Offline-First CRM System

John Abramo

1 Introduction

Lifetree WNY is a locally owned small business providing tree maintenance services throughout Western New York. As they perform work, they need a way to keep track of the status of every customer and their associated appointments, quotes, invoices, and other associated files. Their work frequently brings them to remote areas with poor cell coverage making the ability to do all of this without needing a network connection important. Previous to the start of this project, they used an iOS application, called Routzy. Routzy fulfilled the requirements for their use case, but in recent years, have scaled back support for the application and not fixed bugs occurring on newer versions of iOS. Because of this, the business needed to replace Routzy with an application that was well supported on modern operating systems. After extensive searching, no other application existed that could fulfill all of the requirements that Routzy did. The decision was made to develop a custom application to replace the functionality of Routzy.

2 Features Delivered

The application is separated into two major parts, a frontend progressive web application (PWA) built using React and typescript, and a backend REST API built using typescript with express framework. A progressive web application was selected because it allowed us to create an application that could be installed to any device for offline use without needing to go through the process of publishing a full application using an app store. Additionally, using a PWA gives the ability to use the app on any device without needing to build separate applications for each platform.

To handle data offline, the frontend keeps a local version of the database stored in the browser. Along with the data itself, it also keeps detailed timestamps of when changes are applied to the data so that it can sync to the central backend so that other devices can see the changes. On the backend, it also keeps detailed timestamps of when the data is updated so that it can send the correct updates out to devices when they are syncing. The basic syncing process is that a device will pull all changes that the backend has since the timestamp of its last sync and apply them to its local database. It will then push all of the changes that it has made since the last time it pushed.

The main feature of the app is the ability to manage customer information and other information relating to work to be performed for customers. The main page of the application has a list of all customers and the ability to add a new customer or select an existing customer to gather more information about them. It contains contact information about the customer as well as links to other data associated with them such as upcoming appointments, quotes, invoices, and job site photos.

Besides just being able to store information in a database, some of the records have special features associated with them. The job site photos page allows the user to capture or upload an image, then draw on the image using a finger or stylus to annotate the work to be performed. The proposals and invoices page allows for generating a pdf of the proposal or invoice as well as the ability for the customer to immediately sign the contract agreeing to the terms and conditions. When sending an email from the application, the user can select a predefined template with variable placeholders to generate the email to send to a specific client with their information filled in.

Integration with other systems was also a requirement for this application. Data about the jobs ready to be completed needed to be added to the application the field crew uses to manage jobs. This was accomplished by using a REST API to create and update jobs when an invoice was synced to the backend API. Emails needed to be sent to customers with their proposals and invoices. This was accomplished using the Gmail API as it allowed sending from the gmail account that the business uses to communicate with customers. Appointments needed to be added to a calendar. This was accomplished by using the Google Calendar API to create events for each appointment and adding additional information to the calendar event.

3 Future Improvements

3.1 Improve integration with Quickbooks

Currently Quickbooks integration is supported by allowing the user to select a range of dates and exporting the data for the date range as a csv file that is able to be uploaded and imported to Quickbooks Online. Quickbooks Online has a REST API that can be used to update data. It would be better if, in the future, this API was used to upload proposal details when the data is synced to the backend automatically.

3.2 Automated Customer Follow-ups

A feature could be added to send follow-up emails to clients after a job is completed. This could be used to collect feedback from clients or send any reminders that need to be sent after the job is completed. It could also be used to direct customers (potentially based on their feedback) to leave reviews on public platforms. This would not only improve customer engagement but also facilitate the collection of reviews on platforms like Google to help the business's online presence.

3.3 Online Payments

A feature could be added to allow customers to approve and pay their proposals online. This would allow for automating the payment collection process and reduce the time the business spends tacking down missing payments and be more convenient for customers.