A Software Design Specification Template

by Brad Appleton <brad@bradapp.net>
http://www.bradapp.net

Adapted for LAFS by Elliott McCrory, 6/13/2023 1:10:00 AM

Table of Contents

1. Introduction

The following is an attempt to put together a complete, yet reasonably flexible template for the specification of software designs. Wherever possible, I have tried to provide guidelines (instead of prescribing requirements) for the contents of various sections and subsections of the document. Some may prefer to require more detailed subsections of a particular section, choosing one or more of the subsection topics from the list of guidelines provided. In this sense, this document is really a template for a template.

In devising this template, I have gleaned information from many sources, including various texts on Software Engineering (Pressman, Sommerville, and Van Vliet), Object-Oriented Development (Booch, Rumbaugh, Berard, and Wirfs-Brock), various SEI reports, DoD-Std and Mil-Std documentation requirements (2167/2167A), and IEEE documentation standards (particularly IEEE-1016 for software designs, and IEEE-830 for software requirements). I have made every effort not to assume or impose a particular software development methodology or paradigm, and to place more emphasis on content than on format.

It is my desire that a completed software design specification meet the following criteria:

- It should be able to adequately serve as training material for new project members, imparting to them enough information and understanding about the project implementation, so that they are able to understand what is being said in design meetings, and won't feel as if they are drowning when they are first asked to create or modify source code.

- It should serve as "objective evidence" that the designers and/or implementers are following through on their commitment to implement the functionality described in the requirements specification.

- It needs to be as detailed as possible, while at the same time not imposing too much of a burden on the designers and/or implementers that it becomes overly difficult to create or maintain.

Please note that there are no sections in this document for describing administrative or business duties, or for proposing plans or schedules for testing or development. The sections in this document are concerned solely with the design of the software. While there are places in this document where it is appropriate to discuss the effects of such plans on the software design, it is this author's opinion that most of the details concerning such plans belong in one or more separate documents.

1.1. Document Outline

Here is the outline of the proposed template for software design specifications. Please note that many parts of the document may be extracted automatically from other sources and/or may be contained in other, smaller documents. What follows is just one suggested outline format to use when attempting to present the architecture and design of the entire system as one single document. This by no means implies that it literally is a single document (that would not be my personal preference):

- Introduction
- System Overview
- Design Considerations
    - Assumptions and Dependencies
    - General Constraints
    - Goals and Guidelines
    - Development Methods
- Architectural Strategies
    - strategy-1 name or description
    - strategy-2 name or description
    - ...
- System Architecture
    - component-1 name or description
    - component-2 name or description
    - ...
- Policies and Tactics
    - policy/tactic-1 name or description
    - policy/tactic-2 name or description
    - ...
- Detailed System Design
    - module-1 name or description
    - module-2 name or description
    - ...
- Glossary
- Bibliography

The above outline is by no means exclusive. A particular numbering scheme is not necessarily required and you are more than welcome to add your own sections or subsections where you feel they are appropriate. In particular you may wish to move the bibliography and glossary to the beginning of the document instead of placing them at the end.

The same template is intended to be used for both high-level design and low-level design. The design document used for high-level design is a "living document" in that it gradually evolves to include low-level design details (although perhaps the "Detailed Design" section may not yet be appropriate at the high-level design phase).

The ordering of the sections in this document attempts to correspond to the order in which issues are addressed and in which decisions are made during the actual design process. Of course it is understood that software designs frequently (and often fortunately) don't always proceed in this order (or in any linear, or even predictable order). However, it is useful for the purpose of comprehending the design of the system to present them as if they did. Frequently, one of the best ways to document a project's design is to keep a detailed project journal, log, or diary of issues as they are mulled over and bandied about and to record the decisions made (and the reasons why) in the journal. Unfortunately, the journal format is not usually organized the way most people would like it for a formal review. In such cases, for the purpose of review, the journal can be condensed and/or portions of it extracted and reorganized according to this outline. However, if this is done then you need to choose whether to update and maintain the design document in the journal format, or the formal review format. It is not advisable to try and maintain the design document in both formats. (If you have an automated method of converting the journal into a formal document, then this problem is solved.)

1.2. Document Description

Here is the description of the contents (by section and subsection) of the proposed template for software design specifications:

1.2.1. Introduction

1.2.1.1 Purpose

The document covers the criteria to develop a web scraping tool to extract accurate and up-to-date doctor data from various medical websites and databases to support healthcare services. Patients, for example, may search for doctors at any time and seek items like ratings, pricing, experiences, and so on. This data can then be used for many purposes, such as medical research, marketing, and analysis.

1.2.1.2 Product Scope

The final product is meant to aggregate information on doctors from several reliable medical/healthcare websites and to enable patients to choose doctors by searching for certain criteria such as reviews, costs, experiences, and so on.

1.2.1.3 Intended Audience and Reading Suggestions

This SRS document is intended for product designers, developers, marketing personnel, and product testers. This paper will provide them with a thorough grasp of the software requirements, allowing them to design, sell, and test a user-friendly solution at the end. We recommend reading the paper from beginning to end to acquire a thorough understanding of the final product requirements.

1.2.1.4 References

• IEEE standard 610.12-1990 IEEE standard Glossary of Software engineering terminology

• IEEE standard 730-1998 IEEE standard for Software Quality Assurance plans

• IEEE standard 830-1990 IEEE Recommended Practice for Software requirements specification.

1.2.1.5 Document Conventions

This document has been typed using 'Times' font, with a font size of 12 and line spacing of 1.5. The document follows standard MLA format.

1.2.1.6 Summary of Project

A web-based system to extract and collect data about doctors from various health sites/news/blogs. The data to be collected may include doctors' names, contact information, the doctors' area of expertise (including main / sub medical specialties), comments/ratings made by his patients with its date info, diseases he has previously diagnosed/treated, institutions/hospitals/clinics he works/worked for information and any other relevant information available on the website or database, to make it easy for patient to search about what they need.

1.2.2. System Overview

We plan to develop a develop a scraping tool for extracting and gathering doctor-related information from various health-related websites/news/blogs. Information's such as doctors' names, contact information, the doctor's area of expertise (including main / sub medical specialties), comments/ratings made by his patients with date information, diseases he has previously diagnosed/treated, institutions/hospitals/clinics he works/worked for information, and any other relevant information available on the website or database, to allow patients to easily search for what they need.

2. Design Considerations

2.1. Assumptions and Dependencies

- Related software or hardware: The product depend on specific web scraping tools or libraries to function properly. Additionally, it may assume that users have a stable internet connection and adequate computing resources to run the product.

- Operating systems: The product is designed to work on specific operating systems, such as Windows, Mac OS, or Linux. It may also require specific versions of these operating systems to function properly.

- End-user characteristics: The product assume that users have a basic understanding of web scraping concepts and techniques. It may also assume that

users have access to the necessary data sources and permissions to scrape the desired data.

- Possible and/or probable changes in functionality: The product assume that changes in the data sources or website structures will be rare and can be easily accommodated. It may also assume that any changes to the product functionality or features can be communicated to users effectively and without disrupting their workflows.

2.2. General Constraints

There are several global limitations or constraints that may have a significant impact on the design of the system's software:

- Hardware or software environment: The system's software design must be compatible with the hardware and software environment on which it will run. This includes compatibility with the operating system, web server, and database software.

- End-user environment: The software design must consider the end-user environment and the resources that will be available. This includes considerations such as the type of device being used, screen size, and input capabilities.

- Availability or volatility of resources: The software design must consider the availability and volatility of resources, including memory and storage capacity.

- Standards compliance: The software design must comply with relevant industry standards and best practices to ensure compatibility and interoperability with other systems.

- Interoperability requirements: The software design must consider interoperability requirements with other systems and platforms.

2.3. Goals and Guidelines

Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system's software. Such goals might be:

- The KISS principle ("Keep it simple stupid!")
- Emphasis on speed versus memory use
- working, looking, or "feeling" like an existing product

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.

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.

3. Architectural Strategies

- Use of programming languages and frameworks: the system may be built using **NodeJS** for the backend, **HTML-CSS-JS** for the frontend, **Python** and **Beautifulsoap** library for the scraper app, and **MySQL** for the database management system.

- Reuse of existing software components: The system may incorporate existing software components, such as open-source libraries (Beautifulsoap**)**, to implement various parts and features of the system. This can save development time and effort and reduce the risk of errors and bugs.

- Future plans for extending or enhancing the software: The system may be designed with future plans for extending or enhancing its features and functionality. This could involve designing the system to be modular and scalable, allowing for new components to be added or existing ones to be modified without requiring significant changes to the system architecture.

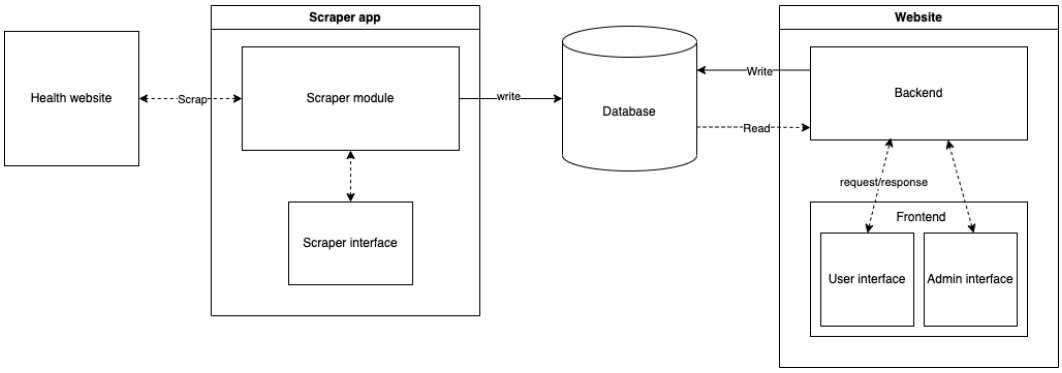
- User interface paradigms: The system may employ specific user interface paradigms, such as a single-page application or a mobile-first design approach.

- Error detection and recovery: The system may be designed with specific error detection and recovery mechanisms, such as automated testing, exception handling, and logging.

- Communication mechanisms: The system may employ specific communication mechanisms, such as **REST APIs** to enable communication between different system components.

- Distributed data or control over a network: The system may be designed with distributed data or control over a network, such as using a microservices architecture.

4. System Architecture



4.1. Subsystem Architecture

- **Website**: This component serves as the primary means of providing users with access to information about doctors, as well as giving administrators the ability to manage and control the system. This component has two parts: the Frontend, which is responsible for the user and admin interface, and the backend, which handles requests, authentication, and database operations.

- **Database**: The Database component is tasked with the responsibility of storing data related to users and doctors. It facilitates the CRUD (Create, Read, Update, and Delete) operations associated with the data.

- **Scraper app**: It is a software program designed to collect data on doctors from health websites and store it in the database for use on the website. It is composed of two main components: the scraping module, which is the central sub-component of the app and contains the script for the scraping process, and an interface that allows users to access and utilize the app's scraping capabilities.

- **Health website**: is the website from which we will gather information on doctors

5. Policies and Tactics

- Consistency: The design of the system should be consistent throughout all its components, from the color scheme and typography to the placement of buttons and icons. This will boost the user experience, which will get more users to use the website.

- Responsiveness: The system should be designed to be responsive across different devices and screen sizes. So, the user shall be able to use the website smoothly with their mobile phone.

- Error handling: The system should be designed to handle errors gracefully and provide clear feedback to the user when something goes wrong. This includes providing helpful error messages, validating user input, and preventing data loss.

- Security: The system should be designed with security in mind, implementing measures such as encryption, password hashing, authentication, and access control to protect user data and prevent unauthorized access.

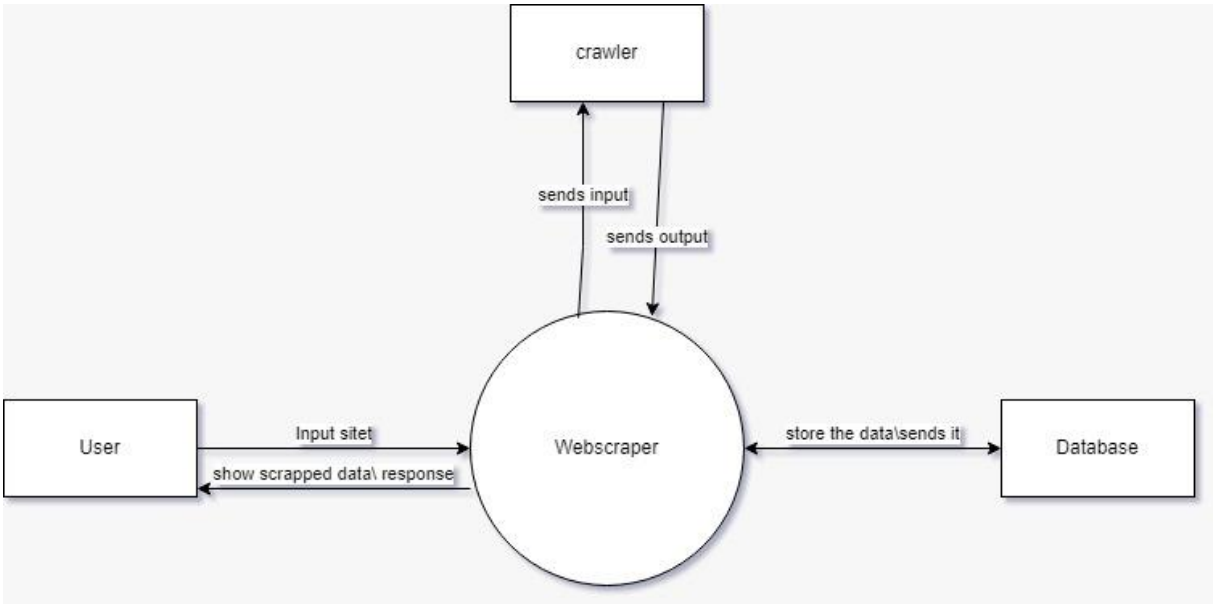
6. Detailed System Design

Our web scraping website (DocFinder) for doctors' data has been designed with a focus on efficiency and accuracy. We have created detailed design diagrams that outline the various components and their interactions within the system. These diagrams capture the data flow, data storage, and processing mechanisms, ensuring that the scraping process is smooth and reliable. With these design diagrams as a foundation, our system is well-equipped to extract and organize doctors' data effectively.

6.1. Classification

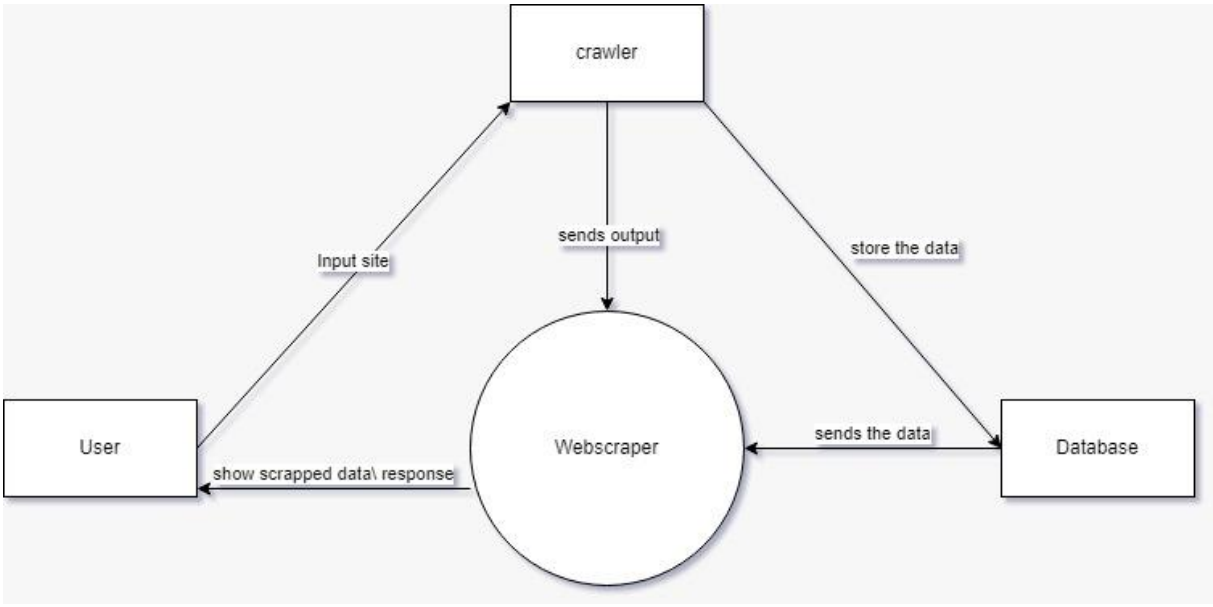
Here we outline the types of components used in the system. Each component serves a specific purpose in the web scraping process and contributes to the overall functionality of the system. All are as shown bellow.

1. Context Diagram



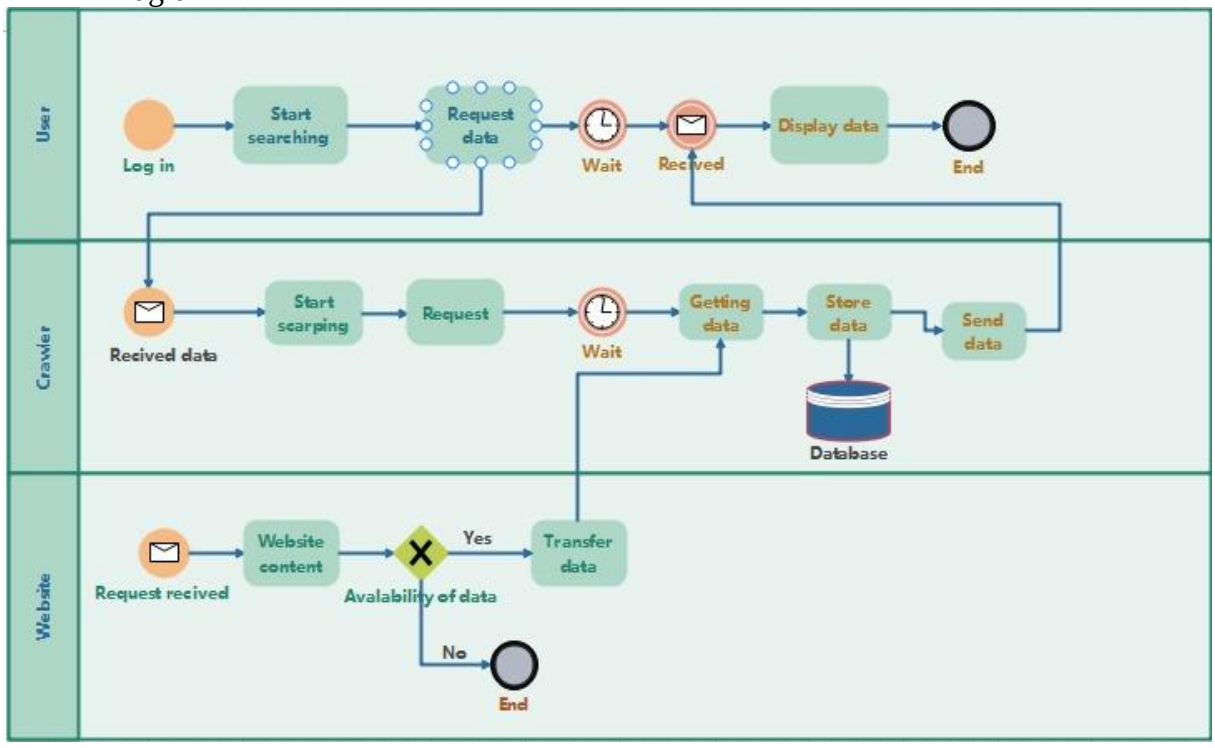
This context diagram for our web scraping system for doctors' data depicts the system at the center, surrounded by its primary interactions. The system interacts with three main components: the user, the database, and the crawler. The user initiates the scraping process by providing input and receiving output from the system. The database serves as the storage for the extracted doctors' data, enabling efficient data retrieval and management. The crawler component is responsible for navigating the target websites, extracting the relevant information, and feeding it into the system for processing. The context diagram presents a concise overview of how the system interacts with its key components, facilitating a clear understanding of its role and relationships.

2. Dataflow diagram



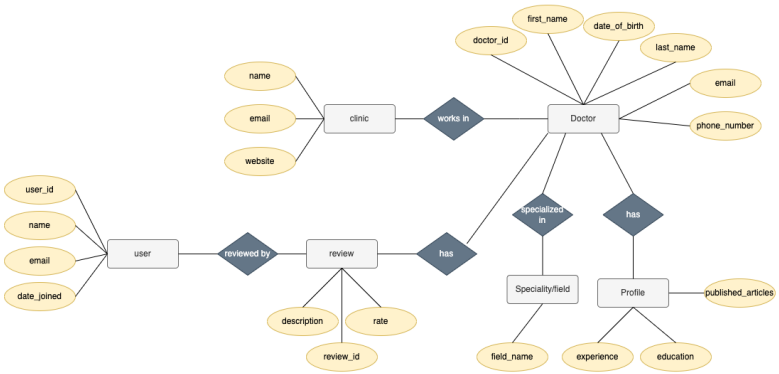
This is data flow diagram (DFD) for our web scraping system for doctors' data. It illustrates the flow of information within the system. The main process interacts with the user, receiving input such as website URLs and data requirements. It communicates with the crawler component to extract doctors' data, which is stored in a database. The system provides outputs to the user, such as data reports or notifications. The DFD highlights the system's data flow and interactions with external entities, aiding in understanding its functionality and data storage components.

3. BPMN Diagram

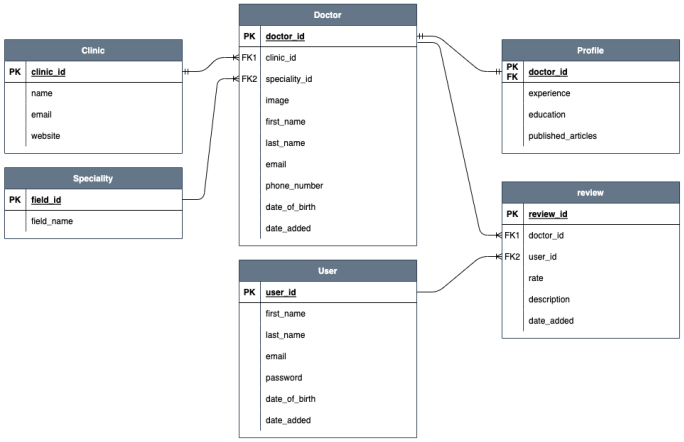


This is the BPMN diagram for our web scraping system for doctors' data. It depicts the system's workflow. It starts with the user initiating the process, followed by tasks such as user input when searching for a doctor, data extraction, data processing, and data storage. The diagram highlights the interaction between the main process and the crawler component, as well as the storage of extracted data in the database. Outputs, such as data reports or notifications, are represented, and decision points may be included. Overall, the BPMN diagram provides a concise and comprehensive visualization of the system's processes, interactions, and data flow.

4. ER Diagram:



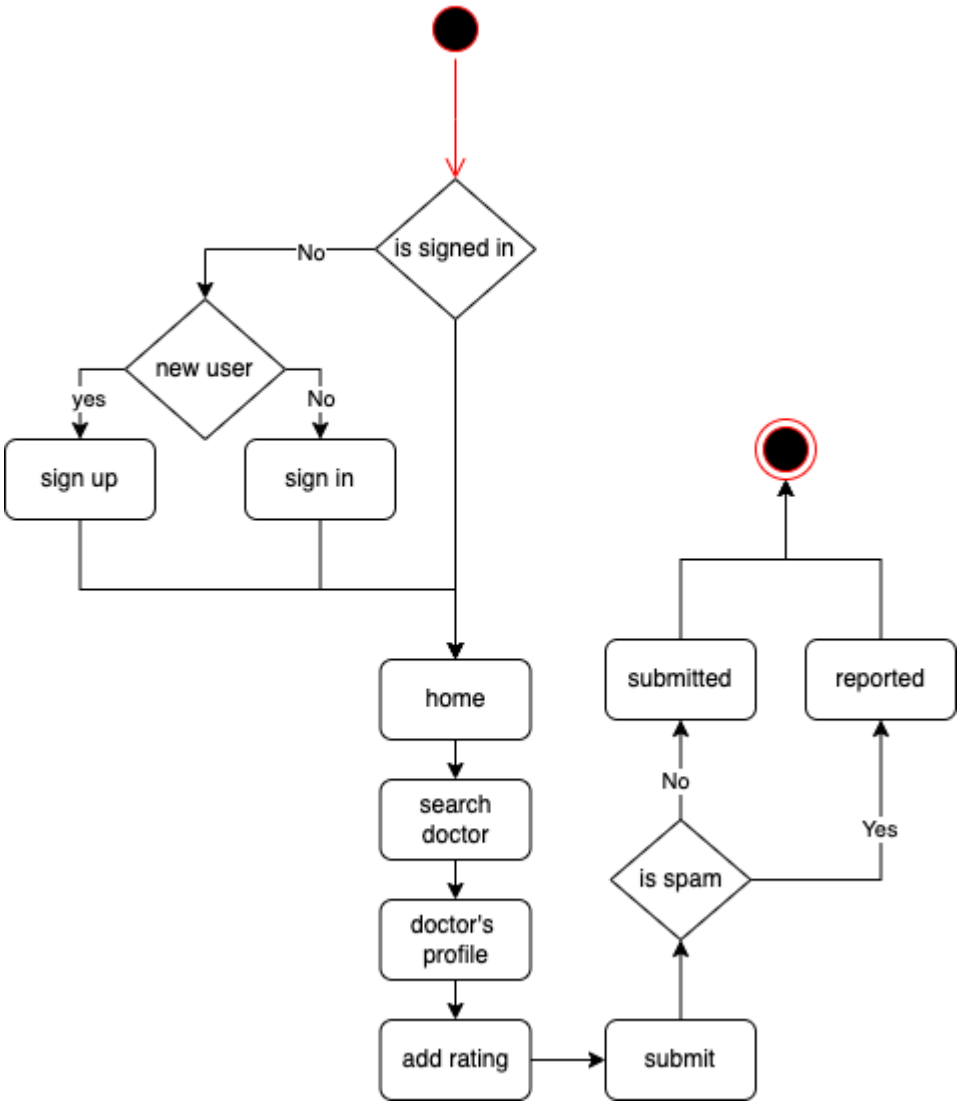
Our ER Diagram has the following entities, the user which has the user_id, name, email, date_joined. Doctors which have some information about those doctors. Review which has the description, rate, and review_id. The SpecialityField which contains field_name. The profile which contains the experiences, education and published articles. Additionally, we have clinics table which has clinics name, email, website columns. We also provided a more detailed database management diagram as shown bellow:



This shows the primary and foreign keys in addition to how those tables are related with each other.

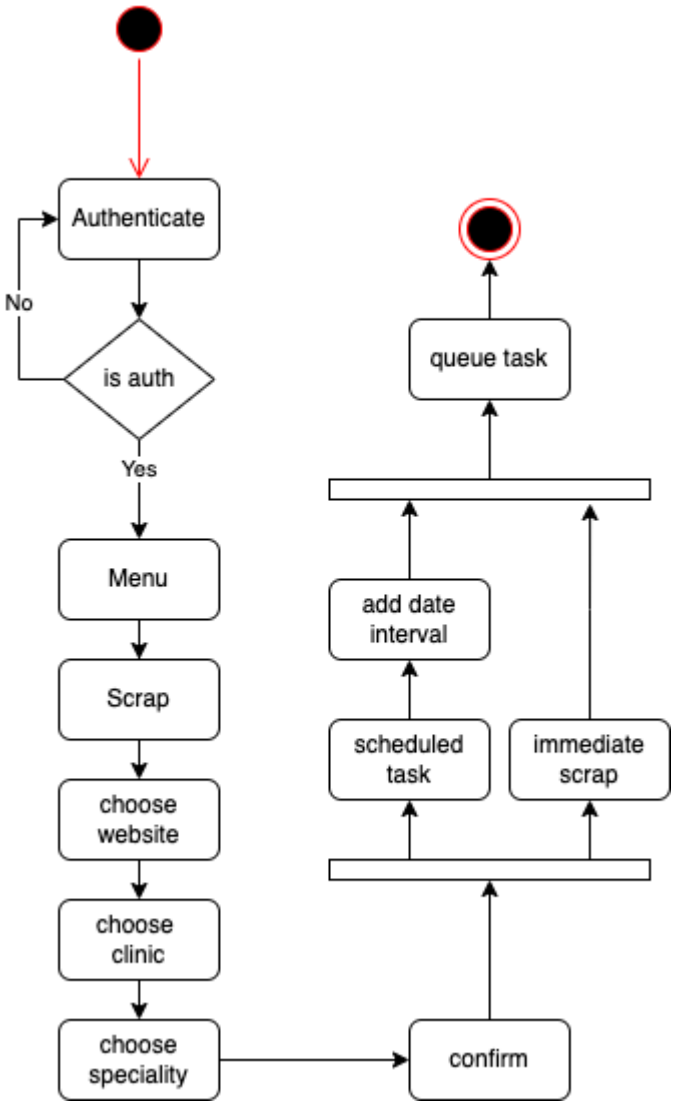
Activity Diagrams:

4.1 User add review:



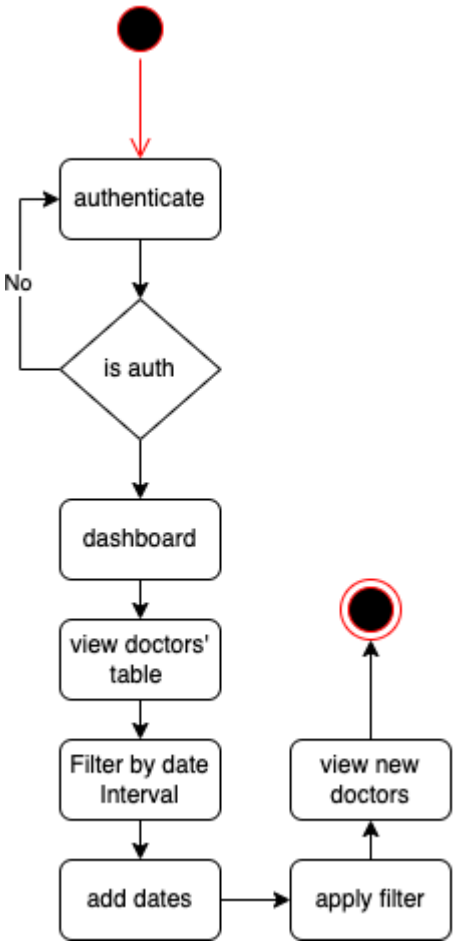
This activity diagram illustrates the process of a user adding a review for a doctor. It starts with the user logs in then searching for a doctor. After that the user selects a doctor, then filling out a review form with feedback, ratings, and comments. The system validates the review, and if it passes, the review is submitted, stored in the database, and associated with the respective doctor and user. The doctor's ratings and statistics are updated accordingly. The activity diagram concludes with the completion of the review addition process.

4.2 Scrap doctors:



This activity diagram illustrates the process of the system scraping doctors' data from external sources. It starts with selecting the source and then proceeds to scrape the data using web scraping techniques. Afterward, the system processes and cleans the scraped data before storing it in the database. The activity diagram concludes with the completion of the scraping process. The flow of the process is shown clearly in this diagram.

4.3 Admin track new doctors:



This activity diagram is for the admin tracking new doctors. It starts with the authentication process, where the administrator logs into the system. After successful authentication, the system enters the admin's dashboard and then he can go to view doctors' table and the "Track New Doctors" activity. It continuously checks for new doctor registrations. If new doctors are found, the system notifies the administrator. The administrator reviews the new doctors' information and decides to approve or reject them. The system updates the doctors' status accordingly. Finally, the administrator views the list of newly approved doctors. Additionally, admin can filter the tracked data by dates.

6.2. Definition

For our web scraping project that extracts doctors' data, we will describe the specific purpose and semantic meaning of the components involved.

Scraper Module: This module is responsible for accessing and retrieving data from external websites that contain doctors' information. It uses web scraping techniques to extract relevant data such as names, specialties, contact details, and other pertinent information.

Data Processing Module: This module processes the scraped data, performing tasks such as data cleaning, validation, and transformation. It ensures that the extracted information is accurate, consistent, and compatible with the system's data format.

Database Component: This component is responsible for storing the scraped doctors' data. It provides a structured storage mechanism to organize the collected information for efficient retrieval and usage.

Search Functionality: This component enables users to search for doctors based on different criteria such as specialties, locations, or names. It utilizes the scraped and stored data to facilitate efficient and accurate doctor search.

The semantic meaning of these components is to enable the automated extraction and storage of doctors' data from various sources, ensuring its availability for search and retrieval by users. The components work together to provide an efficient and reliable system for accessing up-to-date information about doctors."

6.3. Responsibilities

Component	Responsibilities	Role	Services
Scraper Module	Access external websites and extract doctors' data	Collect relevant information	Provide a mechanism to retrieve and compile data from various sources into a standardized format for further processing

Data Processing Module	Process scraped data to ensure accuracy and compatibility	Clean, validate, and transform data	Ensure reliability and integrity of doctors' data through data manipulation and enhancements
Database Component	Store scraped doctors' data in a structured manner	Provide efficient storage and retrieval	Enable persistence and organization of doctors' data, facilitating easy access and retrieval by other system components and users
Search Functionality	Enable users to search for doctors based on different criteria	Utilize scraped and stored data for search	Empower users to find doctors based on specialties, locations, or names, enhancing their ability to discover and connect with healthcare professionals

6.4. Constraints

This table provides a clear overview of the assumptions, limitations, and constraints specific to each component within the system. It covers the various aspects such as assumptions about external resources, limitations on performance and reliability, and constraints related to data validation, storage, access, and security.

Component	Assumptions	Limitations	Constraints
Scraper Module	Assumes availability and accessibility of external websites	Subject to performance and reliability of external websites	- Constraints on scraping rate to avoid overloading websites and potential IP blocking
Data Processing Module	Assumes consistent and expected input data format	- Must adhere to predefined data validation rules and formats	- Proper handling of exceptions or errors during data processing
Database Component	Assumes availability of the database system and resources	- Data storage must conform to defined database schema and constraints	- Proper handling of concurrent access to maintain data consistency and integrity
Search Functionality	-	- Efficient indexing and search algorithms should be employed to optimize search performance	- Search queries must be validated and sanitized to prevent potential security vulnerabilities - Ensure appropriate access controls for data protection

6.5. Composition

Here we describe the use and meaning of the subcomponents that are part of the main component. This section helps in understanding the hierarchical structure and relationships between the components within the system. We added the scraper Module component.

Component: Scraper Module

Subcomponents:

URL Manager: Responsible for managing the list of URLs to be scraped.

Web Scraper: Handles the actual extraction of doctors' data from the web pages.

Data Parser: Extracts relevant information from the scraped web content.

Data Storage: Stores the extracted data for further processing.

The URL Manager subcomponent is used to maintain a list of URLs from which the data will be scraped. It ensures proper management and organization of the scraping targets.

The Web Scraper subcomponent is responsible for accessing the web pages, retrieving the HTML content, and navigating through the website's structure to extract the necessary data.

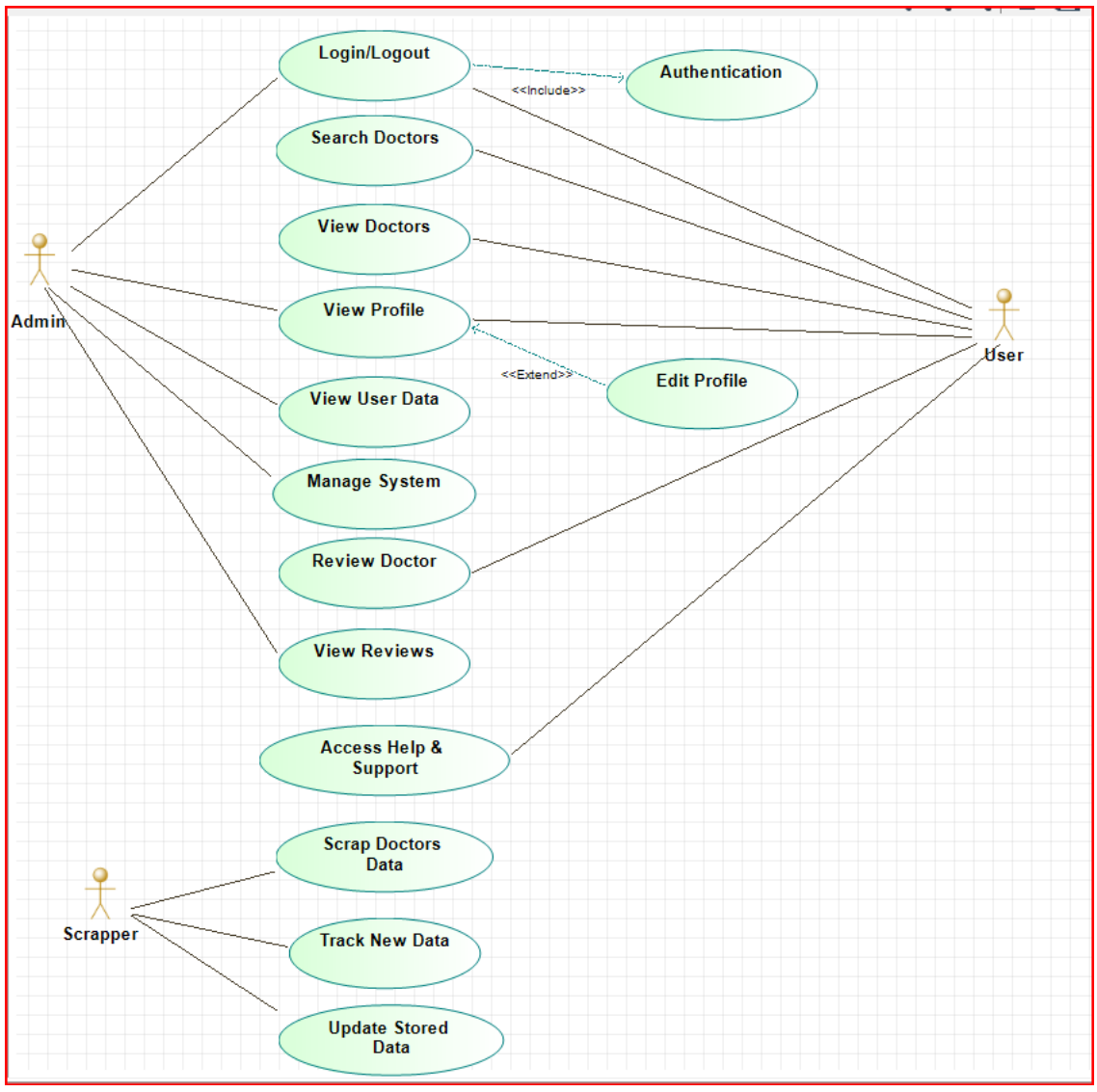
The Data Parser subcomponent processes the scraped HTML content and extracts the required information, such as doctors' names, specialties, contact details, and other relevant data points.

The Data Storage subcomponent is responsible for storing the extracted data, ensuring its availability for further processing and retrieval by other components.

These subcomponents work together to form the Scraper Module, enabling the extraction and storage of doctors' data from the web pages in a structured and organized manner.

6.6. Uses/Interactions

1. Use Case Diagram:



This use case explains major functionalities of the system as shown.

Use case narratives:

Use case1:
Title: Search Doctors
Actors: User, the person searching for doctors.
Description: this use case describes the process of searching for doctors on the system. The user wants to find doctors based on specific criteria, such as location, specialty, or name.
Preconditions:
• The user is logged into the system.
• The system is accessible and operational.
Use case scenario:
• The user accesses the search page within the system.

• The user provides search criteria, such as location, specialty, or name, through the user interface.
• The user interface validates and sends the search criteria to the doctor search controller.
• The doctor search controller receives the search criteria and initiates the search process.
• The doctor search controller communicates with the doctor search service to retrieve matching doctors based on the provided criteria.
• The doctor search service performs the search algorithm, querying the database for doctors that meet the search criteria.
• The database retrieves and returns the list of matching doctors to the doctor search service.
• The doctor search service collects the list of doctors and sends it back to the doctor search controller.
• The doctor search controller receives the list of doctors and formats the results.
• The formatted search results are sent to the user interface for display.
• The user interface presents the search results to the user.
• The user can view detailed information about a specific doctor by selecting the corresponding option.
• The user can choose to refine or modify the search criteria and repeat the search process if needed.
• Postconditions:
• The user is presented with a list of doctors matching the search criteria.
• The user can view detailed information about specific doctors.
• The user can refine or modify the search criteria and perform another search if desired.
Alternative scenario:
• If no doctors match the search criteria, the system displays a message indicating no results found.
• If there are any technical issues during the search process, such as a connection failure or database error, an error message is displayed to the user, prompting them to try again later or contact support.
Exceptions:
• If the user is not logged into the system, they are redirected to the login page before accessing the search functionality.

Use Case 2:
Title: Scrape Doctors' Data
Actors: Scraper, The component responsible for scraping doctors' data from target websites.
Description: this use case describes the process of scraping doctors' data from target websites using a scraper component. The scraper autonomously retrieves relevant information about doctors, such as their names, specialties, contact details, and locations.
Preconditions:

• The system is accessible and operational.
• The target websites to be scraped are identified and accessible.
Use case scenario:
• The scraper component is initialized within the system.
• The scraper retrieves the list of target websites to be scraped from the configuration or a predefined source.
• For each target website:
○ The scraper component navigates to the website using web crawling techniques.
○ It identifies and extracts the relevant data based on predefined scraping rules or patterns.
○ The extracted data is stored in a temporary data storage area or memory.
○ Once all target websites have been scraped, the scraper proceeds to process the extracted data.
• The scraper component performs any necessary data processing or formatting on the extracted data.
• The processed data is stored in the system's database for further use and analysis.
• The scraper component generates a completion message indicating the success of the scraping process.
• The completion message is logged and made available for system monitoring and auditing purposes.
Postconditions:
• The system contains the scraped doctors' data, including their names, specialties, contact details, and locations.
• The extracted doctors' data is stored in the system's database for further use.
Alternative Scenario:
• If there are any technical issues during the scraping process, such as website unavailability or connection errors, the scraper component logs an error and attempts to retry the scraping process after a specified interval.
• In the event of persistent issues or failures, the system may trigger an alert or notification to administrators or technical support for further investigation.
Exceptions:
• If the target websites become inaccessible or undergo significant structural changes, the system may require manual intervention to update the scraping rules or adapt to the changes.

Use Case 3:
Title: Review a Doctor
Actors: User, The person providing a review for a doctor.
Description: this use case describes the process of reviewing a doctor in the system. The user, who has had an experience with a specific doctor, shares their feedback, rating, and comments to help others make informed decisions.
Preconditions:

• The user is logged into the system.
• The system is accessible and operational.
• The doctor being reviewed is registered in the system.
Use case main scenario:
• The user navigates to the page for reviewing doctors within the system.
• The user selects the doctor they wish to review from a list or by searching for the doctor's name.
• The system presents a review form to the user, allowing them to provide their feedback, rating, and comments about the doctor.
• The user fills in the review form, rating the doctor on various aspects and sharing their comments.
• Upon submission, the system validates the review, ensuring all required fields are filled.
• The system stores the review in the database, associating it with the respective doctor and the user who submitted it.
• The system updates the doctor's overall rating and statistics based on the newly added review.
• The user receives a confirmation message indicating that their review has been successfully submitted.
Postconditions:
• The doctor receives a new review, which is associated with their profile in the system.
• The doctor's overall rating and statistics are updated based on the received review.
• The user receives a confirmation message indicating the successful submission of their review.
Alternative Scenario:
• If the user attempts to submit a review without filling in all the required fields, the system displays an error message, prompting the user to complete the necessary information.
• In the case of a technical error or database issue during the review submission process, the system displays an error message and advises the user to try again later or contact support.
Exceptions:
• If the user is not logged into the system, they are redirected to the login page before accessing the review functionality.

Use Case Glossary:

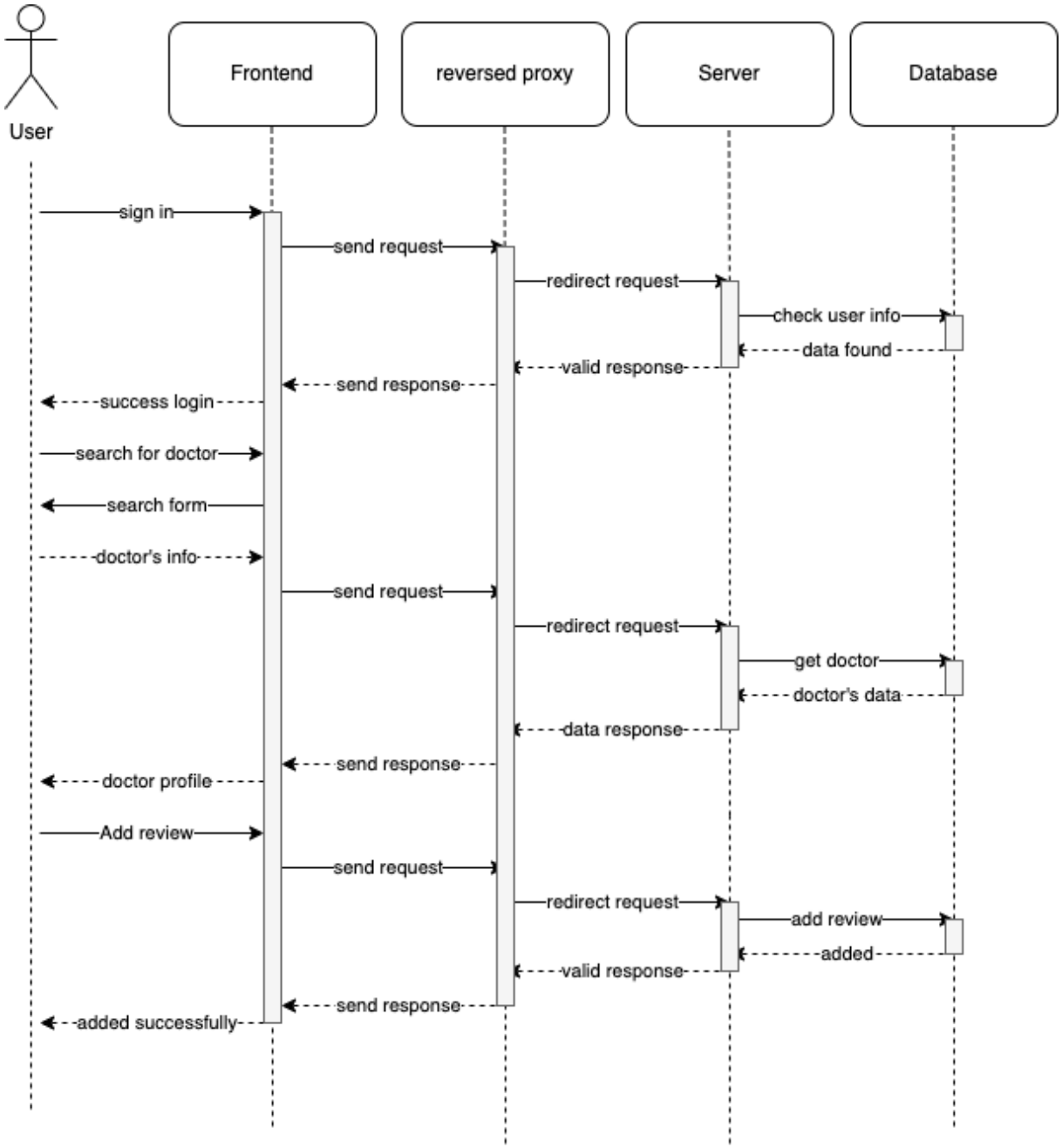
Use Case	Description	Actors	Preconditions	Postconditions

Login/Logout	The process of logging into or out of the system	User	None	User is logged in or logged out of the system
Authentication	The process of verifying the identity of a user	User	User has provided valid credentials	User is granted access based on authentication
Search Doctors	The process of searching for doctors based on criteria	User	User is logged in	User is presented with a list of matching doctors
View Doctors	The process of viewing detailed information about a doctor	User	User is logged in	User can view a doctor's profile and details
View Profile	The process of viewing the user's own profile information	User	User is logged in	User can view their own profile information
View User Data	The process of viewing data related to	User	User has appropriate permissions	User can view data of other users

	other users			
Edit Profile	The process of modifying and updating user's profile	User	User is logged in	User's profile information is updated
Manage System	The process of managing and configuring system settings	Administrator	Administrator is logged in	System settings and configurations are modified
View Reviews	The process of accessing and viewing reviews for doctors	User	User is logged in	User can read and access reviews for doctors
Access Help/Support	The process of accessing help and support resources	User	User is logged in	User can access help documentation or contact support
Scrape Doctors Data	The process of automatically extracting doctors' data	Scraper	System is operational and target websites are accessible	Doctors' data is extracted and stored in the system's database

Track New Data	The process of monitoring and identifying new doctors' data	Scraper	System is operational and target websites are accessible	New doctors' data is identified for extraction
Update Stored Data	The process of updating existing doctors' data	Scraper	System is operational and target websites are accessible	Existing doctors' data is refreshed and updated

2. Sequence diagram:



This sequence diagram illustrates the interactions and flow of messages between user, frontend, server and database components of the system. This is the case of a user adding a review where the steps evolved in this process are captured. Here is the sequence in details:

1. User signs in to the website.
2. The system's frontend sends a request to reversed proxy.
3. The reversed proxy directs this request to the server.
4. The server sends this information to be checked in the database.
5. In this case, database finds the user information so it confirms the server that data of this user is found.

6. The server sends a valid response to reversed proxy.
7. The reversed proxy assures that valid response to frontend component.
8. The frontend indicates the success login for the user.
9. Now user will search for a doctor.
10. A search form will appear to user.
11. User looks for doctors' information.
12. This request is sent to server and thus be found in database of the system.
13. The database retrieves doctor's data and send it to server.
14. The server sends response to frontend and thus the doctors profile will appear to the user.
15. Now the user can add his review upon that doctor.
16. The review will be added to the database through reversed proxy and system's server.
17. The database confirms that the review is added and thus sends a valid response back to the server and the server confirms this to frontend.
18. Now the user's review is added successfully.
19. Other users can view that review for that doctor.

6.7. Resources

Here, we describe the various resources that are managed, affected, or needed by the entity/component. These resources are typically external to the design and can include memory, processors, printers, databases, software libraries, and more. Additionally, this section should address any potential race conditions or deadlock situations that may arise and propose resolutions for them.

Component	Resources	Resolution Strategies
Scraper Module	- Internet Connection	- Ensure stable and reliable internet connection.
	- Computing Resources	- Allocate sufficient computing resources for scraping.
	- External Websites/APIs	- Handle API rate limits and timeouts.

Data Processing Module	- Computing Resources - Memory - Software Libraries	- Allocate adequate computing resources and memory. - Optimize data processing algorithms and code. - Utilize efficient software libraries and frameworks.
Database Component	- Database Server - Storage Space - Network Connection	- Deploy and configure a robust database server. - Allocate enough storage space for data storage. - Ensure a reliable network connection to the database.
Search Functionality	- Computing Resources - Memory - Search Index/Data Structure	- Allocate sufficient computing resources and memory for search functionality - Optimize search index/data structure for efficient search operations.

6.8. Processing

This table provides a concise overview of the processing aspects of each component, including their descriptions, algorithms used, changes of state, time/space complexity, and error handling mechanisms.

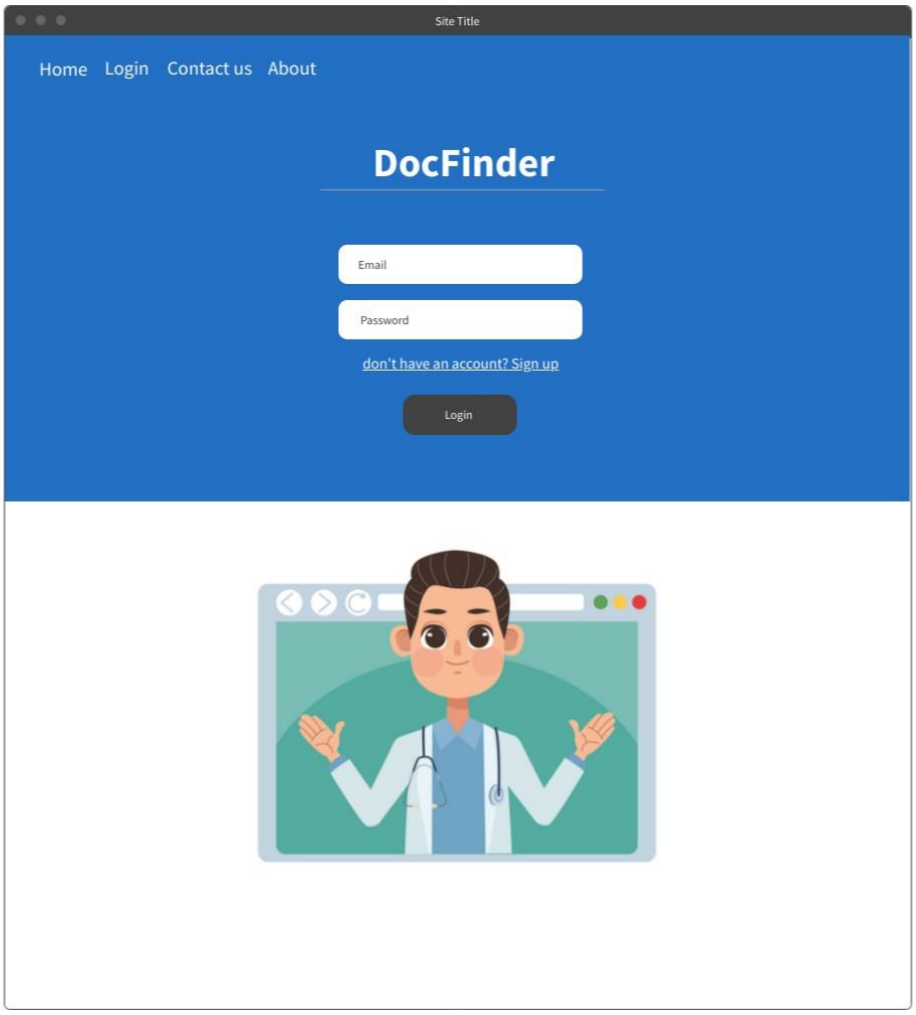
Component	Description
Scraper Module	- Responsible for extracting doctors' data from external websites. - Utilizes web scraping techniques such as HTML parsing and DOM traversal. - Undergoes changes of state as it navigates web pages, retrieves data, and stores it in the database. - Time/Space Complexity: Varies based on the number of pages and size of data being scraped. - Handles concurrent scraping tasks for improved efficiency. - Created and initialized during system startup, establishing connections and initializing resources. - Performs cleanup processes upon completion or system shutdown. - Incorporates error handling mechanisms to address exceptional conditions.
Data Processing	- Responsible for processing and analyzing the

Module	scraped doctors' data. - Applies data cleaning, transformation, and enrichment techniques. - Implements algorithms for data analysis and generating insights. - Manages changes of state during data processing operations. - Time/Space Complexity: Depends on the complexity of data processing algorithms and the volume of data being processed. - Utilizes computing resources and memory efficiently for data processing tasks. - Incorporates error handling mechanisms to handle exceptions and ensure data integrity.
Database Component	- Responsible for managing and storing the scraped doctors' data. - Establishes connections to the database server and manages data retrieval and storage. - Handles changes of state as data is inserted, updated, or queried. - Time/Space Complexity: Depends on the scale of data storage and the complexity of database operations.

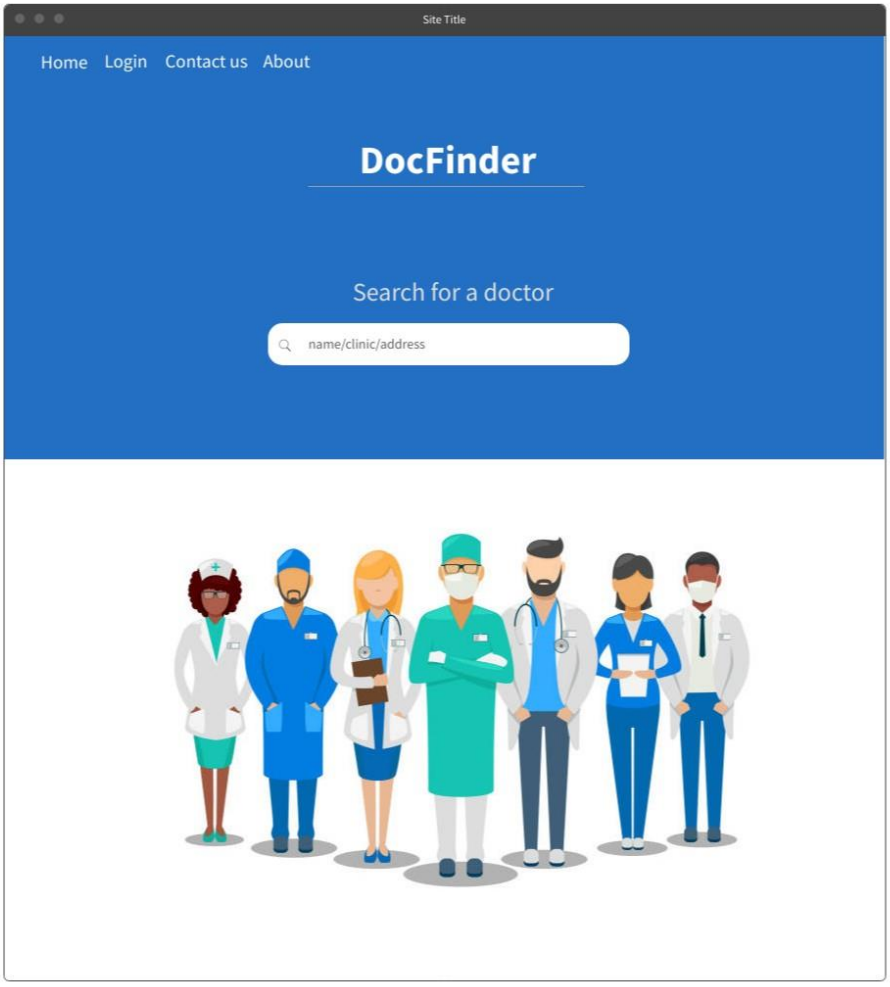
	- Ensures data integrity, consistency, and security. - Implements efficient indexing and querying mechanisms for fast data retrieval. - Incorporates error handling mechanisms to handle database-related exceptions and ensure data reliability.
Search Functionality	- Implements search functionality for users to find doctors based on various criteria. - Utilizes search algorithms and data structures to optimize search operations. - Handles changes of state as search queries are executed and results are displayed. - Time/Space Complexity: Depends on the complexity of the search algorithms and the size of the search index. - Provides efficient and accurate search results to users - Incorporates error handling mechanisms to handle search-related exceptions and ensure smooth operation.

6.9. Interface/Exports

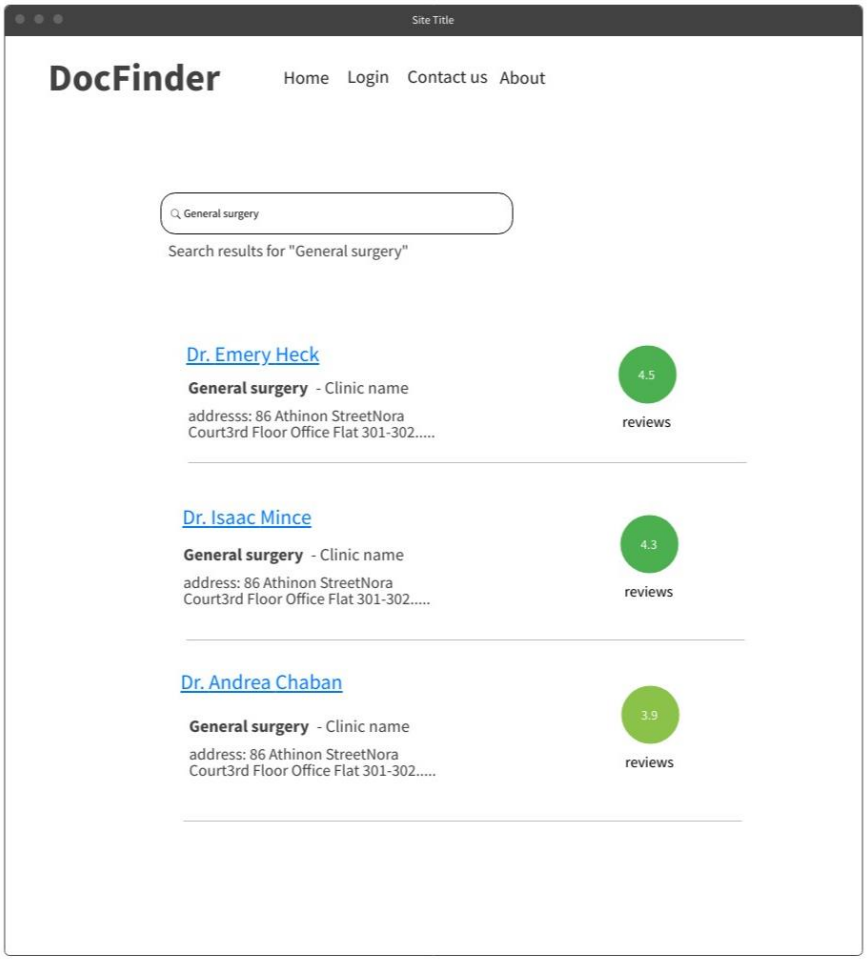
Here we provided some of the user interface designs that we have made for the system.



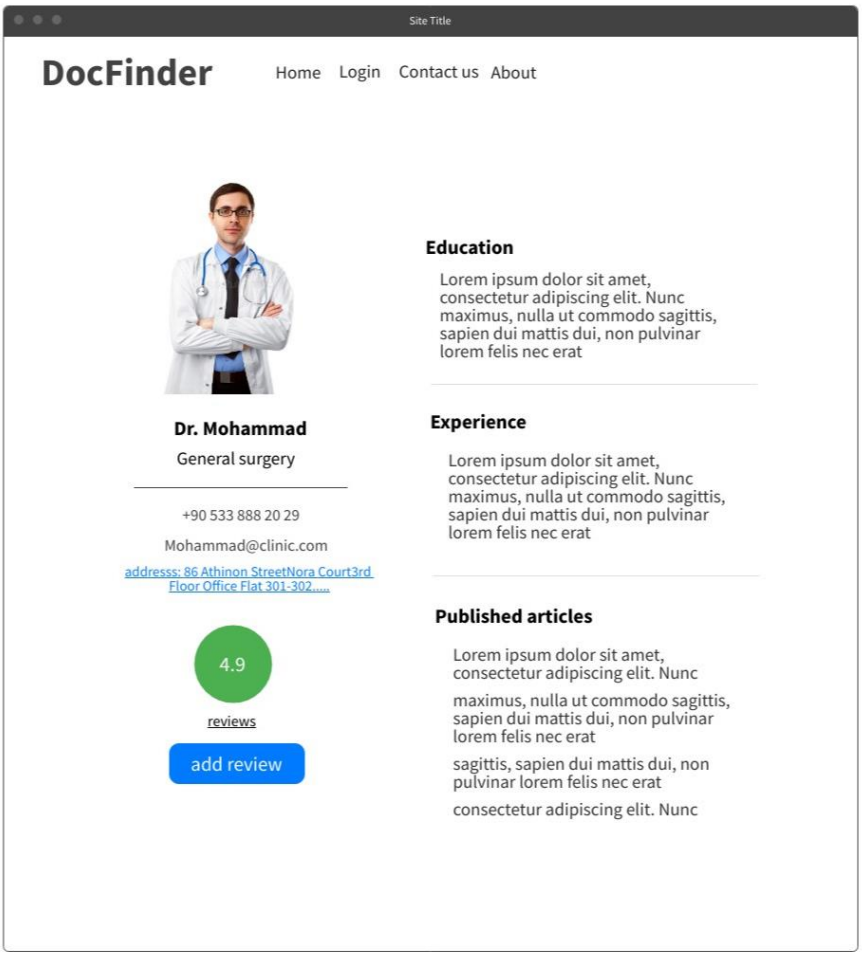
This is the log in page for the system where users can get access to the sytem.



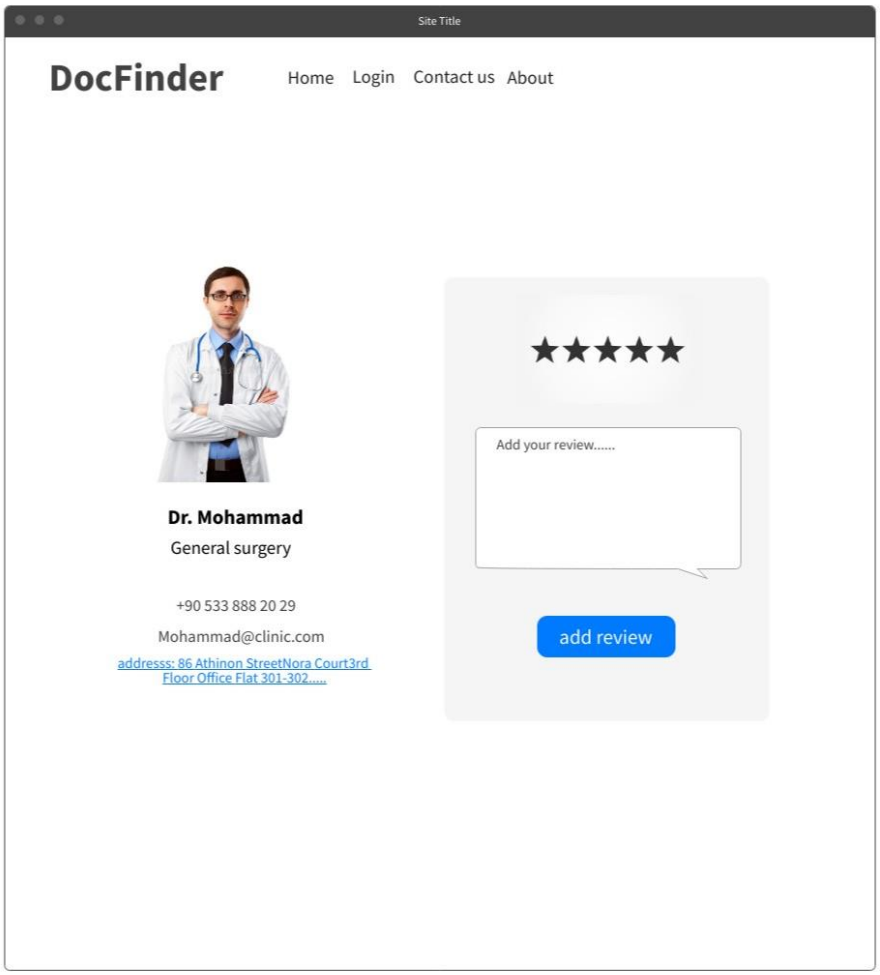
At this page, is the main functionality, which is searching about a doctor. Where user can search using any keywords and results will be shown.



This page shows what will the user see after searching for a doctor as shown in the image.



Here the profile of a doctor is shown in details.



One of the main functionalities of our system is to allow the users to rate and review doctors, and this screen shows how the user interface will be.

7. Glossary

API – Application Program Interface
CSS – Cascading Style Sheets
JS - Java Script
DB - Database
HTML – Hyper-Text Markup Language
IDE – Integrated Development Environment
OS – Operating System
SDS – Software Design Specification
SQL – Structured Query Language
SRS – Software Requirements Specification
UI – User Interface

8. Bibliography

A list of referenced and/or related publications.

Brad Appleton <brad@bradapp.net>
http://www.bradapp.net