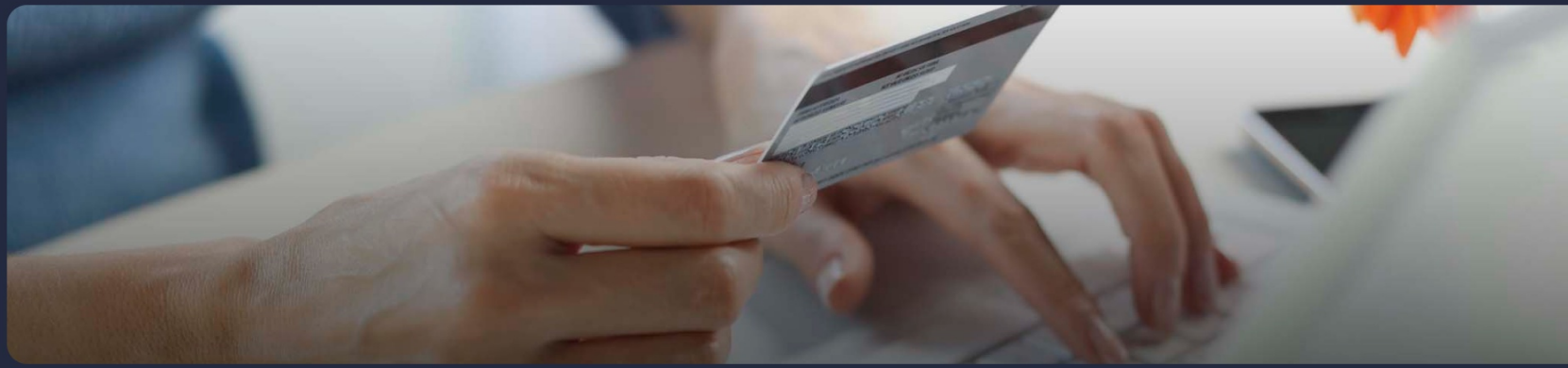


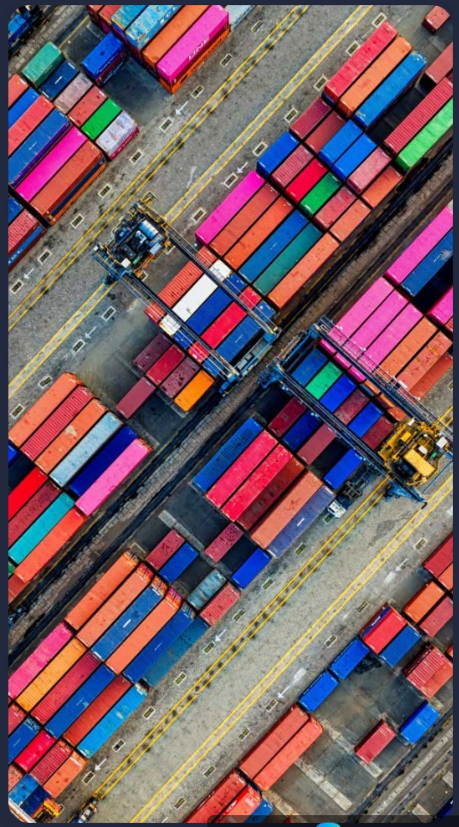
TechNova Online Store Project

This presentation showcases the TechNova Store project, an innovative e-commerce prototype developed by Karim Abouelfadl during my MSc Computer Science studies at City, University of London, leveraging modern technologies such as MySQL, Node.js, and Docker for a fully integrated web application.

Karim Abouelfadl

Comprehensive Overview of E-commerce Prototype Development

Exploring the goals, integrations, and deployment strategies for a robust e-commerce platform



1. **Objective: Create a robust e-commerce prototype**

   The primary goal is to build a functional e-commerce prototype that showcases essential features and user experience for potential investors and stakeholders.

2. **Full-stack integration of technologies**

   Integrating the entire stack including the database, API, and frontend components ensures seamless functionality and efficient data flow within the application.

3. **Utilizing Docker Compose for deployment**

   Docker Compose simplifies the deployment process by allowing the application and its dependencies to run in isolated environments, enhancing scalability and ease of management.

Core Features of Our Technology Stack

A comprehensive overview of the innovative technologies powering our platform

Robust MySQL database implementation

Utilizes a **MySQL** database with seed data to ensure reliable and efficient data management for our application.

Scalable Node.js/Express API

The application features a **Node.js** and **Express** based REST API, designed for high performance and scalability.

Efficient Nginx static file serving

Nginx acts as the frontend server delivering static files, optimizing performance and resource utilization for end users.

Dynamic product listing capabilities

Our platform includes a dynamic product listing feature, allowing for real-time updates and user interactions with product data.

Fully containerized deployment

The application is fully containerized, enhancing deployment efficiency and consistency across different environments.

Organizational Structure of Project Folders

A well-structured folder organization enhances project efficiency and clarity.

- **artefacts/ folder**
  This directory contains all necessary project documentation and deliverables that substantiate the development process and outcomes.
- **code/ directory**
  The coding files and scripts essential for the application, featuring organized source code to facilitate collaboration and version control.
- **database/ section**
  Contains database schemas, migration scripts, and relevant data files needed for the application to function correctly and efficiently.
- **media/ files**
  This folder holds all visual and audio assets such as images, videos, and graphics used in the project for marketing or presentation purposes.
- **docker/ configurations**
  Includes Docker files and configurations that allow for containerization of the application, ensuring a consistent environment across different systems.
- **clean organization**
  A clean and modular structure is essential for enhancing team collaboration, making it easier to find necessary files and promoting efficient workflow.

Comprehensive Database Design for Uber's Business Model

An overview of the essential tables and their relationships in the Uber database architecture

Users table stores customer data.

Contains essential information including user IDs, names, emails, and addresses to identify and communicate with Uber users.

Products table lists available services.

Details all the transportation options including ride types and pricing, enabling user selection based on preferences.

Orders table tracks ride requests.

Records each ride request made by users along with timestamps and statuses, essential for operational management.

OrderItems links orders to services.

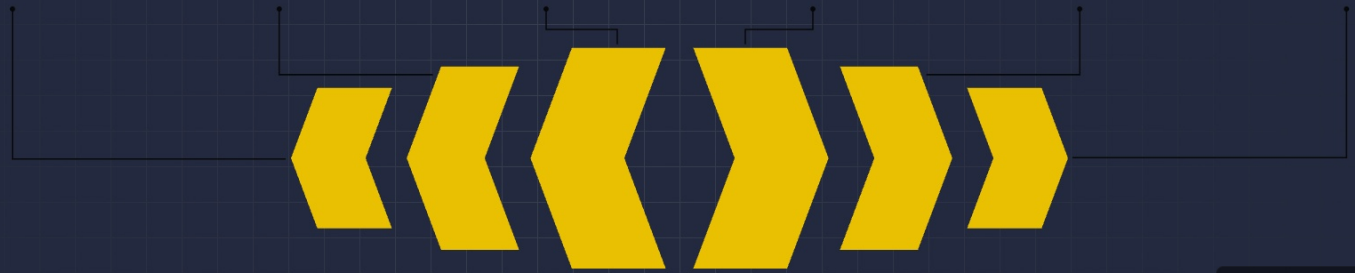
Associates specific rides or services to each order, facilitating detailed analysis of user preferences and service utilization.

Cart table handles user selections.

Allows users to add services to their cart before confirming their ride, enhancing user experience and efficiency.

CartItems tracks individual selections.

Stores details of each item selected in the cart, ensuring accurate processing during order confirmation.



Understanding Docker Compose Architecture for Application Deployment

A visual representation of services involving MySQL, Node/Express, and Nginx with readiness checks

Services Overview

This diagram illustrates the primary services utilized in the Docker Compose architecture, which are essential for application deployment.

Database Management with MySQL

MySQL serves as the database service within this architecture, managing data storage and retrieval for the application effectively.

Backend Implementation with Node.js

The backend is developed using Node.js with the Express framework, providing a robust environment for handling API requests.

Frontend Delivery via Nginx

Nginx functions as the web server for the frontend, efficiently handling HTTP requests and serving client-facing content.

Wait-for Scripts for Readiness

Scripts are implemented to ensure all services are fully operational before the application starts, preventing runtime errors.

Executing Docker Compose to Run Services

Understanding the Command to Build Images and Run Services in Docker

Command: docker-compose up --build

This command initiates the **building** of images and starts services.

Builds images efficiently

This process compiles all necessary application components into **images**.

Runs services seamlessly

After building, it **launches** the required services for the application.

Logs confirm connections

Logs provide real-time feedback on **service connections** and performance.

Live Demo

Identifying Limitations and Future Enhancements

Exploring areas of improvement and potential features for our project.

Single-page prototype

Current design lacks multiple pages.

No user authentication

Users cannot securely log in.

No cart checkout

Purchasing features are absent.

Future navigation improvements

Planning for multi-page navigation.

User sessions

Adding features for user sessions.

Admin features

Introducing administrative capabilities.

Conclusion of Fully Containerized E-Commerce Prototype

A comprehensive demonstration of integration and deployment in a modern e-commerce environment.

- **Fully containerized prototype**
  This prototype showcases a complete and integrated e-commerce solution.
- **Integration of DB**
  Demonstrates effective integration of database systems with other components.
- **API functionality**
  Highlights the seamless operation of the API within the e-commerce environment.
- **Frontend demonstration**
  Illustrates the user interface and experience of the e-commerce application.
- **DevOps deployment skills**
  Showcases skills in deploying applications using DevOps methodologies.
- **Comprehensive testing**
  Ensures all components are tested and function together cohesively.