**Digitization of Registration Form & Creation of Database for Shalom Health Fair**

*EPICS @ BUTLER UNIVERSITY*

*Spring 2024*

*Shalom Health Center Team*

*Reece Parks, Matt Fleschner, Rachel Maier, Hallie Nuccio, Miles Mann*

Table of Contents

[Chapter 1: Introduction 3](#_Toc165823641)

[Organization of Report 3](#_Toc165823642)

[Problem Statement & Objectives 3](#_Toc165823643)

[Motivation & Rationale 3](#_Toc165823644)

[Customer/Developer Description 4](#_Toc165823645)

[Approach to the Problem 4](#_Toc165823646)

[Glossary 4](#_Toc165823647)

[Chapter 2: Specification of Requirements 5](#_Toc165823648)

[Functional Requirements 5](#_Toc165823649)

[Non-Functional Requirements 6](#_Toc165823650)

[Assumptions & Constraints 6](#_Toc165823651)

[Chapter 3: Architecture 7](#_Toc165823652)

[Overview of High-Level System Architecture 7](#_Toc165823653)

[Decomposition & Style 7](#_Toc165823654)

[System Information Description 7](#_Toc165823655)

[Chapter 4: Design 7](#_Toc165823656)

[Description of User Interface 7](#_Toc165823657)

[Chapter 5: Implementation 12](#_Toc165823658)

[Selection of Languages 12](#_Toc165823659)

[Coding Standards & Comments 12](#_Toc165823660)

[Implementation & Distribution of Work 13](#_Toc165823661)

[Organization of the Code Base 13](#_Toc165823662)

[Chapter 6: Quality Assurance & Testing 15](#_Toc165823663)

[Selection of Testing Objectives 15](#_Toc165823664)

[Testing Approach & Methods 15](#_Toc165823665)

[Defect Detection & Management 15](#_Toc165823666)

[Chapter 7: Project Organization and Management 16](#_Toc165823667)

[Roles & Contribution 16](#_Toc165823668)

[Roles & Their Propose 16](#_Toc165823669)

[Team Member & Their Contributions 17](#_Toc165823670)

[Management Process 18](#_Toc165823671)

[Weekly Status Reports 18](#_Toc165823672)

[Detailed User Manual 19](#_Toc165823673)

[Chapter 8: Future Work 21](#_Toc165823674)

[Explanation of Next Steps 21](#_Toc165823675)

[Implementation Needs 21](#_Toc165823676)

[References & Bibliography 22](#_Toc165823677)

[Appendices 22](#_Toc165823678)

# **Chapter 1: Introduction**

## Organization of Report

This report is a comprehensive analysis of our progress working with Shalom Health Care Center in EPICS for the Spring Semester of 2024. This consists of an overview of our project including the specifications of both the functional and non-functional requirements, the organization and management of the project, and details and suggestions for future teams. The order of the report is as follows:

1. Introduction to the Report – description of client, the goals of the project, and the approach.
2. Specification of Requirements – outlines specifications of each aspect of the project.
3. Architecture of Different Systems Used – displaying/describing the building of each system of the project.
4. Design of the User Interface – description of the form we created.
5. Implementation of the Work – background on the work and why we chose certain paths.
6. Quality Assurance & Testing – description of the error testing from both front-end and back-end.
7. Project Organization & Management – description of how the group worked together/the responsibilities of each member.
8. Outline of Future Work – describe what future work needs to be completed on the project.

For more detailed information about the report's layout, refer to the previous page's Table of Contents.

## Problem Statement & Objectives

Every summer, Shalom Health Care Center hosts their Dr. Danneé Neal Back-To-School Health Fair. This event provides the opportunity for people in the greater Indianapolis area to receive vaccinations, participate in check-ups, and get any other additional healthcare they may need. In the past, Shalom has been using paper registration forms that everyone is required to fill out upon arrival. This caused attendees to miss out on valuable time exploring different vendors. Shalom wanted to figure out a solution to this problem, and they believed that the best solution was to create a digitized version of their form. This would allow for a more efficient registration process, allowing their patients to get more out of the Health Fair.

For this semester, our main objectives included creating this registration form website for health fair participants to fill out and linking it to an AWS back-end database. Furthermore, we also wanted to create a home page as well which would assist in offering paths in future implementations for options for not only vendors who attend the health fair, but also admins as well who may want to pull information from the database to gather these key statistics. It is important to note that the AWS database needs to be HIPAA certified since the information being stored is personal patient information.

## Motivation & Rationale

The creation and formation of this digitalized registration form and database assists Shalom heavily. Currently, all information and registration forms filled out during the annual health fair are completed and filed on paper making it difficult to gather these key statistics about patients who register and overall attendee information. Our digitalized version of this form can help achieve a more efficient registration process in which Shalom can more easily store information about health fair participants, and participants can enjoy more of what the fair has to offer instead of spending valuable time filling out forms. Furthermore, if future implementation and progress allows for vendor options, more valuable time can be saved if a participant’s information is universal across the vendors present.

## Customer/Developer Description

Shalom Health Care Center is a non-profit agency located on the west side of Indianapolis that provides quality health care that is patient-centered, affordable, and accessible in the community. Overall, they serve 14,000 patients annually, with over 3,000 individuals attending their annual health fair each year. The goal of this fair is to provide health screenings, community resources, back-to-school essentials, and more to those in need. Shalom has been partnering with Butler EPICS over the last two years, with the goal of completing projects before their 2025 Health Fair.

## Approach to the Problem

When beginning to tackle the problem at hand, we knew we wanted to create a registration form that was both function focused and stayed true to the paper forms Shalom has used previously for health fairs. This would allow us to stay consistent with their requirements and have a simplified design that is clear and concise and would not be confusing for participants who attend the fair. Before beginning any actual code, we began by creating a wire frame and template for the design for the homepage along with the registration form to decide exactly how we wanted everything to be laid out in the end.

The other problem we had to address is the back-end database. Shalom requested AWS for this, and we knew we would have to consider HIPAA compliance in its formation. This would be created simultaneously with the registration form, so from the start, both the front-end and back-end team would need to work together to complete our goals.

The other major obstacle and decision we had to address before even beginning our own work was to decide on how we were going to utilize or build off last semester’s progress on this same project. The semester before us had achieved creating a database on AWS, but the HTML form utilized was too basic for any further implementations or additions. Due to this, we decided to utilize REACT instead and create a new website that consisted of a home page in addition to the form. We met with the previous group and learned a great deal about the back end, so we were able to salvage and utilize their approach for the AWS database, implementing some new features due to the change in registration form. ￼

## Glossary

The following table is a glossary and compilation of various terms used throughout this report. Each term is followed by its definition and location in the report:

|  |  |  |
| --- | --- | --- |
| TERM | WHAT IT MEANS | WHERE USED |
| Component | A Front-end object that has its own functionality and style | Front-end Development, React-MUI-TypeScript |
| npm | Package manager, handles importing and use of additional APIs, and frameworks in use for project | Front-end Development, React-MUI-TypeScript |
| React | A JavaScript UI Library made by Facebook used for Web Development. | Front-end Development |
| MUI | Material UI, additional API library to make react components look \*pretty\* | Front-end Development |
| AWS-Amplify | Allows linkage between and AWS database and a React webpage | Front-end Development,  Back-end Development |
| Non-relational database | Database does not use traditional table based relational database structure allowing for more flexibility. | Back-end Development |
| Node.js | Extension of JavaScript, React, and Next.js | React-MUI-TypeScript, Front-End Development |
| DynamoDB | SQL Database on AWS | Back-End Development |
| API Gateway | Helps manage, publish, and maintain the API that accesses the data | Back-End Development |
| Console | Area behind-the-scenes of a browser, that is receiving the compiled code. | Front-End Development and Back-End Development |
| AWS- AppSync | Used to connect application to data with services like Graph QL | Back-End Development |
| Themes | (themes.ts) collection of Shalom’s Websites fonts and website colors that can be used and referenced throughout the code base | Front- End Development |
| Types | (types.ts) custom keywords that can be used to describe what is needed for specific components | Front-End Development |
| Mutations | Operations that modify the state of a database | Back-End Development |
| Schema | Blueprint/ framework that defines the structure of the data in a database | Back- End Development |

# **Chapter 2: Specification of Requirements**

## Functional Requirements

This project's functional requirements from Shalom Health Center can be separated into two main categories, the front and back end, as the client wanted the creation of a registration form and an AWS database.

For the front end, the requirements included:

* + A registration form for Health Fair attendees the enter information about themselves (demographic, health care provider, etc.)
  + The ability for those attendees to add additional information about adults accompanying them along with children
  + A simplistic design that is function allowing attendees to fill out the form without much confusion or difficulty
  + Language functionality for both English and Spanish
  + Additional options for vendors to add full form universality in the future

For the back end, the requirements included:

* + An AWS database to be able to store the information about the attendees for a moderate length of time rather than having to store physical records
  + The database must be HIPAA-Certified and secure as it is holding sensitive, patient-centered information
  + The creation of tables in the database for not only the main patients filling out the form, but also the accompanying adults and children
  + Full compatibility with the front-end form
  + The ability to pull information from the database to gather key statistics

## Non-Functional Requirements

For the non-functional requirements, there are a few crucial concepts that we really saw a need to address. For one, since we are dealing with patient information and the AWS database needed to be HIPAA-compliant, there was a great need for increased security. Many considerations went into this including only authenticated/authorized usage being allowed when pushing, pulling, and accessing the database. Besides this, scalability was another concern. This includes not only the database size, as we knew there could be upwards of 3,000 health fair attendees filling out the form, but also, with the ability to add future implementations. Shalom has requested that there are options for the vendors along with administrative tools, so we have chosen to design our front-end with these concepts in mind to be added in the future. Through React, additional implementations can be made easily and effectively. Lastly, we made usability our goal to create a registration form that is detailed in information, yet also quick and concise when filling out with limited confusion for the attendees who may or may not have technical experience.

## Assumptions & Constraints

There are many assumptions that are made when creating this project. For one, there is the concept of user behavior. Although we did extensive testing to ensure that the users filling out the form do not encounter errors, there are still many different moving components that need to be accounted for, especially with many users. We had to assume that users are filling out the information correctly with accurate information for themselves. In addition, we also had to make assumptions about the scalability of the project in the future along with future groups having the developmental resources and knowledge to build off from the point we have arrived at. A last assumption that was made was that the client would have the available technology and internet connectivity to be able to use this form and website and allow users to fill it out during the health fair.

When it comes to constraints, there were only two large obstacles to address. Time was a large factor, as our team was given only one semester to not only gain familiarity with last semester’s progress, but also add our own additions and get the project in a place that is ready for next semester to begin with. In addition, as a team we also lacked technological experience and knowledge with React, so the experience allowed for learning and insight.

# **Chapter 3: Architecture**

## Overview of High-Level System Architecture

## Decomposition & Style

The digitalized registration form created for Shalom is web-based with multiple pages including a home page, patient registration page, and placeholder pages for future admin and vendor functionality. When entering the website, the user has the option to choose and navigate between the different pages. In future implementations, there will be log in requirements for users who attempt to access the vendor and admin pages. As of now this webpage is locally hosted, but in the future, this webpage could be given to Shalom to host themselves.

On the back-end side of things, the database is a non-relational database including two tables: patient and children/additional adults. The information stored in these tables is pulled directly from user input from the registration form and includes the same details as the text boxes on the form.

## System Information Description

The registration form created uses the React JS library and imports from the MUI component library. For the back end, the database was formed through AWS using DynamoDB, AWS-App Sync, Lambda, and API Gateway, with the connection to the front end being established through AWS Amplify. More information about the specifics of the code's implementation can be found in this report's implementation section.

# **Chapter 4: Design**

## Description of User Interface

#### Home Page:

The primary home page of the Shalom Health Care Center Health Fair Website displays a large image of the Shalom Logo with helper text below instructing the user what role each person who is accessing the website identifies themselves as for the health fair. There are 3 circular buttons below the text for the user to select from: a Vendor Button, a Patient Button, and an Admin Button. Each one of those buttons will take the user to the correlated page (explanations below in other sections). At the bottom of the screen there are two buttons to toggle between English and Spanish. The home page is pictured below:

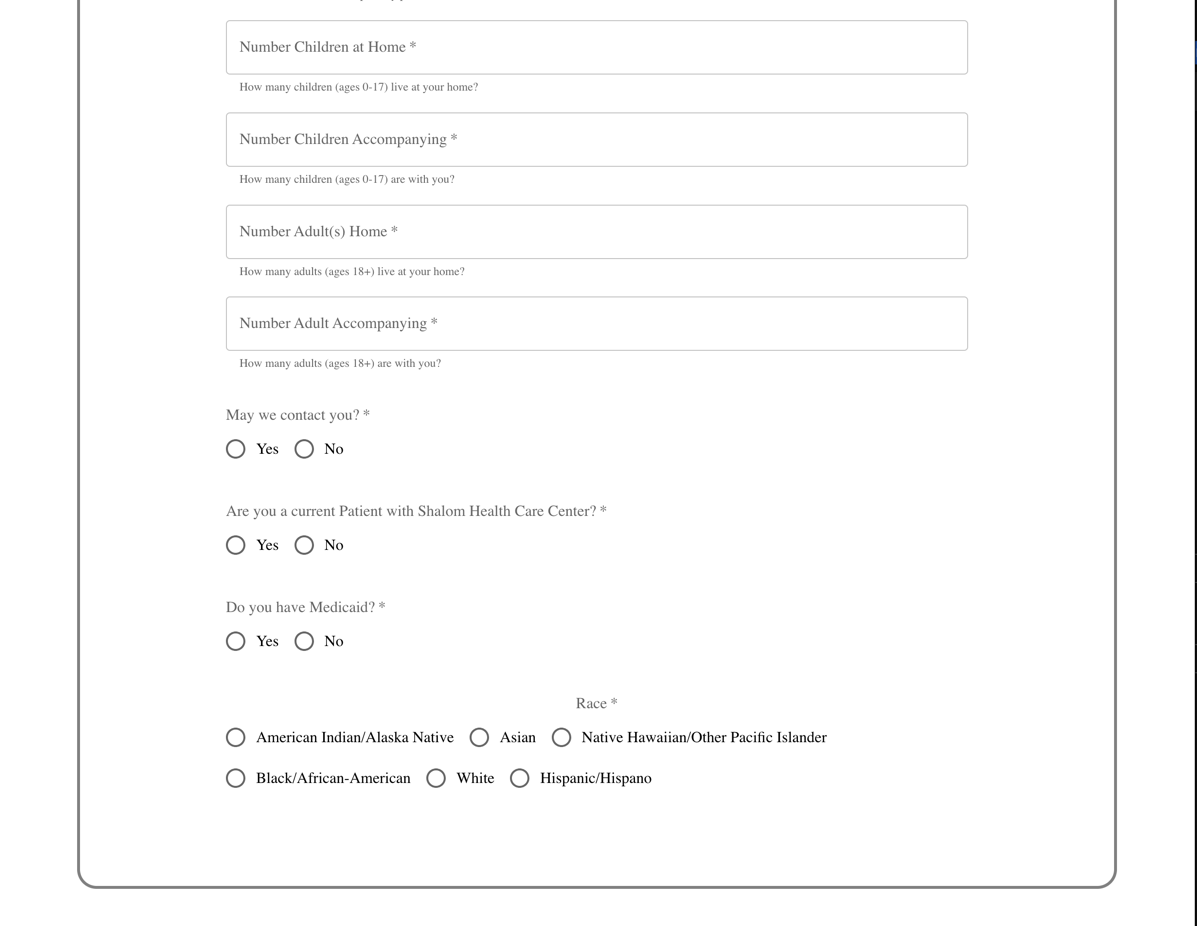
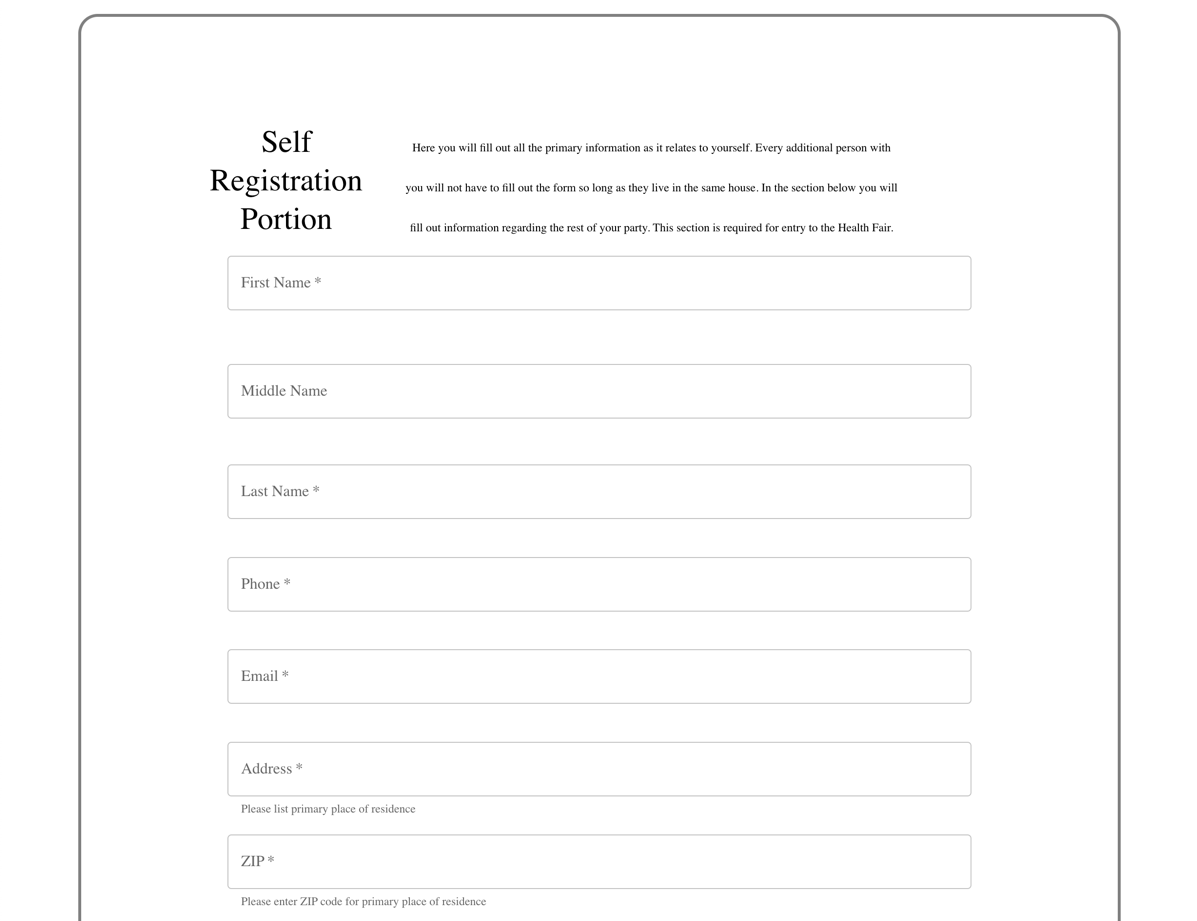


#### Patient Page:

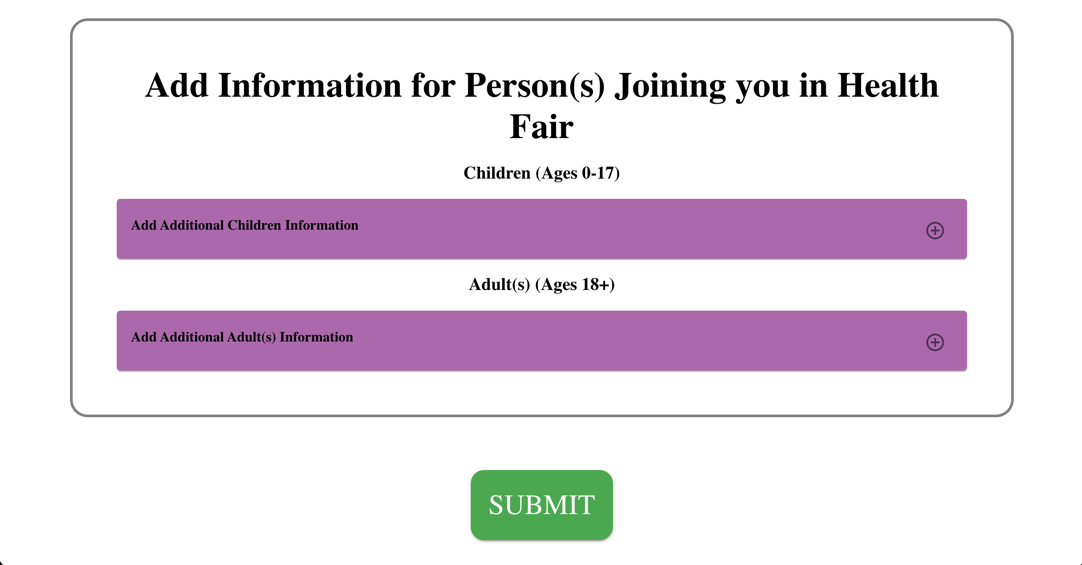
When a user selects “Patient” it routes them to the registration page for the health fair. There are 3 sections that have been broken up for the website layout. At the top is the logo again with some helper text describing this page's purpose. There is also no sign in required for this page so that it is accessible for everyone. In addition, a language toggle button can be seen below the description. The following image displays the patient page:



The next segment of the patient page contains the self-registration portion. This segment is contained in its own box with a heading and short descriptive text explaining the details of this section's purpose. Below the description, there are 7 text boxes, 6 of which are required. The text boxes are ordered by First Name, Middle Name, Last Name, Phone, Email, Address, and Zip with the only text box not required being Middle Name. In the 2nd image of the self-registration portion, there are 4 text boxes and 4 radio button sections. All these sections require input. The text boxes here only allow numbers to be entered and they include: “Number of Children at Home”, “Number Children Accompanying”, “Number Adult(s) at Home”, and “Number Adult Accompanying”. The Radio buttons are “May we contact you?”, “Are you a current Patient with Shalom Health Care Center”, “Do you have Medicaid?”, and “Race”. The self-registration portion of the form can be seen below:



The last section of the Patient page is the additional information for any other persons accompanying the primary person registering including a section for additional children and a section for additional adults. There is a plus sign on the right portion of each section that the user clicks to add information regarding the additional person(s) registering alongside the main attendee filling out the form. Both portions have text fields for full name, phone number, email, address, and zip code. The additional information for person(s) joining section of the form can be seen below:



#### Work-In-Progress Page:

Lastly, there is a work-in-progress page featuring the Shalom logo on the front of the page and text saying, “Construction in Progress!”. This page is taking the place for the admin and vendor pages due to lack of implementation.



#### Admin Page (Plans):

The admin page is currently not implemented but it will begin with a required log in prompt for the staff at Shalom. In addition, there will be required dual authorization to consistently comply with the HIPAA Regulations. In the admin page, once signed in, admins can look up a list of patients based on certain criteria with a dropdown menu of preset queries and a button to generate the list to allow for easy search and list generation. There will also be an area where they can search and pull up individual patients' information. Admins will be able to create accounts for vendors that will be part of that year's health fair along with having access to an interface to clear and delete all content in the database with additional confirmation to prevent accidental deletion.

#### Vendor Page (Plans):

The plans for the vendors page have less plans due to lack of information. The basis of this is that the vendors will have a dual authorization sign in like the admin and then be able to pull up specific information regarding specific patients. If HIPAA compliance allows, they will also be able to input specific information regarding the health information for their organization and the patient; however, it is currently not known if this is allowed.

# **Chapter 5: Implementation**

## Selection of Languages

Our project’s front end was created with React JS which consists of JavaScript, HTML/CSS, and TypeScript. React is a professional Web Development tool that is significantly effective if used correctly, and the UI has the potential to look much more appealing than a simple HTML webpage. We chose React not only because it would make our webpage look more professional overall for the client and the attendees of the health fair, but also because it has so much to offer in terms of making a fully functional site with multiple pages.

The back-end database used to store the information from the form was created using AWS. AWS has many tools that assist with the construction of a database that were utilized such as DynamoDB, AWS-AppSync and API Gateway. AWS Amplify is a framework utilized to allow the linkage between the Shalom registration form and the AWS backend database.

## Coding Standards & Comments

Besides due to its ability to make the form look professional, another reason why we had Selected React JS and Typescript as our primary language was to accommodate proper coding practices and increase code reusability. When increasing the usability of the code, it increases overall readability by having custom naming practices for components/ methods and to expand explanations and documentation of how the system operates.

The main process for how we went about our programming was to begin by creating custom React components utilizing the MUI library to allow for specific visualization of items depending on their purpose and functionality. Then another folder was created that was labeled as pages to hold various files of the pages of the website. Whenever there was a different aspect of either functionality, navigation, layout, or structure added to the web application, we created a new folder in the proper scope and added separate files to allow for further documentation of each element.

Each component was labeled by a name to easily identify its purpose in our application. There are 4 primary elements that went into each component; (a) a folder to hold the remaining 3 files, (b) a React-TypeScript file, (c) a TypeScript index file, and (d) a Jest React-TypeScript testing file. The titling of elements (a), (b), and (d) all held the name of the component, whereas the title of every (c) file was titled index.ts for all components. Files (b), (c), and (d) were all contained in their own folder, (a), which allows for each components index, file (c), to hold the same name. This file was to export each component within its folder (a) that is in an inner scope of the whole overarching project to be accessed everywhere else. File (d) can be identified as having an extension of “.test.tsx” postfixing the name, and lastly file (b) has an extension of “.tsx” postfixing the name.

Our documentation of our code extends to comments for each of the components. Some components do not have complete documentation currently due to some time constraints. The components that do follow the structure of a brief description of what the component is supposed to do and description of all parameters and valid input types.

## Implementation & Distribution of Work

Rachel created the React Application and the necessary components, while Miles formed the Home Page and implemented the Icon Buttons that Rachel created into the page. After that, Rachel created the layout of the Patient Registration Form as it is, along with creating a Vendor Page and Admin Page for upcoming groups to implement. She had the submit button on the registration form print all the inputs into the console as a temporary function until Matt and Reece were able to connect the Back end, and re-direct those inputs into the database. While that was happening, Miles implemented the Language Button into the different pages, and created functionality to switch the text between English and Spanish. Rachel created the functionality to route to the different pages from the click of a button. Finally, Miles and Rachel both took on the error-handling to prevent the application from being able to crash.

On the back-end side of things, Matt was the lead developer working with the database. The AWS database was formed by Reece with help from the previous semester’s group, and Matt worked on creating the schema and tables for the patients who register on the form and the additional children/adults accompanying through DyanmoDB. A lot of time was spent understanding the various tools and connections across AWS to form the database such as working with App-Sync and API Gateway. Matt and Reece both worked towards establishing the connection with the database and form through the utilization of AWS Amplify.

## Organization of the Code Base

When creating a React application there is a preset organization of the code base that auto generates for the programmer, making it easier to navigate and be able to structurally organize as you proceed in development. Below is an image with numbers besides different folders and tabs to explain the organization of our code base. The image is a screen shot Visual Studio Code file organization tab.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | 1 | Folder to hold any additional images/ assets developers wish to use for website | | 2 | React auto generated file, all additional API packages being used in project, can ignore | | 3 | React auto generated-- ignore | | 4 | Where all the stuff happens, primary place where development happens | | 5 | All component folders and files are located for the patient registration form and home page | | 6 | Example component folder, inside is where all component related files can be found | | 7 | File where all of Shalom’s color codes are found and Shalom fonts | | 8 | Same type of thing as number 5 but holds all components for admin page | | 9 | Additional documentation files, currently is empty | | 10 | Folder that holds all website page folders and files | | 11 | Example of a file of a website page | | 12 | Folder holding routing file that allows website navigation | | 13 | File holding routing address for page navigation throughout website | | 14 | The file that runs when running/starting the application | | 15 | Shalom logo image | | 16 | Custom variable types that can be used in the project | | 17 | File that holds dependencies for projects and versions – DO NOT EDIT, reference only | | 18 | Where you can write scripts for testing, debugging or other development, also track authors of project | | 19 | Internal Readme file | | 20 | GitHub Readme file | |

# **Chapter 6: Quality Assurance & Testing**

## Selection of Testing Objectives

To decide what our objectives were for testing, we had to think as if we were an end-user normally filling out the Registration Form. What are the possible issues that we could accidentally cause that would break the program? Since we knew that the program worked efficiently with inputs we expected, how would the program react if we decided to input values the programmer would not expect? This led us to make our objective to essentially search for failures within the program.

## Testing Approach & Methods

Many types of testing are available for QA Testing, but we decided to take a Unit Testing approach to the program. This includes creating test cases and documenting each test case while also marking down whether the case failed or not. In addition, another form of testing utilized in this project that is only specific to React is “jest” testing. You can test for expected outputs of functions with jest, or you can test whether a certain React component renders perfectly. Since we already had tested for expected outcomes in functions. In the end, we only used jest to test the rendering of components like the CircleIconButtons, the Language Buttons, etc.

## Defect Detection & Management

To neatly organize our test cases, we made a table with 7 columns (ID, Purpose, Precondition, Input, Expected Output, Output, and Pass/Fail) to document these cases in. Here is a sample size of four test cases used in our testing:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Purpose** | **Precondition** | **Input** | **Expected**  **Output** | **Output** | **Pass/Fail** |
| 1 | Ensure we cannot input special characters or letters in certain text fields | “How Many Children Are with You Today?” | Four! | Error Message | Submits Normally | Fail |
| 2 | Submit a blank form | Registration Form | Click Submit Blankly | Error Message | Error Message Displays | Pass |
| 3 | Add past the maximum number of children (8) | The Add/Drop Accordion of ChildAccs | Click “+” 9 times | Doesn’t go past 8 | Doesn’t go past 8 | Pass |
| 4 | Going to a random web path | Home Page is open | “/fjhedoiufeoirfj” in search bar | Our 404-page returns | Our 404-page pops up | Pass |

Testing is essential for a program because programmers usually only test inputs expected to be inputted. Testers try to break the program, and if they can, they will report it back to the programmers so they can isolate the error and implement error-handling for that specific problem. For Jest Testing, we tested the rendering of components, and those tests passed.

# **Chapter 7: Project Organization and Management**

## Roles & Contribution

### Roles & Their Propose

|  |  |
| --- | --- |
| Client Liaison & Documentation | The Client Liaison keeps the client informed of the team's progress while setting up meetings between the group and client. The documentation process included keeping track of daily accomplishments and ensuring the Scrum Master had everything they needed to fill out the WSR. |
| Front-End Developer | The Front-End Developer oversees creating the visual and interactable components for the website. They work with ensuring that the data moving around on the website is connected properly and accessed so that the backend can connect areas appropriately. Back or front-end can deal with encryption of data through actual website (front end). |
| Back-End Developer | The Back-End Developer creates and works with the code behind the scenes, more specifically forming the database and tables necessary to store and pull information. Back-end developers can help with data encryption and link the database to the front-end. |
| QA (Quality Assurance) & Tester | The QA Tester ensures that there are no cracks in the programs, and that the program is ultimately functioning as both the end user and programmer intends for it to. If an error is found, the tester will either report it to the programmer, or fix the error themself. |
| Scrum Master/Team Leader | The Scrum Master or team leader ensures that the team is on the right path towards the client’s requirements for the project. This includes forming goals for the team, leading the daily stand-ups, assisting with both the front and back-end work, and managing the group through ensuring completion of task and duties. |

### Team Member & Their Contributions

#### Reece Parks

As team lead, my responsibilities included managing the entire group, ensuring that each team member completed their assigned tasks and duties, and helping when needed on both the front and back end. In addition, I also created Weekly Status Reports for each week and helped guide the team towards forming and accomplishing goals for each sprint. I spent most of my technical time assisting Matt on the back end, more specifically linking the front end React form to the database through AWS Amplify.

#### Matt Fleschner

I was the primary back-end developer for the team and spent a lot of the time focusing on the AWS back-end database. This included forming the schema and creating tables for the patients registering and accompanying adults/children. I spent a lot of time working with the various AWS tools such as DynamoDB, API Gateway, and AWS App-Sync. In addition, with the assistance of Reece, we established the connection with the front-end form by utilizing the AWS Amplify framework.

#### Rachel Maier

I was the primary front-end developer on the team. I had managed to accomplish creating the React application, installing front-end packages (such as APIs and Libraries), and added documentation for how other team members and future groups install proper packages to get the application running on their own computer. I also created components for the registration form and ensured that the dynamically allocated variables held proper unique identification values and their contents were stored as they were entered. I was primarily responsible for the setup of the application and development for how the website looked and behaved on the surface. I assisted with finding a way to dynamically allocate variables so that the back end could access the values from the webpage and creating some baseline tests with Miles.

#### Miles Mann

I was the project's QA Tester and secondary front-end developer. My role consisted of mainly helping out Rachel where I was able to with the implementation of React Components made by Rachel. After we finalized the input fields for our project, I started testing them so we could isolate any errors we may have caused during production of our product. I was able to find some errors we needed to fix, and I started working on solutions to those errors. This led me to implement our error-handling based on the failed test cases I found and the specifics our client asked for. Mainly, my job was to just implement functionality for the components that Rachel made, and make sure they were working properly.

#### Hallie Nuccio

I was the client liaison and in charge of documentation for this team. This meant that I oversaw contacting the client with any updates as well as working with them when changes to our project needed to be made. I made sure to send them emails whenever large accomplishments were made, including the first draft of the online registration form and the connection of the front-end and back-end. I also set up Google Drive which we used to keep track of all the project documents, including information from the client, presentations we made, and files we created.

## Management Process

There is a lot that went into our management process this semester. The team’s primary form of communication was through a text group chat in which everyone would update with their progress, ask questions, or request assistance if needed. Every Tuesday and Thursday, the team would also meet in person during the scheduled class time to share ideas and updates on goals. Tuesdays would usually be dedicated to discussing the Weekly Status Report while the focus of Thursday would be the daily stand up. The stand up consisted of the team discussing what they are currently working on, what issues they might be having, and what they would be working on next. There were many instances in which the team would meet outside of the designated class time as well, especially when it came to being fully prepared for weekly sprints.

Another useful tool to keep them on track this semester was the Trello Board. Each sprint, we would take off the most critical goals from the product backlog and assign them as goals for the sprint. When goals were completed, they would be marked as accomplishments. This was a very effective way to distribute the work and gave everyone the opportunity to always know what goals needed to be addressed and worked on. Our shared Google Drive created and updated by Hallie each week was also critical in staying organized and keeping all our progress documented and in order.

The last and most important part of the management process was how we engaged with the client. Hallie was the main form of contact with Shalom, and she would send frequent updates regarding our progress on the project along with notes that were taken during our meetings. She would also set up virtual meetings with Shalom as well. These meetings were usually bi-weekly during class time, and although they were not long, they were very crucial in receiving updates and feedback from the client regarding the registration form and database.

## Weekly Status Reports

Each week, Reece would fill out a Weekly Status Report about the accomplishments and important information from the week before. These reports would include possible red flags, or obstacles that are significantly hindering or halting the developmental process along with smaller issues/problems as well. Luckily, throughout the semester, the team did not deal with any substantial red flags; however, there would be small obstacles to overcome each week like setting up React, database authentication, and linking the front and back end.

Besides this, the weekly status reports also outline the accomplishments from the week before. This can include anything such as updates on the front and back end, completion toward sprint deliverables, client updates and scheduled meetings, and just overall goal completion. Usually, these accomplishments would be separated by the date of completion.

Finally, the last part of the weekly reports included goals for the upcoming week. Again, these could be major developmental goals or more non-functional duties for the week ahead such as sprint preparation. Every Tuesday, the team would discuss and go over these reports to gain insight on the current velocity of the project and get an overview of where the project is heading next. The full list of Weekly Status Reports for each week can be found on GitHub as linked in.

## Detailed User Manual

The following are detailed instructions on how to operate and set up the React application. Please refer to the readme file in the GitHub repository for further instruction.

#### Set Up:

- Install node and npm on your system (Look up online docs how to do this as it can

take place in your terminal on your computer).

- Install [reacts dev tools] (https://react.dev/learn/react-developer-tools), in CLI "npm install -g react-dev tools".

- In the package.json file, under author, add your year to the Butler Epics Team.

#### Handling the Repository:

Not all the files will be used, as some are automatically generated from the usage of React, jest, and mui. You will edit the version number every time you update the code base and publish it to the world. You can identify scripts you can run for this application in "scripts" in the package.json file. For the most part, you will not alter many things in this file.

Here is a list of File you will generally use or find helpful to look at:

- /src/types.ts -- this file defines custom types

- /src/components -- folder, this has the custom components for the project's patient page

- /src/documentation -- folder, this has documentation of the components from components folder

- /src/components/themes.ts -- this illustrates a consistent theme for the project

- /src/components/index.ts -- this is the custom calls for components in the project

- /src/App.js - this is the main page that connects to the web

- assets folder - this holds all the assets for the projects (imgs, ico, svg etc.)

- package.json - this is where you update version number on each segment going forward

- /src/App.test.js - this is the tests for the overall application, it uses jest

- /src/Pages - folder, holds all the different website pages and related react typescript files

- /src/components\_admin - folder, holds folders for admin page components, these components have not been created yet

- /src/Routes/Routes.tsx - react typescript file, holds the routing addresses for the page navigation throughout website

This is a list of primary dependencies you will be using for this code base project:

- React/Next.js - framework

- jest - testing for React

- MUI - pre-existing component library for React

-Amplify – linking the form to the database backend

#### Running the Application:

After you have set up npm, node, and next (if on Windows), clone the repo and cd into the shalom folder "cd shalom". After that, run ***npm -i*** to install all the dependencies followed by npm audit fix (do NOT do the --force). Lastly, type ***npm start*** to run the application. More info can be found here: https://www.geeksforgeeks.org/how-to-download-a-react-project-from-github-and-run-in-my-pc/

**\*Make sure after you have installed node and npm on your system you type in *npm -i* in the terminal line, and it will install all related packages and dependencies related to this project**\*

\**If you run into an error that looks like this:*

*npm ERR! code ERESOLVE*

*npm ERR! ERESOLVE unable to resolve dependency tree*

*npm ERR!*

*npm ERR! While resolving: shalom@0.1.0*

*npm ERR! Found: jest@27.5.1*

*npm ERR! node\_modules/jest*

*npm ERR! dev jest@"^27.5.1" from the root project*

*npm ERR!*

*npm ERR! Could not* r*esolv*e *dependency:*

*npm ERR! peer jest@"^29.0.0" from ts-jest@29.1.2*

*npm ERR! node\_modules/ts-jest*

*npm ERR! dev ts-jest@"\*" from the root project*\*

Enter the following line: "npm config set legacy-peer-deps true"

#### How to Run the Scripts:

In the terminal, type npm followed by the name of the script. This can be found in the "scripts" section of package.json.

#### How to not “BREAK” the Repository:

The main branch may be locked -- if it isn't, following these steps will help not to break the code/repo. There is an easy way to update the repo and not break the main working branch.

Steps:

1. Go to the issues tab on the GitHub repository page.

2. Click "new issue" to create a new issue and enter in the "Title" space the name of the thing that you are planning to work on -- this will be the title of the new branch so keep it very short and simple.

3. You can leave the description blank and assign the issue to yourself or who is supposed to work on it. You can add a label if you wish.

4. Hit "Submit new issue" and it will redirect you to a new page.

5. On the new page on the right side, you will see "Development". Under that, there is a link to "Create a branch" click it and create the branch

6. Now go to your text edit/terminal/putty and type in the prompts listed and refresh your coding IDE. You will be switched to the proper branch, and you can start editing.

7. Once you are done and you have finalized the contents of the issue, merge it with the main/master branch. (there is easy way to do this on the GitHub repo page)

8. Once successfully merged close the branch and issue, then update and sync main branch in IDE.

#### Helpful Information and Links

For help keeping in line with the national the color guidelines to ensure accessible visual color contrast codes click here (https://webaim.org/resources/contrastchecker/)

You may contact anyone on the team if you need additional assistance.

# **Chapter 8: Future Work**

## Explanation of Next Steps

There are areas of the Registration/Patient Page that still need to be finalized below is a list of what needs to be accomplished in future work:

1. Check box at the end of form (above submit button) with legal jargon authorizing the patients to have this information shared with the vendors and Shalom that are involved in the health fair.
2. Front-end encryption & decryption of the variable storage from registration form to backend database.
3. QR code generated after patient registers to give to vendors to have access to basic registration information or some other way to grant vendors easy access to the registration information.
4. Complete translation of components in registration forms.
5. Link the values from registration to the back-end database storage.

Other areas that need to be completed:

1. Dual Authorization for admin/vendor sign in.
2. Area for admin/ vendor to search and generate information on patients that have registered for the health fair.
3. Area for admin to register/create accounts for vendors.
4. Get list of Queries from Shalom that they would like to search by.
5. Other things related to items in above numbers.

## Implementation Needs

Primary functionality needed is the admin sign in and page to search for queries. The areas in the section above are the primary things that need to be completed. Next group should focus on numbers 2, 5, and 1 from the first part, and numbers 1, 4, and 2 (admin first) from the second part.

# **References & Bibliography**

Include all references and/or websites you used.

#### Websites:

MUI Component Docs:

<https://mui.com/material-ui/all-components/>

Npm Installation guide:

<https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

React project setup guide:

<https://www.geeksforgeeks.org/how-to-download-a-react-project-from-github-and-run-in-my-pc/>

AWS Amplify Docs:

<https://aws.amazon.com/amplify/>

Getting familiar with React:

<https://react.dev/learn>

Jest Testing Docs:

<https://jestjs.io/docs/getting-started>

Accessibility Color Contrast Checker (AA standard):

<https://webaim.org/resources/contrastchecker/>

# **Appendices**

The source code can be found in the GitHub repository. Our sprints, weekly status reports, and other information from Google Drive can be found in the GitHub repository. The client evaluation can be found below. All other information not included in the GitHub can be given after contact with group members if needed.

#### Client Evaluation

A paper with blue writing on it

Description automatically generated