Software Design Document
for
L.A.P.A.
Version 1 approved

Prepared by Brian Kan, Araceli Lopez, Kaila Mayho, Daniel Bollinger, Patricia Luz, Sharon Lake, Phillip Han

L.A. City Bureau of Engineering
September 23, 2017
Table of Contents

1. Introduction...........................................................................................................<pg 4>
   1.1. Purpose...........................................................................................................<pg 5>
   1.2. Document Conventions..................................................................................<pg 5>
   1.3. Intended Audience and Reading Suggestions..............................................<pg 6>
   1.4. System Overview............................................................................................<pg 6>

2. Design Considerations..........................................................................................<pg 6>
   2.1. Assumptions and dependencies......................................................................<pg 6>
   2.2. General Constraints.......................................................................................<pg 6, 7>
   2.3. Goals and Guidelines.....................................................................................<pg 7>
   2.4. Development Methods.....................................................................................<pg 8>

3. Architectural Strategies.......................................................................................<pg 8>

4. System Architecture............................................................................................<pg 9>
   4.1. .......................................................................................................................<pg 10-13>
   4.2. .......................................................................................................................<pg 13-15>

5. Policies and Tactics..............................................................................................<pg 15>
   5.1. Specific Products Used..................................................................................<pg 15>
   5.2. Requirements traceability.............................................................................<pg 16>
   5.3. Testing the software.....................................................................................<pg 16>
   5.4. Engineering trade-offs.................................................................................<pg 16>
   5.5. Guidelines and conventions..........................................................................<pg 16>
   5.6. Protocols.......................................................................................................<pg 16>
   5.7. Maintaining the software.............................................................................<pg 16>
   5.8. Interfaces......................................................................................................<pg 16>
   5.9. System's deliverables....................................................................................<pg 16>
   5.10. Abstraction.................................................................................................<pg 16>

6. Detailed System Design.......................................................................................<pg 16>
   6.x Name of Module..............................................................................................<pg 16>
   6.x.1 Responsibilities............................................................................................<pg 17>
   6.x.2 Constraints..................................................................................................<pg 17>
   6.x.3 Composition..................................................................................................<pg 17>
   6.x.4 Uses/Interactions..........................................................................................<pg 17>
   6.x.5 Resources....................................................................................................<pg 17>
   6.x.6 Interface/Exports..........................................................................................<pg 17>

7. Detailed Lower Level Component Design..........................................................<pg 18>
   7.x Name of Class or File......................................................................................<pg 18>
   7.x.1 Classification...............................................................................................<pg 18>
   7.x.2 Processing Narrative(PSPEC)......................................................................<pg 18>
   7.x.3 Interface Description....................................................................................<pg 18>
   7.x.4 Processing Detail..........................................................................................<pg 18>
   7.x.4.1 Design Class Hierarchy.............................................................................<pg 18>
7.x.4.2 Restrictions/Limitations...........................................................................<pg 18>
7.x.4.3 Performance Issues..................................................................................<pg 18>
7.x.4.4 Design Constraints..................................................................................<pg 18>
7.x.4.5 Processing Detail For Each Operation....................................................<pg 18>

8. Database Design

9. User Interface
   9.1. Overview of User Interface.................................................................<pg 20>
   9.2. Screen Frameworks or Images..............................................................<pg 21>
   9.3. User Interface Flow Model......................................................................<pg 22,23>

10. Requirements Validation and Verification....................................................<pg 24>
11. Glossary........................................................................................................<pg 25>
12. References....................................................................................................<pg 25>
## Revision History

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Reason For Changes</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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
Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.

The purpose of this document is four-fold:

a) Completely define a full set of requirements for the L.A.P.A. - Section 3.0
b) Completely define the design for the L.A.P.A. – Section 4.0.
c) Define and partially implement feasible modules for the L.A.P.A. – Section 5.0.
d) Completely define the Test Plan for the L.A.P.A. – Section 6.0.

The complete definition of all L.A.P.A. requirements provides the source requirement inputs for the development of the subsequent supporting software subsystems documents.

The scope of this document includes the following:

- All functional and nonfunctional requirements on the L.A.P.A. are captured. This includes Verification & Validation (V&V) requirements, as well as inter-software subsystems requirements.
- A complete set of L.A.P.A. Requirements. These requirements are organized by key L.A.P.A. functional units shown on the Level 1 DFD. The Level 1 DFD is shown on page
- A trace matrix, relating all L.A.P.A. functional requirements to functional subunits as expanded in lower level DFDs. Level 2 and higher DFDs are provided on pages.
- The functional requirements defined in the L.A.P.A. Requirements section have been expanded to include more specific hardware requirements.

1.2 Document Conventions
Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.
1.3 Intended Audience and Reading Suggestions
Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

Developers, project managers, testers and documentation writers. The SRS contains an overall description of the project as well as the specific functional requirements of the app itself.

1.4 System Overview
Provide a general description of the software system including its functionality and matters related to the overall system and its design (perhaps including a discussion of the basic design approach or organization)

The final product is to have a functional application where multiple photos can be taken and labeled Before and After. Also the project will be able to mark the primary photo along with give the photos the appropriate categories to be organized within the database.

2. Design Considerations
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:
- The phone is running at least KitKat(4.0) Android.
- The phone is capable of internet access.
- The phone has a functioning browser.
- The phone is equipped with a camera.

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):

- Don’t have access to
  - Hardware or software environment
  - End-user environment
  - Availability or volatility of resources
  - Standards compliance
  - Interoperability requirements
  - Interface/protocol requirements
  - Data repository and distribution requirements
  - Security requirements (or other such regulations)
  - Memory and other capacity limitations
  - Performance requirements
  - Network communications
  - Verification and validation requirements (testing)
  - Other means of addressing quality goals
  - Other requirements described in the requirements specification

You will not need to include all of these. Only the ones that will influence the design of your software

TBD

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

- A fully functional application
- Change the User Interface
- Add before and After Photo option along with categories

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.

TBD

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).

- Implement an iOS version of the application

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.

[Diagram showing SQL Server Database, Azure Storage Container, Mobile App (Android/iOS), with arrows indicating flow of Image/Tag Information and Image File]
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.
DFD Level 1
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 Database Interface class, so that changing the database from MySQL to Oracle or PostGreSQL is simply a matter of rewriting the Database Interface 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 screenshots 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.
- Be able to sign in
- Select the tags for their photo
- Select the photo within the app or in their personal gallery
- Populate the given fields for the categories chosen
- View all of their recent upload history

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.

Requirements Associated with Application Launching

<table>
<thead>
<tr>
<th>Requirement No.</th>
<th>Requirement Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1</td>
<td>LAPA shall be modified so that it can be launched from a web application link that contains a username and Repair Task ID.</td>
</tr>
<tr>
<td>4.1.2</td>
<td>LAPA shall be modified to prompt the user to enter their name and permit number or Job identification number.</td>
</tr>
</tbody>
</table>

Requirements Associated with Application Uploading

<table>
<thead>
<tr>
<th>Requirement No.</th>
<th>Requirement Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.3</td>
<td>LAPA shall be modified to prefill the username and Repair Task ID fields based on the link used to launch the application.</td>
</tr>
<tr>
<td>4.1.4</td>
<td>LAPA shall provide a list of photo categories and subcategories for the user to choose from.</td>
</tr>
<tr>
<td>4.1.5</td>
<td>LAPA shall upload photos to a Microsoft Azure Blob Container.</td>
</tr>
<tr>
<td>4.1.6</td>
<td>LAPA shall be modified to insert records with the Microsoft Azure Blob Storage link, Repair Task ID, username, latitude, longitude, before/after, and primary image into a Microsoft Azure SQL table.</td>
</tr>
</tbody>
</table>

Application Pulling Photos

<table>
<thead>
<tr>
<th>Requirement No.</th>
<th>Requirement Description</th>
</tr>
</thead>
</table>
### 4.1.7
LAPA shall be modified to pull pre-existing photos associated with the Repair Task ID from the Microsoft Azure Blob Storage.

### 4.1.8
LAPA shall pull a list of categories and subcategories, Azure Blob Storage links, Repair Task ID, before/after, latitude and longitude from the Microsoft Azure SQL DB.

## 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.

L.A.P.A: Los Angeles Photo Archive
DB: Database
App: An Android-Specific Application.

## 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](http://www.bradapp.net)
[https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate_Fall08.doc](https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate_Fall08.doc)
PABOE SRS v1.0