Project bhargav  
i want a frontend webpage that contains eccomerce image on the top in middle bhargav eccomerce store and under that 3 buttons with names shoes,watches and paintings button if possible add shoe image at top of button and painting image on top of paintings button and watch image on top of watch button

next when i click on shoes button i should get welcome to shoes selling page which consists of name of the shoe then user can enter the name then selling price of the shoe then user should enter the selling price then their should be a add button to add this user data and that data should be visible under lists and their should be a delete button to delete this data simillarly do for paintings and watches

so i want entire proper working code with database connection

like frontend code backend code and database code

Certainly! I will provide you a complete full-stack solution using the following technology stack for an e-commerce frontend with backend and database:

* Frontend: React (JavaScript)
* Backend: Node.js with Express
* Database: SQLite (simple file-based database)

Features:

* Top centered e-commerce image/title.
* 3 buttons with images: Shoes, Watches, Paintings.
* Clicking a button shows the respective selling page.
* Input fields: Name of item, Selling price.
* Add button stores data in database.
* List displays added items with delete button.
* Fully working backend APIs to add/delete/list items from DB.

Folder structure:

copy

javascript

1ecommerce-store/

2 ├─ backend/

3 └─ frontend/

Step 1: Backend code (backend/index.js)

copy

js

1const express = require('express');

2const cors = require('cors');

3const sqlite3 = require('sqlite3').verbose();

4const path = require('path');

5

6const app = express();

7const PORT = 4000;

8

9app.use(cors());

10app.use(express.json());

11

12// Initialize SQLite database

13const db = new sqlite3.Database('./ecommerce.db', (err) => {

14 if (err) {

15 console.error('Error opening database:', err.message);

16 } else {

17 console.log('Connected to SQLite database.');

18 }

19});

20

21// Create tables for shoes, watches, paintings

22const categories = ['shoes', 'watches', 'paintings'];

23

24categories.forEach(cat => {

25 db.run(

26 `CREATE TABLE IF NOT EXISTS ${cat} (

27 id INTEGER PRIMARY KEY AUTOINCREMENT,

28 name TEXT NOT NULL,

29 price REAL NOT NULL

30 )`,

31 (err) => {

32 if (err) {

33 console.error(`Error creating ${cat} table:`, err.message);

34 }

35 }

36 );

37});

38

39// API to get list of items by category

40app.get('/api/:category', (req, res) => {

41 const category = req.params.category.toLowerCase();

42 if(!categories.includes(category)) {

43 return res.status(400).json({error: 'Invalid category'});

44 }

45

46 db.all(`SELECT \* FROM ${category}`, [], (err, rows) => {

47 if(err) {

48 return res.status(500).json({error: err.message});

49 }

50 res.json(rows);

51 });

52});

53

54// API to add an item to a category

55app.post('/api/:category', (req, res) => {

56 const category = req.params.category.toLowerCase();

57 if(!categories.includes(category)) {

58 return res.status(400).json({error: 'Invalid category'});

59 }

60

61 const { name, price } = req.body;

62 if(!name || !price) {

63 return res.status(400).json({error: 'Name and price are required'});

64 }

65

66 const stmt = db.prepare(`INSERT INTO ${category} (name, price) VALUES (?, ?)`);

67 stmt.run(name, price, function(err) {

68 if(err) {

69 return res.status(500).json({error: err.message});

70 }

71 res.json({id: this.lastID, name, price});

72 });

73 stmt.finalize();

74});

75

76// API to delete an item by id in category

77app.delete('/api/:category/:id', (req, res) => {

78 const category = req.params.category.toLowerCase();

79 const id = req.params.id;

80

81 if(!categories.includes(category)) {

82 return res.status(400).json({error: 'Invalid category'});

83 }

84 if(!id) {

85 return res.status(400).json({error: 'ID is required'});

86 }

87

88 const stmt = db.prepare(`DELETE FROM ${category} WHERE id = ?`);

89 stmt.run(id, function(err) {

90 if(err) {

91 return res.status(500).json({error: err.message});

92 }

93 if(this.changes === 0) {

94 return res.status(404).json({error: 'Item not found'});

95 }

96 res.json({message: 'Item deleted successfully'});

97 });

98 stmt.finalize();

99});

100

101app.listen(PORT, () => {

102 console.log(`Server running on http://localhost:${PORT}`);

103});

copy

Save this as backend/index.js  
Run npm init -y and install dependencies:  
npm install express cors sqlite3

Step 2: Frontend code (frontend/src/App.js)

copy

jsx

1import React, { useState, useEffect } from 'react';

2

3const BASE\_URL = 'http://localhost:4000/api';

4

5const categoryInfo = {

6 shoes: {

7 title: 'Welcome to Shoes Selling Page',

8 buttonText: 'Add Shoe',

9 image:

10 'https://cdn-icons-png.flaticon.com/512/6797/6797594.png', // shoe icon

11 },

12 watches: {

13 title: 'Welcome to Watches Selling Page',

14 buttonText: 'Add Watch',

15 image:

16 'https://cdn-icons-png.flaticon.com/512/1077/1077164.png', // watch icon

17 },

18 paintings: {

19 title: 'Welcome to Paintings Selling Page',

20 buttonText: 'Add Painting',

21 image:

22 'https://cdn-icons-png.flaticon.com/512/742/742922.png', // painting icon

23 },

24};

25

26function App() {

27 const [category, setCategory] = useState(null);

28 const [name, setName] = useState('');

29 const [price, setPrice] = useState('');

30 const [items, setItems] = useState([]);

31

32 // Load items when category changes

33 useEffect(() => {

34 if (!category) return;

35 fetch(`${BASE\_URL}/${category}`)

36 .then(res => res.json())

37 .then(data => setItems(data))

38 .catch(err => alert('Error fetching data: ' + err.message));

39 }, [category]);

40

41 function handleAdd() {

42 if (!name.trim() || !price.trim() || isNaN(price)) {

43 alert('Please enter valid name and numeric price.');

44 return;

45 }

46 const payload = { name: name.trim(), price: parseFloat(price) };

47 fetch(`${BASE\_URL}/${category}`, {

48 method: 'POST',

49 headers: { 'Content-Type': 'application/json' },

50 body: JSON.stringify(payload),

51 })

52 .then(res => {

53 if (!res.ok) throw new Error('Failed to add item');

54 return res.json();

55 })

56 .then(newItem => {

57 setItems(prev => [...prev, newItem]);

58 setName('');

59 setPrice('');

60 })

61 .catch(err => alert('Error adding item: ' + err.message));

62 }

63

64 function handleDelete(id) {

65 fetch(`${BASE\_URL}/${category}/${id}`, { method: 'DELETE' })

66 .then(res => {

67 if (!res.ok) throw new Error('Failed to delete item');

68 return res.json();

69 })

70 .then(() => {

71 setItems(prev => prev.filter(item => item.id !== id));

72 })

73 .catch(err => alert('Error deleting item: ' + err.message));

74 }

75

76 return (

77 <div style={{ maxWidth: 800, margin: 'auto', fontFamily: 'Arial, sans-serif' }}>

78 {/\* Header \*/}

79 <header style={{ textAlign: 'center', margin: '20px 0' }}>

80 <img

81 src="https://cdn-icons-png.flaticon.com/512/263/263142.png"

82 alt="Ecommerce"

83 style={{ width: 120, height: 120, objectFit: 'contain' }}

84 />

85 <h1>Bhargav Ecommerce Store</h1>

86 </header>

87

88 {/\* Category buttons \*/}

89 <div

90 style={{

91 display: 'flex',

92 justifyContent: 'center',

93 gap: 40,

94 marginBottom: 30,

95 }}

96 >

97 {['shoes', 'watches', 'paintings'].map(cat => (

98 <button

99 key={cat}

100 onClick={() => setCategory(cat)}

101 style={{

102 cursor: 'pointer',

103 border:

104 category === cat ? '3px solid #007bff' : '2px solid #ccc',

105 borderRadius: 10,

106 padding: 10,

107 width: 160,

108 backgroundColor: category === cat ? '#e7f1ff' : '#fff',

109 display: 'flex',

110 flexDirection: 'column',

111 alignItems: 'center',

112 fontWeight: 'bold',

113 fontSize: 16,

114 }}

115 >

116 <img

117 src={categoryInfo[cat].image}

118 alt={cat}

119 style={{ width: 80, height: 80, marginBottom: 5 }}

120 />

121 {cat.charAt(0).toUpperCase() + cat.slice(1)}

122 </button>

123 ))}

124 </div>

125

126 {/\* Selling Page \*/}

127 {category && (

128 <section

129 style={{

130 borderTop: '2px solid #ddd',

131 paddingTop: 20,

132 }}

133 >

134 <h2>{categoryInfo[category].title}</h2>

135 <div style={{ marginBottom: 20 }}>

136 <label>

137 {category.charAt(0).toUpperCase() + category.slice(1)} Name:{' '}

138 <input

139 type="text"

140 value={name}

141 placeholder="Enter name"

142 onChange={e => setName(e.target.value)}

143 style={{ marginLeft: 10, padding: 5, width: 300 }}

144 />

145 </label>

146 </div>

147 <div style={{ marginBottom: 20 }}>

148 <label>

149 Selling Price:{' '}

150 <input

151 type="number"

152 value={price}

153 placeholder="Enter price"

154 onChange={e => setPrice(e.target.value)}

155 style={{ marginLeft: 10, padding: 5, width: 300 }}

156 />

157 </label>

158 </div>

159 <button

160 onClick={handleAdd}

161 style={{

162 padding: '8px 20px',

163 backgroundColor: '#007bff',

164 border: 'none',

165 color: 'white',

166 cursor: 'pointer',

167 borderRadius: 5,

168 fontSize: 16,

169 }}

170 >

171 {categoryInfo[category].buttonText}

172 </button>

173

174 {/\* List of items \*/}

175 <h3 style={{ marginTop: 30 }}>Listing Items:</h3>

176 {items.length === 0 ? (

177 <p>No items added yet.</p>

178 ) : (

179 <table

180 style={{

181 borderCollapse: 'collapse',

182 width: '100%',

183 marginTop: 10,

184 }}

185 >

186 <thead>

187 <tr style={{ backgroundColor: '#f2f2f2' }}>

188 <th style={{ border: '1px solid #ddd', padding: 8 }}>ID</th>

189 <th style={{ border: '1px solid #ddd', padding: 8 }}>Name</th>

190 <th style={{ border: '1px solid #ddd', padding: 8 }}>

191 Price (₹)

192 </th>

193 <th style={{ border: '1px solid #ddd', padding: 8 }}>

194 Actions

195 </th>

196 </tr>

197 </thead>

198 <tbody>

199 {items.map(item => (

200 <tr key={item.id}>

201 <td style={{ border: '1px solid #ddd', padding: 8 }}>

202 {item.id}

203 </td>

204 <td style={{ border: '1px solid #ddd', padding: 8 }}>

205 {item.name}

206 </td>

207 <td style={{ border: '1px solid #ddd', padding: 8 }}>

208 {item.price.toFixed(2)}

209 </td>

210 <td style={{ border: '1px solid #ddd', padding: 8 }}>

211 <button

212 onClick={() => handleDelete(item.id)}

213 style={{

214 padding: '5px 10px',

215 backgroundColor: '#dc3545',

216 border: 'none',

217 color: 'white',

218 cursor: 'pointer',

219 borderRadius: 3,

220 }}

221 >

222 Delete

223 </button>

224 </td>

225 </tr>

226 ))}

227 </tbody>

228 </table>

229 )}

230 </section>

231 )}

232 </div>

233 );

234}

235

236export default App;

copy

Step 3: Create React app

copy

bash

1npx create-react-app frontend

2cd frontend

3# Replace src/App.js with above code

4# Also remove src/App.css to avoid style conflicts or keep as you want

5npm start

Step 4: Run backend:

copy

bash

1cd backend

2npm init -y

3npm install express cors sqlite3

4node index.js

How to run:

* First, start your backend: node backend/index.js (run inside backend folder)
* Then start frontend: npm start inside frontend folder created by create-react-app
* Open browser at http://localhost:3000
* Click shoes/watchings/paintings buttons
* Add items and see list update from DB
* Delete removes the item from DB and UI

Explanation:

* Backend uses SQLite for persistence with three tables: shoes, watches, paintings.
* Express APIs for GET(list), POST(add), DELETE(delete) per category.
* Frontend shows the main page and after clicking category, shows form + live list pulling data from backend.
* Images from public icon URLs displayed on buttons.
* Fully functional CRUD for small dataset per category.

MONGO\_URI=mongodb://localhost:27017/recipiedb

PORT=3000

JWT\_SECRET=myverysecuresecret