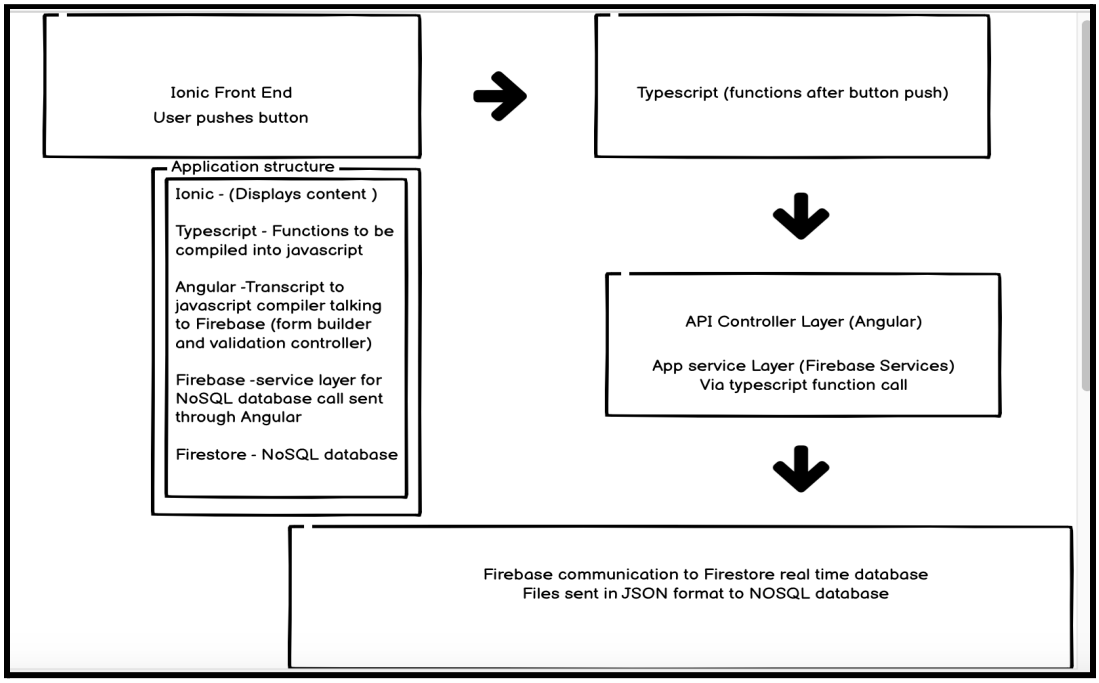
Design: Service Layer

Kurtis Henry
SWDV 691

For my service layers, I have decided to use Firebase for its firestore real-time database integration and Firebase Authentication for my password and login by email features on the app:

What is Firebase?
Firebase is a service layer within my app that is responsible for doing background computation and allows for more time to fine-tune user experience with the Rio Blog Builder app and its database. Although there are limitations in the usage of each service provided by Firebase, I have observed that its services work well for a small to mid-sized application. Firebase has various components. As mentioned in my database design, I am using the firebase service layer to include a Firestore realtime database.

App Architecture in layers:
Validator and Firebase service result (page 4)
Firebase Authentication service (page 5)



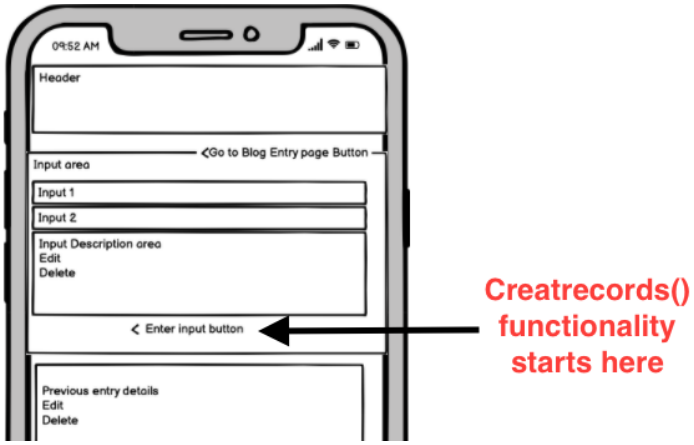
Firestore within my Firebase service layer:
C - Create Blog functionality
R - Read Blog functionality
U - Update functionality
D - Delete functionality

```
@Injectable({
  providedIn: 'root'
})
export class FirebaseService {

  collectionName = 'BlogEntries';

  constructor(
    private firestore: AngularFirestore
  ) { }

  create_blog(record) {
    console.log(record);

    return this.firestore.collection(this.collectionName).add(record);
  }

  read_blog() {
    return this.firestore.collection(this.collectionName).snapshotChanges();
  }

  update_blog(recordID, record) {
    this.firestore.doc(this.collectionName + '/' + recordID).update(record);
  }

  delete_blog(record_id) {
    this.firestore.doc(this.collectionName + '/' + record_id).delete();
  }
}
```

Each of the following functions within my typescript file will communicate with my service layer when a user selects an option. Each CRUD call (via a user pressing a button) will be handled by a function within the Typescript file that sends the call to Firebase/Firestore.
Front facing app will have create, delete, edit, and update buttons.
Corresponding Tyspript:

Error thrown:
If the record fails to create, Console.log will throw an error message***
See **CreateRecord()**
***validators in the angular control layer ensure input is entered in order for the entry to be saved

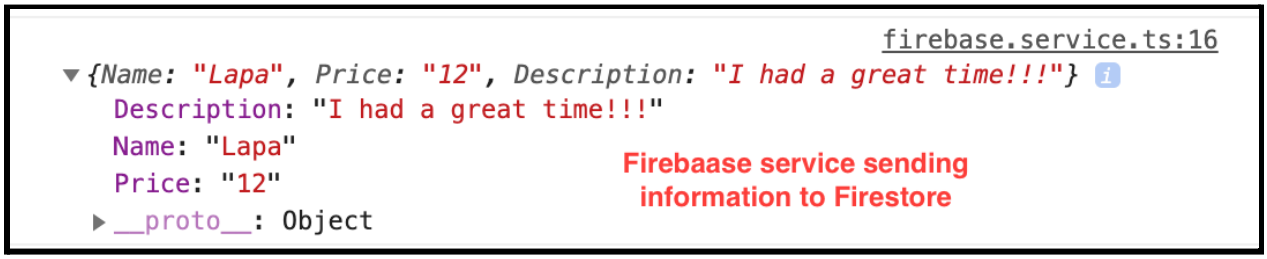
```
  this.firebaseService.read_blog().subscribe(data => {

    this.blogList = data.map(e => {
      return {
        id: e.payload.doc.id,
        isEdit: false,
        Name: e.payload.doc.data()['Name'],
        Price: e.payload.doc.data()['Price'],
        Description: e.payload.doc.data()['Description'],
      };
    })

  });
}

CreateRecord() {
  this.firebaseService.create_blog(this.blogForm.value)
    .then(resp => {
      //Reset form
      this.blogForm.reset();
    })
    .catch(error => {
      console.log(error);
    });
}

RemoveRecord(rowID) {
  this.firebaseService.delete_blog(rowID);
}

EditRecord(record) {
  record.isEdit = true;
  record.EditName = record.Name;
  record.EditPrice= record.Price;
  record.EditDescription= record.Description;
}

UpdateRecord(recordRow) {
  let record = {};
  record['Name'] = recordRow.EditName;
  record['Price'] = recordRow.EditPrice;
  record['Description'] = recordRow.EditDescription;
  this.firebaseService.update_blog(recordRow.id, record);
  recordRow.isEdit = false;
}
```



Example of HTML code call for **CreateRecord()**:

```
<ion-item>
  <ion-button (click)="CreateRecord()" [disabled]="blogForm.invalid">
    <ion-icon size="small" slot="icon-only" name="add"></ion-icon>
     Add Blog Entry
  </ion-button>
</ion-item>
```

Createrecord() Database entry with validator controller example:




```
this.blogForm = this.fb.group({
  Name: ['', [Validators.required]],
  Price: ['', [Validators.required]],
  Description: ['', [Validators.required]]
})
```

Validators from API layer

FireBase Authentication Service Layer:

Angular Firestore Authentication will be added as a service layer and will handle logins and registrations. Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to the Rio Blogger app. It supports authentication using passwords, phone numbers, popular identity providers like Google, Facebook and Twitter, and more. For the project, I will use a password and email for authentication.

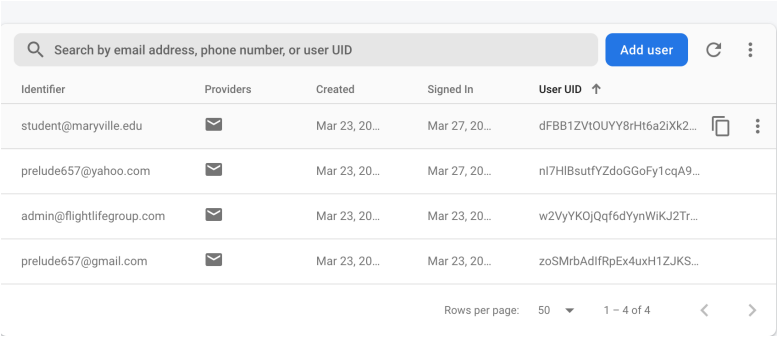
FireBase Authentication Injection:

```
import { AngularFireAuth } from '@angular/fire/auth';

@Injectable({
  providedIn: 'root'
})
export class AuthenticateService {

  constructor(
    private afAuth: AngularFireAuth
  ) { }

  registerUser(value) {
    return new Promise<any>((resolve, reject) => {

      this.afAuth.createUserWithEmailAndPassword(value.email, value.password)
        .then(
          res => resolve(res),
          err => reject(err))
    })

  }
```

FireBase Authentication

Registration Authentication

Firebase not only provides ready-made email authentication but also provides authentication using a variety of social logins. Within Firebase, users can be manually added,deleted, and passwords can be reset by admin. Firebase is a great Backend-as-a-Service (BaaS) platform that has allowed me to easily understand the integration process within my final project.



(continued on next page…..)

I will continue to use angular form validation in order to ensure users input the proper information. Users will not be able to proceed unless form validation requirements are met.

```
validations_form: FormGroup;
errorMessage: string = '';
successMessage: string = '';

validation_messages = {
  'email': [
    { type: 'required', message: 'Email is required.' },
    { type: 'pattern', message: 'Enter a valid email.' }
  ],
  'password': [
    { type: 'required', message: 'Password is required.' },
    { type: 'minlength', message: 'Password must be at least 5 characters long.' }
  ]
};
```