JGi JAIN DEEMED-TO-BE UNIVERSITY | FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

"Restaurant Automation"

By:-

RAJAT SURANA (17BTRCS053)
TANISHQ DALAL(17BTRCS055)
KAUSTUV GIRI (17BTRCS070)

Table of Contents

1. INTRODUCTION

The goal for this project is to introduce automation in privately-owned restaurants, that is, small- to medium-sized establishments. Typical problems restaurant personnel are facing include: Coordination of their work activities Anticipating and handling periods of low/high patron traffic

Recognizing trends early enough to take advantage of bestsellers or abandon the flops. Lowering operating costs, and increasing efficiency/productivity and profits Many restaurants are still operated using pen and paper methods, with little or no automation. Patrons enter the facility to be greeted by a host, who often times has a “dry erase” diagram of the tables, maintained on a blackboard. The host can see the status of the tables based on whether or not they or someone else physically updates the diagram. Once seated a waiter tends to the costumers by noting down the orders onto a piece of carbon paper and delivers it to the kitchen for proper food preparation. The waiter then has to periodically check back to find out when the meal is ready. When the food is done, the piece of carbon paper is saved for proper record keeping by the management. This “old fashion” system works but yields a large amount of tab receipts, wastes a lot of time and is simply out-of-date. In old fashion systems, waiters have to carry pads around to take orders, always have a working pen and be sure to keep each bill organized and “synchronized” with the proper table.

Another issue is record maintenance. In the old system, when everything is done by paper, the management is responsible to keep all information saved and organized, which is no easy task. Everyday tabs are collected, data needs to be organized and employees need to get paid. This requires a great deal of time and attention from the managers. This project computerizes restaurant operation so that all information pertaining to patron’s orders and staff activity will be conveniently shared and stored over the restaurant’s intranet. Hosts will be able to view table status with a click of a button. The wait staff will be able to enter the patron’s orders quickly and efficiently and then have it electronically delivered to the kitchen. The kitchen staff will be able to view the incoming orders and notify the proper wait staff when the food is ready. Bus boys will be able to view real-time floor status allowing them to know which tables are clean, dirty, or occupied. Most importantly, all of the restaurant information is organized and saved in the system database for the management viewing and archival. The analysis will consist of by-the-day and by-the-hour breakdowns. All data is automatically collected and processed allowing management to focus on analyzing the data rather than calculating it.

2. PROBLEM DESCRIPTION

Chef: Working in a kitchen can be strenuous(energetic) and it can be difficult to keep track of everything that needs to be done. Orders need to be filled in a reasonable amount of time while simultaneously ensuring that any dish accommodations are accounted for. This needs to be done with accuracy or else an order could be made incorrectly or even worse, a person with a food allergy could get hurt as a result of a waiter's poor handwriting. We want to get our orders out with accuracy and speed to make sure we have a happy returning customer. When dishes are complete and the waiter does not pick it up for a few minutes, it is a pain because the food gets colder and it takes up space on the counter that could be used otherwise. Also, at the end of a busy day of serving hundreds of customers the inventory can be depleted to the point of needing restocking and it's difficult to keep track of what's expired and what chef need more of. Chef have so many responsibilities that it can be difficult to keep track of everything in the pantry. Chef have to make sure that the expired ingredients are cycled out and that the low stock ingredients are replenished in time for the upcoming week. It would be helpful to have a tool that would track the expiration dates of food to make sure everything is fresh.
It would be a great help if chef had something that clearly displays all incoming orders and the modifications that have been requested by the customers. Also, if multiple dishes come in with the same ingredients, it will be helpful if the chef was notified to make more of a certain ingredient at one time versus cooking a certain part of a dish twice. This can save more time in the kitchen and make the cooking process more efficient. Finally, a way to signal to the waitresses and waiters that the order is finished and ready to be taken to the table to reduce the time the food sits on the counter.

Host/Hostess: Whenever while welcoming customers into restaurant,they aim to make the start of their experience as flawless as possible. As parties of multiple people arrive, they tend to have different requests and requirements based on how many people they have and where they want to sit. Sometimes can have a party of 8 looking for a round table in a corner, where as other times can get a family of 4 wanting a booth.Host/Hostess love to find them their preferred seating, but sometimes there just isn't an available spot, and host/hostess aren't really sure how long it will be for an appropriate spot to open up.
Finding an exceptional table for guests can sometimes be a challenge, especially when host/hostess unsure of which tables are ready to go. When host/hostess welcoming customers in, it would be great if there was a way to easily keep track of which tables are currently cleaned and empty and which are occupied, without having to go through endless reams of paper. At the same time, it'd be

nice to keep track of how long each table has already been occupied. This will be a big help in providing guests with accurate estimations on when a valid seating arrangement may become available.
How will project help?
Using a handy tablet/phone they can keep it with them, can mark tables as occupied, vacant, clean, etc. as the day goes on. Whenever they seat new guests, can mark the table as occupied. This starts a timer that let them know how long it has been since guests have arrived there. Once the guests leave, that table can be marked as vacant, notifying a busboy that the table is ready to be cleaned. Once the busboy is finished,busboy can mark it as cleaned which let them know that the table is ready to be used once again.

Customer (Restaurant Patron): They love going out to eat, one thing they don’t love is how slow things can be. When they seated, it’d be really great if they could place an order for drinks or appetizers without waiting for a waiter to get to patron, especially if the restaurant is incredibly busy this could really cut down on time spent waiting. Even if patron have a special request immediately after their initial order or need to notify the waiter for anything, it can be rough getting someone’s attention.It’s usually very difficult to get their attention unless they walk right past near the dining of patron, and if no one ever walks by,patron might be waiting extended periods of time before anyone notices them.When this hasn’t been properly planned it can be difficult to figure out how to fix it.
How will project help? I think a tablet on the table with a menu that patron party could use to place orders immediately after being seated would be a great idea. It would also be a great way to have more interactive menus that could give patron’s more information on each item, since if they have any questions, usually they have to wait for a server to come answer. A great solution to waiting for a server to walk by to help them, would be the ability to press a button in an app to request a server visits patron’s table.

Waiter/Waitress: Working as a waiter/waitress is an extremely demanding job. Servers have to run around the restaurant taking customers orders, keeping track of which table ordered what, and returning orders. Servers have to interact with chefs/cooks and get food to people right away so the food does not get cold. They also have to enter checks when a customer or group is ready to pay. Since there are many tables in this restaurant, it is somewhat difficult to connect a certain order to a certain table, and also to determine whether a table needs to be cleaned and set up for the next customer. Unfortunately, sometimes a waiter may have to send a dish back if the customer does not like it or finds something wrong. The software should be able to account for this.

There are many ways in which a computer software can help make servers' lives easier everyday at work. A software suite could enter all the orders in a list of the servers', as well as number each table in this restaurant and match it to a given order. Additionally, when the customers are done using the table, the software would mark the table "not clean" and so the waiters will clean it or send the busboy to clean it. If the customers need anything from servers, they can also have an option to send them a ping which will come through as a push notification to server's device, allowing them to know they need service. Server and chef coordination are also very important. The chef should be able to enter a time of completing a dish, and the server should be able to see how much time the dish has been out. The software then sends a reminder to the waiter/waitress to serve an order. Lastly, the system could automatically take payments and tips.

Managers: Managing a restaurant and a full staff of employees is no easy task, there are many different things that they have to do throughout their day and it can be a little overwhelming at times.They really hoping that project will be able to help them with ensuring that the restaurant is adequately stocked at all times by actively keeping track of inventory since the counts provided by the busboys never actually lineup with when they run out of supplies and therefore they constantly guessing at when the proper time to restock is which can lead to scrambling around at the last second to fulfill a customer request.
There are also difficulties with figuring out how many employees manager need to schedule for a given day and time, it seems that over the weekends restaurant generally need more staff, but they themself not sure if there are trends throughout the week and some days are busier than others because they do sometimes find that restaurant is understaffed, but manager cannot seem to find the proper balance of workers to work. Another issue with employees is keeping track of them, like whether or not they made it to their shift on time or are in the building and calculating how much to compensate the employees for their time. Any way to reduce the amount of effort that manager have to put into any of these tasks would be immensely helpful.

How will project help? Utilizing the Android application once the order has been placed via the Android device that has the software on it, the ingredients used to cook the meal will automatically be deducted from inventory so that inventory is always up to date and accurate, if the user has a manager account then they may input a threshold value for items in inventory, and when supplies fall below that threshold a notification will be sent to the manager, informing the manager that they need to pick up more supplies. In addition to this, the application will track the traffic in the restaurant allowing management to determine if there are any trends with when people eat at the restaurant so that they are not over or under staffed. It also serves as an employee portal, allowing them to clock in and out, log break time, and will calculate the proper amount to provide to them for compensation. The application will also verify the employees' location based on IP address or GPS location so that employees can only clock in while they are on site

3. SYSTEM REQUIREMENTS

3.1 Stakeholders

There are many stakeholders who have an interest in this system, and ultimately its success:
i.Restaurant Owners - Have a direct interest in using the system as it will help optimize efficiency in the restaurant and provide better service for patrons.
ii.Employees - Have an an interest in the system since it will result in a more streamlined process while working making their job much easier.
iii.Restaurant Visitors - The system will result in an enriched dining experience for the restaurant visitor, as they will be interacting with and using the system.
iv. Developers - Have an interest in working to design and implement solutions to create the system.

3.2 Actors and Goals

Initiating Actors
Actor Role Goal
Customer The customer is a restaurant visitor who may choose to dine in or take-out food, view the menu, order a meal, eat, and pay for service.
The goal of the customer is to have an excellent dining experience with minimal wait times and smooth service.

Guest The guest is a customer that does not have an account and chooses to opt out of the feature that come with an account, other than that a guest is the same of a customer.
The goal of the guest is to have an excellent dining experience with minimal wait times and smooth service.
Employee The employee is any type of worker at the restaurant, except for the manager.
The goal of the employee is to provide an excellent dining experience for the customers.

Manager The manager is the employee who has the additional responsibility of managing all needs of the restaurant.
The goal of the manager is to manage employees and scheduling, keeping track of inventory, and monitoring revenue and losses while they also ensure that restaurant customers are accounted for.

Bartender The bartender is the employee responsible for preparing and mixing drinks that are ordered by a customer. The bartender receives a queue of customer orders, and begins mixing the drinks accordingly. Once complete, the drink is brought to the customer's table via the waiter.
Busboy The busboy is the employee responsible for cleaning the dishes and tables, and maintaining overall cleanliness of the restaurant. The busboy receives notifications from the database that customers have left their table, marking the table as dirty. Once the table is cleaned, it will be marked as clean.

Chef The chef is the employee responsible for cooking and preparing the food that is ordered by a customer. The chef receives a queue of orders, preparing them in the order they come in. The customer is updated on the status of their order (submitted, preparing, ready).

Database The database is a system that records a customer's order, reservation/table selection, and menu ratings. It essentially acts as the server in which the application receives most of its information for automation.
Host/Hostess The host/hostess is the employee responsible for greeting incoming customers and assigning seats to them. In the event that the guests have already reserved a table, the host/hostess will escort them to the table and marked it as occupied. The host/hostess receives notifications when a table is marked as clean.

Waiter/Waitress The waiter/waitress is the employee responsible for taking orders from customers and sending them to the kitchen queue, as well as serving the food when it is ready. The waiter/waitress receives notifications that a meal is ready, so that they can pick it up and serve it. They are also notified which table number to serve it to, and get notified when a customer needs additional assistance.

3.3 Hardware requirements

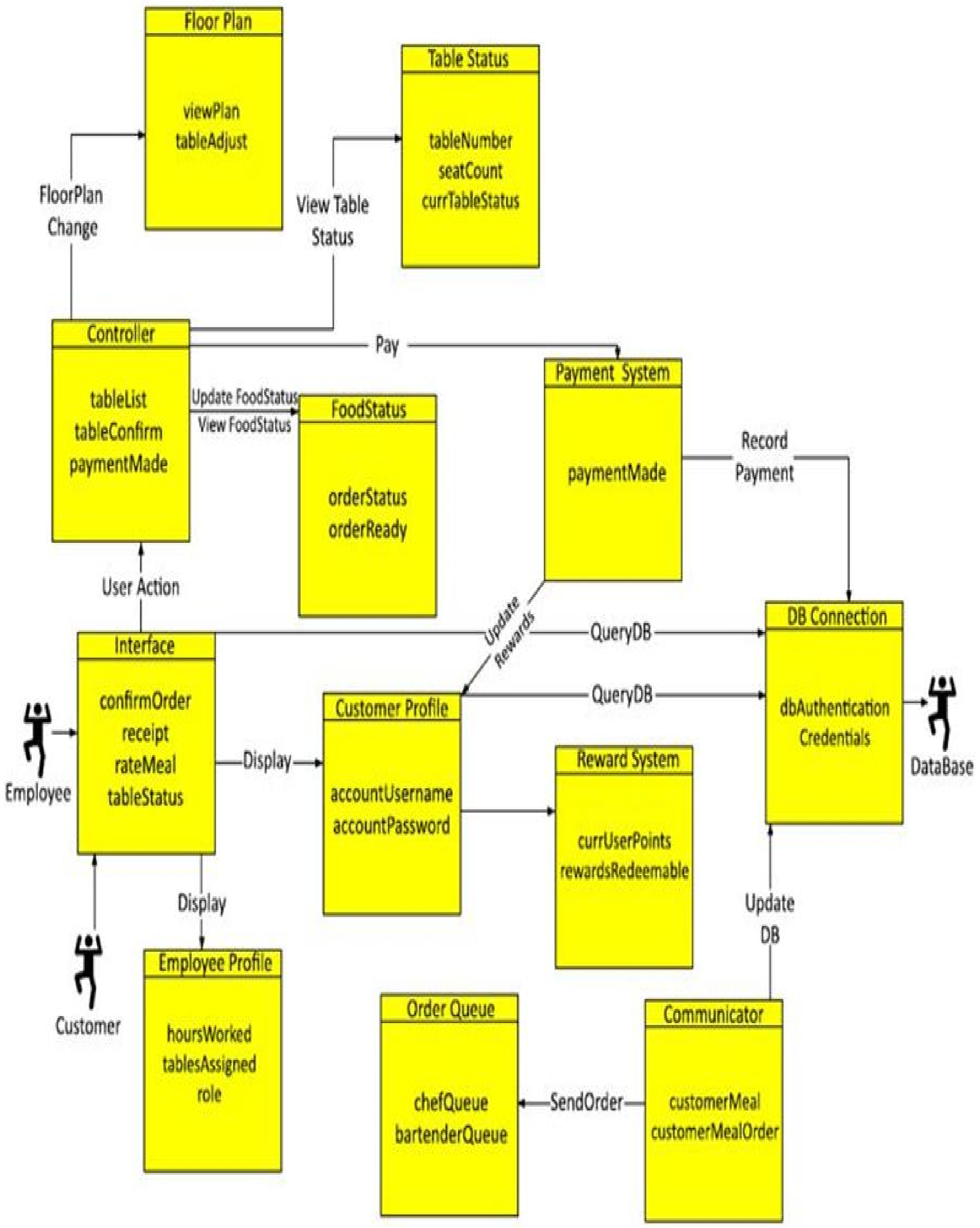
- Intel i3 or higher
- 4 GB RAM

- 1 GB Hard Disk space

3.4 Software Requirements

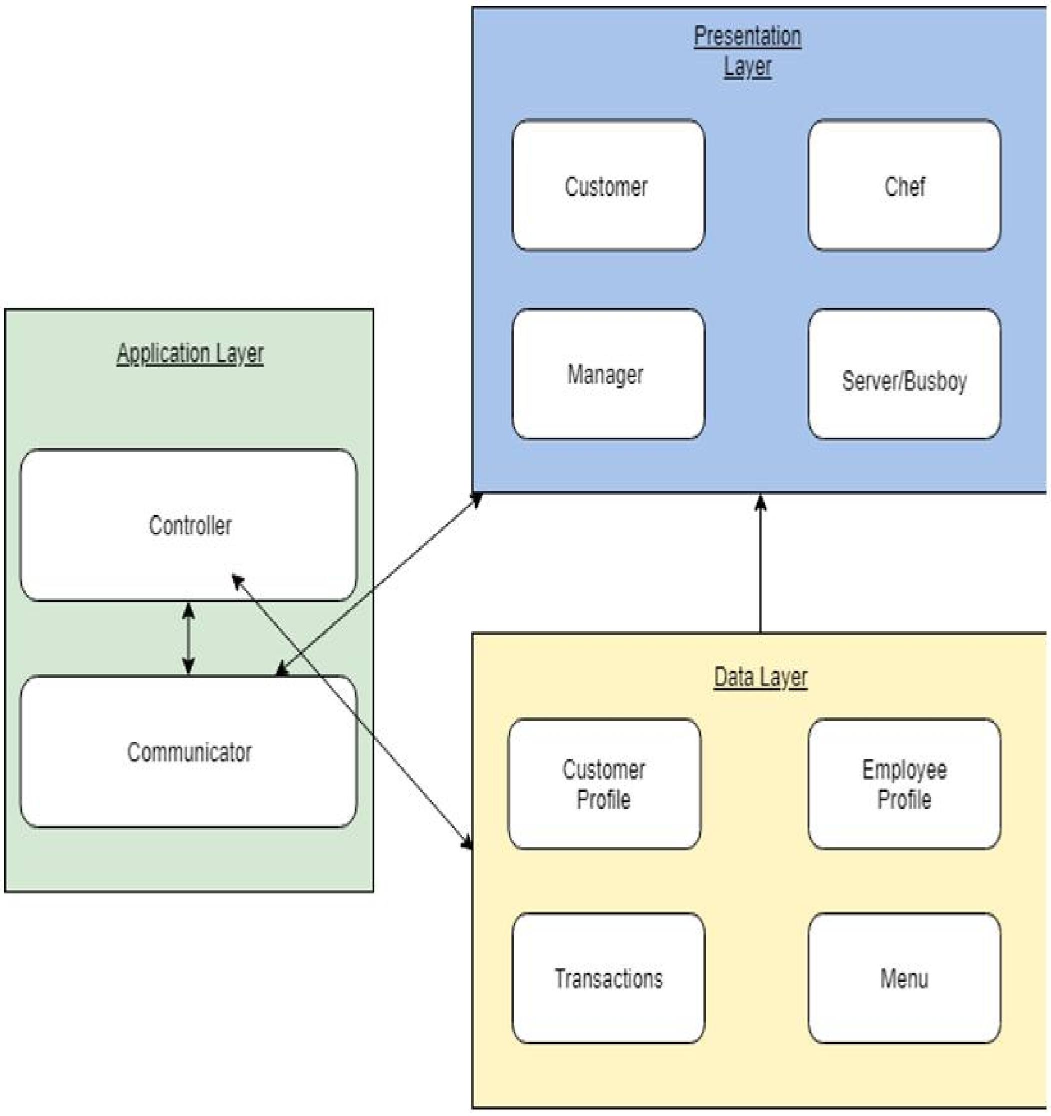
- NodeJS
- MongoDB
- XML
- Java
- Web Browser
- Android App
- Operating System

4. MODEL DIAGRAM



This diagram is a visual representation of the domain concepts in our project and how they interact with one another. Each individual domain concept was previously explained in the above sections.

4.1 UML DIAGRAM



The subsystem above displays our three-layer architecture - which consists of a Presentation Layer, an Application Layer, and a Data Layer. Each layer has a specific responsibility to the entire system. The presentation layer handles the user interface level. This layer holds the information dealing with the Bartender/Chef, Manager, Busboy/Waiter/Waitress, and Customer Interfaces. The application layer is the middle tier of this architecture, which handles communication between the top and bottom layers, essentially managing the overall operation. It will run the business logic, which is the set of rules required for running the application for the guidelines given by the organization. Contained within this layer is the Communicator and Controller. The Controller is utilized to facilitate tasks between the layers and the Communicator is used to pass information between the layers. The lowest layer is the data layer, which deals with the storage and retrieval of data. This tier will hold information pertaining to the Employee and Customer Profiles, as well as the available menu and restaurant transactions

4.2 INTERACTION DIAGRAM

PAYMENT

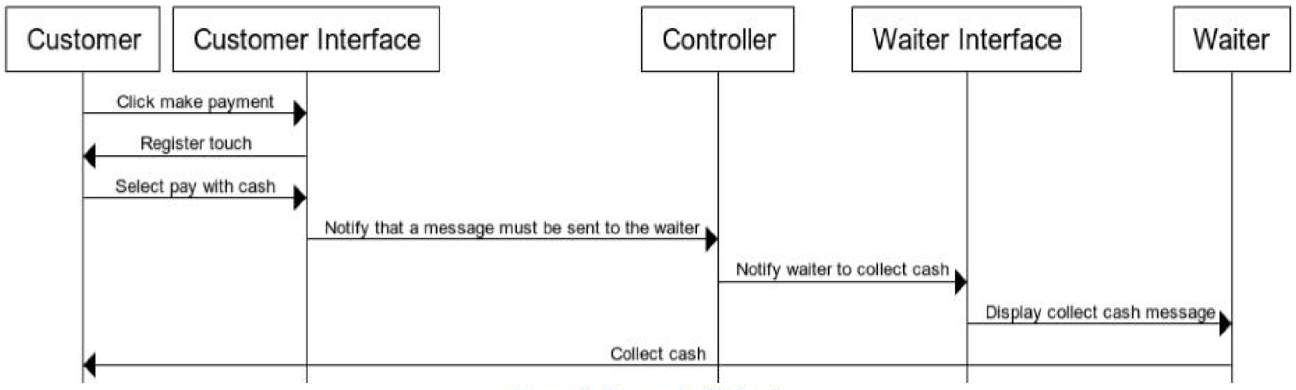
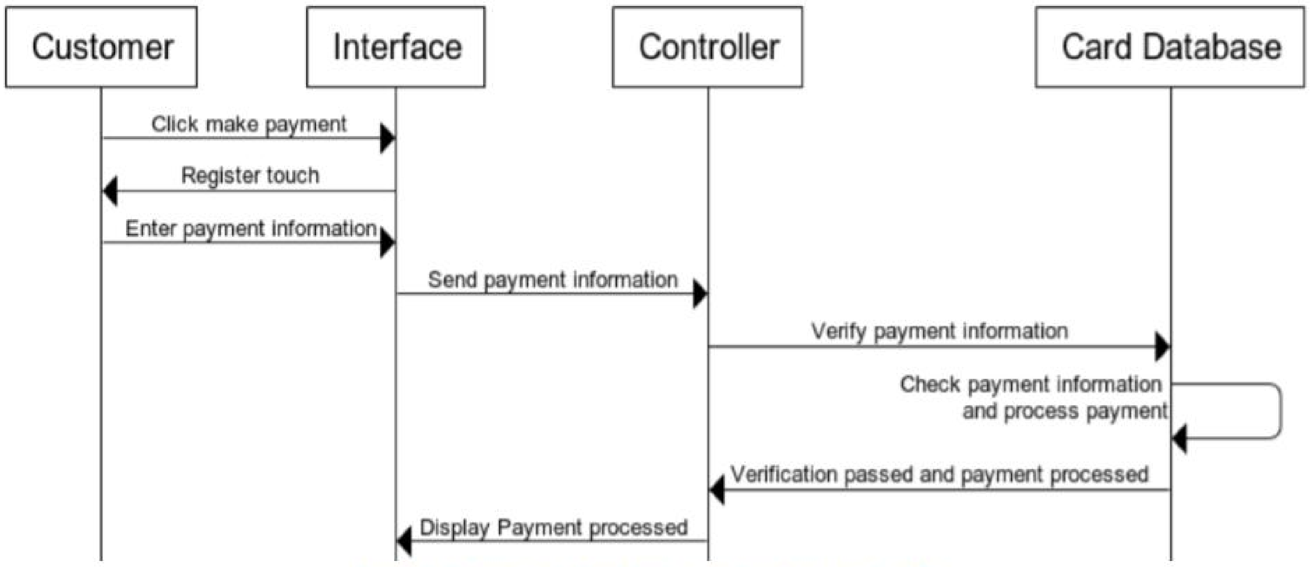


Figure 1: Payment with Cash



Customer
Interface
Controller
Card Database
Click make payment
Register touch
Enter payment information
Send payment information
Verify payment information
Check payment information and process payment
Verification passed and payment processed
Display Payment processed

Figure 2: Payment with Card Verification Successful on First Try

These sequence diagrams display some of the potential paths that could be taken when attempting to pay utilizing the TurboYums payment processing. The second diagram shows a best case scenario while attempting to pay for a meal with a credit or debit card, after the customer interacts with the interface and enters their payment information, the information will be sent to the controller, which will then send the data to the specific database for the card (for e.g.: VISA, American Express, MasterCard etc.) in order to verify that the information sent is in fact a valid. The second diagram shows this process running smoothly with valid card information being entered on the first try. The third diagram shows a similar process but with invalid card information being entered, which results in a loop until the user enters valid information that may be used for payment.

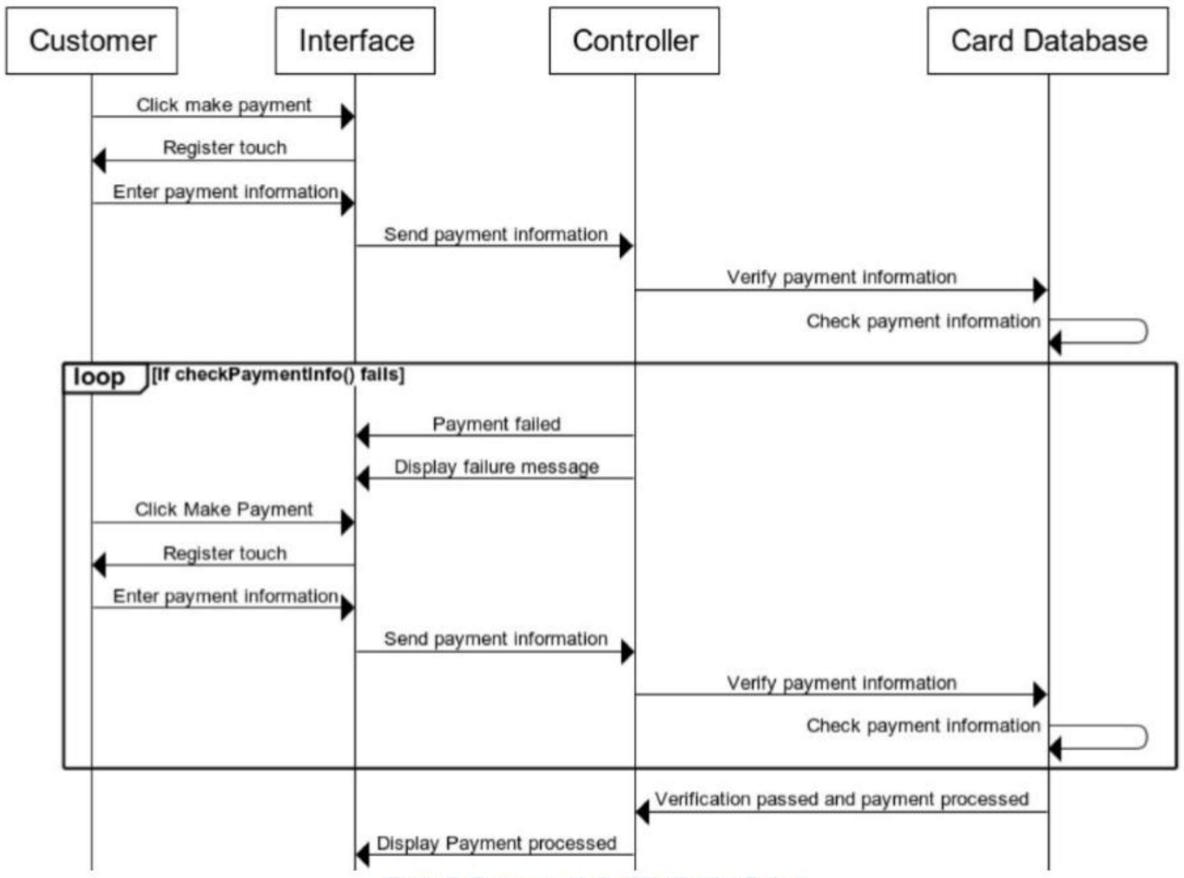

Customer
Interface
Controller
Card Database
Click make payment
Register touch
Enter payment information
Send payment information
Verify payment information
Check payment information
loop
[If checkPaymentInfo() fails]
Payment failed
Display failure message
Click Make Payment
Register touch
Enter payment information
Send payment information
Verify payment information
Check payment information
Verification passed and payment processed
Display Payment processed

Figure 3: Payment with Card Verification Failure

The first diagram shows what would happen if a customer attempts to pay with cash, which results in additional involvement of a waiter since a waiter must be notified of the payment in order to pick the cash up off of the table. The high cohesion and low coupling principles were heavily utilized in various different locations while creating the diagrams. The employment of a controller utilizes the high cohesion principle because it allows many of the other objects, such as the interfaces and database to perform their intended tasks (interacting with the user and checking the card information) without having to send messages to other objects involved in the diagram about the status of their actions. Instead of the interface sending the card information to the database, or the customer interface sending information to the waiter interface that cash must be received, the interfaces send this information to the controller so that the controller may then perform the appropriate tasks as needed. The low coupling principle was largely utilized in the cash payment process in order to keep things quick and clean. Information is telephoned down the line from one object to another, through small connections and links, especially between the controller, waiter interface and waiter, this end of the diagram is mostly singly linked objects.

VIEW MENU

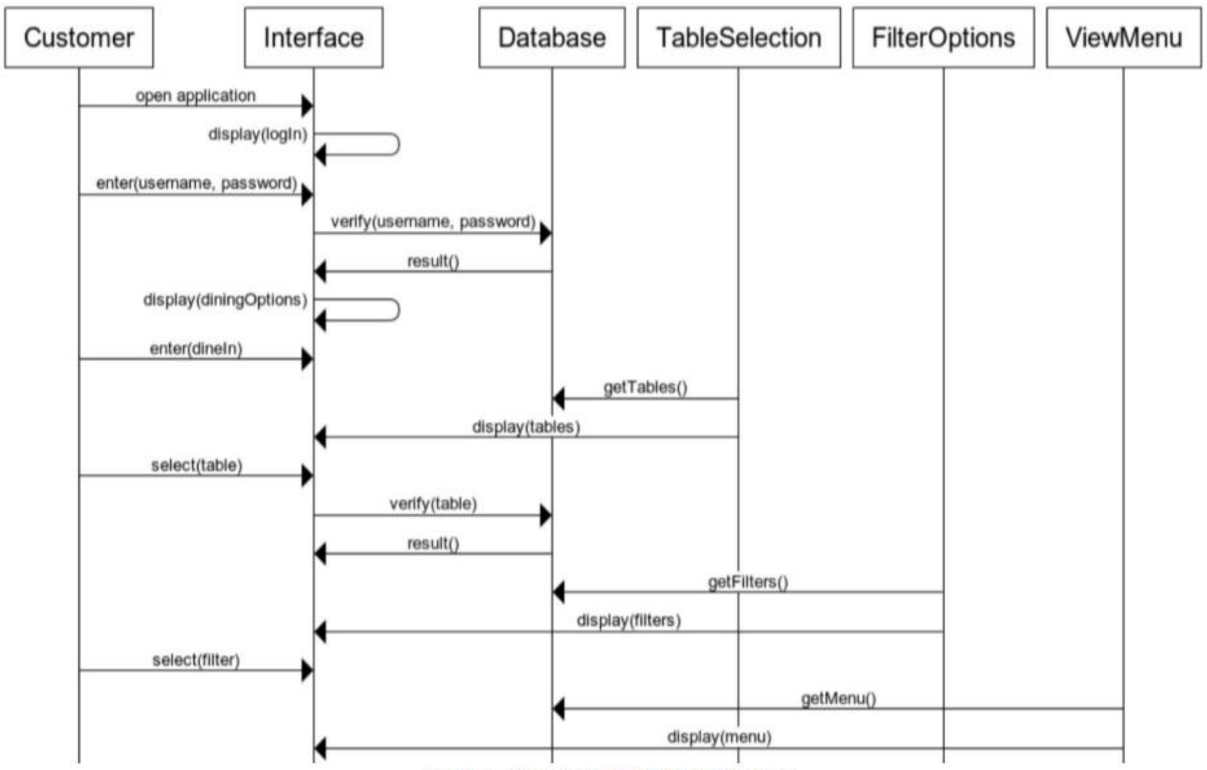


Figure 4: View Menu with Dine-In Selection

One design principle employed in this sequence is the High Cohesion Principle. The High Cohesion Principle says that an object should not take on too many computational responsibilities. This principle is utilized in that each class only takes on responsibilities that have to do with its specific functionalities and doesn't take on more than it can handle at a time. Another design principle used was the Expert Doer Principle, which says that each class is an expert in a specific function. This is shown in the diagram because, as an example, the only function for TableSelection is simply to display the table layout and have the guest select a table. After this function is over, responsibility is passed on to FilterOptions, where the sequence continues on.

CLOCKING IN/OUT

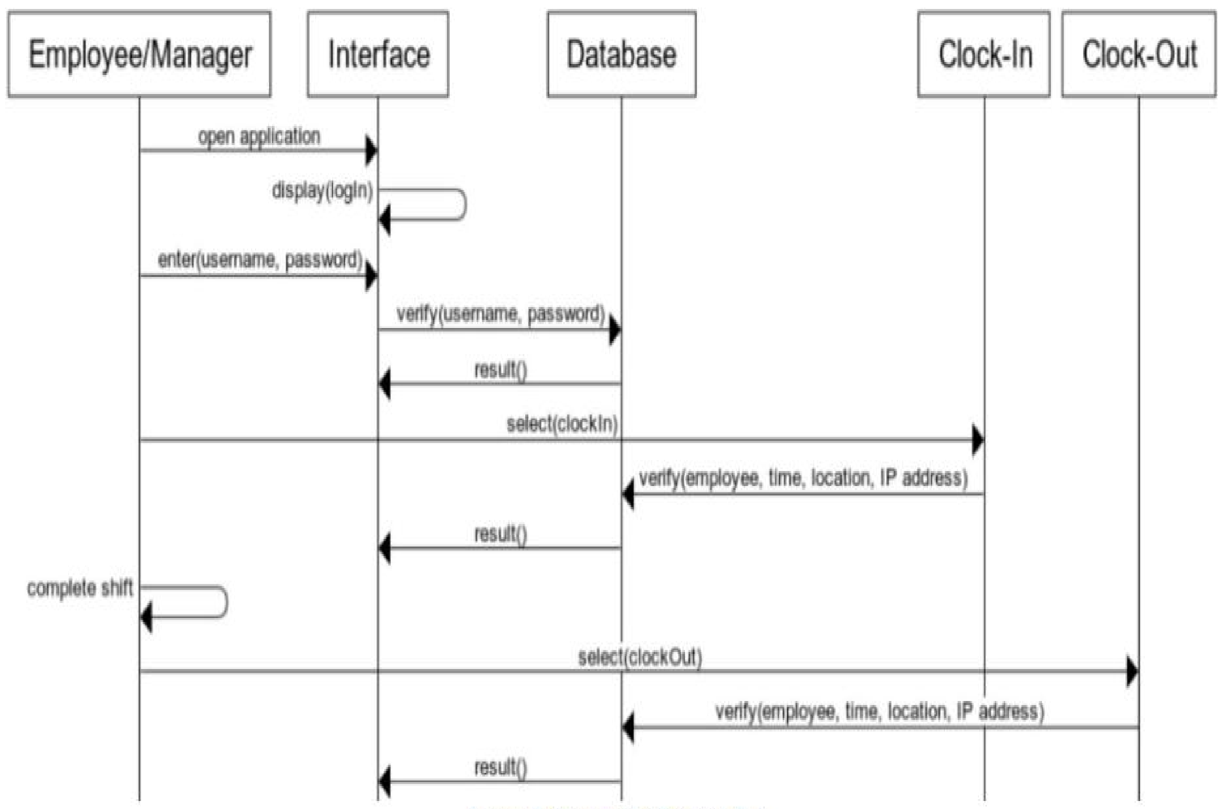
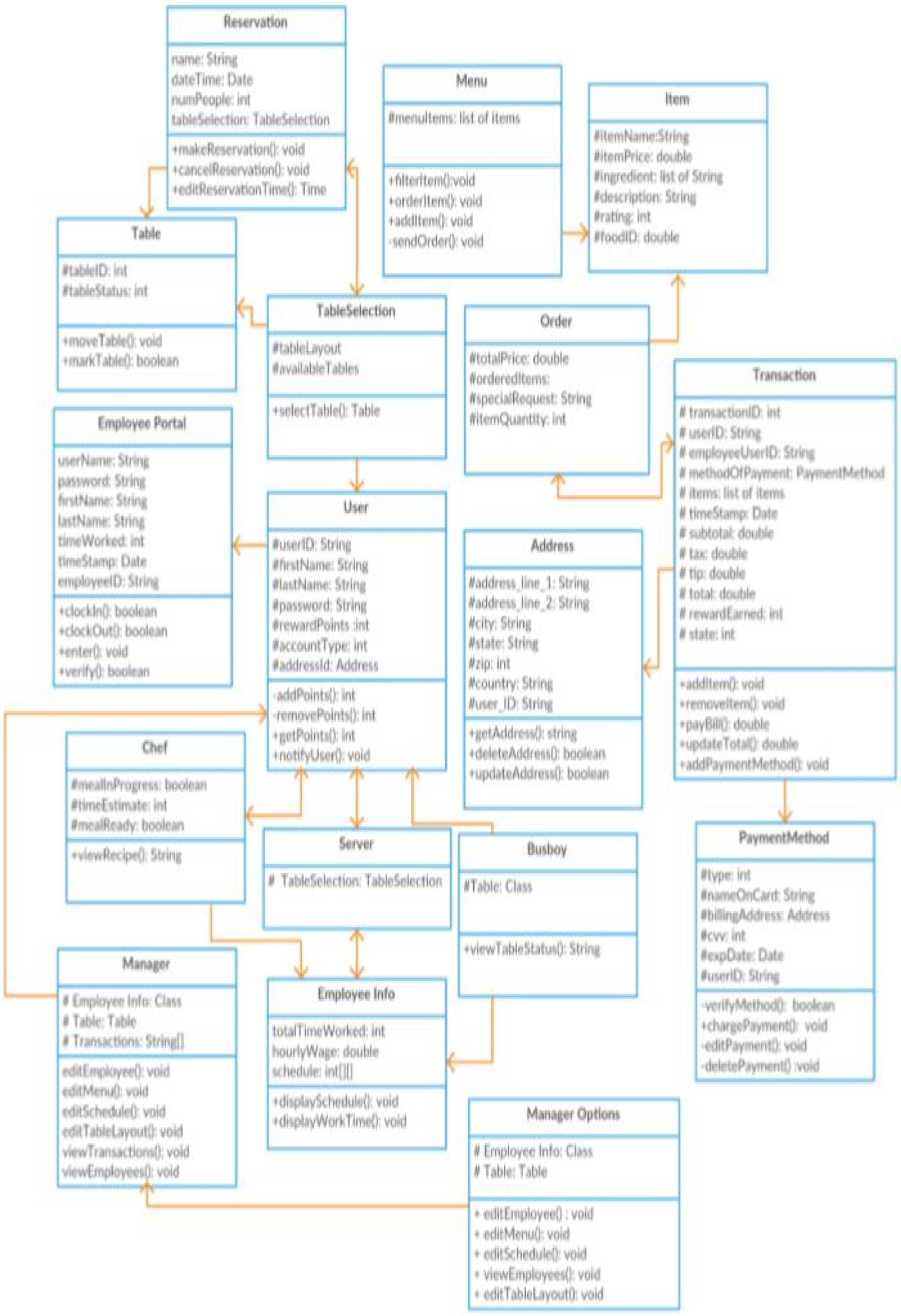


Figure 7: Clocking In/Clocking Out

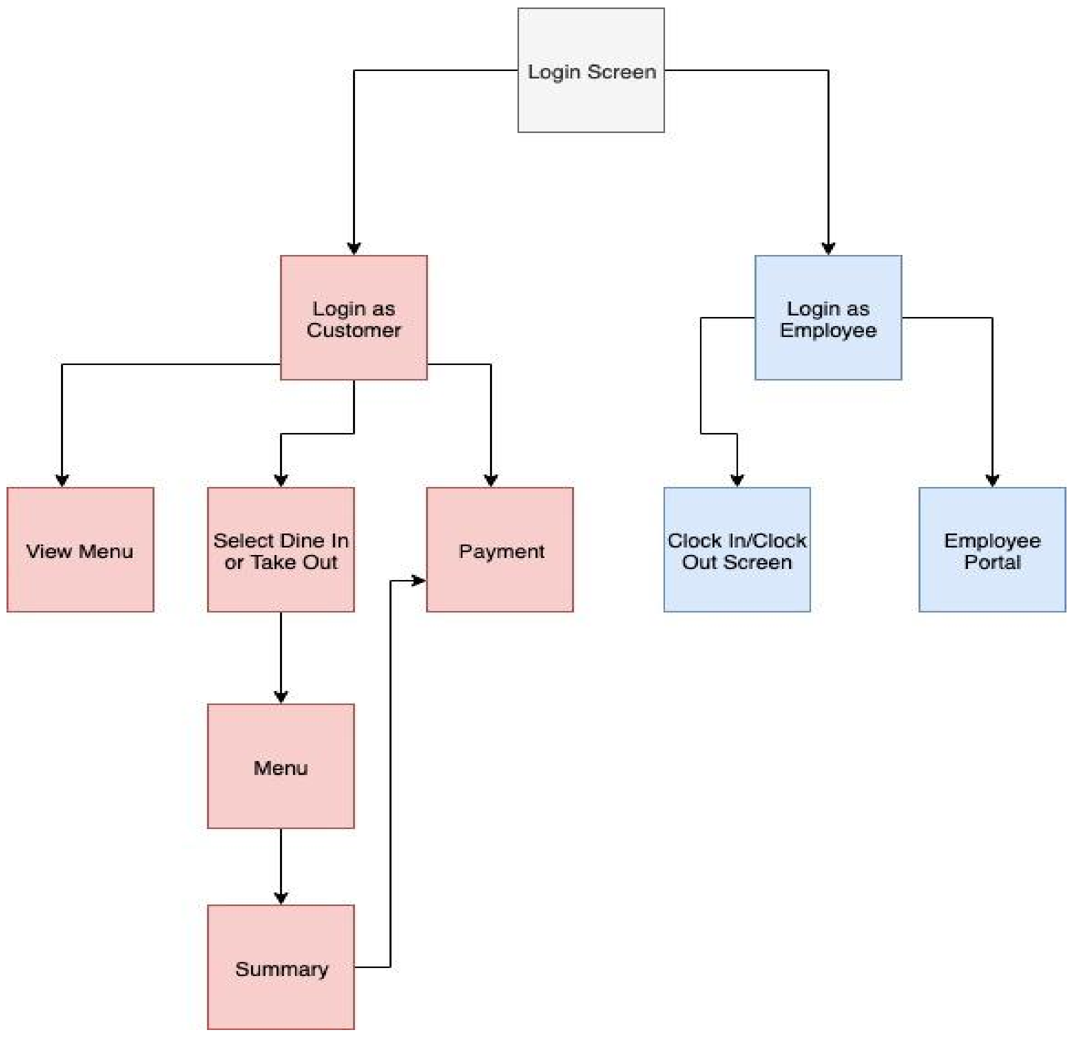
The diagram above demonstrates the interaction between objects and/or actors in UC-7: Clocking In/Out. The user must interact with the interface in order to open the application and log in. Once the user enters their login details, the interface will check them with the database and confirm their credentials. The user is then brought to an interface where they are presented with the option to clock in. Once they select the option to clock in, the system logs the user's name, time of clock-in, location, and ip address. Later on in the day, the user is able to use the system again to clock out, where the system again logs the user's name, time, location, and IP address.

One design principle that is used is the High Cohesion Principle. All of the objects don't take on many computation responsibilities. Each object only focuses on the tasks they are specialized to do. The database has all the logged information required for objects to perform their task, while the objects simply

4.3 CLASS DIAGRAM



INTEGRATION TESTING



Clock in or Clock Out
- To Clock In select Clock In Button
- To Clock Out select Clock Out Button

- ■ Employee Portal
  - View Tables
  - View Schedule
  - View Staff
  - View Menu

- ○ Customer UI
  - ■ Choose option of “Dine In” or “Take Out”
  - ■ View Menu
  - ■ Payment
- ○ Menu: either accessed by “Dine In”, “Take Out”, or “Continue as Guest”
  - ■ Select items to add to cart
  - ■ After selection, are brought to Summary Page where the list of items selected are shown and total price

- ○ Payment UI
  - ■ Option to add a card for payment or use an existing card
  - ■ If add new card, can put in card information
  - ■ Can pay for order successfully

Integration Testing:

- Q: When user adds an item twice to cart, does it show on summary?

A: Yes, the food item shows up twice on summary screen

2. Q: When Employee clocks in does it save in database?

   A: Yes, the timestamp and location with ip address is saved for that employee id

3. Login:
   - Enter the Username
   - Enter the associated Password
   - Press the Login Button

4. Create Account:
   1. From first screen, press create a new account
   2. Create by entering name, username, password, etc.
   3. Press Create Account

5. Create an Order
   1. Once logged in select dine in or take out
   2. Select an item to be added
   3. Select add to order
   4. If more items are desired press back button twice

5 ALGORIYHM AND DATA STRUCTURE

```
const Sequelize = require('sequelize'); const sequelize =
require('./sequelizeConf.js');

module.exports = (sequelize, DataTypes)=> {     return sequelize.define('item',
{        itemName: Sequelize.STRING,        itemPrice: Sequalize.DOUBLE,
ingredient: Sequalize.STRING,        description: Sequalize.STRING,
rating: Sequalize.INTEGER,        foodId: Sequalize.DOUBLE     }) }
item.belongsToMany(ingredient); ingredient.belongsToMany(item);es
```

5.1 Algorithms

For employee working information and clocking in/out, Team C's algorithms will be calculations for how much an employee works. One calculation will be put in a method called hoursWorked(). This method will take the timeOut, and timeIn variables, and convert both to "military time" (24-hour time). This will make calculations simple. Next the method will subtract timeIn from timeOut, and return the result. The Salary() method will calculate the annual salary of the given worker given hourly wage (wage), and number of hours worked (hoursWorked). This method gives a running total of a worker's salary for the year. The worker's salary for the day is calculated by hoursWorked * wage, this salary is added to expectedPay, then this process is repeated for every working day throughout the year.

When a customer is making a transaction, the order which they are making will be converted into a transaction object. When the transaction is created, a subtotal is calculated by adding the price of all items in the order, then a tax rate is applied to calculate a total. Once this information is presented to the user, and they enter their tip, that is added to calculate the total amount due for the transaction. The stripe API will be used to implement payment, and so all the information attached to this transaction object will be used to call the stripe API. Once the transaction goes through, the rewards system will increment a counter of how many visits the visiting customer has. Once they reach a set amount of visits (decided by the restaurant), they will earn a reward, which will be a set amount off of their next purchase (as decided by the restaurant as well)

For the menu side of our application, which will be worked on by Team A, customers will have the option to view a menu. Once customers view the menu, they will have the option to browse through everything in the menu or they can filter things out. For example, if a customer has a food allergy they can chose to filter out all items that have that ingredient in the meal. This is very helpful for customers with food allergies and also other dietary restrictions. Once we an ingredient is selected on the filter, out system will check the database for all the items including that ingredient. Then our database will send back the list of foods without the filtered ingredient and only display food items without the filtered ingredient. Once customers know what they want, they will be able to add and remove food items from their order as they please.

5.2 Data Structures
We can utilize is queue data structure for the Chef Interface. A queue is an ordered collection of items where the addition of new items happens at one end and the removal of existing items occur at the other end. This structure of a queue using the ordering principle of first in first out also known as “first come first served”. In the Chef Interface, the queue is used when orders come in from the system the chef is notified first of that one and works on that meal and then serves it out which removes it from the queue. This performance of accessing is O(n) and to remove is O(1) this is a very efficient data structures to use for this scenario.

There will be a menu table within the SQL database. A database is good for querying and selecting items based on attributes. This tables primary key will be foodId and the other columns necessary will be food name, food price, food ingredients and to improve efficiency it may be helpful to add common allergies and restrictions as columns for example, contains nuts in a meal.

In order to keep track of employee working information, such as time worked and current status, we will use a SQL database. The table employee in the database will have primary key, employee’s username, so that we can easily find an entry pertaining to that employee. The columns keep track of the employee’s time clocked in, time clocked out, whether or not the clocking was done on site, their current clock status, the total numbers of hours worked, each employee’s individual wage, and the estimated pay that is to be received on the next pay cycle.

When a customer is preparing an order, the system will create an order object, which will include a collection of items included in the order, as well as an associated table, customer, and server. When the customer goes to complete a purchase, this order object will be used to create a transaction object, which will include method of payment information, as well as a total. The method of payment will be represented by an object, there will be a field dictating what method of payment this is (cash, card, etc..), a field associating it with a user, and also fields detailing the information regarding the method of payment (card number, expiration date, billing address…). When a payment goes through, This will also update the User object’s rewards information. All the objects used are stored tables in our relational SQL database

6. SOFTWARE TOOLS USED

- **AUTOMATION TESTING TOOL**

AUTOMATION TESTING means using an automation tool to execute your test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Test Automation demands considerable investments of money and resources.

Successive development cycles will require execution of same test suite repeatedly. Using a test automation tool, it's possible to record this test suite and re-play it as required. Once the test suite is automated, no human intervention is required. This improved ROI of Test Automation. The goal of Automation is to reduce the number of test cases to be run manually and not to eliminate Manual Testing altogether.

Automation Tool Best Practices

To get maximum ROI of automation, observe the following

- The scope of Automation needs to be determined in detail before the start of the project. This sets expectations from Automation right.
- Select the right automation tool: A tool must not be selected based on its popularity, but it's fit to the automation requirements.
- Choose an appropriate framework
- Scripting Standards- Standards have to be followed while writing the scripts for Automation. Some of them are-
  - Create uniform scripts, comments, and indentation of the code
  - Adequate Exception handling - How error is handled on system failure or unexpected behavior of the application.
  - User-defined messages should be coded or standardized for Error Logging for testers to understand.

- Measure metrics- Success of automation cannot be determined by comparing the manual effort with the automation effort but by also capturing the following metrics.

    - Percent of defects found
    - The time required for automation testing for each and every release cycle
    - Minimal Time is taken for release
    - Customer Satisfaction Index
    - Productivity improvement

Automation Testing Tools

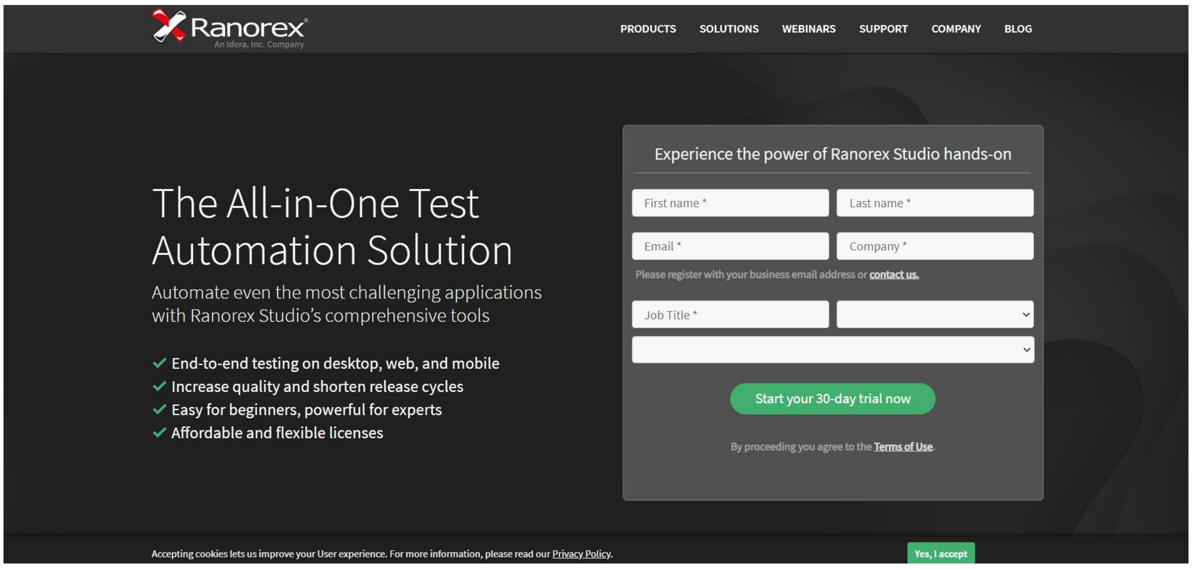
There are tons of Functional and Regression Testing Tools available in the market. Here are best tools certified by our experts

Ranorex Studio

Ranorex Studio is an all-in-one tool for automating functional UI tests, regression tests, data-driven tests and much more. Ranorex Studio includes an easy to use click-and-go interface to automate tests for web, desktop, and mobile applications.

Features:

- Functional UI and end-to-end testing on desktop, web, and mobile
- Cross-browser testing
- SAP, ERP, Delphi and legacy applications.
- iOS and Android
- Run tests locally or remotely, in parallel or distribute on a Selenium Grid
- Robust reporting



The All-in-One Test Automation Solution

Automate even the most challenging applications with Ranorex Studio's comprehensive tools

✓ End-to-end testing on desktop, web, and mobile
✓ Increase quality and shorten release cycles
✓ Easy for beginners, powerful for experts
✓ Affordable and flexible licenses

Experience the power of Ranorex Studio hands-on

First name *	Last name *
Email *	Company *

Please register with your business email address or contact us.

Job Title *	

Start your 30-day trial now

By proceeding you agree to the Terms of Use.

Accepting cookies lets us improve your User experience. For more information, please read our Privacy Policy.

Yes, I accept

Ranorex Studio fundamentals

This section describes the tools and concepts you need to **start automating basic tests** in Ranorex Studio. Each topic includes easy-to-follow step-by-step guides and self-contained examples so that you can gain hands-on experience immediately. If you enjoy learning from a **written tutorial**, we recommend starting with ⇢ Ranorize yourself in 20 minutes to get a feel for Ranorex Studio while producing your first test. If you prefer learning from a **video tutorial**, we recommend watching the Ranorex Studio Quick Start screencast series.

This section is structured as follows:

Explanation of symbols

Method: Contains the detailed description of a process. This includes an explanation of the process, the goal(s) of the process, and a step-by-step explanation of the actions necessary to successfully complete the process.

Concept: Detailed description of a principle, technical concept or key idea applied within one or more Ranorex Studio tools and/or methods. These concepts are important for understanding and using the referenced tool or method

Overview

Ranorex Studio supports development of automated test modules using standard programming languages such as C# and VB.NET.

Main features

- GUI object recognition, filtering GUI elements using the company's proprietary technology *RanoreXPath.*
- Object-based record and replay, using *Ranorex Recorder*, which records the user's interaction with a desktop or web-based application and creates user-maintainable scripts that can be edited with the Ranorex Studio action editor.[1][5] The recorded actions are available as both C# and VB.NET code.[8] Record and replay is supported on mobile devices for actions such as key presses and touch gestures.

Supported technologies

- Windows desktop client applications such as .NET,[4][10] WPF, Win32, VB6, Java, MFC, Embarcadero Delphi.
- Web technologies such as HTML, HTML5, JavaScript Frameworks,[4][10] Ajax, Silverlight, Flash, and Flex.
- Cross-browser testing for Chrome, Safari, Microsoft Edge, Internet Explorer, and Firefox[4][10]

- Mobile Apps
    - native iOS apps
    - native Android apps

System environment

Ranorex Studio runs on Microsoft Windows and Windows Server