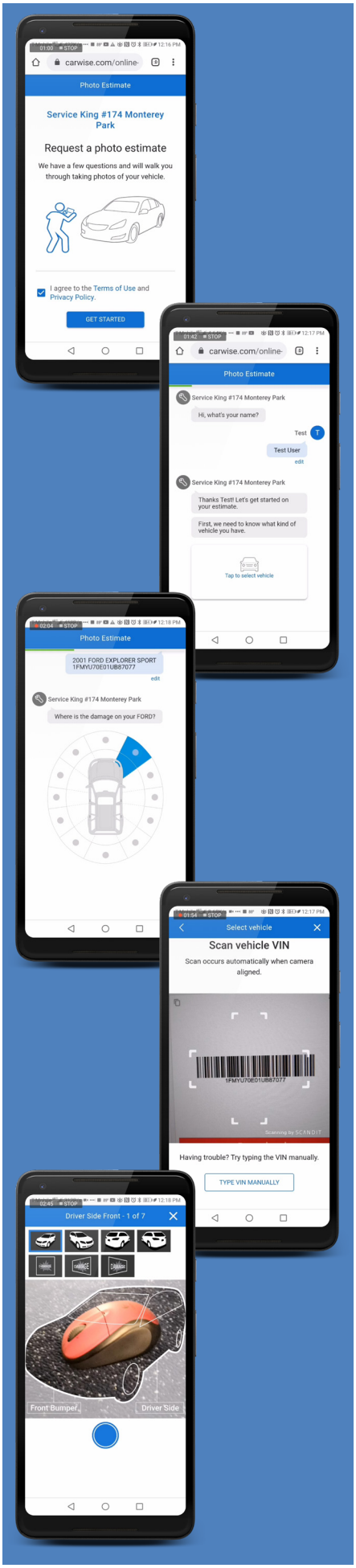
Brett Brewer Full Stack Web Developer



Work Sample

Carwise Photo Estimate Mobile Web App

Led development of a photo estimate mobile web app developed in ES6/Vue.js/Java8/SpringBoot that uses a “conversational” UI (chat-bot style interface) to allow users to submit photos and other info to auto body shops and receive repair estimates.

View Demo

This is a screen recording I made of the online estimate tool in action.

https://drive.google.com/file/d/1V5_uuguNIvO9-vhwifGt5bwKSXUY1Ofr/view?usp=sharing

You can see a live version here, but please don’t complete your photo submission as that annoys the body shops:
https://www.carwise.com/auto-body-shops/book-appointment/fix-auto-buena-park-buena-park-ca-90621/461953

Synopsis

My team developed this app over a period of about 6 months. A UX team provided us with mockups and vector based UI assets. My team implemented the app as 2 separate projects in a single Git repository. As we were using Maven (no choice), there was a main parent POM pointing to 2 child modules, a “frontend” and “backend” that could be built/run/tested separately, but deployed in production as single app. I handled virtually all of the initial setup of both the frontend and back-end projects, including all the complex configuration of the build tools such as Vue-cli/Weback on the frontend project and Spring Boot/Maven for the backend project.

Frontend (UI)

I led all aspects of the frontend implementation and wrote most of the javascript, scss and handled all the image creation/optimzation issues. (graphics were designed and provided by a UX team) . Built in Vue.js with the Vuetify Material Design component framework. Vuex was used to maintain an organized data structure and handle async actions, mutations, etc. IndexDB (native browser storage) was used to cache the app state locally so that accidental page refreshes do not reset the user’s progress while using the estimate tool. Barcode scanning is provided by a commercial scanning library which is integrated with our custom UI to allow users to scan the VIN number of their vehicle to quickly identify year/make/model, etc. To integrate the final frontend build with the backend project, special Webpack and Maven configurations were created to build/transform the frontend image/js/scss resources and customize the app entry page template (index.html) which was really a Java Thymelaf template. This required special consideration since Thymeleaf template variables break webpack as they are interpreted as JSX variables and parsed by Webpack during the build. If you manage to make Webpack happy, you inevitably break the Thymeleaf variable syntax and crash the Java app when the templates are parsed by the backend.

Backend

Java 8 - built with Spring Boot v2.1.7 using built-in Tomcat server for local dev and debugging, and deployed on Weblogic. Oracle DB was used for the persistence layer. Uses spring-boot-dependencies for automatic dependency management (of most dependencies), along with specific spring-boot-start dependencies, Google Guava, Google Gson, etc.

On the backend I was primarily responsible for designing the data format that was consumed by the frontend, writing much of the functionality that transforms backend objects into JSON responses, building the APIs that load the frontend app config and handle REST-style requests and validation for the persistence layer.