29.11.2019: 29.11.2019

Version: 1.0

A close up of a logo

Description automatically generated

Project Leader: Vladimir Vladinov-3856771

Project Member: Alexandru Emilian-3913120

Pizza sale obliterationator 3000

by a55

Problem description

We were given a situation which resembles a real-life problem that some restaurants face. In particular pizza shops that have at least 2 employees.

Usually the process of ordering a pizza consists of the following steps:

* Taking the order from the customer
* Forwarding that order to the kitchen
* Preparing the pizza
* Baking the pizza
* Finalizing the order

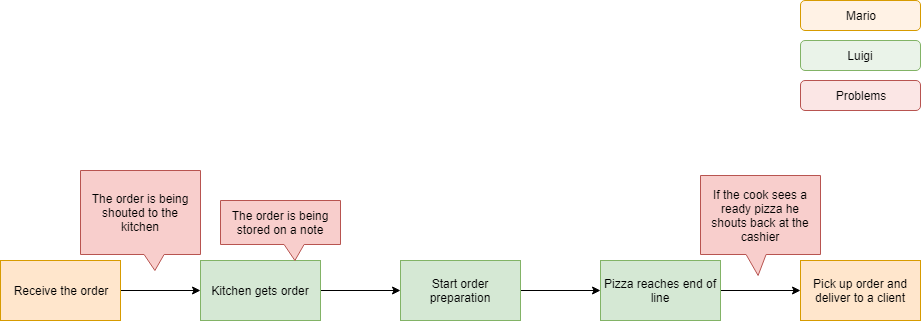
In our case we had exactly two workers: one responsible for the cashed duties and customer interactions and one for preparing the pizzas themselves. The way of preparing a single order had several problems, bottlenecking the entire process.

For finishing a single order these steps are taken:

* The cashier receives the order from the customer
* The cashier shouts the order to the kitchen
* The cook writes the order down on a piece of paper
* The cook starts preparing the order
* The cook puts the pizza in the oven (which is a conveyor-based one)
* The cook periodically checks whether the pizza reached the end of the oven
* If the cook sees a ready pizza, he shouts to the cashier
* The cashier picks up the pizza and gives it to the customer

The flowchart for the process looks something like this:

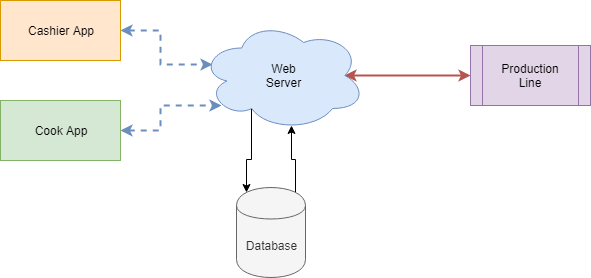
(The cashier’s name is Mario and the cook’s name is Luigi)



Firstly, there is an obvious issue with the communication between the cashier and the cook. The shouting back and forth is not an efficient way to transfer information to the kitchen and back. Furthermore, the order itself is stored on a flying note, if heard by the cook at all. That note can get lost or damaged, rendering the order non-existent. Additionally, when the order is ready for serving, there’s an unnecessary waiting time for the cook to see the baked pizza at the end of the line, followed by a shout to the cashier, informing him he can complete an order. And that’s the main bottleneck of the process.

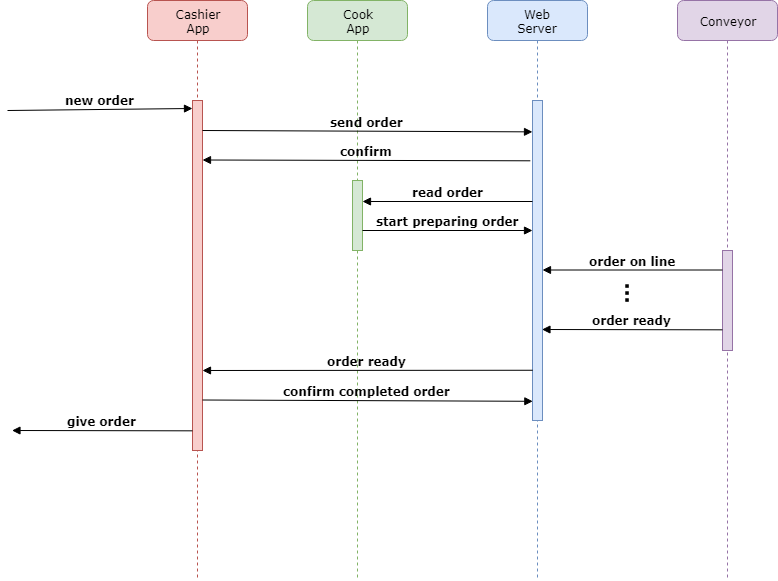
Project Description

Our solution to all of these problems is developing a system to manage and keep track of all orders. The overall architecture looks like this:



We have two distinct applications for the cashier and the cook, connected to our custom web server, which processes all requests and stores the orders in a database. The server also has a wired connection to the production line, which is automated to check for ready orders via sensors.

The following software detailed design schematic showcases the dynamic overview of the whole process:



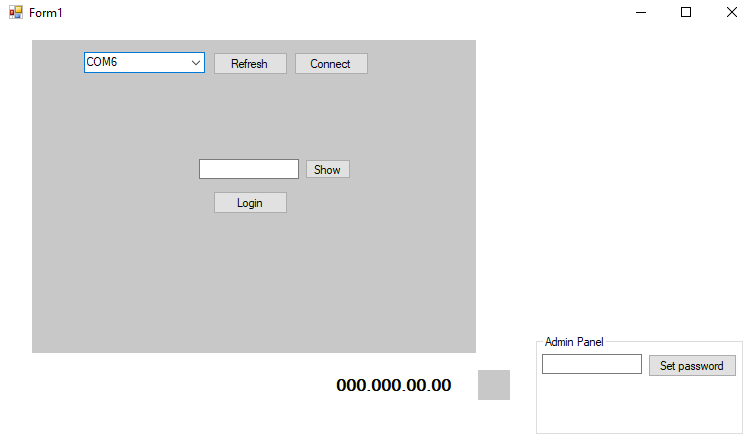
The cashier app would be able to create different orders that would be stored in the web server database. From there the cook app can read each one of them and update its status to “Being prepared”. The conveyor oven will automatically detect whenever the pizza is being baked and when the order is ready. These updates will be dynamically updated on the server, from where the cashier app will be able to show which orders are ready. This system will drastically change the workflow to be way more efficient.

This table describes each component of our system, what actions it’s responsible for and how it’s implemented:

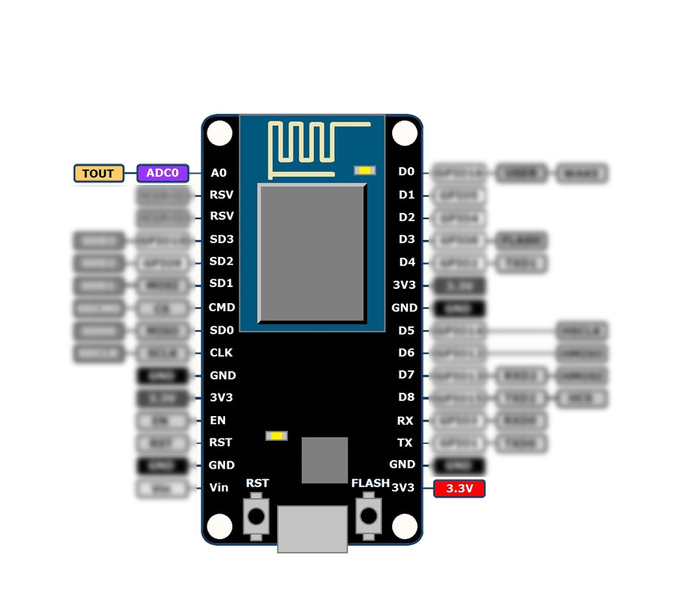
|  |  |  |
| --- | --- | --- |
| Component | Implementation | Responsible for |
| Cashier App | C# Application | * Creating orders * Finishing orders |
| Cook App | C# Application | * Reading orders * Editing orders |
| Web Server | NodeMCU ESP8266 board | * Establishing communication |
| Database | Arduino Uno | * Storing orders |
| Conveyor oven | HC-SR04 sensor | * Keeping track of orders * Editing orders |

setup and usage

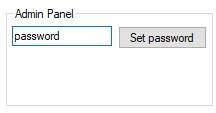
The first step is setting up the server. Connect the NodeMCU board to the computer and run the setup app. You will be presented to this screen:



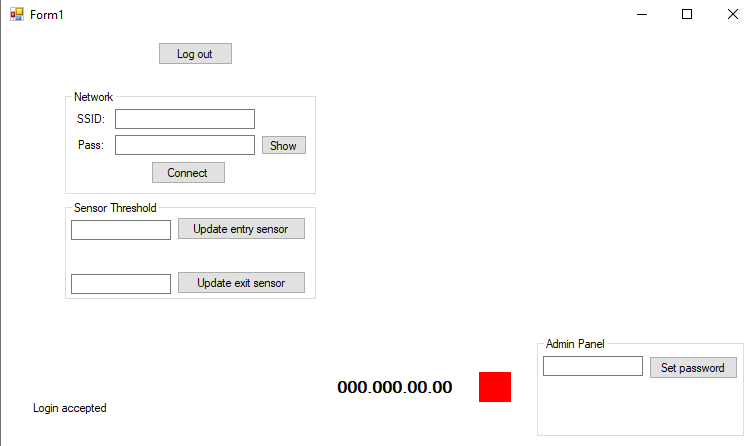
Select the USB port, connected to the board and click connect. The next step is setting up our admin password. In order to do that we need to set the server into root mode. To do that, take a cable and connect the 3V and A0 pins together. You can see where those pins are of the following diagram:



After that you can freely change the password to whatever you want by using the Admin panel.



Now you can login using the same password. Upon successful login you will be presented to the main app screen.



From here you can connect the board to the WiFi network to act as an access point using the Network section. If the connection was successful, the red panel will switch to green and the server IP address will be display on the left like so:



Additionally, you can switch the threshold of the oven sensors from here. Type in the distance in the corresponding box in cm and click the button.

The next step is wiring everything up. Firstly, we need to connect the server and the database together. To do that we will need 3 wires, connected like so:

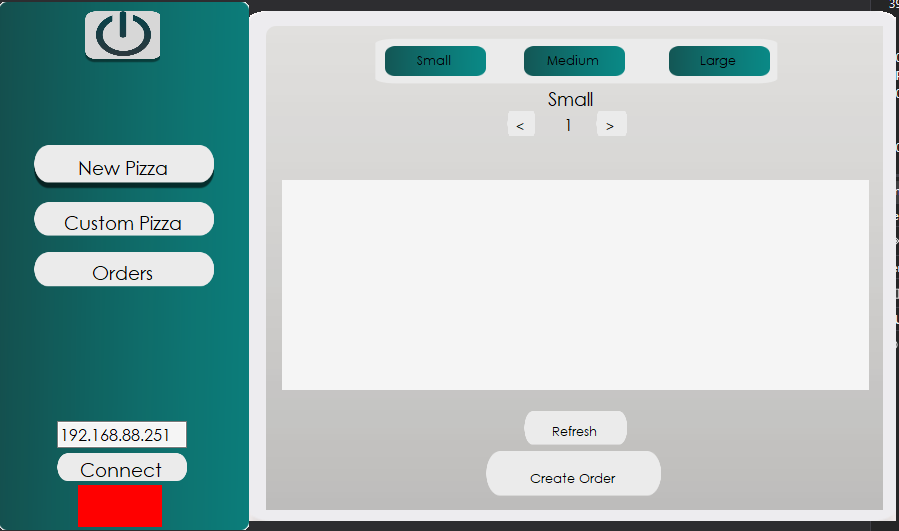
|  |  |
| --- | --- |
| Server Board | Database Board |
| Any ground pin | Any ground pin |
| D5 | 5 |
| D6 | 6 |

Lastly, after establishing the UART connection, we need to wire up the two sensors to the Arduino board. The connection in described in the following table:

|  |  |
| --- | --- |
| Arduino Uno | Sensor Pin |
| Any 5V pin | Vcc (Sensor 1) |
| 8 | Trig (Sensor 1) |
| 9 | Echo (Sensor 1) |
| Any ground pin | Gnd (Sensor 1) |
| Any 5V pin | Vcc (Sensor 2) |
| 10 | Trig (Sensor 2) |
| 11 | Echo (Sensor 2) |
| Any ground pin | Gnd (Sensor 2) |

If a buzzer is available, it can be connected to Arduino pin number 3.

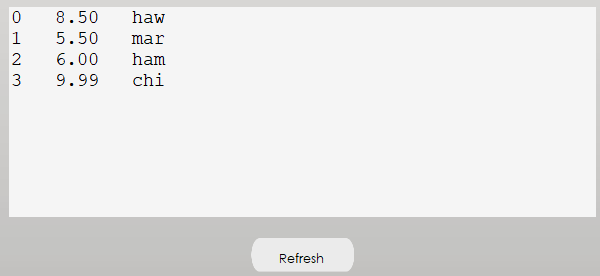
And with that the setup procedure is finished. Now we can run the cashier app which looks like this:



In order to connect the app to the server you need to write the IP address in the bottom left box and click connect. If the connection has been established the panel below will turn green.



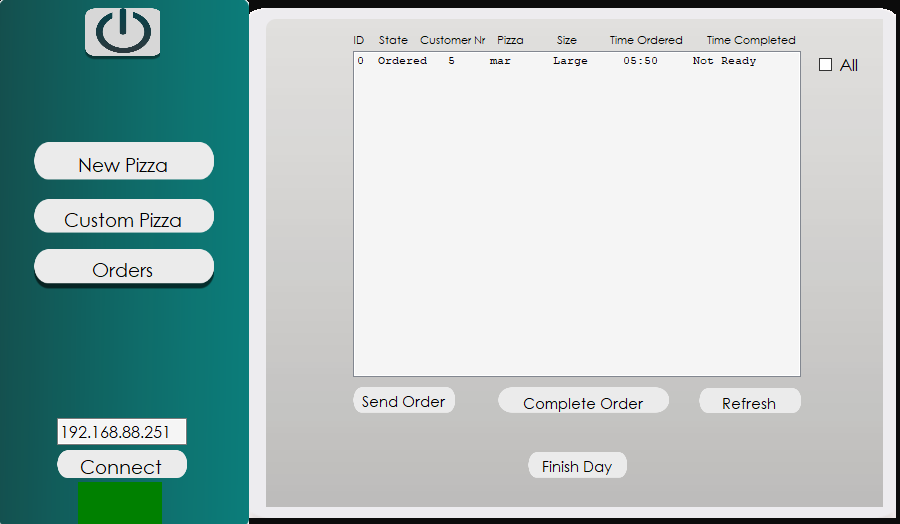
The connection itself is a simple handshake between the app and the server making sure that everything is working fine. From here when clicking the “Refresh” button the app will update its list of pizzas with the ones that are stored in the database.



Each row consists of the pizza’s ID, price and name. Note that the names are limited to 3 characters only due to the small amount of memory the Arduino board offers.

In order to create a new order, we need to select the pizza’s size from one of the top three buttons, the customer’s number and the pizza he ordered. If a customer wants more than one pizza, we need to create an order for each pizza and leave the same customer number for them all.

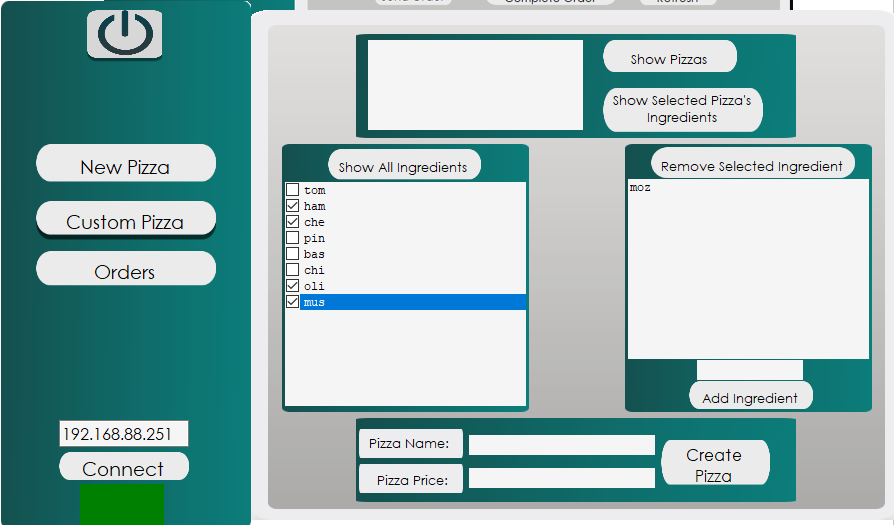
After we created a new order we can see it in the “Orders” tab. After pressing the “Refresh” button the app updates the orders list with the ones stored in the database.



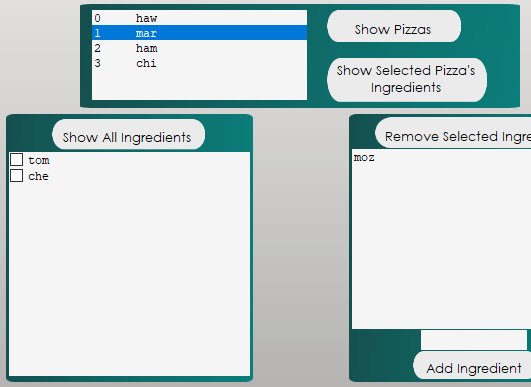
The ordered time is automatically populated with the computer’s current time. The list on itself shows only the non-finished orders unless we check the “All” box.

When a pizza exits the oven its state will have changed to “Ready”. Then the cashier can select the pizza and finalize it by clicking the “Complete order” button. If by any chance the given pizza is returned by the customer with complains you can re-send a finished order back to the kitchen by using the “Send order”. Note that this option works only for pizzas who have been completed first. When the work day is finished and we don’t need the saved orders no more we can wipe out all orders from the database by clicking the “Finish Day” button.

If the customer requires a custom pizza, one can be made using the “Custom Pizza” tab. The app can create a brand new pizza for the menu plus all of its ingredients so the cook can see them automatically.



Let’s set up the ingredients for our new pizza. Upon clicking the “Show All Ingredients” button the list below displays all distinct ingredients in the database. From here we can check the ones we want in our new pizza. If we want to add a new ingredient we can do that from the list on the right. Simply input the name of the new ingredient and click on “Add ingredient”. If a mistake has been made the new ingredient can be deleted by selecting it and pressing the “Remove Selected Ingredient”. Furthermore, we can display only the ingredients of a single pizza in the menu on the left list. By pressing the “Show Pizzas” the list above displays all the pizzas in the menu. From here we can select a single pizza and click on the “Show Selected Pizza’s Ingredients” button. This will set the left list’s ingredients to the selected pizza’s.

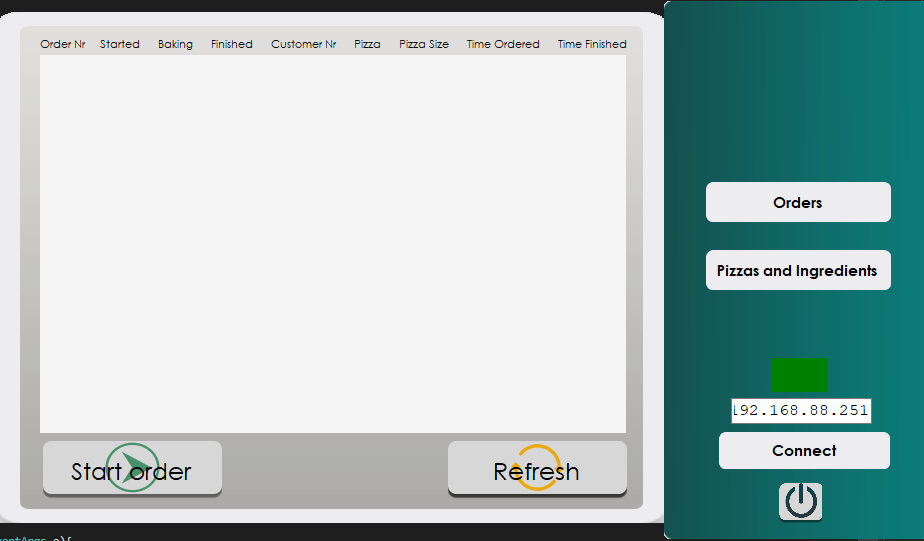


After all the ingredients we want are in the left and right list we can input a price for our pizza and a name for it in the boxes below. After that clicking on the “Create Pizza” button will put our new pizza and all of its ingredients into the database and we can select it from the “New Pizza” tab.

The application can be closed by pressing the power off button on the top left corner.

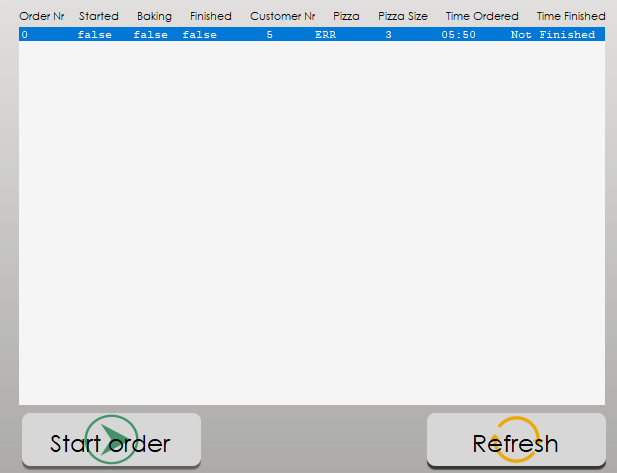


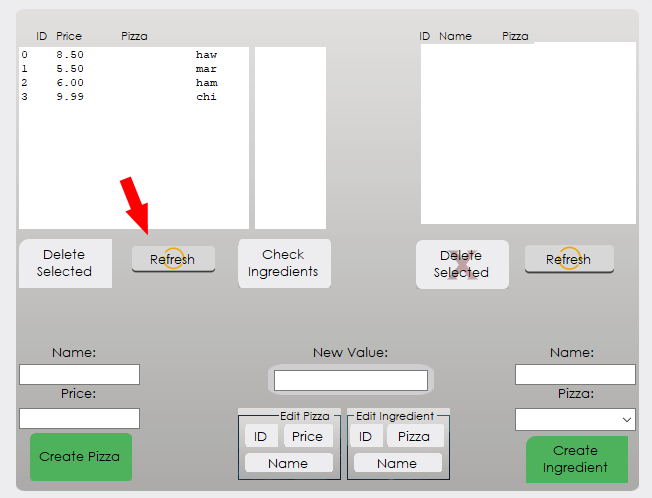
The cook application has a very similar interface.



In order to use we will need to connect it to the server the same way: by writing the server’s IP address in the bottom right text field and clicking “Connect”. From here we can refresh the orders list by pressing the “Refresh” button.

Now the list will populate with all orders from the database. If you can see they word “ERR” instead of the pizza name that means you haven’t updated your pizzas list. You can do that by opening the “Pizzas and Ingredients” tab and pressing the “Refresh” button on the pizzas list as shown below.





Now upon refreshing the orders list the correct pizza name will appear.

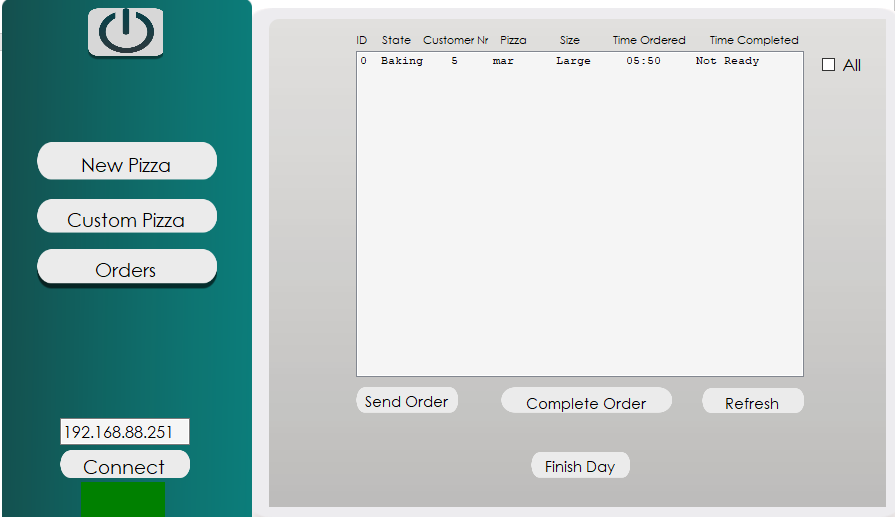


Whenever the cook wants to start preparing an order he should select the specific order from the list and click on the “Start order” button below. This will change the Started state of the order to true



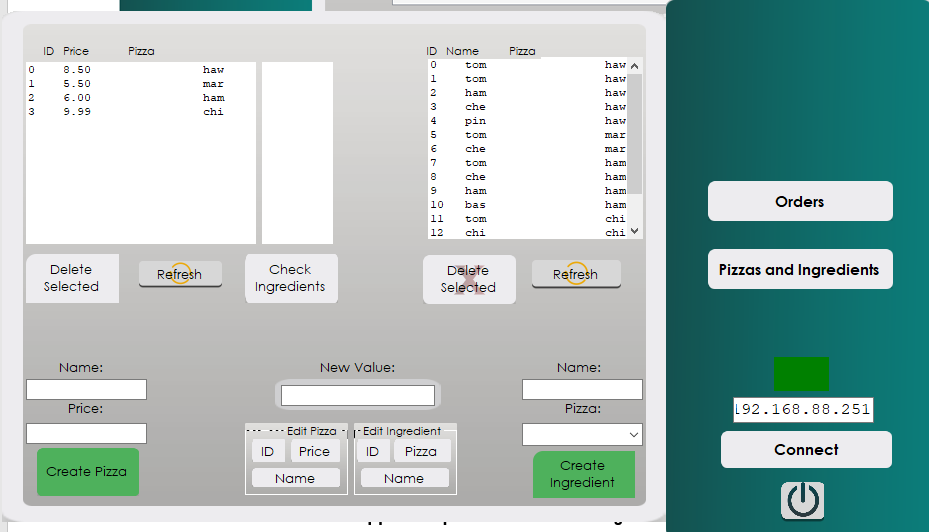
Now whenever the first oven sensor detects a pizza going in the oven it will automatically update the first started order and set its state to baking.

Note that the progress can also be seen on the cashier app



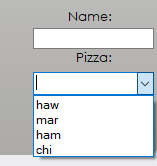
When the pizza is ready it will be detected by the second sensor which will activate the buzzer, notifying the employees that the pizza is ready and will update the state to “Finished”.

The second tab of the cook app is responsible for creating new pizzas, new ingredients and editing already existing ones.

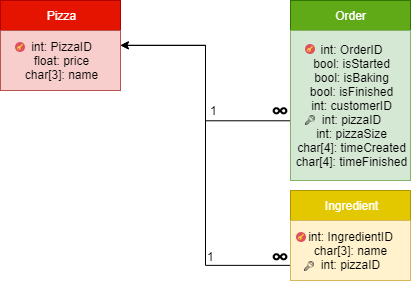


By clicking the “Refresh” buttons we can see all entries in the database. All pizzas are listed on the left, while the ingredients and their corresponding pizzas – on the right. If we select a pizza/ingredient and click on the “Delete Selected” button the selected pizza/ingredient will be removed from the database. Note that the ID numbers will be re-calculated after the deletion. If an ingredient is linked to a deleted pizza its pizza name will be empty.

If we need to create a new pizza, we can use the boxes on the bottom left corner to give it a price and a name. After that we can create the individual ingredients with the bottom right text boxes. Note that when creating an ingredient, you need to specify the pizza it’s linked to via the drop-down menu.



We can also edit each pizza/ingredient’s values using the buttons in the bottom middle. Simply select which pizza or an ingredient you want to change the value of, write the new value in the text field and press the corresponding button. Note that editing anything’s ID can break the database relations to I advise you to do it only if you are familiar with the structure of the database. For references the database looks like this:



PROBLEMS FACED DURING THE DEVELOPMENT

The biggest hurdle we faced was the absence of a team member. Due to that the amount of work each one of us had to do skyrocketed. Another difficulty we faced was an obvious communication issues and bad team planning. We had the overall idea but we never did bother to go into details and describe everything inside-out.

EVALUATION OF THE DELIVERABLE

The developed system will drastically increase the performance of the pizza show. All of the problems mentioned in the first pages were taken into consideration and a solution for each one of them has been integrated into the final product.