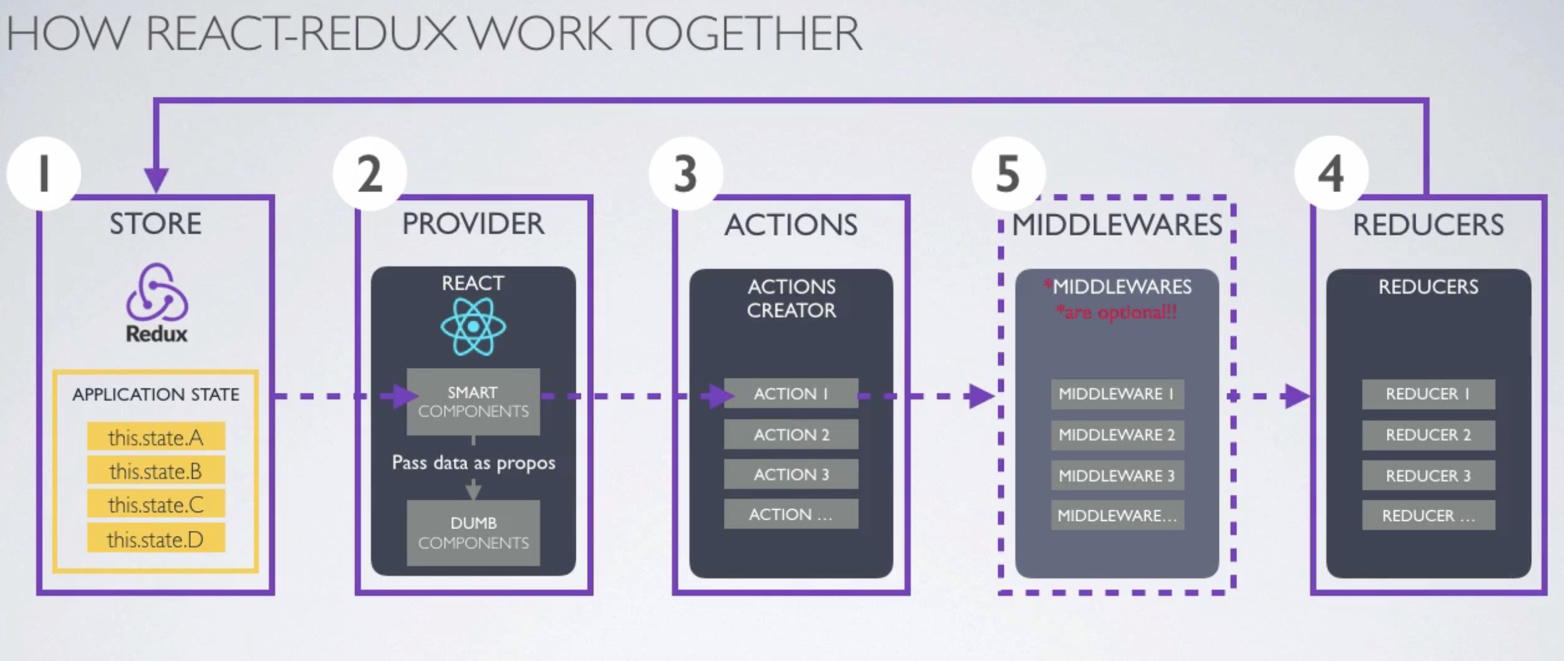
**REDUX – REACT**

**Introduction**



1. Store holds the application state
2. Provider wraps the react application and passes the state of the application to the components. The components that trigger /dispatch action are called smart components.
3. Actions are plain javascript objects that provide information. This information is sent using store.dispatch(type, action) where type must always be specified and action that contains payload of information is optional.

e.g store.dispatch(“POST\_BOOK”, book)

1. The reducer is a pure function that takes the previous state and an action, and returns the next state. The functions such as splice, push should not be used in reducer. Use concat, slice, spread operator instead.
2. Middlewares are optional and can exist if any additional task that

is needed once action is dispatched. For e.g. redux-logger can be configured as middleware to review logs before and after action is dispatched or store state is changed, redux-thunk can be used as middleware along with axios while making rest API call.

**Principles of Redux:**

1. The state of application is stored in a single object tree called a store
2. The state is read-only. Only way to change the state is to dispatch an action
3. Change is made using pure functions called reducers

**Implementation Steps:**

1. Create store using reducer (or combined reducer)
2. Write action
3. Write reducer for the action
4. Export component with connect so that it gets connected to store
5. Component dispatches action: uses mapDispatchToProps and bindActionCreators
6. Reducer updates the store and changes its state
7. Component retrieves the changed state from store: uses mapStateToProps
8. React updates UI and re-renders the change

**Example: GPBookShop - React Redux Application**

**Objective:** To demonstrate implementation of Redux and React

**Structure:**

**actions/** - contains actions for book components and cart components

**reducers/** - contains reducers for book components and cart components

**reducers/index.js** – combines the reducers

**app.js** – imports combined reducers, imports components, creates

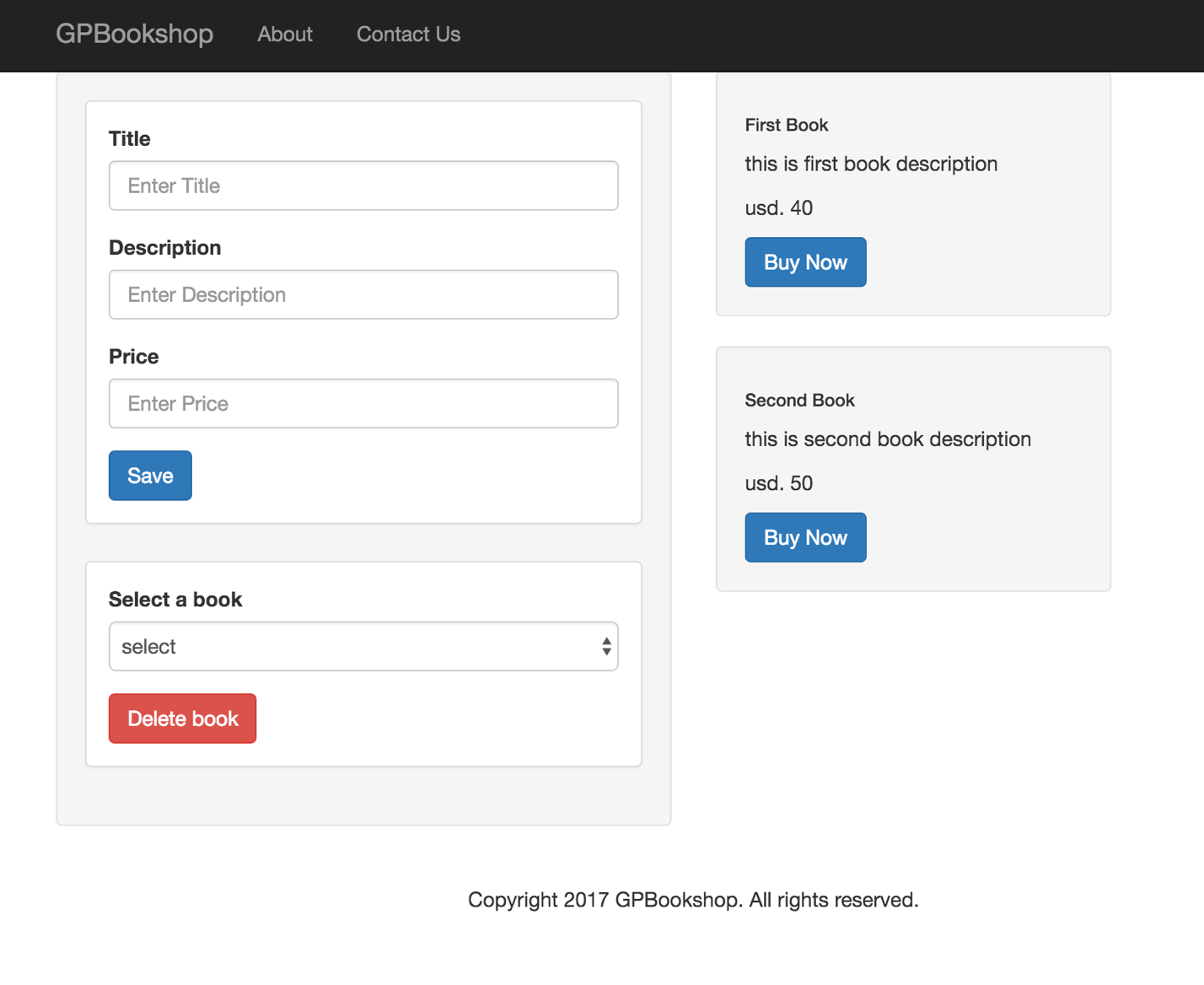
store; provides routes using router

**main.js** – used to display menu, footer and child components

**components/** - contains display only react components

**components/pages** – contains smart react components that dispatch action

**public/** - index.html launches application at localhost:3000

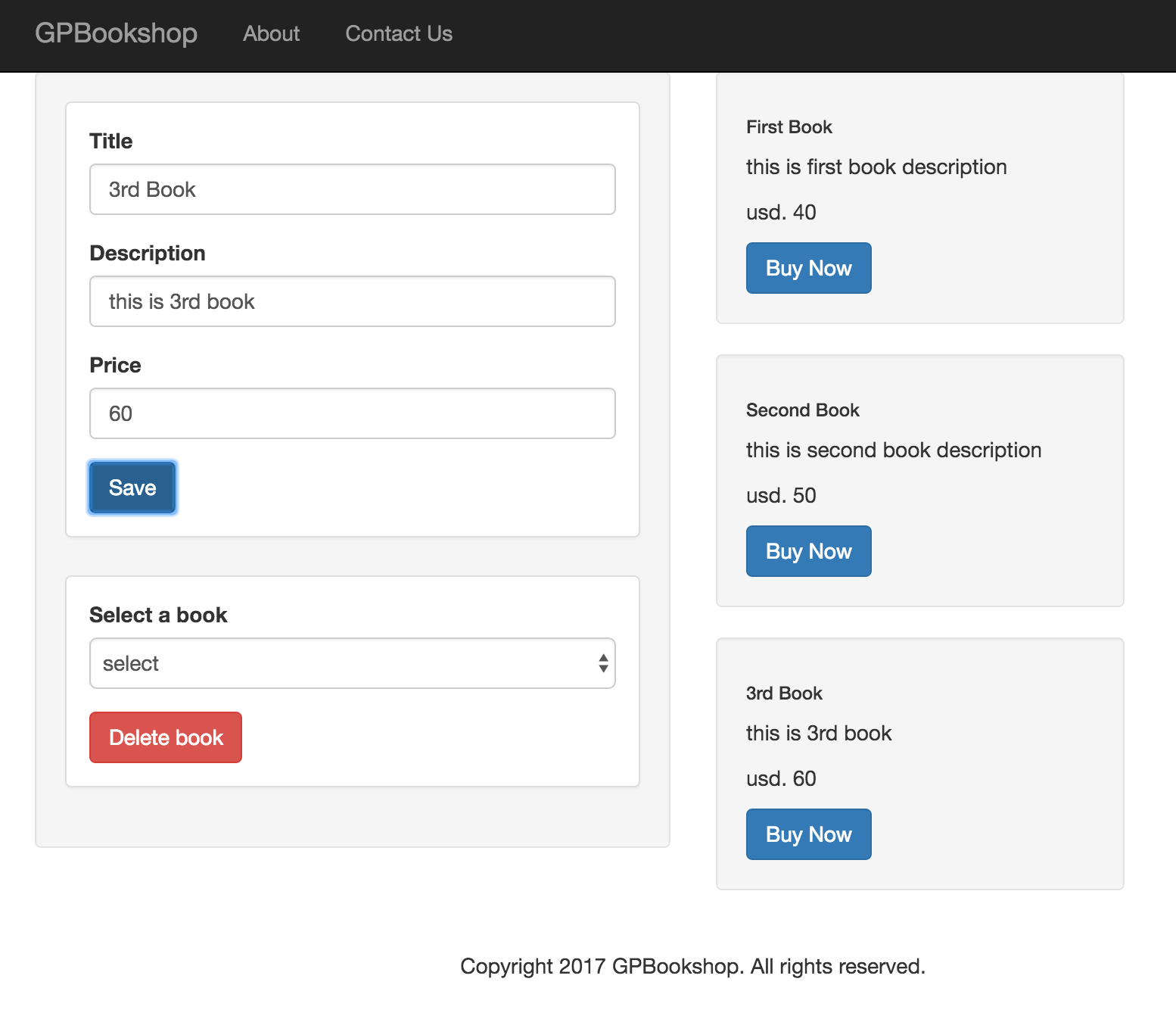


GPBookShop displays BookList component by default along with menu and footer.

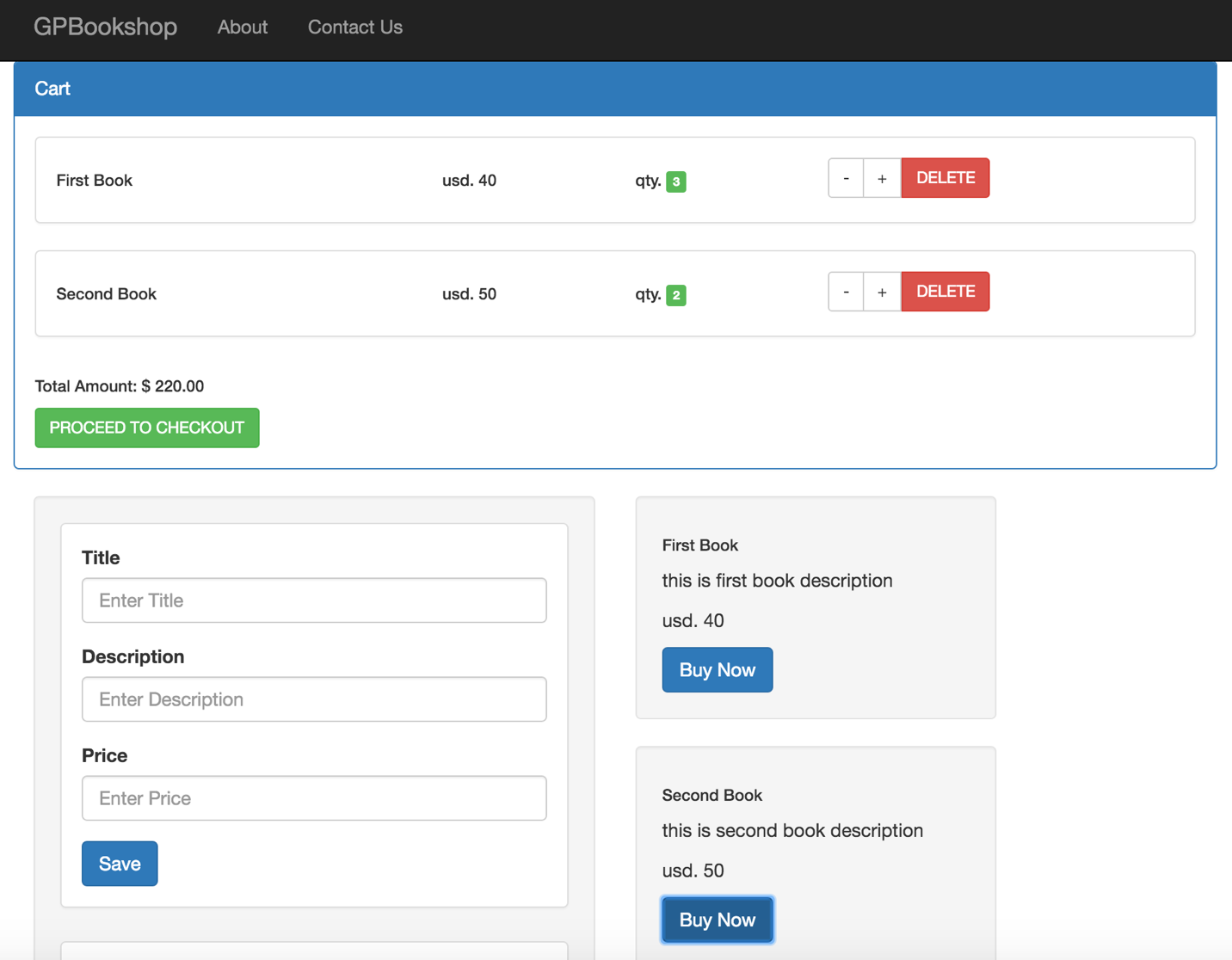
BookList component has child three components:

1. BookForm component allows to add new book to book shop and delete a selected book
2. BookItem component lists all books in store
3. Cart component displays all items in the cart that are selected for purchase

These three components access the recent state of application via redux store.



User enters new book details and clicks on save. New book is displayed in the books list



User clicks on ‘buy now’ button.

Selected book gets added to cart.

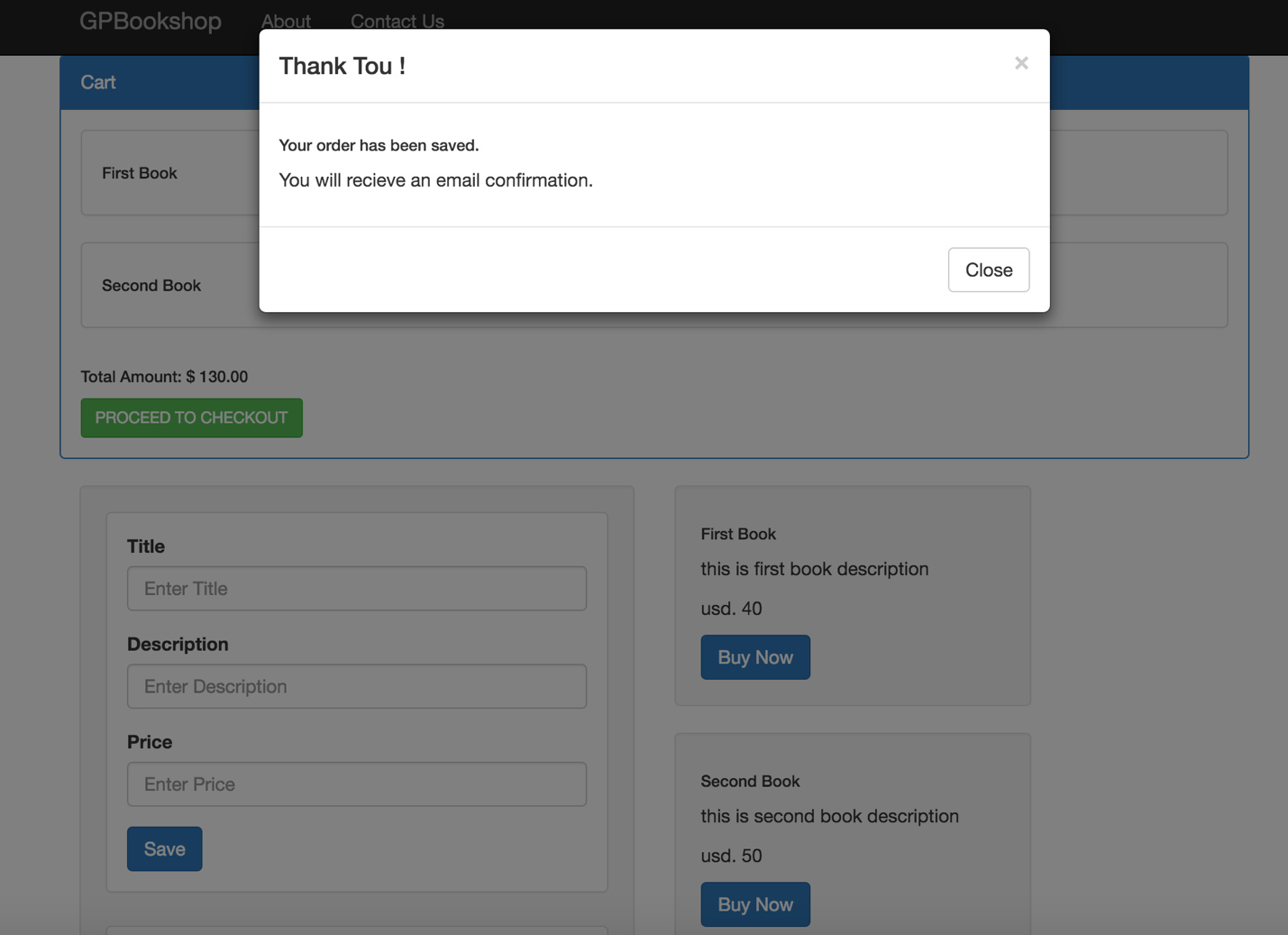
User clicks on ‘+’ button to increment the qty and ‘–‘ button to

decrement the qty of selected book.

User clicks on delete button to remove an item from cart.

Total amount is displayed depending on the number of items

selected and qty of selected item.



User clicks on proceed to checkout button.

Bootstrap Modal window is displayed with thank message.

**References:**

<http://redux.js.org/>

<https://survivejs.com/blog/redux-interview/>

(Dan Abramov – creator of Redux and React hot loader)

<https://www.sitepoint.com/>

<https://www.udemy.com/>