ProverB

ARCHITECTURAL DESIGN COUNSELING MANAGEMENT SOFTWARE

Connor Low

CST-451 Capstone Project Final Architecture & Design

Grand Canyon University

Instructor: Professor Mark Reha

Revision: 2

Date: April 10, 2019

# Design Planning Summary

There are insufficiencies in existing counseling management systems for the specialized field of biblical counseling. A software that meets the specific needs of biblical counselors to store counselee information and track counselee progress is proposed. Although the current project is focused on a single client, due to the specialized nature of biblical counseling within churches, it has potential to appeal to a larger market. This application would allow administrative and counselor users to manage caseloads of counselees. An additional function of the application would be to provide access to biblical resources that may be given to counselees. Requirements for the application will be largely influenced by the client, Pastor Jim Low. To consider the product “complete,” the application will need to support administrator users, who will manage counselees and caseloads, and counselor users, who will manage their individual caseload. The main contributors to this project will be Connor Low, the developer, and Pastor Jim Low, the client. The general schedule for development is: the planning and design phase of the project will be completed by early November; the development phase will begin in November and will have a draft ready by February, and a complete product by March; the testing phase will begin in March; and the deployment phase will be completed by mid-April.

|  |
| --- |
| History and Signoff Sheet |

Change Record

|  |  |  |
| --- | --- | --- |
| Date | Author | Revision Notes |
| November 11, 2018 | Connor | Initial draft for review/discussion |
| April 10, 2019 | Connor | Version 0.1.0 architecture changes |
| April 14, 2019 | Connor | Version 0.1.1 architecture changes. Updated Technologies list. Updated UMLs. Updated resolved Dates for Risks. Added Delivery plan. |

|  |
| --- |
| Overall Instructor Feedback/Comments |

|  |
| --- |
| Overall Instructor Feedback/Comments |

Integrated Instructor Feedback into Project Documentation

Yes  No

# Table of Contents

[Design Planning Summary 2](#_Toc6171708)

[Table of Contents 3](#_Toc6171709)

[Feature Delivery Schedule 4](#_Toc6171710)

[Design Overview 4](#_Toc6171711)

[Detailed High-Level Solution Design 6](#_Toc6171712)

[Logical Solution Design: 8](#_Toc6171713)

[Deployment Diagram 9](#_Toc6171714)

[Physical Solution Design: 9](#_Toc6171715)

[Detailed Technical Design 10](#_Toc6171716)

[General Technical Approach: 10](#_Toc6171717)

[Key Technical Design Decisions: 11](#_Toc6171718)

[Database ER Diagram: 13](#_Toc6171719)

[Flow Charts/Process Flows: 14](#_Toc6171720)

[Sitemap Diagram: 20](#_Toc6171721)

[User Interface Diagrams: 21](#_Toc6171722)

[UML Diagrams: 31](#_Toc6171723)

[Service API Design: 32](#_Toc6171724)

[Non-Functional Requirements: 33](#_Toc6171725)

[Operational Support Design: 35](#_Toc6171726)

[Other Documentation: 36](#_Toc6171727)

[Appendix 37](#_Toc6171728)

[Appendix A – Technical Issue and Risk Log 37](#_Toc6171729)

[Appendix B - References 38](#_Toc6171730)

[Appendix C – External Resources 40](#_Toc6171731)

# Feature Delivery Schedule

Due to the learning curve and complex nature of the application, many of the features considered in-scope at the beginning of the project are out of scope for the final submission. However, these features will still be implemented in the future. The following schedule will be used to benchmark progress.

|  |  |  |
| --- | --- | --- |
| Feature | Target date | FRs |
| Detach Login from main application | 5/1/19 | N/A (update requried) |
| Counselee management view | 5/1/19 | 1.12, 3.5 |
| Assign Counselee to Counselor | 5/15/19 | 3.6 |
| Counselor view caseload | 5/15/19 | 1.3, 3.6 |
| View counselee information | 5/15/19 | 1.12 |
| Assign Resource to counselee | 6/1/19 | 1.11 |
| Add contact information | 6/1/19 | 1.10.3, 3.3, 3.5, 3.8 |
| Manage/Add documents | 6/15/19 | 1.5, 1.6, 1.7 |
| View counselee documents | 6/15/19 | 1.4 |
| User Settings | 7/1/19 | 1.10 |
| Update counselee status | 7/1/19 | 1.13, 3.4 |
| Implement roles | 7/15/19 | 3.1 |
| Reporting | 8/1/19 | 3.9 |

# Design Overview

A secure application is needed to manage data pertaining to counselees in a counseling process. Specifically, the client requires the ability to manage information regarding several forms that are used before, during, and after the counseling process. In addition, the client requested the ability to assign resources often used in biblical counseling, and the ability to gain insight into other counselor’s processes by reporting basic metrics tracked by the application.

The application centers around four primary data models: user, counselee, resource, and document/form. A user will be able to access one or multiple caseloads of counselees. These counselees will have: a descriptive profile summarizing their counseling topic and personal information; a collection of documents pertaining to their counseling process; a list of assigned biblical resources; and a status indicator regarding counseling process progression. A user with admin privileges will be able to see a list of all counselees currently entering, in the middle of, or exiting the counseling process. This user will be able to assign the counselee to a counselor. The counselee will then appear on that counselor’s caseload. A user with counselor privileges will be able to manage the counselee’s documents and assigned biblical resources. The counselor will also be able to update the counselee’s status in the system. The average counseling process duration and caseload size will be tracked for reporting.

# Detailed High-Level Solution Design

The React library will be used to create and update user interfaces. Data will be queried by the React application using an API provided by an Express application running on a NodeJS server. Managers will be used to handle business logic within the Node application for users, counselees, forms, and resources. The DAO design pattern will be used when implementing a data access layer for users’, counselees’, forms’, and resources’ persistence logic. A Mongo Database will be used for persistence.

|  |  |  |
| --- | --- | --- |
| Proof of Concepts | | |
| Description | Outcome | Decisions |
| Save/query data in a MongoDB | Successfully inserted data. Performed multi-part queries that included limits, sorting, and conditionals. | Continue with plan to use MongoDB. |
| Create a simple react application that interacts with an Express API. | Successfully created a React clock component that queried the current time from the Express API. Used React state to perform live updates to the DOM. | Continue with plan to use React. |
| Create a multi-page React application. | Looked at two solutions: Express routing and React routing. Express did not cover needs. React routing would work in theory but was not successfully implemented. | Use React components (.jsx files) as views. |
| Interact with MongoDB through a JavaScript application. | Successfully inserted and retrieved data. Performed multi-part queries from the Node.js backend using the MongoDB driver. | Before moving on to the final phase, attempt a proof of concept using Mongoose, a library built on top of the MongoDB driver. |

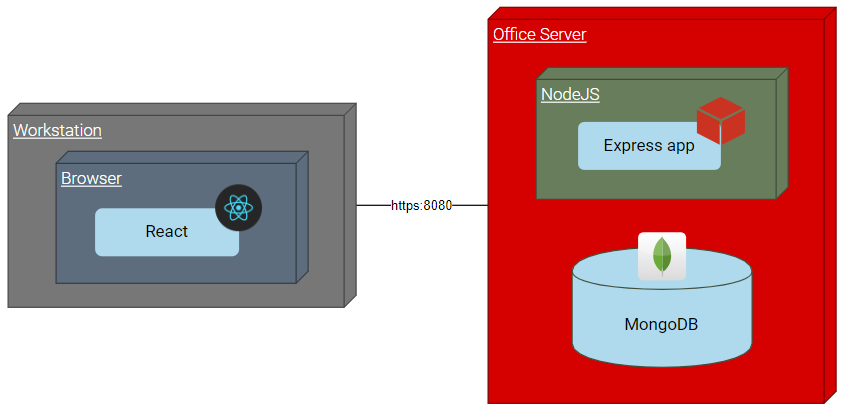
|  |  |
| --- | --- |
| Hardware and Software Technologies | |
| **React** | Render views and manage state. |
| **Material-UI** | Responsive Design |
| **Node** | JavaScript in the server. |
| **Express** | A framework for Node. |
| **MongoDB** | NoSQL persistence technology. |
| **Mongoose** | JavaScript interface for MongoDB. |
| **Babel** | Browser-compliant JavaScript compiler. |
| **JWT (JSON web token)** | Security. |
| **Chart.js** | A charting library. |
| **SJCL** | Encryption library. |

## Logical Solution Design:

A screenshot of a cell phone

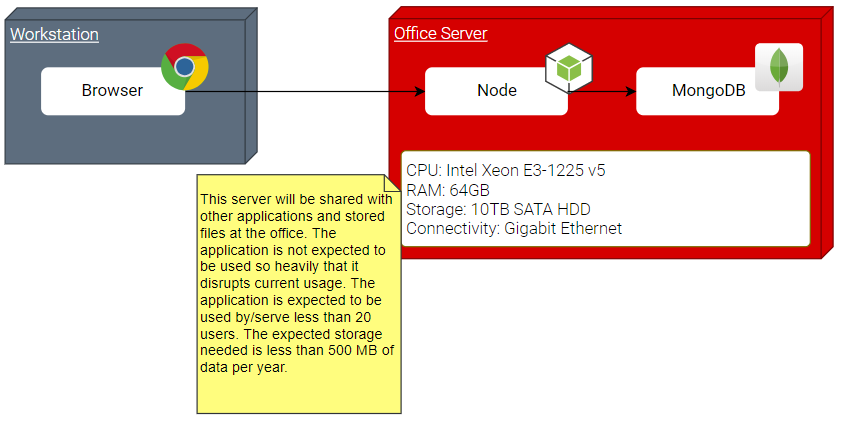
Description automatically generated

## Deployment Diagram



## Physical Solution Design:

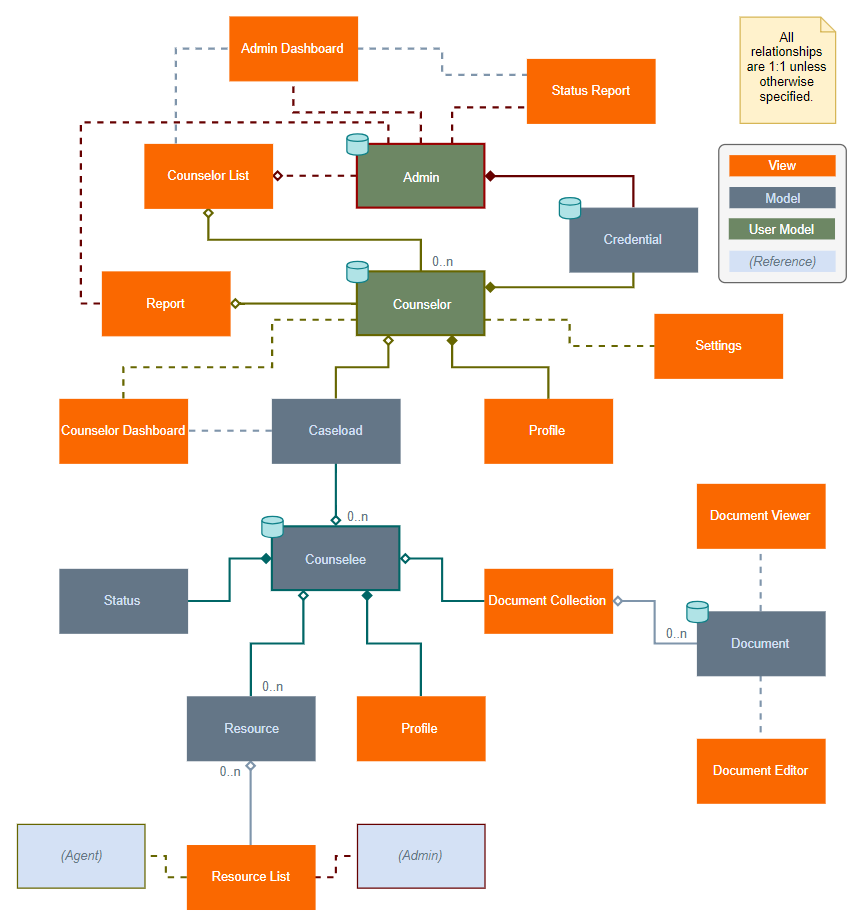
The application will run on an internal church office server as part of the office intranet. Workstations must be wired in to the office network to access the server. The office and therefore the server are small, so it is expected that the application will not take on a heavy user-load.



# Detailed Technical Design

## General Technical Approach:

A map representing the basic entity-relation structure of the project:



## Key Technical Design Decisions:

The following technologies will be used in the design:

### React

A flexible component-based JavaScript library for the frontend. React will be used to create views and determine how data is displayed. React components will use the .jsx extension.

* License: MIT
* Usage: presentation layer
* Install source: npm

### Material-UI

A React component library for consistent and responsive UIs.

* License: MIT
* Usage: presentation layer
* Install source: npm

### Node

Environment for server applications coded in JavaScript. Node will be used as the runtime environment. Node also comes with a package manager, npm.

* License: open source (see github.com/nodejs/node)
* Usage: runtime environment
* Install source: npm

### Express Framework

A Framework for Node. Express will be used to create an internal API for React to consume.

* License: MIT
* Usage: business and data access layers
* Install source: npm

### MongoDB

A NoSQL database. MongoDB will be used to persist data. MongoDB stores data in a JSON format which is optimal for JavaScript based applications.

* License: Server Side Public License (copy and distribution ok; see github.com/mongodb/mongo)
* Usage: database
* Downloads: Mongo

### Mongoose

“Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment. (npm, Inc., 2018)” Mongoose will be used as an application server interface for accessing and updating the database.

* License: Public License (copy and distribution ok; see github.com/mongodb/mongo)
* Usage: database driver
* Npm: npm install mongoose

### Babel

A JavaScript complier for browser friendly JavaScript. Babel will be used to translate JavaScript so that is functions in all browsers.

* License: opensource license by LearnBoost (see github.com/Automattic/mongoose)
* Usage: compatibility tool
* Npm: npm install --save-dev @babel/core

### jsonwebtoken

A JavaScript library for Json Web Tokens (JWT). JWTs are used to authenticate users.

* License: MIT
* Usage: security
* Npm: npm install jsonwebtoken

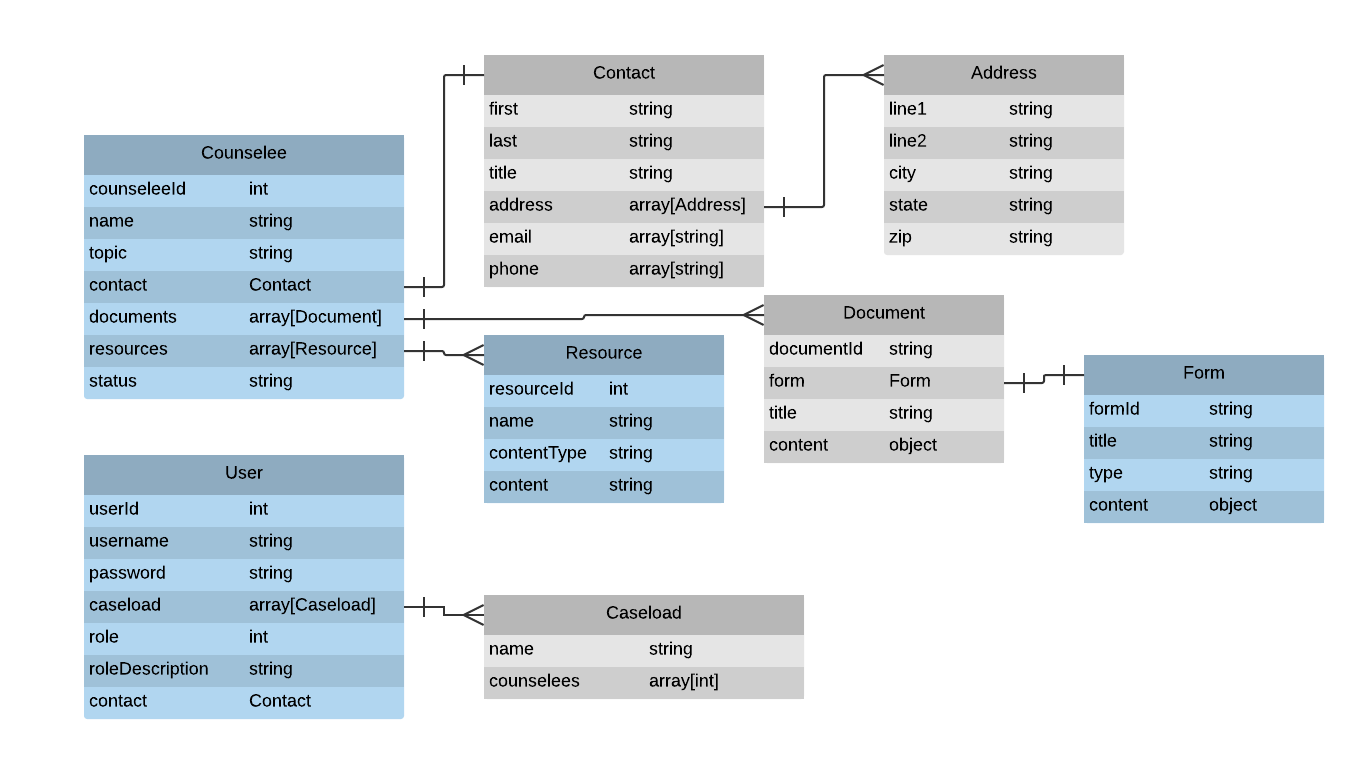
### Chart.js

A JavaScript library for rendering charts and graphs.

* License: MIT
* Usage: graphing, charts
* Npm: npm install chart.js --save

## Database ER Diagram:

MongoDB is a NoSQL database that stores information in a BSON (Binary JSON) format. Rather than relate tables with keys, objects and arrays can be nested inside database objects. Below, tables in blue represent parent database objects and tables in grey represent the architecture for child objects.



Notes:

* An inter-tabular relational key exists between the Counselor.counselees array of integers and the Counselee.counseleeId.
* The Form table will include few objects, since only a few forms exists. These are copied into Document objects for meta-data and editing.

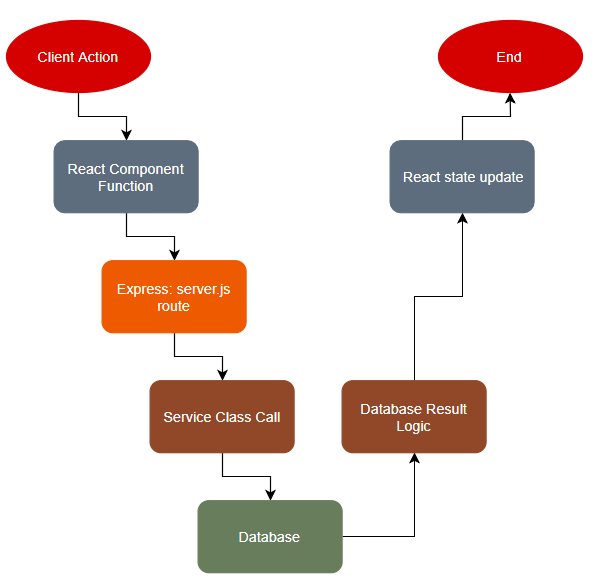
Since NoSQL is unstructured, a DDL script is not applicable. For a representation of the general data structure in JSON format, see the API design under **Appendix C**.

## Flow Charts/Process Flows:

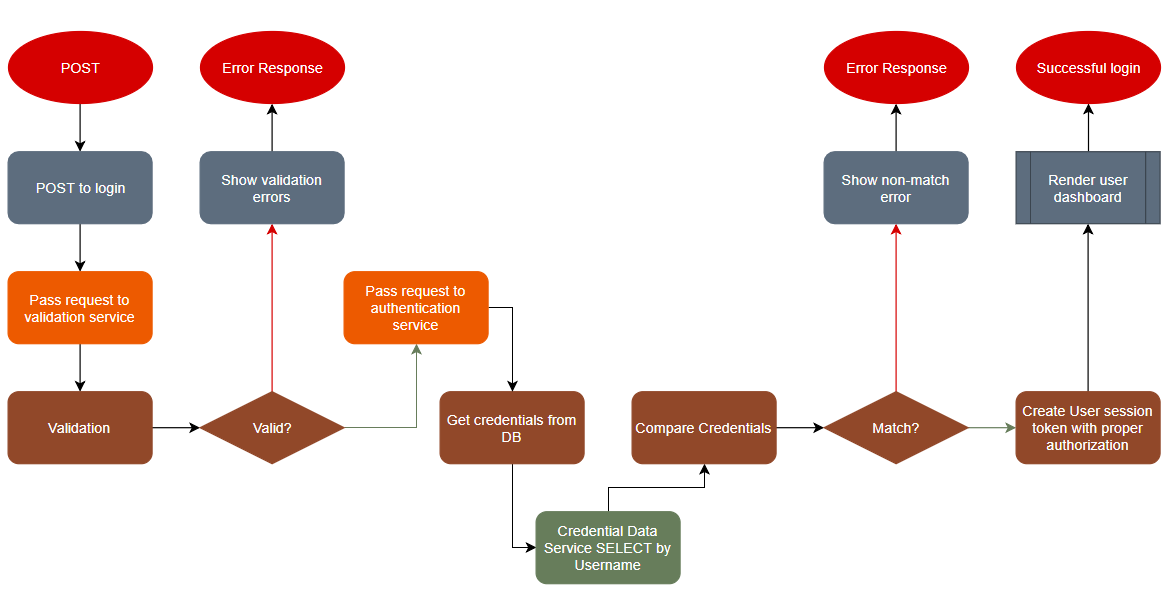
The logical processes of the application’s major functionality are covered in this section

### General Process Flow Template

In general, client interaction with the DOM will trigger React functions. These will often require React to ask the server for data (blue). Requests will be handled by Express which will subsequently call business service functions through specific controller methods (orange). These business services (brown) will retrieve data from a cache or a data service (green).

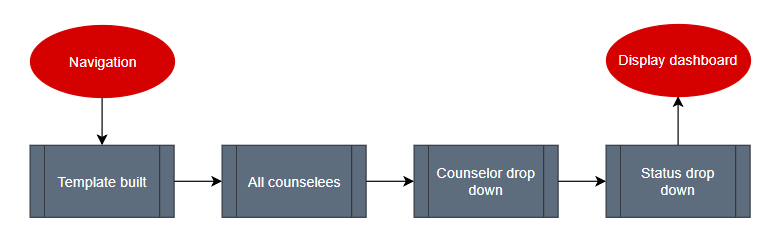


### Login

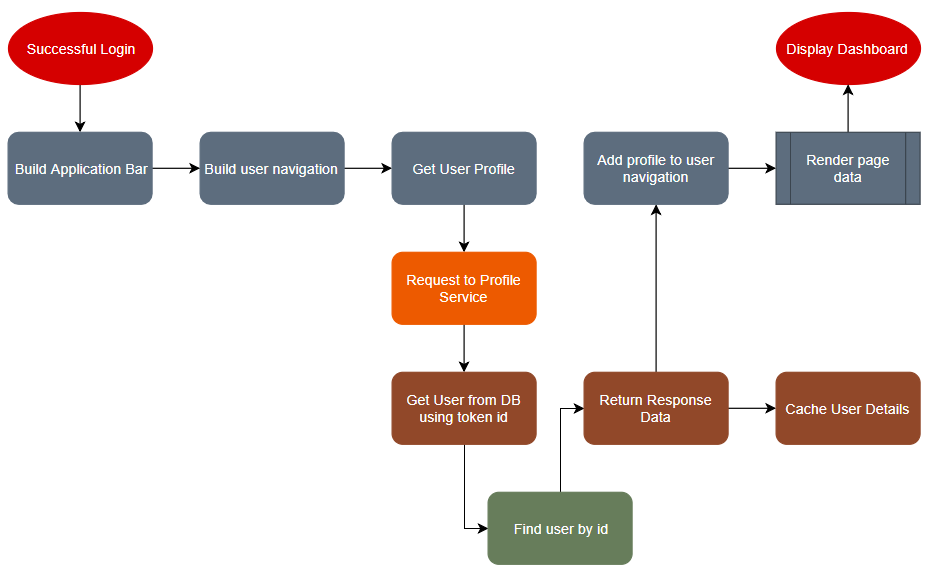


### Render Dashboard

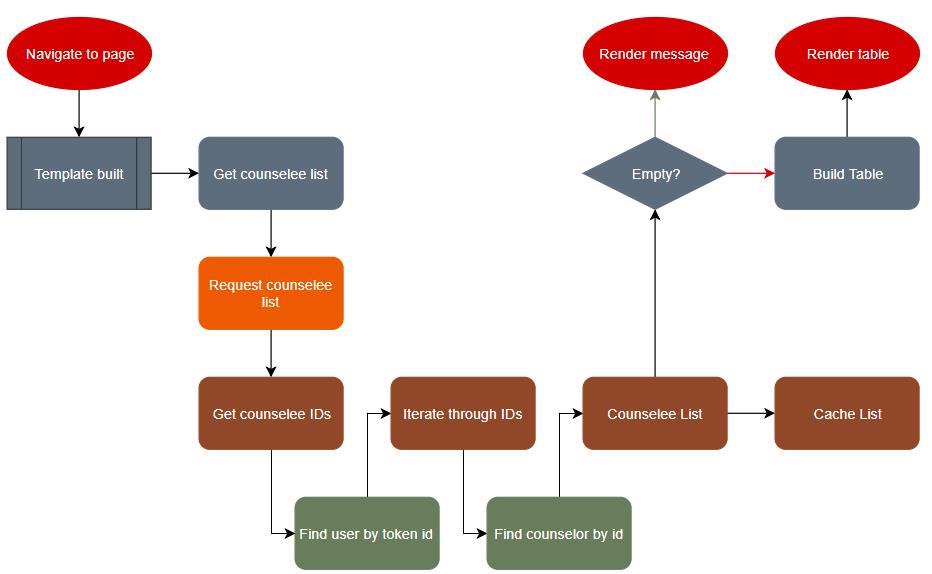
Overview



Template

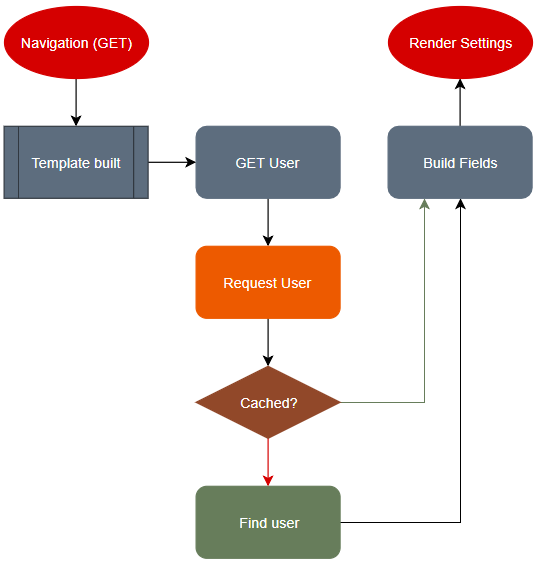


Counselee List

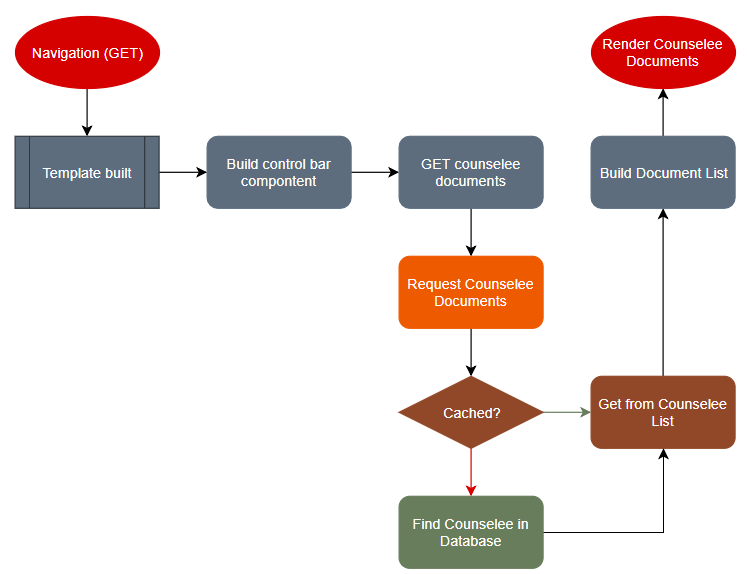


### User Details

N.B. Counselee Details and Counselor Details similar process and not diagramed.



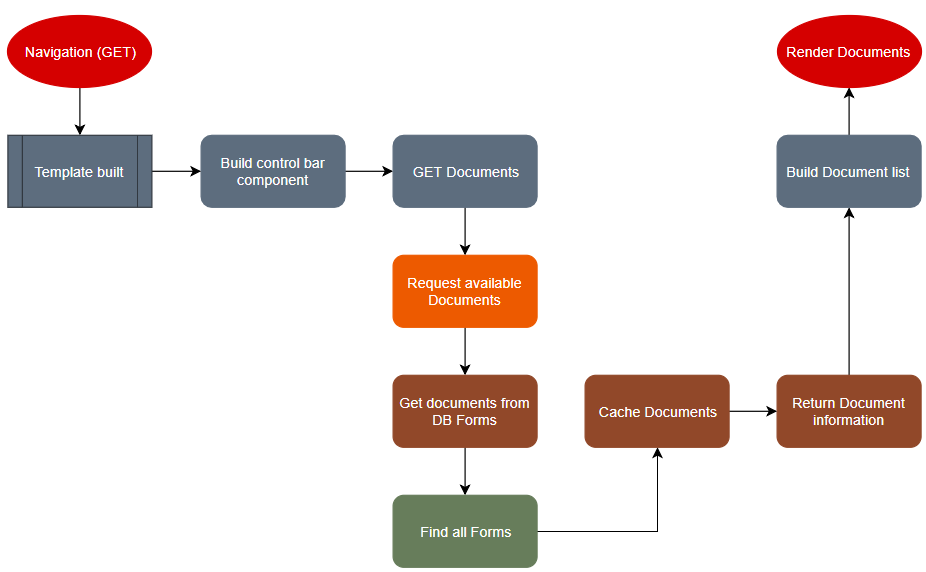
### Counselee Documents



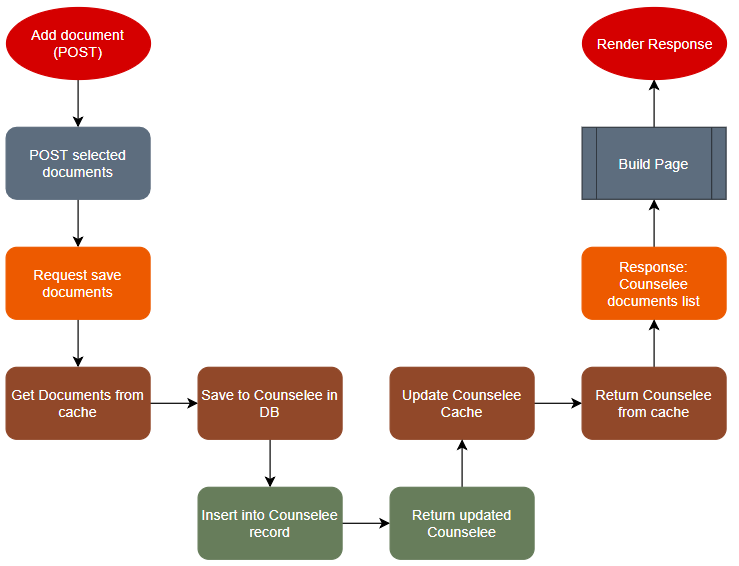
### Add Document

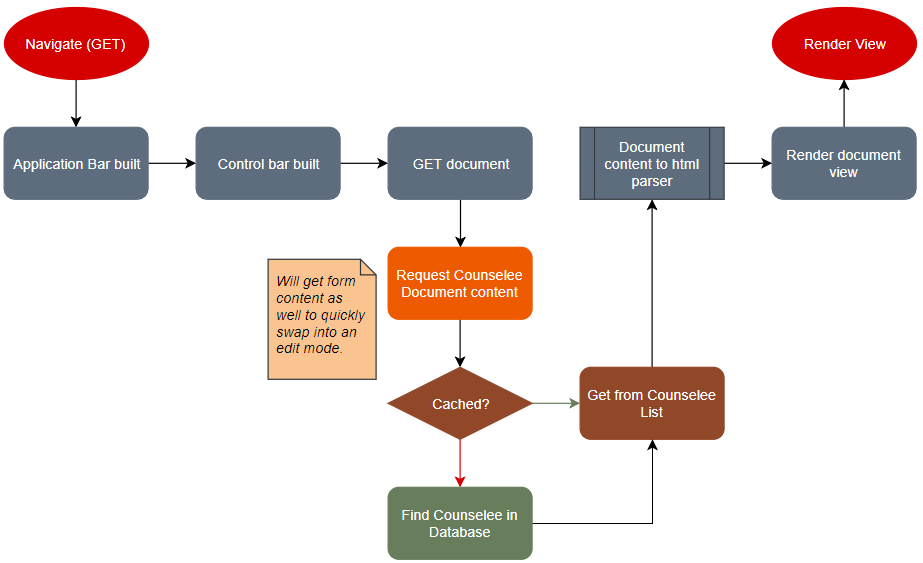
*N.B. Add resource to counselee process similar and not diagramed.*

GET

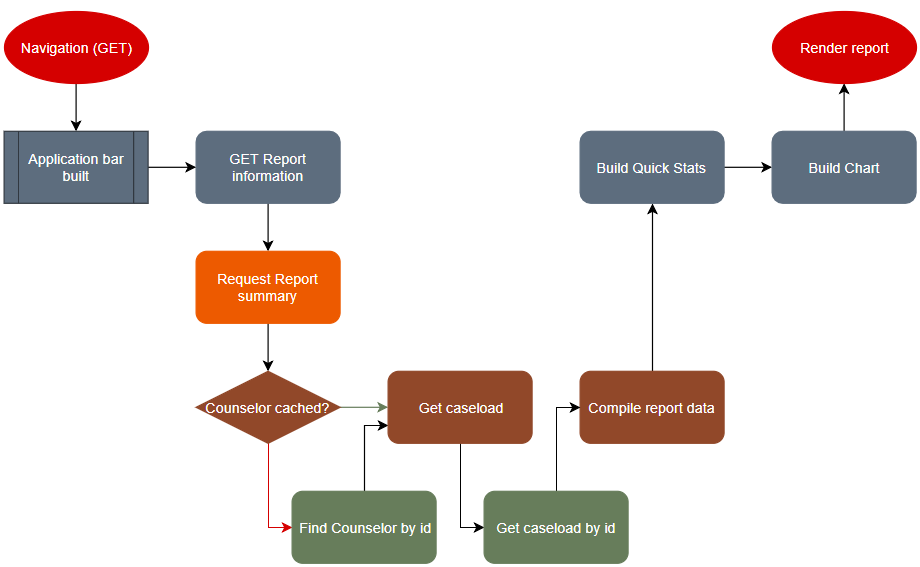


POST



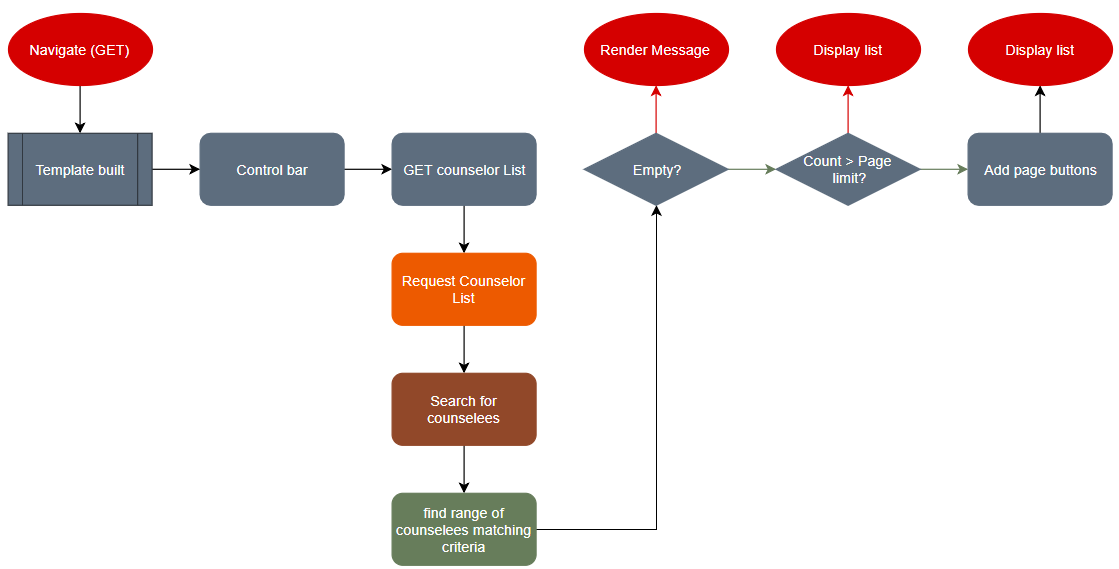
Document Viewer/Editor

### Get Report



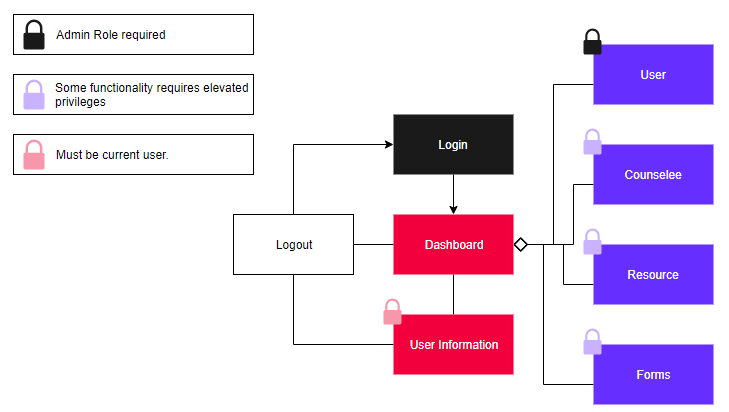
### Counselor List

N.B. *List resources process similar and not diagramed.*

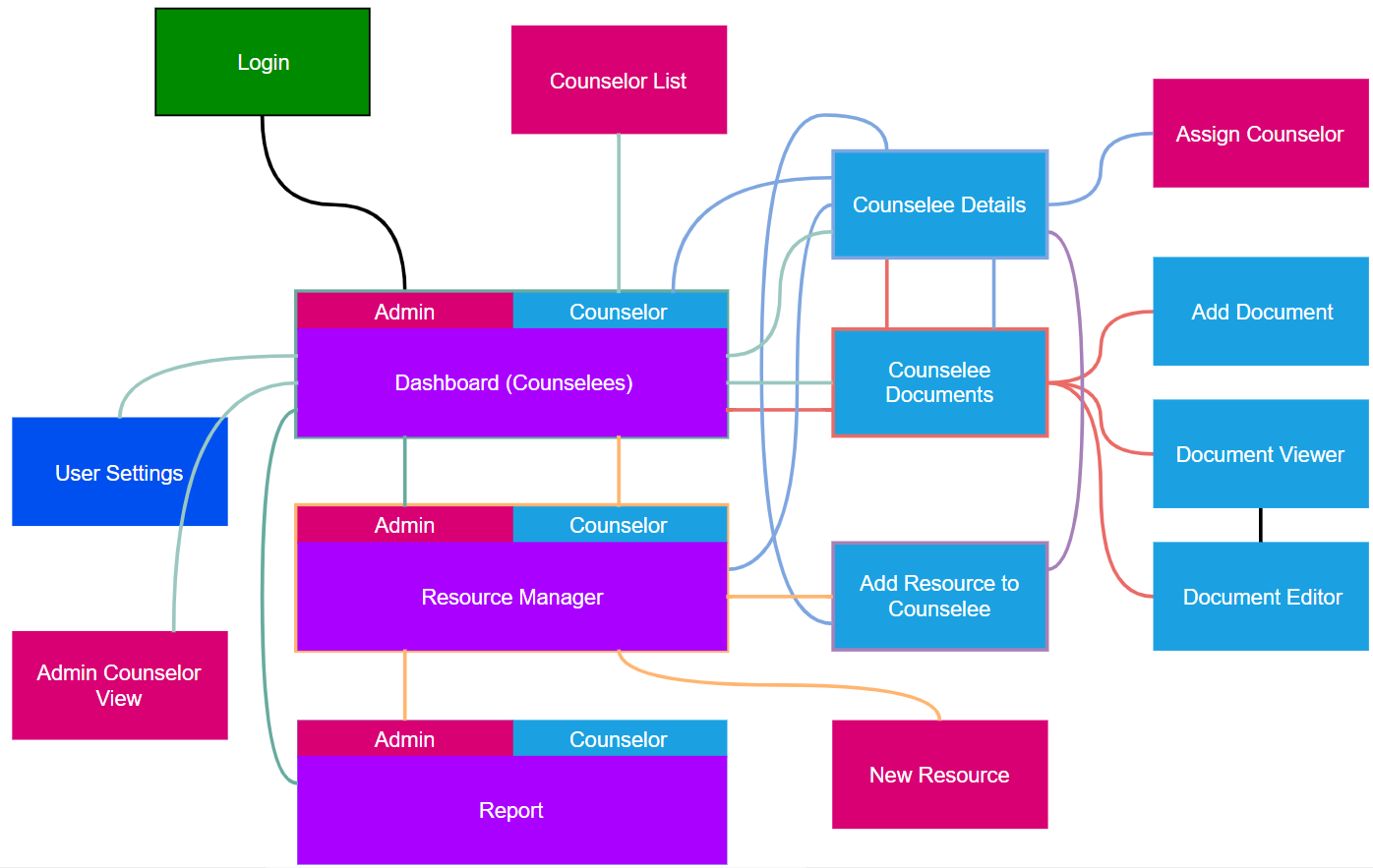


## Sitemap Diagram:

### Version 0.1.0



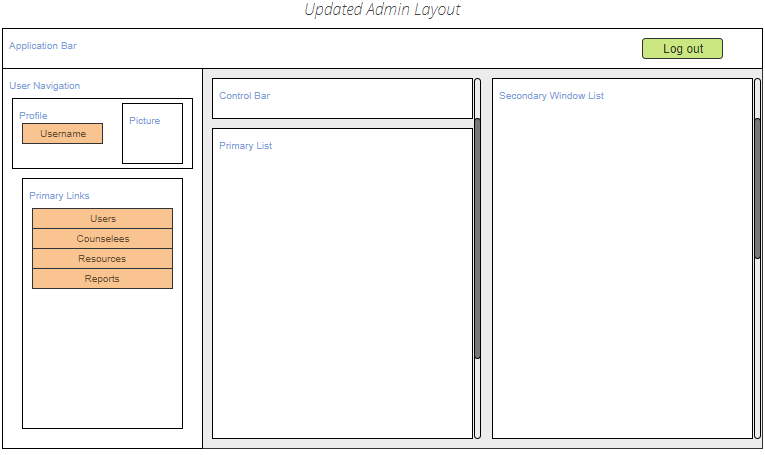
### Initial Version

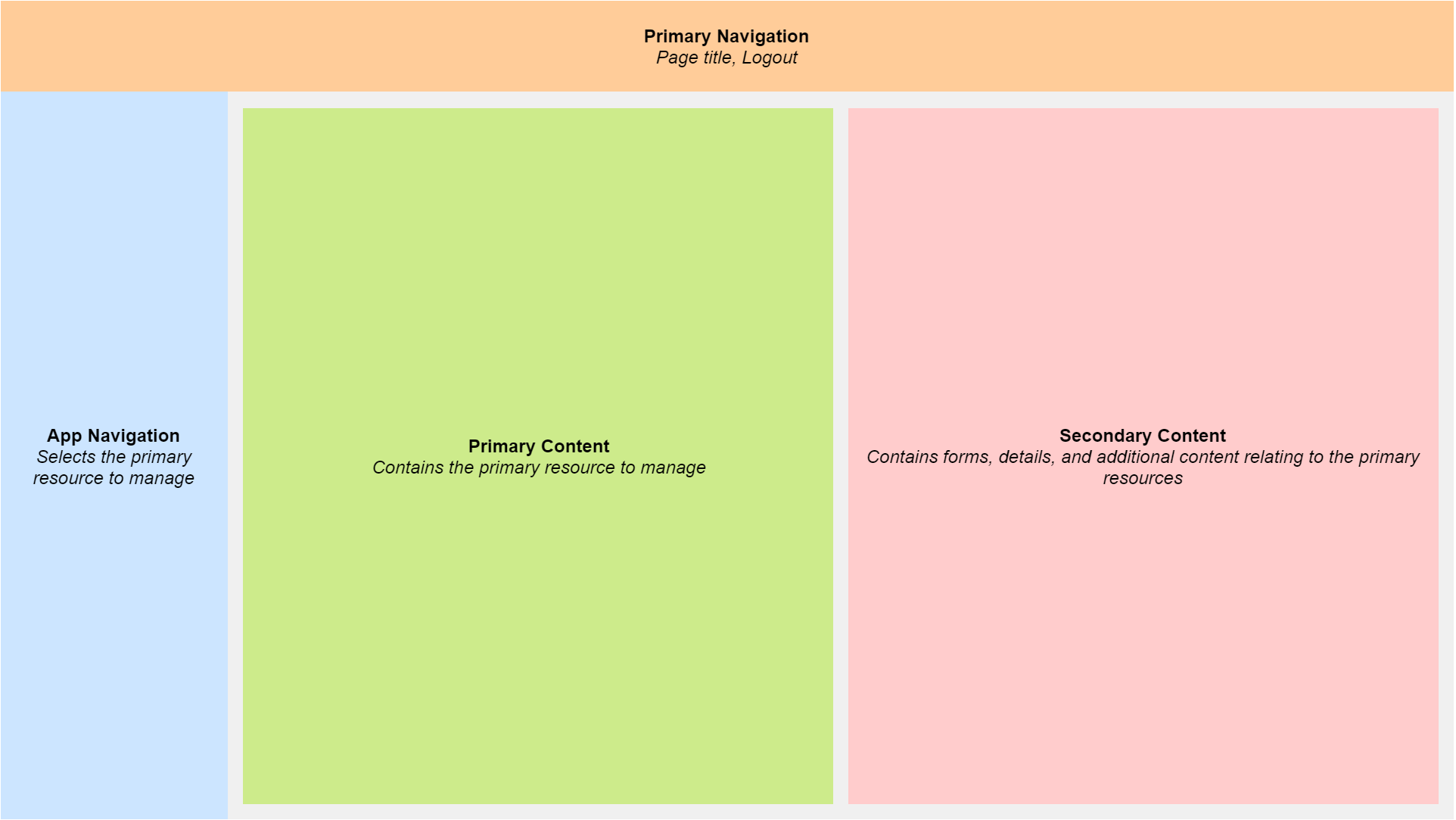


Note: all routes are two-way. Logout functionality is permissible from any point within the application.

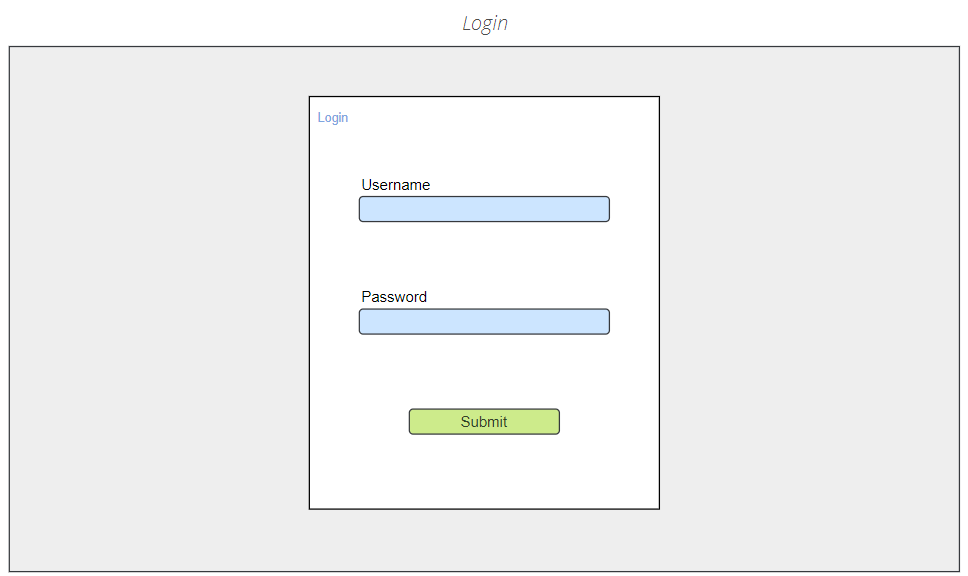
## User Interface Diagrams:

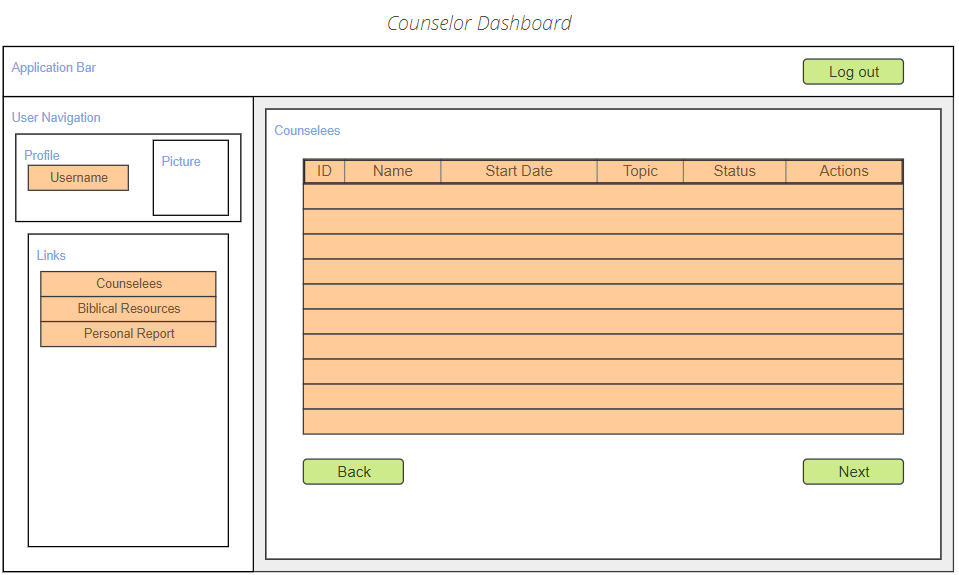
### Version 0.1.0

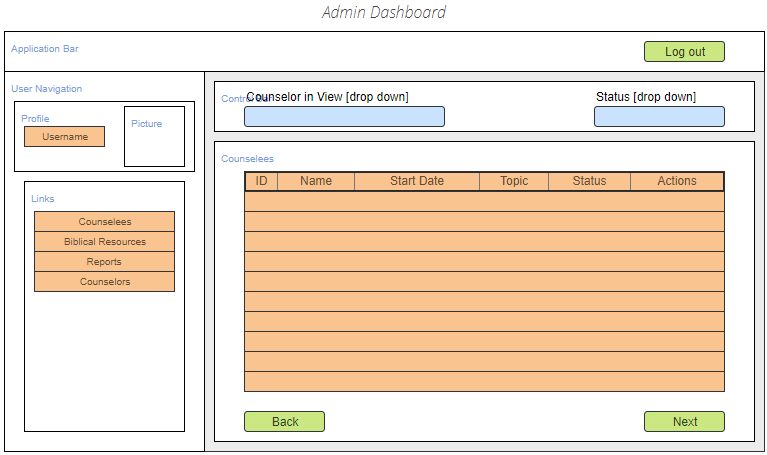


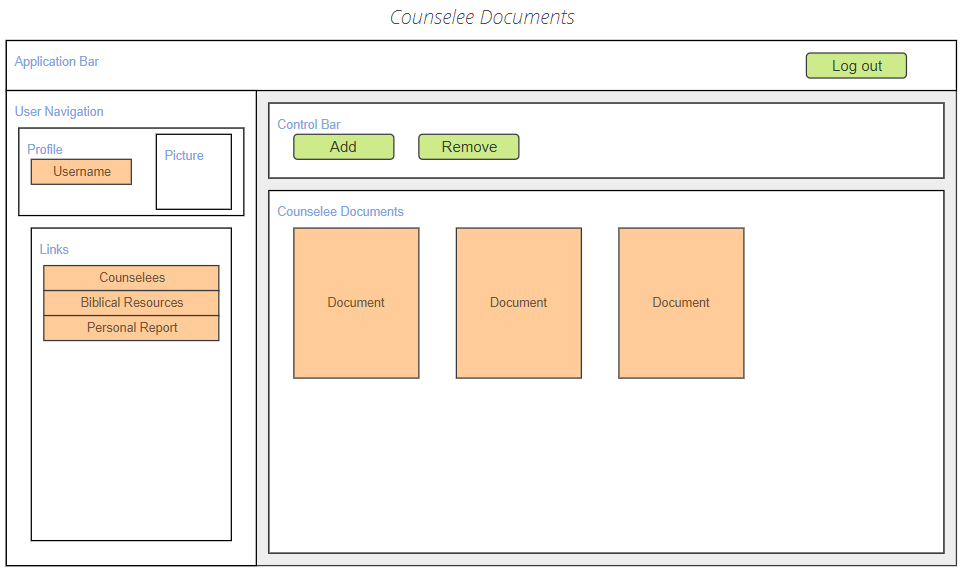


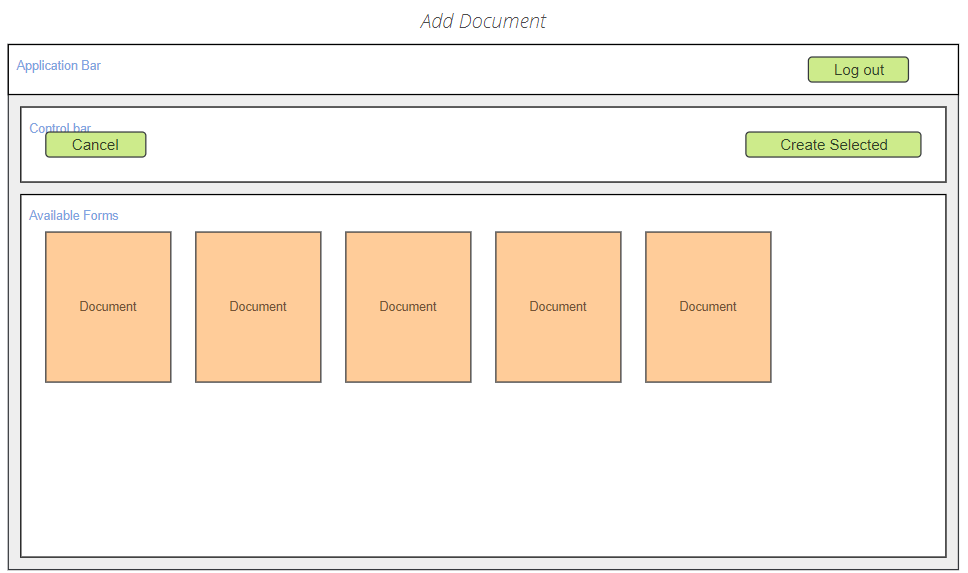
### Initial Version

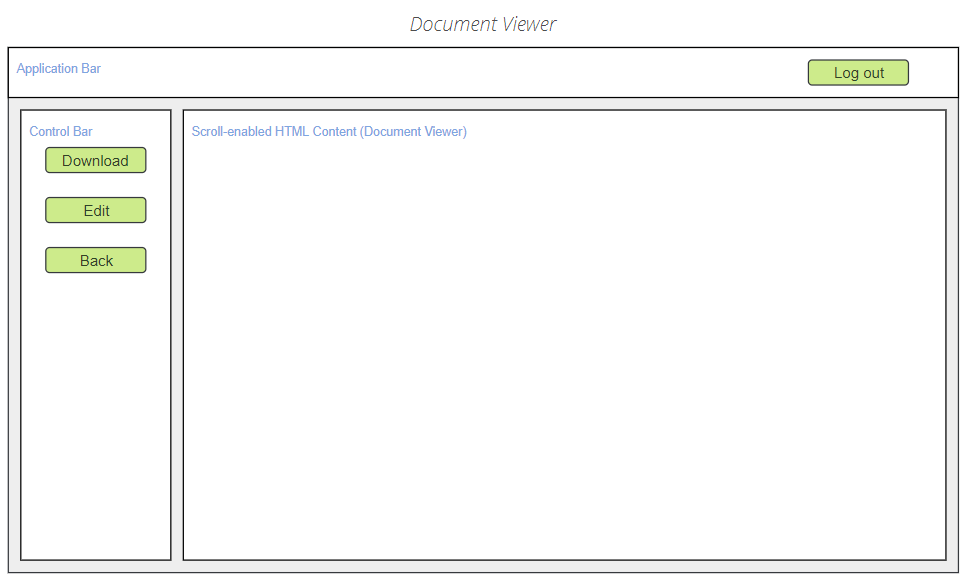


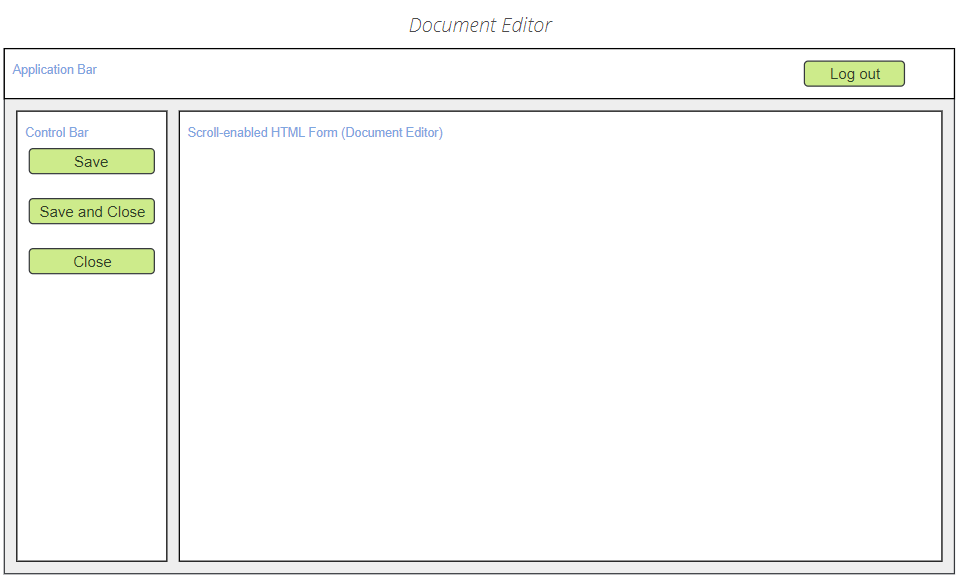


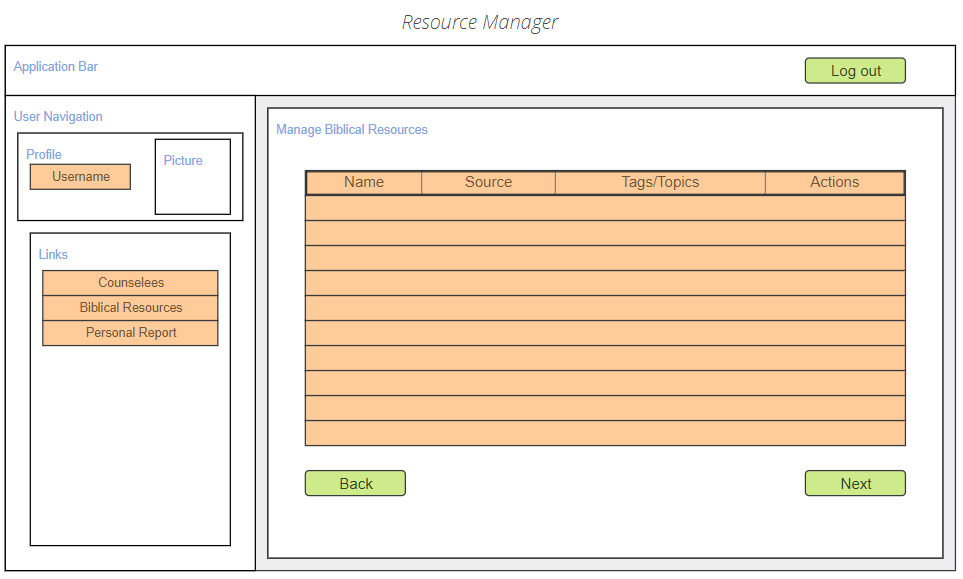


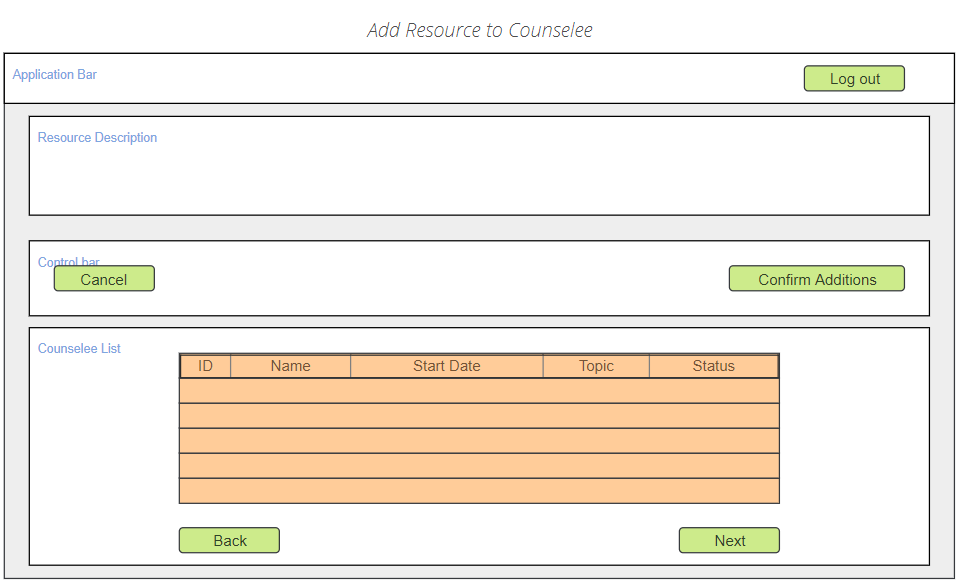


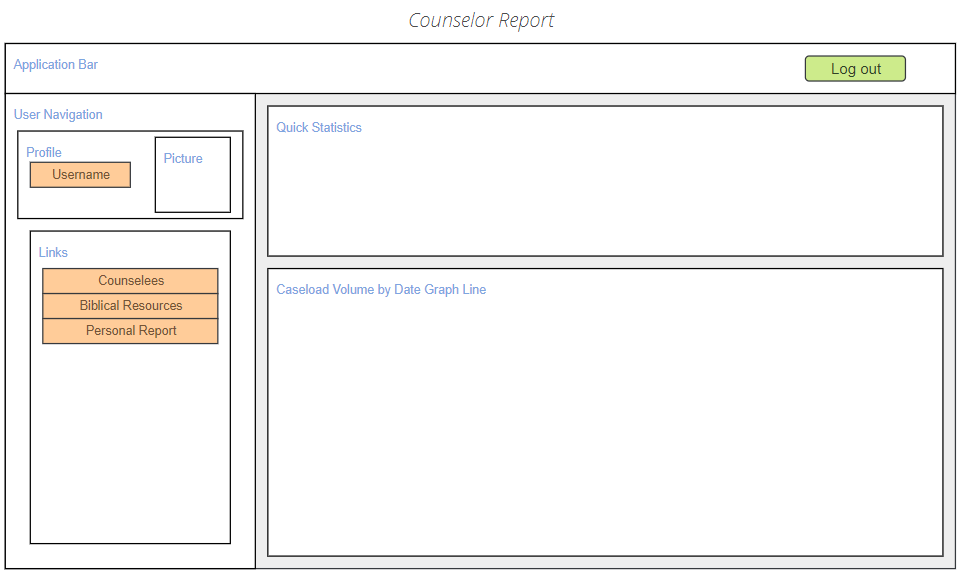


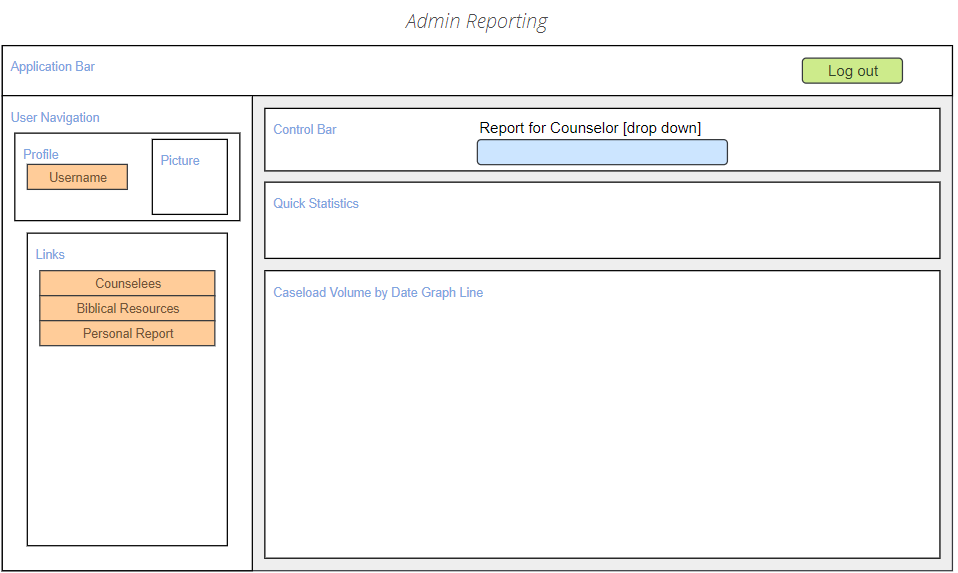


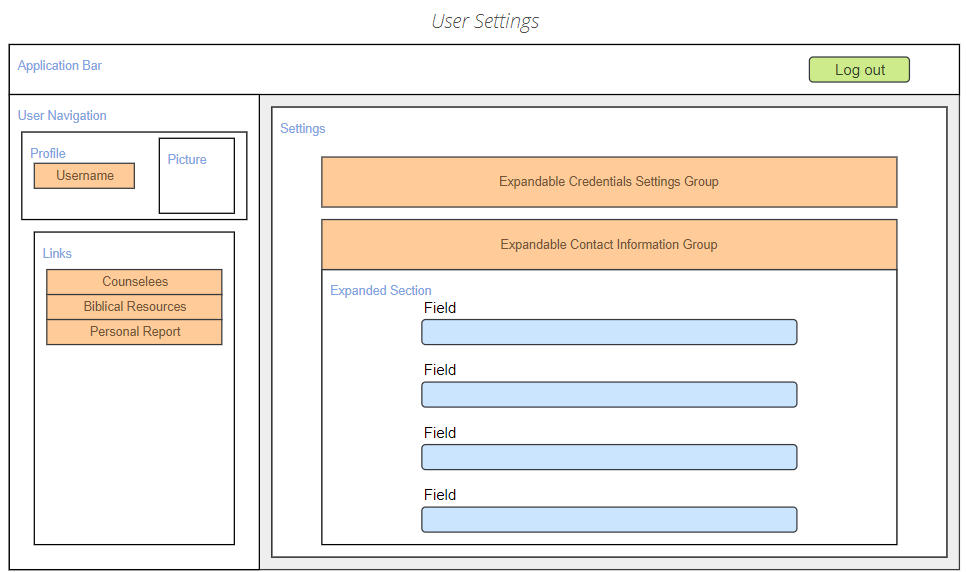


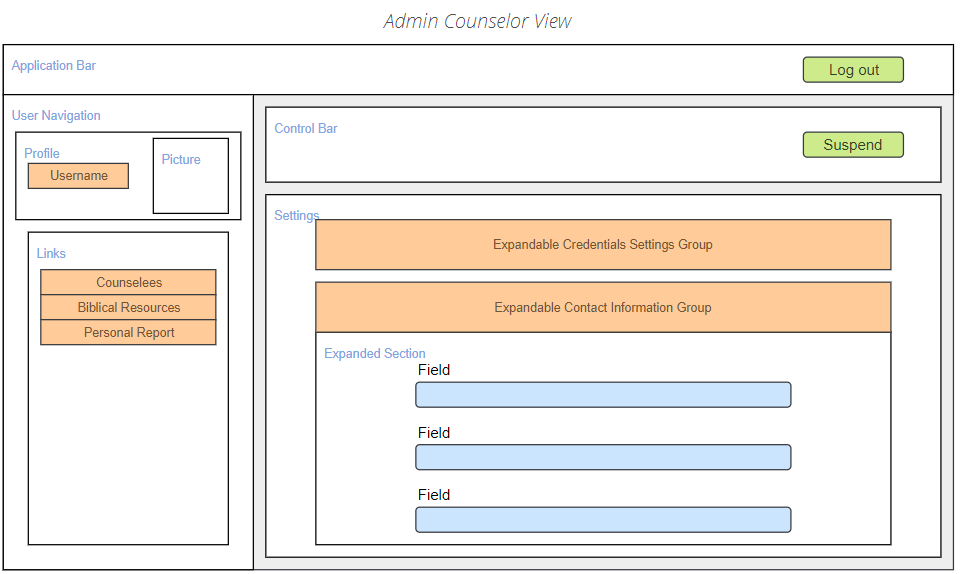


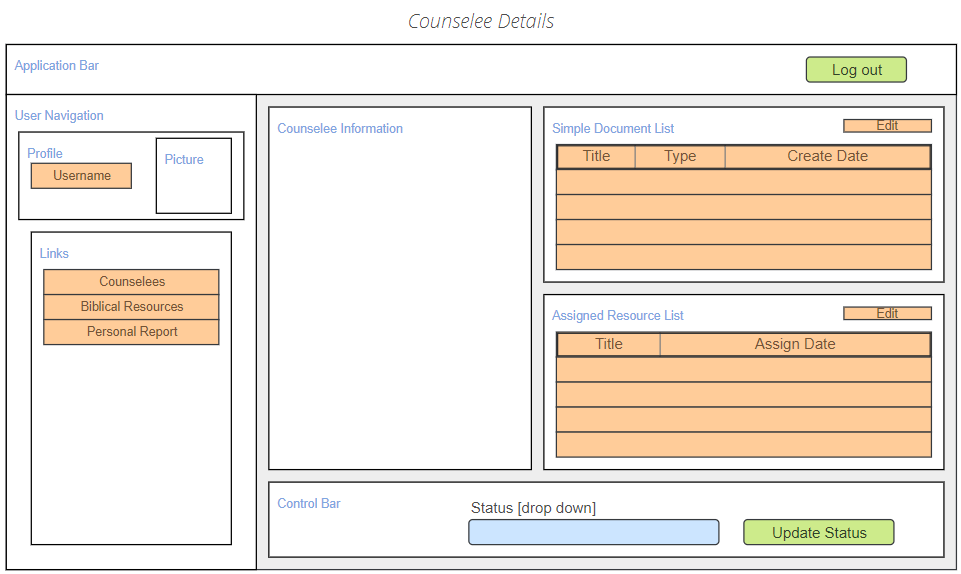


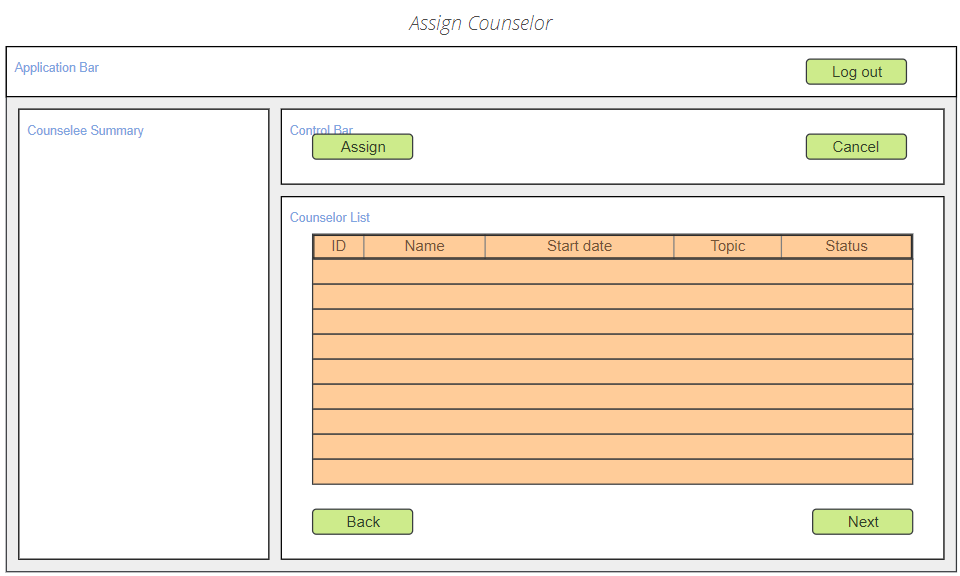


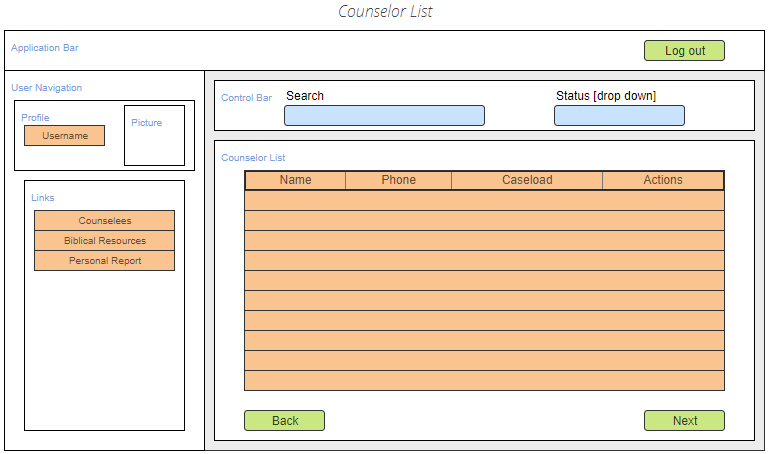


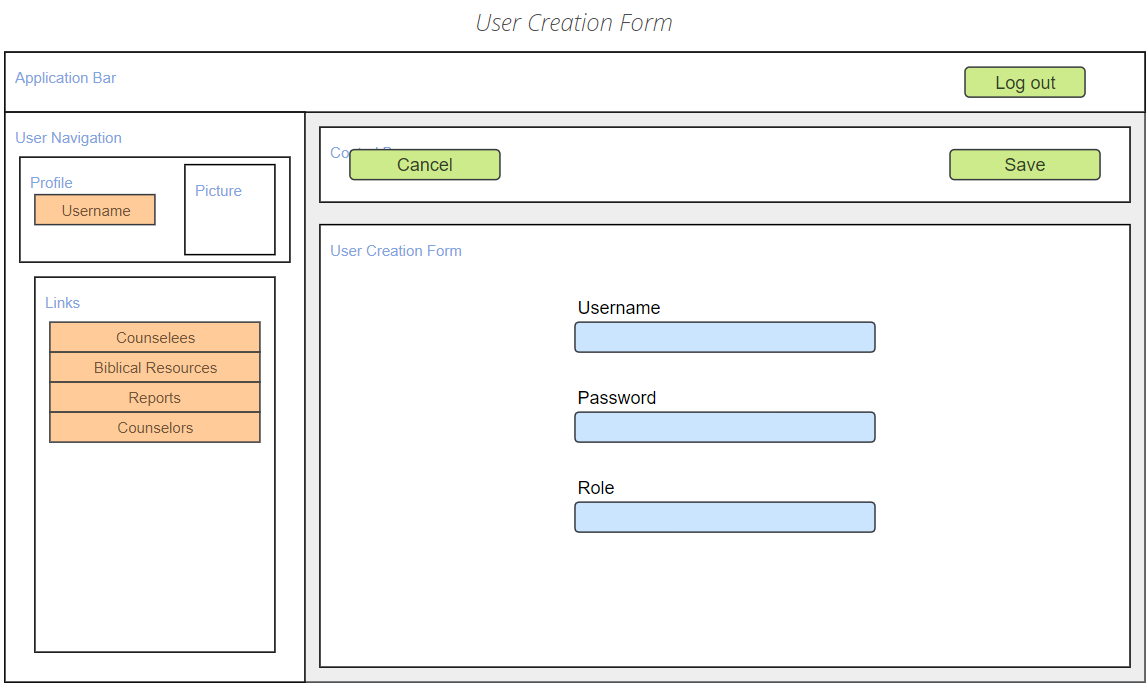












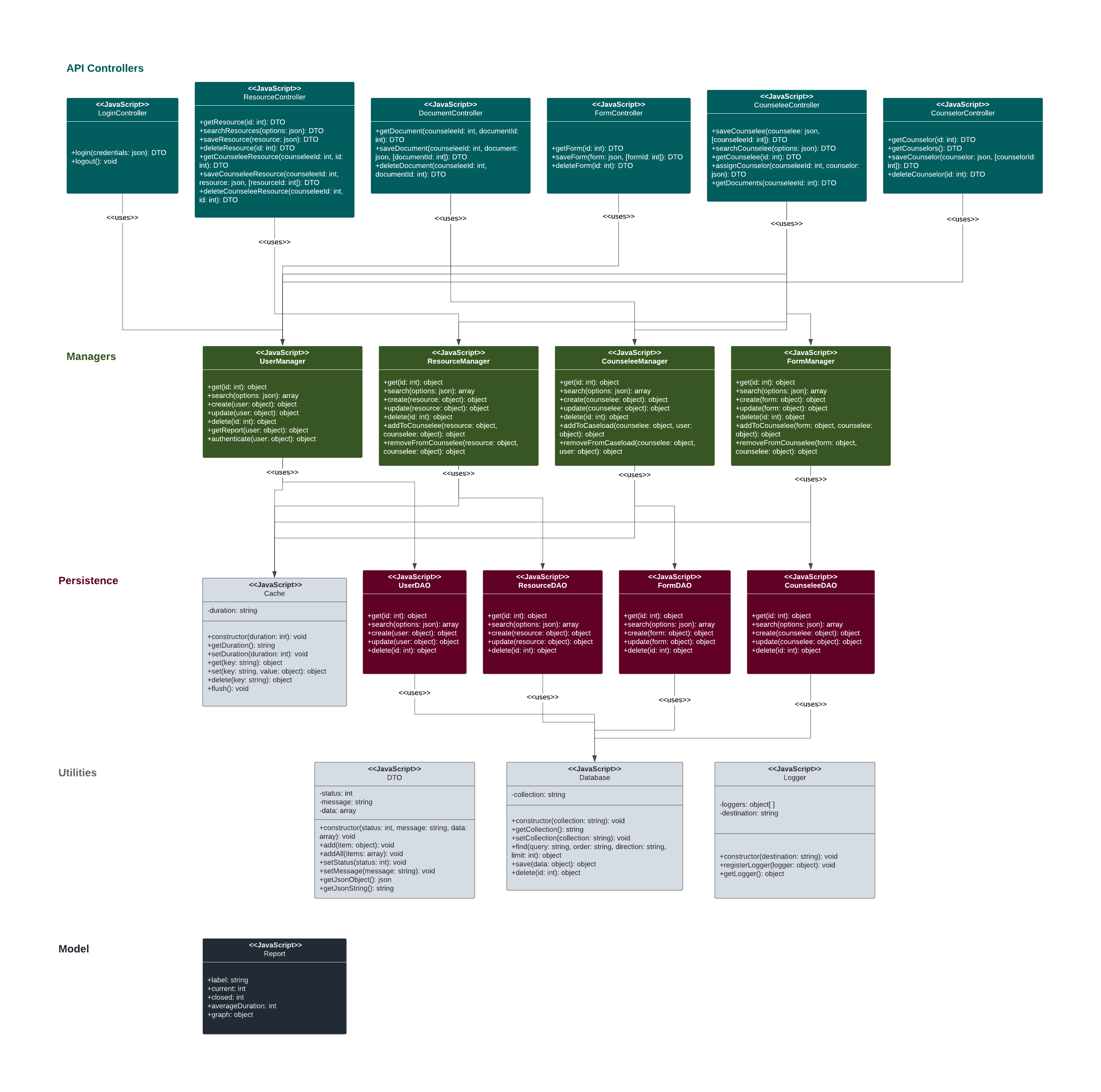
## UML Diagrams:

### Version 0.1.0

A screenshot of a video game

Description automatically generated

### Initial Diagrams



## Service API Design:

Express.js will be used to create an internal API for the application. The API provides a full REST service for reading and modifying the main application data models, in addition to a few helper methods for accessing some sub-objects, such as a counselee’s documents. The documentation for this API can be found on Swagger: https://app.swaggerhub.com/apis-docs/ConnorJamesLow/Proverb/1.0.1.

## Non-Functional Requirements:

### Availability

The application should be available for 105 hours each week during work hours (5 A.M. – 8 P.M.). The application should support usage for at least 20 employees.

* The office server should always be running at the office.
* The current counseling staff who would be using the application will be less than 20 employees. The current server usage is light.

### Backup:

The database should create daily backups.

* The MongoDB installation includes a mongodump command. A background process will run daily to create backups.

### Compliance/Privacy:

The software must comply with PI data laws and regulations defined in the HIPAA act.

N.B. It is possible that the PI form used in the counseling process does not need to be entered into the application; if this is the case, HIPPA will no longer be a concern for the initial product. This should be confirmed for the final version of this report.

* To comply with HIPPA, which requires reasonable threat anticipation, the application will be tested for confidentiality, integrity, and availability of all PHI. A cyber-security student or faculty should be consulted during the testing phase of the project.
* For security implementation, see **Security**.

### Documentation/Scalability:

The application is expected to grow in the future. The Client may require a scope change necessitating a rewrite of certain parts of the application; documentation will smoothen these changes. Additionally, the project has potential to become a COTS application.

* In addition to this design document, code will be commented throughout to clarify sections that are not self-explanatory.
* Each layer of the application will use full encapsulation to ensure low coupling.
  + The Express application will be designed as an API fully separable from the front-end React Application.
  + The separation and encapsulation of managers (business logic) and data access objects (persistence logic) will permit future expansion.

### Logging:

All data-modifications should be logged.

* A logging service will be implemented and utilized in the business layer.
* See **Operational Support Design** for implementation details.

### Security:

N.B. It is possible that the PI form used in the counseling process does not need to be entered into the application; however, the application should use the same security features regardless.

The application data should maximize the confidentiality, integrity, and availability of data within the application. Data will be encrypted. Data will not be altered without record. Data only will be available to counselors and administrators.

* OAuth will be implemented with JWTs.
  + Except for the login and logout API paths, all routes are protected and require a valid JWT.
  + The implementation of JWTs should follow some best practices, discussed in <https://auth0.com/blog/a-look-at-the-latest-draft-for-jwt-bcp/>:
    - The most recent version of 0Auth’s jsonwebtoken library will be used to implement the most up-to-date security and defend against vulnerabilities found in earlier versions and libraries.
    - To prevent “alg: none” attacks, explicitly define in the code which algorithm to use and consider all other JWTs to be invalid.
    - Since the API will be internal, no public keys and HS algorithms will be used to manage data accessibility.
    - Validate all claims match expected parameters (e.g. verify the JWT doesn’t contain claims that are not used in the application).
    - Keys will be at least 32 characters long and avoid dictionary words.
* Flush cookies and cache data when users log out, or a timeout occurs.
* For data integrity, see **logging**.
* All data at rest (MongoDB) will be kept confidential through using encryption built in to Mongo.
* Passwords will be hashed.

### Usability:

The target client has moderate computer experience. The application should logically flow from a business standpoint.

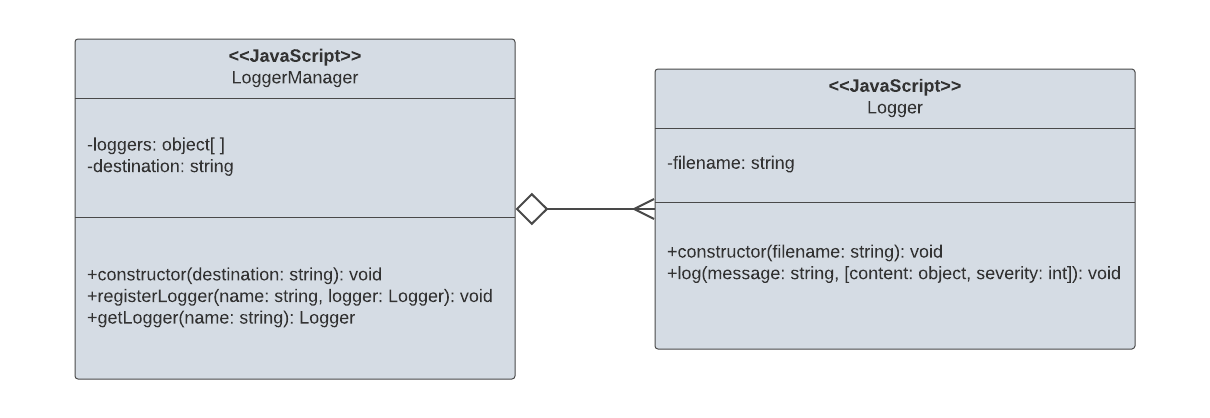
* Ship Shape’s Shepherd JavaScript library provides a powerful library for creating interactive tutorials on a site. A script using the Shephard library will be used to guide new users through the application.
* Like with counseling, the main data models in the application are counselees (resources and forms attach to counselees, counselee caseload appears as the default view with the dashboard).
* CSS design can be used to draw attention to controls by using sufficiently contrasting colors.

## Operational Support Design:

Logger utility classes will be used to create log files. Three loggers will be created:

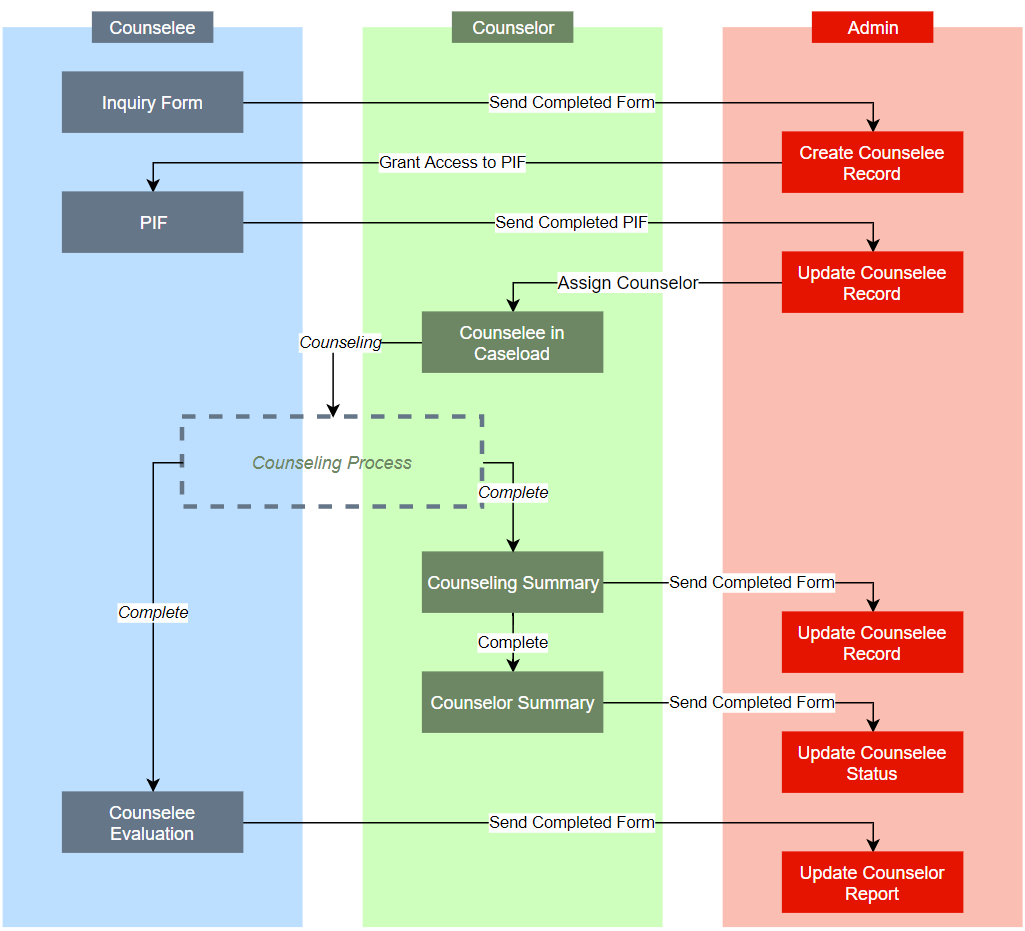
* System logger – creates messages about the system for development and maintenance purposes.
* Activity logger – creates messages and records the state of content as users make changes with the application.
* Exception logger – specifically logs uncaught exceptions.

These will be managed by a LoggerManager. Each logger can be accessed by a string “name” value. Each logger will write to the same folder but create files with different names.



## Other Documentation:

### Process overview



# Appendix

## Appendix A – Technical Issue and Risk Log

1. Use the template to identify and monitor project issues and risks.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issues and Risk Log | | | | | | | | | |
| Issue or Risk | Description | Project Impact | Action Plan/Resolution | Owner | Importance | Date Entered | Date to Review | Date Resolved |
| R | Unsure whether application will need to manage PI after all. Should discuss further with client | Low. | I will be able to meet with the client during the week of 11/18-11/25 | The developer | 1 | 11/11/18 | 11/25/18 |  |
| R | Unfamiliar with using JWTs in Node. | Medium. |  | The developer | 2 | 11/11/18 | 12/1/18 | 2/1/19 |
| R | Unfamiliar with Mongoose library for MongoDB | Medium. | Update current PoC to use Mongoose instead of directly accessing the MongoDB driver | The developer | 3 | 11/11/18 | 12/1/18 | 12/1/18 |

## Appendix B - References

Atto, E. (2018, April 30). *A Beginner’s Guide to React with ES6*. Retrieved from Medium: https://medium.com/the-andela-way/a-beginners-guide-to-react-with-es6-a2ed0b5c977e

Chart.js. (2018). *Chart.js*. Retrieved from Chart.js: http://www.chartjs.org/

Facebook, Inc. (2018). *React*. Retrieved from ReactJs: https://reactjs.org/

GitHub, Inc. (2018). *facebook/react*. Retrieved from GitHub: https://github.com/facebook/react

GitHub, Inc. (2018). *jsonwebtoken*. Retrieved from GitHub: https://github.com/auth0/node-jsonwebtoken

GitHub, Inc. (2018). *mongoose*. Retrieved from GitHub: https://github.com/Automattic/mongoose

GitHub, Inc. (2018). *nodecache*. Retrieved from GitHub: https://github.com/mpneuried/nodecache

GitHub, Inc. (2018). *nodejs/node*. Retrieved from GitHub: https://github.com/nodejs/node

Lawrence, B. (2017, March 23). *5 Vital Tips for Developing HIPPA Compliant Mobile Apps: A Checklist*. Retrieved from NowSecure: https://www.nowsecure.com/blog/2017/03/23/5-vital-tips-developing-hipaa-compliant-mobile-apps-checklist/

Linux Foundation, Node.js Foundation, Joyent, Inc. (2018). *NodeJs*. Retrieved from NodeJs: https://nodejs.org/en/

McLean, T. (2015, March 13). *Critical vulnerabilities in JSON Web Token libraries*. Retrieved from Auth0: https://auth0.com/blog/critical-vulnerabilities-in-json-web-token-libraries/

MongoDB. (2018). *Queries*. Retrieved from MongoDB Node.JS Driver 1.4.9 documentation: https://mongodb.github.io/node-mongodb-native/markdown-docs/queries.html

MongoDB, Inc. (2018). *db.collection.find()*. Retrieved from MongoDB: https://docs.mongodb.com/manual/reference/method/db.collection.find/#db-collection-find

MongoDB, Inc. (2018). *Json and Bson*. Retrieved from MongoDB: https://www.mongodb.com/json-and-bson

npm, Inc. (2018). *mongoose*. Retrieved from npm: https://www.npmjs.com/package/mongoose

npm, Inc. (2018). *path-to-regexp*. Retrieved from npmjs: https://www.npmjs.com/package/path-to-regexp

Peyrott, S. (2018, April 11). *A Look at The Draft for JWT Best Current Practices*. Retrieved from Auth0: https://auth0.com/blog/a-look-at-the-latest-draft-for-jwt-bcp/

Ship Shape. (2018). *Shepherd*. Retrieved from Shepherd: https://shipshapecode.github.io/shepherd/docs/welcome/

StrongLoop, IBM, and other expressjs.com contributors. (2018). *Express*. Retrieved from Express: https://expressjs.com/

U.S. Department of Health & Human Services. (2013, July 26). *Summary of the HIPAA Security Rule*. Retrieved from HHS.gov: https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html

## Appendix C – External Resources

GitHub Repository: <https://github.com/ConnorJamesLow/capstone>

Swagger API docs: https://app.swaggerhub.com/apis-docs/ConnorJamesLow/Proverb/1.0.1