CAPSTONE PROJECT

DATA SCIENCE

Presented By:
Mohammed Abubacker S
(2021311314)
B.Tech Petroleum and Technology
ACTech Anna University

edunet foundation

OUTLINE

- Problem Statement
- Proposed System/Solution
- System Development Approach
- Algorithm & Deployment
- Result
- Conclusion
- Future Scope
- References

PROBLEM STATEMENT

The current manual billing process causes delays for customers and inefficiencies for the organization, presenting an opportunity for improvement. Implementing a computer-based billing system could optimize resource allocation. This system allows for streamlined entry of client, employee, and payment information, enhancing record management and meeting various organizational data requirements. Key drawbacks of the current system include:-

- Limited ability to modify data-
- Dependence on manual operator oversight
- Excessive paper usage
- Difficulty in accessing information promptly
- Challenges in maintaining systematic records
- Paper waste

PROPOSED SOLUTION

The Restaurant Management System (RMS) is a software application that makes restaurant billing easier and faster. Its user-friendly interface allows staff to generate customer bills quickly, helping to reduce waiting times. RMS can handle large volumes of data, storing and retrieving information like billing history, reservations, and employee details.

As a desktop-based system, RMS requires little paperwork. It automates tasks like calculating bills and applying discounts, which helps reduce errors and speeds up the billing process. All data is kept in a secure database, reducing the risk of loss and removing the need for manual record-keeping. By implementing RMS, restaurants can operate more efficiently, streamlining essential functions like billing and reservations, while offering customers a more seamless experience.

SYSTEM APPROACH

Building a restaurant management system in Python involves a structured approach, consisting of the following steps:

1. **Understanding Requirements:**
Start by analyzing the system's needs, covering menu management, order processing, reservations, inventory management, and reporting.

2. **Modular Organization:**
Structure the system into separate modules, each dedicated to specific functions such as Menu Management, Order Processing, Reservation Handling, Inventory Management, and Reporting.

3. **Class Definition**:
Define classes within each module to represent entities and actions. These could include Menu Items, Orders, Reservations, and Inventory Items.

4. **Establishing Relationships:**
Establish connections between classes/modules to illustrate their interactions. For example, how orders relate to menu items and how inventory levels are affected by orders

SYSTEM APPROACH

5. Implementing Functionality:
Develop methods and functions within each class/module to execute necessary tasks, like updating menus, processing orders, managing inventory, and handling reservations.

6. Optional User Interface:
Depending on requirements, create a user interface, whether it's a command-line interface, a graphical interface using libraries like Tkinter or PyQt, or a web-based interface with Flask or Django.

7. Testing and Refinement:
Conduct comprehensive testing to ensure the system functions correctly and meets requirements. Refine the system based on feedback and test results.

8. Documentation:
Document the system's design, functionalities, and usage instructions to facilitate future maintenance and development efforts.

ALGORITHM & DEPLOYMENT

Revamped Structure for Restaurant Management:

1. **Efficient Menu Organization**:
Develop a structured menu item class capturing vital details like name, price, and description. - Establish a dedicated system for managing menu items, ensuring smooth addition, removal, and display processes.

2. **Streamlined Order Processing:**
Create a robust order framework, integrating essential elements such as order ID, selected items, and total price. - Implement seamless functionalities for adding items, precise price calculation, and efficient order processing.

3. **Seamless Reservation Handling:**
Define a comprehensive reservation structure, encompassing key details like reservation ID, customer name, date, and table allocation. - Design an intuitive reservation management system, enabling easy reservation creation, availability checks, and cancellations.

4. **Effective Inventory Management:**
Introduce a well-organized inventory framework, highlighting critical attributes like item name, quantity, and price. - Develop efficient inventory management tools for smooth stock addition, dynamic quantity updates, and proactive low stock alerts.

ALGORITHM & DEPLOYMENT

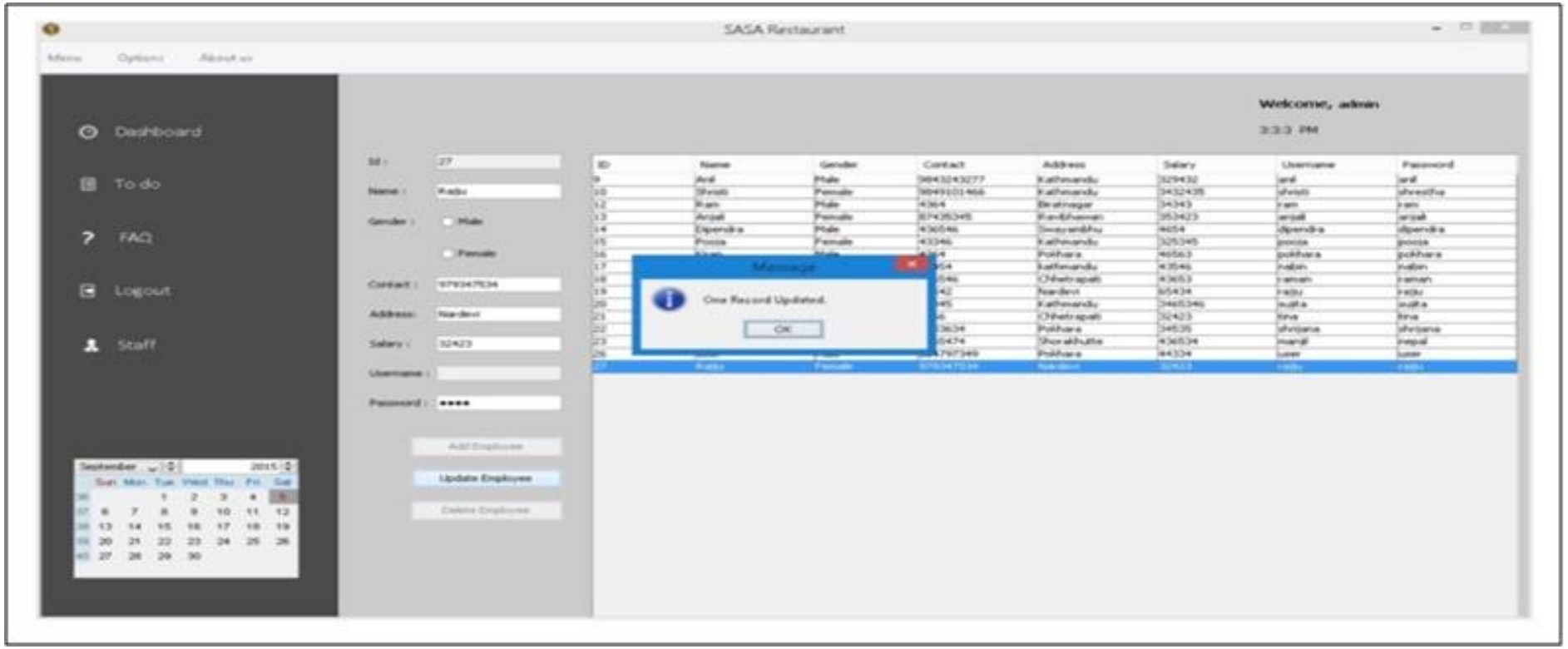
5. **Insightful Reporting Mechanisms:**
Incorporate advanced reporting functionalities to generate customized reports, including sales and inventory insights tailored to specific user needs.

6. **User-Centric Interface Design:**
Craft an intuitive interface to enhance user experience, offering flexible options such as command-line, web-based, or GUI interfaces tailored to user preferences.

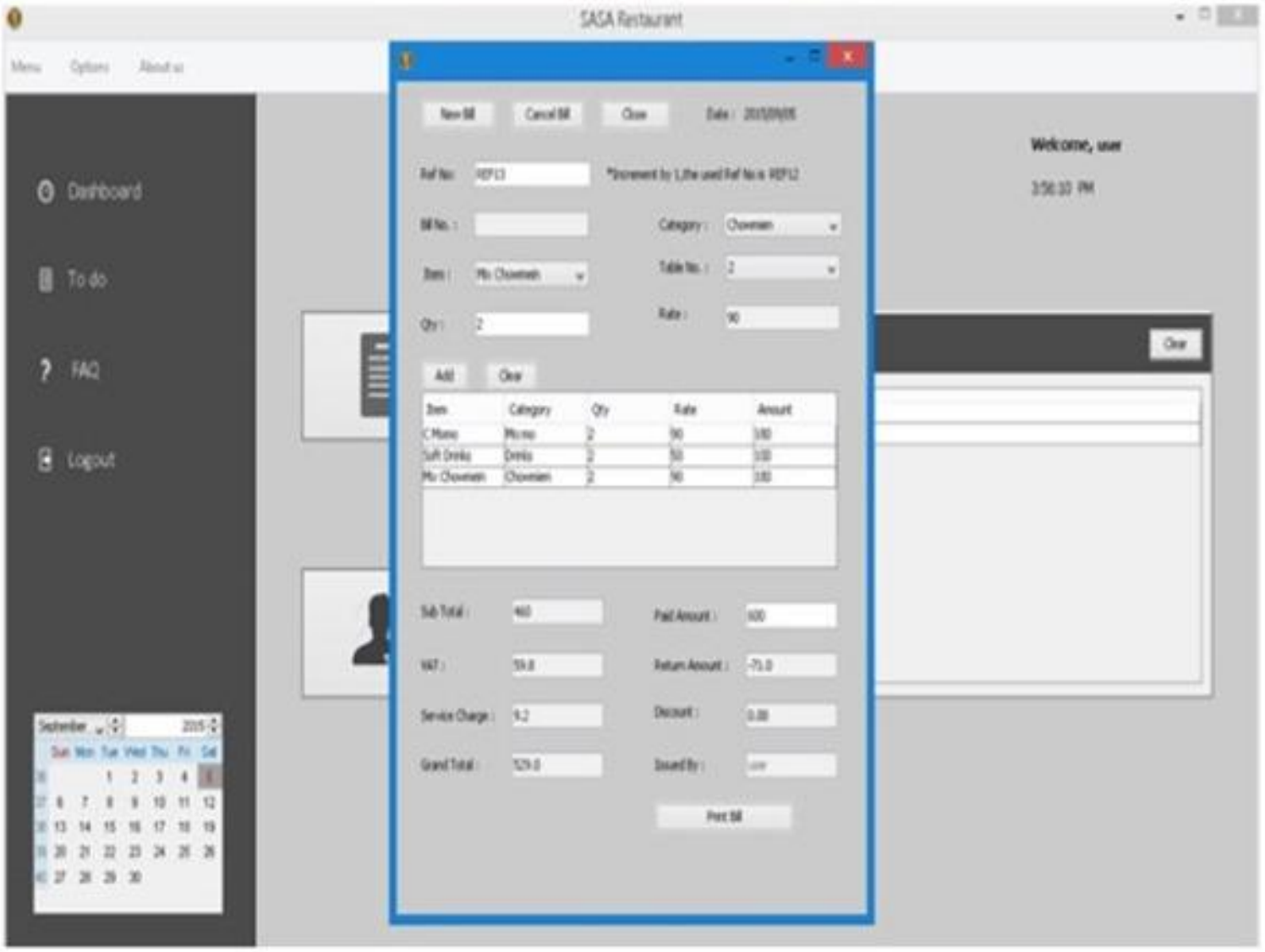
7. **Flexible Deployment Strategies:**
Explore diverse deployment approaches, including local deployment with PyInstaller, web-based deployment using Flask or Django, mobile app deployment via Kivy or React Native, containerization with Docker, and hybrid deployment for comprehensive coverage.

RESULT

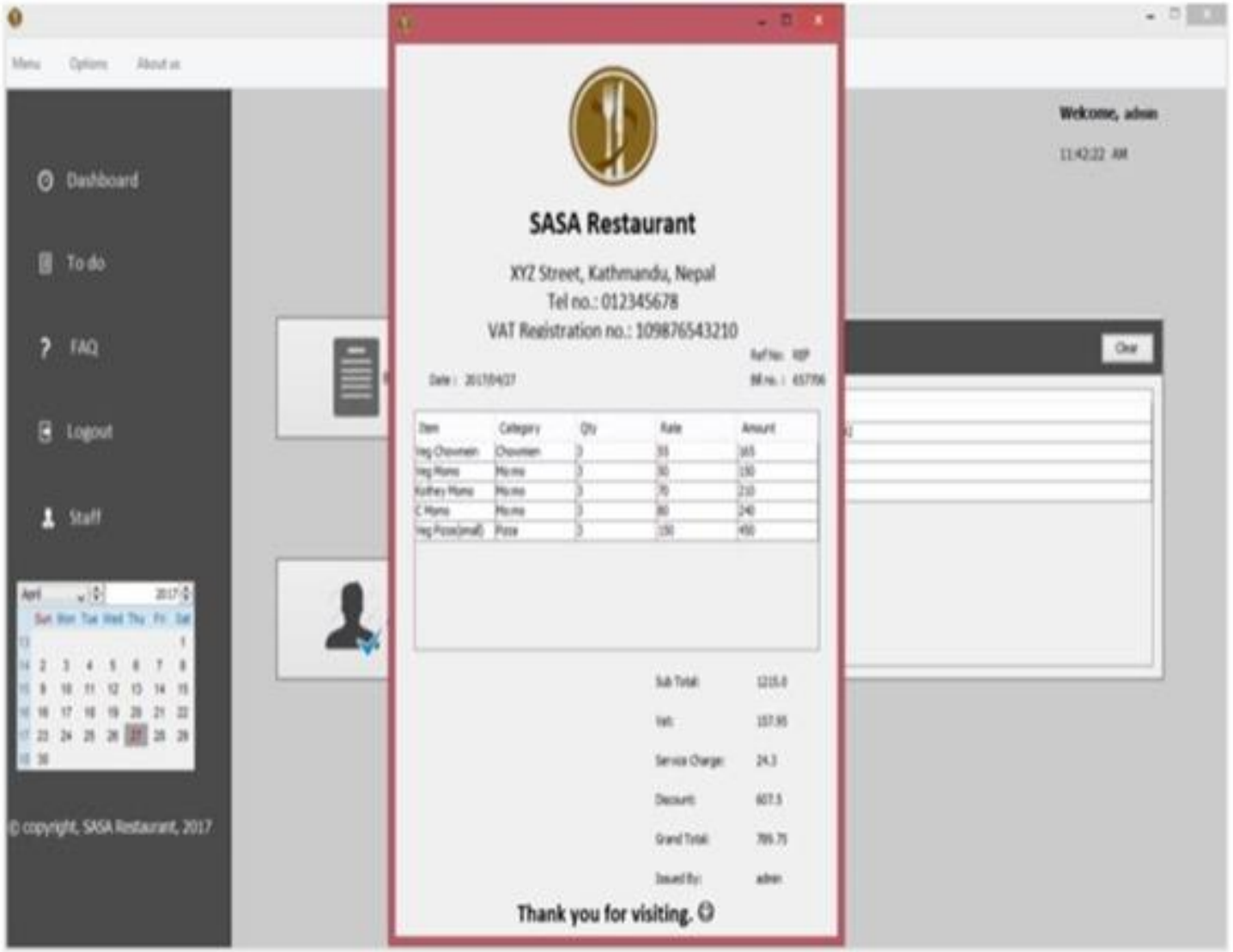
Regarding the outcomes of the Restaurant Billing System, it effectively manages staff information, facilitates customer reservations, handles billing tasks, and more. Analyzing the results, the system allows for comprehensive staff management, including adding, editing, updating, and deleting staff information. Additionally, it supports reservation creation and cancellation, as well as printing bills for customers. The system outputs are illustrated in the figure below.



RESULT



RESULT



edunet foundation

CONCLUSION

The documentation provides a thorough explanation of the Restaurant Management system's structure and coding. Creating this program was no small feat, involving extensive analysis, research, and specialized skills. Writing this report has been a rewarding journey, filled with lessons from the challenging tasks at hand. Designing a restaurant system required a mix of research and coding know-how, and achieving smooth operations took considerable time and effort.

Despite the obstacles, the system was successfully built with a robust design and smooth workflow. The billing system was the most challenging aspect. Extracting data from the database for billing required complex SQL queries and involved managing multiple changes across databases, which was time-consuming and required careful planning.

Overall, this project provided significant coding experience and underscored the importance of effective time management and teamwork in software development.

FUTURE SCOPE

Restaurant management software (RMS) is a robust solution that caters to diverse needs across restaurant operations. From staff management and order processing to billing, menu maintenance, reservation handling, and beyond, it offers a wide array of functionalities. As we look forward, the software holds immense potential for further enhancements. Future upgrades, such as advanced inventory management, wireless tableside ordering and payment systems, real-time alerts, online ordering integration, and mobile management tools, stand to significantly elevate revenue streams and operational efficiencies.

REFERENCES

Restaurant Billing System:
https://www.scribd.com/doc/283903672/Online-Ordering-System-
Project Objective :
https://www.scribd.com/document/36253350/04-Project-Billing-
System Scopes and Limitation:
https://kungfumas.files.wordpress.com/2017/09/099.pdf
Feasibility study:
http://www.slideshare.net/alok104/synopsis-on-billing-system-27487568
UML Diagram :
https://www.techopedia.com/definition/3243/unified-modeling-language-uml/
Use Case Diagram:
http://whatis.techtarget.com/definition/use-case-diagram
Class Diagram:
http://searchmicroservices.techtarget.com/definition/class-diagram
Sequence Diagram :
https://creately.com/blog/diagrams/sequence-diagram-tutorial/
ER Diagram:
http://searchcrm.techtarget.com/definition/entity-relationship-diagram
Interfaces : https://www.youtube.com/watch?v=9K5sS7j5wWI

THANK YOU

edunet foundation