ACKNOWLEDGEMENT

I would like to express my thankfulness towards a number of people who have been instrument in making of this project. First of all I thank the college management who has been with me and provide most helpful facilities throughout the completion of the project. I am greatly indebted to **Rev.Dr. Joshy George CMI, Principal Kristu Jyoti College**, who whole heartedly gave me the permission and whose advice was a real encouragement. I am deeply indebted to **Ms. Tintu Varghese, Project Co-Ordinator and Ms. Soumya Koshy, Project Guide**, for the valuable discussion and helpful counselling. I am sincerely indebted to Asst. Prof. **Mr. Roji Thomas, HOD, Department of Computer Applications**, for the valuable guidance. Last but not least, i wish to express my gratitude and heartfelt thanks to my friends, family and whom with their valuable advice, encouragement and support for the successful completion of the project. Above all I am thanking God Almighty.

Lijo Joseph

INDEX

INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

The ***BookMySnacks*** project is an innovative digital platform engineered to transform the traditional snack-ordering experience in theaters. Designed as a web-based system, it empowers moviegoers to pre-order snacks seamlessly from their seats, eliminating long queues and enhancing the overall cinematic experience. By creating a synchronized ecosystem for users, shop administrators, theater managers, and system administrators, BookMySnacks introduces operational efficiency, real-time ordering, and frictionless service delivery.
This system leverages intelligent module separation and real-time data flow to ensure that every stakeholder—be it a user placing an order or a shop admin managing inventory—operates within a fluid, intuitive environment. With a strong focus on user experience, speed, and accuracy, BookMySnacks addresses a major pain point in the entertainment industry: managing snack demand during intermissions.

Through its secure authentication process, smart seat-linked ordering, and modular control panels for shops and theaters, the system guarantees a streamlined, enjoyable experience. The real-time visibility into orders, seat locations, and showtimes ensures precise delivery and efficient management, while reducing congestion and human error.

Each module—User, Shop Admin, Theater, and Super Admin—has been architected to play a specific role in this connected ecosystem:
The User Module allows customers to browse, select, and purchase snacks based on their theater and seat, all before or during the show.

The **Shop Admin Module** enables counter staff to manage orders, update menus, and track delivery status in real-time.

The **Theater Module** manages theater-specific data, including show schedules and seat mappings.

The **Super Admin Module** ensures system-wide governance, onboarding, and data integrity across all entities.

Together, these modules form the backbone of BookMySnacks—enabling faster service, greater convenience, and a reimagined theater experience that aligns with modern digital expectations.

1.2 PROBLEM STATEMENT

Snack purchasing during movie intermissions is traditionally a chaotic and inefficient process. Long queues, delayed service, and manual order handling often result in frustrated customers, missed movie moments, and overwhelmed staff. The absence of a centralized, pre-ordering system makes it difficult to manage high-volume demand within limited break times, leading to operational bottlenecks and subpar user experiences.

The ***BookMySnacks*** project addresses these challenges by introducing a web-based platform that streamlines snack ordering through digital pre-booking, real-time request handling, and seat-linked delivery. By replacing manual operations with automated workflows and user-friendly interfaces, the system significantly improves efficiency, reduces crowding, and enhances customer satisfaction for both users and theater vendors.

1.3 SCOPE AND RELEVANCE OF THE PROJECT

The ***BookMySnacks*** project aims to transform the theater concession experience by delivering a fast, seamless, and contactless snack-ordering system. Through its intuitive, web-based platform, the system enables users to pre-order snacks linked to their theater seat and showtime, ensuring swift delivery without disrupting their movie experience. The platform minimizes human error through input validation and provides clear feedback, making it reliable and accessible even to non-technical users.

The scope encompasses four core modules—*User*, *Shop Admin*, *Theater*, and *Super Admin*—that work together to create a unified, responsive environment. Users can browse menus, place orders, and track deliveries with ease. Shop Admins manage real-time order queues and inventory. Theater Admins handle scheduling, seating, and shop linkage. Super Admins oversee all data, user verification, and operational integrity across the platform.

By automating the snack-ordering workflow and offering real-time coordination, *BookMySnacks* addresses the inefficiencies of manual operations. It enhances user convenience, reduces queue congestion, and optimizes service delivery, making it a highly relevant and future-ready solution for modern cinema chains.

1.4 OBJECTIVES

The objective of this project is to develop an online snack ordering and management system, ***BookMySnacks***, designed to address and eliminate the inefficiencies of traditional, manual concession operations in movie theaters. This platform aims to streamline the entire workflow by automating key tasks such as snack pre-ordering, seat-based delivery coordination, real-time inventory management, and admin oversight.

The system is tailored to ensure smooth interaction between users, shop admins, theater staff, and super admins, offering a user-friendly and efficient solution. By incorporating real-time updates, contactless ordering, and secure data handling, *BookMySnacks* enhances the overall moviegoing experience and service delivery, ultimately contributing to a faster, safer, and more convenient cinema environment.

SYSTEM ANALYSIS

CHAPTER - 2

SYSTEM ANALYSIS

2.1 INTRODUCTION

The first stage of any project, sometimes called the preliminary assessment is brief investigation of the system under consideration, system study and analysis deals with the study of the current system, this is the critical process of information development. It can be defined as problem solving which consist of four phases that can be successfully completed by applying appropriate skill and carefully addressing each dimension of the information system.

The purpose of preliminary study phase is to determine the initial feasibility of a project work. The product of the phase is a feasibility survey that is presented to a steering committee for a decision on whether the project should be developed.

After feasibility analysis, the next phase is the study of the current system. The purpose of this phase is to learn how the current system operates. The analyst identifies the problems, limitations and constraints forms preliminary solutions finally. The analyst updates the feasibility estimates and presents the findings as a problem statement for final study of phase reports.

The third phase of the system analysis is to define end-user requirements for a new system. The purpose of this phase is to identify what the new and improved information system must be able to do. The product of this phase is the requirement statement.

The fourth phase to select a feasible solution from alternatives that are evaluated in terms of operational, technical, and economic feasibility the analyst will recommend the best solution to the management for approval. A cost benefit analysis determines the expected system development lifetime, cost for a new system and the benefits of the new system.

2.2 EXISTING SYSTEM

Managing blood donation and fulfilling blood requests is often a cumbersome process due to the lack of an organized and centralized system. Currently, finding blood donors relies heavily on manual methods such as contacting individuals or relying on outdated registries, which is time-consuming and inefficient. Hospitals and patients often struggle to locate suitable donors in critical situations, leading to delays that can have serious consequences.

Verification of donors, patients, and requests is inconsistent, increasing the risk of errors and misuse. Additionally, there is no streamlined way to track donations, requests, or donor availability. The lack of a user-friendly platform makes it challenging for donors, patients, and hospitals to interact seamlessly. To address these challenges, we propose *Blood Connect*, an automated system that simplifies the process, enhances reliability, and ensures timely access to blood resources.

2.2.1 LIMITATIONS OF EXSTING SYSTEM

- Reliance on manual processes causes delays and inefficiencies in managing requests.
- Lack of design integrity.
- Inconsistent verification raises safety and authenticity concerns.
- Poor record management leads to incomplete or outdated information.
- Snacks selection from multiple stores may makes confusion

2.3 PROPOSED SYSTEM

The proposed system, ***BookMySnacks***, is a web-based platform designed to eliminate the inefficiencies of traditional snack purchasing during movie intermissions. It offers a centralized, pre-ordering system that facilitates smooth interaction between users, shop admins, theater admins, and the super admin, ensuring a seamless experience before the show or during breaks.

Users can explore theater-linked snack menus, place pre-orders for their selected items, and receive digital tokens or order IDs for quick pickup from designated counters. Shop Admins manage order processing, inventory updates, and availability in real time. Theater Admins handle shop assignments, showtime linking, and overall coordination. Super Admins monitor platform health, verify registrations, and ensure system-wide integrity.

Future enhancements include real-time order status notifications, queue-time estimations, and dynamic menu updates based on availability. These features will reduce waiting time, avoid overcrowding, and modernize the snack-ordering experience—offering users convenience without disrupting their moviegoing flow.

ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features:

- Streamlined Process.
- User-Friendly Interface.
- Real-Time Accessibility.
- Efficient Record Management.
- Enhanced Security.

2.4 FEASIBILITY STUDY

The main objective of the feasibility study is to test the Technical, Operational and Economic feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. The feasibility study is undertaken to determine the possibility of either improving the present system or developing a completely new system.

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

2.4.1 TECHNICAL FEASIBILITY

Generally, new system brings new technology into an organization. The proposed system requires technology and equipment, which can be obtained. The operating system has the technical capacity to hold the data required to use the proposed system. The present equipment technology assures technical guarantee of accuracy, reliability, and ease of access.

2.4.2 OPERATIONAL FEASIBILITY

The operational feasibility of the *Blood Connect* project is essential for ensuring it meets user needs and integrates effectively into existing workflows. Key considerations include:

- ***User Support and Management Buy-in*****:** Strong support from management and users is crucial. Stakeholder engagement sessions will gather input and ensure alignment with user requirements, fostering acceptance of the system.
- **System Usability and Functionality:** The platform is designed with a user-friendly interface for individuals with varying technical expertise. Comprehensive testing will ensure it operates effectively in real-world scenarios.
- **Resistance to Change:** To minimize resistance, training sessions and informative materials will be provided, emphasizing the system's benefits. Engaging users throughout the development process will help facilitate a smooth transition from manual processes.

By addressing these aspects, *BookMySnacks* aims to reduce crowds at theater snack counters while ensuring efficient order management and a smooth customer experience.

2.4.3 ECONOMIC FEASIBILITY

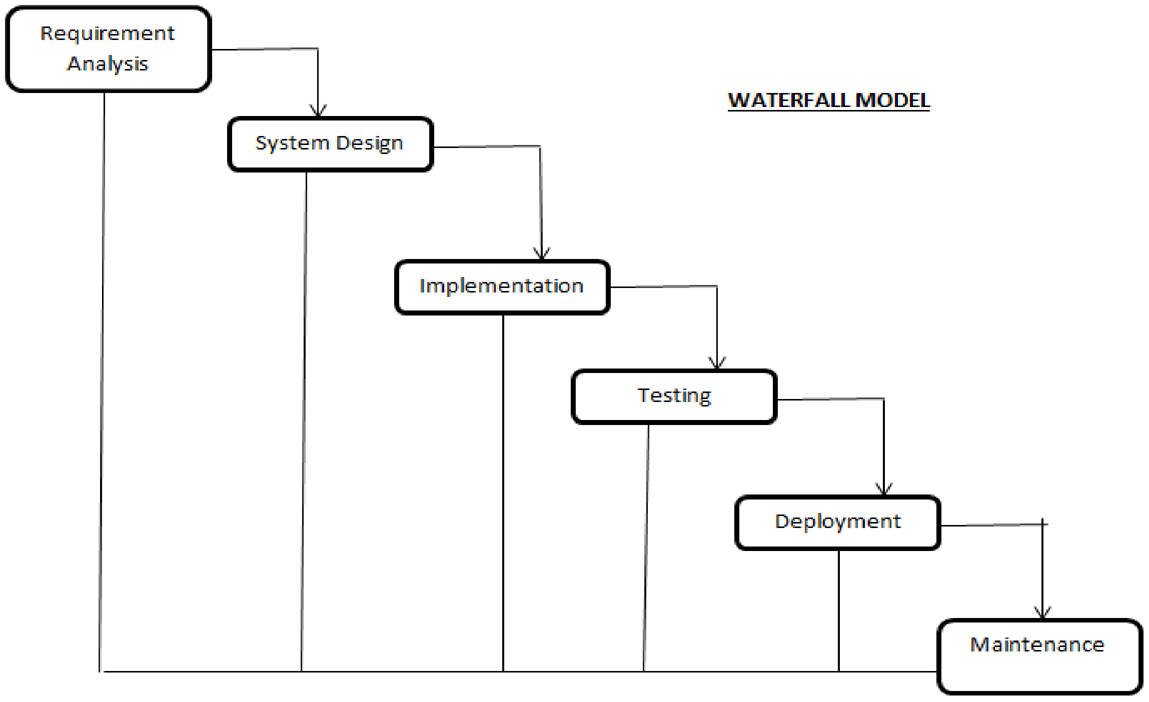
A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any additional hardware or software.

2.5 SOFTWARE ENGINEERING PARADIGM APPLIED

Software process is the way in which we produce the software. Software engineering principles are applied to obtain economically developed software that is reliable and efficient. The chosen software engineering paradigm for “Blood connect” is the Waterfall Model, where the development process is divided into sequential phases. This model ensures a systematic approach to software development, meeting quality, time, and budgetary requirements.

In summary, the proposed bookstore system aims to enhance efficiency, security, and user experience in book transactions, addressing the limitations of the existing manual system. The feasibility study evaluates technical, operational, and economic aspects, and the project follows the Waterfall Model in the software development life cycle.

The following illustration is a representation of the different phrases of the Waterfall Model:


Requirement Analysis
WATERFALL MODEL
System Design
Implementation
Testing
Deployment
Maintenance

SYSTEM DESIGN

CHAPTER 3
SYSTEM DESIGN

3.1 INTRODUCTION

System design is the solution to the creation of a new system. This phase is composed of several systems. This phase focuses on the detailed implementation of the feasible system. Its emphasis on translating design specification to performance specification. System design has two phases of development logical and physical design.

During logical design phase the analyst describes inputs(sources),

Outputs(destination), databases (data stores) and procedures (data flows) all in a format that meats the uses requirements. The analyst also specifies the user needs and level that virtually determines the information flow into and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design.

The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specification, which tell the programmers exactly what the candidate system must do. The programmers write the necessary programs that accepted data through call and produce the required report on a hard copy or display it on the screen.

3.2 DATABASE DESGIN

Database design is the process of producing a detailed data model of a database. The logical model contains all the needed logical and physical design and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for entity. The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system.

NORMALIZATION

The process of normalization is concerned with the transformation of the conceptual schema to a computer represent able form. Normalization reduces the redundancies and anomalies.

THE FIRST NORMAL FORM

First normal form does not allow multi valued and composite valued attributes. It states that the domain of an attribute must include only atomic values and that value of any attribute in a table must be single value form the domain of that attribute.

THE SECOND NORMAL FORM

In second normal form, for relation where primary key contains multiple attributes, on key attribute should not be functionally dependent on a part of the primary key.

THE THIRD NORMAL FORM

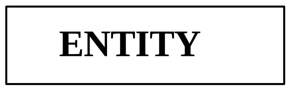
In Third normal form, relation should not have a non-key attribute functionally determined by non-key attribute. That is there should be no transitive dependency of a non-key attribute on the primary key.

3.2.1 ENTITY RELATIONSHIP MODEL

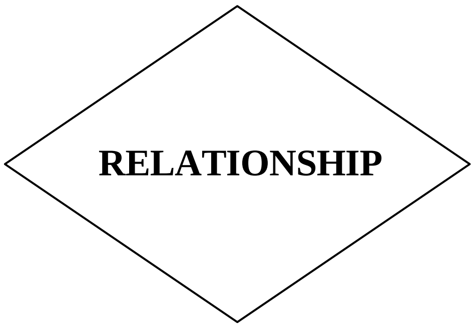
Database designs also includes ER (Entity-Relationship model) diagrams. An ER diagram is a diagram that helps to design databases in an efficient way. Attributes in ER diagrams are usually modelled as an oval with the name of the attribute, linked to the entity or relationship that contains the attributes. Within the relational model, the final step can generally be broken down into two further steps that of determining what are the basic objects about which information, is being stored and then determining the relationships, or objects. This step is not necessary with an object database.

There are 5 main components of an ERD:

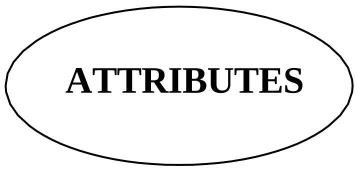
- **Entities,** which are represented by rectangles. An entity is an object or concept about which you want to store information. A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.



- **Relationships,** which are represented by a diamond shapes, show how to entities share information in the database. In some supervise other employee.



- **Attributes,** which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, employee’s social security number might be the employee’s key attribute.



- **Connecting lines,** solid lines that connect attributes to show the relationships of entities in the diagram.

Steps involved in creating an ERD include:

1. Identifying and defining the entities.
2. Determining all interactions between the entities.
3. Analysing the nature of interactions/determining the cardinality of the relationship.
4. Creating the ERD.

E-R diagram shows the different entities in the system and the relationship among the different entities. Each entities have its own attributes associated with it; the relationship shows the association among the different entities.

ER Diagram

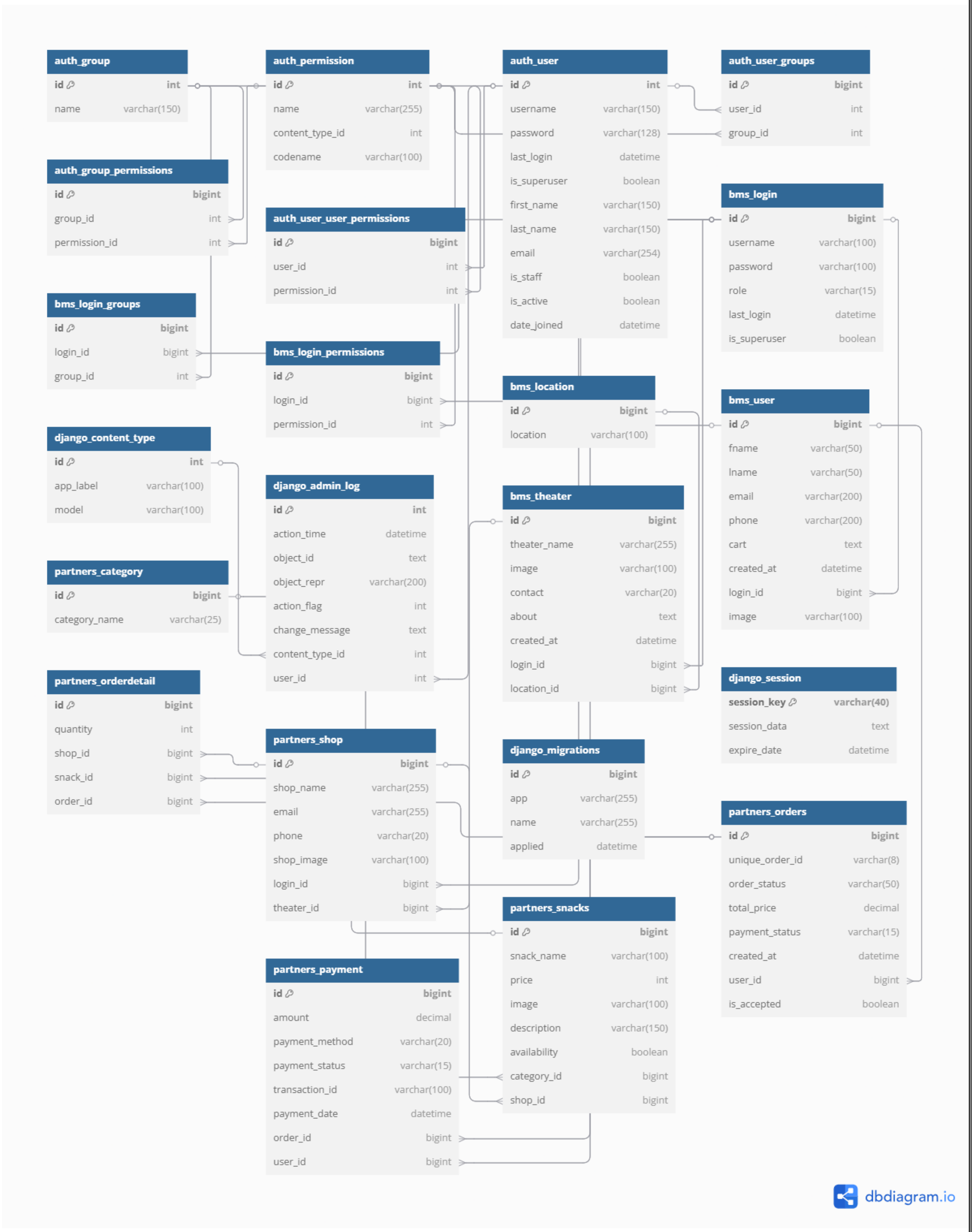

auth_group
id int
name varchar(150)
auth_permission
id int
name varchar(255)
content_type_id int
codename varchar(100)
auth_user
id int
username varchar(150)
password varchar(128)
last_login datetime
is_superuser boolean
first_name varchar(150)
last_name varchar(150)
email varchar(254)
is_staff boolean
is_active boolean
date_joined datetime
auth_user_groups
id bigint
user_id int
group_id int
auth_group_permissions
id bigint
group_id int
permission_id int
auth_user_user_permissions
id bigint
user_id int
permission_id int
bms_login
id bigint
username varchar(100)
password varchar(100)
role varchar(15)
last_login datetime
is_superuser boolean
bms_login_groups
id bigint
login_id bigint
group_id int
bms_login_permissions
id bigint
login_id bigint
permission_id int
bms_location
id bigint
location varchar(100)
bms_user
id bigint
fname varchar(50)
lname varchar(50)
email varchar(200)
phone varchar(200)
cart text
created_at datetime
login_id bigint
image varchar(100)
django_content_type
id int
app_label varchar(100)
model varchar(100)
django_admin_log
id int
action_time datetime
object_id text
object_repr varchar(200)
action_flag int
change_message text
content_type_id int
user_id int
bms_theater
id bigint
theater_name varchar(255)
image varchar(100)
contact varchar(20)
about text
created_at datetime
login_id bigint
location_id bigint
partners_category
id bigint
category_name varchar(25)
django_session
session_key varchar(40)
session_data text
expire_date datetime
partners_orderdetail
id bigint
quantity int
shop_id bigint
snack_id bigint
order_id bigint
partners_shop
id bigint
shop_name varchar(255)
email varchar(255)
phone varchar(20)
shop_image varchar(100)
login_id bigint
theater_id bigint
django_migrations
id bigint
app varchar(255)
name varchar(255)
applied datetime
partners_orders
id bigint
unique_order_id varchar(8)
order_status varchar(50)
total_price decimal
payment_status varchar(15)
created_at datetime
user_id bigint
is_accepted boolean
partners_snacks
id bigint
snack_name varchar(100)
price int
image varchar(100)
description varchar(150)
availability boolean
category_id bigint
shop_id bigint
partners_payment
id bigint
amount decimal
payment_method varchar(20)
payment_status varchar(15)
transaction_id varchar(100)
payment_date datetime
order_id bigint
user_id bigint
dbdiagram.io

TABLE DESIGN:

bmsAdmin_login

Field Name	Data Type	Size	Description	Constraints
id	bigint		Primary key	NOT NULL, GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY
username	varchar	100	Login username	NOT NULL, UNIQUE
password	varchar	100	Hashed password	NOT NULL
role	varchar	15	Role (e.g., admin, staff)	NOT NULL
last_login	timestamp with time zone		Last login time	Nullable
is_superuser	boolean		Superuser flag	NOT NULL

bmsAdmin_user

Field Name	Data Type	Size	Description	Constraints
id	bigint		Primary key	NOT NULL, GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY
fname	varchar	50	First name	NOT NULL
lname	varchar	50	Last name	NOT NULL
email	varchar	200	Email address	NOT NULL, UNIQUE
phone	varchar	200	Phone number	NOT NULL
cart	jsonb		Shopping cart data	Nullable
created_at	timestamp with time zone		User creation timestamp	NOT NULL
login_id	bigint		Login reference	NOT NULL, FK -> bmsAdmin_login(id), UNIQUE
image	varchar	100	Profile image path	NOT NULL

bmsAdmin_location

Field Name	Data Type	Size	Description	Constraints
id	bigint		Primary key	NOT NULL, GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY
location	varchar	100	Location name	NOT NULL

bmsAdmin_theater

Field Name	Data Type	Size	Description	Constraints
id	bigint		Primary key	NOT NULL, GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY
theater_name	varchar	255	Theater name	NOT NULL
image	varchar	100	Image path	NOT NULL
contact	varchar	20	Contact number	NOT NULL
about	text		About the theater	NOT NULL
created_at	timestamp with time zone		Creation timestamp	NOT NULL
login_id	bigint		Login reference	NOT NULL, FK -> bmsAdmin_login(id), UNIQUE
location_id	bigint		Location reference	NOT NULL, FK -> bmsAdmin_location(id)

partners_category

Field Name	Data Type	Size	Description	Constraints
id	bigint		Primary key	NOT NULL, GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY
category_name	varchar	25	Snack category name	NOT NULL

partners_orders

Field Name	Data Type	Size	Description	Constraints
id	bigint		Primary key	NOT NULL, GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY
unique_order_id	varchar	8	Unique order ID	NOT NULL, UNIQUE
order_status	varchar	50	Status of the order	NOT NULL
total_price	numeric	10,2	Total price of the order	NOT NULL
payment_status	varchar	15	Payment status	NOT NULL
created_at	timestamp with time zone		Timestamp of order creation	NOT NULL
user_id	bigint		User reference	NOT NULL, FK -> bmsAdmin_user(id)
is_accepted	boolean		Order accepted flag	NOT NULL

partners_orderdetail

Field Name	Data Type	Size	Description	Constraints
id	bigint		Primary key	NOT NULL, GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY
quantity	integer		Quantity ordered	NOT NULL
shop_id	bigint		Shop reference	NOT NULL, FK -> partners_shop(id)
snack_id	bigint		Snack reference	NOT NULL, FK -> partners_snacks(id)
order_id	bigint		Order reference	NOT NULL, FK -> partners_orders(id)

partners_payment

Field Name	Data Type	Size	Description	Constraints
id	bigint		Primary key	NOT NULL, GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY
amount	numeric	10,2	Amount paid	NOT NULL
payment_method	varchar	20	Payment method (e.g., card, cash)	NOT NULL
payment_status	varchar	15	Status of payment	NOT NULL
transaction_id	varchar	100	Transaction ID	Nullable, UNIQUE
payment_date	timestamp with time zone		Payment date and time	NOT NULL
order_id	bigint		Order reference	NOT NULL, FK -> partners_orders(id), UNIQUE
user_id	bigint		User reference	NOT NULL, FK -> bmsAdmin_user(id)

partners_shop

Field Name	Data Type	Size	Description	Constraints
id	bigint		Primary key	NOT NULL, GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY
shop_name	varchar	255	Shop name	NOT NULL
email	varchar	255	Shop email	NOT NULL
phone	varchar	20	Shop phone number	NOT NULL
shop_image	varchar	100	Image path	NOT NULL
login_id	bigint		Login reference	NOT NULL, FK -> bmsAdmin_login(id), UNIQUE
theater_id	bigint		Theater reference	NOT NULL, FK -> bmsAdmin_theater(id)

partners_snacks

Field Name	Data Type	Size	Description	Constraints
id	bigint		Primary key	NOT NULL, GENERATED BY DEFAULT AS IDENTITY, PRIMARY KEY
snack_name	varchar	100	Name of the snack	NOT NULL
price	integer		Price of the snack	NOT NULL
image	varchar	100	Image path	NOT NULL
description	varchar	150	Description	NOT NULL
availability	boolean		Availability status	NOT NULL
category_id_id	bigint		Category reference	NOT NULL, FK -> partners_category(id)
shop_id_id	bigint		Shop reference	NOT NULL, FK -> partners_shop(id)

3.3 PROCESS DESIGN-DATAFLOW DIAGRAM

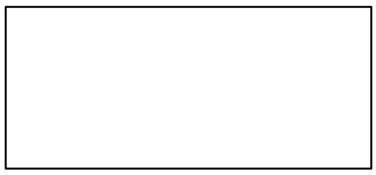
A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system, modelling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing. A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

Data flows

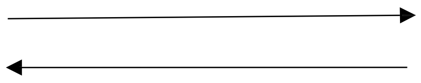
Data flows show the passage of data in the system are represented by the lines joining System components. An arrow indicates the direction of flow and the line is labelled by name of data flow.

Basic data flow diagram symbols are: -

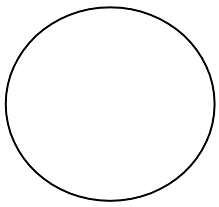
- A rectangle defines a source or destination of the data.



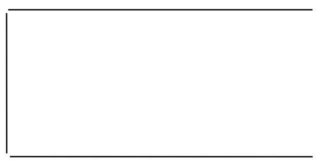
- An arrow identifies data flow, data in motion. It is a pipeline through which information flows.



- A circle or bubble represents a process at transforms incoming data flows into outgoing data flows.



- An open rectangle is a data store.



One of the tools of structured analysis is the diagram. A data flow diagram is a graphical representation of the system. The analyst can use data flow diagram to explain this understanding about the system.

- Data flows are an initiative way of showing how data is processed by a system

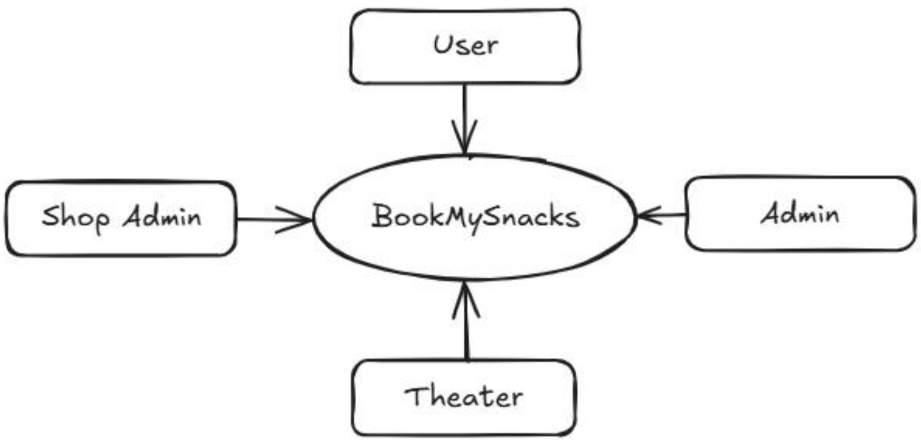
- Data flow models are used to show how data flows through a sequence of processing steps.

Four steps are commonly used to construct a DFD:

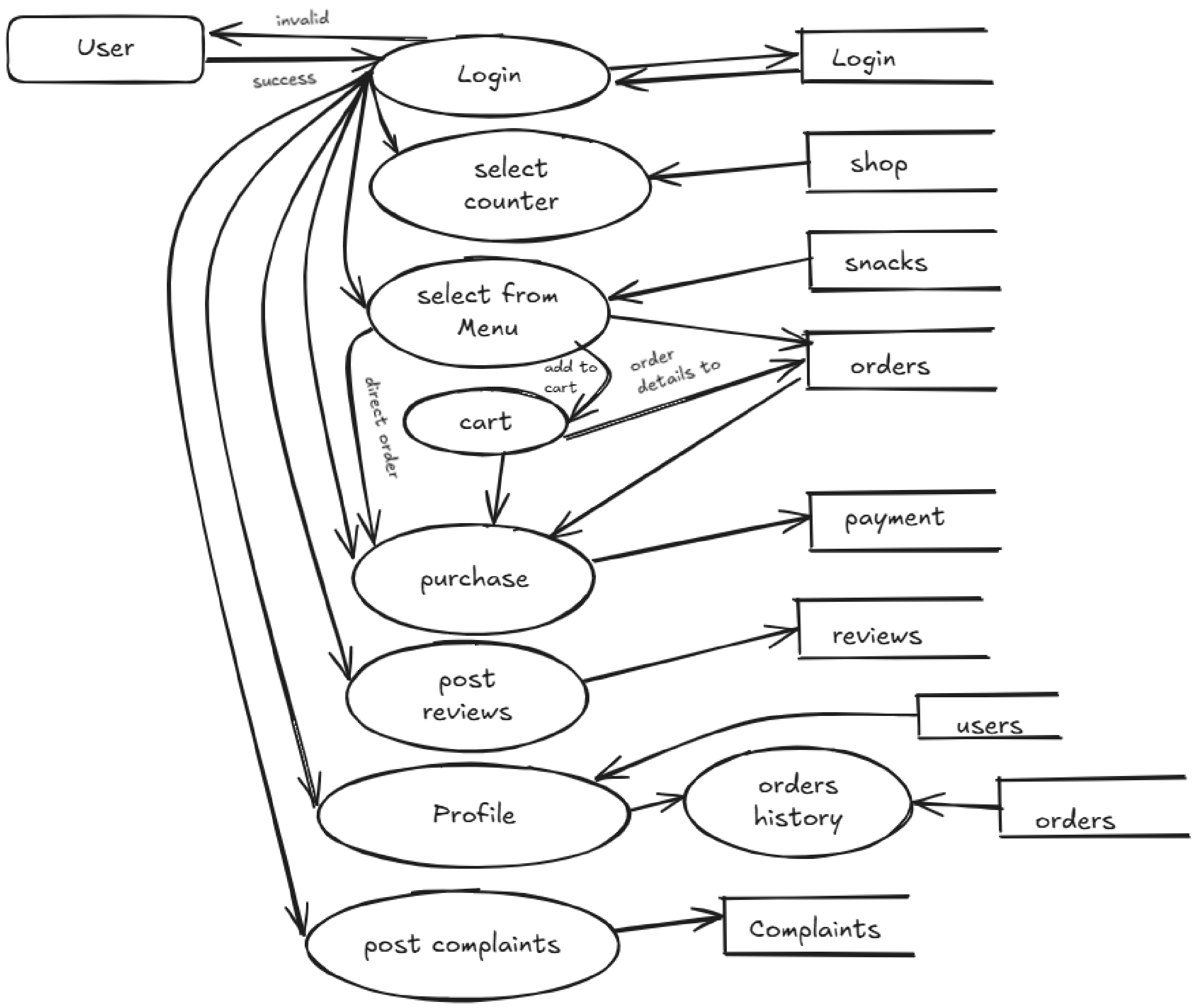
1. Process should be named and numbered for easy references.
2. The direction of flow is from top to bottom and from left to right.
3. When a process is exploded into lower level details, they are numbered.
4. The name of data stores, sources and destinations are written in capital letters.

LEVELS

Level 0

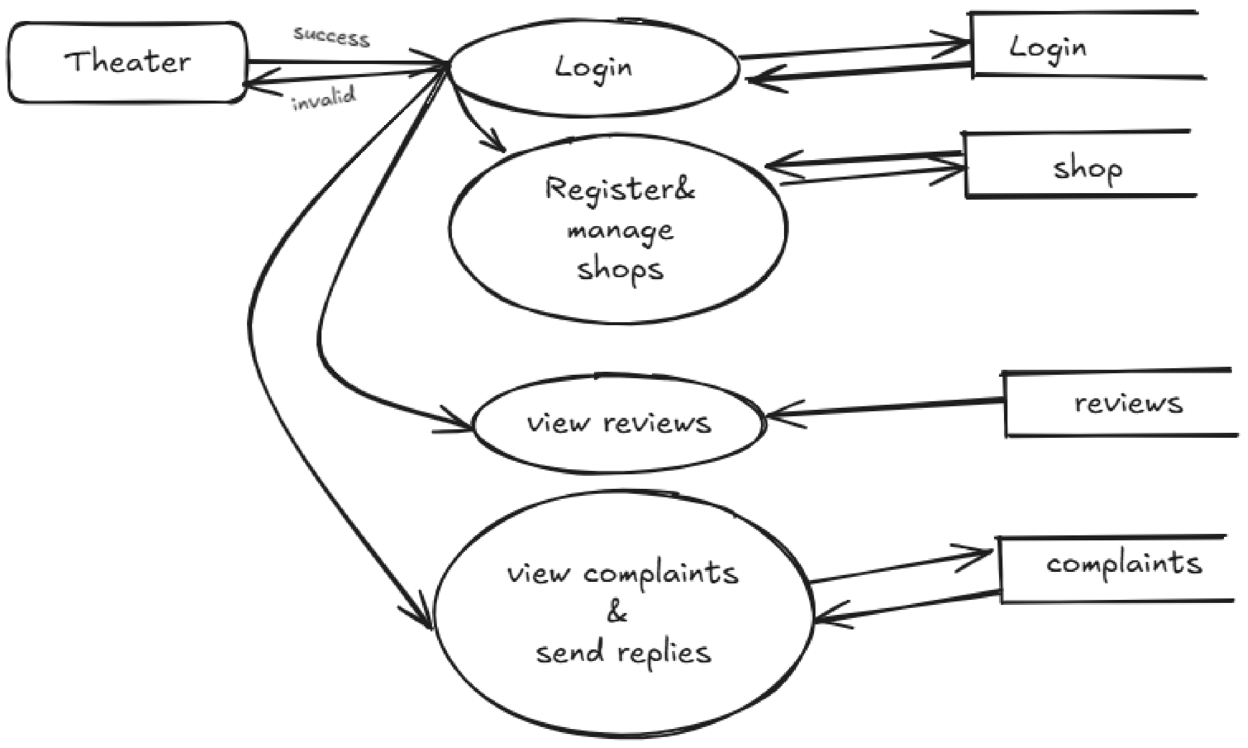


User
Level 1

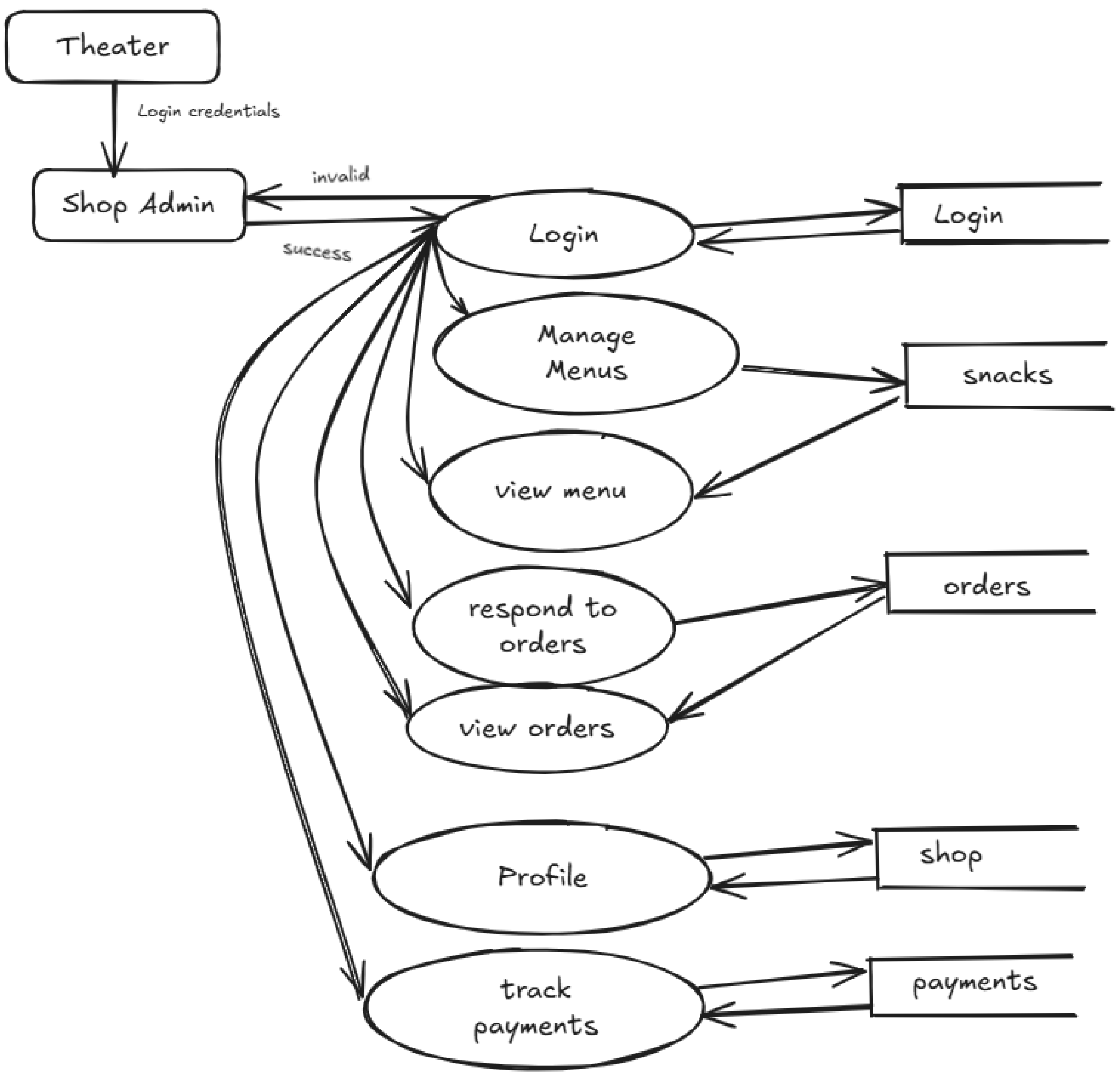

User
invalid
success
Login
Login
select counter
shop
select from Menu
snacks
orders
add to cart
order details to
direct order
cart
purchase
payment
post reviews
reviews
users
Profile
orders history
orders
post complaints
Complaints

Theater

Level 1

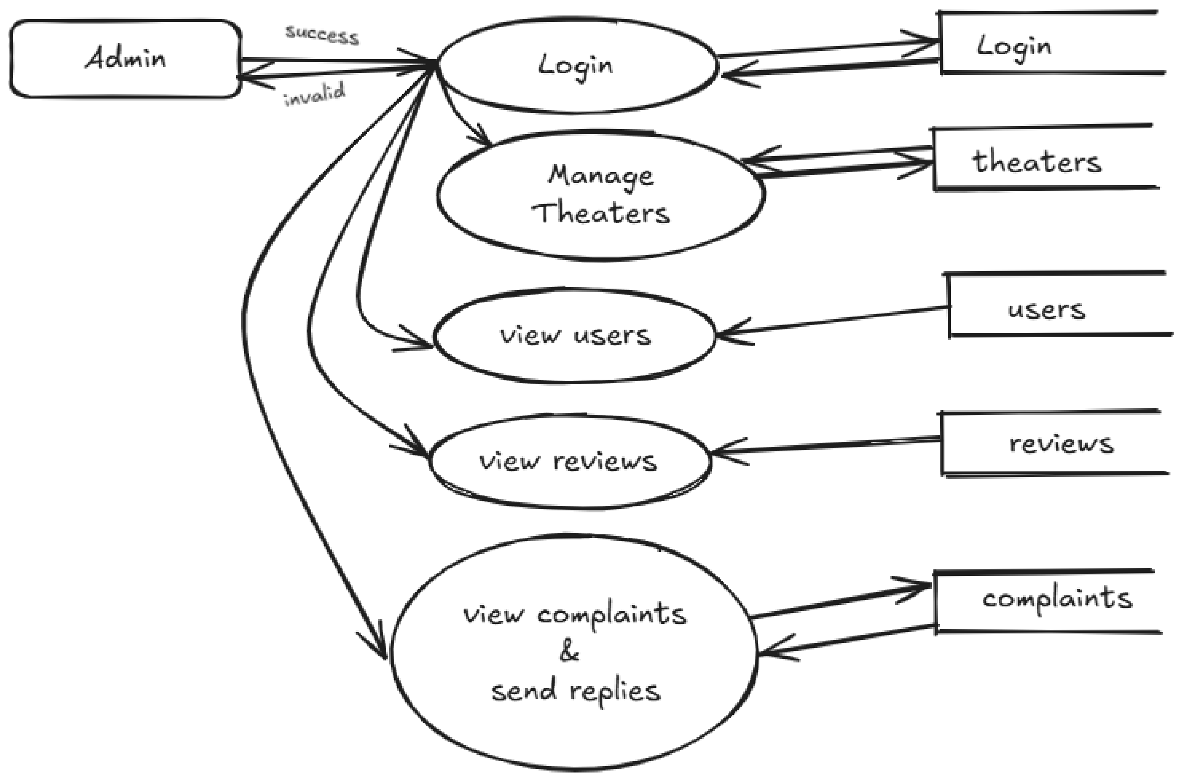


Shop
Level 1


Theater
Login credentials
Shop Admin
invalid
success
Login
Login
Manage Menus
snacks
view menu
respond to orders
orders
view orders
Profile
shop
track payments
payments

Super User

Level 1



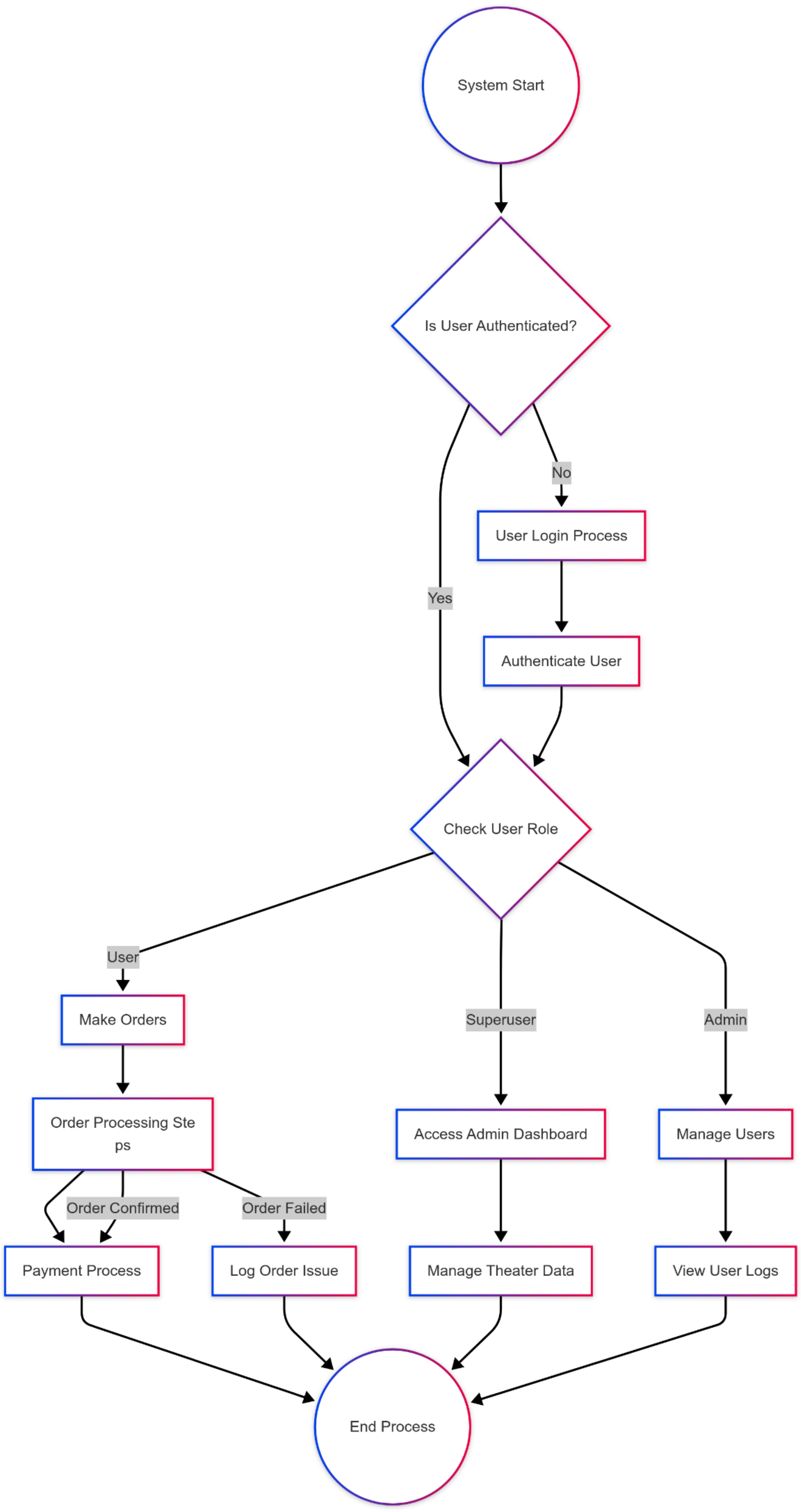
3.4 OBJECT ORIENTED DESIGN-UML DIAGRAMS

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

3.4.1 ACTIVITY DIAGRAM

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.
An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state).

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

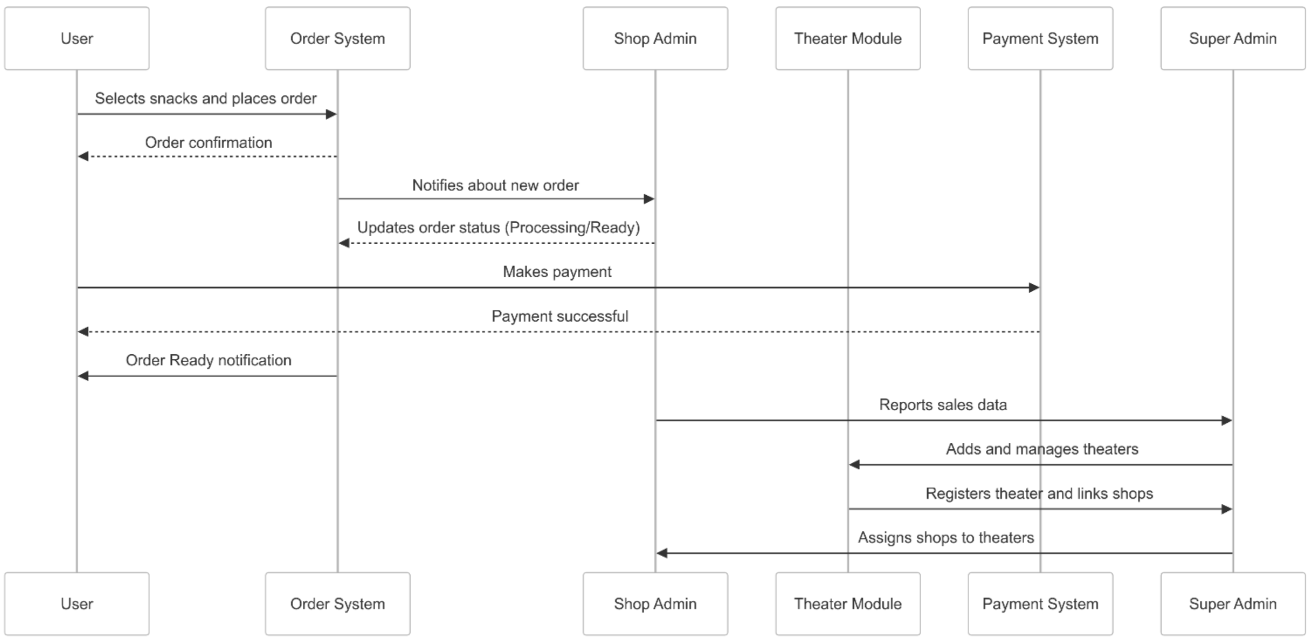

System Start
Is User Authenticated?
No
Yes
User Login Process
Authenticate User
Check User Role
User
Superuser
Admin
Make Orders
Order Processing Ste ps
Access Admin Dashboard
Manage Users
Order Confirmed
Order Failed
Payment Process
Log Order Issue
Manage Theater Data
View User Logs
End Process

3.4.2 SEQUENCE DIAGRAM

The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them. However, an organization's business staff can find sequence diagram useful to communicate how the business currently works by showing how various business objects interact.

An organization's technical staff can find sequence diagram useful in documenting how a future system should behave. During the design phase, architects and developers can use the diagram to force out the system's object interactions, thus fleshing out overall system design.

One of the primary uses of sequence diagram is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams. In addition to their use in designing new systems, sequence diagram can be used to document how objects in an existing system currently interact. This documentation is very useful when transitioning a system to another or organization.

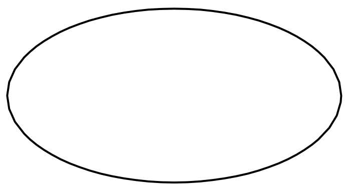


3.4.3 USECASE DIAGRAM

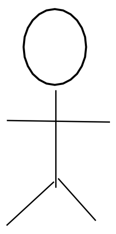
A use case diagram in the unified modeling language (UML) is a type of behavioral diagram defined by and created from a use-case analysis. The main purpose of a use case diagram is to show what system functions are performed for which actor. A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

An actor is a person, organization, or external system that plays a role in one or more interactions with the system. A rectangle is drawn around the use cases, called the system boundary box.

Symbols used in use case are,

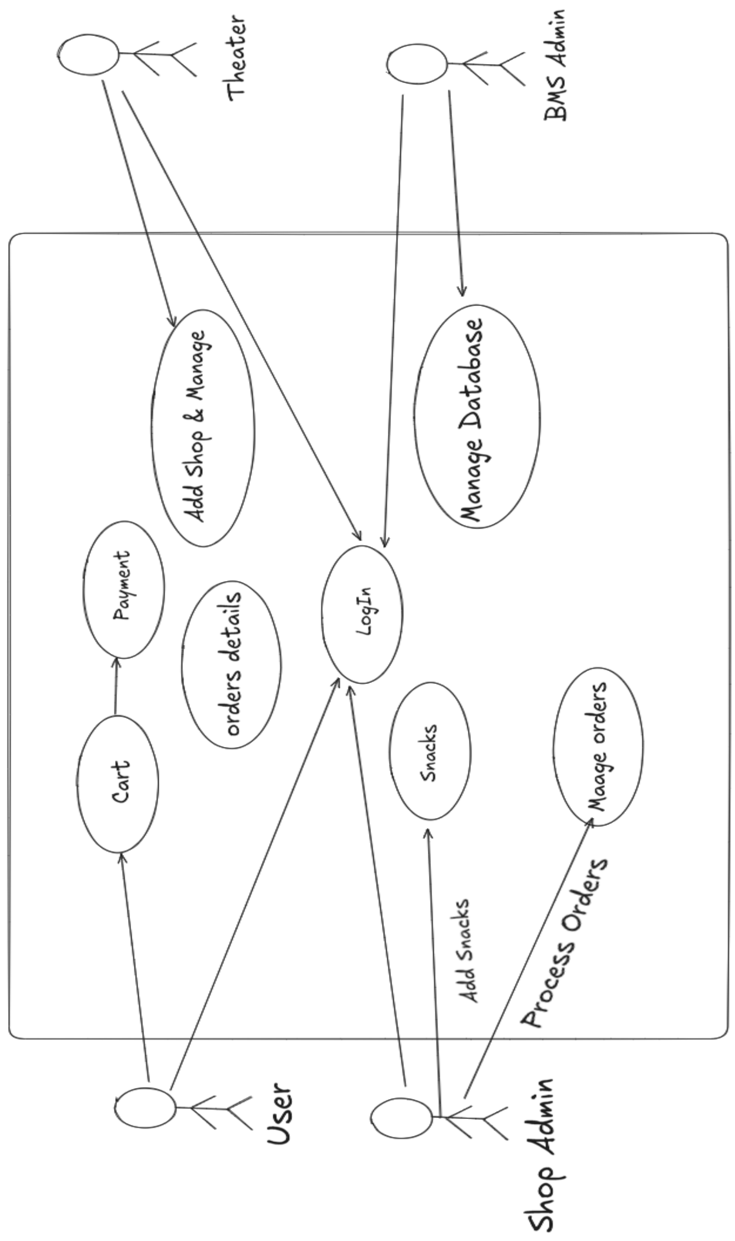


Represent the use case.



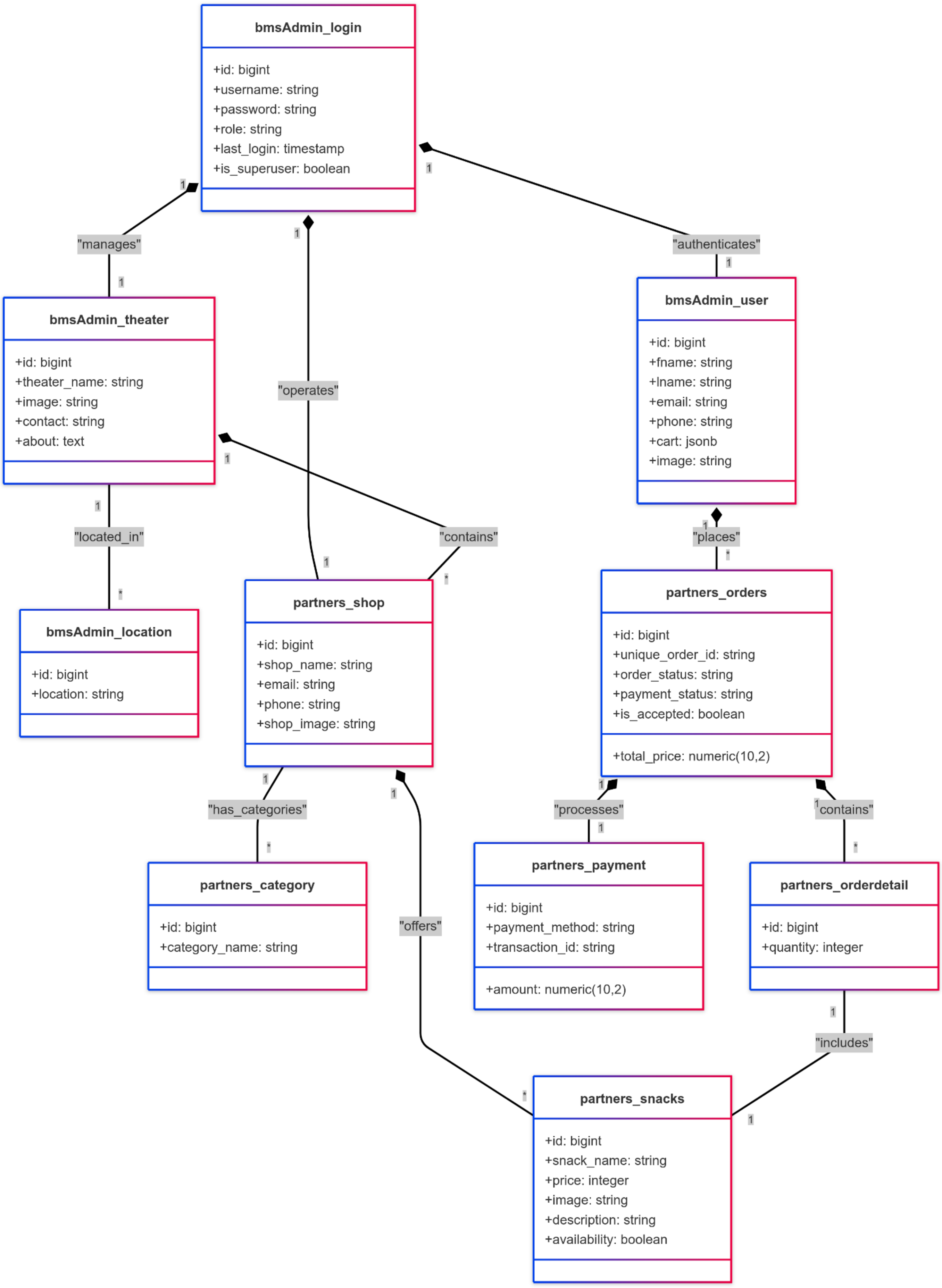
Actors are the entities which interact
With the system.

USE CASE DIAGRAM



3.4.4 CLASS DIAGRAM

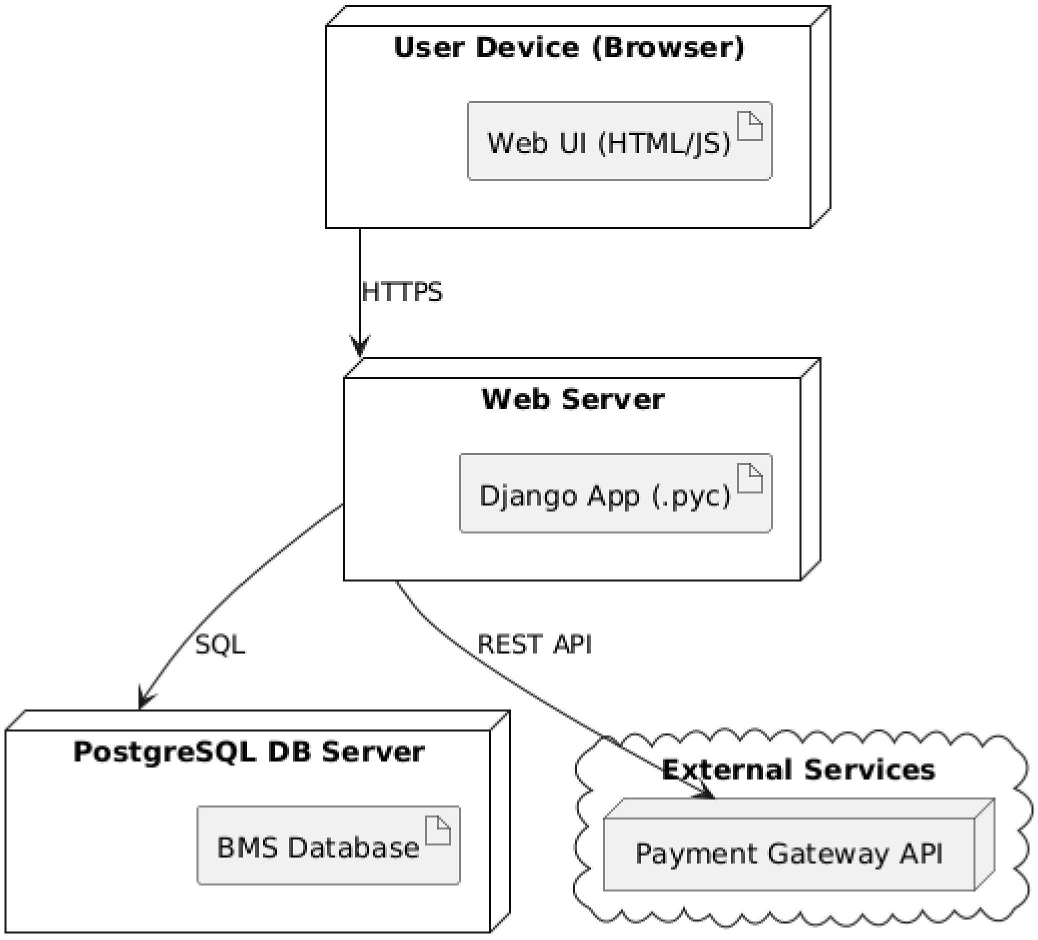
A UML (Unified Modeling Language) Class Diagram is a type of static structure diagram that describes the blueprint of a system by showing its classes, their attributes, methods (operations), and the relationships among the classes. It is widely used in object-oriented design to represent the structure of a software application.


bmsAdmin_login
+id: bigint
+username: string
+password: string
+role: string
+last_login: timestamp
+is_superuser: boolean
"manages"
"operates"
"authenticates"
bmsAdmin_theater
+id: bigint
+theater_name: string
+image: string
+contact: string
+about: text
bmsAdmin_user
+id: bigint
+fname: string
+lname: string
+email: string
+phone: string
+cart: jsonb
+image: string
"located_in"
"contains"
"places"
bmsAdmin_location
+id: bigint
+location: string
partners_shop
+id: bigint
+shop_name: string
+email: string
+phone: string
+shop_image: string
partners_orders
+id: bigint
+unique_order_id: string
+order_status: string
+payment_status: string
+is_accepted: boolean
+total_price: numeric(10,2)
"has_categories"
"processes"
"contains"
partners_category
+id: bigint
+category_name: string
"offers"
partners_payment
+id: bigint
+payment_method: string
+transaction_id: string
+amount: numeric(10,2)
partners_orderdetail
+id: bigint
+quantity: integer
"includes"
partners_snacks
+id: bigint
+snack_name: string
+price: integer
+image: string
+description: string
+availability: boolean

3.4.5 DEPLOYMENT DIAGRAM

A Deployment Diagram in UML (Unified Modeling Language) is a type of diagram that shows the physical architecture of a system. It illustrates how software artifacts (like executables, libraries, or components) are deployed on hardware nodes (such as servers, devices, or networks).
Deployment diagrams are used to model the distribution of software across hardware environments, helping visualize system topology, hardware details, and how software components communicate over the network.

Detailed

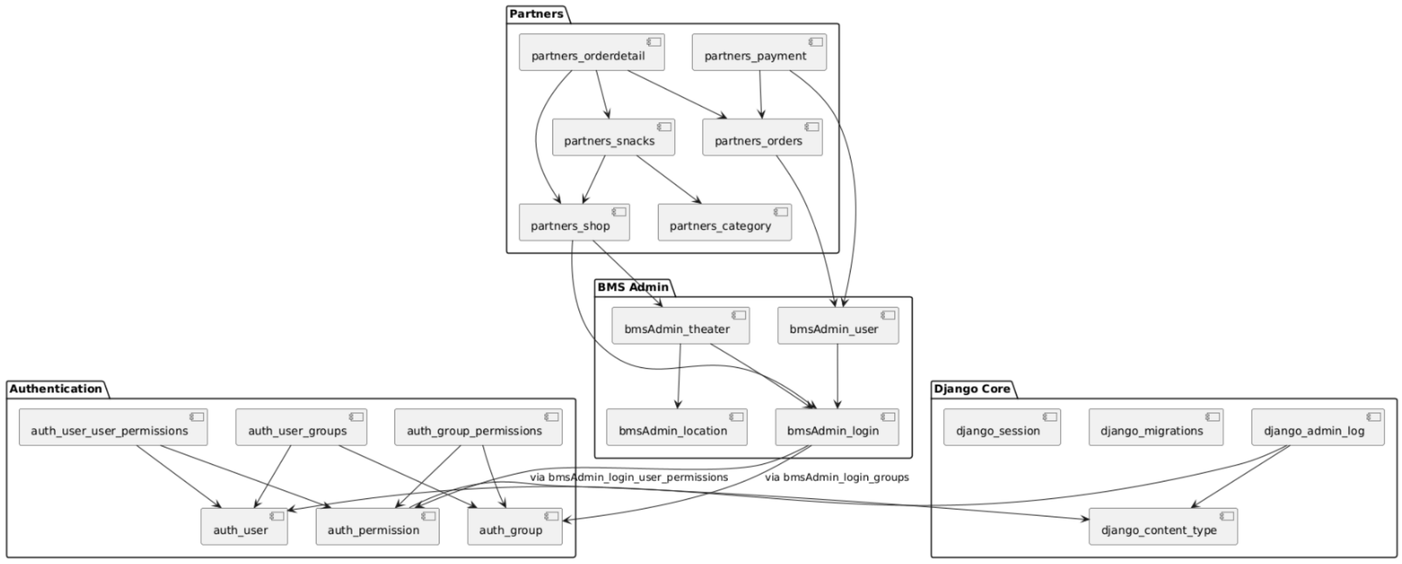


Summarized/original

3.4.6 COMPOUND-BASED DIAGRAM

A Component Diagram in UML (Unified Modeling Language) is used to visualize the organization and dependencies among components in a system. It shows how a system is broken down into modular, reusable, and replaceable components, and how these components interact through interfaces and

connectors.
It is primarily used in high-level system design to model the software architecture, particularly in large, complex applications.



3.5 INPUT DESIGN

Input design is the part of the overall systems, which requires very careful attention. The main objective of the input design is

- To produce a cost-effective +method of input.
- To achieve highest level of accuracy.
- To ensure that the input is acceptable to and understand by the user staff.

The activities to be carried out as part of the overall input processors are as follows:

- Data recording.
- Data transcription.
- Data conversion.
- Data verification.
- Data control.
- Data transmission.
- Data validation.
- Data correction.

One of the early activities of the input design is to determine the nature of the input data. In addition to identify the input types, the analyst needs to consider their impacts on the system as a whole and on other systems.

The analyst should consider the following points when designing the input:

- Nature of the input processing.
- Flexibility and thoroughness of validation rules.
- Handling of priorities with the input procedure.
- Use of composite input documents to reduce the number of different ones.
- Relation with the other system and files.

Careful stages of input after the input medium have been chose involves attention to error handling, batching, and validation procedures.

In the case of this system the error handling has analysed as two major categories.

- User errors.
- System errors.

User errors occur when a user wrongly presses a key or an unsuitable key for a particular operation. These kind of errors are generally non rectifiable but at least can be avoided to greater extend with care and clear attention given when entering data.

System errors occur due to performance of system, hardware malfunctioning etc. These kinds of errors are normally avoided by good maintenance; proper operation and correct procedural follow up. Control and batching comprised of various batching techniques that are used for easy and efficient computing that is, clubbing the necessary program with appropriate functions, usage of needful files and avoiding the usage of unnecessary variables etc.

Basically batching is nothing but the process of joining the cogent order so they are not only meaningful but also easy and effective while operating. The validation procedures are the general error handling characteristics of a system. The basic validation such as non-blank field, non-zero field, is all taken care. The essential validation like master child relationship entries, cascade deletions, deletions only on confirmation are also done as per requirement of the project specification.

3.6 OUTPUT DESIGN

A quality output is one, which meets the requirements of end user and presents the information clearly. In any system result of processing are communicated to the user and to the other system through outputs. In the output design it is determined how the information is to be displayed for immediate need. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationships with the user and helps in decision making.

The objective of the output design is to convey the information of all the past activities, current status and to emphasis important events. The output generally refers to the results and information that is generated from the system. Outputs from computers are required primarily to communicate the results of processing to the users.
At the beginning of the output design various types of outputs such as external, internal, operational, and interactive and turnaround are defined. Then the format, content, location, frequency, volume and sequence of the outputs are specified. The content of the output must be defined in detail. The system analysis has two specific objectives at this stage.

- To interpret and communicate the results of the computer part of a system to the users in a form, which they can understand, and which meets their requirements.
- To communicate the output design specifications to programmers in a way in which it is unambiguous, comprehensive and capable of being translated into a programming language.

SYSTEM ENVIRONMENT

CHAPTER 4

SYSTEM ENVIRONMENT

An important matter in building an application is selecting hardware and software. The hardware drives the software to facilitate solutions. Factors like cost performance and reliability etc are taken into consideration during the purchase of the hardware component for any computerized system.

4.1 INTRODUCTION

Hardware and software requirements for the installation and smooth functioning of this project could be configured based on the requirements needed by the component of the operating environment that works as front-end system here we suggest minimum configuration for the both hardware and software components. Working of with this software is requirements concrete on system environments. It includes two phases:

- Software Requirements
- Hardware Requirements

SOFTWARE REQUIREMENTS:

- Operating System: WINDOWS 10
- Front End: HTML, JavaScript, CSS
- Back End: PostgreSQL
- Web Browser: Brave/MS Edge
- Platform: Visual Studio Code

HARDWARE REQUIREMENTS:

- Processor: Pentium III Processor/above
- RAM: 512 MB RAM
- Hard Disk: 40 GB or above
- Monitor: Standard Colour   Monitor
- Clock speed: 2.24 GHz

4.3 TOOLS, PLATFORM

Python

Python is a high-level, interpreted programming language known for its simplicity and readability. Created by Guido van Rossum in 1991, it supports multiple programming styles and is widely used in web development, data science, automation, and AI. Python’s syntax is clean and easy to learn, making it popular among beginners and professionals. It runs on most operating systems, is open-source, and has a large standard library along with a vast ecosystem of third-party packages. Python is maintained by the Python Software Foundation and is freely available under an open-source license.

Py-Django

Django is a high-level, open-source web framework written in Python, designed to make the development of secure and maintainable websites faster and easier. It was originally developed by Adrian Holovaty and Simon Willison in 2003 and released publicly in 2005. Django follows the Model-View-Template (MVT) architecture and includes many built-in tools such as an admin interface, authentication system, routing, and an Object-Relational Mapper (ORM) to simplify database operations. It promotes rapid development, code reusability, and the “Don’t Repeat Yourself” (DRY) principle, making it ideal for building scalable web applications. Django is highly customizable and supports major databases including PostgreSQL, MySQL, SQLite, and Oracle. It works well with front-end technologies and can be integrated with REST APIs and JavaScript frameworks. The framework is maintained by the Django Software Foundation and has a strong community and a rich ecosystem of third-party packages. Django is distributed under the BSD license and is compatible with most operating systems and deployment platforms.

HTML

HTML stands for Hypertext Mark-up Language, was invented by Tim Burners Lee. It is a simple text formatting language used to create hypertext documents. It is a platform independent language unlike most other programming languages. HTML is neutral and can be used on any platform or desktop. It is this feature of HTML that makes it popular as on the WWW. This versatile language allows the creation of hypertext links, also known as hyperlinks. The language used to develop web pages is called Hyper Text Mark-up Language (HTML). HTML is the language interpreted by a browser. HTML is specified as TAGS in an HTML document (i.e. the web page).

HTML TAGS

Tags are instructions that are embedded directly into the text of the document. An HTML tag is a signal to a browser that it should do something other than just throw text up on the screen. By convention all HTML tags begin with an open angle bracket (<). and end with a close angle bracket (>).

THE STRUCTURE OF AN HTML PROGRAM

Every HTML program has a rigid structure. The entire web page is enclosed within <html>
</html> tags. Within these tags two distinct sections are created using the tags <head> </head>and the <body> </body> tags.

JAVASCRIPT

JavaScript is an object based, cross-platform, loosely typed multiuse programming language that is used to add interactivity to the web pages. A JavaScript is a program that is included on an HTML page. Because it is enclosed in the<script>tag, the text of the script does not appear on the user's screen, and the Web browser knows to run the JavaScript program. The<script>> tag is most often found within the <head> section of the HTML page. Scripts that write text to the screen or that write HTML is best put in the body section. JavaScript allows you create an active interface, giving the users feedback as they navigate your pages. JavaScript can be used to make sure that your users enter valid information in forms, which can save time and money. If the forms require calculations, you can do them in JavaScript the user's machine without needing to use a complex server CGI.

With JavaScript, you have the ability to create custom HTML pages depending on actions that the user takes. JavaScript controls the browser, because JavaScript has a set of date and time features. Java script deals with commands called event handles. An action by the user on the page triggers an event handler in your script. JavaScript is case sensitive. Scripts can be put in either of two places on an HTML page: between the<head> and </head> tag or between the <body> and </body> tag.

One of the main uses of JavaScript is to provide feedback to people browsing your site. An alert window can be created that pops up and gives the user the vitally important information that they need to know about the page. Different languages versions can be have had on different scripts on one page. One script might be for any JavaScript version, another for JavaScript1.1 and higher, and a third for JavaScript1.2. In the case of JavaScript, the function is a set of JavaScript statements that performs a task. Function can be called as many times as needed.

PostgreSQL

PostgreSQL is an open-source, object-relational database management system (ORDBMS) known for its advanced features, high performance, and extensibility. It runs on multiple platforms, including Windows, macOS, Linux, and Unix, making it a highly versatile solution for developers and organizations. PostgreSQL is renowned for its support for advanced data types, such as JSON, XML, and hstore, as well as its adherence to SQL standards, ensuring compatibility with most modern applications.

Designed for high concurrency, PostgreSQL can efficiently manage large datasets and handle complex queries, making it ideal for data warehousing, business intelligence, and web applications. It is equipped with powerful features such as multi-version concurrency control (MVCC), which ensures high transaction isolation and prevents data anomalies in high-traffic environments.

PostgreSQL offers advanced indexing techniques, including B-trees, hash indexes, and GiST (Generalized Search Tree) indexes, improving the speed of query execution and ensuring efficient data retrieval. It also supports full-text search, custom functions, and triggers, making it an excellent choice for developers who need to build sophisticated, database-driven applications. The database's extensibility allows developers to add new functionality through custom data types, operators, and procedural languages.

For data integrity and security, PostgreSQL provides strong features such as user authentication, SSL connections, and data encryption. It also supports advanced replication methods like streaming replication and logical replication, enabling high availability and data redundancy. PostgreSQL's ACID compliance (Atomicity, Consistency, Isolation, Durability) ensures reliable transaction processing, and its robust handling of NULL values and constraints further enhances data consistency.

PostgreSQL is highly scalable, supporting both vertical and horizontal scaling, and integrates well with various programming languages, including Python, Java, PHP, and C++. The system is available under the PostgreSQL License, a permissive open-source license, making it free to use and modify for both commercial and non-commercial purposes. Its widespread adoption in industries ranging from financial services to web applications underscores its reliability, performance, and versatility.

OPERATING SYSTEM

This project work is done in Windows 10, which is the operating system. An operating system is a set of software tools designed to make it easy for people or programmers to make optimum use of the computer. People can be separated into two groups, users and programmers. The user wants a convenient set of commands to manage files of data or programs, copy and run application packages while a programmer uses a set of tools that can be held together and debug programs. No matter where you are working, your computer will easier to use and manage, because Microsoft Windows 10 is more compatible and powerful than any workstation you have used.

The main features of Windows 10 are:

1. Easier to use
2. Easier to manage
3. More compatible
4. More powerful

SYSTEM IMPLEMENTATION

CHAPTER 5
SYSTEM IMPLEMENTATION

5.1.1 INTRODUCTION

Implementation means converting a new design into operation. During implementation there must be a strong interaction between the developer and the users. This is the phase where the new system is given full chance to prove its worth and to minimize the reluctance to change. The proposed system may be entirely new, replacing an existing one or it may be modifications to the existing system. In either case, proper implementation is necessary to provide a reliable system to meet organizational requirements. The implementation stage involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.
- Evaluation of the changeover method.

The method of implementation and the time scale to be adopted are found out initially. Next the system is tested properly and the same time users are trained in the new procedures.

Implementation of hardware refers to the final installation of the package in the real environment, to the satisfaction of the intended users and the successful operation of the system. In many organizations, those who commission the hardware development project will not be the one to operate them. In the initial stage, the person who is not sure that the hardware will make the jobs easier will doubt about the hardware. But we must ensure that the resistance does not build one makes sure that

- The active user must be aware of the benefits of using the system.
- Their confidence in the system is build up
- Proper guidance is imparted to the user so that he is comfortable in using the application.

Implementation is the stage of the project where the theoretical design is turned into a working system. At this stage, the main work load, the greatest upheaval and the major impact on the existing system shifts to the department. If the implementation is not carefully planned and controlled, it can cause confusion.

Implementation includes all those activities that take place to convert from the old system to the new one. Proper implementation is essential to provide a reliable system to meet the organizational requirements. Successful implementation may guarantee improvement in the organization using the new system, but improper installation will prevent it.

The process of putting the developed system into the actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system, can be implemented only after through testing is done and if it is found to be working according to the specification of the system.

The most crucial stage is achieving a new successful system and giving confidence on the new system for the user that it will work efficiently. It involves careful planning, investigation of the current system and is constraints on implementation, design of methods to achieve the changeover. The more the complex system being implemented is, the more involved will be the system analysis and the design effort required for its implementation.

5.1.2 CODING

5.1.3 SAMPLE CODES

```
def AdminLog(request):
    error_message = None
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        try:
            user = Login.objects.get(username=username)
            if user.password == password:
                user.last_login = timezone.now()
                user.save()
                login(request, user)
                # return redirect('home')
                return render(request, 'bmsAdmin/home.html')
            else:
                error_message = "Invalid Password"
        except Login.DoesNotExist:
            error_message = "Invalid Username"
    return render(request, 'bmsAdmin/login.html', {'error_message': error_message})

def userSignUp(request):
    if request.method == 'POST':
        fname = request.POST.get('fname')
        lname = request.POST.get('lname')
        username = request.POST.get('username')
        password = request.POST.get('password')
        confirm_password = request.POST.get('confirm_password')
        email = request.POST.get('email')
        phone = request.POST.get('phone')
        image = request.FILES.get('image')

        if password != confirm_password:
            messages.error(request, "Passwords do not match.")
            return redirect('bmsUsers:register')

        newUserLogin = Login(username=username, password=password, role='user')
        newUserLogin.save()

        if not image:
            image = f"{settings.STATIC_URL}images/default_pic.jpg"

        newUser = User(
            login_id = newUserLogin.id,
            fname=fname,
            lname=lname,
            email=email,
            phone=phone,
            image=image
        )

        newUser.save()
        messages.success(request, f"{username} registered successfully!")
        return redirect('bmsUsers:register')


    return render(request, 'bmsUsers/register.html')

def loginUser(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        try:
            user = Login.objects.get(username=username, role="user")
            if user.check_password(password):
                user.last_login = timezone.now()
                user.save()
                login(request, user)
                return redirect('bmsUsers:page1')
            else:
                messages.error(request, "Invalid Password")
        except Login.DoesNotExist:
            messages.error(request, "Invalid Username")

    return render(request, 'bmsUsers/login.html')

def viewCartOrders(request, order_id):
    order = get_object_or_404(Orders, id=order_id)
    details = OrderDetail.objects.filter(order=order)

    cart_items = []
    for detail in details:
        cart_items.append({
            'snack_name': detail.snack.snack_name,
            'quantity': detail.quantity,
            'price': detail.snack.price,
            'image': detail.snack.image,
            'subtotal': (detail.quantity * detail.snack.price)
        })

    total_price = sum(item['price'] * item['quantity'] for item in cart_items)
    content = {
        'cart_items': cart_items,
        'total_price': total_price,
        'order_uid': order.unique_order_id,
        }

    return render(request, 'bmsUsers/viewCartOrders.html', content)

def add2Cart(request, snack_id):
    quantity = request.POST.get('quantity')

    if isinstance(quantity, str):
        quantity = int(quantity)
    if quantity is None:
        quantity = 1

    user = get_object_or_404(User, login=request.user)
    snack = get_object_or_404(Snacks, id=snack_id)



    cart = user.cart or {"snacks": []}

    # Check if the snack is already in the cart
    for item in cart["snacks"]:
        if item["snack_id"] == snack_id:
            item["quantity"] += 1
            break
    else:
        cart["snacks"].append({"snack_id": snack_id, "quantity": quantity})

    user.cart = cart
    user.save()

    messages.success(request, f"{snack.snack_name} added to cart!")
    return redirect("bmsUsers:cart")

def buy_now(request, sid):
    user = get_object_or_404(User, login=request.user)
    snack = get_object_or_404(Snacks, id=sid)
    quantity = request.POST.get('quantity')

    if isinstance(quantity, str):
        quantity = int(quantity)

    if quantity is None:
        quantity = 1


    total_price = snack.price * quantity

    # Create & Save Order before storing in session
    order = Orders.objects.create(
        user=user,
        order_status='Pending',
        total_price=total_price,
        payment_status='Pending'
    )

    OrderDetail.objects.create(
        order=order,
        snack=snack,
        quantity=quantity,
        shop=snack.shop_id
    )
    request.session["pending_order_id"] = order.id
    request.session.modified = True

    return redirect("bmsUsers:process_payment")

def editSnacks(request, sId):
    snack = get_object_or_404(Snacks, id=sId)
    categories = Category.objects.all()

    if request.method == 'POST':
        snack_name = request.POST.get('snack_name')
        price = request.POST.get('price')
        description = request.POST.get('description')
        availability = request.POST.get('availability') == 'on'
        category_id = request.POST.get('category_id')
        image = request.FILES.get('image')


        snack.snack_name = snack_name
        snack.price = price
        snack.description = description
        snack.availability = availability
        snack.category_id = Category.objects.get(id=category_id)


        if image:
            if snack.image:
                old_image_path = os.path.join(settings.MEDIA_ROOT, str(snack.image))
                if os.path.isfile(old_image_path):
                    os.remove(old_image_path)


            snack.image = image

        snack.save()
        return redirect('partners:manageSnacks')
    content = {
        'snack': snack,
        'categories': categories
    }

    return render(request, 'partners/shops/editSnacks.html', content)

def contactShop(request):
    user = request.user
    theater_id = get_object_or_404(Theater, login=user)
    shops = Shop.objects.filter(theater_id=theater_id)

    shop_data = []
    for shop in shops:
        contact_info = f"Email: {shop.email}\nPhone: {shop.phone}"

        qr = qrcode.make(contact_info)
        buffer = BytesIO()
        qr.save(buffer, format="PNG")
        img_str = base64.b64encode(buffer.getvalue()).decode()

        shop_data.append({
            'shop': shop,
            'qr_code': img_str
        })

    context = {
        'active_page': 'contact',
        'shop_data': shop_data
    }
    return render(request, 'partners/theater/home.html', context)
```

5.1.4 CODE VALIDATION AND OPTIMIZATION

Validation is the process of evaluating software at the end of the software development to ensure compliance with software requirements. Testing is a common method for validation. Validation succeeds when the software functions in a manner that can be reasonably expected by the customer. Form data validation comes in a couple different forms. Data can be validated at the field level when it is entered by the user, and it can be validated at the form level (i.e., all fields) when the form is submitted or printed. These types of validation have different, complimentary purposes and for a complete form design it is a good practice to use a combination of the two methods.

Field level validation:

The purpose of Field Level Validation is to verify that the input to a single field is entered correctly. For example, for an e-mail field, the job of the validation script is to make sure the entered text matches the standard email format, i.e., two sets of strings separated by an "@" symbol and to check whether there is a dot(.) separating the domain name. The most common way to implement a text pattern test like this is to use a regular expression. Most of the time validation scripts are used to match input text against a pattern using a regular expression.

Form level validation:

Form level validation is used to ensure all the required form data is filled in, and or to make sure that any data dependencies between fields are met before the form is submitted.

5.2 UNIT TESTING

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing. It is performed by using the White Box Testing method. Unit Testing is the first level of software testing and is performed prior to Integration Testing. It is normally performed by software developers themselves or their peers. In rare cases, it may also be performed by independent software testers.

Unit Testing Benefits:

- Unit testing increases confidence in changing/ maintaining code. If good unit tests are written and if they are run every time any code is changed, we will be able to promptly catch any defects introduced due to the change. Also, if codes are already made less interdependent to make unit testing possible, the unintended impact of changes to any code is less.

- Codes are more reusable. In order to make unit testing possible, codes need to be modular. This means that codes are easier to reuse.

- Development is faster. Writing tests takes time but the time is compensated by the less amount of time it takes to run the tests. Unit tests are more reliable than 'developer tests'. Development is faster in the long run too. The effort required to find and fix defects found during unit testing is very less in comparison to the effort required to fix defects found during system testing or acceptance testing.

- The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels. Compare the cost (time, effort, destruction, humiliation) of a defect detected during acceptance testing or when the software is live.

- Debugging is easy. When a test fails, only the latest changes need to be debugged. With testing at higher levels, changes made over the span of several days/weeks/months need to be scanned.

- Codes are more reliable

SYSTEM TESTING

CHAPTER 6
SYSTEM TESTING

6.1 INTRODUCTION

Once source code can be generated, software must be tested to uncover (and correct) as many errors as possible before delivery to customer. Our goal is to design a series of test cases that have a high likelihood of finding errors. To uncover the errors software techniques are used. These techniques provide systematic guidance for designing test that exercise the internal logic of software components, and exercise the input and output domains of the program to uncover errors in program function, behavior and performance. Software is tested from two different perspectives:

- Internal program logic is exercised using - white box/ test case design techniques.

- Software requirements are exercised using – black box/ test case design techniques.

In both cases, the intent is to find the maximum number of errors with the minimum amount of effort and time.

6.2 INTEGRATION TESTING

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These “design items”, i.e. a group of units, are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested, and individual subsystems are exercised through their input interfaces.
Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure callsor procedure activations, and this is done after testing individual modules, i.e. unit testing.
The overall idea is a “building block” approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages. Some different types of integration testing are big bang, top-down and bottom-up. Other integration patterns are: Collaboration Integration, Backbone Integration, Layer Integration, Client/Server integration, distributed service Integration and High-frequency Integration.

BIG BANG

In this approach, all or most of the modules are coupled together to form a complete software system or major part of the system and then used for integration testing. The big bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of Integration testing. A type of Big Bang Integration testing is called Usage Model Testing can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes on optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment. For integration testing, Usage Model testing can be more efficient and provides better test than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confident that the integrated environment will work as expected for the target customers.

TOP-DOWN AND BOTTOM-UP

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher-level components. This process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of percentage. Top down testing is an approach to integrated testing where the top integrated modules are tested, and branch of the module is tested step by step until the end of the related module.

Sandwich Testing is an approach to combine top down testing with bottom up testing. The main advantage of the bottom-up approach is that bugs are more easily found. With top-down, it is easier to find a missing branch link.

6.1.1 SYSTEM TESTING

System Testing is the type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested. System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specification or in the context of both. System testing tests the design and behavior of the system and the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirement specification (SRS). System Testing is basically performed by a testing team that is independent of the development team that helps to best the quality of the system impartial. It has both functional and non-functional testing. System Testing is a Black-box testing. System Testing is performed after the integration testing and before the acceptance testing.

6.1.2 TEST PLAN AND TEST CASES

A test plan is a document detailing the objectives, target market, internal beta team, and processes for a specific beta test for a software or hardware product. The plan typically contains a detailed understanding of the eventual workflow.

Test Case 1

TEST CASE FOR SUPER ADMIN

Test case ID	Input specifications	Input data	Expected result	Test case result
1	Run the application on browser		Application is up and home screen is displayed	Pass
2	Enter a valid username and password then press login button	Username and Password	Admin log into the page	Pass
3	Enter an invalid username or password	Invalid username or password	"Invalid username or password" is displayed	Pass
4	Admin can add location, Categories	Click on add button	Successfully added	Pass
5	Admin can view user info		Details are listed	Pass

TEST CASE FOR USER

Test case ID	Input specifications	Input data	Expected result	Test case result
1	Running the application on browser		Home page is displayed	pass
2	If a user registers	Click on the icon user and fill the required fields	User register	pass
3	Enter a valid username and password then press login button	Username & password	User gets logged into the page	pass
4	Enter an invalid username or password	Invalid credentials	“Invalid username or password” is displayed	pass
5	User select hometown and theater	State-> theater	Display theaters who has partnership with bms	pass
6	User select shops	Choose from list	Redirect to snacks list	pass
7	Order placement	Select a snack, click buy now	Redirect to payment	pass
8	Cart	Click on ‘add to cart’ button	Item added to cart	pass
9	If user wants to view orders history	Click on the orders icon	History will be displayed	pass
10	User profile	Select profile icon	Redirect to profile	pass

TEST CASE FOR THEATER

Test case ID	Input specifications	Input data	Expected result	Test case result
1	Running the application on browser		Home page is displayed	pass
2	If a theater registers	Click on the icon user and fill the required fields	Theater registered	pass
3	Enter a valid username and password then press login button	Username & password	User gets logged into the page	pass
4	Enter an invalid username or password	Invalid credentials	“Invalid username or password” is displayed	pass
5	Admin can add shops	Filllup registration form	Shop has registered	pass
6	Admin can manage shops	Click the manage shops option	List of shops will appear	pass
7	User profile	Select profile option	Redirect to profile	pass

TEST CASE FOR SHOP

Test case ID	**Input specifications**	**Input data**	**Expected result**	**Test case result**
1	Running the application on browser		Home page is displayed	pass
2	Enter a valid username and password then press login button	Username & password	User gets logged into the page	pass
3	Enter an invalid username or password	Invalid credentials	"Invalid username or password" is displayed	pass
4	Admin can add snacks	Fillup the new snacks form	"Snacks Added"	pass
5	Manage snacks	Choose 'our snack'	Snacks details will be listed	pass
6	Order placement	Accept/reject	Updated in orders list	pass
7	History	Select orders history	Orders history will be displayed	pass
8	Revenue	Select revenue option	Payment History will be displayed	pass
10	User profile	Select profile icon	Redirect to profile	pass

SYSTEM MAINTENANCE

CHAPTER 7

SYSTEM MAINTENANCE

7.1 INTRODUCTION

The results obtained from the evaluation process help the organization to determine whether its information systems are effective and efficient or otherwise. The process of monitoring, evaluating, and modifying of existing information systems to make required or desirable improvements may be termed as System Maintenance.

System maintenance is an ongoing activity, which covers a wide variety of activities, including removing program and design errors, updating documentation and test data and updating user support. For the purpose of convenience, maintenance may be categorized into three classes, namely:

1) Corrective Maintenance:

This type of maintenance implies removing errors in a program, which might have crept in the system due to faulty design or wrong assumptions. Thus, in corrective maintenance, processing or performance failures are repaired.

2) Adaptive Maintenance:

In adaptive maintenance, program functions are changed to enable the information system to satisfy the information needs of the user. This type of maintenance may become necessary because of organizational changes which may include:

a) Change in the organizational procedures.
b) Change in organizational objectives, goals.
c) Change in forms.
d) Change in information needs of managers.
e) Change in system controls and security needs, etc.

3) Perfective Maintenance:

Perfective maintenance means adding new programs or modifying the existing programs to enhance the performance of the information system. This type of maintenance undertaken to respond to user's additional needs which may be due to the changes within or outside of the organization. Outside changes are primarily environmental changes, which may in the absence of system maintenance, render the information system ineffective and inefficient. These environmental changes include:

a) Changes in governmental policies, laws, etc.
b) Economic and competitive conditions.
c) New technology.

7.2 SYSTEM MAINTENANCE

The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is to make it adaptable to the changes in the system environment. They may be social technical and other environment changes which affect a system which is implemented. Software product enhancement may involve providing new functional capabilities, improving user displace and mode of interaction, upgrading the performance characteristics of the system. Only through proper system maintenance procedures, system can be adapted to cope up with this change. Software maintenance is of course far than “Finding mistakes”. We may define maintenance by describing four activities that are undertaken to after a program is released.

The first maintenance activity occurs became it is unreasonable to assume that software texting will uncover errors in a large software system. During the use of any large program, errors will occur and report to the developer. The process that includes the diagnosis and correction of one or more errors is called corrective measures.

The second activity that contributes to a definition of maintenance occurs because of the rapid change that is encountered in every aspect of computing. Therefore, maintenance is an activity that modifies software to properly interface with a changing environment id both necessary and common place.

Third activity that may apply to a definition of maintenance occurs when a software package is successful. As the software is used, recommendations for new capabilities, modifications to existing functions and general enhancements are received from users. To satisfy requests in this category, perfective maintenance is performed. This activity accounts for the majority of all efforts expended on software maintenance.

SYSTEM SECURITY

CHAPTER 8
SYSTEM SECURITY

8.1 INTRODUCTION

Security is a fundamental component of the BookMySnacks platform, ensuring safe and reliable access for users, theater owners, shop admins, and super admins. Given that the platform involves sensitive user data and financial transactions via online payments, robust multi-level security practices have been implemented to safeguard system integrity, prevent unauthorized access, and protect data privacy.

8.2 OPERATING LEVEL SECURITY

- The server hosting *BookMySnacks* is configured with a hardened Linux-based OS, minimizing vulnerabilities by disabling unused ports and services.
- Regular OS updates and patches are applied to prevent exploitation of known security flaws.
- Access to the production environment is restricted via SSH keys and role-based permissions, reducing the risk of unauthorized administrative control.

8.3 DATABASE LEVEL SECURITY

- All user data, including login credentials, order history, and payment details, is securely stored in a relational database with encrypted connections (SSL/TLS).
- SQL Injection protection is enforced using ORM features in Django, preventing unauthorized query execution.
- Database access is restricted to the backend application layer; no direct access is provided to users or external services.
- Admin credentials and sensitive data fields are hashed using industry-standard algorithms like bcrypt or PBKDF2.

8.4 SYSTEM LEVEL SECURITY

- Authentication is enforced across all user types (users, shop admins, theater owners, super admin) via secure login sessions.
- CSRF (Cross-Site Request Forgery) and XSS (Cross-Site Scripting) protections are implemented using Django’s built-in middleware.
- Sessions auto-expire after inactivity to prevent session hijacking.
- Payment operations are securely handled via Paytm integration with token-based authentication and server-side verification.
- Logging and monitoring tools are in place to track suspicious activity, failed

login attempts, and unauthorized access patterns.

- Admin and Super Admin dashboards are isolated from user interfaces, with two-factor authentication planned as a future enhancement.

SYSTEM PLANNING & SCHEDULING

CHAPTER 9
SYSTEM PLANNING & SCHEDULING

9.1 INTRODUCTION

System planning and scheduling are crucial in ensuring the successful development and timely delivery of the *BookMySnacks* platform. Effective planning outlines the project's roadmap, allocates resources efficiently, anticipates challenges, and maintains progress toward clearly defined milestones.

9.2 PLANNING A SOFTWARE PROJECT

Planning the *BookMySnacks* system required a structured approach involving collaboration between developers, designers, testers, and stakeholders. The project followed an agile-like methodology with iterative milestones and continuous feedback.

9.3 STEPS INVOLVED IN PLANNING A SYSTEM

- **Requirement Analysis**: Identification of core modules: User, Shop Admin, Theater Admin, Super Admin. Features like snack pre-booking, real-time order tracking, and payment integration were prioritized
.
- **Feasibility Study:** Assessed technical, operational, and economic feasibility, confirming the viability of the system.
- **Resource Allocation:** Assigned tasks to developers, UI designers, and testers based on skills and module complexity.
- **System Design:** Prepared data flow diagrams, ER models, and wireframes to visualize the entire system.
- **Development Scheduling:** Created detailed time estimates and deliverables for each module.
- **Testing & Review Planning:** Defined schedules for unit testing, integration testing, and UAT (User Acceptance Testing).

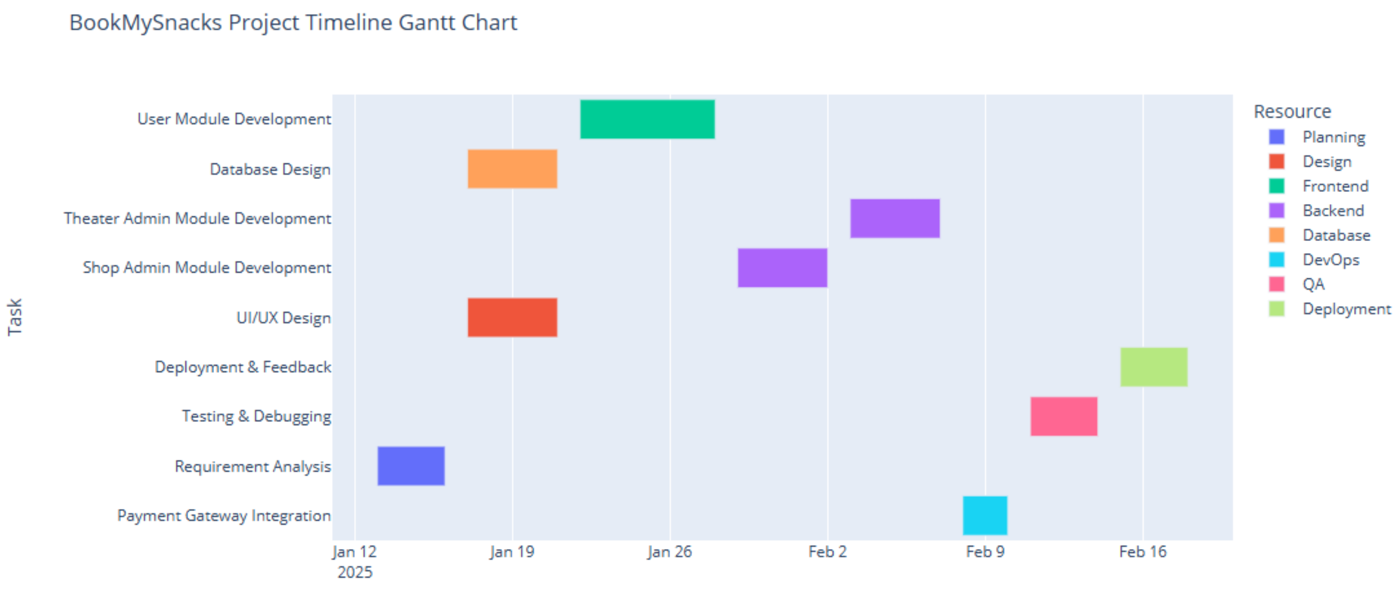
9.4 GANTT CHART

A GANTT chart is a bar chart that represents the project schedule. It visualizes the start and end dates of individual tasks and their progression over time.

Purpose:

In the BookMySnacks project, the GANTT chart helped in tracking:

- Progress of each module (User, Admin, Shop Admin)
- Overlapping tasks (e.g., UI design and backend setup)
- Delays or early completions



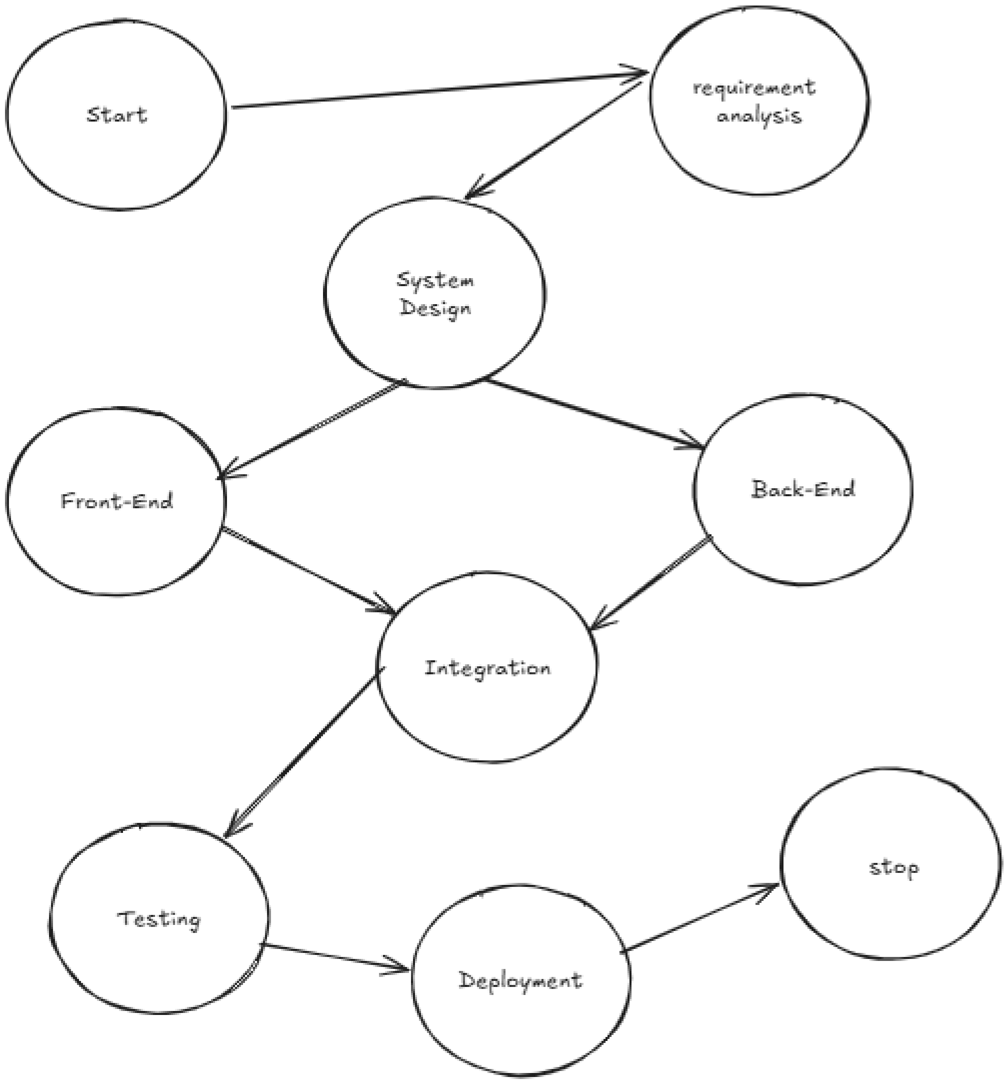
9.5 PERT CHART

A PERT (Program Evaluation Review Technique) chart is a network diagram that maps the sequence and dependencies of tasks in a project. It's used to identify the critical path — the longest chain of dependent tasks that determine the project's minimum completion time.

Purpose:

In *BookMySnacks*, the PERT chart was used to:

- Understand task dependencies (e.g., UI design must precede frontend integration)
- Calculate earliest and latest start times
- Identify tasks that can be done in parallel (e.g., User and Admin module development)


Start
requirement analysis
System Design
Front-End
Back-End
Integration
Testing
Deployment
stop

SYSTEM COST ESTIMATION

CHAPTER 10
SYSTEM COST ESTIMATION

10.1 INTRODUCTION

System cost estimation is an essential phase in project planning and execution. It involves forecasting the required resources in terms of effort, time, and cost to complete the software successfully. Accurate cost estimation helps avoid under- or over-budgeting and ensures efficient allocation of manpower, tools, and infrastructure. For the *BookMySnacks* project, Lines of Code (LOC) Based Estimation has been chosen due to its direct correlation with the size and complexity of the software modules being developed.

10.2 LOC BASED ESTIMATION

Lines of Code (LOC) Based Estimation is a traditional and widely-used method in software engineering. It estimates project effort based on the expected number of source lines in the complete application. This technique is effective for projects where the development team has prior experience and can reasonably predict the code size.

Eg: *productivity assumption: Average productivity = 10 LOC/hour*

The LOC-based estimation for *BookMySnacks* provides a practical approach to assess development effort and cost. With a total of approximately 3500 lines of code projected, the expected effort is 350 person-hours. This estimation aids in scheduling, budgeting, and resource planning for the smooth execution of the project. As the system evolves, these estimations can be refined for greater accuracy.

FUTURE ENHANCEMENT AND SCOPE OF FURTHER DEVELOPMENT

CHAPTER 11

FUTURE ENHANCEMENT AND SCOPE OF FURTHER DEVELOPMENT

11.1 INTRODUCTION

Using **HTML, CSS, JavaScript, Python Django, and PostgreSQL**, the software entitled **“BookMySnacks”** has been designed, developed, and tested for diverse use cases. The system is meticulously planned and designed to offer a seamless experience for users, shops, theater, and administrators. Its robust architecture ensures flexibility, allowing future modifications to incorporate new features as per evolving requirements. The system delivers high performance and efficiency.

11.2 MERIT OF THE SYSTEM

- **Queue-Free Experience**: Eliminates long wait times by enabling pre-ordering of snacks.
- **Efficient Order Handling**: Streamlines the flow of orders for Shop Admins and ensures faster fulfillment.
- **Theater-Linked Menus**: Displays only relevant snack options based on theater and show selection.
- **Real-Time Coordination**: Enables live order tracking and status updates for users and shop admins.
- **Centralized Management**: Unified platform for users, shops, theaters, and admins to operate collaboratively.
- **Reduced Intermission Congestion**: Orders are placed in advance, cutting down rush during breaks.

11.3 DEMERIT OF THE SYSTEM

- **Dependency on Internet:** Requires consistent internet connectivity for full functionality.
- **No Seat Delivery**: Users must still collect snacks manually, which may cause minor delays.
- **Technical Challenges:** Potential for downtime or technical glitches in high-traffic scenarios.
- **Maintenance Overhead:** Regular updates and security checks are essential to ensure smooth operation.
- **Theater Collaboration Required**: System effectiveness relies on active

participation from theater and shop admins.

- **Limited Offline Functionality**: Does not support orders via SMS or offline booking methods.
- **Initial Adoption Resistance**: Users unfamiliar with pre-order systems may be hesitant to use the platform.
- **Overload Risk During Peak Hours**: Simultaneous high-volume orders could lead to temporary slowdowns or queue buildup at counters.

11.4 FUTURE ENHANCEMENT OF THE SYSTEM

Future development is crucial for any project as it introduces the latest features, improves system performance, and addresses user feedback. Enhancements ensure the system stays relevant and continues to meet the evolving needs of users. For the **BookMySnacks** system, future enhancements aim to integrate dynamic features and improve user experience. These proposed enhancements are explained briefly below:Reporting module with real time mechanism.

- **Digital Queue Management**: Smart queue tokens with live counter updates to reduce physical crowding.
- **Order Recommendation Engine**: Suggest popular combos or snacks based on past orders and movie genres.
- **UPI & Wallet Integration**: Enable fast, secure payments via UPI apps, wallets, and loyalty points.
- **Dynamic Menu Updates**: Real-time availability adjustments to avoid order failures or stockouts.
- **Analytics Dashboard**: Insights for shop and theater admins to track sales, peak times, and customer preferences.
- **Guest Ordering**: Allow one-time users to order without full registration for faster onboarding.
- **Theater Announcements**: Notify users about offers, queue movement, or counter changes in real-time.
- **Multilingual Support**: Interface available in regional languages to reach a wider audience.

CONCLUSION

CHAPTER 12

CONCLUSION

The project titled **"BookMySnacks"** is a robust web-based application developed to optimize and modernize the snack-ordering experience in movie theaters. It streamlines the ordering process by allowing users to pre-order snacks based on their selected theater and movie timing, reducing the rush and wait times during intermissions. The system empowers users, shop admins, and theater administrators to interact through dedicated modules, ensuring smooth operations and effective coordination.

The platform prioritizes usability and security, offering a simple and intuitive interface for users to browse, select, and pay for snacks, while enabling shop admins to manage inventory and orders in real-time. All modules have been rigorously tested with a variety of input scenarios to ensure stability and reliability under actual usage conditions.

Built with scalability in mind, **BookMySnacks** is designed to accommodate future enhancements such as UPI-based payments, multilingual support, personalized recommendations, and digital queue management. These features will further elevate the user experience and operational efficiency of the system.

In conclusion, **BookMySnacks** effectively meets its objectives by addressing the chaos and delays associated with traditional snack ordering in theaters. It delivers a convenient, organized, and modern alternative for both cinema-goers and snack vendors. With its adaptable architecture and future-ready design, the system holds strong potential to revolutionize the in-theater snack experience across a broader landscape.

BIBLIOGRAPHY

CHAPTER 13

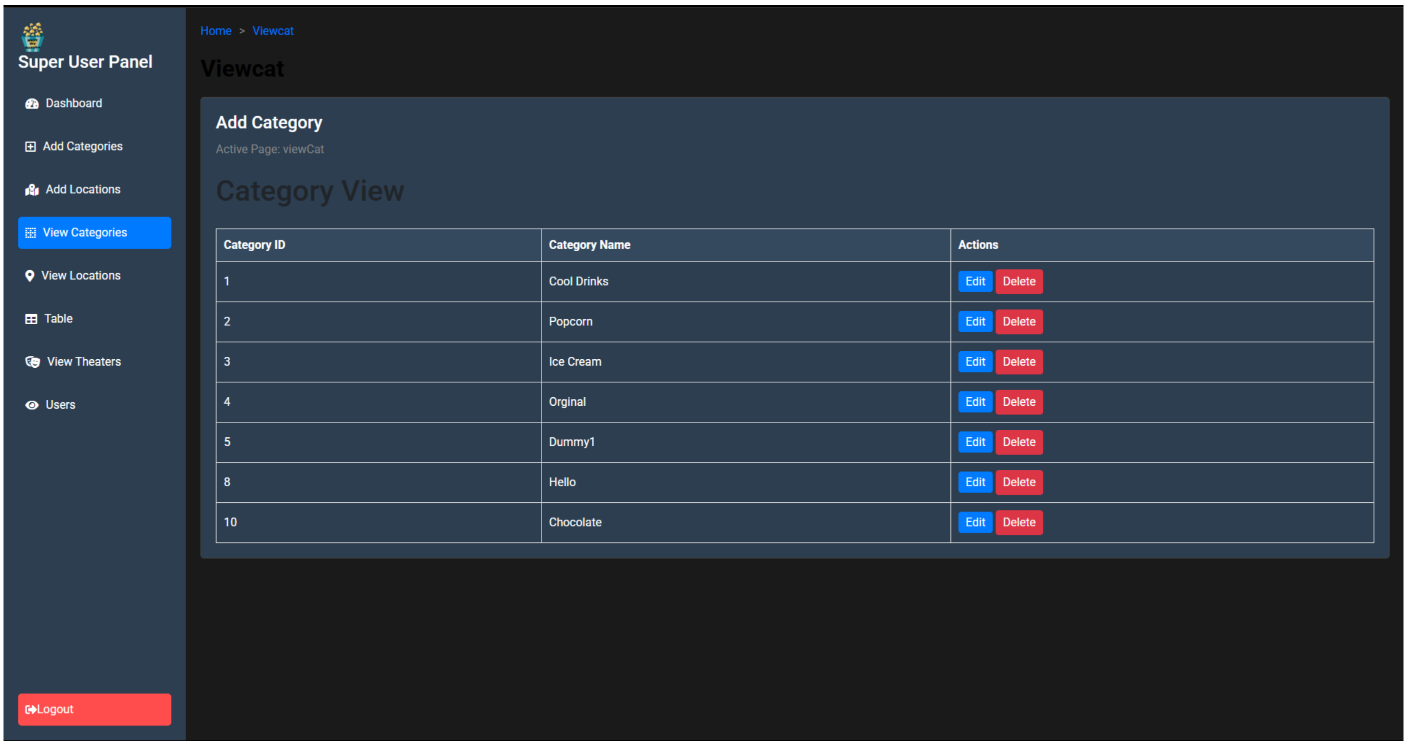
BIBILOGRAPHY

Web References

- **https://www.w3schools.com**
- **https://docs.djangoproject.com**
- **chatgpt.com**
- **youtube.com**
- **https://github.com**
- **https://www.postgresql.org/**
- **https://www.python.org/**
- **https://getbootstrap.com/**
- **GeeksforGeeks**
- **https://fontawesome.com**
- **https://www.zomato.com/**
- **https://www.swiggy.com/**
- **Plotly – Python Graphing Library for Gantt & Data Visualization**
- **TutorialsPoint – Full Stack Resources**

APPENDIX

Super User



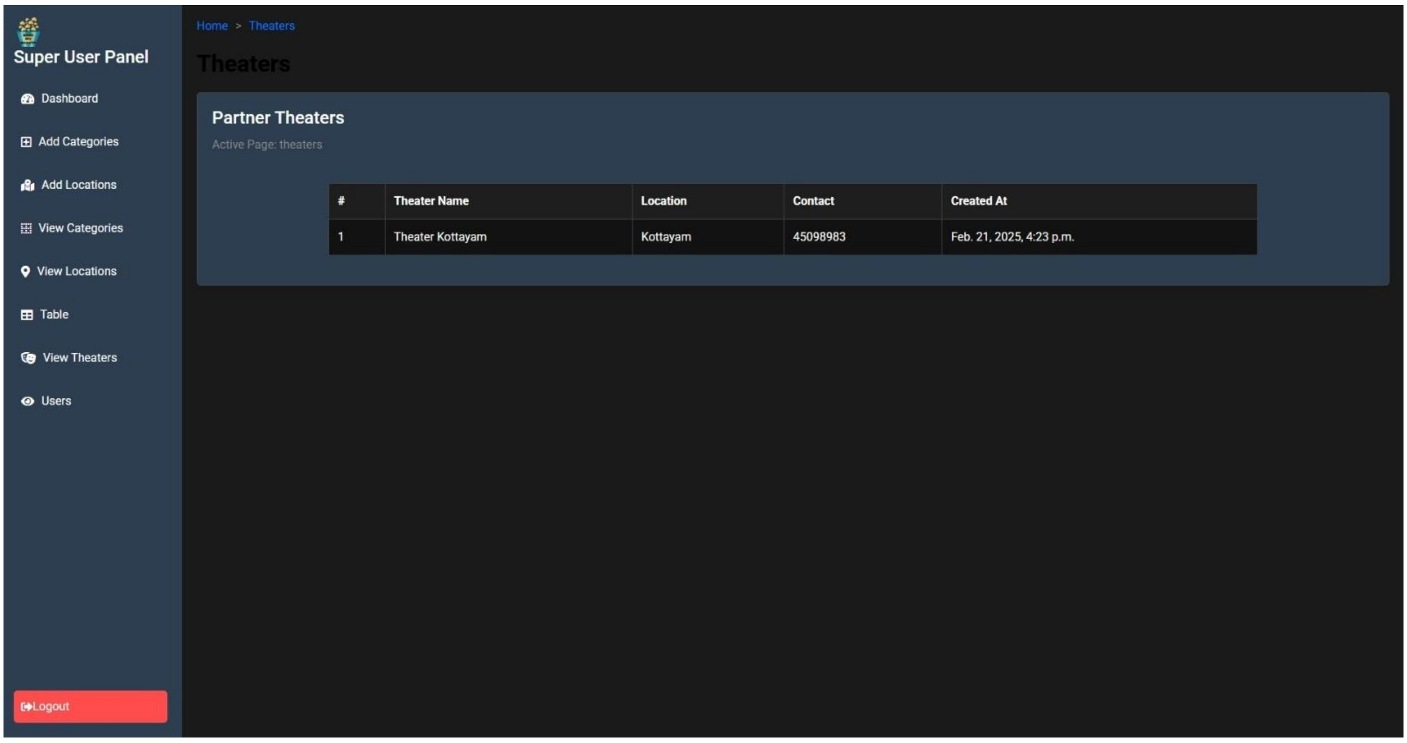
Viewcat

Add Category

Active Page: viewCat

Category View

Category ID	Category Name	Actions
1	Cool Drinks	Edit Delete
2	Popcorn	Edit Delete
3	Ice Cream	Edit Delete
4	Orginal	Edit Delete
5	Dummy1	Edit Delete
8	Hello	Edit Delete
10	Chocolate	Edit Delete



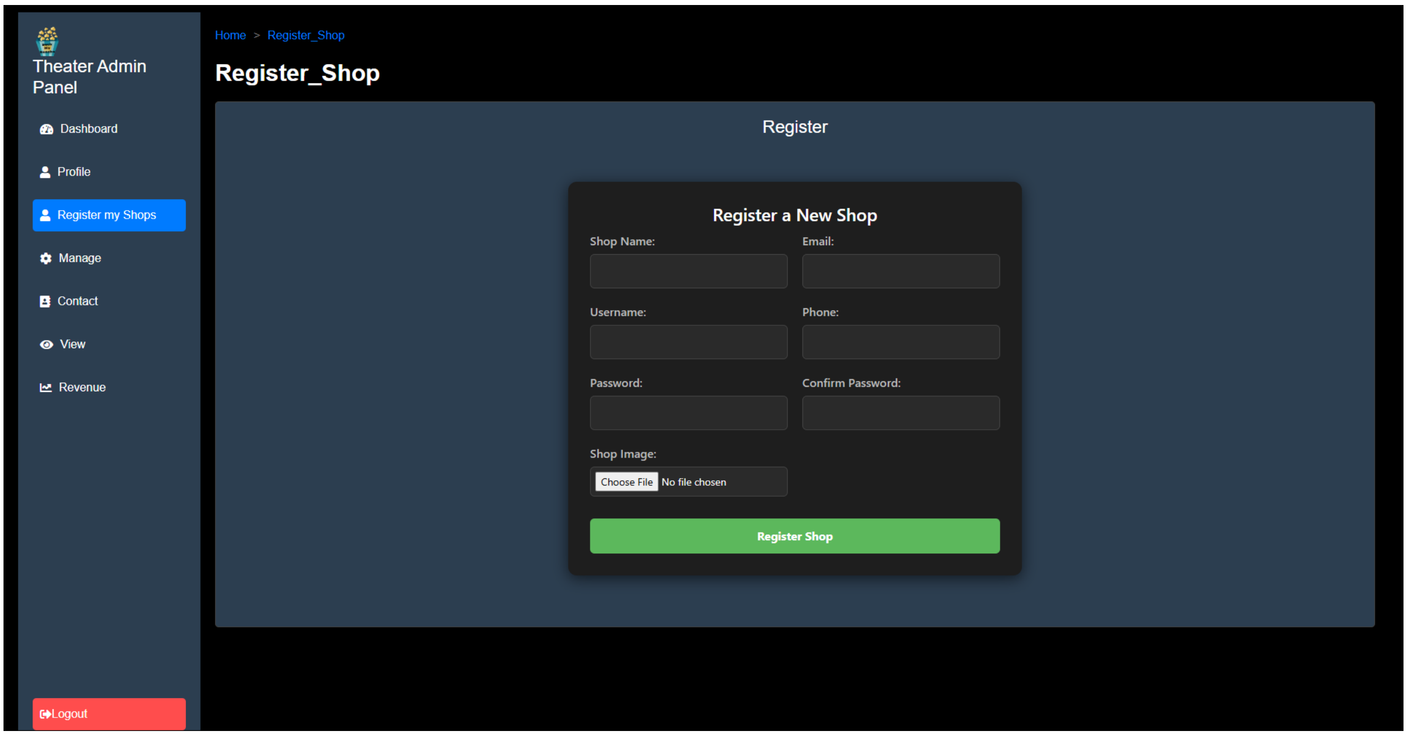
Theaters

Partner Theaters

Active Page: theaters

#	Theater Name	Location	Contact	Created At
1	Theater Kottayam	Kottayam	45098983	Feb. 21, 2025, 4:23 p.m.

Theater



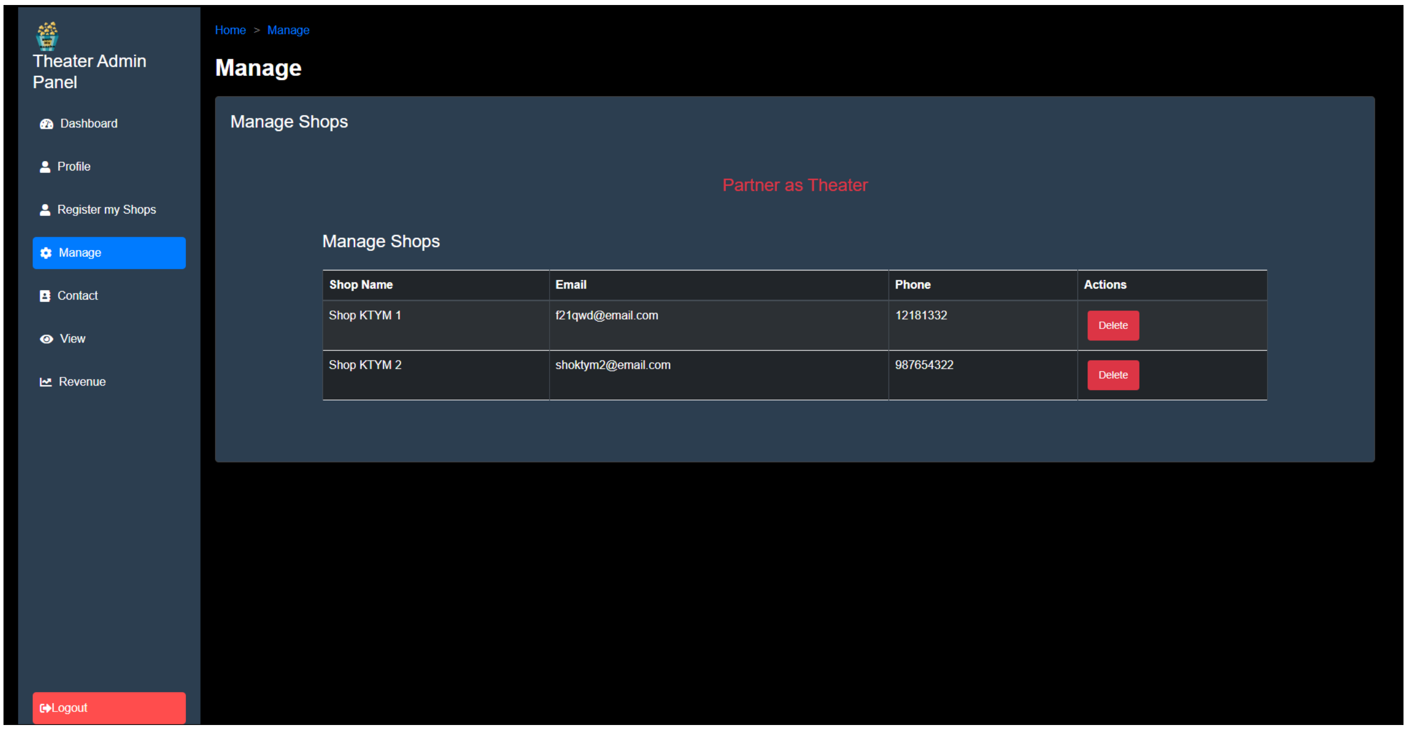
Register_Shop

Register

Register a New Shop

Shop Name:
Email:
Username:
Phone:
Password:
Confirm Password:
Shop Image:
Choose File No file chosen

Register Shop



Manage

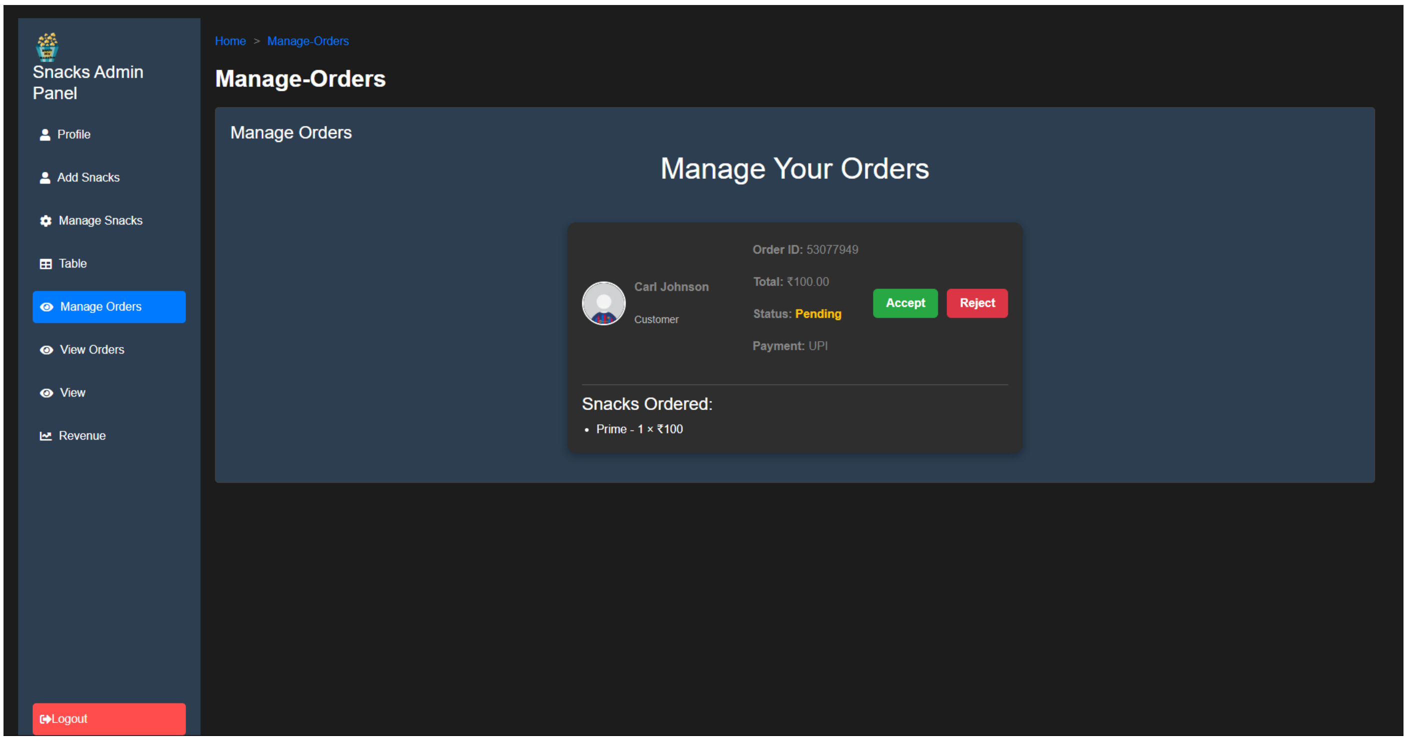
Manage Shops

Partner as Theater

Manage Shops

Shop Name	Email	Phone	Actions
Shop KTYM 1	f21qwd@email.com	12181332	Delete
Shop KTYM 2	shoktym2@email.com	987654322	Delete

Shop



Manage-Orders

Manage Orders

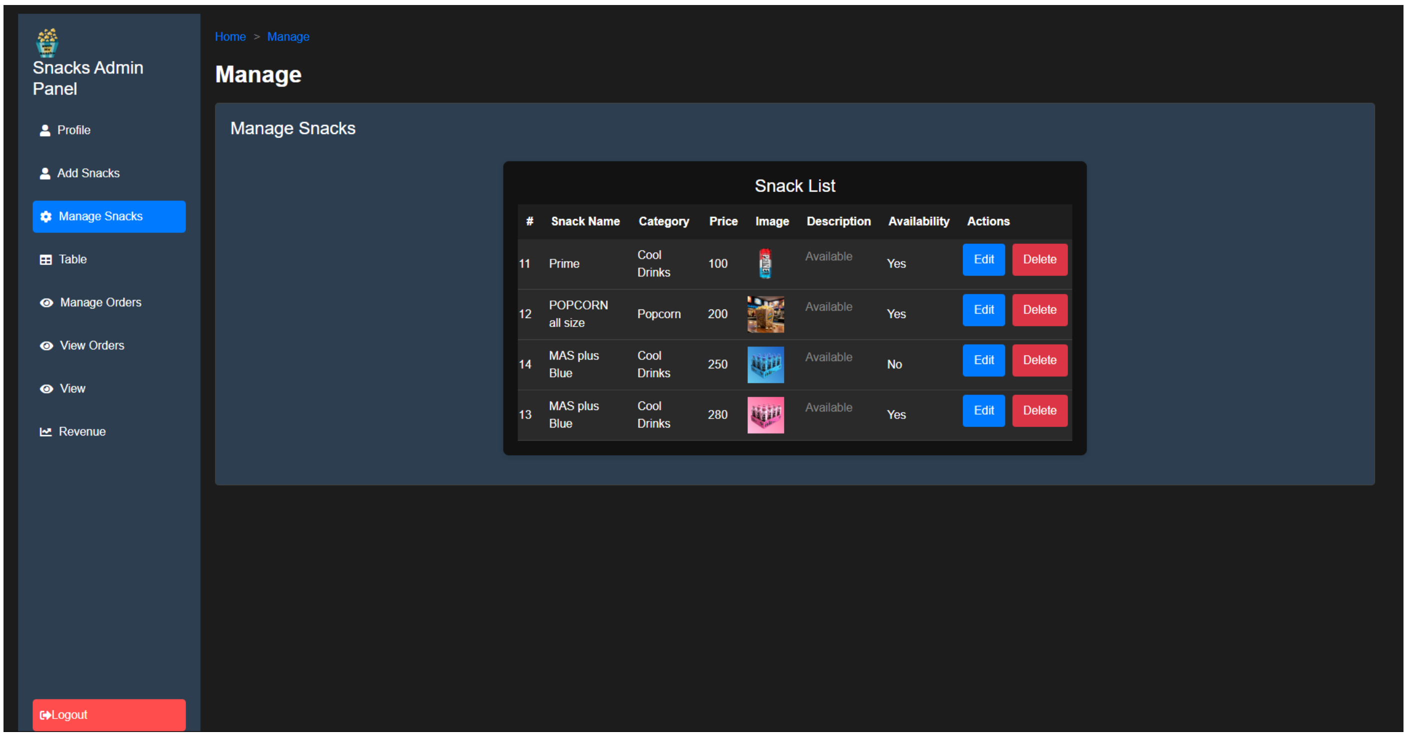
Manage Your Orders

Carl Johnson
Customer

Order ID: 53077949
Total: ₹100.00
Status: Pending
Payment: UPI

Snacks Ordered:

- Prime - 1 × ₹100



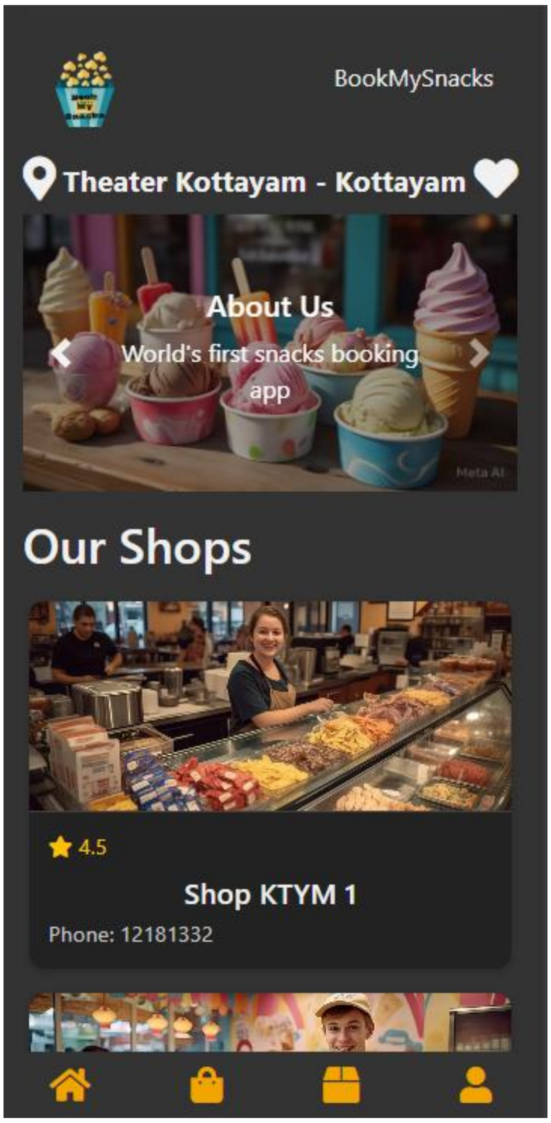
Manage

Manage Snacks

Snack List

#	Snack Name	Category	Price	Image	Description	Availability	Actions
11	Prime	Cool Drinks	100		Available	Yes	Edit Delete
12	POPCORN all size	Popcorn	200		Available	Yes	Edit Delete
14	MAS plus Blue	Cool Drinks	250		Available	No	Edit Delete
13	MAS plus Blue	Cool Drinks	280		Available	Yes	Edit Delete

User


BookMySnacks

Theater Kottayam - Kottayam

About Us

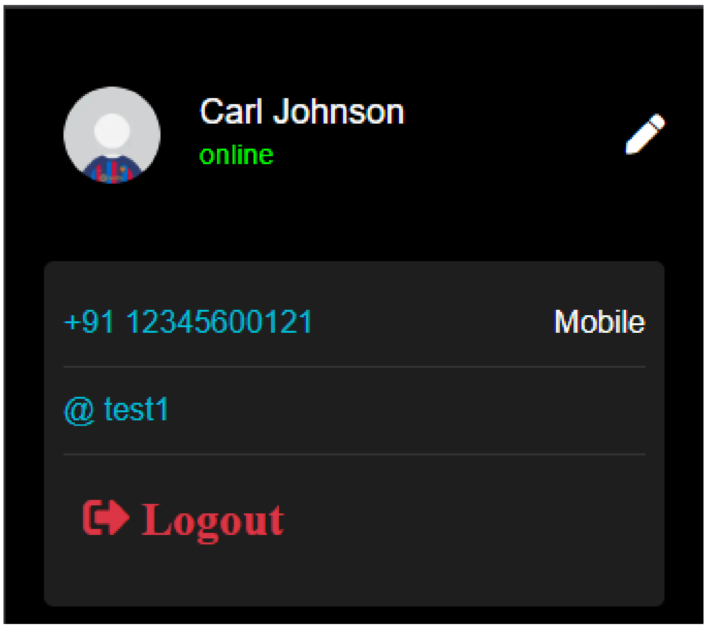
World's first snacks booking app

Our Shops

4.5

Shop KTYM 1

Phone: 12181332


Carl Johnson

online

+91 12345600121 Mobile

@ test1

Logout