Turf Management System

App.jsx:

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-router-dom';
import Welcome from './pages/Welcome';
import Login from './pages/Login';
import Register from './pages/Register';
import AdminDashboard from './pages/AdminDashboard';
import UserDashboard from './pages/UserDashboard';
import Cart from './pages/Cart';
import Checkout from './pages/Checkout';
import EditTurf from './pages/EditTurf';


function App() {
  const user = JSON.parse(localStorage.getItem('user'));

  return (
    <Router>
      <div className="App">
        <Routes>
          {/* Public routes */}
          <Route path="/" element={<Welcome />} />
          <Route
            path="/login"
            element={!user ? <Login /> : <Navigate to={user.role === 'admin' ? '/admin' : '/user'}
/>}
          />
          <Route
            path="/register"
            element={!user ? <Register /> : <Navigate to={user.role === 'admin' ? '/admin' :
'/user'} />}
          />

          {/* Protected admin routes */}
          <Route
            path="/admin"
            element={user && user.role === 'admin' ? <AdminDashboard /> : <Navigate
to="/login" />}
          />
          <Route
            path="/edit-turf/:id"
            element={user && user.role === 'admin' ? <EditTurf /> : <Navigate to="/login" />}
          />

          {/* Protected user routes */}
          <Route
            path="/user"
            element={user && user.role === 'user' ? <UserDashboard /> : <Navigate to="/login"
/>}
          />
          <Route
            path="/cart"
            element={user && user.role === 'user' ? <Cart /> : <Navigate to="/login" />}
          />
          <Route
            path="/checkout"
            element={user && user.role === 'user' ? <Checkout /> : <Navigate to="/login" />}
          />

          {/* Catch all route - redirect to home */}
          <Route path="*" element={<Navigate to="/" />} />
        </Routes>
      </div>
    </Router>
   );
}

export default App;
```

AdminNavbar.jsx:

```
import React from 'react';
import { Navbar, Nav, Container } from 'react-bootstrap';
import { useNavigate } from 'react-router-dom';

const AdminNavbar = () => {
 const navigate = useNavigate();

 const handleLogout = () => {
  localStorage.removeItem('user');
  navigate('/');
 };

 return (
  <Navbar bg="dark" variant="dark" expand="lg">
```

```
      <Container>
        <Navbar.Brand href="#home">Turf Admin</Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="me-auto">
            <Nav.Link onClick={() => navigate('/admin')}>Home</Nav.Link>
            <Nav.Link onClick={() => navigate('/admin')}>Manage Turfs</Nav.Link>
            <Nav.Link onClick={() => navigate('/admin')}>View Bookings</Nav.Link>
          </Nav>
          <Nav>
            <Nav.Link onClick={handleLogout}>Logout</Nav.Link>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
};

export default AdminNavbar;
```

TurfCard.jsx:

```
import React from 'react';
import { Card, Button, Badge } from 'react-bootstrap';

const TurfCard = ({ turf, onEdit, onDelete, onAddToCart, isAdmin = false }) => {
  return (
    <Card style={{ width: '18rem', margin: '10px' }}>
      <Card.Body>
```

```
          <Card.Title>{turf.title}</Card.Title>
          <Card.Text>
            {turf.description}
            <br />
            <Badge bg="info">Location: {turf.location}</Badge>
            <br />
            <Badge bg="success">Price: ${turf.price}</Badge>
          </Card.Text>
          {isAdmin ? (
            <>
              <Button variant="primary" onClick={() => onEdit(turf)}>Edit</Button>{' '}
              <Button variant="danger" onClick={() => onDelete(turf.id)}>Delete</Button>
            </>
          ) : (
            <Button variant="primary" onClick={() => onAddToCart(turf)}>Add to Cart</Button>
          )}
        </Card.Body>
      </Card>
  );
};

export default TurfCard;
```

UserNavbar.jsx:

```
import React from 'react';
import { Navbar, Nav, Container, Badge } from 'react-bootstrap';
import { useNavigate } from 'react-router-dom';
import { useState, useEffect } from 'react';
import { cartService } from '../services/ApiService';

const UserNavbar = () => {
  const navigate = useNavigate();
  const [cartItems, setCartItems] = useState([]);

  useEffect(() => {
    cartService.getCart().then(response => {
      setCartItems(response.data);
    });
  }, []);

  const handleLogout = () => {
    localStorage.removeItem('user');
    navigate('/');
  };

  return (
    <Navbar bg="primary" variant="dark" expand="lg">
      <Container>
        <Navbar.Brand>Turf User</Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="me-auto">
            <Nav.Link onClick={() => navigate('/user')}>Browse Turfs</Nav.Link>
            <Nav.Link onClick={() => navigate('/cart')}>
              Cart <Badge bg="secondary">{cartItems.length}</Badge>
            </Nav.Link>
            <Nav.Link onClick={() => navigate('/checkout')}>Checkout</Nav.Link>
          </Nav>
          <Nav>
            <Nav.Link onClick={handleLogout}>Logout</Nav.Link>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
};

export default UserNavbar;
```

AdminDashboard.jsx:

```
import React, { useState, useEffect } from 'react';
import { Container, Row, Col, Button, Modal, Alert } from 'react-bootstrap';
import { Formik, Form, Field } from 'formik';
import * as Yup from 'yup';
import { useNavigate } from 'react-router-dom';
import AdminNavbar from '../components/AdminNavbar';
import TurfCard from '../components/TurfCard';
import { turfService, bookingService } from '../services/ApiService';

const TurfSchema = Yup.object().shape({
 title: Yup.string().required('Required'),
 location: Yup.string().required('Required'),
 price: Yup.number().required('Required').positive('Price must be positive'),
 description: Yup.string().required('Required'),
});

const AdminDashboard = () => {
 const navigate = useNavigate();
 const [turfs, setTurfs] = useState([]);
 const [bookings, setBookings] = useState([]);
 const [showModal, setShowModal] = useState(false);
 const [editingTurf, setEditingTurf] = useState(null);
 const [error, setError] = useState('');
 const [activeTab, setActiveTab] = useState('turfs');

 useEffect(() => {
   loadTurfs();
   loadBookings();
 }, []);

 const loadTurfs = () => {
   turfService.getAll()
     .then(response => setTurfs(response.data))
     .catch(error => setError('Failed to load turfs'));
 };

 const loadBookings = () => {
   bookingService.getAll()
     .then(response => setBookings(response.data))
     .catch(error => setError('Failed to load bookings'));
 };
```

```
const handleCreateTurf = (values, { setSubmitting, resetForm }) => {
  const turfData = {
    ...values,
    price: Number(values.price)
  };

  turfService.create(turfData)
    .then(() => {
      loadTurfs();
      setShowModal(false);
      resetForm();
      setError('');
    })
    .catch(error => setError('Failed to create turf'))
    .finally(() => setSubmitting(false));
};

const handleEditTurf = (turf) => {
  setEditingTurf(turf);
  navigate(`/edit-turf/${turf.id}`);
};

const handleDeleteTurf = (id) => {
  if (window.confirm('Are you sure you want to delete this turf?')) {
    turfService.delete(id)
      .then(() => {
        loadTurfs();
        setError('');
      })
      .catch(error => setError('Failed to delete turf'));
  }
};

const handleCloseModal = () => {
  setShowModal(false);
  setEditingTurf(null);
};

return (
  <>
    <AdminNavbar />
    <Container className="mt-4">
      <Row>
        <Col>
          <h2>Admin Dashboard</h2>
          <Button variant="primary" onClick={() => setActiveTab('turfs')} className="me-2">
            Manage Turfs
          </Button>
          <Button variant="secondary" onClick={() => setActiveTab('bookings')}>
            View Bookings
          </Button>
        </Col>
      </Row>

      {error && <Alert variant="danger" className="mt-3">{error}</Alert>}
```

```
{activeTab === 'turfs' && (
 <>
  <Row className="mt-3">
   <Col>
    <Button variant="success" onClick={() => setShowModal(true)}>
     Add New Turf
    </Button>
   </Col>
  </Row>

  <Row className="mt-3">
   {turfs.map(turf => (
    <Col md={4} key={turf.id}>
     <TurfCard
      turf={turf}
      onEdit={handleEditTurf}
      onDelete={handleDeleteTurf}
      isAdmin={true}
     />
    </Col>
   ))}
  </Row>

  <Modal show={showModal} onHide={handleCloseModal}>
   <Modal.Header closeButton>
    <Modal.Title>Add New Turf</Modal.Title>
   </Modal.Header>
```

```
            <Modal.Body>
             <Formik
              initialValues={{ title: '', location: '', price: '', description: '' }}
              validationSchema={TurfSchema}
              onSubmit={handleCreateTurf}
             >
              {({ errors, touched, isSubmitting }) => (
               <Form>
                <div className="mb-3">
                 <label htmlFor="title" className="form-label">Title</label>
                 <Field
                  type="text"
                  name="title"
                  className={`form-control ${errors.title && touched.title ? 'is-invalid' : ''}`}
                 />
                 {errors.title && touched.title && (
                  <div className="invalid-feedback">{errors.title}</div>
                 )}
                </div>

                <div className="mb-3">
                 <label htmlFor="location" className="form-label">Location</label>
                 <Field
                  type="text"
                  name="location"
                  className={`form-control ${errors.location && touched.location ? 'is-invalid' :
''}`}
                 />
```

```
                  {errors.location && touched.location && (
                    <div className="invalid-feedback">{errors.location}</div>
                  )}
                </div>

                <div className="mb-3">
                  <label htmlFor="price" className="form-label">Price</label>
                  <Field
                    type="number"
                    name="price"
                    className={`form-control ${errors.price && touched.price ? 'is-invalid' : ''}`}
                  />
                  {errors.price && touched.price && (
                    <div className="invalid-feedback">{errors.price}</div>
                  )}
                </div>

                <div className="mb-3">
                  <label htmlFor="description" className="form-label">Description</label>
                  <Field
                    as="textarea"
                    name="description"
                    className={`form-control ${errors.description && touched.description ? 'is-
invalid' : ''}`}
                  />
                  {errors.description && touched.description && (
                    <div className="invalid-feedback">{errors.description}</div>
                  )}
```

```
            </div>


            <Button type="submit" variant="primary" disabled={isSubmitting}>
              {isSubmitting ? 'Creating...' : 'Create Turf'}
            </Button>
          </Form>
        )}
      </Formik>
    </Modal.Body>
  </Modal>
</>
)}


{activeTab === 'bookings' && (
  <Row className="mt-3">
    <Col>
      <h3>Bookings</h3>
      {bookings.length === 0 ? (
        <p>No bookings found.</p>
      ) : (
        <table className="table table-striped">
          <thead>
            <tr>
              <th>ID</th>
              <th>User</th>
              <th>Turf</th>
              <th>Date</th>
            </tr>
```

```
         </thead>
         <tbody>
          {bookings.map(booking => (
           <tr key={booking.id}>
            <td>{booking.id}</td>
            <td>{booking.userName}</td>
            <td>{booking.turfTitle}</td>
            <td>{new Date(booking.date).toLocaleDateString()}</td>
           </tr>
          ))}
         </tbody>
        </table>
       )}
      </Col>
     </Row>
    )}
   </Container>
  </>
 );
};

export default AdminDashboard;
```

Cart.jsx:

```
import React, { useState, useEffect } from 'react';
import { Container, Row, Col, Table, Button, Alert } from 'react-bootstrap';
import { useNavigate } from 'react-router-dom';
import UserNavbar from '../components/UserNavbar';
import { cartService } from '../services/ApiService';

const Cart = () => {
  const navigate = useNavigate();
  const [cartItems, setCartItems] = useState([]);
  const [error, setError] = useState('');

  useEffect(() => {
    loadCart();
  }, []);

  const loadCart = () => {
    const userId = JSON.parse(localStorage.getItem('user')).id;

    cartService.getCart()
      .then(response => {
        const userCart = response.data.filter(item => item.userId === userId);
        setCartItems(userCart);
      })
      .catch(error => setError('Failed to load cart'));
  };

  const handleUpdateQuantity = (id, newQuantity) => {
    if (newQuantity < 1) return;

    cartService.updateCartItem(id, { quantity: newQuantity })
      .then(() => loadCart())
      .catch(error => setError('Failed to update quantity'));
```

```
};

const handleRemoveItem = (id) => {
  cartService.removeFromCart(id)
    .then(() => loadCart())
    .catch(error => setError('Failed to remove item'));
};

const getTotalPrice = () => {
  return cartItems.reduce((total, item) => total + (item.price * item.quantity), 0);
};

return (
  <>
    <UserNavbar />
    <Container className="mt-4">
      <Row>
        <Col>
          <h2>Shopping Cart</h2>
        </Col>
      </Row>

      {error && <Alert variant="danger">{error}</Alert>}

      {cartItems.length === 0 ? (
        <Row className="mt-3">
          <Col>
            <p>Your cart is empty.</p>
      <Button variant="primary" onClick={() => navigate('/user')}>
        Browse Turfs
      </Button>
    </Col>
  </Row>
) : (
  <>
    <Row className="mt-3">
      <Col>
        <Table striped bordered hover>
          <thead>
            <tr>
              <th>Turf</th>
              <th>Price</th>
              <th>Quantity</th>
              <th>Total</th>
              <th>Action</th>
            </tr>
          </thead>
          <tbody>
            {cartItems.map(item => (
              <tr key={item.id}>
                <td>{item.title}</td>
                <td>${item.price}</td>
                <td>
                  <Button
                    variant="outline-secondary"
                    size="sm"
              onClick={() => handleUpdateQuantity(item.id, item.quantity - 1)}
            >
              -
            </Button>
            <span className="mx-2">{item.quantity}</span>
            <Button
              variant="outline-secondary"
              size="sm"
              onClick={() => handleUpdateQuantity(item.id, item.quantity + 1)}
            >
              +
            </Button>
          </td>
          <td>${item.price * item.quantity}</td>
          <td>
            <Button
              variant="danger"
              size="sm"
              onClick={() => handleRemoveItem(item.id)}
            >
              Remove
            </Button>
          </td>
        </tr>
      ))}
    </tbody>
  </Table>
</Col>
```

```
          </Row>

          <Row>
            <Col className="text-end">
              <h4>Total: ${getTotalPrice()}</h4>
              <Button variant="success" onClick={() => navigate('/checkout')}>
                Proceed to Checkout
              </Button>
            </Col>
          </Row>
        </>
      )}
    </Container>
  </>
 );
};

export default Cart;
```

Checkout.jsx:

```
import React, { useState, useEffect } from 'react';
import { Container, Row, Col, Card, Button, Alert } from 'react-bootstrap';
import { useNavigate } from 'react-router-dom';
import UserNavbar from '../components/UserNavbar';
import { cartService, bookingService } from '../services/ApiService';

const Checkout = () => {
  const navigate = useNavigate();
const [cartItems, setCartItems] = useState([]);
const [error, setError] = useState('');
const [success, setSuccess] = useState('');

useEffect(() => {
  loadCart();
}, []);

const loadCart = () => {
  const userId = JSON.parse(localStorage.getItem('user')).id;

  cartService.getCart()
    .then(response => {
      const userCart = response.data.filter(item => item.userId === userId);
      setCartItems(userCart);
    })
    .catch(error => setError('Failed to load cart'));
};

const handleCheckout = () => {
  const user = JSON.parse(localStorage.getItem('user'));
  const bookingDate = new Date().toISOString();

  // Create a booking for each cart item
  const bookingPromises = cartItems.map(item => {
    const booking = {
      userId: user.id,
      userName: user.name,
      turfId: item.turfId,
      turfTitle: item.title,
      date: bookingDate,
      total: item.price * item.quantity
    };

    return bookingService.create(booking);
  });

  Promise.all(bookingPromises)
    .then(() => {
      // Clear the cart
      return cartService.clearCart();
    })
    .then(() => {
      setSuccess('Checkout successful!');
      setTimeout(() => navigate('/user'), 2000);
    })
    .catch(error => setError('Checkout failed. Please try again.'));
};

const getTotalPrice = () => {
  return cartItems.reduce((total, item) => total + (item.price * item.quantity), 0);
};

return (
  <>
    <UserNavbar />
```

```
<Container className="mt-4">
 <Row>
  <Col>
   <h2>Checkout</h2>
  </Col>
 </Row>

 {error && <Alert variant="danger">{error}</Alert>}
 {success && <Alert variant="success">{success}</Alert>}

 {cartItems.length === 0 ? (
  <Row className="mt-3">
   <Col>
    <p>Your cart is empty.</p>
    <Button variant="primary" onClick={() => navigate('/user')}>
     Browse Turfs
    </Button>
   </Col>
  </Row>
 ) : (
  <>
   <Row className="mt-3">
    <Col md={8}>
     <Card>
      <Card.Header>
       <h4>Order Summary</h4>
      </Card.Header>
      <Card.Body>
```

```
              {cartItems.map(item => (
                <div key={item.id} className="d-flex justify-content-between mb-2">
                  <div>
                    {item.title} x {item.quantity}
                  </div>
                  <div>${item.price * item.quantity}</div>
                </div>
              ))}
              <hr />
              <div className="d-flex justify-content-between">
                <strong>Total:</strong>
                <strong>${getTotalPrice()}</strong>
              </div>
            </Card.Body>
          </Card>
        </Col>
      </Row>

      <Row className="mt-3">
        <Col>
          <Button variant="success" onClick={handleCheckout}>
            Confirm Checkout
          </Button>
        </Col>
      </Row>
    </>
  )}
</Container>
```

```
    </>
  );
};

export default Checkout;
```

EditTurf.jsx:

```
import React, { useState, useEffect } from 'react';
import { Formik, Form, Field } from 'formik';
import * as Yup from 'yup';
import { Container, Row, Col, Card, Button, Alert } from 'react-bootstrap';
import { useNavigate, useParams } from 'react-router-dom';
import AdminNavbar from '../components/AdminNavbar';
import { turfService } from '../services/ApiService';

const TurfSchema = Yup.object().shape({
  title: Yup.string().required('Required'),
  location: Yup.string().required('Required'),
  price: Yup.number().required('Required').positive('Price must be positive'),
  description: Yup.string().required('Required'),
});

const EditTurf = () => {
  const navigate = useNavigate();
  const { id } = useParams();
  const [turf, setTurf] = useState(null);
  const [error, setError] = useState('');
```

```
useEffect(() => {
  loadTurf();
}, [id]);

const loadTurf = () => {
  turfService.getById(id)
    .then(response => setTurf(response.data))
    .catch(error => setError('Failed to load turf'));
};

const handleSubmit = (values, { setSubmitting }) => {
  const turfData = {
    ...values,
    price: Number(values.price)
  };

  turfService.update(id, turfData)
    .then(() => {
      navigate('/admin');
      setError('');
    })
    .catch(error => setError('Failed to update turf'))
    .finally(() => setSubmitting(false));
};

if (!turf) {
  return (
    <>
```

```
      <AdminNavbar />
      <Container className="mt-4">
        <Row>
          <Col>
            <p>Loading...</p>
          </Col>
        </Row>
      </Container>
    </>
  );
}

return (
  <>
    <AdminNavbar />
    <Container className="mt-4">
      <Row className="justify-content-center">
        <Col md={6}>
          <Card>
            <Card.Header as="h3">Edit Turf</Card.Header>
            <Card.Body>
              {error && <Alert variant="danger">{error}</Alert>}
              <Formik
                initialValues={{
                  title: turf.title,
                  location: turf.location,
                  price: turf.price,
                  description: turf.description
        }}
        validationSchema={TurfSchema}
        onSubmit={handleSubmit}
       >
        {({ errors, touched, isSubmitting }) => (
         <Form>
          <div className="mb-3">
           <label htmlFor="title" className="form-label">Title</label>
           <Field
            type="text"
            name="title"
            className={`form-control ${errors.title && touched.title ? 'is-invalid' : ''}`}
           />
           {errors.title && touched.title && (
            <div className="invalid-feedback">{errors.title}</div>
           )}
          </div>

          <div className="mb-3">
           <label htmlFor="location" className="form-label">Location</label>
           <Field
            type="text"
            name="location"
            className={`form-control ${errors.location && touched.location ? 'is-invalid' :
''}`}
           />
           {errors.location && touched.location && (
            <div className="invalid-feedback">{errors.location}</div>
          )}
        </div>

        <div className="mb-3">
          <label htmlFor="price" className="form-label">Price</label>
          <Field
            type="number"
            name="price"
            className={`form-control ${errors.price && touched.price ? 'is-invalid' : ''}`}
          />
          {errors.price && touched.price && (
            <div className="invalid-feedback">{errors.price}</div>
          )}
        </div>

        <div className="mb-3">
          <label htmlFor="description" className="form-label">Description</label>
          <Field
            as="textarea"
            name="description"
            className={`form-control ${errors.description && touched.description ? 'is-
invalid' : ''}`}
          />
          {errors.description && touched.description && (
            <div className="invalid-feedback">{errors.description}</div>
          )}
        </div>
```

```
           <Button type="submit" variant="primary" disabled={isSubmitting}
className="me-2">
            {isSubmitting ? 'Updating...' : 'Update Turf'}
           </Button>
           <Button variant="secondary" onClick={() => navigate('/admin')}>
            Cancel
           </Button>
          </Form>
         )}
        </Formik>
       </Card.Body>
      </Card>
     </Col>
    </Row>
   </Container>
  </>
 );
};

export default EditTurf;
```

Login.jsx:

```
import React from 'react';
import { Formik, Form, Field } from 'formik';
import * as Yup from 'yup';
import { Container, Row, Col, Card, Button, Alert } from 'react-bootstrap';
import { useNavigate } from 'react-router-dom';
import { authService } from '../services/ApiService';
```

```
const LoginSchema = Yup.object().shape({
  email: Yup.string().email('Invalid email').required('Required'),
  password: Yup.string().required('Required'),
});

const Login = () => {
  const navigate = useNavigate();
  const [error, setError] = React.useState('');

  const handleSubmit = (values, { setSubmitting }) => {
    setError('');
    authService.login(values.email, values.password)
      .then(response => {
        if (response.data.length > 0) {
          const user = response.data[0];
          localStorage.setItem('user', JSON.stringify(user));

          if (user.role === 'admin') {
            navigate('/admin');
          } else {
            navigate('/user');
          }
        } else {
          setError('Invalid email or password');
        }
        setSubmitting(false);
      })
    .catch(error => {
      setError('Login failed. Please try again.');
      setSubmitting(false);
    });
  };

  return (
    <Container className="mt-5">
      <Row className="justify-content-center">
        <Col md={6}>
          <Card>
            <Card.Header as="h3">Login</Card.Header>
            <Card.Body>
              {error && <Alert variant="danger">{error}</Alert>}
              <Formik
                initialValues={{ email: '', password: '' }}
                validationSchema={LoginSchema}
                onSubmit={handleSubmit}
              >
                {({ errors, touched, isSubmitting }) => (
                  <Form>
                    <div className="mb-3">
                      <label htmlFor="email" className="form-label">Email</label>
                      <Field
                        type="email"
                        name="email"
                        className={`form-control ${errors.email && touched.email ? 'is-invalid' : ''}`}
                      />
                      {errors.email && touched.email && (
                        <div className="invalid-feedback">{errors.email}</div>
                      )}
                    </div>

                    <div className="mb-3">
                      <label htmlFor="password" className="form-label">Password</label>
                      <Field
                        type="password"
                        name="password"
                        className={`form-control ${errors.password && touched.password ? 'is-
invalid' : ''}`}
                      />
                      {errors.password && touched.password && (
                        <div className="invalid-feedback">{errors.password}</div>
                      )}
                    </div>

                    <Button
                      type="submit"
                      variant="primary"
                      disabled={isSubmitting}
                    >
                      {isSubmitting ? 'Logging in...' : 'Login'}
                    </Button>
                  </Form>
                )}
              </Formik>
```

```
          </Card.Body>
        </Card>
      </Col>
    </Row>
  </Container>
 );
};

export default Login;

Register.jsx:
import React from 'react';
import { Formik, Form, Field } from 'formik';
import * as Yup from 'yup';
import { Container, Row, Col, Card, Button, Alert } from 'react-bootstrap';
import { useNavigate } from 'react-router-dom';
import { authService } from '../services/ApiService';

const RegisterSchema = Yup.object().shape({
 name: Yup.string().required('Required'),
 email: Yup.string().email('Invalid email').required('Required'),
 password: Yup.string().min(6, 'Password must be at least 6
characters').required('Required'),
});

const Register = () => {
 const navigate = useNavigate();
 const [error, setError] = React.useState('');
const [success, setSuccess] = React.useState('');

const handleSubmit = (values, { setSubmitting }) => {
  setError('');
  setSuccess('');

  // Check if email already exists
  authService.login(values.email, values.password)
    .then(response => {
      if (response.data.length > 0) {
        setError('Email already exists');
        setSubmitting(false);
      } else {
        // Register new user
        const newUser = {
          name: values.name,
          email: values.email,
          password: values.password,
          role: 'user'
        };

        authService.register(newUser)
          .then(() => {
            setSuccess('Registration successful. Please login.');
            setTimeout(() => navigate('/login'), 2000);
            setSubmitting(false);
          })
          .catch(error => {
            setError('Registration failed. Please try again.');
            setSubmitting(false);
          });
      }
    })
    .catch(error => {
      setError('Registration failed. Please try again.');
      setSubmitting(false);
    });
};

return (
  <Container className="mt-5">
    <Row className="justify-content-center">
      <Col md={6}>
        <Card>
          <Card.Header as="h3">Register</Card.Header>
          <Card.Body>
            {error && <Alert variant="danger">{error}</Alert>}
            {success && <Alert variant="success">{success}</Alert>}
            <Formik
              initialValues={{ name: '', email: '', password: '' }}
              validationSchema={RegisterSchema}
              onSubmit={handleSubmit}
            >
              {({ errors, touched, isSubmitting }) => (
                <Form>
                  <div className="mb-3">
  <label htmlFor="name" className="form-label">Name</label>
  <Field
    type="text"
    name="name"
    className={`form-control ${errors.name && touched.name ? 'is-invalid' : ''}`}
  />
  {errors.name && touched.name && (
    <div className="invalid-feedback">{errors.name}</div>
  )}
</div>

<div className="mb-3">
  <label htmlFor="email" className="form-label">Email</label>
  <Field
    type="email"
    name="email"
    className={`form-control ${errors.email && touched.email ? 'is-invalid' : ''}`}
  />
  {errors.email && touched.email && (
    <div className="invalid-feedback">{errors.email}</div>
  )}
</div>

<div className="mb-3">
  <label htmlFor="password" className="form-label">Password</label>
  <Field
    type="password"
    name="password"
```

```
                    className={`form-control ${errors.password && touched.password ? 'is-
invalid' : ''}`}
                  />
                  {errors.password && touched.password && (
                    <div className="invalid-feedback">{errors.password}</div>
                  )}
                </div>

                <Button
                  type="submit"
                  variant="primary"
                  disabled={isSubmitting}
                >
                  {isSubmitting ? 'Registering...' : 'Register'}
                </Button>
              </Form>
            )}
          </Formik>
        </Card.Body>
      </Card>
    </Col>
  </Row>
</Container>
);
};

export default Register;
```

UserDashboard.jsx:

```
import React, { useState, useEffect } from 'react';
import { Container, Row, Col, Alert } from 'react-bootstrap';
import UserNavbar from '../components/UserNavbar';
import TurfCard from '../components/TurfCard';
import { turfService, cartService } from '../services/ApiService';

const UserDashboard = () => {
  const [turfs, setTurfs] = useState([]);
  const [error, setError] = useState('');

  useEffect(() => {
    loadTurfs();
  }, []);

  const loadTurfs = () => {
    turfService.getAll()
      .then(response => setTurfs(response.data))
      .catch(error => setError('Failed to load turfs'));
  };

  const handleAddToCart = (turf) => {
    const cartItem = {
      turfId: turf.id,
      title: turf.title,
      price: turf.price,
      quantity: 1,
      userId: JSON.parse(localStorage.getItem('user')).id
```

```
 };

 cartService.addToCart(cartItem)
   .then(() => {
     alert('Added to cart successfully!');
     setError('');
   })
   .catch(error => setError('Failed to add to cart'));
};

return (
 <>
   <UserNavbar />
   <Container className="mt-4">
     <Row>
       <Col>
         <h2>Browse Turfs</h2>
       </Col>
     </Row>

     {error && <Alert variant="danger">{error}</Alert>}

     <Row className="mt-3">
       {turfs.map(turf => (
         <Col md={4} key={turf.id}>
           <TurfCard
             turf={turf}
             onAddToCart={handleAddToCart}
```

```
            />
          </Col>
        ))}
      </Row>
    </Container>
  </>
 );
};

export default UserDashboard;
```

Welcome.jsx:

```
import React from 'react';
import { Container, Row, Col, Card, Button } from 'react-bootstrap';
import { useNavigate } from 'react-router-dom';

const Welcome = () => {
  const navigate = useNavigate();

  return (
    <Container className="mt-5">
      <Row className="justify-content-center">
        <Col md={8}>
          <Card className="text-center">
            <Card.Header as="h1">Welcome to Turf Management System</Card.Header>
            <Card.Body>
              <Card.Text>
                Manage and book turfs with our easy-to-use system
              </Card.Text>
              <Button variant="primary" className="me-3" onClick={() => navigate('/login')}>
                Login
              </Button>
              <Button variant="secondary" onClick={() => navigate('/register')}>
                Register
              </Button>
            </Card.Body>
          </Card>
        </Col>
      </Row>
    </Container>
  );
};

export default Welcome;
```

ApiService.jsx:

```
import axios from 'axios';

const API_BASE_URL = 'http://localhost:3001';

const api = axios.create({
  baseURL: API_BASE_URL,
});

export const authService = {
  login: (email, password) => api.get(`/users?email=${email}&password=${password}`),
  register: (userData) => api.post('/users', userData),
};

export const turfService = {
  getAll: () => api.get('/turfs'),
  getById: (id) => api.get(`/turfs/${id}`),
  create: (turfData) => api.post('/turfs', turfData),
  update: (id, turfData) => api.put(`/turfs/${id}`, turfData),
  delete: (id) => api.delete(`/turfs/${id}`),
};

export const cartService = {
  getCart: () => api.get('/cart'),
  addToCart: (item) => api.post('/cart', item),
  updateCartItem: (id, item) => api.put(`/cart/${id}`, item),
  removeFromCart: (id) => api.delete(`/cart/${id}`),
  clearCart: () => api.get('/cart').then(response => {
    const deletePromises = response.data.map(item => api.delete(`/cart/${item.id}`));
    return Promise.all(deletePromises);
  }),
};

export const bookingService = {
  getAll: () => api.get('/bookings'),
  create: (bookingData) => api.post('/bookings', bookingData),
};

export default api;
```