SWEN325 Assignment 1 Report

Philip Oliver 300398228

Application Architecture

Mobile application architecture is split into 4 layers. These layers are: Presentation, Business, Data Access, and Service. In an ideal architecture, these 4 layers will be seperate from each other.

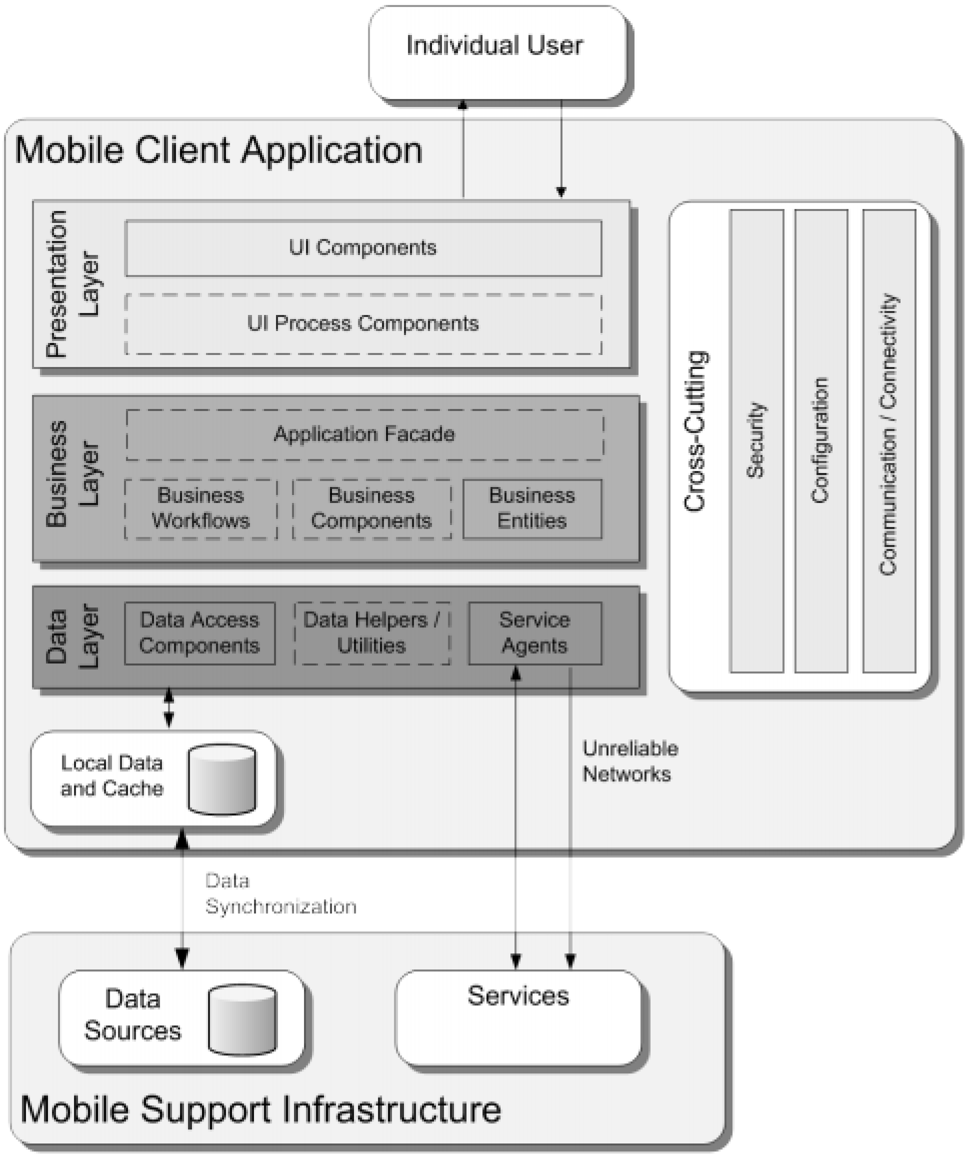
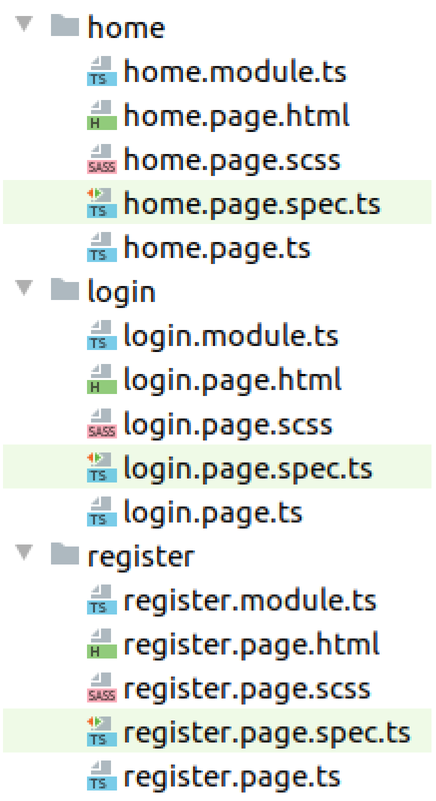


Image from 'Mobile Architecture Guide': Microsoft

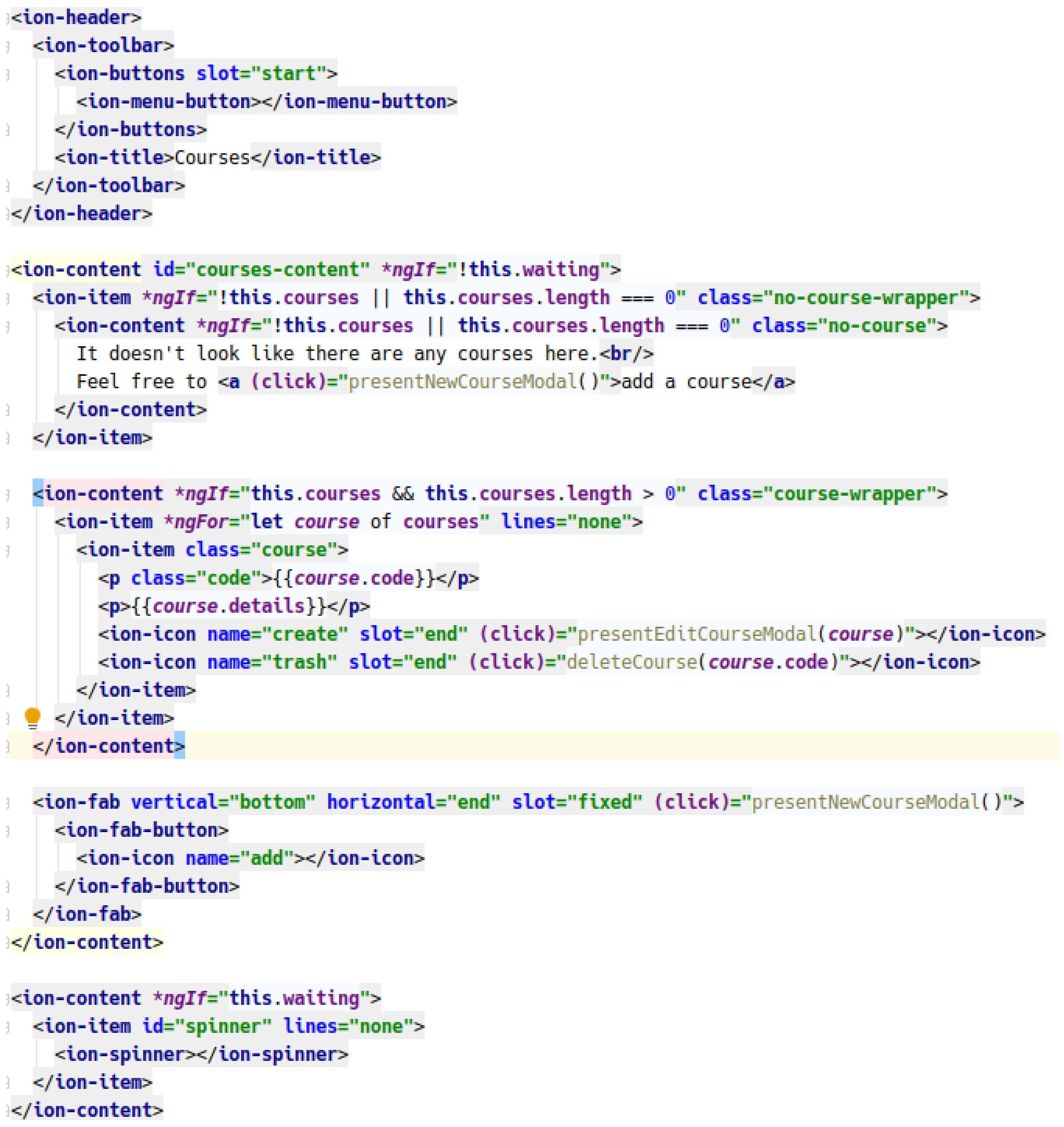
Presentation Layer

The presentation layer in an Ionic application is typically represented in the HTML and CSS files, and the Ionic components which are converted to HTML elements for display. Due to Ionic automatically separating the HTML, CSS, and typescript files, the presentation layer is separated from the other layers.



```
▼ home
    home.module.ts
    home.page.html
    home.page.scss
    home.page.spec.ts
    home.page.ts
▼ login
    login.module.ts
    login.page.html
    login.page.scss
    login.page.spec.ts
    login.page.ts
▼ register
    register.module.ts
    register.page.html
    register.page.scss
    register.page.spec.ts
    register.page.ts
```

The separate files can be seen in the above image.

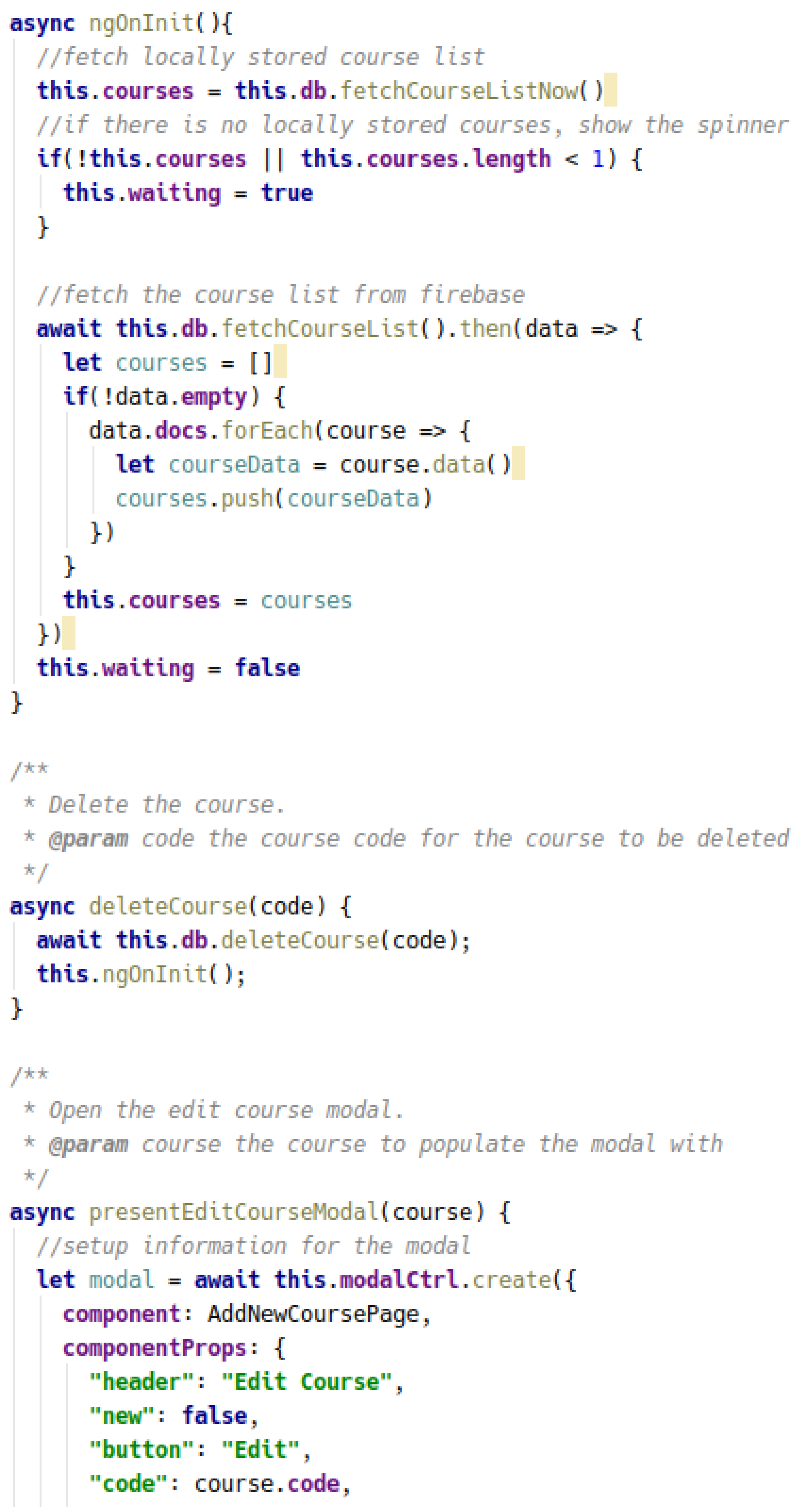


```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button></ion-menu-button>
    </ion-buttons>
    <ion-title>Courses</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content id="courses-content" *ngIf="!this.waiting">
  <ion-item *ngIf="!this.courses || this.courses.length === 0" class="no-course-wrapper">
    <ion-content *ngIf="!this.courses || this.courses.length === 0" class="no-course">
      It doesn't look like there are any courses here.<br/>
      Feel free to <a (click)="presentNewCourseModal()">add a course</a>
    </ion-content>
  </ion-item>

  <ion-content *ngIf="this.courses && this.courses.length > 0" class="course-wrapper">
    <ion-item *ngFor="let course of courses" lines="none">
      <ion-item class="course">
        <p class="code">{{course.code}}</p>
        <p>{{course.details}}</p>
        <ion-icon name="create" slot="end" (click)="presentEditCourseModal(course)"></ion-icon>
        <ion-icon name="trash" slot="end" (click)="deleteCourse(course.code)"></ion-icon>
      </ion-item>
    </ion-item>
  </ion-content>

  <ion-fab vertical="bottom" horizontal="end" slot="fixed" (click)="presentNewCourseModal()">
    <ion-fab-button>
      <ion-icon name="add"></ion-icon>
    </ion-fab-button>
  </ion-fab>
</ion-content>

<ion-content *ngIf="this.waiting">
  <ion-item id="spinner" lines="none">
    <ion-spinner></ion-spinner>
  </ion-item>
</ion-content>
```

The above HTML file shows the use of standard HTML components such as <p> tags as well as the <ion> tags.

Business Layer

The business layer in an Ionic application is represented by the page.ts files. These files are the logic required by the application in processing user inputs, how to send data to the presentation layer, and the relationships between pages.

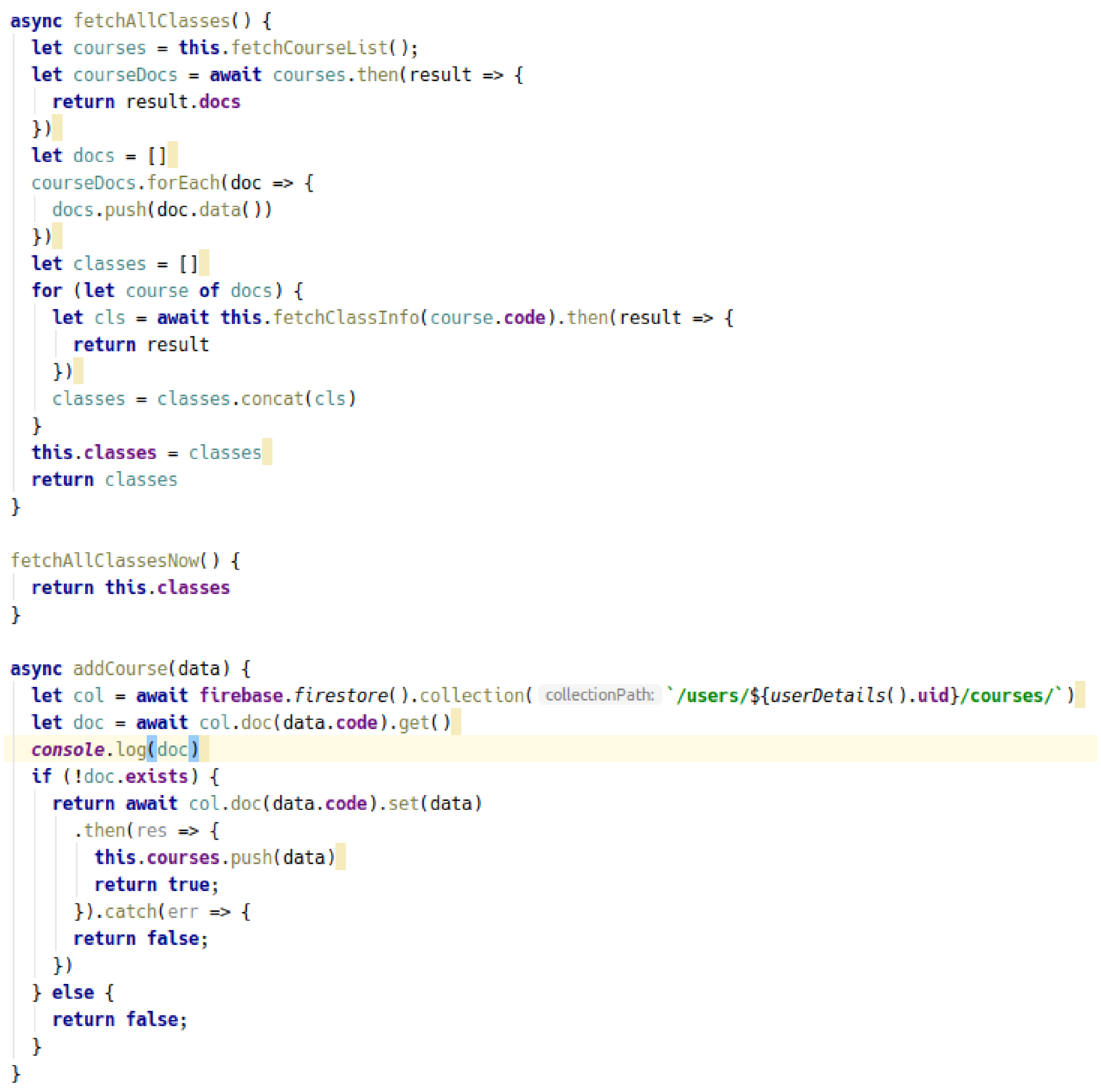


```
async ngOnInit(){
  //fetch locally stored course list
  this.courses = this.db.fetchCourseListNow()
  //if there is no locally stored courses, show the spinner
  if(!this.courses || this.courses.length < 1) {
    this.waiting = true
  }

  //fetch the course list from firebase
  await this.db.fetchCourseList().then(data => {
    let courses = []
    if(!data.empty) {
      data.docs.forEach(course => {
        let courseData = course.data()
        courses.push(courseData)
      })
    }
    this.courses = courses
  })
  this.waiting = false
}

/**
 * Delete the course.
 * @param code the course code for the course to be deleted
 */
async deleteCourse(code) {
  await this.db.deleteCourse(code);
  this.ngOnInit();
}

/**
 * Open the edit course modal.
 * @param course the course to populate the modal with
 */
async presentEditCourseModal(course) {
  //setup information for the modal
  let modal = await this.modalCtrl.create({
    component: AddNewCoursePage,
    componentProps: {
      "header": "Edit Course",
      "new": false,
      "button": "Edit",
      "code": course.code,
```

The above image shows these processes: the fetchCourseList and deleteCourse functions provide the bridge between the presentation layer and the data access layer though this processing, and the presentEditCourseModal provides some navigation between pages.

Data Access Layer

My implementation of the data access layer in my Ionic app was through the use of services. These services were stored separately from the pages in a separate folder and provided an interface between the business layer and the services. The services created were an authentication service, a database service, and a utility service, which provided functionality which was shared by multiple pages in the business layer.



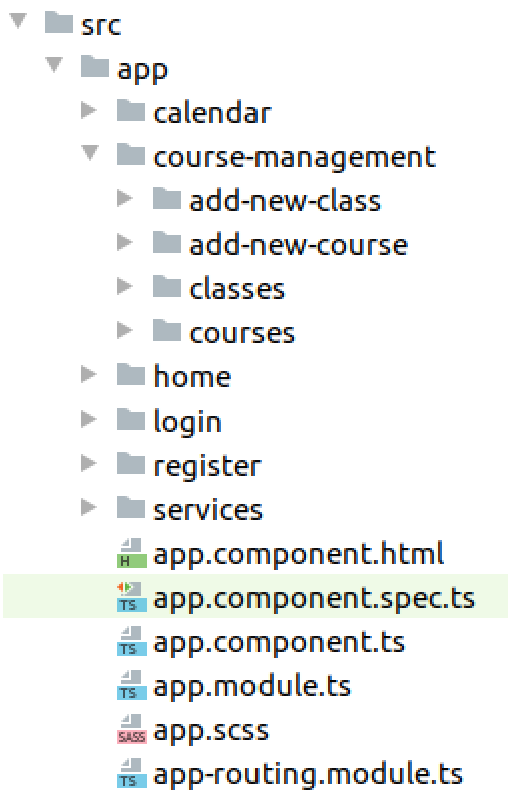
```
async fetchAllClasses() {
  let courses = this.fetchCourseList();
  let courseDocs = await courses.then(result => {
    return result.docs
  })
  let docs = []
  courseDocs.forEach(doc => {
    docs.push(doc.data())
  })
  let classes = []
  for (let course of docs) {
    let cls = await this.fetchClassInfo(course.code).then(result => {
      return result
    })
    classes = classes.concat(cls)
  }
  this.classes = classes
  return classes
}

fetchAllClassesNow() {
  return this.classes
}

async addCourse(data) {
  let col = await firebase.firestore().collection( collectionPath: `/users/${userDetails().uid}/courses/`)
  let doc = await col.doc(data.code).get()
  console.log(doc)
  if (!doc.exists) {
    return await col.doc(data.code).set(data)
      .then(res => {
        this.courses.push(data)
        return true;
      }).catch(err => {
      return false;
    })
  } else {
    return false;
  }
}
```

The above image shows 2 main pieces of functionality provided by the database service. The functions fetchAllClasses and addCourse provide an interface for accessing and manipulating the data stored in the services.

Services

The service agents in the data access layer communicate with firebase in order to provide authentication and user storage. Between these two layers is a local cache. This cache stores data received from firebase and edits the local copies when a user edits the items before sending the changes or fetching from the externally stored firebase. This allows for quicker response times when fetching data, and semi-persistent storage in case the phone's connection is unreliable.

Organisation



```
▼ src
  ▼ app
    ▶ calendar
    ▼ course-management
      ▶ add-new-class
      ▶ add-new-course
      ▶ classes
      ▶ courses
    ▶ home
    ▶ login
    ▶ register
    ▶ services
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
      app.scss
      app-routing.module.ts
```

The organisation of my folders can be seen above. The main pages are each in their own folders, with the pages relating to courses and classes as subfolders within the course-management folder. The services are separate in the services folder, and any overall app design information is stored in the top-level files, as is by default in an Ionic application.

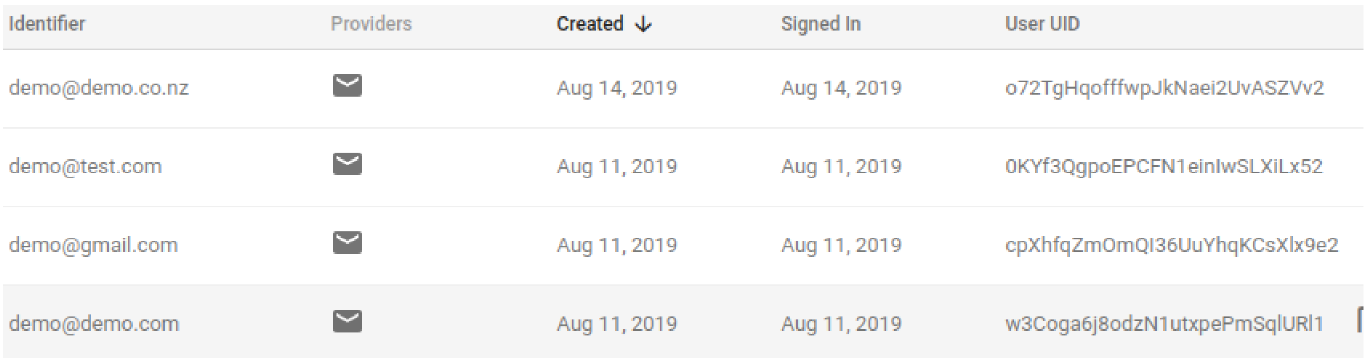
Cross-Cutting

A cross-cutting component I made use of in my application was an authentication guard. This authentication guard provided security by only allowing access to pages that were not the login or register page to logged in users.

Major External Component

The major external component I used for my application was Firebase. I made use of both the authentication and database services of Firebase.

Authentication



Identifier	Providers	Created ↓	Signed In	User UID
demo@demo.co.nz	✉	Aug 14, 2019	Aug 14, 2019	o72TgHqofffwpJkNaei2UvASZVv2
demo@test.com	✉	Aug 11, 2019	Aug 11, 2019	0KYf3QgpoEPCFN1einIwSLXiLx52
demo@gmail.com	✉	Aug 11, 2019	Aug 11, 2019	cpXhfqZmOmQI36UuYhqKCsXlx9e2
demo@demo.com	✉	Aug 11, 2019	Aug 11, 2019	w3Coga6j8odzN1utxpePmSqlURl1

Database