Pharmacy Automation: Database-Based Medication Management

1st Abdulkadir Can Kinsiz
Dept. of Information System Engineering of Kocaeli University
Kocaeli, Turkey
kadircankinsiz@gmail.com

2nd Şevval Zeynep Ayar
Dept. of Information System Engineering of Kocaeli University
Kocaeli, Turkey
zeynepayar2949@gmail.com

3rd Sude Deniz Suvar
Dept. of Information System Engineering of Kocaeli University
Kocaeli, Turkey
sudesuvar51@gmail.com@gmail.com

Abstract—This report presents the development of a pharmacy automation system, aimed at streamlining various processes within a pharmacy setting. Leveraging technologies such as PHP CodeIgniter 4[6], MySQL[5], HTML[2], CSS[3] with Bootstrap 5[7], JavaScript[4], and AJAX[9][11], the system offers comprehensive automation solutions for tasks such as inventory management, prescription tracking, sales monitoring, and customer management. Through a user-friendly interface, pharmacists can efficiently manage their inventory, track prescriptions, and generate reports. The integration of these technologies enhances the overall efficiency and accuracy of pharmacy operations, ultimately leading to improved customer service and satisfaction.

Keywords—(*Pharmacy automation, PHP CodeIgniter 4, MySQL, HTML, CSS, Bootstrap 5, JavaScript, AJAX****)***

I. Introduction

The efficient management of pharmacies is crucial in today's dynamic healthcare landscape. With increasing demand and complexities, traditional manual methods struggle to keep pace. Hence, the development of a pharmacy automation system becomes imperative.

By integrating modern technologies into a cohesive platform, the pharmacy automation system revolutionizes pharmacy operations. It enhances efficiency and accuracy while empowering pharmacists with intuitive tools and real-time insights.

In subsequent sections, we'll explore the system's architecture, features and development methodologies. Through this analysis, we highlight the significance of pharmacy automation in modernizing healthcare services and improving patient outcomes.

II. Nomenclature

In this section, key terminologies and abbreviations used throughout the report are defined for clarity:

- MVC: Model-View-Controller [8]
- PHP: Hypertext Preprocessor [6]
- MySQL: Structured Query Language [5]
- HTML: Hypertext Markup Language [2]
- CSS: Cascading Style Sheets [3]
- AJAX: Asynchronous JavaScript and XML[9]

III. Detailed Analysis of Implementation Strategies and Algorithms

The provided controller file showcases not only the implementation strategies and algorithms but also leverages various features offered by the CodeIgniter 4 framework. Below is a detailed analysis, incorporating CodeIgniter's built-in features and their significance within the pharmacy automation system:

Model-View-Controller (MVC) Architecture:

CodeIgniter follows the MVC architectural pattern, promoting separation of concerns and modular development. The controller file effectively implements this pattern by organizing application logic into controllers, views for presentation, and models for data manipulation.

Security Features:

CodeIgniter provides built-in security features to mitigate common web application vulnerabilities, such as cross-site scripting (XSS)[12], SQL injection[13], and cross-site request forgery (CSRF)[14]. Features like CSRF protection, input data filtering, output escaping, and encryption are crucial for safeguarding sensitive data and preventing malicious attacks.

Database Abstraction and Query Builder:

CodeIgniter offers a powerful database abstraction layer and a query builder library, simplifying database interactions and enhancing portability across different database systems. The controller leverages these features to execute database queries in a secure and efficient manner, abstracting low-level database operations.

Session Management:

CodeIgniter facilitates session management through its session library, allowing developers to store and retrieve user-specific data across multiple requests. The controller utilizes session variables to maintain user state, manage cart items, and authenticate users securely.

Form Handling and Form Validation:

The framework provides helpers and libraries for handling form submissions and validating form inputs. Methods like $_POST, setRules(), run(), and validation_errors() aid in processing form data and ensuring its validity before further processing.

Error Handling and Logging:

CodeIgniter offers comprehensive error handling and logging mechanisms to debug and monitor application errors effectively. The controller utilizes try-catch blocks and logging functions to capture and log exceptions, ensuring robustness and maintainability.

Encryption and Security Utilities:

CodeIgniter includes encryption and security utilities for encrypting sensitive data, generating secure hashes, and handling passwords securely. Features like MD5[10] encryption library, hashing functions, and password hashing enhance the security of user data and credentials.

HTTP Request and Response Handling:

The framework simplifies HTTP request[15] and response handling through its HTTP library and helper functions. Methods like redirect(), base_url(), and site_url() facilitate URL generation, redirection, and response management, enhancing user experience and application flow.

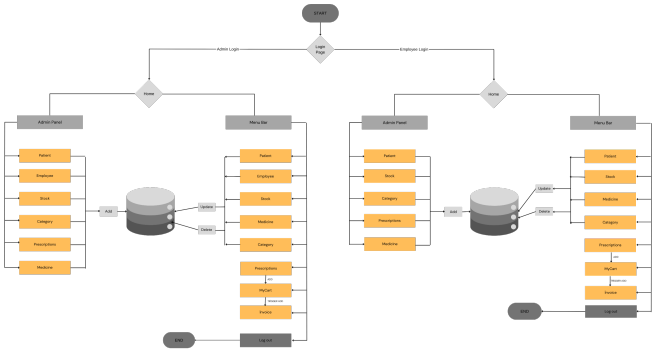
Internationalization and Localization:

CodeIgniter supports internationalization and localization features, allowing developers to create multilingual applications easily. Although not explicitly used in the provided controller, these features are beneficial for applications targeting diverse user bases.

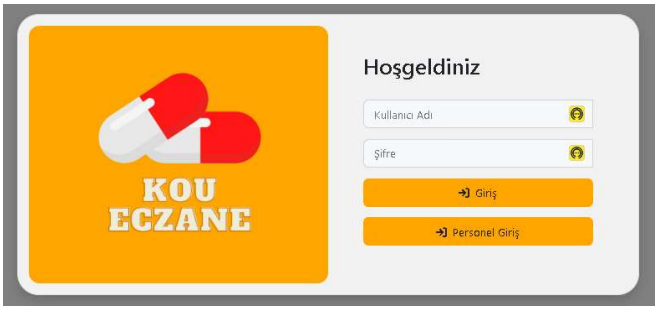
In summary, the controller file incorporates various features provided by the CodeIgniter 4 framework, enhancing security, performance, and maintainability of the pharmacy automation system. By leveraging these features alongside implementation strategies and algorithms, the system achieves robustness, scalability, and user satisfaction.

IV. CONCLUSION

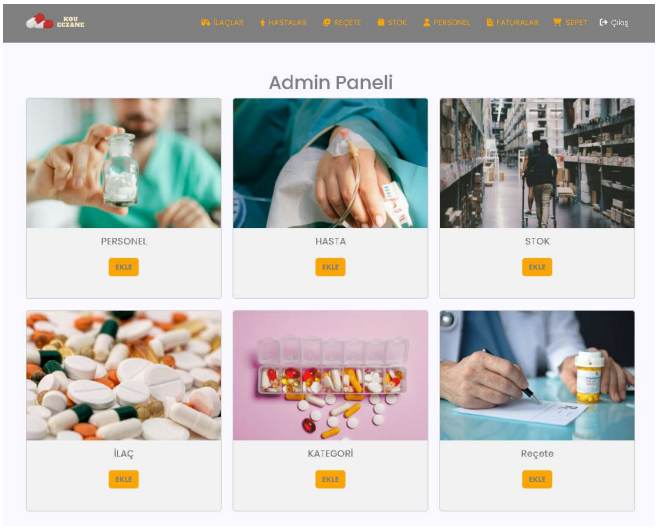
"In conclusion, the pharmacy automation system developed for this database course showcases the successful integration of PHP CodeIgniter 4 and database management principles to address real-world challenges in pharmacy management. By adhering to the MVC architecture and leveraging CodeIgniter's features, the system demonstrates effective organization and security measures. Through abstraction and query building, data interactions are simplified, while validation and encryption ensure data integrity and security. This project underscores the importance of applying theoretical database concepts into practical solutions, highlighting the potential for technology to revolutionize healthcare operations. Moving forward, the insights gained from this project can serve as a foundation for further exploration and innovation in database-driven applications within the healthcare sector."



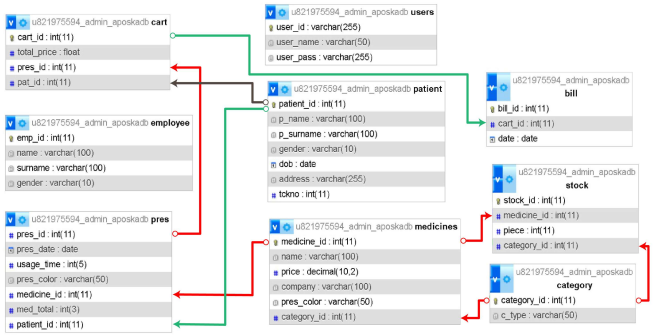
(Flow Chart)



(Login Panel)



(Admin Panel)



(Database Diagram)

ACKNOWLEDGMENT

We extend our sincere gratitude to all individuals who contributed to the development and implementation of the pharmacy automation system. Our team members' dedication, expertise, and collaboration were crucial in bringing this project to fruition. We also appreciate the guidance and support provided by our mentors and advisors, as well as the valuable feedback from users and stakeholders during testing and refinement. Additionally, we acknowledge the contributions of the open-source community, whose resources and frameworks have been instrumental in advancing our work. We would also like to express our thanks to the Stack Overflow[1] community for their invaluable assistance and insights throughout the development process. Together, these collective efforts have enabled us to create a robust and efficient solution for pharmacy management.

REFERENCES

[1]Stack Overflow. (n.d.). Stack Overflow. Retrieved May 5, 2024, from https://stackoverflow.com/

[2]W3Schools. (n.d.). HTML Tutorial. Retrieved May 5, 2024, from https://www.w3schools.com/html/

[3]W3Schools. (n.d.). CSS Tutorial. Retrieved May 5, 2024, from https://www.w3schools.com/css/

[4]W3Schools. (n.d.). JavaScript Tutorial. Retrieved May 5, 2024, from https://www.w3schools.com/js/

[5]W3Schools. (n.d.). SQL Tutorial. Retrieved May 5, 2024, from https://www.w3schools.com/sql/

[6]CodeIgniter. (n.d.). User Guide. Retrieved May 5, 2024, from https://www.codeigniter.com/userguide3/

[7]Bootstrap. (n.d.). Documentation. Retrieved May 5, 2024, from https://getbootstrap.com/docs/4.1/getting-started/introduction

[8]GeeksforGeeks. (n.d.). MVC Framework - Introduction. Retrieved May 5, 2024, from https://www.geeksforgeeks.org/mvc-framework-introduction/

[9]W3Schools. (n.d.). AJAX - What is AJAX?. Retrieved May 5, 2024, from https://www.w3schools.com/js/js_ajax_intro.asp

[10]GeeksforGeeks. (n.d.). What is the MD5 Algorithm?. Retrieved May 5, 2024, from https://www.geeksforgeeks.org/what-is-the-md5-algorithm/

[11]jQuery. (n.d.). jQuery. Retrieved May 5, 2024, from https://jquery.com/

[12]Cloudflare. (2023). Cross-site scripting (XSS). Retrieved May 5, 2024, from https://www.cloudflare.com/learning/security/threats/cross-site-scripting/

[13]Cloudflare. (2023). SQL injection. Retrieved May 5, 2024, from https://www.cloudflare.com/learning/security/threats/sql-injection/

[14]Cloudflare. (2024). Cross-site request forgery (CSRF). Retrieved May 5, 2024, from https://www.cloudflare.com/learning/security/threats/cross-site-request-forgery/

[15]Kinsta. (n.d.). What is an HTTP request? [Kinsta Knowledge Base]. Retrieved May 5, 2024, from https://kinsta.com/knowledgebase/what-is-an-http-request/