System Design Document (SDD) Project:

System Design Document (SDD)
Purpose of the Document

The system design document is meant to track the design of the project and to guide the team in development. The design of the system will be broken down while paying attention to the specific requirements listed in the requirement analysis document (RAD) and following the design goals set forth by the bits team and the Potentia mentors. This document will break down the system into subsystems with explanations on how these systems are supposed to be designed and how they will interact with the other subsystems to complete the whole.

Table of Contents

1. Introduction .............................................................................................................. 5
   1.1 Purpose of the System ...................................................................................... 5
   1.2 Design goals ..................................................................................................... 5-8
   1.3 Definitions, acronyms, and abbreviations .................................................. 8-9
   1.4 References ....................................................................................................... 9
       1.4.1 Related Documentation ................................................................. 9
       1.4.2 Software Architecture Constraints ............................................. 10
   2. Current software architecture ............................................................................ 10
       2.1 Overview ................................................................................................... 10
   3. Proposed software architecture ......................................................................... 11
       3.1 Overview ................................................................................................. 11
       3.2 Subsystem decomposition .......................................................................... 12-13
       3.2.2 Hardware/software mapping .......................................................... 14
       3.2. Relational Database Table Diagram .................................................... 15
3.2.4 Deployment Diagram.................................................................................................................. 15

3.2.5 Persistent data management .................................................................................................. 16

3.2.6 Access control and security .................................................................................................. 16

3.2.7 Global software control ....................................................................................................... 16

3.2.8 Boundary conditions ........................................................................................................... 17

4. Subsystem services ..................................................................................................................... 17-18
About the System Design Document

VERSION CONTROL

Version No. iv

Name(Title)  Author  Date  Authorization

2.0 SDD Report January 20, 2019
2.1 SDD Final January 20, 2019

Document Properties

Item Details

Document Title

Authors: Nour Farhat, Nick Ramsey, Jeremy DeVries, Brennan Powers, Grant Tolarco

Creation Date 01/18/2019

Last Updated 4/30/2019
1. Introduction

Potentia Analytics is a software provider focused on improving hospital workflow. Their scheduling software Symphony has been released to multiple hospitals with over 45 million hours scheduled. With new software Bernoulli and Foresight released, they have started looking into ways to compare hospital statistics for displaying the power of their scheduling algorithms and for advertising on how hospitals can benefit from using Potentia’s software. The Hospital comparison tool will be a way for Potentia to use publicly gathered data and present it in an easy to understand format. The tool will have compared information on a huge number of hospitals in the U.S. and will allow the user to look up hard to understand data, receive it in a digestible format with side by side comparisons to the national and state averages, as well as getting information on how Potentia can improve their hospitals performance.

1.1 Purpose of the System

The Hospital Comparison Tool will be used as a marketing tool for comparing various hospital data to national and state averages. As a marketing tool, this will help Potentia show where some hospitals need improvement and to provide information to the customers on why they should consider Symphony. The system will display various hospital statistics for hospitals to identify problem areas and learn more about how Potentia can help improve these areas. Alternatively, the tool is meant to be used by patients to find, compare, and gather information about hospitals.

1.2 Design goals

1.2.1 Performance

<table>
<thead>
<tr>
<th>Design Criterion</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Time</td>
<td>A user request is acknowledged quickly.</td>
</tr>
<tr>
<td>Throughput</td>
<td>The system can accomplish all needed tasks in a fixed period.</td>
</tr>
<tr>
<td>Memory</td>
<td>The system will run using minimum memory required to run most web application.</td>
</tr>
</tbody>
</table>
### 1.2.2 Dependability

<table>
<thead>
<tr>
<th>Design Criterion</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robustness</td>
<td>The web application will continue without crashes due to invalid/incorrect input.</td>
</tr>
<tr>
<td>Reliability</td>
<td>Capability of the system to operate as expected regardless of user input/operation in the Django server environment.</td>
</tr>
<tr>
<td>Security</td>
<td>Integrity of the database system will be protected from corruption and invalid input.</td>
</tr>
<tr>
<td>Availability</td>
<td>The web application should always be available for any user to use at any time.</td>
</tr>
<tr>
<td>Relevance</td>
<td>The Django server will stay up-to-date with the public records used to create the comparisons.</td>
</tr>
</tbody>
</table>

### 1.2.3 Cost

<table>
<thead>
<tr>
<th>Design Criterion</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Cost</td>
<td>The time and resources of 5 software developers.</td>
</tr>
<tr>
<td>Deployment Cost</td>
<td>The cost of deploying on Digital Ocean to host the server and the cost of hardware or cloud space to host the database.</td>
</tr>
<tr>
<td>Upgrade Cost</td>
<td>Potentia will need to pay for upgrades to the software, since they own the rights to it.</td>
</tr>
<tr>
<td>Maintenance cost</td>
<td>Potentia will need to pay for the hosting service which will host the application.</td>
</tr>
<tr>
<td>Administration Cost</td>
<td>Potentia will need to pay for someone to administrate the software.</td>
</tr>
</tbody>
</table>
### 1.2.4 Maintenance

<table>
<thead>
<tr>
<th>Design Criterion</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensibility</td>
<td>It is recommended that additional modules to the system be added by an experienced developer.</td>
</tr>
<tr>
<td>Modiﬁability</td>
<td>The system is not easily modifiable due to the lack of admin privileges.</td>
</tr>
<tr>
<td>Adaptability</td>
<td>The system can be easily used on supported browsers.</td>
</tr>
<tr>
<td></td>
<td>Known Supported Browsers:</td>
</tr>
<tr>
<td></td>
<td>- Chrome, Firefox, Internet Explorer, Mobile, Edge, Safari</td>
</tr>
<tr>
<td>Readability</td>
<td>The code should be easily readable, comprehensible, and have enough comments to understand it.</td>
</tr>
<tr>
<td>Traceability of requirements</td>
<td>The ability to map code to specific requirements</td>
</tr>
</tbody>
</table>

### 1.2.5 End User Criteria

<table>
<thead>
<tr>
<th>Design Criterion</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utility</td>
<td>The system should be able to support the end user in finding hospital data.</td>
</tr>
<tr>
<td>Usability</td>
<td>The user should have the ability to use the system, with minimal knowledge of basic web-browser navigation.</td>
</tr>
<tr>
<td>Technology to be Used</td>
<td>PostgreSQL, Django, AngularJS, Mapeal</td>
</tr>
</tbody>
</table>
1.2.6 Scope Trade-off

<table>
<thead>
<tr>
<th>Trade-off</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space vs. Speed</td>
<td>The priority for this system would fall more towards speed than space. The web application needs to be just as responsive as any other web app commonly used today. However, with modern servers’ capabilities this is not something we really need to consider as any kind of speed we could gain from taking more space would be completely negligible from the users perspective.</td>
</tr>
<tr>
<td>Delivery Time vs. Functionality</td>
<td>With our deadline at the end of the semester, our delivery time is more important than any kind of additional functionality. The core functions, as defined in our RAD will be completed by the delivery date.</td>
</tr>
<tr>
<td>Delivery Time vs. Quality</td>
<td>With the end of semester deadline, there may be some sacrifice of quality. However, we will endeavor to complete a system with no bugs by the completion date.</td>
</tr>
</tbody>
</table>

1.3 Definitions, acronyms, and abbreviations

**Actor:** any person who is going to use the system.

**Analysis:** act of exploring the problem domain to gain a better understanding of the problem.

**Authorization:** The granting of access to an entire server or files and directories on it.

Authorization can be restricted by criteria including host names.
**BITS Team:** Jeremy DeVries, Nour Farhat, Brennan Powers, Grant Tulacro, Nick Ramsey.

**Database:** A collection of data stored on a computer that is easy to manage and access.

**User:** Any person accessing the web application through a standard web browser interface.

**Web Application:** A software application that runs on a remote server.

**FR:** Functional Requirement

**NFR:** Non-Functional Requirement

**UC:** Use Cases

**HCT:** Hospital Comparison Tool.

**Landing page:** The page that contains a search bar and the interactive US State map.

**RAD:** Requirement Analysis Document.

**Result page:** The page that shows the user the results of their search.

**Hospital Dashboard:** The page that shows the hospital statistics and allows for search of more detailed information.

**HTTP:** Hypertext Transfer Protocol.

**TCP:** Transmission Control Protocol.

### 1.4 References

This document is the design of the system based on the functional requirements from the Requirement Analysis document (RAD) and the Project Inception Document.

### 1.4.1 Related Documentation

**Project Homepage:** Can be found at [https://www2.cs.siu.edu/~hmt/](https://www2.cs.siu.edu/~hmt/).

**Project Proposal:** Can be found on the projects Homepage.

**Requirement Analysis Document:** Can be found on the projects Homepage.
1.4.2 Software Architecture Constraints

Potentia has given a few constraints on the systems architecture. The system must be a 3-tier client-server-database application. This makes the most sense for the systems overall architecture. We also received some more specific requirements for the software architecture. The first is that the application be made using Django web framework. This is because it is what Potentia uses for their own web related systems and it will allow them to more easily work on the system in the future. Since we will not be providing maintenance on this system, it will fall to the Potentia Team to fix any bugs that appear or any changes they may want to make. The second architectural constraint Potentia gave us is to use PostgreSQL relational database system for the hospital information database. This is what the Potentia Team is familiar with and fits well with the kind of data we will be managing. The last constraint Potentia has given us is to use AngularJS. With the type of application, we are creating this fits our needs very well.

1.5 Overview

The Hospital Comparison Tool that we propose is a brand-new system with no predecessor available. If someone wanted to look at hospital data, they would need to download the archive and look through the CSV files themselves. Attempting to find the relevant data they are looking for can be very difficult due to the many different files and data fields. Our system will give the user an easy way to see several commonly desired data fields in an easily understandable way. The user would not need to download anything compared to the old method. The old method had no real search or data analysis ability while our application will allow for easy searching and comprehensive data analysis and comparison. Additionally, our system will offer options on how a hospital can improve their statistics using Potentia’s products.
2. Current software architecture

2.1 Overview

As mentioned above the system currently used is the raw database tables, and the Hospital Comparison Tool will be recreated from scratch therefore we will not mention previous software architecture.

3. Proposed software architecture

3.1 Overview

The proposed system will use the Django App framework which follow the Model-View-Controller (MVC) architecture, this architecture is split up into three different parts:

1. The Model which is the logical data structure behind the entire application and is represented by a database Postgres database.
2. The View is the user interface, which will be represented by HTML/CSS/AngularJS files.
3. The Controller is the middleman that connects the view and model together, meaning that it is the one passing data from the model to the view.
3.2 Subsystem decomposition

The following are subsystems identified within our system. These subsystems are to be developed by the BITS team and incorporated into the overall system.

- Database Management Engine:
- Analytic Engine
- Interactive Engine

**Database Management Engine:**

The database management engine will incorporate a PostgreSQL database connected to the Django framework through migrations to be utilized inside the web application processes. The database will be where all hospital data is to be stored and maintained. Access to the database will be restricted to self-update scripts and user query. Scripts in the Interactive engine are to be utilized after a search function so that the correct data can be found inside the database and brought to the Analytic engine for computation. Some key functions of the Database management engine include:

- Manage database table utilization from other subsystems
  - Push correct analytical data to the Analytic engine.
  - Push correct display data to the Interactive engine for display
- Update database tables through scripts utilizing the csv files.
  - Utilize scripts to convert CSV files to PostgreSQL table format.
  - Utilize scripts to update existing tables with new CSV files.
- Keep data secure in a read only status to users.

**Analytic Engine:**

The application will use Queries and the NumPy python package to manipulate and analyze the provided data by the Database Management Engine. In order to provide the user with an interactive summary of the results. The analytics engine is responsible for querying the needed data from the tables and hand off the results to the Interactive engine which will display the results for to the user.
Below are the main tasks the Analytic Engine is responsible for:

- Retrieve the national and state averages for multiple data points. Such as:
  - ER Wait Time
  - Hospital Rating
  - Median Length of Stay
  - Median time between decision and admission
  - Median length of stay
  - Percent of patients left without being seen
- Retrieve national and state median values for multiple data points, such as:
  - Median Length of Stay
  - Median time between decision and admission
  - Median length of stay
- Retrieve hospital ratings using multiple data points
  - Staff responsiveness rating
  - Efficient use of medical imaging.
- Analyze Hospital performance for data points of interest, such as:
  - Hospital Overall Rating

Interactive Engine:

The user engine will handle the connection between the user and the system and allow the user to interact with the system in various ways. Some of the functions that the User engine will allow are

- Search functionality (FR1 from Requirement analysis document “RAD”)
  - Search function which queries the database utilizing the “DME” (database management engine)
- User Interactions (FR9 from Requirement analysis document “RAD”)
  - Interactions with web application UI.
- Representing data in a consumable graphical manner.
- Representing patient timeline using multiple points of data.
3.2.2 Hardware/software mapping

Hardware configuration of the system will be using a server machine that would host the database and the structure of the system, and a client machine for the browser.

The different Nodes will be connecting through TCP/IP protocols. Off-the-shelf components, Mapael, Django, AngularJS, PostgreSQL are to be deployed at the client side’s node.

Fig 3.2.3 Software Architecture diagram for Hospital Comparison tool
### 3.2.3 Relational Database Table Diagram

<table>
<thead>
<tr>
<th>DATABASE: PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>The PostgreSQL database will be responsible for maintaining and storing hospital data. Migrations in the Django Server will serve as the connection between the web application’s framework and the relational PostgreSQL database as well as giving the web application the ability the query and pull data to the front end (user view).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Django Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Django software will act as the server platform and framework to base all connections between components. Django allows connections to databases using migrations. Django is also the base server that will compile and run the web application on the web server. This framework contains subsections known as “apps”, html, CSS, and scripts will be stored and ran in these subsections.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AngularJS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AngularJS is a web application framework used to connect JavaScript based code to HTML (hypertext markup language) for easier programming of interactivity inside the web application.</td>
</tr>
</tbody>
</table>
3.2.4 Deployment Diagram

<table>
<thead>
<tr>
<th>Technology</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Django Framework</td>
<td>Off the shelf server framework used to house and connect program code and database connectivity</td>
</tr>
<tr>
<td>AngularJS</td>
<td>Open source software for easier programming of interactivity inside web applications</td>
</tr>
<tr>
<td>Mapael</td>
<td>A plugin from the jQuery software used to display dynamic vector maps used for the interactive map</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>An off the shelf Object-Relational database system used for storage and access of hospital data for the Web application system.</td>
</tr>
<tr>
<td>Browser:</td>
<td>Extra software ran on user machines to access the world wide web to run the web application, used for normal access of the system and testing of the web application.</td>
</tr>
<tr>
<td>- Safari</td>
<td></td>
</tr>
<tr>
<td>- Chrome</td>
<td></td>
</tr>
<tr>
<td>- Firefox</td>
<td></td>
</tr>
</tbody>
</table>
3.2.5 Persistent data management

The publicly available hospital information that our system uses updates once every 4 months. Therefore, this data must be scraped on a 4-month rotation and the database updated. We will be using PostgreSQL to manage the data. Our database will be completely relational. If we were to use objects, the complexity of each object would make things very difficult to manage.

3.2.6 Access control and security

The system should not allow any user to edit the database. No system is being created to allow an admin access to the system. All changes to the system will need to be implemented by a developer.

3.2.7 Global software control

The system is susceptible to synchronization, Since the user of the HCT is to have read permission only. The web server on which our application is to be deployed needs to just handle more than one user request at a time. This will be taken care of using multithreading and multiprocessing.

HTTP(Hypertext Transfer Protocol ) is used to structure requests and responses over the internet between user and the server. The transfer of resources happens using TCP (Transmission Control Protocol). Where the TCP manages the channels between the browser and the server. 
3.2.8 Boundary conditions

The system should be started up on a dedicated server machine. The system is designed to run continuously. Restarts should only be performed by the server administrator. Any changes required to the server should only be performed by the server administrator. Shutdowns should be rare and are not required.

4. Subsystem services

Database Management Engine:
Incorporate a PostgreSQL database connected to the Django framework through migrations to be utilized inside the web application processes.
Access to the database will be restricted to self-update scripts and user query.
Scripts in the Interactive engine are to be utilized after a search function so that the correct data can be found inside the database and brought to the Analytic engine for computation.

Analytic Engine:
The application will use Queries and the NumPy python package to manipulate and analyze the provided data by the Database Management Engine. In order to provide the user with an
interactive summary of the results. The analytics engine is responsible for querying the needed
data from the tables and hand off the results to the Interactive engine which will display the
results for to the user.

**Interactive Engine:**
The user engine will handle the connection between the user and the system and allow the user to
interact with the system in various ways. Some of the functions that the User engine will allow
are

**Mapeal:** Mapael is a jQuery plugin based on raphael.js that allows the display of dynamic vector
maps. This will provide our system with an interactive map of the United States, that is to be
used in the Landing page, and provide an alternative searching and data summarizing ability of
the system.

**Django:** Python based web framework, its primary goal is to ease the creation of the database.
This framework stress reusability and "pluggability" of components.

**AngularJS:** JavaScript web app framework. It aims to simplify the development and the testing
of the application. And provides sub-services or functions that are built in services or custom
design ones that are limited to AngularJS application.

**PostgreSQL:** An object-relational database management system (ORDBMS) with an emphasis
on extensibility and standards compliance. It can handle workloads ranging from small single-
machine applications to large Internet-facing applications. It is available on different Operating
systems such as Microsoft Windows, Linux and Mac.