A Java Full Stack Project Report on

FOOD-DELIVERY-APP

Submitted

In partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE and ENGINEERING

By

Devi Harini - 231FA04853

Sai geethika - 231FA04C67

Deepika - 231FA04866

Poojitha - 231FA04844

Under the Guidance of

R. AKHILESH
BYTEXL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF COMPUTING AND INFORMATICS

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University)
Vadlamudi, Guntur -522213, INDIA.
MAY, 2025

CERTIFICATE

This is to certify that the field project entitled “***FOOD-DELIVERY-APP”***

is being submitted by **[**Devi Harini**], [**231FA04853**]**, **[**Sai geethika**], [**231FA04C67**]**, **[**Deepika**], [**231FA04866**]**, **[**Poojitha**], [**231FA04844**]** in partial fulfilment of the requirements for the degree of **Bachelor of Technology (B.Tech.) in Computer Science and Engineering** at Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India.

This is a bonafide work carried out by the aforementioned students under my guidance and supervision.

Project Review Committee **byteXL SME** **HoD,CSE**

DECLARATION

Date:

We hereby declare that the work presented in the field project titled”***FOOD-DELIVERY-APP”*** is the result of our own efforts and investigations.

This project is being submitted under the supervision of **Mrs.G.Navya, Assistant professor,CSE** in partial fulfillment of the requirements for the Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering at Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, India.

Devi Harini	- 231FA04853	Signature:
Sai geethika	- 231FA04C67	Signature:
Deepika	- 231FA04866	Signature:
Poojitha	- 231FA04844	Signature:

ABSTRACT—This project presents a Food Delivery Application integrated with Artificial Intelligence (AI) to enhance user experience by providing personalized food recommendations. The app leverages machine learning algorithms to analyze user preferences, past orders, and real-time data such as location and time of day to suggest relevant meal options. By incorporating AI-powered recommendation systems, the app aims to streamline the decision-making process for users, increase customer satisfaction, and boost sales for partner restaurants. Additionally, the platform supports essential food delivery features including order tracking, secure payments, and user reviews, creating a seamless and efficient food ordering experience. This intelligent approach not only improves convenience for consumers but also offers valuable insights for restaurants to optimize their menus and marketing strategies.

Keywords—Food Delivery Application, AI-Powered Recom- mendations, Personalized Food Suggestions, Machine Learning, User Preference Analysis, ReactJS Frontend, FastAPI Backend, RESTful API, SQL Database, Order Management System, Real- Time Order Tracking, Secure Payment Integration, User Au- thentication, Recommendation Algorithm, Data-Driven Insights, Location-Based Services, Responsive UI/UX, Performance Opti- mization, Scalability, Cloud Deployment

I. INTRODUCTION

This Food Delivery Application is designed to provide users with a seamless, efficient, and personalized food ordering experience by leveraging modern web technologies and AI-powered recommendations. The app's frontend is built using React, enabling a dynamic and responsive user interface that delivers real-time updates and smooth navigation. The backend utilizes FastAPI, a high-performance Python framework that facilitates rapid API development and asynchronous processing for handling concurrent user requests efficiently. Data is securely stored and managed using MySQL, a relational database system known for its reliability and scalability. Together, these technologies support a robust platform capable of managing user authentication, order processing, payment integration, and AI-driven personalized food suggestions that enhance customer satisfaction and streamline restaurant operations.

FastAPI (Backend)

FastAPI serves as the backbone of the application's server- side logic. It is a modern Python framework that allows the creation of RESTful APIs with high throughput, thanks to its asynchronous request handling capabilities. FastAPI automates data validation, serialization, and API documentation (using OpenAPI standards), enabling fast and

maintainable development cycles. The backend manages key functionalities such as user authentication, order lifecycle management, secure payment processing, and communication with the AI recommendation engine. Its scalability and efficiency make it well-suited for handling the concurrent requests typical of a food delivery platform, ensuring low latency and high availability.

React (Frontend)

React powers the client-side of the application, providing an interactive and user-friendly interface. Using a component- based architecture, React enables the development of reusable UI elements, which simplifies maintenance and scalability. The framework's virtual DOM mechanism optimizes rendering performance, delivering a fluid experience on both desktop and mobile devices. React manages key frontend tasks including displaying restaurant menus, enabling order placement, tracking delivery status, and presenting AI-powered personalized food recommendations. State management techniques in React ensure that the application remains responsive and consistent, seamlessly interacting with the FastAPI backend through RESTful API calls.

MySQL (Database)

MySQL is utilized for persistent storage of all critical application data. This relational database management system efficiently organizes structured data such as user profiles, restaurant information, menu items, orders, payment transactions, and feedback. Its support for ACID transactions guarantees data integrity, especially important for handling orders and payments accurately. The database schema is optimized through normalization and indexing to facilitate fast query execution, which supports both backend processing and AI analytics. MySQL's robustness and scalability allow the platform to manage increasing data volumes and user activity as the application grows.

II. BACK GROUND STUDIES:

The rapid growth of online food delivery services has transformed how consumers access meals, providing convenience, variety, and speed. With the increasing number of restaurants and menu options available on digital platforms, users often face decision fatigue and struggle to find meals that match their preferences quickly. To address this, integrating Artificial Intelligence (AI) and machine learning techniques

into food delivery applications has become a growing trend to enhance user experience through personalized recommendations.

Evolution of Food Delivery Services:

The food delivery industry has witnessed rapid growth due to the convenience it offers consumers in accessing diverse cuisines without leaving their homes. Advances in mobile technology and internet penetration have made food delivery apps an essential part of daily life, facilitating easy ordering, quick payments, and real-time tracking.

Challenges in Food Selection:

With an abundance of restaurants and menu choices available on digital platforms, users often face difficulties in making quick, satisfactory food choices. Traditional apps usually depend on simple filters and user reviews, which do not fully capture individual preferences or contextual factors, leading to suboptimal user experience.

Role of AI in Personalization:

Artificial Intelligence and machine learning have emerged as powerful tools to address the challenge of personalization in food delivery apps. AI recommendation systems analyze historical user data, order patterns, time of day, location, and other contextual inputs to generate tailored food suggestions. This enhances user engagement, improves satisfaction, and boosts repeat orders, giving businesses a competitive edge.

Technology Stack for Modern Food Delivery Apps:

To build a scalable and responsive food delivery platform, integration of robust frontend, backend, and database technologies is essential. React is widely used for frontend development due to its component-based architecture and efficient rendering capabilities. FastAPI, a high-performance Python framework, is ideal for backend development, providing asynchronous processing and automatic API documentation. SQL databases like MySQL offer reliable, structured data storage with ACID compliance to ensure transaction integrity and efficient query processing.

Integration of AI with Core Technologies:

Combining AI-driven recommendation engines with this tech stack enables the creation of intelligent food delivery systems that dynamically adapt to user preferences in real-time. Such integration not only elevates the user experience but also provides restaurants with actionable insights to optimize menus and marketing strategies.

III. PROBLEM STATEMENT

Food Delivery App with AI-Powered Recommendations: Integrate AI to suggest restaurants and dishes based on user preferences and location.

Code:

chatbot.py

```
def suggest_dish(order_text):
    order_text = order_text.lower()
    if "pizza" in order_text:
        return "You might like Margherita Pizza!"
    elif "burger" in order_text:
        return "Try our Double Cheeseburger!"
    elif "chicken" in order_text:
        return "Grilled Chicken sounds perfect!"
    return "How about trying today's special combo meal?"
```

database.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, declarative_base

DATABASE_URL = "mysql+pymysql://root:sruthi%4013@localhost/foodapp"



engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

main.py

```
from fastapi import FastAPI, File, UploadFile, Form, Depends, HTTPException
from sqlalchemy.orm import Session
from whisper_handler import transcribe
from chatbot import suggest_dish
from database import get_db, Base, engine
from models import User, Restaurant, Order
from pydantic import BaseModel
import spacy
import os

# Load spaCy model
nlp = spacy.load("en_core_web_sm")

# Initialize DB tables
Base.metadata.create_all(bind=engine)

# FastAPI instance
app = FastAPI()

from fastapi.middleware.cors import CORSMiddleware

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # Or replace "*" with your frontend URL for security
    allow_credentials=True,
    allow_methods=["*"],  # Allows POST, GET, OPTIONS etc.
    allow_headers=["*"],
)


# Utility: Extract location from text using spaCy
def extract_location(text: str) -> str:
    doc = nlp(text)
    for ent in doc.ents:
        if ent.label_ == "GPE":  # Geo-Political Entity
            return ent.text
    return "Unknown"

# Root endpoint
@app.get("/")
def read_root():
    return {"message": "Welcome to the Food Delivery API!"}

# Prevent favicon error
@app.get("/favicon.ico")
async def favicon():
    return {}

# 📤 Upload and process voice command
@app.post("/upload-audio/")
async def handle_audio(
    user_id: int = Form(...),
    file: UploadFile = File(...),
    db: Session = Depends(get_db)
):
    path = f"temp_{file.filename}"
    with open(path, "wb") as f:
        f.write(await file.read())

    try:
        text = transcribe(path)
        suggestion = suggest_dish(text)
        location = extract_location(text)

        new_order = Order(user_id=user_id,
dish_name=suggestion, location=location)
        db.add(new_order)
        db.commit()

        return {
            "order_text": text,
            "suggestion": suggestion,
            "location": location
        }
    finally:
        if os.path.exists(path):
            os.remove(path)

# □ Register user (form data - not used by frontend)
@app.post("/register/")
def register_user(
    username: str = Form(...),
    password: str = Form(...),
    db: Session = Depends(get_db)
):
    existing_user =
db.query(User).filter(User.username ==
username).first()
    if existing_user:
        raise HTTPException(status_code=400,
detail="Username already exists")

    new_user = User(username=username,
password=password)
    db.add(new_user)
    db.commit()
    db.refresh(new_user)

    return {
        "message": "User registered",
        "user_id": new_user.id
    }

# ✅ Login or Register with JSON (used by
frontend)
class UserRequest(BaseModel):
    username: str
    password: str

@app.post("/register_or_login")
def register_or_login(user: UserRequest, db:
Session = Depends(get_db)):
    existing_user =
db.query(User).filter(User.username ==
user.username).first()

    if existing_user:
        if existing_user.password != user.password:
            raise HTTPException(status_code=400,
detail="Incorrect password")
        return {
            "message": "User logged in",
            "user_id": existing_user.id
        }

    # Register new user
    new_user = User(username=user.username,
password=user.password)
    db.add(new_user)
    db.commit()
    db.refresh(new_user)

    return {
        "message": "User registered",
        "user_id": new_user.id
    }

# 📦 Get order history by user ID
```

```
@app.get("/orders/{user_id}")
def get_orders(user_id: int, db: Session = Depends(get_db)):
    return db.query(Order).filter(Order.user_id == user_id).all()
```

models.py

```
from sqlalchemy import Column, Integer, String
from database import Base

class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True, index=True)
    username = Column(String(50), unique=True)
    password = Column(String(100))

class Restaurant(Base):
    __tablename__ = 'restaurants'
    id = Column(Integer, primary_key=True)
    name = Column(String(100))
    location = Column(String(255))
class Order(Base):
    __tablename__ = 'orders'
    id = Column(Integer, primary_key=True)
    user_id = Column(Integer)
    dish_name = Column(String(100))
    location = Column(String(255))
```

requirements.txt

```
fastapi
uvicorn
sqlalchemy
pymysql
openai-whisper
geopy
python-multipart
```

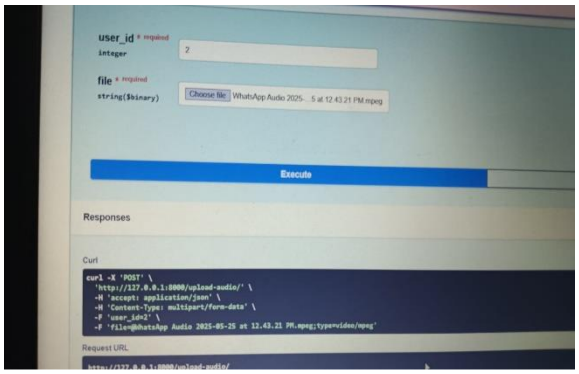
whisper_handler.py

```
import whisper

model = whisper.load_model("base")

def transcribe(audio_path):
    result = model.transcribe(audio_path)
    return result["text"]
```

O/P:



user_id * required
integer
2

file * required
string($binary)
Choose file WhatsApp Audio 2025-...5 at 12.43.21 PM.mpeg

Execute

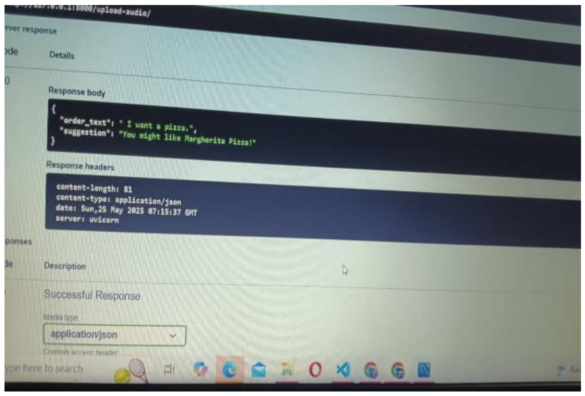
Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/upload-audio/' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'user_id=2' \
  -F 'file=@WhatsApp Audio 2025-05-25 at 12.43.21 PM.mpeg;type=video/mpeg'
```

Request URL

```
http://127.0.0.1:8000/upload-audio/
```



```
http://127.0.0.1:8000/upload-audio/
```

Server response

Code Details

Response body

```
{
  "order_text": " I want a pizza.",
  "suggestion": "You might like Margherita Pizza!"
}
```

Response headers

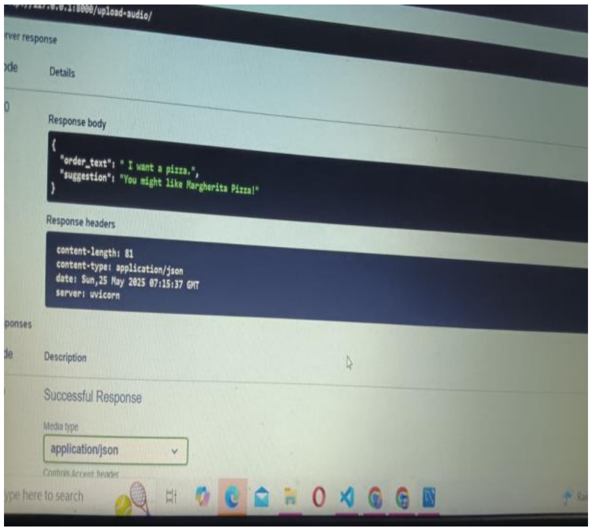
```
content-length: 81
content-type: application/json
date: Sun,25 May 2025 07:15:37 GMT
server: uvicorn
```

Responses

Code Description

Successful Response

Media type
application/json
Controls Accept header



```
.../127.0.0.1:8000/upload-audio/
```

Server response

Code Details

Response body

```
{
  "order_text": " I want a pizza.",
  "suggestion": "You might like Margherita Pizza!"
}
```

Response headers

```
content-length: 81
content-type: application/json
date: Sun,25 May 2025 07:15:37 GMT
server: uvicorn
```

Responses

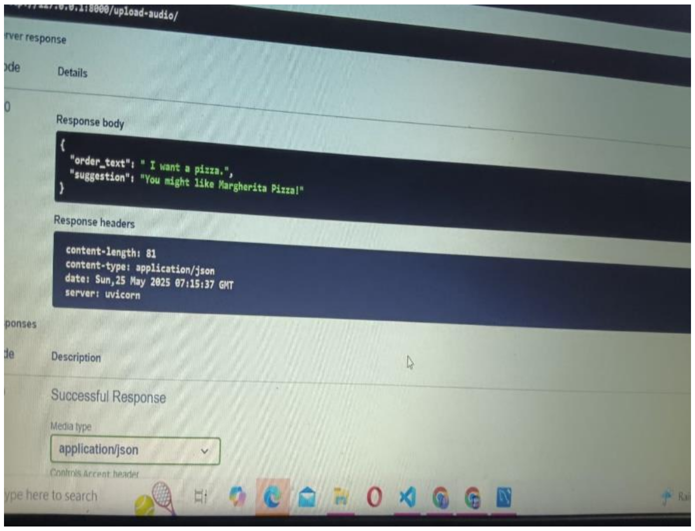
Code Description

Successful Response

Media type

application/json

Controls Accept header



```
.../127.0.0.1:8000/upload-audio/
```

Server response

Code Details

Response body

```
{
  "order_text": " I want a pizza.",
  "suggestion": "You might like Margherita Pizza!"
}
```

Response headers

```
content-length: 81
content-type: application/json
date: Sun,25 May 2025 07:15:37 GMT
server: uvicorn
```

Responses

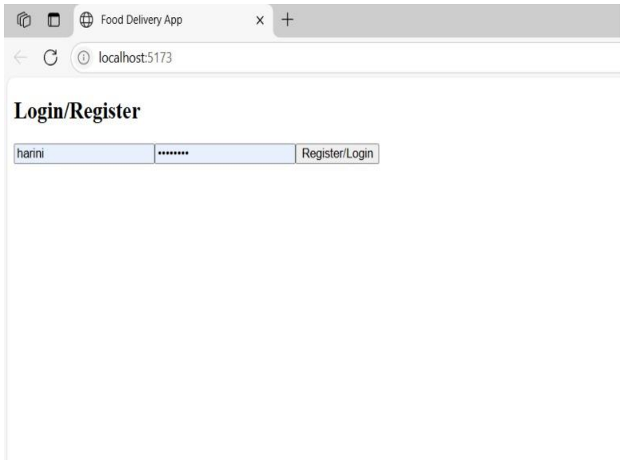
Code Description

Successful Response

Media type

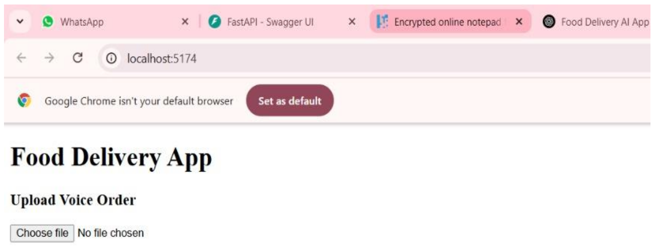
application/json

Controls Accept header



Login/Register

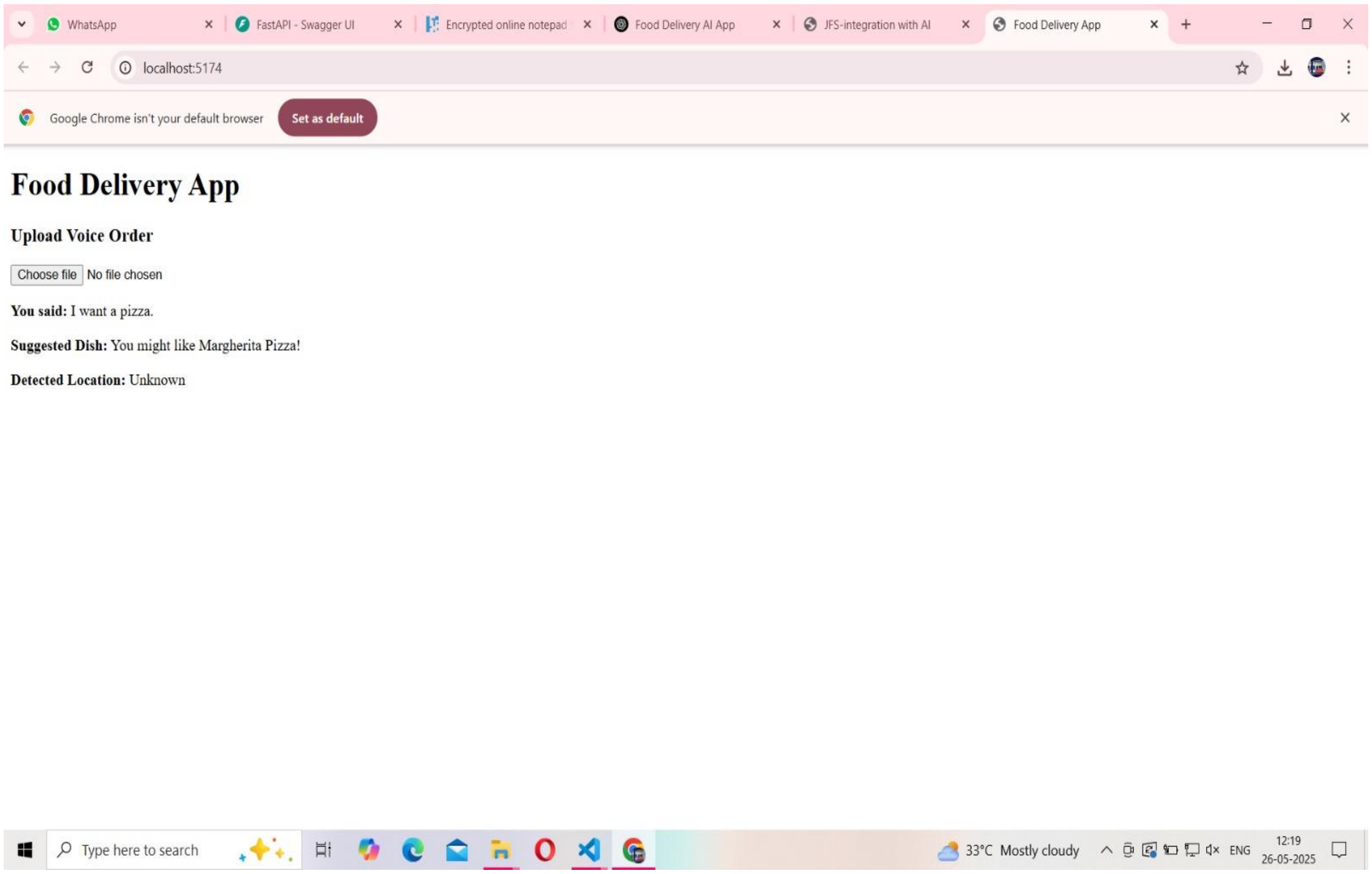
harini | •••••••• | Register/Login



Food Delivery App

Upload Voice Order

Choose file No file chosen



Food Delivery App

Upload Voice Order

Choose file No file chosen

You said: I want a pizza.

Suggested Dish: You might like Margherita Pizza!

Detected Location: Unknown