NATIONAL UNIVERSITY
COLLEGE OF COMPUTING AND
INFORMATION TECHNOLOGIES

DATABASE SYSTEMS

U-BELT EXPRESS (DOCUMENTATION)

FINAL PROJECT

Prepared by:

Rae Paulos
Tristan Jay Sevilla
Jared Pilapil

Submitted to:

Prof. Jensen Santillan

Table of Content

1. Introduction

1.1 Project Overview

U-Belt Express is a food delivery application designed for users within Manila area, with a particular focus on supporting low-scale and local restaurants. The platform allows users to easily browse local restaurants, place food orders, and manage transactions, all through an intuitive and user-friendly interface. Inspired by established delivery platforms like Foodpanda, UBELT-ExPress aims to prioritize affordability, convenience, and ease of use for users around Manila.

U-Belt Express focuses on the practical implementation of relational database design using a **locally hosted MySQL database**. The project simulated a food delivery platform where structured data handling, real-time updates, and secure transactions are the core components of the system.

While the current setup uses a local database, the team is open to transitioning to the cloud in the future to allow remote access and better scalability. This would require additional research and planning, particularly in selecting the right cloud platform, configuring database hosting, and ensuring data security and availability.

1.2 Learning Objectives

By the end of the project, students will be able to apply the following skills to create scalable and efficient applications, ensuring they are well-equipped for future software development challenges:

#	SKILL	DESCRIPTION
1	Application of Relational Database Design Principles	Demonstrate an understanding of relational database architecture, including entity relationships, data integrity, and query structuring using MySQL.
2	Implementation of Database Normalization Techniques	Apply normalization rules (1NF, 2NF, 3NF) to optimize data storage, eliminate redundancy, and maintain consistency within the database schema.
3	Development of Object-Oriented Software Solutions	Utilize object-oriented programming paradigms in Java to construct modular, maintainable, and reusable components of the application.
4	Application of User-Centered	Employ UI/UX principles to design intuitive,

	Interface Design	accessible, and responsive interfaces tailored to the needs of student users.
5	Integration and Utilization of Web APIs	Explore the implementation of third-party APIs (e.g., Google Maps API) to enhance system functionality and enable dynamic, real-time interactions.
6	Data Cleaning and Preprocessing Techniques	Perform data cleaning and transformation to ensure accuracy, consistency, and usability of external data before database integration.
7	Practice of Version Control and Collaborative Development	Utilize Git and GitHub for effective source code management, version tracking, and team-based development workflows.
8	Engagement in Collaborative Practices	Participate in a structured development environment, emphasizing communication, task delegation, and cooperative problem-solving.

2. Background

2.1 Introduction to Advanced Database Systems

A Database Management System (DBMS) is software that enables efficient storage, retrieval, and manipulation of application data. Traditional relational DBMS (RDBMS) organize data into tables, but can struggle with horizontal scalability, geo-distribution, and unstructured data types (Gillis, n.d.).

Advanced database systems go beyond traditional setups by offering features like splitting data across servers for speed, keeping backup copies for safety, using flexible data structures that adapt when needed, running across multiple locations for reliability, and letting users adjust how precise the data needs to be. Moreover, an app like U-Belt Express, where multiple users are browsing menus and ordering food, ensures the app responds quickly, handles lots of users smoothly, and stays reliable even if something fails.

2.2 Chosen Advanced Database Model/Technology

U-Belt Express will adopt MySQL, one of the most widely used open-source relational database systems. Although MySQL is often thought of as a “traditional” RDBMS, it incorporates several advanced features that make it well suited to a modern, student-focused food-delivery platform:

1. ***Client–Server Architecture***
    - MySQL runs as a multi-threaded server that handles client connections over TCP/IP, allowing the application backend (Java) to open multiple concurrent sessions.
    - This separation ensures that database tuning and application logic remain decoupled, which simplifies maintenance and scaling.

3. System Design

3.1 System Architecture

U-Belt Express employs a **three-tier client-server architecture** designed to deliver a scalable food delivery platform for Manila-area users and local restaurants.

Architecture Overview:

Presentation Layer (Frontend)

- JavaFX-based desktop application with SceneBuilder for UI construction
- Figma-designed interface optimized for student users
- Handles user interactions, order placement, and restaurant browsing

Application Layer (Backend)

- Java core application managing business logic and API orchestration
- User authentication and session management
- Order processing and transaction handling
- Google Maps API integration for delivery ETA calculations

Data Layer (Database)

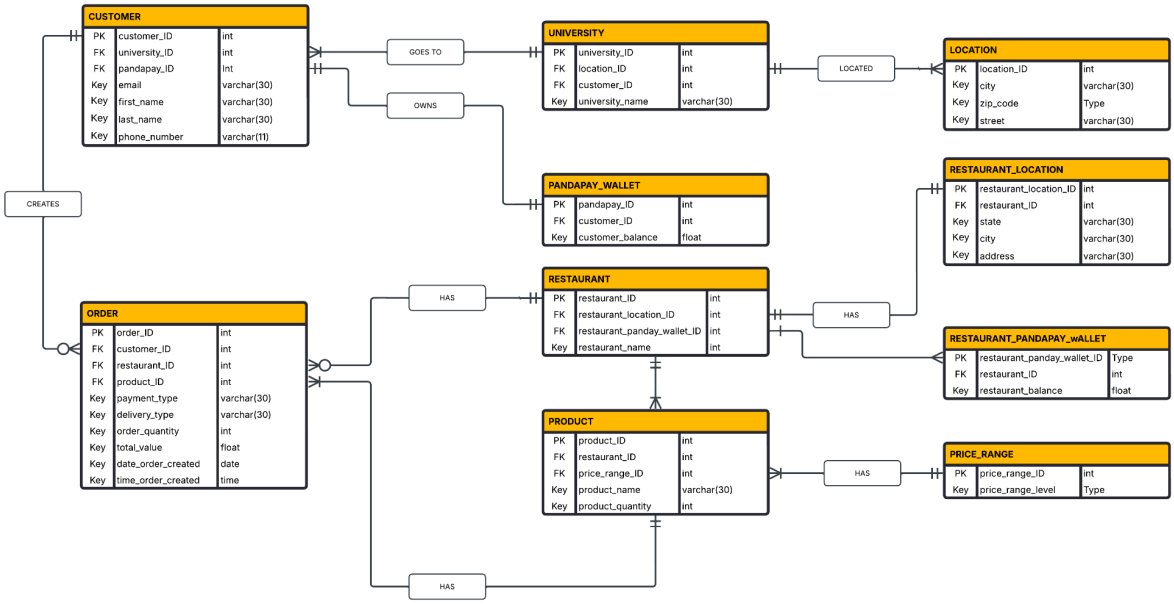
- MySQL relational database with workbench
- Foreign key constraints ensuring referential integrity
- Normalized schema supporting customers, restaurants, products, and orders

3.2 Design Diagrams

This section outlines the database structure of U-Belt Express. The system connects customers and local restaurants through a set of well-defined relational tables. Each table has a specific role in managing users, locations, payments, orders, and menu items. The Entity

Relationship Diagram (ERD) below illustrates the logical relationships between these entities and their attributes:

U-Belt Centric Food Delivery App: A Solution for Convenient and Affordable Dining



Data Dictionary for Element: CUSTOMER

Stores personal and contact details of users using the food delivery app.

Name	*Data Type*	*Constraint*	*Description*
customer_ID	varchar(100)	PK	Unique identifier for each customer
customer_location_ID	varchar(100)	FK	References the customer's location
customer_phone_number	varchar(11)	Key	Customer's phone number
customer_email	varchar(30)	Key	Customer's email address
customer_password	varchar(50)	Key	Customer's account password
customer_first_name	varchar(30)	Key	Customer's first name

customer_last_name	varchar(30)	Key	Customer's last name

Data Dictionary for Element: CUSTOMER_LOCATION

Stores location information for customers including city, zip code, and street address.

Name	*Data Type*	*Constraint*	*Description*
customer_location_ID	varchar(15)	PK	Unique identifier for customer location
city	varchar(30)	Key	City where the customer is located
zip_code	varchar(4)	Key	Postal/zip code of customer's location
street	varchar(30)	Key	Street address of the customer

Data Dictionary for Element: RESTAURANT_LOCATION

Stores location information for restaurants including state, city, and address details.

Name	*Data Type*	*Constraint*	*Description*
restaurant_location_ID	varchar(100)	PK	Unique identifier for restaurant location
state	varchar(30)	Key	State where the restaurant is located
city	varchar(50)	Key	City where the restaurant is located
address	varchar(30)	Key	Street address of the restaurant

Data Dictionary for Element: RESTAURANT

Stores restaurant information including basic details and location reference.

Name	*Data Type*	*Constraint*	*Description*
restaurant_ID	varchar(100)	PK	Unique identifier for each restaurant
restaurant_location_ID	varchar(15)	FK	References the restaurant's location
price_range	varchar(10)	FK	References the restaurant's price range category
restaurant_name	varchar(100)	Key	Name of the restaurant

Data Dictionary for Element: PRICE_RANGE

Stores price range categories to classify restaurants by their pricing levels.

Name	*Data Type*	*Constraint*	*Description*
price_range_ID	varchar(15)	PK	Unique identifier for price range category
price_range_level	Type	Key	Level or category of pricing (e.g., ~~₱~~, ~~₱₱~~ , ~~₱₱₱~~)

Data Dictionary for Element: BUSINESS_OWNER

Stores information about restaurant owners and their contact details.

Name	*Data Type*	*Constraint*	*Description*
business_owner_ID	int	PK	Unique identifier for each business owner
restaurant_ID	int	FK	References the restaurant owned
owner_first_name	varchar(50)	Key	Business owner's first name
owner_last_name	varchar(50)	Key	Business owner's last name
owner_email	varchar(100)	Key	Business owner's email address

owner_password	varchar(30)	Key	Business owner's account password

Data Dictionary for Element: PRODUCT

Stores information about food items and menu products offered by restaurants.

Name	*Data Type*	*Constraint*	*Description*
product_ID	int	PK	Unique identifier for each product
restaurant_ID	int	FK	References the restaurant offering the product
product_name	varchar(30)	Key	Name of the food product
product_description	varchar(100)	Key	Description of the food product
product_quantity	int	Key	Available quantity of the product
product_price	decimal(10, 2)	Key	Price of the product

Data Dictionary for Element: CART

Stores temporary shopping cart information for customers before order placement.

Name	*Data Type*	*Constraint*	*Description*
customer_ID	varchar(100)	PK/FK	References the customer who owns the cart
product_ID	varchar(100)	PK/FK	References the product in the cart
restaurant_ID	varchar(100)	FK	References the restaurant of the product
quantity	int	Key	Quantity of the product in cart

Data Dictionary for Element: ORDER

Stores completed order information including customer details and delivery information.

Name	Data Type	Constraint	Description
order_ID	varchar(100)	PK	Unique identifier for each order
customer_ID	varchar(100)	FK	References the customer who placed the order
restaurant_ID	varchar(100)	FK	References the restaurant fulfilling the order
delivery_type	varchar(30)	Key	Type of delivery (pickup, delivery, etc.)
order_amount	decimal(10, 2)	Key	Total amount of the order

4. Implementation

In the development of U-Belt Express, a food delivery application, the selection of appropriate technologies and a structured implementation process were critical to ensuring system reliability, efficiency, and user accessibility. This section outlines the rationale behind the chosen technology stack, including development tools, programming languages, and integration services. Furthermore, it details the step-by-step implementation process, highlighting technical challenges encountered and the corresponding solutions applied.

Overview

Table 1
Technology Stack and Description of Tools Used in U-Belt Express Development

TECH STACK	DESCRIPTION
Frontend	- Designed using Figma for high-fidelity prototyping. - Implement with JavaFX and SceneBuilder for UI development.
Backend	- Java used for core application logic (authentication, order processing, database interactions and other features). - API integration a. Google Maps API for delivery ETAs.

Database	-	MySQL was chosen for data management due to its relational model, ensuring data integrity and consistency.
Development Tools	-	Visual Studio Code (VSCode) as the integrated development environment (IDE).
	-	Git and GitHub used for version control and collaboration.

Frontend - UI/UX Design

The design process commenced with the creation of a high-fidelity prototype using **Figma**, **a collaborative interface design tool widely adopted in professional UI/UX workflows (Jain, 2024).** Figma enabled the development team to conceptualize and visualize user flows, layout structures, and interface elements in a collaborative environment. The resulting prototype provided a comprehensive reference for frontend implementation, facilitating alignment between design specifications and functional development.

The user interface is to be implemented using **JavaFX**, **a robust framework for developing rich desktop applications in Java.** To streamline the design-to-development process, the team intends to use **SceneBuilder**, **a graphical interface builder that allows for the visual construction of JavaFX components without manual coding.** This approach is expected to enhance consistency between the design prototype and the actual application, while also improving development efficiency and reducing UI-related errors. The planned integration of Figma, JavaFX, and SceneBuilder aims to ensure that the final user interface is both intuitive and aligned with modern design standards.

Backend - Programming Languages

The backend of U-Belt Express utilizes both Java and Google API, each selected for specific tasks to optimize development.

1. **Java - Core Application Development**
   Java is used for the core application logic, including user authentication, order processing, database interactions, and other planned features. Its robustness, scalability, and integration with JavaFX for frontend-backend communication made it ideal for constructing the main application.

2. **Google - API Integration**
   Python supports several specialized tasks in the backend, including integration with external APIs (Jeremiah & guide, 2024), leveraging its versatility and rich library ecosystem.

A. Google Maps API

The Google Maps API is integrated through Python to calculate the estimated time of arrival (ETA) for deliveries (*Distance Matrix API Overview | Distance Matrix API (Legacy)*, n.d.). By using location data from both the user and the restaurant, Python communicates with the API to provide accurate, real-time delivery time predictions.

Database - Data Management

For data storage and management, MySQL was selected as the database management system for U-Belt Express. The relational model provided by MySQL is particularly well-suited to handle structured data (Erickson, 2024), such as user information, restaurant details, menu items, and order transactions, which are inherently interrelated. The relational database's ability to define and enforce relationships between tables ensures data integrity and consistency, making it an optimal choice for applications that require a high level of organization and data accuracy.

Tools - Development Environment and Version Control

The development of U-Belt Express is planned to be conducted using Visual Studio Code (VSCode) as the primary integrated development environment (IDE). VSCode has been selected for its flexibility, support for multiple programming languages, and extensive range of extensions, which will enhance productivity during the development process (*Why Did We Build Visual Studio Code?*, n.d.). Its built-in Git integration is expected to facilitate version control, making it an ideal choice for managing the project's codebase and collaborative workflows.

For version control and team collaboration, Git and GitHub will be utilized. Git, as a distributed version control system, will allow the team to efficiently manage code changes, track revisions, and maintain code integrity (Torvalds & Jain, 2024). GitHub, as a cloud-based platform, will enable seamless collaboration, providing features such as branching and pull requests to manage code merges and resolve conflicts (Maheshwari, 2024). This version control setup will ensure that the development process remains organized and transparent while supporting collaborative contributions.

4.2 Implementation Process

We are currently facing challenges in implementing the Google API for our delivery operations, specifically in optimizing the mapping system to efficiently allocate deliveries based on real-time distance calculations. Our goal is to enhance the accuracy of distance-based allocation by integrating real-time data from Google Maps, ensuring that deliveries are assigned to the most suitable stores based on proximity to both the pick-up and drop-off locations.

One of the key aspects of this implementation is the ability to dynamically calculate distances between stores, pick-up points, and drop-off destinations. By leveraging Google API's

mapping and routing capabilities, we aim to streamline the delivery process, reduce transit times, and improve overall efficiency. However, we are encountering difficulties in configuring the API to provide precise real-time distance measurements and seamlessly integrate them into our allocation system.

5. Evaluation

5.1 Evaluation Criteria

To evaluate the U-Belt Express food delivery system, we will set key criteria focusing on technical performance and user experience. These will include:

- **Software Selection & Use (25%)**
  - Assess the appropriateness and effective utilization of chosen technologies.
- **Understanding of Data Models (25%)**
  - Evaluate the comprehension and implementation of relational database concepts.
- **SQL Implementation (25%)**
  - Assess the quality and efficiency of database operations and queries.
- **Error Handling & Optimization (25%)**
  - Evaluate system reliability and performance optimization strategies.

5.2 Testing Methodology

We will use several testing methods to check if the U-Belt Express system will work effectively. These will include:

- **Manual UI Testing** – We will manually test the interface designed in Figma and JavaFX to see if it will be clear and user-friendly.
- **Unit Testing** – We will test small parts of the system, like login and database functions, to make sure they will work correctly.
- **Database Validation** – We will run queries in MySQL to make sure the database structure will support proper relationships between users, orders, and restaurants.

5.3 Evaluation Results

Strengths

- **User Interface Clarity** – The UI is simple and easy for users to use, based on early feedback.
- **Organized Database** – The ERD supports multiple users and restaurants, helping the system scale properly.
- **Core Function Readiness** – Early tests suggest that basic features like login and order tracking will perform as planned.

Weaknesses

Incomplete Integration – Since the system is still in early stage and under development, some features like real-time updates and e-wallet transactions are not fully functional at first.

The current system design will allow for easier updates and smooth integration of new features in the future. We will ensure that the weaknesses are handled clearly and in a user-friendly manner to improve the overall experience. In preparation for future enhancements in using the API Integration and business-side local restaurant management, we will plan ahead for increased data volume and implement necessary security measures to maintain system performance and reliability.

6. Discussion

6.1 Software Solutions for Data Manipulation

The U-Belt Express project explored several software solutions for CRUD operations and data manipulation within the relational database model:

Java Database Connectivity (JDBC) was selected as the primary solution for database interactions. JDBC provides direct SQL execution capabilities, allowing for complex queries and transactions essential for food delivery operations. Its strengths include excellent performance for relational operations and seamless integration with the Java backend. However, JDBC requires extensive boilerplate code and manual SQL query construction, which can be error-prone and time-consuming.

MySQL Workbench served as the database administration tool for schema design and data manipulation during development. Its visual query builder and ERD generation capabilities streamlined database design, while its built-in data export/import features facilitated testing scenarios. The limitation lies in its desktop-only nature, restricting collaborative database management.

JavaFX integrated database controls were implemented for real-time data binding between the user interface and database. This solution provides immediate visual feedback for data changes and simplifies UI development. However, it can create tight coupling between presentation and data layers, potentially affecting system maintainability.

The chosen combination effectively supports the project's requirements for reliable order processing, user management, and restaurant data handling, though it requires careful error handling and connection management to ensure system stability.

6.2 Implementation of SQL Storage Systems

MySQL's workbench was utilized as the core SQL storage system for U-Belt Express. The implementation focused on maintaining referential integrity across the complex relationships between customers, local restaurants, orders, products, and etc.

Indexing Strategy: Primary and foreign key indexes were established on frequently queried columns (customer_ID, restaurant_ID, order_ID) to optimize performance. Composite indexes were created for location-based queries combining city and zip_code fields to support delivery radius calculations.

Challenges Encountered: The main challenge involved managing concurrent access during peak ordering times. Customers attempting to order from many restaurants simultaneously created potential race conditions for inventory management. This was addressed through row-level locking and optimistic concurrency control.

Schema Evolution: As the project developed, schema modifications required careful migration planning to maintain data integrity. The normalized design (achieving 3NF) occasionally resulted in complex JOIN operations for comprehensive order views, requiring query optimization and consideration of denormalization for performance-critical operations.

The MySQL implementation successfully supports the application's core functionality while providing a foundation for future scalability improvements.

7. Conclusion

7.1 Project Summary

The U-Belt Express project successfully demonstrates the application of advanced database concepts in developing a functional food delivery platform. Key accomplishments include the design and implementation of a fully normalized relational database supporting complex business operations, development of a comprehensive Entity Relationship Model with nine interconnected entities managing customers, restaurants, products, and orders, and integration of modern technologies including JavaFX for user interface, Google Maps API for delivery calculations, and MySQL for data management.

The project achieved its learning objectives by applying database normalization principles, implementing object-oriented design patterns, and demonstrating practical API integration. The resulting system provides essential food delivery functionality including user authentication, restaurant browsing, order management, and location-based services tailored for the Manila local markets.

7.2 Limitations and Future Improvements

Current limitations include the local database deployment restricting multi-user access and scalability, incomplete real-time features such as live order tracking and instant notifications, limited payment integration with only basic transaction recording, and absence of advanced features like recommendation algorithms and delivery optimization.

Future improvements should focus on cloud migration to enable remote access and improved scalability, implementation of real-time connections for live updates, integration of comprehensive payment gateways including e-wallets and credit cards, development of mobile applications for broader accessibility, and addition of analytics dashboards for business owners and administrators.

Performance enhancements could include database sharding for handling larger datasets, implementation of caching mechanisms for frequently accessed data, and optimization of complex queries through advanced indexing strategies.

Some features, such as real-time updates and e-wallet transactions, might not be completely operational at first because the system is still in the development stage.

7.3 Contribution to Advanced Database Systems

This project contributes to the field of advanced database systems by demonstrating practical implementation of relational database principles in a real-world application context. The work showcases effective integration of traditional RDBMS capabilities with modern application requirements, proving that well-designed relational systems remain relevant for complex business applications.

The project's approach to handling concurrent transactions, maintaining data integrity across multiple related entities, and integrating external APIs while preserving database consistency provides valuable insights for similar applications. The documented challenges and solutions offer practical guidance for developers implementing database-driven applications in competitive markets.

Furthermore, the project illustrates the importance of thoughtful database design in supporting business scalability, demonstrating how proper normalization and relationship modeling create a foundation for future enhancements and feature expansion in evolving application requirements.

8. References

Distance Matrix API overview | Distance Matrix API (Legacy). (n.d.). Google for Developers. Retrieved May 4, 2025, from https://developers.google.com/maps/documentation/distance-matrix/overview

Erickson, J. (2024, August 29). *MySQL: Understanding What It Is and How It's Used*. Oracle. Retrieved May 4, 2025, from https://www.oracle.com/ph/mysql/what-is-mysql/

Gillis, A. S. (n.d.). *What is a relational database? | Definition from TechTarget*. TechTarget. Retrieved May 6, 2025, from https://www.techtarget.com/searchdatamanagement/definition/relational-database

Jain, S. (2024, September 26). *What is Figma?* GeeksforGeeks. Retrieved May 4, 2025, from https://www.geeksforgeeks.org/what-is-figma/

Jeremiah, O., & guide, s. (2024, November 1). *Python APIs: A Guide to Building and Using APIs in Python*. DataCamp. Retrieved May 4, 2025, from https://www.datacamp.com/tutorial/python-api

Maheshwari, M. (2024, June 30). *Introduction to Github*. GeeksforGeeks. Retrieved May 4, 2025, from https://www.geeksforgeeks.org/introduction-to-github/

MySQL :: MySQL 8.4 Reference Manual :: 17.1 Introduction to InnoDB. (n.d.). MySQL :: Developer Zone. Retrieved May 6, 2025, from https://dev.mysql.com/doc/refman/8.4/en/innodb-introduction.html

Patel, A. (2025, March 7). *Introduction of DBMS (Database Management System)*. GeeksforGeeks. Retrieved May 6, 2025, from https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/

Torvalds, L., & Jain, S. (2024, February 14). *What Is a GIT Repository?* GeeksforGeeks. Retrieved May 4, 2025, from https://www.geeksforgeeks.org/what-is-a-git-repository/

Why did we build Visual Studio Code? (n.d.). Visual Studio Code. Retrieved May 4, 2025, from https://code.visualstudio.com/docs/editor/whyvscode

9. Appendix (Optional)

Include any additional information that may be relevant but not essential for the main body of the document.
This could involve detailed code listings, complex diagrams, or supplementary data used in the evaluation.