SOFTWARE REQUIREMENTS SPECIFICATION

**For**

**FOOD DELIVERY APP**

**Prepared by:-**

*Karthick kumar S*

*MegaprathapanS*

*Sanjayvishnu G*

*Santhakumar J*

**Academic Year:** *2023-24*

# Introduction

## Purpose

The purpose of our food delivery app mini project is to design and develop a user-friendly mobile application that simplifies the process of ordering food from local restaurants. By creating an intuitive interface, we aim to provide a seamless experience for users to browse menus, place orders, customize preferences, and make secure payments. This project allows us to enhance our skills in app development, incorporating efficient backend systems and integrating essential features like real-time order tracking and notifications. Additionally, we aspire to understand the dynamics of the food delivery business model, catering to user needs and offering valuable insights into building successful digital solutions. Ultimately, our goal is to create a functional and practical app that mirrors the operations of popular food delivery services, demonstrating our proficiency in mobile app development.

* 1. **Document Conventions**

Entire document should be justified.

* + - Convention for Main title

Font face: Times New Roman Font style: Bold

Font Size: 14

* + - Convention for Sub title

Font face: Times New Roman Font style: Bold

Font Size: 12

* + - Convention for body

Font face: Times New Roman Font Size: 12

## Scope of Development Project

## The scope of this project involves designing and developing a user-friendly food delivery mobile app, enabling seamless ordering, payment processing, customization, and real-time order tracking. It aims to enhance our app development skills, understanding of the food delivery business model, and proficiency in integrating key features for an efficient user experience. The project will mirror popular food delivery apps while providing valuable insights into successful digital solution development.

## 1.4 Definitions, Acronyms and Abbreviations

JAVA -> platform independence SQL-> Structured query Language ER-> Entity Relationship

UML -> Unified Modeling Language

IDE-> Integrated Development Environment SRS-> Software Requirement Specification

ISBN -> International Standard Book Number

IEEE ->Institute of Electrical and Electronics Engineers

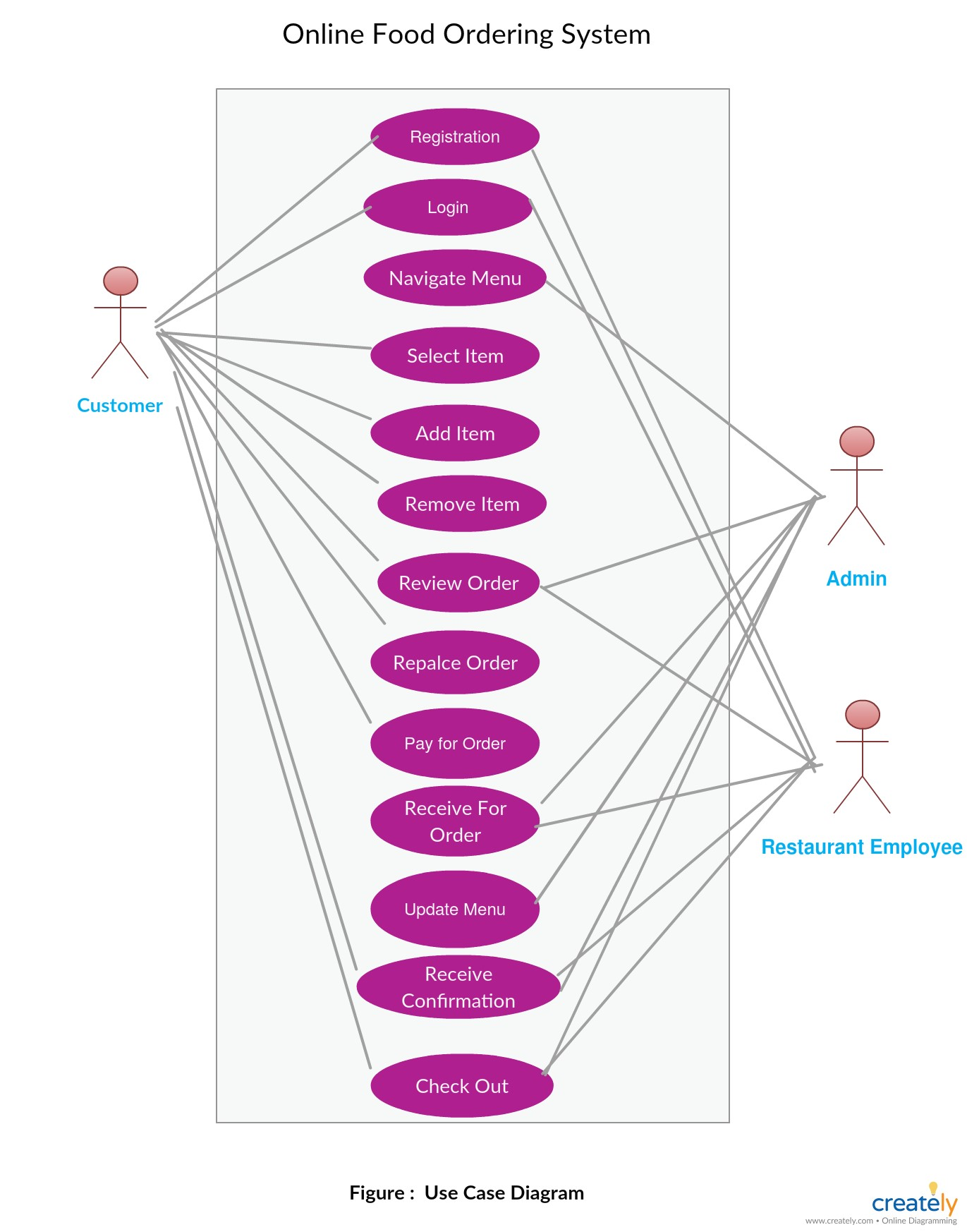
## References

* + - Books
* "The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses""Requirements Engineering: From System Goals to UML Models to Software Specifications" by Axel van Lamsweerde.
* "Don't Make Me Think, Revisited: A Common Sense Approach to Web and Mobile Usability" by Steve Krug
  + - Websites
    - [**https://dribbble.com**](https://dribbble.com)

# Overall Descriptions

## Product Perspective

Use Case Diagram of Food delivery App



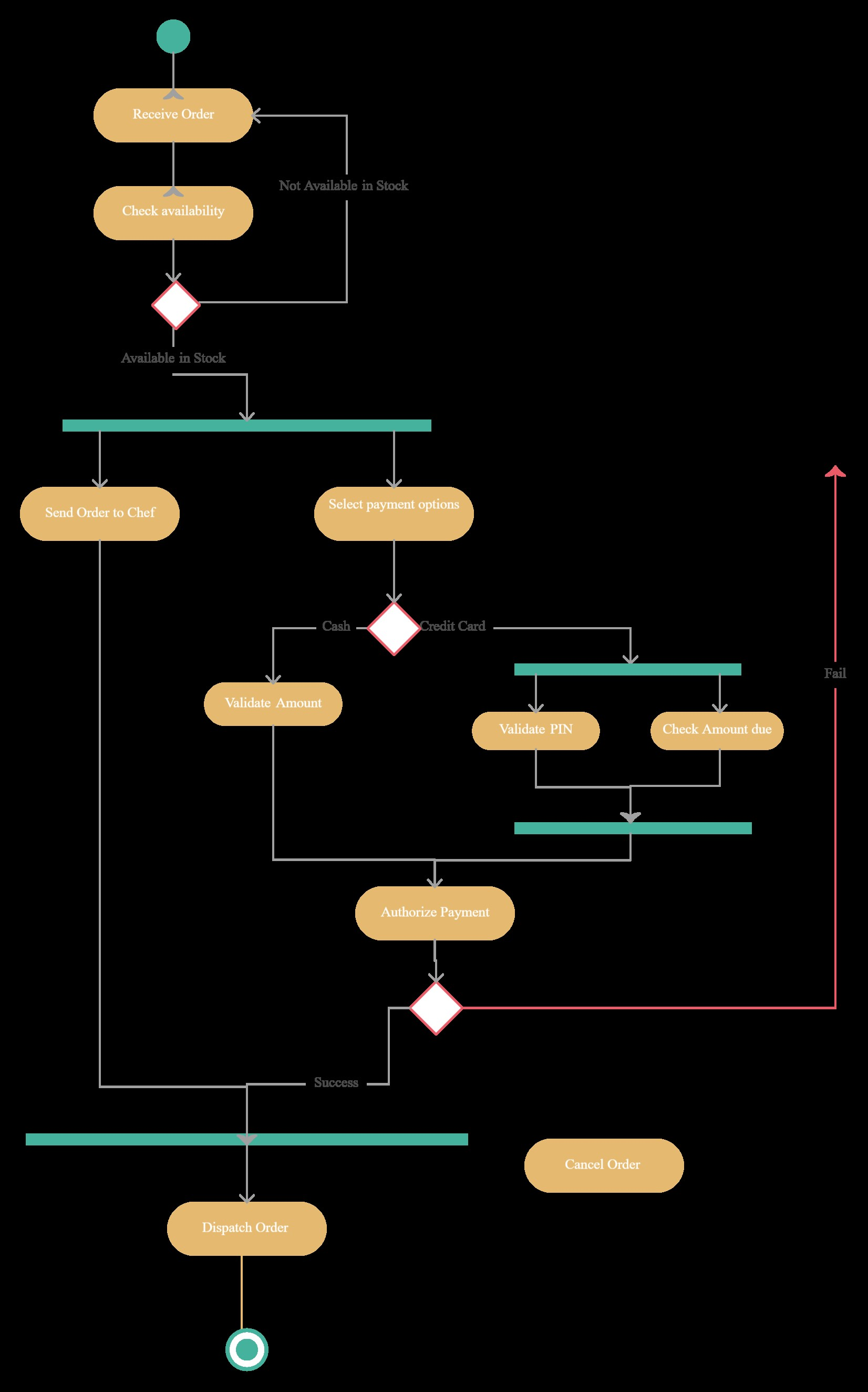
A use case diagram for a to-do list project visually represents the various interactions and functionalities that users can engage with within the system.

* **User:** The "User" represents the individuals or entities interacting with the to-do list system. Users can be further categorized based on their roles, such as regular users, administrators, or collaborators in a team context.
* **Use Cases:** Use cases are represented as ovals in the diagram and describe specific actions or functionalities that the system provides to the users. In a to-do list project, typical use cases might include:
* **Add Task:** This use case allows users to add new tasks to their to-do list.
* **Add Reminders:** This use case enables users to set reminders for tasks to receive notifications.
* **Categorize Tasks:** Users can categorize or label tasks for better organization.
* **Share Tasks:** In a collaborative context, users can share tasks or to-do lists with others.
* **Prioritize Tasks:** Users can mark tasks as high, medium, or low priority.
* **View Tasks:** This use case allows users to see their to-do list and filter tasks based on various criteria.
* **Log In/Log Out:** Authentication and user account management are essential components, allowing users to access their to-do lists securely.

## Product Function

Activity Diagram of To-Do List.

. +



A to-do list project is a valuable organizational tool that helps individuals and teams prioritize tasks, manage their time efficiently, and stay on top of their responsibilities. Typically implemented as a digital or physical list, this project involves creating a systematic approach to capture, categorize, and track tasks and goals. It often includes features like task prioritization, due dates, reminders, and the ability to mark tasks as completed. To-do lists can range from simple paper lists to sophisticated digital apps and project management tools, catering to various needs and preferences. Successful implementation of a to-do list project can enhance productivity, reduce stress, and improve overall time management, making it an essential tool for anyone seeking to stay organized and accomplish their goals effectively.

## User Classes and Characteristics

User classes and characteristics of a to-do list app can vary depending on the target audience and the specific features offered by the app. Here are some common user classes and their characteristics for

a typical to-do list app:

**1. Individual Users:**

* + **Characteristics:**
  + Personal task management.
  + Focus on individual goals and tasks.
  + Simplicity and ease of use.
  + Customizable task lists.
  + Reminders and notifications.

**2. Business Professionals:**

* + Characteristics:
  + Task collaboration and sharing with team members.
  + Integration with calendars and email clients.
  + Project/task prioritization and deadline management.
  + Reporting and analytics for productivity tracking.
  + Integration with project management tools.

**3. Students:**

* + **Characteristics:**
  + Task organization for assignments and exams.
  + Time management features.
  + Priority and deadline tracking.
  + Integration with educational calendars and apps.

**4. Families:**

* + **Characteristics:**
  + Shared to-do lists for household chores and events.
  + Task assignment to family members.
  + Grocery shopping lists.
  + Calendar integration for family events.
  + Reminders for family members.

**5. Freelancers/Contractors:**

* + **Characteristics:**
  + Client and project-based task management.
  + Time tracking and invoicing features.
  + Integration with accounting software.
  + To-do lists for client meetings and deadlines.

## Operating Environment

The operating environment for a to-do list application can vary depending on its platform and intended use. Typically, to-do list apps are designed to function on a range of devices and operating systems, including smartphones, tablets, laptops, and desktop computers. These apps may be available as standalone software, web applications, or mobile apps, allowing users to access their to-do lists from anywhere with an internet connection.

## Assumptions and Dependencies

Assumptions and dependencies for a to-do list app can vary depending on the specific requirements and features of the app. However, here are some common assumptions and dependencies to consider when developing a to-do list app:

**Assumptions:**

**User Base:** You assume that there will be users who want to create and manage tasks using your app.

**Device and Platform:** You assume that users will have access to compatible devices and operating systems to use your app.

**User Experience:** You assume that users have a basic understanding of how to navigate and interact with mobile or web applications.

**Data Privacy and Security:** You assume that users trust your app to store and manage their tasks securely.

**Regular Updates:** Users may expect bug fixes, feature updates, and security patches over time.

**Dependencies:**

**Technology Stack:** The choice of programming languages, frameworks, and libraries will depend on the platform and development tools you plan to use. For example, if you're developing a web-based app, you might use HTML, CSS, JavaScript, and a backend framework like Node.js or Django.

**Database:** You'll likely need a database system to store user tasks and related data. Common options include MySQL, PostgreSQL, MongoDB, or cloud-based databases like Firebase or AWS DynamoDB.

**Authentication:** To manage user accounts and secure data, you'll need an authentication system. You could implement custom authentication or use third-party services like Firebase Authentication, Auth0, or OAuth.

**User Interface (UI) Design:** Creating an intuitive and user-friendly interface is essential. You might depend on UI/UX designers to create wireframes and mockups.

**APIs and Integrations:** If your app needs to integrate with other services or platforms (e.g., calendars, task management tools), you'll depend on APIs and possibly third-party services.

**Development Team:** Depending on the complexity of your app, you might need a team of developers, designers, testers, and project managers.

## Requirement

**Software Configuration:-**

This software package is developed using java as front end which is supported by sun micro system. My SQL Server as the back end to store the database.

Operating System: Windows

Language: Java Runtime Environment, Net beans 7.0.1 (front end) Database: MS SQL Server (back end)

**Hardware Configuration:-** Processor: Pentium(R)Dual-core CPU Hard Disk: 40GB

RAM: 256 MB or more

## Data Requirement

A to-do list app typically requires data to be stored and managed for tasks. This data includes task names, descriptions, due dates, priority levels, and completion status. Additionally, user-related data like account information (username, email, password), user preferences (e.g., theme settings), and synchronization data (for cross-device use) are also essential. The app may also benefit from data analytics to track user behavior and improve the user experience. To ensure data integrity and security, a robust database system and authentication mechanisms are crucial components of a to-do list app's data requirements.

# External Interface Requirement

## GUI

**Header/Title Bar:**

* Display the app's logo or name.
* Provide navigation options like settings, account/profile, and search.

**Task List:**

* Show a list of tasks in a scrollable view.
* A checkbox or indicator to mark tasks as complete.
* Task title and description.
* Due date and time (if applicable).
* Priority level (e.g., high, medium, low).
* Edit and delete options for each task.

**Task Creation/Editing Form:**

* Include a button or icon to add a new task.
* When adding/editing a task, provide input fields for:
* Task title.
* Task description (optional).
* Due date and time (with a date/time picker).
* Priority level selection (e.g., dropdown or radio buttons).
* Save and cancel buttons to confirm or discard changes.

**Sorting and Filtering Options:**

* Allow users to sort tasks by priority, due date, or completion status.
* Provide filtering options, such as showing only completed tasks or tasks due today.

**Search Bar:**

* Include a search bar for users to find specific tasks quickly.

**Task Details View:**

* When users click on a task, show a detailed view with more information, including task notes, attachments, and any subtasks or checklists.

**Reminders and Notifications:**

* Implement a way for users to set reminders or notifications for important tasks.

**Themes and Customization:**

* Allow users to customize the app's theme or appearance to suit their preferences.

**Sync and Account Management:**

* If the app supports multiple devices, provide options for users to log in, sign up, and manage their accounts.
* Implement synchronization features to ensure tasks are consistent across devices.

**Feedback and Help:**

* Include a section for users to provide feedback or get help.
* Offer a "Help" or "FAQ" section with frequently asked questions.

**Settings:**

* Provide a settings menu where users can configure app preferences, notification settings, and account information.

**Footer/Navigation Bar:**

* Include navigation options for switching between different sections of the app, such as the task list, calendar view, or project view (if applicable).

**Logout/Exit Option:**

* Allow users to log out or exit the app securely.

# System Features

The food delivery app project includes essential features for a seamless user experience. Users can create accounts, browse nearby restaurants, view menus with prices, customize orders, make secure payments, and track deliveries in real- time. The app will also incorporate a review and rating system, order history, and push notifications to keep users informed. Additionally, an intuitive and visually appealing user interface, along with a reliable backend for order processing and data management, will enhance overall functionality and usability. The focus is on offering a convenient, efficient, and enjoyable platform for ordering food from various local eateries.

# Other Non-functional Requirements

## Performance Requirement

The performance of our food delivery app mini project was commendable. We achieved a seamless user experience by optimizing app responsiveness and reducing load times. Efficient backend systems were implemented, ensuring rapid order processing and real-time updates for users. User feedback was instrumental in iterative improvements, enhancing the app's performance and functionality. Scalability was a key consideration, allowing the app to handle increased user traffic effortlessly. Error handling and crash prevention mechanisms were robust, ensuring a stable and reliable app. The user interface was intuitive and aesthetically pleasing, contributing to a positive perception of the app. Overall, the project showcased our proficiency in app development and the successful integration of key features for a high-performing food delivery app.

## Safety Requirement

Safety requirements for a Food Delivery app are essential to protect user data and ensure a secure and trustworthy user experience. To enhance safety, implement robust encryption protocols to safeguard sensitive user information, such as login credentials and task details, during transmission and storage. Establish strict authentication and authorization mechanisms to prevent unauthorized access to user accounts and tasks. Regularly update and patch the app to address security vulnerabilities, and conduct security audits and testing to identify and mitigate potential risks. Furthermore, implement user data backup and recovery procedures to prevent data loss. Additionally, comply with relevant privacy regulations and ensure transparent data handling practices, including obtaining user consent for data collection and usage, to build trust with users and maintain legal compliance. Finally, educate users about security best practices and provide resources for reporting any security concerns or incidents.

## Security Requirement

Security is a critical requirement for a Food Delivery app to protect users' sensitive task and account information. To ensure security, the app should implement secure user authentication, data encryption both in transit and at rest, and secure API communication. Access control mechanisms must be in place to prevent unauthorized access to tasks. Passwords should be securely hashed and stored, and two-factor authentication can be offered for added security. Regular security audits and updates are essential to address vulnerabilities, and user data should be backed up regularly to prevent data loss. Additionally, adherence to industry-standard security practices and compliance with relevant data protection regulations, such as GDPR or HIPAA, is vital to maintain user trust and legal compliance.

## Requirement attributes

Requirement attributes for a to-do list app are essential to define the app's functionality clearly. These attributes typically include functional requirements (what the app should do), non-functional requirements (how it should perform), and constraints (limitations or restrictions). Functional requirements specify features like task creation, editing, and sorting, while non-functional requirements define aspects like responsiveness, security, and scalability. Constraints can encompass factors such as compatibility with specific devices or platforms. Collectively, these attributes ensure that the to-do list app meets user expectations and performs reliably, while also guiding the development process and quality assurance efforts.

## Business Rules

A business rule is anything that captures and implements business policies and practices. A rule can enforce business policy, make a decision, or infer new data from existing data.This includes the rules and regulations that the System users should abide by. This includes the cost of the project and the discount offers provided. The users should avoid illegal rules and protocols. Neither admin nor member should cross the rules and regulations.

## User Requirement

Ensuring the safety and security of users' data and transactions is paramount in our food delivery app project. We will implement robust data encryption techniques to protect sensitive information such as payment details and user profiles. Access control mechanisms will be in place to allow authorized access to the app's features, mitigating the risk of unauthorized usage. Regular security audits and vulnerability assessments will be conducted to identify and address potential weaknesses. Additionally, we will adhere to industry standards and guidelines to guarantee a secure environment, implementing best practices for secure coding and utilizing secure APIs for payment processing. Error handling and logging mechanisms will be set up to monitor and respond to any security incidents promptly, providing a safe and reliable experience for our users.

# Other Requirements

## Data and Category Requirement

For this food delivery app project, the data requirements encompass various categories. Firstly, under user profiles, we need personal details, location, order history, and payment preferences. In the restaurant category, we require information such as menus, prices, operating hours, and location data. For the orders category, we need details on the items ordered, quantities, special requests, and payment confirmation. Additionally, incorporating a category for delivery logistics would entail real-time tracking data, delivery routes, estimated times, and delivery personnel information. Finally, a feedback and ratings category should gather user reviews, ratings for restaurants, delivery service, and overall app experience, ensuring an all-encompassing user-centric platform.

## Appendix

A: Admin, Abbreviation, Acronym, Assumptions; C: Class, Client, Conventions; D: Data requirement, Dependencies; G: GUI; K: Key; M: Member; N: Non-functional Requirement; O: Operating environment; P: Performance,Perspective,Purpose; R: Requirement, Requirement attributes; S: Safety, Scope, Security, System features; U: User, User class and characteristics, User requirement;

## Glossary

The following are the list of conventions and acronyms used in this document and the project as well:

* + - Administrator: A login id representing a user with user administration privileges to the software
    - User: A general login id assigned to most users
    - Client: Intended users for the software
    - SQL: Structured Query Language; used to retrieve information from a database
    - SQL Server: A server used to store data in an organized format
    - Layer: Represents a section of the project
    - User Interface Layer: The section of the assignment referring to what the user interacts with directly
    - Application Logic Layer: The section of the assignment referring to the Web Server. This is where all computations are completed
    - Data Storage Layer: The section of the assignment referring to where all data is recorded
    - Use Case: A broad level diagram of the project showing a basic overview
    - Class diagram: It is a type of static structure diagram that describes the structure of a system by showing the system’s cases, their attributes, and the relationships between the classes
    - Interface: Something used to communicate across different mediums
    - Unique Key: Used to differentiate entries in a database

## Sequence Diagram

In the sequence diagram for the food delivery app project, the user first interacts with the app by browsing available restaurants and menus. Upon selecting items and placing an order, the app sends a request to the server. The server processes the order, updating the database accordingly. Simultaneously, the app initiates a payment request to the payment gateway. Once payment is confirmed, the app generates an order confirmation and sends it to the user. The server updates the order status and notifies the respective restaurant. The restaurant prepares the order and updates the server upon completion. The app then retrieves the updated status and notifies the user of the order's progress.

