Report Three: Fort Hays State Movie Theaters

CSCI 441

Professor: Hieu Vu
Members: Jakob Schaefer, Derek Litke, Andrew Carter, Hanyong Yoon, David Sowles, Ryan Smith

GitHub - https://github.com/SrgtKillerSpark/fhsumovietheaterwebsite
YouTrack - https://fhsmt.youtrack.cloud/dashboard?id=169-0

Member Contribution

Jakob Schaefer – summary of changes, formatting, responsibility matrix, design of tests

Hanyong Yoon – combining information from previous reports, final checks and confirmations

Derek Litke – summary of changes, user interface design and implementation

Andrew Carter – history of work

David Sowles – NA

Ryan Smith - Effort Estimation using Use Case Points, Formatting, Use Case Diagram, Class Diagram, Design Patterns, Object Constraint Language

Responsibility Matrix

Sub-Project / Task	Jakob	Hanyong	Derek	Andrew	David	Ryan
Cover Page & Contributions	16%	16%	16%	16%	16%	16%
Work Assignment	16%	16%	16%	16%	16%	16%
Customer Statement of Requirements	20%	20%	20%	20%	0%	20%
System Requirements	20%	20%	20%	20%	0%	20%
Functional Requirements Specification	20%	20%	20%	20%	0%	20%
Effort Estimation using Use Case Points	20%	20%	20%	20%	0%	20%
Interaction Diagrams	20%	20%	20%	20%	0%	20%
System Architecture	20%	20%	20%	20%	0%	20%
Algorithms and Data Structures	20%	20%	20%	20%	0%	20%
History of Work	15%	15%	15%	40%	0%	15%
References	16%	16%	16%	16%	16%	16%

Summary of Changes

- Changed diagrams to white with black lettering
- Expanded detailed narratives for all actors
- Setting up database during demo 2 instead of demo 1
- Setting up theater manager/employee in demo 2 instead of demo 1
- Based class diagram and OCC added to report 3

Table of Contents

    b. Architecture styles
    c. Mapping subsystems to Hardware
    d. Connectors and network protocols
    e. Global control flow
    f. Hardware Requirements
7. Algorithms and Data Structures
8. History of Work
9. References

Customer Statement of Requirements

Problem Statement

Customer (Actor)

As a person that goes to the movies often, buying a ticket has always been a problem for me. Most of the time when the movie is a popular release, I often find myself waiting in a long line to purchase a ticket. Sometimes, I would go through the trouble of waiting just for the ticket to be sold out. There are many problems besides the long line and the unavailable ticket. For example, even when the ticket is available, I don't know what seat is available until it's my turn to purchase the ticket. This is problematic, especially more so if I am with my family or friends, that's because we want to sit together and the only seats remaining are scattered or not enough seats are available for my group.

Another issue I face with the process of buying a movie ticket is not receiving an efficient confirmation, when I purchase a movie ticket it's usually just a small stub that can be easily lost. While some theaters send an email confirmation in case you lose the ticket, old theaters don't have a system like this. If I were to lose my ticket, there is no way to prove my purchase history, and I will have to rebuy if I still want to watch the movie, all while dealing with this frustrating situation. Beyond this issue, I like to keep track of what I've watched in the past, part of the theater experience is remembering the movie. It's almost impossible to remember every movie I've watched in the past, and without a proper system that shows the history of movies I've watched, I would have to rely on going through the collections of stubs that I've collected over the years. Even then, over time, stubs get worn out and it's hard to read the prints or sometimes impossible to read them. Sometimes stubs can get lost, and I'm faced with the problem of trying to remember the movie that I've watched.

In a busy theater, I cannot stress the importance of convenience. Many businesses are modernizing their services, whether it's for ordering food, buying tickets, or shopping. With services evolving with time, it's important for movie theaters to adapt to this evolution.

Instead of making customers feel like they're going through an unnecessary obstacle, the process of buying a ticket should be simple. Sometimes, I feel like I spend more time dealing with the tedious process of buying tickets than I do enjoying the movie.

I think about accessibility too. While I am blessed enough to be in a healthy body, that is not the case for everyone. Everyone can't stand in a long line for a long time. Whether it's the elderly, parents with young children, or people with disabilities, the process of purchasing a ticket can be overwhelming. While kiosks might be a good option to solve this problem, they're often outdated and confusing. A more user-friendly system must be employed for everyone to enjoy the "movie theater" experience.

With all these issues combined, it makes going to the movies less appealing. Why shouldn't I just stay home and watch something at the comfort of my home using various streaming platforms. Instead, every time I go to the movies, I feel like I am fighting through tedious procedures of outdated system that should be simple and convenient. What I want is a simple and reliable solution. I want a system where I can log in, browse what movies are available, and see the available seating charts. I want to pick exactly where I want to sit, purchase tickets without having to wait in line, and get an email confirmation of my purchase right away. I want access to the history of my purchases to view what I've seen in the past to remember my experiences. Generally, what I want is for the ticket-buying process to be stress-free. I want to enjoy going to the movies and not face the frustration of purchasing a ticket.

Staff/Admin (Theater Employee) (Actor)

Working at a movie theater can be rewarding, that's if everything is going according to plan. One problem I face is keeping the theater information accurate; Some examples are having the correct movies posted, accurate movie times, proper ticket prices, list of concession items. I must update all the information related to the movie theater manually. This takes up a lot of time and can sometimes be the reason for mistakes. If I forget to post the correct information, customers will get confused and possibly angry.

Another issue I face is with real-time updates. There are times when movie tickets sell out fast and the showing for it needs to be updated or I must cancel a showing because technical issues like projector not working properly. Making these changes isn't simple, there is no central system I can use to update everything. I would have to adjust everything manually and often leave notes for coworkers, which can lead to miscommunication. Without real-time updates it's difficult to answer the customer's questions like "How many tickets are left?" or "What's the most popular movie currently?" Finding the correct answers for these questions means going through multiple spreadsheets that are manually updated, which can be time consuming and inaccurate.

What I need is a reliable system where I can update everything in one place. I need a system where I can log in with my employee account, update movies and showtimes and concession items, change ticket pricing without worrying about if the information is correct or not. I need the system to have real-time update so I can answer customer questions without hesitation. I need a system that will improve the quality of life when it comes to my job. The system will help me make less mistakes and with it I can provide better service for the customer.

Theater manager (Actor)

As a manager my job is to make sure the entire theater operation is running smoothly. The most difficult part of my job is keeping track of employees. I need to keep track of who's working, what time they've arrived, and how many hours they worked to process their payroll. Currently, this is a manual and time-consuming process. Using time sheets, paper schedules, and outdated clock-in system is not ideal in this day and age. Keeping accurate reports is important for me as a manager. I must keep track of sales numbers, ticket performance, and customer activity. However, without a centralized system keeping track of these reports is a tedious task that can sometimes take hours.

To make my jobs easier, I need a modern centralized system where employees can clock in and out, track their working hours, and calculate their payroll. It would save me time and

headache if I could rely on the system to handle the accuracy of these records automatically. This system would help me tremendously in a way where I wouldn't have to spend hours checking employee timesheets or fixing mistakes that are caused by manual entry. Overall, I need to be able to save time, reduce errors, and make the theater a place where both the customers and staff can have positive experience.

Glossary of Terms

Technical Terms

Digital ticketing – Electronic process of purchasing a ticket through the app. Tickets are delivered digitally

Showtime - A scheduled movie screening with a fixed start time, shows specific auditorium it's linked too.

Purchase History - Record showing all previously purchased items from the movie theater.

Admin panel - Digital interface where employees and managers can update movie schedules, ticket pricing, seat availability, and concession.

User Account - User profile containing login credentials.

Customer Dashboard – Personalized area in the app that shows past purchases, payment preferences.

Non-Technical Terms

Auditorium - The room where the movie is played with projector, screen, and seats.

Seating Charts – A map of available and sold seats for specific showtime. Customers use this to pick out their seats.

Concessions – Snacks and drinks sold at the theater can be purchased using the app.

Accessibility Features – Design consideration that makes the app and theater experience usable for everyone.

Concept definition

Account – Stores customer credentials (email, password) and profile details

Authentication – validates login information for secure access

Catalog UI – Interface that displays movie listings

Search Service – allows searching for movies

Movie – official record of film details

Scheduling Service – handles the creation and management of showtimes

Showtime – represents date/time and auditorium for a movie

Seating UI – displays seating charts with availability

Seat Map Template – defines the layout of seats in an auditorium

Seat Inventory – tracks available, sold, or blocked seats

Seat Hold Manager – holds seats temporarily and releases them if not purchased

Pricing Engine – calculates ticket prices, fees, and charges

Order Service – creates orders based on selected seats

Order – stores order details including items, fees, taxes, and status

Ticket Issuer – issues tickets once payment is confirmed

Ticket – stores finalized ticket details

Notification Service – sends confirmation/cancellation emails

Cancellation Service – handles order cancellations

Refund – records and processes returned payments

Reporting Service – generates sales and booking reports

System Requirements

Functional Requirements

Req ID	Priority	Requirement
REQ-1	High	The system shall allow users to create an account with their email
REQ-2	High	The system shall display movies and showtimes
REQ-3	High	The system shall allow customers to search for movies
REQ-4	High	The system shall provide seat maps for each show time
REQ-5	High	The system shall allow customers to purchase tickets and select seats
REQ-6	Medium	The system shall allow customers to cancel purchased tickets
REQ-7	High	The system shall send ticket confirmations via email
REQ-8	Medium	The system shall allow customers to view past ticket history
REQ-9	High	The system shall allow admins to add/remove movies
REQ-10	High	The system shall allow admins to update showtimes
REQ-11	High	The system shall allow admins to adjust ticket pricing
REQ-12	Medium	The system shall provide admins with sales and booking reports

Nonfunctional Requirements (FURPS+)

Req ID	Priority	Requirement
REQ-13	High	The system shall load the homepage in under 2 seconds
REQ-14	High	The system shall support mobile and desktop devices
REQ-15	High	The system shall remain available at least 99% of the time
REQ-16	Medium	The system shall support accessibility features
REQ-17	High	The system shall encrypt all sensitive data, including payment information
REQ-18	Medium	The system shall scale to support an increasing number of users and theaters

User Interface Requirements

Req ID	Priority	Requirement
REQ-19	High	The system shall present customers with a clear movie listings page, showing title, showtime, rating, and availability.
REQ-20	High	The system shall provide an interactive seating chart for customers to select seats
REQ-21	Medium	The system shall provide admins with a simple dashboard to manage movies and showtimes
REQ-22	Medium	The system shall provide managers with a dashboard summarizing sales, staffing, and payroll

Functional Requirements Specification

Stakeholders

The following are the stakeholders that will benefit the most from our movie ticketing system.

- Customers – Moviegoers, purchases tickets, choose seats, manage purchase history, and receive confirmations.
- Theater Employees – Responsible for updating movies, showtimes, prices, and aids with customer's needs.
- Theater Manager – Overlooks the entire theater operations, sales, staffing, and payrolls.
- Theater Owner – Focuses on the whole theater's profitability, compliances, and customer satisfaction.

Actors and Goals

Initiating Actors

Customers – Find showtimes, see available seats, choose seats, buy ticket, get ticket confirmation email, view/cancel purchases.

Theater Employees – Add and remove listings, showtimes, and pricing in real-time. Help with customers' needs.

Theater Manager - Access manager dashboard and audit reports.

Participating Actors

Payment Gateway - Authorize payments securely.

Email Service - Send ticket purchase confirmation and cancellation email.

Use Cases

Casual Description

Customer

UC-C1: Create Account

Customers sign up using email and password.

REQ-1

UC-C2: Browse and Search Movies

Customers can view movie listings and search for titles.

REQ-2, REQ-3, REQ-19

UC-C3: View Showtimes and Seat Map

Customers can view shows for movies and view seats using seating charts that have availability indicators and accessibility seating.

REQ-4, REQ-20

UC-C4: Purchase Tickets

Customer selects seats, enters payment, confirms order. System processes payment and issues order confirmation.

REQ-5, REQ-7

UC-C5: Cancel Ticket

Customer cancels an order. Purchase gets refunded and the seat used becomes available again.

REQ-6

UC-C6: View Purchase History and Customer Dashboard

Customers can view previous purchases made from the theater.

REQ-8

Theater Employees

UC-E1: Manage Movies

Employees can add new movies and remove outdated ones. Change the needed metadata and assets.

REQ -9

UC-E2: Manage Showtimes

Employees create and edit showtimes and can update changes in real-time.

REQ-10

UC-E3: Adjust Ticket Pricing

Employees can update prices for the tickets.

REQ-11

UC-E4: View Sales and Booking reports

Employees can view reports for sales, booking by its movies, date range, and auditorium.

REQ-12

UC-E5: Use Admin Dashboard

Employees can access a simplified dashboard to manage movies and showtimes from one place.

REQ-21

Theater Manager

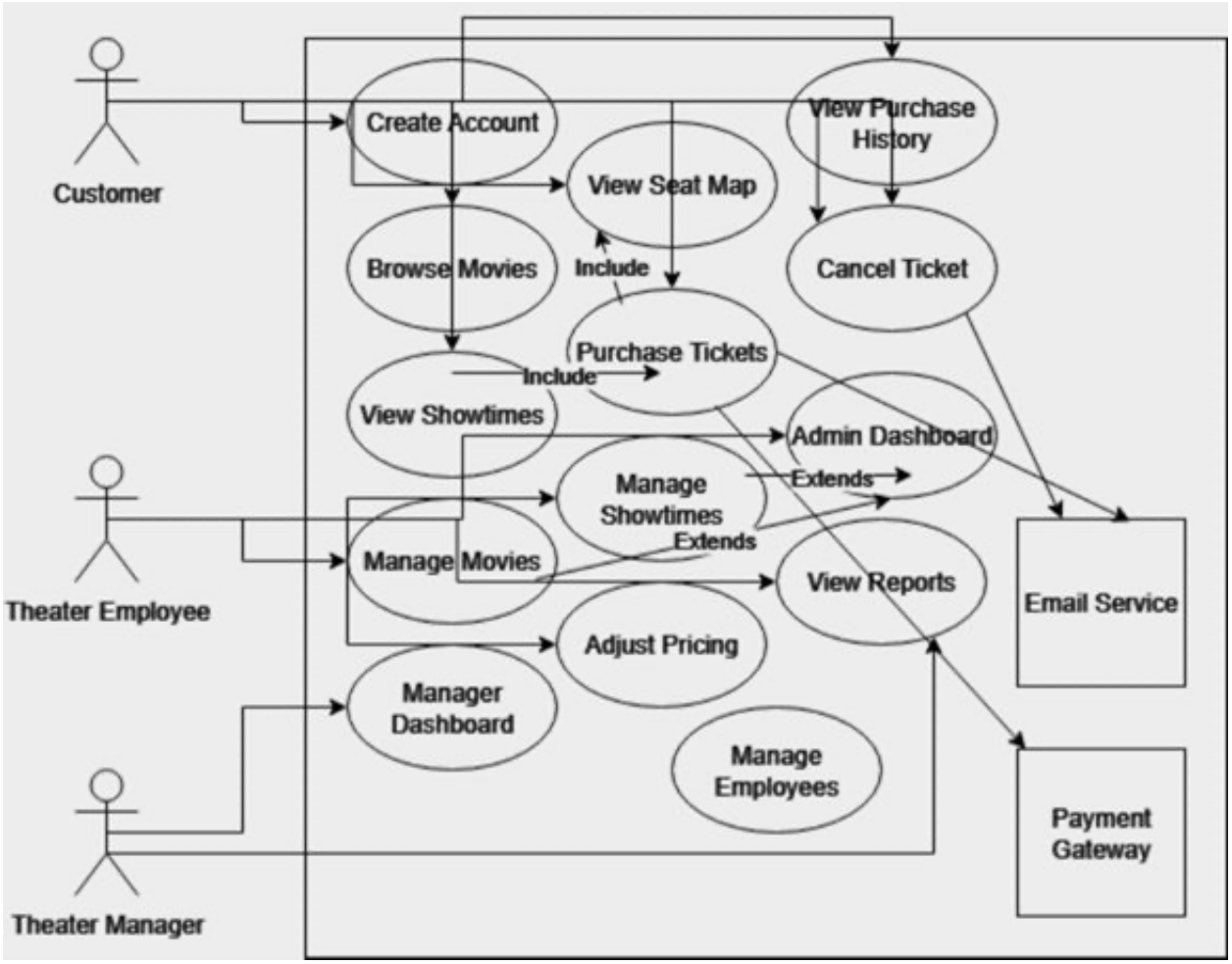
UC-M1: View Manager Dashboard

Managers have access to the Manager dashboard that shows summaries for sales, staffing/payroll information.

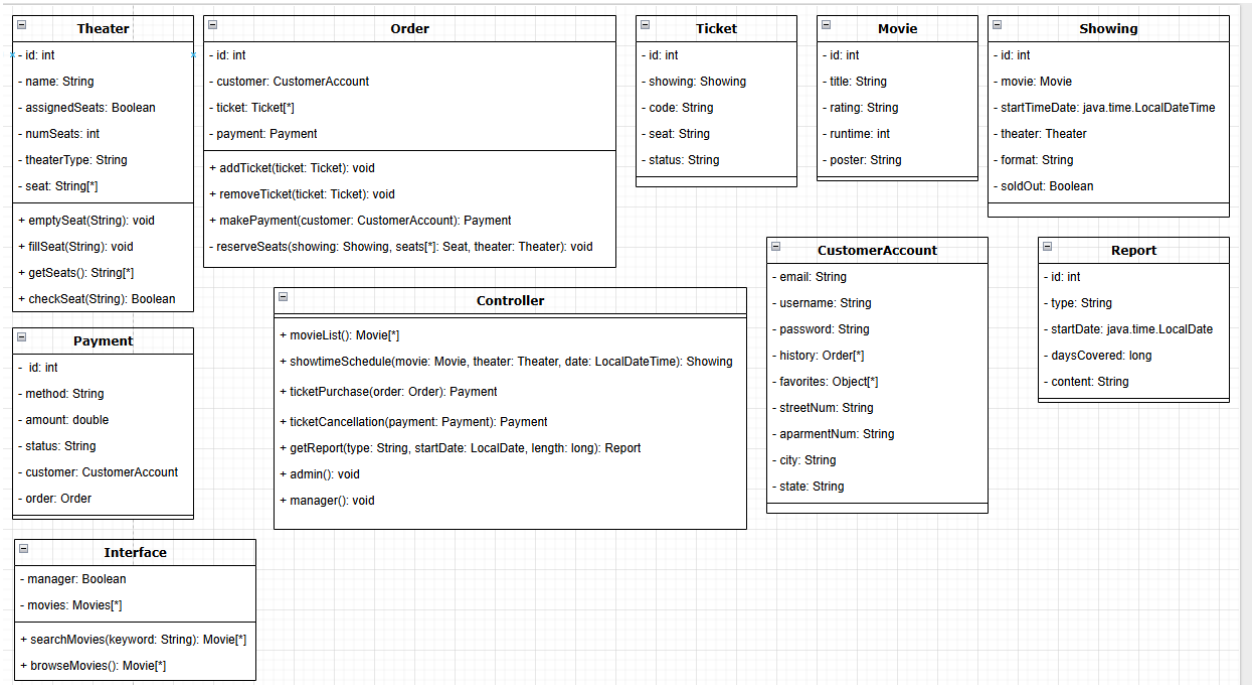
REQ-22, REQ-12

Requirements responding to all use cases: REQ-13, REQ-14, REQ-15, REQ-16, REQ-17, REQ-18.

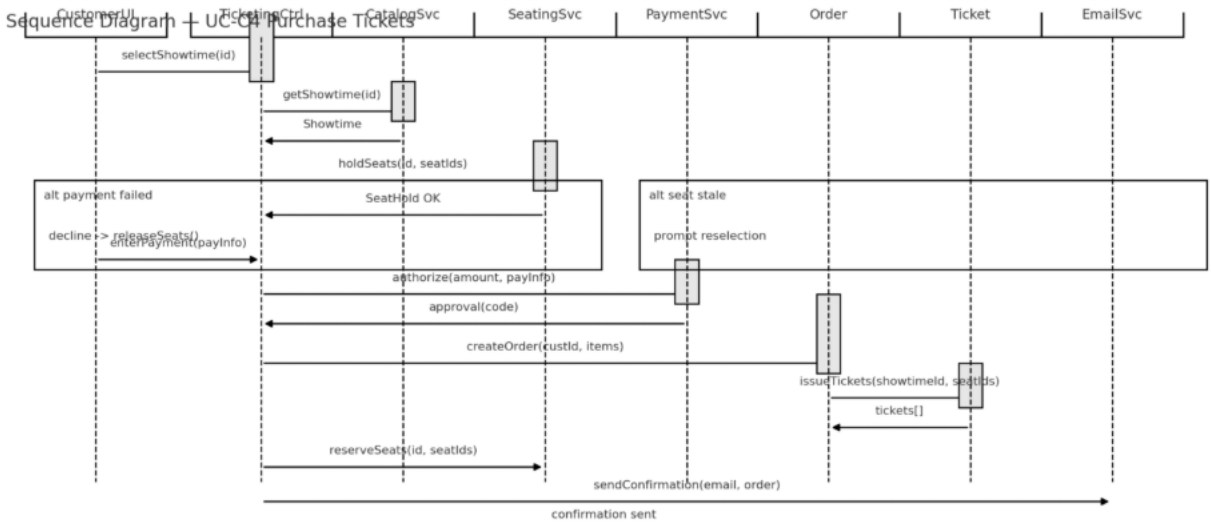
Use Case Diagram

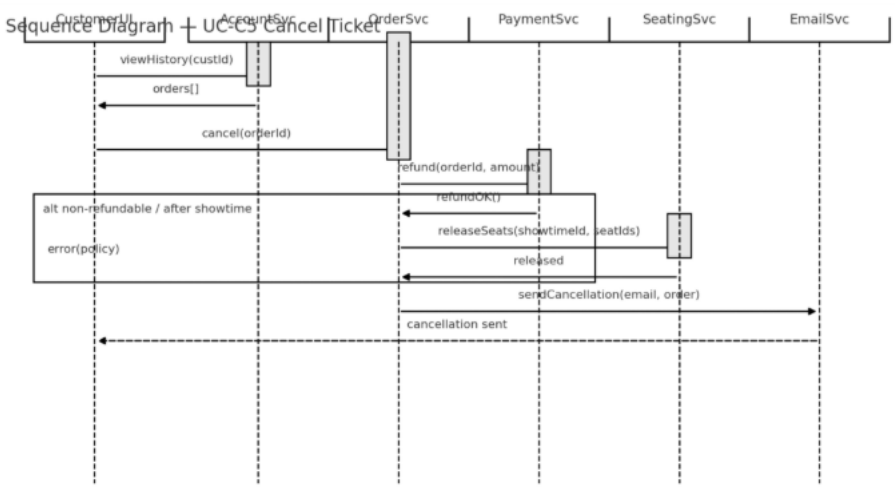


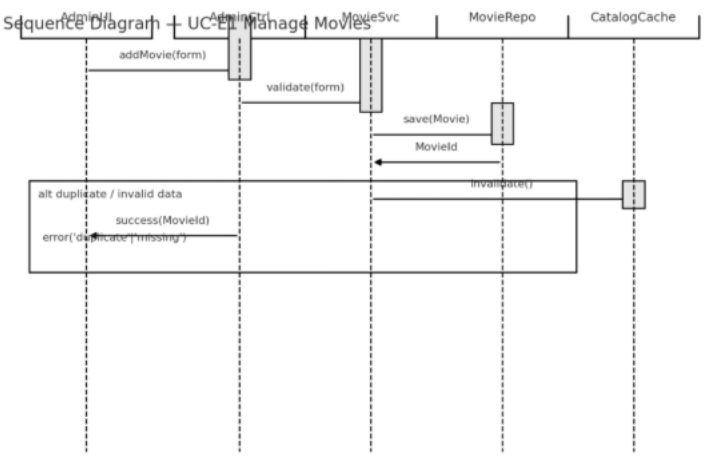
Class Diagram



Interaction Diagram







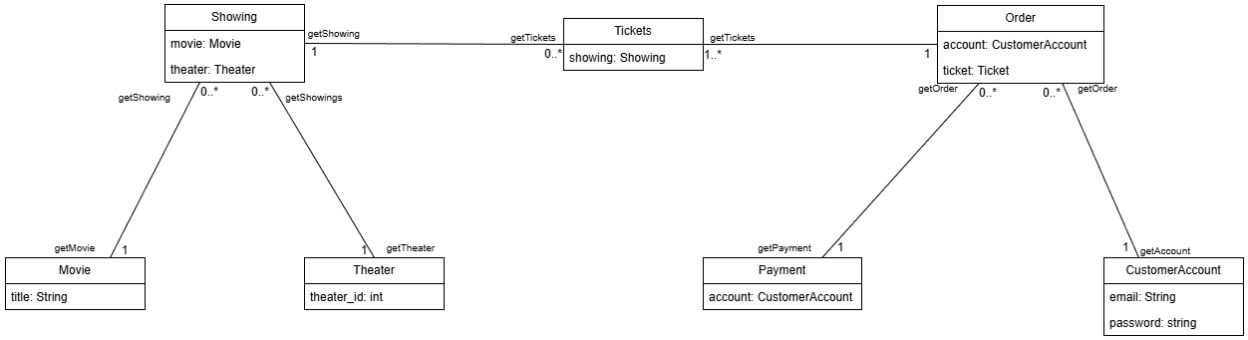
The design pattern that helped shape our program is the command design pattern. We decided that as a web-based service using the command pattern would be the most beneficial for processing multiple requests and for managing our seating inventory and keeping that as up to date as possible.

Traceability Matrix

	UC-C1	UC-C2	UC-C3	UC-C4	UC-C5	UC-C6	UC-E1	UC-E2	UC-E3	UC-E4	UC-E5	UC-M1
REQ1	X											
REQ2		X										
REQ3		X										
REQ4			X									
REQ5				X								
REQ6					X							
REQ7				X								
REQ8						X						
REQ9							X					
REQ10								X				
REQ11									X			
REQ12										X		X
REQ13	X	X	X	X	X	X	X	X	X	X	X	X
REQ14	X	X	X	X	X	X	X	X	X	X	X	X
REQ15	X	X	X	X	X	X	X	X	X	X	X	X
REQ16	X	X	X	X	X	X	X	X	X	X	X	X
REQ17	X	X	X	X	X	X	X	X	X	X	X	X
REQ18	X	X	X	X	X	X	X	X	X	X	X	X
REQ19		X										
REQ20			X									
REQ21											X	
REQ22												X

Our domain concepts evolved into our classes very naturally. There were some minor adjustments needed for some situations where there was an outlying item.

Object Constraint Language


Showing
movie: Movie
theater: Theater
getShowing
1
getTickets
0..*
Tickets
showing: Showing
getTickets
1..*
1
Order
account: CustomerAccount
ticket: Ticket
getShowing
0..*
0..*
getShowings
getOrder
0..*
0..*
getOrder
getMovie
1
1
getTheater
getPayment
1
1
getAccount
Movie
title: String
Theater
theater_id: int
Payment
account: CustomerAccount
CustomerAccount
email: String
password: string

Fully Dressed Descriptions

UC-C4 Purchase Tickets

Primary Actor – Customer

Participating Actors – Payment Gateway, Email Service

Precondition – Customer is logged in, selects a movie, and a showtime

Postcondition – Ticket purchased, confirmation email sent, seat marked as unavailable

Flow of Events

1. Customer selects movie and showtime
2. System displays seating chart with availability
3. Customer chooses seat(s)
4. Customer enters payment details
5. System sends payment to payment gateway
6. Payment gateway returns approval
7. System issues ticket, reserves seat, and records purchase
8. System triggers email service to send confirmation email

Alternate Flows

1. If payment fails, customer is shown error and asked to retry
2. If seats become unavailable mid-process, system prompts customer to reselect

UC-C5 - Cancel Ticket

Primary Actor – Customer

Participating Actors – Payment Gateway, email service

Precondition – Customer has an active ticket

Postcondition – Ticket Cancelled, seat available, refund processed, cancellation email sent

Flow of events

1. Customer navigates to Purchase History
2. Customer selects a ticket to cancel
3. System verifies eligibility (before showtime, refundable policy)

4. System contacts payment gateway to process refund
5. System updates ticket status and makes seat available
6. System triggers email service to send cancellation confirmation

Alternate flows

1. If non-refundable, system displays an error
2. If refund fails, system alerts customer to contact support

UC-E1 – Manage Movies

Primary Actor – Theater Employee

Precondition – Employee logged into Admin Dashboard

Postcondition – Movie listing created, updated, or deleted

Flow of Events

1. Employee logs into Admin Dashboard
2. Employee selects manages movies
3. Employee adds new movie (title, description, poster, runtime, rating)
4. System validates data and saves movie to database
5. Updated listings are visible to customers in real time

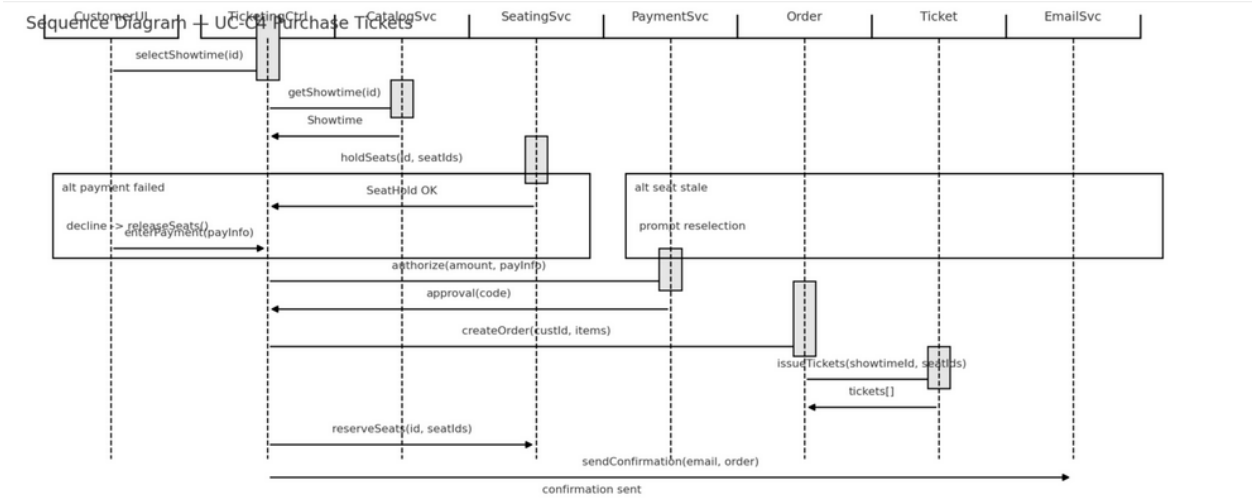
Alternate Flows

1. If data is missing, system prompts employee to complete form
2. If duplicate movie exists, system alerts employee

System Sequence Diagrams

Purchase tickets diagram

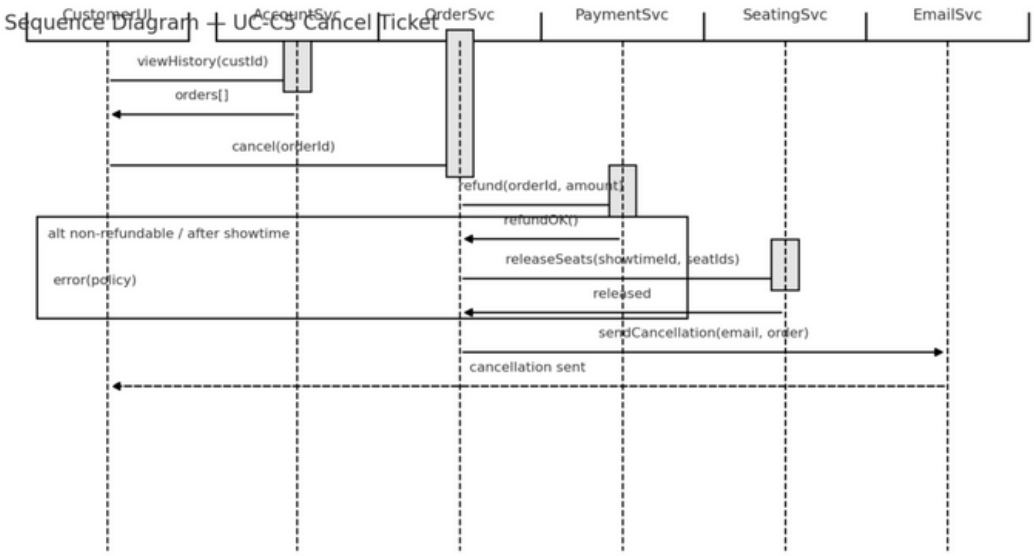
Shows the order of interactions between the Customer and different system services involved in buying a movie ticket.



1. Customer selects showtime.
2. System gets seating info.
3. Customer picks seats and enters payment info.
4. System sends payment requests to Payment service.
5. Payment is approved or declined.
6. System displays confirmation back to the customer.

Cancel Ticket diagram

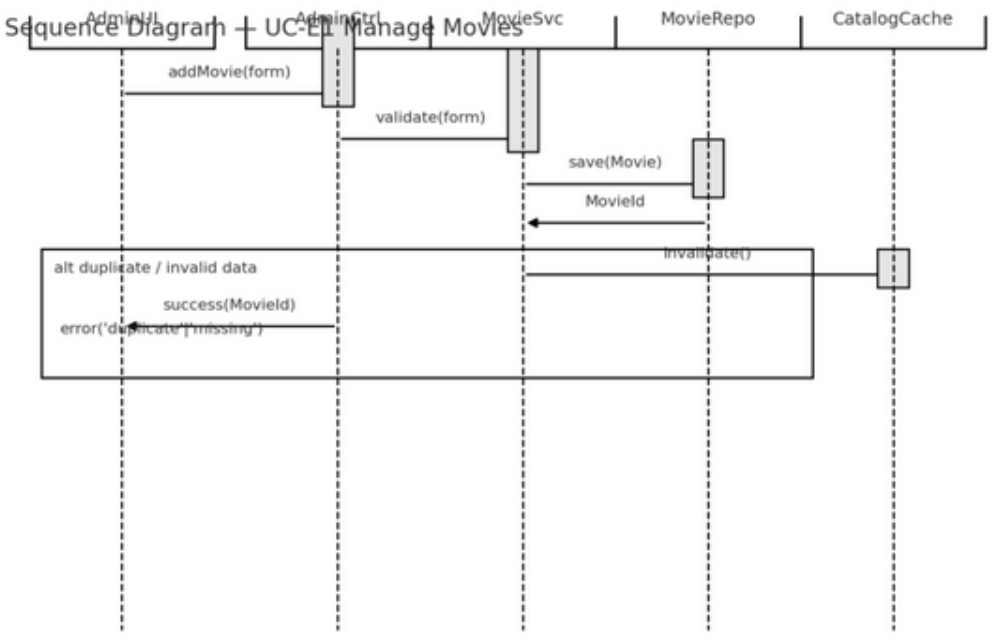
This diagram shows how different parts of the system interact when a customer cancels a previously purchased ticket.



1. Customer views purchase history.
2. Customer selects an order to cancel.
3. System processes refund.
4. Seats are released back to availability.
5. Confirmation Email Sent.
6. Customer UI receives Success Message.

Manage Movie Diagram

This diagram shows how a Theater employee uses the system to add a new movie to the theater listings through the Admin Dashboard



1. Admin fills out the movie form and submits it.
2. Controller sends the form to business logic.
3. Service validates the movie information.
4. Validation succeeds or fails.
5. System is updated so the customers can see the new movie.
6. Response is returned to the UI.

Effort Estimation using Use Case Points

Unadjusted Actor Weight (UAW)		
Actor Type	Description	Weight
Simple	The actor is another system which interacts with our system through a defined application programming interface (API).	1
Average	The actor is a person interacting through a text-based user interface, or another system interacting through a protocol, such as a network communication protocol.	2
Complex	The actor is a person interacting via a graphical user interface.	3

Unadjusted Use Case Weight (UUCW)		
Use Case Category	Description	Weight
Simple	Simple user interface. Up to one participating actor (plus initiating actor). Number of steps for the success scenario: ≤ 3. If presently available, its domain model includes ≥ 3 concepts.	5
Average	Moderate interface design. Two or more participating actors. Number of steps for the success scenario: 4 to 7. If presently available, its domain model includes between 5 and 10 concepts.	10

Complex	Complex user interface or processing. Three or more participating actors. Number of steps for the success scenario: ≥ 7. If available, its domain model includes ≥10 concepts.	15

Use Case	UAW	UUCW	Total
UC-C1	5	10	15
UC-C2	4	5	9
UC-C3	4	10	14
UC-C4	6	10	16
UC-C5	6	10	16
UC-C6	4	5	9
UC-E1	4	5	9
UC-E2	4	5	9
UC-E3	4	5	9
UC-E4	4	10	14
UC-E5	4	5	9
UC-M1	4	10	14
UCP Total			143

Technical Factor	Description	Weight	Perceived Complexity	Calculated Factor (Weight x P.C.)
T1	Distributed system (running on multiple machines)	2	0	0
T2	Performance objectives (are response time and throughput performance critical?)	1	1	1
T3	End-user efficiency	1	2	2
T4	Complex internal processing	1	0	0
T5	Reusable design or code	1	1	1

T6	Easy to install (are automated conversion and installation included in the system?)	0.5	4	2
T7	Easy to use (including operations such as backup, startup, and recovery)	0.5	5	2.5
T8	Portable	2	1	2
T9	Easy to change (to add new features or modify existing ones)	1	2	2
T10	Concurrent use (by multiple users)	1	5	5
T11	Special security features	1	1	1
T12	Provides direct access for third parties (the system will be used from multiple sites in different organizations)	1	0	0
T13	Special user training facilities are required	1	0	0
Total				16.5

Environmental Factor	Description	Weight	Perceived Impact	Calculated Factor
E1	Familiar with the development process (e.g., UML-based)	1.5	2	3
E2	Application problem experience	0.5	2	1
E3	Paradigm experience (e.g., object-oriented approach)	1	3	3
E4	Lead analyst capability	0.5	3	1.5
E5	Motivation	1	4	4
E6	Stable requirements	2	2	4

E7	Part-time staff	-1	5	-5
E8	Difficult programming language	-1	0	0
Total				11.5

UUCP	TCF	ECF
143	0.7	1.1

UCP
110.11

UCP	PF	Duration
110.11	28	3083.08

System Architecture

Identifying Subsystems

1. Customer Interface
    a. Provides the web and mobile frontends for customers to browse movies, select seats, and purchase tickets.
2. Admin Interface
    a. Provides the web dashboard for theater staff to manage movies, showtimes, and ticket pricing.
3. Backend Services
    a. Implements business logic for authentication, user management, ticketing, and payment processing.
4. Database Subsystem
    a. Stores user accounts, movies, showtimes, tickets, and sales history.

Architecture Styles

The system follows a client server architecture, where the client communicates with a central backend server that handles logic and database operations. The system can be described as a layered architeture:

1. Presentation Layer (Customer/Admin Interfaces)
2. Business Logic Layer (Backend Services)
3. Data Layer (Database Subsytem)

Mapping Subsystems to Hardware

1. Client
    a. Runs in a standard web browser or mobile browser
2. Server
    a. Runs backend services on a web server
3. Database
    a. Runs on a database server or cloud-hosted database system

Connectors and Network Protocols

1. HTTP/HTTPS
    a. Used for communication between clients and backend server
2. JDBC/SQL
    a. Used for communication between backend services and the database

3. No custom socket programming or RMI is required

Global Control Flow

1. The system is event driven. Users can perform actions in any order (log in, browse movies, purchase tickets, cancel orders).
2. The backend listens for requests and responds accordingly
3. The system is not real time, it is asynchronous and response based
4. No timers or periodic processes are required for core functionality.

Hardware Requirements

1. Client Side
    a. Any modern web or mobile browser with internet access, screen display, and basic input devices.
2. Server Side
    a. A machine capable of running a web server and backend framework (Express.js or Java/Spring) with sufficient CPU/RAM to handle concurrent users
3. Database
    a. Persistent storage system for user, movie, ticket data
4. No specialized hardware is required.

Algorithms and Data Structures

Algorithms

- Seat Selection and Hold Algorithm
    - Ensure that customers can select seats in real time without conflicts
    - Check the requested seat IDs against the SeatInventory for availability
    - If all seats are available, temporarily hold the seats in SeatHoldManager and set a timeout
    - If payment is not complete within the timeout period, automatically release held seats
- Order Processing Algorithm
    - Convert held seats into a confirmed order and associated tickets
    - Calculate total price using the PricingEngine
    - Authorize payment through PaymentGateway
    - On successful payment, created an Order object and issue corresponding ticket objects
    - Update seat status in SeatInventory from HELD to RESERVED

- Cancellation and Refund Algorithm
    - Cancel ordders and release seats while issuing refunds
    - Check if the order is eligible for cancellation
    - Update order status to “Cancelled”
    - Process refund using Refund service
    - Update seat availability in SeatInventory
    - Notify the customer vis NoticationService
- Reporting Algorithm
    - Generate sales and booking reports efficiently
    - Query orders and ticket information for the specified date range and filters
    - Aggregate data by movie, auditorium, or date
    - Format and return report object for display on manager/admin dashboard

Data Structures

- Arrays/Lists
    - Used for storing movies (MovieCatalog), showtimes (ShowtimeSchedule), tickets in an order (Order.ticket), and seat lists (Theater.seat)
- Hash Maps/Dictionaries
    - Used for mapping seat IDs to seat objects in SeatInventory and mapping movie IDs to Movie objects in MovieCatalog
- Queues
    - Used for managing seat hold timeouts in SeatHoldManager
- Objects/Classes
    - Core domain objects include CustomerAccount, Move, Showtime, Ticket, Order, Payment, and Seat

Concurrency

- Our system has limited concurrency concers, primarily in handling multiple users reserving seats simultaneously
    - Threads
        - Seat reservation and timeout release could potentially be handled in separate threads for efficiency
        - SeatHoldManager runs a background thread to release expired seat holds
- Synchronization
    - SeatInventory updates are synchronized to prevent race conditions when multiple customers attempt to reserve the same seat
    - Locks or atomic operations are used to ensure that each seat can only be held or reserved by one customer at a time

Design of Tests

Test Cases for Unit and Integration Testing

Customer Scenario – Create Account

1) Navigate to sign-up page
2) Enter all information requested
3) Submit sign-up
4) Receive email notification of account creation

Tests the creation of an account.

Customer Scenario – Login

1. Select “Login”
2. Enter username and password
3. Click “Login”
4. Directed to “Home Page”

This test case will cover one of the most common usage scenarios. Testing this will allow all our other customer use cases to start from the home page instead of making the user log in for every test.

Customer Scenario – Purchase Tickets

1) Browses available movies
2) Selects movie
3) Look at movie’s showtimes
4) Select a showtime
5) View available seats in theater
6) Select seats
7) Enter payment info
8) Confirm purchase
9) Receive confirmation, receipt, and tickets

Purchasing tickets at a location other than the movie theater is one of the most important functions of our software. This will evaluate the processes of navigating and selecting movies, theaters, and seats as well as testing our payment process.

Customer Scenario – Refund Tickets

1) Go to purchase history
2) Select purchase to refund
3) Confirm to refund purchase
4) Receive confirmation and receipt of refund

This will test our purchase history and our payment return process.

Employee Scenario – Login

1. Select “Login”

2. Enter username and password
3. Click “Submit”
4. Directed to “Admin Dashboard”

Will test login for user will elevated permissions.

Employee Scenario – Add Movie

1) Select “Add Movie”
2) Enter movie information and upload image(s)
3) Submit new movie

Tests that an employee can add a movie, and a normal user can’t add a movie.

Employee Scenario – Add Showing

1) Select movie
2) Select “Add Showing”
3) Input showtime, date, and theater
4) Submit showing

Tests that an employee can add a showing, and a normal user can’t add a showing.

Employee Scenario – Modify Showing

1. Select showing
2. Click “Modify”
3. Make changes to theater, movie, and showtime
4. Submit changes to showing

Tests that an employee can modify a showing, and a normal user can’t modify a showing.

Employee Scenario – Delete Showing

1. Select showing
2. Click “Modify”
3. Click “Delete Showing”
4. Confirm the deletion of showing

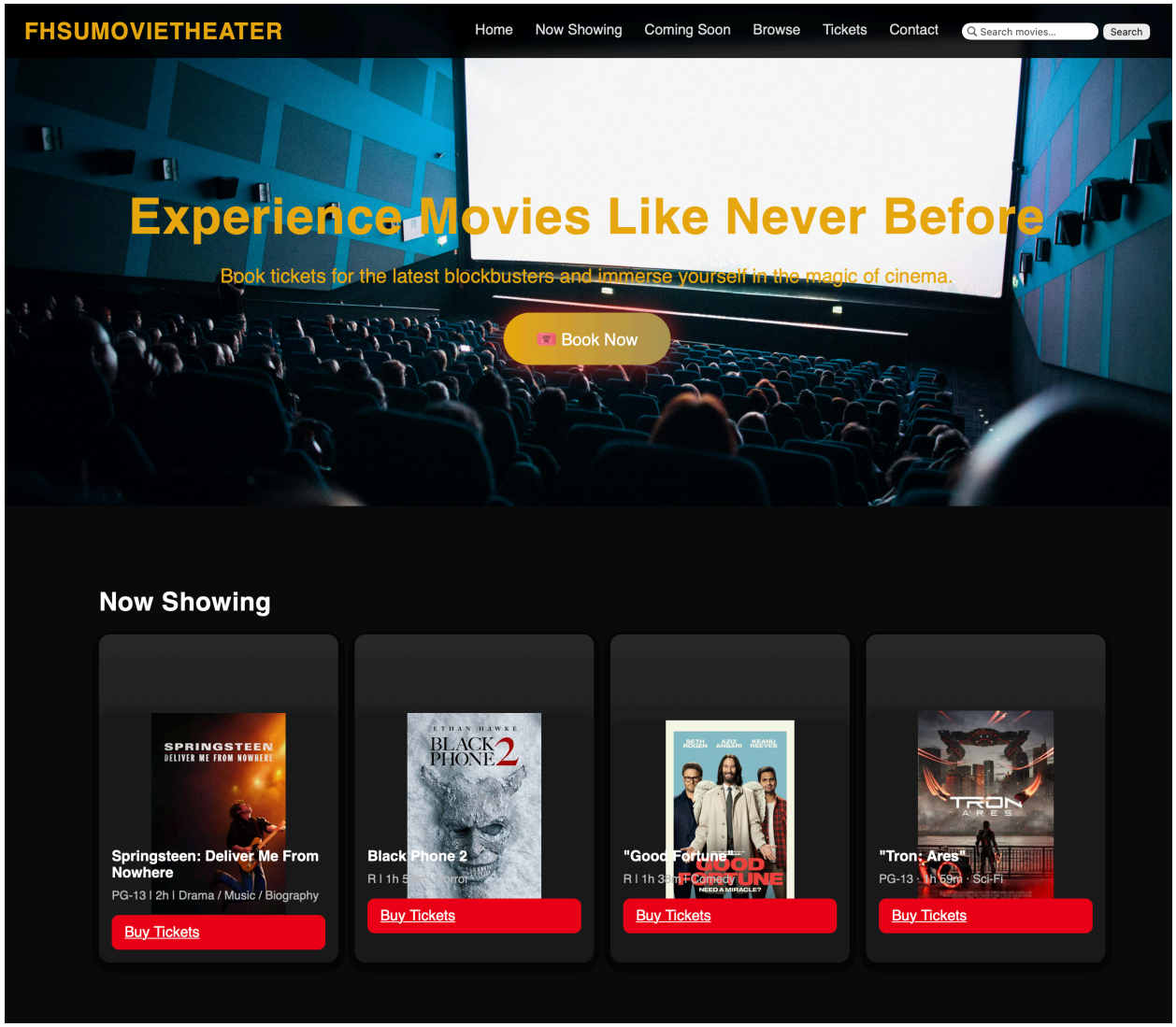
Tests that an employee can delete a showing, and a normal user can’t delete a showing.

Our coverage for testing will be at a high level. For our unit testing, we will primarily be using white box testing to develop our classes and switch to black box testing, once the class passes white box testing, to verify that the class functions as intended. Our testing process will extensively cover the specifications required for our software. This will be accomplished because of our selected test cases that will evaluate many of our classes numerous times throughout the process.

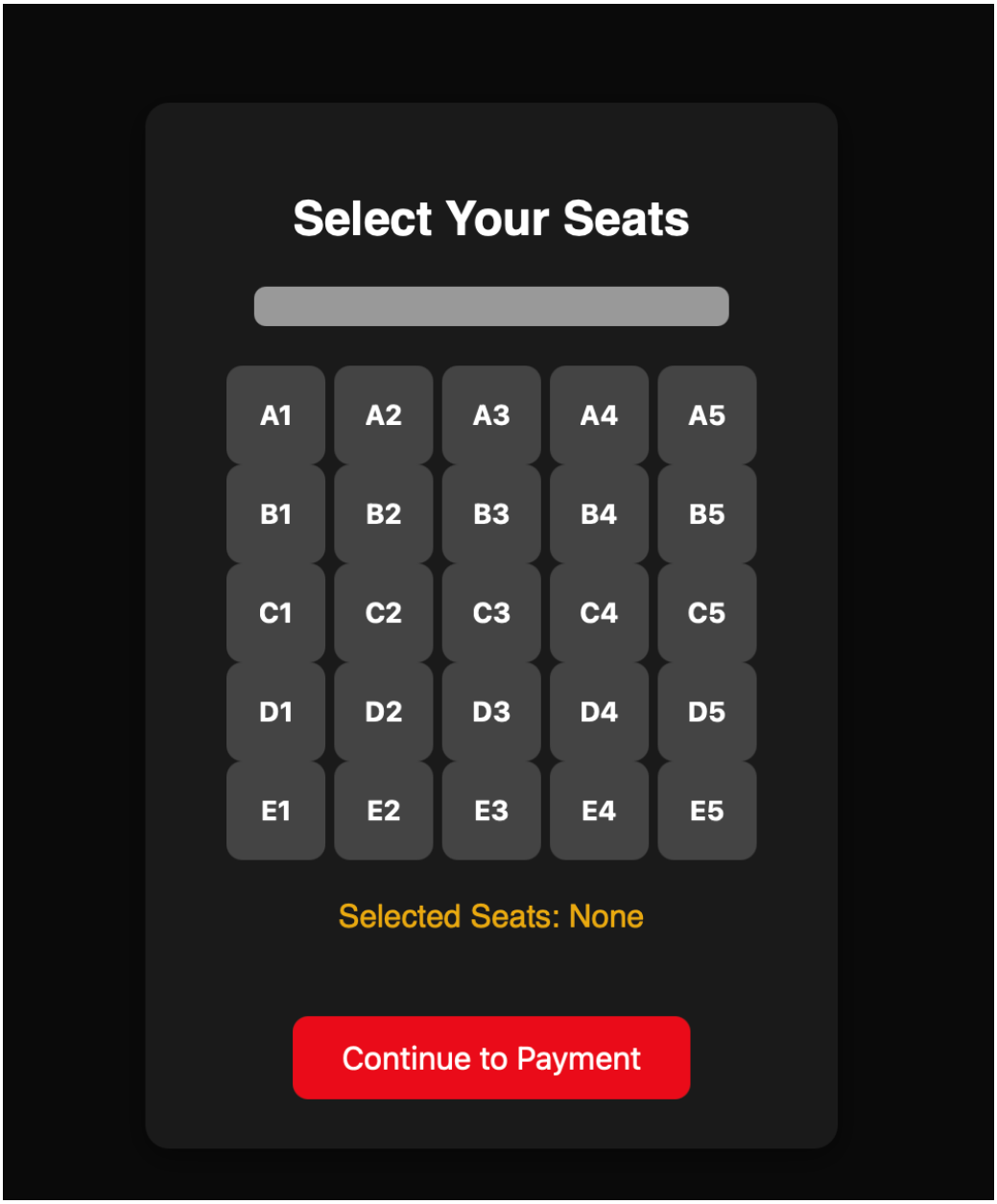
For our integration strategy we will use vertical integration testing as it aligns with our agile development approach. The test cases (user stories) listed above will be how we will break down our development approach. These user stories have been selected because they are typical usage scenarios, and their inner workings overlap to cover the testing for our entire system.

User Interface Design and Implementation

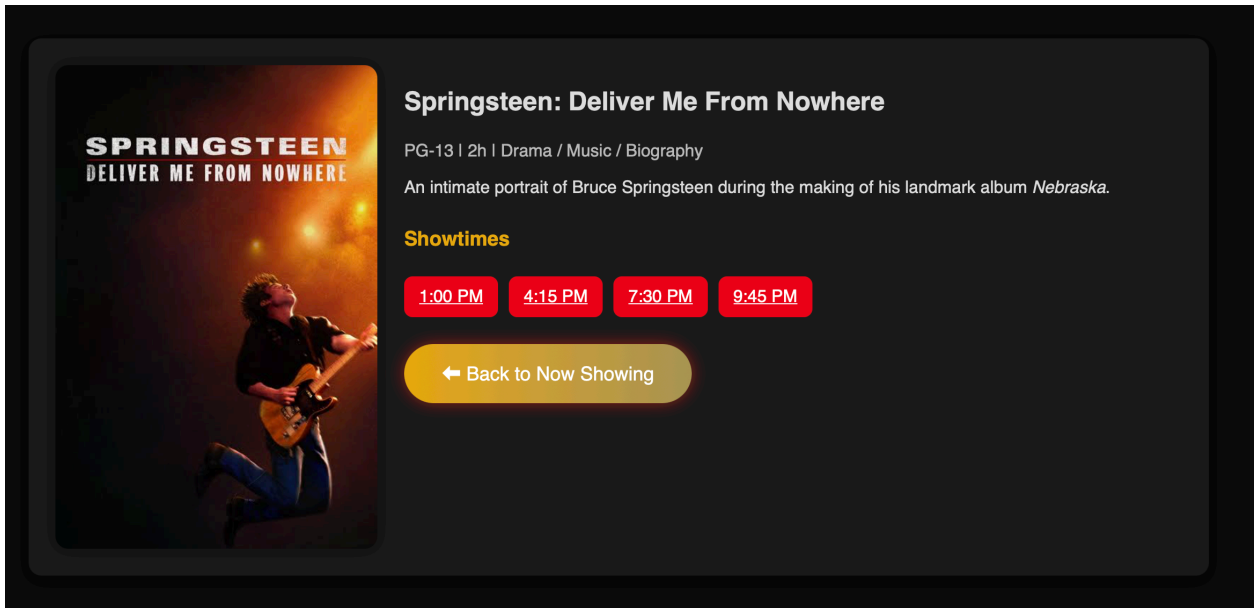
Customer Homepage:
The customer can see what movie is showing on the front page. Homepage contains the option for the customers to search for the app for movies and give options for customers to sign in to their account and view their purchased tickets and purchase history. So far for Demo 1, we have the current movies in theaters and the homepage created.



Seat Map Selection Interface:
The user can select available seats for a chosen movie through the seat map selection interface. The interface displays the theater layout, and shows which seats are available, reserved, and taken using clear legends. The design prevents double booking using realtime seat availability updates and provides seats that are accessibility friendly.

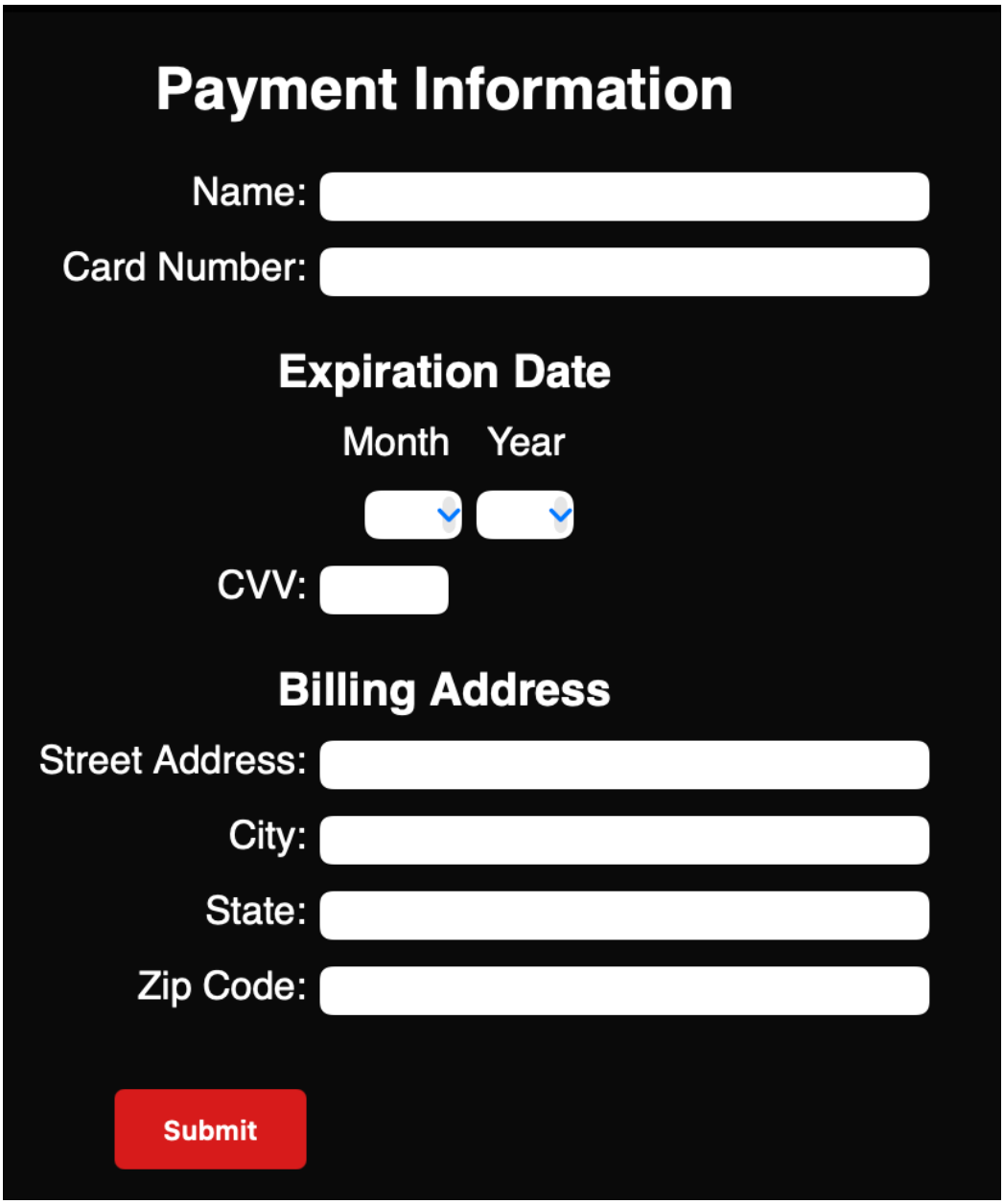


Movie Details Interface:
The Movie Details Interface shows information about the selected movies within the app. The interface shows available date, showtimes, auditorium, movie length, poster, rating, Genre. Users can conveniently select preferred showtime to purchase their tickets, and it will bring them to the seat selection interface for their seats.



Checkout Interface:

Checkout Interface is the final step in the ticket purchasing process, User can review their tickets to make sure everything is correct before purchasing it. This page shows the summary of the user's booking details, showing movie title, seat, showtime, auditorium id, and total cost. Users can choose their preferred method to pay and apply a discount code if available.



History of Work, Current Status, and Future Work

This section summarizes how the work on the Fort Hays State Movie Theaters system evolved over the semester, compares it to the original plans from Reports #1 and #2, and outlines the current status and potential directions for future work.

History of Work

At the beginning of the project (Report #1), our team focused on understanding the problem domain and capturing what the customer, staff, and theater manager needed from a modern ticketing system. We produced the customer statement of requirements,

glossary, and initial system requirements, including both functional and non-functional requirements such as performance, availability, and security

In Report #2, the emphasis shifted from “what” the system should do to “how” it would behave. We identified stakeholders and actors, defined and refined use cases, built the use case diagram, and created the traceability matrix connecting use cases to requirements. We also produced system sequence diagrams, interaction diagrams, and began shaping the overall architecture and data structures that would support ticket purchasing, cancellations, admin management, and reporting.

For Report #3, our work focused on consolidating and refining the earlier deliverables and aligning them with the implementation and final demo. We expanded the narratives for all actors, updated diagrams for visual consistency, and produced more detailed descriptions of algorithms, data structures, concurrency concerns, and effort estimation using Use Case Points. As part of this iteration, we adjusted the original milestone plan: database setup and the theater manager/employee functionality were moved from Demo 1 to Demo 2 to better match the actual pace of the team and the complexity of implementation.

Throughout the project we used GitHub for version control and YouTrack for task tracking and milestones. These tools helped us coordinate contributions across team members, manage changes to requirements and diagrams, and keep the implementation aligned with the evolving design and schedule.

Key Accomplishments

Our main accomplishments in this project can be summarized as:

- Defined a clear problem statement and stakeholder needs for customers, staff/admins, and the theater manager.
- Specified functional and non-functional requirements, including UI and accessibility requirements.
- Modeled the system using use cases, system sequence diagrams, interaction diagrams, and a traceability matrix.
- Designed a layered client–server architecture with identified subsystems, connectors, and hardware requirements.
- Described core algorithms for seat selection and holding, order processing, cancellation/refund, and reporting, along with supporting data structures and concurrency considerations.
- Estimated project effort using Use Case Points, technical factors, and environmental factors.

- Implemented a working web prototype that demonstrates core customer and admin flows consistent with the use cases defined in earlier reports.

Current Status

For the final demo the system exists as a functional prototype of an online movie theater platform. Customers can browse movies and showtimes, view seating information, and walk through the ticket-purchase flow at the user interface level. Theater staff can conceptually manage movies, showtimes, and pricing through the admin functionality described in our architecture and use cases. Some parts of the backend behavior (such as full reporting, payment integration, and payroll management) are represented at the design level and partially mocked in the prototype rather than fully deployed in a production environment.

Future Work

There are several directions in which this project could be extended:

- **Full backend and database integration**: Replace mock data with a persistent database implementation for movies, showtimes, tickets, and user accounts. This could also include fully implementing the reporting subsystem.
- **Real payment and email services**: Integrate with a real payment gateway and email/notification service to handle secure transactions and confirmations end-to-end.
- **Staff scheduling and payroll**: Implement the manager-focused features for employee time tracking, scheduling, and payroll reporting that were identified in the requirements but not fully implemented.
- **Enhanced accessibility and responsiveness**: Further improve accessibility (WCAG compliance) and optimize the UI for a wider range of devices and assistive technologies.
- **Scalability and multi-theater support**: Extend the system to support multiple theaters/locations, higher user loads, and more advanced analytics dashboards for owners and managers.
- **Security hardening and auditing:** Add stronger authentication, authorization, logging, and audit trails to better support real-world deployment and security requirements.

By completing the current prototype and documenting the system in detail, we have created a strong foundation that can be used to continue development into a fully deployed, production-ready movie theater ticketing system.

References

https://eceweb1.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf - textbook