System Design Document

CSC301 Project

Inwit

Rayhan Fazal
Longyu Li
Arthur Ng
Sabih Sarowar
Raghav Sharma
Kevin Shin

Table of Contents

1. Introduction

1.1 Product Overview

Inwit is a food container service startup that allows users to order food using reusable containers. Restaurants partnered with Inwit agree to offer an option to deliver food through reusable Inwit containers and can be ordered from through the Inwit web application. For reference, the final product would be similar to Uber Eats or DoorDash.

1.2 Project Goals

The primary goal of this project is to implement new features for the service. New features that will be implemented include but are not limited to: container management, order management and mobile UI. A secondary goal of the project is to improve the user experience by improving response time of the app.

2. General Design Approach and Considerations

2.1 Environment

The application is managed through WordPress and uses MySQL as a database management system. Files are accessed through cPanel, which we have managed to link our GitHub repository to.

2.2 Considerations

The Inwit team has provided an existing codebase to work with. The current application contains basic features and plugins. Features of the web application are contained within individual plugins, which can be enabled through the WordPress interface.

2.3 Approach

Our team will add functionality to the web application by creating plugins. Plugins can then be enabled through the WordPress interface. A separate plugin will be created for each distinct feature.

3. Software Architecture

3.1 Architecture Design

Our team was unable to choose the architecture for this project as we were given an existing codebase. We believe, however, that the existing codebase resembles the microservices architecture (or alternatively, a service-oriented architecture). A web application in WordPress is constructed by combining various plugins. Each individual plugin acts a separate, modular service and can be attached or detached from the web application. Plugins are independent and are solely responsible for their own functionality, meaning they can be worked on without rebuilding the entire application.

Link for more about the microservices architecture
https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices

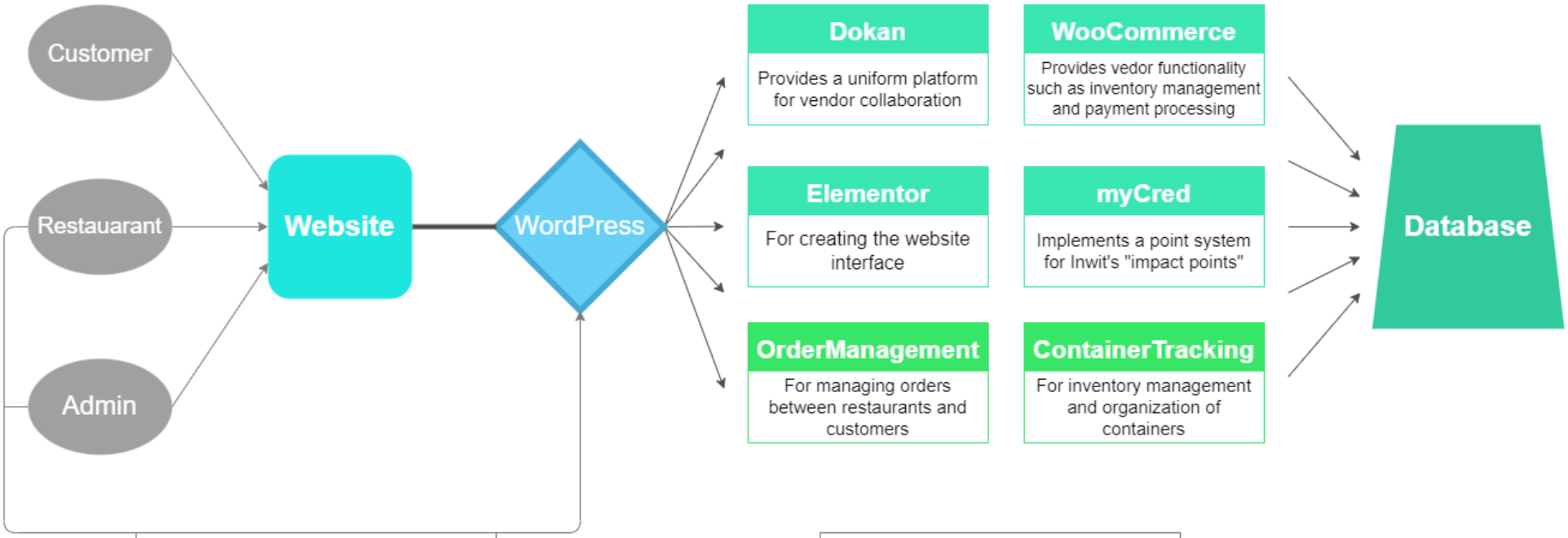
3.2 Software Architecture Diagram

A diagram has been created to illustrate how the Inwit web application uses the microservices architecture. It has been labelled “Microservice Architecture Diagram” and is viewable in the next page.

3.3 Provided/Existing Diagrams

The Inwit team has also provided a diagram which shows the responsibilities and relationships of existing components and the main structure/flow of the website. The provided diagrams are viewable after the software architecture diagram specified in 3.2.

Microservice Architecture Diagram



Plugins can be enabled and added to the website through the WordPress interface

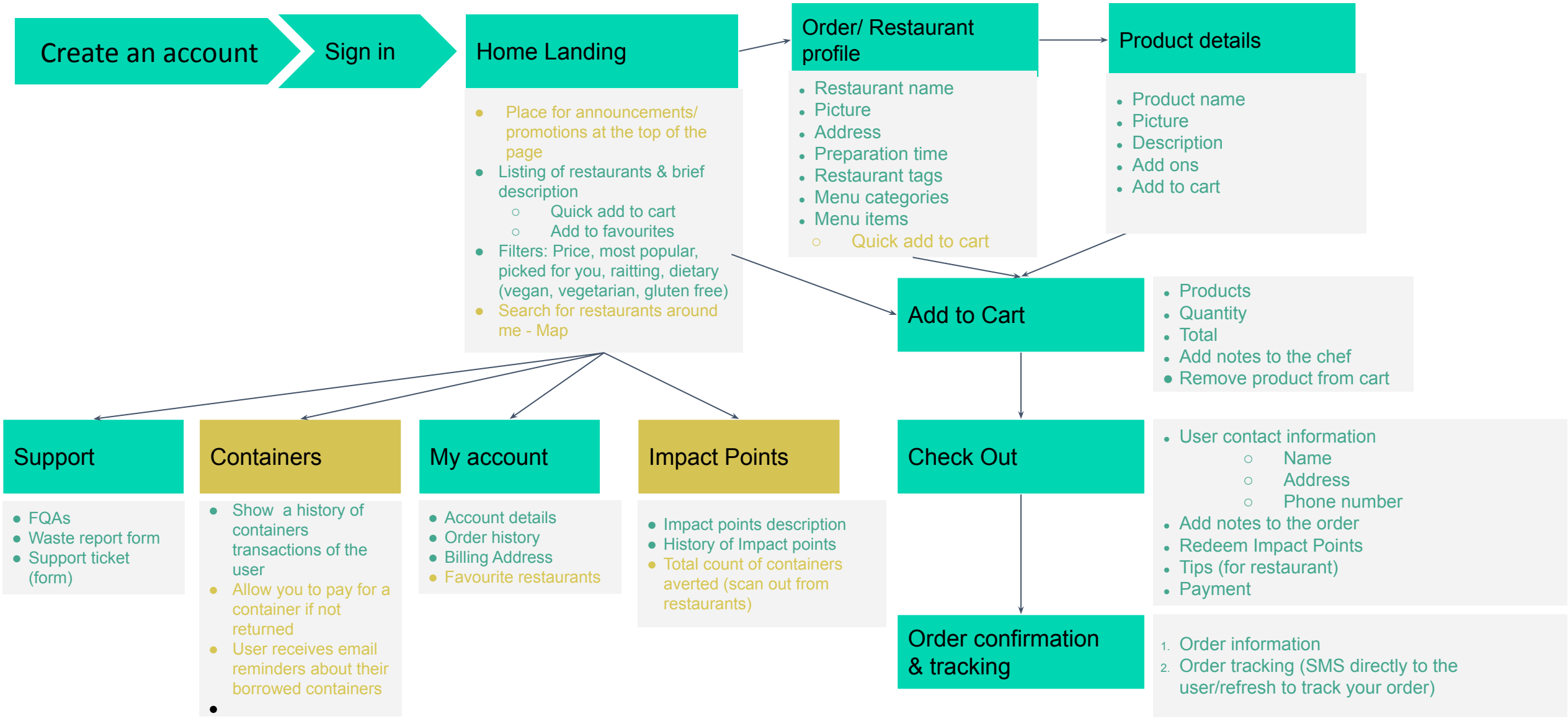
Each plugin is an independent, modular service that can be attached or removed from the application

already implemented in some form

targeted features for the project

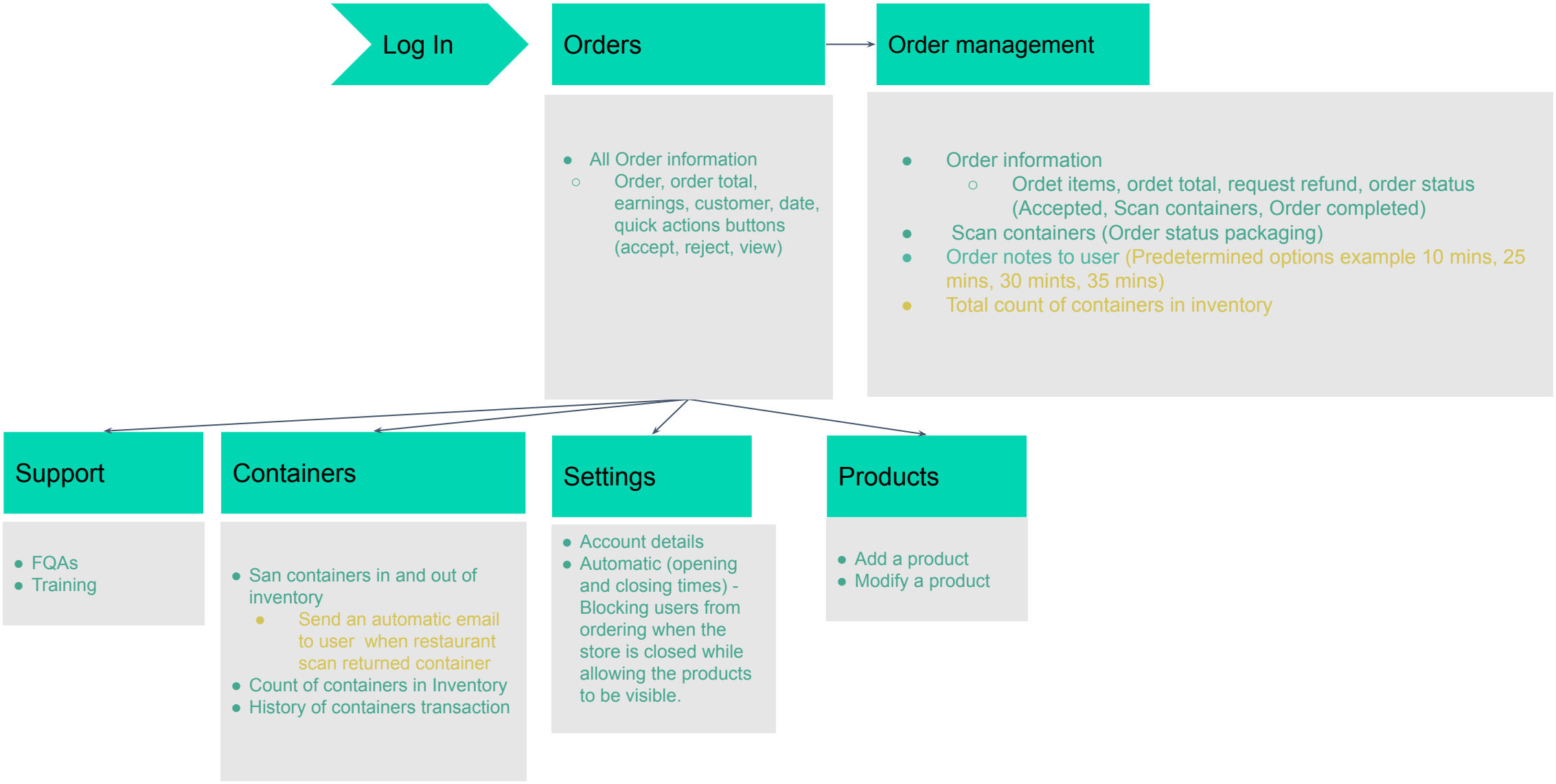
Users

As a customer, I'm able to create an account, identify quickly restaurants around me, order food in desired restaurants, I am able to filter food by my dietary preferences, I am able to pay quickly and track my order to know when my order is accepted and my food ready for pickup. I receive Impact Points for every dollar spent on the platform and for returning containers, It is easy for me to know how many containers I have into my account and how long I have left to bring them back to any restaurant.



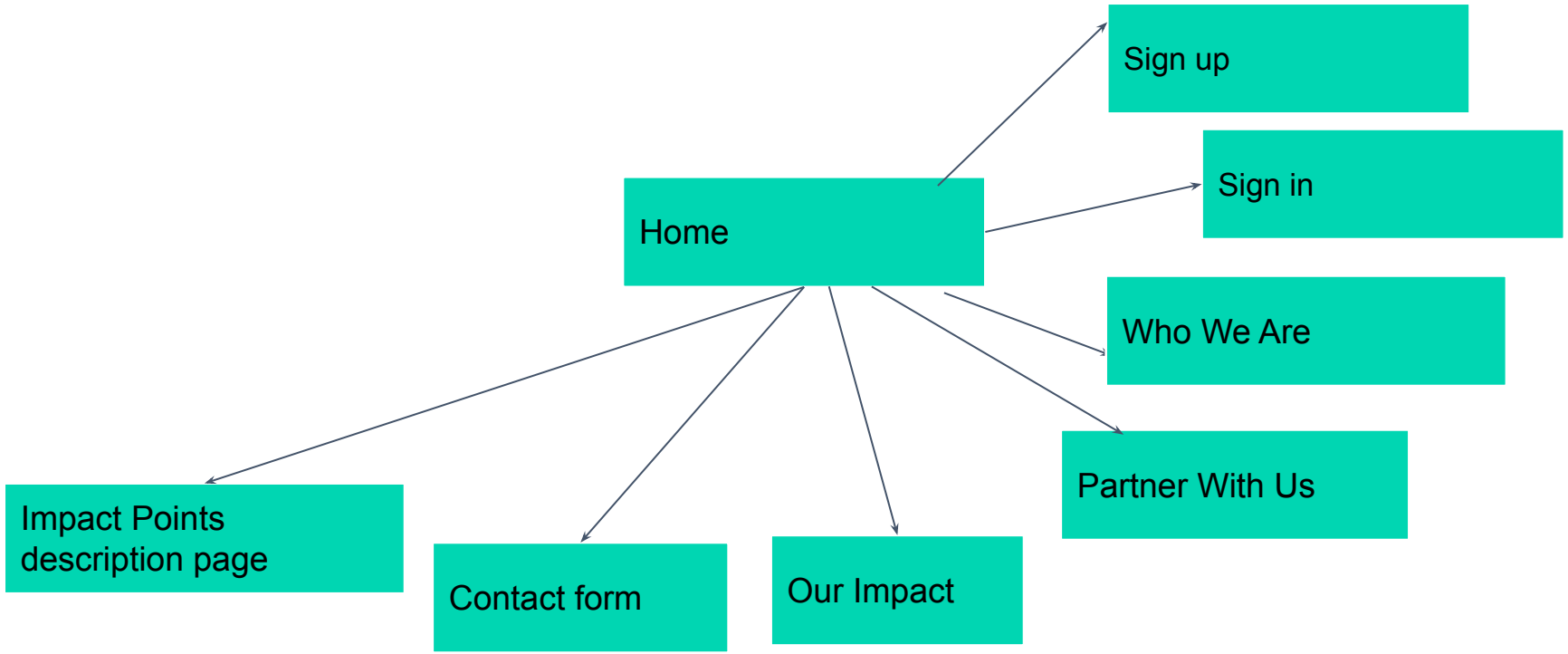
Restaurant

As a restaurant, I'm able to access my vendor site, to receive orders, accept them and set a preparation time and scan containers to the order.



Public Pages

Informative



4. CRC Cards

Container	
Represent an instance of a physical container Contain information and current status about the container	ContainerManager (Customer) ContainerManager (Restaurant) ContainerView (Customer) ContainerView (Restaurant)

ContainerManager (Customer)	
Keeps track of the number of containers Remove tracked containers after return	ContainerView (Customer) Container

ContainerManager (Restaurant)	
Keeps track of container inventory Update database upon container loan/return Notify point system upon return	ContainerView (Restaurant) Container myCred (points system)

ContainerView (Customer)	
Display containers that need to be returned to user Display information about container status	ContainerManager (Customer) Container

ContainerView (Restaurant)	
Display containers inventory Display information about container status	ContainerManager (Restaurant) Container

Order	
Represent an order for a restaurant Contain information and status about the order	OrderManager (Customer) OrderManager (Restaurant) OrderView (Customer) OrderView (Restaurant)

OrderManager (Customer)	
Let customers send orders to restaurants Send notifications to restaurants upon order Receive order status updates from restaurants	OrderManager (Restaurant) OrderView (Customer) Order

OrderManager (Restaurant)	
Let restaurants receive orders from clients Receive notifications from customers upon order Order status editing Send order status updates to customers Mark orders as finished Update container status	OrderManager (Customer) OrderView (Restaurant) Order ContainerManager (Restaurant)

OrderView (Customer)	
Display status of orders Display information about orders	OrderManager (Customer) Order

OrderView (Restaurant)	
Display orders received from customers Display notifications	OrderManager (Restaurant) Order