INFO30005 Project: Business Requirements

Photo: Alvin Mahmudov, Unsplash

Your team has been contracted to design and build web apps for *Snacks in a Van*, a new startup company operating in Melbourne. *Snacks in a Van* runs a fleet of food trucks that work as popup cafes.

Our vendors drive around Melbourne, parking wherever they think there are customers. They have no fixed schedule and are mobile and flexible. Some vendors tend to go to the same places each day, while others are more random, or try to park near big events like sporting matches and university open days.

While our vans don't have fixed locations, their *current* location is very important to our customers, who need to be able to find nearby vans when they're hungry. We make this information available to customers via a web app. To save time, the app also allows registered customers to order their snacks in advance of arriving at the van.

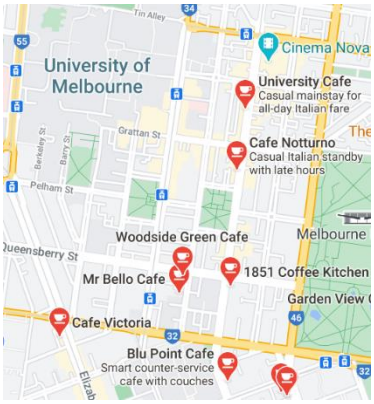
Photo: Jon Tyson, Unsplash

We expect to have around 100 vans and 100k customers. We give each van a fun name like "Tasty Trailer". Vendors are identified as van names in our apps – we don't make staff names public.

You need to make two web apps: one for customers and one for vendors.

Customer app

When a customer opens their app, they first need to see where the nearest vans are. To do this, the app needs to get the customer's location, access current van locations from the database, calculate the five nearest vans, and display their details. This display could simply be a list (closest van at the top), or for *bonus marks*, you can show their locations on a map.



After the customer chooses the van they want to purchase from (it won't necessarily be the closest one) a menu should appear which lists all the snacks available. Every van offers the same menu, because we are a chain - but customers order from a specific van.

To find a van and see the menu, customers don't need to be logged in. But to proceed beyond this point and place an order, they need to log in. (If a prospective customer doesn't already have an account, they'll need to register and create one.) In our database we want to store the customer's family and given names, a login ID (email address) and password. Customers should be able to change their profile details (name and password) later.

Use good security practices such as authentication and session management to handle logins and registrations and to appropriately restrict access to features and data.

photo: Nathan Dumlao, Unsplash

Having seen the menu, a logged-in customer can now place an order. An order consists of one or more snacks, with quantities for each (e.g. "two cappuccinos, a flat white, and three small cakes"). When the customer has entered their order details, ask them to confirm, then save the order in the database and make it visible to the chosen vendor, who must now prepare the snacks. Orders need to be timestamped, because we give customers a 20% discount if snacks are not ready in 15 minutes. Once an order has been placed, customers may want to monitor its status, such as whether it has been fulfilled and is ready for pickup.

Customers tend to change their mind a lot, and we need to design for this. We allow customers to change or cancel their order within 10 minutes of placing it and before it is fulfilled: after that they must pay for the original order. A cancelled order should be invisible to customer and vendor, but not deleted from the database, as we must keep track of this behaviour. Changing an order may involve adding or subtracting items or changing any of the quantities. If a customer changes their order, the timestamp is reset and their 10-minute change window starts again. The company may want to fine-tune these global time limits (10 and 15 minutes), so don't hard-code them.

Payment is automatically handled by an external payments system, so you don't need to implement payment in your app.

For *bonus marks*, allow the customer to rate their experience after picking up snacks. Ratings should be 1-5 with an optional comment.

Customers will mostly use their app on phones (of all kinds) but we need to cater to all screen sizes including desktops.

Vendor app


Photo: Nathan Dumlao, Unsplash

The vendors working in our vans also need an app. They log in using the van name rather than their personal name or email.

When a vendor has parked and is ready to start selling, they need to send their location to the database and mark themselves as open-for-business. Allow them to enter a short text address (e.g. "on Grattan st outside Melbourne Uni"). At the same time, have the app capture their geo-location. When they leave this location to drive somewhere else or quit for the day, allow the vendor to mark this in the database.

Vendors need a screen with a list of their van's outstanding orders, i.e. orders which have been placed but not yet fulfilled. This list needs to be in time order, with the older, more urgent orders at the top.

The vendor needs to be able to click on an order in the list to display the details of that order, including the order number, customer's given name, the items that were ordered, and how many minutes remain before a discount must be awarded.

Once an order is fulfilled (the snacks are prepared), the vendor needs to be able to click a button to mark this, and when the customer picks up the snacks, the vendor needs to be able to mark this too. Fulfilled orders need to be shows as such in the vendor's order list, while picked-up orders should disappear from the list – though they are not deleted from the database, and should be easily findable again in case customers return to discuss their order. If the 20% discount for late fulfillment was applied, make sure you record this.

We expect that vendors will use an iPad placed on a shelf in the van. Their webpages need to be usable at this screen size, and be clear and well-designed so that operators don't make mistakes in these busy workplaces.

The Menu

All of our vans offer the same menu. But don't hard-code it in HTML; rather, store it in the database. Include photos and prices for each snack. Make sure your photos do not breach copyright: we suggest you source them from Unsplash.

The list of snacks that we currently sell is: Cappuccino, Latte, Flat white, Long black, Plain biscuit, Fancy biscuit, Small cake, Big cake

Locations, distances, and maps

Locations are recorded as a pair of decimal numbers representing longitude and latitude. A precision of four decimal points corresponds to about 10 metres in Melbourne, which is good enough for our business. You can get the location of a web user via the Geolocation API and some simple client-side JavaScript.

Calculate the distance between two locations using the Euclidean formula. In other words, regard distance as the length of a straight line between the points: there is no need to calculate a walking path or take obstacles or traffic into account.

Displaying locations on a map is an optional challenge for *bonus marks*. You could use Google Maps or Open Street Maps. Instructions for using these can be found online.

"Live" pages

Some screens would work better if they updated live: for example, when customers and vendors are monitoring orders. This is an optional challenge for *bonus marks*.

Database management

Some software features are designed to manage the database: e.g., placing an order. On the other hand, there are data which are not managed by features discussed in this requirements document: e.g., it is not required that you build an admin feature that lets staff maintain the menu. In these cases, you can simply place the data in the database yourself, using any tool e.g. Robo 3T.

[end of requirements]