CIS 4328: Information Systems Senior Project II

Final Project Report

Quality Measure Dashboard System and Administrator File Application

Development Team: *Waxwings*

Hunter Blake
JohnArmon Antolin
Trey Cox
Wilson Jativa

Instructor: Karthikeyan Umapathy

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

May 2, 2022

Table of Contents

Final Report

1. Executive Summary

1.1 Summary

Waxwings group developed two applications for MASS Clinic: the Quality Measure Dashboard System and the Administrator File Application. The Quality Measure Dashboard System displays graphs of MASS Clinic's patient data while the Administrator File Application allows an admin user to upload a file of patient data so that it is graphed and displayed on the Quality Measure Dashboard System application. Waxwings began the development process from August 2021 to May 2022 and completed both applications by the planned deadline of the end of April 2022.

The MASS Clinic provides free, volunteer-based care for non-emergency cases of acute or chronic diseases. The problem of unclear information regarding patient medical results affects MASS Clinic's impact of gaining potential donors and representing the data in a more visual manner. The major requirements of a successful solution would allow an administrator user to upload an excel file that contains the patient data, the data would then be stored into a database successfully, a website application will graph all the patient data properly by their respective categories, each graph should have a drilldown option that is based on year if appropriate to that data, and the design should be obvious for potential donor users when interacting with the elements on the page. The Dashboard application follows a MVC model, more specifically a MVR model where the routes takes the role of the controller in the conventional Model-View-Controller model. The Dashboard application and Administrator application communicate on a 3-Tier Client Server model where the Dashboard application requests and receives data from a server and the server requests and receives data from a database. The Administrator application communicates with the database through a pipe-and-filter model because the donor user has no need to be aware of the administrator user side of the product. The entire product uses HTML 5, CSS 2.1, JavaScript ES5, the Node.JS and Angular.JS frameworks, and Microsoft Azure Services; the main web browser used is the latest Google Chrome.

The initial test results showed that a majority of the functionality for the requirements were achieved with two exceptions: one chart on the Dashboard application was not rendering and a part of the Admin Application contained an unnecessary red box. These issues have been corrected. The client also suggested reorganizing the Last Year Seen data, HTN Diabetic data and the Outcome data; these changes were implemented before the last delivery. With the support of our mentor, Raymond McDermott and reassurance of our client from MASS Clinic, Faisal Sayed, the product with the expected requirements of the client was delivered which solves the need of an application that is able to visually display patient quality data from an Excel file. Waxwings had four main key learnings from this project. First, the student development team learned the importance of consistent and concise communication. The student team had an open channel of communication that was available at any time and the team met in person at least twice a week to check on progress and speak about what the next objectives to be completed were. Waxwings also met with their mentor and client every other week to make sure that their vision, goals, and completions were on track with the client's requirements. Second, Waxwings learned how to develop software as a team. A majority of software development

is team development, and this experience taught the student team how to complement each other's perspectives, soft skills and hard skills. The student development team learned how to collaboratively work and plan accordingly depending on time, place and pace of work. From working on this process from the Fall and Spring semesters, Waxwings learned and experienced the entire software development process so as to be familiar with it in future work. Waxwings gathered and analyzed the client's requirements, designed the product, coded and implemented two applications, tested both programs and deployed the product to MASS Clinic. This third learning taught Waxwings the importance of documentation along this software development process as well which was an idea supported greatly by the teams' mentor. Finally, Waxwings last key learning was the improvement of their technical skills especially in HTML, CSS and JavaScript. The team also used Node.JS as a runtime environment and Angular as a framework in the admin application. ReactJS was also used on the dashboard. MongoDB was used for the admin backend. Familiarity in using a library was also implemented with the use of Chart.JS for the dashboard graphs. SQL skills were also tested and improved with the use of Microsoft's Azure database. Waxwings also learned how to host and deploy the product using Microsoft's Azure services.

2. Introduction

2.1 Community Partner/Client

Waxwing's client was MASS Clinic with Faisal Sayed as the product owner. MASS Clinic provides free, volunteer-based care management to the uninsured population of Duval County. By collaborating with local healthcare institutions, We Care clinics, and other faith-based organizations, our work improves public health, lowers healthcare costs and reduces the burden of non-emergency care and the treatment of acute and chronic diseases placed on local hospital emergency rooms.

2.2 Problem Statement

MASS Clinic as a volunteer-based care facility relies on donors and philanthropy to sustain their work. Donors and possible donors should have the opportunity to view the patients' quality data so as to make better decisions on their donorship. This data was previously organized on the MASS Clinic website in data tables with just labels and numbers. The problem of unclear information regarding patient quality outcomes affects MASS Clinic in which it was more difficult to convert potential donors into donors. A successful solution would be to create a dashboard that displays and organizes dynamic data on patient outcomes into more visually engaging graphs.

2.3 Background

In this situation, the business opportunity is less about monetary profit but more focused on gaining people to join the MASS Clinic cause which is to provide free care management to the uninsured population in the local area. The key market of people this product is trying to reach is people and foundations looking to donate to nonprofits. People within this category reach from givers of smaller donations, often younger within the 20-39 age range, and donors of larger donations which tend to be in an older demographic form 39 and up. The reason for this is individual donation is often dependent on personal income which is correlated with age. Foundations are hard to segment because often a foundation has to pass the places it proposes to donate through multiple hands that are in a number of different demographic groups.

The potential users of the product would be donors and possible donors that want to see the success and impact of MASS Clinic to the local patients they serve and an administrator user that will load the data. There are two kinds of donors:

- Individuals - These donors are most likely looking for places to donate from their home. They are spending a few minutes looking into each place they want to donate to. They are looking to know about the place they are thinking about donating to and what that place produces.
- Groups/Companies - These donors are often spending time in groups looking over the places they want to donate to. Portfolios are put together on the places they want to donate to. These conversations are most likely taking place in an office or over zoom.

Both of these kinds of donors currently have access to the info over call or through the chart available on the current website. The administrator user will likely work from an office and spend no more than a few minutes putting new data into the system. The existing system that MASS Clinic had was tables that contained labels and data which were hard coded by someone; MASS Clinic decided to remove that system and replace it with the Waxwing product.

2.4 Project Goals and Objectives

The deliverables of this product would create two applications. First, there will be a Patient Quality Measurement Dashboard that a donor can use that views all the patient data. These graphs will need to be grouped together into smaller dashboards and each graph should have the capability to be viewed by itself. If applicable to the data, the graph should have a drilldown option that is based on year. This Dashboard application should also have navigation capabilities. The second part of the product would be the Administrator File Application which is not accessible to the donor user. This application should have a sign-in option for the admin user, an Excel file upload capability and be able to create another admin user account. The data uploaded into the admin application should be the same data displayed on the donor dashboard application.

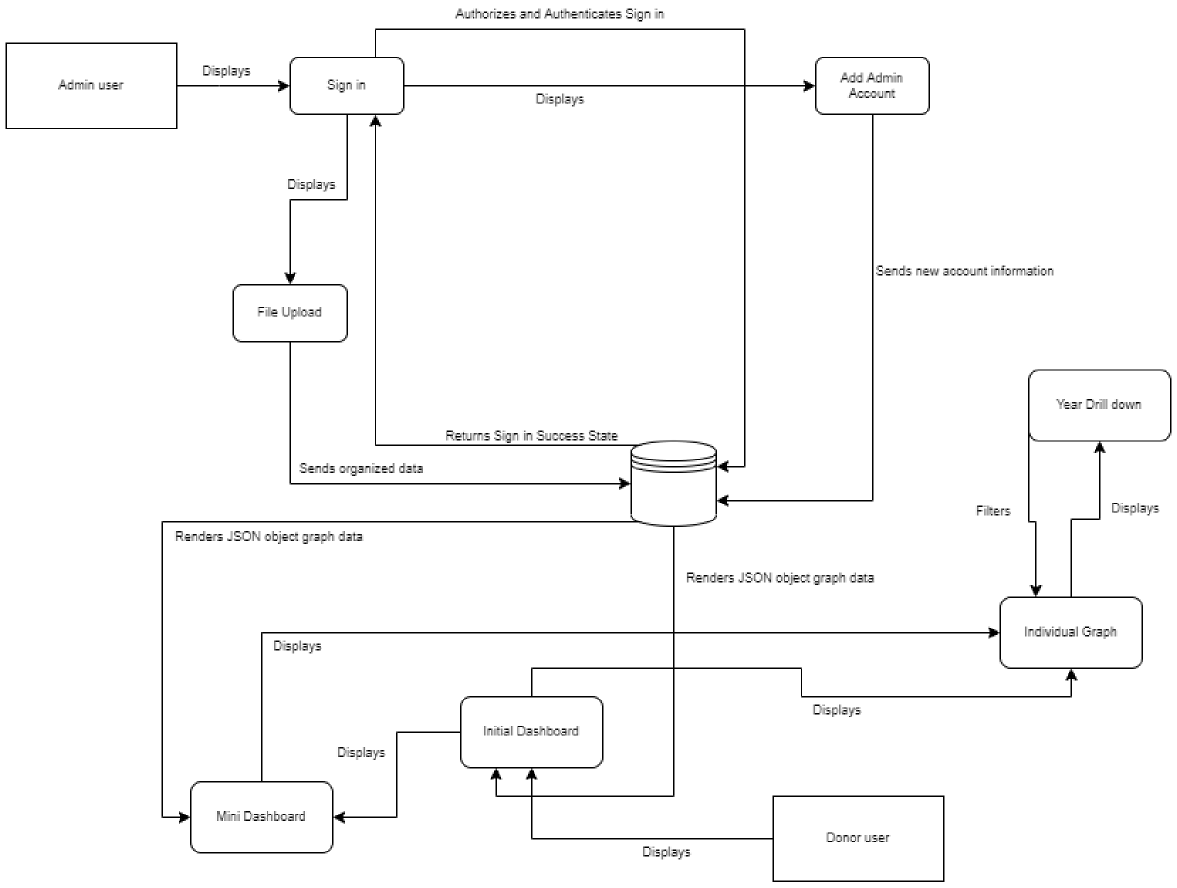
Waxwing's product visually organizes the patient data into graphs instead of just tables with labels and graphs which helps in better analysis of the data and identifying patterns more quickly. Donors and possible donors do not have to just read rows of numbers, but instead view the data story quickly and accurately so that they can decide about their donorship. This also allows MASS Clinic to have a quick system in checking their efficiency in their care and impact in the community.

2.5 Solution Developed

The solution developed by Waxwings are two web applications. In abstraction, the admin application was designed to have separate access from the dashboard application because the donor user does not need to know about how the data is inserted into the dashboard.

The Patient Quality Dashboard Application was created with a landing dashboard that contained certain graphs approved by the client. On that landing dashboard, there is a dropdown option where the donor user can choose to view on specific graph. There is also a navigation bar to other dashboards that contain graphs that are directly related to each other. On the landing and the other dashboards, the graphs are all organized into cards which have renderings of small graphs. The card's "see more" link can be clicked to view the graph on its own view page and if applicable, will have the implementation of drilldown option for choosing a specific year of data. This application solves the problem of just having data in tables and makes the data more visually appealing and easily understandable. This application used HTML 5, CSS 2.1, JavaScript ES5, Node.JS runtime environment and Chart.JS library. This application receives its data from the Microsoft Azure database. It was hosted using Microsoft's Azure services and most compatible to Google Chrome.

The Administrator File Application was created with a landing sign in page that prompts a user id and password; this capability has authentication and authorization capabilities. The following page after a successful sign in, is a file upload page which also checks if the file is in the proper format; if the file is formatted incorrectly, then the file errors will show and not load into the database. If successfully uploaded, a submission successful screen will show. This application also has the capability of adding another admin user. MASS Clinic expressed that less than 3 people will have this type of access. This application solves the issue of hardcoding data into the website graphs individually and renders all the data at the same time. This application used HTML 5, CSS 2.1, JavaScript ES5, and Angular as its framework. This application also uses MongoDB. This application sends its data to the Microsoft Azure database. It was hosted using Microsoft's Azure services and most compatible to Google Chrome.



2.6 Software Development Methodology

The software development process for Waxwings was organized through 7 deliverables which all contain either major artifacts for the product and/or a product release demonstration. The deliverables were:

1. Deliverable 1
    a. Artifacts
        i. Product Vision
        ii. Product Backlog
        iii. Product User Stories
        iv. Sprint Backlog
        v. Software Development Plan
    b. Product Release Demo
        i. All team members must have similar IDE configurations for the development of the project source code
        ii. Project source code repository created
        iii. All team members are connected to the repository and should have contributed to the source codebase
        iv. Work on a user story (should be different from the rest of the team members)
        v. The codebase for the user story should reflect the implementation of the MVC pattern and the team selected programming platform
        vi. The implementation of the user story may not be complete, but it should demonstrate the approach taken to address the requirement described in the user story
2. Deliverable 2
    a. Artifacts
        i. List of identified use cases and relevant user story alignment
        ii. Outline Use Case Specification
        iii. Complete Use Case Specification
        iv. User Interface Specifications (Sketches and Wireframes)
        v. Sprint Backlog
        vi. Updated artifacts
    b. Product Release Demo
        i. All team members must have achieved the product release 1 individual and team expectations.
        ii. All team members must have created a student account with Azure, AWS, or another relevant cloud service provider
        iii. Cloud resource for web server and database instance must be created as the team's stage environment
        iv. All team members are connected to the team cloud resources and should have pushed their codebase to the team's cloud resource
        v. Work on more than two user stories relevant to a use case (should be different from the rest of the team members)
        vi. The codebase for the user stories should reflect the implementation

of the MVC pattern and the team selected programming platform

vii. The implementation of the user story may not be complete, but it should demonstrate the approach taken to address the requirement described in the user story

3. Deliverable 3
    a. Artifacts
        i. Activity Diagrams
        ii. Analysis Model
        iii. Sequence Diagrams
        iv. Preliminary Database Design
        v. Sprint Backlog in JIRA
        vi. Product Backlog in JIRA
        vii. Updated artifacts
    b. Product Release Demo
        i. All team members must have achieved the product releases 1 and 2 individual and team expectations
        ii. All team members should be connected to and utilizing the team cloud web server and database instance resources
        iii. The team's cumulative codebase, at a minimum, must be addressing four key use cases
        iv. The team's cumulative codebase must be satisfactorily addressing at a minimum of 25% of the project's stated requirements
        v. At a minimum, work on one use case (should be different from the rest of the team members) and satisfactorily address 25% of the requirements
        vi. The codebase should reflect the implementation of the MVC pattern and the team selected programming platform
        vii. The codebase must be committed to the team git repository and deployed in the team's cloud resources
        viii. The implementation of the use case may not be complete, but a considerable portion of the stated requirements should be working (i.e., runnable and demonstratable)
4. Deliverable 4
    a. Artifacts
        i. Data Flow Diagram – Context Diagram
        ii. Data Flow Diagram – Level 0 Diagram
        iii. Architecture model
        iv. Detailed Design
        v. Package diagram
        vi. Class diagram for each package
        vii. Sprint Backlog in JIRA
        viii. Product Backlog in JIRA
        ix. Updated artifacts
    b. Product Release Demo
        i. Patient Quality Measurement Dashboard Application:
            1. Landing Dashboard

            2. Smaller Organized Dashboards
            3. Dropdown navigation
            4. Bar navigation
            5. 25% of graph views
            6. Graph data from database active
        ii. Administrator File Application:
            1. Angular framework is completed
            2. Sign in page created
            3. File upload page created without functionality
            4. Successful upload page created without functionality
        iii. Host both applications with Microsoft Azure Services

5. Deliverable 5
    a. Artifacts
        i. Database Design
        ii. User Interface Design
        iii. Hardware and Software Specification
        iv. Sprint Backlog on JIRA
        v. Updated artifacts
    b. Product Release Demo
        i. Patient Quality Measurement Dashboard Application:
            1. 75% of graph views
            2. 25% complete cards on dashboards rendering smaller graph views
        ii. Administrator File Application:
            1. Sign in page authorization and authentication on local device
            2. File upload page created with functionality
            3. Successful upload page created with functionality
            4. Create new admin user page without functionality
        iii. Host both applications with Microsoft Azure Services
6. Deliverable 6
    a. Artifacts
        i. Test Cases
        ii. Client Testing Summary Report
        iii. Deployment Plan
        iv. Sprint Backlog on JIRA
        v. Updated artifacts
    b. Product Release Demo
        i. Patient Quality Measurement Dashboard Application:
            1. 100% of graph views
            2. 80% complete cards on dashboards rendering smaller graph views
            3. Debug entire application
        ii. Administrator File Application:
            1. Sign in page authorization and authentication on hosted backend
            2. Create new admin user page with functionality

            3. Debug entire application functionality
        iii. Host both applications with Microsoft Azure Services and start testing with client environment
7. Deliverable 7
    a. Artifacts
        i. Final Report
        ii. Source Code files and instructions
        iii. Product Presentation Video
    b. Product Release Demo
        i. Patient Quality Measurement Dashboard Application completed
        ii. Administrator File Application completed
        iii. Both frontends and backends are hosted with Microsoft Azure Services and transferred and deployed to client

2.7 Academic Course Works

The Waxwings student development team used their previous experience and skills from other courses, work, and personal projects in developing the MASS Clinic product. The web development class helped the team understand the basic of the MVC structure which structure the Dashboard application. A course in system security and cloud computing helped one of the members develop the Admin application using the Agular framework. The User Interface Design course helped the team settle with a minimalistic design that will be easy for the client to change and personalize to their new website in the future. One of the team members experience from their internship work helped in using Microsoft Azure hosting service. The course in databases helped the team be familiar with SQL to be able to query in Microsoft Azure's database. Previous experience also helped the team use GitHub, GitHub Desktop, and git for collaborative development. Waxwings also looked at other resources and tutorials to complete the product.

2.8 Mentor

Waxwings regularly met with their mentor, Raymond McDermott. Ray was very helpful throughout the process and provided a lot of insight in the software development stages and in the team members' future career endeavors. Ray was adamant about documentation and keeping requirements detailed in writing which Waxwings followed. This helped greatly especially during the development stage where the team had to code specific patient data. Ray was also very helpful in helping with the cloud hosting environment. He also provided Waxwings with many resources that helps in growing our development knowledge. Raymond was very encouraging and gave us career tips in where to apply, how to apply for future jobs, what a good development culture looks like, how to give ourselves the best opportunities for success and the possibility of working under contract as an individual.

3. System Requirements

3.1 Overview

The major requirements of the product that were given by the client, created by Waxwings and were clarified and finalized with the client were:

- Dashboard application that displayed patient data into graphs

- The application should take in the excel file and graph that data
- Ability to click on graphs to make them bigger and view their details (Client used elements from Power BI as a visual reference, but specified that it does not have to be exactly like Power BI)
- A drill down option to view a graph's data by a specified year, if that is applicable to that data category

Overall, an admin user will load data into the application, that data will be organized into different graphs and those graphs will be viewed by donor users. A typical situation of a possible donor user would be: the donor will go to MASS Clinic's website, click on the link to go to the Patient Quality Measurements, view a landing page dashboard that has various graphs of patient data, can click on individual graphs to view them individually and possibly have a drill down option for a year. The admin user would just be responsible for loading the patient data file.

3.2 User Stories and Non-Functional Requirements

- Admin user
    1. As an administrator, I want to access a sign in page for the file upload so that only specific people can upload data.
        - Status: COMPLETED in February iteration
        - Details:
            - This requires a user ID and password that are strings
            - Also, if the sign in is incorrect or not authorized, an error message will appear.
    2. As an administrator, I want to be able to upload an excel file that contains all the patient data to be graphed so that it is easier than manually plugging in values.
        - Status: COMPLETE in April iteration
        - Details:
            - An error message will appear if the Excel file that is uploaded is missing specific tabs and formatting. The errors will specify what categories are missing.
    3. As an administrator, I want to know if my file upload was successful so that I know that the data made it to the backend.
        - Status: COMPLETE in April iteration
    4. As an administrator, I want to be able to add another admin account user so that it is not centralized control.
        - Status: COMPLETE in March iteration
        - Details:
            - This will ask for a user ID, first name, last name, email address and password in string types. An error will occur if information is in the wrong format.
- Donor user
    1. As a donor, I want to be able to access the quality dashboard application from a link on the MASS Clinic website.
        - Status: COMPLETE in February iteration
        - Details:

        - MASS Clinic is going through a new website development process, the Waxwings product will be implemented during or after that process is mostly complete.

2. As a donor, I want to see patients served data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
3. As a donor, I want to see patient encountered data and be able to view specific years of data if applicable.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
4. As a donor, I want to see diabetic patients served data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other patients served data
5. As a donor, I want to see hypertensive patients served data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other patients served data
6. As a donor, I want to see diabetic and hypertensive patients served data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other patients served data
7. As a donor, I want to see medical volunteer data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
8. As a donor, I want to see non-medical volunteer data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other volunteer data
9. As a donor, I want to see patient screenings provided data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:

        - Bar graph
        - Combined with other volunteer data

10. As a donor, I want to see patient referrals to MASS Clinic data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other referral MASS Clinic data
11. As a donor, I want to see patient referrals from MASS Clinic data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other referral MASS Clinic data
12. As a donor, I want to see diabetic patients referred to Telehealth data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other referral Telehealth data
13. As a donor, I want to see hypertensive patients referred to Telehealth data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other referral Telehealth data
14. As a donor, I want to see diabetic and hypertensive patients referred to Telehealth data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other referral Telehealth data
15. As a donor, I want to see follow-up encounters with diabetic patient data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other follow up encounters data
16. As a donor, I want to see follow-up encounters with hypertensive patient data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
        - Bar graph
        - Combined with other follow up encounters data
17. As a donor, I want to see follow-up encounters with diabetic and

hypertensive patient data and be able to view specific years of data.

  - Status: COMPLETE in February iteration
  - Details:
    - Bar graph
    - Combined with other follow up encounters data

18. As a donor, I want to see medical insurance applications completed data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
      - Bar graph
19. As a donor, I want to see prescription assistance applications completed data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
      - Bar graph
20. As a donor, I want to see demographics data and be able to view specific years of data.
    - Status: COMPLETE in February iteration
    - Details:
      - Pie graphs
21. As a donor, I want to see patient arrival time data and be able to view specific years of data.
    - Status: COMPLETE in March iteration
    - Details:
      - Pie graph
22. As a donor, I want to see patient accessibility data and be able to view specific years of data.
    - Status: COMPLETE in March iteration
    - Details:
      - Pie graph
23. As a donor, I want to see patient no show data and be able to view specific years of data.
    - Status: COMPLETE in March iteration
    - Details:
      - Bar graph
24. As a donor, I want to see unique patients' data and be able to view specific years of data.
    - Status: COMPLETE in March iteration
    - Details:
      - Bar graph
25. As a donor, I want to see appointment type patient data and be able to view specific years of data.
    - Status: COMPELTE in MARCH iteration
    - Details:
      - Bar graph

26. As a donor, I want to see last year seen patient data that shows how long a patient receives care at MASS Clinic by having its patient enrollment and unique patients served data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Line graph
27. As a donor, I want to see 20% of MASS diabetic patients will be referred to the Telehealth as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other 20% relevant data
28. As a donor, I want to see 20% of MASS hypertensive patients will be referred to the Telehealth as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other 20% relevant data
29. As a donor, I want to see 20% of MASS dually diagnosed diabetic and hypertensive patients will be referred to the Telehealth as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other 20% relevant data
30. As a donor, I want to see 50% of identified MASS clinic diabetic patients will attend follow-up appointments as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other 50% relevant data
31. As a donor, I want to see 50% of identified MASS clinic hypertensive patients will attend follow-up appointments as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other 50% relevant data
32. As a donor, I want to see 50% of identified MASS clinic dually diagnosed diabetic and hypertensive patients will attend follow-up appointments as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration

        - Details:
            - Bar graph
            - Combined with the other 50% relevant data

33. As a donor, I want to see 20% of patients screened through the program who are eligible for health care coverage will complete applications as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
34. As a donor, I want to see 85% of patients found eligible to obtain prescription assistance will complete applications as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
35. As a donor, I want to see 95% of patients who are screened as chronic care patients will be provided with information regarding community resources during the duration of the program data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
36. As a donor, I want to see 25% of MASS diabetic patients referred to Telehealth will complete at least 1 follow-up visit as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other 25% relevant data
37. As a donor, I want to see 25% of hypertensive patients referred to Telehealth will complete at least 1 follow-up visit as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other 25% relevant data
38. As a donor, I want to see 25% of MASS dually diagnosed diabetic and hypertensive patients referred to Telehealth will complete at least 1 follow-up visit as measured by patient records data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other 25% relevant data
39. As a donor, I want to see patients who had at least one HbA1c test within

one year prior to their most recent encounter data and be able to view specific years of data.

- Status: COMPLETE in April iteration
- Details:
  - Bar graph
  - Combined with the other HbA1c relevant data

40. As a donor, I want to see patients who did not have at least one HbA1c test within one year prior to their most recent encounter or whose last HbA1c test was > 9% data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
      - Bar graph
      - Combined with the other HbA1c relevant data
41. As a donor, I want to see patients who had at least one HbA1c test within one year prior to their post recent encounter and whose last HbA1c test was < 7% data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
      - Bar graph
      - Combined with the other HbA1c relevant data
42. As a donor, I want to see patients who had at least one HbA1c test within one year prior to their post recent encounter and whose last HbA1c test was <= 8% data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
      - Bar graph
      - Combined with the other HbA1c relevant data
43. As a donor, I want to see patients who had at least one LDL cholesterol test within one year prior to their most recent encounter data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
      - Bar graph
      - Combined with the other cholesterol relevant data
44. As a donor, I want to see patients who had at least one LDL cholesterol test within one year prior to their most recent encounter and whose last LDL cholesterol was <100 mg/dL data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
      - Bar graph
      - Combined with the other cholesterol relevant data
45. As a donor, I want to see patients whose blood pressure at their last encounter was <140/80 data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:

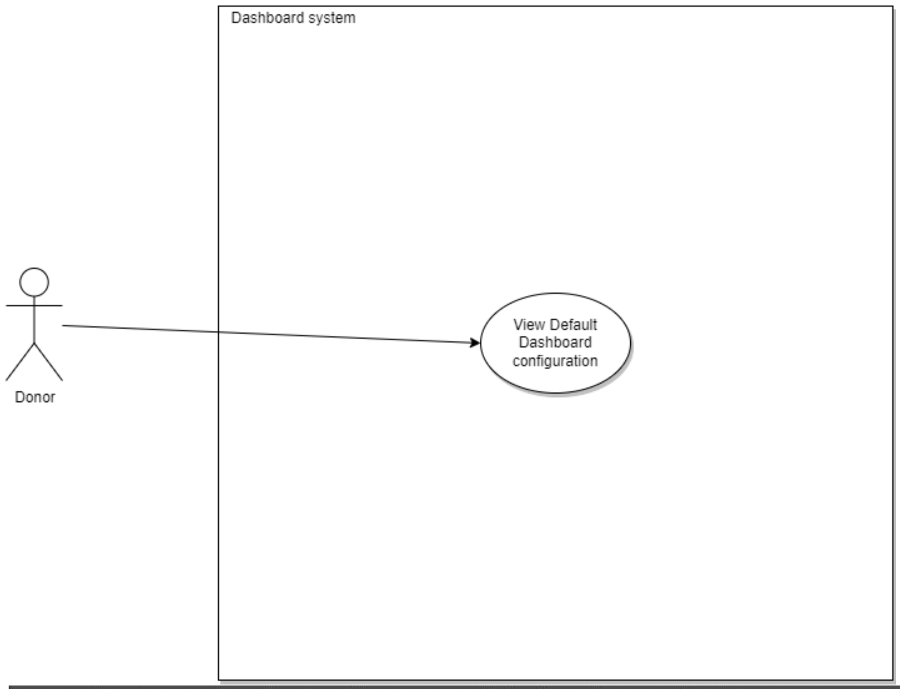
        - Bar graph
        - Combined with the other blood pressure relevant data

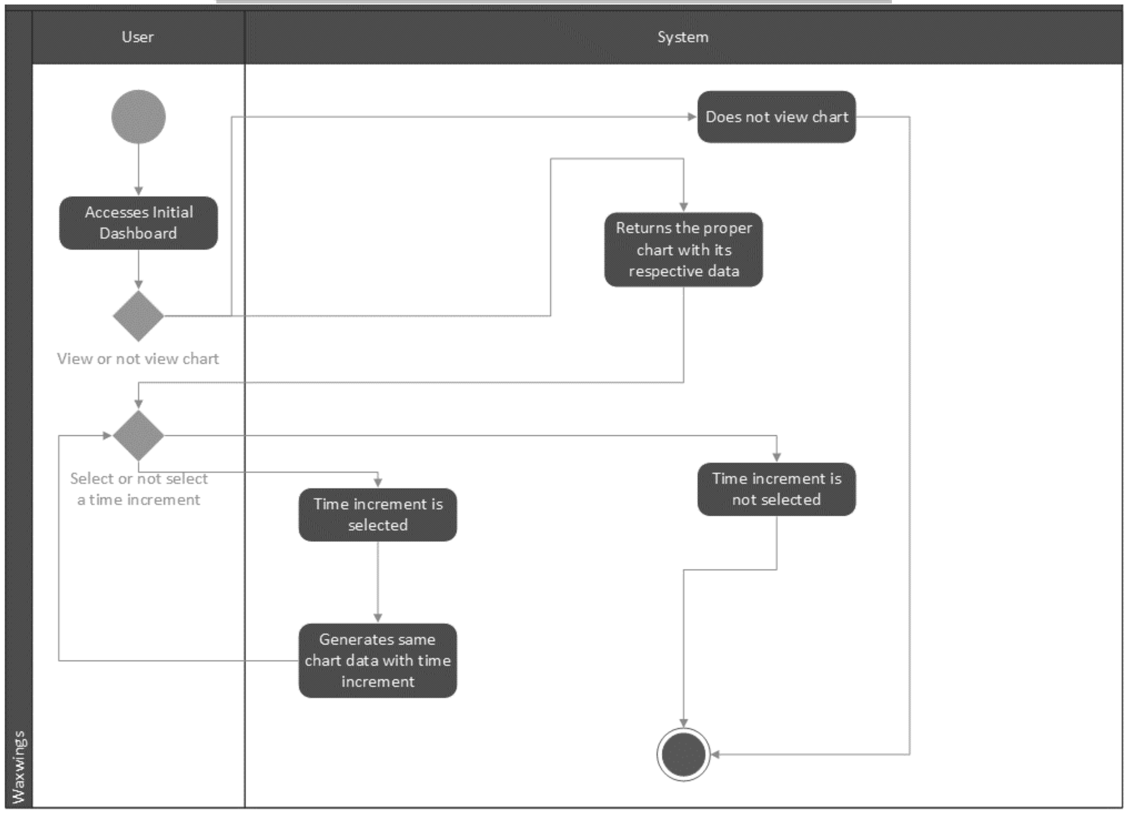
46. As a donor, I want to see patients who had a blood pressure measurement taken at their last encounter data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other blood pressure relevant data
47. As a donor, I want to see patients whose blood pressure at their last encounter was <= 140/90 data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other blood pressure relevant data
48. As a donor, I want to see patients whose blood pressure at their last encounter was > 140/90 data and be able to view specific years of data.
    - Status: COMPLETE in April iteration
    - Details:
        - Bar graph
        - Combined with the other blood pressure relevant data
49. As a donor, I want to be able to navigate to specific graphs data and be able to view specific years of data.
    - Status: COMPLETE in January iteration
    - Details:
        - Dropdown navigation
50. As a donor, I want to be able to navigate to specific groupings of graphs on smaller dashboards data and be able to view specific years of data.
    - Status: COMPLETE in January iteration
    - Details:
        - Navigation bar

3.3 Functional Requirements

3.3.1 Default Configuration of User Dashboard

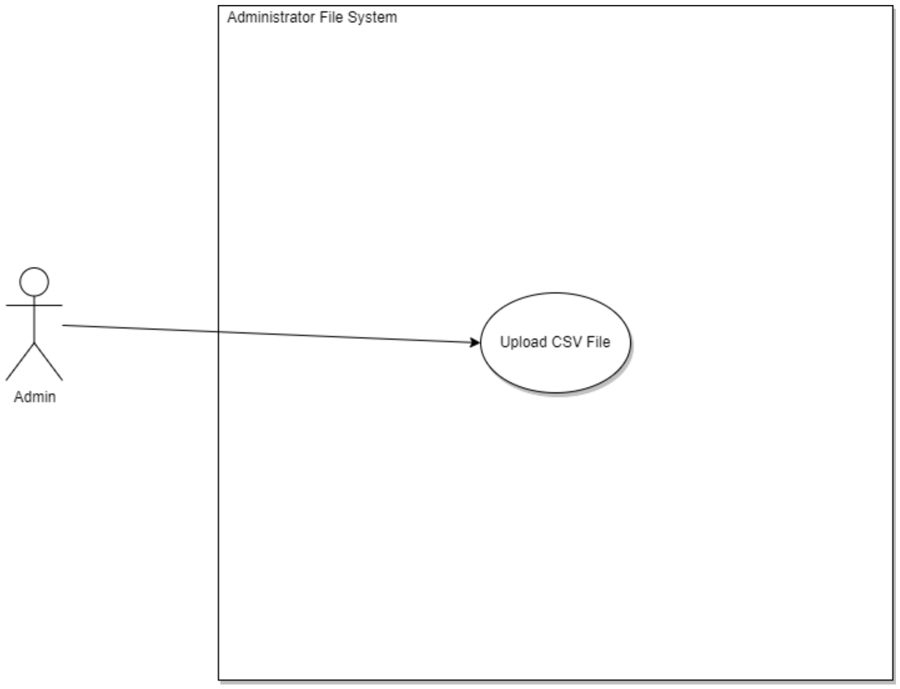
The donor user goes to the MASS clinic website. The donor will be able to access this dashboard from the MASS website with a link on a tab. This use case should allow the donor to see the landing page of the dashboard with the primary default configuration of the graphs. The landing page renders all the graphed data from the JSON objects it receives from the database.

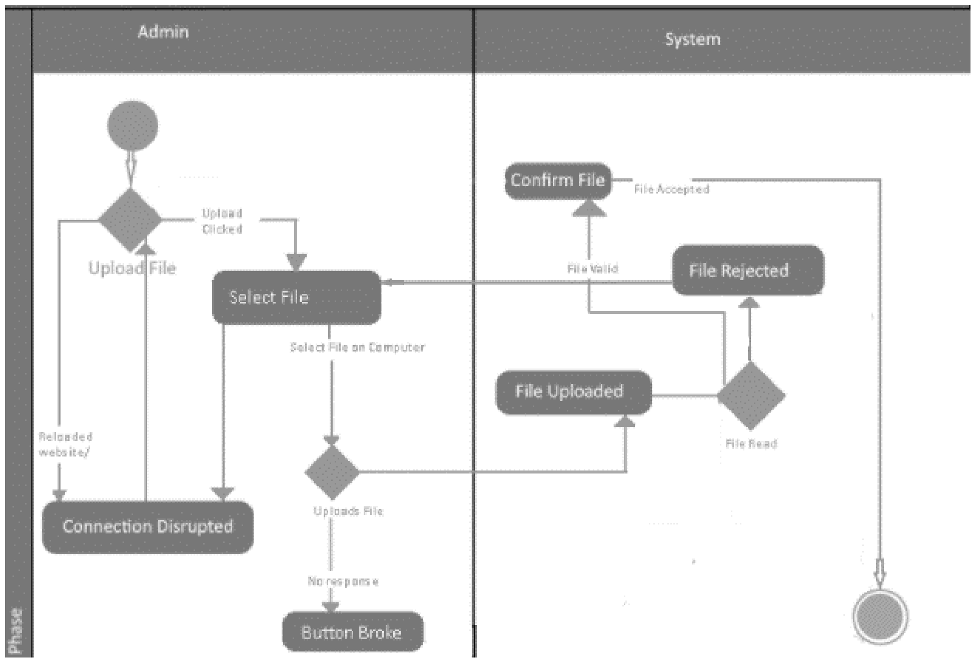



User
System
Waxwings
Accesses Initial Dashboard
View or not view chart
Does not view chart
Returns the proper chart with its respective data
Select or not select a time increment
Time increment is selected
Time increment is not selected
Generates same chart data with time increment

3.3.2 *Management of CSV Files*

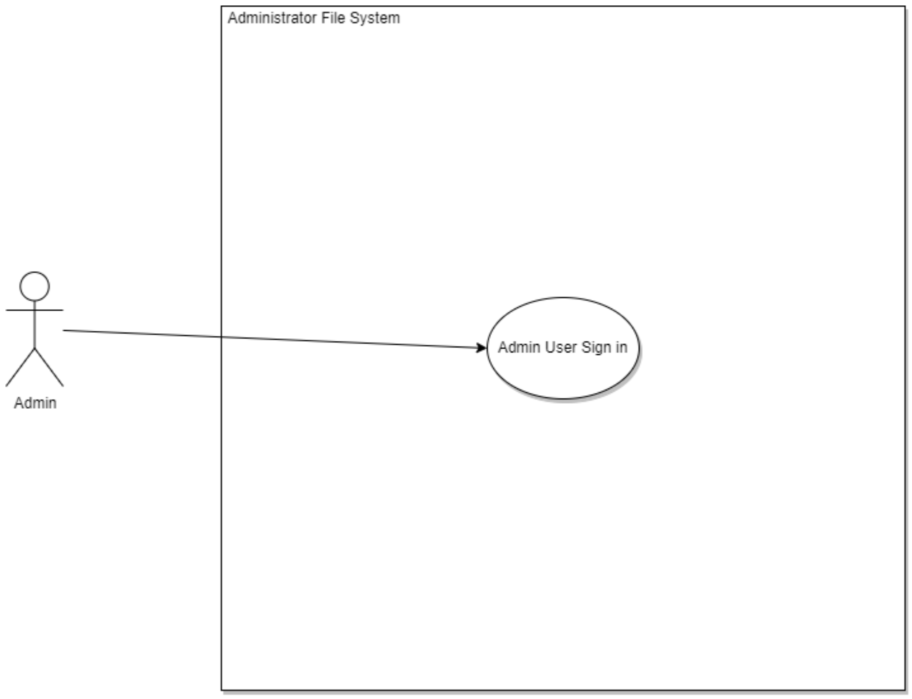
Authorized administration will be able to upload an Excel file into the system which then the dashboard will re-display the newly uploaded content. Administration will check, filter, and clean csv files/data beforehand and then upload the Excel file into the application. The application will be able to check if the Excel file is in the proper formatting. The database will receive the data and send it to the dashboard graphs.

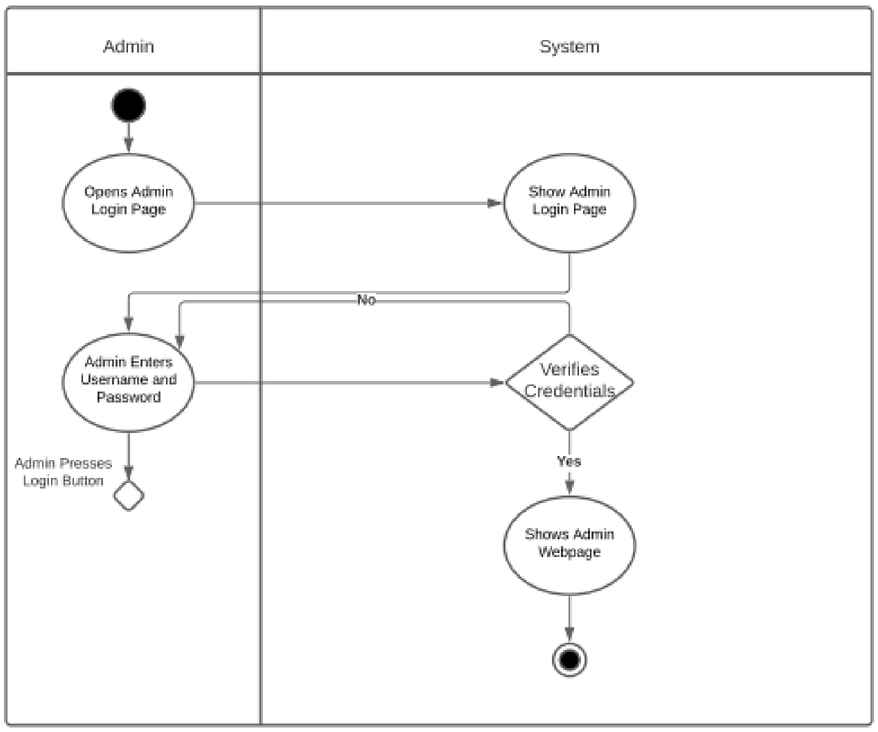




3.3.3 *Administration User Sign-in*

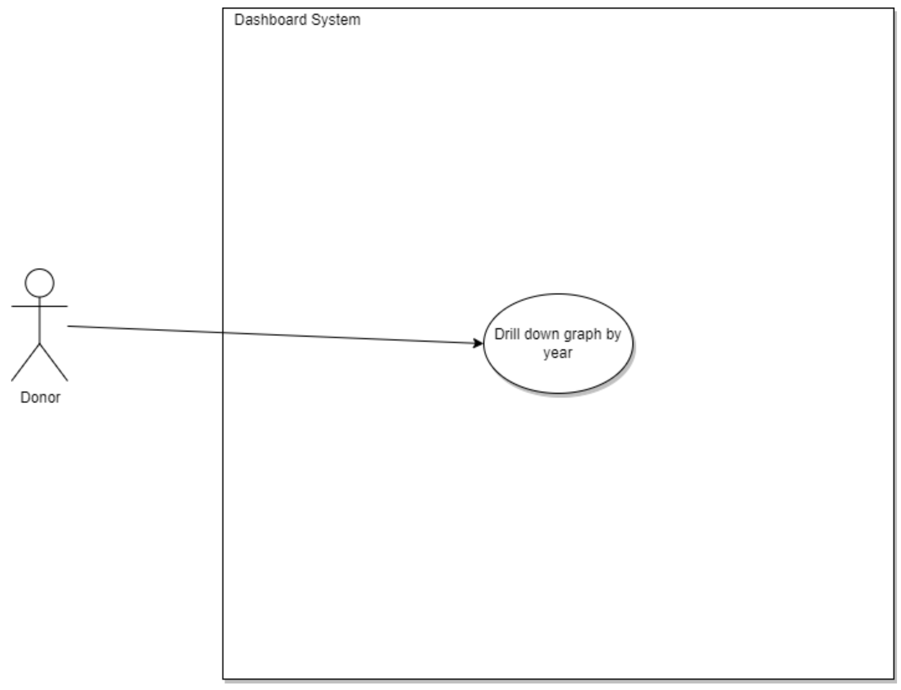
The point of the admin login use case is to make sure only authenticated users who are authorized can edit or view the patient quality information. The users will enter a username and password, which if are the correct pair will grant them access for a file upload. The system will be able to authorize the correct sign in information.

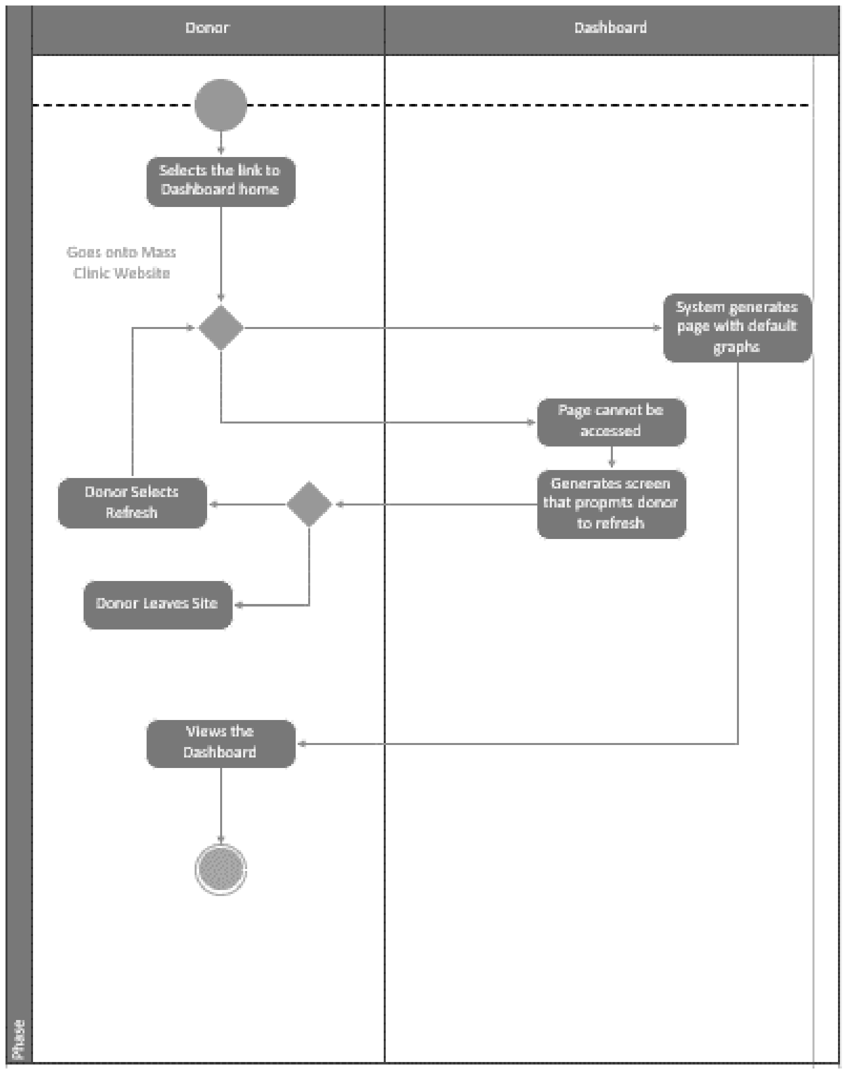




3.3.4 *Editing Individual Graphs by Year*

The donor will have the ability to choose specific graphs to view in a larger more detailed manner. There will also be a drill down option for specific graphs to display data by a certain year. This year data will already be part of the JSON object received from the database to render the graph initially.


Dashboard System
Drill down graph by year
Donor


Donor
Dashboard
Selects the link to Dashboard home
Goes onto Mass Clinic Website
System generates page with default graphs
Page cannot be accessed
Generates screen that propmts donor to refresh
Donor Selects Refresh
Donor Leaves Site
Views the Dashboard
Phase

3.4 Mandatory and Special Requirements

There was no other special requirement that were expressed by MASS Clinic that was relevant to the successful functioning of the software system. All requirements have been listed previously in prior sections.

4. Detailed Design

4.1 Architectural Design

4.1.1 Architecture Models Used

4.1.1.1 MVC

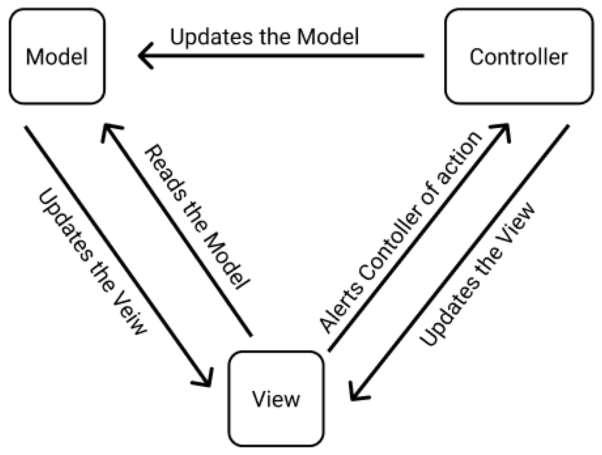
This architecture breaks the web application into 3 pieces: the model, the view, and the controller. Using these three parts the developers can follow a pattern, allowing them to build the application iteratively.

Models: The models are files that connect to the database and hold the shape of different pieces of data. Models can pull down arrays of data from the database and pass them through the controllers into the views.

Controllers: The controllers are files that control the communication between the views and the models. A controller will allow a view to using just one or many models when the user needs it. The controller also calls the different views when the user needs them. In the code and only specifically for this product, controllers and routes are interchangeable due to how they are accomplishing the same task.

Views: The views are the part of the application that the users interact with. The views either just display themselves or the data that was given to them from the models. The views are flexible and allow different data to be displayed based on the data given to them from the model.

Waxwings have chosen to use the MVC architecture for our project because of the ability to design in iteration. Without knowing at the start of the project all that will be needed using the MVC pattern allows the developers to have a way to continue to build and change and not worry about adding code in a way that will disrupt other code already written. By creating a three-part system all of the developers will know where the data is coming from and can easily find where it is being used.



4.1.1.2 3-Tier Client Server

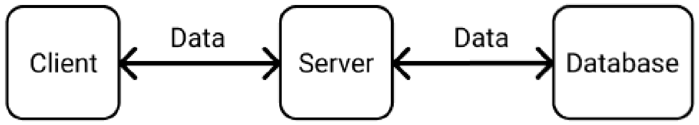
This approach separates the application into three distinct parts and controls how the user can access data. In this architecture, there are three parts: the client, the server, and the database. The client is the interface with the user who sends messages/data to the server and then the server sends the data to the database.

Client: The client is the part of the application that the user interacts with. It displays data, pages, and takes input from the user. Once the user has given the client input or made a request the client begins to communicate with the server. It will either give the server information/data or will make a request to the server for the pages the user is looking for.

Server: The server takes requests from the client and uses those requests to call data from the database. It then takes the data it is looking for from the database and gives it back to the client. The server also can transform the data before it gives it to the client or the database.

Database: The database controls all of the data. It only talks to the server and has no direct communication with the client. With the correct authorization, the database will allow the server to get any data it wants and then sent it to the client.

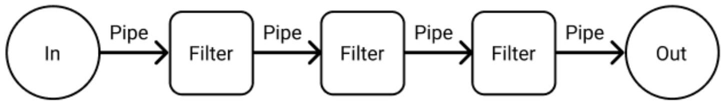
Waxwings have chosen this method because a large portion of our project is allowing one person to upload data and allowing many people to access it. By creating separation between the client, server, and database we allow the server to process any data going into or out of the database and make sure the right people are taking those actions. Waxwings can make sure that the right form of data is going in through validation on the server and we can make sure the users who are only allowed to read the data are not able to directly add new data to the database. It also allows the developers to test ahead of time in Azure rather than just running the code all in one place.



4.1.1.3 Pipe and Filter

This architecture uses a pipeline that takes in data, often in a raw or unprocessed form, and then runs it through a series of filters that change the data. The filters can be stacked up and used in sequence on the data so that the data comes out the other end of the process in a usable way.

In the application, Waxwings are creating a dashboard that takes data from different tables in an azure database and displays that data into different graphs for the end-users. To have this data become displayable we have to run it through several filters. Waxwings need to filter most of our data by year, some data by patient type, and some data by a specific place. To avoid writing the same code repeatedly we can make filters that can be used in multiple places.

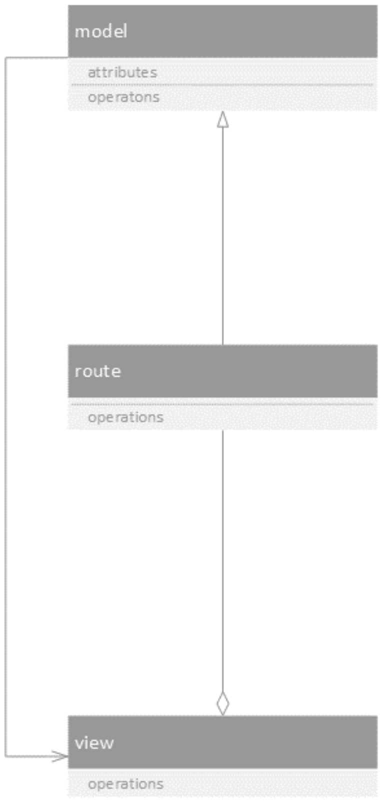


4.2 Design Specifications

To accomplish what the development team wanted, Waxwings are using a Node.js application with Express for the dashboard application. Express allows the team to use the MVC format using models, views, and routes. It also allows the views to be generated on the client-side without allowing access to the server-side of the application. On the side of the dashboard application, Waxwings also use Sequelize in the models that allow the models to access the database.

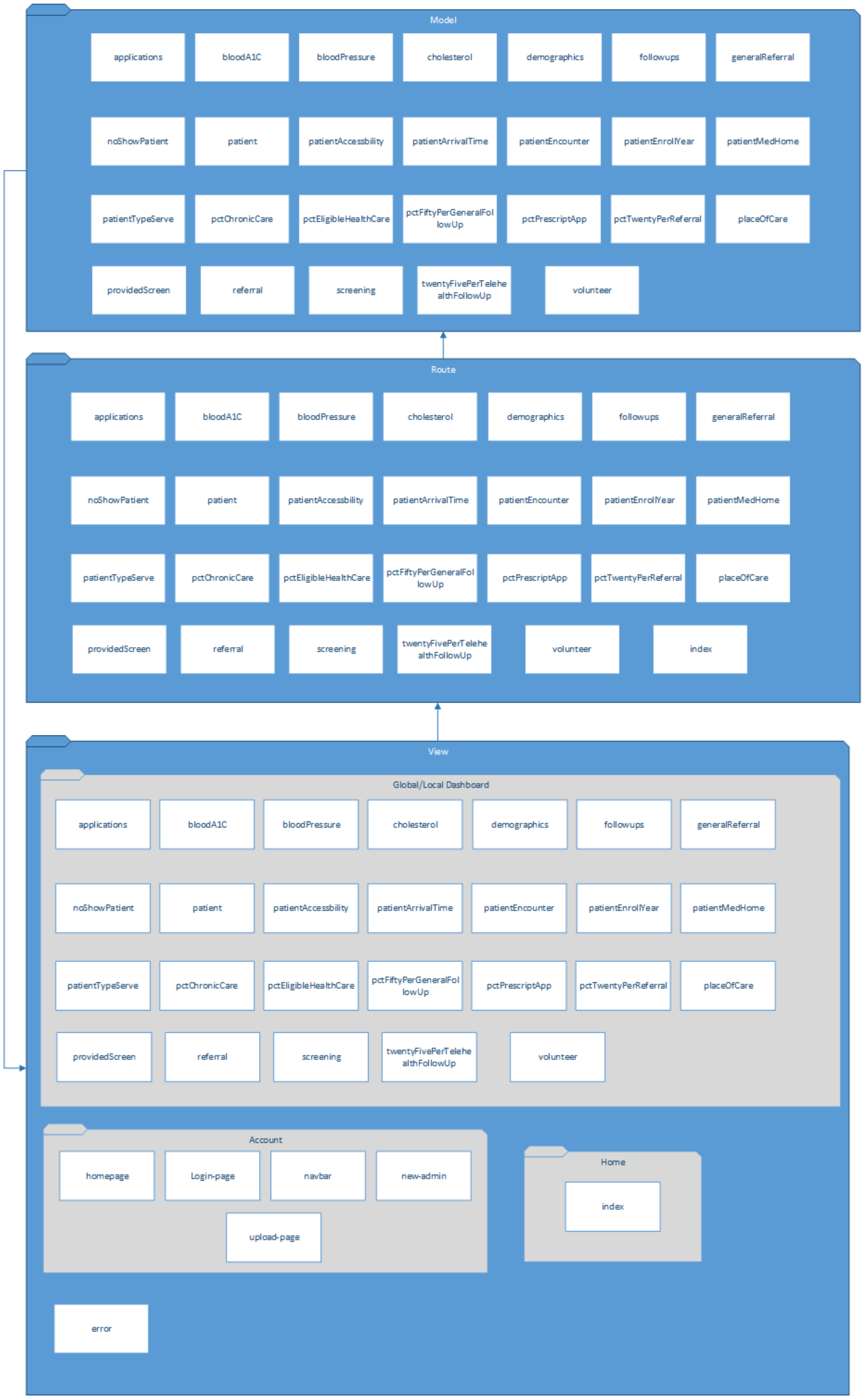
For the Data Admin application, Waxwings are using Angular with JavaScript and Typescript for our frontend and using ASP .NET for our backend. HTML designs are created from Bootstrap templates redesigned for our project purposes. Security will include JWT Tokens for Angular, and the security for the backend is tbd. Angular utilizes components and services which deviate from the typical MVC architecture; however much of the organization and hierarchy of the files will be similar. For the security portion of the application, JWT tokens, Bcrypt, Bearer Header Tokens and tbd security technologies will be permeated throughout the project.

To host both of the applications, Waxwings are using Azure app services and to host the database we are using azure SQL server. This allows us easy access to both applications and databases and will allow easy transfer when we move the code from our azure accounts onto the clients.



For an overall picture, the image above summarizes how the classes are related in respect to the package they belong to as labeled model, route and view. Their operations and attributes are listed in the detailed files as seen in the full diagram below (note: to view better, expand picture but for the sake of fitting into this document it was restricted to the seen size).

For the model operations in the code, the respective Azure database is called, a constant is made given the specific table it is pulling from and those attributes. Finally, an export of the module with all the data and attributes is sent to the respective route. The package diagram is seen below along with its details.


Model
applications
bloodA1C
bloodPressure
cholesterol
demographics
followups
generalReferral
noShowPatient
patient
patientAccessbility
patientArrivalTime
patientEncounter
patientEnrollYear
patientMedHome
patientTypeServe
pctChronicCare
pctEligibleHealthCare
pctFiftyPerGeneralFollowUp
pctPrescriptApp
pctTwentyPerReferral
placeOfCare
providedScreen
referral
screening
twentyFivePerTelehealthFollowUp
volunteer
Route
applications
bloodA1C
bloodPressure
cholesterol
demographics
followups
generalReferral
noShowPatient
patient
patientAccessbility
patientArrivalTime
patientEncounter
patientEnrollYear
patientMedHome
patientTypeServe
pctChronicCare
pctEligibleHealthCare
pctFiftyPerGeneralFollowUp
pctPrescriptApp
pctTwentyPerReferral
placeOfCare
providedScreen
referral
screening
twentyFivePerTelehealthFollowUp
volunteer
index
View
Global/Local Dashboard
applications
bloodA1C
bloodPressure
cholesterol
demographics
followups
generalReferral
noShowPatient
patient
patientAccessbility
patientArrivalTime
patientEncounter
patientEnrollYear
patientMedHome
patientTypeServe
pctChronicCare
pctEligibleHealthCare
pctFiftyPerGeneralFollowUp
pctPrescriptApp
pctTwentyPerReferral
placeOfCare
providedScreen
referral
screening
twentyFivePerTelehealthFollowUp
volunteer
Account
homepage
Login-page
navbar
new-admin
upload-page
Home
index
error

This package diagram displays the MVC framework of the Quality Measure Dashboard System by its models, views, routes and their classes. Due to using JavaScript and Node, Express routes are implemented. These routes act as the Controller in the typical MVC framework, but in this project is able to achieve the same result.

Model: The models are files that connect to the database and hold the shape of different pieces of data. Models can pull down arrays of data from the database and pass them through the controllers into the views.

1. applications: This model defines all attributes of applications for both medical insurance applications and prescription assistance applications that have been completed. Attributes defined here are RowID, ApplicationType, PatientAmount, and RecordedYear.
2. bloodA1C: This model defines all attributes of measured A1C levels in the blood. Attributes defined here are RowID, TimePeriod, NumDom, TotalPercent, PatientAmountSpecific and PatientAmountTotal.
3. bloodPressure: This model defines all attributes of measured blood pressure for diabetic patients and for hypertensive patients. Attributes defined here are RowID, TimePeriod, NumDom, Total Percent, PatientAmountSpecific, and PatientAmountTotal.
4. cholesterol: This model defines all attributes of measured cholesterol. Attributes defined here are RowID, TimePeriod, NumDom, Total Percent, PatientAmountSpecific, and PatientAmountTotal.
5. demographics: This model defines all attributes of demographics which include gender, age and race. Attributes defined here are RowID, DemographicType, PatientAmount, and RecordedYear.
6. followups: This model defines all attributes of patient follow ups. Attributes defined here are RowID, PatientType, PatientAmount, and RecordedYear.
7. generalReferral: This model defines all attributes of referrals to MASS clinic that are **not** towards Telehealth. Attributes defined here are RowID, ToFrom, PatientAmount, and RecordedYear.
8. noshowPatient: This model defines all attributes of how many patients did not show up for their scheduled appointment. Attributes defined here are RowID, RecordedYear, and PercentAmount.
9. patient: This model defines all attributes of patients served who have only diabetes, only hypertension and patients who have both diabetes and hypertension. Attributes defined here are RowID, PatientType, PatientAmount, and RecordedYear.
10. patientAccessbility: This model defines all attributes of how accessible the patient was to an appointment availability in the clinic. Attributes defined here are RowID, AmountOfTime, PercentAmount, TotalAmount and SpecificAmount.
11. patientArrivalTime: This model defines all attributes of when patients arrived at the clinic. Attributes defined here are RowID, TimeArrived, PercentAmount, and PatientAmount.
12. patientEncounter: This model defines all attributes of how many patients the clinic has encountered in a specified amount of time. Attributes defined here are RowID, PatientAmount, RecordedYear, and TargetNumber.
13. patientEnrollYear: This model defines all attributes of how many patients enrolled in the clinic per year. Attributes defined here are RowID, PatientAmount, RecordedYear,

and TargetNumber.

14. patientMedHome: This model defines all attributes of measured patients from a medical home. Attributes defined here are RowID, AmountOfYears and PatientAmount.
15. patientTypeServe: This model defines all attributes of the type of patient that is served. Attributes defined here are RowID, PatientAmount, RecordedYear, and PatientType.
16. pctChronicCare: This model defines all attributes of the outcome percentage of patients that need chronic care. Attributes defined here are RowID, TargetPercent, PercentAchieved, RecordedYear, Numerator and Denominator.
17. pctEligibleHealthCare: This model defines all attributes of the outcome percentage of those eligible for health care. Attributes defined here are RowID, TargetPercent, PercentAchieved, RecordedYear, Numerator and Denominator.
18. pctFiftyPerGeneralFollowUp: This model defines all attributes of the outcome percentage of follow ups for patients that are only diabetic, only hypertensive and those who are both. Attributes defined here are RowID, TargetPercent, PercentAchieved, RecordedYear, Numerator and Denominator.
19. pctPrescriptApp: This model defines all attributes of the outcome percentage of patients who have prescription applications. Attributes defined here are RowID, TargetPercent, PercentAchieved, RecordedYear, Numerator and Denominator.
20. pctTwentyPerReferral: This model defines all attributes of outcome percentage of referred patients that are only diabetic, only hypertensive and those who are both. Attributes defined here are RowID, TargetPercent, PercentAchieved, RecordedYear, Numerator and Denominator.
21. placeOfCare: This model defines all attributes of patients in various places of care. Attributes defined here are RowID, ApptypeGroup, ApptypeSpecific, PatientsEncounters, and UniquePatients.
22. providedScreen: This model defines all attributes of measured provided screenings for patients. Attributes defined here are RowID, PatientAmount, and RecordedYear.
23. referral: This model defines all attributes of referrals from MASS clinic that **are** towards Telehealth. Attributes defined here are RowID, PatientType, PatientAmount, and RecordedYear.
24. screening: This model defines all attributes of the total number of screenings done by the clinic. Attributes defined here are RowID, PatientAmount, and RecordedYear.
25. twentyFivePerTelehealthFollowUp: This model defines all attributes of the outcome percentage of Telehealth referred patients who had a follow up that are only diabetic, only hypertensive and those who are both. Attributes defined here are RowID, TargetPercent, PercentAchieved, RecordedYear, Numerator and Denominator.
26. volunteer: This model defines all attributes of the number of medical and non-medical volunteers. Attributes defined here are RowID, VolunteerType, VolunteerAmount, and RecordedYear.

<u>View:</u> The views are the part of the application that the users interact with. The views either just display themselves or the data that was given to them from the models. The views are flexible and allow different data to be displayed based on the data given to them from the model.

1. Global/Local Dashboard: All these view pages are related to the dashboard and either

contain tables, graphs and/or charts of data. These pages are the end view for the donor as it visually summarizes the data they want to see by patient condition.

a. applications: This view displays all data of applications for both medical insurance applications and prescription assistance applications that have been completed in a graph(s).
b. bloodA1C: This view displays all data of measured A1C levels in the blood in a graph(s).
c. bloodPressure: This view displays all data of measured blood pressure for diabetic patients and for hypertensive patients in a graph(s).
d. cholesterol: This view displays all data of measured cholesterol in a graph(s).
e. demographics: This view displays all data of demographics which include gender, age and race in a graph(s).
f. followups: This view displays all data of patient follow ups in a graph(s).
g. generalReferral: This view displays all data of referrals to MASS clinic that are not towards Telehealth in a graph(s).
h. noshowPatient: This view displays all data of how many patients did not show up for their scheduled appointment in a graph(s).
i. patient: This view displays all data of patients served who have only diabetes, only hypertension and patients who have both diabetes and hypertension in a graph(s).
j. patientAccessbility: This view displays all data of how accessible the patient was to an appointment availability in the clinic in a graph(s).
k. patientArrivalTime: This view displays all data of when patients arrived at the clinic in a graph(s).
l. patientEncounter: This view displays all data of how many patients the clinic has encountered in a specified amount of time in a graph(s).
m. patientEnrollYear: This view displays all data of how many patients enrolled in the clinic per year in a graph(s).
n. patientMedHome: This view displays all data of measured patients from a medical home in a graph(s).
o. patientTypeServe: This view displays all data of the type of patient that is served in a graph(s).
p. pctChronicCare: This view displays all data of the outcome percentage of patients that need chronic care in a graph(s).
q. pctEligibleHealthCare: This view displays all data of the outcome percentage of those eligible for health care in a graph(s).
r. pctFiftyPerGeneralFollowUp: This view displays all data of the outcome percentage of follow ups for patients that are only diabetic, only hypertensive and those who are both in a graph(s).
s. pctPrescriptApp: This view displays all data of the outcome percentage of patients who have prescription applications in a graph(s).
t. pctTwentyPerReferral: This view displays all data of outcome percentage of referred patients that are only diabetic, only hypertensive and those who are both in a graph(s).
u. placeOfCare: This view displays all data of patients in various places of care in a graph(s).

v. providedScreen: This view displays all data of measured provided screenings for patients in a graph(s).
w. referral: This view displays all data of referrals from MASS clinic that are towards Telehealth in a graph(s).
x. screening: This view displays all data of the total number of screenings done by the clinic in a graph(s).
y. twentyFivePerTelehealthFollowUp: This view displays all data of the outcome percentage of Telehealth referred patients who had a follow up that are only diabetic, only hypertensive and those who are both in a graph(s).
z. volunteer: This view displays all data of the number of medical and non-medical volunteers in a graph(s).

2. Account: All view pages here are related to account management and file upload. This essentially will act as a "separate" application as the administrator will be the only one to know about this side of the application (i.e., a donor would not have access or knowledge about an account).
    a. homepage: This is the first or landing page that the donor is routed to when they access the admin user application.
    b. login-page: This page will prompt the admin user to login in with a ID and password.
    c. navbar: This bar is the navigation between the login/logout steps.
    d. new-admin: This page will allow for the creation or an update of the admin user.
    e. upload-page: This page will have the options to upload a file that has the data on it so that it can be processed into the database and essentially end up on the front end views for the donors.
3. Home: This is the home page of the application
    a. Index: This is the first or landing page that the donor is routed to when they access the dashboard application/website. This will also contain the initial dashboard that will generate the initial default charts or graphs.
4. Error: This page is an error page that the donor will see if the intended view page was not able to be reached due to data generation errors.

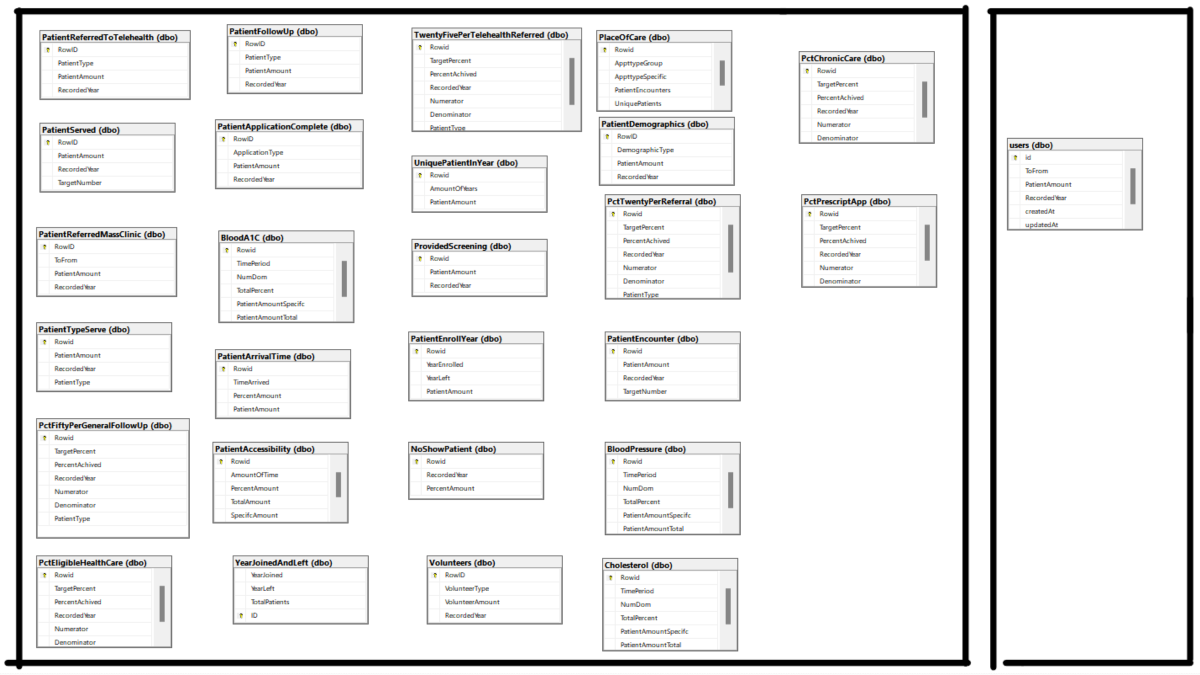
Route: The routes are files that control the communication between the views and the models. A route will allow a view to use just one or many models when the user needs it. The route also calls the different views when the user needs them. In the product, routes and controllers are interchangeable, but this only applies to this product.

1. applications: This route passes information of applications for both medical insurance applications and prescription assistance applications that have been completed to the respective view page.
2. bloodA1C: This route passes information of measured A1C levels in the blood to the respective view page.
3. bloodPressure: This route passes information of measured blood pressure for diabetic patients and for hypertensive patients to the respective view page.
4. cholesterol: This route passes information of measured cholesterol to the respective view page.
5. demographics: This route passes information of demographics which include gender, age and race to the respective view page.

6. followups: This route passes information of patient follow ups to the respective view page.
7. generalReferral: This route passes information of referrals to MASS clinic that are not towards Telehealth to the respective view page.
8. noshowPatient: This route passes information of how many patients did not show up for their scheduled appointment to the respective view page.
9. patient: This route passes information of patients served who have only diabetes, only hypertension and patients who have both diabetes and hypertension to the respective view page.
10. patientAccessbility: This route passes information of how accessible the patient was to an appointment availability in the clinic to the respective view page.
11. patientArrivalTime: This route passes information of when patients arrived at the clinic to the respective view page.
12. patientEncounter: This route passes information of how many patients the clinic has encountered in a specified amount of time to the respective view page.
13. patientEnrollYear: This route passes information of how many patients enrolled in the clinic per year to the respective view page.
14. patientMedHome: This route passes information of measured patients from a medical home to the respective view page.
15. patientTypeServe: This route passes information of the type of patient that is served to the respective view page.
16. pctChronicCare: This route passes information of the outcome percentage of patients that need chronic care to the respective view page.
17. pctEligibleHealthCare: This route passes information of the outcome percentage of those eligible for health care to the respective view page.
18. pctFiftyPerGeneralFollowUp: This route passes information of the outcome percentage of follow ups for patients that are only diabetic, only hypertensive and those who are both to the respective view page.
19. pctPrescriptApp: This route passes information of the outcome percentage of patients who have prescription applications to the respective view page.
20. pctTwentyPerReferral: This route passes information of outcome percentage of referred patients that are only diabetic, only hypertensive and those who are both to the respective view page.
21. placeOfCare: This route passes information of patients in various places of care to the respective view page.
22. providedScreen: This route passes information of measured provided screenings for patients to the respective view page.
23. referral: This route passes information of referrals from MASS clinic that are towards Telehealth to the respective view page.
24. screening: This route passes information of the total number of screenings done by the clinic to the respective view page.
25. twentyFivePerTelehealthFollowUp: This route passes information of the outcome percentage of Telehealth referred patients who had a follow up that are only diabetic, only hypertensive and those who are both to the respective view page.
26. volunteer: This route passes information of the number of medical and non-medical volunteers to the respective view page.

index: This route passes information of the landing page of the initial dashboard information to its respective view page.

4.3 Database Design



As seen in the diagram above, each table in the database does not have relationships with each other and the donor user would not have interaction with the data other than viewing it on the page of the dashboard application. As for the admin user, we simplified the application so that it would not relate with patient data. Nevertheless, the admin user will have a stored user id, password, year and updated time data. The patient data that is split into 26 tables will have their respective columns of data.

Waxwings has justified the use of the database in this way due to the 3-tier client-server architecture of the application. This approach separates the application into three distinct parts and controls how the user can access data. In this way, there are three parts: the client, the server, and the database. The client is the interface with the user who sends messages/data to the server and then the server sends the data to the database. The database then controls all of the data. It only talks to the server and has no direct communication with the client. With the correct authorization, the database will allow the server to get any data it wants and then send it to the client.

Below is the provided scheme from the Azure database when “SELECT * FROM sys.schemas” is queried.

```
name                          schema_id   principal_id

dbo                           1           1
```

guest	2	2
INFORMATION_SCHEMA	3	3
sys	4	4
db_owner	16384	16384
db_accessadmin	16385	16385
db_securityadmin	16386	16386
db_ddladmin	16387	16387
db_backupoperator	16389	16389
db_datareader	16390	16390
db_datawriter	16391	16391
db_denydatareader	16392	16392
db_denydatawriter	16393	16393

4.4 Component Diagram

Within the architecture of the MVC model, the details of how these components interact with each other has been detailed in the previous section. Overall, it is important to note that the database is what is connecting the two different applications making one product, but Dashboard application and Admin application and their functionalities are separate and unique to itself. The components of the applications are:

1. Waxwings' Administrator File Application - Angular (HTML, CSS, TypeScript, Bootstrap), Node.JS (Express), JWT tokens, Bcrypt, Bearer Header Tokens
2. Waxwings' Quality Measure Dashboard Application - Node.JS (Express, HTML, CSS, JavaScript, Chart.JS)
3. Microsoft Azure Services - Application being hosted on cloud service
4. Microsoft SQL Server - Database holding all the data from file and sign-in for both applications

4.5 Deployment Diagram

These are the specifications for the product:

4.5.1 Hardware

- Processor
  - Minimum: 1.9 gigahertz (GHz) x86- or x64-bit dual core processor with SSE2 instruction set
  - Recommended: 3.3 gigahertz (GHz) or faster 64-bit dual core processor with SSE2 instruction set
- Memory
  - Minimum: 2-GB RAM
  - Recommended: 4-GB RAM or more
- Display
  - Minimum: Super VGA with a resolution of 1024 x 768
  - Recommended: HighSpeed HDMI with at least 1080p
- Windows Azure
  - [Recommended] If Express deployment: at least…
    - 1 physical/virtual machine
    - 1 CPUs
    - 8 GB RAM [note: do not use dynamic memory]
    - 40 GB Disk space
  - Or if Distributed deployment: at least…
    - 8 physical/virtual machine
    - 2 CPUs
    - 8 GB RAM [note: do not use dynamic memory]
    - 40 GB Disk space

4.5.2 Software

- Network:
  - Bandwidth greater than 50 KBps (400 kbps)
  - Latency under 150 ms
- Web browsers
  - [Recommended] Google Chrome (latest publicly released version) running on Windows 11, 10 or 8.1
    - Google Chrome (latest publicly released version) running on at least the last two publicly released Mac OS versions
  - Microsoft Edge (latest publicly released version) running on Windows 11, 10 or 8.1
  - Mozilla Firefox (latest publicly released version) running on Windows 11, 10 or 8.1
  - Apple Safari (latest publicly released version) running on at least the last two publicly released Mac OS versions
- Microsoft Office [note: this takes into account the use of Excel]
  - [Recommended] Microsoft 365 or Office 2016
  - Office 2013
  - Office 2010
- Windows Azure [note: will need a subscription to Microsoft Azure Services due to use

of Microsoft Azure Database services and cloud network]
    - Windows Server 2012 R2
        - Microsoft .NET Framework 3.5 Service Pack (SP) 1
        - IIS 8.5 (built in component of Windows Server 2012 R2
        - .NET Framework 4.5 Extended, with ASP.NET
        - Optionally, Microsoft Web Platform Installer 5.2
    - or [Recommended] Windows Server 2016
        - Microsoft .NET Framework 3.5 Service Pack (SP) 1
        - Internet Information Services (IIS) 10 (built in component of Windows Server 2016)
        - .NET Framework 4.5 Extended, with ASP.NET
        - Optionally, Microsoft Web Platform Installer 5.2
- NodeJS
    - Recommended: v16.13.1
- ReactJS
    - Recommended: 18.1.0
- AngularJS
    - Recommended: 13.0.3
- MongoDB
    - Recommended: 4.4
- JavaScript
    - Recommended: ES5
- HTML
    - Recommended: HTML5
- CSS
    - CSS2.1 or newer

4.5.3 *Cloud Deployment*

Waxwings has been using Visual Studio code as our integrated development environment for both the Admin Application and Dashboard Application. The code is pushed to a shared GitHub repository where the main branch is continually merged with different developers' code and is updated daily for both applications.

For the Dashboard application, the GitHub repository is connected to Microsoft's Azure which is where the Database for the application is also hosted. The updated Dashboard application code is pushed to Azure and the web application is live through a link provided. The client has expressed that MASS Clinic uses Azure currently so the integration of this part of this application and database will not be a major problem. The Admin application is still in the process of being hosted into Azure due to it being written in Angular and it should be hosted by the next deliverable due date.

5. Implementation

5.1 Front End

Front-end stack for the Administrator File Application is HTML, CSS, JavaScript, and Angular. The front-end stack for the Patient Quality Measurement Dashboard Application is HTML, CSS, JavaScript, ReactJS, and ChartJS.

5.2 Back end

The backend for the Dashboard Application was NodeJS, and JavaScript connecting to the Azure SQL database while the backend for the Admin Application was compromised of NodeJS, ExpressJS, JavaScript and TypeScript that links to a Mongo database.

5.3 Work Completed

All requirements listed in the System Requirements Sections were completed. This list was also written as “COMPLETE” in those sections.

6. Test Case Results

6.1 Test Cases

6.1.1 Introduction of Use Cases

The purpose of this Test Cases report section is to define the test cases attributed to the use cases that were determined in the early stages of the Quality Measure Dashboard and Admin File Application for MASS Clinic. Both applications have been created by Waxwings. There are four use cases that have been identified by the Waxwings team: two that are applicable to the Administrator File Application and two to the Quality Measure Dashboard Application.

For the Administrator File Application, the first use case is the Admin Sign-in Capability which will be tested by Wilson Jativa. This use case should render the landing page of the admin sign-in which provides fields to enter a user id and password and a button to login. This case should deny anyone who is not an approved administrator account. A successful sign-in will lead to a file upload which is connected to the next use case. This particular use case also allows the admin to add another user account that has an administrator. The prompt will have fields that prompt for a user id, a password, email and a button. The client has clarified that any file uploaded into the application will not have identifiable information for patients so security in this use case was kept to a minimum while the focus was on authorization and authentication.

Another use case for the Administrator File Application is the Data File Upload and Database Process which will be tested by Trey Cox. This use case will bring the admin user to a file upload page where the admin user can click a button to upload a file and a button for upload. This case should deny a file upload that does not meet the field requirements for the specific tables of the database or data that is irrelevant to the Dashboard application. A successful file upload will lead to a page that prompts that the upload was successful and a button to upload a new file. This upload will lead to the data from the file to be inserted into the database; if there was data already in the database, it should be updated. The database then creates a JSON file with objects of the data that needs to be generated on the Dashboard application.

For the Quality Measure Dashboard, the primary use case is Dashboard Rendering & Navigation which will be tested by JohnArmon Antolin. This application is made of four dashboards that organize all the graphs into their respective categories. This test case makes sure that all mini-graphs are rendering correctly on their respective dashboards, each mini-graph properly leads to the view that shows the fullness of the graph, all full graphs are rendering on all their respective view pages, the navigation tabs on all application views are working and that the dropdown navigation options on all application views are working.

Another use for the Quality Measure Dashboard is the Dashboard Drilldown Generation of Years which will be tested by Hunter Blake. This use case checks all the views that contain full graphs so the drill down option for sorting the respective data into different years is rendering correctly.

6.1.2 Test Case for Admin Sign-In Capability

UC Scenario	**Step Description/ Condition**	**Data values**	**Expected Result**	**Actual Result** (if different from expected)	**Successful/ Failed**	**Environment Nbr (if failed)**	**Log Nbr (if failed)**
Scenario 1: Loading login screen	Basic flow	None	Login screen appears		Successful		
Scenario 2: Login process successful	Basic flow	user ID: string password: string Login button clicked	Loads file upload page and login success message		Successful		
Scenario 3: Wrong sign-in information	Basic flow + alternative flow 1	user ID: string password: string Login button clicked	Error message appears; screen stays on login screen		Successful		

Scenario 4: Create another admin user	Basic flow	Create account button clicked	Create another account screen appears		Successful		
Scenario 5: Create another admin user successful	Basic flow	user ID: string first name: string last name: string email: string@string.com password: string Create button clicked	File upload page and new account made successfully message appears		Successful		
Scenario 6: Create another admin user checks email format works	Basic flow + alternative flow 2	user ID: string first name: string last name: string email: string@string.com password: string Create button	Error message appears; screen stays on create another account screen		Successful		

		clicked					
Scenario 7: Create another admin user checks if user ID already exists works	Basic flow + alternative flow 3	user ID: string first name: string last name: string email: string@string.com password: string Create button clicked	Error message appears; screen stays on create another account screen		Successful		
Scenario 8: Create another admin user unsuccessful due to password confirmation not matching	Basic flow + alternative flow 4	user ID: string email: string@string.com password: string Create button clicked	Error message appears; screen stays on create another admin user screen		Successful		
Scenario 9: Logout	Basic Flow	Click Logout button	Login screen appears		Successful		
				Test Case Status	Successful/ Failed		

6.1.3 Test Case for Data File Upload and Database Process

UC Scenario	Step Description/ Condition	Data values	Expected Result	Actual Result (if different from expected)	Successful/ Failed	Environment Nbr (if failed)	Log Nbr (if failed)
Scenario 1: Loading file upload page	Basic flow	Login button clicked	File upload page appears		Successful		
Scenario 2: File upload page successful	Basic flow	Excel file with the proper worksheets (i.e., PersonEngaged, Database Structure, OutCome, Demographic, ArrivalTime, Accessibility, NoShow, LastYearSeen, UniquePatient, AppointmentType, HTN_Diabetic) Upload button clicked	Loads file uploaded successfully page		Successful		
Scenario 3: File upload page unsuccessful check	Basic flow + alternative flow 1	Excel file with the proper worksheets (i.e., PersonEngaged, Database Structure, OutCome, Demographic, ArrivalTime,	Error message appears and lists missing worksheets; screen stays on file upload		Successful		

		Accessibility, NoShow, LastYearSeen, UniquePatient, AppointmentType, HTN_Diabetic) Upload button clicked	screen				
Scenario 4: File upload successful page	Basic flow	Upload button clicked	File upload successful page appears		Successful		
Scenario 5: Upload a new file	Basic flow	Upload new file button clicked	Loads file upload page		Successful		
				Test Case Status	Successful/ Failed		

6.1.4 Test Case for Dashboard Rendering & Navigation

UC Scenario	**Step Description / Condition**	**Data values**	**Expected Result**	**Actual Result** (if different from expected)	**Successful/ Failed**	**Environment Nbr (if failed)**	**Log Nbr (if failed)**
Scenario 1: Loading Main Dashboard	Basic flow	Main Dashboard tab clicked	Main Dashboard page appears and renders 8		Successful		

			mini graphs				
Scenario 2: Loads Patients Encountered Dashboard	Basic flow	Patients Encountered tab clicked	Patients Encountered page appears and renders 6 mini graphs		Successful		
Scenario 3: Loads Patient Goals & Outcomes Dashboard	Basic flow	Patient Goal & Outcomes tab clicked	Patient Goal & Outcomes page appears and renders 4 mini graphs		Successful		
Scenario 4: Loads Demographics Dashboard	Basic flow	Demographics tab clicked	Demographics page appears and renders 2 mini graphs		Successful		
Scenario 5: Loads Patient Appointments Dashboard	Basic flow	Patient Appointments tab clicked	Patient Appointments page appears and renders 5 mini graphs		Successful		
Scenario 6: Having all	Basic flow	“See more” button on any mini	Each “see more” leads to its		Successful		

25 “See more” links working		graph is clicked	respective bigger view of their graph				
Scenario 7: Having all 24 drop down options links working	Basic flow	Drop down options are clicked	Each drop down option leads to its respective bigger view of their graph		Successful		
Scenario 8: Loads application s page	Basic flow + alternative flow 1	application s page clicked	applicatio n graph loads		Successful		
Scenario 9: Loads hbA1C page	Basic flow + alternative flow 2	hbA1C page clicked	hbA1C graph loads		Successful		
Scenario 10: Loads bloodPress ure page	Basic flow + alternative flow 3	bloodPress ure page clicked	bloodPress ure graph loads		Successful		
Scenario 11: Loads cholesterol page	Basic flow + alternative flow 4	cholesterol page clicked	cholesterol graph loads		Successful		

Scenario 12: Loads demographics page	Basic flow + alternative flow 5	demographics page clicked	demographics graph loads		Successful		
Scenario 13: Loads followups page	Basic flow + alternative flow 6	followups page clicked	followups graph loads		Successful		
Scenario 14: Loads noShowPatient page	Basic flow + alternative flow 7	noShowPatient page clicked	noShowPatient graph loads		Successful		
Scenario 15: Loads patientAccessibility page	Basic flow + alternative flow 8	patientAccessibility page clicked	patientAccessibility graph loads		Successful		
Scenario 16: Loads patientArrivalTime page	Basic flow + alternative flow 9	patientArrivalTime page clicked	patientArrivalTime graph loads		Successful		
Scenario 17: Loads	Basic flow + alternative flow 10	patientEncounter page clicked	patientEncounter graph		Successful		

patientEncounter page			loads				
Scenario 18: Loads patientEnrollYear page	Basic flow + alternative flow 11	patientEnrollYear page clicked	patientEnrollYear graph loads		Successful		
Scenario 19: Loads patientReferredMassClinic page	Basic flow + alternative flow 12	patientReferredMassClinic page clicked	patientReferredMassClinic graph loads		Successful		
Scenario 20: Loads patientReferredToTelehealth page	Basic flow + alternative flow 13	patientReferredToTelehealth page clicked	patientReferredToTelehealth graph loads		Successful		
Scenario 21: Loads patientTypeServe page	Basic flow + alternative flow 14	patientTypeServe page clicked	patientTypeServe graph loads		Successful		
Scenario 22: Loads pctChronicCare page	Basic flow + alternative flow 15	pctChronicCare page clicked	pctChronicCare graph loads		Successful		

Scenario 23: Loads pctEligibleHealthCare page	Basic flow + alternative flow 16	pctEligibleHealthCare page clicked	pctEligibleHealthCare graph loads		Successful		
Scenario 24: Loads pctFiftyPerGeneralFollowUp page	Basic flow + alternative flow 17	pctFiftyPerGeneralFollowUp page clicked	pctFiftyPerGeneralFollowUp graph loads		Successful		
Scenario 25: Loads pctPrescriptApp page	Basic flow + alternative flow 18	pctPrescriptApp page clicked	pctPrescriptApp graph loads		Successful		
Scenario 26: Loads pctTwentyPerReferral page	Basic flow + alternative flow 19	pctTwentyPerReferral page clicked	pctTwentyPerReferral graph loads		Successful		
Scenario 27: Loads placeOfCare page	Basic flow + alternative flow 20	placeOfCare page clicked	placeOfCare graph loads		Successful		
Scenario	Basic flow + alternative	providedScreening	providedScreening		Successful		

28: Loads providedScreening page	flow 21	page clicked	graph loads				
Scenario 29: Loads twentyFivePerTelehealthReferred page	Basic flow + alternative flow 22	twentyFivePerTelehealthReferred page clicked	twentyFivePerTelehealthReferred graph loads		Successful		
Scenario 30: Loads uniquePatientInYear page	Basic flow + alternative flow 23	uniquePatientInYear page clicked	uniquePatientInYear graph loads		Successful		
Scenario 31: Loads volunteers page	Basic flow + alternative flow 24	volunteers page clicked	volunteers graph loads		Successful		
				Test Case Status	Successful/Failed		

6.1.5 Test Case for Dashboard Drilldown Generation of Years

UC Scenario	**Step Description/ Condition**	**Data values**	**Expected Result**	**Actual Result** (if different from expected)	**Successful/Failed**	**Environment Nbr (if failed)**	**Log Nbr (if failed)**
Scenario 1: Loads applications year drilldown	Basic flow + alternative flow 1	applications year drilldown clicked	application year drilldown loads		Successful		
Scenario 2: Loads hbA1C year drilldown	Basic flow + alternative flow 2	hbA1C year drilldown clicked	hbA1C year drilldown loads		Successful		
Scenario 3: Loads bloodPressure year drilldown	Basic flow + alternative flow 3	bloodPressure year drilldown clicked	bloodPressure year drilldown loads		Successful		
Scenario 4: Loads cholesterol year drilldown	Basic flow + alternative flow 4	cholesterol payear drilldownge clicked	cholesterol year drilldown loads		Successful		

Scenario 5: Loads demographics year drilldown	Basic flow + alternative flow 5	demographics year drilldown clicked	demographics year drilldown loads		Successful		
Scenario 6: Loads followups year drilldown	Basic flow + alternative flow 6	followups year drilldown clicked	followups year drilldown loads		Successful		
Scenario 7: Loads noShowPatient year drilldown	Basic flow + alternative flow 7	noShowPatient year drilldown clicked	noShowPatient year drilldown loads		Successful		
Scenario 8: Loads patientAccessibility year drilldown	Basic flow + alternative flow 8	patientAccessibility year drill-down clicked	patientAccessibility year drilldown loads		Successful		
Scenario 9: Loads patientArrivalTime year drilldown	Basic flow + alternative flow 9	patientArrivalTime year drilldown clicked	patientArrivalTime year drilldown loads		Successful		

Scenario 10: Loads patientEncounter year drilldown	Basic flow + alternative flow 10	patientEncounter year drilldown clicked	patientEncounter year drilldown loads		Successful		
Scenario 11: Loads patientEnrollYear year drilldown	Basic flow + alternative flow 11	patientEnrollYear year drilldown clicked	patientEnrollYear year drilldown loads		Successful		
Scenario 12: Loads patientReferredMassClinic year drilldown	Basic flow + alternative flow 12	patientReferredMassClinic year drilldown clicked	patientReferredMassClinic year drilldown loads		Successful		
Scenario 13: Loads patientReferredToTelehealth year drilldown	Basic flow + alternative flow 13	patientReferredToTelehealth year drilldown clicked	patientReferredToTelehealth year drilldown loads		Successful		
Scenario 14: Loads patientTyp	Basic flow + alternative flow 14	patientTypeServe year drill-down	patientTypeServe year drilldown loads		Successful		

eServe year drilldown		clicked					
Scenario 15: Loads pctChronicCare year drilldown	Basic flow + alternative flow 15	pctChronicCare year drilldown clicked	pctChronicCare year drilldown loads		Successful		
Scenario 16: Loads pctEligibleHealthCare year drilldown	Basic flow + alternative flow 16	pctEligibleHealthCare year drilldown clicked	pctEligibleHealthCare year drilldown loads		Successful		
Scenario 17: Loads pctFiftyPerGeneralFollowUp year drilldown	Basic flow + alternative flow 17	pctFiftyPerGeneralFollowUp year drilldown clicked	pctFiftyPerGeneralFollowUp year drilldown loads		Successful		
Scenario 18: Loads pctPrescriptApp year drilldown	Basic flow + alternative flow 18	pctPrescriptApp year drilldown clicked	pctPrescriptApp year drilldown loads		Successful		
Scenario	Basic flow + alternative	pctTwentyPerReferra	pctTwentyPerReferral		Successful		

19: Loads pctTwentyPerReferral year drilldown	flow 19	l year drilldown clicked	year drilldown loads				
Scenario 20: Loads placeOfCare year drilldown	Basic flow + alternative flow 20	placeOfCare year drilldown clicked	placeOfCare year drilldown loads		Successful		
Scenario 21: Loads providedScreening year drilldown	Basic flow + alternative flow 21	providedScreening year drilldown clicked	providedScreening year drilldown loads		Successful		
Scenario 22: Loads twentyFivePerTelehealthReferred year drilldown	Basic flow + alternative flow 22	twentyFivePerTelehealthReferred year drilldown clicked	twentyFivePerTelehealth Referred year drilldown loads		Successful		
Scenario 23: Loads uniquePatientInYear year drilldown	Basic flow + alternative flow 23	uniquePatientInYear year drilldown clicked	uniquePatientInYear year drilldown loads		Successful		

Scenario 24: Loads volunteers year drilldown	Basic flow + alternative flow 24	volunteers year drilldown clicked	volunteers year drilldown loads		Successful		
				Test Case Status	Successful/Failed		

6.2 Client Testing Summary Report

6.2.1 *Introduction*

Due to the client's busyness from his engagement at his own professional activities and being out of the country for a considerable amount of time during Waxwings' testing deliverable window for the UNF course, this testing summary report is in its partial stage.

Faisal, our client, was restricted to his cell phone device when viewing the application during the initial test. Consequently, he was able to only see the Administrator File Application through a demo of the product but was provided a link to the Quality Measure Dashboard Application that is hosted on Azure to test on his own computer or laptop device at a later time so that he could test it on a bigger screen.

This report is primarily based on his reaction and comments during our first live meeting which was recorded which demoed and tested both applications by Waxwings on their devices while Faisal provided verbal reactions and comments. This report will be updated with Faisal's written comments in the following deliverables.

6.2.2 *Test Summary*

This report provides a summary of results of the partial test performed on the Administrator File Application and Quality Measure Dashboard Application for MASS Clinic created by Waxwings.

6.2.2.1 Project Name

Waxwings' Administrator File Application

6.2.2.2 System Name

Administrator File Application

6.2.2.3 Version Number

2.0

6.2.2.4 Additional Comments

Demoed for client

Functional

Test Owner: Faisal (verbal communication)

Test Date: 3/22/22

Test Results: Sign-in page working properly with authentication and authorization. Upload file working with file checking. Submit screen prompt working. Admin ability registering a new account with admin access working. Confirmation of a new account made is working. Signing in with a new account is working.

Additional Comments: The authentication and authorization capabilities are being stored locally and will be moved to the database by the next deliverable. Faisal expressed his satisfaction with the product and had no further comments or questions about it.

6.2.2.5 Project Name

Waxwings' Quality Measure Dashboard Application

6.2.2.6 System Name

Quality Measure Dashboard Application

6.2.2.7 Version Number

3.0

6.2.2.8 Additional Comments

Tested with client

Functional

Test Owner: Faisal (verbal communication)

Test Date: 3/22/22

Test Results: Initial dashboard is working as well as the tabs navigation. "See more" link is working and leads to a bigger view page of the graph. Drill down option for year filtering is working and renders the updated data. Navigation bar and dropdown options are working.

Additional Comments: Faisal was very positive in his remarks about this application and how it looks. He explained how to organize the OutCome, LastYearSeen and HTN_Diabetic data better so that it would be beneficial for donors to view in certain graph formats.

6.2.3 Test Assessment

All tests for both applications covered the major functionalities of the requirements that were asked for by the client. The goal was to test if the Azure cloud hosted Dashboard was properly working and if the Admin sign in application could handle the authorization and authentication process; those major functionalities were successful locally and have been implemented unto Azure since then.

6.2.4 Test Results

The results of the demo and testing were satisfactory in terms of large picture goals and Faisal was able to guide Waxwings with how specific data should be graphed. No major software or hardware errors were encountered, but Waxwings would like further testing to occur. Minor issues to fix are logged in Waxwings test cases section of this full document. Those Dashboard issues need to be cross referenced with the data file to see if certain graphs would benefit from having a year drill down option. Also, one page is not rendering data as a bar graph. All these issues have been addressed and corrected by the Waxwings team as of May 2, 2022.

6.2.5 *Suggested Actions*

Recommendations by client are:

- Check the LastYearSeen data and graph it so that it shows the time period of how long certain patients have been with MASS clinic as this would be the most beneficial for donors to see and react to.
- Check the HTN_Diabetic data and group them into 3 groups of A1C, Cholesterol and Blood Pressure so that donors can see the cause and effect numbers more clearly.
- Check the OutCome data and display them similarly to the HTN_Diabetic data where groupings show comparisons better.

As a note, all recommendations listed here was from the initial testing with the client; all these suggestions have been solved. Waxwings recommends the client to continue testing the applications with the hosted links that were provided on MASS Clinic's computers to check compatibility and functionality.

7. Project Installation and Delivery

7.1 Installation Guide

Tools Needed

1. MongoDB
2. Azure Portal
3. Visual Studio Code
    a. Azure App Service Extension
        i. Go into the extension area in vs code
        ii. Search for "Azure App Service"
        iii. Select install
        iv. Go to the new icon
        v. Log into your azure account in the extension

Installing the Database

1. Log into Azure Portal
2. Select "Create a Resource"
3. Search for Azure Sql
4. Select the setting you would like
5. In the query editor insert the given bacpac in attached file
6. Follow these steps https://docs.microsoft.com/en-us/azure/azure-sql/database/database-import?view=azuresql&tabs=azure-powershell

Installing the Application

1. General
    a. Log into Azure Portal
    b. Create a resource group
2. The Dashboard
    a. Go to github and find https://github.com/tcollincox/mass-clinic-dashboard-waxwings
    b. Clone the project to your computer
    c. Open a new terminal inside visual studio code
    d. Run npm install
    e. Create a file in the directory called ".env"
    f. In that file put the following information
        i. USERNAMELOGIN = <$yourdatabaseloginusername>
        ii. PASSWORDLOGIN = <$yourdatabaseloginpassword>
        iii. MASSCLINICDATABASENAME = <$yourdatabasename>
        iv. HOSTNAME = <$yourdatabaseserverlink>
    g. Deploy app
        i. Go to the "Azure App Service Extension"
        ii. Select the deploy app icon(A cloud with an arrow)
        iii. Select the file you are currently using for your file to deploy
        iv. Select the resource group that you made previously
        v. Select Node 16
        vi. Select Windows
        vii. Select your desired region (East US recommended)
        viii. Create a new app service plan
        ix. Select "Skip for now" when prompted with application insights
        x. When prompted if you "always want to deploy to…" select yes
        xi. Once prompted with "Deploy website" go to the Azure App Service Extension and expand the menu options
        xii. Add SCM_DO_BUILD_DURING_DEPLOYMENT as true to Application Settings
        xiii. Right click Application Settings and select Upload Local Settings
            1. Select .env as your option
        xiv. Select the deploy app icon again
        xv. Once website is loaded check to see if you see an empty home screen with a nav bar
3. The Upload page backend 1 (Uploading backend)
    a. Go to github and find https://github.com/tcollincox/mass-clinic-node-excel-backend
    b. Clone the project to your computer
    c. Open a new terminal inside visual studio code
    d. Run npm install
    e. Copy the .env file from The Dashboard and paste it in this projects directory

f. Deploy app
  i. Go to the “Azure App Service Extension”
  ii. Select the deploy app icon(A cloud with an arrow)
  iii. Select the file you are currently using for your file to deploy
  iv. Select the resource group that you made previously
  v. Select Node 16
  vi. Select Linux
  vii. Select your desired region (East US recommended)
  viii. Create a new app service plan
  ix. Select “Skip for now” when prompted with application insights
  x. Right click Application Settings and select Upload Local Settings
    1. Select .env as your option
  xi. Select the deploy app icon again

4. The Upload page backend 2(Login and create account backend)

a. Go to github and find https://github.com/JatJax/mass-clinic-login-server
b. Clone the project to your computer
c. Open a new terminal inside visual studio code
d. Run npm install
e. Create a file in the directory called “.env”
f. In that file put the following information
  i. DB_CONN_STRING=<mongo string with password>
  ii. DB_NAME=<mongo db name>
  iii. MASS_CLINIC_COLLECTION_NAME=<collection name>
  iv. USERS_COLLECTION_NAME=<users collection name>
  v. SECRET= <secret code>
g. Deploy app
  i. Go to the “Azure App Service Extension”
  ii. Select the deploy app icon(A cloud with an arrow)
  iii. Select the file you are currently using for your file to deploy
  iv. Select the resource group that you made previously
  v. Select Node 16
  vi. Select Linux
  vii. Select your desired region (East US recommended)
  viii. Create a new app service plan
  ix. Select “Skip for now” when prompted with application insights
  x. Right click Application Settings and select add from local
    1. Select .env as your option
  xi. Select the deploy app icon again

5. The Upload Frontend

a. Go to github and find https://github.com/JatJax/massclinic-login-server
b. Clone the project to your computer
c. Open a new terminal inside visual studio code
d. Run the command cd ./angular-login/

   e. Run npm install
   f. Name routes to backends
      i. Go to the file src
      ii. Go to the file in that called environment.prod.ts
      iii. In that file add
         1. UPLOADSITEURL: 'https://mass-clinic-upload-backend-final.azurewebsites.net/',
         2. BACKENDURL: 'https://mass-clinic-login-server.azurewebsites.net/'
   g. In the terminal run "ng build --base-href"
   h. Deploy app
      i. Go to the "Azure App Service Extension"
      ii. Select the deploy app icon(A cloud with an arrow)
      iii. Go to browse file
         1. Go to the current project you are in and then go to angular-login/dist/angular-login
         2. Select that file
      iv. Select the resource group that you made previously
      v. Select Node 16
      vi. Select Windows
      vii. Select your desired region (East US recommended)
      viii. Create a new app service plan
      ix. Select "Skip for now" when prompted with application insights
      x. When prompted if you "always want to deploy to…" select yes
      xi. Once prompted with "Go to Website" go to the website and try to log in

Allowing Cors

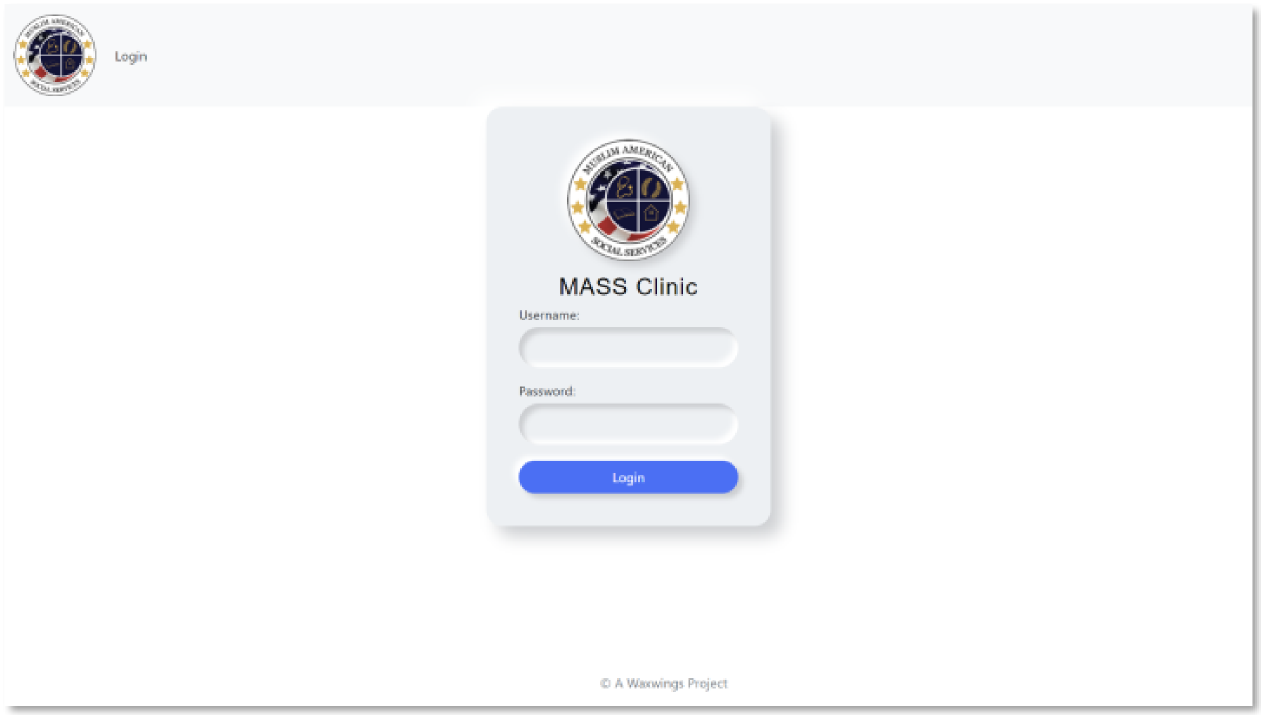
1. Allowing cors on upload backend
   a. Go to Azure Portal
   b. Go to App Services
   c. On the left side search "CORS"
   d. Where it allows you to enter URLs enter the URL for the upload frontend
2. Allowing cors on login backend
   a. Go to Azure Portal
   b. Go to App Services
   c. On the left side search "CORS"
   d. Where it allows you to enter URLs enter the URL for the upload frontend

There is also an accompanying file named "database.bacpac".

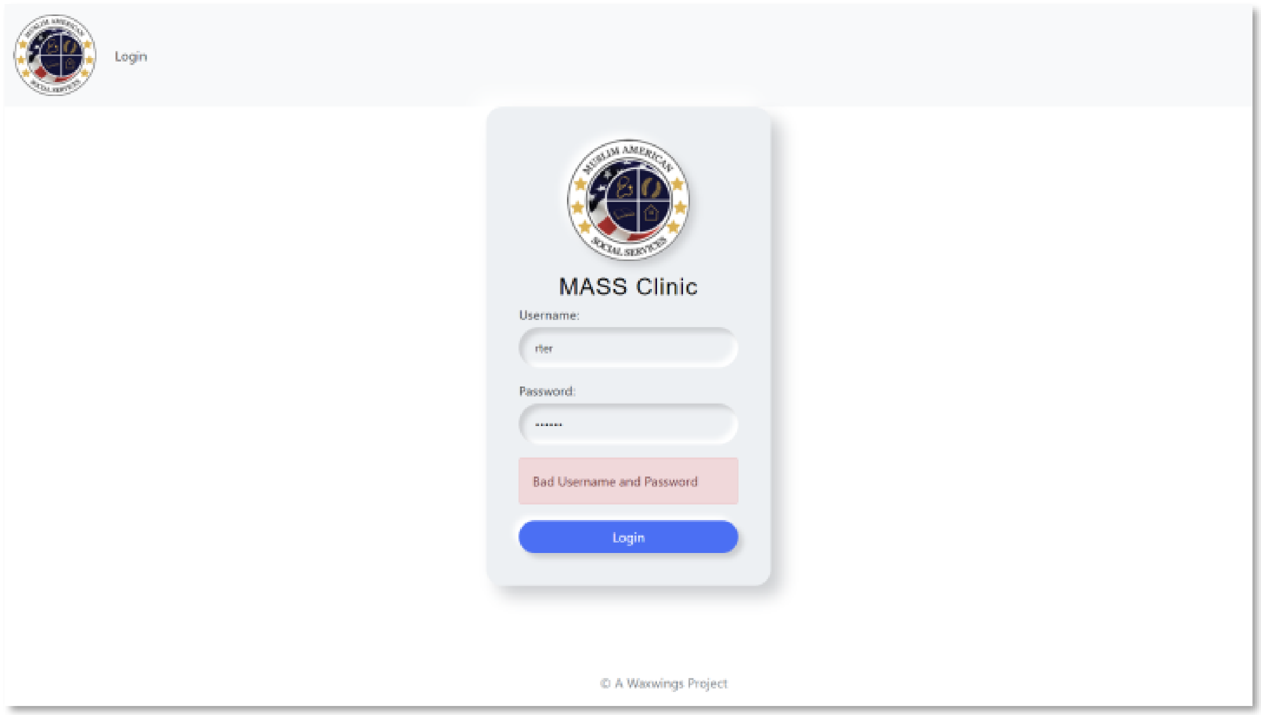
7.2 User Manual

7.2.1 Administrator File Application

Use provided link to get to admin's landing page as seen below:

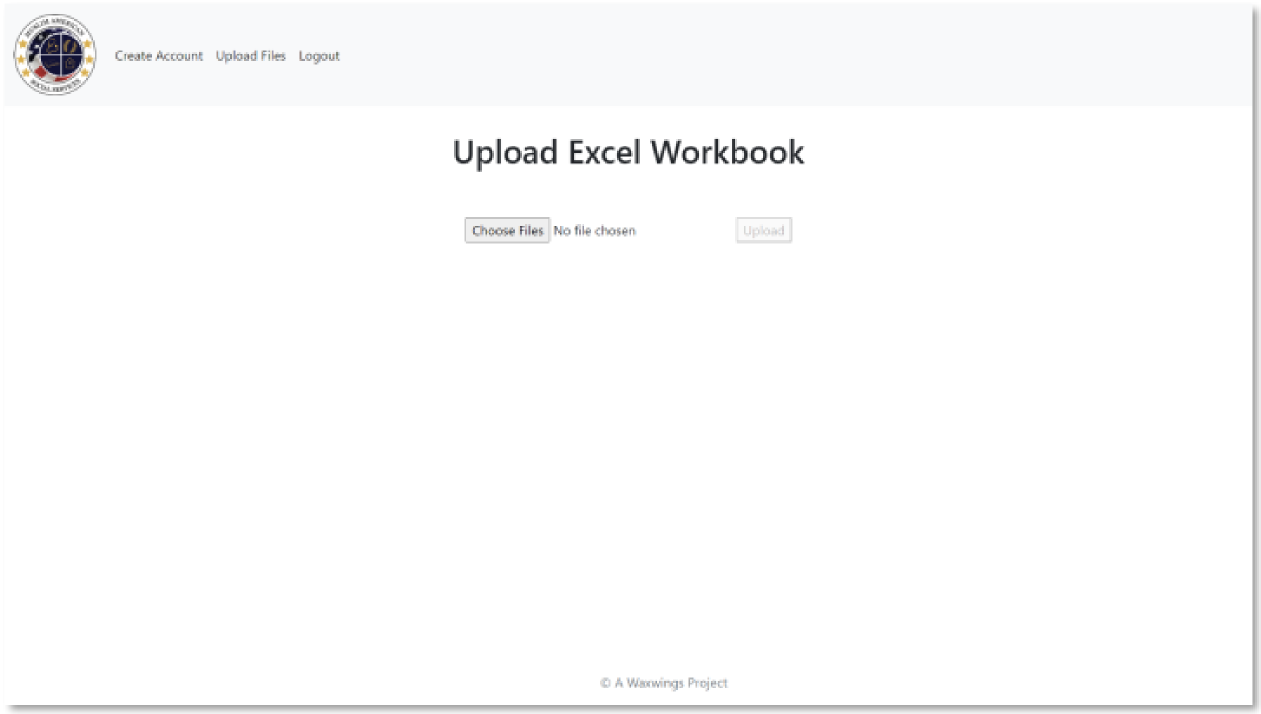


Enter your Username and Password and click the blue "Login" button. If an incorrect username or password is entered, you will see the error message below:



Bad Username and Password

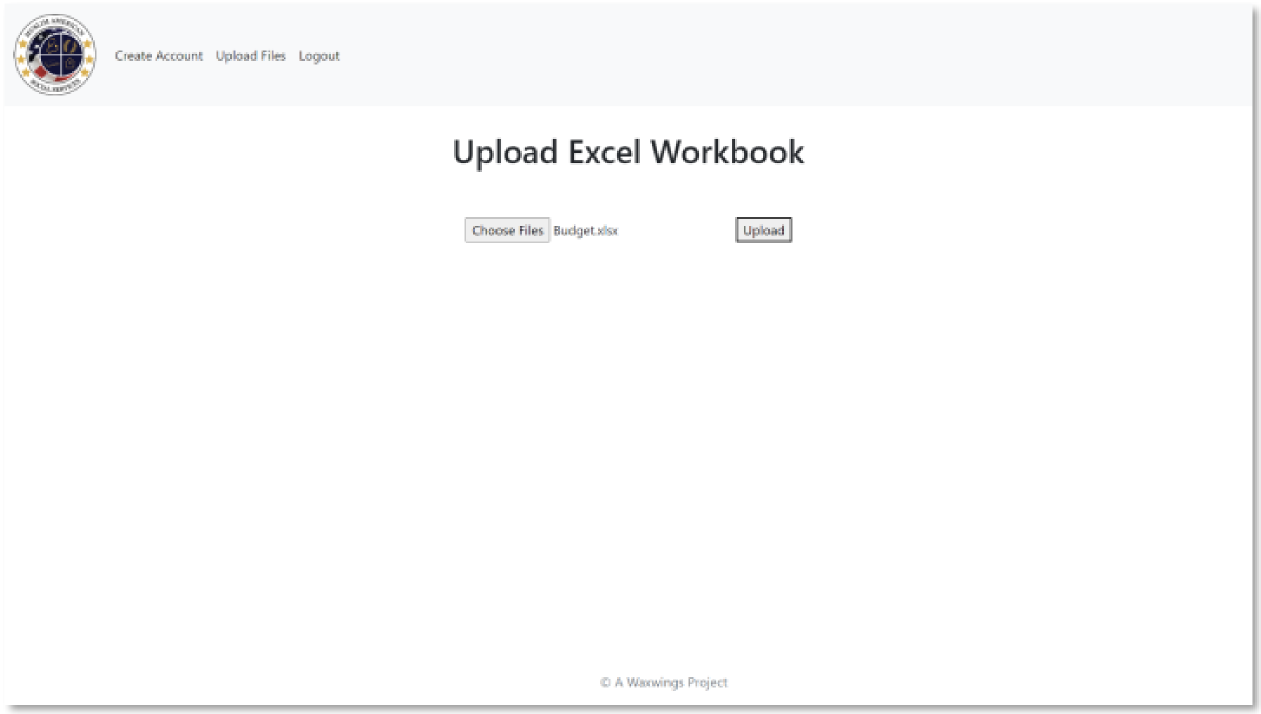
A successful login will lead to the file upload page as seen below:



Upload Excel Workbook

© A Waxwings Project

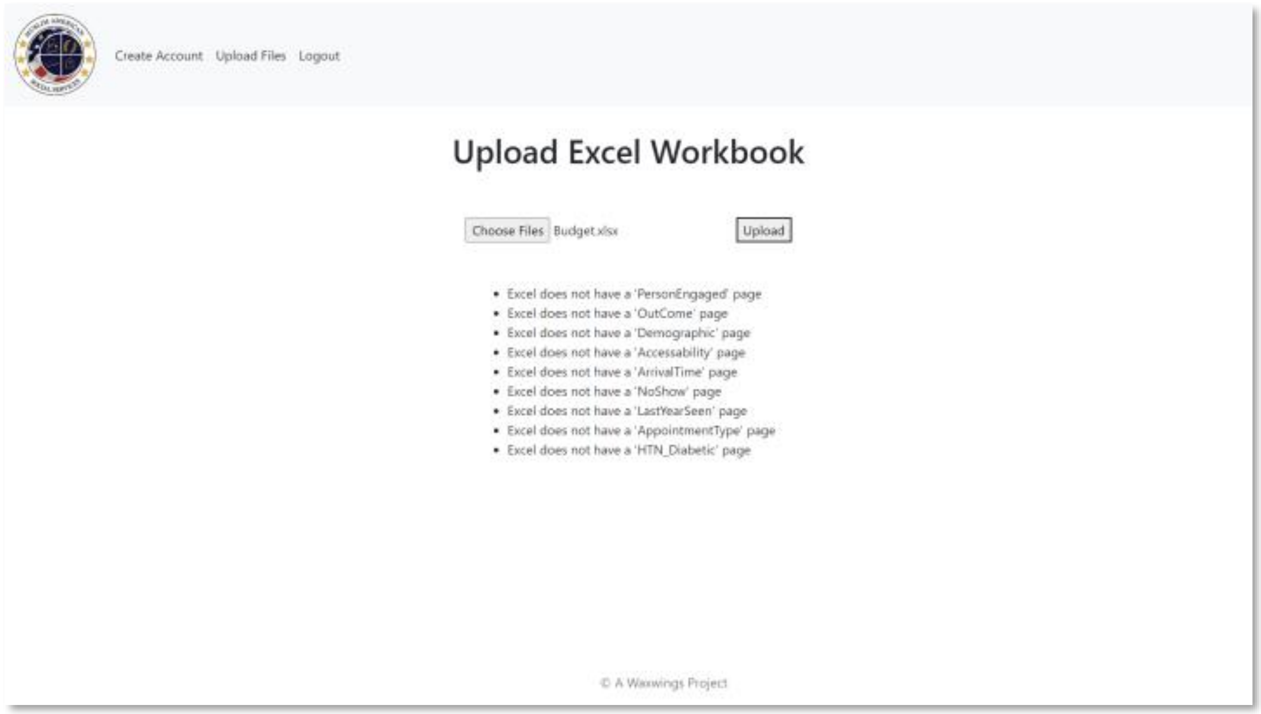
Click "choose files" and click on the file you would like to upload, then click the grey button "upload" as seen below:



Upload Excel Workbook

© A Waxwings Project

If the file does not meet the page requirements that the application will look for, an error message list will show which pages are missing as seen below:

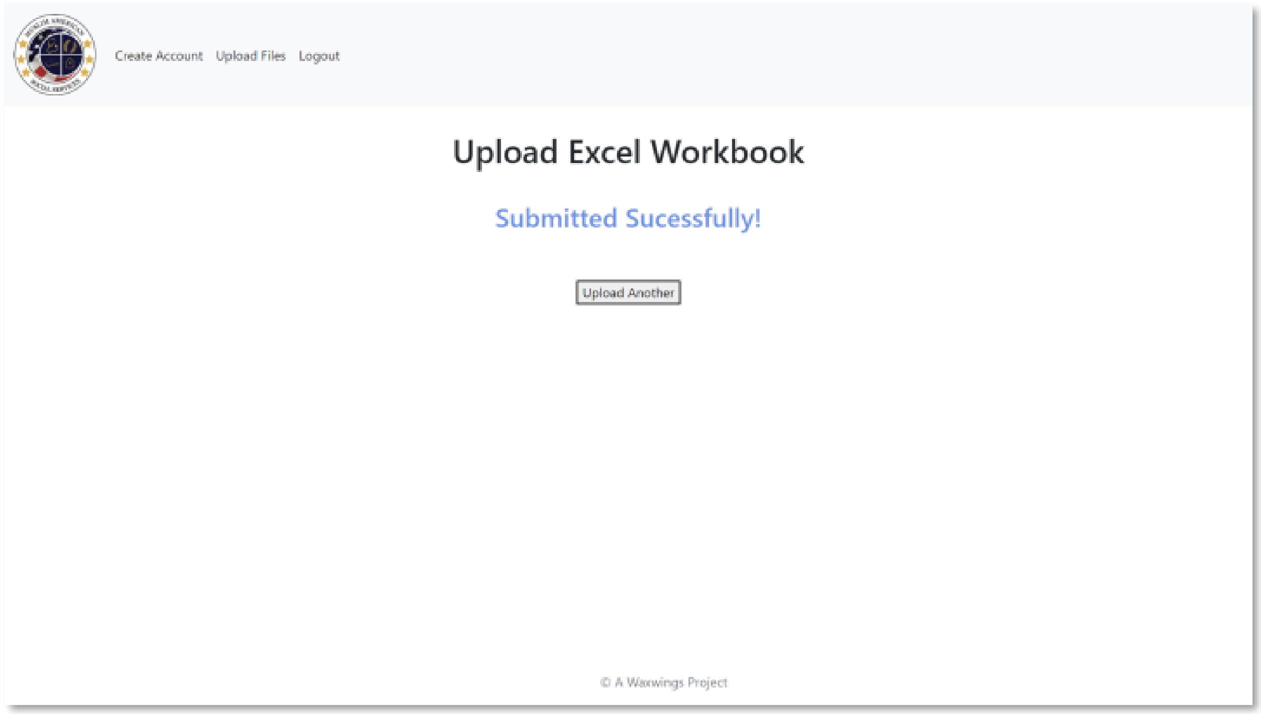


Upload Excel Workbook

- Excel does not have a 'PersonEngaged' page
- Excel does not have a 'OutCome' page
- Excel does not have a 'Demographic' page
- Excel does not have a 'Accessability' page
- Excel does not have a 'ArrivalTime' page
- Excel does not have a 'NoShow' page
- Excel does not have a 'LastYearSeen' page
- Excel does not have a 'AppointmentType' page
- Excel does not have a 'HTN_Diabetic' page

© A Waxwings Project

If a successful upload of a file occurs, you will be directed to the page seen below where you then have the option to upload another new file, create a new admin user or logout. If you click "Upload Another" or "Upload Files" you will be redirected to the upload file page again. If you click "Logout", you will be directed to the home landing page again.

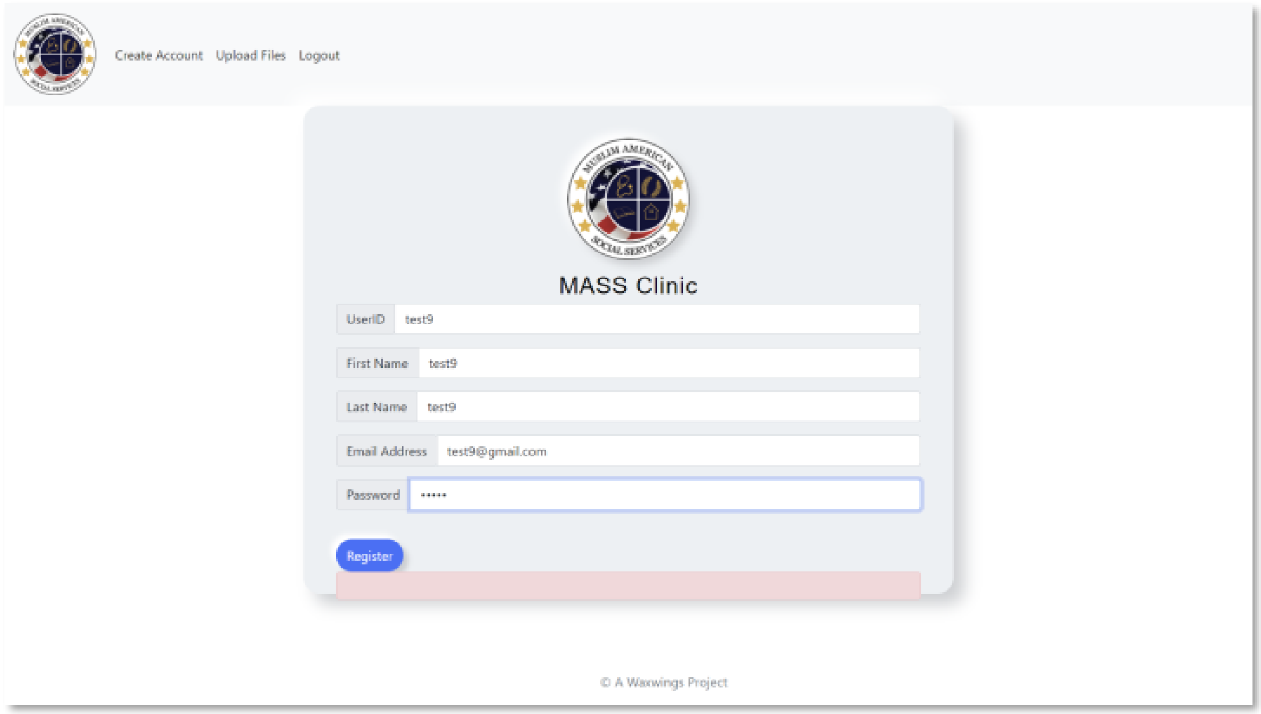


Upload Excel Workbook

Submitted Sucessfully!

© A Waxwings Project

When you click “Create Account” to create another admin account, you will be directed to the page below where you will be prompted to enter a user ID, a first name, a last name, a proper email and a password.

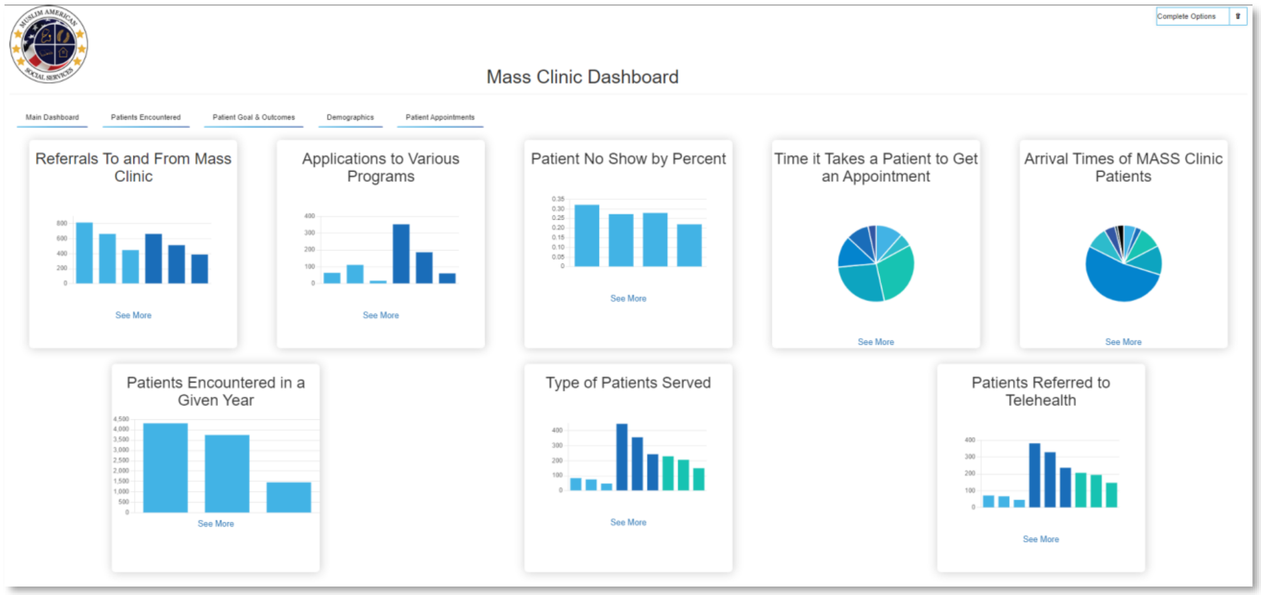


UserID: test9
First Name: test9
Last Name: test9
Email Address: test9@gmail.com
Password: •••••

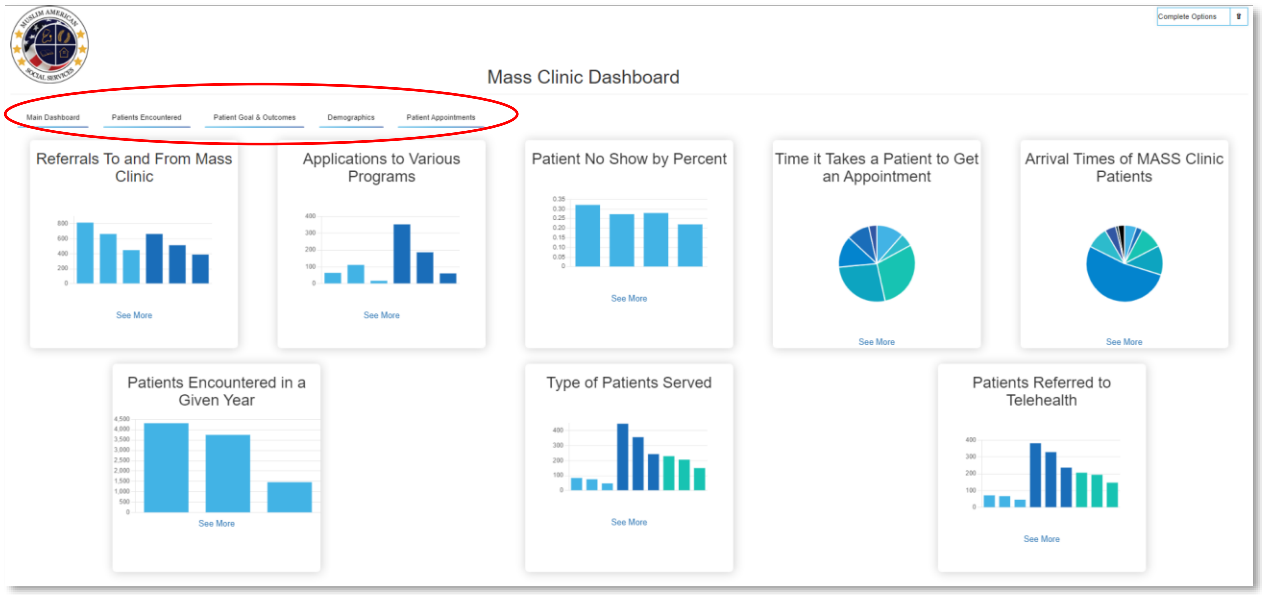
After a successful creation of a new account, you will automatically be sent back to the home landing page.

7.2.2 *Patient Quality Measurement Dashboard Application*

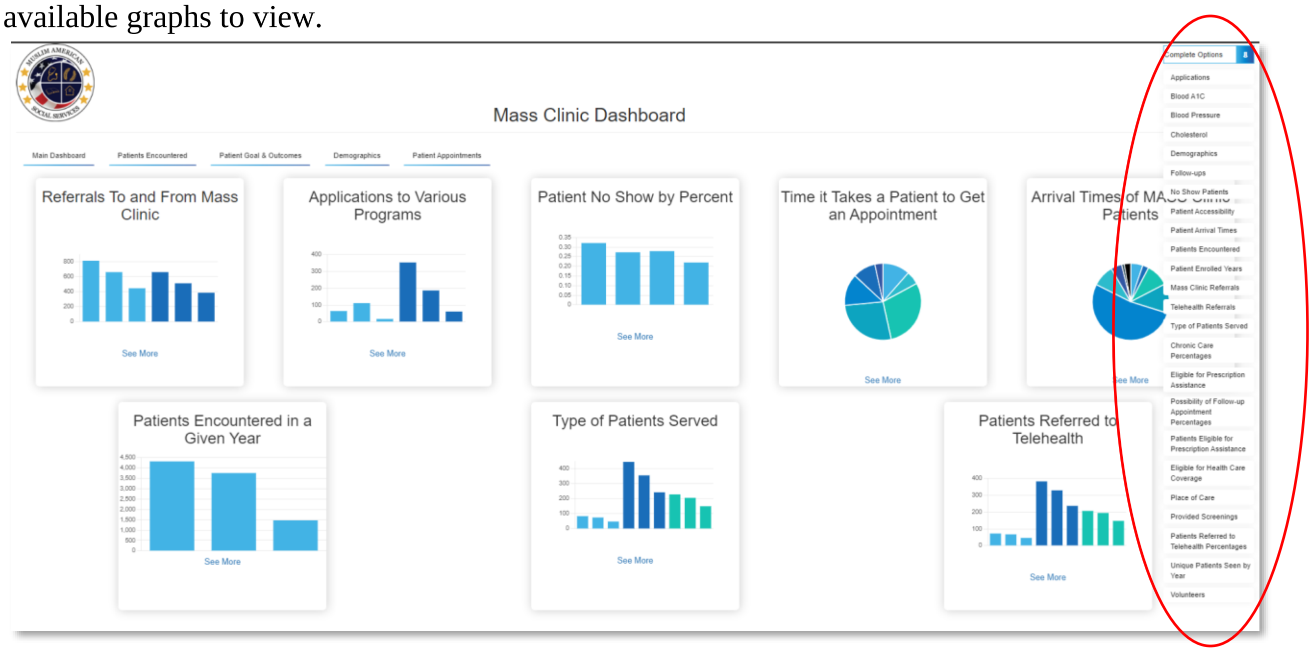
Use provided link to get to dashboard landing page as seen below:


Complete Options
Mass Clinic Dashboard
Main Dashboard
Patients Encountered
Patient Goal & Outcomes
Demographics
Patient Appointments
Referrals To and From Mass Clinic
Applications to Various Programs
Patient No Show by Percent
Time it Takes a Patient to Get an Appointment
Arrival Times of MASS Clinic Patients
Patients Encountered in a Given Year
Type of Patients Served
Patients Referred to Telehealth
See More

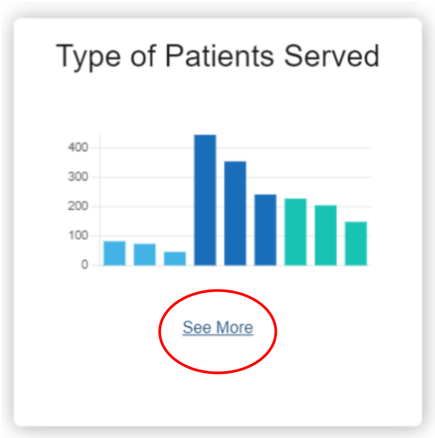
To navigate around the website, you can hover over the navigation bar options to go to other dashboards that group correlated graphs.

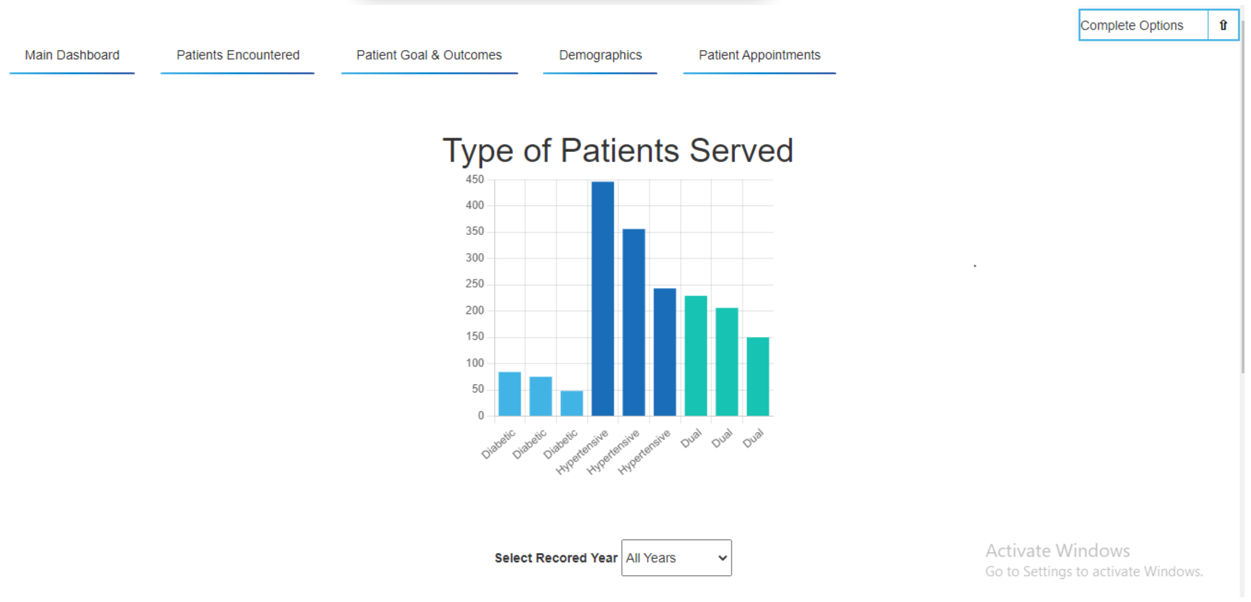

Complete Options
Mass Clinic Dashboard
Main Dashboard
Patients Encountered
Patient Goal & Outcomes
Demographics
Patient Appointments
Referrals To and From Mass Clinic
Applications to Various Programs
Patient No Show by Percent
Time it Takes a Patient to Get an Appointment
Arrival Times of MASS Clinic Patients
Patients Encountered in a Given Year
Type of Patients Served
Patients Referred to Telehealth
See More

You can also go to individual graphs by clicking the dropdown arrow to see a list of all available graphs to view.

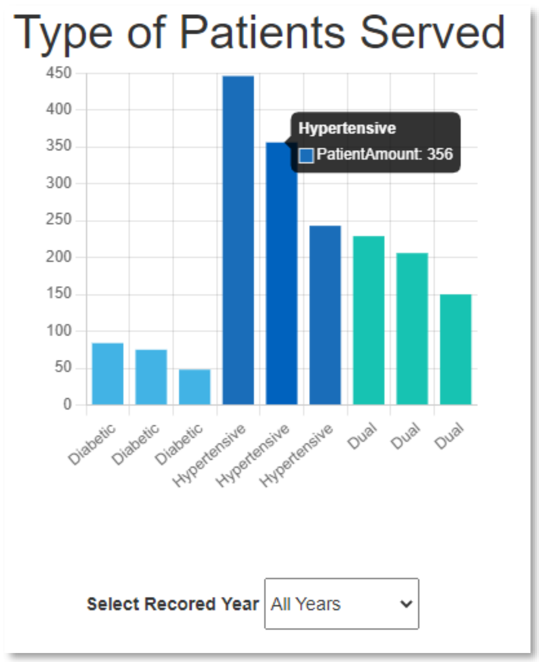

Mass Clinic Dashboard
Main Dashboard
Patients Encountered
Patient Goal & Outcomes
Demographics
Patient Appointments
Referrals To and From Mass Clinic
Applications to Various Programs
Patient No Show by Percent
Time it Takes a Patient to Get an Appointment
Arrival Times of MASS Clinic Patients
Patients Encountered in a Given Year
Type of Patients Served
Patients Referred to Telehealth
See More
Complete Options
Applications
Blood A1C
Blood Pressure
Cholesterol
Demographics
Follow-ups
No Show Patients
Patient Accessibility
Patient Arrival Times
Patients Encountered
Patient Enrolled Years
Mass Clinic Referrals
Telehealth Referrals
Type of Patients Served
Chronic Care Percentages
Eligible for Prescription Assistance
Possibility of Follow-up Appointment Percentages
Patients Eligible for Prescription Assistance
Eligible for Health Care Coverage
Place of Care
Provided Screenings
Patients Referred to Telehealth Percentages
Unique Patients Seen by Year
Volunteers

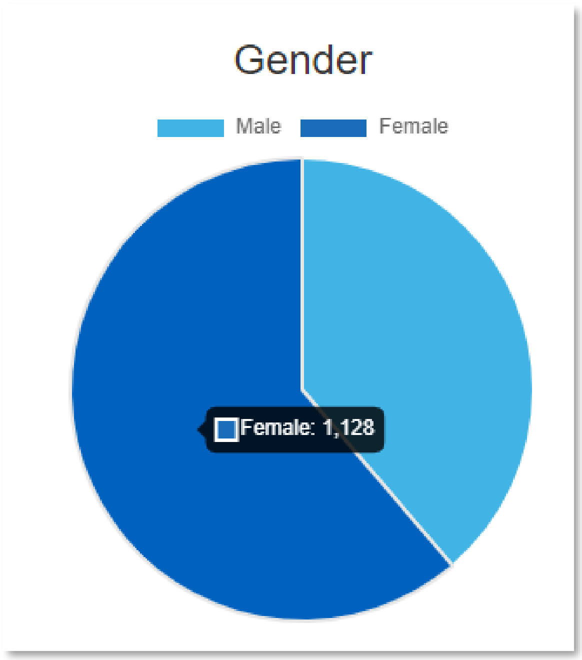
When you are on any of the dashboards, you can simply click on "See More" to view the graph by itself. The examples below shows what will be shown if you click "See More" under the Type of Patients Served and its individual page

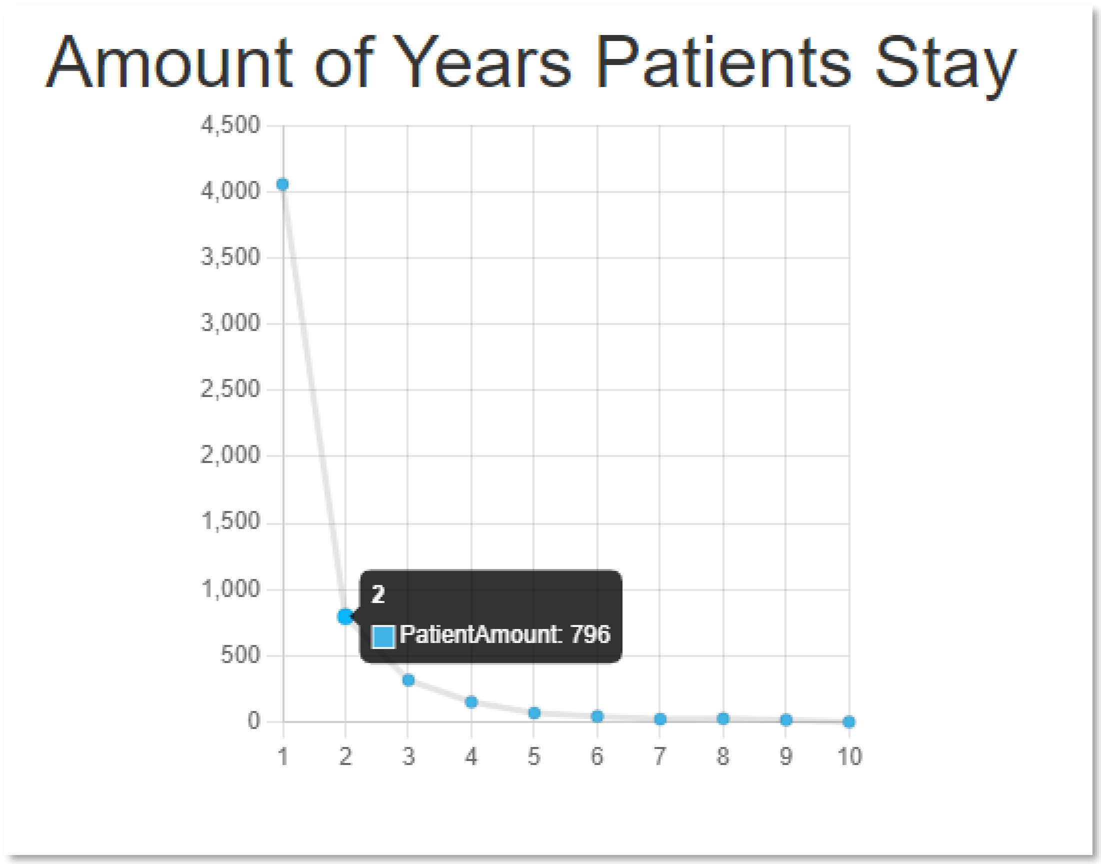




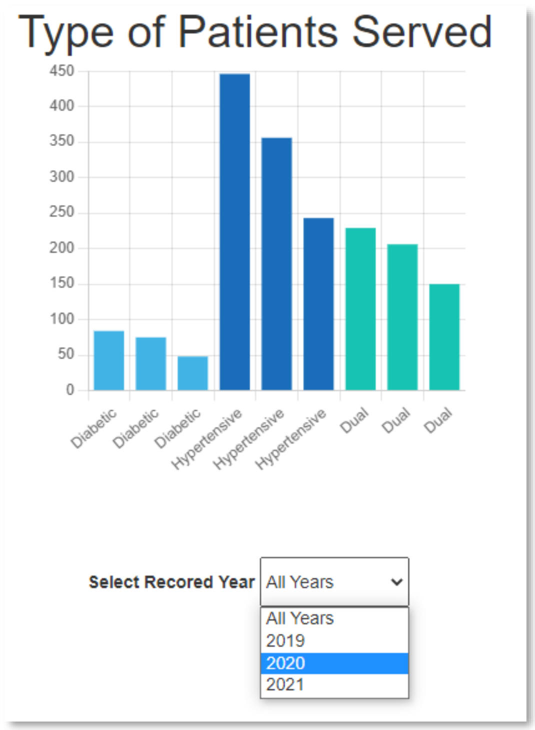
While on the induvial page, you can hover your mouse over the colored bar, pie or line elements to see their data specifications as see in the examples below.

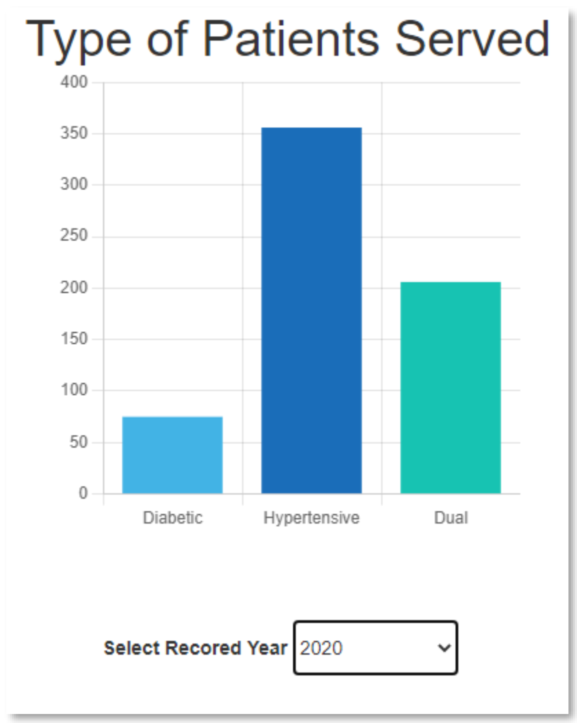






With certain graphs, year drill downs are available. To choose a year, click on the down arrow next to "All Years" and select a year. In the example below, the year "2020" was chosen and the graph then updates itself to show only the 2020 data.





7.2.3 Dashboard Application UI

The layout of the main dashboard and the four other grouped dashboards labeled “patients encountered,” “patients goal & outcomes,” “demographics,” and “patient appointments,” take the same layout of having a the page clearly labeled with the respective title in the center, the complete options dropdown on the top-left and the navigation tabs going across the top as seen in the pictures below. The main body of the page is taken up by rows of “cards” that will preview the possible graph options to choose from; the photos below show examples of those preview cards and Waxwings will complete all the smaller preview graphs by the following deliverable.

The layout of the individual graph pages is similar to the main and sub-grouped dashboards when it comes to navigation, but the main body of those pages will contain only the heading of the graph, the graph and a drilling option.

Referencing the photos for the dashboards and the individual graphs pages above, the titles above each page is crucial for the donor (i.e., user) to know where he or she is in the application. The titles are clearly seen and with further help from a client the wording of each title will be more precise in the following deliverable. Nevertheless, all views in the application do have headings.

When it comes to the individual graphs, each will have its own heading or title, but also the value axes text, value axes labels, graph categories, graph legend and actual shapes (i.e., bar, circle, donut, etc.) all have clear labels. As a note, the labels for the shapes appear when the cursor hovers over the specific series.

Due to the process of developing a new MASS Clinic website, the aesthetics of the Quality Management Application are kept simple so that MASS Clinic can easily integrate the Waxwings application when they are ready. Nevertheless, we have matched our fonts and color scheme to the current MASS Clinic website fonts and color scheme which are from the Open Sans, Arial and sans-serif font families and their tones of blue.

All elements on the dashboard application react mostly on the hover of the cursor. This ensures that donors know what they can view, what they can click, and if the application is responding to their clicking. In the photos below, one can see how the individual cards on the dashboard rise and gain color when hovered over. A color reaction also occurs when hovering over the navigation var tabs and dropdown options.

The quality dashboard application is consistent when it comes to interactions. Each navigation tab changes colors when the cursor hovers over it and it can be clicked just like any other tab. The dropdown menu options also change colors and can be clicked like any other dropdown option. Clicking on a card to view a graph is also consistent with all the graphs on the application.

7.2.4 *Administrator File Application UI*

The simplicity of the design, the labeling of the pages and components and the reactionary components of the elements on the page allows the donor to accomplish what they would like to do (i.e., navigate between views using the dropdown or tabs, look at a graph, choose a drill down option for the graph, etc.) with ease.

The layout of the administrator user application is simpler than the dashboard layout due to the requirements of this application. The prompts for the views on this application are all seen in the middle with the exception of the top bar that has login/logout options and MASS Clinic’s logo. That top bar is static and consistent on all the admin’s application views while the main body of the view layouts the admin’s current step in the sign-in and uploading process.

The prompts for each view on this application clearly shows the content that it is related to. The sign in page only has the sign in prompts. The file upload view only has its relevant content. Lastly, the successful upload page is clearly labeled as such with its “File Uploaded Successfully!” message.

The aesthetics of the admin application is more subdued compared to the dashboard application due to the simpler interaction the user will have in this use case. Nevertheless, the admin application does mimic certain aspects of the MASS Clinic website and Waxwings dashboard application with its use of the same font family and the gray scheme which is the other major color of the current MASS Clinic website along with blue.

Using the parts of the admin application is straightforward and easy to learn due to how it mimics very common actions used in other software applications. The sign-in prompt is similar to other websites and applications’ sign-in options with a username, password and sign in button. The file upload view is similar to how other applications will allow a user to choose a file and show how it is the chosen uploaded option. Finally, the message of a successful upload is common practice so that the user does not have any doubts about the process.

The admin application is primarily consistent with its use of buttons that will generate the sign-in, the file upload, the possible creation of another user and the return to the homepage. For instance, all blue buttons are final actions that lead to another view.

It is minimal effort when it comes to the admin application. The number of views that are seen can be a minimum of three (i.e., sign-in view, upload view, file successful view). The number of interactions with the views are also held to a minimum such as scrolling up/down or clicking on elements.

7.3 Delivery Process Adopted

The source code for our applications can be found on GitHub and is accessible by the client, mentor, team and professor as set up in the fall of the year 2021. The Dashboard application is already in Azure while the Admin Application is in the process of being integrated into Azure. The applications will be pushed from Waxwings' Azure account to our client's Azure account and the process will most likely begin with the Dashboard Application. Once that is turned over, the Dashboard Application will be a public application that can be accessed and viewed from the MASS clinic website while the Admin Application will remain private for MASS clinic's admin users only. The cloud deployment process and the set-up guide are found in previous sections.

Due to the nature of the project and the client we are working with, there is not much training that is needed for Waxwings' applications. Faisal is the only person that needs to be trained. The client has been informed earlier on that they will be using Microsoft's Azure services and database, and they have agreed upon the use of Angular and Node.JS for the applications. The client is also aware of how the applications are most compatible with Google Chrome.

Documentation and the openness of Waxwings to be contacted even in the future can address any questions about the code, how sections of the applications were set-up, integration to the MASS clinic systems, etc. Waxwings will see to it that these applications are delivered, integrated, implemented and used as they were required, developed and intended for on the client's systems.

8. Conclusion

8.1 Contributions and Limitations

The software developed covers all the major requirements that was given by the client. Some satisfactory aspects of the developed software product are the abstraction of the Admin application from the Dashboard application, ability to read an Excel file instead of CSV file, live graph rendering changes, the security of having chosen admin users, organizing all the data into visually appealing graphs and the ability to simply create a connection link from the MASS Clinic website to the Dashboard application. Some disadvantages of the created software are some CSS lack of responsive design, higher security for the administrator application and having separate applications.

8.2 Benefits to Community Partner

From the software given, MASS Clinic will be able to visualize their data. Visualizing data allows for better analysis of data and make decision more quickly. It also allows for identifying patters easily to see up or down trends. This not only helps gain donors or let donors see where their contribution is going but it could also help the medical volunteers see patient quality and where they could adjust their work. Visualization of data can also help in identifying data errors.

Another value added to MASS Clinic is how secured the data is because it is on a different admin application. The patient data is only handled by the admin user and the donor user does not even know that the admin application exists. This allows the different users to focus on their specified cases.

8.3 Future Work

There are many ways to build upon the Waxwings products. For the Dashboard application, more graphed data can be added and more drilldown option other than just year can be done. The CSS of that application can be redone to be more responsive as well. The ability to download the charts into picture files for reports or presentations would also be beneficial. The Admin application's security could be heightened if sensitive patient data will become part of the excel file. A different file type could also be programed to be used. The Admin application could have a link that leads to the dashboard after a successful rendering. The entire product is open for growth and further development.

8.4 Summary

Waxwings group created two applications for MASS Clinic: the Quality Measure Dashboard System and the Administrator File Application. The Quality Measure Dashboard System displays graphs of MASS Clinic's patient data while the Administrator File Application allows an admin user to upload a file of patient data so that it is graphed and displayed on the Quality Measure Dashboard System application.

MASS Clinic's problem of unclear information regarding patient medical results affects MASS Clinic's impact of gaining potential donors and representing the data in a more visual manner. The overall finished product is able to have an administrator user to upload an excel file that contains the patient data, the data would then be stored into a database successfully, a website application will graph all the patient data properly by their respective categories, each graph should have a drilldown option that is based on year if appropriate to that data, and the design should be obvious for potential donor users when interacting with the elements on the page. The Dashboard application follows a MVC model, more specifically a MVR model where the routes takes the role of the controller in the conventional Model-View-Controller model. The Dashboard application and Administrator application communicate on a 3-Tier Client Server model where the Dashboard application requests and receives data from a server and the server requests and receives data from a database. The Administrator application communicates with the database through a pipe-and-filter model because the donor user has no need to be aware of the administrator user side of the product. The entire product uses HTML 5, CSS 2.1, JavaScript ES5, the Node.JS and Angular.JS frameworks, ReactJS, MongoDB and Microsoft Azure Services; the main web browser used is the latest Google Chrome.