**Chapter Four**

## Implementation

## 4.1. Introduction/Overview

Implementation of a given data model involves the physical realization of the abstract machine components on a real machine. This chapter aims to provide a comprehensive overview of the issues addressed during the implementation phase of the project.

During implementation, the focus is on translating the conceptual design and logical model into a tangible, working solution. It involves converting the design specifications and requirements into executable code that can be executed on a computer system.

The main objective of the implementation phase is to bring the system to life by constructing the actual software or hardware components that constitute the data model. This phase requires careful attention to detail, as it involves writing code, configuring hardware, setting up databases, and integrating various components.

Some of the key aspects addressed during the implementation phase include:

**1. Coding**: Writing the actual software code based on the chosen programming languages and frameworks. This involves translating the logical model and design specifications into executable code that performs the desired operations and functionalities.

**2. Hardware Configuration**: Setting up and configuring the necessary hardware components required for the implementation. This includes servers, storage devices, networking equipment, and any other hardware infrastructure required to support the system.

**3. Database Implementation**: Creating and configuring the database that will store and manage the data for the system. This involves defining the database schema, creating tables, specifying relationships, and implementing any necessary optimizations for efficient data storage and retrieval.

**4. Integration**: Connecting and integrating various software and hardware components to ensure they work together seamlessly. This includes integrating different modules, APIs, libraries, and third-party services to enable smooth communication and functionality.

**5. Testing:** Conducting thorough testing and quality assurance procedures to ensure that the implemented system functions as expected. This includes unit testing, integration testing, system testing, and performance testing to identify and rectify any issues or bugs.

**6. Deployment**: Deploying the implemented system to the target environment or production environment. This involves installing and configuring the software, setting up the necessary infrastructure, and ensuring the system is operational and accessible to users.

The implementation phase is a critical stage in the project lifecycle as it transforms the abstract concepts and designs into a concrete and functional system. It requires close collaboration between developers, database administrators, system administrators, and other stakeholders involved in the implementation process.

By effectively addressing the various aspects and challenges of implementation, the project can progress towards successful completion and deliver a fully functional and optimized system.

### **4.2. Tools and Technology Utilized for Implementation**

**1. PHP (Php hypertext preprocess):** is a scripting language used on the server side, allowing developers to create dynamic and interactive web pages quickly. PHP scripts are executed on the server and can be written using a simple text editor like Notepad. It is compatible with various servers like Apache and IIS, as well as different platforms such as Windows, Linux, Unix, and Mac OS X.

**2. MYSQL (my structure query language):** is a widely used database management system that forms the backend layer of a system. It enables developers to create and manage databases, providing efficient storage and retrieval of data for software applications. MYSQL facilitates communication between software applications and databases, whether they are on the same computer or across a network, including the Internet.

**3. CSS (Cascading Style Sheets):** is a text document that contains rules defining how elements on an HTML page should be formatted. It allows developers to control the visual appearance of web elements by setting properties such as positioning, sizing, font, and color. CSS enhances the presentation of web pages, enabling developers to create visually appealing and consistent designs across multiple web pages.

**4. HTML (Hyper Text Markup Language):** is the standard markup language used to create web pages. It provides the structural framework for web content and allows web browsers to interpret and display the content correctly. HTML uses tags to define elements and their relationships, providing semantic information about the content. It is not a programming language but a markup language that focuses on structuring web documents.

**5. JavaScript:** is a scripting language that adds interactivity and dynamic functionality to static web applications. It enhances the user experience by enabling personalized and interactive content on web pages. JavaScript is supported by all major web browsers and is widely used for creating features like drop-down menus, animations, form validation, and real-time updates. It enhances the usability and engagement of websites, making them more appealing to visitors.

**4.3. System implementation**

System implementation involves a series of steps to effectively transition from the existing system to the new system. This phase ensures a smooth conversion process and provides necessary documentation for users. Here are the key steps involved:

**A. System Conversion:**

System conversion is a critical step in the implementation process. It involves transitioning from the existing system to the new system or migrating from a manual system to a web-based system (website). Depending on the complexity of the project, different conversion strategies can be employed, such as:

**1. Direct Conversion:** This strategy involves replacing the existing system with the new system in one go. It requires careful planning and coordination to ensure a seamless transition and minimize disruptions to the business operations.

**2. Phased Conversion**: In a phased conversion approach, the new system is implemented in stages or phases. This allows for incremental adoption and provides opportunities to address any issues or challenges that may arise during the implementation process.

**3. Parallel Conversion:** Parallel conversion involves running both the existing system and the new system simultaneously for a certain period. This allows for a gradual transition and provides a fallback option in case any issues arise with the new system.

The chosen conversion strategy depends on factors such as project scope, complexity, resource availability, and organizational readiness.

**B. User Documentation**:

User documentation is a crucial aspect of system implementation. It provides users with a point of reference for understanding and effectively utilizing the new system. The documentation should be clear, concise, and accessible to ensure users can easily navigate and make the most out of the system. Key elements of user documentation include:

**1. User Manuals**: These documents provide step-by-step instructions on how to use the system, including navigation, data entry, and performing specific tasks or functions. User manuals should be well-organized and include screenshots or illustrations to aid understanding.

**2. Training Materials**: Training materials, such as presentations or videos, are created to facilitate user training sessions. These materials should cover the system's key features, functionalities, and best practices for optimal usage.

**3. FAQs and Troubleshooting Guides**: Frequently Asked Questions (FAQs) and troubleshooting guides address common user queries and provide solutions to common issues or errors that may arise during system usage. These resources help users troubleshoot problems independently before seeking additional support.

**4. Help Desk or Support Channels**: Establishing a help desk or support channels, such as a dedicated email address or ticketing system, ensures users have a means to reach out for assistance or report system-related issues. Clear guidelines should be provided on how to access and utilize these support channels effectively.

User documentation serves as a valuable resource for users, empowering them to utilize the system effectively and overcome any challenges they may encounter. It plays a vital role in ensuring a successful implementation and user adoption of the new system.

In summary, system implementation involves careful system conversion from the existing system to the new system or from a manual system to a web-based system. User documentation plays a crucial role in providing users with the necessary guidance and support to effectively utilize the new system. By following these steps, organizations can ensure a successful implementation and smooth transition to the new system.

## System Interfaces

System interfaces are an essential component of system implementation, as they facilitate communication and interaction between different software components or systems. These interfaces define the methods, protocols, and formats through which systems exchange information and collaborate. Here are some key aspects related to system interfaces:

**1. Interface Design:**

During system implementation, interface design focuses on defining the specifications and requirements for the interfaces between various system components. This involves identifying the necessary data inputs, outputs, and functionalities that need to be exchanged between different modules or systems. The design process also considers factors such as data formats, protocols, security measures, and error handling mechanisms.

**2. Application Programming Interfaces (APIs):**

APIs play a crucial role in system interfaces by providing a predefined set of rules and functionalities that allow different software components to interact with each other. APIs define the methods, parameters, and data formats that should be used to access and manipulate specific functionalities or data within a system. Implementing APIs ensures standardized and efficient communication between different components, enabling seamless integration and interoperability.

**3. User Interface (UI):**

The user interface is an important aspect of system interfaces, as it enables users to interact with the system and perform tasks. The UI design focuses on creating intuitive and user-friendly interfaces that facilitate efficient user interactions. This involves designing screens, forms, menus, and controls that allow users to input data, view information, and perform actions in a straightforward manner. The UI design should consider factors such as usability, accessibility, and responsiveness across different devices and platforms.

**4. Integration Interfaces:**

Integration interfaces are used when integrating different systems or components to enable seamless data exchange and collaboration. These interfaces define the methods and protocols for integrating systems, including data synchronization, message passing, or remote procedure calls. Integration interfaces ensure that data and functionalities can be shared and accessed across different systems, enabling them to work together as a unified solution.

**5. Third-Party Interfaces:**

In some cases, systems may need to integrate with third-party services or platforms, such as payment gateways, social media APIs, or external data sources. Third-party interfaces involve implementing the necessary protocols and standards to connect with these external services and exchange data or perform specific actions. This allows systems to leverage the capabilities of third-party services and enhance their functionalities.

Effective system interface design and implementation are crucial for achieving seamless communication and integration between system components. Well-designed interfaces ensure interoperability, flexibility, and scalability, allowing systems to adapt to changing requirements and integrate with external services or future enhancements.

In summary, system interfaces define the methods, protocols, and formats through which systems communicate and collaborate. Designing and implementing robust system interfaces, including APIs, user interfaces, integration interfaces, and third-party interfaces, is essential for successful system implementation and seamless interaction between different software components.

Add image here (landing page)

Fig 14 : Dashboard

## 4.3.2. System Algorithm

In order to ensure a secure login process and protect user information, the system employs a robust encryption algorithm. This algorithm is designed to encrypt sensitive data, such as passwords, during the login process. Additionally, the system utilizes a registration feature that includes a verification code sent to users. Here's an explanation of the system algorithm:

**1. Encryption Algorithm:**

The system employs a state-of-the-art encryption algorithm to safeguard user credentials, particularly passwords, during the login process. Encryption is a technique that transforms plain text into a format that is unreadable without the corresponding decryption key. By encrypting passwords, the system ensures that even if unauthorized access occurs, the stored passwords remain protected.

The encryption algorithm chosen for the system is carefully selected based on its strength, reliability, and resistance to potential attacks. It employs cryptographic techniques to convert the passwords into encrypted form, making it extremely challenging for any malicious entity to decipher them. This ensures that user passwords are securely stored within the system's database.

**2. Registration Process:**

The system includes a registration feature that allows users to create new accounts. During the registration process, users provide their personal information and choose a password. To enhance security, an additional layer of verification is implemented.

After user registration, the system generates a unique verification code. This code is sent to the user's provided contact information, such as their email address or mobile number. The verification code is typically delivered via email, SMS, or through an authentication app.

The user is required to enter the verification code within a specified time frame to confirm the authenticity of their registration. By requiring this verification step, the system ensures that only legitimate users gain access to the system.

**3. Security Measures:**

In addition to encryption and the registration verification process, the system employs various other security measures to enhance overall system security. These measures may include:

**- Secure Socket Layer (SSL) encryption**: The system may utilize SSL certificates to establish a secure connection between the user's device and the system's server, ensuring that data transmitted during the login process remains encrypted and protected from interception.

**- Password complexity requirements**: The system may enforce specific password complexity rules, such as minimum length, combination of alphanumeric characters, and inclusion of special characters, to ensure that users create strong and secure passwords.

**- Account lockout and password reset**: The system may implement mechanisms to detect and prevent brute force attacks by locking user accounts after a certain number of failed login attempts. It may also provide a password reset functionality to allow users to regain access to their accounts in case they forget their passwords.

**- Regular security updates**: The system undergoes regular updates and patches to address any identified security vulnerabilities and ensure that it remains protected against emerging threats.

By implementing a robust encryption algorithm, incorporating a registration verification process, and adopting additional security measures, the system aims to provide a secure login process and protect user information from unauthorized access.

In summary, the system employs a strong encryption algorithm to secure the login process and protect user credentials. Additionally, it includes a registration verification process that involves sending a verification code to the user. These measures, combined with other security features, enhance the overall security of the system and safeguard user information.

**4.3.3. Source Code**

ADD HERE

* 1. **DevOps Integration and Installation Procedures**

DevOps Integration and Installation Procedures for Bekur:

DevOps integration and installation procedures play a vital role in ensuring the successful implementation of Bekur. By adopting DevOps practices, organizations can streamline their development, deployment, and maintenance processes, enabling efficient collaboration between development and operations teams. The following considerations are crucial for integrating DevOps and installing Bekur;

**4.4.1. Continuous Integration and Deployment**

(on version control make changes after meeting)

Implementing a robust Continuous Integration and Deployment (CI/CD) pipeline is essential for efficient software development and release cycles. The CI/CD pipeline automates various stages, including code building, testing, and deployment, ensuring a smooth and reliable process. For Bekur, it is important to establish the following steps within the CI/CD pipeline:

**Version Control**: Utilize a version control system, such as Git, to manage the source code of Bekur. Implementing branching strategies like GitFlow allows for parallel development and feature isolation, facilitating collaboration among developers.

**Automated Testing:** Implement a comprehensive suite of automated tests, including unit tests, integration tests, and end-to-end tests. These tests should cover critical functionalities, edge cases, and performance benchmarks. Integrating the test suite into the CI/CD pipeline ensures that tests are automatically executed whenever code changes are made, guaranteeing the stability and reliability of Bekur.

**Continuous Integration**: Configure a CI server, such as Jenkins, GitLab CI/CD, or CircleCI, to monitor the code repository for changes. The CI server should automatically trigger builds, execute the test suite, and provide feedback on the code quality and test results. Additionally, it can perform code analysis, such as linting and code style checks, to enforce coding standards and best practices.

**Continuous Deployment**: Once the code passes the automated tests, the CI/CD pipeline should automatically deploy the application to the desired environment. This can be a staging environment for further testing or directly to the production environment.

**4.4.2 Infrastructure as Code (IaC)**

Implementing Infrastructure as Code (IaC) allows for the management and provisioning of Bekur's infrastructure in a consistent and reproducible manner. IaC tools, such as Terraform or AWS CloudFormation, enable the definition of infrastructure resources, configurations, and dependencies in code, simplifying infrastructure management.

**Define Infrastructure**: Utilize IaC tools to define and provision the necessary infrastructure components, such as servers, databases, networking, and storage resources, required for running Bekur. Leveraging cloud services like AWS, Azure, or Google Cloud Platform facilitates scalable and flexible infrastructure provisioning.

**Version Control for Infrastructure**: Store the IaC code in the same version control system as the application code. This ensures that infrastructure changes are tracked, reviewed, and managed effectively. Employing branching and tagging strategies allows for the management of different versions and environments.

**Automated Provisioning:** Configure the CI/CD pipeline to automatically provision the infrastructure based on the defined IaC code. This ensures consistency and eliminates manual setup, reducing the chance of errors and providing a reliable and reproducible infrastructure.

**4.4.3 Monitoring and Logging**

Implement robust monitoring and logging practices to ensure the stability and performance of Bekur. Monitoring tools, such as Prometheus or ELK stack (Elasticsearch, Logstash, and Kibana), can provide insights into the system's health, resource usage, and potential issues.

**- Instrumentation**: Integrate monitoring agents or libraries into Bekur code to collect relevant metrics and logs. This allows for real-time monitoring of the application's performance and behavior.

**- Log Management**: Configure centralized log management to aggregate and analyze logs from different components of Bekur. This facilitates troubleshooting and detecting anomalies or potential security breaches.

**- Alerting**: Set up alerting mechanisms to notify the operations team in case of any critical issues or abnormal behavior detected by the monitoring system.

These are some general guidelines for integrating DevOps practices and installing Bekur. However, the specific integration and installation procedures may vary depending on the infrastructure, technology stack, and specific requirements of Bekur. It is essential to adapt these practices to suit the project's needs and leverage the available DevOps tools and technologies effectively.

Chapter5:ConclusionandFuturework 5.1Conclusion 5.2.Futurework 5.3. Closing remarks