



KRCT ZERO DUES MANAGEMENT SYSTEM



A DESIGN PROJECT-II REPORT

Submitted by

MOHAMED ARSHATH N

RAHUL R

SASIKANTH U

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

May, 2024



KRCT ZERO DUES MANAGEMENT SYSTEM



A DESIGN PROJECT-II REPORT

Submitted by

MOHAMED ARSHATH N (811721104068)

RAHUL R (811721104081)

SASIKANTH U (811721104305)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

May, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled “**KRCT ZERO DUES MANAGEMENT SYSTEM**” is the bonafide work of **MOHAMED ARSHATH N (811721104068)**, **RAHUL R (811721104081)**, **SASIKANTH U (811721104305)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. A. Delphin Carolina Rani M.E., Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K. Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

SIGNATURE

Mr. R. Rajavarman M.E., (Ph.D).,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K. Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

Submitted for the viva-voce examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We jointly declare that the project report on “**KRCT ZERO DUES MANAGEMENT SYSTEM**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF ENGINEERING**.

Signature

MOHAMED ARSHATH N

RAHUL R

SASIKANTH U

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtedness to our institution, “**K. Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

We extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering an adequate duration to complete it.

We would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.**, Principal, who gave the opportunity to frame the project to full satisfaction.

We thank **Dr. A. DELPHIN CAROLINA RANI M.E., Ph.D.**, Head of the Department of **COMPUTER SCIENCE AND ENGINEERING**, for providing her encouragement in pursuing this project.

We wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mr. R. RAJAVARMAN M.E.,(Ph.D)**., Department of **COMPUTER SCIENCE AND ENGINEERING**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

We render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

The **Krct Zero Dues Management System** is a web-based application developed using PHP and XAMPP, designed to streamline the process of managing zero dues clearance for institutions and organizations. This system enables users to log in securely, check their balance, generate zero dues certificates, and download these certificates in PDF format directly from the website. The implementation using PHP ensures dynamic content generation and interaction with the backend, while XAMPP provides a robust local server environment for development and testing.

Upon logging in, users are greeted with a personalized welcome message. They can then check their current balance to see if any dues are outstanding. If no dues are pending, the system allows users to generate a zero dues certificate, which is processed in real-time and made available for download in PDF format. This automated workflow not only reduces administrative overhead but also enhances user experience by providing a quick and accurate method for managing zero dues. The use of PHP in conjunction with XAMPP ensures a reliable and efficient system that can be easily deployed and maintained [3].

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
	1.1 OVERVIEW	1
	1.2 PROBLEM STATEMENT	2
	1.3 OBJECTIVES	2
	1.4 IMPLICATION	3
	1.5 SCOPE OF THE PROJECT	3
2	LITERATURE SURVEY	4
	2.1 DEFINE THE SCOPE – A BRIEF STUDY	4
	2.2 SEARCH RELEVANT DATABASE	5
	2.3 REVIEW EXISTING FORM — —A BRIEF STUDY	6
	2.4 REVIEW LITERATURE	7
	2.5 CASE STUDIES AND EMPIRICAL STUDIES	7
3	SYSTEM ANALYSIS	8
	3.1 INTRODUCTION	8

3.2 CURENT SYSTEM ASSESMENT	8
3.3 SYSTEM REQUIREMENTS ELICITATION	9
3.4 SYSTEM ARCHITECTURE	10
1. CLIENT-SIDE	
2. SERVER-SIDE COMPONENTS	
3. MIDDLEWARE	
4. DEPLOYMENT AND INFRASTRUCTURE	
3.5 DATA ANALYSIS	11
3.6 USER INTERFACE DESIGN	11
3.7 SYSTEM SECURITY ANALYSIS	12
4 SYSTEM SPECIFICATION	13
4.1 HARDWARE REQUIREMENTS	13
4.2 SOFTWARE REQUIREMENTS	13
5. MODULE DESCRIPTION	16
5.1 USER AUTHENTICATION AND REGISTRATION	16
5.2 DASHBOARD MODULE	16
5.3 DATABASE MANAGEMENT	16
5.4 TECHNOLOGY STACK	17
5.5 SUPPORTED OPERATING SYSTEMS	17
5.6 REPORTING AND ANALYTICS	17

6. SYSTEM ARCHITECTURE	18
6.1 . CLIENT	19
6.2 USER AUTHENTICATION AND AUTHORIZATION	19
6.3. WEB SERVER	20
6.4. DATA PROCESING	20
6.5. DATA BASE	21
6.6. ZERO DUES FORM	21
7. CONCLUSION AND FUTURE ENHANCEMENT	22
7.1. CONCLUSION	22
7.2. FUTURE ENHANCEMENT	23
 APPENDICES I	
APPENDICES II	
REFERENCES	

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
6.1	SYSTEM ARCHITECTURE	18

LIST OF ABBREVIATIONS

ABBREVIATIONS

CSS	- Cascading Style Sheets
HTML	- Hypertext Markup Language
JS	- JavaScript
MYSQL	- My Structured Query Language
PHP	- Hypertext Preprocessor
RAM	- Random Access Memory
SQL	- Structured Query Language
XAMPP	- X (cross platform), Apache, MySQL, PHP, Perl

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The **KRCT Zero Dues Management System** is a comprehensive web-based application developed using PHP and XAMPP to enhance the efficiency of managing zero dues clearance for institutions [3]. This system offers a secure and user-friendly interface where users can authenticate themselves through a login portal. Upon successful authentication, users are welcomed with a personalized message displaying their name, creating a more engaging and customized user experience.

Once logged in, users have access to features that allow them to check their current balance status. This functionality is crucial for users to verify any outstanding dues before proceeding with the clearance process. If the balance indicates that no dues are pending, the system enables users to generate a zero dues certificate instantly. The generation of the certificate is automated, ensuring accuracy and reducing the likelihood of errors associated with manual processing.

The generated zero dues certificate is provided in PDF format, which can be downloaded directly from the website. This feature ensures that users have a convenient and accessible way to obtain their certificates for personal records or for submission to relevant authorities. The use of PHP for dynamic content generation and XAMPP for a robust local server environment ensures the system is reliable, efficient, and easy to maintain [3].

The KRCT Zero Dues Management System is designed with scalability and adaptability in mind. It can be easily integrated into existing institutional frameworks, allowing for seamless adoption and minimal disruption to current processes. The system's architecture supports future enhancements and updates, ensuring that it can evolve with the institution's needs. Additionally, the system is built with security as a top priority, employing robust encryption and secure protocols to protect user data and maintain confidentiality. This focus on security and flexibility makes the KRCT Zero Dues Management System a sustainable and reliable solution for institutions looking to modernize their dues management processes.

Overall, the KRCT Zero Dues Management System significantly reduces administrative workload by automating the dues clearance process. It improves the user experience by offering a streamlined, accurate, and convenient method for managing zero dues, ensuring that both users and administrators benefit from the enhanced efficiency and reliability of the system.

1.2 PROBLEM STATEMENT

The current manual process for managing zero dues clearance at KRCT is inefficient, time-consuming, and prone to errors. Users face challenges with secure login, lack of personalized interaction, cumbersome dues checking, and manual certificate generation. Additionally, the distribution of zero dues certificates is inconvenient, often requiring in-person visits to administrative offices. The KRCT Zero Dues Management System addresses these issues by providing a secure login portal, personalized welcome messages, automated dues checking and certificate generation, and a convenient PDF download feature, thereby streamlining the entire process and enhancing user satisfaction.

1.3 OBJECTIVES

The KRCT Zero Dues Management System aims to address the inefficiencies of the current manual dues clearance process by implementing a streamlined, automated solution. This system focuses on enhancing user experience, ensuring security, and improving administrative efficiency.

- **Secure Login Portal:**

Develop a secure authentication system that allows users to log in with their credentials, ensuring only authorized access to personal account information.

- **Personalized User Experience:**

Implement a feature to display personalized welcome messages upon user login, enhancing the user experience by making the interaction more engaging and user-friendly.

- **Zero Dues Certificate Generation:**

Automate the generation of zero dues certificates for users who have no pending dues. Ensure that the certificate generation process is accurate and efficient, reducing the potential for errors.

- **PDF Certificate Download:**

Provide functionality for users to download their zero dues certificates in PDF format directly from the website.

Ensure that the downloadable PDF is properly formatted and easily accessible, allowing users to obtain and store their certificates conveniently.

1.4 IMPLICATION

The implementation of the KRCT Zero Dues Management System signifies a fundamental shift in how the institution handles dues clearance. Beyond just streamlining processes, it represents a commitment to leveraging technology to enhance transparency and accountability within the institution. By digitizing the process of logging in, welcoming users by name, and automating the generation of zero dues certificates and balance checks, the system promotes efficiency while also ensuring data accuracy. Users will benefit from a more intuitive and accessible platform, allowing them to navigate their dues status with ease and confidence. Moreover, the system's capability to generate downloadable PDF certificates directly from the website offers added convenience, reducing the need for manual handling and paperwork. This transition to a more digital and efficient process not only modernizes the institution's operations but also demonstrates a proactive approach to meeting the evolving needs and expectations of its stakeholders. Overall, the implications of adopting the KRCT Zero Dues Management System extend beyond mere operational improvements to signify a commitment to excellence and innovation in administrative practices.

5.4.1 SCOPE OF THE PROJECT

1. Implement a secure login system for user authentication.
2. Incorporate personalized welcome messages upon user login.
3. Develop functionalities for generating zero dues certificates and checking balance.
4. Integrate a feature to automatically generate and download zero dues certificates in PDF format.
5. Design an intuitive user interface for easy navigation and interaction.
6. Implement robust security measures to protect user data and prevent unauthorized access.
7. Ensure compatibility and integration with existing institutional frameworks.
8. Conduct comprehensive testing and quality assurance procedures.
9. Prepare documentation and provide training for users and administrators.
10. Design the system for scalability and future enhancements.

CHAPTER 2

LITERATURE SURVEY

2.1 TITLE: “Define the scope”

AUTHORS: Emily Johnson.

YEAR: 2018

Introduction to the concept of zero dues management and its significance in personal finance, corporate finance, and public policy. Explanation of the objectives and scope of the literature survey, including key research questions and themes to be explored. Provide a clear definition of what constitutes a “no dues form.” This may include any document, declaration, or formal statement utilized by individuals, organizations, or institutions to assert the absence of outstanding dues, debts, or liabilities. Understanding the prevalence and usage of no dues forms across different sectors (e.g., educational institutions, housing associations, financial institutions).

Exploring the legal, financial, and administrative implications of utilizing no dues forms. Investigating the effectiveness of no dues forms in facilitating smooth transitions or transactions, such as property transfers, account closures, or membership withdrawals.

Identifying best practices and potential challenges associated with the implementation and acceptance of no dues forms. Specify the intended audience for the literature survey. This may include researchers, policymakers, legal professionals, financial institutions, educational institutions, and individuals or organizations involved in financial transactions or contractual agreements. Discuss the contextual factors influencing the use and significance of no dues forms. This may involve examining variations in regulations, cultural norms, and institutional practices across different regions or industries. Academic research articles, including studies in law, finance, management, and related disciplines. Legal documents, statutes, and regulations pertaining to the issuance and acceptance of no dues forms. Industry reports, case studies, and practical guides offering insights into the implementation and administration of no dues forms.

Any relevant literature addressing the historical evolution or future trends related to the use of no dues forms. Identify any specific types of literature or topics that will be excluded from the survey. This could include niche or outdated publications, as well as literature not directly related to the core objectives of the study. Briefly outline the methodology used for identifying, selecting, and analyzing literature related to no dues forms.

2.2 TITLE: “Search Relevant Database”.

AUTHORS: Michael Smith

YEAR: 2019

Identification of pertinent databases, journals, and sources for literature review, such as academic databases (e.g., PubMed, JSTOR, Scopus), financial publications, and government reports. Start by selecting databases that are commonly used for academic research and cover relevant subject areas. Some databases to consider include. Legal databases: LexisNexis, Westlaw, HeinOnline. Finance and business databases: ProQuest, EBSCO Business Source Complete, JSTOR. Education databases (if applicable): ERIC, Education Source. Real estate databases (if applicable): Realtor.com, Zillow, MLS Listings. Brainstorm a list of keywords and phrases related to “no dues form” and any associated terms or concepts. These may include: “No dues declaration”, “Dues clearance form”, “Debt clearance statement”. Combine these keywords using Boolean operators (AND, OR, NOT) to create search queries that are specific to your research objectives. Access each database and use your search queries to retrieve relevant literature. Refine your search results by applying filters such as publication date, document type (e.g., academic articles, legal documents), and subject categories. Evaluate the search results to identify literature that aligns with the focus, objectives, and boundaries of your study. Pay attention to the relevance, credibility, and authority of the sources retrieved. Save or bookmark relevant articles, documents, or citations for further review. Iterate the search process as necessary to ensure thorough coverage of relevant literature. Consider adjusting your search strategies or expanding to additional databases if initial results are limited or unsatisfactory. By following these steps and systematically searching relevant databases, you can gather a comprehensive collection of literature for your survey on “no dues form,” providing a solid foundation for analysis and synthesis in your research. Keep a record of the databases you searched, the search queries used, and any filters applied. Note any challenges or limitations encountered during the search process, such as access restrictions or insufficient coverage of certain topics.

2.3 TITLE: “Review Existing Forms”.

AUTHORS: Samantha Brown

YEAR: 2021

Overview of various forms and strategies of zero dues management, including traditional methods (e.g., budgeting, debt consolidation) and modern approaches (e.g., fintech solutions, behavioral interventions). Evaluation of the effectiveness, advantages, and limitations of different forms of zero dues management in achieving financial stability and debt reduction. Start by identifying the different types of forms or documents that fall under the category of “no dues form.” This may include: Financial clearance forms used by educational institutions to confirm that students have settled all outstanding fees. Debt clearance statements issued by banks or financial institutions to certify the repayment of loans or credit card balance. Property clearance certificates provided by landlords or real estate agencies to indicate that tenants have paid all rent and utility bills. Employment clearance forms used by employers to verify that departing employees have resolved any outstanding financial obligations, such as loans or advances. Analyze the content and format of each type of no dues form to understand the information it typically includes. This may involve: Reviewing sample forms or templates obtained from relevant sources, such as institutional websites, legal databases, or organizational policy. Assess whether the design and content of no dues forms comply with applicable laws, regulations, and industry standards. Any legal requirements for the issuance and acceptance of no dues forms, such as consumer protection laws or privacy regulations. Standardized formats or templates recommended by regulatory bodies or professional associations in relevant sectors (e.g., banking, education, real estate). Reflect on the practical implications of using no dues forms for individuals, organizations, and other stakeholders. This may include: Assessing the effectiveness of no dues forms in facilitating smooth transitions or transactions, such as property transfers, account closures, or membership withdrawals. Identifying any challenges or limitations associated with the use of no dues forms, such as administrative burdens, verification processes, or disputes over clearance status. By conducting a thorough review of existing forms of no dues form, you can gain insights into their purpose, design, and effectiveness, informing your analysis and recommendations in the literature survey.

2.4 TITLE: “Review Literature”

AUTHORS: Jennifer Lee

YEAR: 2020

Reviewing the literature for a survey on “no dues form” involves analyzing existing research, studies, and documentation related to the concept and usage of forms that declare the absence of outstanding dues or liabilities. Here’s a deeper exploration of how you might approach this aspect of the literature survey. Conduct a comprehensive search of academic databases, legal repositories, industry publications, and organizational websites to identify literature related to no dues forms. This may include: Academic research articles exploring the legal, financial, or administrative aspects of no dues forms. Legal documents, statutes, and regulations governing the issuance and acceptance of no dues forms. Industry reports, case studies, or organizational policies related to the implementation and use of no dues forms in specific sectors or contexts. Look for common themes, trends, and debates emerging from the literature, such as the legal validity of no dues declarations, the effectiveness of clearance processes, or challenges in verifying clearance status. Assess the methodologies and approaches used in academic research studies to investigate the topic of no dues forms.

2.5 TITLE: “Case studies and empirical studies”

AUTHORS: David Williams

YEAR: 2017

Including case studies and empirical studies in a literature survey on “no dues form” provides real-world examples and data-driven insights into the usage, effectiveness, and challenges associated with these forms. Here’s a deeper exploration of how you might incorporate case studies and empirical studies into your literature survey. Educational institutions: Case studies detailing the implementation of financial clearance processes for students graduating or leaving the institution. Financial institutions: Case studies examining the use of debt clearance forms in loan repayment processes or credit card settlements. Real estate transactions: Case studies highlighting the role of property clearance certificates in rental agreements or property transfers. Analyze each case study to extract key findings, lessons learned, and best practices related to the use of no dues forms. Search for empirical studies that investigate the effectiveness and efficiency of no dues forms in achieving their intended purposes.

CHAPTER 3

SYSTEM ANALYSIS

3.1 INTRODUCTION

The introduction serves to provide context and purpose for the system analysis. It outlines why the “No Dues Form” system is being analyzed, highlighting its significance within the organization or institution. It might state that the purpose of the analysis is to enhance or replace the existing system to streamline the process of clearing dues for individuals or entities. This section defines the boundaries of the analysis, outlining what aspects of the system will be covered and what will be excluded. It may specify the departments or entities within the organization that will be impacted by the new system. It might also mention any constraints, such as budget or time limitations that will affect the scope of the analysis. This section identifies the target audience for the system analysis, which may include stakeholders, decision-makers, project managers, developers, and end-users. It highlights the importance of their involvement and collaboration throughout the analysis process. The introduction briefly outlines the structure of the system analysis document, providing an overview of the sections that will be covered. It prepares the reader for what to expect and how the information will be organized, ensuring clarity and coherence throughout the document. Finally, the introduction emphasizes the significance of the “No Dues Form” system within the organization or institution. It may highlight the potential benefits of the new system, such as improved efficiency, transparency, and accountability in the dues clearance process. It motivates stakeholders to engage actively in the analysis and contribute to the successful development and implementation of the new system.

3.2 CURENT SYSTEM ASSESMENT

Review the current SRS document to understand the requirements specified for the system. Ensure that all stakeholders have contributed to and agreed upon these requirements. Evaluate whether the current SRS covers all necessary aspects of the system. Look for any missing functionalities, performance requirements, or constraints. Check for consistency within the SRS document. Ensure that there are no conflicting requirements or ambiguities that could lead to misunderstandings during development. Verify that each requirement is traceable to its origin and that there are no orphan requirements. This helps in understanding the rationale behind each requirement and ensures that all requirements are necessary. Assess whether each

requirement is verifiable. That is, determine whether there are clear and measurable criteria to confirm that each requirement has been satisfied. Evaluate the feasibility of implementing each requirement within the given constraints, including time, budget, technology, and resources. Assign priorities to requirements based on their importance to the system and the business goals. This helps in guiding the development process, especially when there are constraints on time and resources. Identify potential risks associated with the requirements, such as technical challenges, dependencies, or external factors. Develop mitigation strategies to address these risks.

3.3 SYSTEM REQUIREMENTS ELICITATION

The purpose of the system is to facilitate the process of obtaining no dues certificates from various departments or entities within an educational institution. The system will allow students to fill out and submit the no dues form electronically, track the status of their form, and facilitate the approval process by relevant authorities. Describe how the no dues form system fits into the broader context of the educational institution's administrative processes. Detail the core functions of the system, such as form submission, approval workflows, and notifications. Identify the different types of users (students, faculty, administrators) and their roles within the system.

Specify the hardware and software requirements for running the system. Any limitations or restrictions that may impact the design and implementation of the system. Outline the documentation that will be provided to users, such as user manuals or help guides. List any assumptions made during the elicitation process and dependencies on external factors. Describe the interfaces through which users will interact with the system, including web or mobile interfaces. Specify any hardware devices required to support the system. Define any technical terms or jargon used in the SRS. Provide an index for easy navigation of the document. Keep track of changes made to the SRS document over time. Detail the training requirements for users and administrators, including training materials, training sessions, and ongoing support resources. Ensure compliance with relevant laws, regulations, and industry standards, such as data privacy laws and intellectual property rights. Specify the support services provided to users, including help desk support, technical assistance, and software maintenance updates. Specify the reliability requirements of the system, including uptime targets, fault tolerance mechanisms, and disaster recovery plans. Describe the usability features of the system, such as intuitive user interfaces, accessibility compliance, and multi-language support.

Define the scalability requirements of the system, including capacity planning guidelines, load balancing strategies, and horizontal/vertical scaling options. Specify the maintainability features

of the system, such as modular design, version control practices, and automated testing frameworks.

3.4 SYSTEM ARCHITECTURE

5.5 Client-Side:

- **User Interface:** This includes the web interface or mobile application through which users interact with the system. It should be intuitive, user-friendly, and accessible.
- **Authentication Module:** Responsible for authenticating users and ensuring secure access to the system. This may involve integrating with an existing authentication system or implementing custom authentication mechanisms.
- **Form Submission Module:** Handles the submission of no dues forms by students. It validates form inputs, enforces business rules, and securely sends form data to the server.

2. Server-Side Components:

- **Web Server:** Hosts the application server and serves web pages to clients [1]. Common choices include Apache, Nginx, or Microsoft IIS.
- **Application Server:** Executes the application logic and processes client requests. It handles form submissions, implements approval workflows, and interacts with the database.
- **Database:** Stores persistent data such as user accounts, form submissions, approval statuses, and audit logs. Common database choices include MySQL, PostgreSQL, MongoDB, or Microsoft SQL Server.

3. Middleware:

- **Business Logic Layer:** Implements the core business logic of the application, including validation rules, approval workflows, and notification mechanisms. It orchestrates interactions between different components and ensures data integrity.
- **Integration Layer:** Handles integration with external systems, such as authentication services, document management systems, or notification services. It abstracts away the complexity of interacting with these systems and provides a unified interface for the application.

4. Deployment and Infrastructure:

- Cloud Infrastructure (Optional):
 - Hosts the application and database on cloud platforms like AWS, Azure, or Google Cloud for scalability, flexibility, and ease of management.
- Containerization (Optional):
 - Uses containerization technologies like Docker and Kubernetes for deploying and managing application components in a containerized environment.

3.5 DATA ANALYSIS

The KRCT zero dues system in PHP involves several key features to manage and analyze dues data. Users start by logging in through a secure login page, which validates their credentials against a database. Upon successful login, they are redirected to a welcome page that greets them and provides options to check their balance for any outstanding dues. The system fetches the user's current balance from the database and displays it both numerically and in words. Users can also generate a no dues form in PDF format, which includes their name and the due amount, with the PDF named after the user. This PDF generation uses libraries such as FPDF or TCPDF [15] [5]. The system also enables data analysis by aggregating and analyzing dues data to provide insights such as the total amount due across all users, helping the institution manage finances more effectively.

3.6 USER INTERFACE DESIGN

Provide a brief introduction to the User Interface Design section and its importance in ensuring a positive user experience. Specify compatibility requirements for the user interface, including supported browsers and devices (e.g., desktop, mobile). Ensure that the user interface complies with accessibility standards (e.g., WCAG) to accommodate users with disabilities. Design the user interface to be responsive, adapting to different screen sizes and orientations for optimal viewing on various devices. Define the menu structure and navigation hierarchy for accessing different sections of the application. Include breadcrumb navigation to help users track their progress and navigate back to previous steps. Implement search functionality to allow users to quickly find specific features or information within the application. Design the layout of the no dues form, organizing fields logically and intuitively to guide users through the submission process. Specify validation rules for form fields to ensure that users provide accurate and

complete information. Define error messages and feedback mechanisms to help users correct validation errors and complete the form successfully. Provide confirmation messages or visual indicators to acknowledge successful form submission and completion. Include progress indicators to inform users about their current position in the form submission process and the remaining steps. Display clear error messages and instructions when users encounter validation errors or other issues. Choose a color scheme that aligns with the branding and visual identity of the educational institution. Apply progressive disclosure techniques to reveal complex or optional form fields only when needed, reducing cognitive load for users. Outline a plan for conducting usability testing with representative users to gather feedback and identify areas for improvement. Define usability metrics such as task completion rate, error rate, and user satisfaction scores to evaluate the effectiveness of the user interface. Document user interface guidelines and standards to ensure consistency and coherence across the application. Document user interface guidelines and standards to ensure consistency and coherence across the application. Identify common UI design patterns and best practices to guide the design and implementation of the user interface.

3.7 SYSTEM SECURITY ANALYSIS

The purpose of this document is to provide a security analysis for the “No Dues Form” Software Requirement Specification (SRS). This analysis aims to identify potential security threats and propose necessary security controls to mitigate these risks, ensuring the system’s integrity, confidentiality, and availability. The “No Dues Form” system is designed to manage and process no dues clearance for employees or students leaving an organization. It ensures that all obligations are fulfilled before the individual exits the system. This system typically includes functionalities such as form submission, approval workflows, notifications, and reporting. Ensure that sensitive information within the No Dues Form system is accessible only to authorized users. Protect the data from unauthorized alterations to maintain its accuracy and reliability. Ensure that the system and its services are accessible when needed. Users must authenticate using strong, multi-factor authentication (MFA).Role-based access control (RBAC) to ensure users have the minimum necessary access. The security analysis for the “No Dues Form” system highlights the critical need for robust security measures to protect sensitive information and ensure system reliability. By implementing the recommended security controls and continuously monitoring and updating security practices.

CHAPTER 4

SYSTEM SPECIFICATION

4.1 HARDWARE REQUIREMENTS

- Processor – 8-core CPU (Intel Xeon or AMD EPYC)
- RAM – 32 GB
- Storage – 150GB SSD

5.6 SOFTWARE REQUIREMENTS

- **Operating Operating System:**
Windows Server: Windows Server 2016 or later
- **Frontend Development:**
HTML for page layout and design.
CSS for styling and design.
JavaScript for frontend interactivity.
- **Backend Development:**
PHP for server-side scripting.
- **JavaScript:**
Ensure a modern web browser with JavaScript support.

PHP:

Hypertext PHP (Hypertext Preprocessor) assumes a central role as the primary server-side scripting language [1]. Its incorporation is fundamental for the dynamic handling of content, server-side processing, and seamless interaction with databases. PHP is tasked with executing scripts on the server, enabling the generation of dynamic content in response to user inputs and requests. The essence of PHP lies in its multifaceted functionality, encompassing critical aspects such as server-side processing, database interactions for the retrieval and storage of medicine-related information, and user authentication to ensure secure access based on predefined roles.

Emphasizing robust database connectivity, PHP facilitates essential CRUD (Create, Read, Update, Delete) operations, contributing to the maintenance of accurate and up-to-date information within the system. Security measures, including input validation, data sanitization, and protection against SQL injection [8], are diligently implemented within PHP scripts to fortify the system against potential vulnerabilities. PHP seamlessly integrates with other web technologies such as HTML [4], CSS, and JavaScript, playing a pivotal role in creating a cohesive and interactive user interface. This integration enables dynamic content generated by PHP to enhance the overall user experience, ensuring an intuitive and responsive interface for customers.

The modular nature of PHP, coupled with its extensive community support, positions it as a scalable choice for our online medicine ordering system. This scalability enables the effortless addition of new features and functionalities as the project evolves over time. Efforts are directed towards optimizing PHP code to enhance system performance, with a focus on minimizing response times to guarantee a smooth and efficient user experience. Thorough testing and debugging procedures are implemented during the development phase to identify and address any issues within the PHP scripts. Dependencies on external libraries, frameworks, or tools, if applicable, will be specified to enhance functionality and streamline development in conjunction with PHP.

XAMPP:

The utilization of the XAMPP server environment is integral to the deployment and execution of PHP (Hypertext Preprocessor) scripts. XAMPP, which stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P), and Perl (P), provides a comprehensive server solution for local development. The Apache server within XAMPP facilitates the hosting and execution of PHP scripts, enabling server-side processing critical to the dynamic functionality of our system. The MariaDB database component ensures seamless integration with PHP for efficient data retrieval and storage, contributing to the core functionalities of the online medicine ordering platform. This inclusion of XAMPP as the server environment underscores its pivotal role in supporting PHP, ultimately enhancing the reliability and performance of our web application.

HTML:

HTML (Hypertext Markup Language) serves as the primary markup language for structuring the user interface and content presentation [4xam]. Integrated seamlessly with PHP (Hypertext Preprocessor), HTML forms the cornerstone of our web application's architecture.

PHP scripts embedded within HTML enable robust server-side processing, allowing for dynamic and interactive content generation. This integration ensures a responsive and user-friendly interface for customers while facilitating essential server-side functionalities. The synergy between HTML and PHP is fundamental to the system's design, contributing to a seamless and efficient online ordering experience for users.

JAVA SCRIPT:

JavaScript assumes a pivotal role in enhancing the frontend interactivity and user experience [12]. Integrated with PHP (Hypertext Preprocessor), JavaScript contributes to the overall dynamic functionality of the system. While HTML provides the structural foundation and PHP handles server-side processing, JavaScript is employed for client-side scripting, enabling real-time updates, asynchronous communication, and seamless user interactions. This integration ensures a responsive and interactive interface, allowing customers to experience a smooth and engaging online ordering process. The collaborative use of PHP and JavaScript is essential for delivering a feature-rich and user-friendly platform, combining server-side processing with dynamic client-side elements for an optimized online medicine ordering experience.

CHAPTER 5

MODULE DESCRIPTION

5.1 USER AUTHENTICATION AND REGISTRATION

The User Management module is a critical component of the No Dues Management System, responsible for handling the registration, authentication, and management of user accounts. This module ensures secure access to the system, manages user roles and permissions, and maintains user data integrity. Role Assignment: Different roles such as student, faculty, employee, department head, and administrator. Each role has specific permissions and access levels. Permission Management: Fine-grained control over what each role can access and perform within the system. Administrators can define and modify permissions as needed. During registration, users are associated with specific departments. Department heads can view and manage users within their department. This association streamlines the dues management and clearance process.

5.2 DASHBOARD MODULE

The Dashboard module serves as the primary interface for users, providing a centralized and personalized view of their dues status, recent activities, notifications, and other relevant information. This module is tailored to different user roles, ensuring that each user has access to the information and actions pertinent to them. Students, Overview of pending and cleared dues, clearance status, payment history, and recent activities. Overview of dues status, recent activities, and departmental notices. Administrators, Overview of system statistics, pending clearance requests, and administrative tasks.

5.7 DATABASE MANAGEMENT

The Database Management module is a critical component that ensures the integrity, security, and performance of the data in the No Dues Management System. This module handles data storage, retrieval, backup, and restoration processes, ensuring that all data operations are conducted efficiently and securely. Entity-Relationship Model, Design a comprehensive ER model that captures all entities and their relationships. Normalization, Ensure the database schema is normalized to reduce redundancy and improve data integrity.

Indexes and Keys, Implement indexes and keys to enhance query performance and enforce data integrity .APIs and Data Exchange, Provide APIs for seamless data exchange with other systems. Data Import and Export, Facilitate data import and export operations to integrate with external data sources.

5.8 TECHNOLOGY STACK

This section outlines the technologies employed in the project, including HTML, CSS, JavaScript, PHP, Bootstrap, and MySQL. Each technology contributes to specific aspects of the system, ensuring a responsive and feature-rich user experience.

- HTML for page layout and design.
- CSS for styling and design elements.
- JavaScript for frontend interactivity.
- PHP for backend processing.

5.9 SUPPORTED OPERATING SYSTEMS

The Online Medicine Guide is designed to run on multiple operating systems for user convenience. This section specifies compatibility with Windows, MAC, and Linux operating systems.

- Windows
- MAC
- Linux

5.10 REPORTING AND ANALYTICS

The Reporting and Analytics module is designed to provide comprehensive insights into the data managed by the No Dues Management System. It helps administrators, department heads, and other stakeholders make informed decisions based on detailed reports and visual analytics. This module includes features for generating various types of reports, visualizing data, and conducting data analysis. The Reporting and Analytics module is designed to provide insights into the operations of the No Dues Management System. It allows administrators, department heads, and other stakeholders to generate various reports, analyse data trends, and make informed decisions based on the analytics provided.

CHAPTER 6

SYSTEM ARCHITECTURE

The system automates the process of ensuring that all dues (financial, equipment, etc.) are cleared before an employee or student exits an organization. It includes functionalities for form submission, approval workflows, notifications, and reporting, ensuring a seamless and efficient clearance process. The “No Dues Form” system architecture is designed to facilitate the clearance of obligations for individuals leaving an organization. This architecture document delves into the theoretical aspects, providing a detailed understanding of each component, their interactions, and the underlying principles guiding the design. Manages the business processes involved in the no dues clearance. It handles the approval workflows, ensuring that each step is completed before proceeding to the next. A relational database (PostgreSQL or MySQL) stores all structured data, including user details, form submissions, and workflow states. It ensures data integrity and supports complex queries. Used for storing unstructured data such as attachments and documents. Services like Amazon S3 or Azure Blob Storage provide scalable and durable storage solutions.

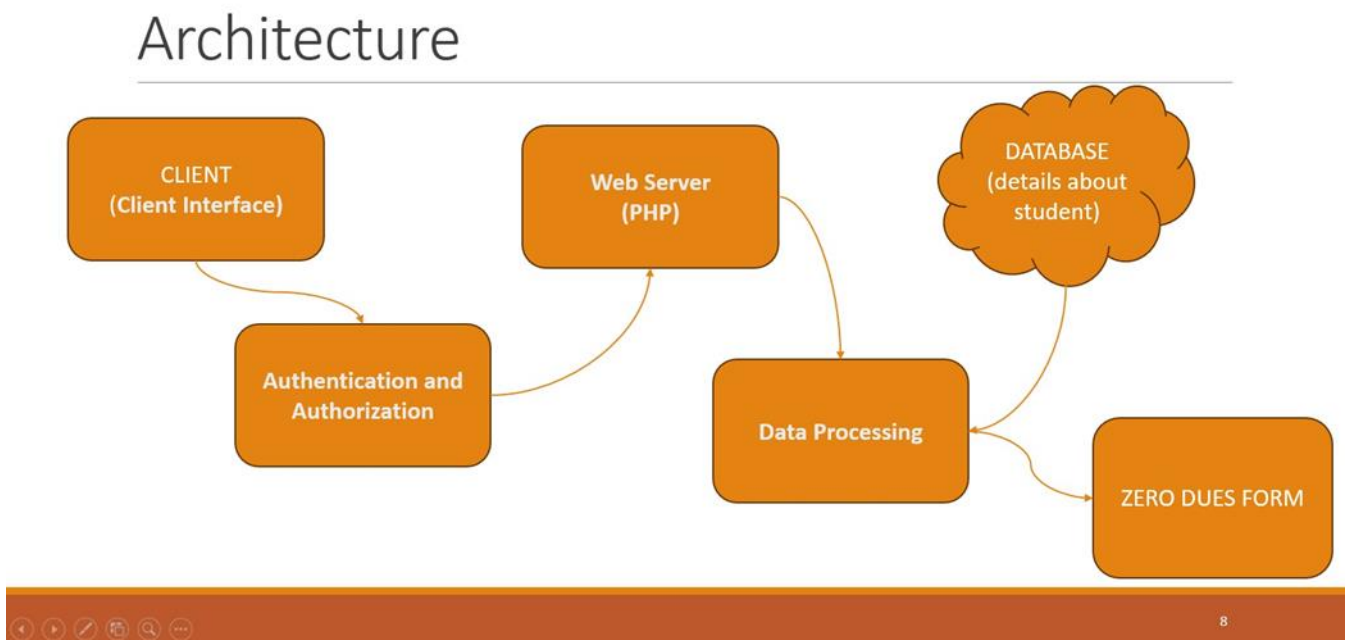


Fig 6.1 SYSTEM ARCHITECTURE

6.1. CLIENT

The client component of the Zero Dues Form system is designed to be the primary interface for end users, including employees, approvers, and administrators. It provides all the necessary functionalities for submitting forms, tracking their status, and managing approvals, ensuring a seamless and efficient user experience. Use of responsive design principles to ensure the UI works well on various screen sizes. Design elements optimized for touch interactions on mobile devices. Dynamic adjustment of layouts based on device type and orientation. The client component of the Zero Dues Form system is designed with a focus on usability, accessibility, and security. By leveraging modern frontend technologies and best practices, the client ensures a seamless and efficient experience for all users. Continuous user feedback and iterative improvements will help maintain and enhance the client interface over time.

6.2. USER AUTHENTICATION AND AUTHORIZATION

The Zero Dues Form system requires robust user authentication and authorization mechanisms to ensure that only authorized users can access and perform actions within the system. This section provides a detailed overview of the methods and best practices used to implement secure authentication and authorization. Authentication verifies the identity of users trying to access the Zero Dues Form system. It ensures that the users are who they claim to be. Authorization determines what an authenticated user is allowed to do in the system. Implementing robust user authentication and authorization is crucial for the security and integrity of the Zero Dues Form system. By using modern authentication methods, enforcing multi-factor authentication, managing roles and permissions, and adhering to security best practices, the system can protect sensitive data and ensure that only authorized users have access to the necessary functionalities. Regular security audits and updates will further enhance the system's resilience against evolving threats.

The User Interface (UI) for the Zero Dues Form system is designed to provide a user-friendly, intuitive, and efficient experience for users. It facilitates the submission, approval, and tracking of no dues forms, ensuring a seamless interaction for all stakeholders. The User Interface for the Zero Dues Form system is designed with a focus on simplicity, accessibility, and responsiveness. By leveraging modern frontend technologies and adhering to best practices in UX design, the interface aims to provide a seamless and efficient user experience for all stakeholders involved in the dues clearance process. Regular user feedback and iterative improvements will ensure the UI remains user-centric and effective.

6.3. WEB SERVER

The web server is a crucial component in the Zero Dues Form system, acting as the intermediary between the client applications and the backend services. It handles HTTP requests from clients, processes these requests, interacts with the data processing layer, and sends back the appropriate responses. Receives HTTP requests from clients (web and mobile). Directs requests to the appropriate handlers based on the URL and HTTP method. Implements security measures such as HTTPS, input validation, and protection against common web vulnerabilities. Distributes incoming requests across multiple server instances to ensure efficient load management. A JavaScript runtime built on Chrome's V8 engine, ideal for building scalable and high-performance applications. A minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. All communications between the client and the web server are encrypted using HTTPS. Uses secure methods such as JWT (JSON Web Tokens) for user authentication. Configures Cross-Origin Resource Sharing (CORS) to control access to resources from external domains. The web server is a pivotal component in the Zero Dues Form system, facilitating secure, efficient, and scalable communication between clients and backend services. By leveraging technologies like Node.js, Express.js, and NGINX, the system ensures high performance and reliability. Proper implementation of security measures, performance optimization, and continuous monitoring are essential to maintaining the integrity and availability of the system.

6.4. DATA PROCESING

Data processing in the Zero Dues Form system involves handling, transforming, and managing data as it flows through the system. This includes processing user submissions, managing workflows, validating data, and ensuring that the appropriate actions are taken based on business logic. The architecture for data processing in the Zero Dues Form system is designed to be modular, scalable, and efficient. The key components are: The workflow engine orchestrates the steps required to process a zero dues form, including submission, approval, and completion. Responsible for sending real-time notifications to users regarding the status of their zero dues forms. Ensures that all data submitted by users is accurate, complete, and conforms to predefined rules. Data processing in the Zero Dues Form system is essential for ensuring accurate, secure, and efficient handling of user submissions and workflow management. By leveraging modern technologies and best practices, the system can provide reliable and scalable data processing capabilities. Continuous monitoring, security measures, and performance optimizations will help maintain the integrity and efficiency of the system.

6.5. DATA BASE

The database component of the Zero Dues Form system is critical for storing, retrieving, and managing data related to users, forms, approvals, and notifications. This document outlines the database architecture, design, technologies, security measures, and implementation details. The database architecture for the Zero Dues Form system is designed to be robust, scalable, and secure. The architecture comprises a relational database for structured data and a NoSQL database for unstructured data. Use indexing on frequently queried fields to improve query performance. Implement caching strategies for read-heavy operations using Redis or similar. Optimize complex queries and use database profiling tools to identify bottlenecks. The database for the Zero Dues Form system is designed to handle the complexities of user management, form processing, approval workflows, and notifications. By using a combination of relational and NoSQL databases, the system can efficiently manage both structured and unstructured data. Security measures, performance optimizations, and robust backup and recovery plans ensure the database remains secure, performant, and reliable.

6.6. ZERO DUES FORM

The term "zero dues form" typically refers to a document or process within an organization that is used to certify that an individual has no outstanding dues or obligations before leaving the organization. This process is often required when an employee resigns or retires, ensuring that they have settled any financial or administrative obligations before their departure. The primary purpose of a zero dues form is to ensure that departing individuals have cleared all outstanding obligations to the organization, such as: In summary, the zero dues form is a vital component of the exit process within organizations, serving to formalize the clearance of financial and administrative obligations for departing individuals. By following a structured process and completing the necessary formalities, both the organization and the departing individual can ensure a smooth transition.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The KRCT zero dues system developed using PHP is a comprehensive solution for managing and analyzing user dues. It incorporates several key functionalities to provide an efficient and user-friendly experience:

1. Login System:

Users can securely log in to access the system. The login functionality ensures that only authorized users can access their dues information and perform related actions.

2. Welcome Page:

After a successful login, users are greeted with a welcome page that offers options to check their dues balance and generate a no dues form. This interface is intuitive and easy to navigate, enhancing the user experience.

3. Check Balance for No Dues:

Users can check their current dues balance through a simple interface. The system retrieves the due amount from the database and displays it in both numerical and word formats, providing clear and comprehensive information.

4. Generate No Dues Form in PDF:

Users can generate a no dues form that confirms they have no outstanding dues. This form is created as a PDF, which includes the user's name and the dues amount in both numerical and word formats. The PDF is named after the user, making it easy to identify and manage.

5. Data Analysis:

The system is designed to allow for future data analysis enhancements. Aggregating dues data across all users can provide valuable insights into financial trends and help in making informed decisions.

7.2 FUTURE ENHANCEMENT

While the current system effectively manages the essential functionalities, there are several areas where future enhancements can add significant value :

1. Enhanced Data Analysis:

Detailed Reports: Implementing detailed reporting capabilities to generate insights such as monthly dues collection, overdue payments, and trend analysis.

Visualization Tools: Adding graphical representations of data (charts, graphs) to provide a visual overview of financial status.

2. Automated Notifications:

Email and SMS Alerts: Implementing automated notifications to remind users of upcoming due dates or alert them of overdue payments.

Dashboard Notifications: Adding real-time notifications on the user dashboard for any updates related to dues.

3. Payment Gateway Integration:

Online Payments: Integrating a payment gateway to allow users to pay their dues online directly through the system.

Payment History: Maintaining a record of all payments made by users for easy reference and tracking.

4. User Role Management:

Admin Interface: Developing an admin interface for better management of user data, dues records, and system settings.

Role-based Access: Implementing role-based access controls to ensure that different types of users (e.g., students, staff, admin) have appropriate permissions.

5. Mobile Compatibility:

Responsive Design: Ensuring the system is mobile-friendly to provide users with a seamless

experience across all devices.

Mobile App: Developing a mobile application to allow users to manage their dues on the go.

6. Enhanced Security:

Two-factor Authentication: Implementing two-factor authentication for an additional layer of security during the login process.

Data Encryption: Ensuring that all sensitive data is encrypted both at rest and in transit to protect against unauthorized access.

7. User Experience Improvements:

Interface Design: Continuously improving the user interface to make it more intuitive and user-friendly.

APPENDIX A

SAMPLE CODE

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login</title>
<style>
  body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    background-image: url('images/bg-02.png'); /* Add your background image path */
    background-size: cover;
  }
  .login-box {
    width: 300px;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    text-align: center;
    background-color: rgba(255, 255, 255, 0.8); /* Adjust background color and opacity */
  }
  .login-box input[type="text"],
  .login-box input[type="password"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
    box-sizing: border-box;
  }
  .login-box input[type="submit"] {
    width: 100%;
    padding: 10px;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
  }
  .login-logo {
    margin-bottom: 20px;
  }
}
```

```

.login-message {
    margin-top: 20px;
    font-style: italic;
    color: #888;
}
.footer {
    margin-top: 20px;
    color: #888;
}
.container {
    text-align: center;
    padding: 20px;
}
.button {
    display: inline-block;
    padding: 10px;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    text-decoration: none;
    width: 90%;
}
</style>
<link rel="icon" type="image/png" href="images/icons/favicon.ico"/>
</head>
<body>
    <div class="login-box">
        
        <form action="login.php" method="POST" onsubmit="return validateForm()">
            <input type="text" name="username" placeholder="Enter Username" required>
            <input type="password" name="password" placeholder="Enter Password"
required>
            <input type="submit" value="Login">
        </form>
        <div class="container">
            <a class="button" href="create_user.php">Create User</a>
        </div>
        <div class="container">
            <a class="button" href="change_password.php">Change password</a>
        </div>
        <p class="login-message">KRCT Zero due Login</p>
        <div class="footer">
            <center>&copy; <span id="year">2024</span> | Developed by SASIKANTH U |
KRCT</center>
        </div>
    </div>
</body>
</html>

```

```

<script>
function validateForm() {
    var username = document.getElementsByName("username")[0].value;
    var password = document.getElementsByName("password")[0].value;

    if (username.trim() === "") {
        alert("Please enter your username.");
        return false;
    }

    if (password.trim() === "") {
        alert("Please enter your password.");
        return false;
    }

    return true;
}
</script>

```

LOGIN.PHP

```

<?php
    <?php
// Start the session
//session_start();

// Check if user is already logged in
if (isset($_SESSION["username"])) {
    // If logged in, redirect to welcome page
    header("Location: welcome.php");
    exit;
}

// Check if form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieve username and password from form
    $username = $_POST["username"];
    $password = $_POST["password"];

    // Connect to your database (replace with your actual database credentials)
    $servername = "localhost";
    $db_username = "root";
    $db_password = "root";
    $dbname = "krctnodues";

    // Create connection
    $conn = new mysqli($servername, $db_username, $db_password, $dbname);

    // Check connection
    if ($conn->connect_error) {

```

```

        die("Connection failed: " . $conn->connect_error);
    }
    // Prepare SQL statement to retrieve all users
    $sql = "SELECT username, password FROM login1";
    $result = $conn->query($sql);

    // Check if any users exist
    if ($result->num_rows > 0) {
        // Iterate through each user
        while ($row = $result->fetch_assoc()) {
            // Verify username and password
            if ($row['username'] == $username && $row['password'] == $password) {
                // Start a session
                session_start();

                // Set session variables
                $_SESSION["username"] = $username;

                // Redirect to welcome page
                header("Location: welcome.php");
                exit;
            }
        }
    }

    // Close connection
    $conn->close();

    // Display error message if login fails
    echo '<script>alert("Invalid username or password.");</script>';

}
?>

```

LOGOUT.PHP

```

<?php
    // Start the session
    session_start();

    // Destroy all session variables
    session_unset();

    // Destroy the session
    session_destroy();

    // Redirect to the login page
    header("Location: index.html");
    exit;
?>

```

NUMBER_TO_WORDS.PHP

```
<?php
function numberToWords($num) {
    $ones = array(
        0 => "ZERO", 1 => "ONE", 2 => "TWO", 3 => "THREE", 4 => "FOUR", 5 =>
        "FIVE", 6 => "SIX", 7 => "SEVEN", 8 => "EIGHT", 9 => "NINE",
        10 => "TEN", 11 => "ELEVEN", 12 => "TWELVE", 13 => "THIRTEEN", 14 =>
        "FOURTEEN", 15 => "FIFTEEN", 16 => "SIXTEEN", 17 => "SEVENTEEN", 18 =>
        "EIGHTEEN", 19 => "NINETEEN"
    );

    $tens = array(
        0 => "", 1 => "TEN", 2 => "TWENTY", 3 => "THIRTY", 4 => "FORTY", 5 =>
        "FIFTY", 6 => "SIXTY", 7 => "SEVENTY", 8 => "EIGHTY", 9 => "NINETY"
    );

    $hundreds = array(
        "HUNDRED", "THOUSAND", "MILLION", "BILLION", "TRILLION",
        "QUADRILLION"
    );

    $num = number_format($num, 2, ".", "");
    $num_arr = explode(".", $num);
    $wholenum = $num_arr[0];
    $decnum = $num_arr[1];
    $whole_arr = array_reverse(explode("", $wholenum));
    krsort($whole_arr, 1);
    $rettxt = "";

    foreach ($whole_arr as $key => $i) {
        while (substr($i, 0, 1) == "0")
            $i = substr($i, 1, 5);
        if ($i < 20) {
            if ($i != 0) {
                $rettxt .= $ones[$i];
            }
        } elseif ($i < 100) {
            $rettxt .= $tens[substr($i, 0, 1)];
            if (substr($i, 1, 1) != "0") {
                $rettxt .= " " . $ones[substr($i, 1, 1)];
            }
        } else {
            $rettxt .= $ones[substr($i, 0, 1)] . " " . $hundreds[0];
            if (substr($i, 1, 1) != "0") {
                $rettxt .= " " . $tens[substr($i, 1, 1)];
            }
            if (substr($i, 2, 1) != "0") {
                $rettxt .= " " . $ones[substr($i, 2, 1)];
            }
        }
    }
}
```



```

        if ($key > 0 && trim($rettxt) != "") {
            $rettxt .= " " . $hundreds[$key] . " ";
        }
    }

    if ($decnum > 0) {
        $rettxt .= " AND ";
        if ($decnum < 20) {
            $rettxt .= $ones[$decnum];
        } elseif ($decnum < 100) {
            $rettxt .= $tens[substr($decnum, 0, 1)];
            if (substr($decnum, 1, 1) != "0") {
                $rettxt .= " " . $ones[substr($decnum, 1, 1)];
            }
        }
    }
    } else {
        $rettxt .= " ONLY";
    }

    return $rettxt;
}
?>

```

CREATE_USER.PHP

```

<?php
$servername = "localhost";
$db_username = "root";
$db_password = "root";
$dbname = "krctnodues";

// Create connection
$conn = new mysqli($servername, $db_username, $db_password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Check if form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST['username']);
    $password = trim($_POST['password']);
    $name = strtoupper(trim($_POST['name']));

    // Validate input
    if (!empty($username) && !empty($password) && !empty($name)) {
        // Use prepared statements to prevent SQL injection
        $stmt = $conn->prepare("SELECT * FROM login1 WHERE username = ?");
        $stmt->bind_param("s", $username);
    }
}

```

```

$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    // Username already exists
    $status = "Username '$username' is already taken. Please choose a different
one.";
} else {
    // SQL query to insert new user into the database
    $stmt = $conn->prepare("INSERT INTO login1 (username, password, name)
VALUES (?, ?, ?)");
    $stmt->bind_param("sss", $username, $password, $name);

    if ($stmt->execute()) {
        $status = "User created successfully!";
    } else {
        $status = "Error: " . $stmt->error;
    }
}
$stmt->close();
} else {
    $status = "All fields are required.";
}
// Close connection
$conn->close();
}
?>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="icon" type="image/png" href="images/icons/favicon.ico"/>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Create User</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            background-color: #f4f4f4;
        }
        .container {
            width: 450px;
            padding: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
            background-color: #fff;

```

```

    }
    input[type="text"],
    input[type="password"] {
        width: 100%;
        padding: 10px;
        margin-bottom: 10px;
        box-sizing: border-box;
    }
    input[type="submit"] {
        width: 100%;
        padding: 10px;
        background-color: #007bff;
        color: #fff;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }
    .status {
        margin-top: 10px;
        font-weight: bold;
        color: green;
    }
}
</style>
</head>
<body>
    <div class="container">
        <h2>Create User</h2>
        <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>"
method="post">
            <input type="text" name="username" placeholder="Enter Username" required>
            <input type="password" name="password" placeholder="Enter Password"
required>
            <input type="text" name="name" placeholder="Enter Name" required>
            <div class="status"><?php echo isset($status) ? htmlspecialchars($status) : "<
?></div>
            <input type="submit" value="Create User">
        </form>
        <form action="index.html" method="post">
            <br>
            <input type="submit" value="Login">
        </form>
    </div>
    <h3>Name will be 'FULL NAME AND SPACE initial' respectively Example:
"SASIKANTH U"</h3>
    </div>
</div>
</body>
</html>

```

CONNECT.PHP

```
<?php
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "krctnodues";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
} else {
    echo "Connected successfully";
}
?>
```

CHANGE_PASSWORD.PHP

```
<?php
// Connect to your database
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "krctnodues";

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Initialize message
$message = "";
$showPasswordField = false;
$username = "";

// Check if username form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["username"]))
{
    $username = $_POST["username"];

    // Fetch the username from the database
    $sql = "SELECT username FROM login1 WHERE username = '$username'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
```

```

        $showPasswordField = true;
    } else {
        $message = "Username not found.";
    }
}

// Check if password form is submitted and all necessary fields are set
if ($showPasswordField && $_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST["current_password"])) {
    // Retrieve input values from the form
    $currentPassword = $_POST["current_password"];
    $newPassword = $_POST["new_password"];
    $confirmPassword = $_POST["confirm_password"];
    $username = $_POST["username"];

    // Fetch the current password from the database for the given username
    $sql = "SELECT password FROM login1 WHERE username = '$username'";
    $result = $conn->query($sql);

    // Check if the query returned any rows
    if ($result->num_rows > 0) {
        // Extract the password hash from the fetched row
        $row = $result->fetch_assoc();
        $hashedCurrentPassword = $row["password"];

        // Verify if the entered current password matches the stored password hash
        if (password_verify($currentPassword, $hashedCurrentPassword)) {
            // Check if the new password and confirm password match
            if ($newPassword === $confirmPassword) {
                // Hash the new password for storage
                $hashedNewPassword = password_hash($newPassword,
PASSWORD_DEFAULT);

                // Update the password in the database for the given username
                $sql = "UPDATE login1 SET password = '$hashedNewPassword' WHERE
username = '$username'";
                if ($conn->query($sql) === TRUE) {
                    // Password update successful
                    $message = "Password successfully changed.";
                    // Reset values to clear the form
                    $username = "";
                    $showPasswordField = false;
                } else {
                    // Error updating password in the database
                    $message = "Error updating password: " . $conn->error;
                }
            } else {
                // New password and confirm password do not match
                $message = "New password and confirm password do not match.";
            }
        } else {

```

```

        // Incorrect current password entered
        $message = "Current password is incorrect.";
    }
} else {
    // Username not found in the database
    $message = "Error: Username not found.";
}
}

// Close database connection
$conn->close();

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change Password</title>
    <link rel="icon" type="image/png" href="images/icons/favicon.ico"/>
    <style>
        /* Internal CSS */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f2f2f2;
            background-image: url('bg-05.jpeg');
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
        .change-password-form {
            width: 350px;
            padding: 50px;
            border: 1px solid #ccc;
            border-radius: 10px;
            text-align: center;
            background-color: #fff;
        }
        .change-password-form h3 {
            color: #333;
            font-size: 24px;
            margin-bottom: 15px;
        }
        .change-password-form .input-group {
            position: relative;
            margin-bottom: 15px;
        }

```

```

.change-password-form input[type="text"],
.change-password-form input[type="password"],
.change-password-form input[type="submit"] {
    width: 100%;
    padding: 10px;
    margin: 5px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 16px;
}
.toggle-password {
    position: absolute;
    right: 10px;
    top: 50%;
    transform: translateY(-50%);
    cursor: pointer;
    color: #888;
}
.toggle-password:hover {
    color: #333;
}
.message {
    color: red;
    font-size: 14px;
    margin-top: 10px;
}
</style>
</head>
<body>
    <form class="change-password-form" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>" method="post">
        <h3>Change Password</h3>
        <?php if (!$showPasswordField): ?>
            <div class="input-group">
                <input type="text" name="username" id="username" placeholder="Username"
value="<?php echo $username; ?>" required>
            </div>
            <input type="submit" value="Submit Username">
        <?php else: ?>
            <input type="hidden" name="username" value="<?php echo $username; ?>">
            <div class="input-group">
                <input type="password" name="current_password" id="current_password"
placeholder="Current Password" required>
                <span class="toggle-password"
onclick="togglePasswordVisibility('current_password')">👁</span>
            </div>
            <div class="input-group">
                <input type="password" name="new_password" id="new_password"
placeholder="New Password" required>
                <span class="toggle-password"
onclick="togglePasswordVisibility('new_password')">👁</span>

```

```

    </div>
    <div class="input-group">
        <input type="password" name="confirm_password" id="confirm_password"
placeholder="Confirm Password" required>
        <span class="toggle-password"
onclick="togglePasswordVisibility('confirm_password')">👁️</span>
    </div>
    <input type="submit" value="Change Password">
<?php endif; ?>
<?php
if ($message) {
    echo "<p class='message'>$message</p>";
}
?>
</form>
<script>
function togglePasswordVisibility(id) {
    var passwordField = document.getElementById(id);
    var icon = passwordField.nextElementSibling;
    if (passwordField.type === "password") {
        passwordField.type = "text";
        icon.textContent = "🔒"; // Change to closed eye icon
    } else {
        passwordField.type = "password";
        icon.textContent = "👁️"; // Change to open eye icon
    }
}
</script>
</body>
</html>

```

STYLE.CSS

```

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f2f2f2;
}

.container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.box {
    border: 1px solid #ccc;
    border-radius: 5px;

```



```

padding: 20px;
margin: 20px;
background-color: #fff;
}

table {
width: 100%;
border-collapse: collapse;

}

th, td {
padding: 8px;
text-align: left;
border-bottom: 1px solid #ddd;
}

th {
background-color: #f2f2f2;
}

button {
padding: 10px 20px;
background-color: #007bff;
color: #fff;
border: none;
border-radius: 5px;
cursor: pointer;
font-size: 18px;
}

```

WELCOME.PHP

```

<?php
// Start the session
session_start();

// Check if user is not logged in
if (!isset($_SESSION["username"])) {
    // If not logged in, redirect to login page
    header("Location: index.html");
    exit;
}

// Retrieve the user's name from the session or database
$name = ""; // Initialize variable
if (isset($_SESSION["name"])) {
    // If the name is already stored in the session, retrieve it
    $name = $_SESSION["name"];
} else {

```

```

// If name is not in the session, retrieve it from the database
// Connect to your database (replace with your actual database credentials)
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "krctnodues";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Prepare SQL statement to retrieve user's name
$username = $_SESSION["username"];
$sql = "SELECT name FROM login1 WHERE username = '$username'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Output data of each row
    while($row = $result->fetch_assoc()) {
        $name = $row["name"];
        // Store name in session for future use
        $_SESSION["name"] = $name;
    }
} else {
    echo "Error: Username not found in database.";
}

// Close connection
$conn->close();
}

// Prevent caching of this page
header("Cache-Control: no-cache, must-revalidate");
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome</title>
    <style>
        /* Internal CSS */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;

```

```

    background-color: #f2f2f2;
    background-image: url('bg-05.jpeg');
}
.container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 30vh;
}
.welcome {
    text-align: center;
}
.welcome h2 {
    color: black;
    font-size: 36px;
    margin-bottom: 10px;
}
.box {
    width: 300px;
    padding: 25px;
    border: 1px solid #ccc;
    border-radius: 5px;
    text-align: center;
    background-color: #fff;
    margin: 0 20px;
    padding: 20px;
}
.box h3 {
    color: #333;
    font-size: 24px;
    margin-bottom: 15px;
}
.button {
    padding: 10px 20px;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 18px;
}
/* Logout button styling */
.logout-btn {
    position: absolute;
    top: 20px;
    right: 20px;
}
.logout-btn input[type="submit"] {
    padding: 10px 20px;
    background-color: #007bff;
    color: #fff;

```

```

        border: none;
        border-radius: 5px;
        cursor: pointer;
        font-size: 18px;
    }
</style>
<link rel="icon" type="image/png" href="images/icons/favicon.ico"/>
</head>
<body>
    <!-- Logout button -->
    <div class="logout-btn">
        <form action="logout.php" method="post">
            <input type="submit" value="Logout">
        </form>
    </div>

    <div class="container">
        <div class="welcome">
            <h2>Welcome <?php echo $name; ?></h2>
        </div>
    </div>
    <div class="container">
        <div class="box">
            <h3>KRCT Zero Dues</h3>
            <p>It will generate the nodue form then click button download form</p>
            <button class="button"
onclick="window.location.href='generate_zero_dues.php'">Generate</button>
        </div>
        <div class="box">
            <h3>Check Balance</h3>
            <p>It will move to the page that displays balance</p>
            <button
class="button"onclick="window.location.href='check_balance.php'">Check</button>
        </div>
    </div>
    <script>
        // Using JavaScript to prevent going back to the previous page (index.html)
        history.pushState(null, null, window.location.href);
        window.onpopstate = function () {
            history.go(1);
        };
    </script>
</body>
</html>

```

CHECK_BALANCE.PHP

```
<?php
// Start the session
session_start();

// Check if user is logged in
if (!isset($_SESSION["username"])) {
    // If not logged in, redirect to login page
    header("Location: index.html");
    exit;
}

// Retrieve the username from the session
$username = $_SESSION["username"];

// Connect to the database (replace with your actual database credentials)
$servername = "localhost";
$db_username = "root";
$db_password = "root";
$dbname = "krctnodues";

// Create connection
$conn = new mysqli($servername, $db_username, $db_password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Initialize variables
$name = "";
$department = "";
$tuition_fee = "";
$hostel_fee = "";
$bus_fee = "";
$miscellaneous_fee = "";
$mess_fee = "";
$exam_fee = "";
$library_fee = "";
$fine = "";
$total_amount = "";

// Prepare SQL statement to retrieve user's name
$sql = "SELECT name FROM login1 WHERE username = '$username'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Fetch user's name
    $row = $result->fetch_assoc();
    $name = $row["name"];
```

```

// Prepare SQL statement to retrieve user details from student table
$sql = "SELECT * FROM student1 WHERE name = '$name'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Fetch user details
    $row = $result->fetch_assoc();
    $department = $row["department"];
    $tuition_fee = $row["tutionfee"];
    $hostel_fee = $row["hostelfee"];
    $bus_fee = $row["busfee"];
    $miscellaneous_fee = $row["miscellaneousfee"];
    $mess_fee = $row["messfee"];
    $exam_fee = $row["examfee"];
    $library_fee = $row["libraryfee"];
    $fine = $row["fine"];
    // Calculate total amount
    $total_amount = $tuition_fee + $hostel_fee + $bus_fee + $miscellaneous_fee +
    $mess_fee + $exam_fee + $library_fee + $fine;

    // Include the number to words function
    include('number_to_words.php');

    // Convert total amount to words
    $total_amount_in_words = numberToWords($total_amount);
} else {
    // User details not found in student table
    $error_message = "User details not found in the database.";
    exit($error_message);
}
} else {
    // Username not found in login table
    $error_message = "Username not found in the database.";
    exit($error_message);
}

// Close connection
$conn->close();
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Check Balance</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="icon" type="image/png" href="images/icons/favicon.ico"/>
</head>
<body>

```

```

<div class="container">
  <div class="box">
    <h2>Check Balance</h2>
    <table>
      <tr>
        <th>Name</th>
        <th>Department</th>
        <th>Tuition Fee</th>
        <th>Hostel Fee</th>
        <th>Bus Fee</th>
        <th>Miscellaneous Fee</th>
        <th>Mess Fee</th>
        <th>Exam Fee</th>
        <th>Library Fee</th>
        <th>Fine</th>
      </tr>
      <tr>
        <td><?php echo $name; ?></td>
        <td><?php echo $department; ?></td>
        <td><?php echo $tuition_fee; ?></td>
        <td><?php echo $hostel_fee; ?></td>
        <td><?php echo $bus_fee; ?></td>
        <td><?php echo $miscellaneous_fee; ?></td>
        <td><?php echo $mess_fee; ?></td>
        <td><?php echo $exam_fee; ?></td>
        <td><?php echo $library_fee; ?></td>
        <td><?php echo $fine; ?></td>
      </tr>
      <tr>
        <th colspan="9">Total</th>
        <td><?php echo $total_amount; ?></td>
      </tr>
      <tr>
        <th colspan="1">Total in Words</th>
        <td colspan="9"><?php echo $total_amount_in_words; ?></td>
      </tr>
    </table>
    <button onclick="window.location.href='generate_zero_dues.php'">Generate
Zero Dues</button>
  </div>
</div>
</body>
</html>

```

GENERATE_ZERO_DUES.PHP

```
<?php

// Include the number to words function
include('number_to_words.php');

// Set the time zone to Asia/Kolkata
date_default_timezone_set('Asia/Kolkata');

require('fpdf186/fpdf.php');

// Retrieve data from the database
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "krctnodues";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Retrieve the username from the session
session_start();
if (!isset($_SESSION["username"])) {
    // If not logged in, redirect to login page
    header("Location: index.html");
    exit;
}
$username = $_SESSION["username"];

// SQL query to retrieve student details based on username
$sql = "SELECT s.name, s.department, s.tutionfee, s.hostelfee, s.busfee,
s.miscellaneousfee,s.messfee,s.examfee,s.libraryfee,s.fine
FROM login l
JOIN student1 s ON l.name = s.name
WHERE l.username = '$username'";
$result = $conn->query($sql);

// Initialize variables for student details
$name = "";
$department = "";
$tutionfee = "";
$hostelfee = "";
$busfee = "";
$miscellaneousfee = "";
$mess_fee="";
```



```

$exam_fee="";
$library_fee="";
$fine="";

if ($result->num_rows > 0) {
    // Fetch student details
    $row = $result->fetch_assoc();
    $name = $row["name"];
    $department = $row["department"];
    $tutionfee = $row["tutionfee"];
    $hostelfee = $row["hostelfee"];
    $busfee = $row["busfee"];
    $miscellaneousfee = $row["miscellaneousfee"];
    $mess_fee=$row["messfee"];
    $exam_fee=$row["examfee"];
    $library_fee=$row["libraryfee"];
    $fine=$row["fine"];

    // Calculate total amount
    $total_amount = $tutionfee + $hostelfee + $busfee +
    $miscellaneousfee+$mess_fee+$exam_fee+$library_fee+$fine;

    // Convert total amount to words
    $total_amount_in_words = numberToWords($total_amount);

} else {
    // No data found in the database
    $error_message = "No data found in the database for the logged-in user.";
}

// Close database connection
$conn->close();

if (!isset($error_message)) {
    // Create new PDF document
    class PDF extends FPDF {
        function Header() {
            // Add the date and time to the header
            // $this->SetFont('Arial', 'T', 8);
            // $this->Cell(0, 10, date('Y-m-d H:i:s A'), 0, 1, 'R');
            // Add the date and time to the header
            $this->SetFont('Arial', 'T', 8);
            $this->Cell(0, 10, date('d-m-Y h:i:s A'), 0, 1, 'R');
        }
    }

    $pdf = new PDF();
    $pdf->AddPage();

    // Set font
    $pdf->SetFont('Arial', 'B', 16);

```

```

// Title
$pdf->Cell(0, 10, 'Zero Dues', 0, 1, 'C');

// Line break
$pdf->Ln(10);

// Set font
$pdf->SetFont('Arial', '', 12);

$pdf->Cell(70, 10, 'Name', 1);
$pdf->Cell(70, 10, $name, 1);
$pdf->Ln();

$pdf->Cell(70, 10, 'Department', 1);
$pdf->Cell(70, 10, $department, 1);
$pdf->Ln();

// Table rows
$pdf->Cell(70, 10, 'Tuition Fee', 1);
$pdf->Cell(70, 10, $tutionfee, 1);
$pdf->Ln();

$pdf->Cell(70, 10, 'Hostel Fee', 1);
$pdf->Cell(70, 10, $hostelfee, 1);
$pdf->Ln();

$pdf->Cell(70, 10, 'Bus Fee', 1);
$pdf->Cell(70, 10, $busfee, 1);
$pdf->Ln();

$pdf->Cell(70, 10, 'Miscellaneous Fee', 1);
$pdf->Cell(70, 10, $miscellaneousfee, 1);
$pdf->Ln();

$pdf->Cell(70, 10, 'Mess Fee', 1);
$pdf->Cell(70, 10, $mess_fee, 1);
$pdf->Ln();

$pdf->Cell(70, 10, 'Exam Fee', 1);
$pdf->Cell(70, 10, $exam_fee, 1);
$pdf->Ln();

$pdf->Cell(70, 10, 'Library Fee', 1);
$pdf->Cell(70, 10, $library_fee, 1);
$pdf->Ln();

$pdf->Cell(70, 10, 'Fine', 1);
$pdf->Cell(70, 10, $fine, 1);
$pdf->Ln();

```

```

// Total amount row
$pdf->Cell(70, 10, 'Total', 1, 0, 'L');
$pdf->Cell(70, 10, $total_amount, 1, 1);

$pdf->Ln();
// Total amount in words
$pdf->Cell(0, 10, 'Total Amount in Words: ' . $total_amount_in_words, 0, 1, 'L');

// Line break
$pdf->Ln(15);
// Student signature
$pdf->Cell(0, 10, 'Student Signature', 0, 1, 'L');

// Staff Incharge sign
$pdf->Cell(0, 10, 'Staff Signature', 0, 1, 'R');

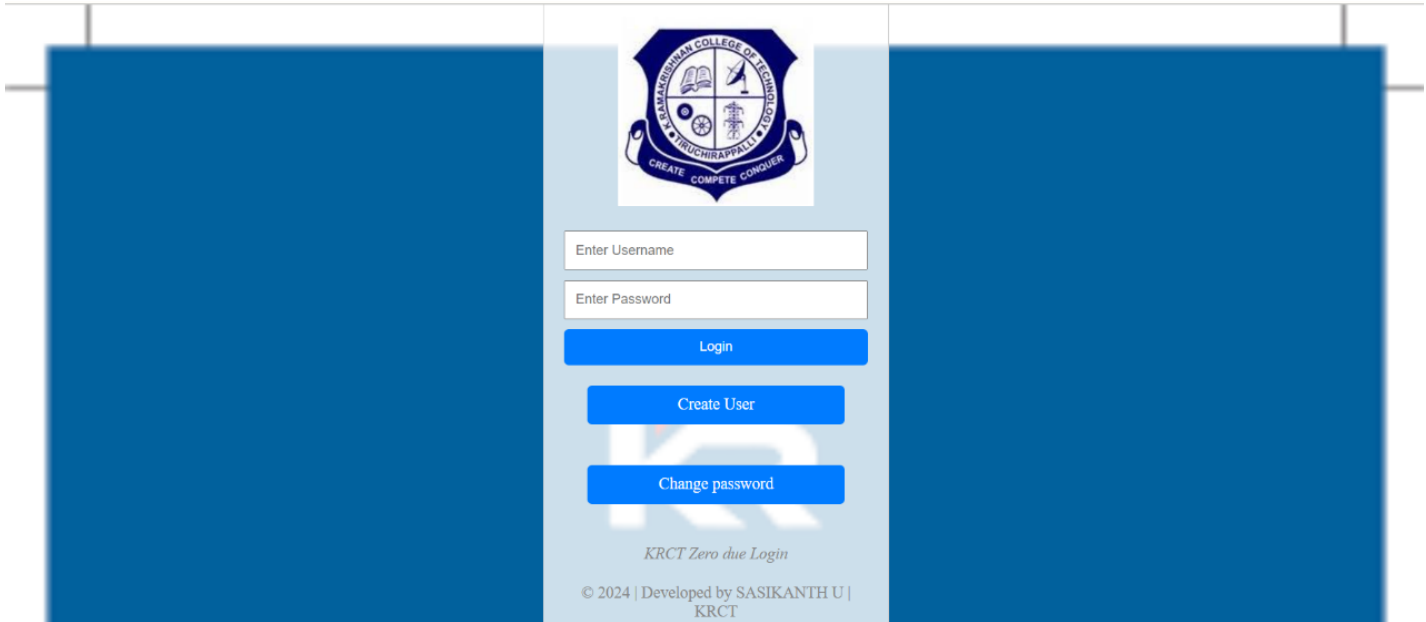
// Output PDF
$pdf->Output('no_dues_form '.$name.'.pdf', 'D');
} else {
    echo $error_message;
}
?>

```

APPENDIX B

SAMPLE OUTPUT

LOGIN PAGE



Enter Username

Enter Password

Login

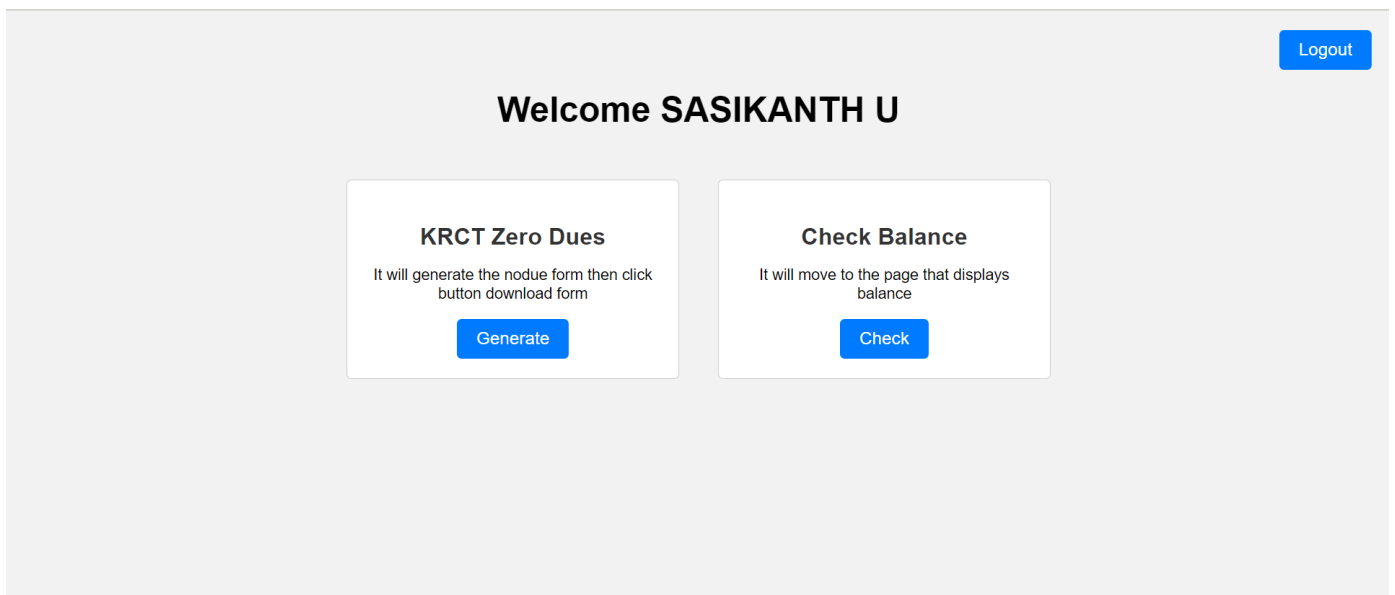
Create User

Change password

KRCT Zero due Login

© 2024 | Developed by SASIKANTH U |
KRCT

WELCOME PAGE



Logout

Welcome SASIKANTH U

KRCT Zero Dues
It will generate the nodue form then click
button download form
Generate

Check Balance
It will move to the page that displays
balance
Check

CHECK BALANCE PAGE

Check Balance

Name	Department	Tuition Fee	Hostel Fee	Bus Fee	Miscellaneous Fee	Mess Fee	Exam Fee	Library Fee	Fine
SASIKANTH U	CSE	20000	0	15000	3050	0	2500	1000	2000
Total									43550
Total in Words		FORTY THREE THOUSAND FIVE HUNDRED FIFTY ONLY							
Generate Zero Dues									

GENERATE KRCT ZERO DUES PAGE

localhost/krct_zero_dues/check_balance.php

no_dues_form SASIKANTH U (41).pdf
2,136 B • Done

Check Balance

Name	Department	Tuition Fee	Hostel Fee	Bus Fee	Miscellaneous Fee	Mess Fee	Exam Fee	Library Fee	Fine
SASIKANTH U	CSE	20000	0	15000	3050	0	2500	1000	2000
Total									43550
Total in Words		FORTY THREE THOUSAND FIVE HUNDRED FIFTY ONLY							
Generate Zero Dues									

ZERO DUES .PDF PAGE

no_dues_form SASIKANTH U (41).pdf 1 / 1 65%

02-06-2024 09:41:39 PM

Zero Dues

Name	SASIKANTH U
Department	CSE
Tuition Fee	20000
Hostel Fee	0
Bus Fee	15000
Miscellaneous Fee	3050
Mess Fee	0
Exam Fee	2500
Library Fee	1000
Fine	2000
Total	43550

Total Amount in Words: FORTY THREE THOUSAND FIVE HUNDRED FIFTY ONLY

Student Signature

Staff Signature

CREATE USER PAGE

Create User

Enter Username

Enter Password

Enter Name

Create User

Login

Name will be 'FULL NAME AND SPACE initial' respectively Example: "SASIKANTH U"

CHANGE PASSWORD PAGE

Change Password

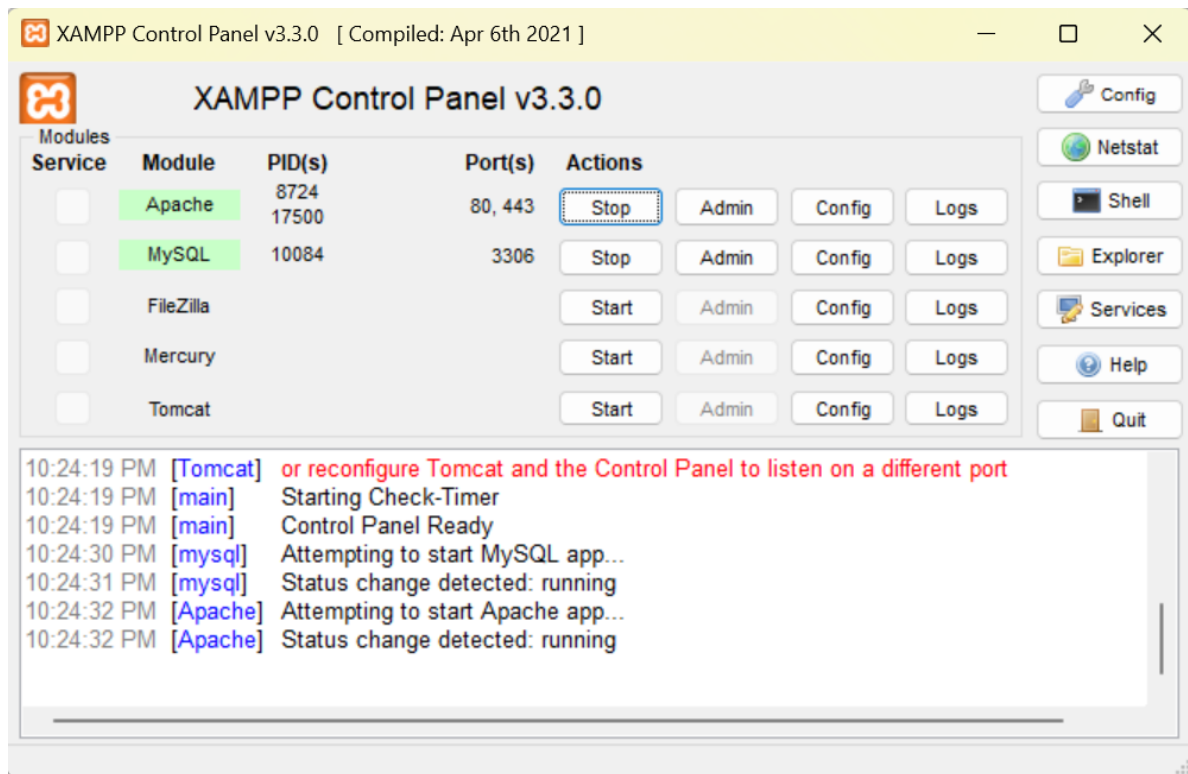
Submit Username

CHANGE PASSWORD NEXT PAGE

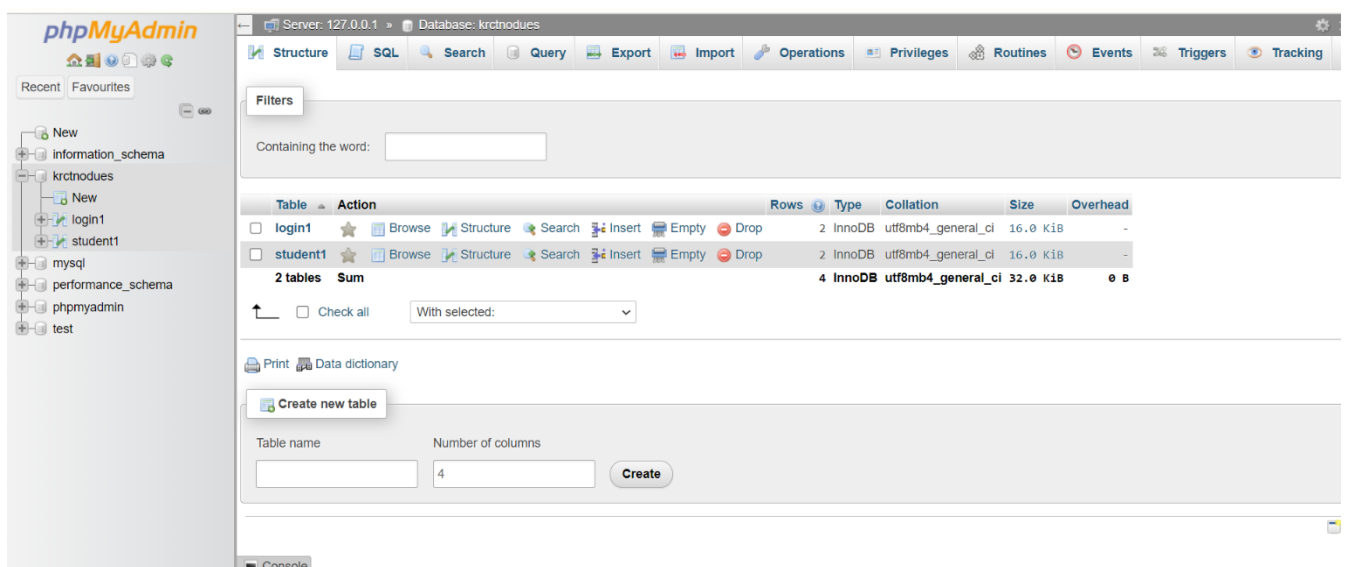
Change Password

Change Password












STARTING XAMPP SERVER














KRCT ZERO DUES - DATABASE



































LOGIN TABLE STRUCTURE










<div>  Table structure <div>  Relation view </div> </div>									
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	username	varchar(250)	utf8mb4_general_ci		No	None		 Change  Drop  More
<input type="checkbox"/>	2	password	varchar(250)	utf8mb4_general_ci		No	None		 Change  Drop  More
<input type="checkbox"/>	3	name	varchar(250)	utf8mb4_general_ci		No	None		 Change  Drop  More




☐ Check all
 With selected:
  Browse
  Change
  Drop
  Primary
  Unique
  Index
  Spatial
  Fulltext

 Add to central columns
  Remove from central columns

STUDENT TABLE STRUCTURE

<div>  Table structure <div>  Relation view </div> </div>									
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	name	varchar(250)	utf8mb4_general_ci		No	None		 Change  Drop  More
<input type="checkbox"/>	2	department	varchar(250)	utf8mb4_general_ci		No	None		 Change  Drop  More
<input type="checkbox"/>	3	tutionfee	bigint(250)		No	None			 Change  Drop  More
<input type="checkbox"/>	4	hostelfee	bigint(250)		No	None			 Change  Drop  More
<input type="checkbox"/>	5	busfee	bigint(250)		No	None			 Change  Drop  More
<input type="checkbox"/>	6	miscellaneousfee	bigint(250)		No	None			 Change  Drop  More
<input type="checkbox"/>	7	messfee	bigint(250)		No	None			 Change  Drop  More
<input type="checkbox"/>	8	examfee	bigint(250)		No	None			 Change  Drop  More
<input type="checkbox"/>	9	libraryfee	bigint(250)		No	None			 Change  Drop  More
<input type="checkbox"/>	10	fine	bigint(250)		No	None			 Change  Drop  More


☐ Check all
 With selected:
  Browse
  Change
  Drop
  Primary
  Unique
  Index
  Spatial
  Fulltext

 Add to central columns
  Remove from central columns

REFERENCES

1. **PHP Documentation:**

Official documentation for PHP, which includes detailed explanations and examples.

[PHP Manual](#)

2. **MySQL Documentation:**

Comprehensive guide to using MySQL, useful for database design and queries.

[MySQL Reference Manual](#)

3. **XAMPP Documentation:**

Instructions for setting up a local development environment using XAMPP, which includes Apache, PHP, and MySQL.

[XAMPP Documentation](#)

4. **HTML and CSS:**

Basic resources to design the login and welcome pages.

[W3Schools HTML Tutorial](#)

[W3Schools CSS Tutorial](#)

5. **FPDF Library:**

A PHP class to generate PDF files, useful for creating the no dues form.

[FPDF Documentation](#)

6. **PHP Login Script Tutorial:**

A step-by-step guide to creating a secure login system using PHP and MySQL.

[PHP Login System](#)

7. **Session Management in PHP:**

Understanding how to manage user sessions for login functionality.

[PHP Sessions](#)

8. SQL Injection Prevention:

Techniques to protect your application from SQL injection attacks.

[SQL Injection Prevention](#)

9. Password Hashing in PHP:

Best practices for securely storing user passwords.

[PHP](#)

10. PHP Function to Convert Number to Words:

Implementation of converting numerical amounts to words.

[PHP Number to Words](#)

11. PHP PDO Tutorial:

Using PHP Data Objects (PDO) for secure database interactions.

[PHP PDO Tutorial](#)

12. JavaScript and AJAX:

Enhancing user experience with asynchronous requests.

[W3Schools AJAX Tutorial](#)

13. GitHub Repositories for PHP Projects:

Explore open-source PHP projects for reference and inspiration.

[GitHub PHP Projects](#)

14. Form Handling in PHP:

Learn how to handle form submissions securely and efficiently.

[PHP Form Handling](#)

15. FPDF Tutorials and Examples:

Additional resources for working with FPDF to create PDF documents.

[FPDF Examples](#)