

TY B.Tech Computer Engineering

2018 Course

CLOUD COMPUTING SCE



SUBMITTED TO:

PROF. VISHAL MESHRAM

**VISHWAKARMA INSTITUTE OF INFORMATION
TECHNOLOGY, PUNE**

COMPUTER ENGINEERING DEPARTMENT

Anish Mudaliar 321002 21810945

Amlan Nanda 321032 21810954

Amey Bobade 321007 21810866

Pritesht Bhutada 321042 21810832

So in our SCE we have discussed about 3 Cloud Services:

1. Hosting a website using MICROSOFT AZURE

At its core, **Azure** is a public cloud computing platform—with solutions including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) that can be **used for** services such as analytics, virtual computing, storage, networking, and much more.

2. Login Portal Authentication using GOOGLE FIREBASE

Firebase is a Backend-as-a-Service (BaaS). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. **Firebase** is categorized as a NoSQL database program, which stores data in JSON-like documents.

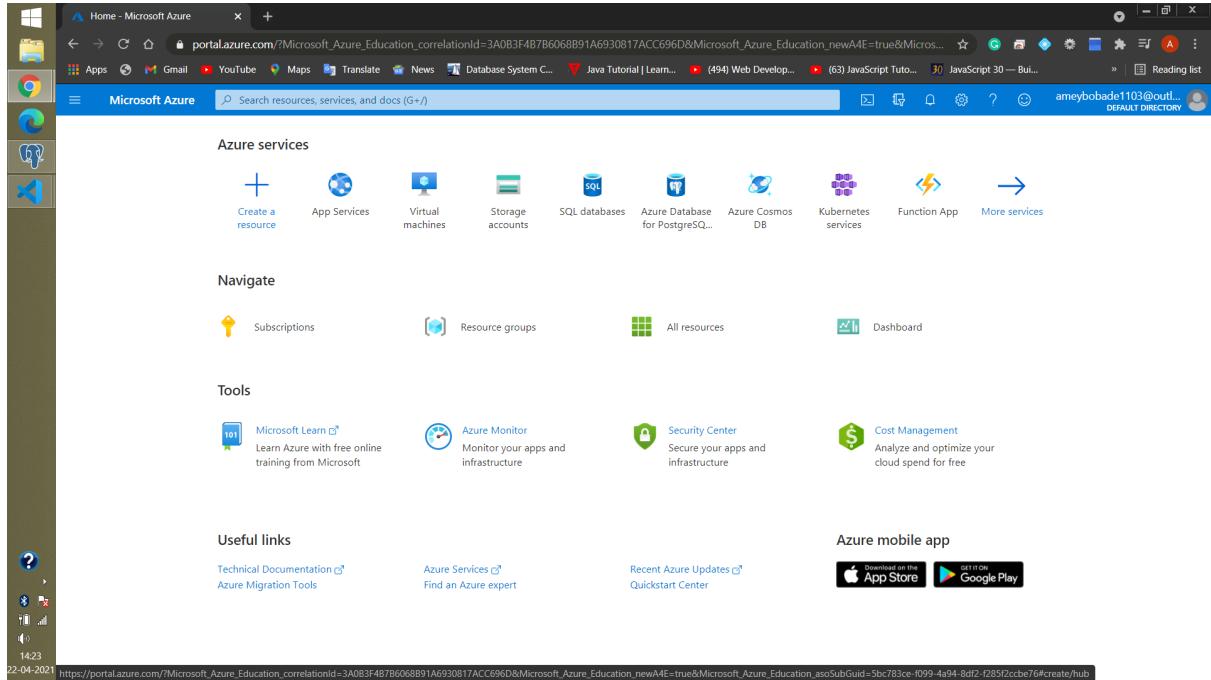
3. Custom app using AMAZON HONEYCODE

Amazon Honeycode is a fully managed service that allows you to quickly build mobile and web apps for teams—without programming. Build **Amazon Honeycode** apps for managing almost anything, like projects, customers, operations, approvals, resources, and even your team.

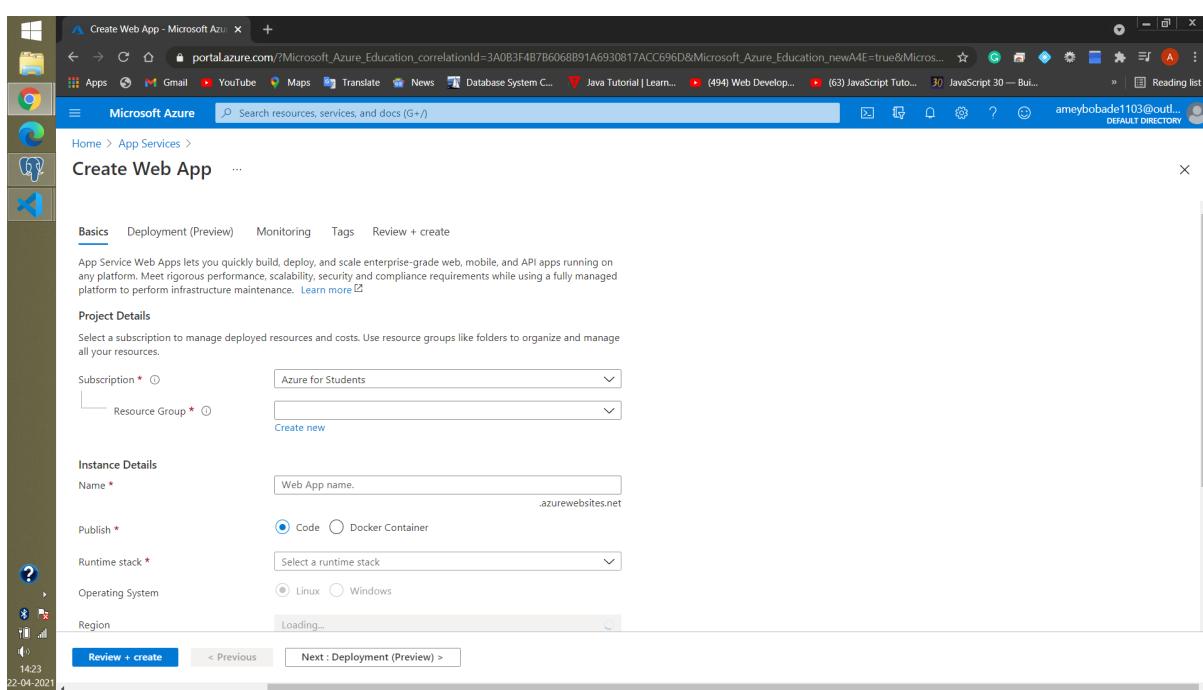
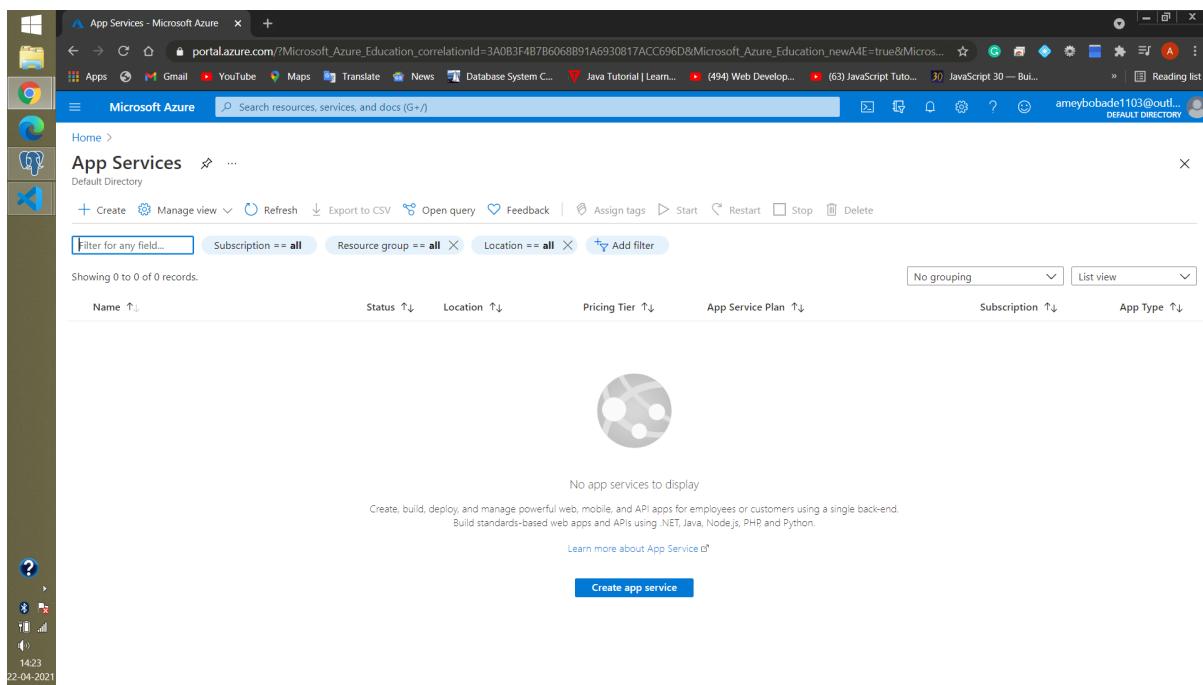


1.Hosting a website using Microsoft Azure

This is the Azure portal where we can view and manage all of your applications in one unified hub—including web apps, databases, virtual machines, virtual networks, storage, and Visual Studio team projects. Enjoy the flexibility of using the Azure portal's graphical experience or the integrated command-line experience provided by Cloud Shell.



Azure Web Apps is a cloud computing based platform for hosting websites, created and operated by Microsoft. It is a platform as a service which allows publishing Web apps running on multiple frameworks and written in different programming languages, including Microsoft proprietary ones and 3rd party ones.



So firstly we'll have to create a azure Storage account. Fill in the given options as we like it. An **Azure storage account** contains all of your **Azure Storage** data objects: blobs, files, queues, tables, and disks. The **storage account** provides a unique namespace for your **Azure Storage** data that is accessible from anywhere in the world over HTTP or HTTPS.

A **resource group** is a container that holds related **resources** for an **Azure** solution. The **resource group** can include all the **resources** for the solution, or only those **resources** that you want to manage as a **group**.

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below.

[Learn more about Azure storage accounts](#)

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription * Resource group *

Instance details

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

Storage account name *
Location * Standard Premium

[Review + create](#) < Previous Next : Networking >

The storage account name has to be unique in the world. A **storage account** provides a unique namespace in **Azure** for your data. Every object that you store in **Azure Storage** has an address that includes your unique **account name**. The combination of the **account name** and the **Azure Storage** service endpoint forms the endpoints for your **storage account**.

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription * Resource group *

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

Storage account name * Location * Standard Premium

Account kind * Replication *

[Review + create](#) < Previous Next : Networking >

Keep going on next on each tab on default options itself

Now the storage account is created and is ready to be deployed.

The screenshot shows the 'Create storage account' wizard in the Microsoft Azure portal. The 'Review + create' tab is selected. A green validation message 'Validation passed' is displayed. The 'Basics' section contains the following configuration:

Subscription	Azure for Students
Resource group	Env
Location	East US
Storage account name	rascalculatorwebsite
Deployment model	Resource manager
Account kind	StorageV2 (general purpose v2)
Replication	Read-access geo-redundant storage (RA-GRS)
Performance	Standard

The 'Networking' section shows:

Connectivity method	Public endpoint (all networks)
Default routing tier	Microsoft network routing

The 'Data protection' section shows:

Point-in-time restore	Disabled
-----------------------	----------

At the bottom, there are 'Create', '< Previous', 'Next >', and 'Download a template for automation' buttons.

The screenshot shows the 'Create storage account' wizard in the Microsoft Azure portal. The 'Review + create' tab is selected. A green validation message 'Validation passed' is displayed. The 'Data protection' section contains the following configuration:

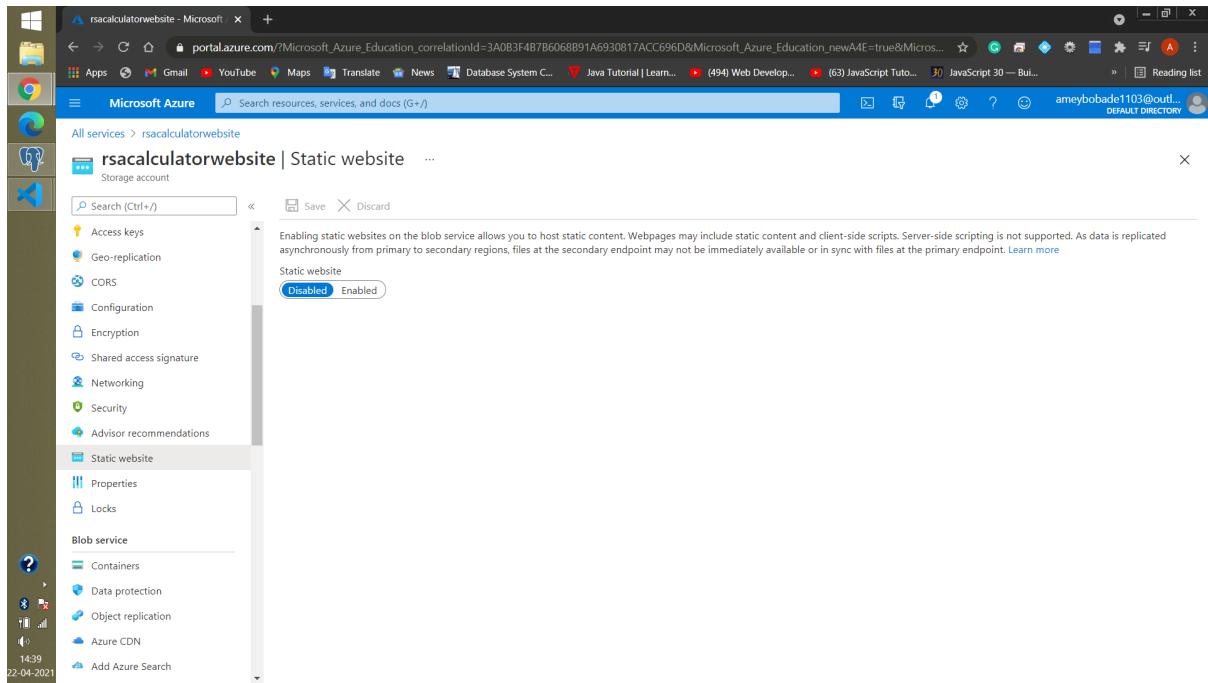
Point-in-time restore	Disabled
Blob soft delete	Disabled
Container soft delete	Disabled
File share soft delete	Disabled
Blob change feed	Disabled
Versioning	Disabled

The 'Advanced' section contains the following configuration:

Secure transfer required	Enabled
Allow storage account key access	Enabled
Minimum TLS version	Version 1.2
Infrastructure encryption	Disabled
Allow Blob public access	Enabled
Blob access tier (default)	Hot
NFS v3	Disabled
Hierarchical namespace	Disabled
Large file shares	Disabled
Customer-managed keys support	Disabled

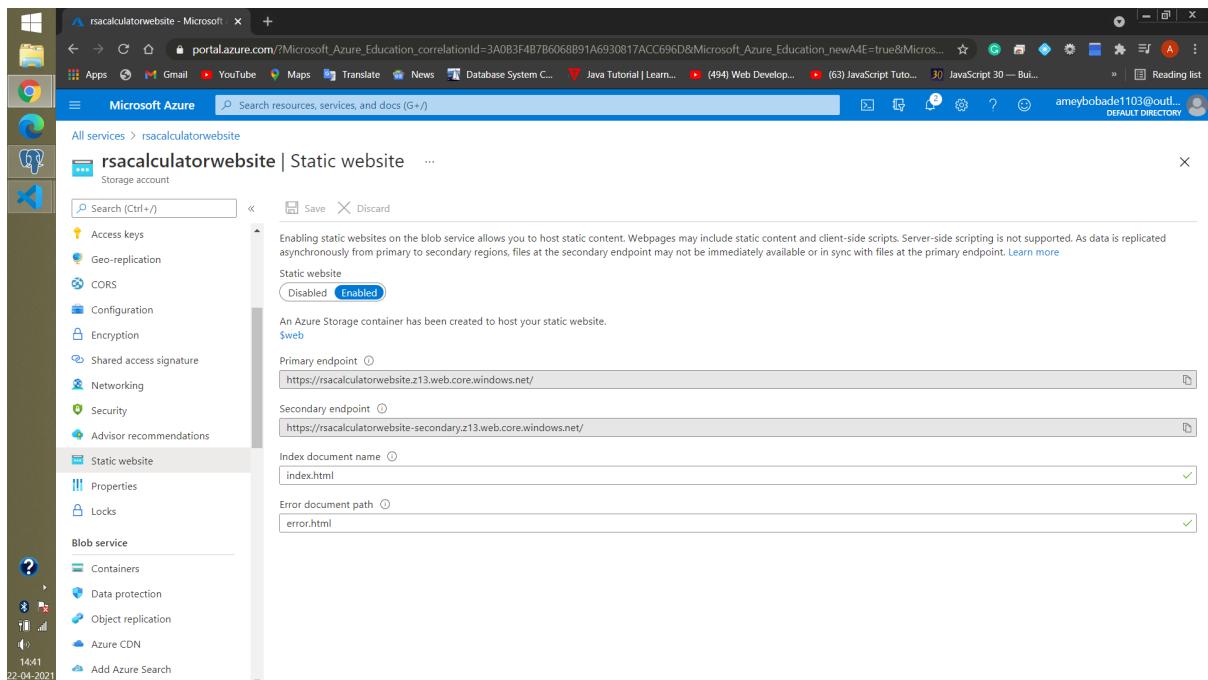
At the bottom, there are 'Create', '< Previous', 'Next >', and 'Download a template for automation' buttons.

Once this is deployed, we need to enable our static website.



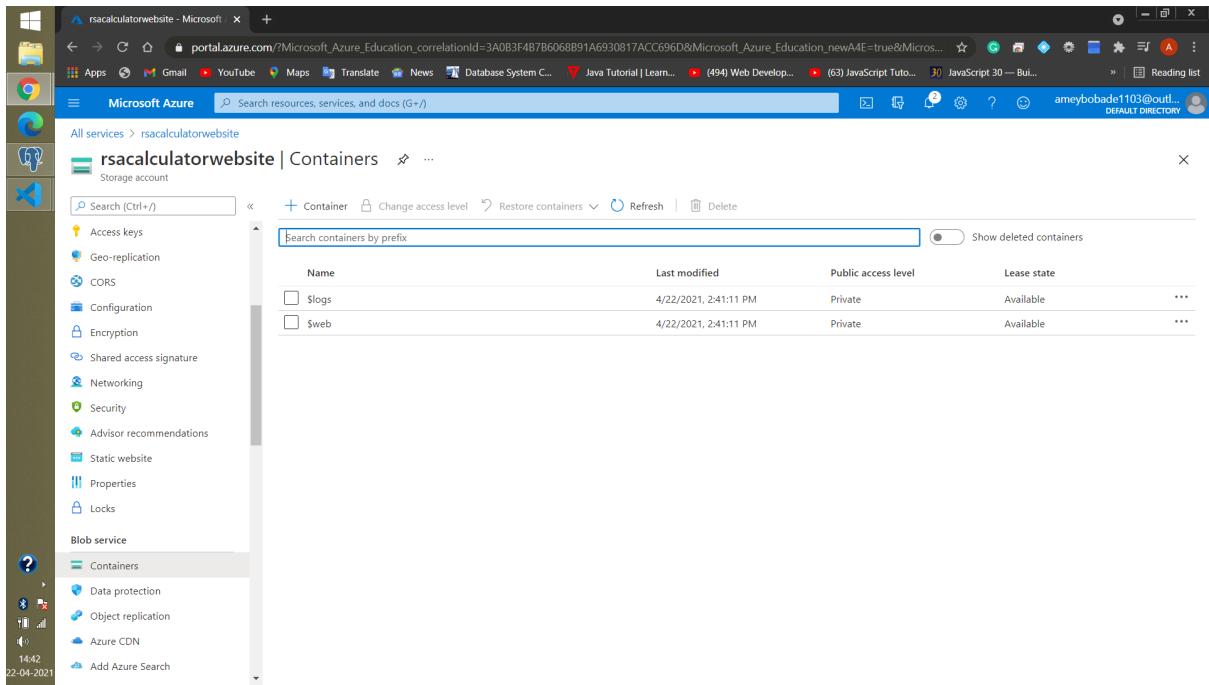
The screenshot shows the Azure portal interface. The left sidebar has 'Storage account' selected under 'All services > rsacalculatorwebsite'. The main content area is titled 'rsacalculatorwebsite | Static website'. It displays a message about enabling static websites on the blob service. A 'Static website' section is open, showing 'Disabled' is selected for the 'Enabled' button. Other options like 'Properties', 'Locks', and 'Blob service' are also visible. The status bar at the bottom shows '14:39 22-04-2021'.

Azure Private Endpoint is a network interface that connects you privately and securely to a service powered by **Azure Private Link**. **Private Endpoint** uses a private IP address from your VNet, effectively bringing the service into your VNet.



This screenshot shows the same Azure portal interface as the previous one, but the 'Enabled' button in the 'Static website' section is now set to 'Enabled'. Below it, the configuration for the primary endpoint is shown: 'Primary endpoint' is set to <https://rsacalculatorwebsite.z13.web.core.windows.net/>. There are also fields for 'Secondary endpoint' (<https://rsacalculatorwebsite-secondary.z13.web.core.windows.net/>), 'Index document name' (index.html), and 'Error document path' (error.html). The status bar at the bottom shows '14:41 22-04-2021'.

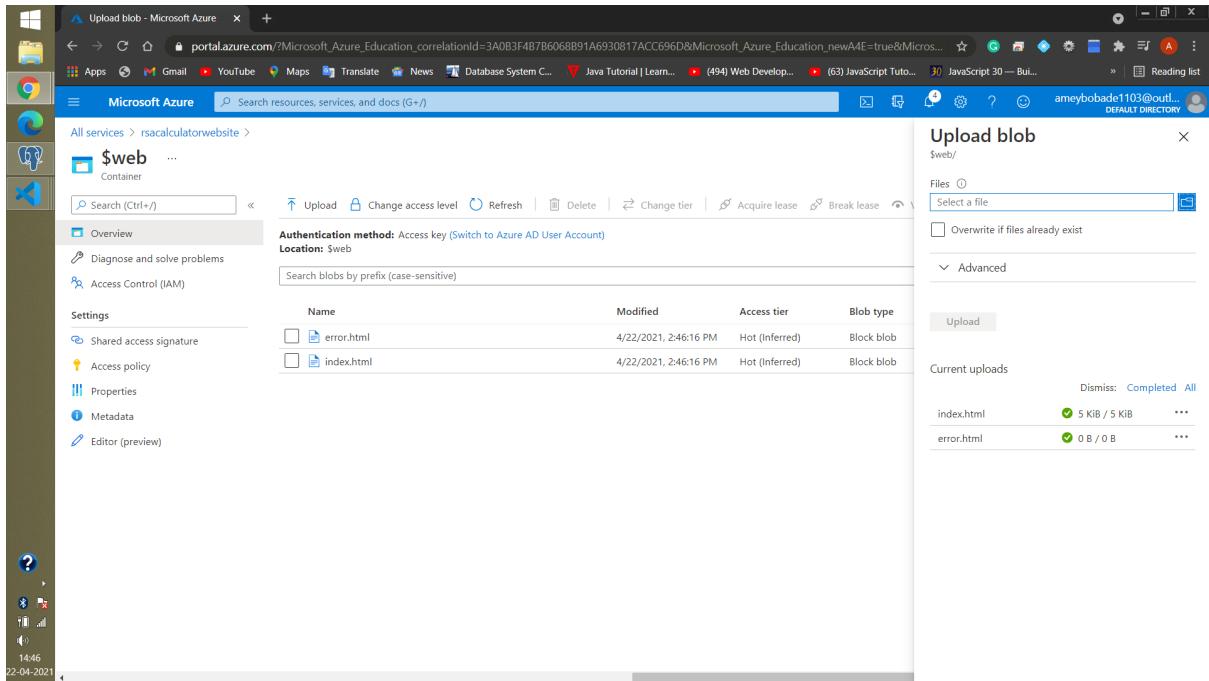
A container is a readymade software environment that has the application code and its dependency's preloaded within an image. ... The **container** engine allows each application (**container**) to run on top of the host operating system but remain isolated from each other.



A screenshot of the Microsoft Azure portal showing the storage account 'rsacalculatorwebsite'. The left sidebar shows various service options like Access keys, Geo-replication, CORS, Configuration, etc. Under Blob service, 'Containers' is selected. The main pane displays a table of containers with two entries: '\$logs' and '\$web'. Both were modified on 4/22/2021 at 2:41:11 PM and are set to 'Private' with 'Available' lease state.

Name	Last modified	Public access level	Lease state
\$logs	4/22/2021, 2:41:11 PM	Private	Available
\$web	4/22/2021, 2:41:11 PM	Private	Available

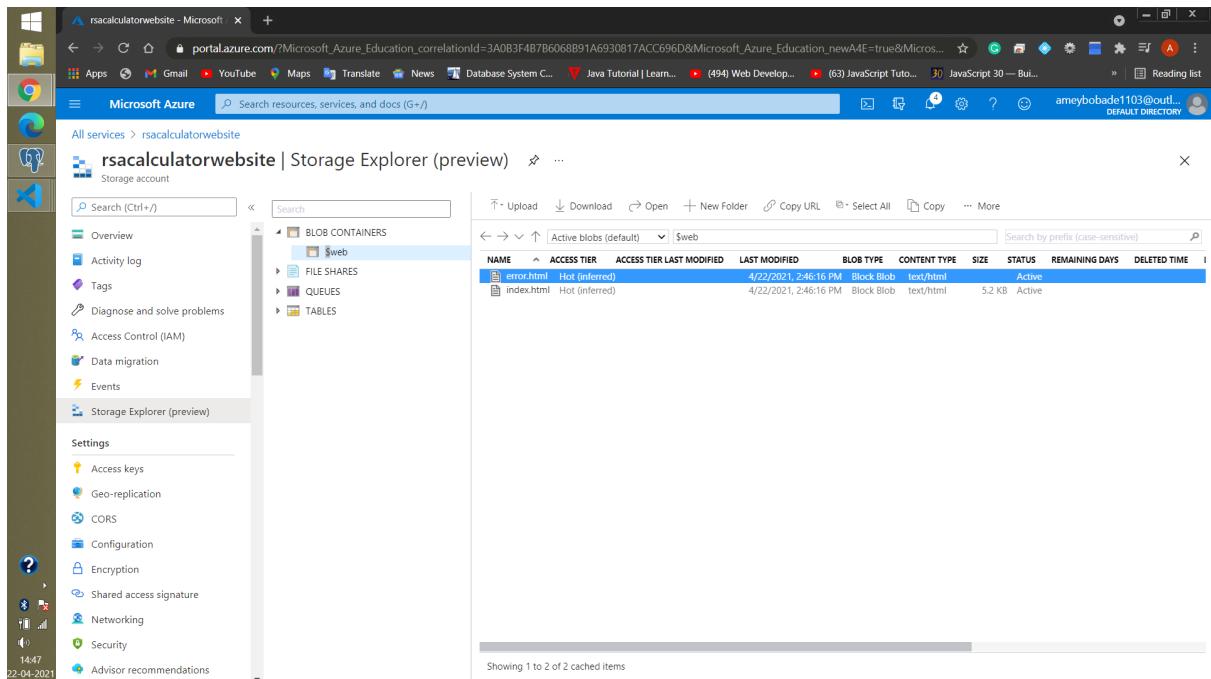
Then you'll see the \$web and \$logs folder. Open the \$web folder and here you'll have to upload your files.



A screenshot of the Microsoft Azure portal showing the '\$web' container details. The left sidebar shows settings like Shared access signature, Access Control (IAM), Properties, Metadata, and Editor (preview). The main pane shows two blobs: 'error.html' and 'index.html', both modified on 4/22/2021 at 2:46:16 PM and categorized as 'Hot (Inferred)' with 'Block blob' type. A right-hand panel titled 'Upload blob' allows for file selection and overwriting existing files. It also shows a 'Current uploads' section with 'index.html' and 'error.html' listed.

A **blob** is a data type that can store binary data. This is different from most other data types used in databases, such as integers, floating point numbers, characters, and strings, which store letters and numbers. Since **blobs** can store binary data, they can be used to store images or other multimedia files.

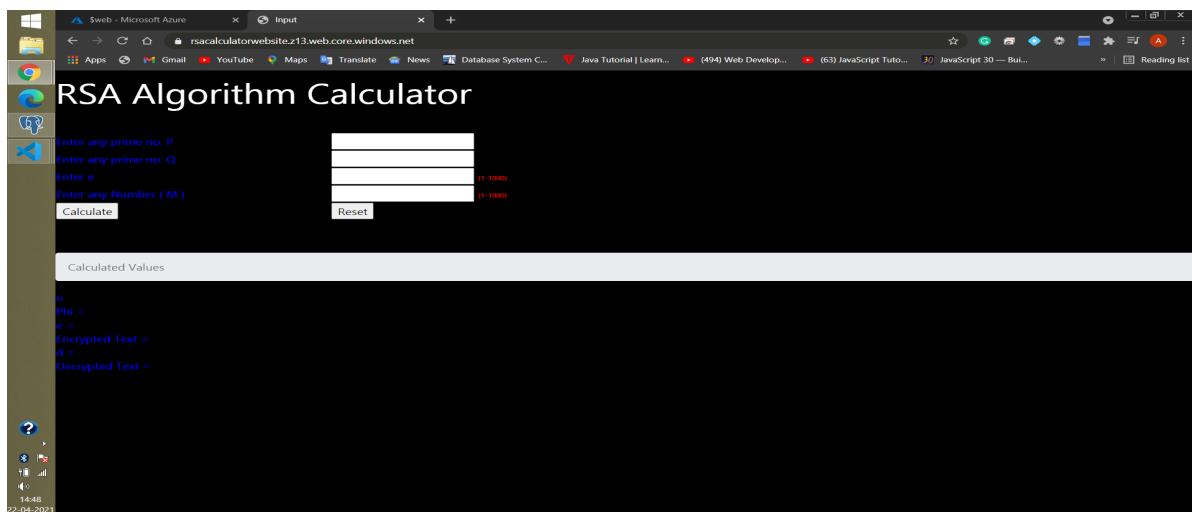
Then open Storage Explorer (preview). **Azure Storage Explorer** is an application which helps you to easily access the **Azure storage** account through any device on any platform, be it Windows, MacOS, or Linux. You can easily connect to your subscription and manipulate your tables, blobs, queues, and files.



Now your website is deployed and ready to go!

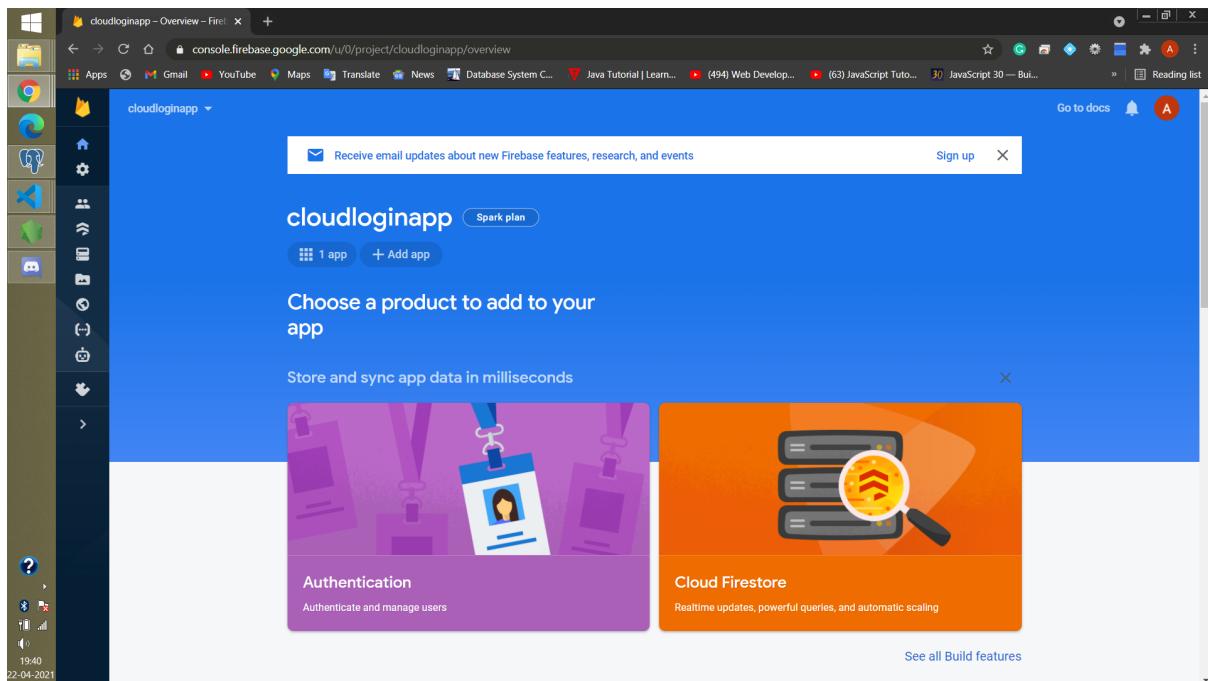
<https://rsacalculatorwebsite.z13.web.core.windows.net/>

RSA is a public-key cryptosystem and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private).

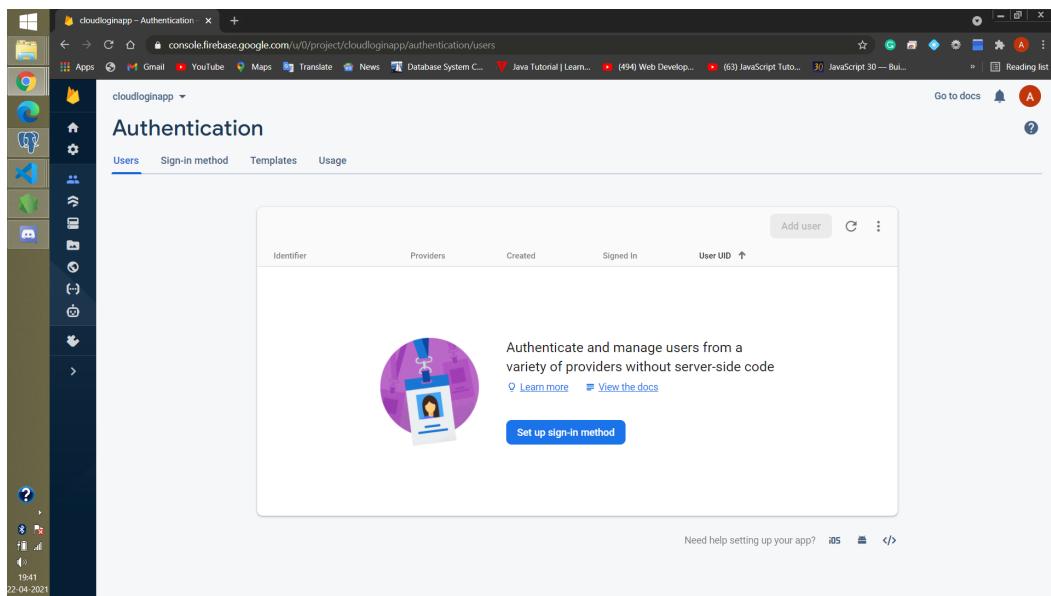


2.GOOGLE FIREBASE AUTHENTICATION

Step1: Login to your firebase account and create a new app. Then, choose any one product for your app mentioned below. Here, we are using authentication. **Firebase Authentication** provides backend services, easy-to-use SDKs, and ready-made UI libraries to **authenticate** users to your app. It supports **authentication** using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.



Step 2: Now , setting up sign in method



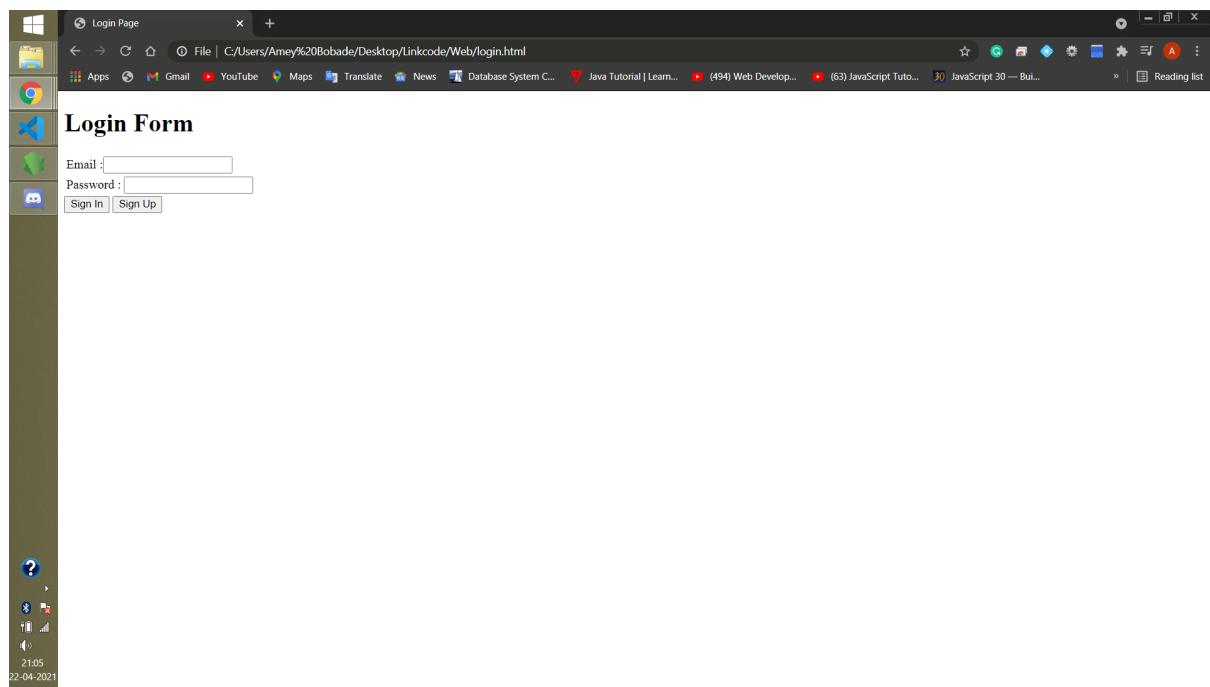
Step 3: Here we are allowing users to sign-up using their email address and password by enabling the service mentioned below.

Provider	Status
Email/Password	Enable
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center	Disabled

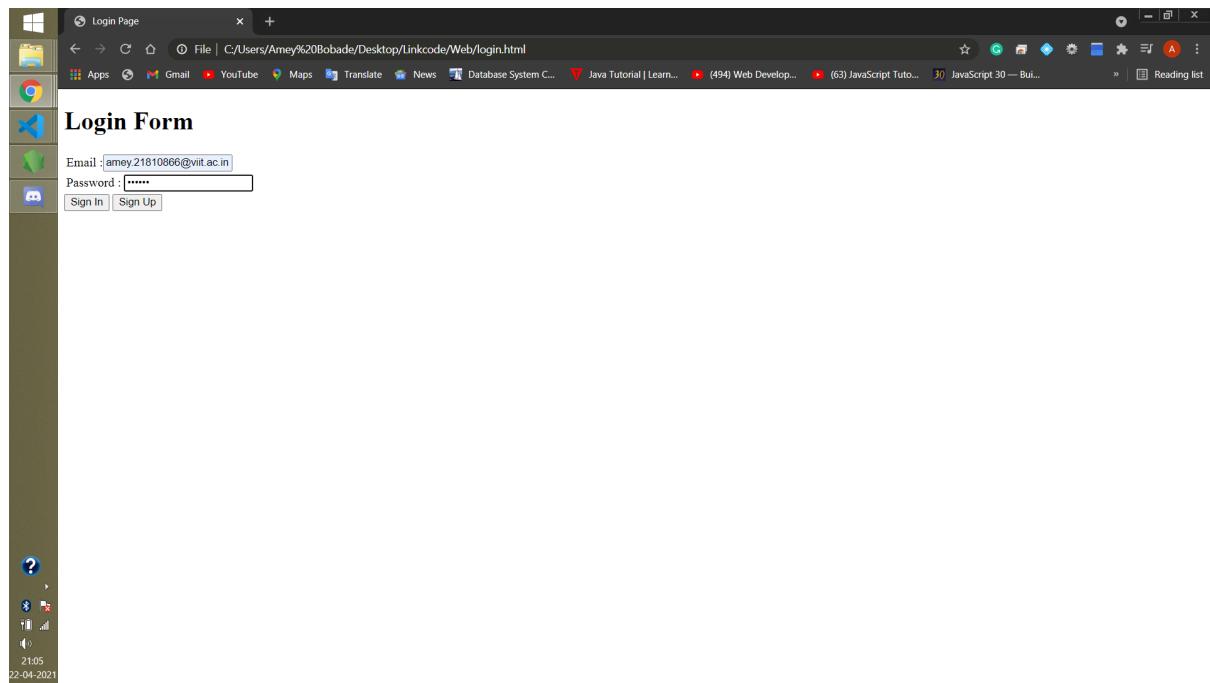
Enabling the feature->

Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center	Disabled
Facebook	Disabled
Twitter	Disabled
Github	Disabled
Yahoo	Disabled
Microsoft	Disabled

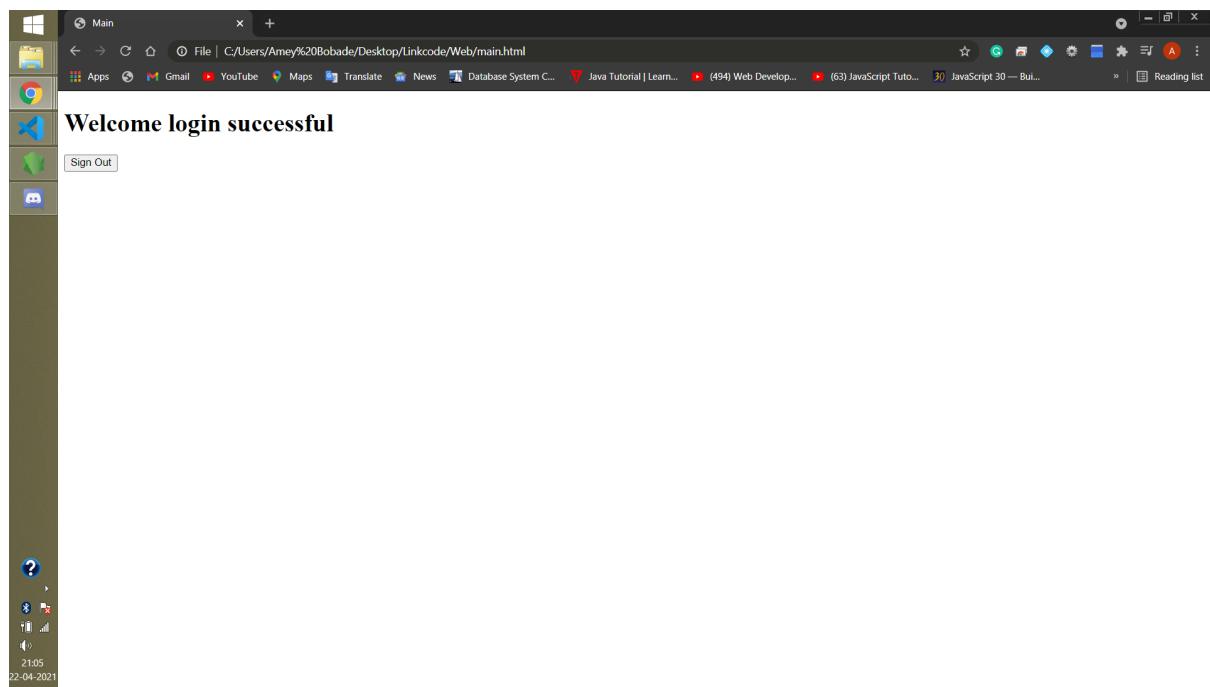
Here is the login form created.



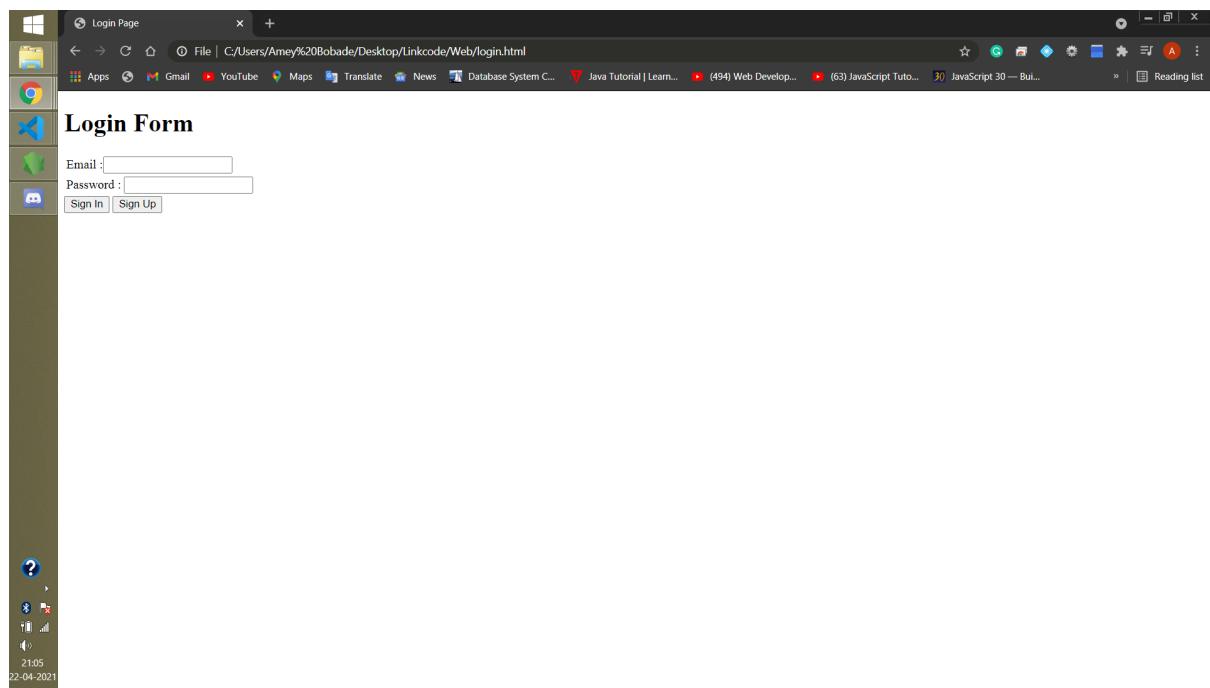
Here, we are trying to Sign-in through a mail and password.



As you can see, login has been successful.



You can also sign-out.



Here is the screenshot of the code we used for authentication.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Page</title>
    <script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-app.js"></script>
    <script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-auth.js"></script>
    <script src="login.js"></script>
</head>
<body>
    <h1>Login Form</h1>
    <div id="formContainer" >
        <div id="header">
            <table>
                <tr>
                    <td> Email :<input type="email" name="email" id="email">
                </td>
                <tr>
                    <td> Password : <input type="password" name="password" id="password"><br>
                </td>
            </table>
        </div>
    </div>
</body>
```

22-04-2021 0 2 hrs 41 mins Azure: ameybobade1103@outlook.com Ln 17, Col 29 Spaces: 4 UTF-8 LF HTML Go Live 2h 33m Flow Prettier

```
// Your web app's Firebase configuration
var firebaseConfig = {
  apiKey: "AIzaSyAh-yw6_jImfqc0npN9SctB12ujVxSiXm0",
  authDomain: "cloudloginapp.firebaseioapp.com",
  projectId: "cloudloginapp",
  storageBucket: "cloudloginapp.appspot.com",
  messagingSenderId: "615208865900",
  appId: "1:615208865900:web:faaef075700ca4422c4849"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);

const auth = firebase.auth();

function signup() {
  var email = document.getElementById("email");
  var password = document.getElementById("password");

  const promise = auth.createUserWithEmailAndPassword(email.value, password.value);
  promise.catch(e => alert(e.message));

  alert("Signed Up");

  document.getElementById('email').value = '';
  document.getElementById('password').value = '';
}

function signin(){
  var email = document.getElementById("email");
}
```

22-04-2021 0 2 hrs 41 mins Azure: ameybobade1103@outlook.com Ln 48, Col 1 Spaces: 2 UTF-8 LF JavaScript Go Live 2h 33m Flow Prettier

A screenshot of Visual Studio Code showing the file `login.js`. The code defines three functions: `signup`, `signin`, and `signout`. The `signup` function creates a user with email and password. The `signin` function signs in with email and password, then redirects to `main.html`. The `signout` function signs out and then redirects back to `login.html`.

```
File Edit Selection View Go Run Terminal Help
login.js - Web - Visual Studio Code

JS loginjs > ...
16     function signup() {
17         var email = document.getElementById("email");
18         var password = document.getElementById("password");
19
20         const promise = auth.createUserWithEmailAndPassword(email.value, password.value);
21         promise.catch(e => alert(e.message));
22
23         alert("Signed Up");
24
25         document.getElementById('email').value = '';
26         document.getElementById('password').value = '';
27     }
28
29
30     function signin(){
31
32         var email = document.getElementById("email");
33         var password = document.getElementById("password");
34
35         const promise = auth.signInWithEmailAndPassword(email.value, password.value);
36         promise.catch(e => alert(e.message));
37
38         window.location.href = 'main.html';
39     }
40
41     function signout(){
42
43         auth.signOut();
44         alert("Signed Out");
45
46         window.location.href = 'login.html';
47     }

Ln 81, Col 1 (209 selected) Spaces: 2 UTF-8 LF JavaScript ⚡ Go Live 2h 33m ⚡ Flow ✨ Prettier ⌂ ⌂
22-04-2021 ⌂ 0 ⌂ 0 2 hrs 45 mins Azure: ameybobade1103@outlook.com
```

A screenshot of Visual Studio Code showing the file `main.html`. The code is an HTML page with a title "Main". It includes meta tags for charset, http-equiv, and viewport. It has a script tag pointing to `login.js`. The body contains a welcome message and a button for signing out.

```
File Edit Selection View Go Run Terminal Help
main.html - Web - Visual Studio Code

main.html > html > body
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Main</title>
8
9      <script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-app.js"></script>
10     <script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-auth.js"></script>
11     <script src="login.js"></script>
12 </head>
13 <body>
14     <h1>Welcome login successful</h1>
15
16     <button id="signout" value="Sign Out" onclick="signout()">Sign Out</button>
17
18 </body>
19 </html>

Ln 15, Col 5 Spaces: 4 UTF-8 CRLF HTML ⚡ Go Live 2h 33m ⚡ Flow ✨ Prettier ⌂ ⌂
22-04-2021 ⌂ 0 ⌂ 0 2 hrs 45 mins Azure: ameybobade1103@outlook.com
```

3.AMAZON HONEYCODE

What Is Amazon Honeycode?

Amazon Honeycode, which is available in beta, is a fully managed service that allows

customers to quickly build powerful mobile and web applications – with no programming

required. Customers who need applications to track and manage things like process approvals, event scheduling, customer relationship management, user surveys, to-do lists,

and content and inventory tracking no longer need to do so by error-prone methods like

emailing spreadsheets or documents, or hiring and waiting for developers to build costly

custom applications.

With Amazon Honeycode, customers can use a simple visual application builder to create highly

interactive web and mobile applications backed by a powerful AWS-built database to perform tasks like

tracking data over time and notifying users of changes, routing approvals, and facilitating interactive

business processes. Using Amazon Honeycode, customers can create applications that range in

complexity from a task-tracking application for a small team to a project management system that

manages a complex workflow for multiple teams or departments.

Is Amazon Honeycode free?

Amazon Honeycode is free to use for teams with up to 20 members and 2,500 rows per

workbook. Upgrade to a paid plan when you need additional capacity.

Benefits of using Amazon Lex:

With Honeycode, one can create web & mobile apps that can work on a desktop browser,

Android, and iOS platforms.

No programming skills are needed so that anyone can be an app developer. Honeycode apps can help teams stay in a loop by giving them access to the same data. Personalized features of Honeycode apps can make teams remain focused. This is possible because these apps show data that are relevant to each team member. Honeycode can automate manual steps

Features of Amazon Honeycode:

Customized apps without programming.

Personalize data for each team member.

Workflow automation

Share the app

Access anywhere and update anytime.

After having basic knowledge about Amazon Honeycode, let's learn about how to start

working with it.

How it works:

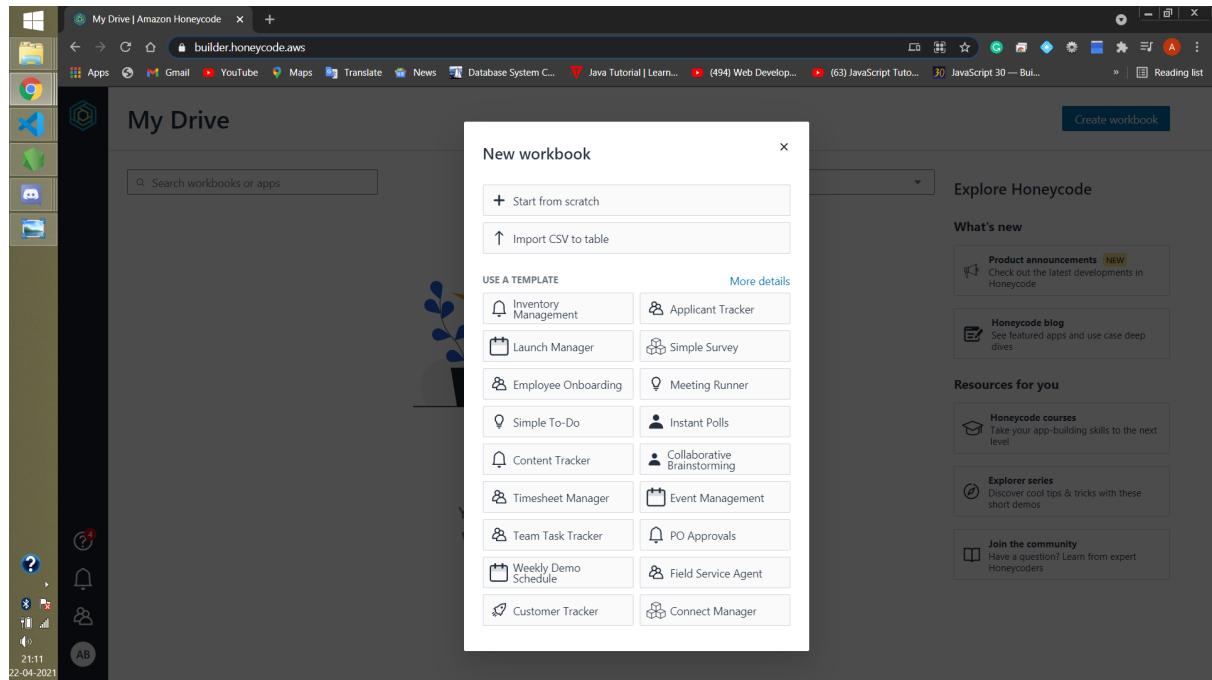
Amazon has built-in templates that can be chosen:

- Simple To-do
- Customer Tracker
- Simple Survey
- Inventory Management
- Content Tracker
- Timeoff Reporting
- Event Management
- Team Task Tracker
- Weekly Demo Schedule
- Field Service Agent
- PO Approvals

Or you can even build one from scratch with a similar process, we'll go with a built-in template.

Steps:-

- 1) GO onto the Amazon Honeycode website, and click on Sign Up.



- 2) Creating an account will lead you to the Honeycode Dashboard

The screenshot shows the Honeycode dashboard with a specific table named 'A_Tasks' open. The table has columns labeled A through I, with headers: Task, Project, Assignee, Due, Status, Order, Priority, Notes, and Links. The data in the table is as follows:

A	B	C	D	E	F	G	H	I
Task	Project	Assignee	Due	Status	Order	Priority	Notes	Links
1 Update Icons for social links	Newsletter	Amey Bobade	12/12/20	Not Started	3	Medium		http://example.com/1
2 Update the distribution list	Newsletter	Jane Roe	4/22/20	In Progress	1	High		
3 New design for "About Us" page to include team bios	Web Redesign	Jane Roe	6/22/21	Blocked	2	Medium		
4 Create new pattern for error notifications	Web Redesign	John Stiles	4/7/21	Not Started	3	Low		Waiting on new asset
5 Update icon library for v2	Web Redesign	John Stiles	7/19/21	In Progress	1	High		
6 Pick a new color palette	Web Redesign	Amey Bobade	9/8/21	Completed	4	Medium		

3) Click on Create a New Workbook on Right Top Corner.

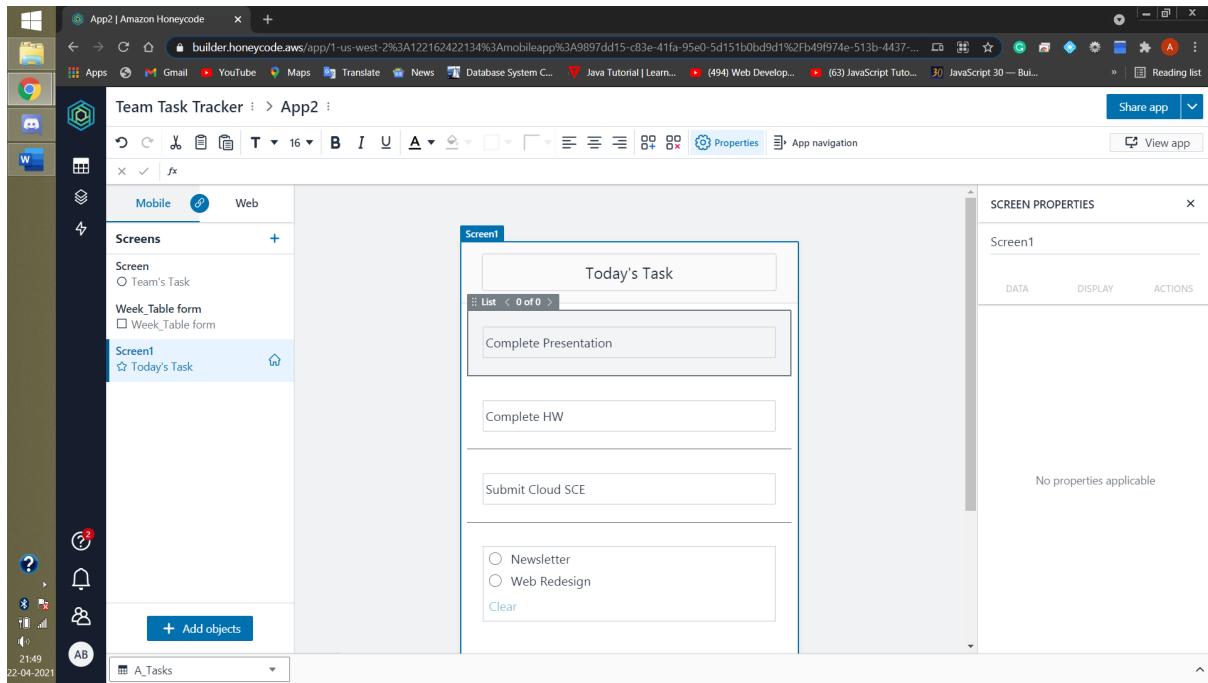
The screenshot shows a web-based spreadsheet application titled "Team Task Tracker". The data is organized into columns: Task, Project, Assignee, Due, Status, and Priority. One row, task 4, has its "Status" field set to "Blocked".

Task	Project	Assignee	Due	Status	Priority
1 Update Icons for social links	Newsletter	Amey Bobade	12/12/20	Not Started	Medium
2 Update the distribution list	Newsletter	Anish	4/22/20	In-Progress	High
3 New design for "About Us" page to include team bios	Web Redesign	Amlan Nanda	6/22/21	Blocked	Medium
4 Create new pattern for error notifications	Web Redesign	Pritesh	4/7/21	Not Started	Low
5 Update icon library for v2	Web Redesign	Vedant	7/19/21	In-Progress	High
6 Pick a new color palette	Web Redesign	Aadarsh	9/8/21	Completed	Medium

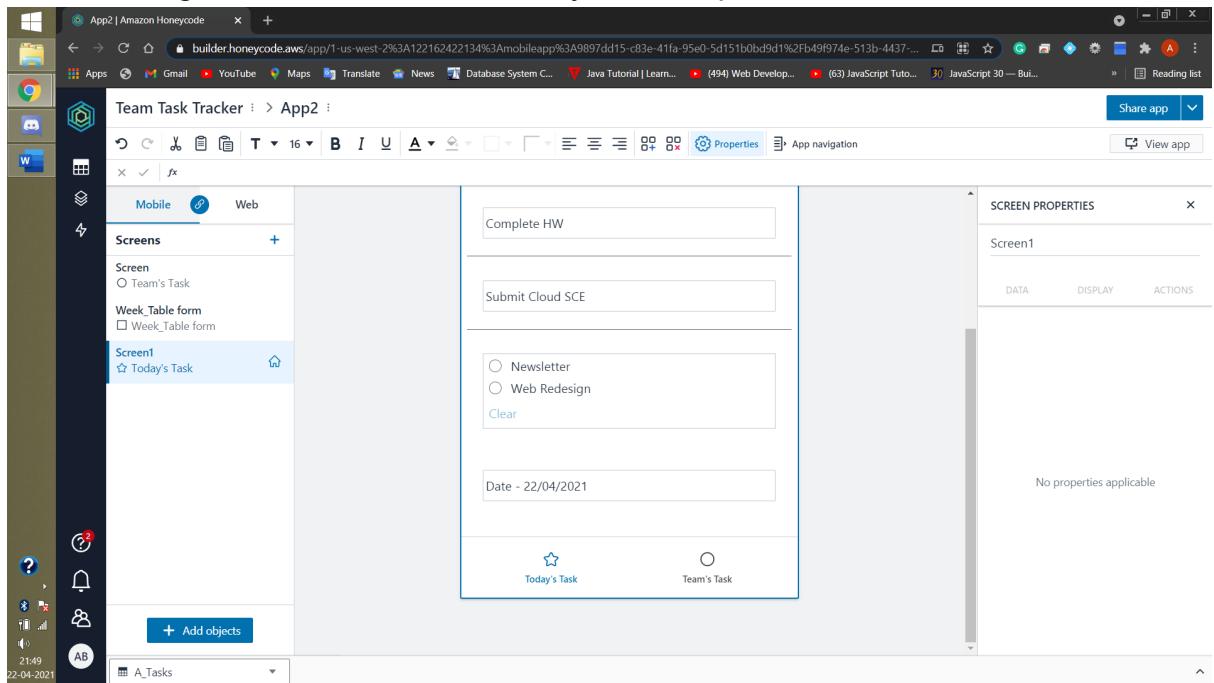
4) Select a Built-In Template for now. We'll go with the To-Do App. We get this spreadsheet for the data in our To-DO App

The screenshot shows the Honeycode builder interface with a project titled "App1". A "Screens" panel on the left lists a "Screen" object. The main workspace displays a "My Today's Tasks" screen with a "ContentBox" object containing the text "Hey there, Builder!". The "Properties" panel on the right shows the "ContentBox" properties.

5) Now, click on the Builder tab on the left panel below the Tables tab, and select ToDo from there.

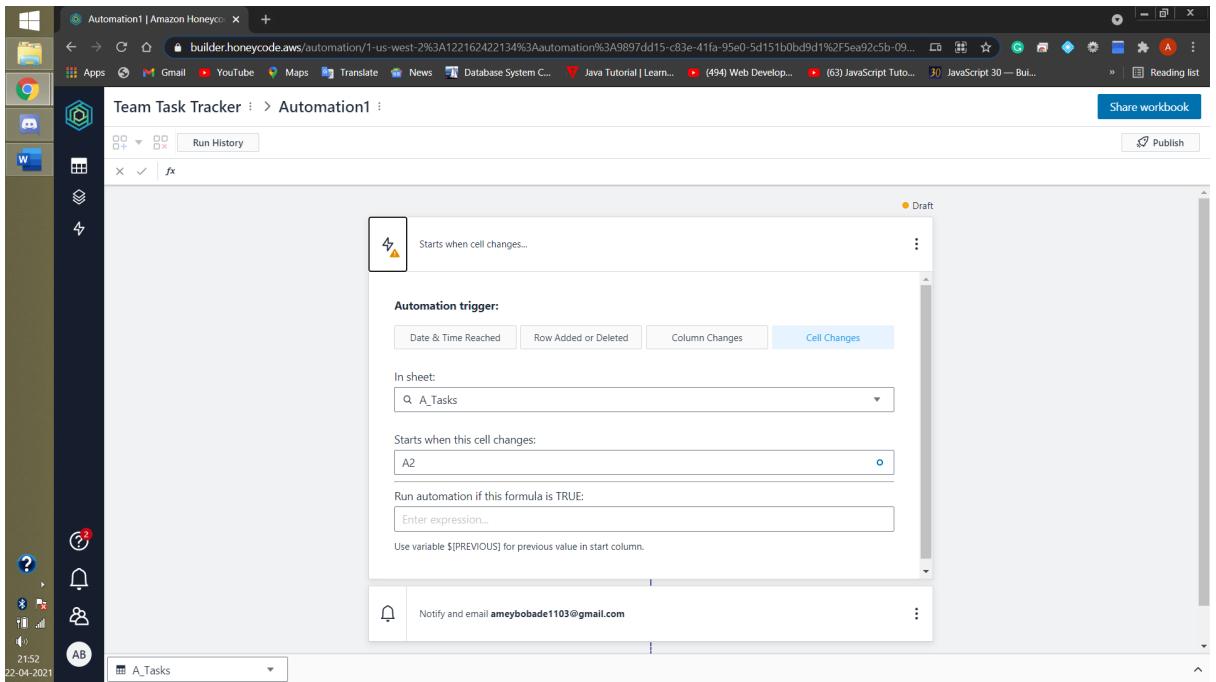


6) Here you can edit anything that should be displayed in your app. From the Bottom Navigation Bar to each and every element present.

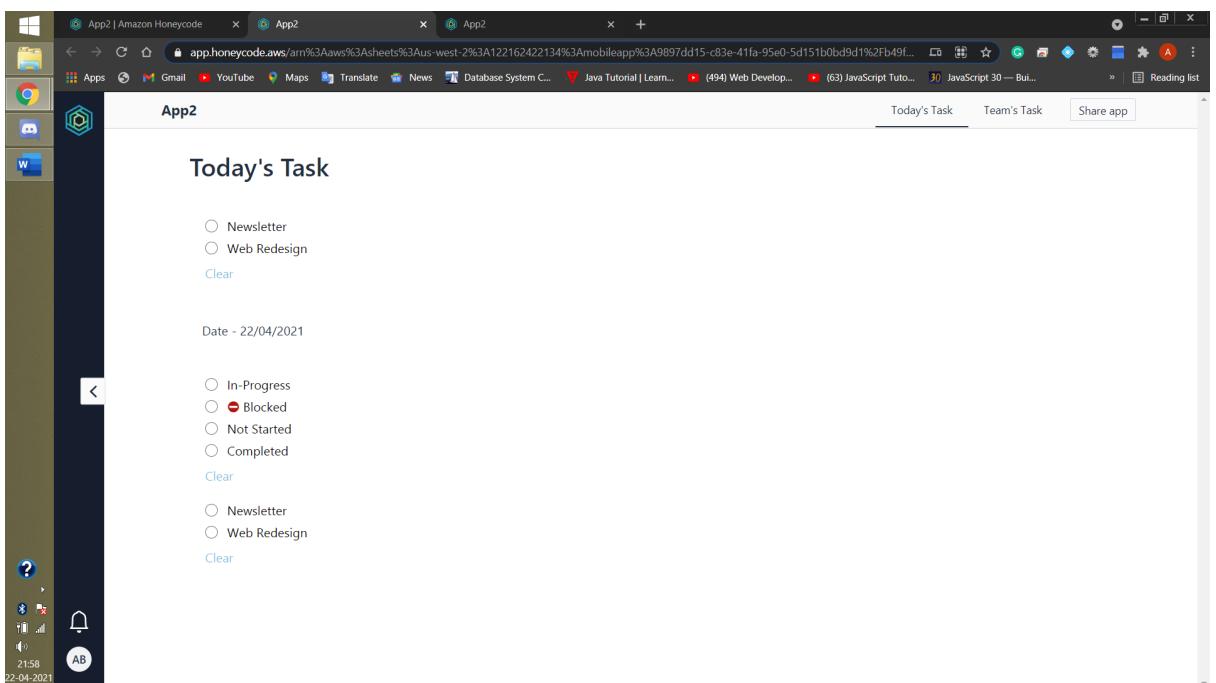


7) We can also add any kind of automations we want from the Automations tab below Builder Tab. Publish the automation from the top-right corner once you

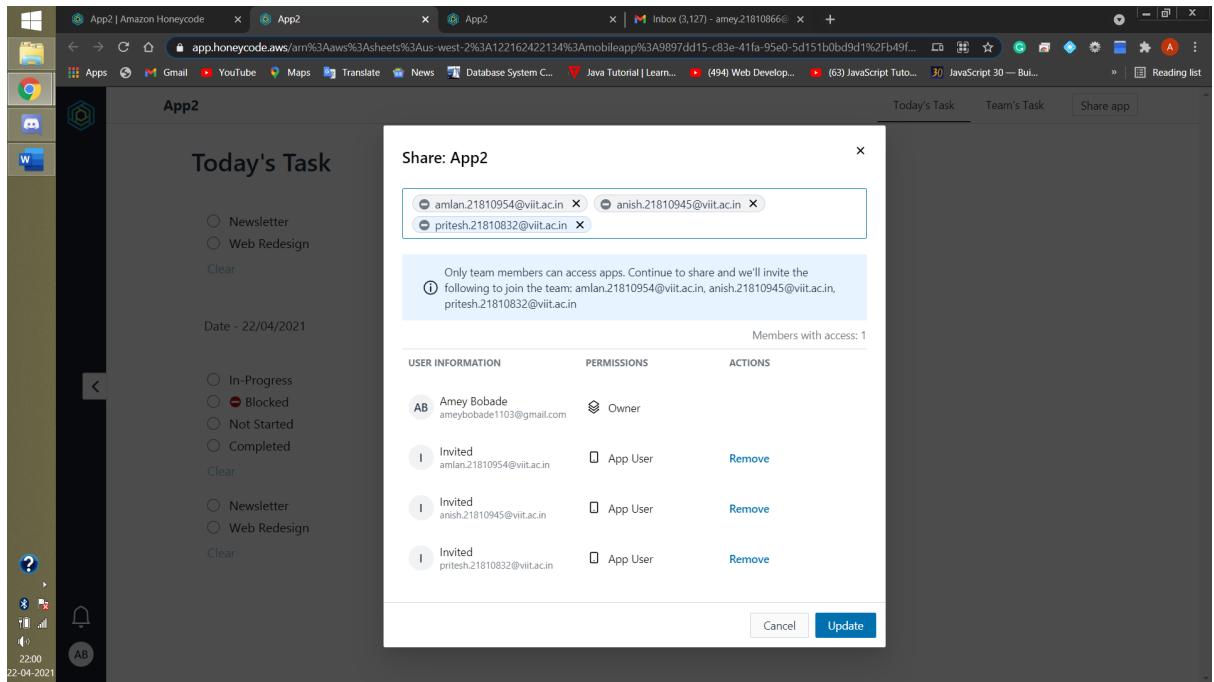
set it up.



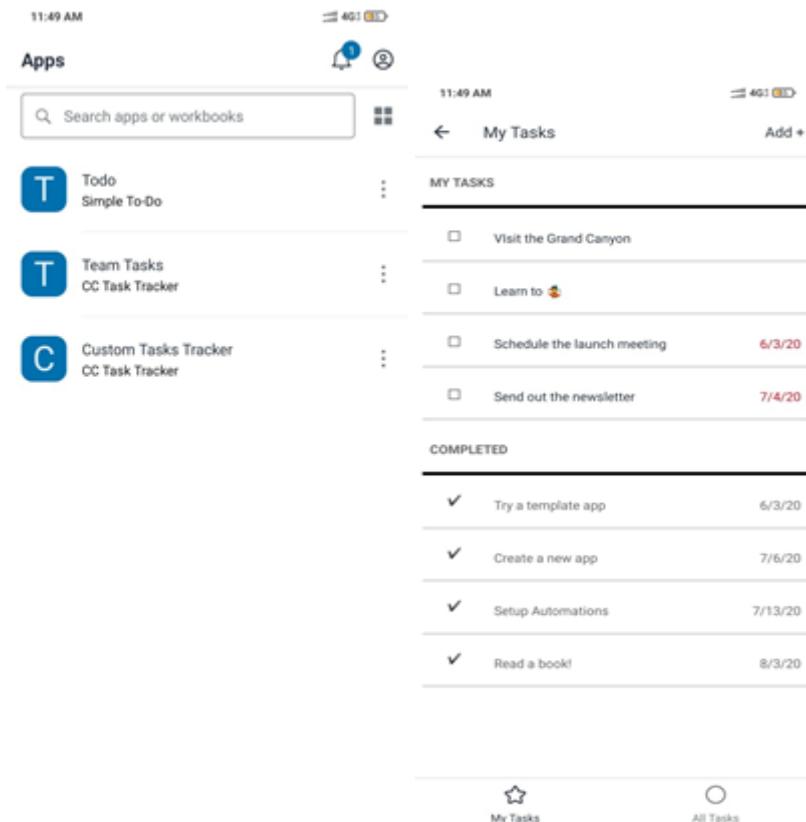
- 8) Go back to the Builder tab and make sure your app looks good, by checking it in the Preview tab.



9) Click on Share App, and add all your teammates' mails.



10) Install Amazon Honeycode app on your mobile phone for easy access. Lets see its various screens now.



And, with that we're done. A new custom app within 30 mins can be set up for your team!