# Documentation

## Brief Documentation Explaining How to Use the System

This document provides a brief overview of how to use the Book Inventory system. Below are the main instructions for navigating and using the key features of the application.

### 1. Using the Frontend

The front-end of the application is built using React and offers an intuitive user interface for managing books in inventory. Users can perform the following actions:  
- Add a new book to the inventory.  
- Update an existing book’s details.  
- Delete a book from the inventory.  
- View all books in a well-organized list or card format.  
- Filter and search for books by title and genre.  
- Export the inventory list to CSV or JSON format.

### 2. Using the Backend

The backend is powered by Django REST Framework and provides a robust API for interacting with the book inventory. Key endpoints include:  
`GET /inventory/`: Retrieve the list of all books.  
`POST /inventory/`: Add a new book.  
`PUT /inventory/<id>/`: Update a book's details.  
`DELETE /inventory/<id>/`: Delete a book.

### 3. API Error Handling

The system is designed to handle API errors gracefully. If an API request fails, the user will receive a detailed error message. For instance, if the ISBN is invalid or a required field is missing, the user will be alerted to fix the issue.

## Design Decisions

Several design decisions were made to enhance the user experience and system performance:  
**Frontend and Backend Separation**: The frontend and backend are developed separately to ensure modularity and scalability. This allows for easier maintenance and the ability to independently update components.  
**Responsive Design**: The frontend is designed to be responsive, providing a seamless experience across various devices, including desktops, tablets, and mobile phones.  
**Modal for Editing**: A decision was made to use modals for editing book details to provide a smoother user experience, but the form-based approach was kept for simplicity.

## Challenges Faced During Development

Several challenges were encountered during development:  
**Handling API Errors**: Ensuring that meaningful error messages were displayed to the user required additional effort in handling API responses and parsing error messages.  
**Cross-Browser Compatibility**: Ensuring the frontend worked consistently across different browsers was a challenge that required extensive testing and tweaking of CSS styles.  
**Database Migrations**: Managing database migrations and ensuring a smooth setup for new developers joining the project involved careful planning and documentation.