# Final Report: [DENTISTRY] Data Mgmt for Sealants Outreach Program

## Summary:

Our app is a data management system designed for use by Texas A&M School of Dentistry to streamline their data collection and entry processes for their Sealants Outreach Program. Our clients wanted an app that could replace their paper-based forms and provide useful overview statistics of collected data that could be used to assess the progress, costs, etc. of the program.

This was not a legacy project, so all current functionality was implemented by our team. Included in the app are a data entry form for specific events, or instances of the program visiting specific schools, as well as forms for patient details, which are divided into personal data, screening, preventive service, and follow up sections. The data entered into these forms is then stored in tables. Specific records can then be easily found using the added search function, which allows the user to search for a record by date or school name in the case of events, or id or school name in the case of patients.These features were added to address the need for a digital form entry process. Additionally, features were added to compile the data entered to produce summary statistics. Three pages were added to show tables and charts with information such as how many children at a specific school received sealants, or how many hours a dentist spent at a specific school. These features were added to address the need for overview statistics to assess the program

.

## Description of User Stories:

**Event Details Page**

- Event Data Collection Form - Add Event Details
    - Points: 4, Status: Complete
    - Created page where event details can be added
- Event Data Collection Form - Event Details
    - Points: 4, Status: Complete
    - Created page where event details can be viewed
- Building Cucumber tests cases for Event data collection form
    - Points: 3, Status: Complete

    - Added initial Cucumber tests for form functionality
- Basic Rspec tests for Event data collection form
    - Points: 3, Status: Complete
    - Added initial Rspec tests for form functionality
- Database connection for Event data collection form:
    - Points: 4, Status: Complete
    - Data entered into the event data collection form updates the database
- Added a View, Styling and Frontend for List of existing Events Details
    - Points: 4, Status: Complete
    - Create a display with styling to show event details
- Added Rspec tests for Event_Details Views
    - Points: 4, Status: Complete
    - Added tests for event_details view
- Added Rspec for Controller of Event_Details
    - Points: 4, Status: Complete
    - Added tests for event details controller
- Modified Event_Details views
    - Points: 4, Status: Complete
    - Format changes made to event details form and display
- Cucumber tests for event details form
    - Points: 4, Status: Complete
    - Added cucumber tests for adding event details
- Cucumber tests for event details display
    - Points: 4, Status: Complete
    - Added cucumber tests for displaying event details

**Patient Details Page**

- Child-Level Data Form - Patient Details
    - Points: 4, Status: Complete
    - Created page where patient details can be viewed
- Styling and Frontend for show Patient Details
    - Points: 4, Status: Complete
    - Apply Styling to Patient Details Page (home)
- Database connection for Screening Section
    - Points: 4, Status: Complete
    - Data entered into screening form updates the database
- Styling and Frontend for show Screening Section
    - Points: 4, Status: Complete
    - Apply Styling to Screening Section

- Database Connection for Preventive Services
    - Points: 4, Status: Complete
    - Data entered into preventive service form updates the database
- Styling and Frontend for Preventive Services
    - Points: 4, Status: Complete
    - Apply Styling to preventive service section
- Child-Level Data Form - FollowUp Section _ Initial Version
    - Points: 4, Status: Complete
    - Create Follow Up Section for patient details
- Styling and Frontend for Follow-up
    - Points: 4, Status: Complete
    - Apply styling to follow up section
- Styling Fixes for Child Level Data Form - Screening
    - Points: 4, Status: Complete
    - Fixed some errors in styling
- Styling Fixes for Child Level Data Form - Preventive
    - Points: 4, Status: Complete
    - Fixed some errors in styling
- Styling Fixes for Child Level Data Form - Follow Up
    - Points: 4, Status: Complete
    - Fixed some errors in styling
- Added Rspec tests for Patient_Details Views
    - Points: 4, Status: Complete
    - Added Rspec tests for patient details view
- Added Rspec for Controller of Patient_Details
    - Points: 4, Status: Complete
    - Added Rspec tests for patient details controller
- Edited Styling and Added logic for unique Patient ID on the patient_details
    - Points: 4, Status: Complete
    - Logic added to uniquely identify patients by patient ID, styling adjusted to accommodate change
- Add PID, PatientID and SchoolName related Logic
    - Points: 4, Status: Complete
    - Logic added to generate at PID based on patient ID and SchoolName
- Cucumber tests for patient details display
    - Points: 4, Status: Complete
    - Cucumber tests added to test patient display
- Creating Rspec Tests for Screening and Preventative Functionality
    - Points: 4, Status: Complete

    - Added Rspec tests to test screening and preventive service forms
- Creating Rspec Tests for Styling Fixes in Follow-Up Child Level Data Form
    - Points: 4, Status: Complete
    - Added Rspec tests for follow up form
- Reconfigure patient details paths and logic to use PID rather than ruby default id
    - Points:4, Status: Complete
    - Any logic or paths that identified a patient entry by Patient ID was changed to instead use PID
- Re-Establishing Database Connection for Screening and Preventative View
    - Points: 4, Status: Complete
    - After changes were made to the screening and preventive view functionality, ensures data from the form properly updates the database and data from the database is properly displayed on the form
- Re-Establishing Database Connection for Follow-Up View
    - Points: 4, Status: Complete
    - After changes were made to the follow up view functionality, ensures data from the form properly updates the database and data from the database is properly displayed on the form
- Creating a Cucumber Test for the Preventive Services Section
    - Points: 4, Status: Complete
    - After changes were made to functionality, new tests needed to be created for the preventive services section

**Add Patient Page**

- Child-Level Data Form - Add Patient Details
    - Points: 4, Status: Complete
    - Created page where patient details can be added
- Styling and Frontend for Add Patient Details
    - Points: 4, Status: Complete
    - Apply styling to Add Patient Page
- Cucumber tests for patient details form
    - Added cucumber tests to test add patient page

**Statistics Page**

- Create Statistics Page for Patient Data
    - Points: 4, Status: Complete
    - Create new page to show summary statistics

- Create a School_Details view for Statistics page
    - Points: 4, Status: Complete
    - Create a new page to show school summary statistics
- Create an Event_details Statistics Page with visualizations for the data
    - Points: 4, Status: Complete
    - Create a new page for event statistics
- Establish database connection for Patient Details Statistics View
    - Points: 4, Status: Complete
    - Data from the database is accessible and properly displayed in patient details stat view
- Establish Database connection for Event Details Statistics View
    - Points: 4, Status: Complete
    - Data from the database is accessible and properly displayed in event details stat view
- Establish database connection for School Details Statistics View
    - Points: 4, Status: Complete
    - Data from the database is accessible and properly displayed in school details stat view
- Fix issues with statistics pages while deploying in heroku
    - Points: 4, Status: Complete
    - Found and solved issues that was causing stats pages to not properly function on heroku

**Miscellaneous**

- Deployment
    - Points: 2, Status: Complete
    - Deploy app for the first time on Heroku
- User Stories
    - Points: 5, Status: Complete
    - Create initial user stories for the project
- Storyboard
    - Points: 4, Status: Complete
    - Create storyboards to map out expected behavior for pages
- Add CI Action to GitHub to allow automatic testing
    - Points: 4, Status: Complete
- Styling and Frontend, design
    - Points: 4, Status: Complete
    - Design an overall look for the application
- Creating Cucumber Test for Sessions

    - Points: 4, Status: Complete
- Deployed the app in heroku for verifying proper documentation
    - Points: 4, Status: Complete
    - Tested deployment instructions located in the GitHub to ensure they were accurate and complete
- Added Rspec tests for Whitelisting, Sessions, Excel Export and Statistics
    - Points: 4, Status: Complete
    - Added more Rspec tests to increase code coverage
- Make Rspec tests for mailers, channels, jobs, and child level details
    - Points: 4, Status: Complete
    - Added more Rspec tests to increase code coverage
- Refactor Code for Improved Coverage and Ruby Standards Compliance
    - Points: 4, Status: Complete
    - Added more Rspec tests to increase code coverage
- Development of Comprehensive Project Deployment Guidelines
    - Points: 4, Status: Complete
    - Added more Rspec tests to increase code coverage

**Unfinished**

- Organize school stats chart by school district
    - Points: 4, Status: Incomplete
    - Add functionality to charts that group statistics based on school district too rather than just individual schools
- School District Field and search function
    - Points: 4, Status: Incomplete
    - Add school district to add event form and add functionality to allow for records to be searched for by school district

## Legacy Project Considerations:

This was not a legacy project, so there was no process used to understand or refactor pre-existing code.

## Team Roles:

The roles of product owner and scrum master were consistent throughout the entire project.

Jaydeep Radadiya (734004131) : Product Owner
Sarah Gullion (530003430): Scrum Master
Manisha Panda (334003453)
Apurva Mandalika (334003575)
Jay Nehete (734004130)
Sushant Shelar (733001479)

## What was Accomplished in Each Iteration:

Iteration 0:
- 0 points
- Create Initial Mock-Ups for App
- Attend Initial Client Meeting
    - Determine Client's requirements for the project
- Create Initial User Stories
- Set Up Repo and Project Management Page

Iteration 1:
- 76 Points
- App is first deployed to Heroku
- Event Details and Patient Details, and Add Patient pages are created
- Initial Cucumber and Rspec Tests are Created

Iteration 2:
- 48 Points
- Functionality was added to Event Details, Patient Details, and Add Patient pages
- Screening, Preventive Service, and Follow Up forms are created
- Styling added to improve app appearance

Iteration 3:
- 72 Points
- Some issues with the styling were fixed
- Statistics Page for patient data and school data were created
- Google OAuth was added
- Code coverage was expanded for areas previously missed

Iteration 4:
- 48 Points
- Statistics page for events was created
- Functionality was added to all Statistics pages
- Screening, Preventive Service, and Follow Up forms are consolidated into a single page

Iteration 5:

- 24 Points
- Patient Statistics page was fixed
- Reached code coverage goal

## Points Per Member:

| | Iteration 0 | Iteration 1* | Iteration 2 | Iteration 3 | Iteration 4 | Iteration 5 | Post Iteration 5 | Total |
|---|---|---|---|---|---|---|---|---|
| Jaydeep Radadiya | 0 | 15 | 8 | 12 | 8 | 4 | 0 | 47 |
| Sarah Gullion | 0 | 11 | 8 | 12 | 8 | 4 | 4 | 47 |
| Manisha Panda | 0 | 14 | 8 | 12 | 8 | 4 | 0 | 46 |
| Apurva Mandalika | 0 | 14 | 8 | 12 | 8 | 4 | 0 | 46 |
| Jay Nehete | 0 | 14 | 8 | 12 | 8 | 4 | 0 | 46 |
| Sushant Shelar | 0 | 8 | 8 | 12 | 8 | 4 | 8 | 48 |
| Total | 0 | 76 | 48 | 72 | 48 | 24 | 12 | 280 |

*At this point in the project, we had not yet settled on how to count points for stories with two people assigned to them. In Iteration 1, we chose to give the full amount of points for the story to both people, which essentially doubled the point value of the story, resulting in an overinflated point total. This approach was not used in any further iterations.

## Customer Meetings:

Iteration 0:

- During this iteration, we met with our clients on:
  - 9/1/2023, 1:30pm
  - 9/6/2023, 1pm
  - 9/12/2023, 12pm

- During these meetings we introduced ourselves to the clients, asked questions about their intended use for their desired app, learned the features they expected to see, and obtained sample data that the app would be expected to take in and store.

Iteration 1:

- Due to scheduling conflicts with our clients, we were unable to hold meetings with them during this iteration.

Iteration 2:

- During this iteration, we met with our clients on:
    - 10/3/23, 12 pm
- During this meeting, we demoed our events details and patient details entry forms, as well as the patient details display. We received feedback about changes our clients wanted about what information about the patient should be entered and what data was most important to them to be displayed about the patient.

Iteration 3:

- During this iteration, we met with our clients on:
    - 10/18/23, 1 pm
    - 10/25/23, 1 pm
- During the 10/18/23 meeting, we demoed our Google login feature and our improved patient detail display format. We received feedback from our clients and they also requested that we create a new page in our app to show overall statistics for individual schools.
- During the 10/25/23 meeting, we demoed our improved screening form and auto-population features.We received feedback on what additional information the clients wanted auto-populated in the future as well as feedback on how the clients wanted us to merge our screening and preventive service forms in order to streamline the process of filling them out.

Iteration 4:

- During this iteration, we met with our clients on:
    - 11/06/23, 1 pm
    - 11/08/23, 1 pm

- During the 11/06/23 meeting, we demoed the app's updated appearance, the addition of search bars to data tables so that users could search for specific patients or events, and the export feature for patients and events that allows users to export the data in our app to an excel file. One of our clients was unexpectedly unable to attend this meeting.

- During the 11/08/23 meeting, we demoed the features from the last meeting for the client that had missed them, as well as the new Statistics Page.

Iteration 5:

- During this iteration, we met with our clients on:
  - 11/15/23, 1 pm

- During this meeting, we demoed all of the app's completed aspects to demonstrate its overall operation. We also discussed the process of transitioning the project in the future.

## Recommendations for Future Development:

For future development, our team recommends that more work be done on the statistics pages. More features can be added such as fore-casting, organizing data by gender, organizing data by school districts, etc. Additionally, more comprehensive and detailed statistical information can be added to these pages.

## BDD/TDD process:

Our team followed a Behavior-Driven Development (BDD) and Test-Driven Development (TDD) process to enhance software quality and collaboration.
When creating new features, we defined expected behavior using natural language and wrote tests based on that expected behavior. Happy paths (passing cases) and sad paths (failing cases) were both integrated into our tests. One problem that arose from this approach was that, when major changes were made to our application, this often caused many test cases to fail, resulting in time needing to be devoted to adapting them to once again match the application's behaviors. For example, tests written to test the Screening, Preventive Service, and Follow Up forms when they were different pages all had to be adapted after they were merged into a single form. Such instances caused our code coverage to be relatively low for the majority of the project. This issue was fixed in our final iteration.

Cucumber Tests:

```
23 scenarios (23 passed)
210 steps (210 passed)
0m14.363s

Share your Cucumber Report with your team at https://reports.cucumber.io

Command line option:    --publish
Environment variable:   CUCUMBER_PUBLISH_ENABLED=true
cucumber.yml:           default: --publish

More information at https://cucumber.io/docs/cucumber/environment-variables/

To disable this message, specify CUCUMBER_PUBLISH_QUIET=true or use the
--publish-quiet option. You can also add this to your cucumber.yml:
default: --publish-quiet

Coverage report generated for Cucumber Features, RSpec to C:/Users/Sgull/OneDrive/Desktop/CSCE606/DentistryProjectV2/Seal
ant-On-Rails-V2/coverage. 319 / 349 LOC (91.4%) covered.
```

Rspec Tests:

```
PS C:\Users\Sgull\OneDrive\Desktop\CSCE606\DentistryProjectV2\Sealant-On-Rails-V2> rspec
................................................................................

Finished in 14.38 seconds (files took 16.98 seconds to load)
80 examples, 0 failures

Coverage report generated for Cucumber Features, RSpec to C:/Users/Sgull/OneDrive/Desktop/CSCE606/DentistryProjectV2/Seal
ant-On-Rails-V2/coverage. 319 / 349 LOC (91.4%) covered.
```

## Configuration Management:

Everyone worked within the same GitHub repo, with development occurring in different branches and six branches total. Before any code from a branch could be merged into the main branch, it had to be reviewed and approved by two team members. As merges to main occurred fairly often, we did not immediately push the code to Heroku after a merge. Instead, we pushed to Heroku on a periodic basis.

## Issues in Production Release to Heroku:

There are several setup steps that need to be followed before the application can be successfully deployed to Heroku.These steps mainly pertain to setup that allows the Google OAuth to function. Only once these instructions have been completed can the app be properly deployed to Heroku.

## Issues with  AWS CLOUD9, GitHub, or other Tools:

AWS Cloud9 was not used for this project. Github was used. The only issue we encountered with GitHub were merge conflicts, which were relatively simple to solve on a case by case basis.

## Gems Used:

**Core Gems**

1. Rails ( rails ): The primary web application development framework written in Ruby. It follows the model-view-controller (MVC) architecture, providing default structures for a database, a web service, and web pages.

2. Sprockets-Rails ( sprockets-rails ): Integrates with Rails to compile and serve web assets like JavaScript, CSS, and images. It's a part of the Rails asset pipeline, optimizing the serving of assets.

3. SQLite3 ( sqlite3 ): A lightweight database engine. Used as the default database for Active Record in Rails, especially in development and testing environments due to its simplicity and minimal setup.

4. Puma ( puma ): A high-performance HTTP server for Ruby/Rails applications. It's designed for speed and concurrency and is often used in production environments.

5. Importmap-Rails ( importmap-rails ): Allows the use of JavaScript with ECMAScript Module (ESM) import maps in Rails, simplifying JavaScript bundling and module Management.

6. Turbo-Rails ( turbo-rails ): Part of Hotwire, it offers Single Page Application (SPA) like functionality by speeding up HTML updates over the wire without needing client-side JavaScript.

7. Stimulus-Rails ( stimulus-rails ): Another part of Hotwire, it's a modest JavaScript framework for enhancing HTML with minimal custom JavaScript code.

8. JBuilder ( jbuilder ): Facilitates the creation of JSON data structures, typically used for building JSON-based APIs in Rails applications.

9. BCrypt ( bcrypt ): Provides mechanisms for securing passwords, used for hashing passwords in Rails applications.

10. OmniAuth ( omniauth and related gems): A flexible authentication system using OAuth. The specific gems mentioned are for integrating Google OAuth2 for user authentication.

11. Dotenv-Rails ( dotenv-rails ): Manages environment variables, useful for keeping sensitive information like API keys out of the codebase.

12. OmniAuth-Rails CSRF Protection ( omniauth-rails_csrf_protection ): Adds CSRF protection for OmniAuth requests, enhancing security.
13. Font-Awesome-Sass ( font-awesome-sass ): Provides the Font Awesome icon set as Sass files, useful for adding icons to a Rails application.

14. Sassc-Rails ( sassc-rails ): Enables the use of SCSS (Sassy CSS) within Rails applications for more advanced styling capabilities.

15. Caxlsx and Caxlsx_Rails ( caxlsx , caxlsx_rails ): Used for generating Excel files, handy for reports and data exports.

16. Tzinfo-Data ( tzinfo-data ): Provides timezone data, particularly important for Windows environments that do not include zoneinfo files.

17. Bootsnap ( bootsnap ): Caches Ruby bytecode to reduce boot times, improving the startup performance of the Rails application.

18. Bootstrap ( bootstrap ): A popular front-end framework for developing responsive and mobile-first websites.

19. JQuery-Rails ( jquery-rails ): Incorporates jQuery, a fast, small, and feature-rich JavaScript library, into the Rails application.

20. Chartkick ( chartkick ): Simplifies the creation of charts and graphs, integrating seamlessly with various JavaScript charting libraries.

**Development and Testing Gems**

1. Byebug, Debug ( byebug , debug ): Debugging tools for Ruby, allowing developers to inspect and troubleshoot their code.

2. Rubocop and related gems ( rubocop , rubocop-performance , rubocop-rails , rubocop-rspec , rubycritic ): Code linters and analyzers to ensure code quality and adherence to Ruby and Rails best practices.

3. Launchy ( launchy ): A helper for launching cross-platform applications, useful in a development environment.

4. Web-Console ( web-console ): Provides an interactive console on exception pages, aiding in debugging.

5. Capybara, Cucumber-Rails, RSpec-Rails and related gems ( capybara , cucumber-rails , rspec-rails , selenium-webdriver , simplecov , webdrivers , ZenTest ): These gems are used for testing the Rails application, covering features like integration testing, behavior-driven development (BDD), and test coverage reporting.

**Production Gems**

1. PostgreSQL ( pg ): The PostgreSQL adapter for Ruby on Rails, used in production environments, particularly for Heroku deployments.

## Code is pushed to GitHub Repo:

All development was done using the git repo:
https://github.com/Data-Mgmt-for-Sealants-Outreach-Program/Sealant-On-Rails-V2

## Describe Repo Contents for Deploying Project:

The README file in the git repository contains instructions for deploying the project. The GitHub repo contains most of the files needed for deployment, but some new files will need to be created, edited, or deleted based on the README instructions. There are also additional setup instructions regarding Google OAuth included that must be followed for the application to function properly.

.

## Resource Links:

Project Management Page:
https://github.com/orgs/Data-Mgmt-for-Sealants-Outreach-Program/projects/1

Github Repo:
https://github.com/Data-Mgmt-for-Sealants-Outreach-Program/Sealant-On-Rails-V2

Heroku App:
https://sealant-on-rails-v2-34df313219f3.herokuapp.com/login

## Links to Presentation and Demo Video:

Presentation Link:
https://docs.google.com/presentation/d/160D34hv8-V04ZE1GD4tA0r-JcqRvh2rt6QyVdLz5rcA/edit?pli=1#slide=id.g2a0cc6726ee_0_667

Video Link:
https://drive.google.com/file/d/11592Wui-Bs8wEZcxgB6Inrj6RlKbmNM2/view?usp=sharing