**Problem Statement:**

**Description**

The Kitchen Story is an e-commerce portal that lets people shop basic food items on their website.

**System Details**

The website needs to have the following features: A search form in the home page to allow entry of the food items to be purchased by the customer. Based on item details entered, it will show available food items with price. Once a person selects an item to purchase, they will be redirected to the list of available items. In the next page, they are shown the complete breakout of the order and details of the payment to be made in the payment gateway. When payment is done, they are shown a confirmation page with details of the order.

**Angular:**

Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps.

Angular is a development platform, built on [TypeScript](https://www.typescriptlang.org/). As a platform, Angular includes:

* A component-based framework for building scalable web applications
* A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
* A suite of developer tools to help you develop, build, test, and update your code

With Angular, you're taking advantage of a platform that can scale from single-developer projects to enterprise-level applications. Angular is designed to make updating as easy as possible, so you can take advantage of the latest developments with a minimum of effort. Best of all, the Angular ecosystem consists of a diverse group of over 1.7 million developers, library authors, and content creators.

**Angular CLI:**

The Angular CLI is a command-line interface tool that you use to initialize, develop, scaffold, and maintain Angular applications directly from a command shell.

npm install -g @angular/cli

**Components:**

Components are the building blocks that compose an application. A component includes a TypeScript class with a @[Component](https://angular.io/api/core/Component)() decorator, an HTML template, and styles. The @[Component](https://angular.io/api/core/Component)() decorator specifies the following Angular-specific information:

* A CSS selector that defines how the component is used in a template. HTML elements in your template that match this selector become instances of the component.
* An HTML template that instructs Angular how to render the component.
* An optional set of CSS styles that define the appearance of the template's HTML elements.

The following is a minimal Angular component.

import { Component } from '@angular/core';

@Component({

selector: 'hello-world',

template: `

<h2>Hello World</h2>

<p>This is my first component!</p>

`

})

export class HelloWorldComponent {

// The code in this class drives the component's behavior.

}

To use this component, you write the following in a template:

<hello-world></hello-world>

When Angular renders this component, the resulting DOM looks like this:

<hello-world> <h2>Hello World</h2> <p>This is my first component!</p></hello-world>

**Templates:**

Every component has an HTML template that declares how that component renders. You define this template either inline or by file path.

Angular extends HTML with additional syntax that lets you insert dynamic values from your component. Angular automatically updates the rendered DOM when your component’s state changes. One application of this feature is inserting dynamic text, as shown in the following example.

<p>{{ message }}</p>

The value for message comes from the component class:

import { Component } from '@angular/core';

@Component ({

selector: 'hello-world-interpolation',

templateUrl: './hello-world-interpolation.component.html'

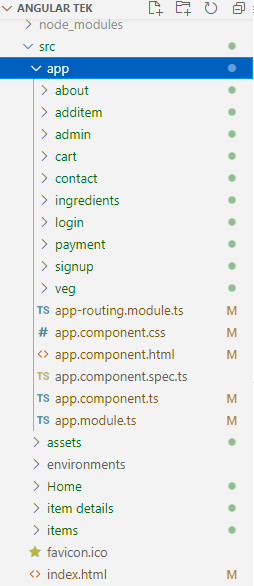
})

export class HelloWorldInterpolationComponent {

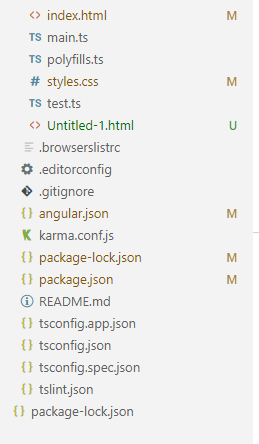
message = 'Hello, World!';

}

**Visual Studio Code Project Files:**



**Figure 1. Files of Visual Studio Code**



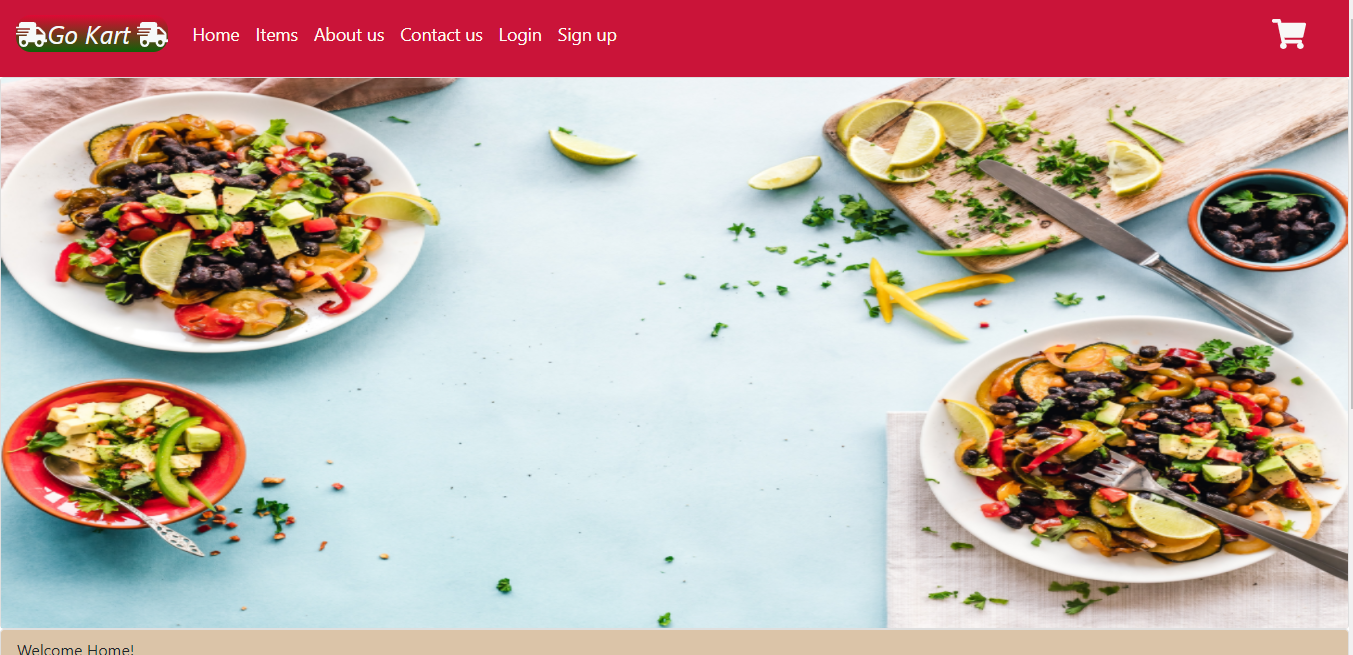
**Figure 2. Files of Visual Studio Code**

**Results and Observations**



**Figure 3. Navigation bar**

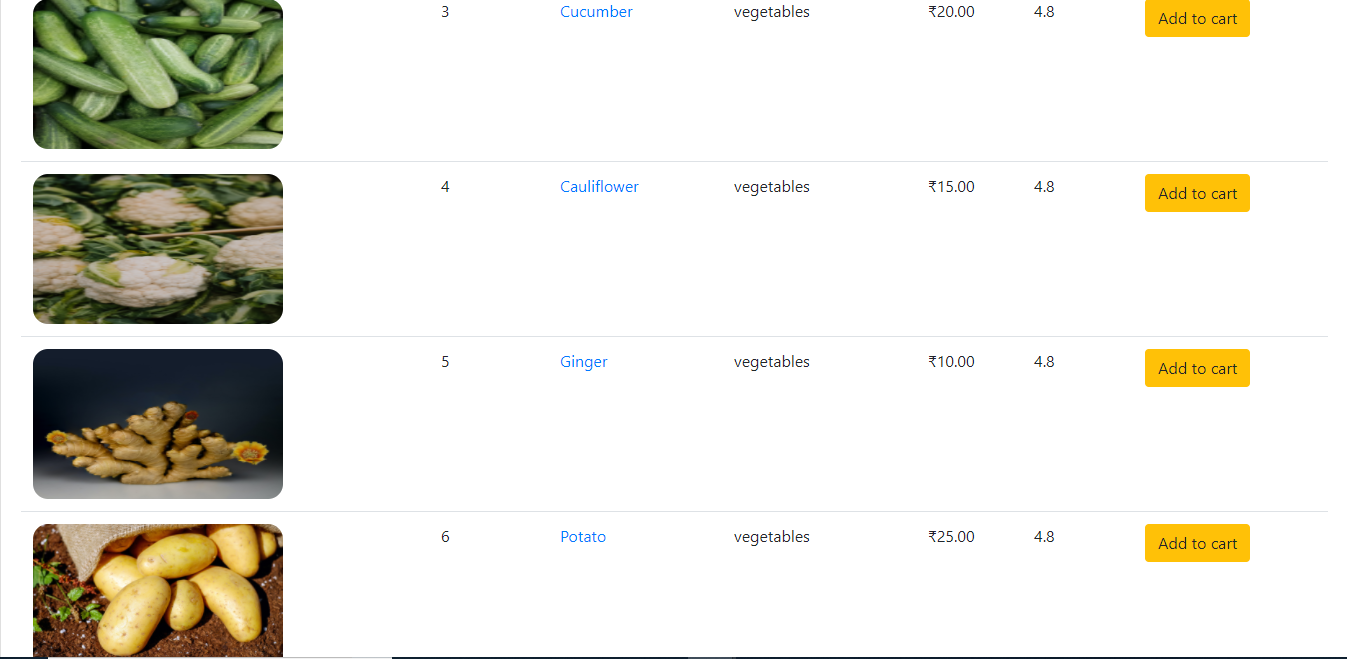
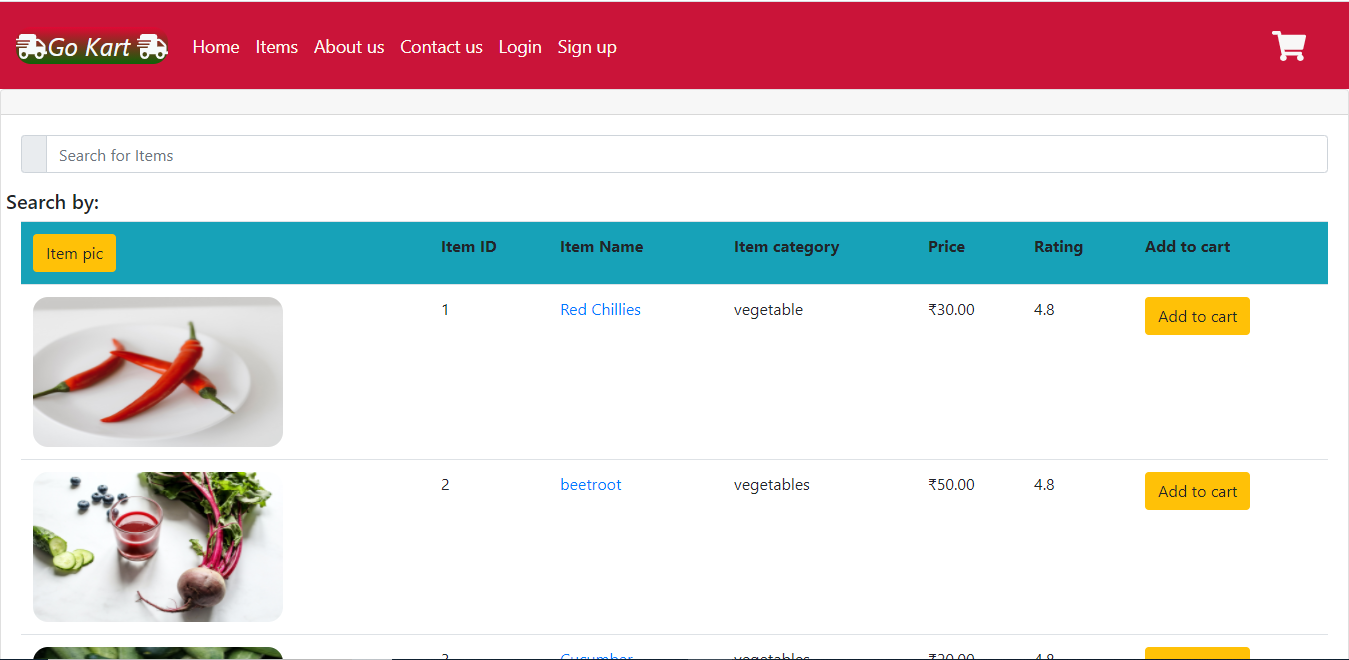
The above diagram describes the application navigation bar.



****

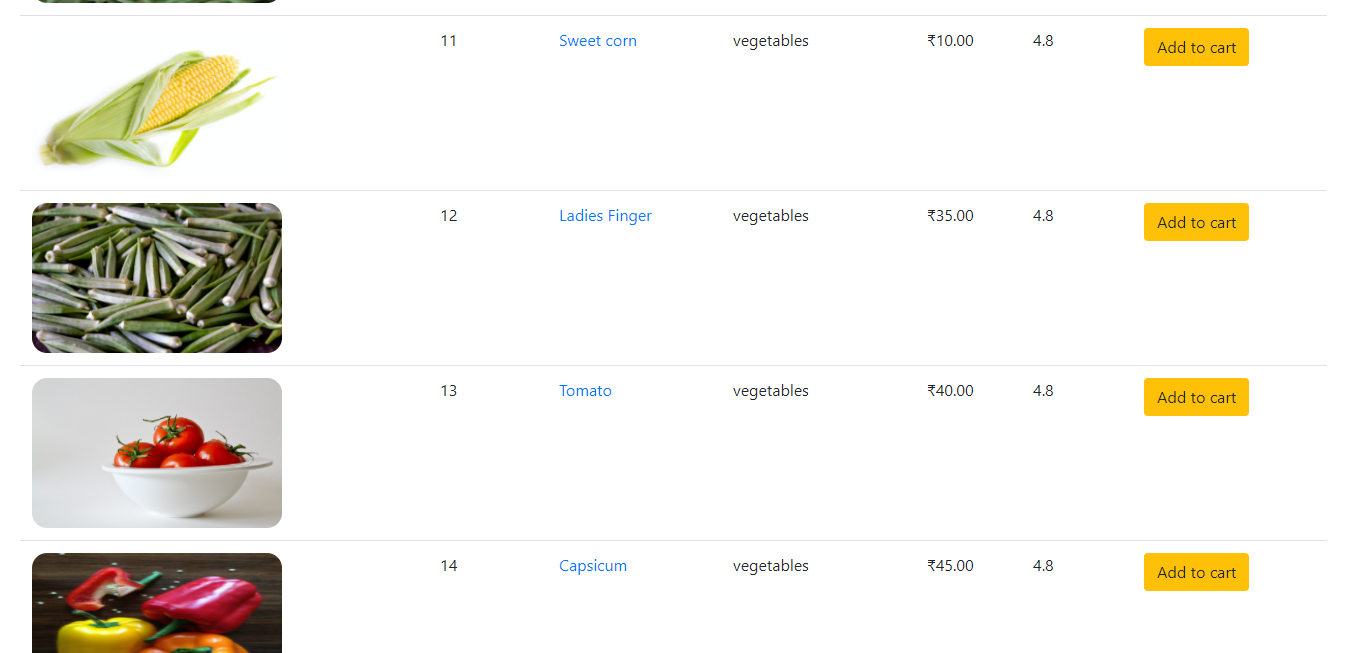
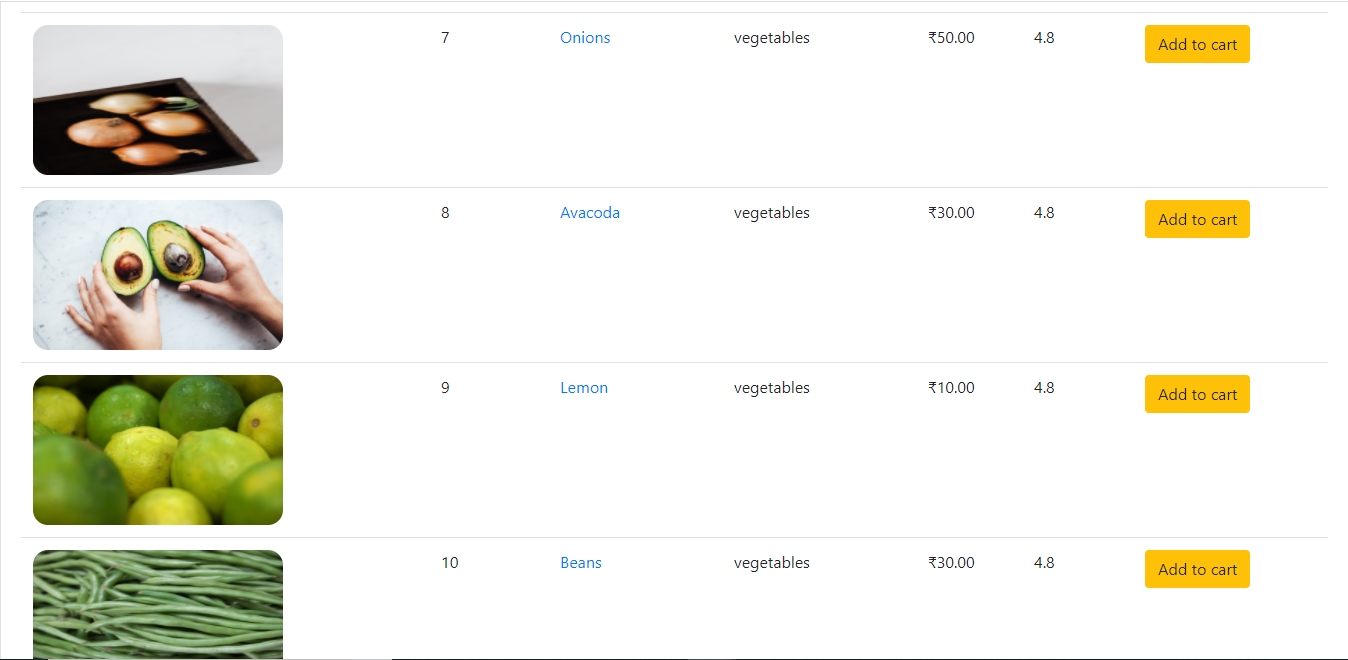
**Figure 4. Home Page**

The above figure describes the home page with logo and content.



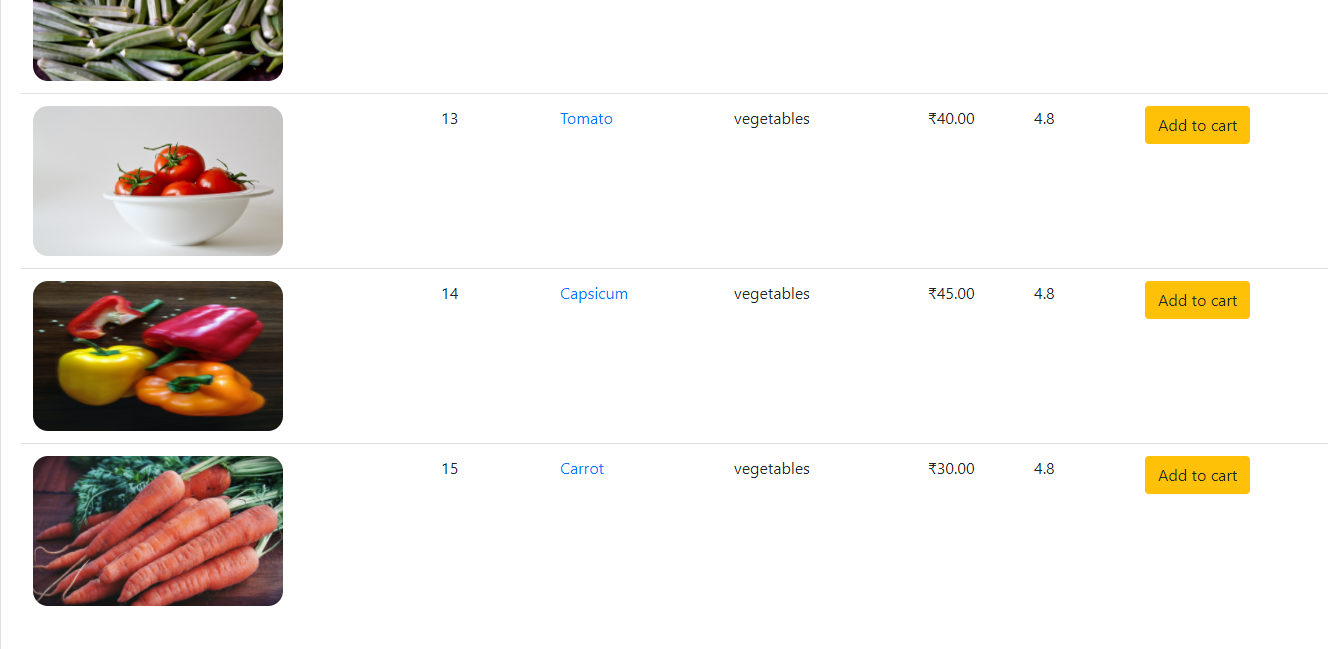
**Figure 5. Items Page**

The above figure represents the items in selling items tab.



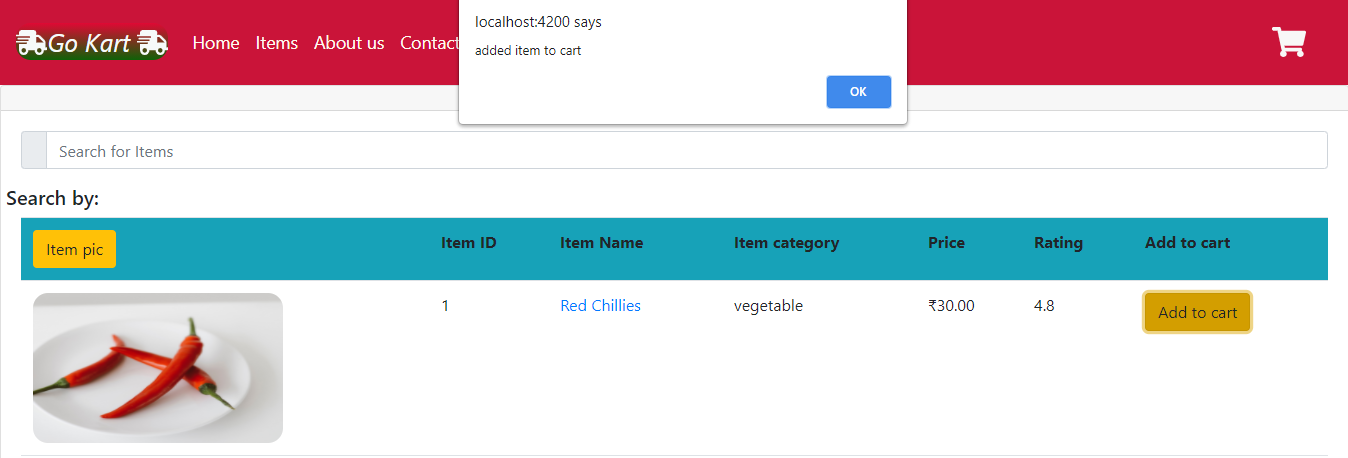
**Figure 6. Items Page**

The above figure represents the items in selling items tab.



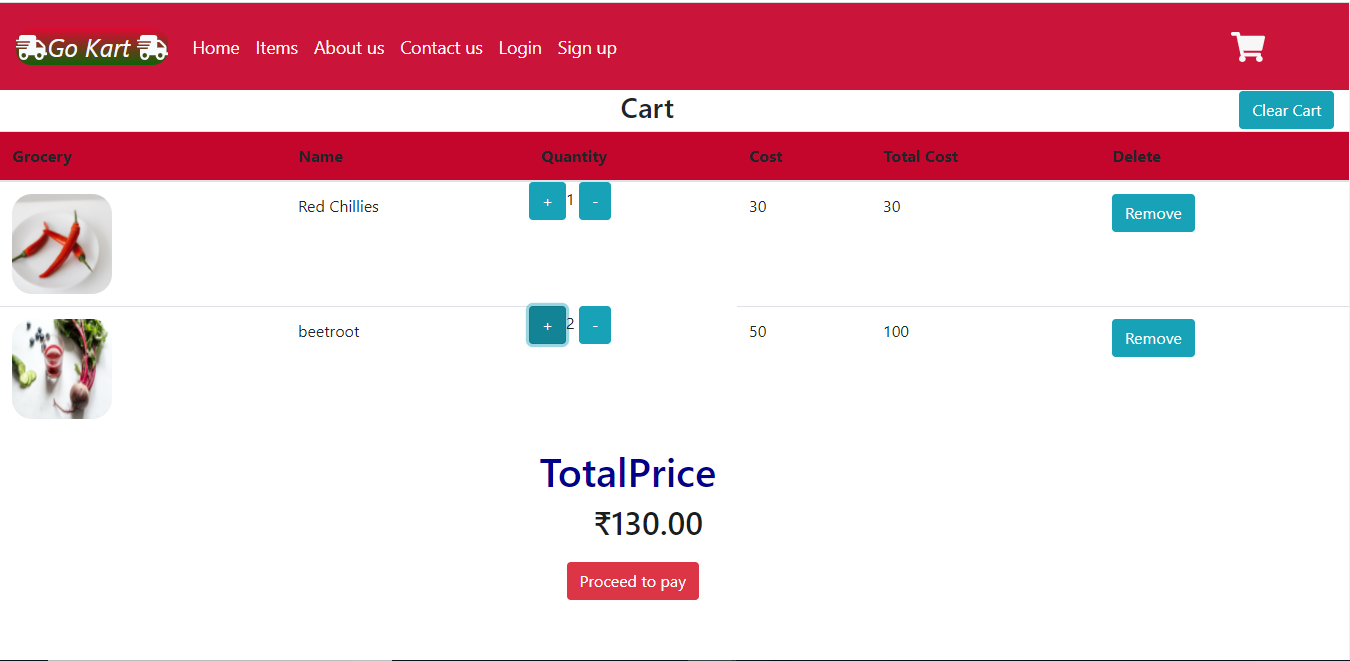
**Figure 7. Items page**

The above figure represents the items in selling items tab.

****

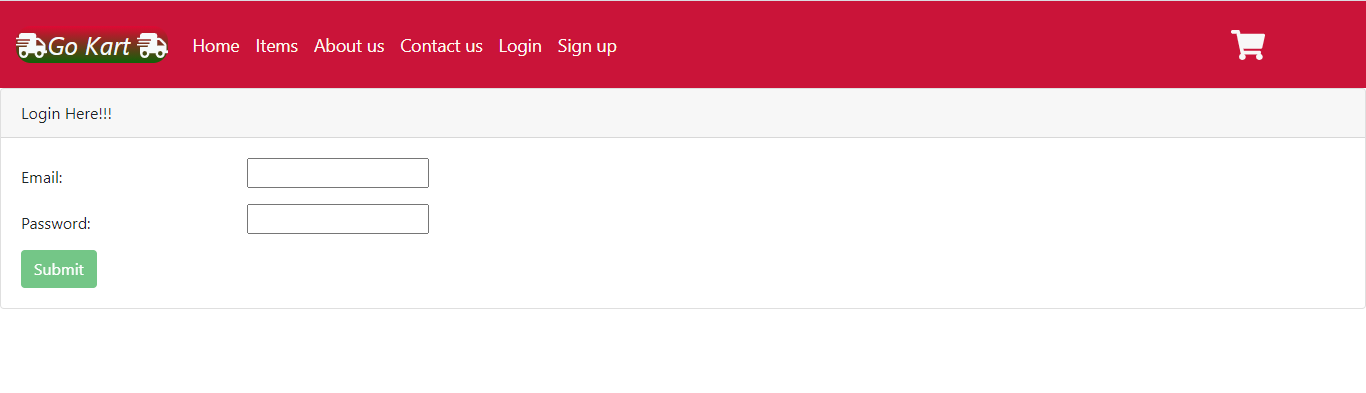
**Figure 8. Add to cart**

The above figure represents the pop up window that comes after clicking on add to cart.

****

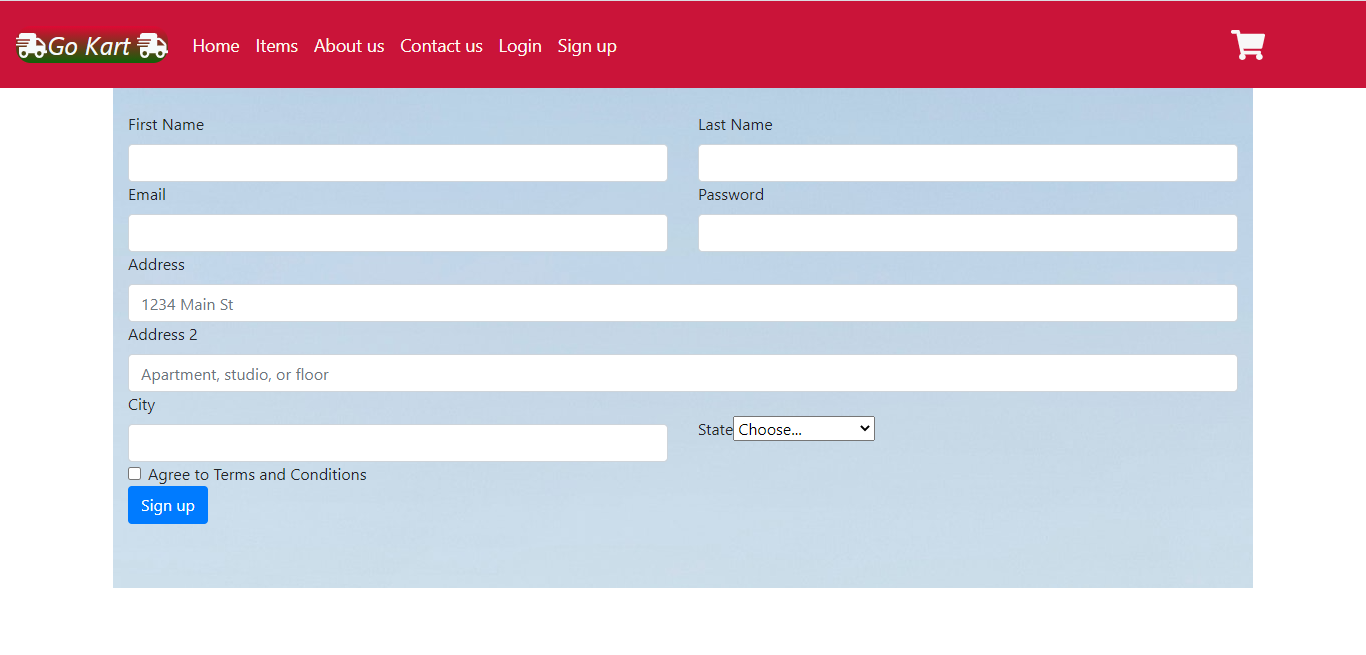
**Figure 9. Cart**

The above figure represents the cart items and total price in the cart.

****

**Figure 10. Login page**

The above figure represents the login page.



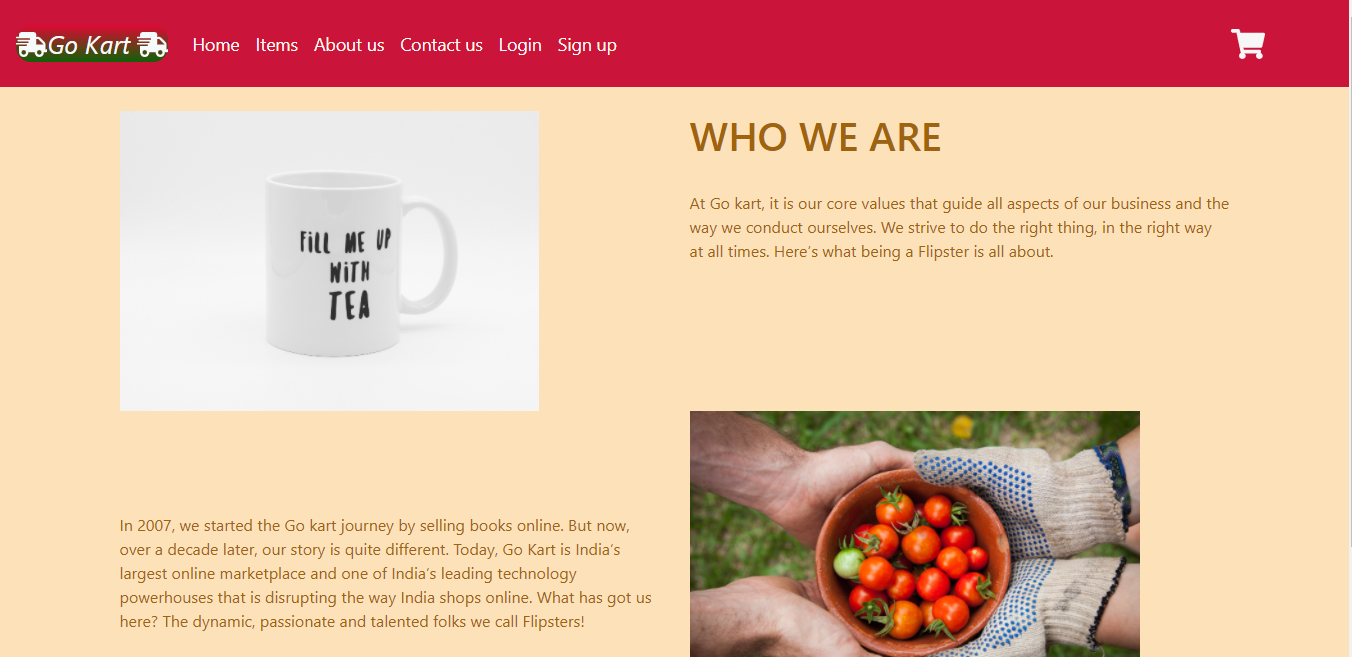
**Figure 11. Sign up page**

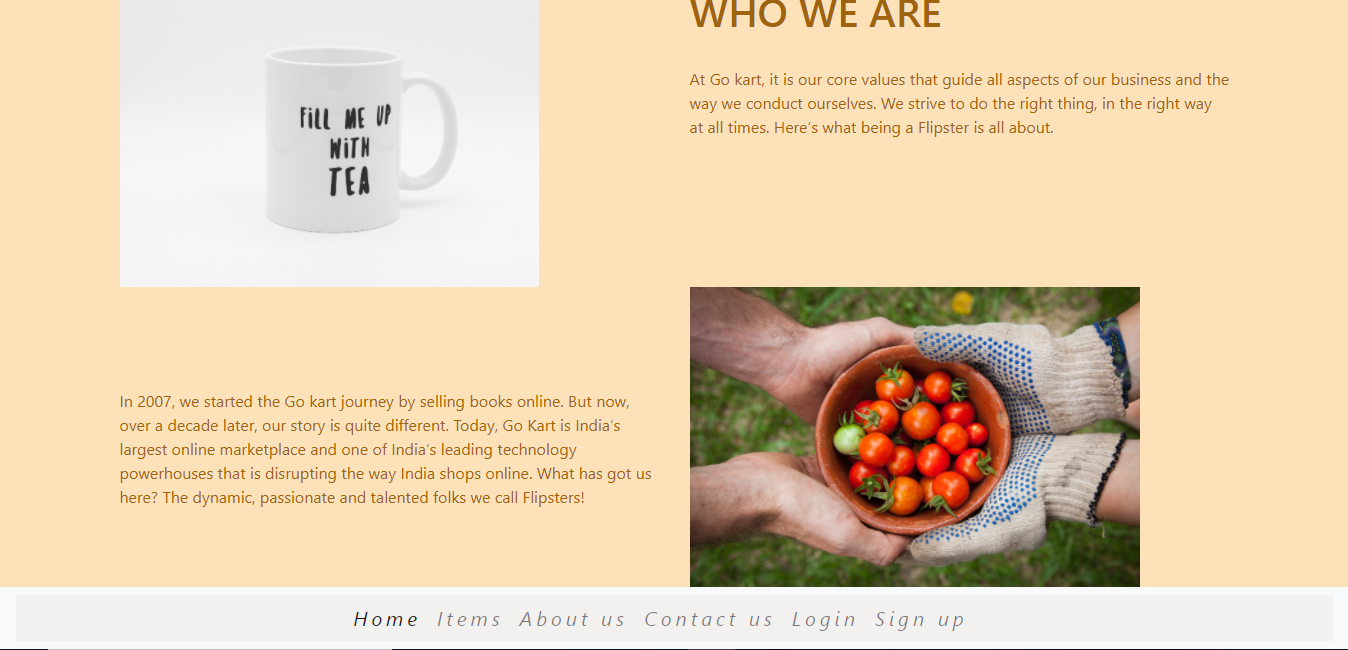
The above figure represents the sign up page.



**Figure 12. Payment page**

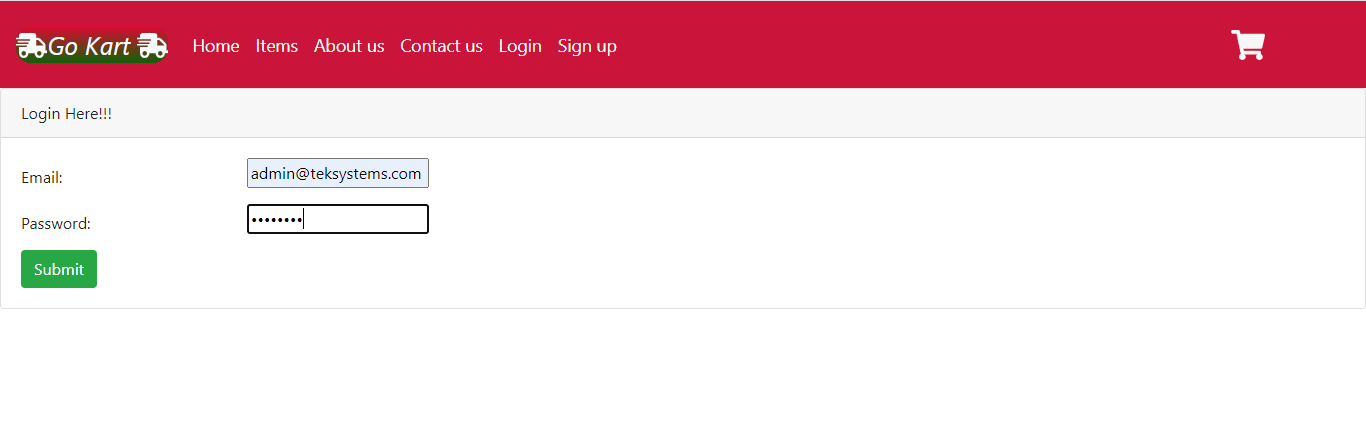
The above figure represents the payment success page.



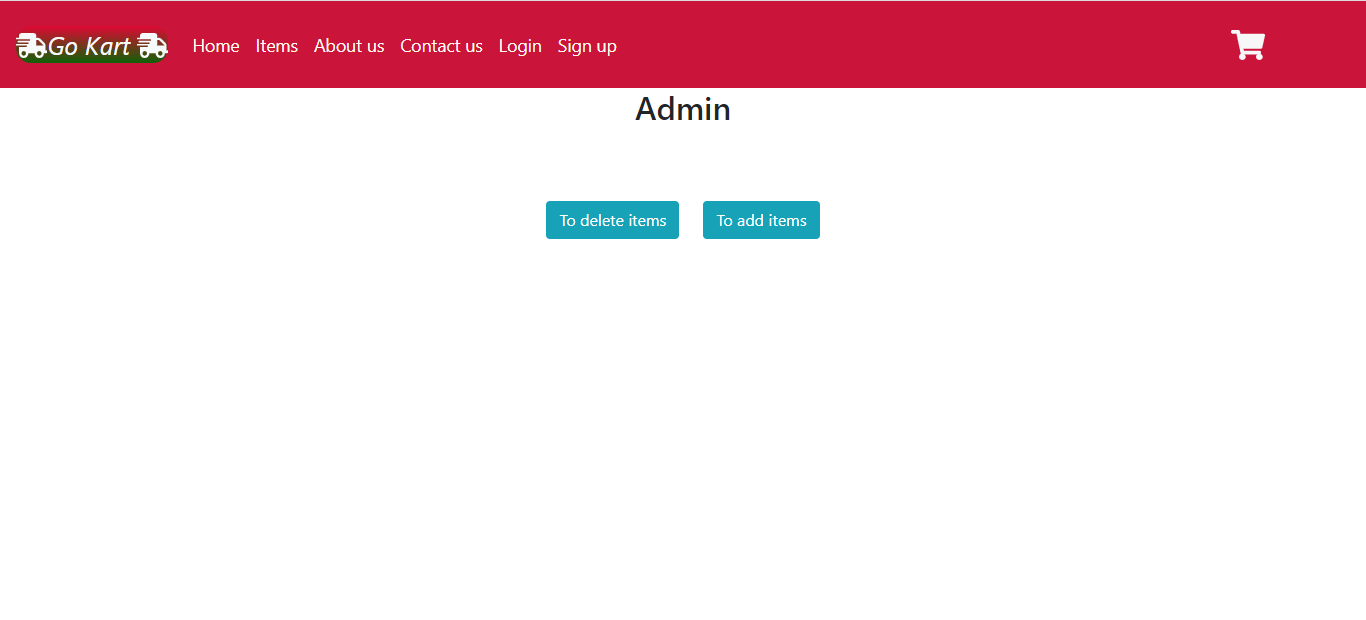


**Figure 13. About us page**

The above figure describes the about us page

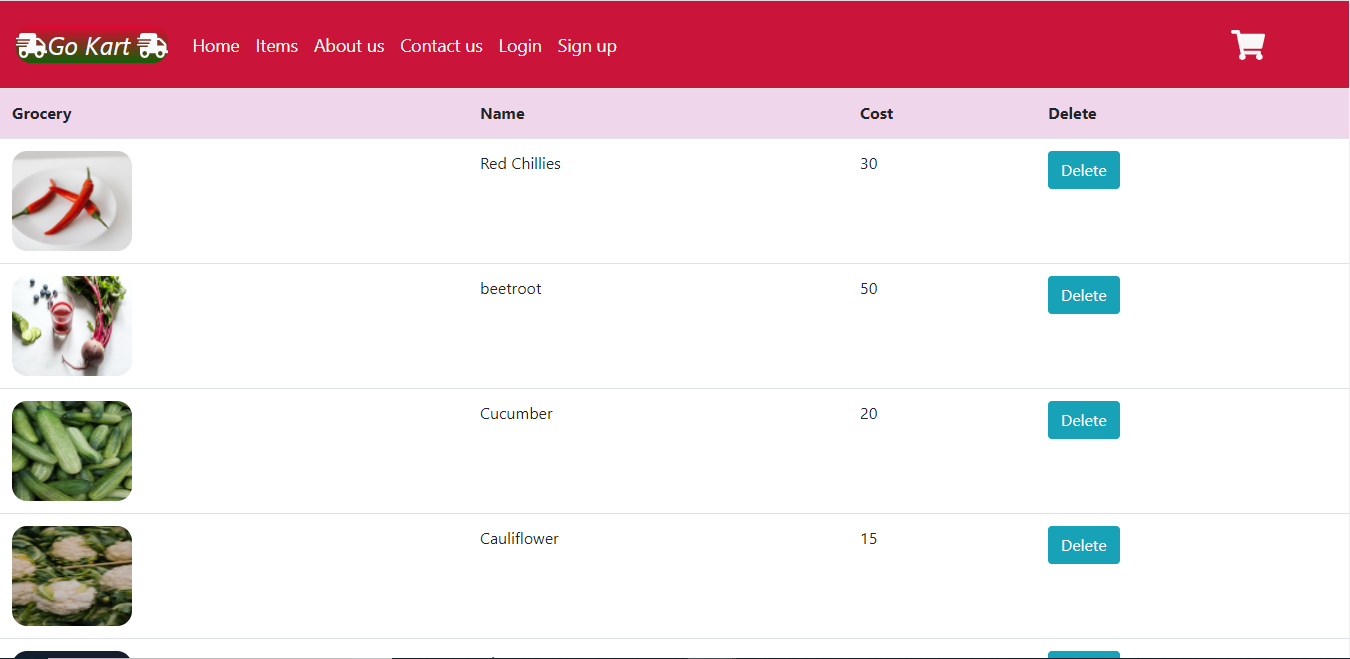
**Figure 14. Admin Login**

The above figure describes the admin login page



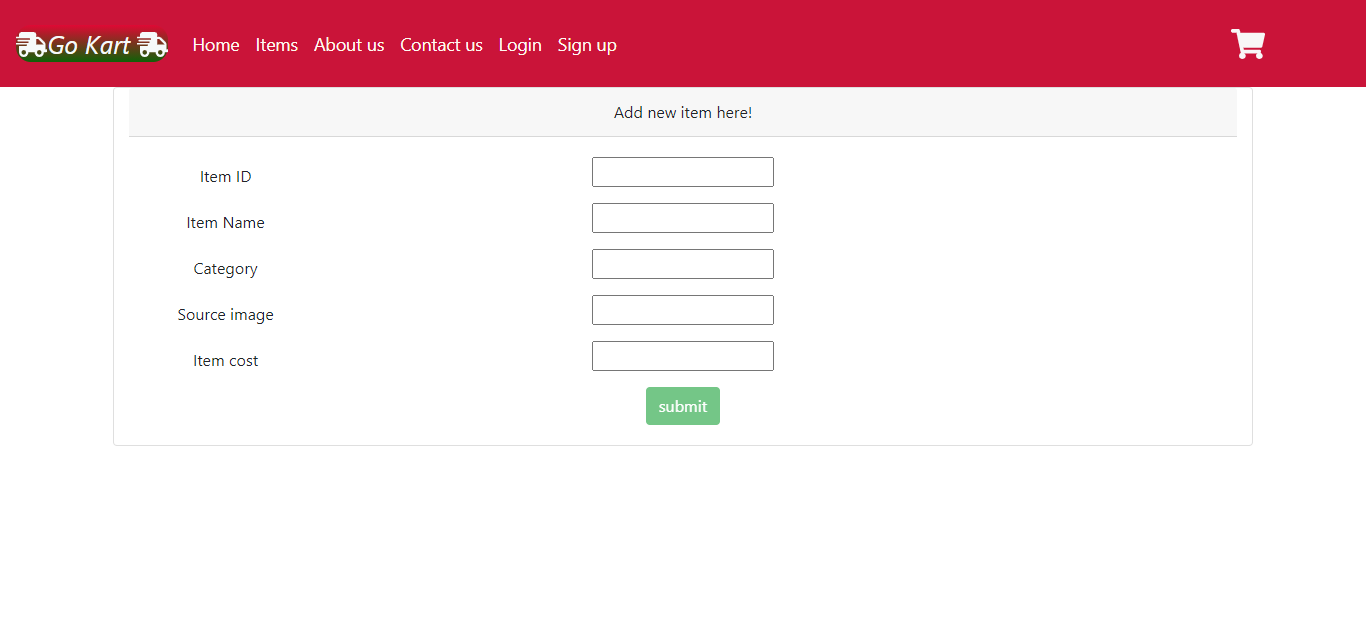
**Figure 15. Admin page**

The above figure describes the admin page where admin can select to delete or add items



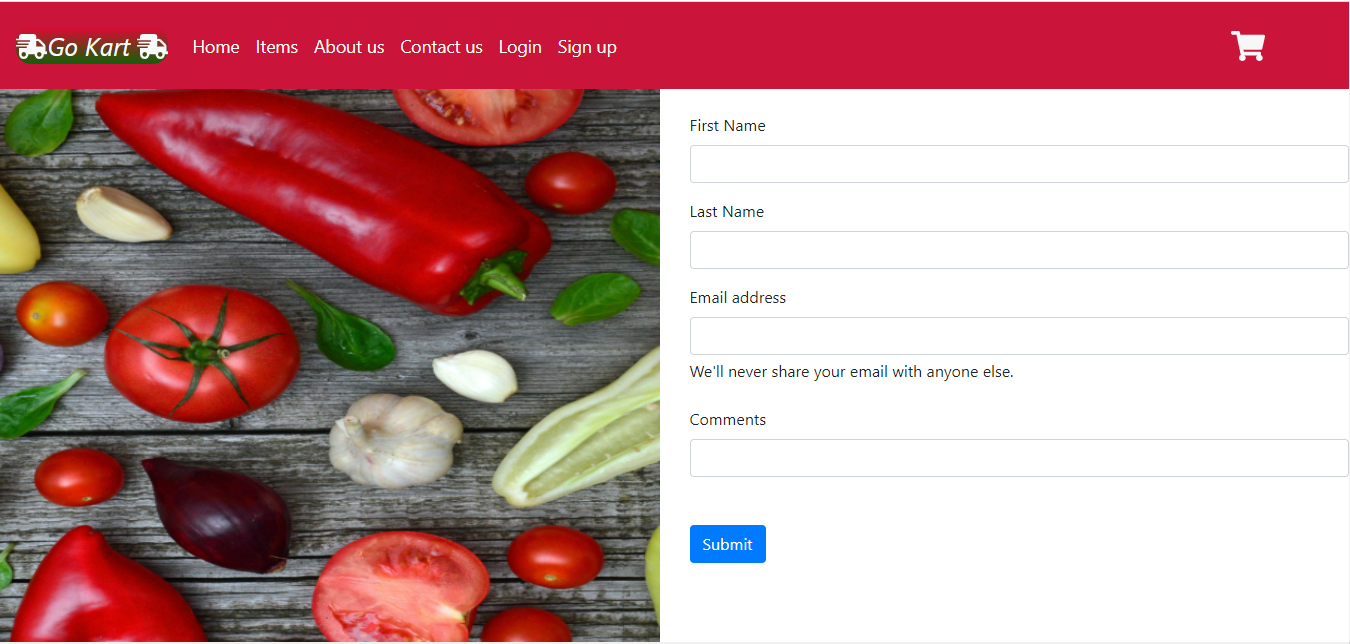
**Figure 16. Delete items page**

The above figure describes the deletion page



**Figure 17. Add items**

The above figure describes the add items page of admin where admin can add items to be sold



**Figure 18. Contact us page**

The above figure describes the contact us page to leave a message to admin or post a question