Where’s My Order

Software Design Document

CONTRIBUTORS:

TEAM GALAHAD

- *ABHIJITH M*
- *VIJAY ANANDAN JM*

INDEX

1. Introduction

1.1 Purpose

This project will develop a real-time bot to give customers instant order updates, improving transparency and reducing support needs.

1.2 Scope

This project aims to create a bot that provides customers with real-time order status updates, including shipping and delivery details, accessible through the company's website and app.

1.3 Overview

This project will implement a real-time bot to give customers instant access to their order status, including shipping and delivery updates, directly through the company's digital platforms.

1.4 Definition and Acronyms

Order Management System (OMS): Software used by the company to track, manage, and fulfil customer orders.

Bot: A software application designed to automate tasks, in this case, responding to customer inquiries about order status.

API: Application Programming Interface; allows the bot to communicate with the OMS for real-time data.

ETA: Estimated Time of Arrival; the expected delivery time of a customer's order.

UI: User Interface; the part of the bot that customers interact with, typically on the company's website or app.

RTOS: Real-Time Order Status; the function of providing immediate updates on order processing and shipping status.

2. Motivation & Problem Statement

Problem Statement: Real-Time Order Status bot

Create a real-time order status bot that allows customers to inquire about the status of their orders. The bot should access the company's order management system to provide accurate updates on shipping, delivery times, and tracking information, ensuring customers have easy access to information without needing to contact support**.**

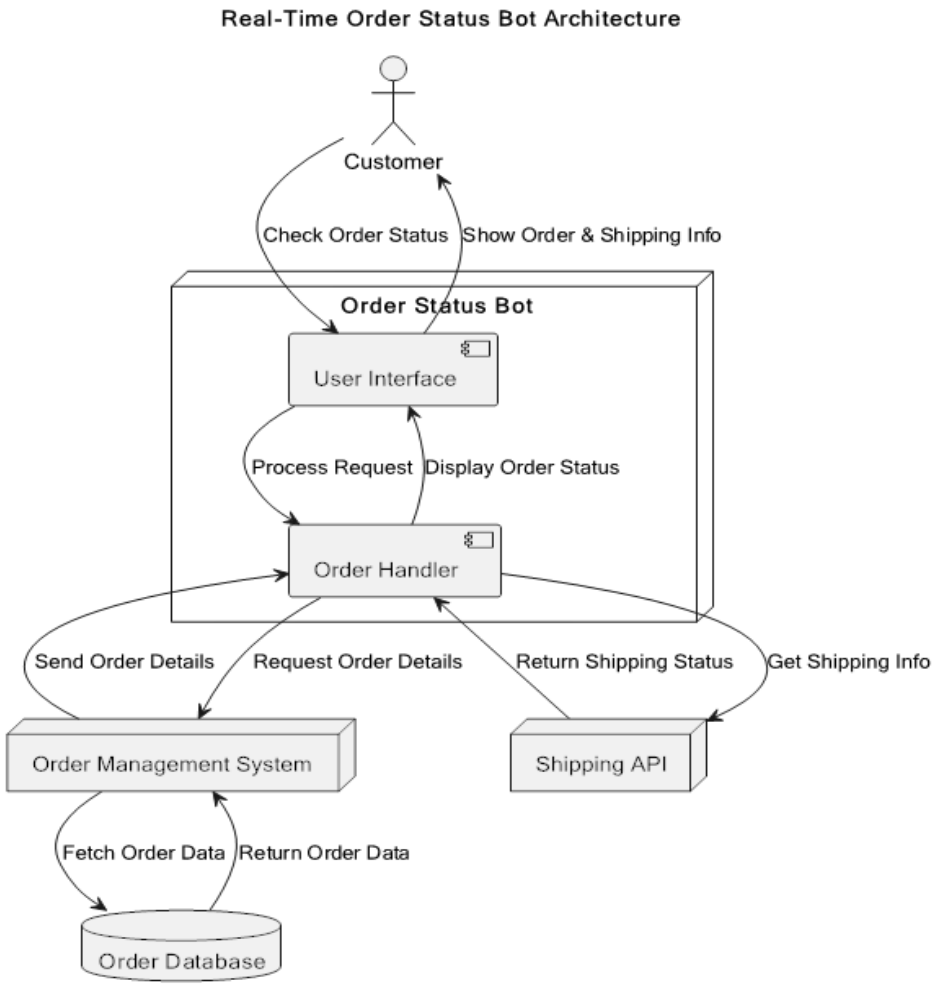
Motivation:

This project was inspired by a frustrating experience my grandparents faced while ordering medicine online when they were unwell. After placing the order, they waited for hours without updates and struggled to reach customer service, leaving them anxious and frustrated. Ultimately, they had to go out to buy the medicine themselves, adding unnecessary stress to an already challenging situation.

This highlighted the need for a reliable way to check order statuses in real-time. If they had access to a chatbot that could provide instant updates on shipping and delivery information, they could have easily tracked their order without the hassle of unresponsive support.

Developing this chatbot could benefit people of all ages—children, the elderly, and everyday customers—by providing easy access to vital information about their orders and reducing the stress associated with delays.

3. Architecture Diagram:



4. Tech stacks used

Node.js:

A JavaScript runtime that allows you to run JavaScript on the server side. It enables building scalable network applications.

Express.js:

A web framework for Node.js that simplifies the creation of web servers and APIs. It provides robust features for web and mobile applications.

JavaScript:

The primary programming language used for both server-side (Node.js) and client-side (browser) code.

HTML:

The standard markup language for creating web pages. In this case, it's used to create the front-end interface where users can input their order ID.

CSS:

CSS is used for styling the HTML elements to improve the user interface.

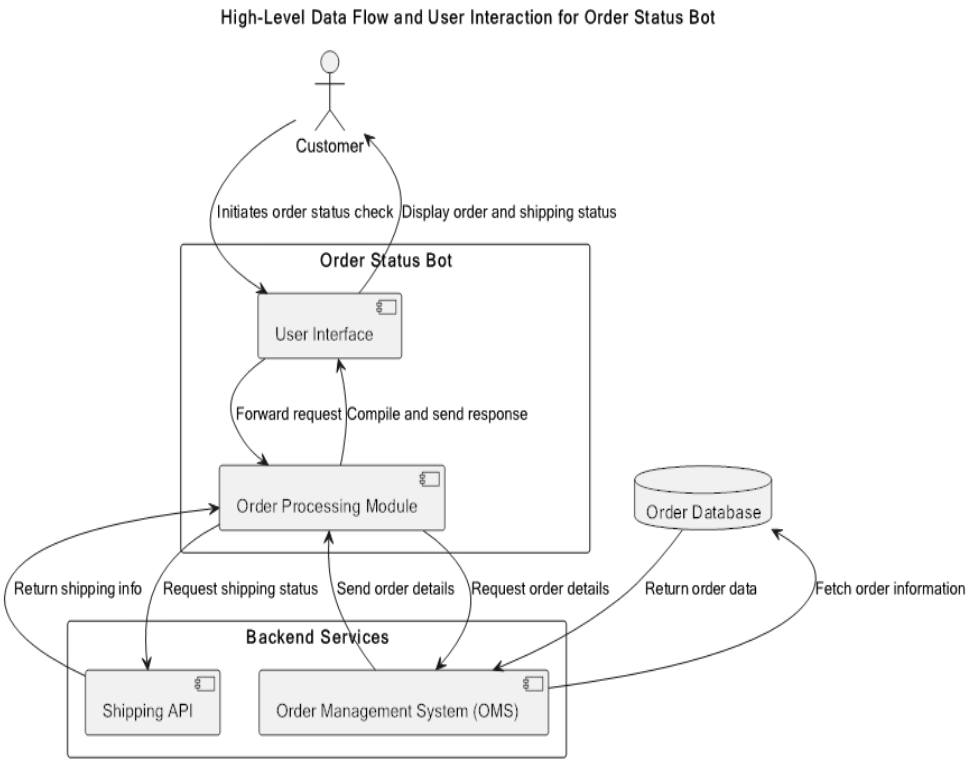
JSON (JavaScript Object Notation):

A lightweight data interchange format used for transmitting data between the server and the client. The API sends order information in JSON format.

Dialog Flow:

Dialog flow to build a real-time order status bot can enhance the user experience by allowing natural language processing for user interactions.

5. Flow Diagram:



6. Key Challenges:

Integration with Order Management System

Challenge: Integrating the bot with the order management system may pose difficulties, particularly if the API is not well-documented or user-friendly.

Solution: We will begin with mock data to develop and test the bot's functionalities. As we gain familiarity with the API, we will progressively integrate real order data.

Error Handling and Data Validation

Challenge: Users might input incorrect order IDs or encounter unexpected errors from the system.

Solution: We will implement robust error handling and input validation mechanisms. Clear and informative error messages will guide users in correcting their inputs.

Real-Time Updates and Scalability

Challenge: Maintaining responsiveness and performance during peak user activity can be challenging.

Solution: To optimize performance, we will employ caching strategies for frequently requested data and conduct load testing to ensure the bot can handle multiple simultaneous requests efficiently.

User Interface and Experience

Challenge: Designing an intuitive interface that is user-friendly and accessible to all.

Solution: We will gather feedback from target users throughout the design process and conduct usability testing to identify areas for improvement. Comprehensive documentation will also be provided to assist users.