SimpliLearn

Assesment project 2: KitchenStory

Source code

Index.html

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>kitchen Club</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">

    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
href="https://fonts.googleapis.com/css2?family=Raleway:ital,wght@0,300;0,500;1,600&
display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap"
rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css2?family=Karla:ital,wght@0,200;0,300;0,400;0,50
0;0,600;0,700;1,200;1,300;1,400;1,500;1,600;1,700&display=swap" rel="stylesheet">

    <link rel="stylesheet" type="text/css"
href="//code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css" />

    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
```

```
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>


</head>

<body>
    <app-root></app-root>
</body>

</html>
```

aad-or-editItems.html

```
<div class="container add-edit-div">
    <div class="row">

        <div class="d-flex flex-row-reverse" style="margin-bottom: 30px;">
            <div class="p-2">
                <button type="button" class="btn btn-outline-primary"
[routerLink]="['/admin/items']">
                    <i class="fa fa-arrow-left"></i>
                    <span style="margin-left: 3px; margin-bottom: 10px;">
                        Display all items
                    </span>
                </button>
            </div>
        </div>

        <div class="col-xs-12 col-md-6 col-md-offset-3">
            <h2 style="margin-bottom: 40px;">{{selectedPath | titlecase }} Item</h2>

            <form [formGroup]="addOrEditItemsForm" (ngSubmit)="onSubmit()">
                <div (click)="hideResponseTexts()">
                    <div class="form-group">
```

```
                    <label for="exampleInputEmail1">Title</label>
                    <input type="itemTitle" class="form-control"
formControlName="itemTitle">
                    <span class="error-text"
*ngIf="addOrEditItemsForm.controls.itemTitle.errors != null &&
addOrEditItemsForm.controls.itemTitle.touched">Please enter a title for this
item.</span>
                </div>
                <div class="form-group">
                    <label for="exampleInputEmail1">Description</label>
                    <input type="itemDesc" class="form-control"
formControlName="itemDesc">
                </div>
                <div class="form-group">
                    <label for="exampleInputEmail1">Price (in ₹)</label>
                    <input type="itemPrice" class="form-control"
formControlName="itemPrice">
                    <span class="error-text"
*ngIf="addOrEditItemsForm.controls.itemPrice.errors != null &&
addOrEditItemsForm.controls.itemPrice.touched">Please enter a valid amount.</span>
                </div>
                <div class="form-group">
                    <label for="exampleInputEmail1">Category</label>
                    <select class="form-select" aria-label="Default select example"
formControlName="itemCategory">
                        <option value="starters">Starters</option>
                        <option value="mains">Mains</option>
                        <option value="alcoholic-beverages">Alcoholic Beverages</option>
                        <option value="desserts">Desserts</option>
                    </select>
                    <span class="error-text"
*ngIf="addOrEditItemsForm.controls.itemCategory.errors != null &&
addOrEditItemsForm.controls.itemCategory.touched">Please select a category.</span>
                </div>
            </div>
            <div class="form-group">
                <label>Thumbnail image</label>

                <div class="form-group mobile-only" *ngIf="previewPath != ""
style="margin-top: 15px;">
```

```
                    <div class="container-fluid no-padding">
                        <img [src]=previewPath alt="" class="img-thumbnail" />
                    </div>

                    <div class="text-center">
                        <button type="button" class="btn btn-md btn-outline-danger"
style="margin-top: 25px" *ngIf="showDeleteBtn == true && previewPath != ""
(click)="onDeleteImage()">Delete image</button>
                    </div>
                </div>

                <input type="file" class="form-control text-nowrap text-truncate"
*ngIf="!isUploaded" accept="image/png, image/jpeg, image/jpg"
formControlName="itemImage" (change)="selectFile($event)">

                <span class="error-text"
*ngIf="addOrEditItemsForm.controls.itemImage.errors != null &&
addOrEditItemsForm.controls.itemImage.touched">Please set and upload an image for
this item.</span>
            </div>

            <div *ngIf="isUploading" class="progress mt-2" style="margin-bottom: 15px;">
                <div class="progress-bar progress-bar-animated bg-warning"
role="progressbar" attr.aria-valuenow="{{ uploadPercentage }}" aria-valuemin="0" aria-
valuemax="100" [ngStyle]="{ width: uploadPercentage + '%' }">
                    {{ uploadPercentage }}%
                </div>
            </div>

            <div class="alert alert-danger" role="alert" *ngIf="(selectedFile) &&
(!isImage)">
                Please select an .jpg, .jpeg or .png file only.
            </div>

            <div class="alert alert-danger" role="alert" *ngIf="(selectedFile) && (isImage)
&& (fileSizeExceeded)">
                Please upload an image smaller than 2 MB.
            </div>

            <div class="alert alert-success" role="alert" *ngIf="onSuccessText != "">
```

```
            {{onSuccessText}}
          </div>

          <div class="alert alert-danger" role="alert" *ngIf="unknownErrorText != ''">
            <h4>Some error occurred.</h4>
            {{unknownErrorText}}
          </div>

          <div class="form-group">
            <button type="submit" *ngIf="isAdd == true && isUploading == false"
style="margin-right: 10px" class="btn btn-md btn-primary"
[disabled]="(!addOrEditItemsForm.valid) || isUploading || fileSizeExceeded ||
!isImage">{{submitBtnText}}</button>

            <button type="submit" *ngIf="isEdit == true" class="btn btn-md btn-primary"
[disabled]="isUploading || fileSizeExceeded || !isImage">{{submitBtnText}}</button>

            <button type="button" class="btn btn-md btn-outline-danger" style="margin-
left: 10px" *ngIf="showDeleteBtn == true" (click)="onDeleteItem()">Delete item</button>

            <button type="button" class="btn btn-md btn-primary" *ngIf="(isSubmitted
== true && isAdd == true) || (unknownErrorText != '' && isAdd == true)"
(click)="addAnotherItem()">{{addAnotherItemBtnText}}</button>
          </div>

        </form>
      </div>


      <div class="col-xs-12 col-md-6 col-md-offset-3 on-mobile-hide" *ngIf="previewPath
!= ''">
        <div class="container-fluid no-padding">
          <div class="row">
            <div class="col-md-12">
              <img [src]=previewPath alt="" class="img-thumbnail" />
            </div>
          </div>
        </div>

        <div class="text-center">
```

```
            <button type="button" class="btn btn-md btn-outline-danger" style="margin-
top: 25px" *ngIf="showDeleteBtn == true && previewPath != ""
(click)="onDeleteImage()">Delete image</button>
          </div>
        </div>

    </div>
</div>
```

add-or-edititems.ts

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { ActivatedRoute, Data, NavigationStart, Router } from '@angular/router';
import { Subscription } from 'rxjs';
import { filter, map } from 'rxjs/operators';
import { Item } from 'src/app/models/item.model';
import { ItemImageService } from 'src/app/services/item-image.service';
import { ItemDataService } from 'src/app/services/item-data.service';

@Component({
  selector: 'app-add-or-edit-items',
  templateUrl: './add-or-edit-items.component.html',
  styleUrls: ['./add-or-edit-items.component.css'],
})
export class AddOrEditItemsComponent implements OnInit {
  addOrEditItemsForm: FormGroup;
  selectedPath: string;

  selectedFile: FileList = null;
  file: File | null;
  uploadPercentage = -1;
  imageUrl: string;
  itemId: string;
  itemCategory: string;

  isAdd: boolean = false;
  isEdit: boolean = false;
  isUploaded: boolean = false;
```

```
isUploading: boolean = false;
isImage: boolean = false;
isSubmitted: boolean = false;
fileSizeExceeded: boolean = false;
showDeleteBtn: boolean = false;

onSuccessText: string = '';
submitBtnText: string;
addAnotherItemBtnText: string;
unknownErrorText: string = '';
previewPath: string = '';

imageUrlSub: Subscription;

validImageTypes: string[] = ['image/png', 'image/jpeg', 'image/jpg'];

item: Item = {
  id: '',
  name: '',
  description: '',
  price: 0,
  category: '',
  imageUrl: '',
  addedOn: '',
  modifiedOn: '',
  isAvailable: true
};

constructor(
  private router: Router,
  private route: ActivatedRoute,
  private itemImageService: ItemImageService,
  private itemDataService: ItemDataService
) {
  //
  this.isAdd = false;
  this.isEdit = false;
  this.isUploaded = false;
  this.isUploading = false;
  this.isImage = false;
```

```
  this.isSubmitted = false;
  this.fileSizeExceeded = false;
  this.showDeleteBtn = false;
  this.selectedFile = null;
  //
}

ngOnInit(): void {
  // creating reactive signup form
  this.addOrEditItemsForm = new FormGroup({
    itemTitle: new FormControl('', [Validators.required]),
    itemDesc: new FormControl(''),
    itemPrice: new FormControl('', [
      Validators.required,
      Validators.pattern('^-?[0-9]+([,.]?[0-9]+)?$'),
    ]),
    itemCategory: new FormControl('', [Validators.required]),
    itemImage: new FormControl(''),
  });

  // get data from route
  this.route.data.subscribe((data: Data) => {
    this.selectedPath = data['path'];
    this.submitBtnText =
      this.selectedPath === 'edit' ? 'Update item' : 'Add item';
    this.isAdd = this.selectedPath === 'add' ? true : false;
    this.isEdit = this.selectedPath === 'edit' ? true : false;
  });

  // get value of path variables from route
  this.route.params.subscribe((value) => {
    this.itemId = value['itemId'];
    this.itemCategory = value['itemCategory'];
  });

  // if it is edit
  if (this.isEdit == true) {
    this.initializeForItemEdit();
  }
}
```

```
async initializeForItemEdit() {
  this.item = (await this.itemDataService.getItemById(
    this.itemCategory,
    this.itemId
  )) as Item;

  this.addOrEditItemsForm.patchValue({
    itemTitle: this.replaceUndefinedOrNull(this.item.name),
    itemDesc: this.replaceUndefinedOrNull(this.item.description),
    itemPrice: this.replaceUndefinedOrNull(this.item.price),
    itemCategory: this.replaceUndefinedOrNull(this.item.category),
  });

  this.previewPath = this.item.imageUrl;
  this.imageUrl = this.item.imageUrl;
  this.fileSizeExceeded = false;
  this.isImage = true;
  this.showDeleteBtn = true;
}

// on selecting a file
selectFile(event: any): void {
  this.setFile(event);

  this.checkImageOrNot(this.file);

  if (this.isImage) {
    this.checkFileSize(this.file);
  }

  if (this.isImage == true && this.fileSizeExceeded == false) {
    this.previewImage();
  }
}

// on clicking submit
onSubmit() {
  // handle the case when disabled attribute for add item button is deleted
  // from html
```

```
  if (this.isAdd) {
    if (
      this.addOrEditItemsForm.invalid == true ||
      this.isImage == false ||
      this.fileSizeExceeded == true
    ) {
      return;
    }
  }

  // call to uploadImage() uploads the selected image and fetches the image URL
  // and then it pushes the item data to Firebase DB
  if (this.isAdd == true && this.selectedFile != null) {
    this.uploadImage();
  }

  // if a new image file is selected on edit page
  // we delete the previous image
  // and upload the new image
  // and update the item data
  if (
    this.selectedFile != null &&
    this.isEdit == true &&
    this.item.imageUrl == ''
  ) {
    this.uploadImage();
  } else if (
    this.selectedFile != null &&
    this.isEdit == true &&
    this.item.imageUrl != ''
  ) {
    this.performDeleteImage();
    this.uploadImage();
  } else if (this.selectedFile == null && this.isEdit) {
    this.updateItemData();
  }
}

private uploadImage(): void {
  const category = this.addOrEditItemsForm.get('itemCategory').value;
```

```
  this.isUploading = true;

  this.itemImageService.pushImageToStorage(this.file, category).subscribe(
    (percentage) => {
      this.uploadPercentage = Math.round(percentage ? percentage : 0);

      if (this.uploadPercentage == 100) {
        this.isUploaded = true;
      }
    },
    (error) => {
      this.unknownErrorText = error;
      this.isUploaded = false;
      this.isUploading = false;
    }
  );

  // get the image url from Firebase
  // then push item data to Firebase Realtime DB
  this.imageUrlSub = this.itemImageService
    .getimageUrlObservable()
    .subscribe((url) => {
      if (url != '' && url != null && url != undefined) {
        this.imageUrl = url;
        // unsubscribe here so that pushItemData() isn't called more than once
        this.imageUrlSub.unsubscribe();
        if (this.isAdd) {
          this.pushItemData();
        } else if (this.isEdit) {
          this.updateItemData();
        }
      }
    });
}

// saves item data in Firebase DB
private pushItemData() {
  this.item.name = this.addOrEditItemsForm.get('itemTitle').value;
  this.item.description = this.addOrEditItemsForm.get('itemDesc').value;
```

```
  this.item.price = this.addOrEditItemsForm.get('itemPrice').value;
  this.item.category = this.addOrEditItemsForm.get('itemCategory').value;
  this.item.imageUrl = this.imageUrl;
  this.item.addedOn = new Date().toLocaleString();

  this.itemDataService.addItemData(this.item).subscribe(
    (response) => {
      if (response != null) {
        this.isSubmitted = true;
        this.onSuccessText = 'Item added!';
        this.addAnotherItemBtnText = 'Add another item';
        // Firebase returns an unique identifier on adding some data via POST request
        // We fetch it from response.name
        this.itemDataService.setItemId(response.name);
      }
    },
    (error) => {
      this.unknownErrorText = error;
      this.isSubmitted = false;
      this.addAnotherItemBtnText = 'Try again?';
    }
  );
}

// updates item data
updateItemData() {
  this.item.name = this.addOrEditItemsForm.get('itemTitle').value;
  this.item.description = this.addOrEditItemsForm.get('itemDesc').value;
  this.item.price = this.addOrEditItemsForm.get('itemPrice').value;
  this.item.category = this.addOrEditItemsForm.get('itemCategory').value;
  this.item.imageUrl = this.imageUrl;
  this.item.modifiedOn = new Date().toLocaleString();

  this.performItemDataUpdate(this.item)
    .then((res) => {
      this.isSubmitted = true;
      this.onSuccessText = 'Item updated!';
    })
    .catch((error) => {
      this.unknownErrorText = error;
```

```
      this.isSubmitted = false;
    });
}

async performItemDataUpdate(item: Item) {
  await this.itemDataService.updateItemData(
    item,
    this.itemCategory,
    this.itemId
  );
}

addAnotherItem() {
  this.addOrEditItemsForm.reset();

  this.file = null;
  this.selectedFile = null;
  this.uploadPercentage = -1;
  this.isImage = false;
  this.isUploaded = false;
  this.isUploading = false;
  this.isSubmitted = false;
  this.fileSizeExceeded = false;

  if (this.isAdd) {
    this.submitBtnText = 'Add item';
  }

  if (this.isEdit) {
    this.submitBtnText = 'Update item';
  }

  this.unknownErrorText = '';
  this.previewPath = '';
  this.onSuccessText = '';

  this.item = {
    id: '',
    name: '',
    description: '',
```

```
    price: 0,
    category: '',
    imageUrl: '',
    addedOn: '',
    modifiedOn: '',
    isAvailable: true
  };
}

onDeleteImage() {
  if (this.isEdit != true && this.showDeleteBtn != true) {
    return;
  }
  this.performDeleteImage();
}

async performDeleteImage() {
  await this.itemImageService
    .deleteImage(this.item.imageUrl)
    .then(() => {
      // only when no image file selected
      // hide image preview
      if (this.selectedFile == null) {
        this.previewPath = '';
        this.showDeleteBtn = false;
        this.item.imageUrl = '';
      }
    })
    .catch(() => {
      this.unknownErrorText = 'Some error occurred while deleting image.';
    });

  await this.itemDataService.deleteImageUrl(this.item.category, this.itemId);
}

onDeleteItem() {
  if (this.isEdit != true && this.showDeleteBtn != true) {
    return;
  }
```

```
  this.performDeleteItem();
}

async performDeleteItem() {
  // first delete the thumbnail image and then delete the item data
  if (this.previewPath != '') {
    await this.performDeleteImage();
  }

  await this.itemDataService
    .deleteItemData(this.item.category, this.item.id)
    .then(() => {
      this.onSuccessText = 'Item deleted!';

      setTimeout(() => {
        this.router.navigate(['admin/items']);
      }, 1500);
    })
    .catch(() => {
      this.unknownErrorText = 'Some error occurred while deleting item.';
    });
}

/**
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */

/** Utility functions */

private setFile(event: any) {
  this.selectedFile = event.target.files;
```

```
  if (this.selectedFile) {
    this.file = this.selectedFile.item(0);
  }
}

// preview the selected image
// uses FileReader and reads the selected file
private previewImage() {
  const fileReader = new FileReader();
  fileReader.onload = () => {
    this.previewPath = fileReader.result as string;
  };
  fileReader.readAsDataURL(this.file);
}

private checkImageOrNot(file: File) {
  if (this.validImageTypes.indexOf(file.type) != -1) {
    this.isImage = true;
  } else {
    this.isImage = false;
  }
}

private checkFileSize(file: File) {
  const limit = 2048;

  if (Math.round(file.size / 1024) > limit) {
    this.fileSizeExceeded = true;
  } else {
    this.fileSizeExceeded = false;
  }
}

hideResponseTexts() {
  this.unknownErrorText = '';
  this.onSuccessText = '';
}

replaceUndefinedOrNull(v: any): string {
  if (v == undefined || v == null) {
```

```
      return '';
    }
    return v;
  }
}
```

add-or-edititems.css

```
.add-edit-div {
    padding-top: 70px;
    padding-bottom: 180px;
    min-height: 70vh;
}

.form-group {
    margin-bottom: 20px;
}

label {
    font-size: 20px;
    font-weight: 400;
}
```

displayItem.html

```
<main>
    <div class="album py-5">
        <div class="container">

            <div class="d-flex flex-row-reverse" style="margin-bottom: 30px;" #startersRef>
                <div class="p-2">
                    <button type="button" class="btn btn-outline-primary"
(click)="onAdd()">Add a new item</button>
                </div>
            </div>

            <nav class="sticky-top category-navbar" *ngIf="showCategoryNavbar == true">
```

```
            <div class="container">
                <ul class="nav nav-pills nav-justified text-light category-ul-nav">
                    <li class="nav-item">
                        <a class="nav-link" [ngClass]="{'active': currentActive === 1}"
(click)="scrollTo(startersRef, 1)">
                            <app-starters-icon></app-starters-icon>
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" [ngClass]="{'active': currentActive === 2}"
(click)="scrollTo(mainsRef, 2)">
                            <app-mains-icon></app-mains-icon>
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" [ngClass]="{'active': currentActive === 3}"
(click)="scrollTo(dessertsRef, 3)">
                            <app-desserts-icon></app-desserts-icon>
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" [ngClass]="{'active': currentActive === 4}"
(click)="scrollTo(alcoholicBeveragesRef, 4)">
                            <app-drinks-icon></app-drinks-icon>
                        </a>
                    </li>
                </ul>
            </div>
        </nav>

        <div class="loader-div" *ngIf="isLoading == true && isLoaded == false">
            <app-loader></app-loader>
        </div>

        <section class="section" id="starters" style="scroll-margin-top: 0rem;">
            <div class="items-div" *ngIf="isLoaded == true">
                <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
                    <span class="fs-2">Starters</span>
                </div>
```

```
            <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
                <div class="col" *ngFor="let item of starters">
                    <div class="card">
                        <img class="img-loading card-img-top" src={{item.imageUrl}}
role="img" aria-label="item-image" preserveAspectRatio="xMidYMid slice"
focusable="false">

                        <div class="card-body">
                            <p class="card-text"><strong>{{item.name}}</strong></p>
                            <span class="badge bg-light text-dark item-
category">{{item.category}}</span>

                            <div class="d-flex justify-content-between align-items-center">
                                <small class="text-dark item-price">₹{{item.price}}</small>

                                <div class="btn-group">
                                    <div class="form-check form-switch" style="margin-right:
10px;">
                                        <input class="form-check-input" type="checkbox"
id="flexSwitchCheckDefault" [checked]="item.isAvailable"
(click)="setAvailabilityStatus(item)">
                                        <label class="form-check-label"
for="flexSwitchCheckDefault">Available?</label>
                                    </div>
                                    <button type="button" class="btn btn-sm btn-outline-
primary" (click)="onEdit(item.category, item.id)">Edit</button>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </section>

            <section class="section" id="mains" #mainsRef>
                <div class="items-div" *ngIf="isLoaded == true">
                    <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
                        <span class="fs-2">Mains</span>
                    </div>
```

```
            <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
                <div class="col" *ngFor="let item of mains">
                    <div class="card">
                        <img class="img-loading card-img-top" src={{item.imageUrl}}
role="img" aria-label="item-image" preserveAspectRatio="xMidYMid slice"
focusable="false">

                        <div class="card-body">
                            <p class="card-text"><strong>{{item.name}}</strong></p>
                            <span class="badge bg-light text-dark item-
category">{{item.category}}</span>

                            <div class="d-flex justify-content-between align-items-center">
                                <small class="text-dark item-price">₹{{item.price}}</small>

                                <div class="btn-group">
                                    <div class="form-check form-switch" style="margin-right:
10px;">
                                        <input class="form-check-input" type="checkbox"
id="flexSwitchCheckDefault" [checked]="item.isAvailable"
(click)="setAvailabilityStatus(item)">
                                        <label class="form-check-label"
for="flexSwitchCheckDefault">Available?</label>
                                    </div>
                                    <button type="button" class="btn btn-sm btn-outline-
primary" (click)="onEdit(item.category, item.id)">Edit</button>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </section>

        <section class="section" id="desserts" #dessertsRef>
            <div class="items-div" *ngIf="isLoaded == true">
                <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
                    <span class="fs-2">Desserts</span>
```

```
            </div>

            <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
                <div class="col" *ngFor="let item of desserts">
                    <div class="card">
                        <img class="img-loading card-img-top" src={{item.imageUrl}}
role="img" aria-label="item-image" preserveAspectRatio="xMidYMid slice"
focusable="false">

                        <div class="card-body">
                            <p class="card-text"><strong>{{item.name}}</strong></p>
                            <span class="badge bg-light text-dark item-
category">{{item.category}}</span>

                            <div class="d-flex justify-content-between align-items-center">
                                <small class="text-dark item-price">₹{{item.price}}</small>

                                <div class="btn-group">
                                    <div class="form-check form-switch" style="margin-right:
10px;">
                                        <input class="form-check-input" type="checkbox"
id="flexSwitchCheckDefault" [checked]="item.isAvailable"
(click)="setAvailabilityStatus(item)">
                                        <label class="form-check-label"
for="flexSwitchCheckDefault">Available?</label>
                                    </div>
                                    <button type="button" class="btn btn-sm btn-outline-
primary" (click)="onEdit(item.category, item.id)">Edit</button>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </section>

            <section class="section" id="alcoholic-beverages" #alcoholicBeveragesRef>
                <div class="items-div" *ngIf="isLoaded == true">
                    <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
```

```
                    <span class="fs-2">Alcoholic Beverages</span>
                </div>

                <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
                    <div class="col" *ngFor="let item of alcoholicBeverages">
                        <div class="card">
                            <img class="img-loading card-img-top" src={{item.imageUrl}}
role="img" aria-label="item-image" preserveAspectRatio="xMidYMid slice"
focusable="false">

                            <div class="card-body">
                                <p class="card-text"><strong>{{item.name}}</strong></p>
                                <span class="badge bg-light text-dark item-
category">{{item.category}}</span>

                                <div class="d-flex justify-content-between align-items-center">
                                    <small class="text-dark item-price">₹{{item.price}}</small>

                                    <div class="btn-group">
                                        <div class="form-check form-switch" style="margin-right:
10px;">
                                            <input class="form-check-input" type="checkbox"
id="flexSwitchCheckDefault" [checked]="item.isAvailable"
(click)="setAvailabilityStatus(item)">
                                            <label class="form-check-label"
for="flexSwitchCheckDefault">Available?</label>
                                        </div>
                                        <button type="button" class="btn btn-sm btn-outline-
primary" (click)="onEdit(item.category, item.id)">Edit</button>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </section>

        </div>
    </div>
```

```
</main>
```

DisplayItem.ts

```
import {
  AfterViewInit,
  Component,
  ElementRef,
  HostListener,
  OnInit,
  ViewChild,
} from '@angular/core';
import { ActivatedRoute, NavigationExtras, Router } from '@angular/router';
import { Item } from 'src/app/models/item.model';
import { ItemDataService } from 'src/app/services/item-data.service';

@Component({
  selector: 'app-display-items',
  templateUrl: './display-items.component.html',
  styleUrls: ['./display-items.component.css'],
})
export class DisplayItemsComponent implements OnInit, AfterViewInit {
  starters: Item[] = [];
  mains: Item[] = [];
  alcoholicBeverages: Item[] = [];
  desserts: Item[] = [];

  isLoading: boolean = false;
  isLoaded: boolean = false;

  sectionName: string = '';

  showCategoryNavbar: boolean = false;
  currentActive: number;
  @ViewChild('startersRef') startersRef: ElementRef;
  @ViewChild('mainsRef') mainsRef: ElementRef;
  @ViewChild('dessertsRef') dessertsRef: ElementRef;
  @ViewChild('alcoholicBeveragesRef') alcoholicBeveragesRef: ElementRef;
```

```
public startersOffset: Number = null;
public mainsOffset: Number = null;
public dessertsOffset: Number = null;
public alcoholicBeveragesOffset: Number = null;

constructor(
  private router: Router,
  private itemDataService: ItemDataService,
  private route: ActivatedRoute
) {
  this.route.fragment.subscribe((data) => {
    this.sectionName = data;
  });
}

ngOnInit(): void {
  this.fetchItems();
}

ngAfterViewInit() {
  this.startersOffset = this.startersRef.nativeElement.offsetTop;
  this.mainsOffset = this.mainsRef.nativeElement.offsetTop;
  this.dessertsOffset = this.dessertsRef.nativeElement.offsetTop;
  this.alcoholicBeveragesOffset =
    this.alcoholicBeveragesRef.nativeElement.offsetTop;
}

@HostListener('window:scroll', ['$event'])
checkOffsetTop() {
  this.startersOffset = this.startersRef.nativeElement.offsetTop+200;
  this.mainsOffset = this.mainsRef.nativeElement.offsetTop-200;
  this.dessertsOffset = this.dessertsRef.nativeElement.offsetTop-200;
  this.alcoholicBeveragesOffset =
    this.alcoholicBeveragesRef.nativeElement.offsetTop-200;

  if (
    window.pageYOffset >= this.startersOffset &&
    window.pageYOffset < this.mainsOffset
  ) {
    this.currentActive = 1;
```

```
    this.showCategoryNavbar = true;
  } else if (
    window.pageYOffset >= this.mainsOffset &&
    window.pageYOffset < this.dessertsOffset
  ) {
    this.currentActive = 2;
    this.showCategoryNavbar = true;
  } else if (
    window.pageYOffset >= this.dessertsOffset &&
    window.pageYOffset < this.alcoholicBeveragesOffset
  ) {
    this.currentActive = 3;
    this.showCategoryNavbar = true;
  } else if (window.pageYOffset >= this.alcoholicBeveragesOffset) {
    this.currentActive = 4;
    this.showCategoryNavbar = true;
  } else {
    this.currentActive = 0;
    this.showCategoryNavbar = false;
  }
}

scrollTo(el: HTMLElement, v: number) {
  //el.scrollIntoView({ block: 'start', inline: 'nearest' });
  el.scrollIntoView();
  this.currentActive = v;
}

async fetchItems() {
  try {
    this.isLoading = true;

    this.starters = await this.itemDataService.getItemsCategoryWise(
      'starters'
    );
    this.mains = await this.itemDataService.getItemsCategoryWise('mains');
    this.alcoholicBeverages = await this.itemDataService.getItemsCategoryWise(
      'alcoholic-beverages'
    );
    this.desserts = await this.itemDataService.getItemsCategoryWise(
```

```
        'desserts'
      );

      this.isLoading = false;
      this.isLoaded = true;
    } catch (error) {
      console.log(error);
    }
  }

  onEdit(itemCategory: string, itemId: string) {
    this.router.navigate(['admin/items/edit', itemCategory, itemId]);
  }

  onAdd() {
    this.router.navigate(['admin/items/add']);
  }

  setAvailabilityStatus(item: any) {
    const status = item.isAvailable;
    console.log(status);
    item.isAvailable = !status;

    this.itemDataService.setIsAvailable(!status, item.category, item.id);
  }
}
```

DisplayItem.css

```
section {
    scroll-margin-top: 6rem;
}

img {
    width: 100%;
    height: 300px;
    object-fit: cover;
}
```

```
.item-price {
    font-size: 18px;
    color: black;
}

.card {
    font-family: "Karla", sans-serif;
    border-radius: 1.3em;
}

.card-img-top {
    border-top-left-radius: 1.3em;
    border-top-right-radius: 1.3em;
}

.card-text {
    font-size: 20px;
    color: black;
}

.item-category {
    white-space: pre-wrap;
    font-size: 14px;
    margin-bottom: 10px;
}

.item-price {
    margin-right: 10px;
}

.loader-div {
    margin-top: 250px;
    min-height: 60vh;
}

.img-loading {
    background-color: #e2e2e2;
    background-image: linear-gradient(0deg, #e2e2e2 0%, #f7f7f7 100%);
}
```

```
div.card {
    box-shadow: rgba(149, 157, 165, 0.2) 0px 8px 24px;
}

div.card:hover {
    transform: translateY(-0.1rem) scale(1);
    box-shadow: 0 0.5em 3rem -1rem rgba(0, 0, 0, 0.5);
}

.items-div {
    margin-bottom: 80px;
}

.category-navbar {
    background-color: white;
    color: black;
    padding-top: 10px;
    padding-bottom: 10px;
    margin-top: 80px;
    margin-bottom: 80px;
    border-bottom-left-radius: 0.5em;
    border-bottom-right-radius: 0.5em;
    font-size: 18px;
    box-shadow: rgba(0, 0, 0, 0.15) 1.95px 1.95px 2.6px;
}

.nav-pills .nav-link.active,
.nav-pills .show>.nav-link {
    color: rgb(255, 255, 255);
    background-color: #e7e7e7d8;
}

.nav-link {
    color: #000000;
}


/*
##Device = Low Resolution Tablets, Mobiles (Landscape)
```

```
##Screen = B/w 481px to 767px
*/

@media (min-width: 481px) and (max-width: 767px) {
    .nav-link {
        padding-top: 0.5rem;
        padding-right: 0.5rem;
        padding-bottom: 0.5rem;
        padding-left: 0.5rem;
    }
    .category-navbar {
        font-size: 15px;
    }
}

.form-switch {
    word-break: break-word;
}

.form-check-label {
    font-size: 14px;
}

.form-check-input:checked {
    background-color: #33ea00;
    border-color: #33ea00;
}


/*
##Device = Most of the Smartphones Mobiles (Portrait)
##Screen = B/w 320px to 479px
*/

@media (min-width: 320px) and (max-width: 480px) {
    .nav-link {
        padding-top: 0.5rem;
        padding-right: 0.5rem;
        padding-bottom: 0.5rem;
        padding-left: 0.5rem;
```

```
    }
    .category-navbar {
        font-size: 15px;
    }
}
```

DisplayOrder.html

```
<main class="my-order-page">
    <div class="album py-5">
        <div class="container">

            <div class="d-flex flex-row-reverse" style="margin-bottom: 30px;">
                <div class="p-2">
                    <button type="button" class="btn btn-outline-primary"
[routerLink]="['/admin/manage-orders']">
                        <i class="fa fa-arrow-left"></i>
                        <span style="margin-left: 3px; margin-bottom: 10px;">
                            Back to manage orders
                        </span>
                    </button>
                </div>
            </div>

            <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
                <span class="fs-2">{{userName}}'s orders</span>
            </div>

            <div *ngIf="isLoading == true">
                <app-loader></app-loader>
            </div>

            <div class="text-center" *ngIf="isLoaded == true && orderArray.length == 0">
                <h3>The customer hasn't ordered anything yet.</h3>
            </div>

            <div *ngFor="let order of orderArray">
                <div class="card mx-auto" style="max-width: 700px;">
```

```
                <div class="card-header">
                    <strong>Order # {{order.orderNo}}</strong> placed on
{{order.addedOn}}
                </div>
                <div class="card-body">
                    <h5 class="card-title">ID: {{order.orderId}}</h5>
                    <div *ngFor="let oi of order.orderedItems">
                        <p class="card-text">
                            {{oi.name}}: ₹{{oi.price}} × {{oi.quantity}} =
₹{{getItemTotalAmount(oi.price, oi.quantity)}}
                        </p>
                    </div>
                    <hr> <strong>Total amount paid: ₹{{order.totalAmt}}</strong> (inclusive
of 18 %GST)
                </div>
            </div>
        </div>
    </div>
</main>
```

DisplayOrder.ts

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { count } from 'rxjs/operators';
import { Order } from 'src/app/models/order.model';
import { OrderDataService } from 'src/app/services/order-data.service';

@Component({
  selector: 'app-display-orders',
  templateUrl: './display-orders.component.html',
  styleUrls: ['./display-orders.component.css'],
})
export class DisplayOrdersComponent implements OnInit {
  orders: any;
  orderArray: any = [];
  isLoading: boolean = true;
  isLoaded: boolean = false;
```

```
userName: string;
uid: string;

constructor(
  private orderDataService: OrderDataService,
  private route: ActivatedRoute
) {
  this.userName = this.route.snapshot.queryParams['name']
  this.uid = this.route.snapshot.params['uid'];
}

ngOnInit(): void {
  this.fetchOrderData();
}

async fetchOrderData() {
  this.isLoaded = false;
  this.isLoading = true;

  this.orders = await this.orderDataService.getOrderDataById(
    this.uid
  );

  let count = 0;
  for (let orderId in this.orders) {
    count++;

    const orderObj: Order = this.orders[orderId];
    const oia = [];

    for (let oi in orderObj.orderedItems) {
      const o = {
        name: orderObj.orderedItems[oi].name,
        price: orderObj.orderedItems[oi].price,
        quantity: orderObj.orderedItems[oi].quantity,
      };
      oia.push(o);
    }

    const obj: any = {
```

```
        orderNo: count,
        orderId: orderObj.orderId,
        addedOn: orderObj.addedOn,
        orderedItems: oia,
        totalAmt: orderObj.totalAmt,
      };

      this.orderArray.push(obj);
    }

    // reverse it to show latest order first
    this.orderArray.reverse();

    this.isLoaded = true;
    this.isLoading = false;
  }

  getItemTotalAmount(price: number, quantity: number) {
    return Number(price) * Number(quantity);
  }
}
```

DisplayOrder.css

```
.my-order-page {
    min-height: 120vh;
    padding-top: 80px;
}

div {
    font-family: "Karla", sans-serif;
}

.card {
    margin-bottom: 20px;
    font-size: 18px;
}

.card-header {
    background-color: rgb(0 255 167 / 46%);
```

}

ManageOrder.html

```
<main class="manage-orders-page">
   <div class="album py-5">
      <div class="container">

         <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
            <span class="fs-2">Manage orders</span>
         </div>

         <div *ngIf="isLoading == true">
            <app-loader></app-loader>
         </div>

         <div *ngIf="isLoaded == true">

            <div class="row row-cols-1 row-cols-md-3 g-4">
               <div class="col" *ngFor="let user of usersDataArray">
                  <div class="card">
                     <div class="card-body">
                        <h5 class="card-title">{{user.name}}</h5>
                        <div class="card-text">
                           <ul>
                              <li><strong>Email:</strong> {{user.email}}</li>
                              <li><strong>Address:</strong> {{user.address}}</li>
                              <li><strong>User ID:</strong> {{user.uid}}</li>
                           </ul>
                        </div>
                     </div>
                     <div class="card-footer" (click)="goToOrders(user)">
                        Orders <i class="fa fa-angle-right"></i>
                     </div>
                  </div>
               </div>
            </div>
         </div>
      </div>
   </div>
</main>
```

ManageOrder.ts

```
import { Component, OnInit } from '@angular/core';
import { NavigationExtras, Router } from '@angular/router';
import { Order } from 'src/app/models/order.model';
import { User } from 'src/app/models/user.model';
import { OrderDataService } from 'src/app/services/order-data.service';
import { UserDataService } from 'src/app/services/user-data.service';

@Component({
  selector: 'app-manage-orders',
  templateUrl: './manage-orders.component.html',
  styleUrls: ['./manage-orders.component.css'],
})
export class ManageOrdersComponent implements OnInit {
  usersData: any;
  usersDataArray: any[] = [];

  isLoading: boolean = false;
  isLoaded: boolean = false;

  constructor(
    private userDataService: UserDataService,
    private orderDataService: OrderDataService,
    private router: Router
  ) {}

  ngOnInit(): void {
    this.fetchAllUsersData();
  }

  async fetchAllUsersData() {
    this.isLoading = true;

    this.usersData = await this.userDataService.getAllUsersData();
    this.formatUsersData();

    this.isLoading = false;
```

```
    this.isLoaded = true;
  }

  formatUsersData() {
    for (let obj in this.usersData) {
      this.usersDataArray.push(this.usersData[obj]);
    }
  }

  goToOrders(user: User) {
    const navObj: NavigationExtras = {
      queryParams: {name: user.name}
    }
    this.router.navigate(['admin', user.uid, 'orders'], navObj);
  }
}
```

ManageOrder.css

```
.manage-orders-page {
    min-height: 120vh;
    padding-top: 80px;
}

div {
    font-family: "Karla", sans-serif;
}

.card-footer:hover {
    cursor: pointer;
    background-color: rgb(46, 144, 255);
    color: white;
}
```

login.component.html

```
<main class="form-signin">

    <h1 class="h3 mb-3 fw-normal">Log in</h1>

    <form [formGroup]="logInForm" (ngSubmit)="onLogIn()">

        <div (click)="hideResponseErrors()">
            <div class="form-floating">
                <input type="email" class="form-control" [ngClass]="{'error-border':
!isHideResponseErrors && errorObj.logIn.email != null}" id="floatingInput"
placeholder="name@example.com" formControlName="email">
                <label class="floating-label" for="floatingInput">Email address</label>

                <span class="error-text" *ngIf="logInForm.controls.email.errors != null &&
logInForm.controls.email.touched">Please enter a valid email.</span>

                <span class="error-text" *ngIf="!isHideResponseErrors &&
errorObj.logIn.email != null">{{errorObj.logIn.email}}</span>
            </div>
            <div class="form-floating">
                <input type="password" class="form-control" [ngClass]="{'error-border':
!isHideResponseErrors && errorObj.logIn.password != null}" id="floatingPassword"
placeholder="Password" formControlName="password">
                <label class="floating-label" for="floatingPassword">Password</label>

                <span class="error-text" *ngIf="logInForm.controls.password.errors != null &&
logInForm.controls.password.touched">Password length must be greater than
8.</span>

                <span class="error-text" *ngIf="!isHideResponseErrors &&
errorObj.logIn.password != null" style="margin-top: -
100px;">{{errorObj.logIn.password}}</span>
            </div>
        </div>

        <button class="w-100 btn btn-lg btn-primary" type="submit"
[disabled]="!logInForm.valid || isBtnClicked" style="margin-top: 10px;">
```

```
            <i class="fa fa-arrow-right" *ngIf="!isLoggingIn"></i>
            <span class="spinner-border spinner-border-sm" role="status" aria-
hidden="true" *ngIf="isLoggingIn"></span>
        </button>

    </form>

    <button class="w-100 btn btn-lg btn-primary google-btn" type="button"
(click)="onLogInWithGoogle()" style="margin-bottom: 25px;">
        <span class="google-btn-text">Log in with Google</span>
    </button>

    <span class="unknown-error-text" *ngIf="errorObj.logIn.unknown !=
null">{{errorObj.logIn.unknown}}</span>

    <p class="mobile-only">Don't have an account? <a [routerLink]="['/sign-up']">Sign
up</a></p>

</main>
```

login.comopenent.ts

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { Subscription } from 'rxjs';
import { AuthErrorHandlerService } from 'src/app/services/auth-error-handler.service';
import { AuthService } from 'src/app/services/auth.service';
import { UserDataService } from 'src/app/services/user-data.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
})
export class LoginComponent implements OnInit, OnDestroy {
  logInForm: FormGroup;
```

```
email: string;
password: string;

errorObj: any;
isLoggingIn: boolean = false;
isBtnClicked: boolean = false;
isHideResponseErrors: boolean = true;
errorSub: Subscription;

constructor(
  private authService: AuthService,
  private router: Router,
  private authErrorHandler: AuthErrorHandlerService,
  private userDataService: UserDataService
) {
  // creating a subscription to listen to the subject in authService
  // so that we get updated whenever the errorObj changes
  this.errorSub = authErrorHandler.getErrorObservable().subscribe((data) => {
    this.errorObj = data;
  });

  // the errorObj needs to be initialized here
  // calls the next method on subject in authService
  // and we get the initial errorObj data here
  this.authErrorHandler.initializeErrorObj();
}

ngOnInit(): void {
  // creating reactive signup form
  this.logInForm = new FormGroup({
    email: new FormControl('', [Validators.required, Validators.email]),
    password: new FormControl('', [
      Validators.required,
      Validators.minLength(8),
    ]),
  });

  // reset the errorObj
  // so that previous errors don't come up in view
  this.errorObj = {
```

```
    logIn: {
      errorFound: null,
      email: null,
      password: null,
      unknown: null,
    },
    signUp: {
      errorFound: null,
      name: null,
      email: null,
      password: null,
      unknown: null,
    },
  };
}

ngOnDestroy() {
  this.errorSub.unsubscribe();
}

/** method binded to form ngSubmit event */
onLogIn() {
  // handle the case when disabled attribute for submit button is deleted
  // from html
  if (this.logInForm.invalid) {
    return;
  }

  this.isBtnClicked = true;
  this.isLoggingIn = true;

  this.email = this.logInForm.get('email').value;
  this.password = this.logInForm.get('password').value;

  this.authService
    .signIn(this.email, this.password)
    .then((result) => {
      this.isLoggingIn = false;
      this.isHideResponseErrors = true;
```

```
      this.router.navigate(['']);
    })
    .catch((error) => {
      this.isBtnClicked = false;
      this.isHideResponseErrors = false;
      this.isLoggingIn = false;
      this.authErrorHandler.handleAuthError(error, 'logIn');
    });
}

/** on clicking log in with google */
onLogInWithGoogle() {
  this.authService
    .authenticateWithGoogle()
    .then((result) => {
      // save user data for a first time user only
      if (result.additionalUserInfo.isNewUser == true) {
        this.userDataService.createNewUser(
          result.user.displayName,
          result.user.email,
          result.user.uid
        );
      }

      this.router.navigate(['']);
    })
    .catch((error) => {
      console.log(error);
      this.authErrorHandler.handleAuthError(error, 'logIn');
    });
}

/** hides error messages on input click */
hideResponseErrors() {
  if (
    this.authErrorHandler.foundLogInError &&
    this.isHideResponseErrors === false
  ) {
    this.isHideResponseErrors = !this.isHideResponseErrors;
    this.authErrorHandler.clearLogInError();
```

```
    }
  }
}
```

Login.component.css

```
.form-signin {
    width: 100%;
    max-width: 330px;
    margin: auto;
    padding-top: 150px;
    padding-bottom: 180px;
    min-height: 70vh;
}

.floating-label {
    color: rgb(109, 109, 109)
}

.form-signin .checkbox {
    font-weight: 400;
}

.form-signin .form-floating:focus-within {
    z-index: 2;
}

.form-signin input[type="email"] {
    margin-bottom: -1px;
    border-bottom-right-radius: 0;
    border-bottom-left-radius: 0;
}

.form-signin input[type="password"] {
    margin-bottom: 1px;
    border-top-left-radius: 0;
    border-top-right-radius: 0;
}

button[type="submit"]:disabled {
```

```
    background-color: rgb(173, 173, 173);
    border-color: inherit;
}

input[type="email"].ng-invalid.ng-touched {
    border: 1px solid rgb(255, 83, 83);
}

input[type="password"].ng-invalid.ng-touched {
    border: 1px solid rgb(255, 83, 83);
    margin-bottom: -1px;
}

.error-text {
    color: rgb(255, 83, 83);
    font-size: 13px;
    margin-left: 13px;
}

.unknown-error-text {
    color: rgb(255, 83, 83);
    font-size: 13px;
    margin-top: 10px;
}

.error-border {
    border: 1px solid rgb(255, 83, 83);
}

.google-btn {
    margin-top: 10px;
}

.google-btn-text {
    font-size: 16px;
}
```

Signup.component.html

```
<main class="form-signin">

   <h1 class="h3 mb-3 fw-normal">Sign up</h1>

   <form [formGroup]="signUpForm" (ngSubmit)="onSignUp()">

      <div class="form-floating">
         <input type="name" class="form-control" id="name" placeholder="Sreya Saha"
formControlName="name">
         <label class="floating-label" for="floatingInput">Name</label>
         <span class="error-text" *ngIf="signUpForm.controls.name.errors != null &&
signUpForm.controls.name.touched">What's your name?</span>
      </div>

      <div class="form-floating" (click)="hideResponseErrors()">
         <input type="email" class="form-control" [ngClass]="{'error-border':
!isHideResponseErrors && errorObj.signUp.email != null}" id="email"
placeholder="name@example.com" formControlName="email">
         <label class="floating-label" for="floatingInput">Email address</label>

         <span class="error-text" *ngIf="signUpForm.controls.email.errors != null &&
signUpForm.controls.email.touched">Please enter a valid email.</span>

         <span class="error-text" *ngIf="!isHideResponseErrors &&
errorObj.signUp.email != null">{{errorObj.signUp.email}}</span>
      </div>

      <div class="form-floating">
         <input type="password" class="form-control" id="password"
placeholder="Password" name="password" formControlName="password">
         <label class="floating-label" for="floatingPassword">Password</label>
         <span class="error-text" *ngIf="signUpForm.controls.password.errors != null &&
signUpForm.controls.password.touched">Password length must be greater than
8.</span>
      </div>
```

```
        <button class="w-100 btn btn-lg btn-primary" type="submit"
[disabled]="!signUpForm.valid || isBtnClicked || isSignedUp">
            <i class="fa fa-arrow-right" *ngIf="!isSigningUp && !isSignedUp"></i>
            <span class="spinner-border spinner-border-sm" role="status" aria-
hidden="true" *ngIf="isSigningUp"></span>
            <i class="fa fa-check" aria-hidden="true" *ngIf="isSignedUp &&
isBtnClicked"></i>
        </button>

    </form>

    <button class="w-100 btn btn-lg btn-primary google-btn" type="button"
(click)="onSignUpWithGoogle()" style="margin-bottom: 25px;">
        <span class="google-btn-text">Sign up with Google</span>
    </button>

    <span class="unknown-error-text" *ngIf="errorObj.signUp.unknown !=
null">{{errorObj.signUp.unknown}}</span>

    <p class="mobile-only">Have an account? <a [routerLink]="['/login']">Log in</a></p>

</main>
```

Signup.component.ts

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { Subscription } from 'rxjs';
import { AuthErrorHandlerService } from 'src/app/services/auth-error-handler.service';
import { AuthService } from 'src/app/services/auth.service';
import { UserDataService } from 'src/app/services/user-data.service';

@Component({
  selector: 'app-signup',
  templateUrl: './signup.component.html',
  styleUrls: ['./signup.component.css'],
})
export class SignupComponent implements OnInit, OnDestroy {
  signUpForm: FormGroup;
```

```
name: string;
email: string;
password: string;

errorObj: any;
isBtnClicked: boolean = false;
isSignedUp: boolean = false;
isHideResponseErrors: boolean = true;
isSigningUp: boolean = false;
errorSub: Subscription;

constructor(
  private authService: AuthService,
  private router: Router,
  private authErrorHandler: AuthErrorHandlerService,
  private userDataService: UserDataService
) {
  // creating a subscription to listen to the subject in authService
  // so that we get updated whenever the errorObj changes
  this.errorSub = authErrorHandler.getErrorObservable().subscribe((data) => {
    this.errorObj = data;
  });

  // calls the next method on subject in authService
  // and we get the errorObj data here
  this.authErrorHandler.initializeErrorObj();
}

ngOnInit(): void {
  // creating reactive signup form
  this.signUpForm = new FormGroup({
    name: new FormControl('', Validators.required),
    email: new FormControl('', [Validators.required, Validators.email]),
    password: new FormControl('', [
      Validators.required,
      Validators.minLength(8),
    ]),
  });
```

```
  // reset the errorObj
  // so that previous errors don't come up in view
  this.errorObj = {
    logIn: {
      errorFound: null,
      email: null,
      password: null,
      unknown: null,
    },
    signUp: {
      errorFound: null,
      name: null,
      email: null,
      password: null,
      unknown: null,
    },
  };
}

ngOnDestroy() {
  this.errorSub.unsubscribe();
}

/** method binded to form ngSubmit event */
onSignUp() {
  // handle the case when disabled attribute for submit button is deleted
  // from html
  if (this.signUpForm.invalid) {
    return;
  }

  this.isBtnClicked = true;
  this.isSigningUp = true;

  this.name = this.signUpForm.get('name').value;
  this.email = this.signUpForm.get('email').value;
  this.password = this.signUpForm.get('password').value;

  this.authService
    .signUp(this.email, this.password)
```

```
    .then((result) => {
      this.isSignedUp = true;
      this.isSigningUp = false;
      this.isHideResponseErrors = true;

      this.userDataService.createNewUser(
        this.name,
        this.email,
        result.user.uid
      );

      setTimeout(() => {
        this.router.navigate(['']);
      }, 1500);
    })
    .catch((error) => {
      this.isSignedUp = false;
      this.isSigningUp = false;
      this.isBtnClicked = false;
      this.isHideResponseErrors = false;

      this.authErrorHandler.handleAuthError(error, 'signUp');
    });
}

/** on clicking sign up with google */
onSignUpWithGoogle() {
  this.authService
    .authenticateWithGoogle()
    .then((result) => {
      // save user data only the first time
      if (result.additionalUserInfo.isNewUser == true) {
        this.userDataService.createNewUser(
          result.user.displayName,
          result.user.email,
          result.user.uid
        );
      }

      setTimeout(() => {
```

```
          this.router.navigate(['']);
        }, 1500);
      })
      .catch((error) => {
        this.authErrorHandler.handleAuthError(error, 'signUp');
      });
  }

  /** hides error messages on input click */
  hideResponseErrors() {
    if (
      this.authErrorHandler.foundSignUpError &&
      this.isHideResponseErrors === false
    ) {
      this.isHideResponseErrors = !this.isHideResponseErrors;
      this.authErrorHandler.clearSignUpError();
    }
  }
}
```

Signup.component.css

```
.form-signin {
    width: 100%;
    max-width: 330px;
    margin: auto;
    padding-top: 150px;
    padding-bottom: 180px;
    min-height: 70vh;
}

.floating-label {
    color: rgb(109, 109, 109)
}

.form-signin .checkbox {
    font-weight: 400;
}

.form-signin .form-floating:focus-within {
```

```
    z-index: 2;
}

.form-signin input[type="name"] {
    margin-bottom: -1px;
    border-top-left-radius: 0;
    border-top-right-radius: 0;
}

.form-signin input[type="email"] {
    margin-bottom: -1px;
    border-bottom-right-radius: 0;
    border-bottom-left-radius: 0;
}

.form-signin input[type="password"] {
    margin-bottom: 10px;
    border-top-left-radius: 0;
    border-top-right-radius: 0;
}

button[type="submit"]:disabled {
    background-color: rgb(173, 173, 173);
    border-color: inherit;
}

input[type="name"].ng-invalid.ng-touched {
    border: 1px solid rgb(255, 83, 83);
}

input[type="email"].ng-invalid.ng-touched {
    border: 1px solid rgb(255, 83, 83);
}

input[type="password"].ng-invalid.ng-touched {
    border: 1px solid rgb(255, 83, 83);
    margin-bottom: -1px;
}

.error-text {
    color: rgb(255, 83, 83);
    font-size: 13px;
    margin-left: 13px;
}
```

```
.error-border {
    border: 1px solid rgb(255, 83, 83);
}

.unknown-error-text {
    color: rgb(255, 83, 83);
    font-size: 13px;
    margin-top: 10px;
}

.google-btn {
    margin-top: 10px;
}

.google-btn-text {
    font-size: 16px;
}
```

Cart.component.html

```
<nav class="navbar fixed-bottom navbar-expand-lg" id="cart-bar" *ngIf="isCartEmpty == false
&& hideCartBar == false">
    <div class="container" id="cart-bar-div ">
        <span class="navbar-brand ">Cart ({{totalItems}})</span>

        <div>
            <span id="subtotal">Subtotal: ₹{{totalAmt}}</span>
            <button type="button" class="btn btn-outline-danger" id="view-cart-btn"
*ngIf="goToOrders==false" (click)="onContinue()">Continue</button>
            <button type="button" class="btn btn-outline-success" id="view-cart-btn"
*ngIf="goToOrders==true" (click)="placeOrder()">Place order</button>
        </div>
    </div>
</nav>
```

Cart.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { HandleCartService } from '../services/handle-cart.service';
import { HandleLocalStorageService } from '../services/handle-local-storage.service';
```

```
@Component({
  selector: 'app-cart',
  templateUrl: './cart.component.html',
  styleUrls: ['./cart.component.css'],
})
export class CartComponent implements OnInit {
  isCartEmpty: boolean = true;
  totalAmt: number;
  totalItems: number;
  goToOrders: boolean = false;
  hideCartBar: boolean = false;

  constructor(
    private handleCartService: HandleCartService,
    private handleLocalStorage: HandleLocalStorageService,
    private router: Router,
    private route: ActivatedRoute
  ) {
    this.handleLocalStorage.getCartDataObservable().subscribe((data) => {
      // here data is cart data object
      if (data != null && Object.keys(data.items).length > 0) {
        this.isCartEmpty = false;
        this.totalAmt = data.totalAmt;
        this.totalItems = Object.keys(data.items).length;
      }

      if (data == null) {
        this.isCartEmpty = true;
      }
    });
  }

  ngOnInit(): void {
    this.handleCartService.onCartPageObs().subscribe((data) => {
      this.goToOrders = data;
    });

    this.handleCartService.onConfirmOrderPageObs().subscribe((data) => {
      this.hideCartBar = data;
    });
  }

  onContinue() {
    this.router.navigate(['cart']);
```

```
  }

  placeOrder() {
    this.router.navigate(['confirm-order']);
  }
}
```

Cart.component.css

```
#subtotal {
    padding-right: 11px;
}

#clear-cart-btn {
    margin-right: 6px;
}

#cart-bar {
    background-color: white;
    box-shadow: rgba(0, 0, 0, 0.25) 0px 54px 55px, rgba(0, 0, 0, 0.12) 0px -12px 30px, rgba(0, 0, 0, 0.12) 0px 4px 6px, rgba(0, 0, 0, 0.17) 0px 12px 13px, rgba(0, 0, 0, 0.09) 0px -3px 5px;
    color: black;
    /** below css is to make cart bar sticky and always on top without obstructing others*/
    position: sticky;
    font-family: "Karla", sans-serif;
    font-size: 20px;
}


/*
  ##Device = Desktops
  ##Screen = 1281px to higher resolution desktops
*/

@media (min-width: 1281px) {
    /* CSS */
}


/*
    ##Device = Laptops, Desktops
    ##Screen = B/w 1025px to 1280px
  */
```

```
@media (min-width: 1025px) and (max-width: 1280px) {
    /* CSS */
}


/*
    ##Device = Tablets, Ipads (portrait)
    ##Screen = B/w 768px to 1024px
  */

@media (min-width: 768px) and (max-width: 1024px) {
    /* CSS */
}


/*
    ##Device = Tablets, Ipads (landscape)
    ##Screen = B/w 768px to 1024px
  */

@media (min-width: 768px) and (max-width: 1024px) and (orientation: landscape) {
    /* CSS */
}


/*
    ##Device = Low Resolution Tablets, Mobiles (Landscape)
    ##Screen = B/w 481px to 767px
  */

@media (min-width: 481px) and (max-width: 767px) {
    .navbar-brand {
        font-size: 17px;
    }
    #subtotal {
        font-size: 16px;
    }
    #cart-bar-div {
        font-size: 14px;
    }
    #clear-cart-btn {
        font-size: 14px;
    }
    #view-cart-btn {
```

```
        font-size: 15px;
    }
}


/*
    ##Device = Most of the Smartphones Mobiles (Portrait)
    ##Screen = B/w 320px to 479px
  */

@media (min-width: 320px) and (max-width: 480px) {
    .navbar-brand {
        font-size: 17px;
    }
    #subtotal {
        font-size: 16px;
    }
    #cart-bar-div {
        font-size: 14px;
    }
    #clear-cart-btn {
        font-size: 14px;
    }
    #view-cart-btn {
        font-size: 15px;
    }
}
```

CartPage.component.html

```
<main class="cart-page">
    <div class="album py-5">
        <div class="container">

            <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
                <span class="fs-2">Your Cart</span>
            </div>

            <div class="text-center" *ngIf="isCartEmpty == true">
                <h3>No items in cart.</h3>
            </div>

            <div *ngIf="isCartEmpty == false">
                <div *ngFor="let item of cartArray">
```

```
            <div class="card mb-3 mx-auto" style="max-width: 700px;" *ngIf="item.quantity >
0">
                <div class="row g-0">
                    <div class="col-md-4">
                        <img class="item-image" [src]="item.imageUrl">
                    </div>
                    <div class="col-md-8">
                        <div class="card-body">
                            <h5 class="card-title">{{item.name}}</h5>
                            <span class="badge bg-light text-dark item-
category">{{item.category}}</span>
                            <p class="card-text item-price">
                                ₹{{item.price}} × {{item.quantity}} = ₹{{getItemTotalAmount(item.price,
item.quantity)}}
                            </p>
                            <div class="btn-toolbar" role="toolbar" *ngIf="item.quantity > 0">
                                <div class="btn-group" role="group" aria-label="-">
                                    <span type="button" class="btn btn-outline-danger btn-sm"
(click)="onRemove(item)">-</span>
                                    <span class="text-center" style="width:
30px;">{{[(item.quantity)]}}</span>
                                    <span type="button" class="btn btn-outline-success btn-sm"
(click)="onAdd(item)">+</span>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            <div class="text-center cart-btn-group">
                <div class="form-group">
                    <button class="btn btn-outline-danger" (click)="clearCart()">Clear cart</button>
                </div>
            </div>

        </div>
    </div>
</div>
</main>
```

CartPage.component.ts

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Cart } from '../models/cart.model';
import { HandleCartService } from '../services/handle-cart.service';
import { HandleLocalStorageService } from '../services/handle-local-storage.service';

@Component({
  selector: 'app-cart-page',
  templateUrl: './cart-page.component.html',
  styleUrls: ['./cart-page.component.css'],
})
export class CartPageComponent implements OnInit, OnDestroy {
  cartObj: Cart;
  cartArray: any[] = [];
  isCartEmpty: boolean = true;

  constructor(
    private handleLocalStorageService: HandleLocalStorageService,
    private handleCartService: HandleCartService,
    private router: Router
  ) {
    this.cartObj = JSON.parse(this.handleLocalStorageService.getCartData());
  }

  ngOnInit(): void {
    this.populateCartData();
    // hide bottom cart bar when viewing cart page
    this.handleCartService.goToOrders(true);
  }

  ngOnDestroy() {
    // show bottom cart bar when cart page component is destroyed
   this.handleCartService.goToOrders(false);
  }

  populateCartData() {
    if (this.cartObj != null && this.cartObj.items != undefined) {
      this.isCartEmpty = false;
      const itemD = this.cartObj.items;
```

```
    for (let item in itemD) {
      const itemObj = itemD[item];

      const obj = {
        id: itemObj.itemId,
        name: itemObj.name,
        category: itemObj.category,
        price: itemObj.price,
        imageUrl: itemObj.imageUrl,
        quantity: itemObj.quantity,
      };

      this.cartArray.push(obj);
    }
  } else {
    this.isCartEmpty = true;
  }
}

/** add to cart */
onAdd(item: any) {
  item.quantity += 1; //two-way binded
  this.handleCartService.addOrUpdate(item);
}

/** remove from cart */
onRemove(item: any) {
  item.quantity -= 1; //two-way binded
  this.handleCartService.removeItem(item);

  // if not items in cart
  // set isCartEmpty to true
  this.cartObj = JSON.parse(this.handleLocalStorageService.getCartData());
  if(this.cartObj == null) {
    this.isCartEmpty = true;
  }
}

getItemTotalAmount(price: number, quantity: number) {
  return Number(price) * Number(quantity);
}

clearCart() {
  this.isCartEmpty = true;
```

```
    this.cartArray = [];
    this.cartObj = null;
    this.handleCartService.clearCart();
  }
}
```

CartPage.component.css

```
div {
    font-family: "Karla", sans-serif;
}

.cart-page {
    min-height: 120vh;
    padding-top: 80px;
}

img {
    width: 100%;
    height: 200px;
    object-fit: cover;
}

.card-title {
    font-size: 25px;
}

.item-category {
    font-size: 15px;
}

.item-price {
    font-size: 19px;
    color: rgb(68, 68, 68);
}

.card {
    font-family: "Karla", sans-serif;
    border-radius: 1.3em;
}

div.card {
    box-shadow: rgba(149, 157, 165, 0.2) 0px 8px 24px;
}
```

```
div.card:hover {
    transform: translateY(-0.1rem) scale(1);
    box-shadow: 0 0.5em 3rem -1rem rgba(0, 0, 0, 0.5);
}

.item-category {
    white-space: pre-wrap;
    font-size: 14px;
    margin-bottom: 10px;
}

.cart-btn-group {
    margin-top: 80px;
    margin-bottom: 100px;
}


/*
  ##Device = Desktops
  ##Screen = 1281px to higher resolution desktops
*/

@media (min-width: 1281px) {
    .item-image {
        border-top-left-radius: 1.3em;
        border-bottom-left-radius: 1.3em;
    }
}


/*
    ##Device = Laptops, Desktops
    ##Screen = B/w 1025px to 1280px
  */

@media (min-width: 1025px) and (max-width: 1280px) {
    .item-image {
        border-top-left-radius: 1.3em;
        border-bottom-left-radius: 1.3em;
    }
}
```

```
/*
  ##Device = Tablets, Ipads (portrait)
  ##Screen = B/w 768px to 1024px
*/

@media (min-width: 768px) and (max-width: 1024px) {
    .item-image {
        border-top-left-radius: 1.3em;
        border-bottom-left-radius: 1.3em;
    }
}


/*
    ##Device = Tablets, Ipads (landscape)
    ##Screen = B/w 768px to 1024px
  */

@media (min-width: 768px) and (max-width: 1024px) and (orientation: landscape) {
    .item-image {
        border-top-left-radius: 1.3em;
        border-bottom-left-radius: 1.3em;
    }
}


/*
    ##Device = Low Resolution Tablets, Mobiles (Landscape)
    ##Screen = B/w 481px to 767px
  */

@media (min-width: 481px) and (max-width: 767px) {
    .item-image {
        border-top-left-radius: 1.3em;
        border-top-right-radius: 1.3em;
    }
}


/*
    ##Device = Most of the Smartphones Mobiles (Portrait)
    ##Screen = B/w 320px to 479px
  */
```

```
@media (min-width: 320px) and (max-width: 480px) {
   .item-image {
      border-top-left-radius: 1.3em;
      border-top-right-radius: 1.3em;
   }
}
```

CategoryPage.component.html

```
<main>
   <div class="album py-5">
      <div class="container">

         <div class="d-flex flex-row-reverse" style="margin-bottom: 30px;" #startersRef>
            <div class="p-2">

            </div>
         </div>

         <nav class="sticky-top category-navbar" *ngIf="showCategoryNavbar == true">
            <div class="container">
               <ul class="nav nav-pills nav-justified text-light category-ul-nav">
                  <li class="nav-item">
                     <a class="nav-link" [ngClass]="{'active': currentActive === 1}"
(click)="scrollTo(startersRef, 1)">
                        <app-starters-icon></app-starters-icon>
                     </a>
                  </li>
                  <li class="nav-item">
                     <a class="nav-link" [ngClass]="{'active': currentActive === 2}"
(click)="scrollTo(mainsRef, 2)">
                        <app-mains-icon></app-mains-icon>
                     </a>
                  </li>
                  <li class="nav-item">
                     <a class="nav-link" [ngClass]="{'active': currentActive === 3}"
(click)="scrollTo(dessertsRef, 3)">
                        <app-desserts-icon></app-desserts-icon>
                     </a>
                  </li>
                  <li class="nav-item">
                     <a class="nav-link" [ngClass]="{'active': currentActive === 4}"
(click)="scrollTo(alcoholicBeveragesRef, 4)">
                        <app-drinks-icon></app-drinks-icon>
```

```
                    </a>
                </li>
            </ul>
        </div>
    </nav>

    <div class="loader-div" *ngIf="isLoading == true && isLoaded == false">
        <app-loader></app-loader>
    </div>

    <section class="section" id="starters" style="scroll-margin-top: 0rem;">
        <div class="items-div" *ngIf="isLoaded == true">
            <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
                <span class="fs-2">Starters</span>
            </div>

            <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
                <div class="col" *ngFor="let item of starters">
                    <div class="card">
                        <img class="img-loading card-img-top" [ngClass]="{'not-available':
!item.isAvailable}" src={{item.imageUrl}} role="img" aria-label="item-image"
preserveAspectRatio="xMidYMid slice" focusable="false">

                        <div class="card-body">
                            <p class="card-text"><strong>{{item.name}}</strong></p>

                            <div class="d-flex justify-content-between align-items-center">
                                <small class="text-dark item-price">₹{{item.price}}</small>

                                <div class="btn-group" *ngIf="item.isAvailable == true">
                                    <span class="btn btn-sm" *ngIf="item.quantity == 0"
(click)="onAdd(item)">
                                        <app-cart-icon></app-cart-icon>
                                    </span>

                                    <div class="btn-toolbar" role="toolbar" *ngIf="item.quantity > 0">
                                        <div class="btn-group" role="group" aria-label="-">
                                            <span type="button" class="btn btn-outline-danger btn-sm"
(click)="onRemove(item)">-</span>
                                            <span class="text-center" style="width:
30px;">{{[(item.quantity)]}}</span>
                                            <span type="button" class="btn btn-outline-success btn-sm"
(click)="onAdd(item)">+</span>
                                        </div>
```

```
                        </div>
                    </div>
                    <div *ngIf="item.isAvailable == false">
                        <span class="not-available-text">Not Available.</span>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

<section class="section" id="mains" #mainsRef>
    <div class="items-div" *ngIf="isLoaded == true">
        <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
            <span class="fs-2">Mains</span>
        </div>

        <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
            <div class="col" *ngFor="let item of mains">
                <div class="card">
                    <img class="img-loading card-img-top" [ngClass]="{'not-available':
!item.isAvailable}" src={{item.imageUrl}} role="img" aria-label="item-image"
preserveAspectRatio="xMidYMid slice" focusable="false">

                    <div class="card-body">
                        <p class="card-text"><strong>{{item.name}}</strong></p>

                        <div class="d-flex justify-content-between align-items-center">
                            <small class="text-dark item-price">₹{{item.price}}</small>

                            <div class="btn-group" *ngIf="item.isAvailable == true">
                                <span class="btn btn-sm" *ngIf="item.quantity == 0"
(click)="onAdd(item)">
                                    <app-cart-icon></app-cart-icon>
                                </span>

                                <div class="btn-toolbar" role="toolbar" *ngIf="item.quantity > 0">
                                    <div class="btn-group" role="group" aria-label="-">
                                        <span type="button" class="btn btn-outline-danger btn-sm"
(click)="onRemove(item)">-</span>
                                        <span class="text-center" style="width:
30px;">{{[(item.quantity)]}}</span>
```

```
                                    <span type="button" class="btn btn-outline-success btn-sm"
(click)="onAdd(item)">+</span>
                                </div>
                            </div>
                        </div>
                        <div *ngIf="item.isAvailable == false">
                            <span class="not-available-text">Not Available.</span>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>

    <section class="section" id="desserts" #dessertsRef>
        <div class="items-div" *ngIf="isLoaded == true">
            <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
                <span class="fs-2">Desserts</span>
            </div>

            <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
                <div class="col" *ngFor="let item of desserts">
                    <div class="card">
                        <img class="img-loading card-img-top" [ngClass]="{'not-available':
!item.isAvailable}" src={{item.imageUrl}} role="img" aria-label="item-image"
preserveAspectRatio="xMidYMid slice" focusable="false">

                        <div class="card-body">
                            <p class="card-text"><strong>{{item.name}}</strong></p>
                            <!-- <span class="badge bg-light text-dark item-
category">{{item.category}}</span> -->

                            <div class="d-flex justify-content-between align-items-center">
                                <small class="text-dark item-price">₹{{item.price}}</small>

                                <div class="btn-group" *ngIf="item.isAvailable == true">
                                    <span class="btn btn-sm" *ngIf="item.quantity == 0"
(click)="onAdd(item)">
                                        <app-cart-icon></app-cart-icon>
                                    </span>

                                    <div class="btn-toolbar" role="toolbar" *ngIf="item.quantity > 0">
```

```
                                    <div class="btn-group" role="group" aria-label="-">
                                        <span type="button" class="btn btn-outline-danger btn-sm"
(click)="onRemove(item)">-</span>
                                        <span class="text-center" style="width:
30px;">{{[(item.quantity)]}}</span>
                                        <span type="button" class="btn btn-outline-success btn-sm"
(click)="onAdd(item)">+</span>
                                    </div>
                                </div>
                            </div>
                            <div *ngIf="item.isAvailable == false">
                                <span class="not-available-text">Not Available.</span>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </section>

        <section class="section" id="alcoholic-beverages" #alcoholicBeveragesRef>
            <div class="items-div" *ngIf="isLoaded == true">
                <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
                    <span class="fs-2">Alcoholic Beverages</span>
                </div>

                <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
                    <div class="col" *ngFor="let item of alcoholicBeverages">
                        <div class="card">
                            <img class="img-loading card-img-top" [ngClass]="{'not-available':
!item.isAvailable}" src={{item.imageUrl}} role="img" aria-label="item-image"
preserveAspectRatio="xMidYMid slice" focusable="false">

                            <div class="card-body">
                                <p class="card-text"><strong>{{item.name}}</strong></p>
                                <!-- <span class="badge bg-light text-dark item-
category">{{item.category}}</span> -->

                                <div class="d-flex justify-content-between align-items-center">
                                    <small class="text-dark item-price">₹{{item.price}}</small>

                                    <div class="btn-group" *ngIf="item.isAvailable == true">
```

```
                                <span class="btn btn-sm" *ngIf="item.quantity == 0"
(click)="onAdd(item)">
                                    <app-cart-icon></app-cart-icon>
                                </span>

                                <div class="btn-toolbar" role="toolbar" *ngIf="item.quantity > 0">
                                    <div class="btn-group" role="group" aria-label="-">
                                        <span type="button" class="btn btn-outline-danger btn-sm"
(click)="onRemove(item)">-</span>
                                        <span class="text-center" style="width:
30px;">{{[(item.quantity)]}}</span>
                                        <span type="button" class="btn btn-outline-success btn-sm"
(click)="onAdd(item)">+</span>
                                    </div>
                                </div>
                            </div>
                            <div *ngIf="item.isAvailable == false">
                                <span class="not-available-text">Not Available.</span>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </section>

    </div>
  </div>
</main>
```

CategoryPage.component.ts

```
import {
  AfterViewInit,
  Component,
  ElementRef,
  HostListener,
  OnInit,
  ViewChild,
} from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { Cart } from '../models/cart.model';
import { ItemDetails } from '../models/item-details.model';
```

```
import { Item } from '../models/item.model';
import { HandleCartService } from '../services/handle-cart.service';
import { ItemDataService } from '../services/item-data.service';

@Component({
  selector: 'app-category-page',
  templateUrl: './category-page.component.html',
  styleUrls: ['./category-page.component.css'],
})
export class CategoryPageComponent implements OnInit, AfterViewInit {
  _starters: Item[] = [];
  _mains: Item[] = [];
  _alcoholicBeverages: Item[] = [];
  _desserts: Item[] = [];

  // merged item data with cart data
  starters: any[] = [];
  mains: any[] = [];
  alcoholicBeverages: any[] = [];
  desserts: any[] = [];

  isLoading: boolean = false;
  isLoaded: boolean = false;

  // for cart
  cartData: Cart;

  sectionName: string = '';

  showCategoryNavbar: boolean = false;
  currentActive: number;
  @ViewChild('startersRef') startersRef: ElementRef;
  @ViewChild('mainsRef') mainsRef: ElementRef;
  @ViewChild('dessertsRef') dessertsRef: ElementRef;
  @ViewChild('alcoholicBeveragesRef') alcoholicBeveragesRef: ElementRef;

  public startersOffset: Number = null;
  public mainsOffset: Number = null;
  public dessertsOffset: Number = null;
  public alcoholicBeveragesOffset: Number = null;

  constructor(
    private router: Router,
    private itemDataService: ItemDataService,
```

```
  private route: ActivatedRoute,
  private handleCartService: HandleCartService
) {
  this.route.fragment.subscribe((data) => {
    this.sectionName = data;
  });
}

ngOnInit(): void {
  this.fetchItems();
}

/** for controlling the change of nav-pills in navbar on scroll position */
ngAfterViewInit() {
  this.startersOffset = this.startersRef.nativeElement.offsetTop;
  this.mainsOffset = this.mainsRef.nativeElement.offsetTop;
  this.dessertsOffset = this.dessertsRef.nativeElement.offsetTop;
  this.alcoholicBeveragesOffset =
    this.alcoholicBeveragesRef.nativeElement.offsetTop;
}

@HostListener('window:scroll', ['$event'])
checkOffsetTop() {
  this.startersOffset = this.startersRef.nativeElement.offsetTop + 200;
  this.mainsOffset = this.mainsRef.nativeElement.offsetTop - 200;
  this.dessertsOffset = this.dessertsRef.nativeElement.offsetTop - 200;
  this.alcoholicBeveragesOffset =
    this.alcoholicBeveragesRef.nativeElement.offsetTop - 200;

  if (
    window.pageYOffset >= this.startersOffset &&
    window.pageYOffset < this.mainsOffset
  ) {
    this.currentActive = 1;
    this.showCategoryNavbar = true;
  } else if (
    window.pageYOffset >= this.mainsOffset &&
    window.pageYOffset < this.dessertsOffset
  ) {
    this.currentActive = 2;
    this.showCategoryNavbar = true;
  } else if (
    window.pageYOffset >= this.dessertsOffset &&
    window.pageYOffset < this.alcoholicBeveragesOffset
```

```
  ) {
    this.currentActive = 3;
    this.showCategoryNavbar = true;
  } else if (window.pageYOffset >= this.alcoholicBeveragesOffset) {
    this.currentActive = 4;
    this.showCategoryNavbar = true;
  } else {
    this.currentActive = 0;
    this.showCategoryNavbar = false;
  }
}

scrollTo(el: HTMLElement, v: number) {
  //el.scrollIntoView({ block: 'start', inline: 'nearest' });
  el.scrollIntoView();
  this.currentActive = v;
}

/** ---------  */

async fetchItems() {
  try {
    this.isLoading = true;

    this._starters = await this.itemDataService.getItemsCategoryWise(
      'starters'
    );
    this._mains = await this.itemDataService.getItemsCategoryWise('mains');
    this._alcoholicBeverages =
      await this.itemDataService.getItemsCategoryWise('alcoholic-beverages');
    this._desserts = await this.itemDataService.getItemsCategoryWise(
      'desserts'
    );

    this.mergeItemAndCartData();

    this.isLoading = false;
    this.isLoaded = true;
  } catch (error) {
    console.log(error);
  }
}

fetchCartData() {
```

```
  this.cartData = this.handleCartService.getCartData();
}

mergeItemAndCartData() {
  this.fetchCartData();

  for (let key in this._starters) {
    let count = 0;
    const id = this._starters[key].id;

    if (this.cartData != null) {
      const itemDetailsObj = this.cartData.items[id];
      if (
        itemDetailsObj != undefined &&
        itemDetailsObj.quantity != undefined
      ) {
        count = this.cartData.items[id].quantity;
      }
    }

    this.starters[key] = { ...this._starters[key], quantity: count };
  }

  for (let key in this._mains) {
    let count = 0;
    const id = this._mains[key].id;

    if (this.cartData != null) {
      const itemDetailsObj = this.cartData.items[id];
      if (
        itemDetailsObj != undefined &&
        itemDetailsObj.quantity != undefined
      ) {
        count = this.cartData.items[id].quantity;
      }
    }

    this.mains[key] = { ...this._mains[key], quantity: count };
  }

  for (let key in this._desserts) {
    let count = 0;
    const id = this._desserts[key].id;
```

```
    if (this.cartData != null) {
      const itemDetailsObj = this.cartData.items[id];
      if (
        itemDetailsObj != undefined &&
        itemDetailsObj.quantity != undefined
      ) {
        count = this.cartData.items[id].quantity;
      }
    }

    this.desserts[key] = { ...this._desserts[key], quantity: count };
  }

  for (let key in this._alcoholicBeverages) {
    let count = 0;
    const id = this._alcoholicBeverages[key].id;

    if (this.cartData != null) {
      const itemDetailsObj = this.cartData.items[id];
      if (
        itemDetailsObj != undefined &&
        itemDetailsObj.quantity != undefined
      ) {
        count = this.cartData.items[id].quantity;
      }
    }

    this.alcoholicBeverages[key] = {
      ...this._alcoholicBeverages[key],
      quantity: count,
    };
  }
}

/** add to cart */
onAdd(item: any) {
  item.quantity += 1; //two-way binded
  this.handleCartService.addOrUpdate(item);
}

/** remove from cart */
onRemove(item: any) {
  item.quantity -= 1; //two-way binded
  this.handleCartService.removeItem(item);
```

```
  }
}
```

CategoryPage.component.css

```
section {
    scroll-margin-top: 6rem;
}

img {
    width: 100%;
    height: 300px;
    object-fit: cover;
}

.item-price {
    font-size: 20px;
    color: black;
}

.card {
    font-family: "Karla", sans-serif;
    border-radius: 1.3em;
}

.card-img-top {
    border-top-left-radius: 1.3em;
    border-top-right-radius: 1.3em;
}

.card-text {
    font-size: 20px;
    color: black;
}

.item-category {
    white-space: pre-wrap;
    font-size: 14px;
    margin-bottom: 10px;
}

.item-price {
    margin-right: 10px;
}
```

```
.loader-div {
    margin-top: 250px;
    min-height: 60vh;
}

.img-loading {
    background-color: #e2e2e2;
    background-image: linear-gradient(0deg, #e2e2e2 0%, #f7f7f7 100%);
}

div.card {
    box-shadow: rgba(149, 157, 165, 0.2) 0px 8px 24px;
}

div.card:hover {
    transform: translateY(-0.1rem) scale(1);
    box-shadow: 0 0.5em 3rem -1rem rgba(0, 0, 0, 0.5);
}

.items-div {
    margin-bottom: 80px;
}

.category-navbar {
    background-color: white;
    color: black;
    padding-top: 10px;
    padding-bottom: 10px;
    margin-top: 80px;
    margin-bottom: 80px;
    border-bottom-left-radius: 0.5em;
    border-bottom-right-radius: 0.5em;
    font-size: 18px;
    box-shadow: rgba(0, 0, 0, 0.15) 1.95px 1.95px 2.6px;
}

.nav-pills .nav-link.active,
.nav-pills .show>.nav-link {
    color: rgb(255, 255, 255);
    background-color: #e7e7e7d8;
}

.nav-link {
```

```
    color: #000000;
}

.btn-sm,
.btn-group-sm>.btn {
    padding: 0.0rem 0.6rem;
    font-size: 0.875rem;
    border-radius: 0.2rem;
    font-size: 17px;
}

.not-available {
    filter: grayscale(1);
}

.not-available-text {
    font-size: 14px;
    color: red;
}


/*
##Device = Low Resolution Tablets, Mobiles (Landscape)
##Screen = B/w 481px to 767px
*/

@media (min-width: 481px) and (max-width: 767px) {
    .nav-link {
        padding-top: 0.5rem;
        padding-right: 0.5rem;
        padding-bottom: 0.5rem;
        padding-left: 0.5rem;
    }
    .category-navbar {
        font-size: 15px;
    }
}


/*
##Device = Most of the Smartphones Mobiles (Portrait)
##Screen = B/w 320px to 479px
*/
```

```
@media (min-width: 320px) and (max-width: 480px) {
   .nav-link {
      padding-top: 0.5rem;
      padding-right: 0.5rem;
      padding-bottom: 0.5rem;
      padding-left: 0.5rem;
   }
   .category-navbar {
      font-size: 15px;
   }
}
```

ConfirmOrder.ccomponent.html

```
<main class="confirm-order-page">
   <div class="album py-5">
      <div class="container">

         <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
            <span class="fs-2">Confirm order</span>
         </div>

         <div class="text-center" *ngIf="orderArray.length == 0">
            <h3>You need to first add items to your cart.</h3>
         </div>

         <div *ngIf="orderArray.length > 0">
            <div *ngFor="let item of orderArray">
               <div class="card mb-3 mx-auto" style="max-width: 700px;">
                  <div class="row g-0">
                     <div class="col-md-8">
                        <div class="card-body">
                           <h5 class="card-title">{{item.name}}</h5>
                           <p class="card-text item-price">
                              ₹{{item.price}} × {{item.quantity}} = ₹{{getItemTotalAmount(item.price,
item.quantity)}}
                           </p>
                        </div>
                     </div>
                  </div>
               </div>
            </div>

            <div class="text-center" style="margin-top: 80px; margin-bottom: 30px;">
```

```
            <h4>Total amount: ₹{{totalAmt}}</h4>
            <p>(inclusive of 18% GST)</p>
        </div>

        <div class="mx-auto text-center" style="max-width: 500px; margin-top: 20px; margin-bottom: 50px;" *ngIf="addressNotFound == true">
            <div class="alert alert-danger" role="alert">
                You haven't set your address. Please do it otherwise we cannot confirm your order.
            </div>

            <button class="btn btn-md btn-outline-primary" (click)="goToProfile()">Go to profile</button>
        </div>

        <div class="alert alert-danger mx-auto" role="alert" style="max-width: 500px; margin-bottom: 20px;" *ngIf="notAvailableItems.length > 0">
            <h5>Following items are currently not available:</h5>
            <ul>
                <li *ngFor="let item of notAvailableItems">{{item.name}}</li>
            </ul>
        </div>

        <div class="text-center cart-btn-group" *ngIf="isProcessing == false && isOrdered == false">
            <div class="form-group">
                <button class="btn btn-outline-warning" (click)="goBackToCart()" style="margin-right: 15px">Back to cart</button>
                <button class="btn btn-outline-danger" (click)="confirm()" *ngIf="notAvailableItems.length < 1">Confirm</button>
            </div>
        </div>

        <div class="alert alert-success mx-auto" role="alert" style="max-width: 500px;" *ngIf="isOrdered == true">
            Your order has been placed successfully!
        </div>

        <div class="text-center cart-btn-group" *ngIf="isOrdered == true">
            <div class="form-group">
                <button class="btn btn-outline-primary" [routerLink]="['/orders']">My orders</button>
            </div>
        </div>
```

```
          </div>


        </div>
    </div>
</main>
```

ConfirmOrder.component.ts

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Cart } from '../models/cart.model';
import { HandleCartService } from '../services/handle-cart.service';
import { HandleLocalStorageService } from '../services/handle-local-storage.service';
import { ItemDataService } from '../services/item-data.service';
import { OrderDataService } from '../services/order-data.service';
import { UserDataService } from '../services/user-data.service';

@Component({
  selector: 'app-confirm-order',
  templateUrl: './confirm-order.component.html',
  styleUrls: ['./confirm-order.component.css'],
})
export class ConfirmOrderComponent implements OnInit, OnDestroy {
  cartObj: Cart;
  orderArray: any[] = [];
  totalAmt: string;
  isOrdered: boolean = false;
  isProcessing: boolean = false;
  addressNotFound: boolean = false;
  notAvailableItems: any[] = [];
  itemAvailabilityChecked: boolean = false;

  constructor(
    private handleCartService: HandleCartService,
    private handleLocalStorageService: HandleLocalStorageService,
    private router: Router,
    private orderDataService: OrderDataService,
    private userDataService: UserDataService,
    private itemDataService: ItemDataService
  ) {
    this.cartObj = JSON.parse(this.handleLocalStorageService.getCartData());
  }
```

```
ngOnInit(): void {
  this.populateOrderData();
  this.handleCartService.hideCartBar(true);

  this.userDataService.checkAddressPresentOrNot().then((data: string) => {
    if (data == null || data == undefined || data.trim().length < 1) {
      this.addressNotFound = true;
    }
  });
}

ngOnDestroy() {
  this.handleCartService.hideCartBar(false);
}

populateOrderData() {
  if (this.cartObj != null && this.cartObj.items != undefined) {
    const itemD = this.cartObj.items;

    for (let item in itemD) {
      const itemObj = itemD[item];

      const obj = {
        id: itemObj.itemId,
        category: itemObj.category,
        name: itemObj.name,
        price: itemObj.price,
        quantity: itemObj.quantity,
      };

      this.orderArray.push(obj);
    }
  }

  this.calculateTotalAmount();
}

getItemTotalAmount(price: number, quantity: number) {
  return Number(price) * Number(quantity);
}

calculateTotalAmount() {
  const GST_Amt = (18 / 100) * Number(this.cartObj.totalAmt);
  this.totalAmt = (Number(this.cartObj.totalAmt) + GST_Amt).toFixed(2);
```

```
}

goBackToCart() {
  this.router.navigate(['cart']);
}

confirm() {
  // don't allow to confirm order if address is not present
  if (this.addressNotFound === true) {
    return;
  }

  this.onConfirm();
}

async onConfirm() {
  this.isProcessing = true;
  this.notAvailableItems = await this.itemDataService.reportItemAvailability(
    this.orderArray
  );

  // if there are not available items in order
  if (this.notAvailableItems.length > 0) {
    this.isProcessing = false;
  } else {
    // clear cart
    this.cartObj = null;
    this.handleLocalStorageService.removeCartData();

    this.orderDataService
      .addOrderData(this.orderArray, this.totalAmt)
      .subscribe(
        (res: any) => {
          this.isOrdered = true;
          this.orderDataService.setOrderId(res.name);
        },
        (error) => {
          console.log(error);
        }
      );
  }
}

goToProfile() {
```

```
    let _name = this.makeProfilePath(
      this.handleLocalStorageService.getUserName()
    );
    this.router.navigate(['profile', _name]);
  }

  /** utilities */

  /** make profile path from name of the user */
  makeProfilePath(v: string) {
    return v.split(' ').join('-');
  }
}
```

ConfirmPage.component.css

```
.confirm-order-page {
    min-height: 120vh;
    padding-top: 80px;
}

div {
    font-family: "Karla", sans-serif;
}

.card-title {
    font-size: 20px;
}

.item-price {
    font-size: 20px;
}
```

Categories.component.html

```
<div class="categories-outer">

    <div class="container" style="text-align: center;">

        <div class="col-lg-6 mx-auto" id="text-1">
            <p class="display-6">
                Want to get food delivered to you? <br> Place your order now.
            </p>
        </div>
```

```
    <div class="text-center" style="margin-bottom: 60px; margin-top: 60px;">
      <button type="button" class="btn btn-md btn-outline-dark"
(click)="onViewCategory()">Have a look at what we've for you</button>
    </div>

    <div class="row row-cols-1 row-cols-md-2 row-cols-lg-2 row-cols-xl-4">
      <div class="col">
        <a>
          <div class="card">
            <img src="https://firebasestorage.googleapis.com/v0/b/restaurant-21-
club.appspot.com/o/starter.jpg?alt=media&token=2b0a42ab-d931-4421-9c5c-684a2ecb95e1"
class="card-img-top" alt="...">
              <div class="card-body">
                <h5 class="card-title">Starters</h5>
              </div>
          </div>
        </a>
      </div>
      <div class="col">
        <a>
          <div class="card">
            <img src="https://firebasestorage.googleapis.com/v0/b/restaurant-21-
club.appspot.com/o/mains.jpg?alt=media&token=bbf302db-447f-4c7e-9467-3741af3010d9"
class="card-img-top" alt="...">
              <div class="card-body">
                <h5 class="card-title">Mains</h5>
              </div>
          </div>
        </a>
      </div>
      <div class="col">
        <a>
          <div class="card">
            <img src="https://firebasestorage.googleapis.com/v0/b/restaurant-21-
club.appspot.com/o/dessert.jpg?alt=media&token=e8edc62c-da39-4f56-ae8f-94c372a1f3c2"
class="card-img-top" alt="...">
              <div class="card-body">
                <h5 class="card-title">Desserts</h5>
              </div>
          </div>
        </a>
      </div>
      <div class="col">
```

```
            <a>
              <div class="card">
                <img src="https://firebasestorage.googleapis.com/v0/b/restaurant-21-
club.appspot.com/o/alcohol.jpg?alt=media&token=40c06c97-283f-46b6-a843-36d4d93154c6"
class="card-img-top" alt="...">
                <div class="card-body">
                  <h5 class="card-title">Alcoholic beverages</h5>
                </div>
              </div>
            </a>
          </div>
        </div>
    </div>
</div>
```

Categories.component.ts

```
import { Component, OnInit } from '@angular/core';
import { NavigationExtras, Router } from '@angular/router';

@Component({
  selector: 'app-categories',
  templateUrl: './categories.component.html',
  styleUrls: ['./categories.component.css']
})
export class CategoriesComponent implements OnInit {

  constructor(private router: Router) { }

  ngOnInit(): void {
  }

  onViewCategory(){
    this.router.navigate(['menu-page']);
  }
}
```

Categories.component.css

```
.categories-outer {
    max-width: 100%;
    background-color: rgb(250, 250, 250);
    padding-top: 52px;
    padding-bottom: 52px;
```

```
}

#text-1 {
    margin-bottom: 30px;
}

.lead {
    font-weight: 400;
    color: rgb(41, 41, 41);
    font-size: 30px;
}

a.clickable-card,
a.clickable-card:hover {
    color: inherit;
    text-decoration: none;
}

.card {
    border-top-right-radius: 2em;
    border-bottom-left-radius: 2em;
}

.card-img-top {
    border-radius: 2em;
    border-bottom-right-radius: 0em;
    border-top-left-radius: 0em;
    border-bottom-left-radius: 0em;
}


/* div.card:hover {
    box-shadow: rgba(50, 50, 93, 0.25) 0px 13px 27px -5px, rgba(0, 0, 0, 0.3) 0px 8px 16px -8px;
} */


/*
  ##Device = Desktops
  ##Screen = 1281px to higher resolution desktops
*/

@media (min-width: 1281px) {
    .card-img-top {
        width: 100%;
```

```
        height: 15vw;
        object-fit: cover;
    }
}


/*
    ##Device = Laptops, Desktops
    ##Screen = B/w 1025px to 1280px
  */

@media (min-width: 1025px) and (max-width: 1280px) {
    div.card {
        margin: 0 auto;
        float: none;
        margin-bottom: 40px;
        width: 90%
    }
    .card-img-top {
        width: 100%;
        height: 15vw;
        object-fit: cover;
    }
}


/*
    ##Device = Tablets, Ipads (portrait)
    ##Screen = B/w 768px to 1024px
  */

@media (min-width: 768px) and (max-width: 1024px) {
    div.card {
        margin: 0 auto;
        float: none;
        margin-bottom: 40px;
        width: 90%
    }
    .card-img-top {
        width: 100%;
        height: 25vw;
        object-fit: cover;
    }
}
```

```
/*
   ##Device = Tablets, Ipads (landscape)
   ##Screen = B/w 768px to 1024px
 */

@media (min-width: 768px) and (max-width: 1024px) and (orientation: landscape) {
   div.card {
      width: 80%;
      margin: 0 auto;
      float: none;
      margin-bottom: 40px;
   }
   div.card-img-top {
      height: 25vw;
      object-fit: cover;
   }
}


/*
   ##Device = Low Resolution Tablets, Mobiles (Landscape)
   ##Screen = B/w 481px to 767px
 */

@media (min-width: 481px) and (max-width: 767px) {
   div.card {
      width: 80%;
      margin: 0 auto;
      float: none;
      margin-bottom: 40px;
   }
   div.card-img-top {
      height: 25vw;
      object-fit: cover;
   }
   .lead {
      font-size: 22px;
   }
}


/*
```

```
   ##Device = Most of the Smartphones Mobiles (Portrait)
   ##Screen = B/w 320px to 479px
  */

@media (min-width: 320px) and (max-width: 480px) {
   div.card {
      width: 80%;
      margin: 0 auto;
      float: none;
      margin-bottom: 40px;
   }
   div.card-img-top {
      height: 25vw;
      object-fit: cover;
   }
   .lead {
      font-size: 22px;
      margin-left: 20px;
      margin-right: 20px;
   }
}
```

Home.component.html

```
<div class="pt-5 my-5 px-xs-0 px-sm-0 pt-xs-0 pt-sm-0 text-center">
   <div class="overflow-hidden">
      <div class="container">
         <img src="https://firebasestorage.googleapis.com/v0/b/restaurant-21-
club.appspot.com/o/interior2.jpg?alt=media&token=742de6e1-bbe5-45ee-ad1a-4fc074e5795e"
style="width: 100%; margin-top: 20px;" class="border mb-4" alt="Example image"
loading="lazy">
      </div>
   </div>
</div>

<div class="px-4 py-5 my-5 text-center">
   <div class="col-lg-6 mx-auto">
      <p class="lead mb-4">
         Whether you are planning an intimate dinner for friends, a corporate luncheon, or an
extravagant soirée, you will find the perfect setting at kitchen Club. In addition to
accommodating as many as 400 guests, our restaurant feature award-winning off-site catering
         and a full service event production team that will make sure every detail is in place, so
you don’t have to.
      </p>
```

```
    </div>
</div>

<app-categories></app-categories>

<div style="margin-top: 25px;">
    <app-header-image></app-header-image>
</div>

<div class="px-4 py-5 my-5 text-center">
    <div class="col-lg-6 mx-auto">
        <p class="lead mb-4">
            With specialty cocktails, unique starter dishes, and spectacular main courses and desserts, kitchen Club is an amazing eatery. The superb talent and exacting precision of our chefs and staffs will make your experience here a grandeur one.
        </p>
    </div>
</div>
```

Home.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Home.component.css

```
.lead {
    color: black;
    font-weight: 400;
}
```

```
/*
  ##Device = Desktops
  ##Screen = 1281px to higher resolution desktops
*/

@media (min-width: 1281px) {
    .lead {
        font-size: 25px;
    }
}


/*
    ##Device = Laptops, Desktops
    ##Screen = B/w 1025px to 1280px
  */

@media (min-width: 1025px) and (max-width: 1280px) {
    .lead {
        font-size: 25px;
    }
}


/*
    ##Device = Tablets, Ipads (portrait)
    ##Screen = B/w 768px to 1024px
  */

@media (min-width: 768px) and (max-width: 1024px) {}


/*
    ##Device = Tablets, Ipads (landscape)
    ##Screen = B/w 768px to 1024px
  */

@media (min-width: 768px) and (max-width: 1024px) and (orientation: landscape) {}


/*
    ##Device = Low Resolution Tablets, Mobiles (Landscape)
    ##Screen = B/w 481px to 767px
```

```
  */

@media (min-width: 481px) and (max-width: 767px) {
    .lead {
        font-size: 18px;
    }
}


/*
    ##Device = Most of the Smartphones Mobiles (Portrait)
    ##Screen = B/w 320px to 479px
  */

@media (min-width: 320px) and (max-width: 480px) {
    .lead {
        font-size: 18px;
    }
}
```

Cart.model

```
import { ItemDetails } from "./item-details.model";

export interface Cart {
    items: ItemDetails;
    totalAmt: number;
}
```

Item.model

```
export interface Item {
    id: string;
    name: string;
    description?: string;
    price: number;
    category: string;
    imageUrl: string;
    addedOn: string;
    modifiedOn: string;
    isAvailable: boolean;
}
```

Item-details.model

```
export interface ItemDetails {
  [id: string]: {
    addedOn: string;
    quantity: number;
    itemId: string;
    category: string;
    name: string;
    price: string;
    imageUrl: string
  };
}
```

Order.model

```
import { OrderDetails } from "./order-details.model";

export interface Order {
    orderId: string,
    orderedItems: OrderDetails,
    addedOn: string,
    totalAmt: string
}
```

Oder-details.model

```
export interface OrderDetails {
  [id: string]: {
    itemId: string;
    name: string;
    price: number;
    quantity: number;
  };
}
```

User.model

```
export interface User {
  uid: string;
  email: string;
  name: string;
  phone: string;
  address: string;
  role: {
    val: string;
  }
}
```

Orderpage.component.html

```
<main class="my-order-page">
    <div class="album py-5">
        <div class="container">

            <div class="d-flex align-items-center pb-3 mb-5 border-bottom">
                <span class="fs-2">My orders</span>
            </div>

            <div *ngIf="isLoading == true">
                <app-loader></app-loader>
            </div>

            <div class="text-center" *ngIf="isLoaded == true && orderArray.length == 0">
                <h3>You haven't ordered anything yet.</h3>
            </div>

            <div *ngFor="let order of orderArray">
                <div class="card mx-auto" style="max-width: 700px;">
                    <div class="card-header">
                        <strong>Order # {{order.orderNo}}</strong> placed on {{order.addedOn}}
                    </div>
                    <div class="card-body">
                        <h5 class="card-title">ID: {{order.orderId}}</h5>
                        <div *ngFor="let oi of order.orderedItems">
                            <p class="card-text">
                                {{oi.name}}: ₹{{oi.price}} × {{oi.quantity}} = ₹{{getItemTotalAmount(oi.price,
oi.quantity)}}
```

```
                    </p>
                  </div>
                  <hr> <strong>Total amount paid: ₹{{order.totalAmt}}</strong> (inclusive of 18
%GST)
                </div>
              </div>
            </div>
          </div>
    </div>
</main>
```

orderPage.component.ts

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { count } from 'rxjs/operators';
import { Order } from '../models/order.model';
import { OrderDataService } from '../services/order-data.service';

@Component({
  selector: 'app-order-page',
  templateUrl: './order-page.component.html',
  styleUrls: ['./order-page.component.css'],
})
export class OrderPageComponent implements OnInit {
  orders: any;
  orderArray: any = [];
  isLoading: boolean = true;
  isLoaded: boolean = false;

  constructor(private orderDataService: OrderDataService) {
    this.fetchOrderData();
  }

  ngOnInit(): void {}

  async fetchOrderData() {
    this.isLoaded = false;
    this.isLoading = true;

    this.orders = await this.orderDataService.getOrderData();

    let count = 0;
    for (let orderId in this.orders) {
      count++;
```

```
      const orderObj: Order = this.orders[orderId];
      const oia = [];

      for (let oi in orderObj.orderedItems) {
        const o = {
          name: orderObj.orderedItems[oi].name,
          price: orderObj.orderedItems[oi].price,
          quantity: orderObj.orderedItems[oi].quantity,
        };
        oia.push(o);
      }

      const obj: any = {
        orderNo: count,
        orderId: orderObj.orderId,
        addedOn: orderObj.addedOn,
        orderedItems: oia,
        totalAmt: orderObj.totalAmt,
      };

      this.orderArray.push(obj);
    }

    // reverse it to show latest order first
    this.orderArray.reverse();

    this.isLoaded = true;
    this.isLoading = false;
  }

  getItemTotalAmount(price: number, quantity: number) {
    return Number(price) * Number(quantity);
  }
}
```

Order-page.component.css

```
.my-order-page {
    min-height: 120vh;
    padding-top: 80px;
}

div {
```

```
    font-family: "Karla", sans-serif;
}

.card {
    margin-bottom: 20px;
    font-size: 18px;
}

.card-header {
    background-color: rgb(0 255 167 / 46%);
}
```

Admin-auth-guard.service.spec

```
import { TestBed } from '@angular/core/testing';

import { AdminAuthGuardService } from './admin-auth-guard.service';

describe('AdminAuthGuardService', () => {
  let service: AdminAuthGuardService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(AdminAuthGuardService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

Admin-auth-guard.service

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { HandleLocalStorageService } from '../services/handle-local-storage.service';

@Injectable({
  providedIn: 'root',
})
export class AdminAuthGuard implements CanActivate {
  constructor(
    private handleLocalStorage: HandleLocalStorageService,
    private router: Router
```

```
  ) {}

  canActivate(): boolean {
    if (
      this.handleLocalStorage.getIsAdmin() == 'false' ||
      this.handleLocalStorage.getIsAdmin() == null
    ) {
      this.router.navigate(['not-found']);
      return false;
    }

    return true;
  }
}
```

auth-guard.service.spec

```
import { TestBed } from '@angular/core/testing';

import { AuthGuardService } from './auth-guard.service';

describe('AuthGuardService', () => {
  let service: AuthGuardService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(AuthGuardService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

auth-guard.service

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { HandleLocalStorageService } from '../services/handle-local-storage.service';

@Injectable({
  providedIn: 'root',
})
export class AuthGuard implements CanActivate{
```

```
  constructor(
    private handleLocalStorage: HandleLocalStorageService,
    private router: Router
  ) {}

  canActivate(): boolean {
    if (
      this.handleLocalStorage.getIsAuthenticated() === 'false' ||
      this.handleLocalStorage.getIsAuthenticated() == null
    ) {
      this.router.navigate(['login']);
      return false;
    }

    return true;
  }
}
```

```
import { TestBed } from '@angular/core/testing';

import { CustomerAuthGuardService } from './customer-auth-guard.service';

describe('CustomerAuthGuardService', () => {
  let service: CustomerAuthGuardService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(CustomerAuthGuardService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

customer-suth-guard.service.spec

```
import { TestBed } from '@angular/core/testing';

import { CustomerAuthGuardService } from './customer-auth-guard.service';

describe('CustomerAuthGuardService', () => {
  let service: CustomerAuthGuardService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(CustomerAuthGuardService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

Customer-suth-guard.service

```
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { HandleLocalStorageService } from '../services/handle-local-storage.service';

@Injectable({
  providedIn: 'root',
})
export class CustomerAuthGuard {
  constructor(
    private handleLocalStorage: HandleLocalStorageService,
    private router: Router
  ) {}

  canActivate(): boolean {
    if (
      this.handleLocalStorage.getIsAdmin() == 'true' ||
      this.handleLocalStorage.getIsAdmin() == null
    ) {
      this.router.navigate(['not-found']);
      return false;
    }

    return true;
  }
}
```

User.data.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { AngularFireDatabase, AngularFireObject } from '@angular/fire/database';
import { BehaviorSubject, Subject } from 'rxjs';
import { environment } from 'src/environments/environment';
import { User } from '../models/user.model';
import { HandleLocalStorageService } from './handle-local-storage.service';

@Injectable({
  providedIn: 'root',
})
export class UserDataService {
  private userObj: AngularFireObject<any>;
  private userData: User = {
```

```
  uid: null,
  email: null,
  name: null,
  phone: null,
  address: null,
  role: {
    val: 'customer',
  },
};
userDataSub = new BehaviorSubject<User>(this.userData);
isAdminSub = new BehaviorSubject<any>(null);

constructor(
  private afdb: AngularFireDatabase,
  private http: HttpClient,
  private handleLocalStorageService: HandleLocalStorageService
) {}

/** saves new user data in Firebase DB */
createNewUser(name: string, email: string, uid: string) {
  this.userData.name = name;
  this.userData.email = email;
  this.userData.uid = uid;

  this.userObj = this.afdb.object('users/' + this.userData.uid);
  this.userObj.set(this.userData).then(() => {
    this.getUserDataFromFirebase();
  });

  this.userDataSub.next(this.userData);
}

getUserDataObservable() {
  return this.userDataSub.asObservable();
}

// --- replace this with an async method
getUserDataFromFirebase() {
  if (this.handleLocalStorageService.getUser() != null) {
    //console.log('getting user data from firebase');
    this.http
      .get(
        environment.firebase.databaseURL +
          '/users/' +
```

```
          localStorage.getItem('user') +
          '.json'
      )
      .subscribe((data: User) => {
        this.userData = data;
        this.userDataSub.next(this.userData);
        this.handleLocalStorageService.setUserName(this.userData.name);

        // set isAdmin value
        if (data.role.val == 'admin') {
          this.handleLocalStorageService.setIsAdmin('true');
          this.isAdminSub.next(true);
        } else if (data.role.val == 'customer') {
          this.isAdminSub.next(false);
          this.handleLocalStorageService.setIsAdmin('false');
        }
      });
  }
}

getIsAdminObservable() {
  this.isAdminSub.next(this.userData.role.val == 'true' ? true : false);
  return this.isAdminSub.asObservable();
}

updateUserData(userDataParam: User): Promise<void> {
  this.handleLocalStorageService.setUserName(userDataParam.name);
  this.userDataSub.next(userDataParam);

  this.userObj = this.afdb.object('users/' + userDataParam.uid);
  return this.userObj.update(userDataParam);
}

async checkAddressPresentOrNot() {
  if (this.handleLocalStorageService.getUser() != null) {
    return await this.getAddressFromFirebase();
  }
}

async getAddressFromFirebase() {
  return await this.http
    .get(
      environment.firebase.databaseURL +
        '/users/' +
```

```
        localStorage.getItem('user') +
        '/address' +
        '.json'
    )
    .toPromise();
}

async getAllUsersData() {
  const path = environment.firebase.databaseURL + '/users.json';
  return await this.http.get(path).toPromise();
}

clearUserDataLocally() {
  Object.entries(this.userData).forEach(([key, val]) => {
    if (key === 'role') {
      this.userData.role.val = 'customer';
    } else {
      this.userData[key] = null;
    }
  });
}

public set setUid(v: string) {
  this.userData.uid = v;
}

public get getUid() {
  return this.userData.uid;
}

public set setEmail(v: string) {
  this.userData.email = v;
}

public set setName(v: string) {
  this.userData.name = v;
}

public get getName() {
  return this.userData.name;
}

public set setPhone(v: string) {
  this.userData.phone = v;
```

```
  }

  public set setAddress(v: string) {
    this.userData.address = v;
  }
}
```

Orderdata.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { AngularFireDatabase } from '@angular/fire/database';
import { environment } from 'src/environments/environment';
import { OrderDetails } from '../models/order-details.model';
import { Order } from '../models/order.model';
import { HandleLocalStorageService } from './handle-local-storage.service';

@Injectable({
  providedIn: 'root',
})
export class OrderDataService {
  uid: string;

  constructor(
    private http: HttpClient,
    private handleLocalStorageService: HandleLocalStorageService,
    private afdb: AngularFireDatabase
  ) {
    this.uid = this.handleLocalStorageService.getUser();
  }

  // adds item data to Firebase DB
  addOrderData(orderData: any, totalAmt: string) {
    const orderObj: Order = this.formatOrderData(orderData, totalAmt);

    const path =
      environment.firebase.databaseURL + '/orders/' + this.uid + '.json';

    return this.http.post<Order>(path, orderObj);
  }

  formatOrderData(od: any, amt: string): Order {
    let orderDetailsObj = {};
```

```
  for (let key in od) {
    const _obj: OrderDetails = {
      [od[key].id]: {
        itemId: od[key].id,
        name: od[key].name,
        price: od[key].price,
        quantity: od[key].quantity,
      },
    };

    orderDetailsObj = {
      ...orderDetailsObj,
      [od[key].id]: _obj[od[key].id],
    };
  }

  const orderObj: Order = {
    orderId: '',
    orderedItems: orderDetailsObj,
    addedOn: new Date().toLocaleString(),
    totalAmt: amt
  };

  return orderObj;
}

setOrderId(idParam: string) {
  let orderId = 'order' + idParam;

  const orderRef = this.afdb.object('orders/' + this.uid + '/' + idParam);
  orderRef.update({ orderId: orderId });
}

async getOrderData() {
  const uid = this.handleLocalStorageService.getUser();
  const path = environment.firebase.databaseURL + '/orders/' + uid + '.json';

  return await this.http.get(path).toPromise();
}

async getOrderDataById(uid: string) {
  const path = environment.firebase.databaseURL + '/orders/' + uid + '.json';
```

```
    return await this.http.get(path).toPromise();
  }
}
```

Auth.service.ts

```
import { Injectable } from '@angular/core';
import { AngularFireAuth } from '@angular/fire/auth';
import { AngularFirestore } from '@angular/fire/firestore';
import auth from 'firebase/app';
import { Router } from '@angular/router';
import { BehaviorSubject, Subject } from 'rxjs';
import { User } from '../models/user.model';
import { UserDataService } from './user-data.service';
import { HandleLocalStorageService } from './handle-local-storage.service';

/**
 * This service deals with user authentication.
 */

@Injectable({
  providedIn: 'root',
})
export class AuthService {
  private isAuthenticated: boolean = false;
  isAuthSub = new BehaviorSubject<any>(null);
  private authStateData: any = null;
  authStateSubject = new Subject<any>();

  constructor(
    private afs: AngularFirestore,
    private afAuth: AngularFireAuth,
    private router: Router,
    private userDataService: UserDataService,
    private handleLocalStorageService: HandleLocalStorageService
  ) {
    // subscribing to the observable that authState returns
    // so that we get updated whenever the authState data gets manipulated
    this.afAuth.authState.subscribe((user) => {
      if (user) {
        handleLocalStorageService.setUser(user.uid);
        this.userDataService.getUserDataFromFirebase();
        // sync cart data
        this.setIsAuthenticated(true);
```

```
      handleLocalStorageService.setIsAuthenticated('true');
      this.setAuthState(user);
    } else {
      this.setIsAuthenticated(false);
      handleLocalStorageService.clearDataOnLogOut();
    }
  });
}

// sign in with email and password
signIn(email: string, password: string): Promise<any> {
  return this.afAuth.signInWithEmailAndPassword(email, password);
}

// sign up with email and password
signUp(email: string, password: string): Promise<any> {
  return this.afAuth.createUserWithEmailAndPassword(email, password);
}

authenticateWithGoogle(): Promise<any> {
  return this.afAuth.signInWithPopup(new auth.auth.GoogleAuthProvider());
}

autoLogIn() {
  if (localStorage.getItem('user') != null) {
    this.setIsAuthenticated(true);
    this.handleLocalStorageService.setIsAuthenticated('true');
  }
}

logOut() {
  this.afAuth.signOut().then(() => {
    this.handleLocalStorageService.clearDataOnLogOut();
    this.setIsAuthenticated(false);
    this.setAuthState(null);
    this.router.navigate(['']);
  });
}

getIsAuthObservable() {
  this.isAuthSub.next(this.isAuthenticated);
  return this.isAuthSub.asObservable();
}
```

```
  setIsAuthenticated(v: boolean) {
    this.isAuthenticated = v;
    this.isAuthSub.next(this.isAuthenticated);
  }

  //
  getAuthStateObservable() {
    return this.authStateSubject.asObservable();
  }

  setAuthState(data: any) {
    this.authStateData = data;
    this.authStateSubject.next(this.authStateData);
  }
  //
}
```

Auth-erro-handler.service.ts

```
import { Injectable } from '@angular/core';
import { Subject } from 'rxjs';

@Injectable({
  providedIn: 'root',
})
export class AuthErrorHandlerService {
  private errorSubject = new Subject<any>();

  private errorObj = {
    logIn: {
      errorFound: null,
      email: null,
      password: null,
      unknown: null,
    },
    signUp: {
      errorFound: null,
      name: null,
      email: null,
      password: null,
      unknown: null,
    },
  };
```

```
constructor() {}

getErrorObservable() {
  return this.errorSubject.asObservable();
}

initializeErrorObj() {
  this.errorSubject.next(this.errorObj);
}

foundLogInError() {
  return !(this.errorObj.logIn.errorFound === null);
}

foundSignUpError() {
  return !(this.errorObj.signUp.errorFound === null);
}

handleAuthError(errorParam: any, callerParam: string) {
  if (callerParam === 'logIn') {
    this.errorObj.logIn.errorFound = true;

    switch (errorParam.code) {
      case 'auth/user-not-found':
        this.errorObj.logIn.email = 'Email not registered.';
        break;

      case 'auth/wrong-password':
        this.errorObj.logIn.password = "Password doesn't match.";
        break;

      default:
        this.errorObj.logIn.unknown = errorParam.message;
        break;
    }
  } else if (callerParam === 'signUp') {
    switch (errorParam.code) {
      case 'auth/email-already-in-use':
        this.errorObj.signUp.email = 'Email already in use.';
        break;

      case 'auth/invalid-email':
        this.errorObj.signUp.email = 'Please enter a valid email.';
```

```
          break;

        default:
          this.errorObj.signUp.unknown = errorParam.message;
          break;
      }
    }

    this.errorSubject.next(this.errorObj);
  }

  // set all error logIn fields to null
  clearLogInError() {
    Object.entries(this.errorObj.logIn).forEach(([key, val]) => {
      this.errorObj.logIn[key] = null;
    });

    this.errorSubject.next(this.errorObj);
  }

  // set all error signUp fields to null
  clearSignUpError() {
    Object.entries(this.errorObj.signUp).forEach(([key, val]) => {
      this.errorObj.signUp[key] = null;
    });

    this.errorSubject.next(this.errorObj);
  }
}
```

Fetch-header-image.service.ts

```
import { Injectable, OnInit } from '@angular/core';
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { environment } from 'src/environments/environment';

@Injectable({
  providedIn: 'root',
})
export class FetchHeaderImageService implements OnInit {

  constructor(private http: HttpClient) {}

  ngOnInit() {}
```

```
  fetchImage() {
    return this.http.get(environment.UNSPLASH_SOURCE_API);
  }
}
```

Herader-cart.service.ts

```
import { Injectable, OnInit } from '@angular/core';
import { Cart } from '../models/cart.model';
import { ItemDetails } from '../models/item-details.model';
import { HandleLocalStorageService } from './handle-local-storage.service';
import { AngularFireDatabase } from '@angular/fire/database';
import { ItemDataService } from './item-data.service';
import { HttpClient } from '@angular/common/http';
import { environment } from 'src/environments/environment';
import { map } from 'rxjs/operators';
import { BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root',
})
export class HandleCartService implements OnInit {
  cartObj: Cart;
  itemsArray: ItemDetails[] = [];
  itemDetails: ItemDetails;
  uid: string;
  postPath: string;
  getPath: string;

  onCartPageSub = new BehaviorSubject<boolean>(false);
  onConfirmOrderPageSub = new BehaviorSubject<boolean>(false);

  constructor(private handleLocalStorageService: HandleLocalStorageService) {}

  ngOnInit() {
    this.cartObj = this.getCartData();
  }

  /** add or update items in cart */
  addOrUpdate(item: any) {
    // get cart data from local storage
    this.cartObj = this.getCartData();
```

```
// add cart object for the first time
if (this.cartObj == null) {
  const cart: Cart = {
    items: {
      [item.id]: {
        addedOn: new Date().toLocaleString(),
        quantity: item.quantity,
        itemId: item.id,
        category: item.category,
        name: item.name,
        price: item.price,
        imageUrl: item.imageUrl,
      },
    },
    totalAmt: item.price,
  };

  this.handleLocalStorageService.addCartData(cart);
} else {
  // add a new item to cart
  if (this.cartObj.items[item.id] == undefined) {
    const itemD: ItemDetails = {
      [item.id]: {
        addedOn: new Date().toLocaleString(),
        quantity: item.quantity,
        itemId: item.id,
        category: item.category,
        name: item.name,
        price: item.price,
        imageUrl: item.imageUrl,
      },
    };

    // any better way?
    this.cartObj = {
      items: {
        ...this.cartObj.items,
        [item.id]: itemD[item.id],
      },
      totalAmt: this.getCartTotalAmount(item.price, true),
    };

    this.handleLocalStorageService.addCartData(this.cartObj);
  } else {
```

```
      // update quantity for existing item
      const itemD = this.cartObj.items[item.id];
      itemD.quantity += 1;
      this.cartObj.items[item.id] = itemD;

      // update total amount
      this.cartObj.totalAmt = this.getCartTotalAmount(item.price, true);

      this.handleLocalStorageService.addCartData(this.cartObj);
    }
  }
}

/** remove an item from cart */
removeItem(item: any) {
  this.cartObj = this.getCartData();

  if (this.cartObj != null) {
    const itemD = this.cartObj.items[item.id];

    if (itemD.quantity > 1) {
      // decrease the quantity
      itemD.quantity -= 1;
      this.cartObj.items[item.id] = itemD;
    } else if (itemD.quantity == 1) {
      // when quantity is 1
      // remove the item
      delete this.cartObj.items[item.id];
    }

    this.cartObj.totalAmt = this.getCartTotalAmount(item.price, false);
  }

  this.handleLocalStorageService.addCartData(this.cartObj);
}

/** calculate total cart amount */
getCartTotalAmount(price: number, add: boolean): number {
  let amt: number;

  if (add == true) {
    amt = Number(this.cartObj.totalAmt) + Number(price);
  } else {
    amt = Number(this.cartObj.totalAmt) - Number(price);
```

```
    }

    return amt;
  }

  /** check for cart data in local storage or Firebase */
  getCartData() {
    if (this.handleLocalStorageService.getCartData() != null) {
      return JSON.parse(this.handleLocalStorageService.getCartData());
    }

    return null;
  }

  /** clear cart */
  clearCart() {
    this.cartObj = null;
    this.handleLocalStorageService.removeCartData();
  }

  onCartPageObs() {
    this.onCartPageSub.next(false);
    return this.onCartPageSub.asObservable();
  }

  goToOrders(v: boolean) {
    this.onCartPageSub.next(v);
  }

  onConfirmOrderPageObs() {
    return this.onConfirmOrderPageSub.asObservable();
  }

  hideCartBar(v: boolean) {
    this.onConfirmOrderPageSub.next(v);
  }
}
```

Handle-local-storage.service.ts

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';
import { Cart } from '../models/cart.model';
```

```
@Injectable({
  providedIn: 'root',
})
export class HandleLocalStorageService {
  cartDataSub = new BehaviorSubject<Cart>(null);

  constructor() {}

  setUser(value: string) {
    localStorage.setItem('user', value);
  }

  getUser() {
    return localStorage.getItem('user');
  }

  setIsAuthenticated(value: string) {
    localStorage.setItem('isAuthenticated', value);
  }

  getIsAuthenticated(): string {
    return localStorage.getItem('isAuthenticated');
  }

  setIsAdmin(value: string) {
    localStorage.setItem('isAdmin', value);
  }

  getIsAdmin() {
    return localStorage.getItem('isAdmin');
  }

  setUserName(value: string) {
    localStorage.setItem('name', value);
  }

  getUserName() {
    return localStorage.getItem('name');
  }

  clearDataOnLogOut() {
    localStorage.removeItem('user');
    localStorage.removeItem('isAdmin');
    localStorage.removeItem('isAuthenticated');
```

```
    localStorage.removeItem('name');
  }

  // cart data
  addCartData(cart: Cart) {
    localStorage.setItem('cartData', JSON.stringify(cart));

    const obj = JSON.parse(localStorage.getItem('cartData'));

    // check if items in cart is empty
    if (Object.keys(obj.items).length == 0) {
      this.removeCartData();
    }

    this.cartDataSub.next(JSON.parse(this.getCartData()));
  }

  removeCartData() {
    if (localStorage.getItem('cartData') != null) {
      localStorage.removeItem('cartData');
    }
    this.cartDataSub.next(null);
  }

  getCartData() {
    return localStorage.getItem('cartData');
  }

  getCartDataObservable() {
    this.cartDataSub.next(JSON.parse(this.getCartData()));
    return this.cartDataSub.asObservable();
  }
}
```

Item-data.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { AngularFireDatabase } from '@angular/fire/database';
import { map } from 'rxjs/operators';
import { environment } from 'src/environments/environment';
import { Item } from '../models/item.model';
```

```
@Injectable({
  providedIn: 'root',
})
export class ItemDataService {
  item: Item;

  constructor(private http: HttpClient, private afdb: AngularFireDatabase) {}

  // adds item data to Firebase DB
  addItemData(itemData: Item) {
    this.item = itemData;
    const path =
      environment.firebase.databaseURL +
      '/items/' +
      itemData.category +
      '.json';

    return this.http.post<Item>(path, itemData);
  }

  // sets item id in Firebase DB
  setItemId(idParam: string) {
    let modifiedIdParam: string;

    if (this.item.category === 'starters') {
      modifiedIdParam = 'S' + idParam;
    } else if (this.item.category === 'mains') {
      modifiedIdParam = 'M' + idParam;
    } else if (this.item.category === 'alcoholic-beverages') {
      modifiedIdParam = 'AB' + idParam;
    } else if (this.item.category === 'desserts') {
      modifiedIdParam = 'D' + idParam;
    }

    const itemRef = this.afdb.object(
      'items/' + this.item.category + '/' + idParam
    );
    itemRef.update({ id: modifiedIdParam });
  }

  /** get item by category */
  async getItemsCategoryWise(category: string) {
    const path =
      environment.firebase.databaseURL + '/items/' + category + '.json';
```

```
  return await this.http
    .get(path)
    .pipe(
      map((responseData) => {
        const itemsArray: Item[] = [];

        for (const key in responseData) {
          if (responseData.hasOwnProperty(key)) {
            itemsArray.push(responseData[key]);
          }
        }
        return itemsArray;
      })
    )
    .toPromise();
}

/** get item by id */
async getItemById(category: string, id: string) {
  const pathItemId = this.getPathItemId(id);

  const path =
    environment.firebase.databaseURL +
    '/items/' +
    category +
    '/' +
    pathItemId +
    '.json';

  return await this.http
    .get(path)
    .pipe(
      map((data) => {
        return data;
      })
    )
    .toPromise();
}

/** delete item data from Firebase DB */
async deleteItemData(itemCategory: string, itemId: string) {
  const pathItemId = this.getPathItemId(itemId);
  const itemRef = this.afdb.object(
```

```
    'items/' + itemCategory + '/' + pathItemId
  );
  return await itemRef.remove();
}

/** delete imageUrl value when image deleted from storage */
async deleteImageUrl(itemCategory: string, itemId: string) {
  const pathItemId = this.getPathItemId(itemId);
  const itemRef = this.afdb.object(
    'items/' + itemCategory + '/' + pathItemId
  );
  return await itemRef.update({ imageUrl: '' });
}

/** update item data */
async updateItemData(item: Item, itemCategory: string, itemId: string) {
  const pathItemId = this.getPathItemId(itemId);
  const itemRef = this.afdb.object(
    'items/' + itemCategory + '/' + pathItemId
  );
  return await itemRef.update(item);
}

/** set/toggle item availability status */
async setIsAvailable(v: boolean, itemCategory: string, itemId: string) {
  const pathItemId = this.getPathItemId(itemId);
  const itemRef = this.afdb.object(
    'items/' + itemCategory + '/' + pathItemId
  );
  return await itemRef.update({ isAvailable: v });
}

/** checks item availability status before confirm order */
async reportItemAvailability(orders: any[]) {
  let notAvailableItems: any[] = [];

  for(let i in orders) {
    const obj: Item = await this.getItemById(orders[i].category, orders[i].id) as Item;

    if(obj.isAvailable == false){
      notAvailableItems.push({name: obj.name, id: obj.id});
    }
  }
```

```
  return notAvailableItems;

}

/** utilities */

getPathItemId(itemId: string): string {
  let pathItemId = '';

  if (!itemId.startsWith('-')) {
    let parts: string[] = itemId.split('-');
    for (let i = 1; i < parts.length; i++) {
      pathItemId += '-' + parts[i];
    }
  } else {
    pathItemId = itemId;
  }

  return pathItemId;
}

/** utilities end */

async getAllItems() {
  const path = environment.firebase.databaseURL + '/items.json';
  let res: any;

  return await this.http
    .get(path)
    .pipe(
      map((responseData) => {
        const itemsArray: Item[] = [];
        for (const category in responseData) {
          if (responseData.hasOwnProperty(category)) {
            for (const item in responseData[category]) {
              itemsArray.push(responseData[category][item]);
            }
          }
        }
        return itemsArray;
      })
    )
    .toPromise();
}
```

}

Item.data.service.ts

```
import { Injectable } from '@angular/core';
import { AngularFireDatabase } from '@angular/fire/database';
import { AngularFireStorage } from '@angular/fire/storage';
import { Observable, Subject } from 'rxjs';
import { finalize } from 'rxjs/operators';

@Injectable({
  providedIn: 'root',
})
export class ItemImageService {
  private basePath = '/uploads/images/';
  private imageUrl: string = '';
  private imageUrlSub = new Subject<any>();

  constructor(
    private afdb: AngularFireDatabase,
    private storage: AngularFireStorage
  ) {}

  // upload image file to Firebase storage
  pushImageToStorage(file: File, itemCategory: string): Observable<number> {
    const filePath = this.basePath + itemCategory + '/' + file.name;
    const storageRef = this.storage.ref(filePath);
    const uploadTask = this.storage.upload(filePath, file);

    uploadTask
      .snapshotChanges()
      .pipe(
        finalize(() => {
          storageRef.getDownloadURL().subscribe((downloadUrl) => {
            this.imageUrl = downloadUrl;
            this.imageUrlSub.next(this.imageUrl);
          });
        })
      )
      .subscribe((data: any) => {
        // console.log(data);
      });

    return uploadTask.percentageChanges();
```

```
  }

  async deleteImage(imageUrl: string) {
    return await this.storage.refFromURL(imageUrl).delete().toPromise();
  }

  getimageUrlObservable() {
    return this.imageUrlSub.asObservable();
  }
}
```

User-profile.component.html

```
<div class="container profile-div">
    <div class="row">
        <div class="col-xs-12 col-md-6 col-md-offset-3">
            <h2 style="margin-bottom: 40px;">Your profile</h2>

            <form [formGroup]="userProfileForm" (ngSubmit)="onUpdate()">
                <div class="form-group">
                    <label for="exampleInputEmail1">Name</label>
                    <input type="name" class="form-control" formControlName="name">
                    <span class="error-text" *ngIf="userProfileForm.controls.name.errors != null &&
userProfileForm.touched">Name cannot be blank.</span>
                </div>
                <div class="form-group">
                    <label for="exampleInputEmail1">Phone</label>
                    <input type="phone" class="form-control" formControlName="phone">
                    <span class="error-text" *ngIf="userProfileForm.controls.phone.errors != null">
                    Enter a 10 digit valid phone number.
                </span>
                </div>
                <div class="form-group">
                    <label for="exampleInputEmail1">Email</label>
                    <input type="email" class="form-control" formControlName="email" readonly>
                </div>
                <div class="form-group">
                    <label for="exampleInputEmail1">Address</label>
                    <input type="address" class="form-control" formControlName="address">
                </div>
                <div class="form-group">
                    <label for="exampleInputEmail1">Role</label>
                    <input type="role" class="form-control" formControlName="role" readonly>
                </div>
```

```
            <div class="alert alert-success mx-auto" role="alert" *ngIf="isUpdateSuccess ==
true">
                Profile updated!
            </div>

            <div class="alert alert-danger mx-auto" role="alert" *ngIf="isUpdateFailure == true">
                Some error occurred.
            </div>

            <button type="submit" [disabled]="!userProfileForm.valid" class="btn btn-md btn-
primary">
                Update
            </button>
        </form>
      </div>
   </div>
</div>
```

User-profile.component.ts

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { ActivatedRoute, Params, Router } from '@angular/router';
import { Subscription } from 'rxjs';
import { User } from '../models/user.model';
import { HandleLocalStorageService } from '../services/handle-local-storage.service';
import { UserDataService } from '../services/user-data.service';

@Component({
  selector: 'app-user-profile',
  templateUrl: './user-profile.component.html',
  styleUrls: ['./user-profile.component.css'],
})
export class UserProfileComponent implements OnInit, OnDestroy {
  userProfileForm: FormGroup;
  private userDataSub: Subscription;
  private userData: User;
  private pathVar: string = '';

  isUpdateSuccess: boolean = false;
  isUpdateFailure: boolean = false;

  constructor(
```

```
  private userDataService: UserDataService,
  private router: Router,
  private route: ActivatedRoute,
  private handleLocalStorageService: HandleLocalStorageService
) {
  // get the path variable
  this.pathVar = this.route.snapshot.params['name'];
}

ngOnInit(): void {
  // show 404 if profile path name doesn't match logged in user's username
  if (this.pathVar != '' && this.pathVar != undefined) {
    let _name = this.makeProfilePath(
      this.handleLocalStorageService.getUserName()
    );

    if (this.pathVar != _name) {
      this.router.navigate(['not-found']);
    }
  }

  this.userDataSub = this.userDataService
    .getUserDataObservable()
    .subscribe((data) => {
      this.userData = data;

      if (this.userData != undefined) {
        if (this.userData.name != undefined) {
          this.userProfileForm.patchValue({
            name: this.userData.name,
          });
        }
        if (this.userData.phone != undefined) {
          this.userProfileForm.patchValue({
            phone: this.userData.phone,
          });
        }
        if (this.userData.email != undefined) {
          this.userProfileForm.patchValue({
            email: this.userData.email,
          });
        }
        if (this.userData.address != undefined) {
          this.userProfileForm.patchValue({
```

```
          address: this.userData.address,
        });
      }
      /* if (this.userData.role != undefined && this.userData.role.val != undefined) {
      this.userProfileForm.patchValue({
        role: this.userData.role.val
      });
      */
    }
  });

  // creating reactive signup form
  this.userProfileForm = new FormGroup({
    name: new FormControl(this.userData.name, [Validators.required]),
    phone: new FormControl(this.userData.phone, [
      Validators.pattern('^((\\+91-?)|0)?[0-9]{10}$'),
    ]),
    email: new FormControl(this.userData.email),
    address: new FormControl(this.userData.address),
    role: new FormControl(this.userData.role.val),
  });
}

ngOnDestroy(): void {
  this.userDataSub.unsubscribe();
}

// updates user data
onUpdate() {
  // handle the case when disabled attribute for submit button is deleted
  // from html
  if (this.userProfileForm.invalid) {
    return;
  }

  this.userData.name = this.userProfileForm.get('name').value;
  this.userData.phone = this.userProfileForm.get('phone').value;
  this.userData.address = this.userProfileForm.get('address').value;
  this.userDataService.setUid = this.userData.uid;

  this.userDataService
    .updateUserData(this.userData)
    .then(() => {
      this.isUpdateSuccess = true;
```

```
        let _name: string;
        _name = this.makeProfilePath(this.userDataService.getName);
        this.router.navigate(['profile', _name]);
      })
      .catch(() => {
        this.isUpdateFailure = true;
      });

      setTimeout(() => {
        this.isUpdateFailure = this.isUpdateSuccess = false;
      }, 2000);
  }

  /** utility functions */

  replaceUndefinedOrNull(v: any): string {
    if (v == undefined || v == null) {
      return ' ';
    }
    return v;
  }

  /** make profile path from name of the user */
  makeProfilePath(v: string) {
    return v.split(' ').join('-');
  }
}
```

User-profile.component.css

```
.profile-div {
    padding-top: 100px;
    padding-bottom: 180px;
    min-height: 70vh;
}

.form-group {
    margin-bottom: 20px;
}

label {
    font-size: 20px;
    font-weight: 400;
```

```
}
```

App.component.html

```
<app-navbar></app-navbar>

<router-outlet></router-outlet>

<app-footer></app-footer>

<app-cart></app-cart>
```

App.component.ts

```
import { Component } from '@angular/core';

import { AuthService } from './services/auth.service';
import { UserDataService } from './services/user-data.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  constructor(
    private authService: AuthService,
    private userDataService: UserDataService
  ) {}

  ngOnInit() {
    // auto log in user if local storage has the uid returned by firebase
    this.authService.autoLogIn();
  }
}
```

App.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
```

```
import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';

import { NavbarComponent } from './global/navbar/navbar.component';
import { HeaderImageComponent } from './home/header-image/header-image.component';
import { CategoriesComponent } from './home/categories/categories.component';
import { HomeComponent } from './home/home.component';
import { LoaderComponent } from './global/loader/loader.component';
import { LoginComponent } from './auth/login/login.component';
import { SignupComponent } from './auth/signup/signup.component';
import { FooterComponent } from './global/footer/footer.component';
import { CartComponent } from './cart/cart.component';

import { AngularFireModule } from '@angular/fire';
import { AngularFireAuthModule } from '@angular/fire/auth';
import { AngularFirestoreModule } from '@angular/fire/firestore';
import { AngularFireStorageModule } from '@angular/fire/storage';
import { AngularFireAnalyticsModule } from '@angular/fire/analytics';

import { environment } from 'src/environments/environment';
import { CartPageComponent } from './cart-page/cart-page.component';
import { CategoryPageComponent } from './category-page/category-page.component';
import { UserProfileComponent } from './user-profile/user-profile.component';
import { AddOrEditItemsComponent } from './admin/add-or-edit-items/add-or-edit-
items.component';
import { DisplayItemsComponent } from './admin/display-items/display-items.component';
import { NotFoundComponent } from './global/not-found/not-found.component';
import { StartersIconComponent } from './global/starters-icon/starters-icon.component';
import { MainsIconComponent } from './global/mains-icon/mains-icon.component';
import { DrinksIconComponent } from './global/drinks-icon/drinks-icon.component';
import { DessertsIconComponent } from './global/desserts-icon/desserts-icon.component';
import { CartIconComponent } from './global/cart-icon/cart-icon.component';
import { OrderPageComponent } from './order-page/order-page.component';
import { AuthGuard } from './route-guards/auth-guard.service';
import { ConfirmOrderComponent } from './confirm-order/confirm-order.component';
import { ManageOrdersComponent } from './admin/manage-orders/manage-orders.component';
import { DisplayOrdersComponent } from './admin/display-orders/display-orders.component';

@NgModule({
  declarations: [
    AppComponent,
```

```
    NavbarComponent,
    HeaderImageComponent,
    CategoriesComponent,
    HomeComponent,
    LoaderComponent,
    LoginComponent,
    SignupComponent,
    FooterComponent,
    CartComponent,
    CartPageComponent,
    CategoryPageComponent,
    UserProfileComponent,
    AddOrEditItemsComponent,
    DisplayItemsComponent,
    NotFoundComponent,
    StartersIconComponent,
    MainsIconComponent,
    DrinksIconComponent,
    DessertsIconComponent,
    CartIconComponent,
    OrderPageComponent,
    ConfirmOrderComponent,
    ManageOrdersComponent,
    DisplayOrdersComponent
  ],
  imports: [
    AngularFireModule.initializeApp(environment.firebase),
    AngularFireAuthModule,
    AngularFirestoreModule,
    AngularFireStorageModule,
    AngularFireAnalyticsModule,
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule,
  ],
  providers: [AuthGuard],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

AppRouting.module

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes, ExtraOptions } from '@angular/router';
import { AppComponent } from './app.component';
import { HeaderImageComponent } from './home/header-image/header-image.component';
import { HomeComponent } from './home/home.component';
import { LoginComponent } from './auth/login/login.component';
import { NavbarComponent } from './global/navbar/navbar.component';
import { SignupComponent } from './auth/signup/signup.component';
import { UserProfileComponent } from './user-profile/user-profile.component';
import { DisplayItemsComponent } from './admin/display-items/display-items.component';
import { AddOrEditItemsComponent } from './admin/add-or-edit-items/add-or-edit-
items.component';
import { NotFoundComponent } from './global/not-found/not-found.component';
import { CategoryPageComponent } from './category-page/category-page.component';
import { CartPageComponent } from './cart-page/cart-page.component';
import { OrderPageComponent } from './order-page/order-page.component';
import { AuthGuard } from './route-guards/auth-guard.service';
import { AdminAuthGuard } from './route-guards/admin-auth-guard.service';
import { CustomerAuthGuard } from './route-guards/customer-auth-guard.service';
import { ConfirmOrderComponent } from './confirm-order/confirm-order.component';
import { ManageOrdersComponent } from './admin/manage-orders/manage-orders.component';
import { DisplayOrdersComponent } from './admin/display-orders/display-orders.component';

const routerOptions: ExtraOptions = {
  scrollPositionRestoration: 'enabled',
  anchorScrolling: 'enabled',
  scrollOffset: [0, 100]
};

const routes: Routes = [
  {
    path: '',
    component: HomeComponent,
  },
  {
    path: 'login',
    component: LoginComponent,
  },
  {
    path: 'sign-up',
    component: SignupComponent,
  },
```

```
{
  path: 'profile/:name',
  component: UserProfileComponent,
  canActivate: [AuthGuard]
},
{
  path: 'admin/items',
  component: DisplayItemsComponent,
  canActivate: [AdminAuthGuard]
},
{
  path: 'admin/items/add',
  component: AddOrEditItemsComponent,
  data: { path: 'add' },
  canActivate: [AdminAuthGuard]
},
{
  path: 'admin/items/edit/:itemCategory/:itemId',
  component: AddOrEditItemsComponent,
  data: { path: 'edit' },
  canActivate: [AdminAuthGuard]
},
{
  path: 'admin/manage-orders',
  component: ManageOrdersComponent,
  canActivate: [AdminAuthGuard]
},
{
  path: 'admin/:uid/orders',
  component: DisplayOrdersComponent,
  canActivate: [AdminAuthGuard]
},
{
  path: 'menu-page',
  component: CategoryPageComponent
},
{
  path: 'cart',
  component: CartPageComponent
},
{
  path: 'orders',
  component: OrderPageComponent,
  canActivate: [AuthGuard]
```

```
  },
  {
    path: 'confirm-order',
    component: ConfirmOrderComponent,
    canActivate: [AuthGuard]
  },
  {
    path: 'not-found',
    component: NotFoundComponent,
  },
  {
    path: '**',
    redirectTo: 'not-found',
  },
];

@NgModule({
  imports: [RouterModule.forRoot(routes, routerOptions)],
  exports: [RouterModule],
})
export class AppRoutingModule {}
```