Grant Hall Website

Created By
CDT Antoine Davis
CDT Austin Pearson
CDT Spencer Welton

Problem Description

The USCC Dietician is testing field implementation of a new system in Grant Hall that will help to determine the volume of cadets that visit Grant Hall and the types of food most frequently consumed. There has been some concern over USMA health, including cadets, faculty, and staff, in terms of food choices and the Superintendent would like to enforce a stricter standard as a result. He would like to know how often cadets are eating in Grant Hall and the times at which the highest number of cadets, staff and faculty are eating at Grant Hall.

The objective of this database is to store order information in order to view times of heavy traffic at Grant and see which items are being purchased the most. The health related information of the food is also stored in order to provide the USCC Dietician, Superintendent, and Grant Hall Staff with data that is useful.

User Vignettes / Use Cases

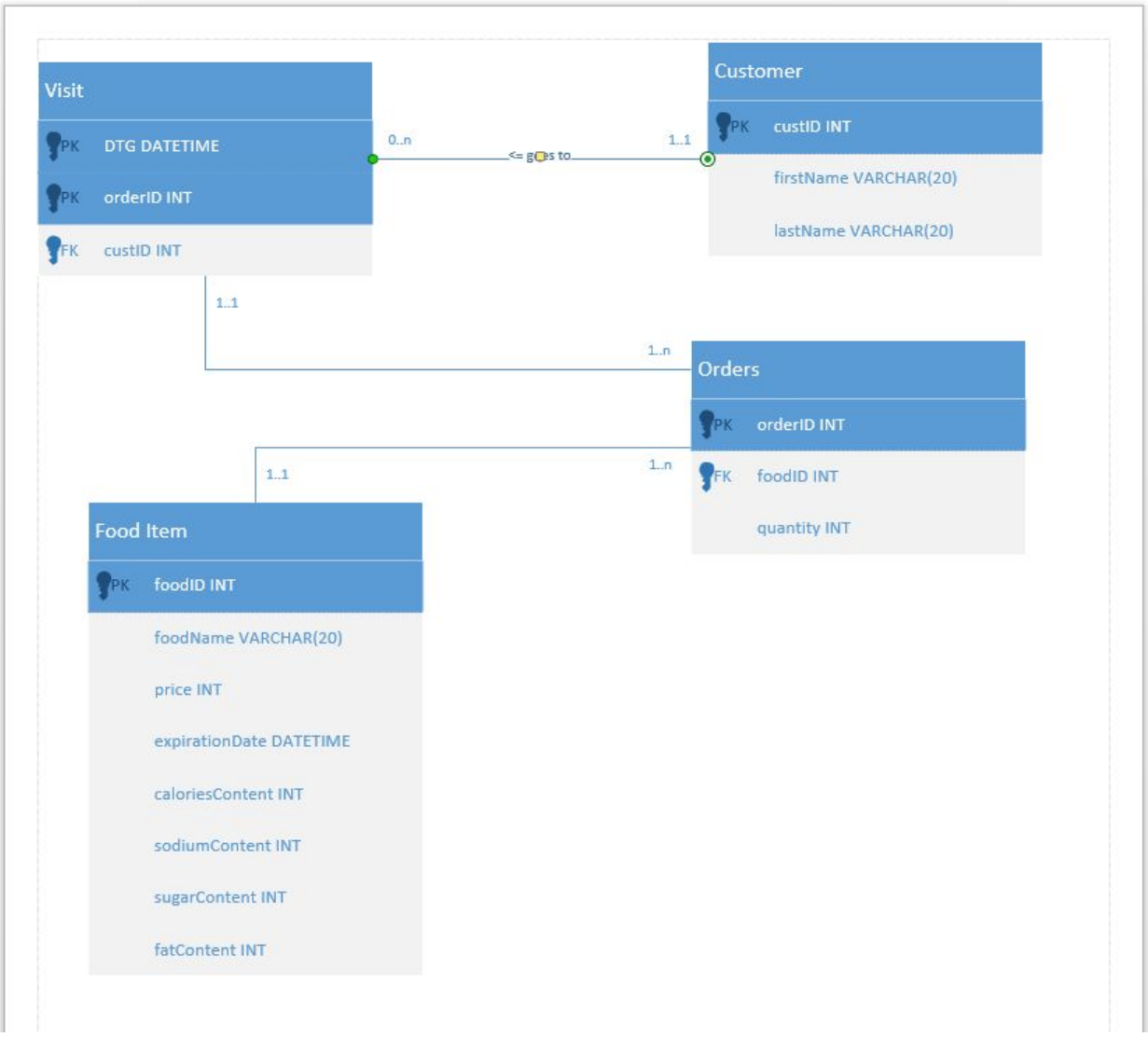
USE CASE 1: The USCC Dietician can view nutrition facts and information associated with the food that Grant Hall sells as well as view which cadets are consuming these food items.
USE CASE 2: The USCC Dietician and Superintendent and Staff can view Grant Hall customers that have visited Grant in the last 7 days.
USE CASE 3: The Grant Hall Finance Manager and The Grant Hall Head Manager can view times of heavy traffic at Grant Hall (where visits exceed expected amount within one hour).
USE CASE 4: The Grant Hall Finance Manager can view cost and revenue associated with orders made by customers.
USE CASE 5: Grant Hall Cashiers and the Grant Hall Head Manager can add or update customer information.
USE CASE 6: The Grant Hall Supply Manager and The Grant Hall Head Manager can add or update food menu information.

Database Design

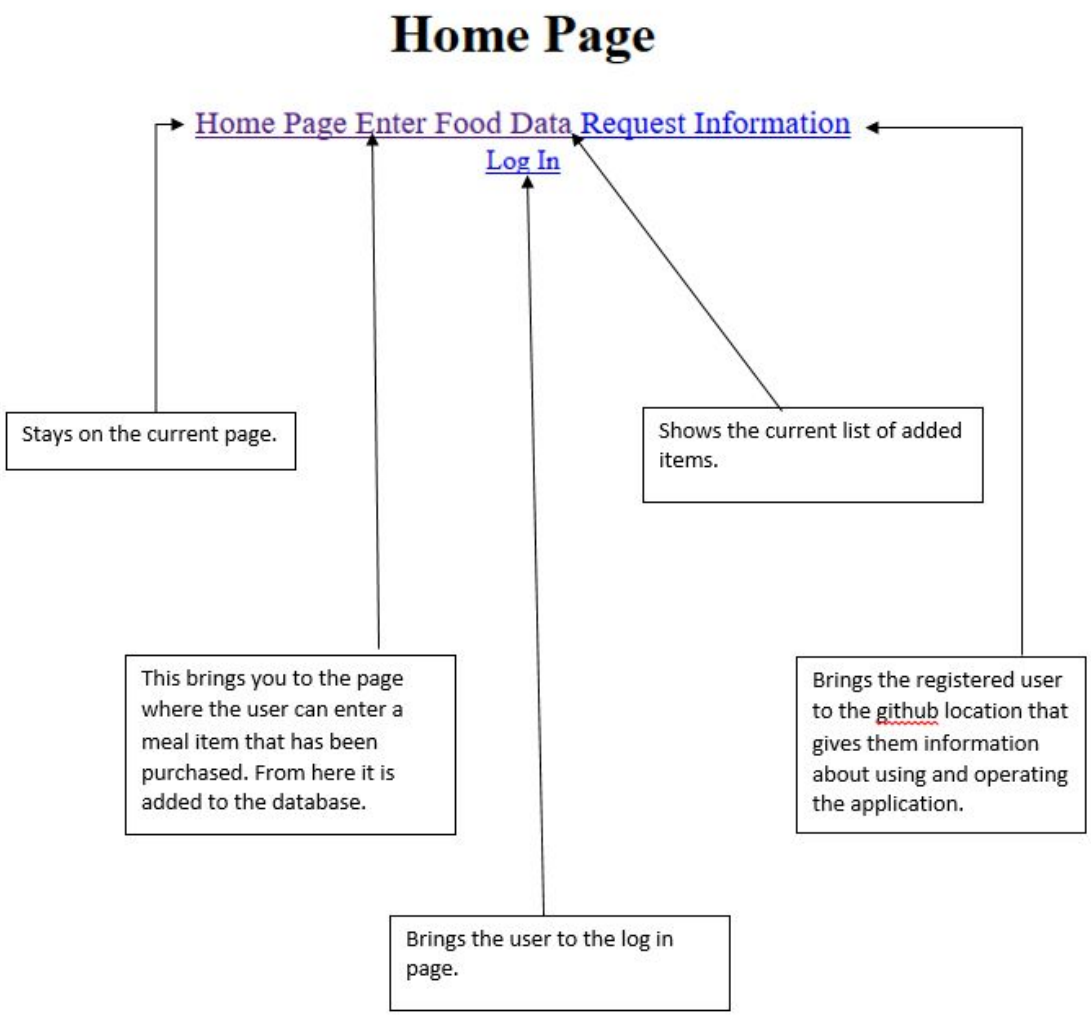
The database was designed to follow the use case paragraphs and allow the user vignettes to accomplish their intended tasks. The food served at Grant Hall has various nutrition facts available to view. The times that a customer visits Grant Hall are also stored in the database and accessing this information can reveal times of heavy traffic which would be relevant to Grant Hall management staff in order to allocate resources accordingly. Order information is stored and

upon viewing, reveals data such as revenue, popular items, and which customers are ordering which items. Finally, customers and food items can be updated and changed in the database to account for new additions or changes to their data.

We had originally modelled a supply request using the database diagram. This proved to be unnecessary for the user vignettes that we were trying to target. The supply request was intended to create a new order for food that was out of stock and relay that information to a supplier but our group agreed that allow that system may be useful, it would be better off as a standalone system rather than being a part of customer orders.



Interaction Design

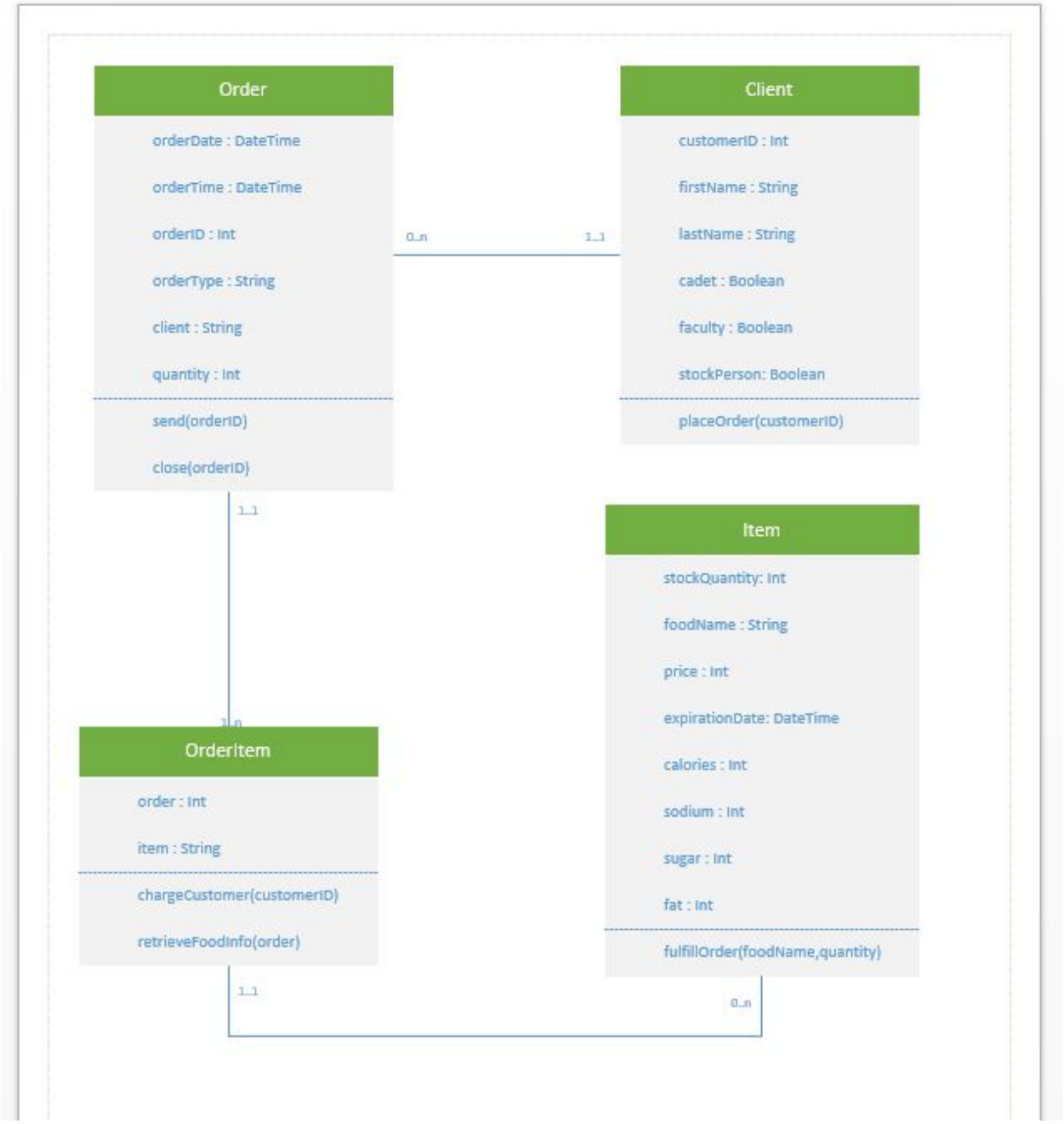


Class Diagram

We have four major classes that demonstrate the functionality of our system. Our client class retains relevant information that we are modelling for a customer that places a food order at Grant Hall. When an order is placed, the order has information that is unique to itself. Immediately, the order generates a unique ID and timestamp. The relational class 'OrderItem' links the information between Order and Item. This class contains functions that are relevant to

both Order and Item. Finally, our Item class retains relevant information for all of the food items in Grant Hall and allows for a completed feedback call to fulfill orders by replying with specific food items for an order request.

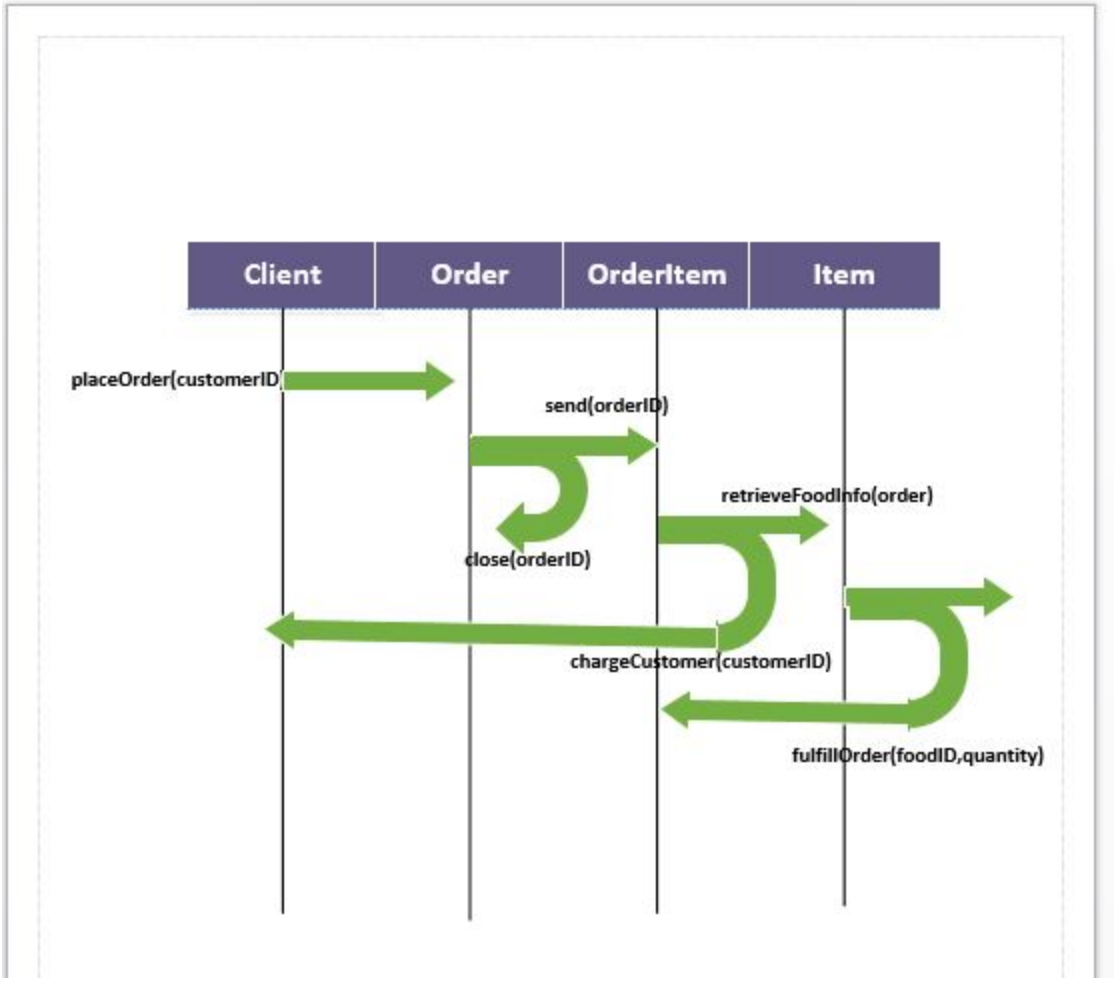
Originally, we had a supplier class and a delivery class that represented a logical flow of functionality that would occur when an order was placed without the food item being in stock. We chose to update this functionality as it was outside of the immediate scope that we were trying to achieve with our system.



Sequence Diagram

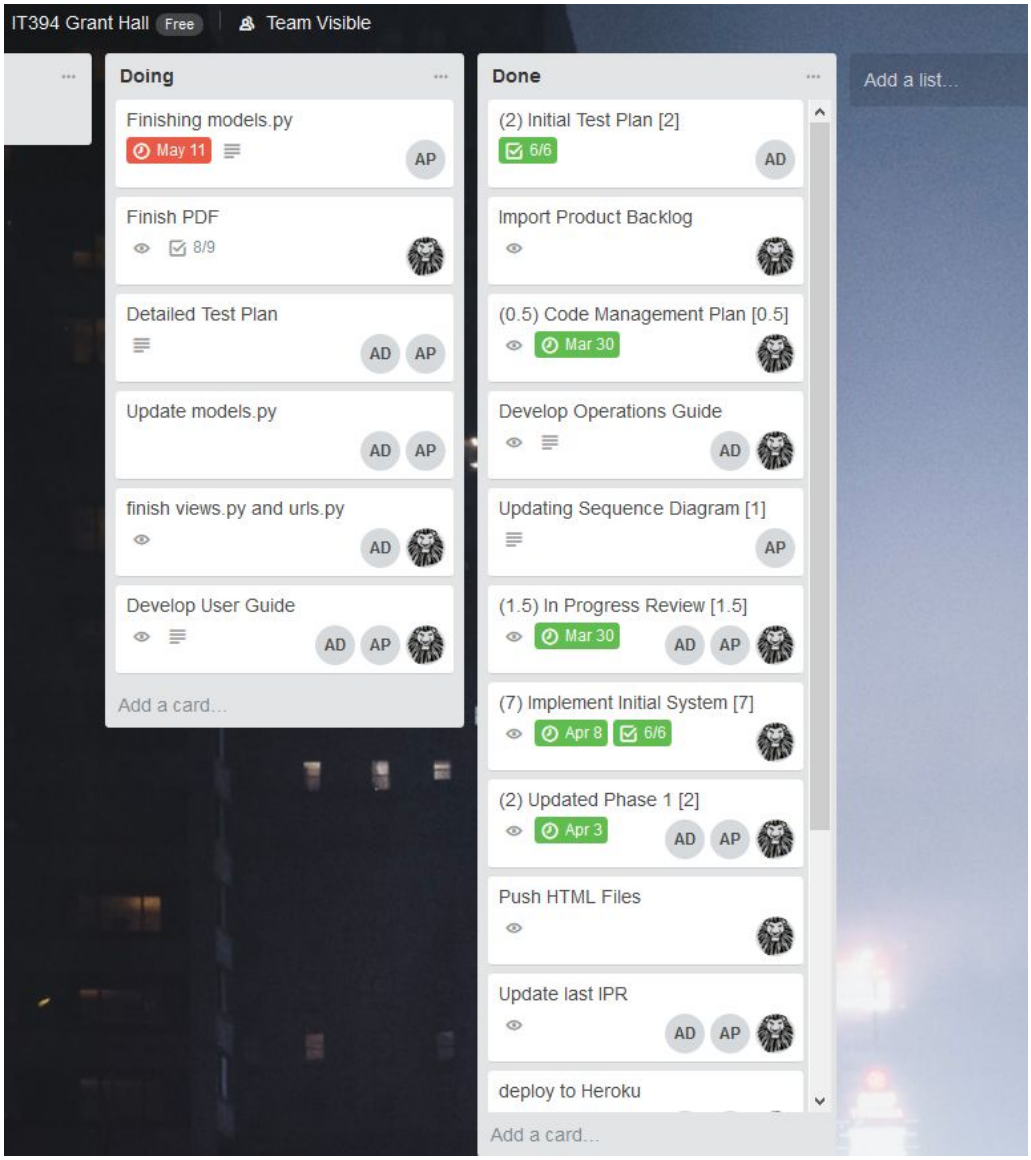
A Client places an order using their unique customer ID. Then the order is either executed or closed based on sufficient food quantity from Grant Hall's shelf stock. If the order is sent through, food items are confirmed using the food class and the order output is passed to the food providers and feedback is sent to the client to confirm their order.

In our previous sequence diagram, we had created a process that sent closed orders to a delivery class that would generate supplier information and delivery requests that would ready a resupply option for the item stock quantity. This was later changed to reflect the scope of the project.



Product backlog

For our product backlog we used Trello to track our progress and understand what was still needed before we would be considered complete. This allowed us to easily maintain our task and keep track of who was responsible for what tasks. It also allowed us to set the conditions for what finished means for some of the task as well as set due dates for them. You can find the most up to date version of the backlog at the following location:
https://trello.com/b/3tAz3e8r



This is what our Backlog looked like on Trello.

Detailed Test Plan

In order to complete our project we need to meet the definition of done for our product. We accomplish this by making our product releasable, deployed on a demo server, and that it was both acceptance and integration tested. We also test to verify user authentication functionality. It is important that only the admin can access the design page. We also had to make sure data that all the data we obtain from the user is entered correctly and mitigate the faulty ones. From there, when it came to accessing the available data, we needed to ensure that the user only received what was requested and what was permitted.

Code Management Plan

For out code management plan we used a shared repository in order to mitigate risk of merge conflicts. This also allowed us to clone our project from a central repository. By using the shared repository plan we could all push and pull when we needed.