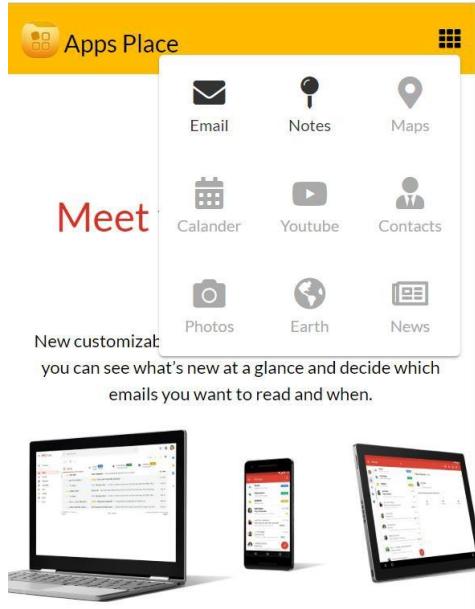




Appsus

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. The result should be functional, beautiful, and responsive.

Time-Table and Delivery Guidelines

Code shall be delivered in Git and GitHub Page.

Each team member should lead one app, but both are accountable for the project's overall success.



Meetings with the team leader

- Meeting 1 - UX and Code Skeleton
- Meeting 2 - Status meetings
- Meeting 3 – Project presentation

Recommended Development Steps

- Review together our last in-class project: cars-app.
- Learn the requirements and research the real apps.
- Create the app skeleton as described below - so each team member would have a place to work in.
- Setup git and make sure the team can push and pull from the repository, setup Github Pages.
- Decide how to work together, split and work separately and together whenever you see fit.
- Push and pull every few hours to coordinate your code and practice the workflow.
- Use this opportunity to improve the code by mutual code reviews.

Folder Structure

Let's review the folder structure for the Appsus app.

Reusable Components

Here are some components that are used by both apps:

- <LongTxt> - gets the text to format as a prop and provides a more/less functionality
- <UserMsg> - Used for showing success / error messages
- <LabelPicker> - Used for picking multiple labels from a (fixed) list of labels

Critical	✓
Family	✓
Work	✓
Friends	✓
Spam	✓
Memories	✓
Romantic	✓

Show the selected labels on the email / note



LABELS

Critical

Work

Invest in your demo data

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

queryParams

A query-param 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-params are a useful way to communicate between routes adding optional parameters. Remember them when integrating the apps.

For example: a link: 'Send as Email' in a `<NotePreview>` component to the `<EmailCompose>` component may look like that:

/email/compose?subject=my note&body= note about the rain

Here is an example, the following link adds an event in google calendar:

`https://calendar.google.com/calendar/render?action=TEMPLATE&text=My Event&details=Event description text&dates=20260305T103000/20260305T184500&location=Tel Aviv`



How the customer explained it



How the project leader understood it



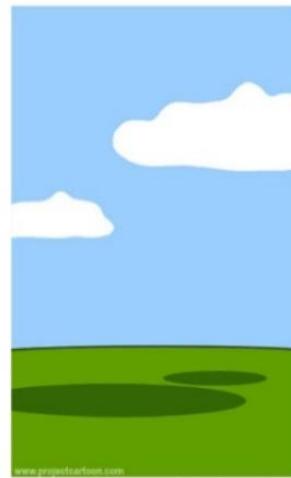
How the analyst designed it



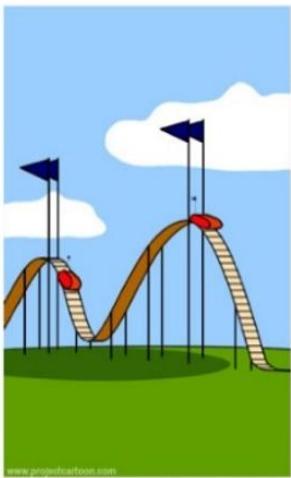
How the business consultant described it



How the programmer wrote it



How the project was documented



How the customer was billed



What the customer really needed