Team 6: Evan Yang, Eura Shin, Matthew Phelan, Trent Woods
CS 499-001 Fall 2019
October 18th, 2019

Word Count:

Evan: 282
Eura: 561
Matthew: 652
Trent: 884

1. High Level Design

Our main goal for the semester is to refine the integration of the backend and frontend for the mobile application. Figure 1 shows the existing architecture in white and our proposed additions in green. The user will enter information to a form which is stored locally on an encrypted SQLite database. This applies to the Android and iOS versions of the app. When connected to the internet, the locally stored information is sent to the cloud and stored in a database hosted on a UK server.

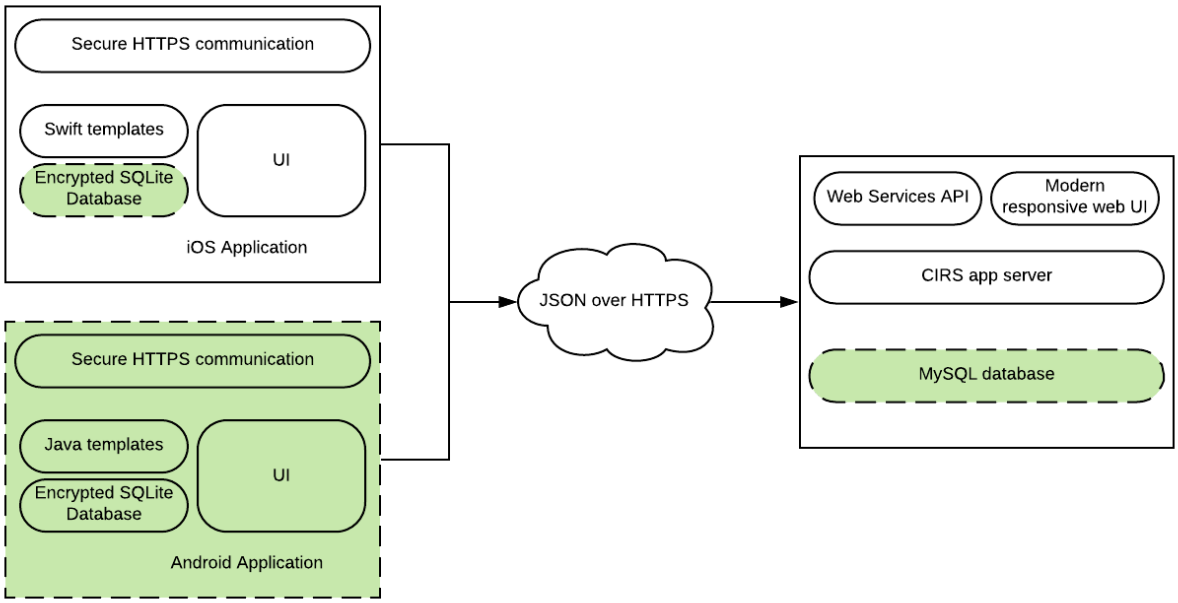

Figure 1. Architecture Diagram. The features our group intends to add are shaded and dotted, while the existing framework is solid.

2. Detailed Design

2.1. PDL for PHP Methods

Authenticate.php:
Authenticates a user at login

```
CONNECT to database
IF connection failed
    THEN connection error

READ POST username, password

QUERY username, server_password
IF !username
    THEN “no username found” error
IF password != server_password
    THEN “incorrect password” error
```

register.php
Registers a new user to the database

```
CONNECT to database
IF connection failed
    THEN connection error

READ POST username, password, firstname, lastname

QUERY username
IF matching username exists
    THEN “username exists” error

INSERT INTO database
    Investigator(username, password, firstname, lastname)
```

retrieve.php
Retrieves database information for specified section of an open case

```
CONNECT to database
IF connection failed
    THEN connection error

READ POST casenum
```

```
a = QUERY SELECT FROM section WHERE casenum
DO FOR each row in a:
    ENCODE row to application/json
END
PRINT application/json data
```

start_death_scene_investigation.php
Starts a new death scene investigation form

```
CONNECT to database
IF connection failed
    THEN connection error

READ POST section, casenum

QUERY casenum
IF matching casenum exists
    THEN "case number exists" error

INSERT INTO database NEW DeathSceneInvestigation(casenum)
IF successful
    THEN INSERT INTO database NEW NextOfKin(casenum)
    INSERT INTO database NEW IncidentInformation(casenum)
    INSERT INTO database NEW Investigation(casenum)
    INSERT INTO database NEW NarrativeComments(casenum)
    INSERT INTO database NEW PillsOnScene(casenum)
```

edit_"section".php
There is an "edit" function for each form section (i.e. edit_case_history, edit_investigation…). Each one performs essentially the same function but for different entities.

```
CONNECT to database
IF connection failed
    THEN connection error

READ POST casenum, attribute_data

QUERY ALL sections == casenum
count = MAX section versions
IF count > 5
    THEN DELETE section version == 1
    DO FOR all sections version 2…5:
        version++
```

```
INSERT new section with version = count + 1
IF matching casenum exists
    THEN "case number exists" error
```

2.2 PDL for Swift Files

SignInViewController.swift

- viewDidLoad()
    - Any additional setup needed after loading initial view
- SubmitButton()
    - Ensure both fields are filled in before checking credentials. If credentials are correct, go to the home page. If not, let the user know and try again.

Investigation.swift

- pickerView()
    - Adds functionality for the fields that require a picker view rather than a text field or toggle button. Set the options for the values that can be picked.
- Save_1(), Save_2(), … , Save7()
    - Save the values of each field to assigned variables for the initial investigation page. Go to the homepage upon completion.
- viewDidLoad()
    - Any additional setup needed before loading initial page. Assign the general layout for each of the pages of the investigation. Assign the values for the options for pickerView fields. Assign the values of each entry to a variable that can be accessed later.

DecedentInformation.swift

- pickerView()
    - Adds functionality for the fields that require a picker view rather than a text field or toggle button. Set the options for the values that can be picked.
- Save_1(), Save_2(), Save_3()
    - Save the values of each field to assigned variables for the decedent information page. Go to the homepage upon completion.
- viewDidLoad()
    - Any additional setup needed before loading decedent information page. Assign the general layout for each of the pages of the investigation. Assign the values for the options for pickerView fields. Assign the values of each entry to a variable that can be accessed later.

IncidentInformation.swift

- pickerView()
    - Adds functionality for the fields that require a picker view rather than a text field or toggle button. Set the options for the values that can be picked.
- Save_1(), Save_2(), Save_3()

    - Save the values of each field to assigned variables for the incident report. Go to the homepage upon completion.
- viewDidLoad()
    - Any additional setup needed before loading incident information. Assign the general layout for each of the pages of the investigation. Assign the values for the options for pickerView fields. Assign the values of each entry to a variable that can be accessed later.

NewFormViewController.swift

- viewDidLoad()
    - Assign the layout for the new form screen.

PendingFormViewController.swift

- PendingButtion()
    - Set the value of all variables to pending status.
- viewDidLoad()
    - Assign the layout for the pending view screen.

CaseHistory.swift

- Save_1(), Save_2(), … , Save7()
    - Save the values of each field to assigned variables for the initial investigation page. Go to the homepage upon completion.
- viewDidLoad()
    - Any additional setup needed before loading case history view. Assign the general layout for each of the pages of the case history. Assign the values for the options for pickerView fields. Assign the values of each entry to a variable that can be accessed later. If the variable is a toggle button, switch the button to correspond to the variable value (yes or no).

PublishedViewController.swift

- viewDidLoad()
    - Assign the layout for the published forms screen.

NextofKin.swift

- Save()
    - Save the values of each field to assigned variables for the next of kin page. Go to the homepage upon completion.
- viewDidLoad()
    - Any additional setup needed before loading the next of kin page. Assign the general layout for the next of kin page. Assign the values of each entry to a variable that can be accessed later. If the variable is a toggle button, switch the button to correspond to the variable value (yes or no).

NarrativeComments.swift

- UploadData()

    - Connect to the backend database hosted on a UK server. Use a post request for enhanced security.
    - Implement a secret password that must match upon accessing of the database. This prevents unauthorized and unwanted connection to our application.
    - Convert each of the text fields to a string to use with the php files for database processing. For the toggle fields, set a boolean value.
    - Convert this string with all of the variable values to .utf8 format in order to transfer and implement in the database.
    - Initiate the upload to the server. Alert the user if there was an issue (loss in connection with the server, for example). Use a return string that should equal "1" if the upload was complete. If so, alert the user the upload was successfully completed.
    - Set all the variables to a published value. Return to homepage upon completion.
- viewDidLoad()
    - Any additional setup needed for the final page.

2.3 User Interface Design

The user interface was modeled after the project specifications that were set forth by Dr. Brown. The user interface design is described in Figure 2 below. The application has three different users: the coroner/deputies, KYVDRS employees, and medical examiners. KYVDRS employees should only be allowed to create new accounts for coroners and medical examiners. The coroner and the medical examiner are able to login into the application, albeit with different privileges. The coroners and their deputies are the main users for this application. They are the only users that are allowed to start an investigation form. The coroner and medical examiners can edit existing investigation forms and save changes. The coroners should also be able to revert the form to a previous version and submit the finalized form. The user interface design for the Death Scene investigation application is mostly complete, with a few more features that need to be implemented. Screenshots of the iOS Death Scene investigation application itself are shown in Figure 3. Some features that need to be implemented in the future include a navigation controller to organize the flow of the pages of the application, scalability of the application to devices with different dimensions, and improving the aesthetics of the user interface to conform with Apple's human interface guidelines.

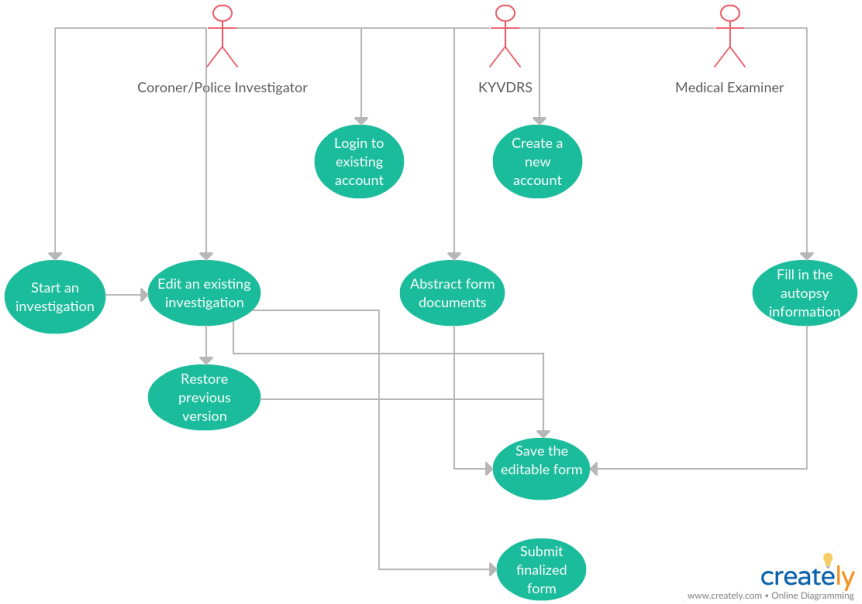

Figure 2. A use case diagram showcasing how different users will interact with the Death Scene Investigation application. The privileges of each user are implied in this diagram.

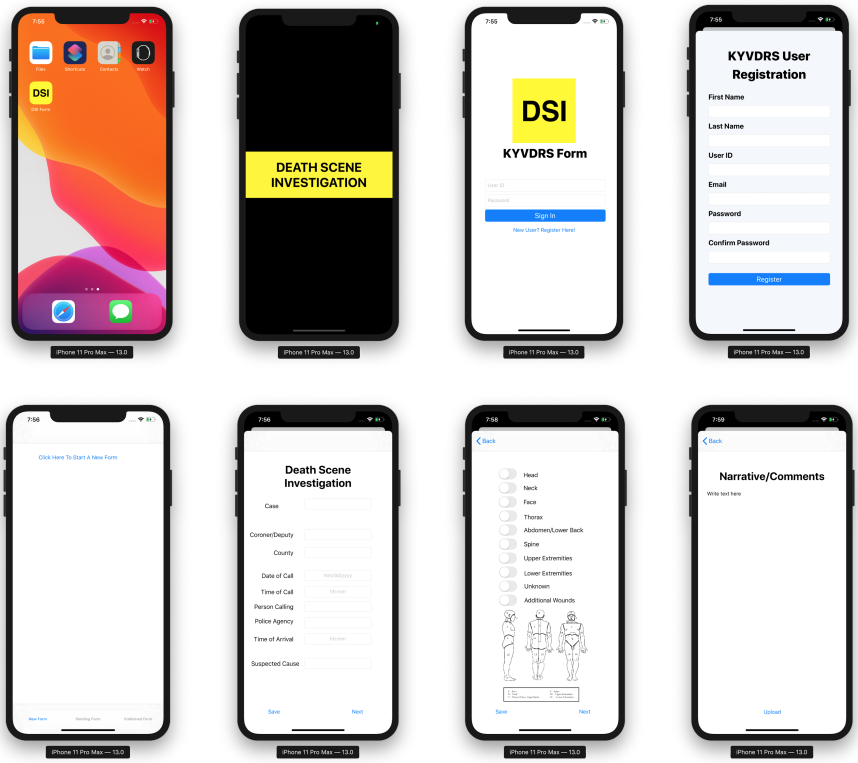

Figure 3. The Death Scene Investigation application on the iOS platform as currently developed. Note the current lack of navigation control and the awkward placing of the Death Scene Investigation form in a modal view.

3. Testing

Our application consists of a user interface and a database stored on a remote server. Users enter information into the iOS application, which is stored in the database. Stored data is also retrieved and displayed on the iOS application. Testing will mostly involve unit testing for individual parts of the iOS application and the database and integration testing to test the functionality of joining these two aspects of our system. Each test case has a unique ID for easy reference and each test consists of test steps where actions are performed or data is entered and expected results, either success or appropriate response to noncomforming input or action. As of now, all tests have not been performed--hence the Not tested status with a yellow background. Once a test is passed, it will indicate "Passed" for the status and the background will be green. If a test fails, the status will be "Failed" and the background will be red.

Test Cases

ID	Type	Description	Test Steps	Expected Results	Status
TC1	Integration	Registering a new user	1. Click register user button 2. Register a new user by entering the required fields	1. System correctly registers a new user and stores their credentials in the database **or** 2. If the user does not enter the required fields, the system prompts them to enter the required fields (now highlighted) until they do	Not tested
TC2	Integration	User login	1. Enter valid user information 2. Click login	1. The system checks to see if the user is registered by retrieving the data from the database (if it exists) **If the user is registered** 2. Accept their credentials and log them in **Else** 3. Reject the user and prompt them to either enter correct credentials or register as a new user	Not tested
TC3	Integration	Saving Data	1. Click "Save form" button on app	**If there is an internet connection** 1. System correctly sends all form data to the database for remote storage **Else** 2. System stores all data locally and backs it up to the server when an internet connection is established	Not tested

TC4	Integration	Edit existing form	1. User selects “Pending form” on main storyboard 2. User selects “Click here to continue form”	1. List of editable forms are shown to user 2. Selected form will be shown to user in the previously saved state 3. Form is editable and can be submitted to DB	Not tested
TC5	Integration	View published forms	1. User selects “Published Form” on main storyboard 2. User selects “View Published Forms Here”	1. Published forms are shown to the user (if they sufficient clearance) 2. Forms can be viewed but not edited	Not tested
TC6	Unit	Start a new form	1. User selects “Start a New Form” on main storyboard	1. New form is presented to the user 2. Data can be input	Passed
TC7	Unit	Upload Photos	1. User selects “Upload Photos” near the end of the form	1. Camera application on the respective device is pulled up. 2. Photos can be taken and uploaded to our app	Not tested
TC8	Unit	Security Clearance	1. Try to access form using each level of clearance	1. System allows access for individuals who are cleared to access the form and rejects those who are not.	Not tested
TC9	Unit	Read Only	1. Try to access form with a user account (District Attorney) that is only cleared to read the form	1. System prevents user from editing	Not tested

4. QA Review Results

After some discussion after review of our current system, Team 6 has found the following issues:

1. We discovered the previous team had not followed Apple's App Store review guidelines (see https://developer.apple.com/app-store/review/guidelines/)
    a. This may have contributed to Apple’s rejection of the application’s publication
2. Some pages in the iOS application (the register page, for example) do not have back buttons and have to be swiped down to return to a previous page. This may not be intuitive to some users, so a back button must be added.
3. Adding a button to return to a previous page will only pull up another instance of the previous page in a modal view. To fix this, navigation controllers must be implemented to manage the flow between the application pages.
4. The preexisting .php files work in connecting the frontend to the backend. However, they are inefficient and create new entities in the database every time the user wishes to save.
5. The interface in general is not user friendly. We will work on editing the current UI and implementing more front-end features in order to ensure the experience is pleasant for the user. This will be a focus before we move on to the Android application.

5. Metrics

1. Estimated story points:

User Stories: 29

1. As a death scene investigator, I need to be able to work on the investigation form offline and sync when online so that I can gather data wherever I need it, regardless of internet connection.

Acceptance Criteria:

- The application is fully functional when there is no connection is available.
- User should be able to view and work with current death scene form and have the data cached on the device until a connection is available.
- When a connection is available, the cached data is imported into the database.

Story Points: 5

2. As a death scene investigator, I need autocomplete on the app so that I can quickly fill out the form without having to type long toxicology terms.

Acceptance Criteria:

- The application can recognize common toxicology terms as the user types and displays a list of closely matching terms for the user to select to fill out a specific field.
- This feature should be implemented for the fields under the Toxicology section of the form.

Story Points: 8

3. As a death scene investigator, I need three levels of login security clearance so that access to sensitive data can be limited to cleared individuals.

Acceptance Criteria:

- The application should show or hide certain information based on the security clearance of the individual.
- The application should also restrict edit permissions for all forms depending on whether a third party or a death scene investigator is accessing the data.
- Death scene investigators should be allowed to access, edit, and view forms that they have created, whereas a third party such as a district attorney can only view forms that they have requested access to.

4. As a death scene investigator, I need the Android and iPhone versions of the application to be identical so that there are no discrepancies in the collection of legal evidence.

Acceptance Criteria:

- The functionality and aesthetics of the application are identical on both versions.
- Both applications should use the same method and format when interfacing with the backend server.

Story Points: 8

5. As a death scene investigator, I need to have access to previous versions of the forms that I have submitted in case a mistake is made in its collection or revision.

Acceptance Criteria:

- Forms that have been submitted can be edited and resubmitted, if need be.
- The user should only be able to access and edit forms that they have submitted in the past.
- The backend database will update the data accordingly.

Story Points: 40

6. As a death scene investigator, I need to be able to upload pictures of the death scene and attach them to the corresponding form that I will be filling out for the death scene.

Acceptance Criteria:

- Both versions of the app (iOS and Android) will be able to successfully access the camera of the device.
- The camera functionality of the application should be able to capture images from the death scene and attach images to the form.
- The application should be able to upload the images corresponding to a specific form to the database.

Story Points: 20

7. As a death scene investigator, I need to ensure that all communication and connections between the mobile application and the database server is secure and that all data being transmitted is encrypted.

Acceptance Criteria:

- Industry-leading security standards will be implemented to ensure all sensitive data is protected. Security is built into the application rather than being an afterthought.
- All data transmitted between the mobile application and the database should be encrypted. This can be done with a mutual authentication scheme, where SSL certificates are exchanged between the application and the database server to ensure that the application is only interfacing with a trusted server (prevents man in the middle attacks).

Story Points: 13

8. As a death scene investigator, I need the application to be easily accessible and usable on different devices of the same platform (iPhones and iPads on iOS, for example).

Acceptance Criteria:

- The format and functionality of the app will be the same regardless of screen size, device specs, etc.
- The application should be able to scale to the appropriate screen size and ensure that all sections and elements of the death scene investigation form are included and are clearly visible on all devices.

Story Points: 5

2. Actual lines of code as implemented:

Swift Files (Front-End)	Lines of Code
CaseHistory.swift	513
RegisterNewUserViewController.swift	33
UsefulFunctions.swift	63
PendingFormViewController.swift	35
NarrativeComments.swift	544
RoundButton.swift	31
IncidentInformation.swift	293
MDSIDocProperties.swift	347
deathsceneInvestigation.swift	252

PublishedViewController.swift	35
Investigation.swift	609
AppDelegate.swift	46
NextofKin.swift	96
UploadPhotos.swift	117
DecedentInformation.swift	358
PillsOnScene.swift	151
NewFormViewController.swift	35
SignInViewController.swift	60
SearchTextField.swift	680
Total	**4298**

PHP Files (Back-End)	**Lines of Code**
start_death_scene_investigation.php	106
edit_next_of_kin.php	47
edit_incident_info.php	72
edit_narrative_comments.php	39
insert.php	33
edit_pills_on_scene.php	49
edit_decedent_info.php	73
edit_investigation.php	137
register.php	54
edit_DSI.php	44
edit_case_history.php	144
authenticate.php	47

retrieve.php	49
Total	**894**

3. McCabe cyclomatic complexity of each module:
    a. authenticate.php: **4**
    b. edit_case_history.php: **3**
    c. edit_decedent_info.php: **3**
    d. edit_DSI.php: **3**
    e. edit_incident_info.php: **3**
    f. edit_investigation.php: **3**
    g. edit_narrative_comments.php: **3**
    h. edit_next_of_kin.php: **3**
    i. edit_pills_on_scene.php: **3**
    j. insert.php: **2**
    k. register.php: **4**
    l. start_death_scene_investigation.php: **4**
4. Complexity of the system: **3** weighted methods per class (in the object oriented Swift files)
5. Product size: **8** user stories, **20** acceptance test plans
6. Product effort (number of person hours):
    a. Evan: **25**
    b. Eura: **25**
    c. Matthew: **25**
    d. Trent: **25**
7. Defects: **4** (refer to section 4)

6. URL for Our Developer Notebook

(click each of our names for individual developer notebooks):
https://github.com/evanfyang/death-scene-investigation/wiki

References

Apple Inc. “App Store Review Guidelines.” *App Store Review Guidelines - Apple Developer*, developer.apple.com/app-store/review/guidelines/.