**Software Design Document**

Cloud Database Migration & LACPD New Hire Enrollment Web Application

**Version 1.1 approved**

**Prepared by Chen Albert, Simon Mai, Shahram Mehri Kalantari, Wilfredo Paz, Michael Loria, Ryan Lee, Ismael Valenzuela, Joshua Perez, Norberto Rosales, Fabio Quintana**

**Law Offices of Los Angeles County Public Defender & Cal State LA**

**11/30/2021**

Table of Contents................................................................................................................<pg 2-3>

Revision History..................................................................................................................... <pg 4>

1. Introduction................................................................................................................ <pg 5>
	1. Purpose........................................................................................................... <pg 5>
	2. Document Conventions……………………................................................. <pg 5>
	3. Intended Audience and Reading Suggestions................................................ <pg 5>
	4. System Overview........................................................................................... <pg 6>
2. Design Considerations............................................................................................... <pg 6>
	1. Assumptions and dependencies..................................................................... <pg 6>
	2. General Constraints........................................................................................<pg 6>
	3. Goals and Guidelines.....................................................................................<pg 7>
	4. Development Methods................................................................................... <pg 7>
3. Architectural Strategies.............................................................................................. <pg 8>
4. System Architecture................................................................................................... <pg 9>
	1. ....................................................................................................................... <pg 9>
	2. ......................................................................................................................<pg 10>
5. Policies and Tactics..................................................................................................<pg 11>
	1. Specific Products Used................................................................................ <pg 11>
	2. Requirements traceability.............................................................................<pg 11>
	3. Testing the software......................................................................................<pg 11>
	4. Engineering trade-offs................................................................................. <pg 11>
	5. Guidelines and conventions..........................................................................<pg 11>
	6. Protocols.......................................................................................................<pg 11>
	7. Maintaining the software............................................................................. <pg 11>
	8. Interfaces......................................................................................................<pg 11>
	9. System's deliverables....................................................................................<pg 12>
	10. Abstraction...................................................................................................<pg 12>
6. Detailed System Design.......................................................................................... <pg 13>

6.x Name of Module.........................................................................................<pg 13>

6.x.1 Responsibilities....................................................................................<pg 13>

6.x.2 Constraints...........................................................................................<pg 13>

6.x.3 Composition.........................................................................................<pg 13>

6.x.4 Uses/Interactions..................................................................................<pg 13>

6.x.5 Resources.............................................................................................<pg 13>

6.x.6 Interface/Exports..................................................................................<pg 14>

1. Detailed Lower level Component Design

7.x Name of Class or File....................................................................................<pg 15>

7.x.1 Classification........................................................................................<pg 15>

7.x.2 Processing Narrative(PSPEC)..............................................................<pg 15>

7.x.3 Interface Description............................................................................<pg 15>

7.x.4 Processing Detail..................................................................................<pg 15>

7.x.4.1 Design Class Hierarchy.................................................................<pg 15>

7.x.4.2 Restrictions/Limitations................................................................<pg 15>

7.x.4.3 Performance Issues......................................................................<pg 15>

7.x.4.4 Design Constraints.......................................................................<pg 15>

7.x.4.5 Processing Detail For Each Operation.........................................<pg 15>

1. User Interface
	1. Overview of User Interface.........................................................................<pg 17>
	2. Screen Frameworks or Images....................................................................<pg 17>
	3. User Interface Flow Model.........................................................................<pg 17>
2. Database Design.......................................................................................................<pg 16>
3. Requirements Validation and Verification................................................................<pg 18>
4. Glossary....................................................................................................................<pg 19>
5. References................................................................................................................<pg 19>

**Revision History**

| Name | Date | Reason For Changes | Version |
| --- | --- | --- | --- |
|  Albert Chen | 11/15 |  initial creation |  1.0 |
| Albert Chen | 11/29 | filled in 1-7 | 1.1 |
|   |   |   |   |
|  |  |  |  |

<Add rows as necessary when the document is revised. This document should be consistently updated and maintained throughout your project. If ANY requirements are changed, added, removed, etc., immediately revise your document.>

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to:

1. Define a full set of software requirements for Public Defender’s Enrollment Application.
2. The scope of the product that this SDD covers will include Public Defender’s Enrollment Application and the migration of PowerApps data to Azure SQL database. The SDD describes only part of the system and provides the information necessary of the system and the software to be assembled.

**1.2 Document Conventions**

N/A

**1.3 Intended Audience and Reading Suggestions**

This document is intended for project managers, users, testers, and documentation writers.

The rest of this SDD will contain Design Considerations, Architectural Strategies, System Architecture, Policies and Tactics, Name of Component Module, Detailed Lower Level Component Design, Database Design, Overview of User Interface, Requirements, Glossary, and References.

**1.4 System Overview**

The software system’s functionality includes the integration of Microsoft Power Apps to another database. Overall system will be two applications of Power Apps and another for the Enrollment Application.The two applications will have its data migrated to AzureSQL Database. The Enrollment Application will be updated to integrate with Box cloud storage, eliminate public access, and implement many smaller changes.

**2. Design Considerations**

Requires access to the LACPD dev environment to work on Power Apps. Also requires an Adobe Sign account to connect the front-end to the back-end of the old Enrollment Application.

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

**2.1 Assumptions and Dependencies**

Describe any assumptions or dependencies regarding the software and its use. These may concern such issues as:

* Related software or hardware
* Operating systems
* End-user characteristics
* Possible and/or probable changes in functionality
* Software: Web Browser, MS Power Platform, MS PowerApps, Adobe Sign Account, Database is required to store documents, and a server to launch the application.

**2.2 General Constraints**

Describe any global limitations or constraints that have a significant impact on the design of the system's software (and describe the associated impact). Such constraints may be imposed by any of the following (the list is not exhaustive):

* Access to the Dev Environment account
* Adobe Sign API Tokens

**2.3 Goals and Guidelines**

Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system's software. For each such goal or guideline, unless it is implicitly obvious, describe the reason for its desirability. Feel free to state and describe each goal in its own subsubsection if you wish. Such goals might be:

* The KISS principle ("Keep it simple stupid!")
* The Software has a mandatory delivery date that must be met (end of the cd3337 class)
* Emphasis on speed versus memory use
* The product should work, look, or "feel" like an existing product

**2.4 Development Methods**

Briefly describe the method or approach used for this software design. If one or more formal/published methods were adopted or adapted, then include a reference to a more detailed description of these methods. If several methods were seriously considered, then each such method should be mentioned, along with a brief explanation of why all or part of it was used or not used.

These would be things such as the ‘Water Fall Development’ methods, ‘Agile Development’, ‘Unplanned Mad Scramble Development’, or other development models and variations. Describe how these were applied in the case of your project.

**3. Architectural Strategies**

Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Such decisions might concern (but are not limited to) things like the following:

* Use of a particular type of product (programming language, database, library, etc. ...)
* Reuse of existing software components to implement various parts/features of the system
* Future plans for extending or enhancing the software
* User interface paradigms (or system input and output models)
* Hardware and/or software interface paradigms
* Error detection and recovery
* Memory management policies
* External databases and/or data storage management and persistence
* Distributed data or control over a network
* Generalized approaches to control
* Concurrency and synchronization
* Communication mechanisms
* Management of other resources

Each significant strategy employed should probably be discussed in its own subsection. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose).

**4. System Architecture**

This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.



This is where the level 0 DFD will probably work best.

At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its modules/components/subsystems (identifying each top-level modules/component/subsystem and the roles/responsibilities assigned to it).

Each subsection (i.e. “4.1.3 The ABC Module”) of this section will refer to or contain a detailed description of a system software component.











Level 1 Data Flow Diagrams (DFD) and Control Flow Diagrams (CFD) should probably go here.



Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Feel free to make use of design patterns, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them. Diagrams that describe a particular component or subsystem in detail should be included within the particular subsection that describes that component or subsystem.

**5. Policies and Tactics**

Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose). Such decisions might concern (but are not limited to) things like the following (Must include 5.1, 5.2, and 5.3. The rest of these categories or custom ones can be added as needed.):

**5.1 Choice of which specific products used**

(IDE, compiler, interpreter, database, library, etc. ...)

**5.2 Plans for ensuring requirements traceability**
 …Describe…

**5.3 Plans for testing the software**

 …Describe…

5.# Engineering trade-offs
 …Describe…

5.# Coding guidelines and conventions

 …Describe…

5.# The protocol of one or more subsystems, modules, or subroutines

 …Describe…

5.# The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality

 …Describe…

5.# Plans for maintaining the software

 …Describe…

5.# Interfaces for end-users, software, hardware, and communications

 …Describe…

5.# Hierarchical organization of the source code into its physical components (files and directories).

 …Describe…

5.# How to build and/or generate the system's deliverables (how to compile, link, load, etc.)

 …Describe…

5.# Describe tactics such as abstracting out a generic DatabaseInterface class, so that changing the database from MySQL to Oracle or PostGreSQL is simply a matter of rewriting the DatabaseInterface class.

For this particular section, it may become difficult to decide whether a particular policy or set of tactics should be discussed in this section, or in the System Architecture section, or in the Detailed System Design section for the appropriate component. You will have to use your own "best" judgement to decide this. There will usually be some global policies and tactics that should be discussed here, but decisions about interfaces, algorithms, and/or data structures might be more appropriately discussed in the same (sub) section as its corresponding software component in one of these other sections.

**6. Detailed System Design**

Most components described in the System Architecture section will require a more detailed discussion. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

This is where Level 2 (or lower) DFD’s will go. If there are any additional detailed component diagrams, models, user flow diagrams or flowcharts they may be included here.

**6.x Name of Component (Module)**

**6.x.1 Responsibilities**

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

**6.x.2 Constraints**

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

**6.x.3 Composition**

A description of the use and meaning of the subcomponents that are a part of this component.

**6.x.4 Uses/Interactions**

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclass’s, and metaclasses.

**6.x.5 Resources**

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

**6.x.6 Interface/Exports**

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code.

**7. Detailed Lower level Component Design**

Other lower-level Classes, components, subcomponents, and assorted support files are to be described here. You should cover the reason that each class exists (i.e. its role in its package; for complex cases, refer to a detailed component view.) Use numbered subsections below (i.e. “7.1.3 The ABC Package”.) Note that there isn't necessarily a one-to-one correspondence between packages and components.

**7.x Name of Class or File**

**7.x.1 Classification**

The kind of component, such as a subsystem, class, package, function, file, etc.

**7.x.2 Processing Narrative (PSPEC)**

A process specification (PSPEC) can be used to specify the processing details

**7.x.3 Interface Description**

**7.x.4 Processing Detail**

**7.x.4.1 Design Class Hierarchy**

Class inheritance: parent or child classes.

**7.x.4.2 Restrictions/Limitations**

**7.x.4.3 Performance Issues**

**7.x.4.4 Design Constraints**

**7.x.4.5 Processing Detail For Each Operation**

**8. Database Design**

Include details about any databases used by the software. Include tables and descriptions.

**9. User Interface**

The user interface is the application, from the point of view of the users. Do your classes and their interactions (the logical and process views) impose restrictions on the user interface? Would removing some of these restrictions improve the user interface? Use some form of user interface flow model to provide an overview of the UI steps and flows. Don't go into too much refinement. You should include screen shots or wireframe layouts of significant pages or dialog elements. Make sure to indicate which of the system level modules or components that each of these user interface elements is interacting with.

**9.1 Overview of User Interface**

Describe the functionality of the system from the user’s perspective. Explain how the user

will be able to use your system to complete all the expected features and the feedback

Information that will be displayed for the user. This is an overview of the UI and its use. The user manual will contain extensive detail about the actual use of the software.

**9.2 Screen Frameworks or Images**

These can be mockups or actual screenshots of the various UI screens and popups.

**9.3 User Interface Flow Model**

A discussion of screen objects and actions associated with those objects. This should include a flow diagram of the navigation between different pages.

**10. Requirements Validation and Verification**

Create a table that lists each of the requirements that were specified in the SRS document for this software.

For each entry in the table list which of the Component Modules and if appropriate which UI elements and/or low level components satisfies that requirement.

For each entry describe the method for testing that the requirement has been met.

**11. Glossary**

An ordered list of defined terms and concepts used throughout the document. Provide definitions for any relevant terms, acronyms, and abbreviations that are necessary to understand the SDD document.  This information may be listed here or in a completely separate document.  If the information is not directly listed in this section provide a note that specifies where the information can be found.

**12. References**

<List any other documents or Web addresses to which this SDD refers. These may include other SDD or SRS documents, user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Brad Appleton <brad@bradapp.net>  <http://www.bradapp.net>

<https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate_Fall08.doc>

https://csns.cysun.org/department/cs/project/view?id=7808904