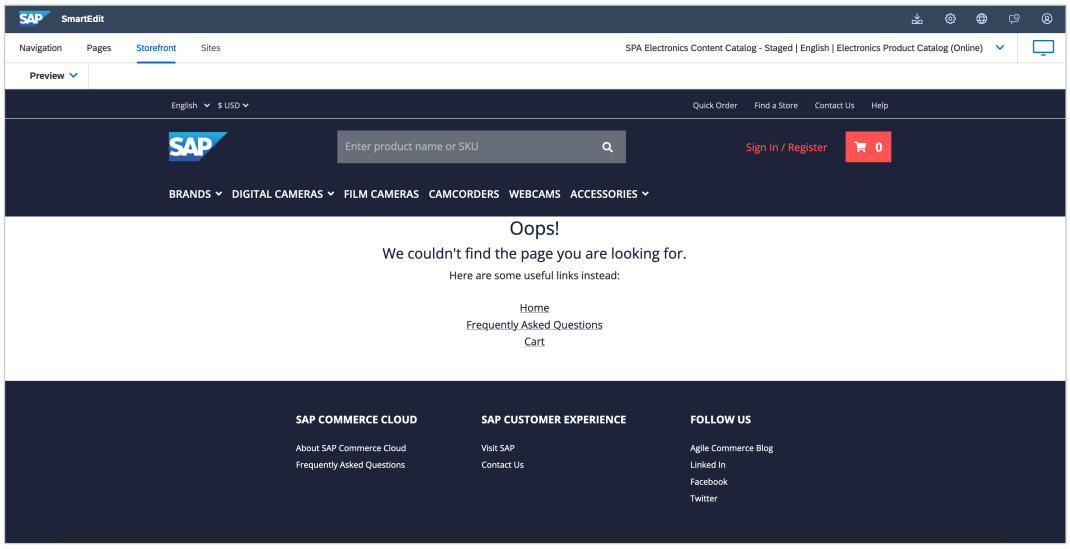
You will see the Staged and Online content catalog versions for the composable electronics storefront:



Click on the **Homepage** in the staged catalog version to see the Homepage’s content slots and components.



If, instead of seeing the homepage, the following appears:

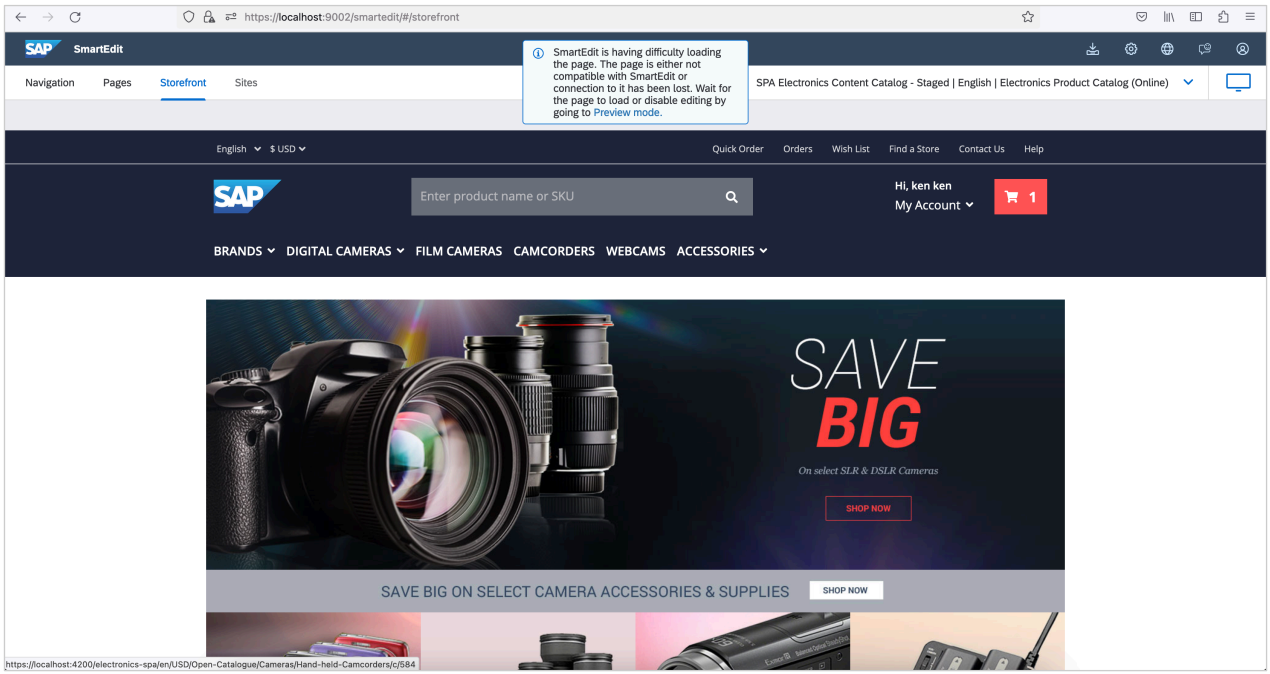


Then no worries, you need to set the "WCMS Cockpit Preview URL" correctly, as follows:

- In Backoffice, navigate to *WCMS* > *Website* > *electronics-spa*, click the *WCMS Properties* tab.
- Set the *WCMS Cockpit Preview URL* to match your composable storefront web site. For example, if your composable storefront homepage url is https://localhost:4200/USD/en/electronics-spa/, set the *WCMS Cockpit Preview URL* to https://localhost:4200/USD/en/electronics-spa/.

After setting the preview URL, please log out of SmartEdit (click on the account management at the top right of the SmartEdit window and click on "Sign out") then log back in with *admin*|*nimda*; when you click on Homepage again, you should now see the homepage.

Optionally, if you can't see the mode's tool bar after clicking on the homepage as shown below, i.e., you can't switch from "Preview" to "Basic edit" mode:



Then, we need to change the placeholder values of `storefrontPreviewRoute` and `allowOrigin`.

In Visual Studio Code, open the `src\app\spartacus\features\smartedit\smart-edit-feature.module.ts` file, update the SmartEdit configuration to look like this:

```
provideConfig(<SmartEditConfig>{
    smartEdit: {
      storefrontPreviewRoute: 'cx-preview',
      allowOrigin: 'localhost:9002',
    },
  })
```

Please logout from SmartEdit, clear your browser cache. Then re-login to SmartEdit and you should see the tool bar now.

If you still can't see the homepage in SmartEdit, you can try the following steps:

- Sign in to SmartEdit as the admin user.
- Click the Settings icon in the top right.
- In the Configuration Editor, scroll down to *whiteListedStorefronts* and add the exact URL of your composable storefront. For this example, it is ["https://localhost:4200"].

The homepage should now appear in SmartEdit.

RECAP

You learned how to setup a composable storefront using the schematics tool and libraries downloaded from the RBSC repository, how to configure the connection with the Commerce Cloud server backend, and integrate it with SmartEdit.

www.sap.com

SAP Customer Experience