Appsus

Your favorite apps in one place

Your challenge is to create a single page application with a set of mini apps:

misterEmail, missKeep, missBooks

The specification for each app is given in a separate document
(missBooks is the app you already have – select one of the team member's projects)

Appsus shall "encapsulate" the different apps and provide a navigation between them. Below is an inspiration photo :)

Time-Table and Delivery Guidelines

Code shall be delivered at your Dropbox as well as in Git and GitHub Page. Fill the URL in the submission form.

Team Lead Meetings:

- Wednesday 15:00-17:00 UX and Code Skeleton – First submission 21:00
- Thursday Status meetings, Second submission 21:00
- Final submission: Saturday 21:00

It is recommended that each team member will be in charge of one of the apps; however, both team members shall be responsible for the successful delivery of the entire project.

Product Definition
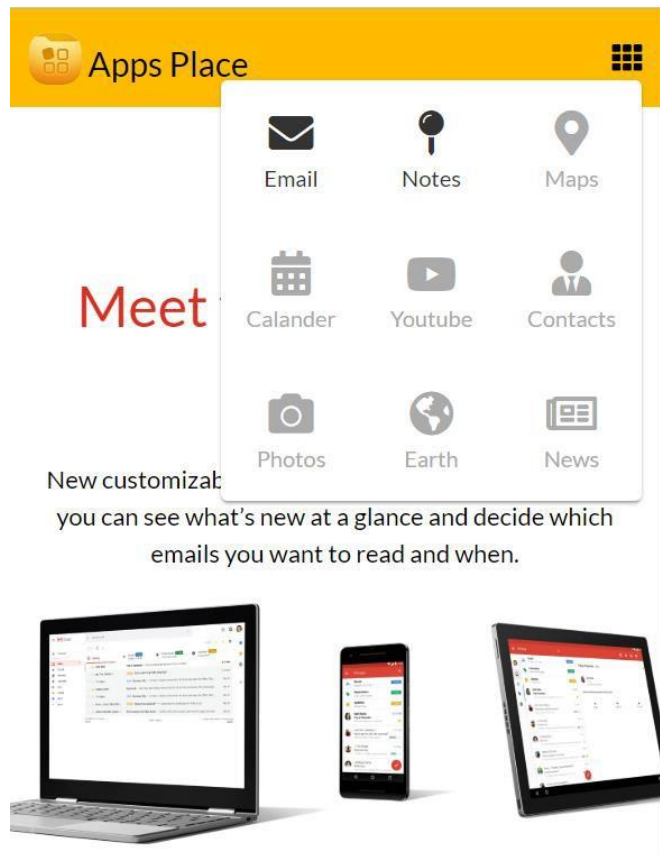
Services load data from localStorage and provide an API with promises.

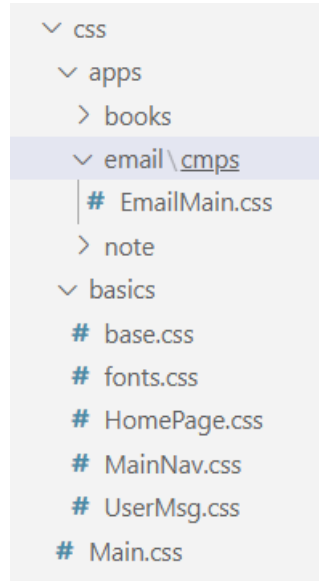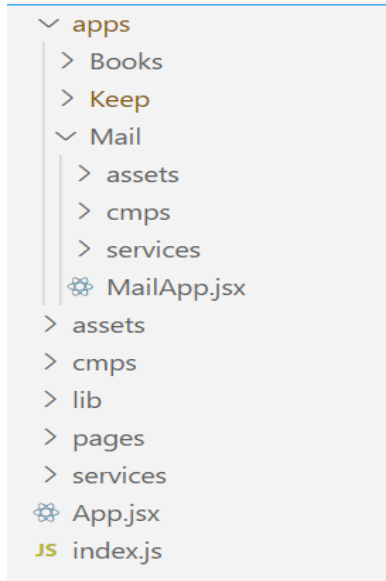The result should be functional, beautiful and responsive.

Learn

- Review together our last CRUD project – Books

Recommended Development Steps

Learn the definition documents and make a plan regarding the structure and working together

Create the following folder structure of the Appsus app:

```
∨ apps
  > Books
  > Keep
  ∨ Mail
    > assets
    > cmps
    > services
    ⚛ MailApp.jsx
  > assets
  > cmps
  > lib
  > pages
  > services
  ⚛ App.jsx
  JS index.js
```

```
∨ css
  ∨ apps
    > books
    ∨ email\cmps
      # EmailMain.css
    > note
  ∨ basics
    # base.css
    # fonts.css
    # HomePage.css
    # MainNav.css
    # UserMsg.css
  # Main.css
```

Setup git and make sure the team can push and pull from the repository

Commit and Push, see it in Github Pages.

Create the app skeleton including home page, app pages, main components, routing, services, etc - so each team member would have a place to work in.

Split and work separately and together whenever you see fit.

Try to push and pull every few hours to coordinate your code and practice the workflow/ Do the first push/pull together

Use this opportunity to improve the code by mutual code reviews

Invest in your test data

It is required to create some meaningful data, so when the app starts, it shows relevant demo data.

eventBus communication

When components are located at different areas of the DOM we sometimes use an eventBus to communicate. Do note that you cannot communicate with a component that does not exist (e.g. – you cannot communicate with component on different route)

queryString Params

A query string is the part of a uniform resource locator (URL) which assigns values to specified parameters:

```
http://example.com/path/to/page?petName=charli&color=black
```

(in this case the param `petName` gets the value *charli,* and the `color` gets *black*)

Query string parameters are a useful way to communicate between routes adding optional parameters. Remember them when integrating the 2 apps.

For example: a link: 'Send as Email' in a <NotePreview> component to the <ComposeEmail> component may look like that:
/email/compose?subject=my note&body= note about the rain