German International University
Faculty of Informatics and Computer Science
Dr. John Zaki
Donia Ali
Nadeen Ramadan
Rahma Ameen

Software Engineering, Spring 2025
Online Event Ticketing System

Project - Task 3

1 Overview

In Task 3, you will develop the frontend of your Online Event Ticketing System using React. This milestone focuses on creating a user-friendly, interactive web interface that communicates with the backend server (built in Task 2). You are expected to implement all major user flows, including authentication, event browsing, event management (for organizers), ticket booking, and profile management, according to the API endpoints and access controls defined in the backend specification

You *can* take Tickets Marche as your reference

2 Requirements

- Use React (with functional components and hooks) as your primary front-end framework.
- Ensure your application is responsive and provides a seamless experience across devices.
- Integrate with your backend API using HTTP requests via **axios**.
- Implement robust error handling and display user-friendly feedback for all interactions.

3 Task 3 description

Note: refere back to task 2 for privacy of APIs (i.e. public/private/admin access,etc.)

a) Authentication & Authorization

- Implement user registration and login forms.
- Use React context to store/access current user.
- Protect routes based on user roles (Standard User, Event Organizer, Admin); redirect unauthorized users as appropriate.
- Implement forget password UI functionality.
- Implement Logout UI functionality, *Hence*; you need to develop a public backend endpoint '/logOut' in auth routes, *Goal: remove cookies.*

b) User Profile Management

- Allow all users to view and update their profiles.

- Display relevant user information fetched from the backend.

c) Event Browsing & Details

- Display a list of approved events for all users.
- Allow users to view detailed information for each event.
- Implement search and filtering options for events.

d) Event Management (Event Organizers)

- Allow organizers to create, edit (number of tickets, date, location), and delete their own events.
- Display organizers' events and provide access to analytics (e.g., a graph showing percentage of tickets booked per event).
- Display event status (approved, pending, declined).

e) Booking Functionality (Standard Users)

Note: The booking status is only Confirmed or Canceled.

- Allow authenticated users to book tickets for events.
- Display real-time ticket availability on the event page (e.g., 'Sold Out', 'Only 5 tickets left', or show a ticket count input based on availability).
- User can book more than one ticket, as long as it doesn't exceed the available number of tickets.
- When user presses book/reserve, the corresponding backend endpoint is called. *Hint*: only one api call
- Display booking history and allow users to cancel bookings.
- Show total price calculation based on ticket quantity and event price once booking is successfully completed.

f) Admin Dashboard

- Allow admins to view all events (approved, pending, declined).
- Allow admins to approve/reject the pending events.
- Provide admin access to user management features (view, update role, delete users).

g) Error Handling & User Feedback

- Display informative messages for errors such as failed authentication, insufficient tickets, or unauthorized actions.
- Show loading indicators and success confirmations where appropriate.

4 Components

a) Shared Components (Used by All Users)

- Navbar/Sidebar
  - Displays links based on login status and user role (User, Organizer, Admin).
  - Includes Logout button when authenticated.
- Footer
  - Static content like copyright.
  - Contact info or links.
- PrivateRoute / ProtectedRoute

  - Wrapper for routes that require authentication.
  - Role-based redirection (e.g., if a User tries to access Admin route).
- Loader / Spinner
  - Reusable loading indicator for data-fetching components.
- Toast Notifications
  - For showing success/error messages (e.g., on login, form submission).
  - Use libraries like react-toastify.

b) Authentication Components

- LoginForm
  - Email, password fields.
  - Shows errors (e.g., incorrect credentials).
  - Calls login from authcontext.
- RegisterForm
  - Name, email, password, confirm password, role selector (User or Organizer).
  - After successful registration, user is redirected to login page.
- ForgotPassword
  - Email input to request password reset/update.

c) User Profile Components (All Roles)

- ProfilePage
  - Shows current user's information.
- UpdateProfileForm
  - Editable fields with client-side validation.
  - Calls backend to update user data.

d) AdminUsersPage

- AdminUsersPage.jsx The main page component
  - Calls the API to fetch all users on mount
  - Displays them in a table
- UserRow.jsx (Reusable subcomponent)
  - Represents a single user in the table
  - Displays: name, email, role
  - Includes action buttons: Update Role, Delete
- UpdateUserRoleModal.jsx (Optional)
  - Modal to select new role for user (User, Organizer, Admin)
  - Calls PUT /users/:id
- ConfirmationDialog.jsx (Optional)
  - For delete confirmation
  - Calls DELETE /users/:id

e) Event Components

- EventList
  - Shows list of approved events for public and users.
  - Includes event cards with basic info: name, date, location, ticket price.
- EventCard
  - Compact, clickable card for each event.

  - Opens single event page.
- EventDetails
  - Full details of a single event.
  - view number of available tickets.
  - BookTicketForm is shown if user is logged in.
- EventForm (Organizer only)
  - Create/edit event form with fields: title, date, location, ticket count, price.
  - Reusable for both creation and editing.
- MyEventsPage (Organizer only)
  - List of events created by the current organizer.
  - Edit/delete buttons for each.
- EventAnalytics (Organizer only)
  - Graph (e.g., using recharts or chart.js) showing ticket booking
- AdminEventsPage (Admin only)
  - Table of all events with filters: approved, pending, declined.
  - Buttons to approve or decline for each event.

f) Booking Components

- BookTicketForm
  - Select quantity, show total price.
  - User can choose more than one ticket if available.
  - 'Book' button that checks availability and calls backend.
- UserBookingsPage
  - Shows bookings made by the user.
  - Each booking includes event name, quantity, price, and cancel button.
- BookingDetails
  - Full details of one booking (optional).
  - Could be modal or separate page.

g) Utility / Infrastructure Components

- Optional: API Service (api.js)
  - Axios instance with base URL and token header.
  - Methods like login, register, getEvents, bookTicket, etc.
- AuthContext
  - Stores current user data (including role).
  - Provides user,login, logout, loading, etc.
- RoleBasedRoute (ProtectedRoute)
  - For separating routes based on roles.

h) Page-Level Views (Routes)

These routes are to guide you and provide a baseline, you are free to add more and use your own routes.

Frontend Route	**Component / Usage**	**Access Role**
`/`	EventList {uses EventCard}	Public
`/login`	LoginForm	Public
`/register`	RegisterForm	Public
`/forgot-password`	ForgotPassword	Public
`/logout`	Navbar logout action	Authenticated
`/profile`	ProfilePage, UpdateProfileForm	All Authenticated Users
`/events/:id`	EventDetails, BookTicketForm	Public, Standard User
`/bookings`	UserBookingsPage {uses EventCard}	Standard User
`/bookings/:id`	BookingDetails	Standard User
`/my-events`	MyEventsPage {uses EventCard}	Organizer
`/my-events/new`	EventForm	Organizer
`/my-events/:id/edit`	EventForm	Organizer
`/my-events/analytics`	EventAnalytics	Organizer
`/admin/events`	AdminEventsPage {uses EventCard}	Admin
`/admin/users`	AdminUsersPage	Admin
`/unauthorized`	UnauthorizedPage	All

Table 1: Frontend Routes with Components, Roles, and EventCard Usage

5 Technology Stack

- React

6 Deliverables

- Fully integrated and functional MERN stack project with MVC architecture.

7 Bonus

- Deployment (1.5%)
- Good UI/UX (1.5%)
- MFA handled in the frontend. (2%)

8 Submission Requirements

- Each team member must collaborate & commit his work through his **branch**.

9 Task Deadline

Task 3 deadline is Saturday, 24th May , 11:59 pm

10 Submission

a) Github repos link to be submitted through this form before the mentioned deadline: https://forms.gle/YuFjDb9aaR8PNvZV7

b) You have to invite the following user to your Github private repository to give us access: username: SEspring25