

Firestore

.1. Server ki zarurat aur Firestore ka Serverless Model:

Traditional systems mein, **Front-End (FED)** se data ko handle karne ke liye ek **backend server** ki zarurat hoti hai. Front-end requests ko backend server pe send karta hai, jahan logic run hoti hai, aur backend server database ko manage karta hai. Yeh model thoda complex hota hai, kyun ke server maintain karna aur secure rakhna bhi zaruri hota hai.

Firestore ne is process ko simple bana diya hai kyun ke yeh ek **serverless** architecture provide karta hai. Firestore ka **Realtime Database** directly front-end se connect ho jata hai, bina kisi backend server ke. Iska matlab yeh hai ke front-end application (jaisa ke website ya mobile app) se aap directly database mein **data add, remove ya update** kar sakte hain. Backend app ka server maintain karne ki zarurat nahi parti. Firestore ki services cloud pe hosted hoti hain, jo saari complexity ko handle karti hain.

2. Firestore Queries for Better Data Search:

Firestore aapko **queries** likhne ka option bhi deta hai jo data search ko efficient banati hain. Firestore ka **Firestore** ya **Realtime Database** aapko query filters provide karta hai jinke zariye aap specific data ko filter kar sakte hain. Misal ke taur par, aap data ko **sort** kar sakte hain, **range** define kar sakte hain (like date se related), ya sirf specific fields ko retrieve kar sakte hain.

Yeh queries backend se directly handled hoti hain, aur aapko complex logic likhne ki zarurat nahi hoti, jo a traditional server setup mein zaruri hoti.

3. Data Types: Database vs Storage Buckets:

Firestore ke database mein aap sirf text ya structured data (like JSON) store karte hain. Lekin jab aapko **files, images, videos** store karni hoti hain, toh Firestore aapko **Cloud Storage** provide karta hai, jise aap **Storage Bucket** bhi kehte hain. For example, agar aapko kisi user ki profile image ya video store karni hai, aap usse Firestore Storage mein upload karenge.

Process: Jab aap file ko Storage mein save karte hain, Firestore aapko us file ka **URL/link** provide karta hai. Aap us link ko apne database mein store kar sakte hain aur jab bhi zarurat ho, aap wo link use karke file ko access kar sakte hain.

4. Realtime Database:

Firestore ka ek unique aur powerful feature **Realtime Database** hai. Iska faida yeh hai ke aap apne data ko **real-time** mein sync kar sakte hain. For example, agar aap ek **chat app** bana rahe hain, toh user ko har baar page reload karne ki zarurat nahi hogi. Jab bhi new message aaye ga, wo real-time mein screen par show ho jaye ga.

Firestore ka **Realtime Database** automatically **listeners** lagata hai jo continuously data changes ko monitor karte hain. Jab bhi database mein koi **add, remove ya update** hota hai, yeh listeners automatically trigger ho jate hain aur front-end pe wo changes show hoti hain. Agar aap traditional backend pe yeh feature khud implement karenin, toh aapko **WebSockets** ka use karna padega, jo kaafi complex hota hai.

5. Firestore Authentication:

Firestore **authentication service** bhi provide karta hai jo multiple methods ke saath kaam karti hai, jaise:

- **Email/Password authentication**
- **Google Sign-in**
- **Facebook Sign-in**
- **Phone number authentication, etc.**

Is service ki madad se aapko complex authentication logic likhne ki zarurat nahi hoti. Firestore khud se secure authentication handle karta hai aur aapko asani se different login methods integrate karne ka option milta hai.

6. Firebase Hosting:

Firebase आपको **hosting service** bhi deta hai, jisse aap apne web apps ko easily deploy kar sakte hain. Firebase hosting ki madad se aap static aur dynamic web apps ko host kar sakte hain, aur yeh platform globally distributed servers ka use karta hai jo aapki app ko fast aur secure banata hai.

Firebase hosting ke saath aap apni website ya web app ko asani se deploy kar sakte hain, bina kisi manual server configuration ke.

7. Firebase Cloud Storage:

Firebase ka **Cloud Storage** files jaise images, videos, documents ko securely store karne ke liye use hota hai. Jab aap koi file (jaise user ki profile image) Cloud Storage mein save karte hain, Firebase आपको us file ka **link** de deta hai. Aap us link ko apne database mein store karte hain aur jab bhi आपको us file ki zarurat hoti hai, aap simple us link se access kar sakte hain.

Example: Agar aap ek social media app bana rahe hain aur users apni profile picture upload karte hain, toh aap image ko **Cloud Storage** mein save karenge aur uska URL apne **database** mein save karenge. Baad mein, jab आपको user ki profile display karni ho, toh aap us URL se image fetch kar lenge.

Initial steps to use firebase :

Firebase ki services ko use karne ke liye आपको kuch **initial steps** follow karne padte hain. Ye steps Firebase project create karne se lekar usko apne web ya mobile application ke saath integrate karne tak ke hain. Main yahaan Firebase ko ek **web project** ke liye configure karne ka process bata raha hoon, lekin yeh steps mobile apps (Android/iOS) ke liye bhi kaafi similar hain.

1. Firebase Account Banana

Sabse pehle, आपको Firebase ki services ko use karne ke liye ek Google account chahiye.

- **Sign up** ya **log in** karein Firebase console par: Firebase Console

2. Firebase Project Create Karna

Firebase ki services ko use karne ke liye आपको ek project create karna hoga. Iske liye:

- Firebase Console par login karne ke baad "**Add Project**" button par click karein.
- Apne project ka **name** dalein.
- Google Analytics enable/disable karne ka option hoga. Aap isse enable bhi kar sakte hain agar analytics chahiye ho, warna disable bhi kar sakte hain.
- Firebase aapka project setup kar dega. Is process mein kuch seconds lagte hain.

3. App ko Firebase Project Mein Add Karna

Jab aapka Firebase project create ho jaye, आपको apni app ko Firebase ke sath integrate karna hoga. Aap yeh integration **Web, Android ya iOS** ke liye kar sakte hain.

Web App Add Karna:

- Firebase Console ke dashboard se **Web (</>)** icon ko select karein.
- Apni app ka **nickname** dalein.

- आपको एक Firebase SDK configuration snippet milega (JavaScript code snippet) jo aapki web app mein integrate hoga. Isse copy kar ke apne HTML file mein `<script>` tag ke andar paste karein.

4. Firebase SDK Ko Apni App Mein Include Karna

Aapko Firebase ki libraries ko apne project mein include karna hoga. Web apps ke liye aap Firebase SDK ka use karte hain jo JavaScript ke zariye hota hai. Firebase SDK ko integrate karne ke liye:

- Apni HTML file mein Firebase configuration snippet paste karein. Example:

```
html Copy code

<!-- Firebase SDKs -->
<script src="https://www.gstatic.com/firebasejs/9.0.0/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/9.0.0/firebase-firestore.js"></script>

<!-- Firebase Configuration -->
<script>
  const firebaseConfig = {
    apiKey: "your-api-key",
    authDomain: "your-app.firebaseio.com",
    projectId: "your-project-id",
    storageBucket: "your-app.appspot.com",
    messagingSenderId: "your-sender-id",
    appId: "your-app-id"
  };

  // Initialize Firebase
  const app = firebase.initializeApp(firebaseConfig);
</script>
```

Is code snippet mein `firebaseConfig` ka use karke Firebase services ko initialize karte hain.

5. Firebase Services Ka Use Karna (e.g., Realtime Database, Firestore, Authentication)

Firebase ko initialize karne ke baad, ab aap Firebase ki different services ko use kar sakte hain. Kuch popular services ka basic example:

a. Firebase Authentication Setup: Firebase Authentication ko setup karne ke liye:

- Firebase Console mein apne project par jaayein.
- **Authentication** section mein jaake **Sign-in method** enable karein (jaise Google Sign-in ya Email/Password).

- Apne JavaScript code mein Firebase Authentication ka use karein

```
javascript Copy code  
  
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";  
  
const auth = getAuth();  
signInWithEmailAndPassword(auth, email, password)  
  .then((userCredential) => {  
    // Signed in  
    const user = userCredential.user;  
  })  
  .catch((error) => {  
    console.error(error);  
  });
```

b. Firestore Database Setup: Firebase Firestore database ko use karne ke liye:

- Firebase Console mein **Firestore Database** section mein jaake ek new database create karein.
- JavaScript mein Firestore ko use karne ka example

```
javascript Copy code  
  
import { getFirestore, collection, addDoc } from "firebase/firestore";  
  
const db = getFirestore();  
try {  
  const docRef = await addDoc(collection(db, "users"), {  
    name: "John Doe",  
    email: "john@example.com"  
  });  
  console.log("Document written with ID: ", docRef.id);  
} catch (e) {  
  console.error("Error adding document: ", e);  
}
```

c. Firebase Storage Setup: Files, images ya videos ko store karne ke liye Firebase Storage ka use karein.

- Firebase Console mein **Storage** section mein jaake ek bucket create karein.
- JavaScript mein Firebase Storage ko use karne ka example

```
javascript Copy code  
  
import { getStorage, ref, uploadBytes } from "firebase/storage";  
  
const storage = getStorage();  
const storageRef = ref(storage, 'some-child');  
  
// File Upload Example  
const file = ...; // Your file here  
uploadBytes(storageRef, file).then((snapshot) => {  
  console.log('Uploaded a blob or file!');  
});
```

6. Deploy Karna (Hosting)

Firebase hosting ko use karne ke liye:

- **Firebase CLI** install karein: *npm install -g firebase-tools* on **cmd prompt**
- Now goto our project folder directory & open terminal there (folder k path bar me cmd type yhen enter)
- Then run cmd **firebase login** it will ask your account ,
- Then on cmd run **firebase init** command chalayein aur hosting select karein.(first option of hosting)
- Select use an existing project & it will ask directory name
- Phr kuch sawal poochy ga uske jawab do
- Apni website ko deploy karne ke liye: **firebase deploy**
- Ye sab one time scene tha , ab jab bhi project me changes kren to cmd “**firebase deploy**”

7.Rollback to previous version :

Firebase k dashboard me ja kr Hosting pe jao whan project k releases show ho rahi hogi wahan kisi pe version pe click kr kr usko rollback kr skty henjis se wo previous “deployed” version pe chala jaye ga

Hosting k page pe ek option milti ha **custom domain** ki usko select krke ham custom domain name bhi rakh skty hen !

Conclusion:

Firebase ko use karne ke liye yeh initial steps follow karte hain:


1. Firebase account aur project create karna.
2. Apne app ko Firebase project mein add karna.
3. Firebase SDK ko include karna.

4. Firebase services ka use karna (authentication, database, storage, etc.)
5. Firebase hosting ke zariye app deploy karna.

Yeh basic steps aapko Firebase ke different services ke saath quickly integrate karne mein madad denge.

Understanding the code: users auth

javascript

 Copy code

```
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";


const auth = getAuth();
signInWithEmailAndPassword(auth, email, password)
  .then((userCredential) => {
    // Signed in
    const user = userCredential.user;
  })
  .catch((error) => {
    console.error(error);
  });
```

Yeh code Firebase Authentication ke zariye user ko **email** aur **password** ka istimaal karte hue sign-in karne ka process hai. Firebase ki authentication service ko use karte hue, hum `signInWithEmailAndPassword` function ka istemal karte hain. Ab main tumhe step by step samjhata hoon:

Code Explanation:

1. Firebase Authentication Ko Import Karna:

javascript

 Copy code

```
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";
```

Is line mein hum Firebase se `getAuth` aur `signInWithEmailAndPassword` functions ko import kar rahe hain. Yeh Firebase ki authentication service se deal karne ke liye zaroori hain.

- `getAuth()` : Yeh function Firebase ka authentication service ka instance return karta hai jo hum use karenge.
- `signInWithEmailAndPassword()` : Yeh function user ko email aur password ke zariye sign-in karne ke liye hota hai.

2. Authentication Object Banana:

javascript

Copy code

```
const auth = getAuth();
```

Yahan `getAuth()` function ko call karke hum Firebase authentication ka ek object `auth` bana rahe hain. Is object ko hum sign-in karte waqt use karenge.

3. User Ko Sign-in Karna:

javascript

Copy code

```
signInWithEmailAndPassword(auth, email, password)
```

Is line mein `signInWithEmailAndPassword` function ko call kiya gaya hai. Isme 3 cheezain pass ho rahi hain:

- `auth`: Jo authentication object humne pehle banaya tha.
- `email`: User ka email jo frontend ya kisi form se mila hoga.
- `password`: User ka password jo frontend ya kisi form se mila hoga.

Ye function Firebase ko request bhejta hai user ko sign-in karne ke liye. Agar email aur password sahi hain, to ye user ko sign-in kar dega, warna error throw karega.

4. Sign-in Success (then block):

javascript

Copy code

```
.then((userCredential) => {  
  // Signed in  
  const user = userCredential.user;  
})
```

- Jab user successfully sign-in ho jata hai, to `.then()` block call hota hai.
- `userCredential`: Ye object return hota hai jab user successfully sign-in ho jata hai.
- `user`: Ye user ka object hota hai jo sign-in ke baad milta hai. Isme user ki information hoti hai, jaise email, UID, etc. Tum is user object ko apne application mein use kar sakte ho.

5. Error Handling (catch block):

javascript

Copy code

```
.catch((error) => {  
  console.error(error);  
});
```

- Agar sign-in mein koi error hota hai, jaise galat email ya password ya Firebase server ka issue, to `.catch()` block execute hota hai.
- `error`: Ye object hota hai jo error ki details provide karta hai. Tum isse log kar sakte ho ya user ko error message dikhane ke liye use kar sakte ho.

Example Scenario:

Agar user `email = "john@example.com"` aur `password = "123456"` dalta hai, aur Firebase ke database mein ye user valid hai:

- `signInWithEmailAndPassword(auth, email, password)` call karega aur Firebase ko verify karega.
- Agar sab kuch sahi hai, to user sign-in ho jayega aur uski information `userCredential` ke zariye milegi.
- Agar galat password ya email hai, to `catch` block execute hoga aur error console mein show hoga.

Summary:

- `getAuth()`: Firebase authentication instance create karta hai.
- `signInWithEmailAndPassword()`: User ko email aur password ke through sign-in karta hai.
- `.then()`: Jab sign-in successful ho, tab user ka data milta hai.
- `.catch()`: Agar sign-in mein koi error ho to ye block run hota hai aur error handle karta hai.

Yeh method asaan aur efficient tareeqa hai Firebase authentication ko use karne ka!

Javascript me type='module' hoto usme event listeners work nahi krty ! jab tak export na ho

```
JS app.js > ...
1 function register(){
2   console.log("test")
3 }
4
5 window.register = register;
```

Ya phr event listener laga do:

```
let registerBtn = document.getElementById("registerBtn");

registerBtn.addEventListener("click", register)
```

```
JS app.js > ...
1 import { initializeApp } from 'https://www.gstatic.com/firebasejs/10.7.0/firebase-app.js'
2
3 import { getAuth, createUserWithEmailAndPassword } from 'https://www.gstatic.com/firebasejs/10.7.0/firebase-auth.js'
4
5
6
7 // Your web app's Firebase configuration
8 const firebaseConfig = {
9   apiKey: "AIzaSyDG6-ZhFAuf2xkIVjIeFKPzYXylqsASeXU",
10  authDomain: "smit-batch-10.firebaseio.com",
11  projectId: "smit-batch-10",
12  storageBucket: "smit-batch-10.appspot.com",
13  messagingSenderId: "333213670128",
14  appId: "1:333213670128:web:b7d83af5afed979bafcc035"
15 };
16
```

```
index.html  JS app.js  X  # style.css
JS app.js > register
21 const register = () => {
22   const email = document.getElementById("email");
23   const password = document.getElementById("password");
24
25   console.log(email.value, password.value)
26
27   createUserWithEmailAndPassword(auth, email, password)
28   .then((userCredential) => {
29     // Signed up
30     const user = userCredential.user;
31     // ...
32   })
33   .catch((error) => {
34     const errorCode = error.code;
35     const errorMessage = error.message;
36     // ..
37   });
38
39 }
```

Same goes for sign in

Ji haan, aap bilkul apne project ki HTML file me ek `<script>` tag use kar sakte hain aur usme external JavaScript file ko import kar sakte hain by specifying `type="module"`. Is tareeke se, aap modular JavaScript code likh sakte hain, jo Firebase ke functions ko bhi use kar sake. Ye process modern browsers me supported hai, aur isse aap Firebase authentication ko ya kisi bhi Firebase service ko integrate kar sakte hain without directly writing all the logic in HTML.

Chaliye step by step samjhte hain kaise aap isko implement kar sakte hain:

1. HTML File Mein Script Tag Dalna

HTML file me aap apni JavaScript file ka reference is tarah de sakte hain:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Firebase Example</title>
</head>
<body>
  <h1>Firebase Authentication</h1>

  <!-- Linking the external JS file with type="module" -->
  <script type="module" src="app.js"></script>
</body>
</html>
```

2. JavaScript File (app.js)

Aap apne JavaScript code ko ek external file app.js me likhenge. Yahan par Firebase ko as a module use karna bahut simple hai:

```
// Firebase SDK ko import karna (Make sure Firebase SDK scripts are included in the project)
import { initializeApp } from "https://www.gstatic.com/firebasejs/9.15.0/firebase-app.js";
import { getAuth, signInWithEmailAndPassword } from "https://www.gstatic.com/firebasejs/9.15.0/firebase-auth.js";

// Aapki Firebase configuration
const firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_PROJECT_ID.firebaseio.com",
  projectId: "YOUR_PROJECT_ID",
  storageBucket: "YOUR_PROJECT_ID.appspot.com",
  messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
  appId: "YOUR_APP_ID"
};

// Firebase app initialize karna
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);

// User sign-in function
function signInUser(email, password) {
  signInWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
      const user = userCredential.user;
      console.log('Signed in:', user.email);
    })
    .catch((error) => {
      console.error('Error signing in:', error);
    });
}

// Example call
signInUser('test@example.com', '123456');
```

```
// User sign-in function
function signInUser(email, password) {
  signInWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
      const user = userCredential.user;
      console.log('Signed in:', user.email);
    })
    .catch((error) => {
      console.error('Error signing in:', error);
    });
}

// Example call
signInUser('test@example.com', '123456');
```

Explanation:

- **type="module"**: Yeh tag aapko JavaScript modules ko import/export karne ki facility deta hai. Isse aap multiple JS files ko apas me link kar sakte hain.
- **import**: Firebase SDK ka specific parts ko import karne ke liye import statement use hota hai (jaise `getAuth`, `signInWithEmailAndPassword`, etc.).
- **Firebase Initialization**: Aapko Firebase ka configuration aur initialization karna hoga, jisme API keys, project ID, etc. aati hain.
- **Function**: `signInUser()` ek function hai jo Firebase ke `signInWithEmailAndPassword()` ko use kar raha hai user ko sign-in karwane ke liye.

Benefits:

- **Separation of Concerns**: Aapka HTML aur JavaScript code alag alag hoga, jo ki ek best practice hai.
- **Modular Code**: Firebase ke specific methods ko use karne ke liye, aap sirf wohi parts import kar rahe hain jo zarurat hain, isliye code lightweight rahega.
- **Modern Approach**: `type="module"` modern browsers me supported hai, jo ki module-based coding ko enable karta hai.

Notes:

1. Make sure ki aapka project correct Firebase SDK version use kar raha ho.
2. HTML file ko agar directly browser me open karein, to Firebase ke kuch features local file se chalne me issues ho sakte hain, isliye project ko local server pe run karna behtar rahega (like Live Server extension in VS Code).
