Software Design Document for Upgrade of Tenant and Owner Portals to Housing Authority Website

Version 1 to be approved

Prepared by Juan Rojas
Simon Bach
Michael Schleicher
Juan Orozco
Ravin Bhakta

Los Angeles County, Housing Authority

December 7, 2018
Table of Contents

Revision History

1. Introduction
   1.1. Purpose
   1.2. Document Conventions
   1.3. Intended Audience and Reading Suggestions
   1.4. System Overview

2. Design Considerations
   2.1. Assumptions and dependencies
   2.2. General Constraints
   2.3. Goals and Guidelines
   2.4. Development Methods

3. Architectural Strategies

4. System Architecture

5. Policies and Tactics
   5.1. Specific Products Used
   5.2. Requirements traceability
   5.3. Testing the software

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

7. Detailed Lower level Component Design
   7.x Name of Class or File
      7.x.1 Classification
      7.x.2 Processing Narrative (PSPEC)
      7.x.3 Interface Description
      7.x.4 Processing Detail
         7.x.4.1 Design Class Hierarchy
         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. User Interface
   8.1. Overview of User Interface
   8.2. Screen Frameworks or Images
   8.3. User Interface Flow Model

9. Database Design
## Revision History

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Reason For Changes</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Draft</td>
<td>9/8/2018</td>
<td>Plan and fill out some material</td>
<td>0.1</td>
</tr>
<tr>
<td>Secondary Draft</td>
<td>12/7/2018</td>
<td>Filled out more material</td>
<td>0.5</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Purpose
The purpose of this Software Design Document, SRS for short, is to outline the Design for the Portal rewrite of the County of Los Angeles Housing Authority portal. The scope of this project only includes the housing authority portal which both tenants and owners use. This portal will connect to an already existing website owned by the county.

1.2 Document Conventions
Every requirement will have its own priority unless specifically stated.

1.3 Intended Audience and Reading Suggestions
The intended audience for this SRS are the students creating the project as well as the sponsors, in this case the county of Los Angeles.

1.4 System Overview
The housing authority portal allows both tenants and owners submit applications for affordable housing as well as check their information and a list of what needs to be done. The purpose for the redesign of the portal is to make it modern not only aesthetically but also functionally by using modern web building practices.
2. Design Considerations

2.1 Assumptions and Dependencies

Assumptions and Dependencies regarding the software and its use:

- ASP.NET MVC
- Entity Framework
- Microsoft SQL
- Web API
- XML

2.2 General Constraints

Currently the software has no general constraints since the project is still in the planning and development phase.

2.3 Goals and Guidelines

The software must be intuitive for the community since it will be used for Housing Section 8 of Los Angeles County.

2.4 Development Methods

The formal development method used is Entity Framework - Database First Approach. Database will be created first, and the Model will be created from the database.
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

The TOPHA architecture is summarized in the Context Diagram (DFD Level 0). The Context Diagram displays the structure of the software modules

![Diagram showing TOPHA architecture]
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
- Microsoft Visual Studio 2017
- Microsoft SQL
- VMware

5.2 Plans for ensuring requirements traceability
In order to ensure requirements traceability is by designing the portal to be intuitive for all users. Our goal is to make the portal to be more intuitive and improve performance from the current portal.

5.3 Plans for testing the software
…Describe…

5.4 Engineering trade-offs
…Describe…

5.5 Coding guidelines and conventions
…Describe…

5.6 The protocol of one or more subsystems, modules, or subroutines
…Describe…

5.7 The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality
…Describe…

5.8 Plans for maintaining the software
…Describe…

5.9 Interfaces for end-users, software, hardware, and communications
5. Describe hierarchical organization of the source code into its physical components (files and directories).

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

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. 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.

8.1 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.
8.2 Screen Frameworks or Images

Tenant and Landlord Portal
Login

Agency Code ID
LACDC

User Name

Password

I'm not a robot

Login

Tenant and Landlord Portal
Registration

Agency Code ID
LACDC

User Type
Tenant

Registration ID

SSN

Full Name

Phone

Company Name

Company Title

UserName

Email

Password

Confirm Password
8.3 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.
9. Database Design

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

<table>
<thead>
<tr>
<th>Requirement Description</th>
<th>Component Modules</th>
<th>UI Elements</th>
<th>Low Level Components</th>
<th>Testing Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example Requirement</td>
<td>Example Module</td>
<td>Example UI</td>
<td>Example Component</td>
<td>Testing Procedure</td>
</tr>
<tr>
<td>Another Example</td>
<td>Another Module</td>
<td>Another UI</td>
<td>Another Component</td>
<td>Another Testing Procedure</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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