

The misterEmail App

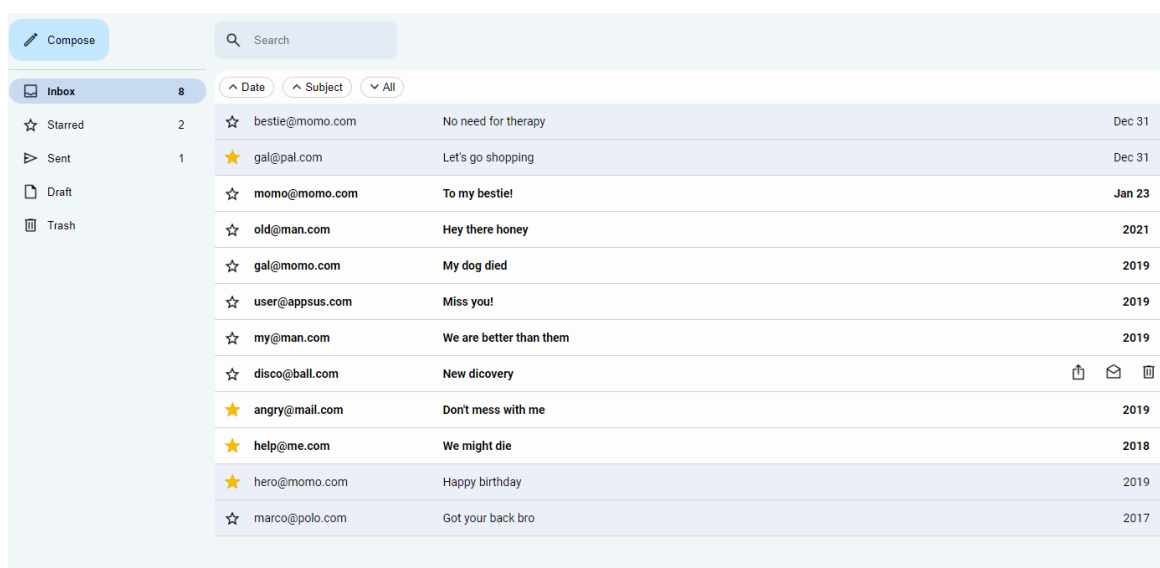
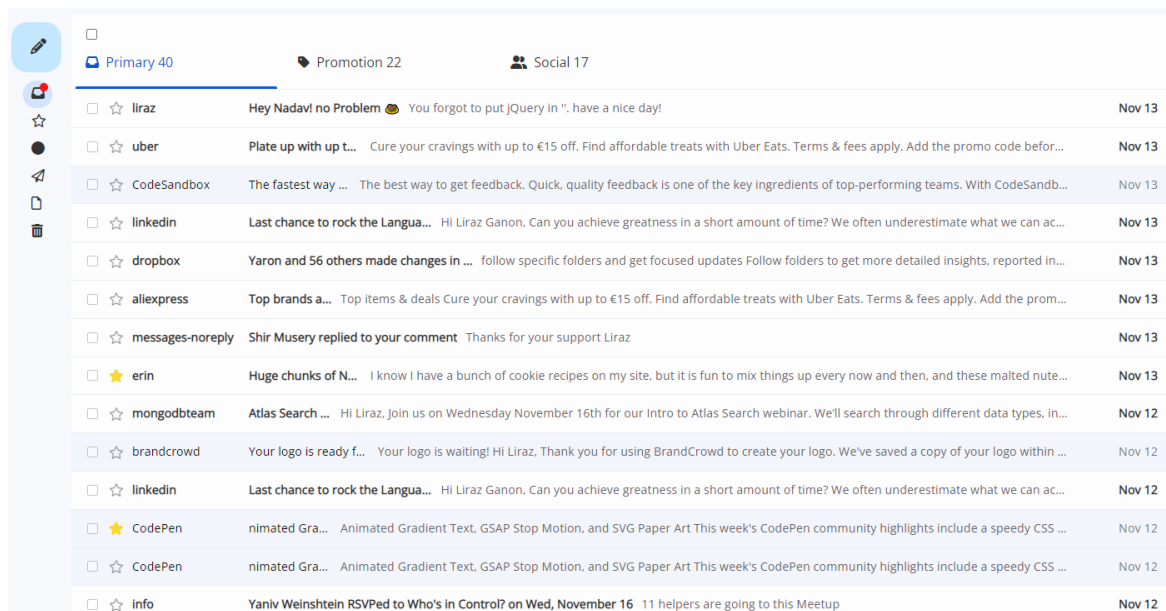
Requirements

Build an email client app – based on Gmail UI and UX.

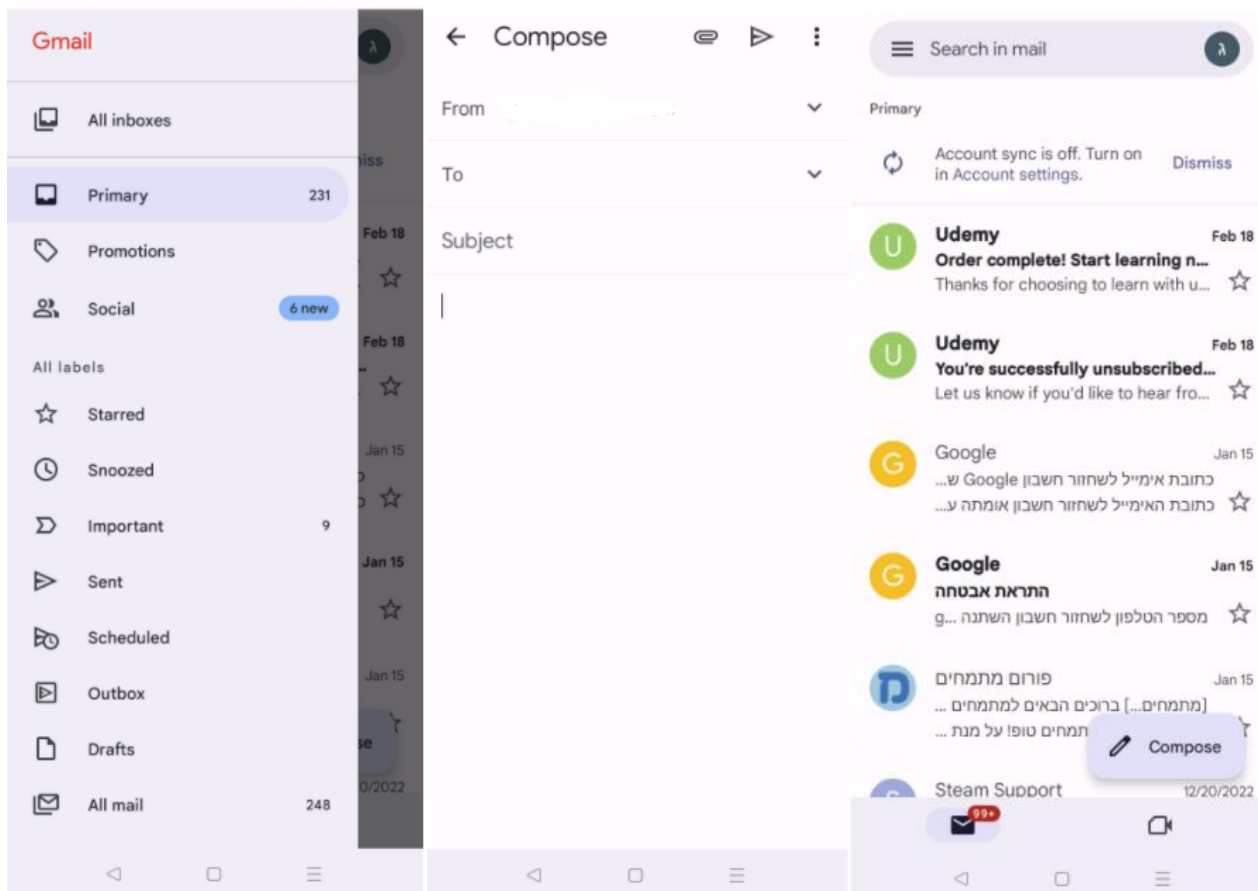
The user will be able to view his inbox and other folders, to look into an email and to compose new emails.

This app will not work with a real server, so emails are not really sent, but the user experience will feel real.

Initial UX








Mobile Version



Some more UX Ideas

Hover states

★ Google	Security alert	- Your Google Account was just signed in to from a n...	 
★ Coding Academy	Wow your app is so amazing!	- WOWWWWWW! This is such an amazing app, Nadav and E...	Jun 12

<input type="checkbox"/> ☆ CodeSandbox	The fastest way ...	The best way to get feedback. Quick, quality feedback is one of the key ingredients of top-performing teams. With CodeSandt	  
<input type="checkbox"/> ☆ linkedin	Last chance to rock the Langua...	Hi Liraz Ganon, Can you achieve greatness in a short amount of time? We often underestimate what we can ac...	Nov 13

Email list

Compose

Inbox 2

Starred

Sent

Drafts

Trash

☆ TED

Eyal, here's why you should watch this talk...

Apr 06

☆ Shoes

Say YES to a discount

Jun 25

☆ eBay

Eyal, new discounts!

Jun 25

☆ Wizz Air

Say YES to a discount

Jun 24

★ Udacity

Use AWS ElastiCache for data storage

Jun 24

☆ Netflix

Eyal, The Politician Season 2 is now on Netflix

Jun 24

Eyal, The Politician Season 2 is now on Netflix

⌵ ↻ ⌵ ⌵

Netflix

<netflix@netflix.net>

This message was mailed to [eyalsk118@gmail.com] by Netflix as part of your Netflix membership. SRC: 12182_en_IL

★ Google

Security alert

Jun 13

★ Coding Academy

Wow your app is so amazing!

Jun 12

emailService

- Model - start with a basic model of emails:

```
const email = {
  id: 'e101',
  subject: 'Miss you!',
  body: 'Would love to catch up sometimes',
  isRead: false,
  isStarred: false,
  sentAt : 1551133930594,
  removedAt : null, //for later use
  from: 'momo@momo.com',
  to: 'user@appsus.com'
}
```

Also, in your email service have a basic user:

```
const loggedInUser = {
  email: 'user@appsus.com',
  fullname: 'Mahatma Appsus'
}
```

Note: this user is hard coded and there is no need to build a login system.

In the **inbox** show all the emails that were sent to the current user, the other emails are **sent** emails and should be displayed at the *sent* folder

The emailService query function should get a criteria(filterBy) object, here is an idea:

```
const filterBy = {
  status: 'inbox/sent/star/trash',
  txt: 'puki', // no need to support complex text search
  isRead: true/false/null, // (optional property, if missing: show all)
}
```

<EmailIndex>

- Gets [emails](#) to show from the service (async) Renders the list and the filter components (both top filter with search, and side filter for different *folders*)

<EmailList>

- Renders a list of [<EmailPreview>](#) pass down an email prop.
- For each EmailPreview render action buttons.

<EmailPreview>

- Present an email preview
- Renders the subject (with text size limit)
- Gives visual indication for read/unread
- Support hover state

<EmailFilter>

- Allow filtering (Start with Search and Read / Unread)

<EmailFolderList>

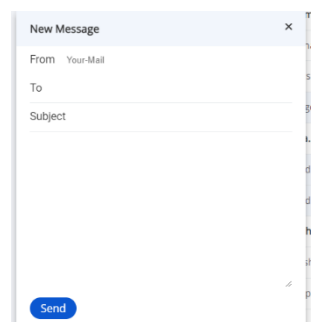
- Allow filtering the emails between different folders (inbox, sent, starred, trash etc..)

<EmailDetails>

- A nested routable component (page) ("/email/:mailId")
- show the entire email
- Allow deleting an email (using the service)
- Allow navigating back to the list

<EmailCompose>

- A component that react to query-string-params ("http:xxx/?[compse=new](#)")
- Has a form with: to, subject and body
- Use the service to send an email (add an email to the list)



Part 1 – React Router + CRUDL

- Create a basic project from the provided starter, make some changes to the title, colors, etc...
- Setup a git repository and deploy your project to GitHub pages
- Remember to make regular commits of your work with meaningful comments
- Once in a while, push to github and check your app from your mobile device
- Make it look Pixel Perfect like the real Gmail App. (Using the devtools to get the right colors, font, sizes and more..)

Development flow

- **Creating main pages**

Use React Router as shown in class to create 3 "pages"

<HomePage> – a simple welcome page

<AboutUs> – try experimenting a bit...

<EmailIndex> – the app we will be building: a CRUDL for emails

- Create a **bookService** which uses the asyncStorageService
- Based on the reference project - build the following components:
 - <EmailIndex>** - renders the filter and the list
 - <EmailList>** - Renders a list of <EmailPreview> components
 - <EmailPreview>** - a preview with basic email details
 - <EmailDetails>** - full details of a specific email
 - <EmailFilter>** - allow the user to filter the emails by text & isRead
 - <EmailCompose>** - (Bonus component) allow the user to send emails using a form.

Start with a simple form which has inputs for the subject and the body of the email and hard code the rest of the data.

Features List

- Display a list of emails (such as inbox)
- Click on an email-preview – opens the email for reading (go to details)
- Give visual indication for read/unread in the email preview, and support hover state (show buttons instead of date)
- Allow removing an email (start by deleting the email from the list)
- Support *staring/unstarring* an email
- Support manually marking an email as read/unread
- Filter emails: by text in the subject or body and by read/unread (use select “read/unread/all”)
- **BONUS:** show a progress bar that displays the unread emails count
- **BONUS:** Compose – create a new email and send it
- **BONUS:** Allow sorting the emails by date and by title
- **BONUS:** Allow more options for emails sorting
- **BONUS:** Allow more options for emails filtering(From, Subject, Date, Has words, etc.) (Filter Modal)

Part 2 – EventBus, Dynamic, Add&Edit

- Continue the work on your email project.
- Remember to make regular commits of your work with meaningful comments. push to github and check your app from your mobile device.

Feature list

- **Folder List** - Allow filtering by different folders: inbox / sent / trash / star
Show an indication for the current folder the user is on.
(params - /mail/:folder)

<EmailFolderList>

- **Inbox** : Show only received emails.
- **Starred** : Show only starred emails in the “starred” folder
- **Sent** : Show only sent emails in the “sent” folder
- **Trash** : When removing an email (e.g. from inbox) it should be displayed in the trash folder (use the removedAt field).
When removing from trash – email gets truly deleted.
- **Drafts**

- **Mail Compose** – create a new email and send it.

<EmailCompose>

should be a rendered according to queryStringParams inside our MailIndex. tip ‘/mail/:folder?compose=new’

Keep in mind that we have 3 different view states (minimized, normal, fullscreen). tip use one piece of state as a string “viewState”

- **Unread count** – show the unread emails count next to the inbox folder
This will require another state (unreadCount) because we filter our emails in the service (aka demo backend).



- **User Msg** – Show a success / error msg to give the user an indication for an action he did (When necessary)
Use **eventBusService** to create communication through out the app.

<UserMsg>

- **SearchParams** – render your **filter** to your app URL.
When user filters it is always shown as query string params – and when we refresh the filter will be determined by the URL.
Example url: /mail/inbox?txt=something&date=15/05/2022
Now do the same for **sort**.
-

- **DRAFT** - Add a draft folder – email that is being composed is auto saved 5 seconds after each time a user changes the email - and can be viewed in the draft folder.
 - **Edit Draft** – Allow the user to edit draft mails and send them as mails. (Will be shown in the sent folder) use **<EmailCompose>** to do so.
-

Finish previous tasks – everything that's left from part 1

- **Filter Modal** - Allow more options for emails filtering(From, Subject, Date, Has words, and any other real Gmail options)
 - **Sort** - Allow sorting the emails by date and by title (and any other real Gmail options)
 - **BONUS : Responsive Web App** – style your app to look like GMAIL on desktop and on a native APP. (using media queries css)
-

Part 3 – React Awesome&Custom

- Continue the work on your email project.
- **Finish previous tasks.**
- Make your app Responsive to mobile and desktop.

New tasks + Features

- **Quick Send (Help)**– Show a link at the home page like so
`<Link to={"/mail/inbox?compose=new&to=help@gmail.com&subject=Help"}>`
When clicked – the user will be redirected to the mail app, and we will use the searchParams to populate the form inside the EmailCompose component.
- **PropTypes Validation**
Add types validations to relevant components.
- **Animations**
Use animate.css to create animations where ever you see fit.
- **Custom Hooks <EmailFilter />**
Use custom hooks in your project – (useEffectUpdate, useForm)
- **NPM Libraries**
 - **FORMIK&MUI**
Upgrade your <EmailCompose /> with FORMIK & MaterialUI
 - **MAPS**
Allow the user to add his location when he sends the mail.
Show a map inside the inbox with the current location of the sender.
use google-map-react (npm)
 - **DASHBOARD (page)**
Create a dashboard page showing statics of your emails.
- **Responsive Web App** – style your app to look like GMAIL on desktop and on a native APP. (using media queries css)

- **BONUS : Multi Select Functionality** – add a checkbox next to each MailPreview -> when clicking it allows selecting the mails.
(selectedMails) . then allow the user to remove \ set as read\unread \ star\unstarred for the selected mails.
 - **BONUS :** Explore and use as many NPM libraries as you'd like.
 - **BONUS :** Use children where-ever you see fit.
 - **BONUS :** Explore some more custom hooks.
-
-