**C868 – Software Capstone Project Summary**

**Task 2 – Section C**

|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | http://www.idevnews.com/views/images/uploads/general/wgu_logo.png  Dossier Management System |
| **Student Name:** | Phuc Thach |

**Table of Contents**

Contents

[Table of Contents 2](#_Toc87892872)

[Application Design and Testing 3](#_Toc87892873)

[Design Document 3](#_Toc87892874)

[Class Design 3](#_Toc87892875)

[UI Design 3](#_Toc87892876)

[Unit Test Plan 3](#_Toc87892877)

[Introduction 3](#_Toc87892878)

[Purpose 3](#_Toc87892879)

[Overview 3](#_Toc87892880)

[Test Plan 3](#_Toc87892881)

[Items 3](#_Toc87892882)

[Features 3](#_Toc87892883)

[Deliverables 3](#_Toc87892884)

[Tasks 3](#_Toc87892885)

[Needs 3](#_Toc87892886)

[Pass/Fail Criteria 3](#_Toc87892887)

[Specifications 3](#_Toc87892888)

[Procedures 3](#_Toc87892889)

[Results 3](#_Toc87892890)

[C4. Source Code 4](#_Toc87892891)

[C5. Link to Live Version 4](#_Toc87892892)

[User Guide 5](#_Toc87892893)

[Introduction 5](#_Toc87892894)

[Installation and Using the Application 5](#_Toc87892895)

[*Login and Signup (An example*) 6](#_Toc87892896)

[*Classes* 6](#_Toc87892897)

[*Create a New Class* 6](#_Toc87892898)

[*Reports* 7](#_Toc87892899)

# Application Design and Testing

# Design Document

## Class Design

The project uses Command Query Responsibility Segregation Design Pattern and SOLID Principles in its class design diagram. CQRS Design pattern maximizes the application performance, scalability, and security. The SOLID principles encourage developers to create more maintainable, understandable, and flexible software while reducing code complexity.

The SOLID Principle of object-oriented design is a set of five programming rules. If followed, the application is likely to be less rigidity, less fragility, less immobility, and less viscosity (Martin R.C, 2000).

* Single Responsibility – A class should only have one responsibility
* Open-closed Principle – A class should be open for extension but closed for modification
* Liskov Substitution – Subclasses should be substitutable for their base classes
* Interface Segregation – Many clients specific interfaces are better than one general-purpose interface
* Dependency Inversion Principle – Depend upon abstracts. Do not depend on concretions.

The mainstream approach to interacting with information systems is via the CRUD approach (create new records, read records, update existing records, and delete records). This convoluted approach merges software design in a single representation, making interaction with information systems confusing, chatty, and underperforming. The CQRS focuses on how the system reads and writes data for a data store by separating operations into query models for reading and command models for writing data (Fowler M., 2006). This separation of concerns enables the minimization of complexity, making the application more maintainable, extensible, and flexible while maximizing the application’s performance, scalability, and security.

The project Class Diagram in Figure 1 and Figure 2 shows relationships among classes. There are three class types; entity classes correlate with database table structures, behavior classes are responsible for reading and writing data per CQRS Design Pattern, event classes handle the side effects of behavior classes. Behavior classes have “CRUD” prefixes followed by the entity class. For example, CreateDossierCommand class follows Single Class Responsibility in SOLID principles as it only creates a dossier and stores it into the database system. Attach to behavior classes are event classes that are responsible for adding audit logs for historical tracking. Events classes are triggered by behavior classes in a series of event collaborations, where classes work together by communicating with each other by sending events when their internal state changes.

A picture containing diagram

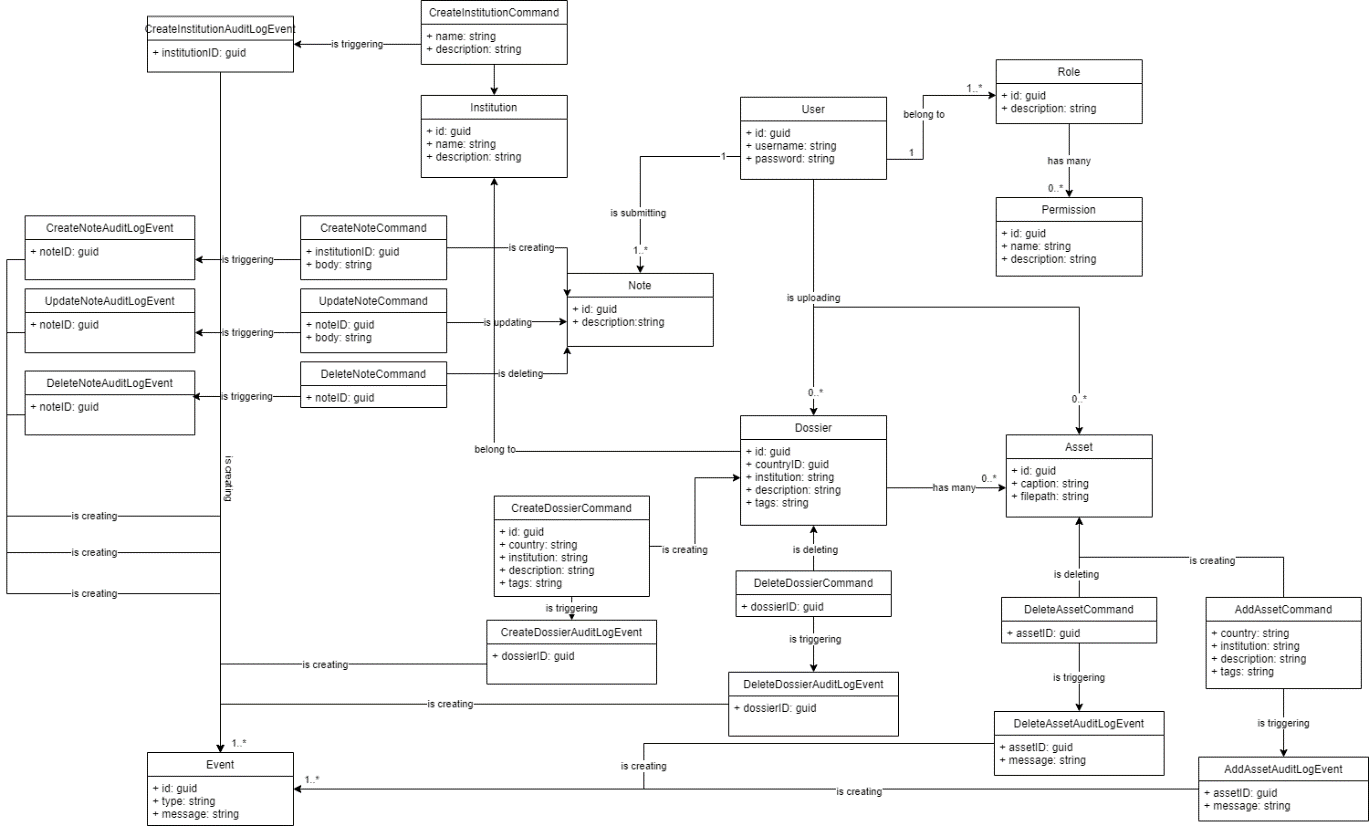
Description automatically generated

Figure 1. Class Design using the CQRS Design Pattern, http://github.com/thachp/dossa/class\_diagram.png

Figure 2. A Dossier is created by CreateDossierCommand, which trigger the CreateDossierAuditLogEvent to create an event log

## UI Design

There are two applications associated with this project: mobile application and web application. The mobile application targets activists who need to securely submit dossiers and other sensitive data via the mobile app. The web application targets analysts who need to download, review, and analyze dossiers. Both applications interface with the same backend services and are integral parts of the Dossier Management System.

The mobile application (Dossa) in Figure 3 shows a mockup of the main screen, requiring the user to press the Connect button before submitting dossiers. Once connected, the user will be taken to the Dossier Submission Form. The user will need to enter required inputs (country, institution, corruption types, description) and optional inputs like hashtags and file assets. There is a VPN connection status label located above the Submit button to indicate their VPN connection status. A green bar means their connection is secured. There are three associated notifications: error, warning, and successful. The Error Notification tells the user there may be issues connecting to the virtual private network. The Warning Notification indicates problems with form validation. Lastly, the Successful Notification indicates dossiers have been submitted successfully.

The web application (Dossier Management), in Figure 4 through Figure 6, shows mockups for three scenes: dashboard, institutions, and reporting. The dashboard scene consists of six widgets: number of total activists, number of dossiers, a bar chart of dossiers submission by months, list of institutions with most dossiers, recently added dossiers, and recently added notes made by other analysts in the organization. The institutions’ scene enables analysts to search institutions by country, name, tag, and corruption types—the Dossier Management web application categories dossiers by the institution. Click on an institution record, and the user will be taken to the Institution Profile scene. The institution profile scene shows the institution name, a list of dossiers related to the institution, and a list of notes left by analysts. Click on a dossier open the Dossier Detail modal, where analysts can see actual dossier description, corruption types, and file assets such as images, audios, videos, documents, etc. Lastly, the reporting scene has four widgets: list countries by dossiers count, a list of institutions ordered by dossiers count, a corruption ration pie-chart displaying corruption types in percentage, and a hashtags widget showing popular hashtags relating to dossiers.

Other features are not in scope but are intended to release in later versions. These features aim to provide analysts enhanced ability to analyze raw dossiers.

* A feature to list people associated with the reported institution. Corruption occurred in the middle-high tiers. Higher tiers need to enforce accountabilities to reduce crime in lower levels. An organization chart will do wonders in understanding the institutions’ internal structure.
* Case management where dossiers can be grouped; create cases, edit a case, assign cases to an institution, associate dossiers to a case profile.
* A way for analysts to bookmark dossiers for viewing later, a scene to view individual asset detail, and functionality that rates institutions.

Graphical user interface

Description automatically generated

Figure 3. Mockup for the mobile application Dossa

Graphical user interface

Description automatically generated with medium confidence

Figure 4. Mockup for the Dashboard scene

Text

Description automatically generated with medium confidence

Figure 5. Mockup for the Institutions List scene

Graphical user interface

Description automatically generated

Figure 6. Mockup for the Reporting scene

# Unit Test Plan

## Introduction

### Purpose

The project test plan intends to find bugs and fix them as soon as possible. The test plan uses three testing techniques: unit testing at the development level, acceptance testing, and black-box testing. Unit testing ensures code stability at the functions level is extremely important as it is the first chance to catch bugs. Acceptance testing determines whether the finished application meets users’ requirements. In black-box testing, testers know what a feature is supposed to do, but testers have no idea how it works. So, testers send random values to form inputs to simulate actual-world experiences. The mobile application, Dossa, will be used widely; therefore, it must be tested rigorously and a priority of all other unit tests.

### Overview

A feature of the Dossier Management System is to provide anonymity for when activists submit dossiers via the mobile application. The following test plan intends to test the functionalities of the dossier submission form and verify that user connections are secured through a third VPN application client called WireGuard Client. A successful test ensures better safety for activists.

## Test Plan

### Items

The test will be performed against the Dossa App, VPN Server, and backend services.

### Features

The test will perform tests on two features; VPN Connection, which provides user anonymity, and the Dossier Submission Form, which allows activists to submit dossiers.

### Deliverables

VPN Status of Connected. A Successful Notification upon dossiers submitted.

### Tasks

As an activist, perform test tasks on the Dossa mobile application.

* + Launch the Dossa App > Click on Connect. Verify that connection status is “connected” in the following screen.
  + Launch the Dossa App > Click on Connect > Enter required inputs > Click Submit. Verify that the application displays a Successful Notification to indicate successful submission.

### Needs

Three of the four subsystems of the Dossier Management System must be running. Dossa must be installed on a mobile device, VPN Server started and running, and the Parse backend services must also be running.

### Pass/Fail Criteria

A pass result entails a successful VPN connected status, and that form submission is completed with a successful notification. A failed test entails failed VPN connection, or that Dossier Submission Form submission has failed. If failed, verify that the VPN server is running. Also, check that required fields are include some values.

## Specifications

Provide sample code that represents what testing code was used. Screenshots are acceptable.

## 

## Procedures

Provide a detailed list of the steps you used to complete the testing process. Be sure to mention if iterations were/are part of the process used and when pass/fail results were provided.

## 

## Results

Here you will describe and provide examples of the testing results. If you were using a testing package, include a screenshot of the interface. Screenshot work best.

# C4. Source Code

All source codes for all subsystems of the application are included with this documentation. It is also accessible via Github.com. Please see <https://github.com/thachp/dossa>

# C5. Link to Live Version

Both the mobile application and a live website for the application are accessible via the URL <https://www.dossa.network>.

Expo is an open-source platform for making universal native Android, iOS, and the web with JavaScript and React. It provides a convenient way for developers to publish mobile apps without going through app submission steps with the Apple App Store or Google Play. Note that to install the mobile application on a mobile device, you must first install the Expo app on IOS or Android.

# References

Fowler, M. (2006, June 19). *Event Collaboration*. Martinfowler.Com. Retrieved November 19, 2021, from https://martinfowler.com/eaaDev/EventCollaboration.html

Fowler, M. (2011, July 14). *bliki: CQRS*. Martinfowler.Com. Retrieved November 19, 2021, from https://martinfowler.com/bliki/CQRS.html

Martin, R. C. (2000). *Design Principles and Design Patterns*. Universidad ORT Uruguay. Retrieved November 19, 2021, from https://fi.ort.edu.uy/innovaportal/file/2032/1/design\_principles.pdf

Millington, S. (2021, May 18). *A Solid Guide to SOLID Principles*. Baeldung. Retrieved November 19, 2021, from https://www.baeldung.com/solid-principles