SUMATOSOFT

SOLUTIONS

Robotics, Order management system, On-premise solution

TECHNOLOGIES

Ruby 2.7.0, Rails 6.0.2.1
React 17.0.1 and Workbox
Python 3.9, Flask 1.1.2
Kotlin 1.3.72, Postgres 12.3
RabbitMQ 3.8, MQTT, STOMP, LCM protocol, Docker and Docker-compose, Terraform
AWS: Lambda, API Gateway, S3, IoT Core, SNS, DynamoDB
Nginx, Gunicorn Sumologic

DURATION

2 years

TEAM

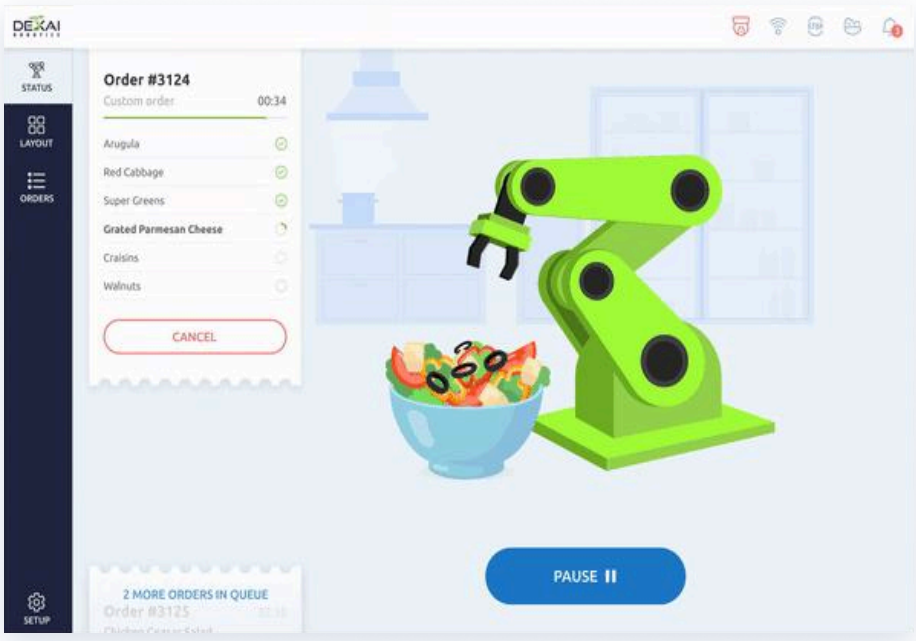
1 Project Manager
1 Account Manager
1 Designer
1 Software Architect / Tech Lead
2 Middle / Senior developers

sumatosoft.com
info@sumatosoft.com

Client: Dexai Robotics
Industries: HoReCa / Restaurants & Hotels / Hospitality
Region: Boston, Massachusetts, USA
Website: dexai.com

Dexai Robotics: graphical user interface for robot operation

A GUI that enabled Dexai Robotics to deploy Alfred to restaurants without requiring an engineer on-site - cutting robot setup time per shift by ~65% and reducing interaction errors by ~50% through real-time visual state monitoring.



Meet Alfred, an automated robotic arm and your personal sous chef! Alfred was born at Dexai Robotics, while SumatoSoft developed a graphic user interface (GUI) that helps to communicate with the robot!

Special features

State Page – displays an action that the robot is performing at the moment. There are unique and animated screens for every state. The design of screens is very bright and simple, so robot operators can easily understand the current state of the robot at a glance.

Error and alert handling: the app informs the operator of any problems and guides them on how to resolve the issue. If the robot runs out of ingredients, needs a bowl, or a utensil is missing, the app shows an alert.

Order queue page – displays the sequence of orders, order details, and the progress of the current order. The operator has an option to change the sequence of orders or cancel them.

SUMATOSOFT

Ingredients monitoring – the app has a page that displays bins with ingredients and the level of ingredients in them. The layout of the bins corresponds to the real placement of bins around the robot.

An option **to pause the robot.**

Business challenge

There was a **robotic arm named Alfred**, but only robotics and software engineers could communicate with it. The client didn't have a user-friendly interface that sends commands to the robot or monitors its state.

The company planned to sell Alfred to restaurants and reached out to the SumatoSoft team to develop a platform with a GUI for interaction between restaurant personnel and multiple robots.

The main challenges

Not all restaurant workers are native English speakers. Thus, the GUI must be intuitive, informative, and simple so every restaurant worker can understand the current Alfred state or an error at first sight on the screen.

Another challenge of this project was the requirement that the main order processing flow should work offline (without global Internet connectivity).

The list of additional requirements includes

- ✓ built-in user management system
- ✓ robot state and actions monitoring
- ✓ sending commands to robots
- ✓ POS systems integration
- ✓ managing the order queue
- ✓ error/edge cases handling

Our solution

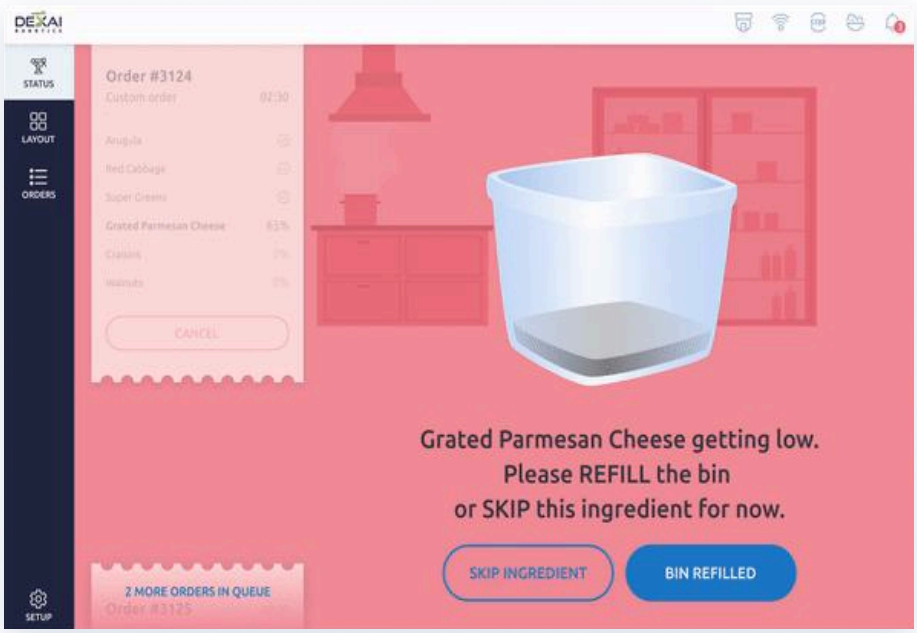
ReactJS was chosen as the main technology for the GUI application, and Ruby on Rails was chosen as the backend technology. The main reason behind this decision was that they are well-suited for the rapid prototyping and startup environment we worked in. The initial architecture included several microservices with HTTP communication. These microservices were a React UI application and a Rails backend server, along with a Kitchen Coordinator service written in Python that was responsible for communicating with the robotics arm.

During the development of the first version, a number of revisions were made both to the requirements and our final solution. First of all, we faced changes in requirements – the customer decided to go only with one robot instead of multiple for the MVP stage. Secondly, we found out that the robot was executing commands asynchronously and wasn't providing the results of these commands. It became clear that the HTTP protocol with its request/response scheme wasn't the best choice in our case, so we abandoned it.

SUMATOSOFT

At that point, we'd also decided to get rid of the Rails server since we could move most of its features to another internal server and the remaining features were deprecated. In addition, the customer wanted a new version of the graphical interface due to some changes in requirements. All this led to a new version of the architecture, where the Rails server was deprecated and all microservices communicated via Message Broker (RabbitMQ) with each other. After implementing this version, we were able to immediately display robot state changes, send commands and orders to it and handle errors from the robot side.

One of the challenges of this project was the requirement that the main **order processing flow should work offline** (without global Internet connectivity). To achieve this goal, we decided to build a PWA (Progressive Web Application) as the main UI for kitchen staff based on our React UI application.



During the next stage, the client needed to develop **several additional microservices and a separate user interface for order placing**.

Our final solution included multiple backend services:

- ✓ service for order creation and storing them in the cloud;
- ✓ order queue management server;
- ✓ microservice for storing and managing recipes;
- ✓ service for synchronizing data between on-premise and cloud service;
- ✓ UI for placing orders written in React JS and the cloud microservices written in Kotlin.

The developed solution allowed us to integrate with third-party POS systems and keep a close watch on the order lifecycle. Each microservice was wrapped into a Docker container, while metrics and logs monitoring relied on SumoLogic.

SUMATOSOFT

Customer’s benefits

By removing the dependency on engineering staff, restaurant personnel at any skill level now commission and control Alfred independently, cutting setup time per shift by ~65%. The GUI's visual state indicators and built-in error handling reduced robot interaction errors by ~50%, enabling reliable multi-robot operations across diverse restaurant environments.

What’s happening with the project right now?

The project was successfully completed, the user interface allows to send commands to the robot arm. The robot functionality is currently being finalized to start real-life commercial deployments.

sumatosoft.com

info@sumatosoft.com

One Boston Place, Suite 2602 Boston, MA 02108, United States