Lab 7 - CRUD Customer Application

Brett Recker - Class ID: 22

Redwan Albadawi - Class ID: 1

Objective

Create a CRUD application for customer data (customer id, customer name, customer email) using rest api’s and nodejs. The application needs to be able to add new customers, list all customers, update customers, and delete customers.

Customers API Service

This service contains all the basic functions performed by the application including getting, posting, updating and deleting customers:

api.service.ts

```
      `body was: ${error.error}`);
    }
    // return an observable with a user-facing error message
    return throwError( error: 'Something bad happened; please try again later.');
  };

  private extractData(res: Response) {
    let body = res;
    return body || { };
  }

  getCustomers(): Observable<any> {
    return this.http.get(apiUrl, httpOptions).pipe(
      map(this.extractData),
      catchError(this.handleError));
  }

  getCustomer(id: string): Observable<any> {
    const url = `${apiUrl}/${id}`;
    return this.http.get(url, httpOptions).pipe(
      map(this.extractData),
      catchError(this.handleError));
  }

  postCustomer(data): Observable<any> {
    return this.http.post(apiUrl, data, httpOptions)
      .pipe(
        catchError(this.handleError)
      );
  }

  updateCustomer(id: string, data): Observable<any> {
    const url = `${apiUrl}/${id}`;
    return this.http.put(url, data, httpOptions)
      .pipe(
        catchError(this.handleError)
      );
  }

  deleteCustomer(id: string): Observable<{}> {
    const url = `${apiUrl}/${id}`;
    return this.http.delete(url, httpOptions)
      .pipe(
        catchError(this.handleError)
      );
  }

}
```

App Routes and app.js

api.service.ts | app.module.ts

```
import {
  MatInputModule,
  MatPaginatorModule,
  MatProgressSpinnerModule,
```

```
    MatSortModule,
    MatTableModule,
    MatIconModule,
    MatButtonModule,
    MatCardModule,
    MatFormFieldModule } from '@angular/material';

const appRoutes: Routes = [
  {
    path: 'customers',
    component: CustomerComponent,
    data: { title: 'Customer List' }
  },
  {
    path: 'customer-details/:id',
    component: CustomerDetailComponent,
    data: { title: 'Customer Details' }
  },
  {
    path: 'customer-create',
    component: CustomerCreateComponent,
    data: { title: 'Create Customer' }
  },
  {
    path: 'customer-edit/:id',
    component: CustomerEditComponent,
    data: { title: 'Edit Customer' }
  },
  { path: '',
    redirectTo: '/customers',
    pathMatch: 'full'
  }
];

@NgModule({
  declarations: [
    AppComponent,
    CustomerComponent,
    CustomerDetailComponent,
    CustomerCreateComponent,
    CustomerEditComponent
  ],
  imports: [
    RouterModule.forRoot(appRoutes),
```

app.js

```
var createError = require( id: 'http-errors');
var express = require( id: 'express');
var path = require( id: 'path');
var favicon = require( id: 'serve-favicon');
var logger = require( id: 'morgan');

var mongoose = require( id: 'mongoose');
mongoose.connect('mongodb+srv://lab7:lab7@cluster0-ydpjq.mongodb.net/book?retryWrites=true')
  .then(() => console.log('connection successful'))
  .catch( onrejected: (err) => console.error(err));

var apiRouter = require( id: './routes/customer');

var app = express();

app.use(logger( format: 'dev'));
app.use(express.json());
app.use(express.urlencoded({extended: false}));
app.use(express.static(path.join(__dirname, 'dist/lab7')));
app.use( fn: '/customers', express.static(path.join(__dirname, 'dist/lab7')));
app.use( fn: '/customer-details/:id', express.static(path.join(__dirname, 'dist/lab7')));
app.use( fn: '/customer-create', express.static(path.join(__dirname, 'dist/lab7')));
app.use( fn: '/customer-edit/:id', express.static(path.join(__dirname, 'dist/lab7')));
app.use( fn: '/api', apiRouter);

// catch 404 and forward to error handler
app.use( fn: function (req, res, next) {
  next(createError(404));
});

// error handler
app.use( fn: function (err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.send(err.status);
});

module.exports = app;
```

Customer Component

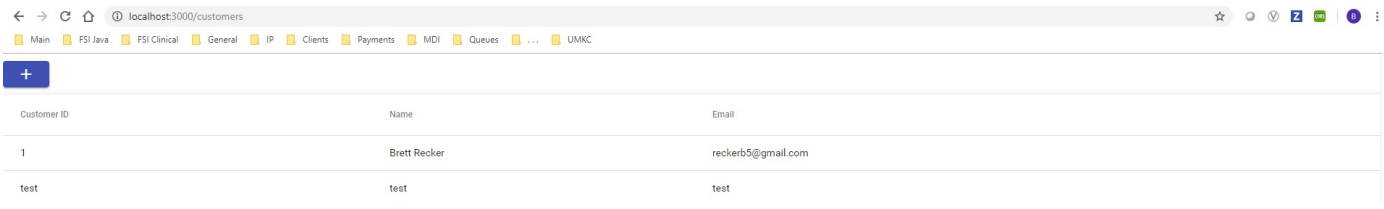
App home screen which shows a list of all existing customers. Also, contains a button to allow for adding new customers:

customer.component.html

```
<div class="button-row">
  <a mat-raised-button color="primary" [routerLink]="['/customer-create']">
    <mat-icon>add</mat-icon>
  </a>
</div>
<div class="example-container mat-elevation-z8">
  <table mat-table #table [dataSource]="dataSource">

    <!--- Note that these columns can be defined in any order.
          The actual rendered columns are set as a property on the row definition" -->

    <!-- ID Column -->
    <ng-container matColumnDef="customer_id">
      <th mat-header-cell *matHeaderCellDef>Customer ID</th>
      <td mat-cell *matCellDef="let element" class="id-col"> {{element.customer_id}} </td>
    </ng-container>

    <!-- Name Column -->
    <ng-container matColumnDef="customer_name">
      <th mat-header-cell *matHeaderCellDef> Name</th>
      <td mat-cell *matCellDef="let element"> {{element.customer_name}} </td>
    </ng-container>

    <!-- Email Column -->
    <ng-container matColumnDef="customer_email">
      <th mat-header-cell *matHeaderCellDef> Email</th>
      <td mat-cell *matCellDef="let element"> {{element.customer_email}} </td>
    </ng-container>

    <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
    <tr mat-row *matRowDef="let row; columns: displayedColumns;" [routerLink]="['/customer-details/', row._id]"></tr>
  </table>
</div>
```

customer.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { ApiService } from '../api.service';
import { DataSource } from '@angular/cdk/collections';
import { Observable } from 'rxjs';

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

  customers: any;
  displayedColumns = ['customer_id', 'customer_name', 'customer_email'];
  dataSource = new CustomerDataSource(this.api);

  constructor(private api: ApiService) { }

  ngOnInit() {
    this.api.getCustomers()
      .subscribe( next: res => {
        console.log('RES' + res);
        this.customers = res;
      }, error: err => {
        console.log(err);
      });
  }
}

export class CustomerDataSource extends DataSource<any> {
  constructor(private api: ApiService) {
    super();
  }

  connect() {
    return this.api.getCustomers();
  }

  disconnect() {

  }
}
```



Customer Create

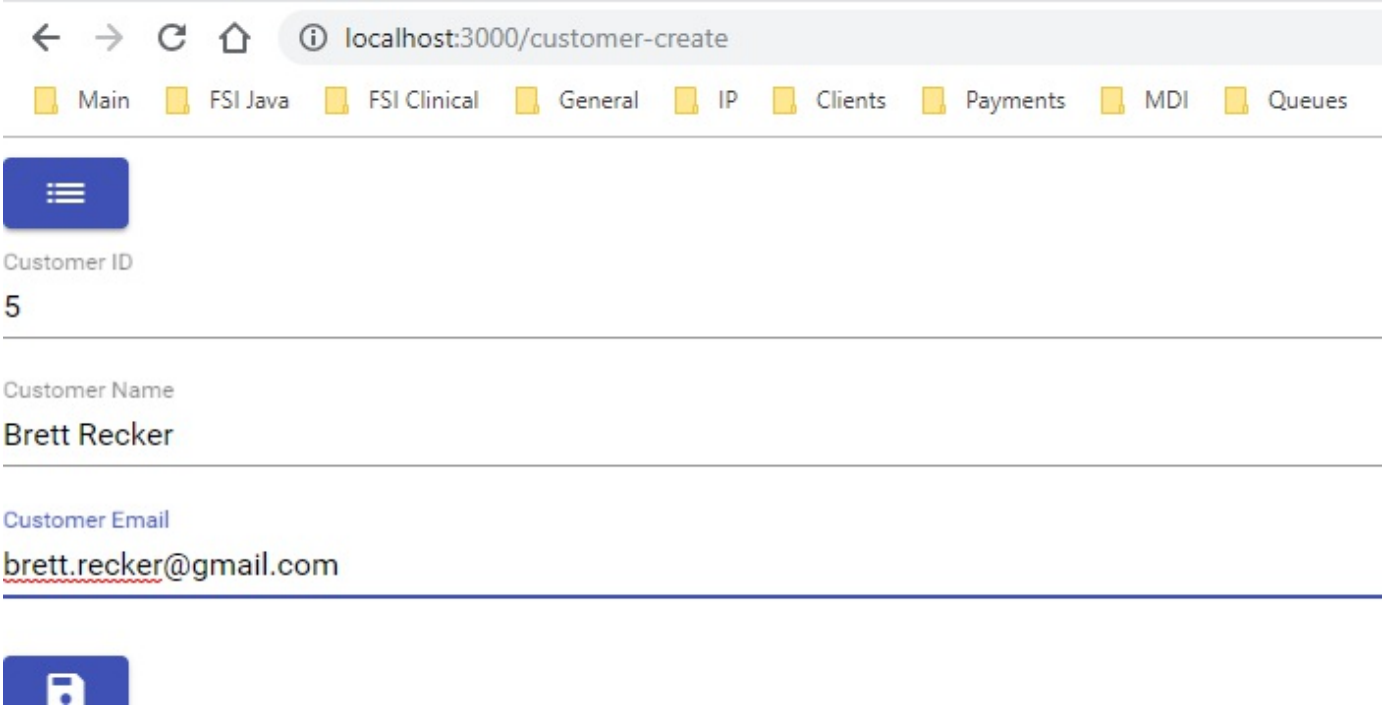
Includes input boxes for customer id, name, and email to post to database:

customer-create.component.html

```
<div class="button-row">
  <a mat-raised-button color="primary" [routerLink]="['/customers']">
    <mat-icon>list</mat-icon>
  </a>
</div>
<form [formGroup]="customerForm" (ngSubmit)="onFormSubmit(customerForm.value)">
  <mat-form-field class="example-full-width">
    <input matInput placeholder="Customer ID" formControlName="customer_id"
           [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!customerForm.get('customer_id').valid && customerForm.get('customer_id').touched">Please enter Customer ID</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <input matInput placeholder="Customer Name" formControlName="customer_name"
           [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!customerForm.get('customer_name').valid && customerForm.get('customer_name').touched">Please enter Customer Name</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <input matInput placeholder="Customer Email" formControlName="customer_email"
           [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!customerForm.get('customer_email').valid && customerForm.get('customer_email').touched">Please enter Customer Email</span>
    </mat-error>
  </mat-form-field>
  <div class="button-row">
    <button type="submit" [disabled]="!customerForm.valid" mat-raised-button color="primary">
      <mat-icon>save</mat-icon>
    </button>
  </div>
</form>
```

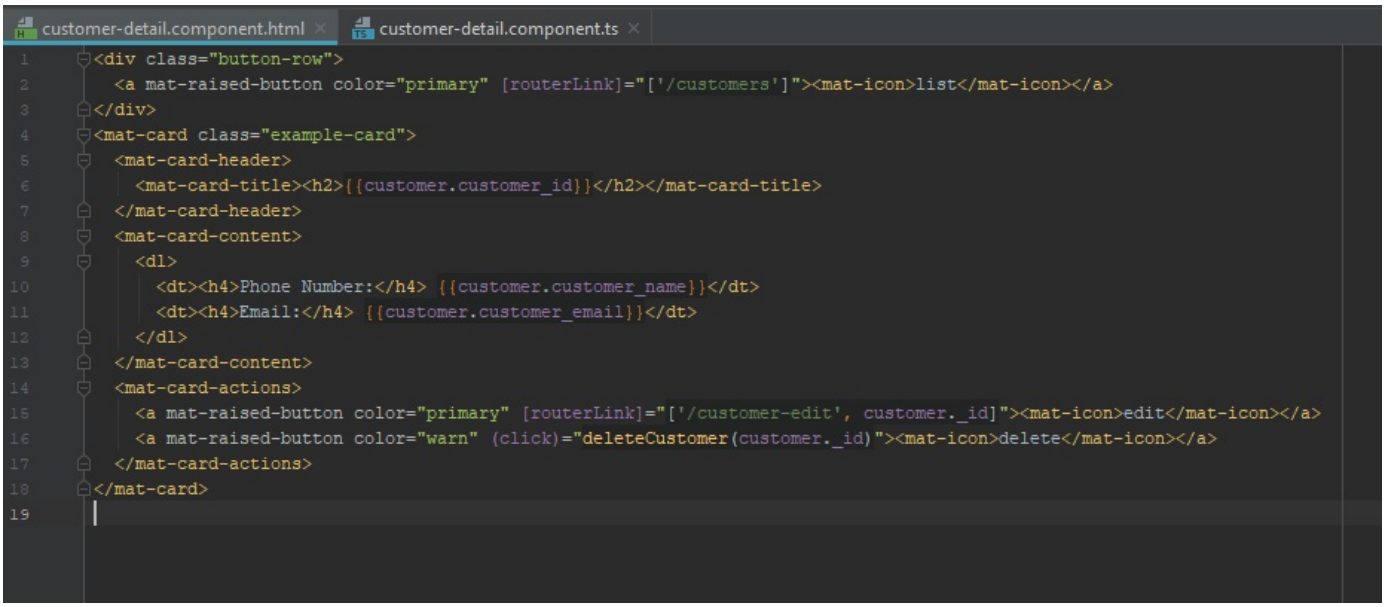
customer-create.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { ApiService } from '../api.service';
import { FormControl, FormGroupDirective, FormBuilder, FormGroup, NgForm, Validators } from '@angular/forms';

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

  customerForm: FormGroup;
  customer_id: string = '';
  customer_name: string = '';
  customer_email: string = '';

  constructor(private router: Router, private api: ApiService, private formBuilder: FormBuilder) { }

  ngOnInit() {
    this.customerForm = this.formBuilder.group( controlsConfig: {
      'customer_id': [null, Validators.required],
      'customer_name': [null, Validators.required],
      'customer_email': [null, Validators.required]
    });
  }


  onFormSubmit(form: NgForm) {
    this.api.postCustomer(form)
      .subscribe( next: res => {
        let id = res['_id'];
        this.router.navigate( commands: ['/customer-details', id]);
      }, error: (err) => {
        console.log(err);
      });
  }

}
```



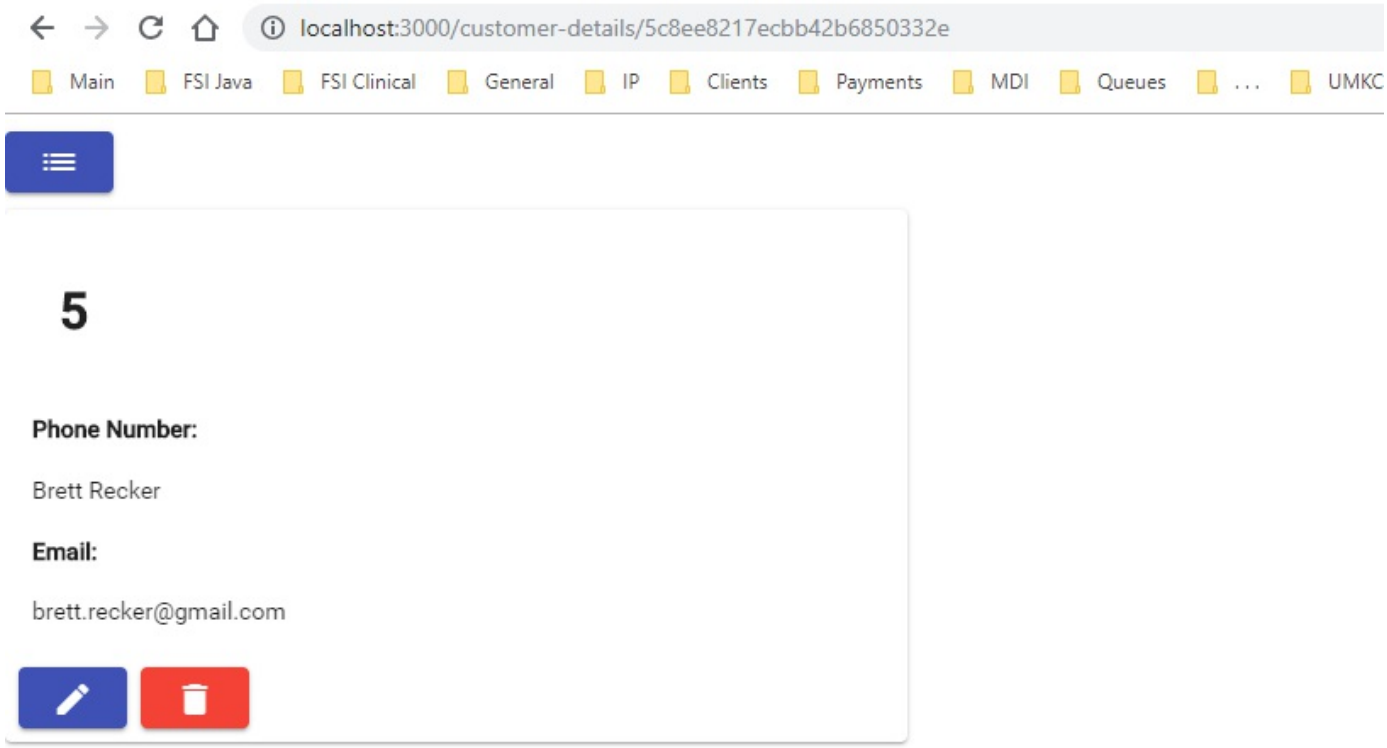
Customer Detail

Shows individual customer details and allows for editing of the customer or deleting:



customer-detail.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { ApiService } from '../api.service';
import { ActivatedRoute, Router } from '@angular/router';

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

  customer = {};

  constructor(private route: ActivatedRoute, private api: ApiService, private router: Router) { }

  ngOnInit() {
    this.getCustomerDetails(this.route.snapshot.params['id']);
  }

  getCustomerDetails(id) {
    this.api.getCustomer(id)
      .subscribe( next: data => {
        console.log(data);
        this.customer = data;
      });
  }

  deleteCustomer(id) {
    this.api.deleteCustomer(id)
      .subscribe( next: res => {
          this.router.navigate( commands: ['/customers']);
        }, error: (err) => {
          console.log(err);
        }
      );
  }

}
```



Customer Edit

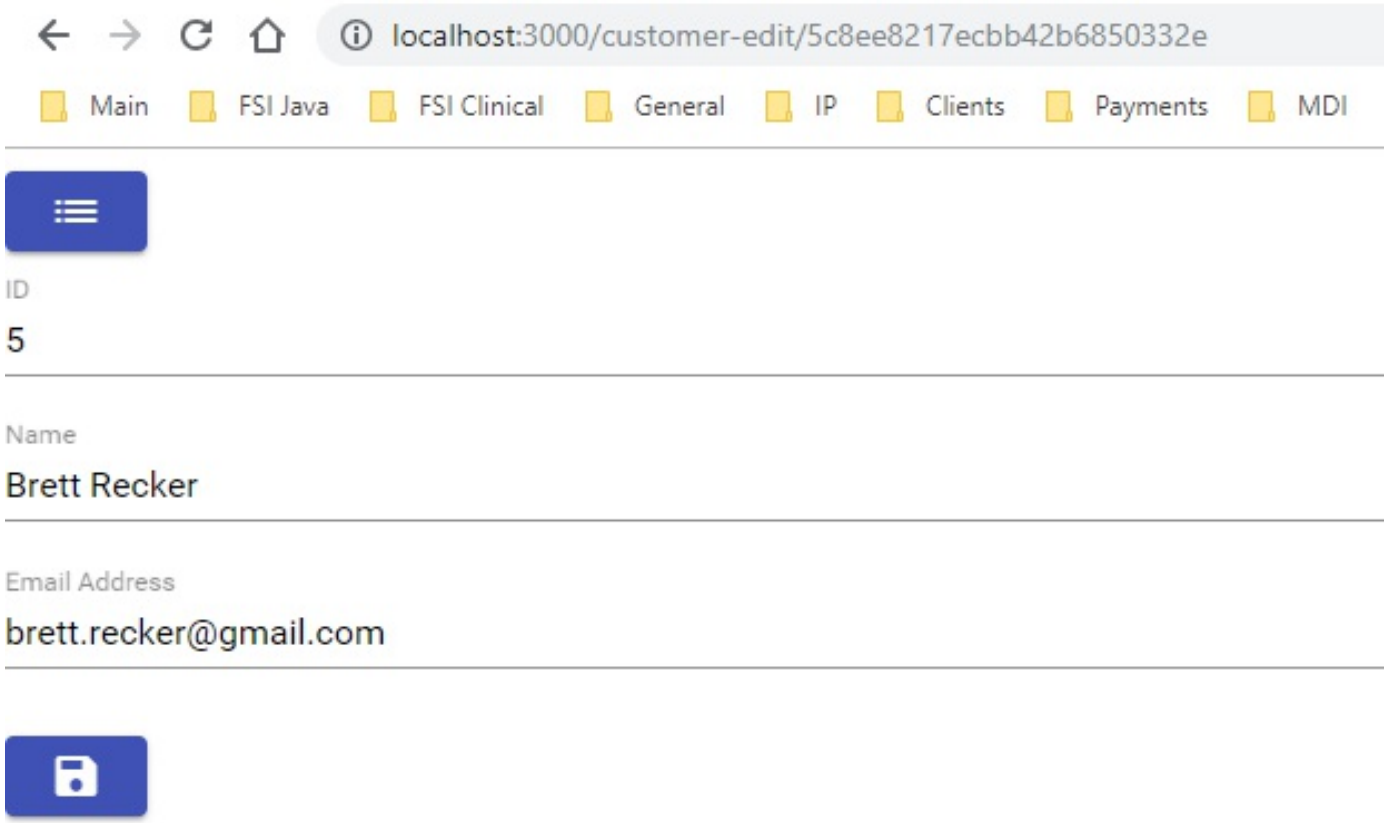
Form similar to adding customer but displays current customer details and allows for updating the customer:

customer-edit.component.html | customer-edit.component.ts

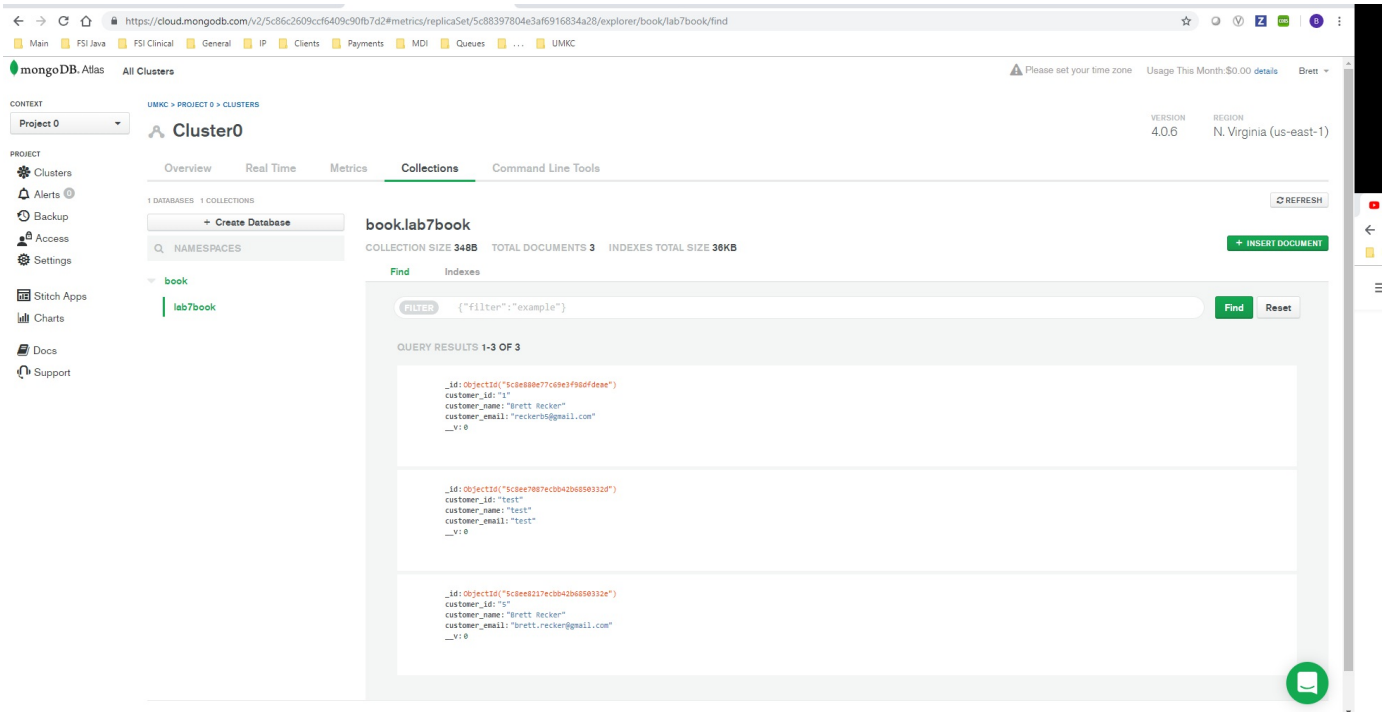
```html
<div class="button-row">
  <a mat-raised-button color="primary" [routerLink]="['/customers']">
    <mat-icon>list</mat-icon>
  </a>
</div>
<form [formGroup]="customerForm" (ngSubmit)="onFormSubmit(customerForm.value)">
  <mat-form-field class="example-full-width">
    <input matInput placeholder="ID" formControlName="customer_id"
          [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!customerForm.get('customer_id').valid && customerForm.get('customer_id').touched">Please enter ID</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <input matInput placeholder="Name" formControlName="customer_name"
          [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!customerForm.get('customer_name').valid && customerForm.get('customer_name').touched">Please enter Customer Name</span>
    </mat-error>
  </mat-form-field>
  <mat-form-field class="example-full-width">
    <input matInput placeholder="Email Address" formControlName="customer_email"
          [errorStateMatcher]="matcher">
    <mat-error>
      <span *ngIf="!customerForm.get('customer_email').valid && customerForm.get('customer_email').touched">Please enter Email Address</span>
    </mat-error>
  </mat-form-field>
    <div class="button-row">
      <button type="submit" [disabled]="!customerForm.valid" mat-raised-button color="primary">
        <mat-icon>save</mat-icon>
      </button>
    </div>
</form>
```

customer-edit.component.html | customer-edit.component.ts

```ts
import { FormControl, FormGroupDirective, FormBuilder, FormGroup, NgForm, Validators } from '@angular/forms';

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

  customerForm: FormGroup;
  id: string = '';
  customer_id: string = '';
  customer_name: string = '';
  customer_email: string = '';

  constructor(private router: Router, private route: ActivatedRoute, private api: ApiService, private formBuilder: FormBuilder) {
  }
  ngOnInit() {
    this.getCustomer(this.route.snapshot.params['id']);
    this.customerForm = this.formBuilder.group( controlsConfig: {
      'customer_id': [null, Validators.required],
      'customer_name': [null, Validators.required],
      'customer_email': [null, Validators.required]
    });
  }

  onFormSubmit(form: NgForm) {
    let id = this.route.snapshot.params['id'];
    console.log(form)
    this.api.updateCustomer(id, form)
      .subscribe( next: res => {
        this.router.navigate( commands: ['/customer-details', id]);
      }, error: (err) => {
        console.log(err);
      });
  }

  getCustomer(id) {
    this.api.getCustomer(id).subscribe( next: data => {
      id = data._id;
      this.customerForm.setValue( value: {
        customer_id: data.customer_id,
        customer_name: data.customer_name,
        customer_email: data.customer_email,
      });
    });
  }
}
```



Lastly, here is the mongodb collection for my app as well as the connection settings:




```
customer.js  app.js

var createError = require( id: 'http-errors');
var express = require( id: 'express');
var path = require( id: 'path');
var favicon = require( id: 'serve-favicon');
var logger = require( id: 'morgan');

var mongoose = require( id: 'mongoose');
mongoose.connect('mongodb+srv://lab7:lab7@cluster0-ydpjq.mongodb.net/book?retryWrites=true')
  .then(() => console.log('connection successful'))
  .catch( onrejected: (err) => console.error(err));

var apiRouter = require( id: './routes/customer');
```

Outcome

This lab was a great learning experience to build off of lab6. I learned a lot about CRUD applications and this will translate very well into our project as we are storing ModelKB artifacts to a MongoDB database as well.