**Acknowledgement**

I would like to express my deep sense of gratitude towards all the faculty of Computer Science and Engineering and peers for their valuable advice at every stage.

I would also like to thank PKware, for giving me the opportunity to intern at their prestigious organization. During my internship, I had the kind association as well as supervision of **Mr. Rajinder Singh**. His exemplary guidance contributed immensely towards achieving an in-depthknowledge and understanding of the project that I was working on. I would also like to express my gratitude towards **Mr. Basant Gurung**, for always guiding me at every step of the way and all the individuals that worked along with me at Pkware; their patience and openness created an enjoyable working environment for me. I acknowledge the help of all of these individuals with an immense sense of gratitude.

Any attempt at any level could not be satisfactorily completed without the support and guidance of my parents and friends.

I am indebted to ‘University Institute of Engineering & Technology, Panjab University, Chandigarh’ for giving me an opportunity to pursue Industrial Training at Dataguise - A Pkware Company. This training has enabled me to empower myself with the knowledge of the existing trends in the world of technology.

**Table of Content**

|  |  |
| --- | --- |
| **Name** | **Page Number** |
| List of Figures and Tables | 3-4 |
| Introduction of project | 5-8 |
| Software requirement specification | 9 |
| Project Practicality | 10 |
| Software design | 11-13 |
| Coding specifications | 14-28 |
| Project Demo | 29-33 |
| Conclusion | 34 |
| Bibliography | 35 |

**List of Figures and Tables**

|  |  |
| --- | --- |
| **Figures** | **Page Number** |
| Fig 1: Pkware Product | 5 |
| Fig 2: Pro Foods | 7 |
| Fig 3: ER Diagram | 10 |
| Fig 4: Sample Database | 11 |
| Fig 5: File structure in Angular project | 12 |
| Fig 6: Event handling example | 17 |
| Fig 7: JDBC Structure | 18 |
| Fig 8: JDBC connectivity | 18 |
| Fig 9: Spring boot architecture | 19 |
| Fig 10: Landing Page | 23 |
| Fig 11: Registration Form | 24 |
| Fig 12: User view | 26 |
| Fig 13: Cart | 27 |
| Fig 14: Admin side manage inventory | 29 |
| Fig 15: Manage Orders | 29 |
| Fig 16: Support Metrics | 30 |
| Fig 17: Pie Chart | 30 |
| Fig 18: Backend project structure in eclipse | 31 |
| Fig 19: DB in PgAdmin | 31 |

**Introduction of project**

**Organization’s Information:**

PKWARE Inc. is an enterprise data protection software company that provides discovery, classification, masking, and encryption solutions, along with data compression software, used by thousands of organizations in financial services, manufacturing, military, healthcare and government. These solutions are used by enterprises that need to comply with data protection regulations such as GDPR, CCPA, [HIPAA,](https://en.wikipedia.org/wiki/HIPAA) PCI DSS, TISAX, [ITAR,](https://en.wikipedia.org/wiki/ITAR) [CDPA,](https://en.wikipedia.org/wiki/CDPA) [LGPD](https://en.wikipedia.org/wiki/LGPD) and other emerging laws. The company is headquartered in Milwaukee, Wisconsin with additional offices in the US, UK, and India.

Rebranded in 2021 after the acquisition of Dataguise, the PK Protect suite encompasses all former PKWARE and Dataguise products and offers solutions for managing personal data and other sensitive data through data discovery, classification, masking, reduction and encryption.

**Methodologies:**

PKWARE provides more sophisticated functionality, monitoring, and consistency across systems in less time when compared to in-house development options.

With its automated redaction solution in place, the corporation is able to take thousands of devices and millions of files out of scope for its PCI audits, simplifying its compliance efforts and maintaining the security of its customers’ data.

The PK Protect Suite—powered by PKWARE and Dataguise technology—offers solutions for managing personal information and other sensitive data, combining extensive expertise, data-driven processes, and proven technologies to balance data usage with data protection for minimal risk and maximum value. Make sensitive data discovery and protection easier with a solution that’s intuitive to implement and use, and saves you time, resources, compliance costs, and complication. With the PK Protect Suite, businesses are empowered to work safely with sensitive data to make better business decisions, gain greater customer insights, move to the cloud, and generate new revenue streams.

**A screenshot of a computer

Description automatically generated with medium confidence**Fig 1: Pkware Product

**Industrial Training Details -**

**Training Period -**

The first 3 months of the internship is the training of the technologies used in products of Pkware. This includes: -

1. Java programming language (Core and Advance)
2. Java Database Connectivity (JDBC) and Hibernate
3. Spring Boot Framework
4. Introduction to Gradle, Tomcat server
5. Introduction to HTML, CSS, Bootstrap
6. JavaScript and typescript programming language
7. Angular framework

In the realm of Java, I deepened my understanding of the language's fundamentals, such as object-oriented programming principles, data types, control structures, and exception handling. I gained hands-on experience in writing efficient Java code to solve real-world problems.

JDBC (Java Database Connectivity) was another significant aspect of my learning. I acquired knowledge about establishing connections to databases, executing queries, and managing result sets using JDBC. This allowed me to interact with various databases and perform CRUD (Create, Read, Update, Delete) operations effectively.

Hibernate, an Object-Relational Mapping (ORM) framework, enabled me to bridge the gap between Java objects and relational databases. I gained proficiency in mapping Java classes to database tables, performing database operations using Hibernate's session management, and optimizing database access for enhanced performance.

Spring Boot played a significant role in my internship as it provided a powerful framework for building APIs. I learned how to leverage Spring Boot's features to create RESTful APIs efficiently. This allowed me to design and implement scalable and modular web services.

In addition to the back-end technologies, I also delved into front-end development. I learned HTML, CSS, and Bootstrap, which are fundamental technologies for creating web pages with attractive layouts and styling. These skills helped me design visually appealing user interfaces.

Moreover, I explored Angular, a powerful front-end framework, which enabled me to build dynamic and responsive web applications. I learned about Angular components, templates, data binding, and routing. This knowledge allowed me to create interactive user interfaces and interact with back-end services.

Throughout my internship, I gained practical experience by working on projects that involved the integration of these technologies. This hands-on approach enhanced my programming skills, problem-solving abilities, and familiarity with industry-standard tools and frameworks.

**Pro Foods (Project Undertaken)**

This industrial training report provides an overview of the development and implementation of Pro Foods, an online food delivery app. Pro Foods is designed to meet the increasing demand for convenient and efficient food ordering and delivery services for individuals with specific requirements. This report aims to detail the objectives, methodologies, and outcomes of the industrial training experience focused on the development of Pro Foods.

In today's fast-paced world, people are seeking hassle-free solutions to meet their daily needs, including food consumption. The emergence of technology has revolutionized the food industry, paving the way for online food delivery services. Pro Foods aims to capitalize on this trend by offering a user-friendly and intuitive platform that connects customers with a wide range of restaurants and facilitates seamless food delivery.

The primary objective of developing Pro Foods is to provide a comprehensive, reliable, and customizable food delivery solution for owners. Through this app, users can browse menus, place orders, check old orders, whereas owners can watch out for metrics, orders and manage inventory. Pro Foods aims to enhance the overall dining experience by ensuring convenience, choice, and timely delivery of high-quality food.

During the industrial training period, the focus was on gaining practical experience in developing a robust and user-centric online food delivery app. The training involved a combination of theoretical knowledge and hands-on implementation, incorporating industry best practices, modern technologies, and agile development methodologies. The training provided an opportunity to understand the software development life cycle (SDLC) and gain proficiency in various stages, including requirement analysis, system design, implementation, testing, and deployment.

This report will outline the methodology used in developing Pro Foods, detailing the requirements, functionality, and architecture of the app. It will also highlight the challenges faced during the training period and the strategies employed to overcome them. The industrial training experience has not only enriched technical skills but has also fostered teamwork, problem-solving abilities, and effective communication within a professional work environment.

Ultimately, the Pro Foods app demonstrates the application of theoretical knowledge in a practical setting, providing valuable insights into the complexities and intricacies of software development for online food delivery services. The industrial training period has been a transformative experience, laying the foundation for a successful career in the field of software development and equipping me with the necessary skills and expertise to contribute to the ever-evolving technology landscape.

This report aims to provide a comprehensive overview of the industrial training experience, serving as a testament to the successful implementation of Pro Foods and the lessons learned throughout the process



Fig 2: Pro Foods Logo

**Software requirement specification**

**Eclipse:**

Eclipse is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. It is the second-most-popular IDE for Java development, and, until 2016, was the most popular. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins

**PgAdmin:**

PgAdmin is a user-friendly software tool designed for managing and administering PostgreSQL databases. It provides a simple and intuitive interface that allows users to perform various tasks, such as creating and managing database objects, executing queries, and monitoring database performance. With pgAdmin, users can efficiently interact with their PostgreSQL databases, making it an essential tool for database administrators and developers.

**Visual Studio Code:**

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for [Windows,](https://en.wikipedia.org/wiki/Windows) Linux and macOS. Features include support for debugging, syntax highlighting, [intelligent code completion,](https://en.wikipedia.org/wiki/Intelligent_code_completion) [snippets,](https://en.wikipedia.org/wiki/Snippet_(programming)) code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

**Git and Github:**

Git and GitHub are powerful software tools used for version control and collaboration in software development projects. Git is a distributed version control system that allows developers to track changes, manage different versions of files, and collaborate with others efficiently. It provides features like branching and merging, which enable teams to work on different tasks simultaneously and merge their changes seamlessly.

GitHub, on the other hand, is a web-based hosting service for Git repositories. It provides a platform for developers to store their Git repositories remotely and collaborate with others. GitHub offers additional features such as issue tracking, pull requests, and code reviews, facilitating smooth collaboration and project management.

**Project Practicality**

The practicability of an online food delivery app is a crucial aspect to consider when evaluating its feasibility and potential success in the market. Here are some key factors to assess:

Market Demand: Conduct a thorough analysis of the market to determine the demand for online food delivery services. Consider factors such as the size of the target market, customer preferences, existing competition, and any emerging trends. This assessment will help validate the need for an online food delivery app and ensure its relevance in meeting customer demands.

Business Model: Evaluate the viability of the business model associated with the app. Assess revenue streams, such as commission fees from partner restaurants or delivery charges, to determine if the app can generate sustainable income. Consider the cost structure, including development and maintenance expenses, marketing efforts, and operational costs, to ensure profitability and financial sustainability.

User Acquisition and Retention: Analyze strategies for acquiring and retaining app users. Determine how to attract a substantial user base, such as implementing effective marketing campaigns, referral programs, or partnerships with restaurants and influencers. Assess the potential for user retention through features like personalized recommendations, loyalty programs, and excellent customer service.

Technology Infrastructure: Evaluate the technical requirements and infrastructure necessary to support the app's functionality. Consider factors such as server capacity, scalability, security measures, and integration capabilities with third-party services, such as payment gateways or mapping APIs. Ensure the availability of robust technology solutions that can handle user traffic and ensure a seamless user experience.

Operational Efficiency: Assess the operational aspects of running the online food delivery app. Consider factors like order management, delivery logistics, and customer support. Evaluate the feasibility of establishing effective processes and systems to manage orders, optimize delivery routes, and provide timely and efficient customer service.

Legal and Regulatory Compliance: Ensure compliance with relevant laws and regulations in the food and delivery industry. Consider aspects such as food safety standards, data privacy regulations, payment processing compliance, and any local licensing requirements. Adhering to legal and regulatory frameworks will help build trust among users and mitigate potential risks.

By evaluating the practicability of your online food delivery app based on these factors, you can assess its potential for success in the market. This assessment will help identify strengths, weaknesses, and areas for improvement, enabling you to make informed decisions and optimize the app's overall performance and profitability.

**Software design**

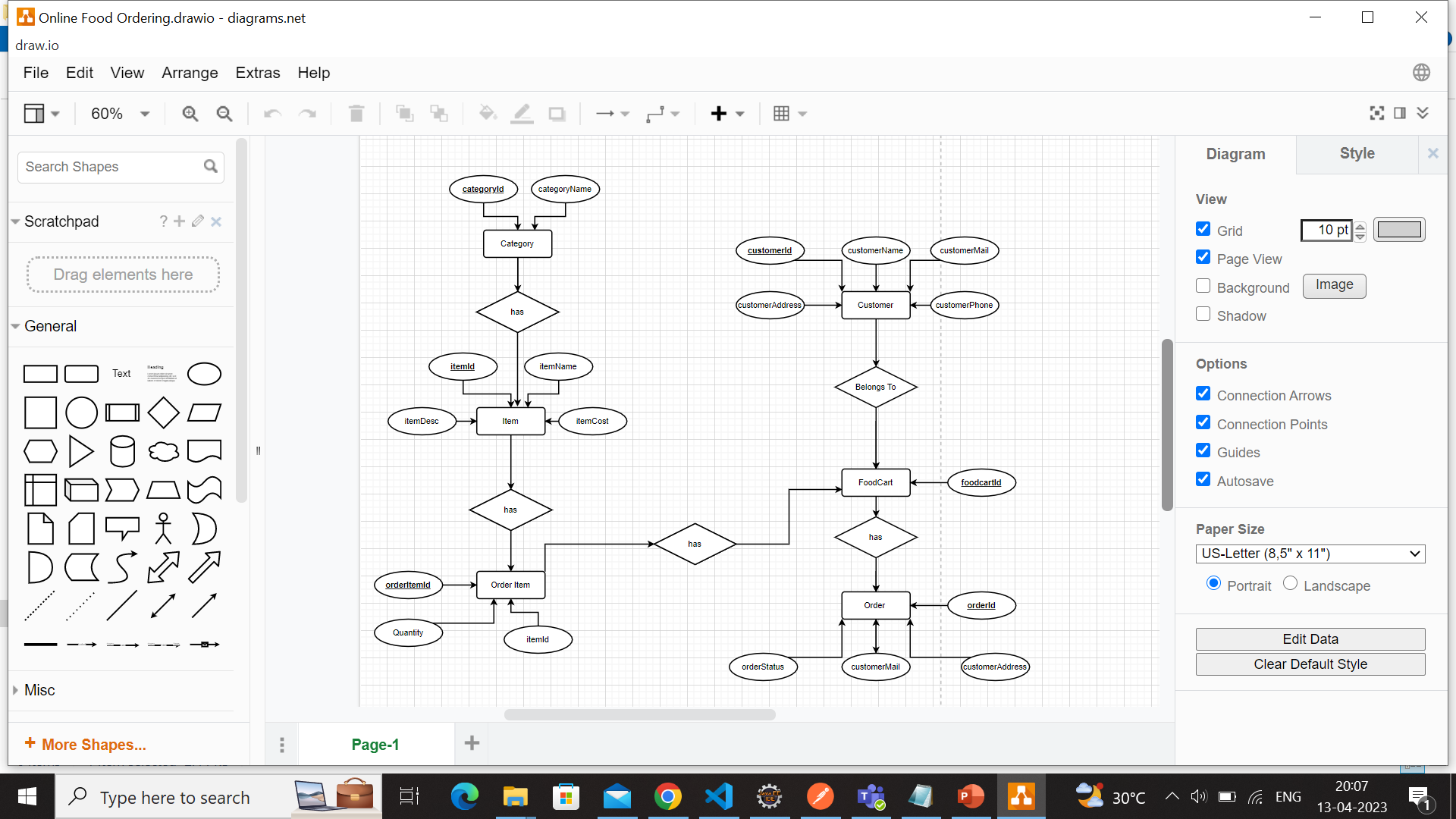
****

Fig 3: ER Diagram

**A screenshot of a computer

Description automatically generated**

Fig 4: Sample database snapshot

**Coding specifications**

**CHAPTER 1: FRONT END**

**1.1 Angular**

Introduction

Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps. It is a TypeScript-based free and open-source [web application framework](https://en.wikipedia.org/wiki/Web_framework) led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

It is a development platform, built on [TypeScript.](https://www.typescriptlang.org/) As a platform, Angular includes:

* A component-based framework for building scalable web applications.
* A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more.
* A suite of developer tools to help you develop, build, test, and update your code.

**1.2 File Structure**

When we create an Angular project, our app architecture looks like Figure 1 1.2.1 e2e

End-to-end (E2E) testing or additionally referred to as integration testing may be a good way to make sure that our application’s area unit operates properly including the dependencies and integrated pieces.

1.2.2 node\_module

In this folder, you can find all the third-party libraries that the application may rely on. This folder is purely for development.

1.2.3 src

The src folder contains the actual source code for the developers.

1.2.4 app folder

The app folder contains your projects logic and information. There is a minimum of one “module” and one “component” in every application and included within the app folder.

The files within contain the following: app.component.css - this can be the css vogue for Root AppComponent. designs here apply to app.component.html. app.component.html - Is associate hypertext mark-up language example related to a root part.

1.2.5 App Component

app.component.specs.ts – Use to test the unity of the root component.

app.component.ts – This is a TypeScript file that defines the basic logic of

AppComponent. AppComponent Linked Display is what becomes the root of the hierarchical rendering when you add components and services to your app.

app.module.ts – This file specifies the basis module of the appliance. The base module is named AppModule. This tells Angular a way to organise your order. Any new parts you boost your app ought to be per the app.module.ts file.

app.routing.module.ts – This file contains the definitions for switching between parts within the Angular app.

1.2.6 assets folder

This folder contains images and other resource files to copy as such when building your application.

1.2.7 environment folder

This is where we save configuration settings to varied environments. Basically, this folder contains 2 files,

I. environment.prod.ts file for the production environment.

II. environment.ts file for the development environment.

1.2.8 favicon.ico

favicon.ico is an icon file that displays on the browser.

1.2.9 index.html

This is the main HTML file, where the application starts, and here we have a defined in the text that matches the appComponent specifier, and it shows appComponent in the body section. If appComponent is different from that defined in the Body Text section, an error will appear.

1.2.10 main.ts

The file is a Typescript file. It's the place to begin our application. Here we can bootstrap (the initialization or loading process) our primary module using the bootstrapModule method like, bootstraps or begin AppModule that may be a root module for all parts in Angular apps.

1.2.11 App.Module.ts

As we know, whenever we create a component with CLI commands like,

Then a new component will be created with the specified name while creating the component

 using the command CLI, and as we can see in the image above we have three components, which are the root component;

Within the supplier array, we are able to see the boot that triggers or starts the part declared within the boot matrix. So as to figure with the Angular app in Index.html, we tend to use a constant name because the selector that was proclaimed within the bootstrap array in App.Module.ts. If Index.html contains an intellectual for AppComponent and for App.Module.ts it throws a bootstrap matrix bootstrap to Test Component so the appliance throws a slip.

1.2.12 polyfills.ts

It provides polyfill scripts (using Polyfill, we may implement functionality that is not supported by the browser) for browser support.

1.2.13 style.css

style.css gives us the ability to add global styles to our applications.

1.2.14 test.ts

The test.ts file prepares the test environment in the application.

1.2.15 angular-cli.json

The standard configuration file for your application is angular-cli.json.

1.2.16 Browserlist

It is configured to share target browsers and Node.js versions among various front-end tools.

1.2.17 editorconfig

This is used to define coding styles and text-editor plugins by which editors can read the file format and stay adhered to the defined style. If you're employed during a team atmosphere, take care that every developer in a team uses a constant setting during this file.

1.2.18 gitignore

gitignore file is used to export files or directories to/from your git repository.

1.2.19 package.json

Package.json file is a standard file, and every Angular node and project includes this file. Basically, this file contains all the information like project name, versions, dependencies, and dev dependency settings.

1.2.20 README.md

Introductory documents for the application.

1.2.21 tsconfig.json

The tsconfig.json file specifies Angular and Typescript base compiler options for the inherited projects. It contains a bunch of settings for your matter translator, therefore, the matter compiler appears at the setup and supports these settings to compile your code in JavaScript.

1.2.22 tsconfig.spec.json

JSON that inherits the tsconfig.json file and contains an inventory of files enclosed (mainly includes profiles) throughout testing, compiler possibility, etc.

1.2.23 tslint.json

It is a method to check for grammatical errors in the written code.

1.2.23 index.html

As we know, Angular is employed for developing single-page apps. Index.html is the beginning page once we launch Angular with the weight unit serve command.

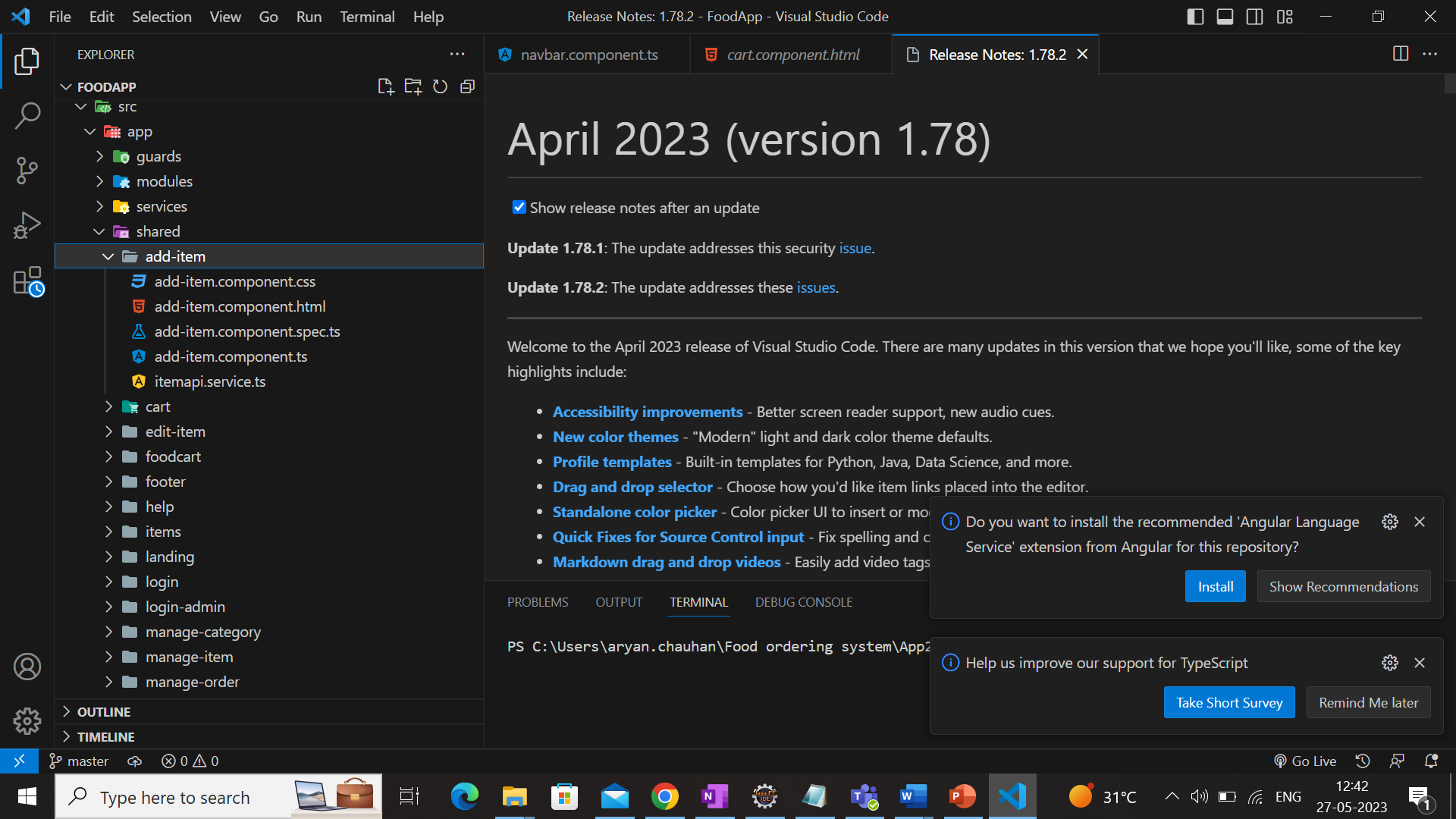
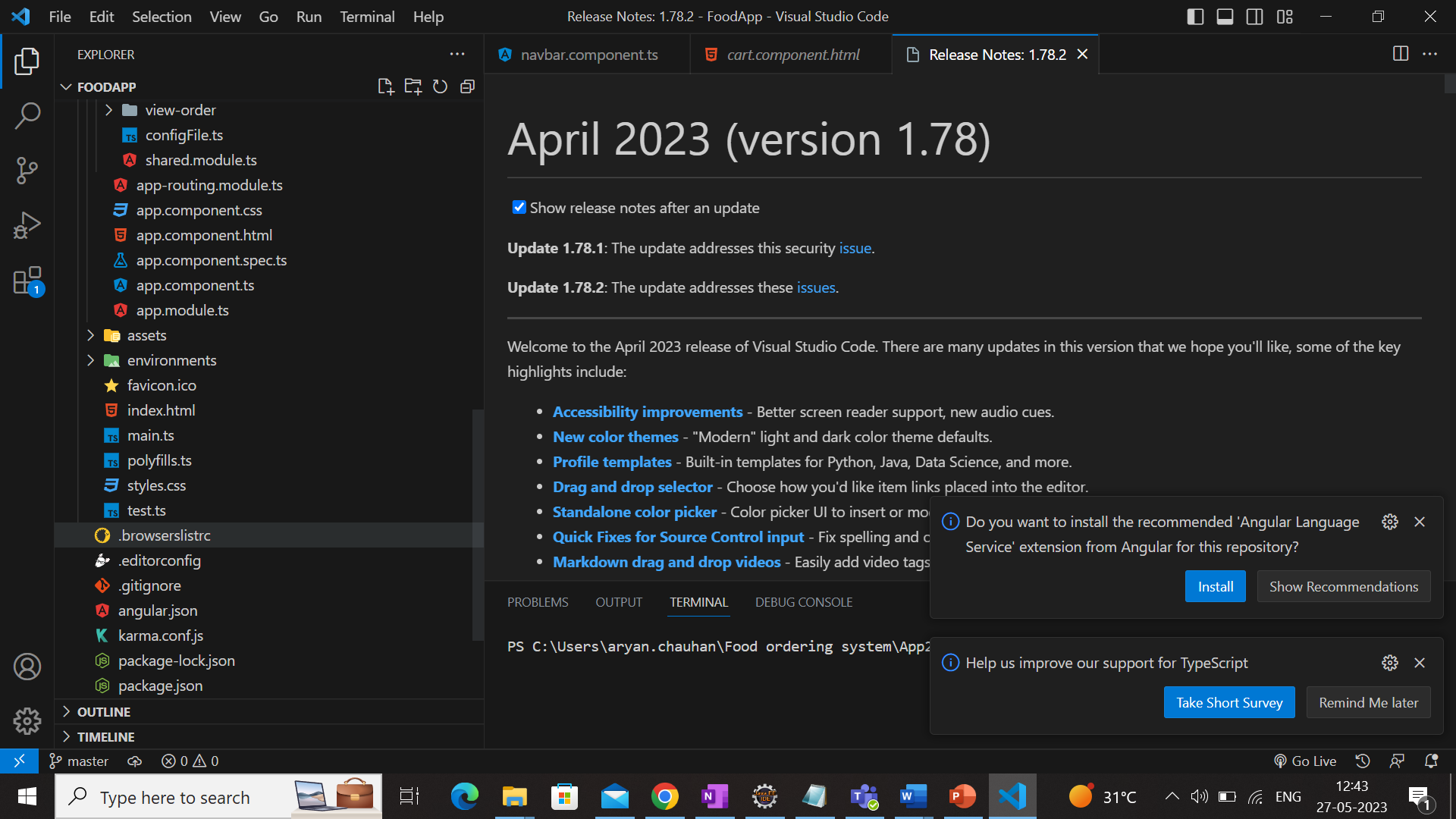
 

Fig 5: File structure in Angular project

**1.3 Event Handling**

In Angular, event handling is used to hear and capture all the events like clicks, mouse

movements, keystrokes etc. It is an important feature that is present in Angular, and it is used in every project irrespective of its size.

Syntax:

<HTML element (event) = function\_name()>

Explanation of Syntax:

HTML elements can be used like <button> tag, input tag etc.

In events, we can use many events that are present like (click), (change) etc.

We need to give a function name in strings, and we need to write the implementation in the ts file.

Approach:

According to the above example declare an event handler with any of the key events in the HTML file.

In ts file write the implementation of the function according to the requirement.

Below is an example where we used different events to use the concept.

The one is (change) event and the second one is (click).

A computer screen shot of a program

Description automatically generated with low confidenceFig 6: Event handling example

**CHAPTER​ 2: BACKEND**

**2​.1 Java Basics**

When we consider a Java program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what class, object, methods, and instance variables mean.

* Object: Objects have states and behaviours. Example: A dog has states - color, name, breed as well as behaviour such as wagging their tail, barking, eating. An object is an instance of a class.
* Class: A class can be defined as a template/blueprint that describes the behaviour/state that the object of its type supports.
* Methods: A method is basically a behaviour. A class can contain many methods. It is in methods where the logics are written, data is manipulated, and all the actions are executed.
* Instance Variables: Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

**2.2 Programming Paradigm**

Java as other modern general purpose programming languages started with fewer programming paradigms and kept adapting to new paradigms as the language editions kept evolving. With its recent editions we can consider Java as a language that supports multi-paradigms.**​**

2.2.1 Object Oriented Programming

Java is an Object-Oriented Programming OOP language at heart. OOP languages model the world by classes. Every class has its own attributes, also known as fields, and a number of methods that might change the state of those fields, also known as behavior. An object is a real instance of a certain class. Similarly, to the real world, classes can be part of other classes that are usually less restricted. Therefore, they can inherit their certain fields and methods from their parent class i.e., superclass.

2.3 Objects & Classes in Java

Let us now look deep into what objects are. If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior. If we consider a dog, then its state is - name, breed, color, and the behavior is - barking, wagging the tail, running. If you compare the software object with a real-world object, they have very similar characteristics.

So, in software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.

A class is a blueprint from which individual objects are created. A class can contain any of the following variable types.

* Local variables: Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
* Instance variables: Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
* Class variables: Class variables are variables declared within a class, outside any method, with the static keyword.

**2.4 Pillars of OOPS**

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

2.4.1 Encapsulation

It is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

1. Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of its own class in which they are declared.
2. As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding.
3. Encapsulation can be achieved by declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

2.4.2 Abstraction

Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essentials units are not displayed to the user.

2.4.3 Inheritance

Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class.

Let us discuss some of frequent used important terminologies:

* Super Class: The class whose features are inherited is known as superclass (or a base class or a parent class).
* Sub Class: The class that inherits the other class is known as subclass (or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
* Reusability: Inheritance supports the concept of “reusability”, i.e., when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

2.4.4 Polymorphism

It refers to the ability of OOPs programming languages to differentiate between entities with the same name efficiently. This is done by Java with the help of the signature and declaration of these entities.

Polymorphism in Java are mainly of 2 types:

* Overloading: Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters or both. Overloading is related to compile-time (or static) polymorphism.
* Overriding: In any object-oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super classes or parent classes. When a method in a subclass has the same name, same parameters or signature, and same return type (or sub-type) as a method in its super-class, then the method in the subclass is said to *override* the method in the super-class.

**2.5 JDBC (Java Database Connectivity)**

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented toward relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

* JDBC-ODBC Bridge Driver,
* Native Driver,
* Network Protocol Driver, and
* Thin Driver

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.

A diagram of a driver

Description automatically generated with low confidence

Fig 7: JDBC structure

The java.sql package contains classes and interfaces for JDBC API. A list of popular interfaces of JDBC API are given below:

* Driver interface
* Connection interface
* Statement interface
* PreparedStatement interface
* CallableStatement interface
* ResultSet interface
* ResultSetMetaData interface
* DatabaseMetaData interface
* RowSet interface

A list of popular classes of JDBC API are given below:

* DriverManager class
* Blob class
* Clob class
* Types class

**2.6 Java Database Connectivity with 5 Steps**

A picture containing text, screenshot, LEGO

Description automatically generated

Fig 8: Steps of Java Database Connectivity

**2.7** **Spring Boot**

2.7.1 Micro Service

Micro Service is an architecture that allows the developers to develop and deploy services independently. Each service running has its own process and this achieves the lightweight model to support business applications.

​2.7.2 Spring Boot

Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run. You can get started with minimum configurations without the need for an entire Spring configuration setup.

Advantages:

Spring Boot offers the following advantages to its developers −

* Easy to understand and develop spring application.
* Increases productivity.
* Reduces the development time.

Goals:

Spring Boot is designed with the following goals −

* To avoid complex XML configuration in Spring
* To develop a production ready Spring applications in an easier way
* To reduce the development time and run the application independently.
* Offer an easier way of getting started with the application.

How​ does it work?

Spring Boot automatically configures your application based on the dependencies you have added to the project by using @EnableAutoConfiguration annotation. For example, if MySQL database is on your classpath, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.

The entry point of the spring boot application is the class contains @SpringBootApplication annotation and the main method.

Spring Boot automatically scans all the components included in the project by using @ComponentScan annotations.

2.7.5 Spring Boot Architecture

A diagram of a service

Description automatically generated with low confidence

Fig 9: Spring Boot Architecture

* Now we have validator classes, view classes, and utility classes.
* Spring Boot uses all the modules of Spring-like Spring MVC, Spring Data, etc. The architecture of Spring Boot is the same as the architecture of Spring MVC, except one thing: there is no need for DAO and DAOImpl classes in Spring boot.
* Creates a data access layer and performs CRUD operation.
* The client makes the HTTP requests (PUT or GET).
* The request goes to the controller, and the controller maps that request and handles it. After that, it calls the service logic if required.
* In the service layer, all the business logic performs. It performs the logic on the data that is mapped to JPA with model classes.
* A JSP page is returned to the user if no error occurred. The main aim of spring boot is to remove the XML and annotations-based configuration settings from the application. Along with this spring boot provides the following benefits such as opinionated (options to later change the configuration), convention over configuration, stand-alone, and production ready.

2.7.8 Spring Boot Application

The entry point of the Spring Boot Application is the class contains @SpringBootApplication annotation. This class should have the main method to run the Spring Boot application. @SpringBootApplication annotation includes Auto- Configuration, Component Scan, and Spring Boot Configuration.

If you added @SpringBootApplication annotation to the class, you do not need to add the @EnableAutoConfiguration, @ComponentScan and @SpringBootConfiguration annotation. The @SpringBootApplication annotation includes all other annotations.

**Project Demo**

The project displayed below is a Online Food delivery app, which was made entirely by me, for the first part of my 6-month industrial training. The purpose of the project was to utilize and showcase our learning from the sessions we were provided on the technologies used in the company, as well as to develop the basic understanding and cross-functionality of the software and the technologies being used across the development of the product “DGSECURE”, before getting started with the product.

Technologies Used-

1. Java - used in backend logic and validations.
2. Spring Boot - for making RESTful API’s.
3. Postgres Database - for storing all data.
4. Angular - for building frontend UI of web application.
5. Bootstrap - CSS styling framework.

Tools Used -

1. Gradle - build tool.
2. PgAdmin - for visualising postgres database
3. Git - Version control system
4. VsCode - IDE

Components –

1. Login and Register
2. View Items
   1. Pagination implemented.
   2. Filters implemented.
3. Place and view orders
4. Manage Inventory
5. Support Metrics
6. Manage Orders

**Snapshots of the project**

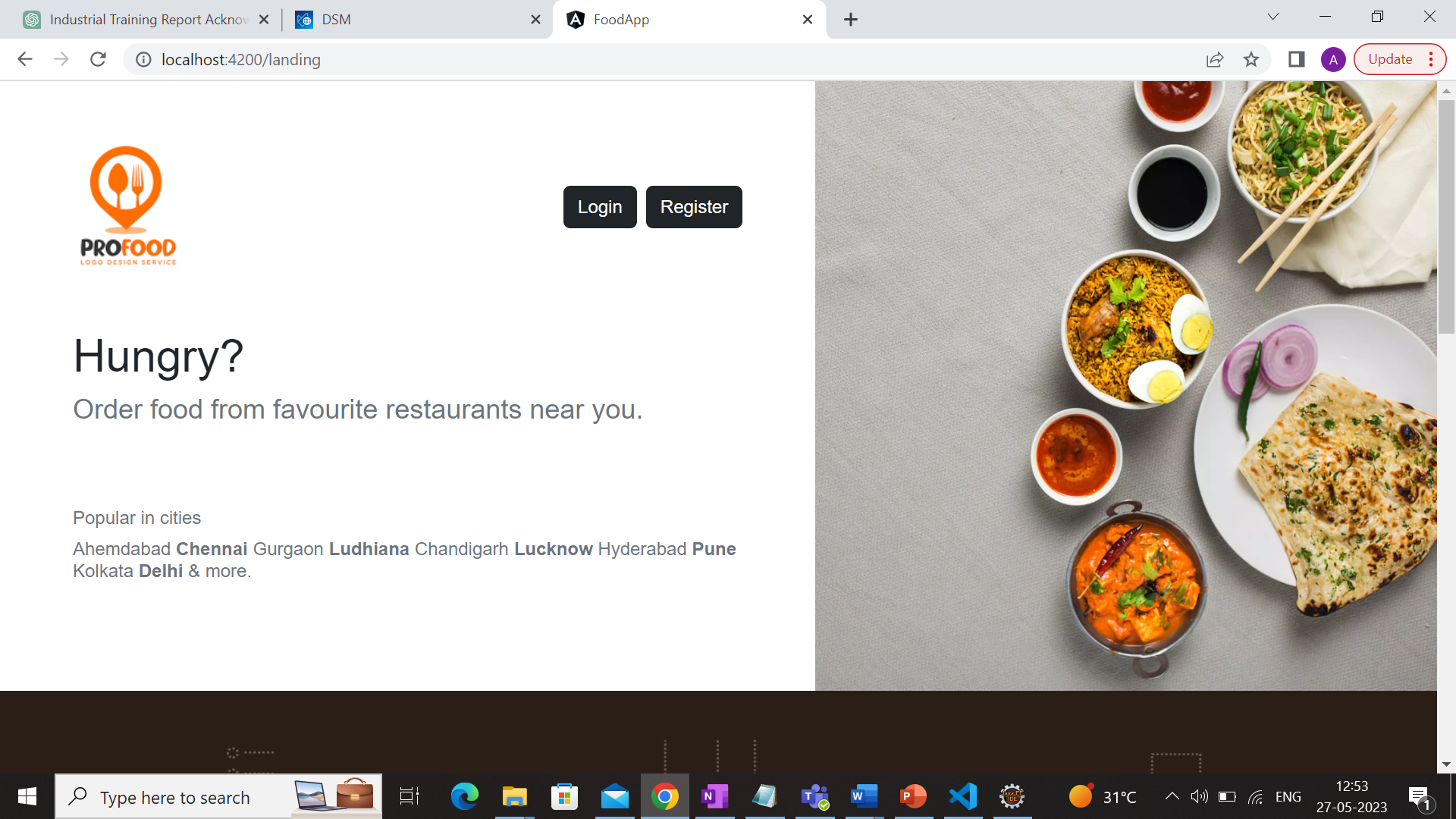
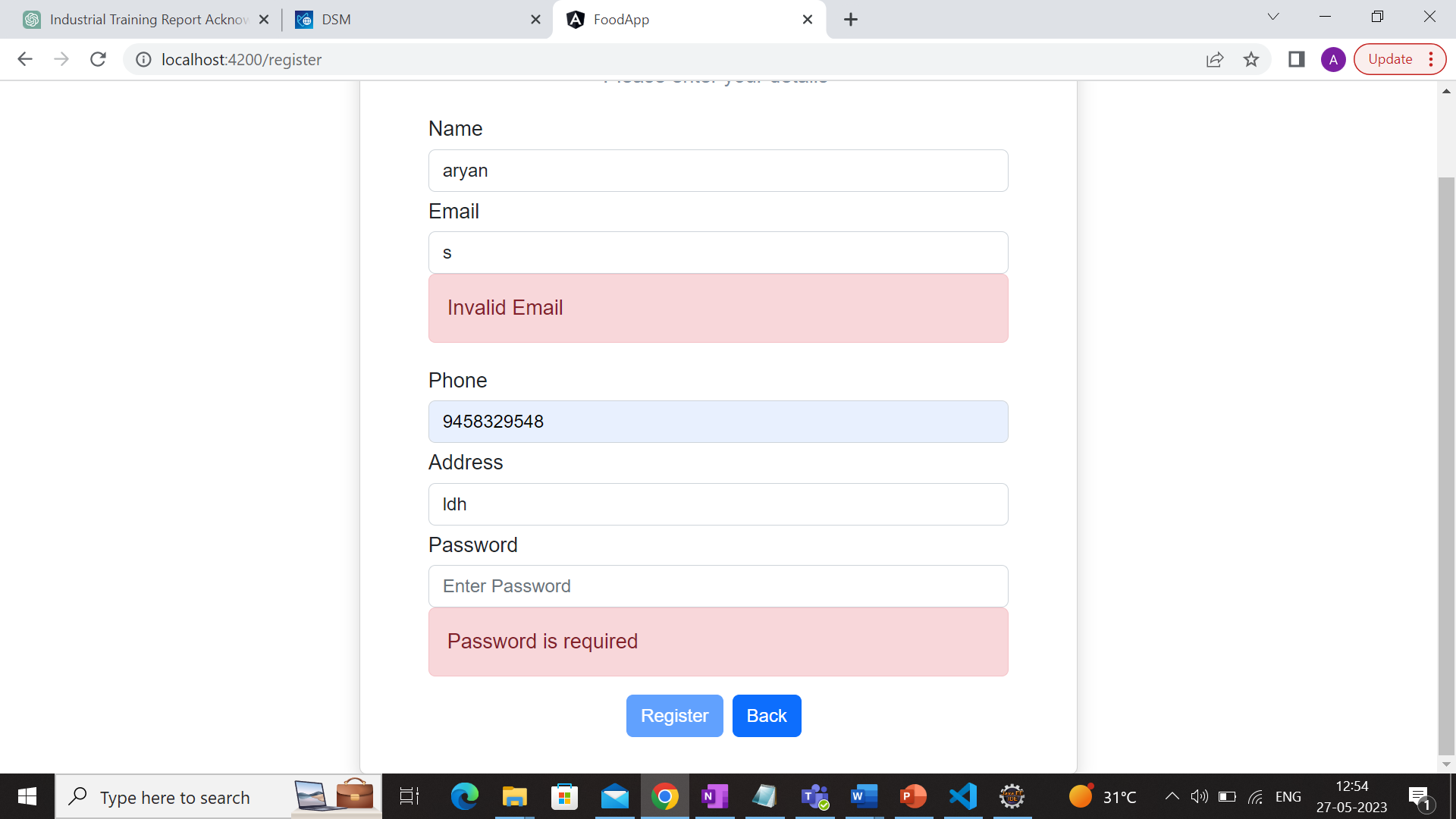


Fig 10: Landing page

 Fig 11: Registration Form

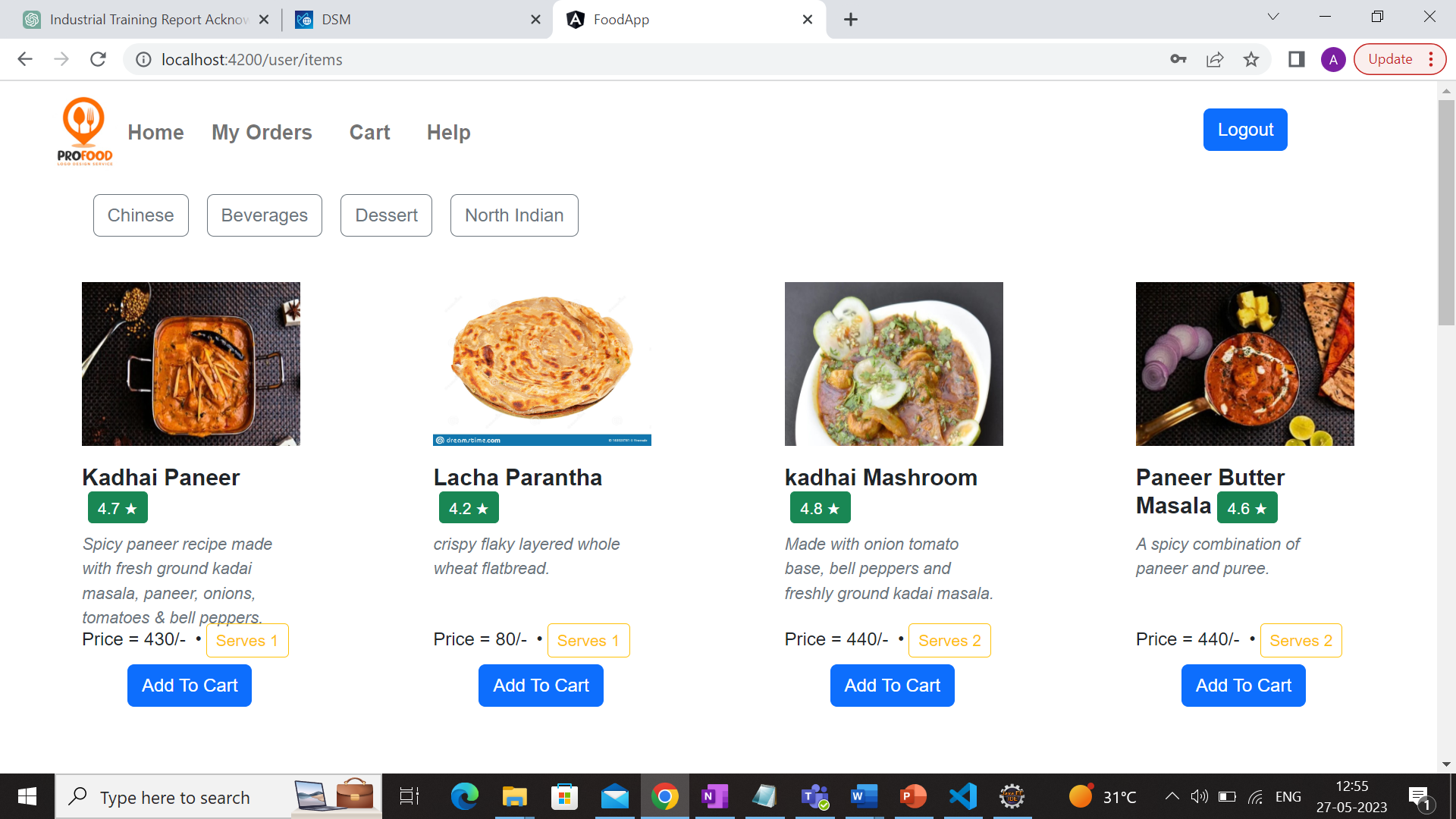


Fig 12: User View

A screenshot of a computer

Description automatically generated Fig 13: Cart

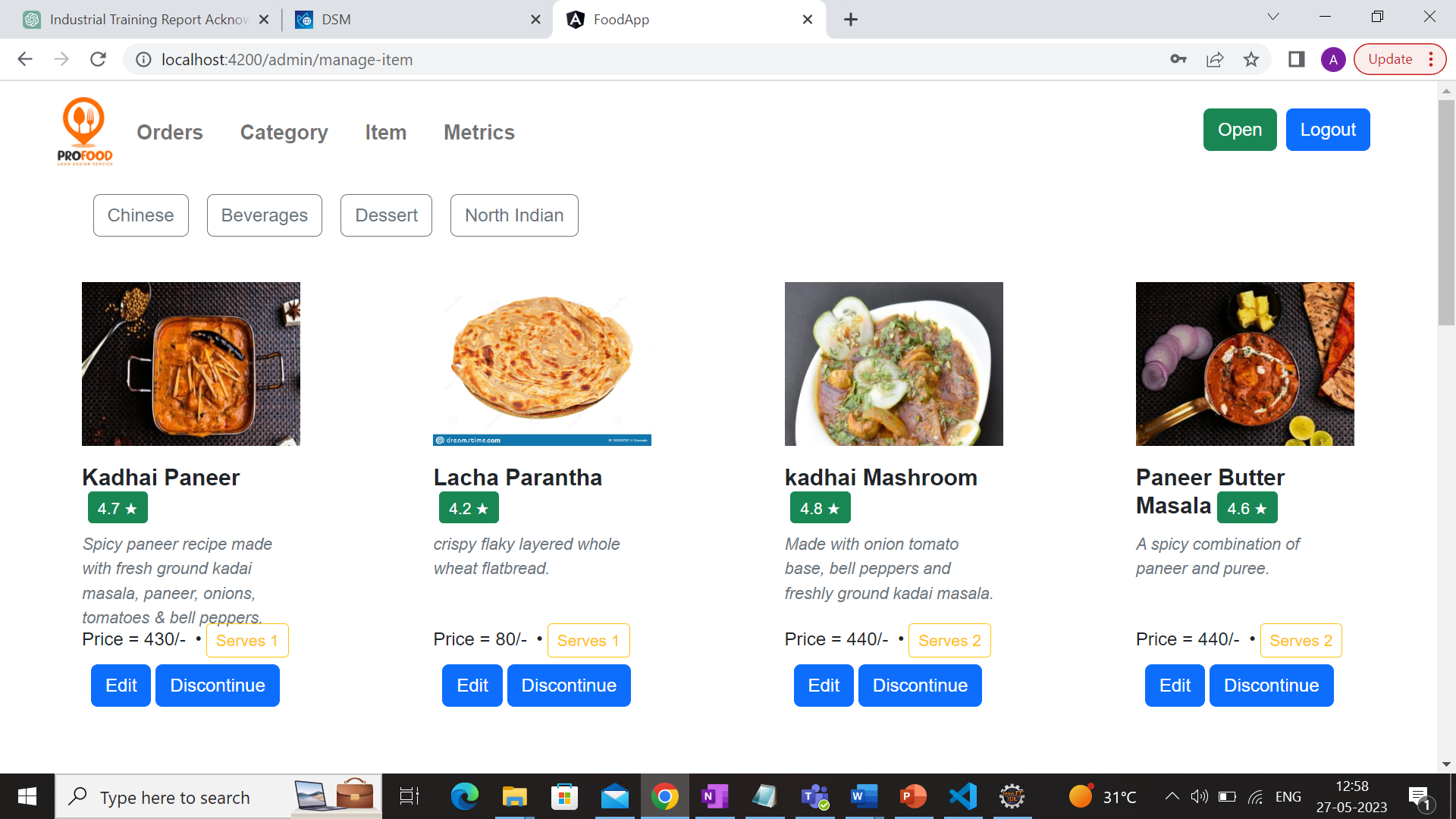
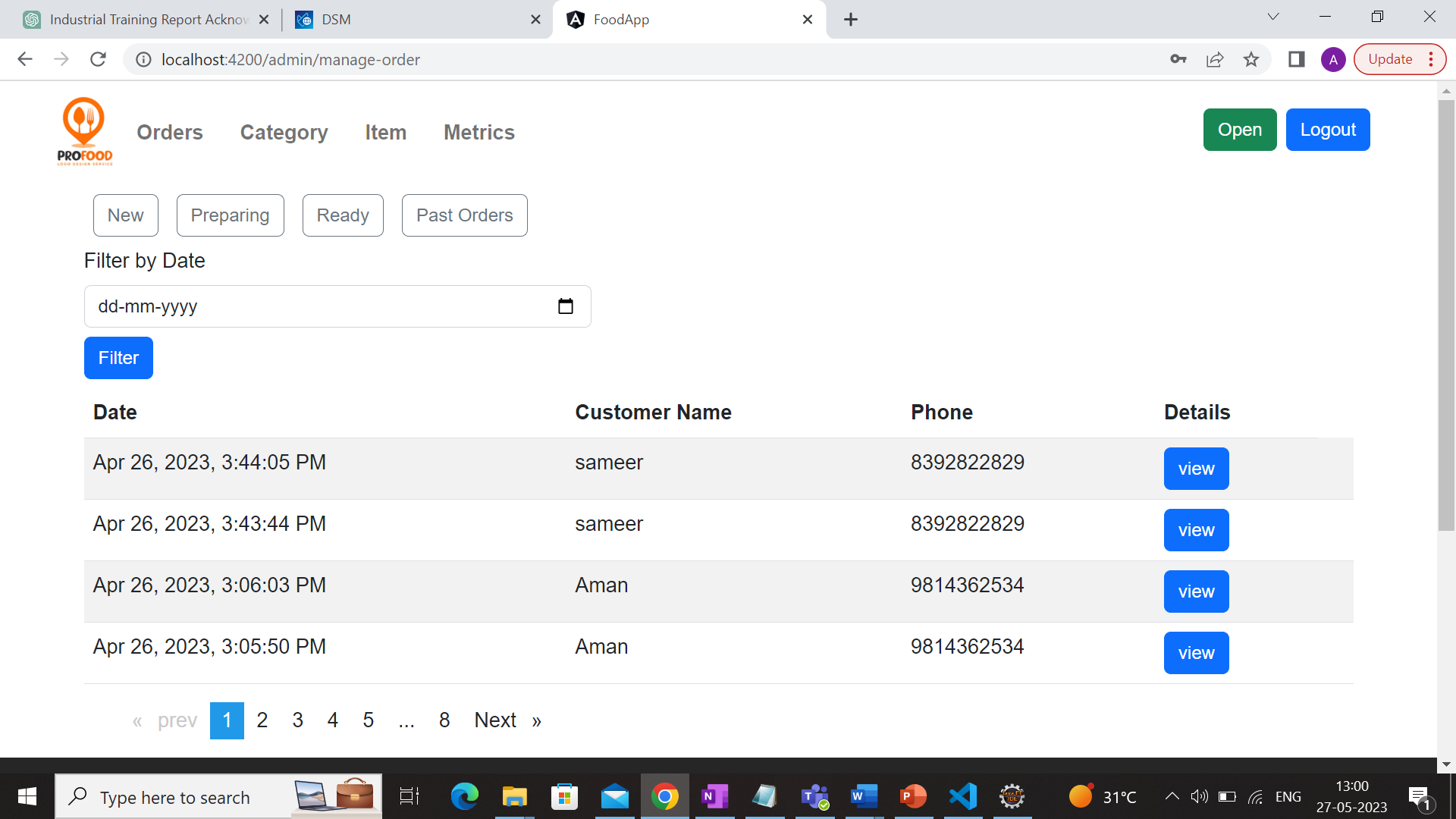


Fig 14 Admin-side Manage Inventory

 Fig 15: Manage Orders

A screenshot of a computer

Description automatically generated Fig 16: Support Metrics

A screen shot of a computer

Description automatically generated with medium confidence

Fig 17 Pie-Chart

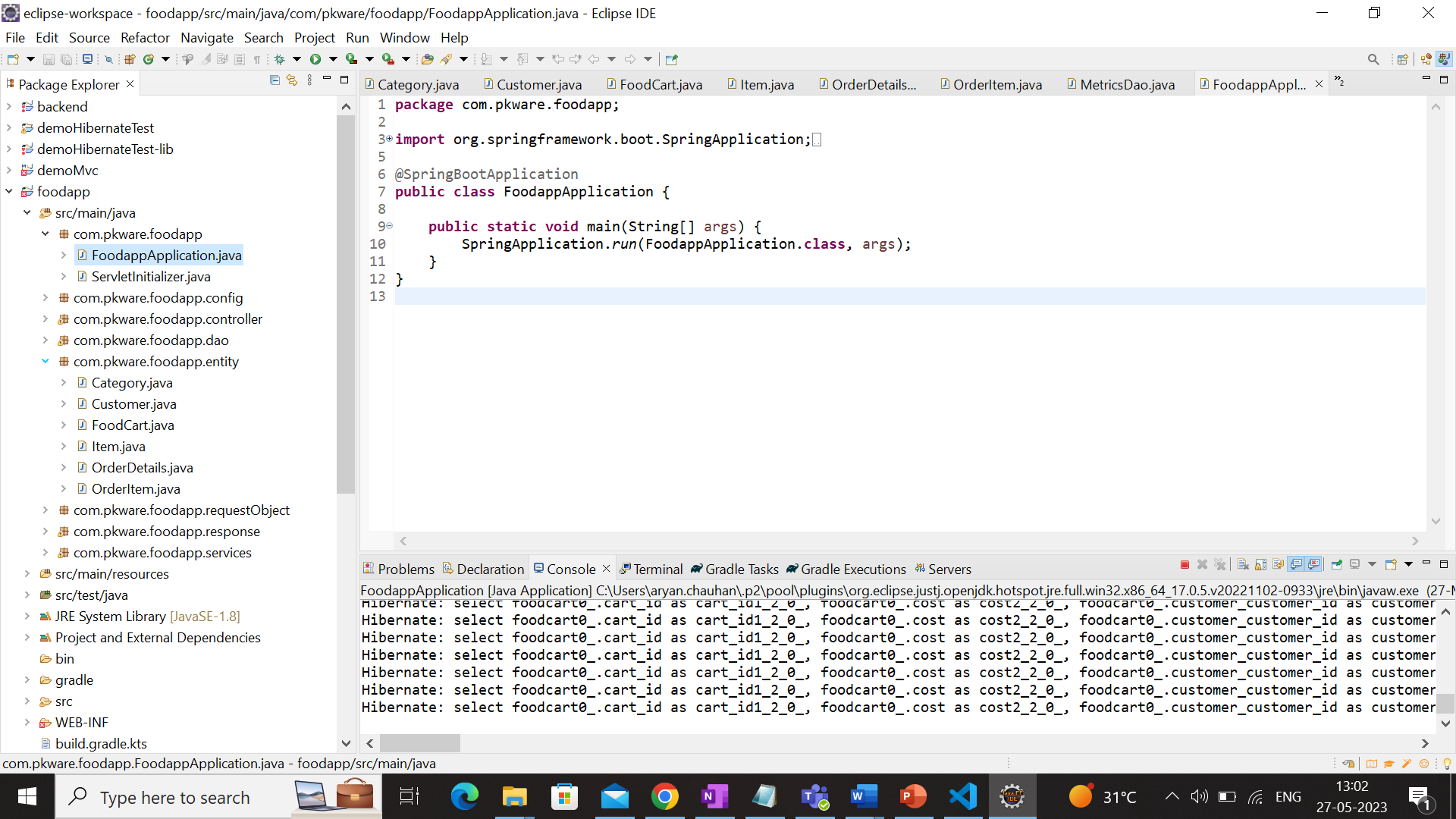


Fig 18: Backend Project structure in Eclipse

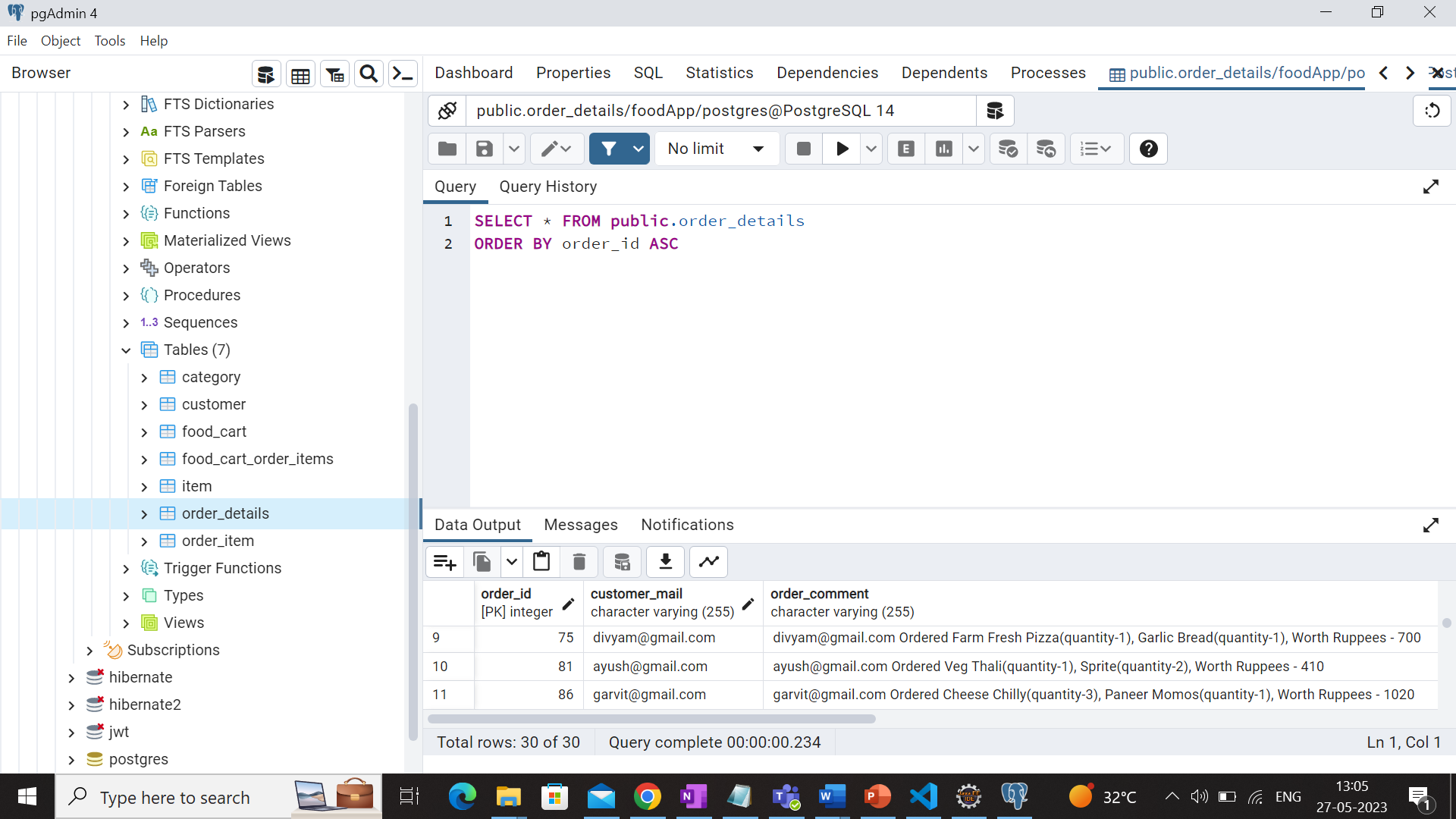


Fig 19: Database tables in Pgadmin

**Conclusion**

In conclusion, my internship with Dataguise – A PKWARE Company as a Java Software Development has been a valuable learning experience that helped me develop my analytical, communication, teamwork, and problem-solving skills.

During these 4 months of Industrial Training, I have learned a lot of new and interesting technologies. It has been a really great experience to watch and learn how a company operates. Being a part of such a talented organisation filled with aspiring and encouraging people has shown me a great path for my career. I would like to learn more and more and develop new and professional skills.

I would really like to thank all the teachers and mentors and UIET, Panjab University for providing me with this opportunity to learn and grow. I am looking forward to contributing more as and when there is an opportunity where I can contribute on what I learnt and will be learning in future and make best use of the experience.

**Bibliography**

1. “The Complete Reference Java, Seventh Edition” by Herbert Shiltd.
2. https://www.java.com/en/
3. https://spring.io/
4. https://spring.io/projects/spring-boot
5. https://gradle.org/
6. https://www.postgresql.org/
7. https://angular.io/
8. https://getbootstrap.com/
9. https://www.geeksforgeeks.org/
10. https://www.pkware.com/