System Design Document

Project: OuRsiders Social Web Application
Purpose of the document

This system design document is meant to provide important information pertaining to the development of OuRsider's social web based application such as processes, data structures, subsystems, and software/hardware components necessary in the implementation of this application.
## Table of Contents

1. Introduction ................................................................. 4  
   1.1 Purpose of the System ........................................ 4  
   1.2 Design goals ......................................................... 4  
   1.3 Definitions, acronyms, and abbreviations .............. 8  
   1.4 References .......................................................... 9  
2. Current software architecture ....................................... 9  
   1.4.1 Overview ......................................................... 9  
3. Proposed software architecture ..................................... 9  
   1.4.2 Overview ......................................................... 9  
   1.4.3 Subsystem decomposition .................................... 9  
   1.4.4 Hardware/software mapping ................................. 13  
   1.4.5 Persistent data management ................................. 14  
   1.4.6 Access control and security ................................. 15  
   1.4.7 Global software control ..................................... 16  
   1.4.8 Boundary conditions ......................................... 17  
4. Subsystem Services ..................................................... 17
About the System Design Document

<table>
<thead>
<tr>
<th>Version No.</th>
<th>Name(Title)</th>
<th>Author(s)</th>
<th>Date</th>
<th>Authorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>SDD Report</td>
<td>Daja Dunbar, Axel Corey, Payton Dudley, Alejandro Narvaez-Colon</td>
<td>2/7/2019</td>
<td></td>
</tr>
</tbody>
</table>

Document Properties

<table>
<thead>
<tr>
<th>Item</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Title</td>
<td>SDD Report</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Daja Dunbar, Axel Corey, Payton Dudley, Alejandro Narvaez-Colon</td>
</tr>
<tr>
<td>Creation Date</td>
<td>2/7/2019</td>
</tr>
<tr>
<td>Last Updated</td>
<td>2/7/2019</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Purpose of the System

The purpose of this system is to help parents connect their children with children who have similar disabilities and interests, controlled solely by their parents. This application aims to help initiate social interactions events for children ages three to eighteen years old by using an algorithm to match users based on similarities of criteria created by each child’s parent/guardian.

1.2 Design Goals

The purpose of outlining the design goals of this web based application is to identify the qualities that are the focus of the system. These goals are derived from the non-functional requirements outlined in the Requirement Analysis document.

These goals can be divided into the following:

- Performance
- Dependability
- Cost
- Maintenance
- End user criteria

Performance Criteria

The system must be capable of handling at least 50 users accessing the web server at any given time, as well as responding to user requests (loading forms, accessing the database, running the matching algorithm) within fifteen seconds at most. The system shall also have high throughput, which is the amount of tasks the system can accomplish in a certain amount of time.
**Dependability Criteria**

The criteria for dependability can be divided into the following:

- **Robustness:** The system must be capable of surviving invalid user inputs, such as: incorrect password, creating an account with an email that already exists in the system, entering an invalid age for a child, etc. The system shall display error notifications pertaining to the invalid user input.

- **Reliability:** It is important that the system displays correct results to the user. This is pertinent for functions such as ensuring that the matching algorithm matches a child with a similar match, having the confirmed match list portray successful matches, and having the matching criteria and information of each child be updated when edited.

- **Availability:** The system should be available to a user at any given time, for optimal user friendliness.

- **Fault Tolerance:** The system should have decent fault tolerance, where it should anticipate the failure of a component and upon such an event preserve data and keep it safe.

- **Security:** The main functions of security will be handled by limiting inputs by validated users. This will help prevent attacks such as sql injection. We will also require the user to have a password of a certain length and complexity to avoid brute force attacks. We will use the ‘devise’ ruby gem to encrypt passwords and other sensitive information.

- **Safety:** The system will be completely parent controlled, and views of other user’s profiles will be limited based on whether they are matched or not. Some views will be limited only to that user, such as names of their children and their respective matching criteria and personal information.

**Cost Criteria**

The criteria for cost can be divided into the following:
- **Development Cost**: The most significant costs of this system are time limitations and developing with new technologies.

- **Deployment Cost**: An objective of this system is to have the user be able to quickly pick up and use this application, which is achievable by having an easy to understand user interface and by having concise descriptions. The deployment cost should be minimal ($5 per month) when using the Digital Ocean service.

- **Upgrade Cost**: Transferring or translating data from the current system to a new system should be low cost due to the relatively low complexity of the current database, which should offer backwards compatibility.

- **Maintenance Cost**: There may be a medium maintenance cost due to relative inexperience with the technologies involved, but should/when bugs arise we will fix them to the best of our abilities. We use components that are supported (Ruby on Rails, MySQL) and well maintained, so moving the data from a database to a newer version should not be a trial. The gem system of Ruby on Rails allows us to easily lock a plugin to a certain version, and even update any gem easily with the ‘bundle install’ command if desired.

- **Administration Cost**: The system should not have a high administration cost, due to operating from only one server and having limited user input should reduce problems that would arise.

**Maintenance**

The criteria for maintenance can be divided into the following:

- **Extensibility**: This system should not be difficult to add new functionalities to. Possible added functionalities could include the introduction of a secondary guardian, a mobile version, and in-app communication.
• **Modifiability:** Building the system as loosely coupled as possible would increase modifiability, so functionalities may be changed with as little impact between subsystems as possible.

• **Adaptability:** This system is not adaptable, as porting the current form developed in RoR to another application platform would be difficult, as RoR has a unique approach to web development with its own gems and file structure. The system would need to be changed greatly to be ported to another application platform.

• **Portability:** The system shall be portable, as there is a desire to have a mobile app version in the future.

• **Readability:** The system will be readable, due in part to the way RoR divides all the different aspects of the system into different subsections. This results in the model, configuration, views, and controllers being separate from each other which will aid in the reading of the program.

• **Traceability of Requirements:** We will have the requirements of the system be traceable by tracking/completing our tasks for each functional requirement and progressing in a fashion that allows us to tackle larger requirements faster. By documenting the progress of the system’s development, it will be easier to determine if and how the requirements were met.

**End User Criteria**

The criteria for the end user can be divided into utility, which is how well the system supports the work of the user, and usability, which is how easy the system is to use.

**Utility:** The system shall support the work of the user by ensuring that the information supplied about each child will be used to reasonably derive a viable match. The provided input will be filtered to ensure the matching algorithm functions properly and makes the user experience as easy as possible.
Usability: The system will have high usability by providing a simple and intuitive user interface which requires little prior information to be useable. Questions for matching criteria shall be straightforward and the system will do the heavy lifting by providing viable matches to the user.

1.3 Definitions, acronyms, and abbreviations

**Commercial off the shelf components**: adjective that describes software or hardware products that are ready-made and available for sale to the general public.

**Logical Architecture**: structural design that gives as much details as possible without constraining the architecture to a particular technology or environment.

**Applications**: software that is primary designed to be used by people

**Business Rules**: a way to represent business policies, processes, and procedures as a series of conditional statements that can be represented as a decision tree of text.

**Algorithm**: series of steps for executing a task, solving a problem, or performing a calculation.

**Physical Architecture**: gives enough detail to implement the architecture on a technology.

**Database**: type of software for accessing, updating, and managing organized data.

**Database index**: optional data structure that can be created for a column or list of columns to speed data access

**Data processing**: series of operations that use information to produce a result.

**Coupling**: Low coupling results in fewer dependencies between each subsystem of the entire system.

**Infrastructure**: system of management applications and controls that is used to manage a power grid.
**Systems**: a class of software that provides services to applications and other systems

**RoR**: Ruby on Rails, a web application framework

**MySQL**: An open source relational database management system

**SQL injection**: An attack caused by user input being treated as sql code, which can result in manipulation and damaging of the database system

**Ruby gem/gem**: a gem is a module/library that can be installed and used in a RoR application.

**Db-** database

**1.4 References**

This system design document is reliant on information provided by the requirement analysis document.

**1.5 Overview**

The system covered is a web-based application software. The system shall perform a clustering algorithm across multiple users’ children based on supplied matching criteria. The purpose is to build a network of matches that are derived from confirmed matches of two users after being introduced by the algorithm.

**2. Current Software Architecture**

**1.4.1 Overview**

There is no prior version of this web based application, so there is no current software architecture in place. DAPA Production’s plans to have an active web-based application that user’s can use to

**3. Proposed Software Architecture**

**1.4.2 Overview**

OuRsiders didn’t have a previous matching algorithm nor web based application for the proposed system. Dapa Productions has agreed to construct both the web based
application along with the matching algorithm to help match children based on their interest, traits, talents, hobbies and disabilities.

**Logical Architecture**

The logical software architecture of the system will be divided into different layers:

- **Presentation Layer**
  - Information views
    - User is able to interact with the database without actually interacting with the database, for example saving their profile, child info etc.

- **Business Logic**
  - This will handle the business rules
    - If a user doesn’t have an account they aren’t able to use the web-based application.
    - If a user wants to block a user to no longer have contact they are allowed to
    - A user’s information will be encrypted so another user can’t access it
    - When a user decides to use DAPA Production’s web-based application they agree to adhere to ourSiders privacy policy and terms and conditions
    - User’s are not allowed to post personal information about their child, including but not limited to pictures, precise location, school.

- **Algorithm**
  - K-means clustering will help with matching users based on their criteria.

- **Data processing**
  - Verifying a user actually has an account

- **Data access Layer**
  - Database
- Database index, for the event the parent needs to find their account by email
- This will hold the user's information along with the child's information, and their location
  - Services
    - This will match the user with a different user in their location, with similar likes, dislikes, hobbies, traits, diagnosis, interests and talents.
  - Systems
    - Infrastructure to help manage the users data

**Physical Software Architecture:**

This will describe how the different components and nodes are arranged within the system. MySQL will be used to control the database. We will be using Ruby on Rails for the actual web-based application, while bootstrap will be used for the front-end, so the interactive part the user will see.

1.4.3 Subsystem decomposition

The following are subsystems identified within our system. Our system will also require the services provided by different commercial off-the-shelf components. We are using the following commercial off-the-shelf components:

1. MySQL
2. Ruby on Rails
3. Bootstrap
**Match Subsystem:**

This service provided by this system will be a child’s traits, hobbies, diagnosis, talents and interest, are stored until another user inputs similar data that will generate a match.

**Permission Handling Subsystem:**

This subsystem is responsible for handling permission of users. It interacts with the database subsystem. This system will make sure that the user of the system are verified to be eligible. This will provide security to each user as a service. This system will require the service provided by the database system.

**Account Handling Subsystem:**

The Parent subsystem will provide users with the option to add, delete a child, create and edit their profile and location. This will also allow the parent to edit any traits, hobbies, diagnosis, talents, and interests. This system will interact with the authentication and database subsystem. It will also be responsible for handling the transfer from an active account to a deactivated user account. It interacts with the database subsystem. This will also include action by the parent. Registration of users

- Adding a child
- Update information of user/child/location

This system will require services from other subsystems such as access control and authentication subsystems which are discussed further below, these will also interact with the database

**Restructured Subsystem Specification:**

The following figure illustrates the relationship among subsystems of ouRsiders web-based application. All subsystems need to read/write/update the database
subsystem. They also have interaction with the commercial off-the-shelf components described above.

Figure 1. Software architecture of OuRsiders system

1.4.4 Hardware/software mapping

The subsystems will be mapped to the components and the components will be mapped to different nodes. The hardware of the system will be consisted of a client machine and a web/application server. The browsers will include FireFox and Chrome which will be used by the client machine, which may includes a home pc, a mobile phone, or a tablet. Our system will have a Database which will be the ouRsiders db component which will be run with MySQL.
### 1.4.5 Persistent data management

<table>
<thead>
<tr>
<th>Table</th>
<th>to</th>
<th>Table</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>to</td>
<td>Child</td>
<td>One-to-Many</td>
</tr>
<tr>
<td>Child</td>
<td>to</td>
<td>Hobbies</td>
<td>One-to-One</td>
</tr>
<tr>
<td>Child</td>
<td>to</td>
<td>Traits</td>
<td>One-to-One</td>
</tr>
<tr>
<td>Child</td>
<td>to</td>
<td>Preferences</td>
<td>One-to-One</td>
</tr>
<tr>
<td>Child</td>
<td>to</td>
<td>Diagnoses</td>
<td>One-to-One</td>
</tr>
<tr>
<td>Child</td>
<td>to</td>
<td>Matches</td>
<td>One-to-Many</td>
</tr>
<tr>
<td>Child</td>
<td>to</td>
<td>Pending_Matches</td>
<td>One-to-Many</td>
</tr>
</tbody>
</table>

*Database Table Relationships*
Database Schema (arrow represents foreign key)
1.4.6 Access control and security

Below an access control matrix that shows the different actors and their privileges within the application.

*Table 1: Access control matrix*

<table>
<thead>
<tr>
<th>Actor</th>
<th>Privilege</th>
</tr>
</thead>
</table>
| User/Parent    | ➢ Registers child information  
                 | ➢ Chooses matches  
                 | ➢ Blocks unwanted matches |
| Administrator  | ➢ May update database items                    |

1.4.7 Global software control

- **Access control**
  - Check unauthorized access by guaranteeing that only the current user’s session is that one able to access the page, use cancan gem

- **Authentication**
  - Use gem devise for authentication since it hashes the password using bcrypt

- **Correctness of system design**
  - Filter passwords so they aren’t logged
  - Authenticity tokens on forms
  - Strong parameters to require user data and only allow modification of name
  - Rack Attack gem to limit the amount of login attempts
  - Use HTTPS for all requests
  - Patch discovered vulnerabilities as soon as possible
  - Bundler-audit gem to check for vulnerabilities in gems Gemfile.lock
  - Brakeman gem for code security analysis (XSS, DoS, SQL Injection, etc.)
  - Separate url for admin portal
  - Forgotten password message should be the same regardless of existing account or not
1.4.8 Boundary conditions

The web archive will be compiled when the first HTTP request arrives. It will then be precompiled for future requests making response time faster. The system initialization will start when the web server, MySQL is up and running. The system shutdown will happen when administrators decide to close the web server. Administrators will be able to start the server, configure the server, and shut down the server.

4. Subsystem services

1.1 Purpose of the system

Our off-the-shelf components are:

- Ruby on Rails: Model View Controller software that depends on gems that enable different functionalities
- Bootstrap: Grid-based, most-popular front-end component library makes it highly compatible, easy to use and good for mobile
• MySQL: A Database Server on which the database of our system is to be deployed.
• Chrome: Browser in which users will interfere with the presentation layer of the system.

Administration Subsystem
Using the cancan gem:
  • It assigns roles to particular users for registering, adding, deleting and updating database data

Match Subsystem:
The match system will run a clustering algorithm with criteria gathered from the user, which is then used to find other children with similar criteria, upon which they will appear in a list of viable matches on the user’s homepage. A user can choose to view a viable matches profile, and confirm the match. Upon the match confirming the match on their end as well, a successful match will be made, and the match will be added to each respective users match list.

Permission Handling Subsystem:
The permission handling system is meant to validate users and ensure that they have the privileges to make submissions to the MySQL database, as well as filtering input to ensure that there is no invalid or potentially dangerous entries in the database that may affect the security of the system.

Account Handling Subsystem:
This will allow the user to decide whether or not they want to deactivate their account. The user also has the ability to delete their account, but the deactivation option will allow for the user to temporarily be removed from the matching cluster algorithm, until the user decides to reactivate their account, putting them back into the cluster algorithm.
Using devise gem:

- Authentication of users through email.
- Full registration of users.
- Update the information of users

A user can:

- Add a child from their account
- Delete a child from their account
- Create and edit their profile
- Edit their child’s profile
- Edit their child’s matching criteria