

Project Submission 1: Requirement Modeling

Restaurant Ordering System: Essential Business Operations: Menu browsing, placing orders, and table management.

Please brainstorm with your group members to compose a Requirement Modeling report for your project that includes the following sections:

1. System Overview (Kyla Harpe)

The restaurant ordering system is designed to create an efficient dining experience for both customers and staff. The system provides a digital platform, accessible through tablets at tables where customers can browse the menu, place orders, and request services without waiting for staff. The system manages everyday restaurant operations, including menu presentation, table management, order processing, and payment handling.

The system offers a simple menu with details such as prices, calories, as well as allergy information. Orders are directly sent to the kitchen for fast preparation and tracked in real-time with updates. The system can also provide tools that manage tables and employee assignments. Management can also use this system to track inventory, monitor sales, and generate reports for business insights. Overall, this system improves efficiency, reduces wait times, minimizes human error in stressful rushes, and enhances customer experience while giving restaurant management ease of operation as well as better control.

2. Output Requirements (Maxwell Benefield)

- The customer orders through a digital menu, probably through a website or a tablet on the table. The menu system must be fast and responsive, so the customer(s) do not have to waste time waiting for things to load or respond to their inputs. When the order is placed, a status message should be displayed, telling them that the order was successful or failed to be placed.

- Also, when a customer places an order, the ordering system must display the order in a timely manner (no more than a minute delay). This ensures that the cooks at the restaurant can prepare their meals promptly, increasing customer satisfaction as their meals arrive sooner.
- The system must produce the current table arrangement (which tables are occupied or not) at least once every 10 minutes, and every 5 minutes during peak periods. This allows waiters (or whomever puts guests to their tables) to seat customers faster without them needing to glance around the whole restaurant to find an unoccupied table. It should differentiate between tables and booths as well, as well as each table having their maximum capacity displayed to assist with different party sizes.

3. Input Requirements (Brenden Luu)

- Menu Data (Items, prices, calories, allergies, etc): This is integral to restaurants as it determines what they sell, how much they sell and a calorie counter which is an important aspect to many people. Allergies are also an important detail as being negligent in this regard can result in death.
- Table Data (Table numbers, occupied tables, servers assigned to tables, etc): Knowing what tables are available and unavailable is important to business operations and knowing if they are staffed (and by who) is also helpful to determine workload balance.
- Reservation Data (Number of people, what time, contact details, etc): This is important because if a reservation is to be made, then we must know how many people there are and what time they will arrive to arrange an adequate table. Having their contacts will be important to notify them of a ready table or to reach them if they are late.
- Employee Data (Number of employees, their roles, their wage, and their hours worked, etc): Employees are the backbone of the business and knowing their numbers lets us adapt to issues that may occur. Their roles are important because if a certain role is lacking, we can move around manpower or hire new help. Wage is the reason why employees are working so knowing this can help (or a reason for lack of work if the wage is inadequate). Hours are important because they let us know the workload

of each employee and how much they should be getting paid based on their wage.

- Payment Data (Card, cash, tips, tax, etc): This is how a company gets their money and tracking it is very important if we want to keep our employees and order supplies. Card is a just payment method that we need to be able to input to withdraw money from the bank. Cash is another payment method but much more direct. Tips are important since most businesses in the food industry do not pay their servers well and this makes up for their lacking salary. So tracking it can help us know if a server is struggling monetarily and allow us a chance to negotiate with them. Tax is important too because you do not receive all the money that is paid towards you since a portion of it is taken by the government via tax. This is important to track so we know how much money we are making.
- Inventory Data (Supplies available, supplies needed, deliveries made/need to be made, etc): Much like employees, without supplies, a business (especially food) cannot run. We need to know what supplies are available so we can order the ones that are lacking. Delivery data is important to keep track of high-quality produce deliverers and being able to contact them to buy supplies and also track the delivery to know their approximate arrival date.

4. Process Requirements (nickolasdaniel)

- a. When placing an order, the system should be able to assign that order to one of the chefs automatically at the right time before the deadline is hit. The chef must be able to handle the number of orders coming in from the system and the system must be able to accurately create a queue and tell the users the time they are likely expected to receive that meal.
- b. When managing the tables, the application must be able to accurately determine the number of tables left. It must account for any tables that were not scheduled by the system. It must take account of the people who are reserving and the amount of time they will stay there in order not to overlap tables from different times.
- c. The menus browsing must show the prices in the local currency of the restaurant, in addition tell the user if an item is out of stock, the wait times

accurately if they order an item (can depend on the busyness of the restaurant).

- d. Payment processing must accept all forms of payment and be able to process them in real time.
- e. If the app or website or server goes down, there needs to be an indicator that says it is down. It should be worked on hastily until it is up. The problematic error must be accounted for and be fixed as soon as possible. Any possible mess ups in payment must be refunded and a report of what happened, and the fix must come shortly after.
- f. The restaurant owner must be able to view, modify, update, delete/cancel, any requests and order that may appear. Any orders/request must be approved by a person before being accepted. This means that the business would need a device to get the necessary information and have an input device to send the information.
- g. All past orders/requests must be securely stored for a certain amount of time. It must be unmodifiable and analyzed/viewable in the future.
- h. The system must have some sort of analytics, and error reporting. It must show every instance of orders/requests per restaurant. System errors must always be actively monitored in case anything goes wrong with the server.
- i. The system's code must have some sort of distributed version control system so every snapshot could be backtracked. This would allow multiple engineers to contribute to the project and avoid merger conflicts. In addition, being able to track who wrote each line of code.
- j. The system must implement some sort of software development and integration, where it would streamline processes, such as automating deployment. It could be deploying a new version of software in the cloud or deploying it locally on a server. This would significantly reduce the amount of time it takes to develop and fix bugs on the platform.
- k. The application would need to be tested every time it is deployed, or code is merged with the main application. It would include both manual and automated tests. For the manual testing, it would require a testing team where their whole goal is to try to break the code or see if there are any errors in the application and give criticism of what went wrong. For the automated tests, this would include writing even more code to test certain parts of the

application. Some include unit testing where it verifies every individual unit/function of code runs correctly, which can lead to early bug detection. Integration testing where it tests if individual models/inputs are functioning correctly as a group, detecting faulty interfaces. Next is regression testing where any new code as a result does not introduce new unexpected features of the program.

- l. The development times must be in reason. The task of developing the software must be broken into small parts so that multiple developers may be able to work on it concurrently. The spiral model will be used for the development of mythology. It divides projects into iterative cycles or "spirals," each containing four main phases: Planning, Risk Analysis, Engineering, and Evaluation, where each cycle builds on the previous one, incorporates feedback, and systematically addresses potential risks to produce evolving versions of the software. This approach would be effective for this project as it is meant for large and complex projects where the risks are high and requirements may be uncertain.
 - m. SEO or search engine optimization where it is the practice of improving rankability on search engines like google, which would lead to more public visibility. This would include making the application adhere to the best practices for search engines, such as improving content quality, identifying keywords and boosting credibility.
 - n. The code structure of the project should be consistent, maintain readability and stability as being very important to maintenance. The code must have lots of good comments explaining what/why the code does. In addition, it must have thorough documentation on every interface/function/class explaining its inner workings.
5. Performance Requirements (NickolasDaniel)
- a. The application should be able to handle many types of performance testing, including load testing where it would simulate heavy user interactions, to identify any bottlenecks. Stress testing where the load would be more than what the hardware can handle, it would be great for testing the queueing/dropping system and finding unexpected behavior. Endurance testing where the app would experience heavy loads over a lot of time,

potentially finding any memory leaks and performance degradation. Scalability testing where it tests the app's ability to scale up and down depending on the resources.

- b. The application should have a good response time; this could include having efficient, low sized, compressed assets, having a good system of caching and lazy loading to make it efficient. The servers should be close enough to the business to limit the amount of delay involved. The response time should be minimal and the time it takes to get the information should be fast.
- c. The server itself should be conservative about resource utilization; it should not use unnecessary processes, to save money and to preserve response times.
- d. The throughput for the server should be at least 100Mbps for the small end and 100 Gbps for the high end. The throughput should not need to form a queue.
- e. The app should be very concurrent, meaning it can handle many responses at the same time. Having good competition would mean dealing with race conduction properly but would introduce better performance.
- f. The app should have rate limiting to reduce the attack threat of denial of service. This will include being able to block IP addresses if they exceed the rate limit. In addition to having a reCAPTCHA to prevent bot scraping.

6. Security Requirements (Brenden Luu)

- RBAC (Role-based access control): It limits what data staff can see and change depending on their roles, so a low-level employee cannot inhibit high level functions of say a manager.
- Authentication of Employee Accounts: Reduces the risk of accounts being accessed by the wrong people and making impactful changes to stuff like menus and prices.
- TSL 1.2+: Prevents the potential for a person with malicious intent to harm others through guest Wi-Fi or devices connected to the backend.
- Encryption: In case of an outside device accessing internal documents, what they can see is limited.
- Fraud/Refund controls: Prevents the business from being taken advantage of or scammed.

- Alert System: In case an attack happens, or a system fails, someone that can fix the issue is alerted as soon as possible.
- Backup System: In case a system becomes inoperable for an indefinite amount of time, having a backup will always be useful.
- Segmented Network: If a network were to be compromised, it would only be a small segment instead of the whole network.
- Consent/Privacy Notices: Help to build and keep trust while letting the recipient know the details of what they are signing up for.
- Dual Control: For actions that are risky or expensive, there must be two higher ups that agree for it to happen (like nuclear launch codes).

7. Conclusion (NickolasDaniel)

This application is a secure, robust and efficient way for restaurant managers/owners to modernize their restaurant reservation and ordering system. The platform gives customers the ability to browse menus, place orders and request reservations fully online and automatic. It provides the tools to owners to modify/add services to their operation, such as tracking inventory, monitoring sales and generating reports. The ordering system gives flexibility and transparency to both the customer and the workers; it supports flexible payment types, when they are expected to receive the order, and their delivery method or a reservation. The online menu feature gives customers lots of data, such as prices, calories, allergies, if it is in stock and how long it takes to prepare. It also handles orders for the cooks; it does the job of automatically assigning orders at the right time before the deadline, if an order is problematic then it can be altered or refunded. Our application is very robust with a team of software engineers who are on call to fix any outages that may occur, it has been tested to be able to handle any stress test. Lastly it has the latest security implementation, such as having Role-based access control per employee, regular backups, TSL encryption, fraud control, and alert system.