# 

# 

DOCKER PROJECT REPORT

**SUBMITTED BY**

| **Dharshan patil** | **1MS22CS405** |
| --- | --- |
| **Abdul Wahab**  **ABID MODI** | **1MS22cs401**  **1MS21CS021** |

As part of the course **Operating Systems- CS53**

SUPERVISED BY

Faculty

**Chandrika Prasad Ma’am**

ASSISTANT PROFESSOR DEPARTMENT OF CSE

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# M S RAMAIAH INSTITUTE OF TECHNOLOGY

**(Autonomous Institute, Affiliated to VTU) BANGALORE-560054**

Oct 2023 - Jan 2024

**TABLE OF CONTENTS**

| **SI. No.** | **TITLE** | **Page No** |
| --- | --- | --- |
| **1.** | Abstract | **1** |
| **2.** | Introduction | **2** |
| **3.** | Objectives | **4** |
| **4.** | Steps for uploading application on cloud | **6** |
| **5.** | Conclusion | **7** |

**ABSTRACT**

The Dockerized ATM Simulation System encapsulates a robust solution for efficiently managing ATM operations within a Docker container. This containerization approach offers several advantages, enhancing the deployment, scalability, and consistency of the system across diverse environments.

By leveraging Docker, the ATM Simulation System, along with its dependencies and runtime environment, is packaged into a single container. This streamlined deployment process reduces the likelihood of errors arising from environmental variations, ensuring a consistent experience across different setups. Developers can easily share the entire application stack, fostering collaboration and minimizing compatibility issues, much like in the Pharmacy Management System scenario.

The containerized deployment of the ATM Simulation System adds portability, enabling it to run seamlessly on various platforms. This versatility enhances the adaptability of the system to different infrastructure setups, making it easier to integrate into diverse banking environments.

The self-contained and isolated environment created by Docker ensures easy maintenance of the ATM Simulation System. This isolation promotes system stability and security, similar to the Pharmacy Management System context. It also facilitates rapid development cycles for future enhancements or updates. Developers can confidently make changes and updates to the system within the Docker container without worrying about unintended consequences in the broader environment.

In summary, the Dockerized approach for the ATM Simulation System simplifies deployment, fosters collaboration among developers, ensures consistency across different environments, enhances system versatility, and promotes system stability and security.

**INTRODUCTION**

The Dockerized ATM Simulation System project represents a strategic amalgamation of traditional banking principles with cutting-edge technological advancements. At its core, this initiative strategically employs Docker containers—an advanced containerization solution designed to optimize critical aspects such as portability, scalability, and operational efficiency within the ATM simulation landscape. The primary objective is to redefine the user interaction paradigm within the realm of simulated ATM operations.

Leveraging contemporary web technologies and containerization principles, the project focuses on encapsulating intricate business logic, frontend components, and essential dependencies within Docker containers. This encapsulation results in a self-contained and reproducible environment, ensuring seamless deployment across diverse infrastructures. This approach addresses compatibility concerns and enhances the overall user experience within the simulated ATM environment.

**Technology Stack:**

* The project employs a robust technology stack that includes PHP, HTML, CSS, JS, Visual Studio Code, Docker, and containers.php
* Html,css,js
* dockers

**Technology Integration:**

A critical aspect of this project involves seamlessly integrating frontend and backend technologies. The frontend, crafted with HTML, CSS, and JavaScript, emphasizes an intuitive and user-friendly interface for simulated ATM interactions. Simultaneously, the backend manages complex business logic and facilitates seamless communication through APIs.

Docker, serving as the foundational containerization platform, introduces ephemeral and portable units that encapsulate software and dependencies. This approach ensures consistent packaging of the ATM Simulation System across various environments, aligning seamlessly with the adoption of a microservices architecture. Docker Compose, an advanced orchestration tool, plays a crucial role in binding these containers together, simplifying the deployment process and contributing significantly to the creation of a robust and fault-tolerant simulated ATM system.

Beyond its encapsulation function, Docker introduces a strategic abstraction layer, effectively isolating the ATM simulation application from the intricacies of underlying infrastructure. This abstraction ensures consistent operation on any environment supporting Docker, mitigating environmental inconsistencies within the simulated banking environment.

**Impact and Future Implications:**

Docker expedites the development lifecycle through rapid and reproducible builds, fostering collaborative development practices and mitigating deployment complexities within the ATM Simulation System. Its impact extends beyond encapsulation, positioning itself as an enabler within an ecosystem where innovation and reliability converge seamlessly in the context of simulated banking operations.

As we navigate the complexities of container orchestration, real-time communication, and security protocols within simulated ATM operations, Docker's transformative influence becomes increasingly apparent. This marks a decisive stride in modernizing simulated banking practices, aligning with contemporary web development standards in the context of ATM simulation.

**OBJECTIVES**

The Dockerized ATM Simulation System project represents a strategic amalgamation of traditional banking principles with cutting-edge technological advancements. At its core, this initiative strategically employs Docker containers—an advanced containerization solution designed to optimize critical aspects such as portability, scalability, and operational efficiency within the ATM simulation landscape. The primary objective is to redefine the user interaction paradigm within the realm of simulated ATM operations.

Leveraging contemporary web technologies and containerization principles, the project focuses on encapsulating intricate business logic, frontend components, and essential dependencies within Docker containers. This encapsulation results in a self-contained and reproducible environment, ensuring seamless deployment across diverse infrastructures. This approach addresses compatibility concerns and enhances the overall user experience within the simulated ATM environment.

**Technology Stack:**

The project employs a robust technology stack that includes PHP, HTML, CSS, JS, Visual Studio Code, Docker, and containers.

PHP

HTML, CSS, JS

Docker

Web-based Containerization:

Objective: Containerize both the frontend and backend components of the Dockerized ATM Simulation System using Docker.

Rationale: Docker containers ensure a consistent and isolated environment for both the frontend (e.g., user interface components) and backend (e.g., transaction logic, APIs) components, simplifying deployment and maintenance of the ATM Simulation System.

**Cross-Browser Compatibility:**

Objective: Ensure cross-browser compatibility for the frontend of the ATM Simulation System.

Rationale: Cross-browser compatibility is crucial for providing a seamless user experience to bank staff and administrators accessing the system from different web browsers during simulated ATM interactions.

**Responsive Design:**

Objective: Implement a responsive design for the frontend of the ATM Simulation System to ensure optimal user experience across various devices and screen sizes.

Rationale: Responsive design enhances accessibility, making the simulated ATM functionalities accessible to users on both desktop and mobile devices within the simulated banking environment.

**Backend API Design:**

Objective: Design a robust and scalable backend API to handle ATM simulation functionalities, including transaction handling, account management, and user interactions.

Rationale: A well-designed backend API ensures smooth communication between the frontend and backend, supporting the core functionalities of the ATM Simulation System in the context of simulated banking operations.

**WebSocket Integration:**

Objective: Explore and implement WebSocket integration for real-time communication between the frontend and backend of the ATM Simulation System, enhancing communication during critical operations.

Rationale: WebSocket enables real-time updates, providing a more interactive and efficient experience for bank staff and administrators interacting with the ATM Simulation System during simulated transactions.

**Docker Compose Configuration:**

Objective: Create a Docker Compose configuration to orchestrate the deployment of both frontend and backend containers of the ATM Simulation System.

Rationale: Docker Compose simplifies the management of multi-container applications, ensuring that both frontend and backend components work seamlessly together in the simulated banking ecosystem.

**Logging and Error Handling:**

Objective: Implement comprehensive logging and error handling mechanisms in both the frontend and backend of the ATM Simulation System to facilitate debugging, monitoring, and ensuring system reliability.

Rationale: Logging and error handling are critical for identifying and resolving issues promptly, improving the overall reliability and performance of the ATM Simulation System during simulated banking operations.

**CONCLUSION**

The inception and evolution of the Dockerized ATM Simulation System project have ushered in a significant exploration of containerization, orchestration, and the incorporation of microservices architecture within the simulated banking domain. By leveraging the capabilities of Docker, the deployment process has been refined, ensuring a consistent and standardized runtime environment. Docker Compose has emerged as a key facilitator in adeptly managing the intricacies of multi-container applications, contributing to heightened adaptability and scalability within the ATM Simulation System. The discernible impact of this containerization approach is reflected in the simulated banking system's enhanced efficiency, portability, and overall robustness. Beyond the immediate benefits of containerization, the skills cultivated throughout this endeavor extend into the realm of scalable and flexible software development practices within the simulated banking landscape. This amalgamation of traditional banking principles with cutting-edge technology not only transforms the current state of simulated ATM operations but also foreshadows a future where innovation and resilience will play pivotal roles in shaping the landscape of banking systems.The Dockerized ATM Simulation System project serves as a testament to the potential of containerization and microservices in revolutionizing the simulated banking sector. The journey undertaken unveils a blueprint for the integration of modern technologies, offering a glimpse into a future where sophisticated software development practices will redefine the standards of efficiency and adaptability within the ATM simulation domain.

**REFERENCES**

1. **Docker Documentation**: Official documentation for Docker, covering installation, usage, and best practices.
2. **OpenMRS:** An open-source medical record system that can be customized for various healthcare scenarios, including pharmacy management.
3. **Clean Code:** A Handbook of Agile Software Craftsmanship by Robert C. Martin: This book provides insights into writing clean, maintainable code.
4. **icroservices** - Martin Fowler: Martin Fowler's articles on microservices offer a deep understanding of this architectural style.