NORTHWEST NAZARENE UNIVERSITY

Database System for the Nampa Family Justice Center

THESIS

Submitted to the Department of Mathematics and Computer Science

in partial fulfillment of the requirements

for the degree of

BACHELOR OF SCIENCE

Matlyn Elizabeth Knott

2018

THESIS

Submitted to the Department of Mathematics and Computer Science

in partial fulfillment of the requirements

for the degree of

BACHELOR OF SCIENCE

By

Matlyn Knott

2018

Database System for the Nampa Family Justice Center

Author: _____Matlyn Knott_____

Matlyn Knott

Approved: __Barry Myers_____

Dr. Barry Myers, Department of Mathematics and Computer Science Faculty Advisor

Approved: _____

Dr. Stephen Riley, Department of Theology
Second Reader

Approved: ___Barry Myers_____

Barry L. Myers, Ph.D., Chair
Department of Mathematics & Computer Science

Abstract

Database System for the Nampa Family Justice Center.

KNOTT, MATLYN (Department of Mathematics and Computer Science), MYERS, DR. BARRY (Department of Mathematics and Computer Science).

The city of Nampa offers help for victims of family violence and sexual assault, through the Family Justice Center. Every year, there are hundreds of people in Canyon County that are petitioning for a no contact order of some kind. They are expected to go to the court room, possibly act as an attorney for themselves, figure the justice system out, and sit in the same room as the respondent, who often happens to be someone who might be physically, mentally, or emotionally abusing them. The Family Justice Center has offered a service to these petitioners that provides an advocate to go with the petitioner. The advocate is there for support, but to also collect data about these hearings. In the past, this data has been collected by pen and paper but the purpose of this project was to create a database system to collect and store this information about the court cases. Throughout this project, I collected information about the system, designed the database system and started the implementation of this database.

Acknowledgments

I would like to thank my parents for being the support system for me these last few years. Specifically, I would like to thank my dad for helping from a technical point of view as I went through this project. I want to thank Dr. Myers and Dr. Hamilton for their confidence in me, even when I did not feel the same about myself. I would like to thank my peers for the hours that they let me try to figure out problems I was running into, out loud, to them. Lastly, I would love to thank my wonderful boyfriend who has been more of an encouragement than I could ever ask for.

Table of Contents

List of Figures

The Nampa Family Justice Center has been serving the community since 2005. They offer a range of services from representation in court hearings, counseling, and even self—defense courses. Over the course of the more than twelve years, much has changed in computer science. The Nampa Family Justice Center has many areas where automation through computer science would be of huge assistance. One of these areas is the advocate program. The advocate program sends volunteers to the courtroom so that those petitioning for a no contact order do not have to show up alone. The petitioners must stand up, act as their own lawyer, and figure out the legal system, all while facing someone who most likely is their abuser. The advocates go and act as a helper in the legal situation. The volunteer advocates know their way around the courthouse, know how to best get the petition granted, and act as emotional support.

The most important job of the advocate, apart from being a guide to the petitioner, is to gather information about the hearing and bring it back to the Nampa Family Justice Center, to report back for grant purposes. The advocates currently collect data with pen and paper, but this is difficult to maintain and keep records. At the end of the year, this data should already be in a database at the Family Justice Center. There is not an application or system already set up for Family Justice Centers advocate program. The current database at the Nampa Family Justice Center is not set up to handle that specific data.

If this database were created to store the information, it would increase the efficiency of the grant writing process. It would make data collection easier. The entire process would be on a phone. Through a series of buttons the database could be polled to see the statistics on particular cases and trends in the data. While this has not yet been created, there is a need for it in this organization.

Preparation

There was a significant learning curve for this project. First, the data collection process had to be learned by shadowing volunteers throughout the summer to see what the actual process of data collection was. Through this research, it was discovered that the easiest way the volunteers could collect data was through an ipad or a tablet. Therefore, it was decided the program needed to work on both iOS and Android. To do this, a web—based application was the best option to achieve a dual platform application. Doing some research on the usability and difficulty, Ionic was considered (Drifty). Others had previously used Ionic on senior projects and it was capable of what was needed, as well as being simple to learn.

Development

The first step to completing this project was to figure out what data needed to be collected, and what data had already been collected. On the first draft of the database there were four tables. As seen in figure 1 below, there was information about the client and judges. This information was not necessary, as the Family Justice Center had previously stored this information somewhere else.

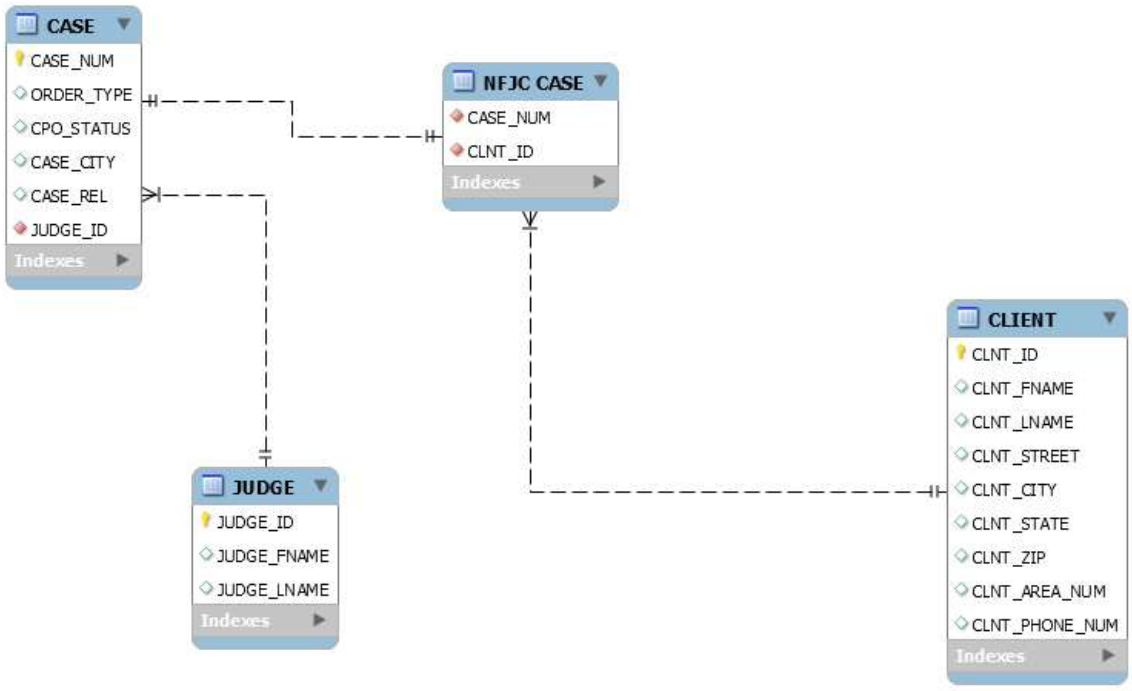

Figure 1 – First ERD

After further research, it was decided the only things that really needed to be in the database were details about each case, including:

- A case number
- Client name
- Results
- Gender of petitioner
- Order type
- An image of the court docket

The ERD came out looking about how figure 2 looks, with a few tweaks.

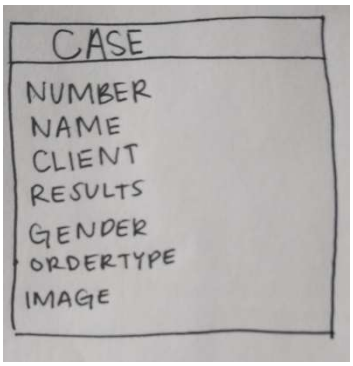

Figure 2 – Final ERD

After the data was organized into what was needed, a GUI for the database system was designed. The application needed to be cross platform. Therefore a webbased program, Ionic, was chosen. Ionic is fairly new and the documentation is easy to find and simple to understand. The documentation Ionic does provide is more than enough to get any developer, no matter the skill level, on the right track. A few of the Ionic apps that were already created as a baseline were used to test most of the code.

Ionic provides installation documentation and information for starting a project. Once node and npm were installed through NodeJS installer, the Ionic and Cordova CLI had to be installed through a simple command line prompt found in the documentation.

A login page was the next thing to implement. The login page is the first thing users need to see upon opening the app. Unfortunately, the login page was never functional with usernames and passwords in the database, but it was able to take in user names and passwords and output the next page. As seen in figure 3, the login page was simple and to the point.

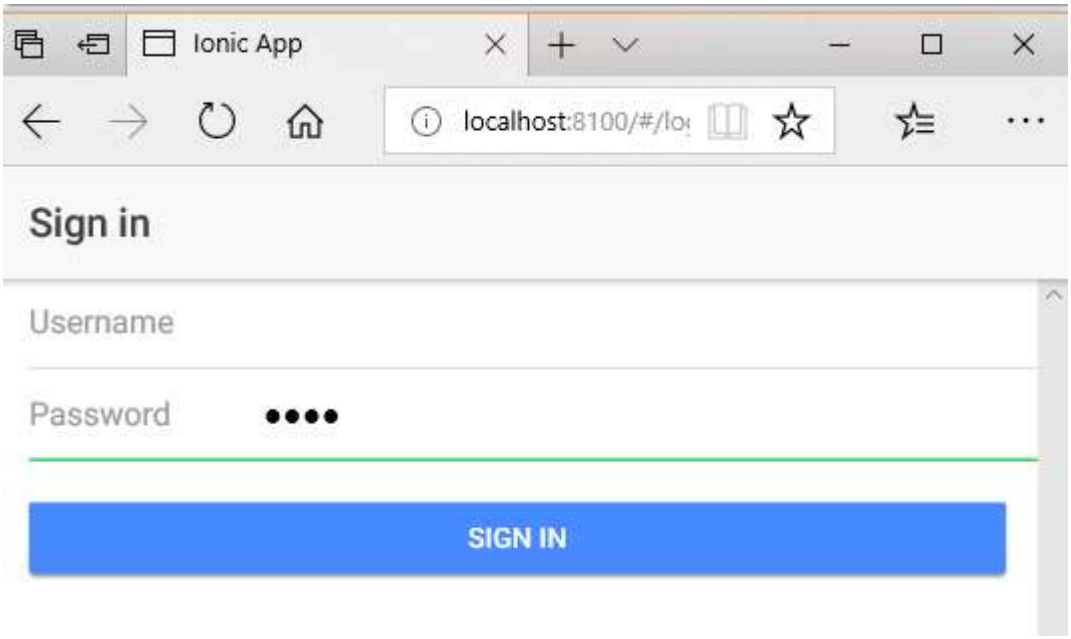

Figure 3 – Log in Page

Next pages were created for display of the cases, for the user search of a specific case, and for reports to be generated. Of these pages, the search bar ended up working properly. The cases page consisted of a list of the current cases, all of which could be expanded to see and edit the details of the case. A plus sign in the top right corner allows the user to add a new case into the list. The reports page did not have any information yet. However this was because the database was not set up and the reports needed to gather information from the database.

Throughout development, it was important to consider not only what the advocates needed from this application, but also what information the grants needed. One particular change that occurred from the beginning of the project was how the image of the docket was stored in the database. When first looking at this information it seemed beneficial to have all of the court case's data in the database. However, not every piece of data was going to be used for the grants. Since it was not all going to be used, collecting it all a second time was a waste of memory. Each court case has what is called a "docket", the document containing all the

information about the hearing, from the judge's name to the city, and everything else publicly available to the Family Justice Center. Most of the information on the case does not need to be entered into the database, but it would be nice to have on file associated with the case. It was decided the best solution was to have the option to either take a picture of the docket with a camera on the ipad/tablet, or to let the user scan in the docket and upload it later. The image of the docket then can be disposed of and the information will stay in the database, associated with the case number.

Something that was very important, was that the application be scalable. The result was that the inputs and displays were fit to screens so that they would grow or shrink depending on the size of the display.

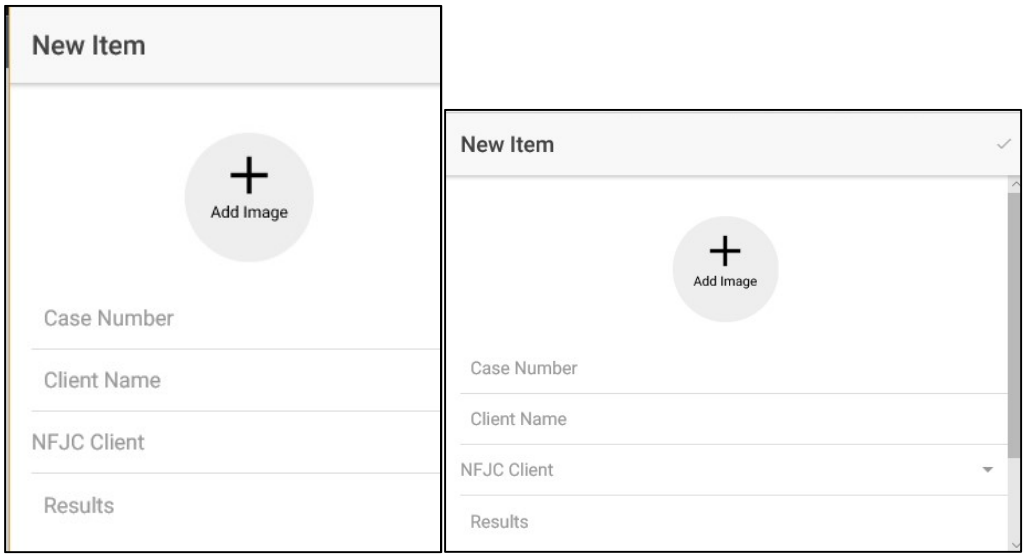

Figure 4 – Scalability

The Ionic programming actually was something that did not pose as many problems as anticipated, but ended up being a useful tool. Ionic's documentation and framework guides helps to easily see what it takes for a piece to be implemented. Their examples give enough help to get even an inexperienced developer somewhere to start.

Future Work

Though most of the project was completed, there was still some implementation that needed to be finished. The first objective to be completedvis that the database and the application needed to be implemented together so that the application can collect data and send it to the database upon submit. Because the database itself is not too large it should not take long to implement. The hard part of the future work is that someone will have to figure out the safest way to implement the database, where to implement it, and how to encrypt and lock this information somewhere. The information being used is sensitive and there will be a lot of work involved in the security aspect of this project. For this system to be functioning, there needs to be encryption, secure log ins, and other safety features to ensure that the data is not compromised.

Another feature needing work is the “submit updates” button. When looking at the information in the court case you can press a button labeled “submit updates” and it will commit any changes that you have made to the information in the court case. Right now, the information can be changed but there is no way to commit it. Figure 5 shows the button that needed to be fixed.

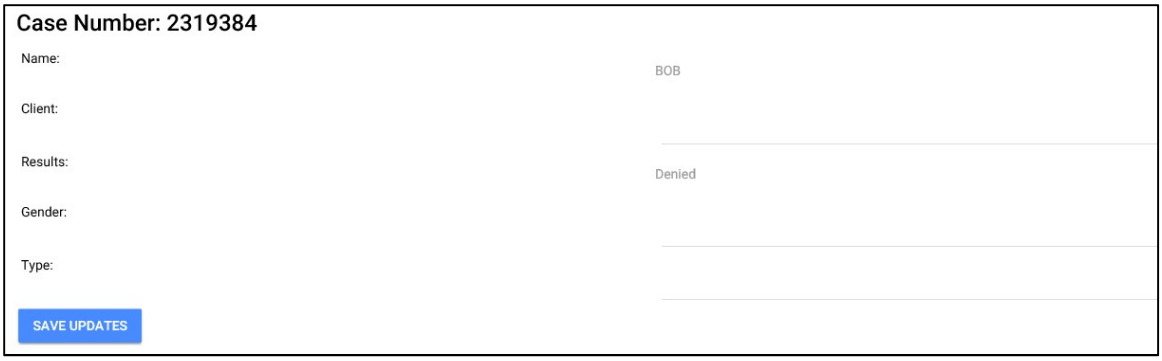

Figure 5 – Save Updates

The final features that should be fixed in future work is the Reports Page. While there is nothing to generate right now, in the future there will be a need for a few different reports to be generated. The Nampa Family Justice Center has asked that there be a report generated for the entire dataset, listed in order by whatever the user asks for (i.e. if the user wants a list generated ordered by client name), the city and the CPO type & outcome, separate groups of NFJC clients and non—clients, and potentially reports based on gender or the presiding judge.

As far as future work goes, there is much more that needs to be done for the Nampa Family Justice Center. While this database might improve some aspects of the organization, more of their systems need to be upgraded or worked on to fit their needs better. There is more to be done with the automation of the center and it is more than a one—person job. However, the Nampa Family Justice Center is solely grant and donation based. This means, for those systems to be improved there needs to be a grant of some sort. Because of the monetary limitations, this change will most likely not come anytime soon.

Challenges

I faced many challenges and learned important lessons. The main thing I learned was how to communicate with clients better. The biggest problem occurred half way through my implementation process. At that time, the entire project was scratched and I was given a new task. It was hard to hear, but I know that I learned through that situation how to ask better questions the next time around.

The main problem that I ran into was knowing whether the project was being created for the program or the grant research. When I went into the courtroom and was gathering information on the program, I was under the impression that the application was supposed to best fit those volunteers. However, as I came to find out, the main purpose of the data was not to be used by those volunteers, but was to be used for grant purposes. If I would have walked into the project knowing what questions to ask, I could have better prepared myself for creating an application that best suit their needs.

Conclusion

I knew when I picked a project I needed it to be something I was passionate and excited about. I had a few options going into that decision—making process, but as soon as this opportunity arose, I knew this was the project that I had been waiting for. The work I was doing was going to directly affect the efficiency of the Nampa Family Justice Center. I knew while it might be a challenge, it was something I wanted to be involved in.

That being said, this project did include many challenges and was frustrating at times. Nevertheless, I learned more about computers, systems analysis, and my abilities through this project than I would have anywhere else. I feel prepared for the real world because I took this project on. While I would have loved to see this project until completion, I know that the work I put in will be a great starting point for another programmer to come in and finish this project. The Nampa Family Justice Center will greatly benefit from this project and the work that I contributed.

References

Drifty. “Ionic Documentation.” *Ionic Framework*, ionicframework.com/docs/.

Murray, Meg Coffin. "Database Security: What Students Need to Know." *Journal of Information Technology Education*, vol. 9, Jan. 2010, pp. IIP61-IIP77. EBSCO*host*, nnu.idm.oclc.org/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=60636410&site=ehost-live&scope=site.

No Author. “Advantages of Stored Procedures”. N.p., 1999. Web. 20 Apr. 2017. <https://docs.oracle.com/cd/F49540_01/DOC/java.815/a64686/01_intr3.htm>.

No Author "Advantages and Drawbacks of Using Stored Procedures for Processing Data." *Segue Technologies*. N.p., 08 Nov. 2016. Web. 20 Apr. 2017. <http://www.seguetech.com/advantages-and-drawbacks-of-using-stored-procedures-for-processing-data/>.

Ozawa, Joseph Paul. "Transformative Justice: Psychological Services in the Criminal, Family, and Juvenile Justice Centres of the Subordinate Courts of Singapore." *Applied Psychology: An International Review*, vol. 51, no. 2, Apr. 2002, p. 218. EBSCO*host*, nnu.idm.oclc.org/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=6384152&site=ehost-live&scope=site.

Perry, Jennifer. "NFJC Meeting." Personal interview. 11 Apr. 2017.

Appendix A – Source Code

Item-Create

HTML

```
<ion-header>

  <ion-navbar>
    <ion-title>{{ 'ITEM_CREATE_TITLE' | translate }}</ion-title>
    <ion-buttons start>
      <button ion-button (click)="cancel()">
        <span color="primary" showWhen="ios">
          {{ 'CANCEL_BUTTON' | translate }}
        </span>
        <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
      </button>
    </ion-buttons>
    <ion-buttons end>
      <button ion-button (click)="done()" [disabled]="!isReadyToSave" strong>
        <span color="primary" showWhen="ios">
          {{ 'DONE_BUTTON' | translate }}
        </span>
        <ion-icon name="md-checkmark" showWhen="core,android,windows"></ion-icon>
      </button>
    </ion-buttons>
  </ion-navbar>

</ion-header>
```

```
<ion-content>
  <form *ngIf="form" [formGroup]="form" (ngSubmit)="createItem()">
    <input type="file" #fileInput style="visibility: hidden; height: 0px" name="files[]" (change)="processWebImage($event)" />
    <div class="profile-image-wrapper" (click)="getPicture()">
      <div class="profile-image-placeholder" *ngIf="!this.form.controls.profilePic.value">
        <ion-icon name="add"></ion-icon>
        <div>
          {{ 'ITEM_CREATE_CHOOSE_IMAGE' | translate }}
        </div>
      </div>
      <div class="profile-image" [style.backgroundImage]="getProfileImageStyle()" *ngIf="this.form.controls.profilePic.value"></div>
    </div>
    <ion-list>
      <ion-item>
        <ion-input type="text" placeholder="Case Number" formControlName="number"></ion-input>
      </ion-item>
      <ion-item>
        <ion-input type="text" placeholder="Client Name" formControlName="name"></ion-input>
      </ion-item>
      <ion-item>
        <ion-label>NFJC Client</ion-label>
        <ion-select formControlName="client">
          <ion-option value="Yes">Yes</ion-option>
          <ion-option value="No">No</ion-option>
        </ion-select>
      </ion-item>
      <ion-item>
        <ion-input type="text" placeholder="Results" formControlName="results"></ion-input>
      </ion-item>
      <ion-item>
        <ion-label>Gender</ion-label>
        <ion-select formControlName="gender">
          <ion-option value="Female">Female</ion-option>
          <ion-option value="Male">Male</ion-option>
        </ion-select>
      </ion-item>
      <ion-item>
        <ion-label>Type of Order</ion-label>
        <ion-select formControlName="type">
          <ion-option value="domesticViolence">Domestic Violence</ion-option>
          <ion-option value="stalking">Stalking</ion-option>
          <ion-option value="both">Both</ion-option>
        </ion-select>
      </ion-item>
    </ion-list>
  </form>
</ion-content>
```

Module TS

```
import { NgModule } from '@angular/core';
import { TranslateModule } from '@ngx-translate/core';
import { IonicPageModule } from 'ionic-angular';

import { ItemCreatePage } from './item-create';

@NgModule({
  declarations: [
    ItemCreatePage,
  ],
  imports: [
    IonicPageModule.forChild(ItemCreatePage),
    TranslateModule.forChild()
  ],
  exports: [
    ItemCreatePage
  ]
})
export class ItemCreatePageModule { }
```

SCSS Style Sheet

```scss
page-item-create {
  .profile-image-wrapper {
    text-align: center;
    margin: 20px 0;

    .profile-image {
      width: 96px;
      height: 96px;
      border-radius: 50%;

      display: inline-block;

      background-repeat: no-repeat;
      background-size: cover;
      background-position: center;
    }

    .profile-image-placeholder {
      display: inline-block;

      background-color: #eee;
      width: 96px;
      height: 96px;
      border-radius: 50%;

      font-size: 12px;

      ion-icon {
        font-size: 44px;
        margin-bottom: -10px;
        margin-top: 10px;
      }
    }
  }
}
```

TS

```
import { Component, ViewChild } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Camera } from '@ionic-native/camera';
import { IonicPage, NavController, ViewController } from 'ionic-angular';

@IonicPage()
@Component({
  selector: 'page-item-create',
  templateUrl: 'item-create.html'
})
export class ItemCreatePage {
  @ViewChild('fileInput') fileInput;

  isReadyToSave: boolean;

  item: any;

  form: FormGroup;

  constructor(public navCtrl: NavController, public viewCtrl: ViewController, formBuilder: FormBuilder, public camera: Camera) {
    this.form = formBuilder.group({
        profilePic: [''],
        number: [''],
        name: ['', Validators.required],
        client: ['',],
        results: [''],
        gender: [''],
        type: ['']
    });

    // Watch the form for changes, and
    this.form.valueChanges.subscribe((v) => {
      this.isReadyToSave = this.form.valid;
    });
  }

  ionViewDidLoad() {

  }

  getPicture() {
    if (Camera['installed']()) {
      this.camera.getPicture({
        destinationType: this.camera.DestinationType.DATA_URL,
        targetWidth: 96,
        targetHeight: 96
      }).then((data) => {
        this.form.patchValue({ 'profilePic': '' + data });
      }, (err) => {
        alert('Unable to take photo');
      })
    } else {
      this.fileInput.nativeElement.click();
    }
  }

  processWebImage(event) {
    let reader = new FileReader();
    reader.onload = (readerEvent) => {
```

```
      let imageData = (readerEvent.target as any).result;
      this.form.patchValue({ 'profilePic': imageData });
    };

    reader.readAsDataURL(event.target.files[0]);
  }

  getProfileImageStyle() {
    return 'url(' + this.form.controls['profilePic'].value + ')'
  }

  /**
   * The user cancelled, so we dismiss without sending data back.
   */
  cancel() {
    this.viewCtrl.dismiss();
  }

  /**
   * The user is done and wants to create the item, so return it
   * back to the presenter.
   */
  done() {
    if (!this.form.valid) { return; }
    this.viewCtrl.dismiss(this.form.value);
  }
}
```

Item Detail

HTML

```
<ion-header>

  <ion-navbar>
    <ion-title>{{ item.number }}</ion-title>
  </ion-navbar>

</ion-header>


<ion-content>
  <div class="item-profile" text-center #profilePic [style.background-image]="'url(' + item.profilePic + ')'">
  </div>
  <div class="item-detail" padding>
    <h2>Case Number: {{item.number}}</h2>
    <div class="row">
      <div class="col">Name: </div>
      <div class="col"><ion-input type="text" placeholder={{item.name}} id="nameHolder"></ion-input></div>
    </div>
    <div class="row">
      <div class="col">Client: </div>
      <div class="col">
        <ion-item>
          <ion-label>{{item.client}}</ion-label>
          <ion-select>
            <ion-option value="Yes">Yes</ion-option>
            <ion-option value="No">No</ion-option>
          </ion-select>
        </ion-item>
      </div>
    </div>
    <div class="row">
      <div class="col">Results: </div>
      <div class="col"><ion-input type="text" placeholder={{item.results}}></ion-input></div>
    </div>
    <div class="row">
      <div class="col">Gender: </div>
      <div class="col">
        <ion-item>
          <ion-label>{{item.gender}}</ion-label>
          <ion-select>
            <ion-option value="Female">Female</ion-option>
            <ion-option value="Male">Male</ion-option>
          </ion-select>
        </ion-item>
      </div>
    </div>
    <div class="row">
      <div class="col">Type: </div>
      <div class="col">
        <ion-item>
          <ion-label>{{item.type}}</ion-label>
          <ion-select>
            <ion-option value="domesticViolence">Domestic Violence</ion-option>
            <ion-option value="stalking">Stalking</ion-option>
            <ion-option value="both">Both</ion-option>
          </ion-select>
        </ion-item>
      </div>
    </div>
    <button ion-button (click)="document.write(hello)">Save Updates</button>
  </div>
</ion-content>
```

Module TS

```
import { NgModule } from '@angular/core';
import { TranslateModule } from '@ngx-translate/core';
import { IonicPageModule } from 'ionic-angular';

import { ItemDetailPage } from './item-detail';

@NgModule({
  declarations: [
    ItemDetailPage,
  ],
  imports: [
    IonicPageModule.forChild(ItemDetailPage),
    TranslateModule.forChild()
  ],
  exports: [
    ItemDetailPage
  ]
})
export class ItemDetailPageModule { }
```

SCSS Style Sheet

```scss
page-item-detail {
  .item-profile {
    width: 100%;
    background-position: center center;
    background-size: cover;
    height: 250px;
  }

  .item-detail {
    width: 100%;
    background: white;
    position: absolute;
  }
}
```

TS

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams} from 'ionic-angular';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { Settings } from '../../providers/providers';
import { Item } from '../../models/item';
import { Items } from '../../providers/providers';

@IonicPage()
@Component({
  selector: 'page-item-detail',
  templateUrl: 'item-detail.html'
})
export class ItemDetailPage {
  item: any;

  constructor(public navCtrl: NavController, navParams: NavParams,public items: Items) {
    this.item = navParams.get('item') || items.defaultItem;
  }

  save(){
     if (document.getElementById("nameUpdated")!=null)
     {
         document.write("Hello");
     }
  }
}
```

List Master

HTML

```
<ion-header>

  <ion-navbar>
    <ion-title>{{ 'LIST_MASTER_TITLE' | translate }}</ion-title>

    <ion-buttons end>
      <button ion-button icon-only (click)="addItem()">
        <ion-icon name="add"></ion-icon>
      </button>
    </ion-buttons>
  </ion-navbar>

</ion-header>

<ion-content>

  <ion-list>
    <ion-item-sliding *ngFor="let item of currentItems">
      <button ion-item (click)="openItem(item)">
        <ion-avatar item-start>
          <img [src]="item.profilePic" />
        </ion-avatar>
        <h2>{{item.name}}</h2>
        <p>{{item.about}}</p>
        <ion-note item-end *ngIf="item.note">{{item.note}}</ion-note>
      </button>

      <ion-item-options>
        <button ion-button color="danger" (click)="deleteItem(item)">
          {{ 'DELETE_BUTTON' | translate }}
        </button>
      </ion-item-options>
    </ion-item-sliding>
  </ion-list>
</ion-content>
```

Module TS

```
import { NgModule } from '@angular/core';
import { TranslateModule } from '@ngx-translate/core';
import { IonicPageModule } from 'ionic-angular';

import { ListMasterPage } from './list-master';

@NgModule({
  declarations: [
    ListMasterPage,
  ],
  imports: [
    IonicPageModule.forChild(ListMasterPage),
    TranslateModule.forChild()
  ],
  exports: [
    ListMasterPage
  ]
})
export class ListMasterPageModule { }
```

TS

```
import { Component } from '@angular/core';
import { IonicPage, ModalController, NavController } from 'ionic-angular';

import { Item } from '../../models/item';
import { Items } from '../../providers/providers';

@IonicPage()
@Component({
  selector: 'page-list-master',
  templateUrl: 'list-master.html'
})
export class ListMasterPage {
  currentItems: Item[];
  constructor(public navCtrl: NavController, public items: Items, public modalCtrl: ModalController) {
    this.currentItems = this.items.query();
  }

  /**
   * The view loaded, let's query our items for the list
   */
  ionViewDidLoad() {
  }

  /**
   * Prompt the user to add a new item. This shows our ItemCreatePage in a
   * modal and then adds the new item to our data source if the user created one.
   */
  addItem() {
    let addModal = this.modalCtrl.create('ItemCreatePage');
    addModal.onDidDismiss(item => {
      if (item) {
        this.items.add(item);
      }
    })
    addModal.present();
  }

  /**
   * Delete an item from the list of items.
   */
  deleteItem(item) {
    this.items.delete(item);
  }

  /**
   * Navigate to the detail page for this item.
   */
  openItem(item: Item) {
    this.navCtrl.push('ItemDetailPage', {
      item: item
    });
  }
}
```

Login

HTML

```
<ion-header>

  <ion-navbar>
    <ion-title>{{ 'LOGIN_TITLE' | translate }}</ion-title>
  </ion-navbar>

</ion-header>


<ion-content>
  <form (submit)="doLogin()">
    <ion-list>

      <ion-item>
        <ion-label fixed>{{ 'USERNAME' | translate }}</ion-label>
        <ion-input type="text" [(ngModel)]="account.username" name="username"></ion-input>
      </ion-item>


      <ion-item>
        <ion-label fixed>{{ 'PASSWORD' | translate }}</ion-label>
        <ion-input type="password" [(ngModel)]="account.password" name="password"></ion-input>
      </ion-item>

      <div padding>
        <button ion-button color="primary" block>{{ 'LOGIN_BUTTON' | translate }}</button>
      </div>

    </ion-list>
  </form>
</ion-content>
```

Module TS

```
import { NgModule } from '@angular/core';
import { TranslateModule } from '@ngx-translate/core';
import { IonicPageModule } from 'ionic-angular';

import { LoginPage } from './login';

@NgModule({
  declarations: [
    LoginPage,
  ],
  imports: [
    IonicPageModule.forChild(LoginPage),
    TranslateModule.forChild()
  ],
  exports: [
    LoginPage
  ]
})
export class LoginPageModule { }
```

```
  // Attempt to login in through our User service
  doLogin() {
    this.user.login(this.account).subscribe((resp) => {
      this.navCtrl.push(MainPage);
    }, (err) => {
      this.navCtrl.push(MainPage);
      // Unable to log in
      let toast = this.toastCtrl.create({
        message: this.loginErrorString,
        duration: 3000,
        position: 'top'
      });
      toast.present();
    });
  }
}
```

TS

```
import { Component } from '@angular/core';
import { TranslateService } from '@ngx-translate/core';
import { IonicPage, NavController, ToastController } from 'ionic-angular';

import { User } from '../../providers/providers';
import { MainPage } from '../pages';

@IonicPage()
@Component({
  selector: 'page-login',
  templateUrl: 'login.html'
})
export class LoginPage {
    //Username is going to be their email
  account: { email: string, password: string } = {
    email: 'test@example.com',
    password: 'test'
  };

  // Our translated text strings
  private loginErrorString: string;

  constructor(public navCtrl: NavController,
    public user: User,
    public toastCtrl: ToastController,
    public translateService: TranslateService) {

    this.translateService.get('LOGIN_ERROR').subscribe((value) => {
      this.loginErrorString = value;
    })
```

Menu

HTML

```
<ion-menu [content]="content">
  <ion-content>
    <ion-list>
      <button menuClose ion-item *ngFor="let p of pages" (click)="openPage(p)">
        {{p.title}}
      </button>
    </ion-list>
  </ion-content>
</ion-menu>

<ion-nav #content [root]="rootPage"></ion-nav>
```

Module TS

```
import { NgModule } from '@angular/core';
import { TranslateModule } from '@ngx-translate/core';
import { IonicPageModule } from 'ionic-angular';

import { MenuPage } from './menu';

@NgModule({
  declarations: [
    MenuPage,
  ],
  imports: [
    IonicPageModule.forChild(MenuPage),
    TranslateModule.forChild()
  ],
  exports: [
    MenuPage
  ]
})
export class MenuPageModule { }
```

TS

```
import { Component, ViewChild } from '@angular/core';
import { IonicPage, Nav, NavController } from 'ionic-angular';

@IonicPage()
@Component({
  selector: 'page-menu',
  templateUrl: 'menu.html'
})
export class MenuPage {
  // A reference to the ion-nav in our component
  @ViewChild(Nav) nav: Nav;

  rootPage: any = 'ContentPage';

  pages: Array<{ title: string, component: any }>;

  constructor(public navCtrl: NavController) {
    // used for an example of ngFor and navigation
    this.pages = [
      { title: 'Sign in', component: 'LoginPage' },
      { title: 'Signup', component: 'SignupPage' }
    ];
  }

  ionViewDidLoad() {
    console.log('Hello MenuPage Page');
  }

  openPage(page) {
    // Reset the content nav to have just this page
    // we wouldn't want the back button to show in this scenario
    this.nav.setRoot(page.component);
  }
}
```

Pages TS

```
// The page the user lands on after opening the app and without a session
export const FirstRunPage = 'LoginPage';

// The main page the user will see as they use the app over a long period of time.
// Change this if not using tabs
export const MainPage = 'TabsPage';

// The initial root pages for our tabs (remove if not using tabs)
export const Tab1Root = 'ListMasterPage';
export const Tab2Root = 'SearchPage';
export const Tab3Root = 'SettingsPage';
```

Search

HTML

```
<ion-header>

  <ion-navbar>
    <ion-title>{{ 'SEARCH_TITLE' | translate }}</ion-title>
  </ion-navbar>

</ion-header>


<ion-content>
  <ion-searchbar (ionInput)="getItems($event)" placeholder="{{ 'SEARCH_PLACEHOLDER' | translate }}"></ion-searchbar>
  <ion-list>
    <button ion-item (click)="openItem(item)" *ngFor="let item of currentItems">
      <ion-avatar item-start>
        <img [src]="item.profilePic" />
      </ion-avatar>
      <h2>{{item.name}}</h2>
      <p>{{item.about}}</p>
      <ion-note item-end *ngIf="item.note">{{item.note}}</ion-note>
    </button>
  </ion-list>
</ion-content>
```

Module TS

```
import { NgModule } from '@angular/core';
import { TranslateModule } from '@ngx-translate/core';
import { IonicPageModule } from 'ionic-angular';

import { SearchPage } from './search';

@NgModule({
  declarations: [
    SearchPage,
  ],
  imports: [
    IonicPageModule.forChild(SearchPage),
    TranslateModule.forChild()
  ],
  exports: [
    SearchPage
  ]
})
export class SearchPageModule { }
```

TS

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';

import { Item } from '../../models/item';
import { Items } from '../../providers/providers';

@IonicPage()
@Component({
  selector: 'page-search',
  templateUrl: 'search.html'
})
export class SearchPage {

  currentItems: any = [];

  constructor(public navCtrl: NavController, public navParams: NavParams, public items: Items) { }

  /**
   * Perform a service for the proper items.
   */
  getItems(ev) {
    let val = ev.target.value;
    if (!val || !val.trim()) {
      this.currentItems = [];
      return;
    }
    this.currentItems = this.items.query({
      name: val
    });
  }
```

```
  /**
   * Navigate to the detail page for this item.
   */
  openItem(item: Item) {
    this.navCtrl.push('ItemDetailPage', {
      item: item
    });
  }

}
```

Tabs

HTML

```
<ion-tabs>
  <ion-tab [root]="tab1Root" tabTitle="Cases"></ion-tab>
  <ion-tab [root]="tab2Root" tabTitle="Search"></ion-tab>
  <ion-tab [root]="tab3Root" tabTitle="Reports"></ion-tab>
</ion-tabs>
```

Module TS

```
import { NgModule } from '@angular/core';
import { TranslateModule } from '@ngx-translate/core';
import { IonicPageModule } from 'ionic-angular';

import { TabsPage } from './tabs';

@NgModule({
  declarations: [
    TabsPage,
  ],
  imports: [
    IonicPageModule.forChild(TabsPage),
    TranslateModule.forChild()
  ],
  exports: [
    TabsPage
  ]
})
export class TabsPageModule { }
```

TS

```
import { Component } from '@angular/core';
import { TranslateService } from '@ngx-translate/core';
import { IonicPage, NavController } from 'ionic-angular';

import { Tab1Root } from '../pages';
import { Tab2Root } from '../pages';
import { Tab3Root } from '../pages';

@IonicPage()
@Component({
  selector: 'page-tabs',
  templateUrl: 'tabs.html'
})
export class TabsPage {
  tab1Root: any = Tab1Root;
  tab2Root: any = Tab2Root;
  tab3Root: any = Tab3Root;

  tab1Title = " ";
  tab2Title = " ";
  tab3Title = " ";

  constructor(public navCtrl: NavController, public translateService: TranslateService) {
    translateService.get(['TAB1_TITLE', 'TAB2_TITLE', 'TAB3_TITLE']).subscribe(values => {
      this.tab1Title = values['TAB1_TITLE'];
      this.tab2Title = values['TAB2_TITLE'];
      this.tab3Title = values['TAB3_TITLE'];
    });
  }
}
```