Pritam Gurung

Software Designs

Textual Analysis

“The Bostall Heath Arts Cinema Club is a not for profit, private members club. It is based in a small, old, purpose-built cinema, inherited by one of the founder members. The club employs a secretary and a part-time bookkeeper, who work from a small office in the building. Membership is renewed annually by subscription, following a reminder being sent out by the secretary. Both single and family memberships are offered. Membership application forms, with a copy of the club’s constitution, and a list of the current executive officers are given to those requesting membership. The completed application forms must be returned with a year’s subscription. They are filed in alphabetical order in a filing cabinet and used thereafter as the primary membership records. A desk diary is used to record when the next renewal subscription date is due. Members are asked to book tickets for a film at least one month in advance to ensure a good chance of getting seats. Each member is eligible to request one additional seat for a guest. This is currently achieved by each member entering their name and membership number with the number of tickets required against each show on a booking- form. The booking form lists the programme schedule prepared in advance by the secretary. Booking -forms for the current season are displayed in the foyer for members to pick up. Each member will also receive advanced booking forms by email twice a year. Those without email will receive booking forms by post. Completed booking forms are returned to the secretary for processing. This is currently done manually by assigning ticket numbers for each show to members as their forms come in. Bookings received for each show are marked on the show’s seating plan. Members who subsequently apply for tickets to a sold out show are informed that they have been unsuccessful. Members with successful bookings are notified that their tickets are available for collection.”

Key Code (some may already be highlighted in other colours, so may list instead below)

Blue – Class
Green – Attributes
Red – Rejected Items
Turquoise – Nouns; Club, Secretary, Bookkeeper, Office, Membership, Application forms, Constitution, Executive officers, filing cabinets, desk diary, tickets, film, member, email, booking forms, season, programme schedule, bookings, show’s seating plan, collection.
Violet/pink – Verbs; inherited, employs, work, renewed, sent, offered, filled, returned, filed, used, record, asked, book, ensure, listed, displayed, pick up, receive, sending, processing, marked, apply, informed, entering, request, assigning.

Task 1

Classes Identification: Identifying Potential Classes:

- Cinema Club: This is the main entity which represents the club itself.
- Member: Individuals who have a membership with the cinema club.
- Subscription: A representation of the membership status of a member.
- Secretary: Role within the club who is responsible for administrative tasks.
- Bookkeeper: Role within the club responsible for financial records.
- Ticket: Physical or digital proof of booking made for a showing.
- Booking: Process and records of a member reserving a ticket for a film.

Attribute Extractions: Properties or characteristics of the classes:

- Cinema Club: Name, Type, Location, membershipList, bookingRecords.
- Member: Name, Email Address, Membership Type, membershipStatus, SubscriptionFee, bookingHistory
- Subscription: SubscriptionDate, RenewalDate, Type.
- Secretary: EmployeeID, Name, Address, Phone, Email, MembershipList, BookingFormList, BookingSchedule, FinancialRecords.
- Bookkeeper: EmployeeID, Name, Address, Phone, email, FinancialRecords, TransactionHistory, PaymentStatus.
- Booking: ShowDate, MemberDetails, GuestName, Status.
- Ticket: CollectionStatus, memberID, showID.

Operations (Methods) Determination: Functions or responsibilities that a class can perform:

- Cinema Club: maintainFacilities, employstaff, manageMemberships, organiseEvents, addNewMember, updateMemberStatus, generateMembershipReports.
- Member: subscribe, bookTicket, reserveSeat, UpdateProfile, PayFees, CancelMembershhip, AddGuestToBooking.
- Subscription: renew, expire, validate.
- Secretary: notifyMembers, manageBookings, maintainRecords, scheduleEvents.
- Bookkeeper: recordTransactions, manageFinances, recordPayments, prepare financialReports.
- Booking: create, updateStatus, assignSeating, processPayment, cancelBookings.
- Ticket: issue, validate, notifyForCollection, generateTicket, checkInGuest.

Rejected Items: Nouns or Phrases (Not Classes):

- **Reminder**: This is an action or process of notifying members about membership renewal, acting as a feature within the notification system rather than a tangible object.
- **Desk Diary**: A physical object used for scheduling, which is a system that would be replaced by a digital calendar or scheduling component.
- **Filing Cabinet**: Physical Storage for documents, not a class within a digital system.

- **Alphabetical Order**: A method of sorting, not a class or attribute but can be an operation.
- **Constitution and list of current executive officers**: Documents and data that would likely be attributes of the Cinema Club class or could be contained within a document management component of the system, not classes on their own.
- **Booking forms and seating plans**: These would be part of the Booking or Cinema system processes, not standalone classes. They may be represented as data structures or interfaces within the system.
- **Office**: The physical office itself would not be part of the software system. The functions that take place within the office might be presented by various system classes.
- **Part-time bookkeeper, secretary**: Specific employment arrangements or titles that are roles played by individuals. The role would be attributes of an Employee Class, not separate classes themselves.

Clear Visual Table For Clarity.

Class	Attributes	Operations	Rejected Item
Cinema Club	Name, Type, Location, MembershipList, BookingRecords	MaintanFacilities, EmployStaff, ManageMemberships, OragniseEvents	-
Member	Name, Email, Address, MembershipTypes, MembershipStatus, SubscriptionFee, BookingHistory	Subscribe, BookTicket, ReserveSeat, UpdateProfile, PayFees, CancelMembership, AddGuestToBooking	-
Subscription	SubscriptionDate, RenewableDate, Type	Renew, Expire, Validate.	-
Secretary	EmployeeID, Name, Address, Phone, Email, MembershipList, BookingFormList, BookingSchedule, FinancialRecords.	NotifyMembers, ManageBookings, MaintainRecords, ScheduleEvents	-
Bookkeeper	EmployeeID, Name, Address, Phone, Email, FinancialRecords, TransactionalHistory, PaymentStatus	Create, UpdateStatus, AssignSeating, ProcessPayment, CancelBookings	
Booking	CollectionStatus, MemberID, ShowID	Issue, Validate, NotifyForCollection, GenerateTicket, CheckInGuest	
Ticket	CollectionStatus, MemberID, ShowID	Issue, Validate, NotifyForCollection, GenerateTicket, CheckInGuest	
-	-	-	Reminder, Subscription as a class, Desk Diary, Filing Cabinet, Alphabetical Order, Constitution, List of Executive Officers, Booking Forms, Seating Plans, Office as a class, Part-time Positions

Task 2

Cinema Club Methods:

- sendReminderForSubscriptionRenewal(): Sends reminder to members for annual subscription renewal.
- registerNewMember(): Registers a new member after receiving the completed application form with payment.
- updateMemberStstaus(): Updates the status of a member's subscription or booking history.
- scheduleShowings(): Organise the calender for upcoming film showings.

Secretary Methods:

- oraganiseBookingForms(): Manages the booking forms available in the foyer for members to pick up.
- processBookings(): Processes completed booking forms returned by members.
- assignTicketNumbers(): Assigns ticket numbers for shows to members bookings.
- notifyTicketAvailability(): Informs memers when their tickets are ready for collection.

Bookkeepers Methods:

- managesFinancialRecords(): Manages the club's financial records, including transcations.
- recordsMembershipPayments(): Keeos track of payments like membership fees.

Member Methods:

- renewMembership(): Renews the member's subscriptionannually.
- bookTickets(): Book tickets for a film at least one month in advance.
- requestAdditionalSeat(): Requests an additional seat for a guest during booking.
- updatePersonalInformation(): Allows members to updatet their contact information and other personal details.

Booking Methods:

- allocateSeats(): Allocate seats for bookings.
- markShowBooking(): Indicate on the seating plan which seats have been booked.
- notifyBookingStatus(): Notify members of successful or unsuccessful bookings.
- manageTicketCollection(): Manage the process for members to collect tickets.

Ticket Methods:

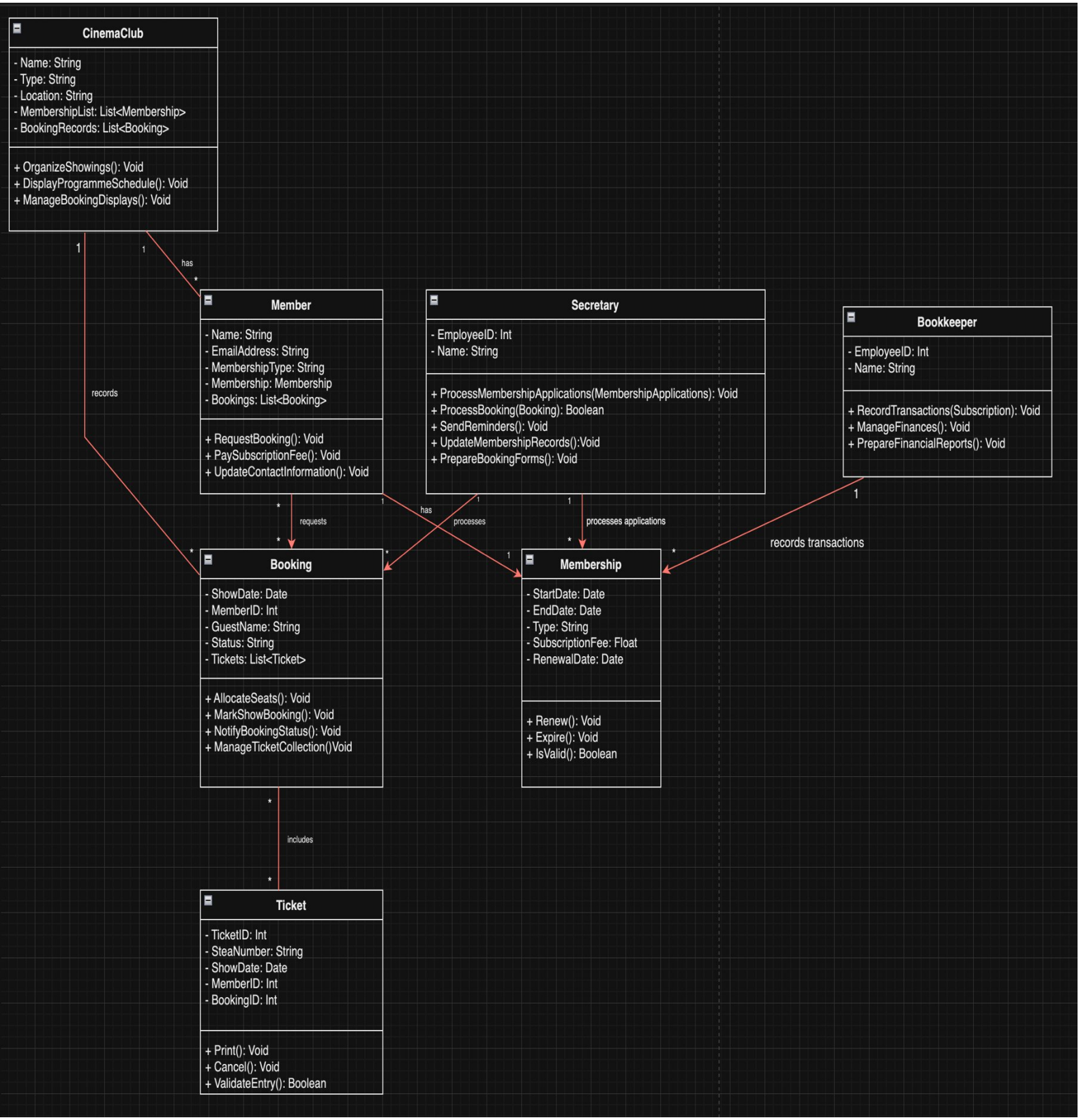
- generateTicket(): Creates a ticket for a members booking.
- markTicketAsCollected(): Marks a ticket as collected by the member.
- validateTicket(): Checks whether the ticket is valid at time of showing.

Subscription Methods:

- checkRenewalDueDate(): Checks when a members membership is due for a renewal.
- recordPayment(): Records the payment of the subscription fee.
- validateActibeSubscription(): Determines whether the members subscription is active or not.

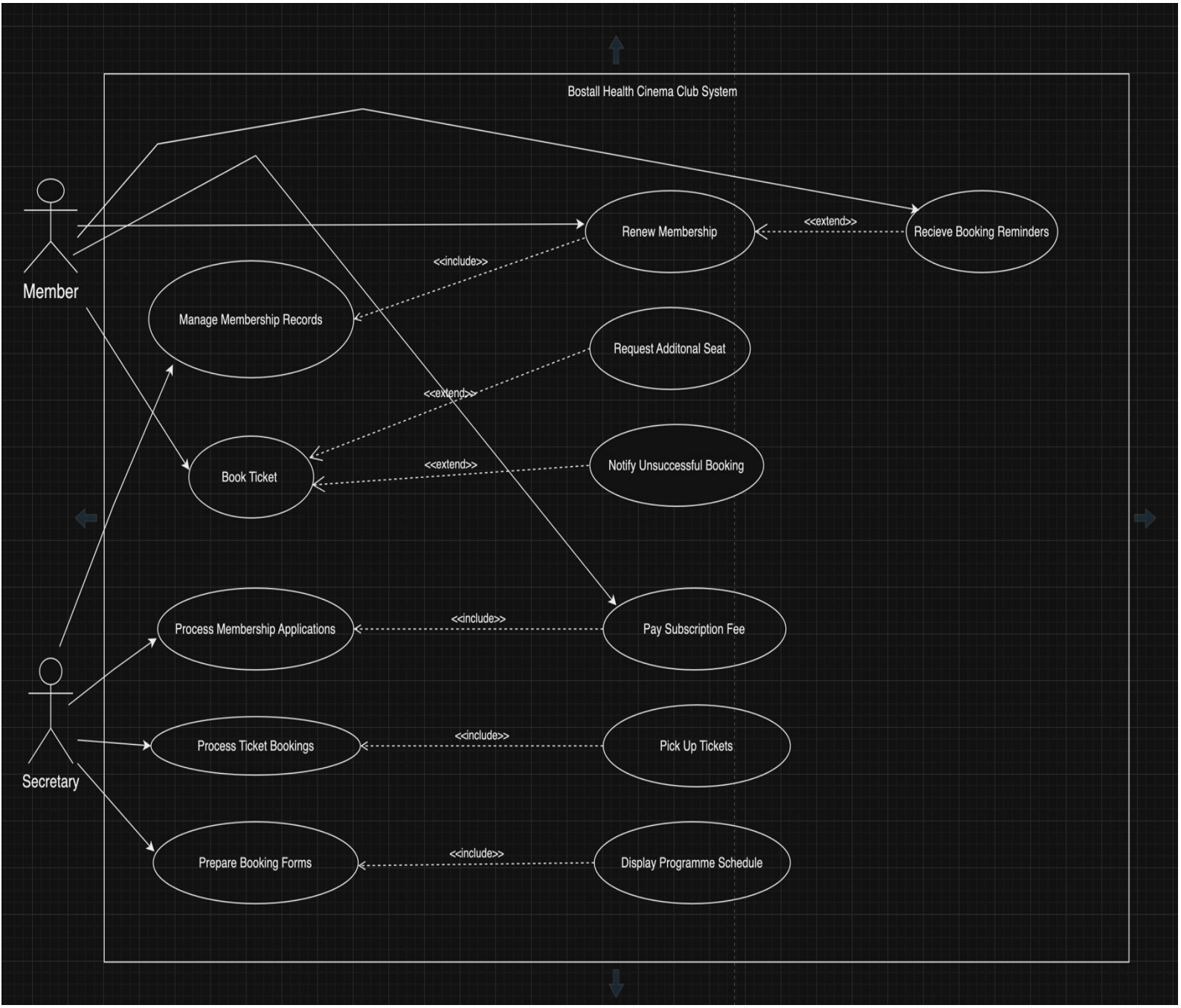
Task 3

Shows a class diagram of the case study, drawn from draw.io.



Task 4

Shows a use case diagram, drawn using draw.io.



Task 5

Use Case Description for Book Ticket

Use Case: Book Ticket
Preconditions: Membership must be valid and active.
Postconditions: The ticket is booked, and members can pick it up.
Actors: Member.
Overview: Member books a ticket for a film showing.

Main Flow of Events:

1. Actors Actions: Member initiates ticket booking.
2. System Response: The system checks for available seats and presents options.
3. Actor Actions: Member selects a preferred seat.
4. Actor Actions: Member may request an additional seat for a guest.
5. System Response: The system processes the additional seat if available.

Alternative Courses:

- If the preferred seat is not available, the system offers alternative seats.
- If no seats are available, the system notifies the member of the next available showing.

Use Case Description for Renew Membership

Use Case: Renew Membership
Preconditions: Membership is near expiration or has expired.
Postconditions: Membership is renewed for the specific term.
Actors: Member
Overview: Member renews their club membership for continued access to services.

Main Flow of Events:

1. Actor Actions: Member shows intentions of renewing membership.
2. System Response: The system presents membership options and fees.
3. Actors Actions: Member selects membership type and completes payment.
4. System Response: The system updates membership, records and confirms renewal.

Alternative Courses:

- If payment is not successful, the system prompts for payment details again.
- If the member decides not to renew, the system archives the membership details.

Use Case Description for Process Membership Applications

Use Case: Process Membership Applications
Preconditions: Applicant has submitted a membership application.
Postconditions: The application is approved or rejected, and the applicant is notified.
Actors: Secretary
Overview: The secretary processes new membership applications.

Main Flow of Events:

1. Actors Actions: The Secretary reviews the membership applications.
2. System Response: The system checks applications against criteria.
3. Actors Actions: The Secretary approves or rejects the applications.
4. System Response: The system updates records and sends a notification to the applicant.

Alternative Courses:

- If application criteria are not met, the secretary requests additional information.
- If the club is at full capacity, the system adds the applicant to a waiting list.

Use Case Description for Receive Booking Reminders

Use Case: Receive Booking Reminders
Preconditions: Member has upcoming bookings.
Postconditions: Member is reminded of their booking dates.
Actors: Member
Overview: The System sends reminders to members about their upcoming film bookings.

Main Flow of Events

1. Actor Actions: Member's booking date approaches.
2. System Response: The system generates a reminder based on the booking date.
3. Actor Actions: Member acknowledges the reminder and marks the date or sets a reminder notification.
4. System Response: The system sends the reminder to the member.

Alternative Courses:

- If the member has opted out of reminders, no reminder is sent.
- If the member's email is not valid, the system attempts to send a reminder by another method, if available.

Use Case Description for Manage Membership Records

Use Case: Manage Membership Records
Preconditions: Membership records exist and are accessible.
Postconditions: Membership records are updated, maintained, and accurate.
Actors: Secretary
Overview: Secretary maintains and updates member's records within the club's database.

Main Flow of Events:

1. Actor Actions: The Secretary accesses the membership records.
2. System Response: The system presents the current membership records.
3. Actor Actions: The Secretary updates any changes to member information or membership status.
4. System Response: The system saves the updates and confirms accuracy.

Alternative Courses:

- If records are found to be missing or incomplete, the Secretary starts the information-gathering process.
- If a record needs to be archived due to membership termination, the Secretary processes the archiving.

Use Case Description for Request Additional Seat.

Use Case: Request Additional Seat
Preconditions: Member has a booking and wishes to add a seat for a guest.
Postconditions: An additional seat is booked for the member's guest, if available.
Actors: Member
Overview: Member requests one additional seat for a film showing alongside their booking.

Main Flow of Events:

1. Actor Actions: Member would like to add a guest seat to their booking.
2. System Response: The System checks for additional seat availability.
3. Actor Actions: If available, the member confirms the addition of the guest seat.
4. System Response: The system reserves the seat and updates the booking details.

Alternative Courses:

- If no additional seats are available, the member is informed, and no changes are made.
- If the member chooses not to proceed with the additional seat, the booking remains unchanged.

Use Case Description for Notify Unsuccessful Booking.

Use Case: Notify Unsuccessful Booking
Preconditions: Member has attempted to book a ticket without success.
Postconditions: Member is informed of the unsuccessful booking attempt.
Actors: Secretary
Overview: The Secretary notifies members when their booking attempts are unsuccessful.

Main Flow of Events:

1. Actors Actions: The Secretary identifies unsuccessful booking attempts.
2. System Response: The System prepares a notification message.
3. Actor Actions: The Secretary sends the notification to the member.
4. System Response: The System logs the communication for record-keeping.

Alternative Courses:

- If member contact information is outdated, the Secretary seeks to obtain current contact information.
- If the member has a preference for communication methods, the Secretary uses the preferred method.

Use Case Description for Pay Subscription Fee

Use Case: Pay Subscription Fee
Preconditions: The membership renewal period is due.
Postconditions: The subscription fee is paid, and membership is extended.
Actors: Member
Overview: Member pays their annual subscription fee to maintain membership status.

Main Flow of Events:

1. Actor Actions: Member proceeds towards the payment of the subscription fee.
2. System Response: The system presents payment options and processes payment.
3. Actors Actions: Member completes the payment using the chosen method.
4. System Response: The system updates the membership status and confirms the transaction.

Alternative Courses:

- If payment fails, the system prompts for a retry or alternative payment method.
- If the member does not complete payment, the system may send a reminder or suspend membership benefits.

Use Case Description for Pick Up Tickets

Use Case: Pick Up Tickets
Preconditions: Member has successfully booked a ticket.
Postconditions: Member has collected their ticket from the club.
Actors: Member
Overview: Member collects their booked tickets from the cinema club.

Main Flow of Events:

1. Actor Actions: Member arrives to pick up tickets.
2. System Response: The system verifies member identity and booking details.
3. Actor Actions: Member confirms ticket collection.
4. System Response: The system logs the ticket collection and provides the tickets.

Alternative Courses:

- If member's booking cannot be found, the Secretary investigates the issue and resolves it.
- If the member wishes to collect tickets for multiple bookings, the secretary processes all collections at once.

Use Case Description for Display Programme Schedule

Use Case: Display Programme Schedule
Preconditions: The programme schedule for the season is available.
Postconditions: Members are informed about the film schedule.
Actors: Secretary
Overview: The Secretary ensures that the programme schedule is visible to members for them to plan their bookings.

Main Flow of Events:

1. Actors Action: The Secretary prepares the programme schedule.
2. System Response: The system displays the schedule in the club's foyer and online platforms.
3. Actor Actions: The Secretary updates the schedule as needed.
4. System Response: The System ensures the most current schedule is always displayed.

Alternative Courses:

- If there are changes or cancellations in the programme, the Secretary promptly updates the schedule.
- If members request a printable schedule secretary provides a printout or downloadable version.

Use Case Description for Process Ticket Bookings

Use Case: Process Ticket Bookings
Preconditions: Member has requested a booking for a ticket.
Postconditions: Ticket booking is processed, and the outcome is communicated to the member.
Actors: Secretary
Overview: The secretary processes the ticket bookings made by members for film showings.

Main Flow of Events:

1. Actors Action: The Secretary receives the ticket booking requested.
2. System Response: The system checks for available shows and seats.
3. Actor Actions: The Secretary confirms the booking if seats are available ppr communicates with the member of not.
4. System Response: The system finalises the booking and updates the records.

Alternative Courses:

- If the desired show is fully booked, the Secretary informs the member and suggests an alternative showtime.
- If the member has special requirements like accessibility needs, the Secretary accommodates these when processing the booking.

Use Case Description for Prepare Booking Forms

Use Case: Prepare Booking Forms
Preconditions: The programme schedule is finalised for the season.
Postconditions: Booking forms are available for members to request tickets.
Actors: Secretary
Overview: The Secretary prepares and distributes the booking forms for members based on the scheduled programme.

Main Flow of Events

1. Actor Actions: The Secretary compiles the details of the film programme into a booking form.
2. System Response: The System provides a template or previous booking forms for reference.
3. Actor Actions: The Secretary populates the booking form with the current programme details.
4. System Response: The System makes the booking forms available to members.

Alternative Courses:

- If there are changes to the film schedule after the forms have been distributed, the secretary updates the redistributes the forms.
- If a member requests a booking form by mail, the secretary sends it through the postal service.

Task 6: Word Count: 995

In the area of software development, the potential to create robust and scalable systems is countless. Object-Oriented Design (OOD) stands as a pillar in this concept, offering a structured approach to problem-solving by modelling software as a collection of interacting objects. This paradigm is dependent on fundamental principles such as encapsulation, polymorphism, and inheritance, which helps software that is modular reusable, and adaptable. Alongside OOD, the basics of efficient computation lie in the adept use of data structures like stacks, queues, vectors, and linked lists. These structures form the backbone support of data storage and manipulation which allows developers to implement complex algorithms and functions with optimised performance and resource management. This report will explain the features of OOD and the fundamentals of data structures that underpin software development, highlighting their synergetic roles in creating solutions that stand the test of time and technology.

Encapsulation is a foundational concept in object-oriented design that restricts direct access to an object's data and methods, bundling the data with the code that operates on it. It acts like a protective wrapper that prevents external interface and misuse of an object's internal workings. For instance, consider a 'Member' class in a cinema club system that contains private attributes such as 'name' and 'membershipNumber'. These details are hidden from direct manipulation and instead, public methods like 'UpdateMemberDetails()' are provided, ensuring that changes to the member's information are only made in controlled ways, ensuring data integrity and security. This control is in place to avoid accidental changes that could disrupt how the system works. It ensures everything stays consistent and operates as expected of the system.

Polymorphism, which means in many forms in Greek, is a concept in object-oriented design that allows objects to be treated as instances of their parent class rather than their actual class. This enables a single function to interact with different types of objectes and perform actions in various ways. For example, in a cinema club, a method 'notify()' could be used across different classes like 'Member', 'Secretary', and 'Bookkeeper'. Each class might implement 'notify()' uniquely – members receive booking confirmation, while staff get operational updates. Despite this, the system can 'notify()' on any object of the club's system, and the correct version will be executed, thanks to polymorphism.

Inheritance is an objected-oriented principle where a new class is created from an existing class, inheriting its properties and behaviours while allowing for a unque features or modifications. This promotes code reusability and hierarchical relationships. For example, the Bostall Health Cinema Club may have a general 'User' class with attributes like 'username' and 'password'. A 'Member' class could inherit from 'User', gaining it's basic login capabiltiies, yet extend functionality by adding 'membershipLevel' and 'bookTicket()'

method. Similarly, a ‘Staff’ class might also derive from ‘User’, adding staff-specific properties such as ‘employeeID’ and methods like ‘scheduleShifts()’. This inheritance structure streamlines the codebase and encapsulates common functionalities in the ‘User’ superclass.

Aggregation and composition are both objected-oriented design concepts that define how classes are associated. Aggregation represents a has-a relationship with a weak association, where the life cycle of the parent and child objected are not tightly coupled. Composition is a stronger from of aggregation where the child’s lifecyle is managed by the parent. For example, consider a ‘CinemaClub’ class that contains ‘Member’ objects. In aggregation, the ‘CinemaClub’ may have members, but members can exist independently of the club. In composition, the ‘Seat’ subjects would no longer exist, reflecting the strong dependent relationship of composition.

A stack is a linear data structure that follows the Last in, First Out principle, where the last element added is the first to be removed. In the context of the Bostall Health Cinema Club, a stack could manage undo operations in the ticket booking system. As changes are made to a booking, each state is pushed onto a stack. If a member makes a mistake, the most recent action can be undone by popping the top state off the stack.

A queue is a data structure that also operates on the First In, First Out principle meaning that first element added is the first to be removed. At the Bostall Health Cinema Club, a queue could manage the processing of ticket bookings. As members request tickets, their bookings are enqueued. The bookings are then processed in the order they were received, ensuring fairness and efficient service as each member is served in turn, akin to customers waiting in line.

Vectors often synonymous with arrays in many programming languages are data structures that store elements of the same type in a continuous block of memory, allowing random access to items via indices. For instance, the Bostall Health Cinema Club might use an array to hold the seating arrangement for a film shown, with each index representing a seat and its occupancy status, enabling quick access and update of seats as members make or cancel bookings.

A linked list is a collection of elements called nodes, each containing data and a reference to the next node, forming a sequence. Linked lists are more flexible than arrays because they don’t need consecutive memory spaces and can grow or shrink by adding or removing nodes, simplifying insertion and deletion of elements. The Bostall Health Cinema Club could use a linked list to manage a waiting list for popular film showings, efficiently adding or removing members as tickets become available or reservations are cancelled.

In conclusion, the principle of object-oriented design and the strategic employment of data structures are foundational to modern software development. Concepts such as encapsulation, polymorphism, inheritance, and aggregation/composition, assist in building software that is both clear and adaptable, facilitating code reuse and system scalability. Similarly, the use of data structures like stacks, queues, vectors, and linked lists enables efficient data management and manipulation, crucial for performance. Together, these

concepts form the foundation of developing sophisticated software solutions, highlighting their indisputable role in tackling complex programming challenges and igniting the flames of innovations in software development.

References:

Case Study

- Assignment Case Study of Bostall Health Cinema Club.

Object-Oriented Design

- Objected-Oriented Design: https://www.geeksforgeeks.org/oops-object-oriented-design/?ref=header_search
- Object-Oriented Design: https://en.wikipedia.org/wiki/Object-oriented_design
- Object-Oriented Design: https://www.geeksforgeeks.org/oops-object-oriented-design/

Encapsulation

- Encapsulation: https://en.wikipedia.org/wiki/Encapsulation_(computer_programming)
- Encapsulation: https://www.sumologic.com/glossary/encapsulation/
- Encapsulation: https://www.techtarget.com/searchnetworking/definition/encapsulation
- Encapsulation: https://www.upgrad.com/blog/encapsulation-in-oops/

Polymorphism:

- Polymorphism: https://en.wikipedia.org/wiki/Polymorphism_(computer_science)
- Polymorphism: https://stackoverflow.com/questions/1031273/what-is-polymorphism-what-is-it-for-and-how-is-it-used
- Polymorphism: https://stackoverflow.com/questions/1031273/what-is-polymorphism-what-is-it-for-and-how-is-it-used
- Polymorphism: https://www.techtarget.com/whatis/definition/polymorphism
- Polymorphism: https://www.sumologic.com/glossary/polymorphism/

Inheritance:

- Inheritance: https://www.sumologic.com/glossary/encapsulation/
- Inheritance: https://medium.com/@andrewkoenigbautista/inheritance-in-object-oriented-programming-d8808bca5021
- Inheritance: https://stackify.com/oop-concept-inheritance/
- Inheritance: https://medium.com/@fullstacktips/oops-concepts-is-a-has-a-inheritance-

fa5caf2d85ea#:~:text=The%20HAS%2DA%20relationship%20defines,Association%2C%20Aggregation%2C%20and%20Composition.

Aggregation/Composition:

- Aggregation: https://medium.com/@bindubc/association-aggregation-and-composition-in-oops-8d260854a446#:~:text=Aggregation%20is%20a%20specialized%20form,or%20child%20and%20vice%20versa

- Aggregation: https://www.infoworld.com/article/3029325/exploring-association-aggregation-and-composition-in-oop.html
- Aggregation/Composition: https://medium.com/swlh/aggregation-vs-composition-in-object-oriented-programming-3fa4fd471a9f
- Aggregation: https://www.ibm.com/docs/en/rsm/7.5.0?topic=diagrams-aggregation-relationships
- Aggregation/Composition: https://www.linkedin.com/advice/1/how-do-you-apply-composition-aggregation

Stacks/Queue:

- Stacks: https://www.geeksforgeeks.org/lifo-principle-in-stack/
- Queue: https://www.geeksforgeeks.org/queue-data-structure/?ref=header_search
- Stacks/Queue: https://byjus.com/gate/difference-stack-and-queue-data-structures/#:~:text=The%20stack%20data%20structure%20implements,known%20as%20the%20pop%20operation.

Vectors:

- Vectors: https://www3.ntu.edu.sg/home/ehchua/programming/cpp/cp8_Template.html#:~:text=(In%20computing%2C%20a%20vector%20refers,mathematical%20n%2Dcomponent%20vector.)

Linked List:

- Linked List: https://www.geeksforgeeks.org/data-structures/linked-list/
- Linked List: https://www.simplilearn.com/tutorials/data-structure-tutorial/types-of-linked-list#:~:text=A%20linked%20list%20is%20a,list%20is%20called%20the%20tail.