CheapTix Project

Christina Daluge

Table of Contents:

Customer Problem Statements and System Requirements

Customer Problem Statement

As a concertgoer, I've faced numerous frustrations when trying to purchase tickets for my favorite events. The current systems are often slow, confusing, and lack transparency. When I visit a concert ticket website, I want a seamless experience: easy navigation, clear information about seating options, and real-time availability updates. It's frustrating to click on a ticket only to find it's no longer available or to encounter hidden fees during checkout. I expect a system that provides a user-friendly interface, allows me to filter events by date, genre, and location, and offers secure payment options. Ultimately, I want to feel confident that I can find and secure the tickets I want without unnecessary hurdles.

Glossary of Terms

- **Ticket Availability:** The status indicating whether tickets for a specific event are still available for purchase.
- **Event Filtering:** The ability to sort and display events based on certain criteria such as date, genre, or location.
- **Checkout Process:** The sequence of steps a customer takes to complete their ticket purchase, including selecting tickets, entering payment information, and finalizing the purchase.
- **User Account:** A personal profile created by customers that stores their purchase history, preferences, and payment information.
- **Dynamic Pricing:** A pricing strategy where ticket prices fluctuate based on demand, time remaining until the event, and other market conditions.

System Requirements

Functional Requirements

No.	Priority Weight	Description
REQ-1	High	The system should display real-time ticket availability for each event.

No.	Priority Weight	Description
REQ-2	High	Users should be able to filter events based on date, genre, and location.
REQ-3	High	The checkout process must allow for secure payment options including credit/debit cards and PayPal.
REQ-4	Medium	The system should provide users with an option to create a user account to store preferences and history.
REQ-5	High	Users should receive a confirmation email after successful ticket purchase, including order details.
REQ-6	Medium	The system should implement a dynamic pricing strategy based on demand and availability.

Nonfunctional Requirements

No.	Priority Weight	Description
NFR-1	High	The system should maintain 99.9% uptime to ensure availability during peak ticket sales.
NFR-2	Medium	The interface should load within 3 seconds for optimal user experience.
NFR-3	High	The platform must comply with security standards (e.g., PCI DSS) for handling payment information.
NFR-4	Medium	The system should be scalable to handle high traffic during major events.

User Interface Requirements

No.	Priority Weight	Description
UI-1	High	The home page should display a list of upcoming events with images and basic details.
UI-2	High	The event details page should provide comprehensive information about the event, including seating chart and ticket prices.
UI-3	Medium	The filtering options should be clearly visible and easily accessible on the events page.

No.	Priority Weight	Description
UI-4	Medium	The checkout page should have a clean layout with clear calls to action and minimal distractions.
UI-5	High	The confirmation page should summarize the order details and provide an easy way to print tickets or add them to a mobile wallet.
UI-6	Medium	The user account section should allow for easy management of preferences, payment methods, and purchase history.

Sketches for User Interface Requirements

1. **Home Page Layout**: A grid of event cards showcasing upcoming concerts, with images, dates, and a “Buy Tickets” button.
2. **Event Details Page**: A detailed view with a seating chart, ticket pricing, and a “Select Tickets” option.
3. **Filtering Options**: Sidebar with checkboxes for genres, date selectors, and location search.
4. **Checkout Page**: A streamlined layout with order summary, payment options, and “Confirm Purchase” button.
5. **Confirmation Page**: A clear summary of the purchase, including a “Download Ticket” option.
6. **User Account Page**: Sections for personal details, payment methods, and order history, with an “Edit” button for easy updates.

By following these guidelines, we can create a concert ticket website that meets customer expectations and provides a smooth, enjoyable ticket purchasing experience.

Functional Requirement Specifications for Concert Ticket Website

Stakeholders:

1. **Customers:** People looking to buy concert tickets
2. **Event Organizers:** Groups that create events and need a way to sell tickets
3. **Website Administrators:** Staff who manage the website and handle transactions
4. **Payment Processors:** Services that process payments for ticket purchases
5. **Marketing Team:** People who promote events and engage with customers

Actors and Goals:

Primary Actors

- **Customer:** A person who wants to look at, choose, and buy concert tickets
- **Event Organizer:** A person who can create and manage events and track ticket sales

Secondary Actors

- **Admin:** A person who oversees the whole system, including events and users
- **Payment Gateway:** A service that safely handles customer payments
- **Notification System:** A service that sends updates and confirmations to customers about their tickets

Use Cases:

Admin Tasks (total: 18)

1. **Add Event:** Admin can create a new concert and set details (4)
2. **Update Event:** Admin can change existing event details (3)
3. **Delete Event:** Admin can remove an event from the site (3)
4. **View Sales Reports:** Admin can see reports on how many tickets were sold (3)
5. **Manage Users:** Admin can check and manage user accounts (2)
6. **Login/Logout:** Admin can log into and out of their account (3)

Customer Tasks (total: 12)

1. **Browse Events:** Customers can see a list of upcoming concerts (2)
2. **Select Tickets:** Customers can pick how many and what kind of tickets to buy (2)
3. **Add to Cart:** Customers can add tickets to their shopping cart (2).
4. **Checkout:** Customers can pay for their tickets (3)
5. **Receive Confirmation:** Customers get a confirmation email for their purchase (2)
6. **View Order History:** Customers can see their past ticket purchases (1)

Event Organizer Tasks (total: 12)

1. **Create Event:** Organizers can set up a new event (4)
2. **Manage Ticket Inventory:** Organizers can change the number of tickets available (3)
3. **View Sales Data:** Organizers can check how many tickets have been sold (3)
4. **Edit Event Details:** Organizers can update event information (2)

Payment Gateway Tasks (total: 6)

1. **Process Payment:** Handles secure payment transactions (3)
2. **Send Payment Confirmation:** Sends a confirmation back after payment (3)

Notification System Tasks (total: 4)

1. **Send Ticket Confirmation:** Sends confirmation emails to customers after they buy tickets (2)
2. **Send Event Reminders:** Sends reminders about upcoming events (2)

Estimation of Implementation Time:

- **Admin Tasks:** 18 points
- **Customer Tasks:** 12 points
- **Event Organizer Tasks:** 12 points
- **Payment Gateway Tasks:** 6 points
- **Notification System Tasks:** 4 points

Total Estimated Workload: 52 points

Estimated Timeline:

1. **Requirements Gathering (2-4 weeks):**
    - Talking to stakeholders to define what's needed. This could take longer depending on how many people are involved in the project
2. **Design Phase (3-5 weeks):**
    - Creating wireframes and designs for how the site will look and work
3. **Development Phase (8-12 weeks):**
    - **Frontend Development:** Building the user interface (4-6 weeks)
    - **Backend Development:** Setting up server logic and databases (4-6 weeks)
4. **Integration and Testing (4-6 weeks):**
    - Connecting payment systems and making sure everything works properly
5. **Deployment (1-2 weeks):**
    - Launching the site and doing final checks
6. **Post-Launch Support (2-4 weeks):**
    - Fixing any issues and gathering feedback for improvements

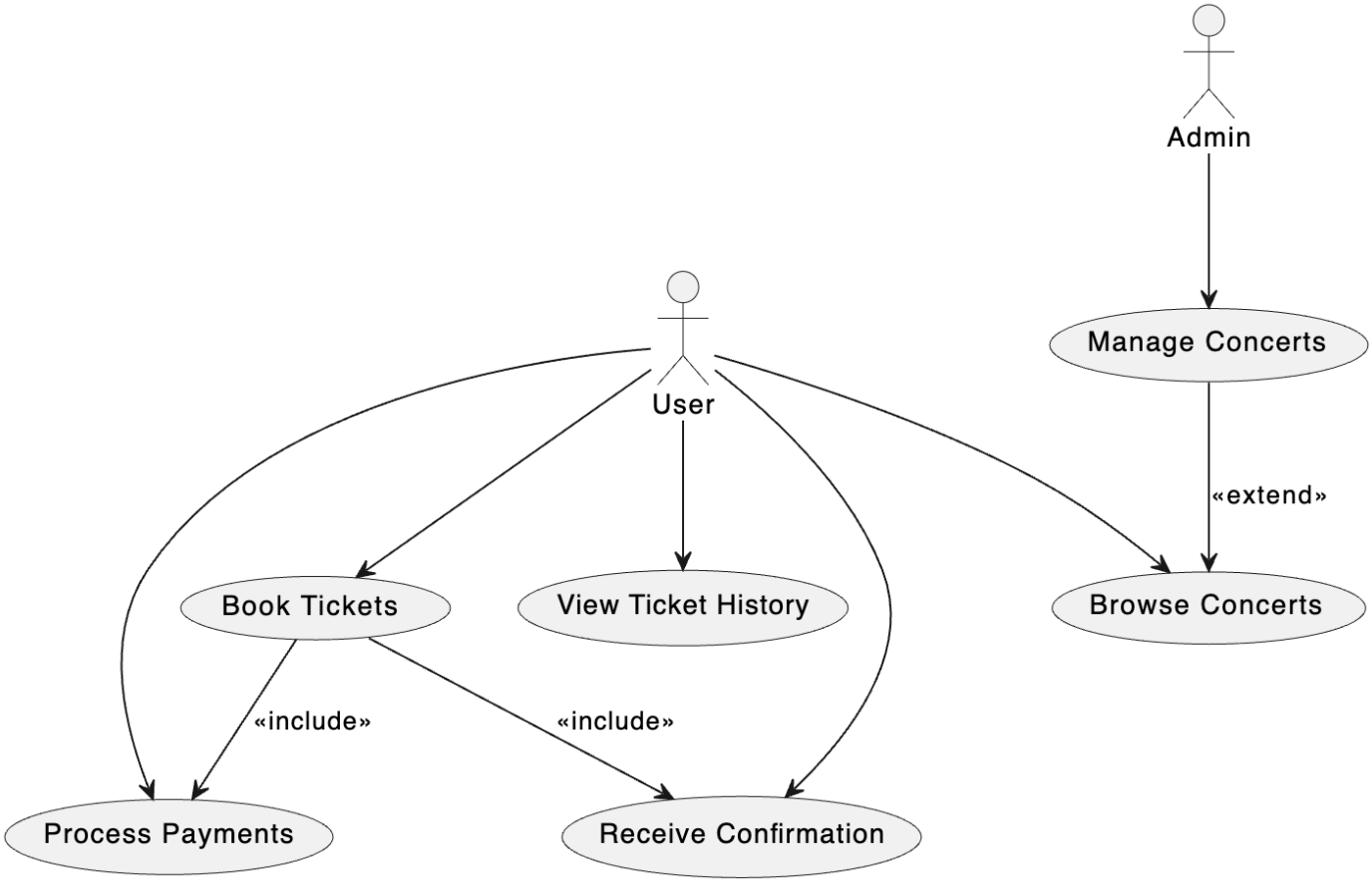
Total Estimated Duration: 20-33 weeks (5-8 months)

Factors Influencing Timeline

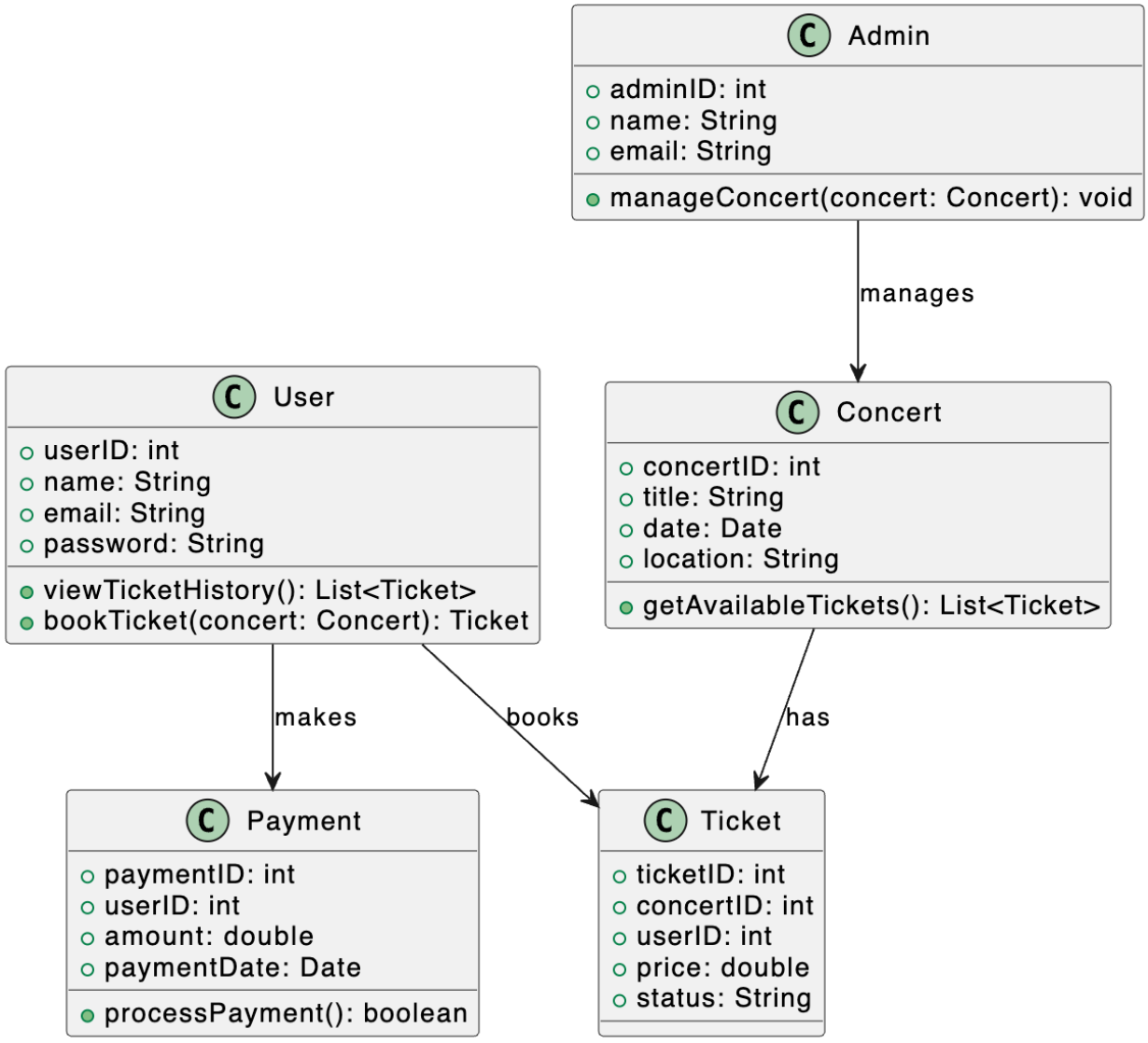
1. **Team Size and Skill:** A bigger, skilled team can finish faster; a smaller team might take longer
2. **Scope Creep:** Adding new features later can delay the project. Clear requirements help avoid this

3. **Technology Choices:** Some tools are faster to work with than others
4. **Compliance:** Meeting payment security rules can take extra time
5. **Testing Needs:** More thorough testing means a longer timeline, especially for security
6. **Feedback Loops:** Getting constant feedback can improve the outcome but might extend the timeline
7. **External Dependencies:** Relying on outside services can introduce delays if they are complex

Use Case Diagram:



Class Diagram:


Admin
adminID: int
name: String
email: String
manageConcert(concert: Concert): void
manages
User
userID: int
name: String
email: String
password: String
viewTicketHistory(): List<Ticket>
bookTicket(concert: Concert): Ticket
Concert
concertID: int
title: String
date: Date
location: String
getAvailableTickets(): List<Ticket>
makes
books
has
Payment
paymentID: int
userID: int
amount: double
paymentDate: Date
processPayment(): boolean
Ticket
ticketID: int
concertID: int
userID: int
price: double
status: String

Sequence Diagrams

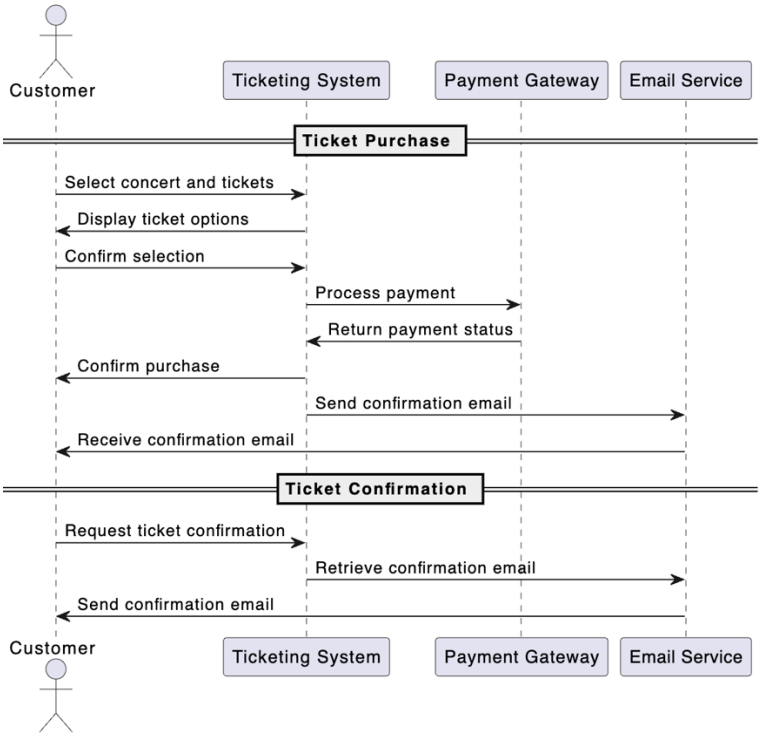
System Sequence Diagram and Activity Diagram Christina Daluge

Sequence Diagram for a Customer buying a concert ticket Actor: Consumer looking for a ticket

Object: Paying for a Ticket

Steps of a customer buying a ticket:

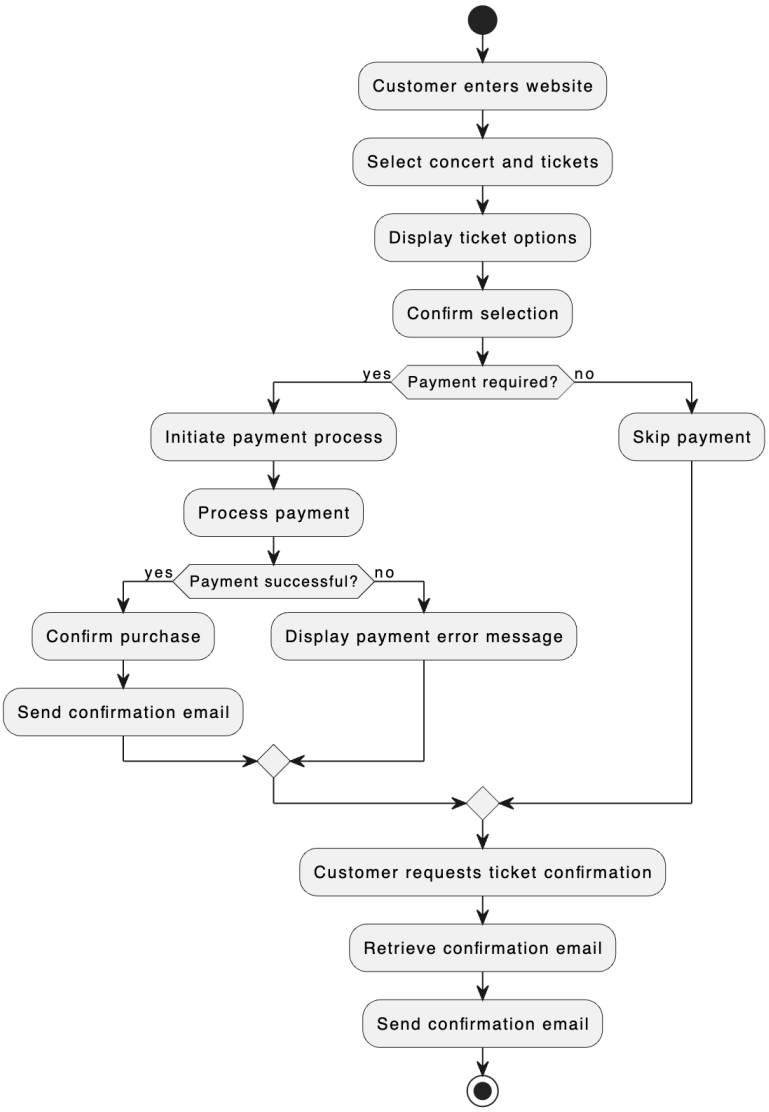
1. Customer uses web browser to access website
2. Customer searches for desired event using the search bar
3. Customer browses through list of events
4. Select desired event: Date/Time, Location, Venue, Seating
5. Select the desired ticket type: General Admission, VIP, etc..
6. Choose the number of tickets needed.
7. Check the seating chart (if applicable) to select specific seats.
8. Click the "Add to Cart " or "Buy Now" button.
9. Proceed to checkout (Check out or Proceed to Payment)
10. Customer enters personal information
11. Customer enters payment information
12. Customer reviews and confirms the tickets are the right ones
13. Customer receives email confirmation of purchase



Activity Diagram:
Activity Diagram for Customer buying a concert ticket

Initial State: The customer is on the ticket buying website homepage looking to buy a ticket

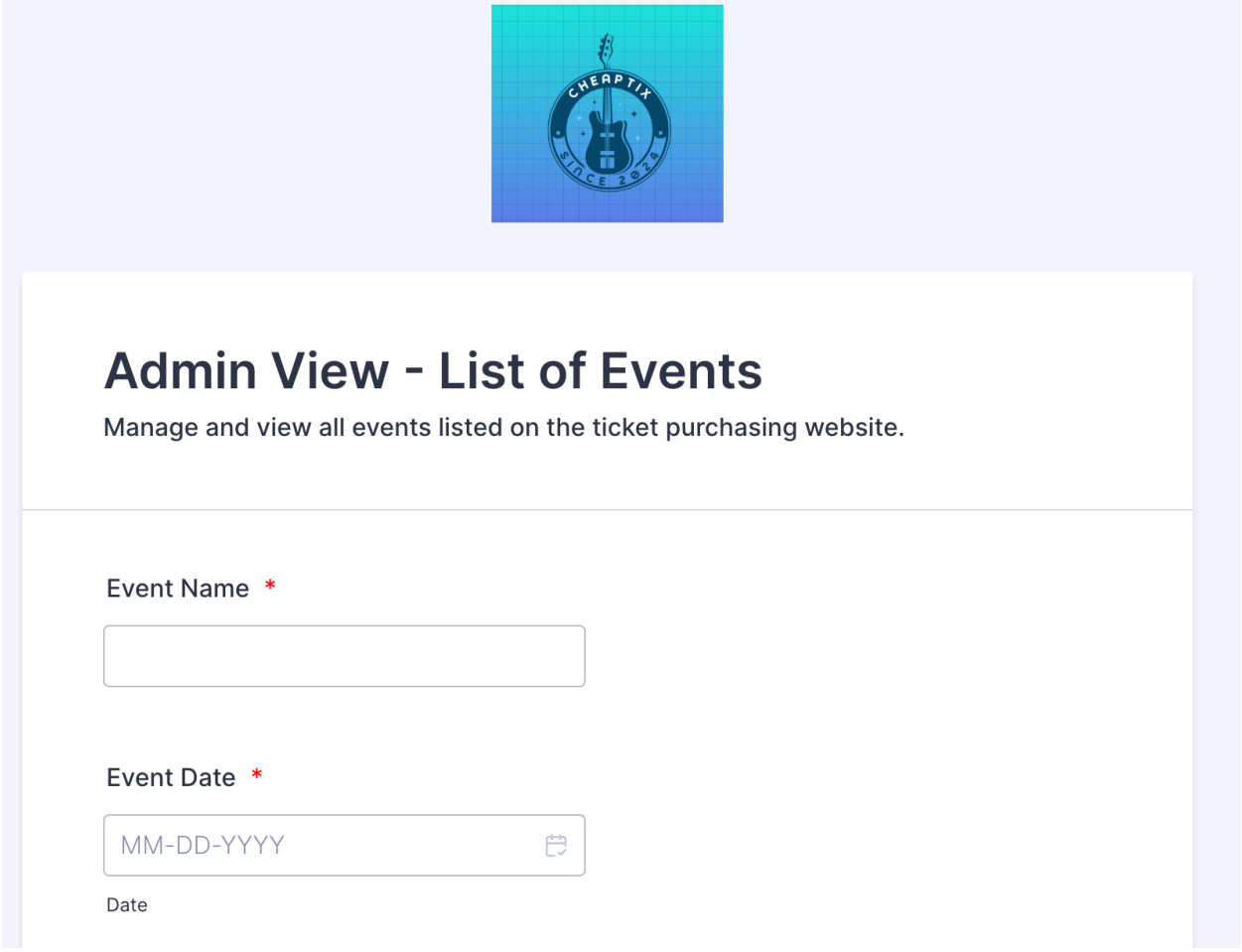
Final Stage: The customer gets a confirmation email that tickets were bought and payment went through



Concert Ticket User Interface Specifications

Preliminary Design:
The User Interface for our Concert Ticket Manager allows users to easily view a list of upcoming concerts, filter events by various criteria, and navigate seamlessly through options. The administrators can add new events, modify existing concert details, or remove events as needed. The top navigation bar also provides quick access to related features, such as adding new music genres or categories.

Viewing Concerts Page:
Admins can see the list of concerts, click buttons to navigate to options for adding, modifying, or deleting concerts.



Adding a Concert

Entering Information:
To add a concert, users enter details into the designated light blue text fields. Each field includes prompts, such as "Enter Concert Name," guiding the user on what to input. Once all information is entered, users can click the blue “Add Concert” button. To cancel, they can click the white “Cancel” button.

Displaying Results:
After adding a concert, a success or error message appears, confirming whether the concert was added successfully. The new concert will also be listed on the Concerts Page.

Navigation Path:
Users follow this path to add a concert:
Login > Home Page > Concerts List > Add Concert

Modifying a Concert

Entering Information:
To edit a concert, users select the concert they wish to modify. The fields are pre-filled with existing details, allowing easy updates. Users can edit the information and click the blue “Submit Edit” button. If they wish to cancel, they can click the white “Cancel” button.

Displaying Results:
Success or error messages confirm if the concert was edited successfully, with changes reflected on the Concerts Page.
Navigation Path:
Users navigate as follows:
Login > Home Page > Concerts List > Edit Concert

Deleting a Concert

Entering Information:
To delete a concert, users click the red “Delete” button next to the desired concert. A confirmation screen appears, prompting users to confirm their choice. They can click the red “Delete Concert” button to finalize the deletion or cancel if they change their mind.

Displaying Results:
A success or error message is shown based on the deletion outcome, and the concert will be removed from the list.

Navigation Path:
Users follow this path to delete a concert:
Login > Home Page > Concerts List > Delete Concert

Adding a Performance Category

Entering Information:
To add a new performance category, users input the category name into a light blue text field with prompts. After entering the information, they can click the blue “Add Category” button or cancel the operation.

Displaying Results:
Users receive confirmation messages indicating whether the category was added successfully, and it appears in the Categories List.

Navigation Path:
Users navigate as follows:
Login > Home Page > Categories > Add Category

User Interaction Example

Adding a New Concert
For instance, if a manager wants to add a concert named "Summer Music Fest," they will log in, access the Concerts List, click “Add Concert,” fill out the required fields (name, date, venue, etc.), and submit.

Total Clicks and Keystrokes:

1. Log in (click)
2. Enter username and password (2 keystrokes each)
3. Access Concerts List (click)
4. Click “Add Concert” (click)
5. Fill in details (total of 5 fields, each click + 60 keystrokes for data entry)
6. Click “Add Concert” (click)

Total: 76 clicks and keystrokes or 12 clicks if already logged in.

Project Plan

Problem Statement:

This concert ticket website is designed to address the common challenge of fans accessing event information in a streamlined, user-friendly, and efficient way. With this platform, users can effortlessly explore concert details from anywhere in the world, whether at home or on the go with their mobile devices. This system ensures fans can easily find information about upcoming shows, including event locations, schedules, and ticket availability. By enhancing discoverability online, we help artists and venues reach a wider audience, fostering growth and attracting more attendees. This website not only keeps the customer updated on the latest concerts but also helps our partners stay competitive in the live entertainment industry. Plus, the technology behind this system can be adapted for any event or business, making it a versatile solution for all your ticketing needs.

Objective of the System:

This concert ticket website is designed with key objectives to streamline the management and accessibility of event information. The aim is to keep all concert details, such as venue locations, schedules, and ticket availability up to date and organized. This system empowers venue owners and promoters to easily add new events and remove those that have ended, ensuring that users have access to the most current information. By increasing transparency, we make it simple for fans to find concert times, locations, and updates on upcoming shows. Additionally, users can favorite their favorite artists and venues, making it easier to discover nearby concerts. Our platform is dedicated to enhancing the live music experience for fans and event organizers alike.

System Requirements:

The requirements for this system are that the system should be able to let customers and venue owners to do these activities:

What customers can do with the system:

- Favorite Artists, Venues, and Events
- Track Prices
- View Upcoming events
- Search Artists, Dates, Locations, and Genre
- Password recovery
- Account Creation/ Secure Logins
- Options for selecting ticket quantity and seating preferences
- Users can leave reviews and rate events after attending

What the Venue Owners/ Event Planner can do with the System:

- Filter Options to help the user find relevant events
- Secure online payment processing (credit/debit cards, digital wallets)
- Admins can create, update, and delete events
- Ability to set event details such as date, time, location, and ticket pricing
- The website should be fully functional on mobile devices.
- Option for a mobile app for enhanced user experience.

- Analytics for tracking ticket sales, user engagement, and event popularity.
- Management tools for monitoring and moderating user content.

Projected Cost:
- Low-End Estimate: $18,000 - $30,000
- High-End Estimate: $50,000 - $80,000

Typical Customers:
This system will have different access and features depending on who is trying to access it:
- Customers (People buying the tickets)
- Venue Managers/ Admins

Project Planning and Development Approach:
1. Software:
    a. Front-end: Can use either HTML, CSS, or JavaScript
    b. Back-End: Node.js, Python (Django or Flask), Ruby on Rails, or PHP (Laravel)
    c. Database: MySQL, PostgreSQL, or MongoDB for data storage
    d. Payment Processing Software: Integration with PayPal, Square, Stripe, or Apple/Google Pay
    e. Security: Firewalls and anti-malware software and security for payments
2. Hardware:
    a. Desktop/PC/Mac
    b. Mobile Devices: Smartphone, Tablet
    c. Servers
3. Network:
    a. Internet Connection
    b. Domain Name
    c. Network Monitoring tools
    d. Firewall and Security

Development Plan- Weekly:
Week 1: Project Planning/ Requirements Gathering
Goals:
- Define Project scope, objectives, and features
- Gather requirements from stakeholders

Tasks:
- Create project charter
- Outline user stories and use cases
- Set up project management tools

Week 2: Wireframing and Prototyping

Goals:

- Design the layout and user interface of the website.

Tasks:

- Create wireframes for key pages (home, event listing, event details, checkout).
- Develop rough prototypes
- Gather feedback on designs from stakeholders

Week 3: Technology Stack Finalization
Goals:

- Decide on the technology stack for the frontend, backend, and database.

Tasks:

- Research and select frameworks and libraries
- Finalize database choices
- Set up a development environment

Week 4: Setting Up the Development Environment
Goals:

- Prepare for coding by setting up tools and repositories

Tasks:

- Set up version control
- Create project repository
- Install necessary development tools and libraries

Week 5: Database Design
Goals:

- Design the database schema

Tasks:

- Create ER diagrams for the database
- Define tables for users, events, tickets, and transactions
- Set up the database and establish connections in the backend

Week 6: Backend Development – User Management
Goals:

- Implement user registration and authentication

Tasks:

- Develop APIs for user registration, login, and password recovery
- Implement JWT or session-based authentication
- Test user management functionalities

Week 7: Backend Development – Event Management
Goals:

- Build the event management features

Tasks:

- Develop APIs for creating, updating, and deleting events
- Implement data validation and error handling
- Test event management functionalities

Week 8: Frontend Development – User Interface
Goals:

- Start building the frontend based on the designs

Tasks:

- Set up the frontend framework (e.g., React)
- Develop components for the homepage, event listings, and event details
- Ensure responsiveness for mobile devices

Week 9: Frontend Development – Ticket Purchasing
Goals:

- Implement ticket purchasing functionalities

Tasks:

- Create components for the ticket selection and checkout process
- Integrate payment gateway
- Implement form validation for the checkout process

Week 10: User Features – Favorites and Notifications
Goals:

- Add user features for favorites and notifications

Tasks:

- Develop APIs for favoriting events
- Implement user notifications for upcoming events
- Integrate notification system (email or in-app)

Week 11: Testing – Unit and Integration Testing
Goals:

- Ensure the application is functional and bug-free

Tasks:

- Write and run unit tests for backend and frontend components
- Perform integration testing for APIs
- Begin user acceptance testing (UAT) with stakeholders

Week 12: Finalizing Features and Bug Fixes
Goals:

- Address bugs and finalize features

Tasks:

- Review feedback from testing phases
- Fix identified bugs and improve UI/UX based on feedback
- Ensure all features are working as intended

Week 13: Performance Optimization
Goals:

- Optimize application for speed and efficiency

Tasks:

- Implement caching strategies (e.g., Redis, CDN)
- Optimize images and assets for faster load times
- Conduct load testing to evaluate performance under stress

Week 14: Deployment Preparation
Goals:

- Prepare the application for deployment

Tasks:

- Set up production environment (server, domain)
- Configure CI/CD pipelines for automated deployment
- Prepare documentation for users and administrators

Week 15: Launch and Post-Launch Activities
Goals:

- Launch the concert ticket website and monitor its performance

Tasks:

- Officially launch the website
- Monitor user interactions and system performance
- Collect user feedback and plan for future updates or enhancements