Fayoum University
Faculty of Science
Mathematics department
Mathematics and Computer Science

City Mall

(Application&&Security System)

A graduation project document submitted to the Dep. of Computer Science as partial fulfillment for the Requirement for the Degree of Bachelor in Computer Science

By

Ahmed Hamada Makhouf

Amr Khaled

Mohammed Ashraf

Mustafa Hamdy

Supervised by

Dr. Mohamed Abd-Elbaky

6/2023

Acknowledgement

We would like to extend my greatest appreciation and deepest gratitude to Dr. Mohamed Abd-Elbaky for his exceptional guidance, unwavering support, and valuable insights throughout the entire duration of our thesis. His dedication to academic excellence have greatly influenced our research journey. We are truly grateful for his patience, encouragement, and commitment to our academic and personal growth. His expert guidance, constructive feedback, and insightful suggestions have immensely shaped the direction and quality of our work. We have been fortunate to benefit from his mentorship, which has not only enriched our understanding of the subject matter but has also inspired us to strive for excellence in all aspects of our academic pursuit. In addition, we would like to express our sincere appreciation to our Computer Science Department coordinator prof. Ahmed Shafey for his guidance, continuous encouragement and support during the courses. At last, we would like to thank all the people who helped, supported and encouraged us to successfully finish the graduation project.

Abstract

Our system aims to utilize deep learning technologies to develop a comprehensive security system for institutions. It will be capable of detecting any anomalies within the mall, such as fires or missing visitors. Additionally, our system will leverage the power of computer vision to calculate the cost of parking garages in the mall. Our focus institution for this project will be the mall, and we will be developing an end-to-end system for it, from the backend to the frontend. Additionally, using data analysis to improve the system's features and enhance visitors' experience, ultimately leading to greater visitor satisfaction.

Table of content

1- Introduction

Motivation

In today's rapidly advancing world, safety and security have become a top priority for public spaces. However, the outdated security systems in many institutions require more employees to achieve high accuracy, which translates to higher costs. Moreover, human error can be costly for institutions, leading to potential losses and damages. Therefore, there is a pressing need for a more advanced and efficient security system that can address these issues. One of the most important problems in any public space is the possibility of a child getting lost. To address this concern, our security system will utilize advanced technologies such as facial recognition and object tracking to detect any missing individuals and quickly alert the security team. In addition, our system will be equipped with features such as real-time notifications and geo-fencing to track and locate lost children and reunite them with their guardians. By utilizing these technologies, we can enhance the safety and security of public spaces, reduce the cost of security personnel, and minimize human error, ultimately improving the visitor experience.

Aim

To address these challenges, our goal is to develop a comprehensive security system for institutions, with a specific focus on a mall, utilizing deep learning technologies. Our system will leverage computer vision to track anomalies, such as fires, in real-time, reducing the number of workers needed in the security department and resulting in cost savings. Furthermore, the system will work 24/7 without breaks and with the same level of concentration, making it more accurate than humans. In addition to security, we plan to utilize the technology in other aspects, such as calculating parking costs in the garage, which will further reduce the cost of workers in the mall.

To achieve these goals, we will create an end-to-end system that includes a backend to handle data, connect the entire system, including the AI component, with users, and a front end to provide an attractive user interface allowing users to interact with the application and its services. The backend will be responsible for managing the data flow between different components of the system, as well as handling the communication with external systems. The AI component will use deep learning algorithms to process the data gathered from the security cameras and detect any anomalies. Finally, the front

end will provide an intuitive and user-friendly interface that will allow users to interact with the system and access its features.
In conclusion, our aim is to create a state-of-the-art security system for institutions that can ensure the safety and security of public spaces while reducing costs and minimizing human error

Outline

The thesis is structured as follows:

- Chapter 2 System Development Cycle and architecture
- Chapter 3 AI and Deep Learning Part
- Chapter 4 Backend and APIs Part
- Chapter 5 Frontend and Android Part
- Chapter 6 Data analysis and insights Part

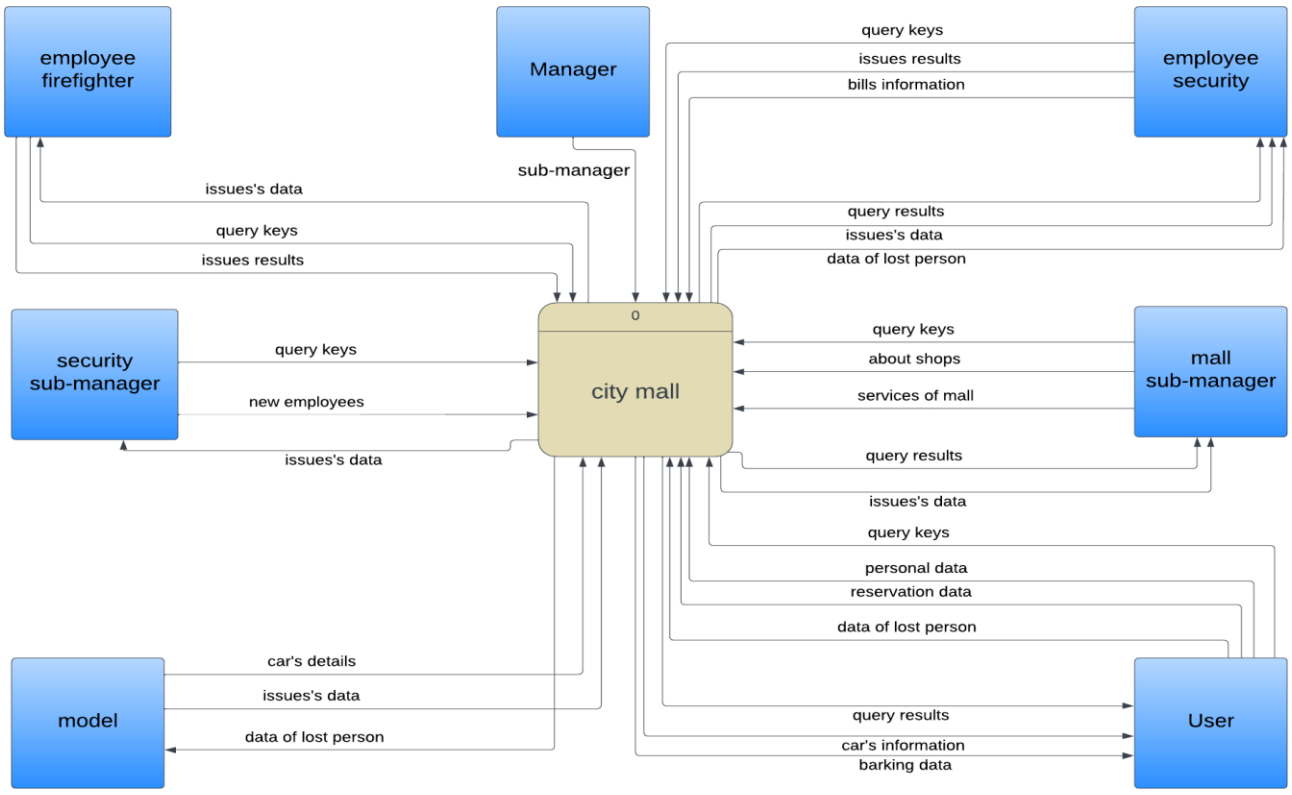
2- System Development Cycle and architecture

2.1 -System Development Cycle

In the beginning stages of the system lifecycle development, one of the key steps is designing the system. This involves creating diagrams to visualize the data flow and processes of each component in the system. By having a clear understanding of the system's architecture and design, we can ensure that all stakeholders are on the same page and that the system is developed to meet their needs and expectations.

2.1.1 - Context Diagram

The first step in designing the system is creating a context diagram. The Level 0 or context diagram represents the high-level view of our system, showcasing the major external entities and the flow of data between them. It depicts the interactions between the mall, the AI models and entities. The Level 0helps us understand the overall structure of the system and its key components.

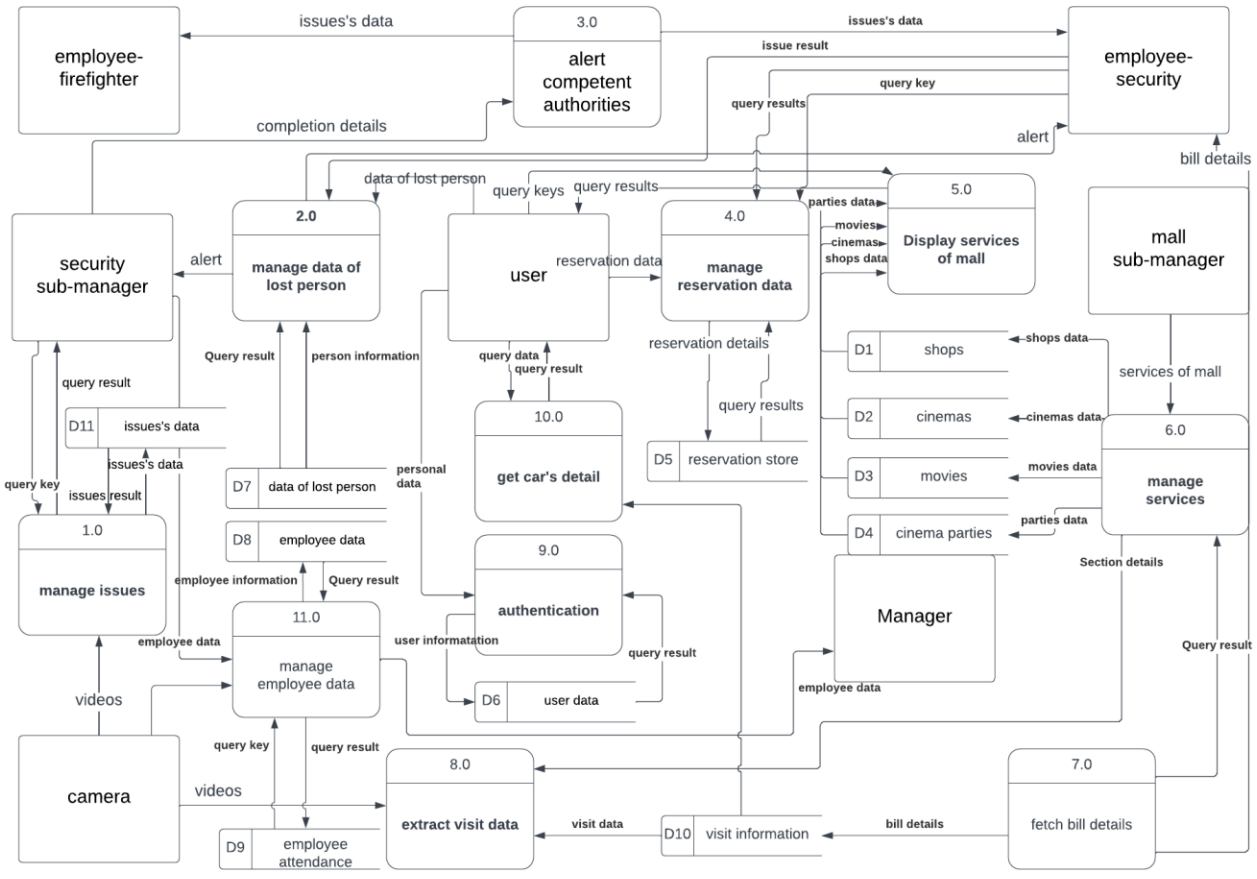


Description of Context Diagram:

At the center of the diagram is the mall, which serves as the main entity interacting with different processes. The mall provides input data to the AI models, including surveillance footage, to detect unfamiliar behaviors such as fire incidents. The AI models analyze the data and generate alerts when anomalies are detected, which are then sent to the backend system for further processing.

The backend system plays a crucial role in receiving and handling alerts from the AI models. It processes the alerts, triggers appropriate actions or notifications, and maintains a log of the detected incidents. Additionally, the backend system integrates with the data analysis module to perform in-depth analysis of the collected data, extracting valuable insights for decision-making and system optimization.

The Android frontend serves as an interface for authorized personnel to interact with the system. It allows employees to check in and check out, view attendance records, and access car tracking information. The frontend communicates with the backend system to retrieve and display relevant data, providing a user-friendly experience for managing security-related aspects within the mall.



2.1.2 - Level 1 Data Flow Diagram

Description of DFD Diagrams:

Data Flow Diagrams (DFDs) are graphical representations that illustrate the flow of data within a system. In our project, we have developed DFD diagrams to visualize the various processes and interactions involved in our integrated security and surveillance system. These diagrams provide a clear overview of how data moves through different components, helping us analyze and optimize the system's functionality.

The Level 1 DFDs dive deeper into the system, breaking it down into more detailed processes and data flows. We have created Level 1 DFD to illustrate functionalities of our system, including unfamiliar behavior detection, employee attendance management, and car tracking. These diagrams demonstrate how data moves within each component and how it is processed or transformed to achieve the desired functionality.

Process examples:

a. Unfamiliar Behavior Detection (process 4 and 2):
This Level 1 DFD focuses on the AI model and its interaction with the backend system. It showcases how surveillance data is collected, processed, and analyzed by the AI model to identify unfamiliar behaviors such as fire incidents. The diagram also highlights how the backend system receives and responds to alerts generated by the AI model, triggering appropriate actions or notifications.

b. Employee data Management (process 11):
This Level 1 DFD illustrates the flow of data related to employee attendance. It showcases how the backend system captures attendance data, integrates it with the employee database, and performs necessary validation and processing.

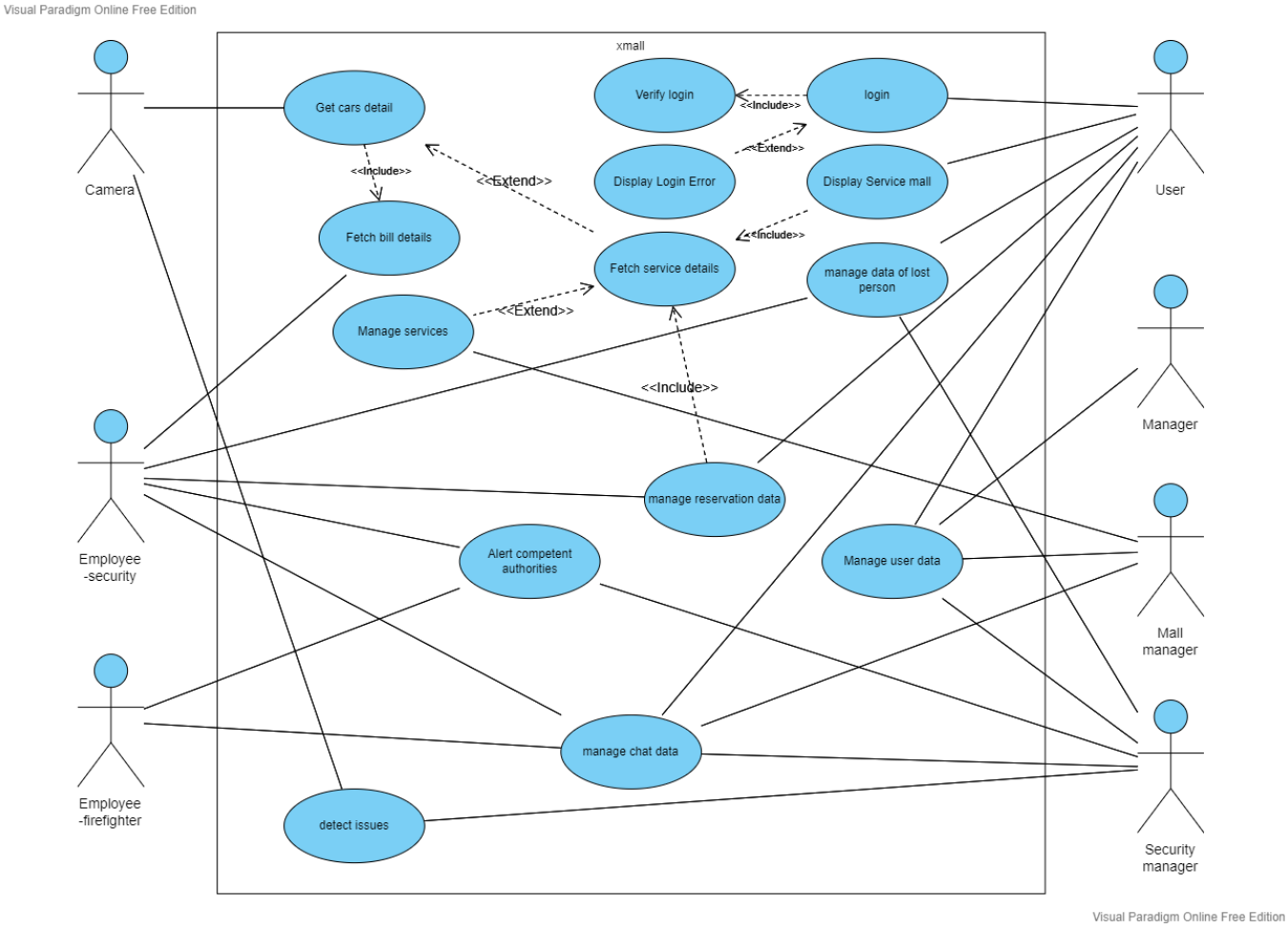
c. Car Tracking(process 8):
The Level 1 DFD for car tracking focuses on the interaction between the backend system, the license plate recognition module, and the database. It demonstrates how the system captures license plate data, processes it through the recognition module, and stores the information in the database. Additionally, it showcases how authorized personnel can access car tracking data through the Android frontend.

These DFD diagrams provide a visual representation of the data flow and interactions within our system, enabling us to identify potential bottlenecks, optimize processes, and

ensure seamless communication between the various components. By utilizing DFDs, we gain a comprehensive understanding of how our integrated security and surveillance system functions, facilitating effective design, development, and maintenance of the overall solution.

2.1.3a – Use Case Diagram

The Use Case Diagram provides a high-level overview of the functionality and interactions of our academic graduation project, an integrated security and surveillance system for a mall.



Description of Use Case Diagrams:

This diagram showcases the different actors or entities that interact with the system and the specific use cases or functionalities they are involved in.

At the center of the Use Case Diagram is the system itself, represented by a box or boundary. Surrounding the system are the actors, which can include various individuals, such as mall staff, security personnel, and visitors, who interact with the system to perform specific tasks or access certain functionalities.

The diagram depicts the main use cases or functionalities offered by the system. These use cases represent the actions or operations that the actors can perform within the system. Examples of use cases in our project.

Arrows between actors and use cases indicate the interactions or associations between them. This signifies which actors are involved in specific use cases and the roles they play in the system. For example, mall staff may have access to administrative functionalities, while security personnel may have additional capabilities for monitoring and responding to security incidents.

The Use Case Diagram provides a visual representation of the system's functionalities and the actors involved, allowing for a better understanding of how different stakeholders interact with the system. It helps identify the various use cases and actors, ensuring that the system caters to the needs and requirements of different user groups. This diagram serves as a valuable communication tool during the analysis, design, and development phases of the project, facilitating collaboration and alignment among project stakeholders.

2.1.3b – Use Case documentation

Use Case Name	Log in
Actors	User
Description	This use case describes the Sign up or login of user using Mobile application or Website.
Preconditions	User using the application without reserve any services from the system
Post conditions	Successfully authorized and can use system service
Main success scenarios	1: The User sign in from mobile app or website (phone-Password). 2: The System validates user entered data and stores it in the database.
Alternative paths	AP: user doesn't have an account: 1. User create an account 2. System confirms account creation

Use Case Name	Display Service mall
Actors	User
Description	This use case describes the showing of mall services
Preconditions	User want to uses or see the services of the mall
Post conditions	The user select or display one of the services and use it
Main success scenarios	1:The system successfully fetched the services of mall 2:The user selected and used one of the services
Alternative paths	AP: failed to fetch data: The system return the user to main home and show "something went wrong message"

Use Case Name	Manage data of lost person
Actors	User , Security manager , Security employee
Description	This use case describes the usage of the lost peoples data or images to give more information about lost people to help finding them
Preconditions	User lost one of his contacts and want security to help him.
Post conditions	Security manager: check the request and confirm it if it was true then the system will alert the security employees with the needed information to help user to find the person
Main success scenarios	1: the user send the data of lost person in the right format without error. 2: The security manager receive the information in the same format. 3:The Security manager confirm the request successfully 4: send the information to the model to search about the person in all cameras 5: The security employee receive more information from the system about the lost person. 6: The user fined the lost person and close the case.
Alternative paths	AP: send corrupted data: Security manager can't accept the case AP: can't find the person: Manager will alert competent authorities

Use Case Name	Manage user data
Actors	User, Manager , Mall manager, Security manager
Description	This use case describes the edit or delete or add new information of user or sub-manager or employee
Preconditions	User: need to sign up on the system. Manager: need to add/edit new sub manager Mall- manager: need to add/edit employee Security - manager: need to add/edit employee
Post conditions	Successfully conformed the wanted edits on data
Main success scenarios	1: The User sign up from mobile app or website (phone-Password). 2: The managers successfully edit on the data.
Alternative paths	AP: user entered invalid data: Alter the user to the right way to enter data AP: user already have account: Alter the user to sign in instead of signup

Use Case Name	Get cars detail
Actors	Employee – security , camera

Description	This use case describes when the car is checked in and checked out and saved the details of the car.
Preconditions	There are car entering or exiting form garage.
Post conditions	detected and send the details of the car
Main success scenarios	1: The User try to enter or exit the garage 2: The model successfully sent the data
Alternative paths	AP: model cannot recognize or detected the details of the car: Security employee will check the details of the car manually AP: the model send repeated or corrupted data: The security employee will entered the details of the car manually

Use Case Name	Fetch bill details
Actors	Employee – security
Description	This use case used to check the garage bill of the exited car
Preconditions	There are car exiting form garage and want to pay the garage bill
Post conditions	Return bill information and remove all details of the car from the system.
Main success scenarios	1: The User try to exit the garage 2: The system successfully returned the data and bill of the car
Alternative paths	AP: The system cannot find data of the car : Security employee with search in all images this section AP: model can't detect the information of the car: The security employee will enter and check the data of the car manually

Use Case Name	detect issues
Actors	camera – security manager
Description	This use case used to check if there is any problem in the mall
Preconditions	there is a problem with the security mall
Post conditions	the problem should be detected by the model
Main success scenarios	1: there is a problem with the security mall 2: The model successfully detect the problem 3: the system successfully alert competent authorities
Alternative paths	AP: The model detect a problem in somewhere and there is no problem there : Security manager should see the video and cancel the issue AP: model can't detect the problem: any employee or user in the mall can send appeal to the system and detail of the issue

Use Case Name	alert competent authorities
Actors	security employee–firefighter employee - security manager
Description	This use case should alert competent authorities when there is a problem
Preconditions	the model detected a problem and send it to security manager
Post conditions	the security manager confirms it and send to the alert competent authorities to solve the problem
Main success scenarios	1: the model detected a problem and send it to security manager 2: the security manager confirms that there is a real problem 3: the system successfully alert competent authorities who is responsible to solve this kind of the problems
Alternative paths	AP: the system doesn't send any alert to any competent authority : Security manager should send alert manual to competent authority AP: the system send alert to wrong competent authority : security manager should cancel the request and send alert manual to competent authority

Use Case Name	manage chat data
Actors	security employee–firefighter employee - security manager-user -mall manager
Description	This use case should manage chatting data
Preconditions	send or receive messages
Post conditions	send or receive messages
Main success scenarios	1:the security or firefighter employee send ok sign or need help to the security manager 2: the user send a request of question to the mall manager 3: the security manager send problem details to the competent authority manually
Alternative paths	AP: the system can’t send or receive messages: all members of system will communicate manually

Use Case Name	manage reservation data
Actors	security employee–user
Description	This use case should manage all reservation data between user and the system for any service
Preconditions	the user need to book/cancel any service in mall
Post conditions	the system should book and send detail of reservation to the user
Main success scenarios	1: the user send the detail of reservation to the system 2: the system should check of the detail of reservation and confirm it
Alternative paths	AP: the system doesn't book tickets : security employee should check the reservation details which sent to user AP: the system doesn't sent reservation data to user : user should send request to the system to check if the reservation was confirmed or not

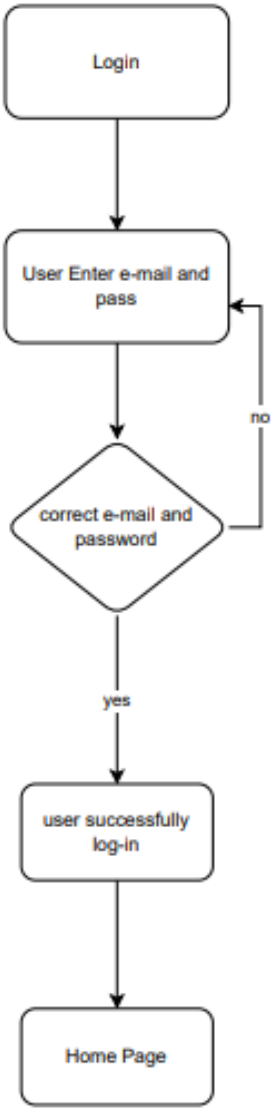
Use Case Name	manage services
Actors	mall manager
Description	This use case used to manage all services in the mall
Preconditions	the mall manager need to add/remove/edit service
Post conditions	the service was edited successfully
Main success scenarios	1: the mall manager add the details of service successfully
Alternative paths	AP: the system doesn't edit all details of the service : mall manager should check and edit the service and add all details which wasn't added

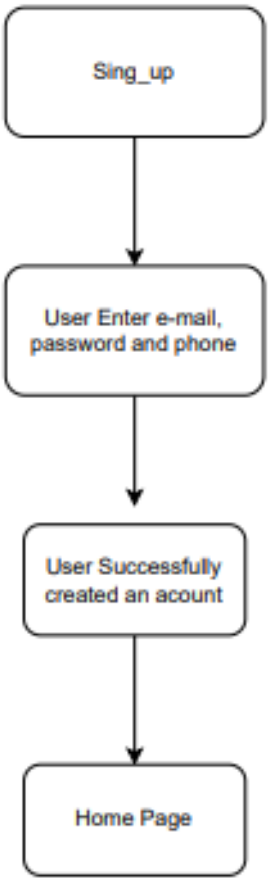
2.1.4 – Activity diagrams

Activity diagrams are powerful tools for visualizing the flow of activities or processes within a system.

Description of Activity Diagrams for login and signup Flow:

In our project, we have created activity diagrams to illustrate the login and signup flows, providing a clear understanding of the steps involved and the decision points within these processes.



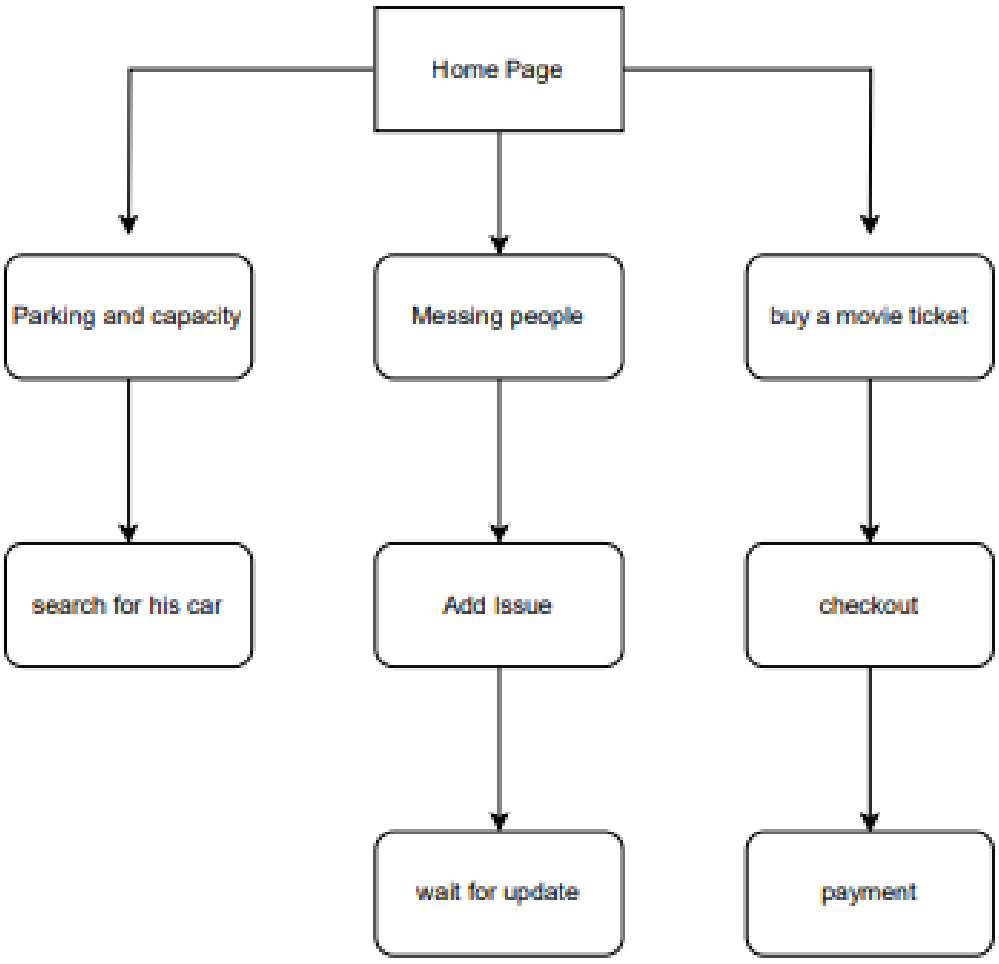


The activity diagram for the login flow showcases the sequence of activities performed when a user attempts to log into the system. It begins with the user providing their credentials, such as username and password. The system then verifies the provided information and checks for authentication. If the authentication is successful, the user is granted access to the system and directed to the home page. If the authentication fails, an error message is displayed, and the user is prompted to retry or seek assistance.

Similarly, the activity diagram for the signup flow depicts the series of activities involved when a user registers for a new account. It starts with the user initiating the signup process and providing the required information, such as their name, email address, and password. The system validates the entered information, checks for any duplicates or errors, and creates a new user account if all requirements are met. Upon successful registration, the user is notified, and they can proceed to log into the system using their

newly created credentials.

Description of Activity Diagrams for Lost People, search for car and payment Flow:



The activity diagram for checkout illustrates the steps involved in the checkout process within the system. It starts with the user initiating the checkout by selecting the desired items or services. The system then prompts the user to provide relevant information such as payment details, delivery address, and any additional preferences. Once the user confirms the checkout, the system processes the transaction, updates inventory if necessary, and generates an order confirmation. The user is then notified about the successful completion of the checkout process.
For car search functionality, the activity diagram showcases the process of searching for a specific car within the system. The user begins by accessing the "Car Search" feature and specifying the search criteria, such as the car's make, model, or license plate

number. The system then retrieves relevant information from the car database and displays the search results to the user. The user can further refine the search or select a specific car from the results for more details.
In the case of the lost people search functionality, the activity diagram outlines the steps involved in searching for lost individuals within the system. The user accesses the "Lost People Search" feature and provides relevant information, such as the person's name, age, or last known location. The system processes the search query, retrieves matching records from the database, and presents the results to the user. The user can then view detailed information about the lost individual and take necessary actions, such as reporting a sighting or contacting the appropriate authorities.

Backend Part

A. Introduction

Welcome to the Backend Documentation for city mall. This documentation aims to provide you with a comprehensive understanding of the backend system and its functionalities. It is intended for developers and technical stakeholders involved in working with the backend infrastructure and APIs.

The backend system serves as the backbone of our application, handling data storage, processing, and the communication between the frontend and various external services. It plays a crucial role in ensuring the smooth operation and functionality of the overall

application.

Key Features:
- Robust API layer for interacting with the frontend and external services.
- Secure authentication and authorization mechanisms to protect user data.
- Efficient data storage and retrieval through the database layer.
- Integration with third-party services and APIs to enhance application capabilities.
- Scalable architecture to handle increasing traffic and workload.
- Automated deployment and continuous integration processes for seamless updates.
- Logging and monitoring systems to track system health and performance.

Target Audience:
This documentation is designed for supervisor doctor, judgmental commission, developers and technical stakeholders who are responsible for maintaining, extending, or integrating with the backend system. It assumes a basic understanding of web development concepts, databases, and APIs. Familiarity with programming languages such as JavaScript or Java would be beneficial.

Throughout this documentation, we will delve into the architecture, API references, database structure, security measures, deployment processes, and other essential aspects of the backend system. By following this guide, you will gain the knowledge and insights necessary to effectively work with the backend infrastructure and contribute to the success of our application.

Next, we will explore the system architecture in detail, providing an overview of how the backend system is structured and the technologies used.

B. Used Technology

This section provides an overview of the technology stack used in the backend system of City Mall. Understanding the technologies involved will give you a better understanding of the system's architecture and development environment.

Backend Technology Stack:

1. Node.js: Node.js is a JavaScript runtime built on the V8 JavaScript engine. It allows running JavaScript code outside of a web browser and is commonly used for building server-side applications.

2. Express.js: Express.js is a fast and minimalist web application framework for Node.js. It provides a robust set of features for building web APIs, handling routes, and managing middleware.

3. Database System: PostgreSQL. The backend system utilizes a database system for persistent data storage and retrieval.

4. Database System: PostgreSQL is a powerful and feature-rich open-source relational database management system (RDBMS). It provides a reliable and scalable solution for storing and retrieving structured data.

5. Sequelize: Sequelize is a popular Object-Relational Mapping (ORM) library for Node.js. It simplifies database interactions by providing an intuitive API for querying and manipulating data in PostgreSQL using JavaScript objects and models.

6. Authentication and Authorization: JSON Web Tokens (JWT) is an industry-standard for representing claims securely between two parties. It is commonly used for authentication and authorization in web applications. JWTs consist of encoded JSON objects that are digitally signed to ensure integrity.
This ensures secure access to protected routes and resources within the backend system.

7. Other Libraries and Packages:

- **Body-parser**: body-parser simplifies parsing request bodies, enabling easy access to JSON or URL-encoded data.
- **cors**: cors handles Cross-Origin Resource Sharing, ensuring proper security and access control for requests originating from different origins.
- **bcrypt**: bcrypt provides a secure way to hash and compare passwords, safeguarding user credentials.
- **multer**: multer simplifies handling file uploads, allowing users to submit files

Key Features of Node.js with express:

1. Asynchronous and Event-Driven: Node.js uses a non-blocking, event-driven

architecture, making it highly efficient and scalable. It can handle a large number of concurrent connections without blocking the execution of other tasks.

2. Single-Threaded and Non-Blocking I/O: Node.js operates on a single thread, using non-blocking I/O operations. This allows it to handle multiple requests simultaneously, resulting in improved performance and responsiveness.

3. JavaScript Everywhere: Node.js enables the use of JavaScript on both the client-side and server-side, allowing developers to write server-side and client-side code using the same programming language. This simplifies development and promotes code reusability.

4. NPM Ecosystem: Node.js has a vast and vibrant ecosystem supported by npm (Node Package Manager). It offers a wide range of open-source libraries and modules that can be easily integrated into Node.js applications, accelerating development and enhancing functionality.

5. Scalability: Node.js is highly scalable due to its event-driven, non-blocking I/O model. It can efficiently handle thousands of concurrent connections, making it suitable for building real-time applications and microservices architectures.

Key Features of Express.js:

1. Minimal and Lightweight: Express.js is a minimalistic web application framework for Node.js. It provides essential features for building web applications without imposing unnecessary overhead, making it lightweight and easy to learn.

2. Routing: Express.js simplifies the handling of HTTP requests through its routing mechanism. Developers can define multiple routes and associate them with specific HTTP methods, making it easier to handle different types of requests and execute corresponding actions.

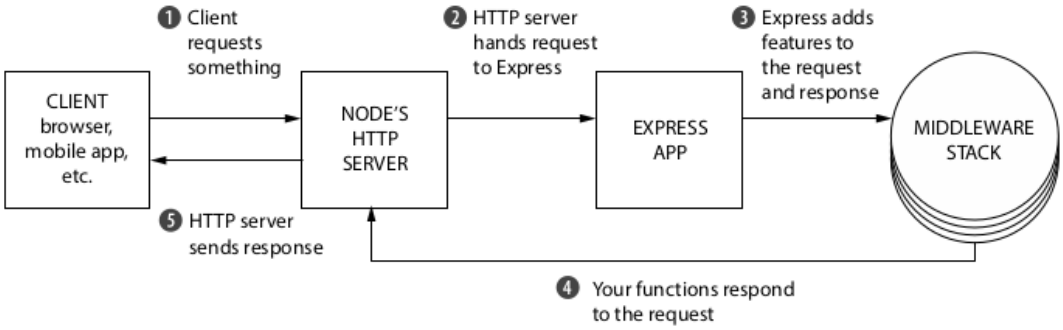
3. Middleware Support: Express.js offers robust middleware support, allowing

developers to define custom middleware functions that can intercept and modify incoming requests or outgoing responses. This enables functionality such as authentication, logging, error handling, and request processing.

4. Extensibility: Express.js can be extended using middleware and third-party packages from the npm ecosystem. This allows developers to add additional functionality to their applications easily and leverage existing community-driven solutions.

5. RESTful API Development: Express.js provides a solid foundation for building RESTful APIs. It allows developers to define routes, handle HTTP methods, and perform CRUD operations on resources, making it a popular choice for developing API-driven applications.

These key features make Node.js and Express.js powerful tools for developing scalable, efficient, and feature-rich web applications. Their combination provides a robust environment for server-side JavaScript development, enabling developers to build high-performance applications and APIs.



Key Features of Sequelize with PostgreSQL:

- Schema definition: Sequelize allows defining database tables, columns, and relationships using JavaScript models, making it easy to work with PostgreSQL schemas.
- Data validation: Sequelize provides built-in validation mechanisms to ensure data integrity and consistency before saving it to the PostgreSQL database.
- Query building: Sequelize offers a fluent API for constructing complex SQL queries, including joins, conditions, and aggregations, without the need to write raw SQL statements.
- Migrations: Sequelize simplifies database schema management through the use of migrations. Migrations enable seamless updates to the database schema while

preserving existing data.

- Associations and relationships: Sequelize supports defining associations between models, such as one-to-one, one-to-many, and many-to-many relationships, making it convenient to work with related data.

C. Project Architecture

The backend application for the City Mall security project follows the Model-View-Controller (MVC) architectural pattern, which provides a logical and modular separation of concerns. This pattern divides the application into three interconnected components: the Model, View, and Controller, each with its specific responsibilities.

1. Model:
In the context of the City Mall security project, the Model component represents the data layer of the application. It encapsulates the logic for data storage, retrieval, and manipulation related to user data, visits, cars, shops, cinemas, offers, movies, issues, and employee attendance. The Model handles interactions with the PostgreSQL database, ensuring the integrity and consistency of the stored data. It incorporates data access code, query execution, and data validation specific to the entities within the City Mall environment.

2. View:
The View component focuses on presenting the application's user interface to the client. In the City Mall security system, the View involves generating and constructing JSON responses, or rendering data in various formats for display. It formats the data received from the Controller in a way that is suitable for presentation to security personnel, administrators, and other system users. The View is decoupled from the underlying data and focuses solely on displaying information effectively.

3. Controller:
The Controller acts as the intermediary between the Model and View components. It receives and handles incoming requests related to managing user data, visits, cars, shops, cinemas, offers, movies, issues, and employee attendance. The Controller processes the data, interacts with the Model to fetch or modify data from the database, and determines the appropriate response to send back to the client. It performs business logic such as authentication, authorization, and data transformation specific to the City Mall security system. The Controller ensures that the correct data is retrieved or updated and prepares it for presentation by the View.

By adopting the MVC structure in the backend application of the City Mall security

project, you achieve several benefits:

1. Separation of Concerns: The MVC pattern separates the responsibilities of data manipulation, user interface, and control flow. This separation allows each component to focus on its specific task, making the codebase more organized and easier to understand.

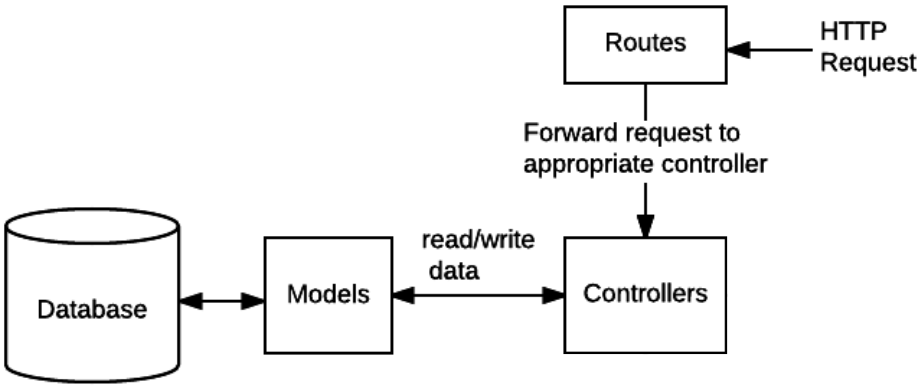
2. Code Reusability: The Models and Views developed for the City Mall security system can be reused across different parts of the application or even in future projects. The decoupling of the View from the data layer enables flexibility in changing the presentation layer without affecting the underlying data structures.

3. Maintainability: The modular structure of MVC makes it easier to maintain and update the City Mall security system. Changes made to one component have minimal impact on the others, promoting a more manageable and scalable codebase.

4. Collaboration: The MVC pattern facilitates collaboration among multiple developers working on the City Mall security system. Each developer can work on a specific component without interfering with others, resulting in parallel development and faster progress.

5. Testability: The separation of concerns in MVC makes it easier to write unit tests for each component individually. Testing the Models, Views, and Controllers independently helps ensure the correctness and reliability of the City Mall security system.

In conclusion, the adoption of the MVC pattern in the backend application of the City Mall security project provides a robust and scalable architecture. It separates the concerns of data manipulation, user interface, and control flow, leading to better organization, code reusability, maintainability, and collaboration among developers. By leveraging the advantages of the MVC structure, the backend system ensures efficient management of security-related aspects within the City Mall environment and allows for future enhancements and scalability.



D. Getting Started

This section will guide you through the initial steps required to set up and start working with the backend system of city mall. By following these steps, you will be able to quickly get our development environment up and running.

Prerequisites:

Before getting started, make sure you have the following prerequisites in place:

1. Operating System: Windows, macOS, Linux

2. Node.js: Install the latest stable version of Node.js. You can download it from the official Node.js website or use a package manager like `npm` or `yarn`.

3. Database: PostgreSQL. Install and configure the database on our local machine or have access to a remote database instance.

4. Text Editor or IDE: Choose a text editor or an integrated development environment (IDE) that you are comfortable with. Some popular options include Visual Studio Code
Installation:

To get started with the backend system, follow these steps:

1. Clone the Repository: Clone the backend repository from City Mall (Backend) using Git or download the source code as a ZIP file and extract it to our desired location.

2. Install Dependencies: Open a terminal or command prompt, navigate to the project's root directory, and run the following command to install the required dependencies:

```
npm install
```

3. Start the Backend Server: After the dependencies are installed and the configuration is in place, start the backend server by running the following command:

```
npm start
```

This will start the backend server on the specified port, and you should see a message indicating that the server is running successfully.

E. API Documentation:

Once the backend server is up and running, you can explore the API documentation to understand the available endpoints, request/response formats, authentication methods, and any additional details specific to the backend system. The API documentation can be accessed at City Mall API.

This section provides detailed documentation for the APIs developed for the graduation project in the previous link. The APIs are designed to support various functionalities of the project, including user management, visits, cars, shops, cinemas, offers, movies, issues, and employee management.

Users API

- Get All Users: Retrieves a list of all users in the system.
- Create a User: Creates a new user in the system.

- Get User by ID: Retrieves a user based on our ID.
- Update User by ID: Updates the details of a specific user.
- Delete User by ID: Deletes a user from the system.

Visits API

- Get All Visits: Retrieves a list of all visits recorded in the system.
- Add Visit: Records a new visit in the system.
- Get Visit by ID: Retrieves details of a specific visit.
- Update Visit by ID: Updates the details of a specific visit.
- Delete Visit by ID: Deletes a visit from the system.

Cars API

- Get All Cars: Retrieves a list of all cars registered in the system.
- Create a Car: Registers a new car in the system.
- Get Car by Plate Number: Retrieves details of a car based on its plate number.
- Update Car by Plate Number: Updates the details of a specific car.
- Delete Car by Plate Number: Deletes a car from the system.

Shops API

- Get All Shops: Retrieves a list of all shops in the system.
- Create a Shop: Registers a new shop in the system.
- Get Shop by ID: Retrieves details of a shop based on its ID.
- Update Shop by ID: Updates the details of a specific shop.
- Delete Shop by ID: Deletes a shop from the system.

Cinemas API

- Get All Cinemas: Retrieves a list of all cinemas in the system.
- Create a Cinema: Registers a new cinema in the system.
- Get Cinema by ID: Retrieves details of a cinema based on its ID.
- Update Cinema by ID: Updates the details of a specific cinema.
- Delete Cinema by ID: Deletes a cinema from the system.

Offers API

- Get All Offers: Retrieves a list of all offers in the system.
- Create an Offer: Creates a new offer in the system.
- Get Offer by ID: Retrieves details of an offer based on its ID.

- Update Offer by ID: Updates the details of a specific offer.
- Delete Offer by ID: Deletes an offer from the system.

Movies API

- Get All Movies: Retrieves a list of all movies in the system.
- Create a Movie: Adds a new movie to the system.
- Get Movie by ID: Retrieves details of a movie based on its ID.
- Update Movie by ID: Updates the details of a specific movie.
- Delete Movie by ID: Deletes a movie from the system.

Issues API

- Get All Issues: Retrieves a list of all issues reported in the system.
- Create an Issue: Reports a new issue in the system.
- Get Issue by ID: Retrieves details of a specific issue.
- Update Issue by ID: Updates the details of a specific issue.
- Delete Issue by ID: Deletes an issue from the system.

Employees API

- Employee Login: Authenticates an employee with the system.
- Create Employee Attendance: Records the attendance of an employee.
- Get All Attendance: Retrieves a list of attendance records for all employees.
- Get Employee Attendance: Retrieves attendance details of a specific employee.
- Get Employee by ID: Retrieves details of an employee based on our ID.

F. Database Design

The backend of the security system for the mall utilizes a PostgreSQL database managed using Sequelize. The following tables have been designed to support the functionalities of the system:

5.1. Car

The `cars` table stores information about the cars entering and leaving the mall. It includes columns such as `plateNum` (primary key) and `color`.

5.2. Cinema

The `cinemas` table stores details about the cinemas within the mall. It includes columns such as `id` (primary key), `name`, `location`, `openAt`, `closeAt`, `phone`, and `imageUrl`.

5.3. EmployeeAttendance

The `employeeAttendance` table tracks the attendance of the mall employees. It includes columns such as `id` (primary key), `loggedIn`, and `loggedOut`.

5.4. Employee

The `employees` table stores information about the mall employees. It includes columns such as `id` (primary key), `name`, `email`, `password`, `imageUrl`, and `role`.

5.5. Issue

The `issues` table records the issues reported within the mall. It includes columns such as `id` (primary key), `type`, `details`, `imageUrl`, and `state`.

5.6. ModelIssue

The `modelIssues` table stores information about the model issues reported within the mall. It includes columns such as `id` (primary key), `type`, `details`, `imageUrl`, and `state`.

5.7. Movie

The `movies` table stores information about the movies being shown in the mall's cinema. It includes columns such as `id` (primary key), `name`, `duration`, `release`, `description`, `genre`, `time`, `ticketPrice`, and `imageUrl`.

5.8. Offer

The `offers` table stores details about the offers available at various shops. It includes columns such as `id` (primary key), `discount`, `startAt`, and `endAt`.

5.9. Shop

The `shops` table stores information about the shops within the mall. It includes columns such as `id` (primary key), `name`, `location`, `openAt`, `closeAt`, `phone`, `imageUrl`, and `shopType`.

5.10. User

The `users` table stores information about the system users, including customers. It includes columns such as `id` (primary key), `name`, `phone`, `email`, `password`, `passwordConfirm`, `passwordChangedAt`, `passwordResetCode`, and `passwordResetExpire`.

5.11. Visit

The `visits` table tracks the visits of cars to the mall. It includes columns such as `id` (primary key), `timeIn`, `timeOut`, `section`, and `cost`.

5.12. Checkout

The `checkouts` table represents the checkouts made by users for movies. It includes columns such as `id` (primary key), `status`, `ticketPrice`, `ticketNum`, and `cost`.

5.13. CinemaMovie

The `cinema_movie` table represents the relationship between cinemas and movies, indicating which movies are being shown in each cinema. It does not include any additional columns.

5.14. IssueEmployee

The `issue_employee` table represents the relationship between issues and employees, indicating which employees are assigned to each issue. It does not include any additional columns.

5.15. ModelIssueEmployee

The `modelIssue_employee` table represents the relationship between model issues and employees, indicating which employees are assigned to each model issue. It does not include any additional columns.

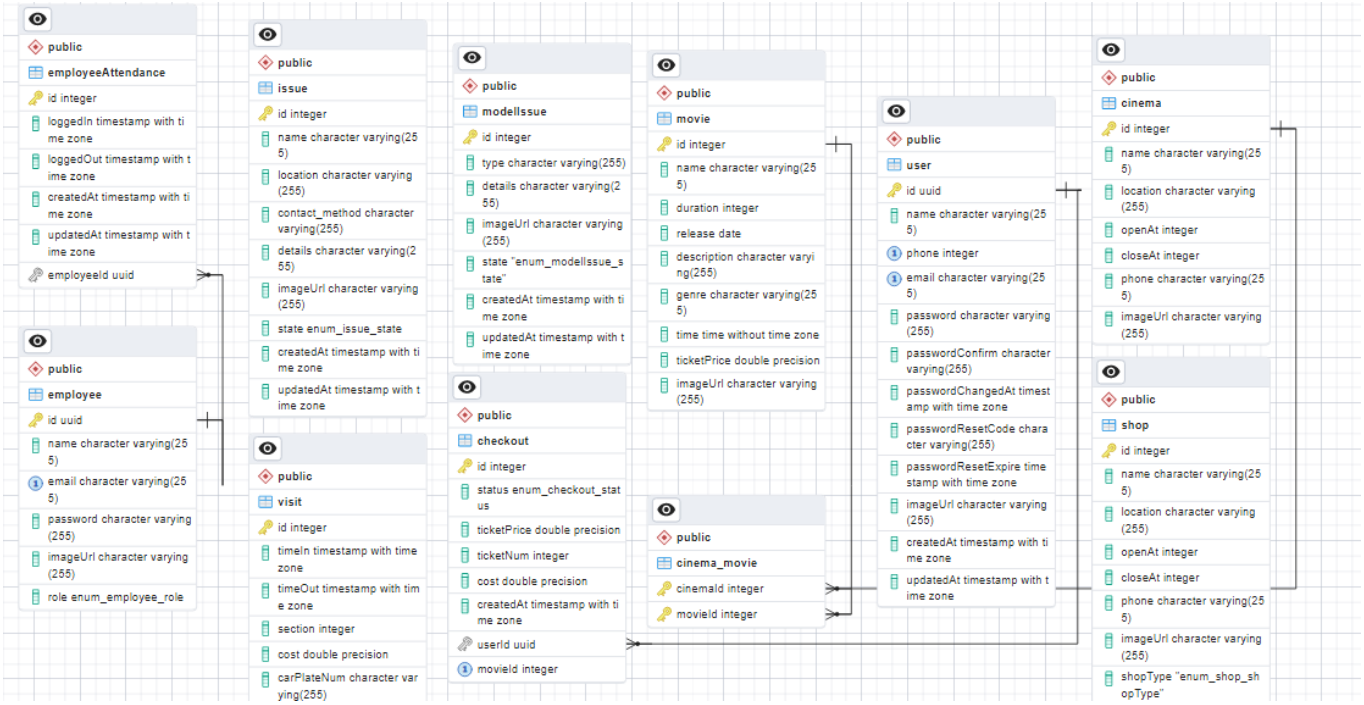
5.16. OfferShop

The `offer_shop` table represents the relationship between offers and shops, indicating which shops have specific offers. It does not include any additional columns.

5.17. UserShop

The `user_shop` table represents the relationship between users and shops, indicating which shops are visited by each user. It does not include any additional columns.

The ERD for the Database



G. Security

Ensuring the security of our project is essential to protect sensitive data, prevent unauthorized access, and mitigate potential security vulnerabilities. In this section, we will discuss the security measures implemented in the backend system, including authentication and authorization, input validation, and security best practices.

1) **Authentication and Authorization**

Authentication is the process of verifying the identity of a user or entity accessing the system. It ensures that only authenticated users can access protected resources. In our graduation project, authentication can be implemented using a combination of user credentials and tokens.

User Credentials

User credentials, such as username (or email) and password, are used to verify the identity of users during the login process. These credentials are securely stored in the `users` table of our database. When a user attempts to log in, the system checks if the provided credentials match the stored values.

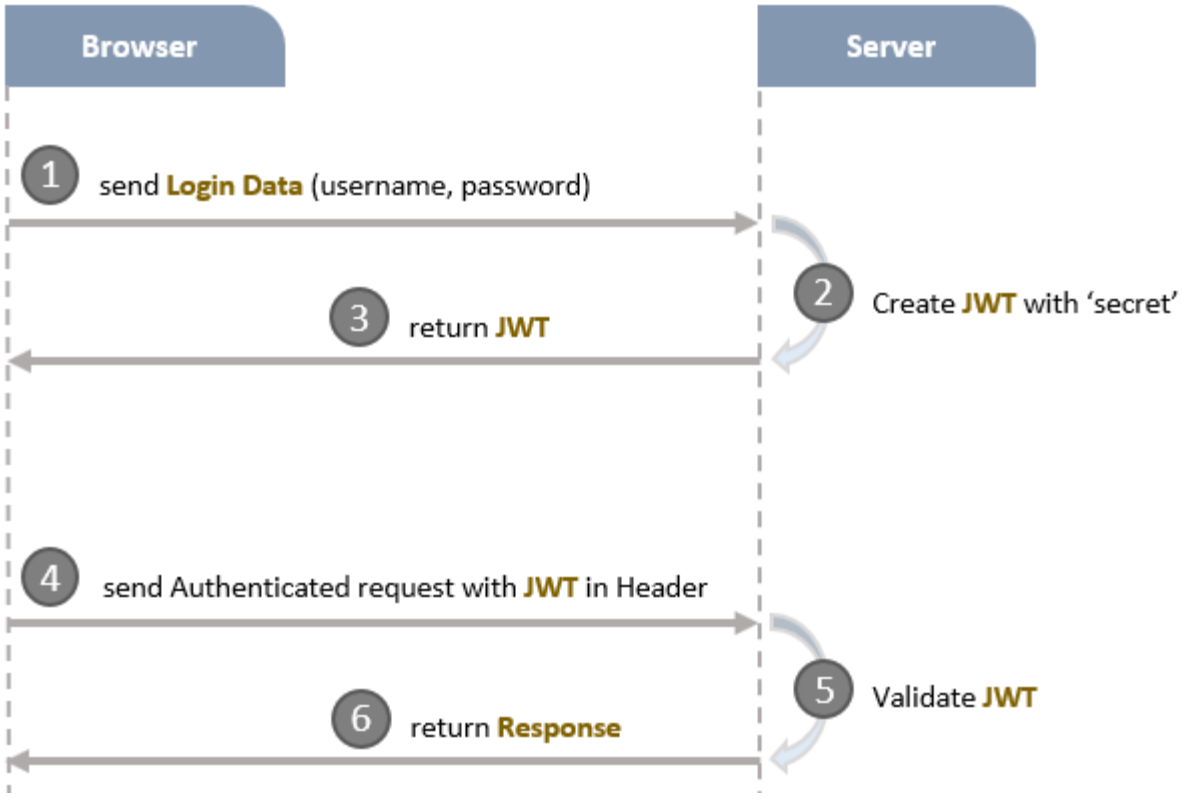
To enhance security, it is recommended to use secure password storage techniques such as hashing and salting. Hashing transforms the user's password into a fixed-length string of characters, making it computationally difficult to reverse-engineer the original password. Salting adds a random value to each password before hashing, further increasing security.

Token-based Authentication

Once a user's credentials are successfully verified, the system generates an authentication token, such as a JSON Web Token (JWT). This token contains information about the user's authentication status and other relevant details. The token is digitally signed by the server to ensure its integrity and authenticity.

The generated token is sent back to the user as part of the authentication response. Subsequently, the user includes this token in the headers (e.g., Authorization header) of our requests to access protected resources within the system.

Flow for Signup & Login with JWT Authentication
The diagram shows flow of User Registration, User Login and Authorization process.



A legal JWT must be added to HTTP x-access-token Header if Client accesses protected resources.

Authorization

Authorization determines what actions an authenticated user is allowed to perform within the system. It ensures that users have the necessary permissions to perform specific operations. In our graduation project, authorization can be based on user roles and our associated permissions.

User Roles and Permissions
User roles define different levels of access and functionality within the system. The `employees` table in our database includes a `role` column, which specifies the role of each employee. Common roles could include 'admin', 'manager', 'employee', or 'customer'.

Each role is associated with a set of permissions that determine what actions a user with that role can perform. For example, an 'admin' role might have full access to all functionalities, while an 'employee' role might have restricted access.

Authorization Service

The system's authorization service receives requests from authenticated users and validates our authentication tokens. It then checks the user's role and associated permissions to determine if the requested action or resource access is allowed.

Based on the user's role, the authorization service consults a set of predefined rules or policies to determine access rights. These rules can be stored in a separate configuration file or database table.

2) Node.js Express Architecture with Authentication & Authorization

The architecture of a Node.js Express application that incorporates authentication and authorization. Node.js and Express are popular choices for building web applications due to our simplicity and scalability. Adding authentication and authorization features enhances the security and control over the application's resources.

Overview of the Architecture

The architecture of a Node.js Express application with authentication and authorization typically involves the following components:

1. **Client:** The client refers to the user interface or frontend of the application, which can be a web browser, mobile app, or any other client application.
2. **Server:** The server is responsible for handling incoming requests from clients, performing authentication and authorization checks, and serving appropriate responses.
3. **Express Framework:** Express is a minimal and flexible web application framework for Node.js. It provides a set of features and utilities to build robust and scalable web applications.
4. **Authentication Middleware:** Authentication middleware is used to verify the identity of the user making the request. It validates user credentials, such as username and password, against a user database or external authentication providers.
5. **Authorization Middleware:** Authorization middleware determines whether the authenticated user has the necessary privileges to access a particular resource or perform a specific action. It checks user roles, permissions, or any custom authorization logic.

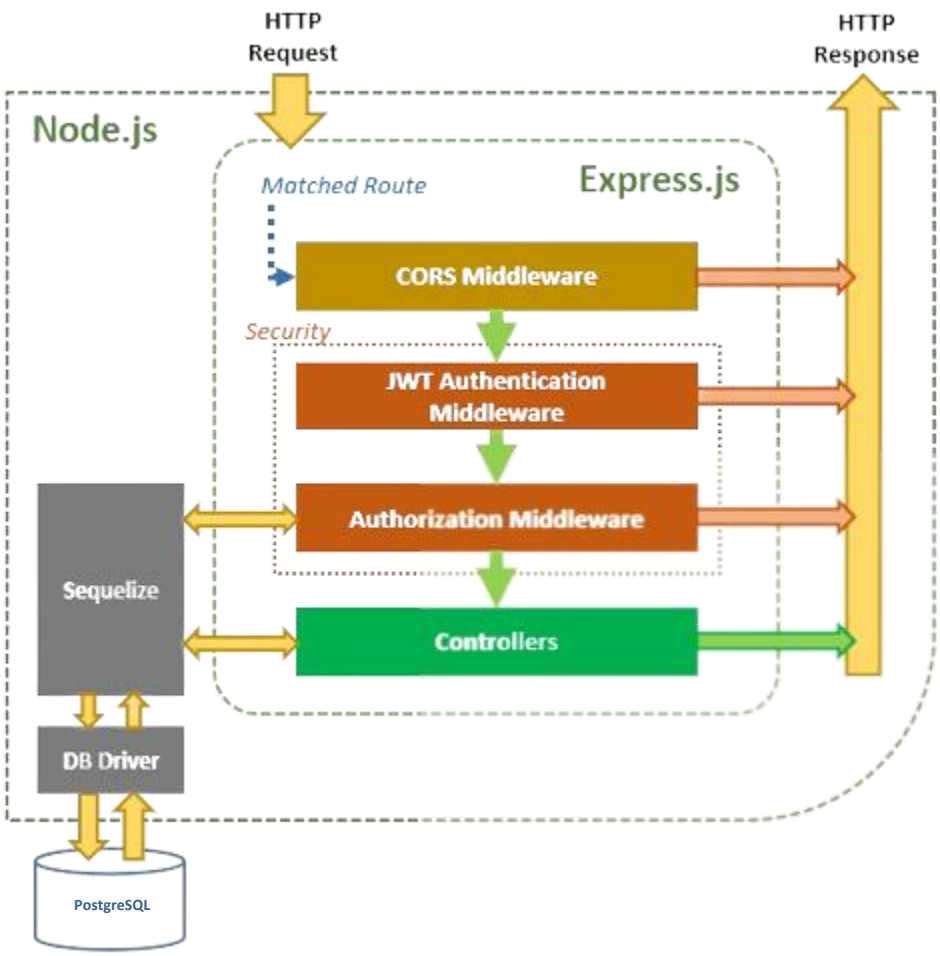
6. **User Database:** A user database stores user information, such as usernames, passwords (hashed), roles, and permissions. Common choices for user databases include relational databases like MySQL or PostgreSQL, NoSQL databases like MongoDB, or even external identity providers like OAuth or LDAP.

Flow of Authentication & Authorization

The flow of authentication and authorization in a Node.js Express application typically follows these steps:

1. **Client Requests Access:** The client initiates a request to the server to access a protected resource or perform a specific action.
2. **Authentication Middleware:** The server's authentication middleware intercepts the request and verifies the user's identity. It checks the provided credentials against the user database or external authentication providers.
3. **Authentication Success:** If the authentication is successful, the server generates a token (such as a JSON Web Token or JWT) and includes it in the response to the client.
4. **Client Includes Token:** The client receives the token from the server and includes it in subsequent requests as an Authorization header or another appropriate mechanism.
5. **Authorization Middleware:** For each subsequent request, the server's authorization middleware intercepts the request, extracts the token, and verifies its validity.
6. **Authorization Check:** The server performs authorization checks based on the user's role, permissions, or any custom authorization logic. It determines whether the user is allowed to access the requested resource or perform the requested action.
7. **Response to Client:** If the user is authorized, the server proceeds with the requested operation and sends the appropriate response back to the client. Otherwise, an error response or access denied message is returned.

Here's a diagram illustrating the flow of authentication and authorization in a Node.js Express application:



The diagram provides a visual representation of the steps involved in authenticating and authorizing users in a Node.js Express application.

3) Input Validation

Input validation is a critical security measure that helps prevent various types of attacks, such as injection attacks and cross-site scripting (XSS). The backend system implements strong input validation techniques to validate and sanitize user input.
Validation: User input is thoroughly validated against predefined rules and constraints to ensure its integrity and adherence to expected formats. Data types, length limits, format patterns, and business-specific validation rules are validated to ensure that the input is valid and secure. Proper validation prevents the system from processing malicious or malformed input that could lead to security vulnerabilities.
Sanitization: Input sanitization is performed to remove potentially harmful or malicious content from user input. The backend system employs techniques to sanitize user input, such as encoding special characters or scripts that could be used for attacks. This prevents the execution of malicious code and protects against common security vulnerabilities.

By incorporating these security measures and following best practices, our backend system will be more resilient against security threats, protecting the confidentiality, integrity, and availability of our data and ensuring a secure user experience

H. Deployment and Environment Management

The deployment and environment management process plays a crucial role in ensuring the successful deployment and operation of our backend system. In this section, we will discuss the deployment strategy used and the environment management practices implemented for our project, with a specific focus on the deployment on Render.

Deployment Strategy

For our project, the deployment strategy involves leveraging the Render platform. Render is a cloud provider that simplifies the deployment and management of web applications, providing scalability, reliability, and ease of use.

The deployment process typically involves the following steps:

1. Configuration: Set up the necessary configuration files, including the project dependencies, environment variables, and deployment settings. This ensures that the deployed application has all the required resources and configurations to function correctly.

2. Build Process: Prepare the project for deployment by building and packaging the application. This may involve compiling code, bundling assets, and generating necessary artifacts for deployment.

3. Deployment: Utilize Render's platform to deploy our backend system. Render supports various deployment methods, such as Docker containers, static site hosting, and serverless functions. The deployment process on Render is simplified, allowing us to focus on the application logic rather than infrastructure management.

4. Scaling and Load Balancing: Render offers automatic scaling and load balancing capabilities, ensuring that our backend system can handle increasing traffic and scale horizontally as needed. This ensures optimal performance and reliability, even during peak usage periods.

Environment Management

Efficient environment management is essential for maintaining consistency, security, and stability across different environments in our project's lifecycle. Here are some key practices employed for environment management:

1. Environment Variables: Utilize environment variables to store sensitive configuration values, such as API keys, database credentials, and other secrets. Render provides a secure and easy-to-use interface for managing environment variables, ensuring that sensitive information remains protected.

2. Separation of Environments: Maintain separate environments for development, staging, and production. This segregation allows for thorough testing and validation of changes before deploying them to the live production environment. Each environment should closely resemble the production environment to identify any potential issues early on.

3. Version Control and Deployment Pipelines: Utilize version control systems, such as Git, to manage our project's source code.

4. Monitoring and Error Reporting: Implement monitoring and error reporting mechanisms to track the health and performance of our deployed application. Utilize tools such as logging frameworks, error tracking services, and performance monitoring solutions to gain insights into application behavior and identify and resolve issues promptly.

By leveraging the Render platform for deployment and adhering to robust environment management practices, you can ensure smooth and reliable deployment of our backend system. Render's features and infrastructure capabilities provide scalability, security, and ease of management, enabling us to focus on building and improving our application while leaving the infrastructure complexities to the platform.

I. Conclusion (Backend):

In conclusion, the backend system that is presents a robust and secure foundation for the City Mall security system. This section provided a comprehensive overview of the backend architecture and its key components.
The backend system, built using Node.js and Express.js, offers a reliable and efficient platform for handling data storage, processing, and communication between the frontend and external services. It incorporates essential features such as a powerful API layer, secure authentication and authorization mechanisms, seamless integration with PostgreSQL for data storage, and compatibility with various third-party services.

The API documentation and reference guide included in this documentation provide developers and technical stakeholders with a clear understanding of the system's endpoints, data structures, and the expected behavior of each API route. It serves as a valuable resource for future development, maintenance, and expansion of the City Mall security system.

With a focus on security, the backend system ensures the protection of sensitive data and the privacy of users. Authentication and authorization are implemented using JSON Web Tokens (JWT), with user credentials securely stored through hashing and salting techniques. Additional security measures, such as input validation and adherence to best practices, contribute to the overall robustness of the system.

The database design, managed using Sequelize, efficiently stores and retrieves data related to users, visits, cars, shops, cinemas, offers, movies, issues, and employees. The defined relationships between these entities facilitate data manipulation and retrieval, ensuring optimal performance and scalability.

By successfully developing the backend system for the City Mall security project, the team has demonstrated our expertise in backend development, database management, and security implementation. The documentation provided offers a comprehensive guide for understanding and utilizing the backend components, enabling future enhancements and integrations.

Overall, the backend system developed for the City Mall security project showcases the team's technical competence, problem-solving abilities, and dedication to creating a secure and efficient application. It lays the foundation for a comprehensive security system that effectively manages user data, visits, cars, shops, cinemas, offers, movies, issues, and employee attendance within the City Mall environment.

Android (front end) PART

A. Introduction

The mobile application developed for this project is a critical component of the security system for the mall. It allows users to purchase cinema tickets, search for their car parking location, and receive notifications in case of a fire or emergency. Additionally, the application uses AI models to detect fires, monitor incoming and outgoing cars, and locate lost people in the mall.

By utilizing a combination of machine learning techniques and software development methodologies, we have created a security system that provides timely and accurate information to reduce risks and protect people's lives and property. Our project demonstrates the potential of AI models and mobile applications in enhancing public safety, and provides a foundation for future research and development in this field.

Ultimately, our goal is to create a safer and more secure environment for everyone in public places like malls. We believe that our security system represents a significant step forward in achieving this goal, and we hope that our work will inspire others to build upon our achievements.

B. Architecture and Technologies Used

Our mobile application is developed using a multi-fragment architecture, which provides several benefits such as better organization of code, easier maintenance, and scalability. The main activity hosts multiple fragments that represent different screens or components of the application. We utilize the bottom navigation bar to allow users to easily switch between different screens.

To manage the data and business logic of the application, we use the SharedViewModel class provided by the Android Architecture Components. The SharedViewModel class is responsible for managing the UI-related data that needs to be shared between multiple fragments or activities. SharedViewModel is scoped to the lifecycle of the activity or fragment that hosts the fragments or activities that need to share data. This allows the SharedViewModel to maintain the same state across different fragments or activities and helps to prevent data inconsistency.

We also use LiveData to observe changes in the data and update the UI accordingly. LiveData ensures that the UI remains responsive and up-to-date with the latest data. Additionally, we use SharedPreferences to store and retrieve user data when they log in. SharedPreferences is a simple way to store small amounts of data on the device.

We use the Navigation Component provided by the Android Jetpack libraries to handle the navigation between different screens. The Navigation Component provides a simple and flexible way to navigate between different screens using safe arguments. This helps to prevent navigation errors and makes it easier to maintain and update the navigation structure of the application.

To bind data to UI elements in our layouts, we use Data Binding. Data Binding allows us to write more concise and readable code, and reduces the amount of boilerplate code required to bind data to UI elements. This also improves the performance of the application by reducing the number of findViewById () calls.

To communicate with our REST API, we use Retrofit, which is a type-safe HTTP client for Android and Java. We define our API interface in a separate package and use Retrofit to create an instance of the API service. This allows us to easily make network requests and handle responses in a structured and efficient way. Additionally, we use Coroutine to show data by a background thread. Coroutine helps us to perform long-running tasks, such as network requests, in a non-blocking and efficient way. Coroutine also simplifies the code by removing the need for callbacks or nested callbacks.

We use CardView to reduce the size of images and make them more visually appealing. CardView provides a simple and flexible way to display images in a consistent and visually appealing way. Additionally, we use Lottie to add animation loading icons while data is not yet loaded. Lottie is a library that allows us to easily add animations to our application.

We use a mix of ViewPager and RecyclerView to display data in a stylish and efficient way. ViewPager is used to display multiple fragments in a swipeable view, while RecyclerView is used to display a list of items. Using a mix of ViewPager and RecyclerView allows us to display data in a visually appealing and efficient way.

To add visual appeal, we use animations like Fade in and Fade out. Animations are used to provide visual feedback to the user and enhance the overall user experience. Additionally, we utilize Notifications to keep users informed of important updates or events. Notifications are a simple and effective way to keep users informed of important information, even when the application is not in use.

We also use Kotlin as the primary programming language in developing our mobile application. Kotlin is fully supported by Android and provides many benefits, such as improved readability, reduced boilerplate code, and improved performance. By using Kotlin, we were able to write more concise and readable code, which helped to reduce development time and improve the overall quality of the application.

Overall, our mobile application uses a combination of modern technologies, including SharedViewModel, LiveData, SharedPreferences, Navigation Component, Data Binding, Retrofit, Coroutines, CardView, Lottie, ViewPager, RecyclerView, Animations, Notifications and Kotlin to provide a seamless and engaging user experience. This architecture and technologies used help to create a well-organized, scalable, and efficient application.

C. Features and Functionality

Our application is designed to provide a convenient, user-friendly, and secure experience for mall visitors. The application offers the following features and functionality:

Authentication: To access the features of our application, users are required to create an account and log in using their email and password. This authentication process ensures that user data is protected and that only authorized users can access the features of the application.

When creating an account, users are prompted to provide their name, email address, phone number, and a secure password.

Cinema Ticket Purchase: Users can purchase cinema tickets directly through the application. The cinema section of the application displays a list of available movies and showtimes, and users can select their preferred movie and showtime, choose their seats, and confirm the purchase. We do not currently offer a payment gateway within the application, but instead provide instructions for users to complete the payment process at the cinema box office.

To ensure a seamless and hassle-free payment experience for our users, we provide clear and concise instructions on how to complete the payment process at the cinema box office. This includes information on the accepted payment methods, ticket pricing, and any applicable discounts or promotions.

Car Parking Search: Users can search for the capacity of each parking section using the application's car parking search feature. The feature displays a list of available parking sections and allows users to select their preferred parking section. Additionally, the feature provides real-time information about the capacity of each parking section, allowing users to find available parking spots quickly and easily.

To provide accurate and up-to-date information on the cost of parking, the application uses AI models to detect the time the car enters and exits the garage. This information is used to calculate the parking duration and cost, which is then displayed to the user. Users can also search for their car by entering their car number, which will provide them with information on the location of their car and the cost of parking.

Lost People: In the unfortunate event that a person is lost within the mall, our application offers a feature that allows users to upload a photo and description of the lost person. The application uses an AI model to analyze the uploaded photo and match it against surveillance camera footage within the mall. If a match is found, the application sends a notification to the user who posted the photo, providing them with the location of the lost person.

This feature is designed to provide a quick and efficient way for users to report and locate lost individuals within the mall. By leveraging the power of AI and real-time notifications, we can help reunite lost individuals with their families and loved ones in a timely manner.

D. User Interface

As this is my first project, designing the user interface can be a challenging task. One of the challenges that I may encounter is ensuring that the user interface is both visually appealing and easy to use.

To address this challenge, I can look at other mall apps to learn from their design choices and best practices. By analyzing these apps, I can gain insights into common design patterns and elements that work well for similar applications. However, it's important to keep in mind that I should create a unique and original design that meets the specific needs and goals of my application.

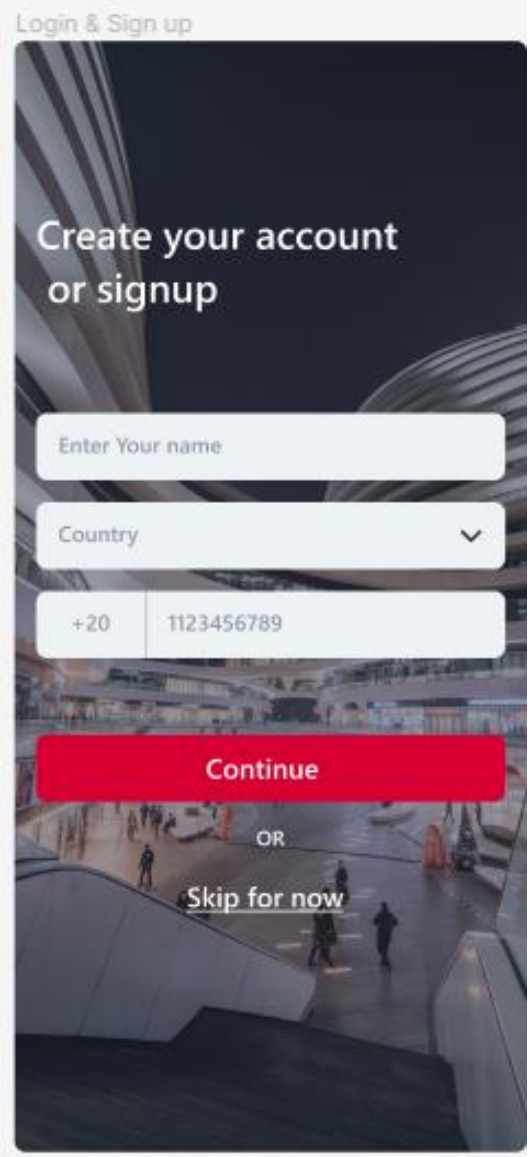
Another challenge I may encounter is ensuring that the user interface is intuitive and easy to navigate. To address this challenge, I can conduct user testing to gather feedback on the usability of the interface. This feedback can be used to refine the design and make it more user-friendly.

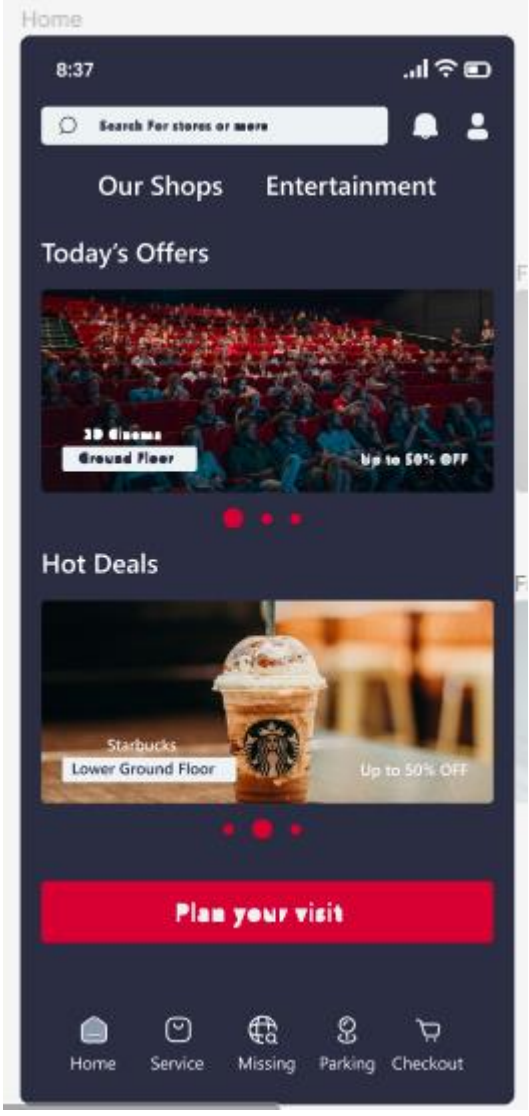
It's also important to keep the user's needs and preferences in mind while designing the interface. By conducting user research, surveys, and focus groups, I can gather insights into the user's behavior, preferences, and expectations while using the application. This will help me create a user interface that is tailored to their needs and preferences, resulting in a better user experience.

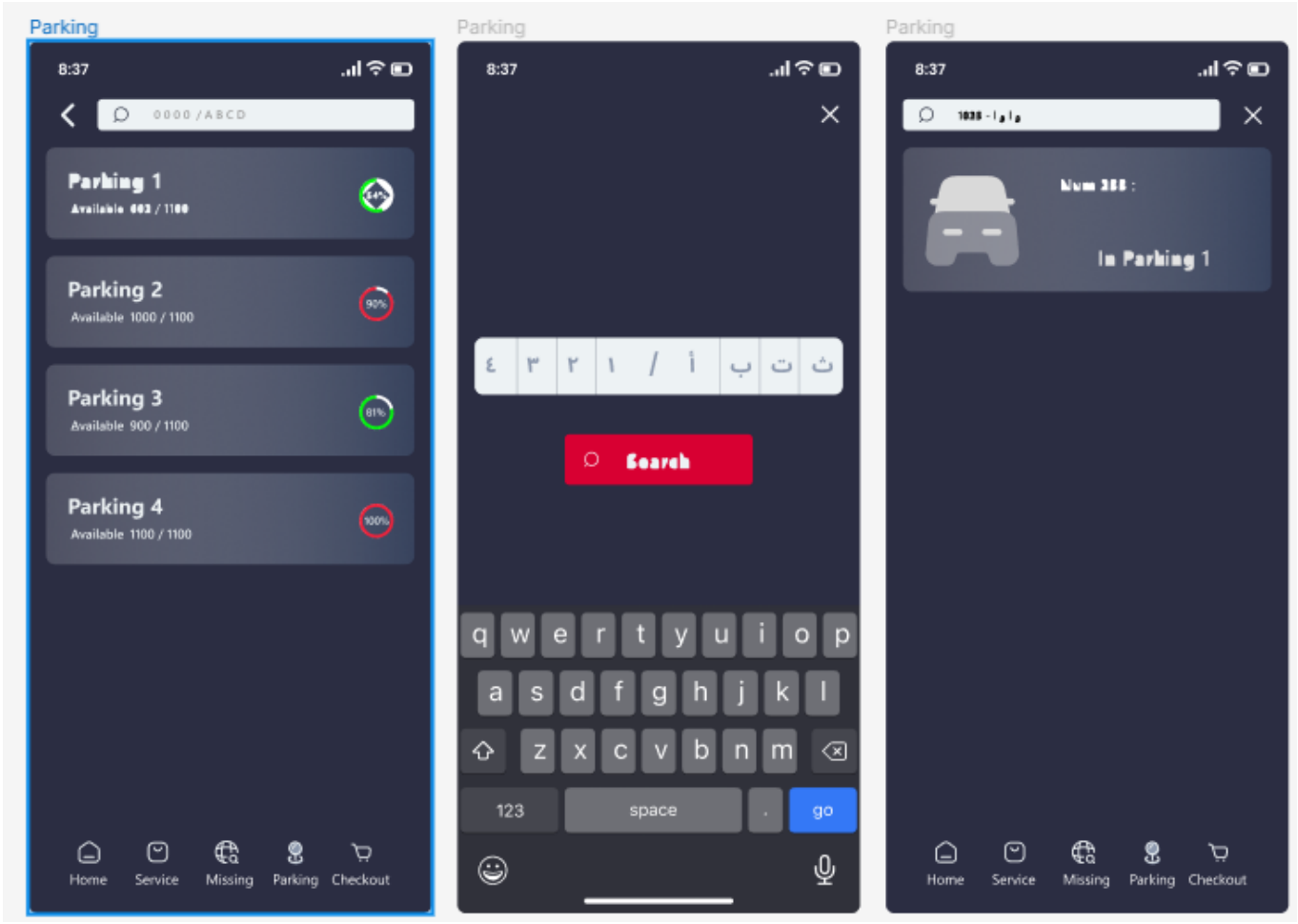
Finally, ensuring the user interface is compatible with different devices and screen sizes can be a challenge. To address this, I can use responsive design techniques to ensure that the user interface can adapt to different screen sizes and resolutions.

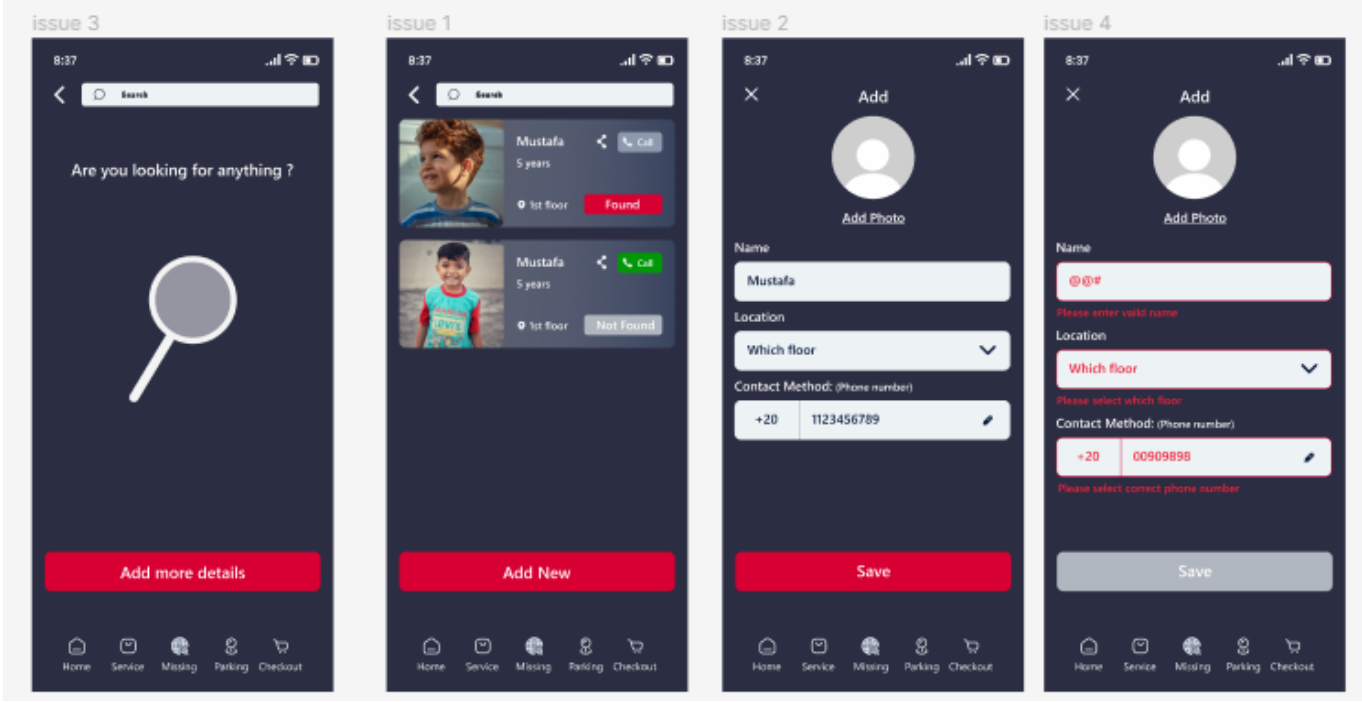
Overall, by looking at other mall apps, conducting user testing and research, and using responsive design techniques, I can create a user interface that is visually appealing, intuitive, and accessible to all users, while also being compatible with a wide range of devices and screen sizes.

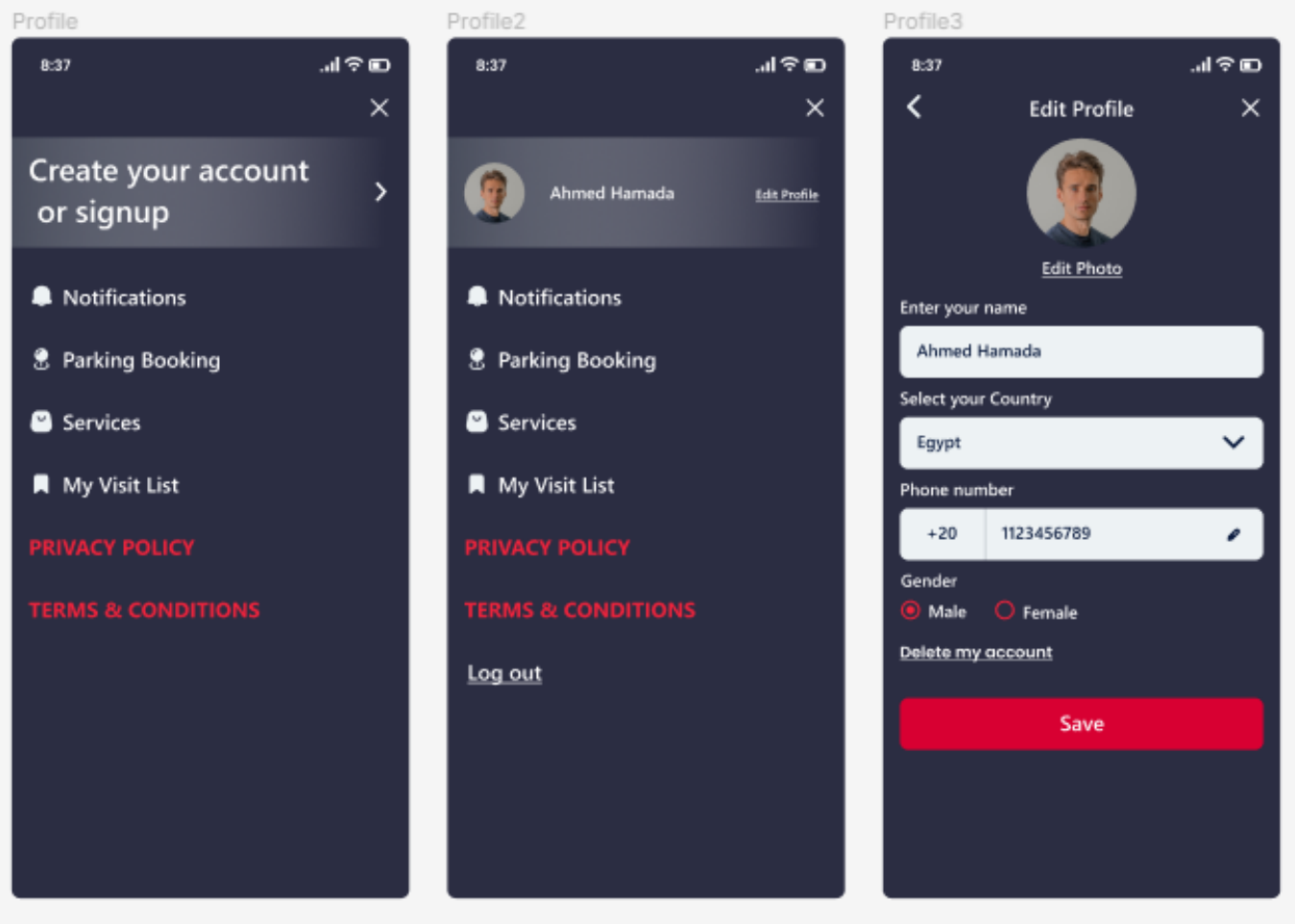
To provide a clear visual representation of the design and functionality of the application, I have included screenshots and mockups in this document. These screenshots and mockups demonstrate how the features of the application have been implemented within the user interface. Including such visual aids is essential to ensure that the application meets the user's needs and expectations, and I believe that they will greatly enhance the understanding and evaluation of my graduation project.






Parking
8:37
0000 / ABCD
Parking 1
Available 603 / 1100
64%
Parking 2
Available 1000 / 1100
90%
Parking 3
Available 900 / 1100
81%
Parking 4
Available 1100 / 1100
100%
Home
Service
Missing
Parking
Checkout
Parking
8:37
Search
Parking
8:37
In Parking 1
Home
Service
Missing
Parking
Checkout


issue 3
Are you looking for anything ?
Add more details
issue 1
Mustafa
5 years
1st floor
Found
Mustafa
5 years
1st floor
Not Found
Add New
issue 2
Add
Add Photo
Name
Mustafa
Location
Which floor
Contact Method: (Phone number)
+20
1123456789
Save
issue 4
Add
Add Photo
Name
@@#
Please enter valid name
Location
Which floor
Please select which floor
Contact Method: (Phone number)
+20
00909898
Please select correct phone number
Save



E. Implementation Details

User Interface: I started developing the application using a multi-activity architecture and created the layouts and implemented the app using static data. However, due to a mistake I made in Android Studio, the app data was lost, and I had to start again from scratch. This time, I decided to use a single activity with multiple fragments to improve the app's navigation and user experience.

The bottom navigation bar was implemented to move between app sections, while a ViewPager was used for the home section to display data for individual shops horizontally. A RecyclerView was used to display data vertically, such as a list of items.

The Navigation Component was used to move between different fragments, and a shared view model and LiveData were used to hide the bottom navigation bar in certain

activities, such as the login and sign up screens.

To display data in the ViewPager, I used the ViewPager feature provided by the Android SDK. Each shop's information was displayed on a separate fragment, and the ViewPager allowed users to swipe left or right to move between different shops.

To display data in the RecyclerView, I used the RecyclerView feature provided by the Android SDK. RecyclerView is a UI component that provides a flexible and efficient way to display large lists of data. In this case, I used the RecyclerView to display a list of items, with each item containing an image and some text data.

To implement this feature, I used the RecyclerView widget in the layout file, and created a custom adapter to bind the data to the RecyclerView. I also used Glide, a popular image loading library for Android, to load images into the ImageView.

To add a rounded corner effect to the images in the RecyclerView, I used the CardView feature provided by the Android SDK. CardView is a UI component that provides a customizable container for displaying content. In this case, I used the CardView to wrap the ImageView that contained the image to be displayed. By setting the card corner radius to a specific value, I was able to create a rounded corner effect for the image.

Backend Integration: The application was linked to the backend using Retrofit and Coroutine to perform API calls in the background. User authentication was implemented, and SharedPreferences were used to store user data locally.

Retrofit is a type-safe HTTP client for Android and Java, which provides an easy-to-use interface for making HTTP requests and handling their responses. Coroutine is a concurrency design pattern that allows for asynchronous and non-blocking code execution.

The backend was responsible for storing and serving data to the client application. The server was built using Node.js and Express, and the database used was PostgreSQL with sequelizeORM. The server API was designed to be RESTful, allowing for easy integration with the client application.

User authentication was implemented using JSON Web Tokens (JWT). When a user logs in, the server generates a JWT token that contains the user's information, which is then stored on the client side. This token is used to make authenticated API requests to the server.

To perform API calls, I used Retrofit and Coroutine to handle the requests asynchronously. The data received from the server was then serialized using Gson, a library that converts Java objects to JSON and vice versa. The deserialized data was then used to update the UI.

Data Storage: For storing user data, Shared Preferences were used to store user data such as their name, email address, and password. Shared Preferences is a lightweight and simple key-value storage system provided by the Android SDK. It allows for the storage of small amounts of data that need to be persisted across app sessions.
When a user creates an account, their data is stored in Shared Preferences as key-value pairs. The data can be retrieved using the corresponding key.

Error Handling: Proper error handling mechanisms were implemented in the application to handle unexpected events, such as network errors or server failures. This included using try-catch blocks to catch and handle exceptions in the code, providing informative error messages to the user, and using logging tools to track errors and monitor application performance.

Data Analysis PART

A. introduction

Data analysis is a crucial component of our academic graduation project, where we leverage the power of data to gain insights, make informed decisions, and enhance the overall functionality of the system. In this section, we will explore the data analysis techniques and methodologies employed to extract valuable information from the collected data.

With the integration of our AI models, backend system, and user interactions, our project generates a significant amount of data. This data includes user profiles, visit records, car information, lost people reports, and various other relevant data points. By analyzing this data, we can uncover patterns, trends, and correlations that can be utilized to improve the system's performance and provide a better user experience.

The data analysis process begins with data collection and preprocessing. We ensure that the data is cleansed, transformed, and organized in a structured manner suitable for analysis. Once the data is prepared, we apply various statistical and analytical techniques to derive meaningful insights. These techniques may include exploratory data analysis, descriptive statistics, data visualization, and machine learning algorithms.

The benefits of data analysis in our project are multifold. Firstly, it allows us to identify patterns and anomalies, such as unusual behaviors like fire incidents or lost individuals within the mall. By detecting such occurrences, we can promptly respond and ensure the safety and security of visitors and employees. Secondly, data analysis helps us optimize the system's performance by identifying areas for improvement, such as optimizing car search algorithms or enhancing user recommendations based on their preferences and past behavior.

Furthermore, data analysis enables us to make data-driven decisions for resource allocation, marketing strategies, and system enhancements. By understanding visitor behavior, peak hours, popular shops, and other relevant metrics, we can allocate resources more efficiently, improve customer engagement, and enhance the overall visitor experience.

B. Define the problem to use Customer Data Analysis:

- The problem that customer data analysis aims to solve is to gain insights into customer behavior and preferences so that businesses can make data-driven decisions to improve customer satisfaction, increase sales, and optimize marketing efforts. By analyzing customer data, businesses can identify patterns and trends that can help them understand how customers interact with their products or services, what their preferences are, and how they make purchasing decisions. This information can be used to personalize marketing messages, improve product offerings, and optimize pricing strategies. However, the challenge lies in collecting and analyzing large amounts of data in a meaningful way and using the insights gained to improve business operations. Customer data analysis is a process that involves the collection, processing, and analysis of large volumes of customer-related data. This data can come from various sources, including customer surveys, social media platforms, website usage metrics, sales data, and more. The goal of customer data analysis is to gain insights into customer behavior, preferences, and needs, and use this information to improve business operations.
- One of the main challenges of customer data analysis is the sheer volume of data that needs to be processed. Businesses may collect data from multiple sources, and this data can be in different formats, making it difficult to analyze. Additionally, it can be challenging to identify meaningful insights from the data, as not all data points may be relevant to the business's goals.

To overcome these challenges, businesses need to have a clear understanding of their objectives and the questions they want to answer using data analysis. They also need to ensure that the data they collect is accurate, relevant, and up-to-date, and use the right tools and techniques to process and analyze the data.

- Some of the common techniques used in customer data analysis include data mining, machine learning, and predictive analytics. These techniques can help businesses identify patterns and trends in customer behavior, predict future customer behavior, and segment customers based on their needs and preferences.
- Overall, customer data analysis is a powerful tool that can help businesses make informed decisions, improve customer satisfaction, and increase profitability. By collecting and analyzing customer data, businesses can gain a competitive edge and stay ahead of the curve in today's data-driven business landscape.

C. Define the goals about use Customer data analysis.

1. Personalize marketing messages: By analyzing customer data, businesses can gain insights into customer preferences and tailor marketing messages to specific customer segments. This can improve the effectiveness of marketing efforts and increase customer engagement.
2. Improve product offerings: Customer data analysis can help businesses identify patterns and trends in customer behavior, which can be used to improve product offerings. For example, businesses can use customer feedback to identify areas of improvement and make changes to products or services accordingly.
3. Optimize pricing strategies: By analyzing customer data, businesses can gain insights into how customers respond to different pricing strategies, such as discounts, promotions, and pricing tiers. This information can be used to optimize pricing strategies and increase sales.
4. Increase customer satisfaction: Customer data analysis can help businesses understand what customers value most and prioritize efforts to improve the customer experience. For example, businesses can use customer feedback to identify pain points and make changes to address them.
5. Identify new opportunities: By analyzing customer data, businesses can identify new opportunities for growth and expansion. For example, businesses can use customer data to identify untapped markets or new product offerings that align with customer needs and preferences.
6. Better decision-making: Data analytics provides organizations with valuable insights and information which can facilitate informed decision-making.
7. Data analytics provide managers with the tools, techniques, and processes necessary for making efficient and effective decisions. By using algorithms and techniques to organize, visualize, and curate data, business analytics enable managers to make decisions that are relevant to the organization's operations and goals.
8. Personalize the customer experience
   Another benefit of data analytics in business is the ability to personalize customer experiences.
   Companies can gain information on customer behavior by creating detailed customer information from these data to provide a unique experience.
9. Retention and loyalty
   Data analytics can also aid in customer retention and loyalty by providing organizations with insights into their customers' preferences, behaviors, and needs.
10. Increase the efficiency of work
    It can be helpful to analyze large data sets quickly and present it in a structured way to help achieve your organization’s goals. It promotes a culture of efficiency and

teamwork by allowing managers to share detailed information with employees.

D. Data collection:

Data collection is a critical component of understanding customers and making informed decisions in business. Without accurate and relevant data, businesses risk making decisions based on assumptions rather than facts, which can lead to lost opportunities and revenue. In this article, we'll discuss the importance of data collection in business and some common methods for collecting data.

Why is Data Collection Important?

Data collection is important in business for several reasons:

1. Understanding Customers: Data collection can help businesses understand their customers better by providing insights into their behavior, preferences, and needs. This information can be used to develop
2. Targeted marketing campaigns, improve products and services, and increase customer satisfaction.
3. Making Informed Decisions: Data collection can provide businesses with the information they need to make informed decisions about marketing, product development, and other business processes. By analyzing data, businesses can identify trends, patterns, and opportunities that might otherwise go unnoticed.
4. Improving Performance: Data collection can help businesses identify areas where they can improve performance. For example, sales data can be used to identify which products are selling well and which are not, allowing businesses to adjust their inventory accordingly.

Common Methods for Data Collection

There are several methods for collecting data in business, including:

1. Surveys: Surveys are a common method for collecting data about customers and their preferences. Surveys can be conducted online, over the phone, orin person, and can cover a range of topics such as customer satisfaction, product preferences, and shopping behaviors.
2. Customer Feedback: Customer feedback can be collected through various channels including social media, email, and customer service interactions. Feedback can provide valuable insights into customer needs and can help businesses identify areas where they can improve.

3. Sales Data: Sales data can be used to understand customer purchasing patterns and trends. Businesses can analyze sales data to identify which products or services are most popular and which customers are most valuable.
4. Demographic Data: Demographic data can provide insights into the characteristics of a business's customer base such as age, gender, income level, and education level. This information can be used to create targeted marketing campaigns and to develop products that meet the needs of specific customer segments.
5. Website Analytics: Website analytics can provide insights into how customers interact with a business's website. This can include information about how customers find the website, which pages they visit, and how long they stay on the site.

Conclusion

Data collection is a critical component of understanding customers and making informed decisions in business. By collecting and analyzing data effectively, businesses can gain valuable insights into their customers, identify areas for improvement, and make data-driven decisions that drive growth and success. It's important for businesses to choose the appropriate methods for collecting data and to ensure that the data is collected in an unbiased and accurate way. By doing so, businesses can maximize the value of their data and use it to drive their success.

E. Exploring Customer Personality Analysis

Understanding your customers is key to building a successful business. One way to gain insight into your customers is to conduct a customer personality analysis. This involves identifying the personality traits and characteristics of your customers, and using this information to tailor your marketing and sales strategies.

The first step in conducting a customer personality analysis is to gather data about your customers. This can be done through surveys, interviews, or by analyzing customer behavior and purchase patterns. Once you have collected this data, you can begin to identify common traits and characteristics among your customers.

Some of the most common personality traits that are analyzed in a customer personality analysis include:

Introversion vs. extroversion: This trait refers to whether people tend to be quiet and reserved (introverted) or outgoing and sociable (extroverted). Knowing whether your customers are more introverted or extroverted can help you tailor your marketing messages and sales approach accordingly.
Openness: This trait refers to how open people are to new ideas and experiences. Customers who are more open may be more likely to try new products or services, while those who are less open may be more resistant to change.
Conscientiousness: This trait refers to how organized and responsible people are. Customers who are more conscientious may be more likely to follow through on purchases and recommendations, while those who are less conscientious may be more impulsive.
Agreeableness: This trait refers to how cooperative and friendly people are. Customers who are more agreeable may be more receptive to marketing messages and may be more likely to recommend your products or services to others.
Neuroticism: This trait refers to how emotionally stable people are. Customers who are more neurotic may be more sensitive to changes in their environment or routine, while those who are less neurotic may be more adaptable.
Once you have identified the personality traits and characteristics of your customers, you can begin to tailor your marketing and sales strategies accordingly. For example, if you know that your customers are more introverted, you may want to focus on building relationships with them through email or social media rather than in-person interactions. Similarly, if you know that your customers are more open to new ideas and experiences, you may want to focus on developing innovative products or services that appeal to their sense of adventure.
By conducting a customer personality analysis, you can gain valuable insights into your customers and develop strategies that are more effective at reaching and engaging them. With this information, you can create a more personalized and targeted marketing approach that resonates with your customers and helps you build long-term relationships with them.
In conclusion, customer personality analysis is a valuable tool for businesses looking to gain insights into their customers and develop more effective marketing and sales strategies. By understanding customers' personality traits and characteristics, businesses can create a more personalized and targeted approach that resonates with customers and helps build long-term relationships with them.

F. Data understanding

Problem Statement

Customer Personality Analysis is a detailed analysis of a company's ideal customers. It helps a business to better understand its customers and makes it easier for them to modify products according to the specific needs, behaviors and concerns of different types of customers.

Customer personality analysis helps a business to modify its product based on its target customers from different types of customer segments. For example, instead of spending money to market a new product to every customer in the company's database, a company can analyze which customer segment is most likely to buy the product and then market the product only on that particular segment.

Work Environment:
Tool: Jupiter Notebook
Programming Language: Python 3
Visualization: Matplotlib, Seaborne

Attributes

People

- ID: Customer's unique identifier
- Year_Birth: Customer's birth year
- Education: Customer's education level
- Marital_Status: Customer's marital status
- Income: Customer's yearly household income
- Kidhome: Number of children in customer's household
- Teenhome: Number of teenagers in customer's household
- Dt_Customer: Date of customer's enrollment with the company
- Recency: Number of days since customer's last purchase
- Complain: 1 if the customer complained in the last 2 years, 0 otherwise

Products

- MntCinemaTicket: Amount spent on wine in last 2 years
- MntFruits: Amount spent on fruits in last 2 years
- MntMeatProducts: Amount spent on meat in last 2 years
- MntFishProducts: Amount spent on fish in last 2 years

- MntSweetProducts: Amount spent on sweets in last 2 years
- MntGoldProds: Amount spent on gold in last 2 years

Promotion

- NumDealsPurchases: Number of purchases made with a discount
- AcceptedCmp1: 1 if customer accepted the offer in the 1st campaign, 0 otherwise
- AcceptedCmp2: 1 if customer accepted the offer in the 2nd campaign, 0 otherwise
- AcceptedCmp3: 1 if customer accepted the offer in the 3rd campaign, 0 otherwise
- AcceptedCmp4: 1 if customer accepted the offer in the 4th campaign, 0 otherwise
- AcceptedCmp5: 1 if customer accepted the offer in the 5th campaign, 0 otherwise
- Response: 1 if customer accepted the offer in the last campaign, 0 otherwise

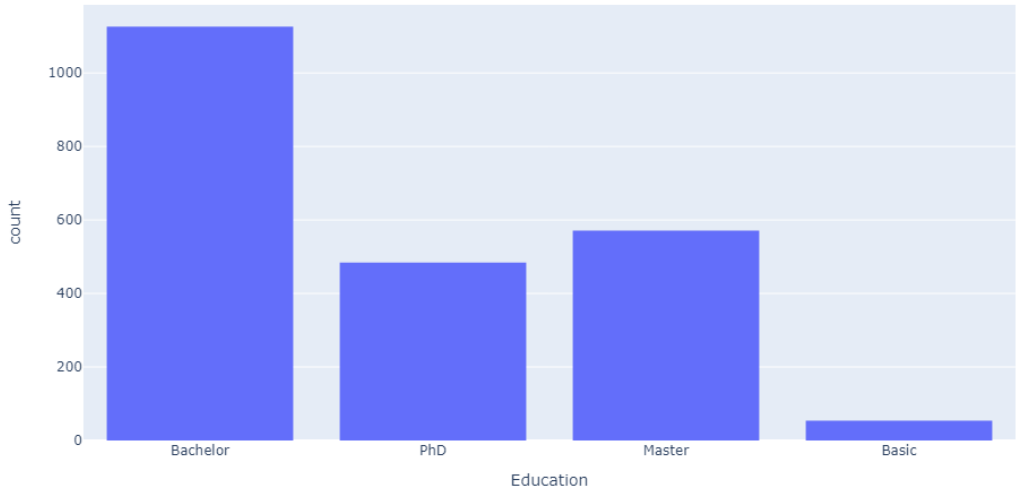
Place

- NumWebPurchases: Number of purchases made through the company's website
- NumCatalogPurchases: Number of purchases made using a catalogue
- NumStorePurchases: Number of purchases made directly in stores
- NumWebVisitsMonth: Number of visits to company's website in the last month

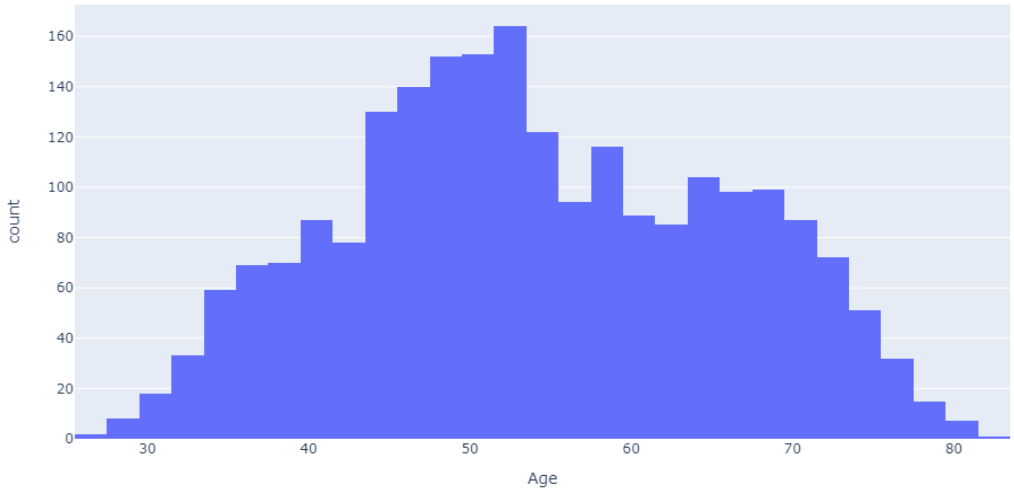
G. Exploring Data Analysis.

Visualization.

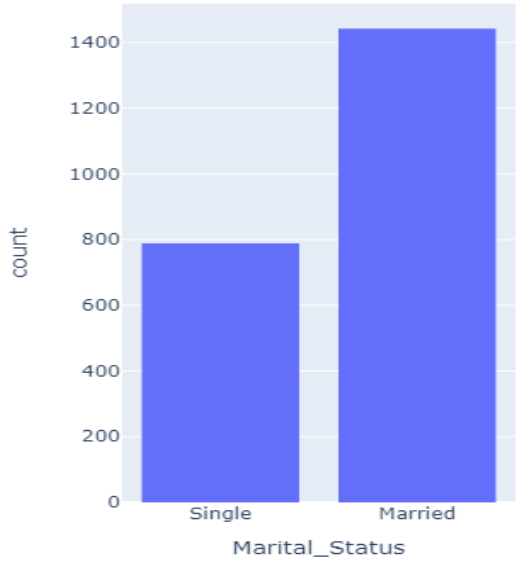
1- One variant.



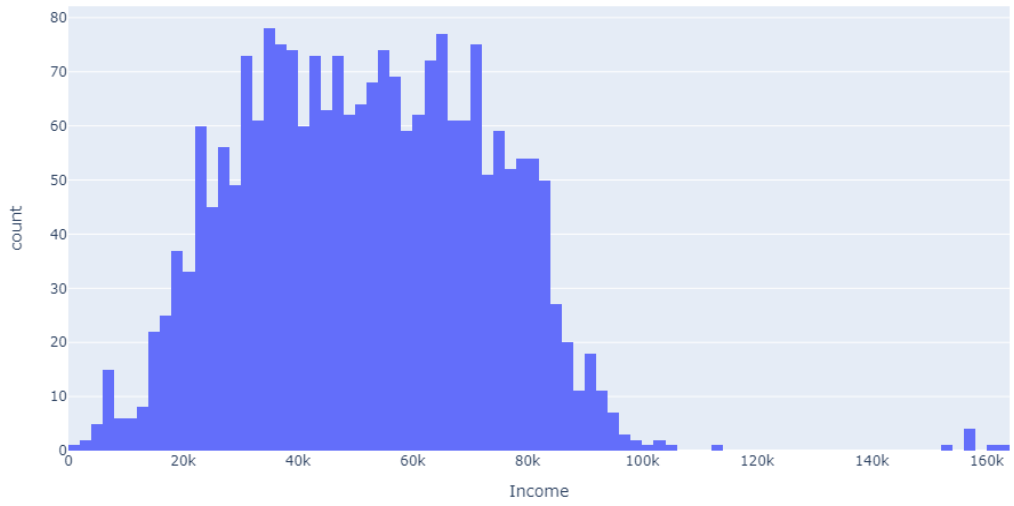
As we show in this figure the majority of clients come from graduate-level educational backgrounds, according to the education graph.



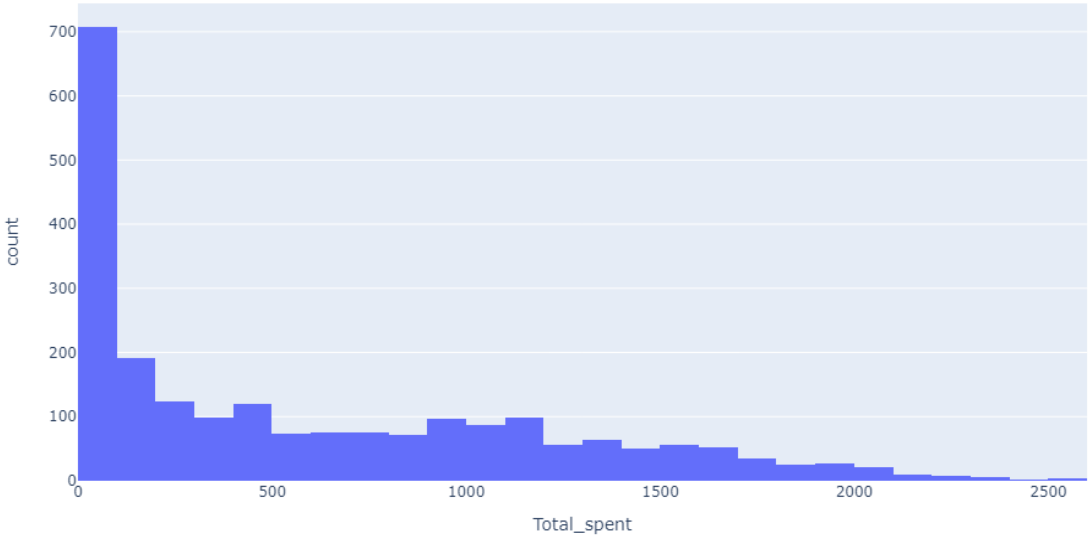
As we show in this figure Age of the customers is nearly normally distributed, with most of the customers aged between 40 and 60.



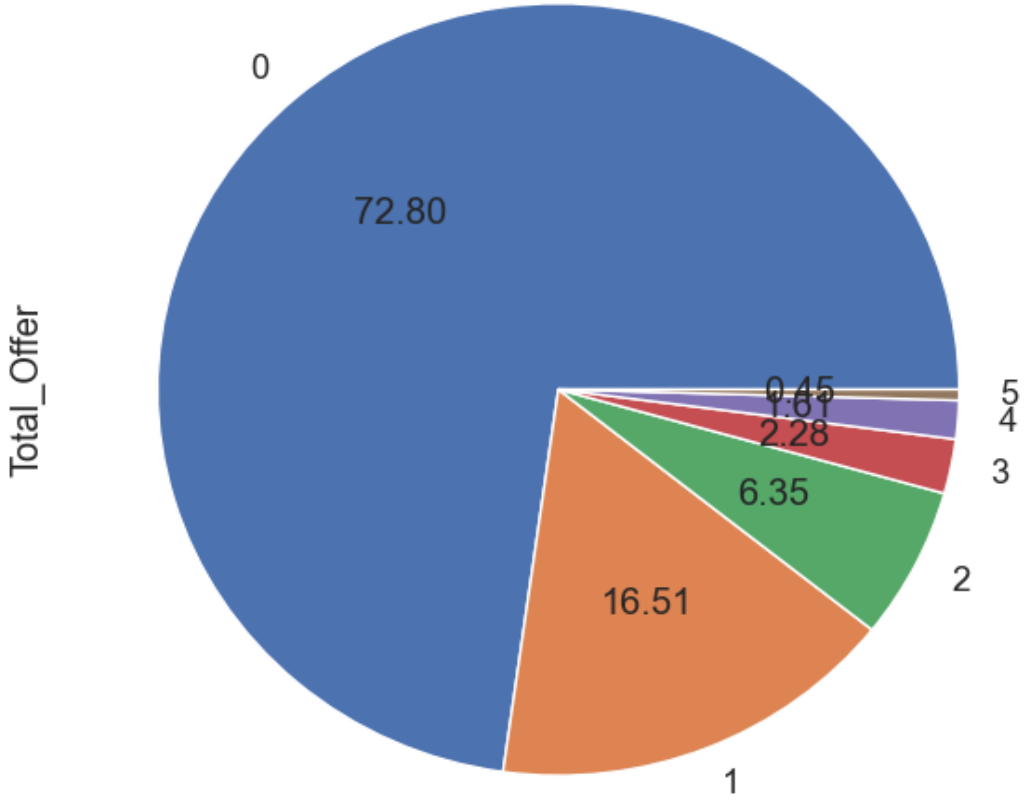
As we show in this figure most clients are married.



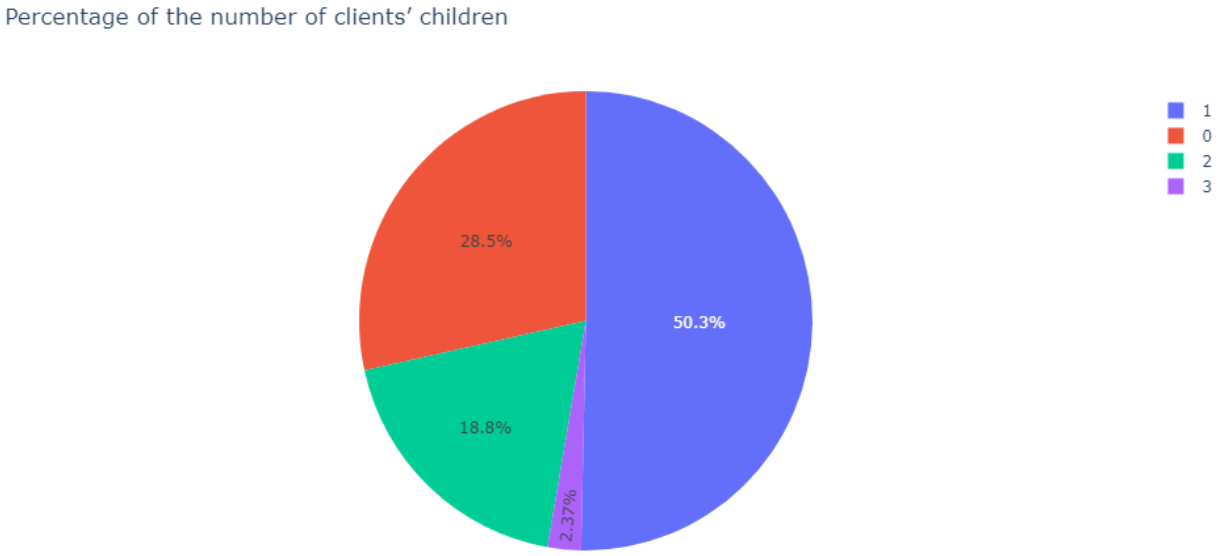
As we show in this figure the salaries of the customers have a normal distribution with most of the customers earning between 25000 and 85000.



This figure show Total spent by the customer

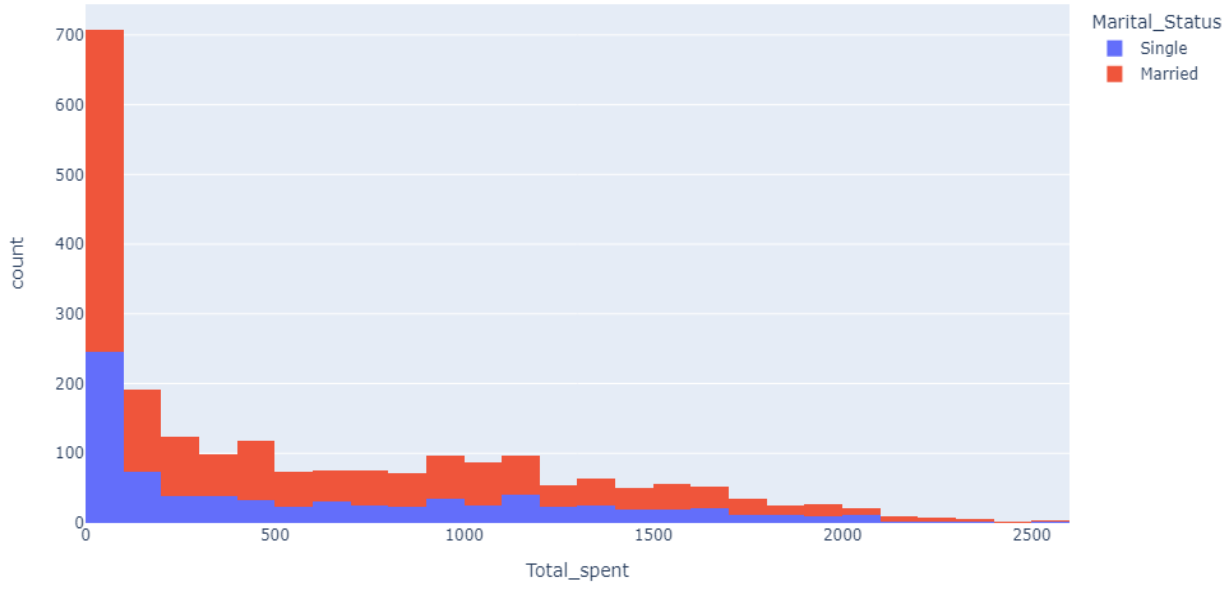


72.80% of customers have not accepted the offer in any of the campaigns

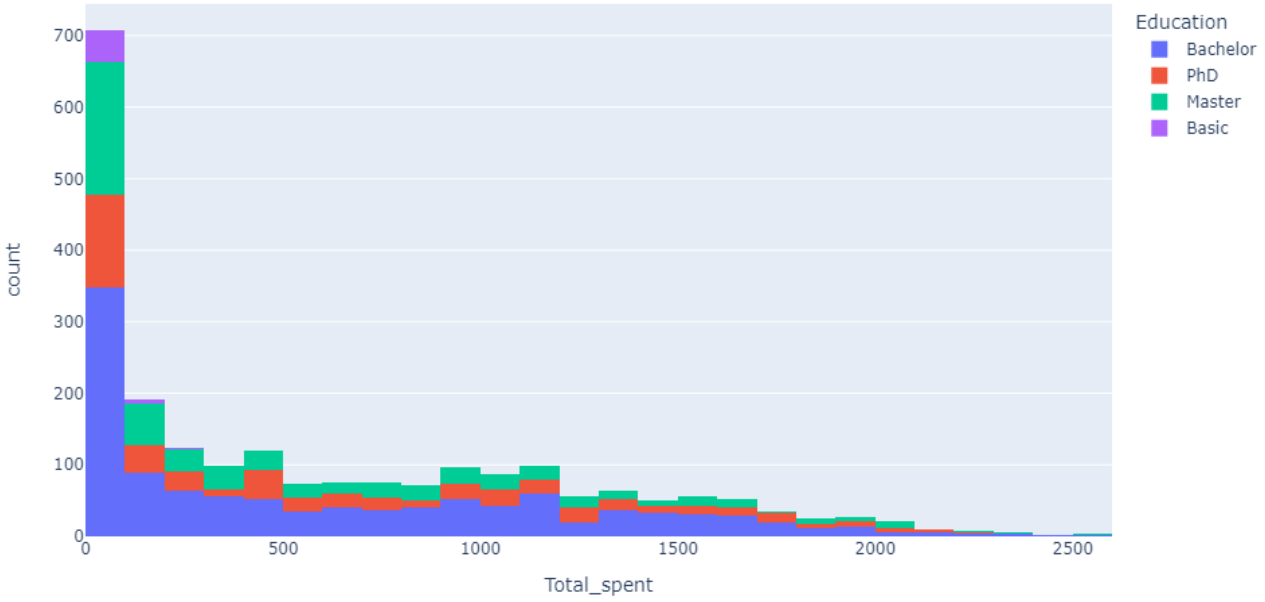


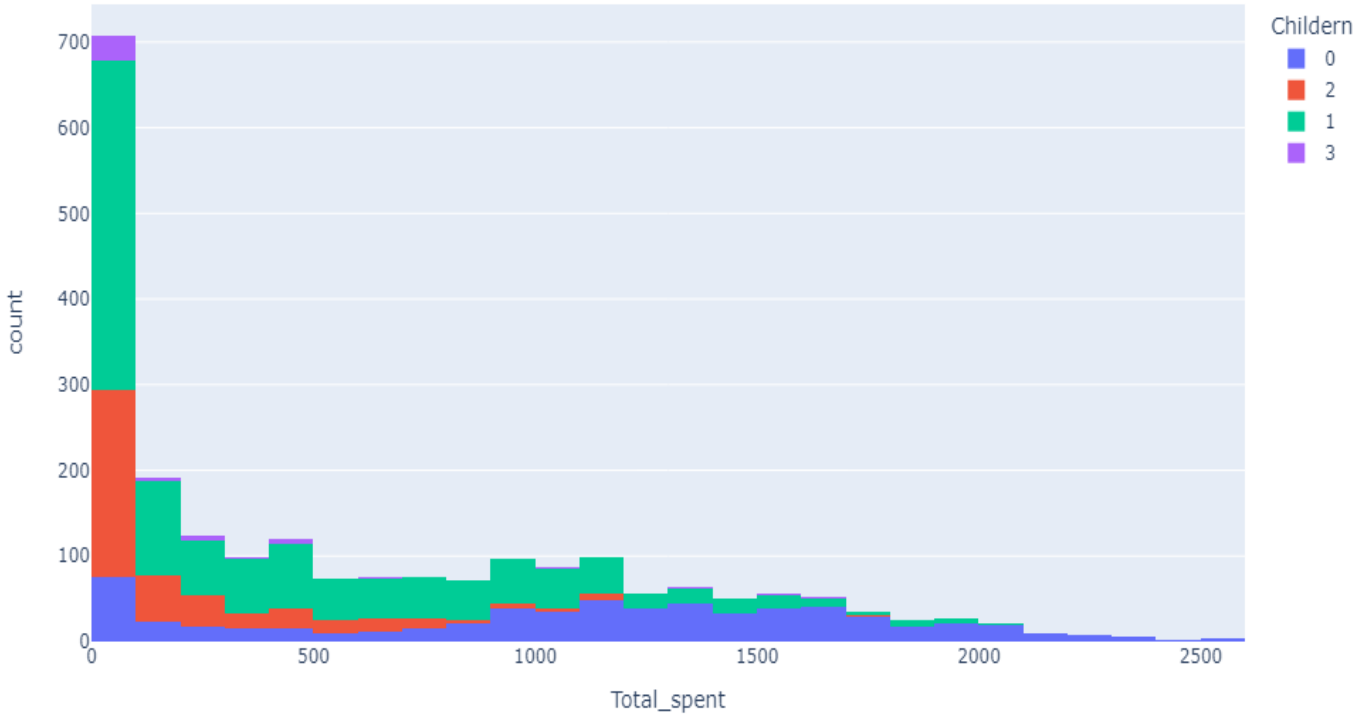
About 50.3% of the customers have only one child.
28.5% of the customers do-not have any children at home while 18.8% of them have 2 children.

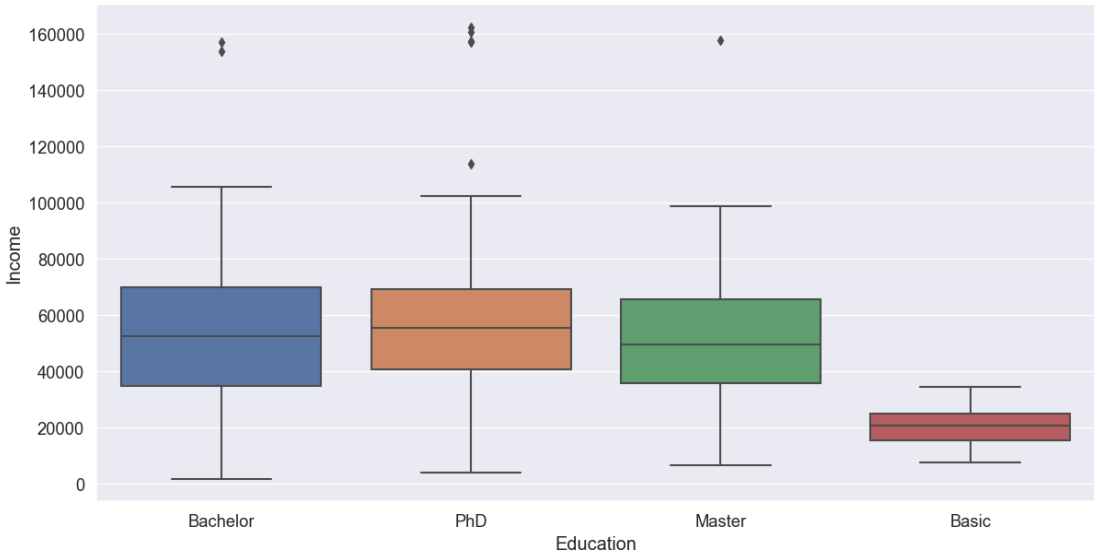
2-Two variant
Spending Distribution by Marital Status
Spending Distribution by Education
Spending Distribution by Children
These plots aim to show the distribution of spending across different categories of



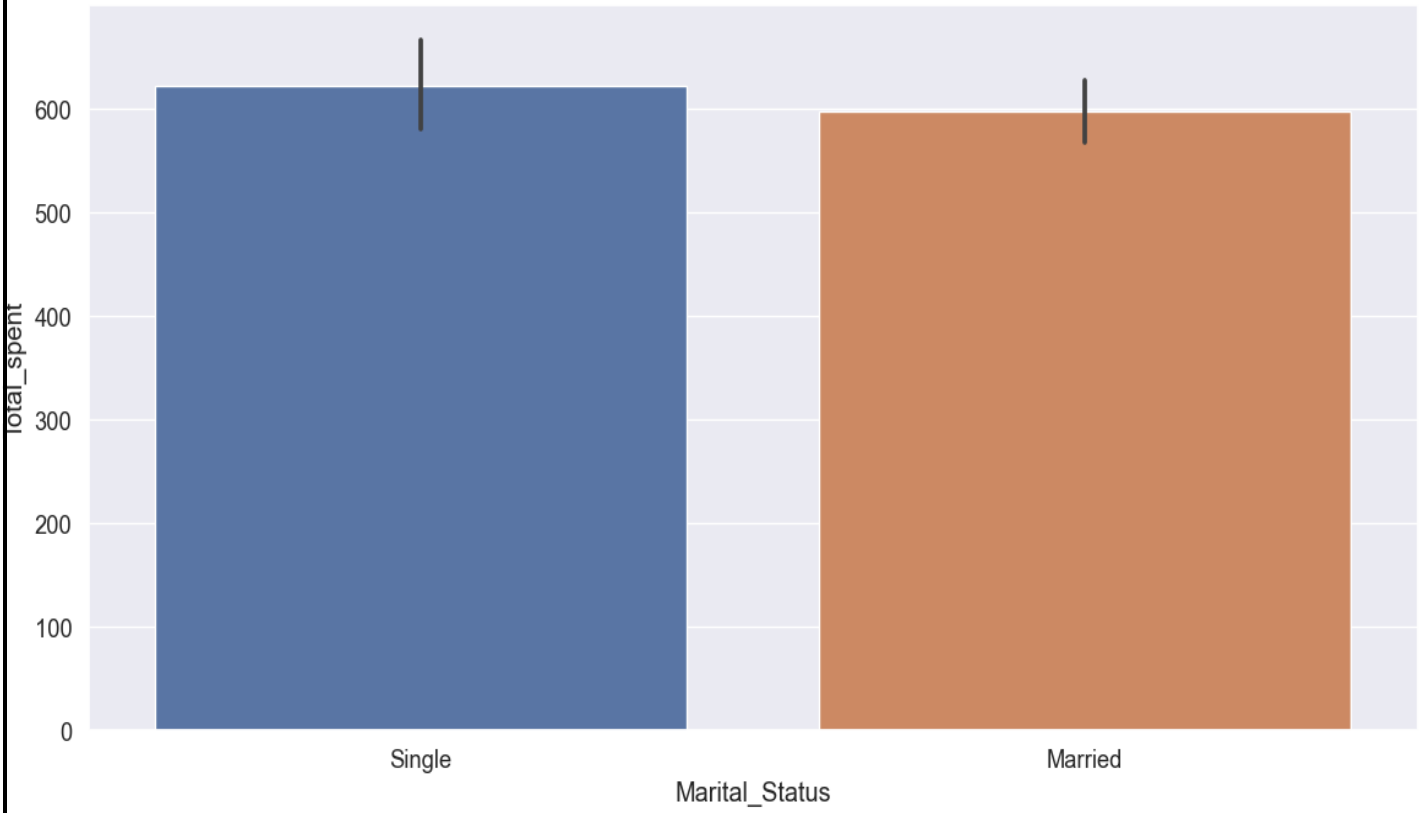
Marital Status, Education Level and Is Parent. The histograms allow us to see the frequency of different spending levels in each category and to compare the spending distributions across categories





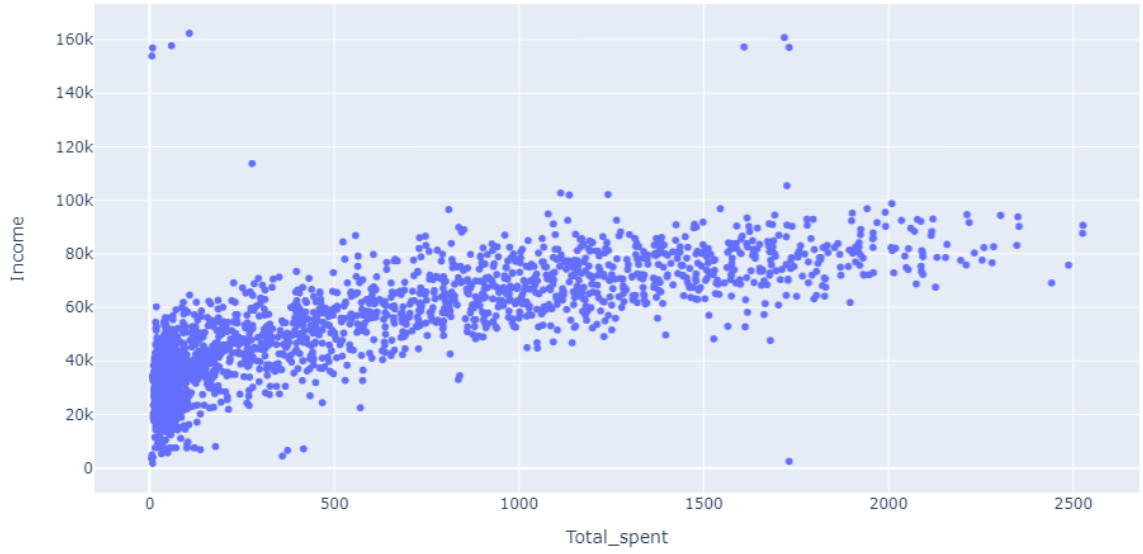


As we show in this figure almost most graduate clients with a master's degree have the

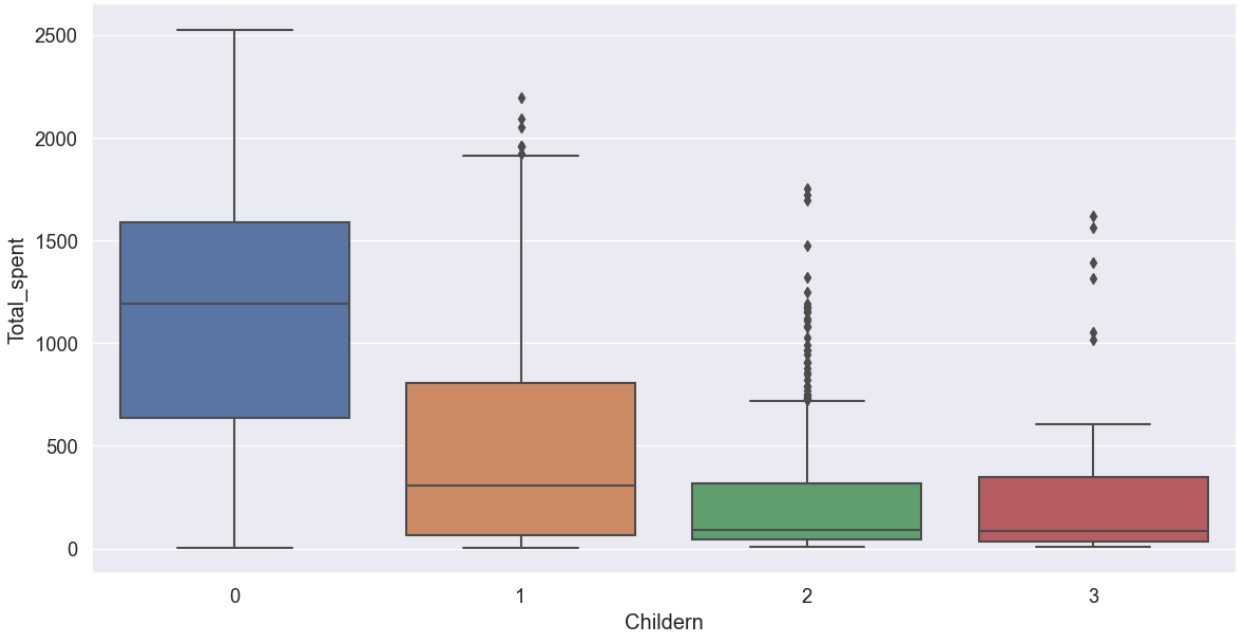


same income

As we show in this figure despite being the minority, the Singles spent more money on the average as compared to the customers having partners



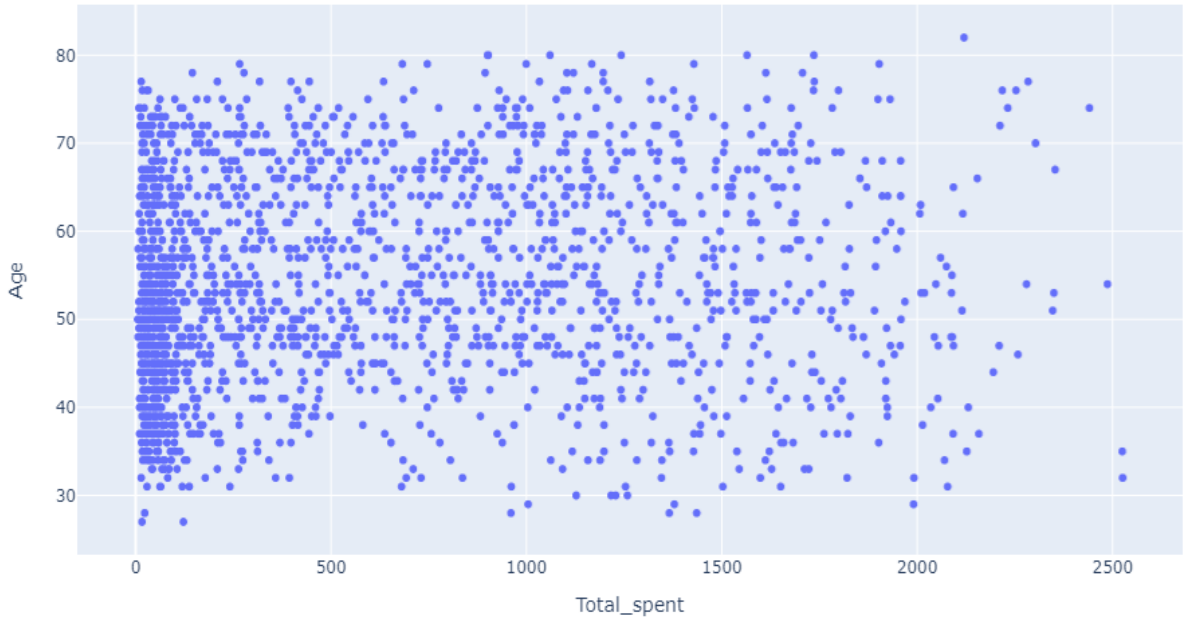
The relationship is linear. Customers having higher salaries are spending more.



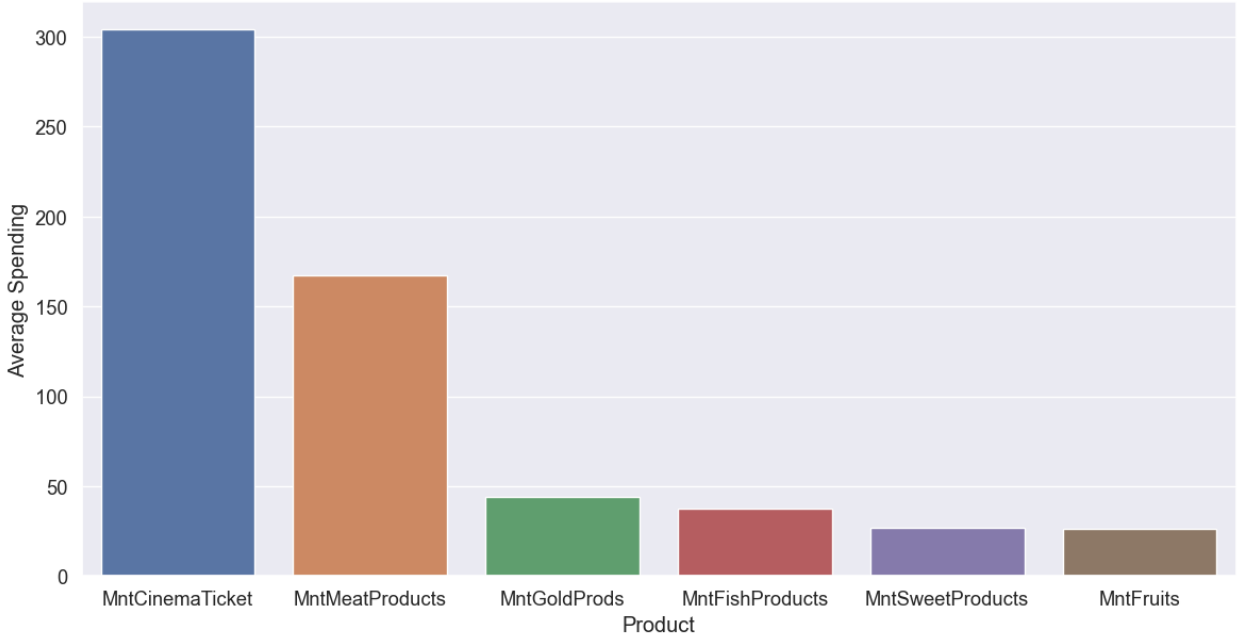
Contrary to the expected total number of customers who have children, the total spent rate is greater

Customers who don't have any children at home spent higher than the customers having 1 children.

The customers having 1 children are spending higher than the customer’s having 2 and 3 children.



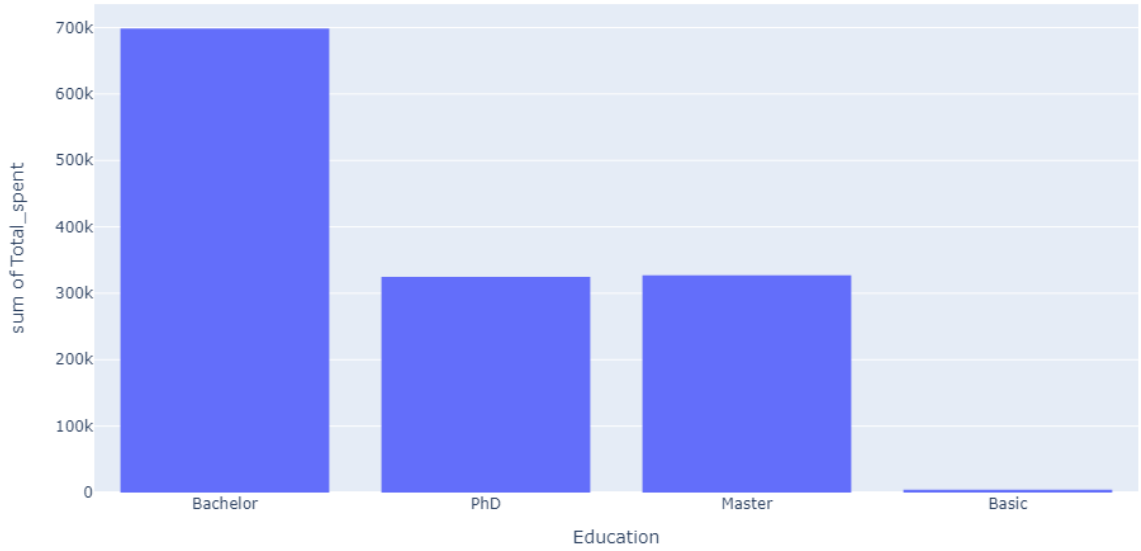
There doesn't seem to be any clear relationship between age of customers and their spending habits.



- Cinema and Meats products are the most famous products among the customers.
- Sweets and Fruits are not being purchased often.

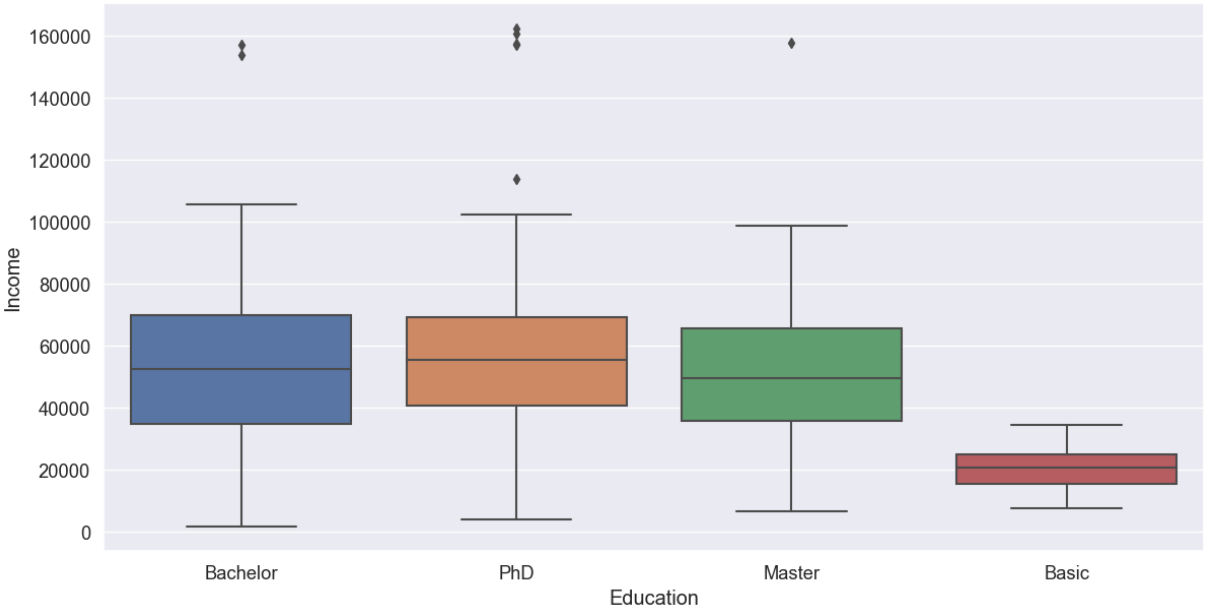
Questions.

1- How much are the total expenses spent by customers according to education?



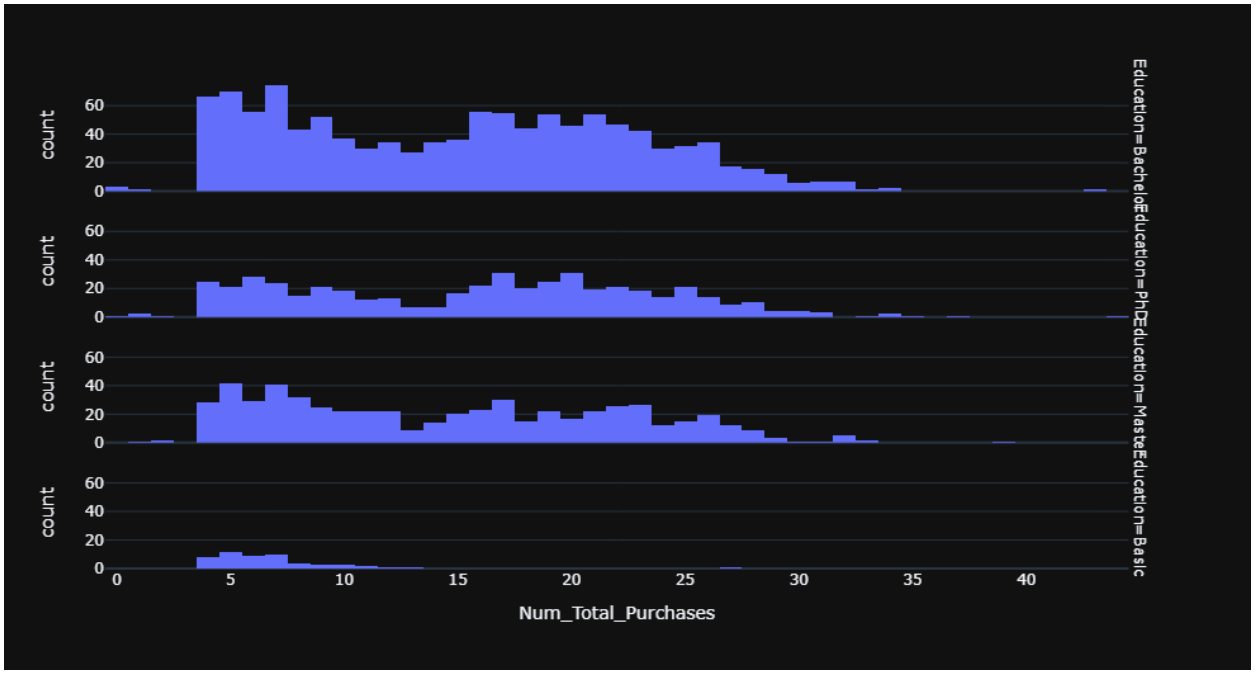
In this figure, it is clear that graduates are the largest of their total expenditures, and those with a master's degree come in the second degree.

2- Who are the clients who have the highest income according to the educational level?



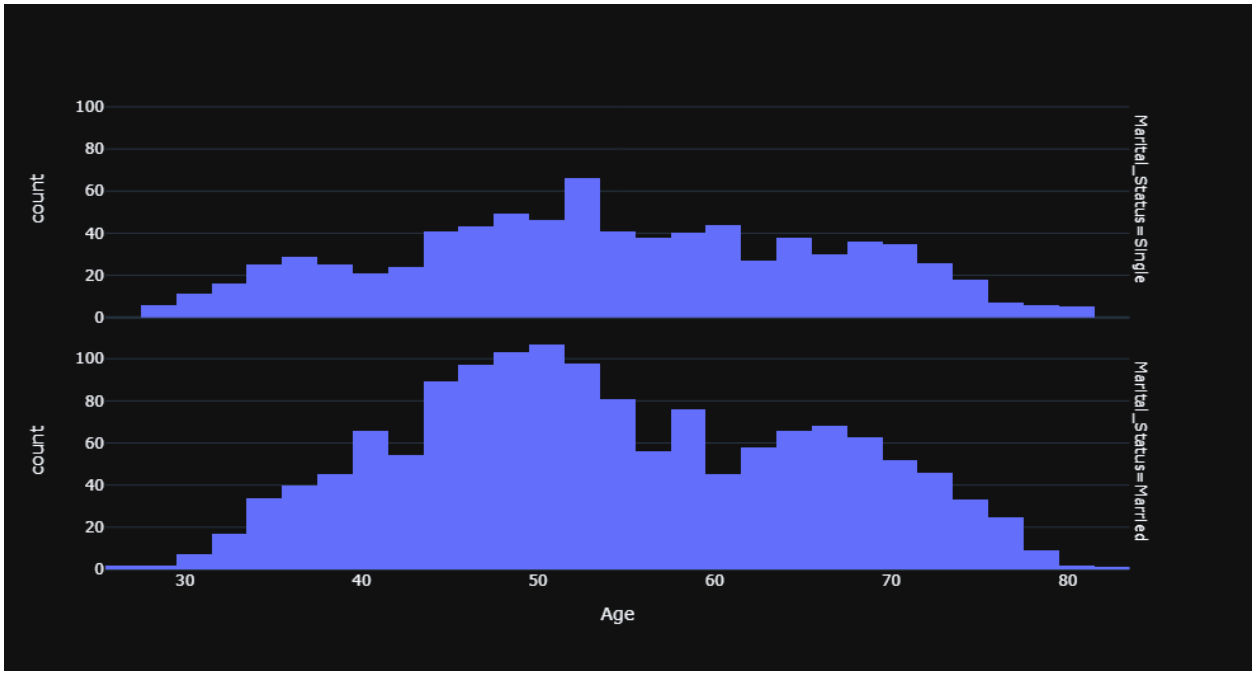
As we show in this figure almost most graduate clients with a master's degree have the same income

3-What is the total customer exchange rate according to education?



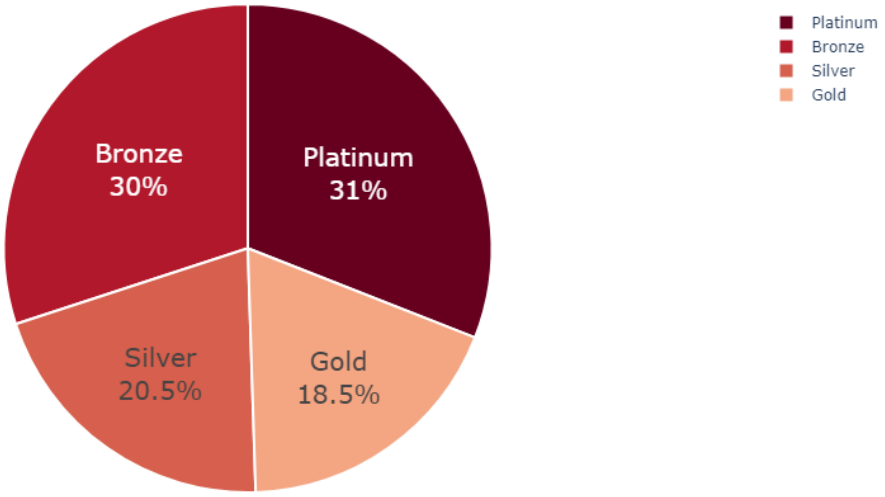
It is an expected percentage because the number of graduate clients is more than the rest.

4- What are the ages of clients according to gender?



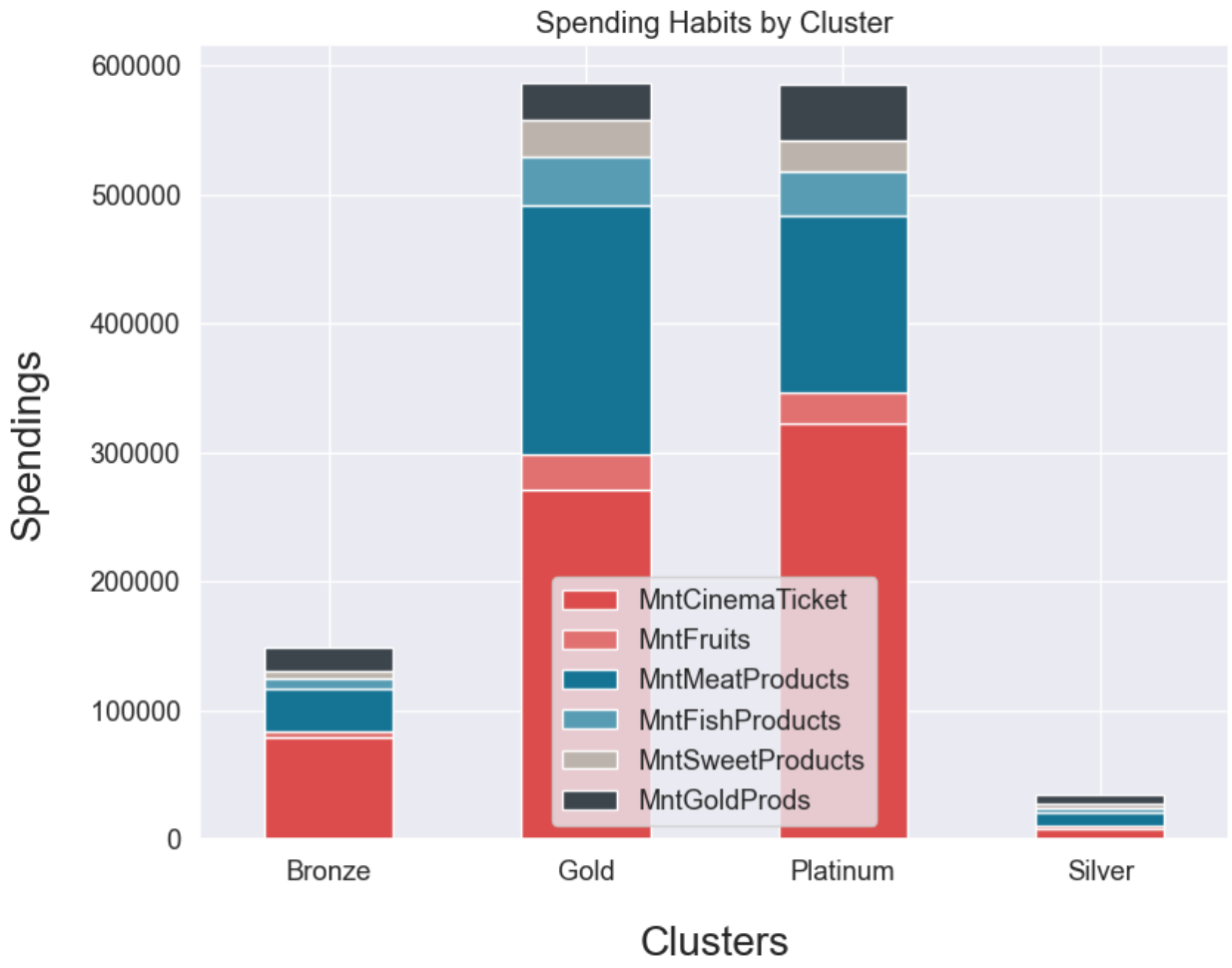
After use Machine Learning Model (K-Means).
Customers were divided into four categories.

- Bronze
- Platinum
- Silver
- gold



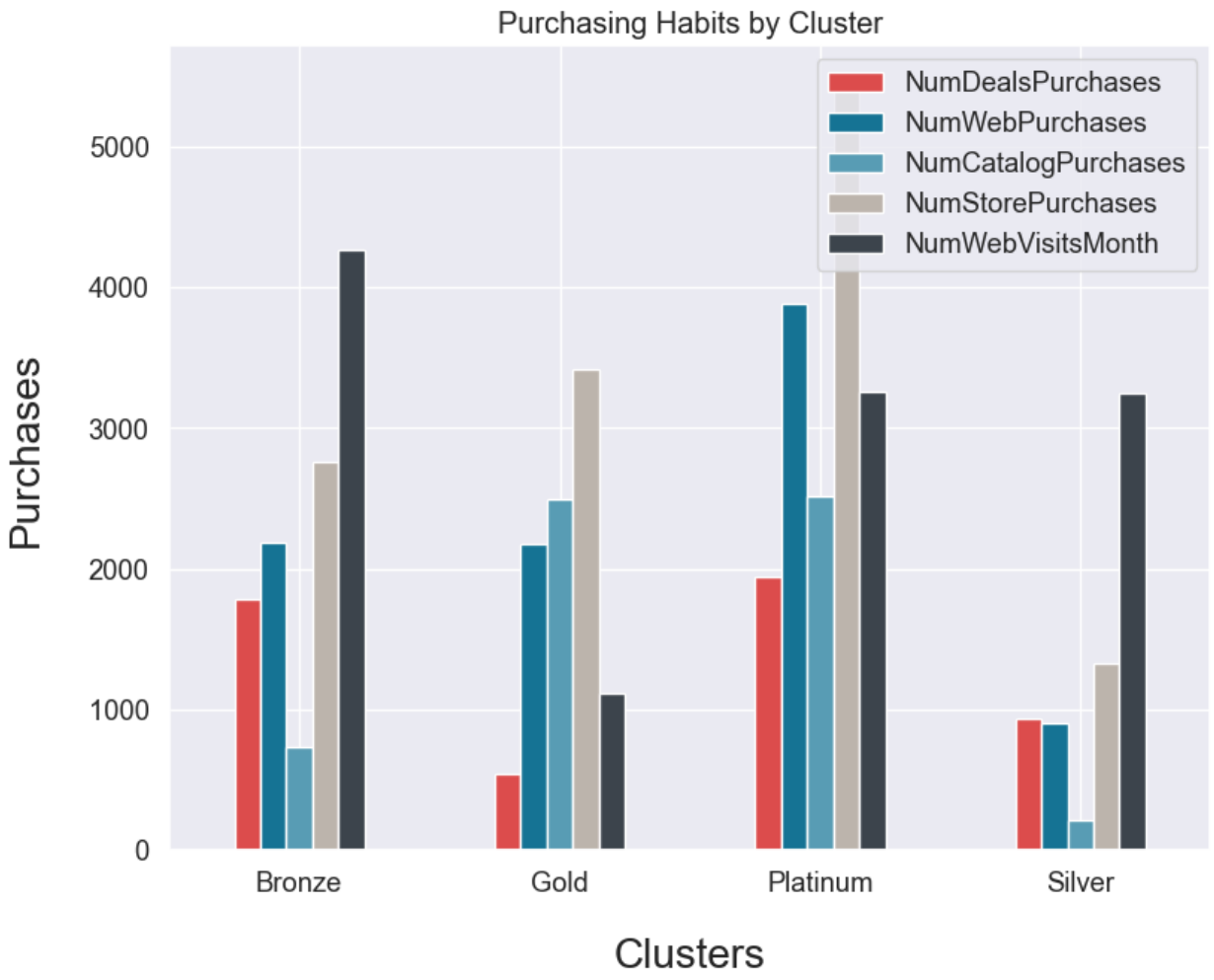
INSIGHTS

- Most of the customers are in the Bronze and platinum category, about 30% and 31% respectively
- Silver is the 3rd popular customer category with 20.5% while only 18.5% occupy the Gold category



INSIGHTS

-Customers from all segments have spent the majority of their money on Cinema and Meat



INSIGHTS

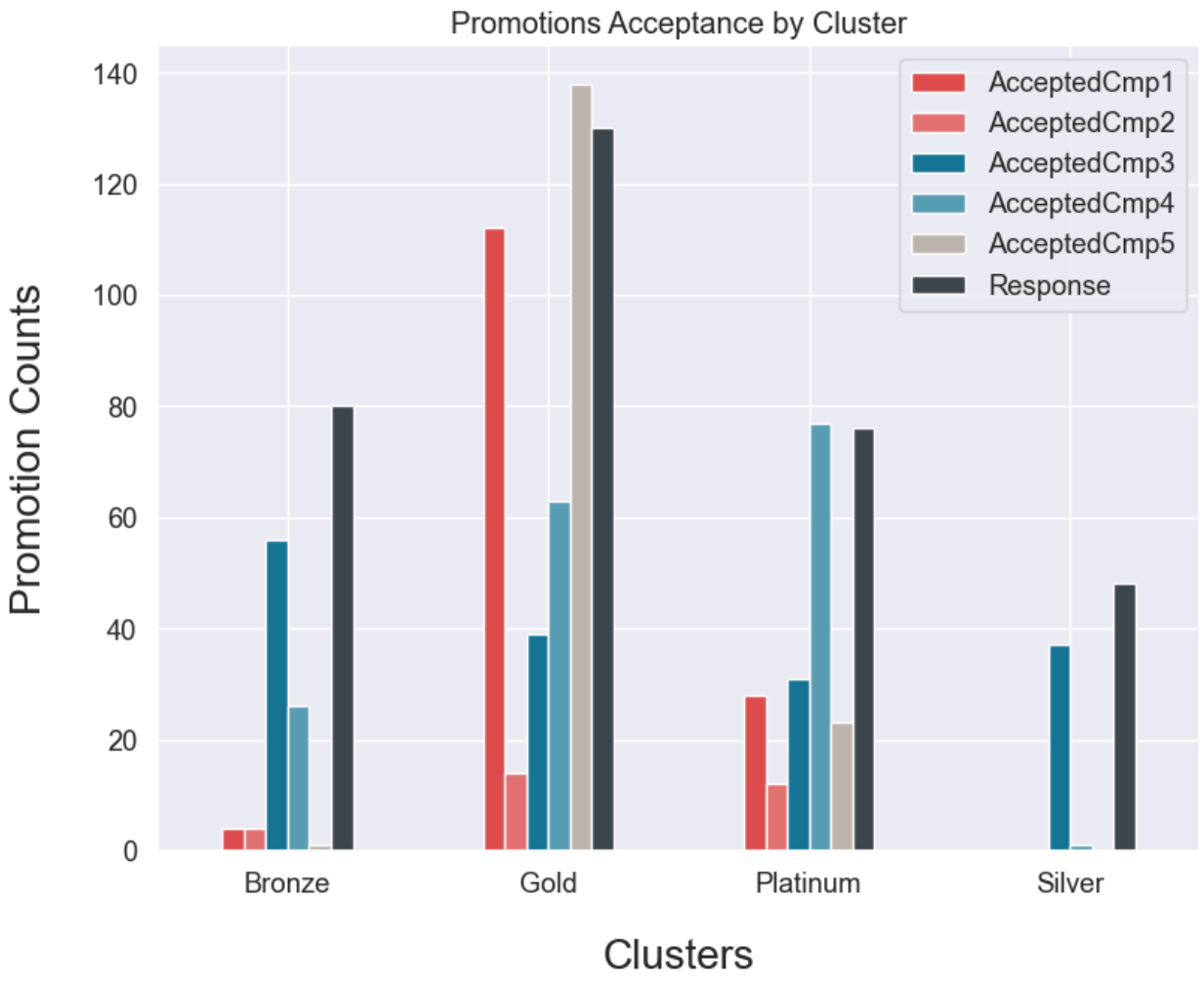
-Platinum and Gold customers are mostly likely to make in-store purchases

-Most web and catalog purchases are also made by customers from the Platinum and Gold segments

-Silver and Gold categories also like to buy from the shop

-Deal purchases are common among Gold and Silver customers

-Silver category customers make the most web visits while customers from the Platinum segment have the least web visits



INSIGHTS
-Platinum subscribers receive most of the offers from the company
-Campaigns 1, 5 and last seem to be the most successful
-The bronze category shows the least interest in a company's promotional campaign¶

F. Answering Question

What are the statistical characteristics of the customers?
The company's customers are mostly married. There are more Middle Aged Adults, aged between 40 and 60 and most of them like to have one child. Most of the customers hold bachelor degree and their earnings are mostly between 25,000 and 85,000.

What are the spending habits of the customers?
Customers have spent more on wine and meat products. Those without children have spent more than those having children. Singles are spending more than the one's with the partners. Middle aged adults have spent more than the other age groups. Store

shopping is the preferred channel for purchasing among the customers. Web and Catalog purchasing also have potential.

Are there some products which need more marketing?
Sweets and Fruits need some effective marketing. Company needs to run promotions for these products in order to increase the revenue from these products. Baskets of the least selling products combined with the most selling products can be effective.

How the marketing can be made effective?
As a marketing recommendation give coupons to the old and high spending customers. Market the cheap and on-offer products to the low income and low spending customers. Web purchasing has some potential. To unlock this give special discounts to the customers who sign up on company's website.

G. Conclusion

1-Most of the customers are university graduates
2-Most of the customers stay with partners
3-Those who live alone have spent more than those who live with a partner
4-Most of the customers have only one child
5-Those who have no children have spent more
6-Middle Age Adults, aged between 40 and 60 years, is a popular age group category
7-Middle-aged adults spend, on average, more than any other age group
8-Most of the customers earn between 25000 and 85000
9-Cinema and Meat Products are very famous among customers
10-Based on income and total expenses, customers are divided into 4 clusters namely Platinum, Gold, Silver and Bronze
11-Most of the customers fall into the Silver and Gold category
12-Those who earn more also spend more
13-Most customers like to buy from a store and then online from the web
14-Platinum customers show more acceptance of promotional campaigns while bronze subscribers show the least interest