Doggy Adoption Network (DAN) SDS

Software Design Specifications

Elizabeth Bailey, Evan Pariser, Nathan Malamud, Jack White

1.0 Revision History

Date	Author Initials	Description
5.8.22	NM	Created draft/outline
	NM, EP	Sections 3-5
5.16.22	EB	Filling out sections 2-4
	EP	Wrote out database documentation
	JW	UML Sequence Diagrams, diagrams of architecture
6.6.22	EP	Updating SDS to fit final release

2.0 System Overview

The Dog Adoption Network (DAN) system is an application that assists users in finding a dog that will fit into their lifestyle and allows dog adoption centers to list their dogs based on behavior and specific needs. The system will be organized into a few major modules:

1. **GUI manager** - displays text, images, and graphics. The GUI manager prompts the user through questions about their lifestyle, living situation, and preferences. Following the initial question phase, the GUI manager displays dogs in the database that best fit their profile. Also takes user input to create and delete dog profiles in the database (admin mode only).
2. **User Profile Development/Quiz (UPD) Module -** contains the logic of the “quiz” to collect relevant information from the user to develop their profile.
3. **Filter Module** - uses methods that query the dog database to narrow down dogs.
4. **MySQL Database and Server -** stores dog profiles on a remote server.

3.0 Software Architecture

Figure 1 is an illustration that shows a logical view (bird's eye) of the modules and their interactions with each other. It is based on the MVC (Model-View-Controller) architecture, which is commonly used in software engineering settings.

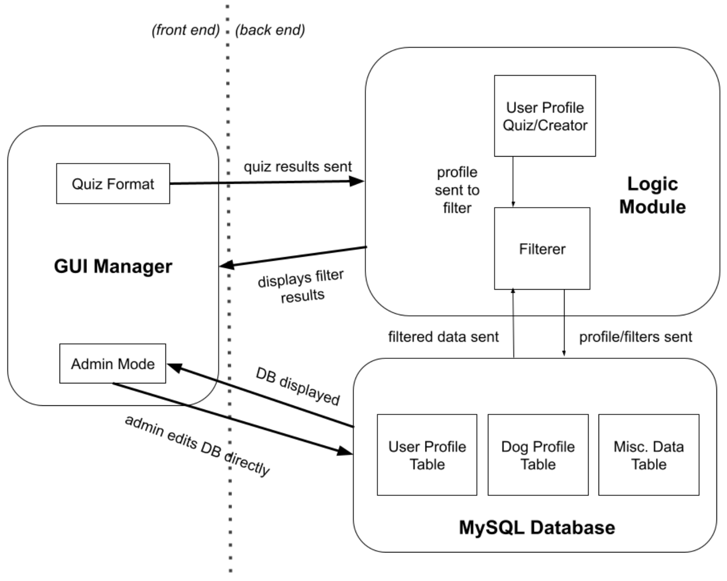

Figure 1. Illustration of software modules categorized into front and back end.

GUI Manager Module - the main purpose is to serve and guide the user through a quiz, and then display one or many dogs that meet the user's specifications. Any other user settings can be found here too. This module will call on the User Profile Development (UPD) Module to move the user through the quiz and develop their profile. This module will use the results from the user's responses to call appropriate methods from the Filter Module. We will implement the front end using Flask. We chose Flask because it is a lightweight framework for building web applications and uses python which we are all familiar with.

User Profile Development/ Quiz (UPD) Module - This module will contain the logic and content of the quiz that is used to develop the user profile, and translates user responses for use by the GUI manager to query the database appropriately. This module will also save the user profile information to the user database on the server.

Filter Module - This module will relay communication between the GUI (user), and the MySQL database. It will use methods that query the SQL database. These methods will be called from the GUI manager based on user input to the quiz module, and then the updated parameters will be sent to the database, which will then send back the requested list of dogs.

MySQL Database - The database we will be using will be MySQL.
There are four tables: `Dog` (containing attributes of each dog), `User` (containing user profile information), `Supplies,` and `Adoption_center`.

The `Dog` profile table can be edited directly in admin mode from the GUI Manager. The `User` profile table will be updated by an individual user (also from input via GUI Manager); users can create a profile and delete a profile they are logged into. Any user profile can also be deleted in admin mode.

The `User,Adoption_center,and Supplies` tables are not used in the current version, but still exist in case we want to expand our program's functionality.

We are choosing MySQL because of Evan's experience, and because it is relational. Relational databases are very easy to scale, and if we want to add new tables, all that needs to be done is to find their relation to the rest of the tables. The database will be hosted on the University of Oregon ix-dev server.

Architecture Rationale
Our system architecture is designed to *prioritize performance*. Most of the components will be hosted locally within the application, and calls to the server are only made 1) to populate the window that displays the dog options currently in the database and 2) when a user logs into a pre-existing account.

This system is based on an MVC (Model-View-Controller) architecture pattern, with some modules broken down additionally to support the system-specific processing of user input for profile development. In our design, the *model* is the database, the *view* is GUI Manager, and the *controller* is composed of two sub-modules: the UPD Module and Filter Module.

4.0 Software Modules

4.1 Graphic User Interface (GUI) Manager
This module is the *view* component, guiding the user through the use of the dynamic pages of the application. Depending on the use case scenario, the sequence of windows the user will interact with changes, but the general states of the GUI are described below.

4.1.1 Landing Page
When the application is first booted up, a landing page will give the user the option to 1) create a new profile, or 2) continue without a profile to view all dogs.

4.1.2 User Profile Development Sequence (Quiz)
The *User Profile Development sequence* is a series of windows that will be driven by the logic stored in the UPD Module. The user will be guided through a series of multiple-choice questions, and saved locally upon completion of the sequence. The user will be required to answer all of the questions in the quiz in order to complete their profile.

4.1.3 Adoptable Dog Display

This view will display the dog profiles from the database. A dog can be selected, bringing the user to the Needs view (see **section 4.1.4**)
If this is viewed from a signed-in user, the dogs will be ordered based on their compatibility with the user's profile (and dogs that are fully incompatible with the user will not be shown at all or the user will have an option to show/ hide those dogs).
If the user is not signed into a completed profile, the application will simply list all of the available dogs in an arbitrary order.

4.1.4 Needs View
When a user selects a dog from the previous view, the application will display the needs and details of that specific dog's profile, and what needs we predict they will be able to easily meet based on their user profile.

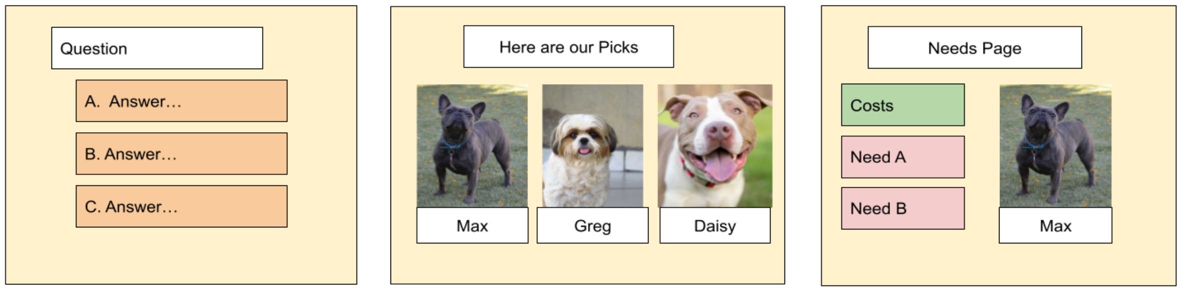

Figure 2. Illustration of the sequence of window views described in **sections 4.1.2 - 4.1.4.**

4.1.5 Admin Mode View
Admins can directly interact with the MySQL database and view, delete, and edit dog profiles. This is all done inside the file dogdb.py. Examples of the functions applications are shown at the bottom and commented out.

4.2 User Profile Development/Quiz (UPD) Module
This module stores the logic and content for the multiple-choice questions a user will answer to complete their user profile. The questions will be displayed in sections including Occupation, Physical Ability, Living Situation, and Finance.

4.3 Filter Module
This module will contain methods that make calls to the dog profile table in the database. Based on the arguments, the methods will return a sorted list of dog profiles in a format that can be used to display information in the adoptable dog display view in the UI. The module uses calls to the MySQL database to receive the dog profiles, but does not directly interact with the database.

4.4 mySQL Database and Server

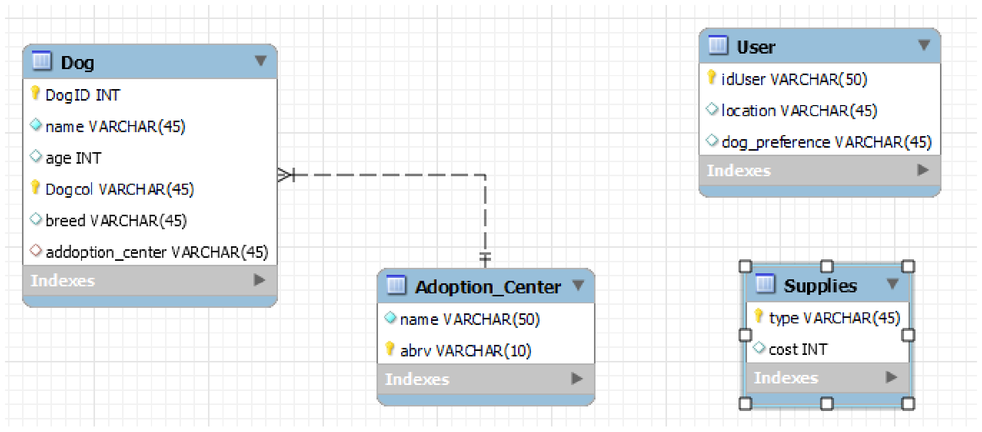

Figure 3 Crowsfoot model of the database. Tables described in 4.4

There are four tables: Dog, User, Supplies, and Adoption_center.
4.4.1 Dog This table contains all dogs and their respective attributes like breed, age, and medical info.
4.4.2 User This table contains all user info and preferences collected from the UPD Module
4.4.3 Supplies This table contains the cost for all food, medicine, or other supplies an owner might need for their dog.
4.4.4 Adoption_center This table contains all adoption centers in our system and their basic info.

5.0 Dynamic Models of Use Cases

User inputs data (e.g. quiz answers) via the GUI Manager. GUI Manager relays data to the Document Manager, which translates the input into an SQL database query (sent to the database). Prior to any module interaction, the IX server must instantiate the SQL database.

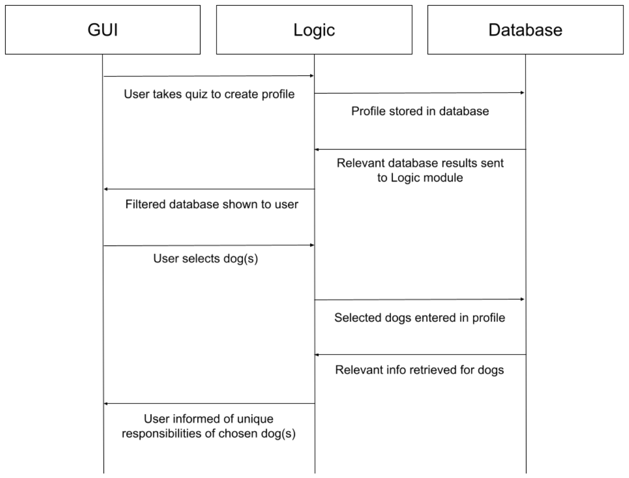
GUI
Logic
Database
User takes quiz to create profile
Profile stored in database
Relevant database results sent to Logic module
Filtered database shown to user
User selects dog(s)
Selected dogs entered in profile
Relevant info retrieved for dogs
User informed of unique responsibilities of chosen dog(s)

Figure 4 Dynamic UML Sequence diagram of Inexperienced User Case.

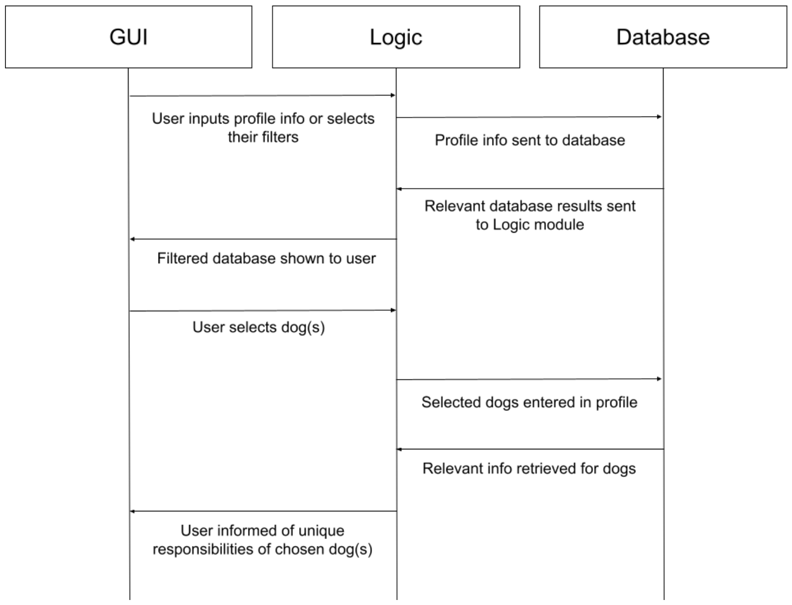
GUI
Logic
Database
User inputs profile info or selects their filters
Profile info sent to database
Relevant database results sent to Logic module
Filtered database shown to user
User selects dog(s)
Selected dogs entered in profile
Relevant info retrieved for dogs
User informed of unique responsibilities of chosen dog(s)

Figure 5 Dynamic UML Sequence diagram of Experienced User Case.

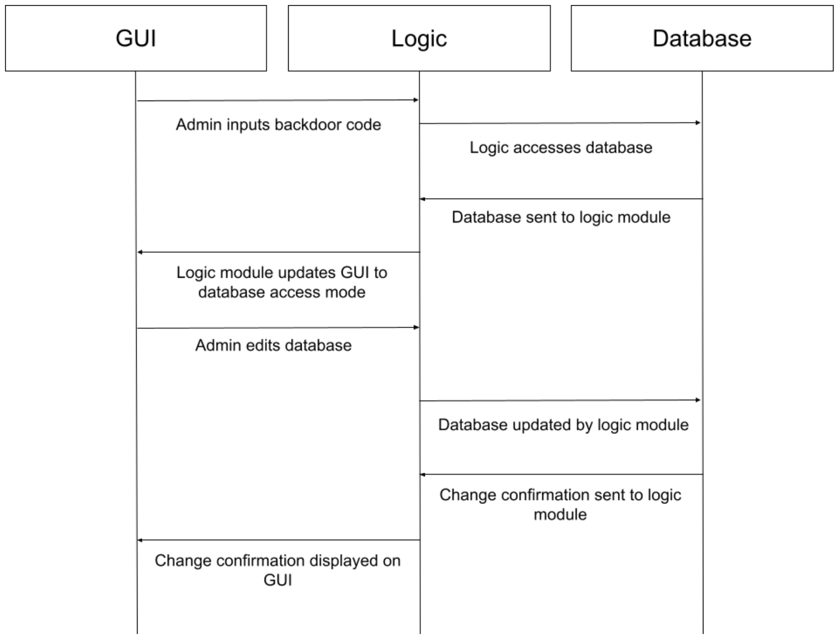
GUI
Logic
Database
Admin inputs backdoor code
Logic accesses database
Database sent to logic module
Logic module updates GUI to database access mode
Admin edits database
Database updated by logic module
Change confirmation sent to logic module
Change confirmation displayed on GUI

Figure 6 Dynamic UML Sequence diagram of Admin User Case.