```css
body{
    background-color: oldlace;
}
.account-container{
    margin-top:10%;
    color: orange;
    padding:0;
    background: white;
}
.top-section{
    font-size: 2.5em;
    background-color:#e1e9e7;
    padding: 5%;
    color: #031c63;

}
.bottom-section{
    padding-left: 5%;
    padding-right: 5%;
    padding-top:2%;
    padding-bottom:2%;
    color: #048ec4;
}
hr{
    padding: 0%;
    margin: 0%;
}
.transaction{
    margin-top: 22%;
    background-color: white;
    padding:0;

}
.transaction span{
    font-size: 0.7em;
}
```

```
<div class="alert alert-success" role="alert">
    Hello! {{userFirstname}}, Your financial picture is here.
  </div>

<div class="container">
    <div class="row">
        <div class="col-sm">
            <div class="account-container">

                <div class="top-section">
                    <div class="balance float-right">
                        Checking Balance: ${{checkingAccountBalance}}
                    </div>
                    <div class="account type">
                        Checking Account
                    </div>
                    <button type="button" class="btn btn-success btn-sm" id="viewCheckingAccountDetails"
(click)="viewCheckingAccountDetails()">Details</button>
                </div>

            </div>

            <br>

            <div class="account-container">

                <div class="top-section">
                    <div class="balance float-right">
                       Savings Balance:  ${{savingAccountBalance}}
                    </div>
                    <div class="account type">
                        Saving Account
                    </div>
                    <button type="button" class="btn btn-primary btn-sm" id="viewSavingAccountDetails"
(click)="viewSavingAccountDetails()">Details</button>
                </div>

            </div>

            <br>

            <div class="account-container">
                <div class="alert alert-secondary" role="alert">
                    Check Book Requests
                </div>
                <table class="table">
                    <thead class="thead-bright">
                        <tr>
                            <th scope="col">Account Number</th>
                            <th scope="col">Quantity</th>
                            <th scope="col">Status</th>
                        </tr>
                    </thead>
                    <tr *ngFor="let r of requests; let i = index">
                        <td>{{r.accountId}}</td>
                        <td>{{r.quantity}}</td>
                        <td>{{r.status}}</td>
                    </tr>
                </table>
            </div>

            <div class="account-container">
                <p>
                    <a class="btn btn-primary" data-toggle="collapse" href="#requestChequeForm" role="button" aria-
expanded="false" aria-controls="requestChequeForm">
                  Request Check Book
                    </a>
                  </p>
                <div class="collapse" id="requestChequeForm">
                    <div class="card card-body">
                  <form #requestChequeForm="ngForm" (ngSubmit)="requestChequeBook(requestChequeForm.value)"
clas="form-inline">
                        <div>
                            <p class="text-dark">Check Order Details:</p>
                         </div>
                         <div class="input-group w-auto justify-content-end align-items-center">
                            Quantity <input type="number" step="1" max="10" min="1" default="1" value="1" id="quantity"
name="quantity" class="quantity-field border-1 border-primary text-center w-25" ngModel>
                         </div>
                        <div class="form-group mx-sm-3 mb-2">
                            <div class="form-check form-check-inline">
                                <input class="form-check-input" type="radio" name="accountFor" id="accountFor"
value="checking" ngModel>
                                <label class="form-check-label" for="inlineRadio1">For Checking Account</label>
                              </div>
                              <div class="form-check form-check-inline">
                                <input class="form-check-input" type="radio" name="accountFor" id="accountFor" value="saving"
ngModel>
                                <label class="form-check-label" for="inlineRadio2">For Saving Account</label>
                              </div>
                        </div>
                        <button type="submit" class="btn btn-primary mb-2">Submit</button>
                  </form>
                    </div>
                </div>
            </div>
        </div>
    <br>

    </div>
</div>
```

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

import { AccountComponent } from './account.component';

describe('AccountComponent', () => {
  let component: AccountComponent;
  let fixture: ComponentFixture<AccountComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ AccountComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(AccountComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AccountsService } from 'src/app/services/accounts.service';
import { ChequeService } from 'src/app/services/cheque.service';


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

  constructor(private router:Router, private service:AccountsService, private chequeService:ChequeService) { }
  checkingAccountBalance;
  savingAccountBalance;
  userFirstname;
  userId;
  requests;

  ngOnInit(): void {

    this.userFirstname = sessionStorage.getItem("firstname");
    this.userId = sessionStorage.getItem("id");

    var checkingAccountId = parseInt(this.userId.toString() + "01");
    var savingAccountId = parseInt(this.userId.toString() + "02");

    this.savingAccountBalance = sessionStorage.getItem("savingAccountBalance")
    this.service.getAccountBalance(checkingAccountId).subscribe(
      response => {
        this.checkingAccountBalance = response;
        sessionStorage.setItem("checkingAccountId", checkingAccountId.toString());
      },
      error =>{
        console.log(error);
      }
    )
    this.service.getAccountBalance(savingAccountId).subscribe(
      response => {
        this.savingAccountBalance = response;
        sessionStorage.setItem("savingAccountId", savingAccountId.toString());
      },
      error =>{
        console.log(error);
      }
    )

    this.chequeService.getChequeBookRequestsByUserId(this.userId).subscribe(
      response => {
        this.requests = response;
      },
      error => {
        console.log(error);
      }
    )
  }

  public viewCheckingAccountDetails() {
    this.router.navigate(["checking"]);
  }

  public viewSavingAccountDetails() {
    this.router.navigate(["saving"]);
  }

  public requestChequeBook(data) {
    let quantity = data.quantity;
    let accountFor = data.accountFor;
    let toSaving = (data.accountFoor == "saving") ? true : false;
    let cheque = new RequestChequeBookBody(Number(this.userId), quantity, toSaving);
    this.chequeService.requestChequeBook(cheque).subscribe(
      response => {
        window.location.reload();
      },
      error => {
        console.log(error);
      }
    )
  }
}

class RequestChequeBookBody {
  userId : number;
  quantity : number;
  saving : boolean;

  constructor(userId : number, quantity : number, saving : boolean) {
    this.userId = userId;
    this.quantity = quantity;
    this.saving = saving;
  }
}
```

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

import { AccountsService } from './accounts.service';

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

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

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

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root',
})
export class AccountsService {
  private url: string = 'http://localhost:8080/accounts';
  constructor(private httpClient: HttpClient) {}

  //Get balance
  public getAccountBalance(accountId: number) {
    return this.httpClient.get(`${this.url}/balance/${accountId}`);
  }

  public deposit(depositBody: any) {
    return this.httpClient.post(`${this.url}/deposit`, depositBody);
  }

  public withdrawal(withdrawalBody: any) {
    return this.httpClient.post(`${this.url}/withdrawal`, withdrawalBody);
  }

  public transfer(transferBody: any) {
    return this.httpClient.post(`${this.url}/transfer`, transferBody);
  }
}
```

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

import { AdminService } from './admin.service';

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

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

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

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';

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

  private url: string = 'http://localhost:8080/admin';
  constructor(private httpClient: HttpClient) { }

  // Login
  public login(user:any) {
    return this.httpClient.post(`${this.url}/login`, user);
  }

  //Update block status
  public updateBlockStatus(updateBody:any) {
    return this.httpClient.post(`${this.url}/updateBlockStatus`, updateBody);
  }

  //Update deposit access
  public updateDepositAccess(updateBody:any) {
    return this.httpClient.post(`${this.url}/updateDepositAccess`, updateBody);
  }

  //Update withdrawal access
  public updateWithdrawalAccess(updateBody:any) {
    return this.httpClient.post(`${this.url}/updateWithdrawalAccess`, updateBody);
  }

  //Update transfer access
  public updateTransferAccess(updateBody:any) {
    return this.httpClient.post(`${this.url}/updateTransferAccess`, updateBody);
  }
}
```

```
<div class="container">
   <div class="row">
      <div class="col-sm">

         <div class="account-container">
            <div class="alert alert-secondary" role="alert">
               Pending Cheque Book Requests
            </div>
            <table class="table">
               <thead class="thead-bright">
                  <tr>
                     <th scope="col">Request ID</th>
                     <th scope="col">Customer</th>
                     <th scope="col">Account</th>
                     <th scope="col">Quantity</th>
                     <th scope="col">Status</th>
                  </tr>
               </thead>

               <tr *ngFor="let r of chequeRequests;">
                  <td>{{r.requestId}}</td>
                  <td>{{r.customerName}}</td>
                  <td>{{r.account}}</td>
                  <td>{{r.quantity}}</td>
                  <td>{{r.status}}</td>
               </tr>
            </table>

            <form #approvalForm="ngForm" (ngSubmit)="approveRequest(approvalForm.value)">
               <div class="form-row align-items-center">
                 <div class="col-auto my-1">
                  <label class="mr-sm-2" for="inlineFormCustomSelect">Approve a request</label>
                  <select class="custom-select mr-sm-2" name=inlineFormCustomSelect id="inlineFormCustomSelect"
ngModel>
                    <option *ngFor="let r of chequeRequests" [value]="r.requestId">
                     {{r.requestId}}
                     </option>
                  </select>
                 </div>
                 <div class="col-auto my-1">
                  <button type="submit" class="btn btn-primary">Submit</button>
                 </div>
               </div>
            </form>
         </div>

         <br>

         <div class="account-container">
            <p>
               <a class="btn btn-primary" data-toggle="collapse" href="#blockForm" role="button" aria-
expanded="false" aria-controls="blockForm">
                 Block/Unblock a User
```

```
                </a>
              </p>
            <div class="collapse" id="blockForm">
                <div class="card card-body">
                  <form #blockForm="ngForm" (ngSubmit)="updateBlockStatus(blockForm.value)" clas="form-inline">
                    <div class="form-group mx-sm-3 mb-2">
                        <input type="test" class="form-control" id="firstname" name="firstname" placeholder="User's First
name" ngModel>
                    </div>
                    <div class="form-group mx-sm-3 mb-2">
                        <input type="test" class="form-control" id="lastname" name="lastname" placeholder="Users's Last
name" ngModel>
                    </div>
                    <div class="form-group mx-sm-3 mb-2">
                        <div class="form-check form-check-inline">
                            <input class="form-check-input" type="radio" name="blockStatus" id="blockStatus" value="block"
ngModel>
                            <label class="form-check-label" for="inlineRadio1">Block</label>
                          </div>
                          <div class="form-check form-check-inline">
                            <input class="form-check-input" type="radio" name="blockStatus" id="blockStatus"
value="unblock" ngModel>
                            <label class="form-check-label" for="inlineRadio2">Unblock</label>
                          </div>
                    </div>
                    <button type="submit" class="btn btn-primary mb-2">Submit</button>
                  </form>
                </div>
            </div>
              <ng-container *ngIf="updateBlockStatusFailed">
                <div class="alert alert-danger" role="alert">
                    {{updateBlockStatusErrorMessage}}
                </div>
                </ng-container>
        </div>

        <br>

        <div class="account-container">
            <p>
                <a class="btn btn-primary" data-toggle="collapse" href="#updateAccessForm" role="button" aria-
expanded="false" aria-controls="updateAccessForm">
                  Change User's Access
                </a>
              </p>
            <div class="collapse" id="updateAccessForm">
                <div class="card card-body">
                  <form #updateAccessForm="ngForm" (ngSubmit)="updateAccess(updateAccessForm.value)"
clas="form-inline">
                    <div class="form-group mx-sm-3 mb-2">
                        <input type="test" class="form-control" id="firstname" name="firstname" placeholder="User's First
name" ngModel>
                    </div>
                    <div class="form-group mx-sm-3 mb-2">
                        <input type="test" class="form-control" id="lastname" name="lastname" placeholder="Users's Last
name" ngModel>
                    </div>
                    <div class="col-auto my-1">
                        <label class="mr-sm-2" for="inlineFormCustomSelect">Select access to change</label>
                        <select class="custom-select mr-sm-2" name=inlineFormCustomSelect id="inlineFormCustomSelect"
ngModel>
                            <option value="deposit">Deposit</option>
                            <option value="withdrawal">Withdrawal</option>
                            <option value="transfer">Transfer</option>
                        </select>
                      </div>
                    <div class="form-group mx-sm-3 mb-2">
                        <div class="form-check form-check-inline">
                            <input class="form-check-input" type="radio" name="blockStatus" id="blockStatus" value="block"
ngModel>
                            <label class="form-check-label" for="inlineRadio1">Block</label>
                          </div>
                          <div class="form-check form-check-inline">
                            <input class="form-check-input" type="radio" name="blockStatus" id="blockStatus"
value="unblock" ngModel>
                            <label class="form-check-label" for="inlineRadio2">Unblock</label>
                          </div>
                    </div>
                    <button type="submit" class="btn btn-primary mb-2">Submit</button>
                  </form>
                </div>
            </div>
              <ng-container *ngIf="updateAccessFailed">
                <div class="alert alert-danger" role="alert">
                    {{updateAccessErrorMessage}}
                </div>
                </ng-container>
        </div>

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

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

import { AdminHomeComponent } from './admin-home.component';

describe('AdminHomeComponent', () => {
  let component: AdminHomeComponent;
  let fixture: ComponentFixture<AdminHomeComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ AdminHomeComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(AdminHomeComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AdminService } from 'src/app/services/admin.service';
import { ChequeService } from 'src/app/services/cheque.service';
import { UsersService } from 'src/app/services/users.service';

@Component({
  selector: 'app-admin-home',
  templateUrl: './admin-home.component.html',
  styleUrls: ['./admin-home.component.css']
})
export class AdminHomeComponent implements OnInit {
  chequeRequests;
  updateBlockStatusFailed : boolean = false;
  updateBlockStatusErrorMessage : string = "";
  updateAccessFailed : boolean = false;
  updateAccessErrorMessage : string = "";


  constructor(private router:Router, private service:AdminService, private chequeService:ChequeService) { }


  ngOnInit(): void {
    this.chequeService.getAllPendingRequests().subscribe(
      response => {
        this.chequeRequests = response;
      },
      error => {
        console.log(error);
      }
    )
  }

  public approveRequest(data) {
    let requestId = data.inlineFormCustomSelect;
    let approveBody = new ChequeApproveBody(requestId, "APPROVED");
    this.chequeService.approveRequest(approveBody).subscribe(
      response => {
        window.location.reload();
      },
      error => {
        console.log(error);
      }
    )
  }

  public updateBlockStatus(data) {
    let firstname = data.firstname;
    let lastname = data.lastname;
    let isBlocked = (data.blockStatus == "block") ? true : false;
    let updateBody = new StatusUpdateBody(firstname, lastname, isBlocked)
    this.service.updateBlockStatus(updateBody).subscribe(
      response => {
        window.location.reload();
      },
      error => {
        this.updateBlockStatusFailed = true;
        this.updateBlockStatusErrorMessage = error.error.errorMessage;
      }
    )
  }

  public updateAccess(data) {
    let firstname = data.firstname;
    let lastname = data.lastname;
    let accessToChange = data.inlineFormCustomSelect;
    let accessible = (data.blockStatus == "block") ? false : true;
    let updateBody = new StatusUpdateBody(firstname, lastname, accessible);
    if (accessToChange == "deposit") {
      this.service.updateDepositAccess(updateBody).subscribe(
        response => {
          window.location.reload();
        },
        error => {
          this.updateAccessFailed = true;
          this.updateAccessErrorMessage = error.error.errorMessage;
        }
      )
    }
    else if (accessToChange == "withdrawal") {
      this.service.updateWithdrawalAccess(updateBody).subscribe(
        response => {
          window.location.reload();
        },
        error => {
          this.updateAccessFailed = true;
          this.updateAccessErrorMessage = error.error.errorMessage;
        }
      )
    }
    else {
      this.service.updateTransferAccess(updateBody).subscribe(
        response => {
          window.location.reload();
        },
        error => {
          this.updateAccessFailed = true;
          this.updateAccessErrorMessage = error.error.errorMessage;
        }
      )
    }
  }

}

class ChequeApproveBody {
  id : number;
  status : string;

  constructor(id : number, status : string) {
    this.id = id;
    this.status = status;
  }
}

class StatusUpdateBody{
  firstname : string;
  lastname : string
  accessible : boolean;

  constructor(firstname : string, lastname : string, accessible : boolean) {
    this.firstname = firstname;
    this.lastname = lastname;
    this.accessible = accessible;
  }
}
```

```
.login-form{
   border: 2px solid green;
   padding: 20px;
}
.error {
   color :red;
   font-weight: bold;
   font-style: italic;
}
```

```
<div class="container mt-5">
   <div class="row">
       <div class="col"></div>
       <div class="col-6">
           <h4>Login Form </h4>
           <form class="login-form" #loginForm="ngForm">
               <div class="form-group">
                 <label for="email">Email address</label>
                 <input type="email" class="form-control" id="email" name="email" #email="ngModel"
[(ngModel)]="login.email"
                 required   aria-describedby="emailHelp"  appInputformator>
                 <small *ngIf="hasError(email)" [class]="hasError(email) ? 'error' : ''">
                   <div *ngIf="email.errors">
                          <span *ngIf="email.errors.required">Email address is a required field !</span>
                          <span *ngIf="email.errors.pattern">Entered email address is not a valid email !</span>
                   </div>
                 </small>
               </div>
               <div class="form-group">
                 <label for="password">Password</label>
                 <input type="password" class="form-control" name="password" #password="ngModel"
[(ngModel)]="login.password" id="password"
                 required minlength="6" maxlength="12">
                 <small *ngIf="hasError(password)" [class]="hasError(password) ? 'error' : ''">
                   <div *ngIf="password.errors">
                          <span *ngIf="password.errors.required">Password is a required field !</span>
                          <span *ngIf="password.errors.minlength">Entered password is not valid password min 6 char!
</span>
                          <span *ngIf="password.errors.maxlength">Entered password is not valid password min 12 char!
</span>
                   </div>
                 </small>
               </div>
               <button type="submit" class="btn btn-primary" (click)="onSubmit(loginForm)">Submit</button>
             </form>
       </div>
       <div class="col"></div>
   </div>
</div>

<br>

<ng-container *ngIf="show">
  <div *ngIf="loginSuccess; then success else failed"></div>
</ng-container>

<ng-template #success>
  <div class="alert alert-success" role="alert">
      Login successfully!
  </div>
</ng-template>

<ng-template #failed>
```

```
  <div class="alert alert-danger" role="alert">
      {{loginFailedMessage}}
  </div>
</ng-template>
```

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

import { AdminLoginComponent } from './admin-login.component';

describe('AdminLoginComponent', () => {
  let component: AdminLoginComponent;
  let fixture: ComponentFixture<AdminLoginComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ AdminLoginComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(AdminLoginComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AdminService } from 'src/app/services/admin.service';

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

  public login :Login = {
    email: '',
    password: ''
  };
  public submitted:boolean = false;
  public show:boolean = false;
  public loginSuccess:boolean = false;
  public loginFailedMessage : string = "";

  constructor(private adminSrv:AdminService, private router:Router) { }

  ngOnInit(): void {
  }

  public onSubmit(loginForm:any) {
    if(loginForm.valid) {
      this.submitted = true;
      this.show = true;
      console.log(this.login);
      this.logIn(this.login)
    } else{
      console.log("Invalid Form !");
      this.validateForm(loginForm);
    }
  }

  public validateForm(form:any){
      Object.keys(form.controls).forEach(field => {
        const control = form.controls[field];
        control.markAsTouched({ onlySelf: true});
      })
  }

  hasError(field:any){
    return (field.invalid && field.touched && field.errors);
  }

  public logIn(loginBody:Login) {
    this.adminSrv.login(loginBody).subscribe(
      response => {
        this.loginSuccess = true;
        this.router.navigate(["admin"]);
      },
      error => {
        this.loginSuccess = false;
        this.loginFailedMessage = error.error.errorMessage;
        console.log(error);
      }
    )
  }

}


interface Login {
  email:string;
  password:string;
}
```

```
<app-header></app-header>

<div class="container-fluid">
    <div class="row min-vh-75 flex-column flex-md-row">

        <main class="col bg-faded py-3 flex-grow-1">
            <router-outlet></router-outlet>
        </main>
    </div>
</div>


<app-footer></app-footer>
```

```
import { TestBed } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'icinbank-webapp'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('icinbank-webapp');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
    expect(compiled.querySelector('.content span')?.textContent).toContain('icinbank-webapp app is running!');
  });
});
```

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'icinbank-webapp';
}
```

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './components/home/home.component';
import { FooterComponent } from './components/footer/footer.component';
import { HeaderComponent } from './components/header/header.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpClient, HttpClientModule } from '@angular/common/http';
import { LoginComponent } from './components/login/login.component';
import { AccountComponent } from './components/account/account.component';
import { RegisterComponent } from './components/register/register.component';
import { CheckingAccountDetailsComponent } from './components/checking-account-details/checking-account-
details.component';
import { SavingAccountDetailsComponent } from './components/saving-account-details/saving-account-
details.component';
import { AdminLoginComponent } from './components/admin-login/admin-login.component';
import { AdminHomeComponent } from './components/admin-home/admin-home.component';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    FooterComponent,
    HeaderComponent,
    LoginComponent,
    AccountComponent,
    RegisterComponent,
    CheckingAccountDetailsComponent,
    SavingAccountDetailsComponent,
    AdminLoginComponent,
    AdminHomeComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AccountComponent } from './components/account/account.component';
import { AdminHomeComponent } from './components/admin-home/admin-home.component';
import { AdminLoginComponent } from './components/admin-login/admin-login.component';
import { CheckingAccountDetailsComponent } from './components/checking-account-details/checking-account-
details.component';
import { HomeComponent } from './components/home/home.component';
import { LoginComponent } from './components/login/login.component';
import { RegisterComponent } from './components/register/register.component';
import { SavingAccountDetailsComponent } from './components/saving-account-details/saving-account-
details.component';
const routes: Routes = [
  { path:"", redirectTo:'/home', pathMatch:'full'},
  { path:"home", component:HomeComponent},
  { path:"login", component:LoginComponent},
  { path:"accounts", component:AccountComponent},
  { path:"register", component:RegisterComponent},
  { path:"saving", component:SavingAccountDetailsComponent},
  { path:"checking", component:CheckingAccountDetailsComponent},
  { path: "adminLogin", component:AdminLoginComponent},
  { path:"admin", component:AdminHomeComponent}
];

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

```css
body{
    background-color: oldlace;
}
.account-container{
    margin-top:10%;
    color: orange;
    padding:0;
    background: white;
}
.top-section{
    font-size: 2.5em;
    background-color:#d9dbdb;
    padding: 5%;
    color: #031c63;

}
.bottom-section{
    padding-left: 5%;
    padding-right: 5%;
    padding-top:2%;
    padding-bottom:2%;
    color: #048ec4;
}
hr{
    padding: 0%;
    margin: 0%;
}
.transaction{
    margin-top: 22%;
    background-color: white;
    padding:0;

}
.transaction span{
    font-size: 0.7em;
}
```

```
<div class="alert alert-primary" role="alert">
  Hello! {{ userFirstname }} - Your checking account details.
</div>

<div class="container">
  <div class="row">
    <div class="col-sm">
      <div class="account-container">
        <div class="top-section">
          <div class="balance float-right">$ {{ balance }}</div>
          <div class="account type">Checking Balance:</div>
        </div>
      </div>

      <br />

      <div class="account-container">
        <p>
          <button
            class="btn btn-success"
            type="button"
            data-toggle="collapse"
            data-target="#depositFrm"
            aria-expanded="true"
            aria-controls="depositFrm"
          >
            Deposit Funds
          </button>
        </p>
        <div id="depositFrm">
          <div class="card card-body">
            <form
              #depositForm="ngForm"
              (ngSubmit)="makeDeposit(depositForm.value)"
              clas="form-inline"
            >
              <div class="form-group mx-sm-3 mb-2">
                <input
                  type="number"
                  class="form-control"
                  id="depositAmount"
                  name="depositAmount"
                  placeholder="Deposit Amount"
                  ngModel
                />
              </div>
              <button type="submit" class="btn btn-primary mb-2">Submit</button>
            </form>
          </div>
        </div>
        <ng-container *ngIf="depositFailed">
          <div class="alert alert-danger" role="alert">
            {{ depositErrorMessage }}
          </div>
        </ng-container>
      </div>

      <br />

      <div class="account-container">
        <p>
          <a
            class="btn btn-warning"
            data-toggle="collapse"
            href="#withdrawalForm"
            role="button"
            aria-expanded="false"
            aria-controls="withdrawalForm"
          >
            Withdraw Funds
          </a>
        </p>
        <div id="withdrawalForm">
          <div class="card card-body">
            <form
              #withdrawalForm="ngForm"
              (ngSubmit)="makeWithdrawal(withdrawalForm.value)"
              clas="form-inline"
            >
              <div class="form-group mx-sm-3 mb-2">
                <input
                  type="number"
                  class="form-control"
                  id="withdrawalAmount"
                  name="withdrawalAmount"
                  placeholder="Withdrawal Amount"
                  ngModel
                />
              </div>
              <button type="submit" class="btn btn-primary mb-2">Submit</button>
            </form>
          </div>
        </div>
        <ng-container *ngIf="withdrawalFailed">
          <div class="alert alert-danger" role="alert">
            {{ withdrawalErrorMessage }}
          </div>
        </ng-container>
      </div>

      <br />

      <div class="account-container">
        <p>
          <a
            class="btn btn-primary"
            data-toggle="collapse"
          href="#transferForm"
          role="button"
          aria-expanded="false"
          aria-controls="transferForm"
        >
          Transfer Funds
        </a>
      </p>
      <div id="transferForm">
        <div class="card card-body">
          <form
            #transferForm="ngForm"
            (ngSubmit)="makeTransfer(transferForm.value)"
            clas="form-inline"
          >
            <div class="form-group mx-sm-3 mb-2">
              <input
                type="test"
                class="form-control"
                id="firstnameTo"
                name="firstnameTo"
                placeholder="Receiver's First name"
                ngModel
              />
            </div>
            <div class="form-group mx-sm-3 mb-2">
              <input
                type="test"
                class="form-control"
                id="lastnameTo"
                name="lastnameTo"
                placeholder="Receiver's Last name"
                ngModel
              />
            </div>
            <div class="form-group mx-sm-3 mb-2">
              <input
                type="number"
                class="form-control"
                id="transferAmount"
                name="transferAmount"
                placeholder="Transfer Amount"
                ngModel
              />
            </div>
            <div class="form-group mx-sm-3 mb-2">
              <div class="form-check form-check-inline">
                <input
                  class="form-check-input"
                  type="radio"
                  name="accountTo"
                  id="accountTo"
                  value="checking"
                  ngModel
                />
                <label class="form-check-label" for="inlineRadio1"
                  >To Checking</label
                >
              </div>
              <div class="form-check form-check-inline">
                <input
                  class="form-check-input"
                  type="radio"
                  name="accountTo"
                  id="accountTo"
                  value="saving"
                  ngModel
                />
                <label class="form-check-label" for="inlineRadio2"
                  >To Saving</label
                >
              </div>
            </div>
            <button type="submit" class="btn btn-primary mb-2">Submit</button>
          </form>
        </div>
      </div>
      <ng-container *ngIf="transferFailed">
        <div class="alert alert-danger" role="alert">
          {{ transferErrorMessage }}
        </div>
      </ng-container>
      </div>

      <div class="account-container">
        <div class="alert alert-secondary" role="alert">Transfer History</div>
        <table class="table">
          <thead class="thead-bright">
            <tr>
              <th scope="col">From</th>
              <th scope="col">From Account</th>
              <th scope="col">To</th>
              <th scope="col">To Account</th>
              <th scop="col">Amount</th>
            </tr>
          </thead>

          <tr *ngFor="let th of transactionHistory; let i = index">
            <td>{{ th.from }}</td>
            <td>{{ th.fromAccount }}</td>
            <td>{{ th.to }}</td>
            <td>{{ th.toAccount }}</td>
            <td>{{ th.amount }}</td>
          </tr>
        </table>
      </div>
      </div>
      </div>
```

</div>

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

import { CheckingAccountDetailsComponent } from './checking-account-details.component';

describe('CheckingAccountDetailsComponent', () => {
  let component: CheckingAccountDetailsComponent;
  let fixture: ComponentFixture<CheckingAccountDetailsComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ CheckingAccountDetailsComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(CheckingAccountDetailsComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormBuilder, Validators} from '@angular/forms';
import { Router } from '@angular/router';
import { from } from 'rxjs';
import { AccountsService } from 'src/app/services/accounts.service';
import { ChequeService } from 'src/app/services/cheque.service';
import { TransactionService } from 'src/app/services/transaction.service';

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

  constructor(private router:Router, private formBuilder:FormBuilder, private service:AccountsService,  private
chequeService:ChequeService, private transactionService:TransactionService) { }
  accountId;
  balance;
  userFirstname;
  saving = false;
  userId;
  depositForm : FormGroup;
  withdrawalFailed : boolean = false;
  depositFailed : boolean = false;
  withdrawalErrorMessage : string = "";
  transferFailed : boolean = false;
  transferErrorMessage : string = "";
  depositErrorMessage : string = "";
  transactionHistory;

  ngOnInit(): void {
    this.accountId = sessionStorage.getItem("checkingAccountId");
    this.userFirstname = sessionStorage.getItem("firstname");
    this.userId = sessionStorage.getItem("id");

    this.service.getAccountBalance(this.accountId).subscribe(
      response => {
        this.balance = response;
      },
      error =>{
        console.log(error);
      }
    )

    this.transactionService.getTransactionHistory(Number(this.accountId)).subscribe(
      response => {
        this.transactionHistory = response;
      },
      error => {
        console.log(error);
      }
    )
  }

  public makeDeposit(data) {
    let depositAmount = data.depositAmount;
    let db = new DepositBody(Number(this.userId), depositAmount, this.saving);
    this.service.deposit(db).subscribe(
      response => {
        window.location.reload();
      },
      error => {
        this.depositFailed = true;
        this.depositErrorMessage = error.error.errorMessage;
        console.log(error);
      }
    )
  }

  public makeWithdrawal(data) {
    let withdrawalAmount = data.withdrawalAmount;
    let wb = new WithdrawalBody(Number(this.userId), withdrawalAmount, this.saving);
    this.service.withdrawal(wb).subscribe(
      response => {
        window.location.reload();
      },
      error => {
        this.withdrawalFailed = true;
        console.log(error);
        this.withdrawalErrorMessage = error.error.errorMessage;
      }
    )
  }

  public makeTransfer(data) {
    let firstnameTo = data.firstnameTo;
    let lastnameTo = data.lastnameTo;
    let transferAmount = data.transferAmount;
    let toSaving = (data.accountTo == "saving") ? true : false;

    let tb = new TransferBody(Number(this.userId), firstnameTo, lastnameTo, transferAmount, this.saving, toSaving);

    this.service.transfer(tb).subscribe(
      response => {
        window.location.reload();
      },
      error => {
        this.transferFailed = true;
        console.log(error);
        this.transferErrorMessage = error.error.errorMessage;
      }
    )
  }

}
```

```
class TransferBody {
  userIdFrom : number;
  firstnameTo : string;
  lastnameTo : string;
  amount : number;
  fromSaving : boolean;
  toSaving : boolean;

  constructor(userId : number, firstnameTo : string, lastnameTo : string, amount : number, fromSaving : boolean,
toSaving : boolean) {
    this.userIdFrom = userId;
    this.firstnameTo = firstnameTo;
    this.lastnameTo = lastnameTo;
    this.amount = amount;
    this.fromSaving = fromSaving;
    this.toSaving = toSaving;
  }
}

class DepositBody {
  userId : number;
  amount : number;
  saving : boolean;
  constructor(userId : number, amount : number, saving : boolean) {
    this.userId = userId;
    this.amount = amount;
    this.saving = saving;
  }
}

class WithdrawalBody {
  userId : number;
  amount : number;
  saving : boolean;
  constructor(userId : number, amount : number, saving : boolean) {
    this.userId = userId;
    this.amount = amount;
    this.saving = saving;
  }
}
```

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

import { ChequeService } from './cheque.service';

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

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

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

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class ChequeService {
  private url: string = 'http://localhost:8080/cheque';
  constructor(private httpClient: HttpClient) { }

  //Get cheque book request by userId
  public getChequeBookRequestsByUserId(userId) {
    return this.httpClient.get(`${this.url}/get/${userId}`);
  }

  //Request cheque book
  public requestChequeBook(chequeRequestBody:any) {
    return this.httpClient.post(`${this.url}/request`, chequeRequestBody);
  }

  //Get all pending requests
  public getAllPendingRequests() {
    return this.httpClient.get(`${this.url}/allPending`);
  }

  //Approve cheque book request
  public approveRequest(approveBody:any) {
    return this.httpClient.post(`${this.url}/approve`, approveBody);
  }
}
```

```
export const environment = {
  production: true
};
```

```
// This file can be replaced during build by using the `fileReplacements` array.
// `ng build` replaces `environment.ts` with `environment.prod.ts`.
// The list of file replacements can be found in `angular.json`.

export const environment = {
  production: false,
  appUrl: 'http://localhost:8080',
};

/*
 * For easier debugging in development mode, you can import the following file
 * to ignore zone related error stack frames such as `zone.run`, `zoneDelegate.invokeTask`.
 *
 * This import should be commented out in production mode because it will have a negative impact
 * on performance if an error is thrown.
 */
// import 'zone.js/plugins/zone-error';  // Included with Angular CLI.
```

```
<footer class="footer">
    <div class="container">
        <div class="text-center">
        </div>
        <div>
            <h4 class="text-danger text-center display-4">This is a demo project. Not a real banking website</h4>
        </div>
    </div>
</footer>
```

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

import { FooterComponent } from './footer.component';

describe('FooterComponent', () => {
  let component: FooterComponent;
  let fixture: ComponentFixture<FooterComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ FooterComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(FooterComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

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

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

  constructor() { }

  ngOnInit(): void {
  }

}
```

```
.topnav-right > li{
    padding-left:30px;
    padding-right:30px;
  }
```

```
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
  <a class="navbar-brand" href="#">ICIN Bank</a>
  <button
    class="navbar-toggler"
    type="button"
    data-toggle="collapse"
    data-target="#navbarNavAltMarkup"
    aria-controls="navbarNavAltMarkup"
    aria-expanded="false"
    aria-label="Toggle navigation"
  >
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
    <div class="navbar-nav">
      <a class="nav-item nav-link active" href="/">Home</a>
      <a class="nav-item nav-link active" href="/register">Register</a>
    </div>
  </div>
  <div class="topnav-right">
    <a class="btn btn-outline-success active" href="/login">Login</a>
    <a class="btn btn-outline-warning" href="/adminLogin">Adminstrator</a>
  </div>
</nav>
```

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

import { HeaderComponent } from './header.component';

describe('HeaderComponent', () => {
  let component: HeaderComponent;
  let fixture: ComponentFixture<HeaderComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ HeaderComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(HeaderComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

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

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

  constructor() { }

  ngOnInit(): void {
  }

}
```

```
.bg-dark {
   background-color: #1f9b44!important;
}
```

```
<body class="text-center">
   <div class="cover-container d-flex w-100 h-100 p-3 mx-auto flex-column">
      <main role="main" class="inner cover">
      <h1 class="cover-heading display-2 text-primary">ICIN Bank</h1>
      <p class="display-4 text-success">Letâ€™s Bank on Each Other.</p>
      <p class="lead">
         <img src="assets/bank-image.jpg" alt="">
      </p>
      </main>
   </div>
</body>
```

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

import { HomeComponent } from './home.component';

describe('HomeComponent', () => {
  let component: HomeComponent;
  let fixture: ComponentFixture<HomeComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ HomeComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(HomeComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
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 {
  }

}
```

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>IcinbankWebapp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

```
.login-form{
   border: 2px solid green;
   padding: 20px;
}
.error {
   color :red;
   font-weight: bold;
   font-style: italic;
}
```

```
<div class="container mt-5">
    <div class="row">
        <div class="col"></div>
        <div class="col-6">
            <h4>Login Form </h4>
            <form class="login-form" #loginForm="ngForm">
                <div class="form-group">
                  <label for="email">Email address</label>
                  <input type="email" class="form-control" id="email" name="email" #email="ngModel"
[(ngModel)]="login.email"
                  required   aria-describedby="emailHelp"  appInputformator>
                  <small *ngIf="hasError(email)" [class]="hasError(email) ? 'error' : ''">
                    <div *ngIf="email.errors">
                            <span *ngIf="email.errors.required">Email address is a required field !</span>
                            <span *ngIf="email.errors.pattern">Entered email address is not a valid email !</span>
                    </div>
                  </small>
                </div>
                <div class="form-group">
                  <label for="password">Password</label>
                  <input type="password" class="form-control" name="password" #password="ngModel"
[(ngModel)]="login.password" id="password"
                  required minlength="6" maxlength="12">
                  <small *ngIf="hasError(password)" [class]="hasError(password) ? 'error' : ''">
                    <div *ngIf="password.errors">
                            <span *ngIf="password.errors.required">Password is a required field !</span>
                            <span *ngIf="password.errors.minlength">Password length is incorrect.</span>
                            <span *ngIf="password.errors.maxlength">Password length is incorrect.</span>
                    </div>
                  </small>
                </div>
                <button type="submit" class="btn btn-primary" (click)="onSubmit(loginForm)">Submit</button>
              </form>
        </div>
        <div class="col"></div>
    </div>
</div>

<br>

<ng-container *ngIf="show">
  <div *ngIf="loginSuccess; then success else failed"></div>
</ng-container>

<ng-template #success>
  <div class="alert alert-success" role="alert">
      Login successfully!
      <a href="/questions" class="badge badge-info">Go to Account Page</a>
  </div>
</ng-template>

<ng-template #failed>
  <div class="alert alert-danger" role="alert">
```

```
    {{loginFailedMessage}}
  </div>
</ng-template>
```

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

import { LoginComponent } from './login.component';

describe('LoginComponent', () => {
  let component: LoginComponent;
  let fixture: ComponentFixture<LoginComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ LoginComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(LoginComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { UsersService } from 'src/app/services/users.service';


export class User {
  constructor(
    private id:number,
    private firstname:String,
    private lastname:String,
    private email:String,
    private username:String,
    private password:String,
    private blocked:boolean,
    private transferAccess:boolean,
    private depositAccess:boolean,
    private withdrawalAccess:boolean)
    { }

}

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

  public login :Login = {
    email: '',
    password: ''
  };
  public submitted:boolean = false;
  public show:boolean = false;
  public loginSuccess:boolean = false;
  public loginFailedMessage : string = "";

  constructor(private userSrv:UsersService, private router:Router) { }

  ngOnInit(): void {
  }

  public onSubmit(loginForm:any) {
    if(loginForm.valid) {
      this.submitted = true;
      this.show = true;
      console.log(this.login);
      this.logIn(this.login)
    } else{
      console.log("Invalid Form !");
      this.validateForm(loginForm);
    }
  }
  public validateForm(form:any){
      Object.keys(form.controls).forEach(field => {
        const control = form.controls[field];
        control.markAsTouched({ onlySelf: true});
      })
  }

  hasError(field:any){
    return (field.invalid && field.touched && field.errors);
  }

  public logIn(loginBody:Login) {

    this.userSrv.login(loginBody).subscribe(
      response => {
        console.log(response),
        sessionStorage.removeItem("password");
        sessionStorage.setItem("id", response["id"]);
        sessionStorage.setItem("firstname", response["firstname"]);
        sessionStorage.setItem("lastname", response["lastname"]);
        sessionStorage.setItem("email", response["email"]);
        sessionStorage.setItem("username", response["username"]);
        sessionStorage.setItem("blocked", response["blocked"]);
        sessionStorage.setItem("transferAccess", response["transferAccess"]);
        sessionStorage.setItem("depositAccess", response["depositAccess"]);
        sessionStorage.setItem("withdrawalAccess", response["withdrawalAccess"]);
        console.log("Json object is ")
        sessionStorage.setItem("userObj",JSON.stringify(response));
        this.loginSuccess = true;
        this.router.navigate(["accounts"]);
      },
      error => {
        this.loginSuccess = false;
        this.loginFailedMessage = error.error.errorMessage;
        console.log(error);
      }
    )
  }

}

interface Login {
  email:string;
  password:string;
}
```

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

```
/**
 * This file includes polyfills needed by Angular and is loaded before the app.
 * You can add your own extra polyfills to this file.
 *
 * This file is divided into 2 sections:
 *   1. Browser polyfills. These are applied before loading ZoneJS and are sorted by browsers.
 *   2. Application imports. Files imported after ZoneJS that should be loaded before your main
 *      file.
 *
 * The current setup is for so-called "evergreen" browsers; the last versions of browsers that
 * automatically update themselves. This includes Safari >= 10, Chrome >= 55 (including Opera),
 * Edge >= 13 on the desktop, and iOS 10 and Chrome on mobile.
 *
 * Learn more in https://angular.io/guide/browser-support
 */

/***************************************************************************************************
***
 * BROWSER POLYFILLS
 */

/**
 * IE11 requires the following for NgClass support on SVG elements
 */
// import 'classlist.js';  // Run `npm install --save classlist.js`.

/**
 * Web Animations `@angular/platform-browser/animations`
 * Only required if AnimationBuilder is used within the application and using IE/Edge or Safari.
 * Standard animation support in Angular DOES NOT require any polyfills (as of Angular 6.0).
 */
// import 'web-animations-js';  // Run `npm install --save web-animations-js`.

/**
 * By default, zone.js will patch all possible macroTask and DomEvents
 * user can disable parts of macroTask/DomEvents patch by setting following flags
 * because those flags need to be set before `zone.js` being loaded, and webpack
 * will put import in the top of bundle, so user need to create a separate file
 * in this directory (for example: zone-flags.ts), and put the following flags
 * into that file, and then add the following code before importing zone.js.
 * import './zone-flags';
 *
 * The flags allowed in zone-flags.ts are listed here.
 *
 * The following flags will work for all browsers.
 *
 * (window as any).__Zone_disable_requestAnimationFrame = true; // disable patch requestAnimationFrame
 * (window as any).__Zone_disable_on_property = true; // disable patch onProperty such as onclick
 * (window as any).__zone_symbol__UNPATCHED_EVENTS = ['scroll', 'mousemove']; // disable patch specified
eventNames
 *
 *  in IE/Edge developer tools, the addEventListener will also be wrapped by zone.js
 *  with the following flag, it will bypass `zone.js` patch for IE/Edge
 *
 *  (window as any).__Zone_enable_cross_context_check = true;
 *
 */

/************************************************************************************************
***
 * Zone JS is required by default for Angular itself.
 */
import 'zone.js';  // Included with Angular CLI.


/************************************************************************************************
***
 * APPLICATION IMPORTS
 */
```

```
.register-form{
   border: 2px solid green;
   padding: 20px;
}
.error {
   color :red;
   font-weight: bold;
   font-style: italic;
}
```

```
<div class="container mt-5">
   <div class="row">
      <div class="col"></div>
      <div class="col-6">
         <h4>New Registration Form : </h4>
         <form class="register-form" [formGroup]="registerForm" (ngSubmit)="onSubmit(registerForm)">
            <div class="form-row">
               <!-- First Name -->
               <div class="col-sm-6">
                  <div class="form-group">
                     <label for="firstname">First Name</label>
                     <input type="text" class="form-control" name="firstname" id="firstname"
formControlName="firstname"
                        aria-describedby="firstnameHelp">
                     <small *ngIf="hasError('firstname')" [class]="hasError('firstname') ? 'error' :''">
                        <div *ngIf="f.firstname.errors">
                           <span *ngIf="f.firstname.errors.required">First Name is a required field</span>
                           <span *ngIf="f.firstname.errors.minlength">First Name is a min 3 char long
                              field</span>
                           <span *ngIf="f.firstname.errors.maxlength">First Name is a max 15 char long
                              field</span>
                        </div>
                     </small>
                  </div>
               </div>
               <!-- Last Name -->
               <div class="col-sm-6">
                  <div class="form-group">
                     <label for="lastname">Last Name</label>
                     <input type="text" class="form-control" name="lastname" id="lastname"
formControlName="lastname"
                        aria-describedby="lastnameHelp">
                     <small *ngIf="hasError('lastname')" [class]="hasError('lastname') ? 'error' :''">
                        <div *ngIf="f.lastname.errors">
                           <span *ngIf="f.lastname.errors.required">Last Name is a required field</span>
                           <span *ngIf="f.lastname.errors.minlength">Last Name is a min 3 char long
                              field</span>
                           <span *ngIf="f.lastname.errors.maxlength">Last Name is a max 15 char long
                              field</span>
                        </div>
                     </small>
                  </div>
               </div>
            </div>

            <div class="form-row">
               <!-- Email -->
               <div class="col-sm-6">
                  <div class="form-group">
                     <label for="email">Email address</label>
                     <input type="email" class="form-control" name="email" id="email"   formControlName="email"
                        aria-describedby="emailHelp" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,3}$">
                     <small *ngIf="hasError('email')" [class]="hasError('email') ? 'error' :''">
```

```
                    <div *ngIf="f.email.errors">
                        <span *ngIf="f.email.errors.required">Email is a required field</span>
                        <span *ngIf="f.email.errors.pattern">Email is a not a valid email</span>
                        <span *ngIf="f.email.errors.email">Email is a not a valid email</span>
                    </div>
                </small>
            </div>
        </div>
        <!-- Password -->
        <div class="col-sm-6">
            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" class="form-control" name="password" id="password"
formControlName="password"
                    aria-describedby="passwordHelp" minlength="6" maxlength="15">
                <small *ngIf="hasError('password')" [class]="hasError('password') ? 'error' :''">
                    <div *ngIf="f.password.errors">
                        <span *ngIf="f.password.errors.required">Password is a required field</span>
                        <span *ngIf="f.password.errors.minlength">Password is a min 6 char long field</span>
                        <span *ngIf="f.password.errors.maxlength">Password is a max 16 char long
                            field</span>
                    </div>
                </small>
            </div>
        </div>
    </div>

    <div class="form-row">
        <!-- Username -->
        <div class="col-sm-6">
            <div class="form-group">
                <label for="username">Username</label>
                <input type="text" class="form-control" name="username" id="username"
formControlName="username"
                    aria-describedby="usernameHelp">
                <small *ngIf="hasError('username')" [class]="hasError('username') ? 'error' :''">
                    <div *ngIf="f.username.errors">
                        <span *ngIf="f.username.errors.required">Username is a required field</span>
                        <span *ngIf="f.username.errors.minlength">Username is a min 3 char long
                            field</span>
                        <span *ngIf="f.username.errors.maxlength">Username is a max 15 char long
                            field</span>
                    </div>
                </small>
            </div>
        </div>
    </div>

    <button type="submit" class="btn btn-success btn-block">Submit</button>
  </form>
 </div>
 <div class="col"></div>
</div>
</div>
```

```html
<br>
<br>

<ng-container *ngIf="show">
    <div *ngIf="success; then success else failed"></div>
</ng-container>

<ng-template #success>
    <div class="alert alert-success" role="alert">
        User registered successfully!
        <a href="/login" class="badge badge-info">Login to continue</a>
    </div>
</ng-template>

<ng-template #failed>
    <div class="alert alert-danger" role="alert">
        Something went wrong! Please try again!
    </div>
</ng-template>
```

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

import { RegisterComponent } from './register.component';

describe('RegisterComponent', () => {
  let component: RegisterComponent;
  let fixture: ComponentFixture<RegisterComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ RegisterComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(RegisterComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormBuilder, Validators, FormControl } from '@angular/forms';
import { UsersService } from 'src/app/services/users.service';

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

  public registerForm: FormGroup;
  public submitted: boolean = false;
  public show:boolean = false;
  public success:boolean = false;

  constructor(private fromBuilder: FormBuilder, private userSrv: UsersService) {
    this.registerForm = this.fromBuilder.group({
      firstname: ['', [Validators.required, Validators.minLength(3),Validators.maxLength(15)]],
      lastname: ['',[Validators.required, Validators.minLength(3),Validators.maxLength(15)]],
      email: ['', [Validators.required, Validators.email]],
      password: ['', [Validators.required, Validators.minLength(6),Validators.maxLength(15)]],
      username: ['', [Validators.required, Validators.minLength(3),Validators.maxLength(15)]]
    });
  }

  ngOnInit(): void {
  }

  public onSubmit(loginForm: any) {
    if (loginForm.valid) {
      this.submitted = true;
      this.show = true;
      this.success = true;
      console.log(this.registerForm.value);
      this.register(this.registerForm.value);
    } else {
      console.log("Invalid Form !");
      this.validateForm(loginForm);
    }
  }

  public validateForm(form: any) {
    Object.keys(form.controls).forEach(field => {
      const control = form.controls[field];
      if (control instanceof FormControl) {
        control.markAsTouched({ onlySelf: true });
      } else{
        this.validateForm(control)
      }
    })
  }
  public register(registerBody:any) {
    this.userSrv.addUser(registerBody).subscribe(
      (res) => {
        console.log(res);
      },
      (error) => {
        console.log(error);
      }
    )
  }

  hasError(field: any) {
    return (this.registerForm.get(field)?.invalid && this.registerForm.get(field)?.touched &&
this.registerForm.get(field)?.errors);
  }
  get f() {
    return this.registerForm.controls;
  }
}
```

```
body{
    background-color: oldlace;
}
.account-container{
    margin-top:10%;
    color: orange;
    padding:0;
    background: white;
}
.top-section{
    font-size: 2.5em;
    background-color:#babcc2;
    padding: 5%;
    color: #031c63;

}
.bottom-section{
    padding-left: 5%;
    padding-right: 5%;
    padding-top:2%;
    padding-bottom:2%;
    color: #048ec4;
}
hr{
    padding: 0%;
    margin: 0%;
}
.transaction{
    margin-top: 22%;
    background-color: white;
    padding:0;

}
.transaction span{
    font-size: 0.7em;
}
```

```
<div class="alert alert-primary" role="alert">
  Hello! {{ userFirstname }}, Here details of your saving account.
</div>

<div class="container">
  <div class="row">
    <div class="col-sm">
      <div class="account-container">
        <div class="top-section">
          <div class="balance float-right">$ {{ balance }}</div>
          <div class="account type">Savings Balance:</div>
        </div>
      </div>

      <br />

      <div class="account-container">
        <p>
          <a
            class="btn btn-success"
            data-toggle="collapse"
            href="#depositForm"
            role="button"
            aria-expanded="false"
            aria-controls="depositForm"
          >
            Deposit Funds
          </a>
        </p>
        <div id="depositForm">
          <div class="card card-body">
            <form
              #depositForm="ngForm"
              (ngSubmit)="makeDeposit(depositForm.value)"
              clas="form-inline"
            >
              <div class="form-group mx-sm-3 mb-2">
                <input
                  type="number"
                  class="form-control"
                  id="depositAmount"
                  name="depositAmount"
                  placeholder="Deposit Amount"
                  ngModel
                />
              </div>
              <button type="submit" class="btn btn-primary mb-2">Submit</button>
            </form>
          </div>
        </div>
      </div>

      <br />
```

```
<div class="account-container">
  <p>
    <a
      class="btn btn-warning"
      data-toggle="collapse"
      href="#withdrawalForm"
      role="button"
      aria-expanded="false"
      aria-controls="withdrawalForm"
    >
      Withdraw Funds
    </a>
  </p>
  <div id="withdrawalForm">
    <div class="card card-body">
      <form
        #withdrawalForm="ngForm"
        (ngSubmit)="makeWithdrawal(withdrawalForm.value)"
        clas="form-inline"
      >
        <div class="form-group mx-sm-3 mb-2">
          <input
            type="number"
            class="form-control"
            id="withdrawalAmount"
            name="withdrawalAmount"
            placeholder="Withdrawal Amount"
            ngModel
          />
        </div>
        <button type="submit" class="btn btn-primary mb-2">Submit</button>
      </form>
    </div>
  </div>
  <ng-container *ngIf="withdrawalFailed">
    <div class="alert alert-warning" role="alert">
      {{ withdrawalErrorMessage }}
    </div>
  </ng-container>
</div>

<br />

<div class="account-container">
  <p>
    <a
      class="btn btn-primary"
      data-toggle="collapse"
      href="#transferForm"
      role="button"
      aria-expanded="false"
      aria-controls="transferForm"
    >
    Transfer Funds
  </a>
</p>
<div id="transferForm">
  <div class="card card-body">
    <form
      #transferForm="ngForm"
      (ngSubmit)="makeTransfer(transferForm.value)"
      clas="form-inline"
    >
      <div class="form-group mx-sm-3 mb-2">
        <input
          type="test"
          class="form-control"
          id="firstnameTo"
          name="firstnameTo"
          placeholder="Receiver's First name"
          ngModel
        />
      </div>
      <div class="form-group mx-sm-3 mb-2">
        <input
          type="test"
          class="form-control"
          id="lastnameTo"
          name="lastnameTo"
          placeholder="Receiver's Last name"
          ngModel
        />
      </div>
      <div class="form-group mx-sm-3 mb-2">
        <input
          type="number"
          class="form-control"
          id="transferAmount"
          name="transferAmount"
          placeholder="Transfer Amount"
          ngModel
        />
      </div>
      <div class="form-group mx-sm-3 mb-2">
        <div class="form-check form-check-inline">
          <input
            class="form-check-input"
            type="radio"
            name="accountTo"
            id="accountTo"
            value="checking"
            ngModel
          />
          <label class="form-check-label" for="inlineRadio1"
            >To Checking</label
          >
        </div>
```

```
          <div class="form-check form-check-inline">
            <input
              class="form-check-input"
              type="radio"
              name="accountTo"
              id="accountTo"
              value="saving"
              ngModel
            />
            <label class="form-check-label" for="inlineRadio2"
              >To Saving</label
            >
          </div>
        </div>
        <button type="submit" class="btn btn-primary mb-2">Submit</button>
      </form>
    </div>
  </div>
  <ng-container *ngIf="transferFailed">
    <div class="alert alert-danger" role="alert">
      {{ transferErrorMessage }}
    </div>
  </ng-container>
</div>

<div class="account-container">
  <div class="alert alert-secondary" role="alert">Transfer History</div>
  <table class="table">
    <thead class="thead-bright">
      <tr>
        <th scope="col">From</th>
        <th scope="col">From Account</th>
        <th scope="col">To</th>
        <th scope="col">To Account</th>
        <th scop="col">Amount</th>
      </tr>
    </thead>

    <tr *ngFor="let th of transactionHistory; let i = index">
      <td>{{ th.from }}</td>
      <td>{{ th.fromAccount }}</td>
      <td>{{ th.to }}</td>
      <td>{{ th.toAccount }}</td>
      <td>{{ th.amount }}</td>
    </tr>
  </table>
</div>
```
```
    </div>
  </div>
</div>
```

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

import { SavingAccountDetailsComponent } from './saving-account-details.component';

describe('SavingAccountDetailsComponent', () => {
  let component: SavingAccountDetailsComponent;
  let fixture: ComponentFixture<SavingAccountDetailsComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ SavingAccountDetailsComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(SavingAccountDetailsComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { FormGroup, FormBuilder } from '@angular/forms';
import { AccountsService } from 'src/app/services/accounts.service';
import { ChequeService } from 'src/app/services/cheque.service';
import { TransactionService } from 'src/app/services/transaction.service';

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

  constructor(private router:Router, private formBuilder:FormBuilder, private service:AccountsService,  private
chequeService:ChequeService, private transactionService:TransactionService) { }

  accountId;
  balance;
  userFirstname;
  saving = true;
  userId;
  depositForm : FormGroup;
  withdrawalFailed : boolean = false;
  withdrawalErrorMessage : string = "";
  transferFailed : boolean = false;
  transferErrorMessage : string = "";
  transactionHistory;

  ngOnInit(): void {
    this.accountId = sessionStorage.getItem("savingAccountId");
    this.userFirstname = sessionStorage.getItem("firstname");
    this.userId = sessionStorage.getItem("id");

    this.service.getAccountBalance(this.accountId).subscribe(
      response => {
        this.balance = response;
      },
      error =>{
        console.log(error);
      }
    )

    this.transactionService.getTransactionHistory(Number(this.accountId)).subscribe(
      response => {
        this.transactionHistory = response;
      },
      error => {
        console.log(error);
      }
    )
  }
  public makeDeposit(data) {
    let depositAmount = data.depositAmount;
    let db = new DepositBody(Number(this.userId), depositAmount, this.saving);
    this.service.deposit(db).subscribe(
      response => {
        window.location.reload();
      },
      error => {
        console.log(error);
      }
    )
  }

  public makeWithdrawal(data) {
    let withdrawalAmount = data.withdrawalAmount;
    let wb = new WithdrawalBody(Number(this.userId), withdrawalAmount, this.saving);
    this.service.withdrawal(wb).subscribe(
      response => {
        window.location.reload();
      },
      error => {
        this.withdrawalFailed = true;
        console.log(error);
        this.withdrawalErrorMessage = error.error.errorMessage;
      }
    )
  }

  public makeTransfer(data) {
    let firstnameTo = data.firstnameTo;
    let lastnameTo = data.lastnameTo;
    let transferAmount = data.transferAmount;
    let toSaving = (data.accountTo == "saving") ? true : false;

    let tb = new TransferBody(Number(this.userId), firstnameTo, lastnameTo, transferAmount, this.saving, toSaving);

    this.service.transfer(tb).subscribe(
      response => {
        window.location.reload();
      },
      error => {
        this.transferFailed = true;
        console.log(error);
        this.transferErrorMessage = error.error.errorMessage;
      }
    )
  }

}

class TransferBody {
  userIdFrom : number;
  firstnameTo : string;
  lastnameTo : string;
  amount : number;
  fromSaving : boolean;
  toSaving : boolean;

  constructor(userId : number, firstnameTo : string, lastnameTo : string, amount : number, fromSaving : boolean,
toSaving : boolean) {
    this.userIdFrom = userId;
    this.firstnameTo = firstnameTo;
    this.lastnameTo = lastnameTo;
    this.amount = amount;
    this.fromSaving = fromSaving;
    this.toSaving = toSaving;
  }
}

class DepositBody {
  userId : number;
  amount : number;
  saving : boolean;
  constructor(userId : number, amount : number, saving : boolean) {
    this.userId = userId;
    this.amount = amount;
    this.saving = saving;
  }
}

class WithdrawalBody {
  userId : number;
  amount : number;
  saving : boolean;
  constructor(userId : number, amount : number, saving : boolean) {
    this.userId = userId;
    this.amount = amount;
    this.saving = saving;
  }
}
```

```
/* You can add global styles to this file, and also import other style files */
```

```
// This file is required by karma.conf.js and loads recursively all the .spec and framework files

import 'zone.js/testing';
import { getTestBed } from '@angular/core/testing';
import {
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting
} from '@angular/platform-browser-dynamic/testing';

declare const require: {
  context(path: string, deep?: boolean, filter?: RegExp): {
    keys(): string[];
    <T>(id: string): T;
  };
};

// First, initialize the Angular testing environment.
getTestBed().initTestEnvironment(
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting(),
  { teardown: { destroyAfterEach: true }},
);

// Then we find all the tests.
const context = require.context('./', true, /\.spec\.ts$/);
// And load the modules.
context.keys().map(context);
```

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

import { TransactionService } from './transaction.service';

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

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

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

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

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

  private url: string = 'http://localhost:8080/transactions';
  constructor(private httpClient: HttpClient) { }

  //Get transaction history
  public getTransactionHistory(accountId:number) {
    return this.httpClient.get(`${this.url}/getHistory/${accountId}`);
  }
}
```

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

import { UsersService } from './users.service';

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

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

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

```
import { HttpClient, HttpParams, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';

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

  private url: string = 'http://localhost:8080/users';
  constructor(private httpClient: HttpClient) { }


  // Login
  public login(user:any) {
    return this.httpClient.post(`${this.url}/login`, user);
  }

  //Registe
  public addUser(user:any) {
    return this.httpClient.post(`${this.url}/add`, user);
  }
}
```