**i-SHOP**

**(Smart Shopping)**

**Abstract:**

Existing system of shopping where normally customer need to physically go to the retail shop and buy the products that customers want. Online shopping is an easy and comfortable way of shopping from a large range of products at home. It understands customer buying habits and traits to predict future trends.

**Existing System:**

Existing system of shopping where normally customer need to physically go to the retail shop and buy the products that customers want.

* Customer Needs to visit the retail store physically.
* Enquire for products availability.
* Have to wait in the queue if have lot of customers.
* Search for product in the store physically.
* Need to make a purchase requisition.

**Draw Backs:**

* Customer needs to allocated time physically to visit the store.
* Unable to find products easily.
* Unable to filter products by price range.
* Wait for billing.

**Proposed System:**

Online shopping is an easy and comfortable way of shopping from a large range of products. It allows customers to view, search and buy products from home using internet.

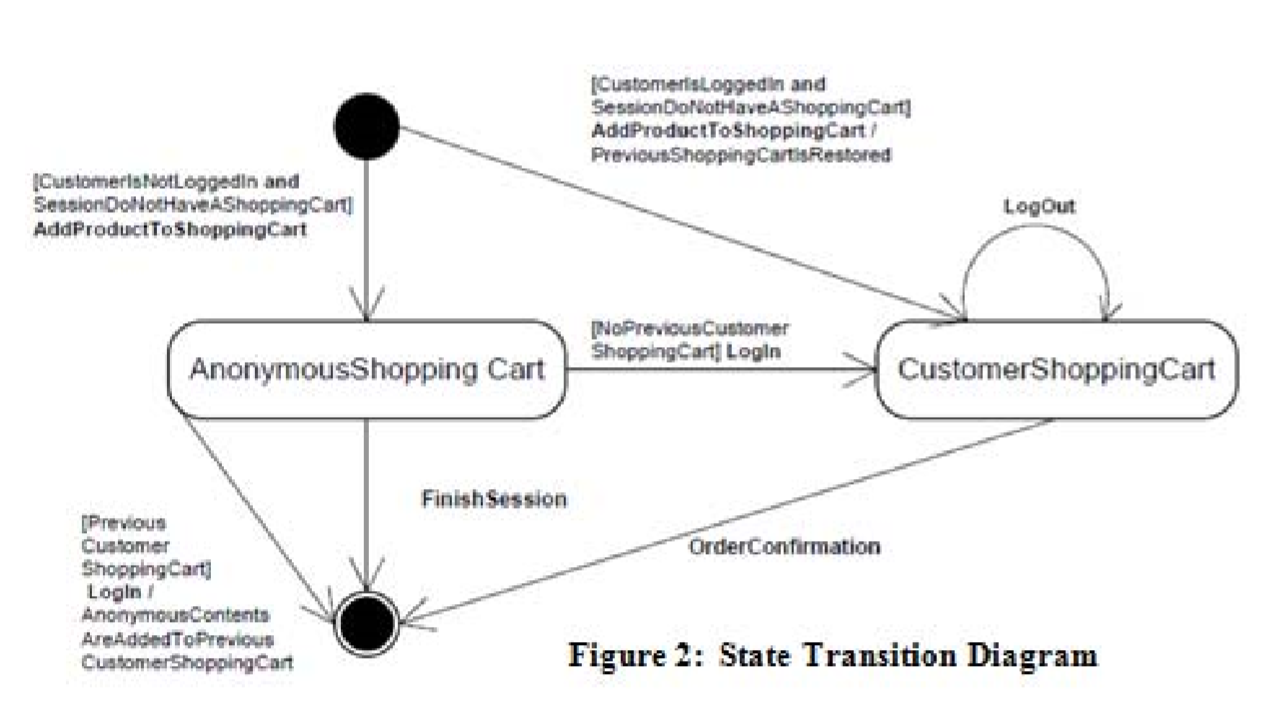
**Advantages:**

* Customer need not to visit the store physically.
* He can browse a large range of products from various vendors.
* Can make payments online, billing processes will be easy.
* It saves the purchasing time for customer.

**Modules in I-Shop**

1. **Registration Module**  
   It allows users and customers to register on the i-shop website in order to use it.
2. **Products Browse and Products Search Module**Allow the customers easy access to search the website for their preferred products.
3. **Shopping Cart Module**Has multiple currency capability to allow customers to use the website in their preferred currency.
4. **Shipping and Billing**Allows the vendor to control the amount of shipping charges.
5. **Payment**Allows the customer many methods to make payments.
6. **Admin [User, Catalog, Order] Management**Allow the admin of i-shop model to manage, control and monitor the overall website in a seamless manner.

**State Transition Diagram**



**System Configuration**

Hardware Configuration

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Hardware** | **Configurations** |
| 1 | Operating System | Windows 7 or Above, Mac, Linux |
| 2 | RAM | 2GB |
| 3 | Processor Speed | Intel Pentium IV (3.0 GHz) and Upwards |
| 4 | Hard Disk Space | 40GB |
| 5 | Monitor | 14’ CRT |

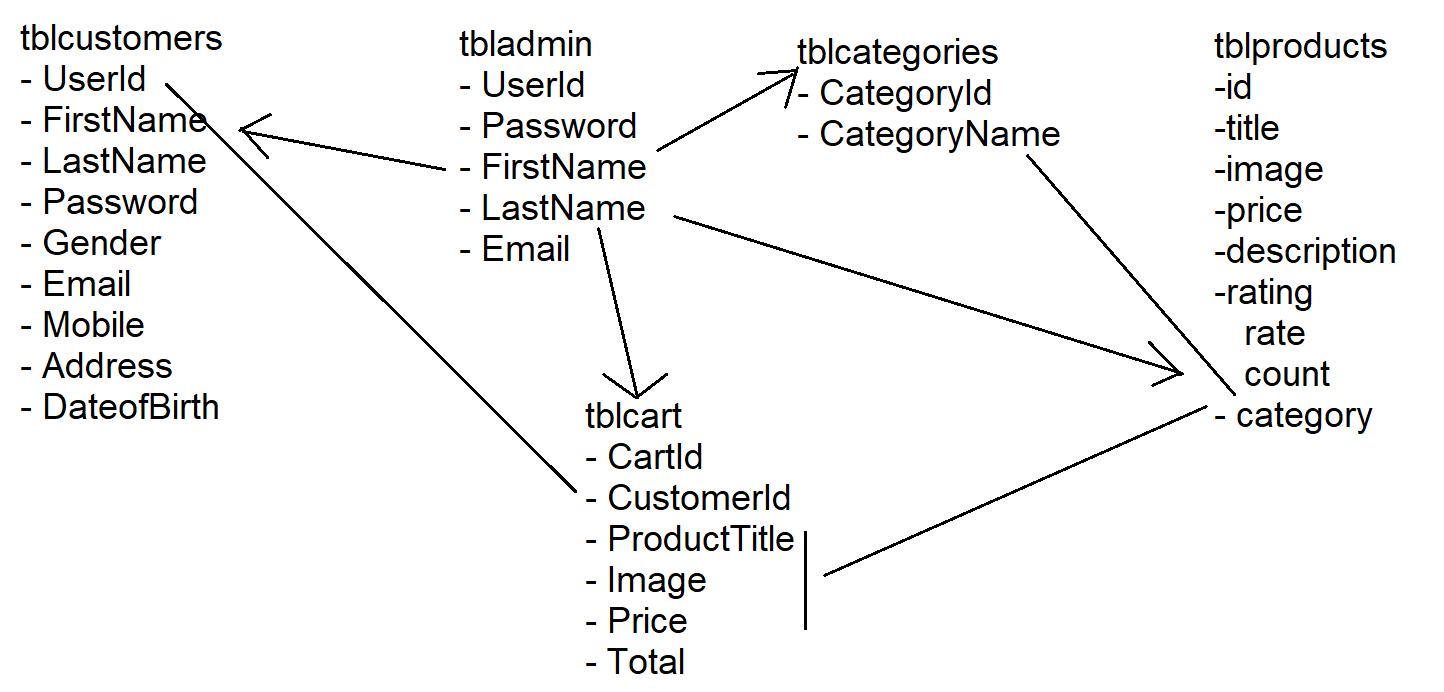
Software Configuration

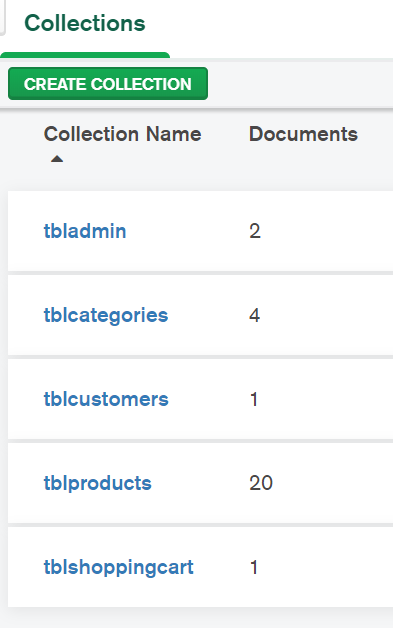
|  |  |  |
| --- | --- | --- |
| **S.NO** | **software** | **Configurations** |
| 1 | Mark Up Language | HTML |
| 2 | Style Sheets | CSS |
| 3 | Client Side | JavaScript, Jquery, React JS |
| 4 | Server Side | Node JS |
| 5 | Database | MongoDb |
| 6 | Middleware | Express |
| 7 | CMS [IDE] | Visual Studio Code |
| 8 | Web Server | Internet Information Services Manager (IIS) |
| 9 | Browser | Chrome, Edge, Internet Explorer, Safari, FireFox |

**Data Models**

1. Conceptual Model  
   Specifies the modules required with data.   
   **Customer Registration and Login   
   Products and Categories  
   Admin Registration and Login  
   Shopping Cart**
2. Physical Model  
   Specifies the table name, field name and data type
3. Logical Model  
   Specifies the relationship between modules
4. Entity Model  
   Specifies the table, field, data type and relation.

**Database Name: ishopdb**





**Create API using Express and Node.js**

* Install following libraries   
  > npm install mongodb --save   
  > npm install express --save   
  > npm install cors --save
* Add a new File “index.js”

**Index.js**

const express = require('express');

const mongoClient = require('mongodb').MongoClient;

const cors = require('cors');

const connectionString = 'mongodb://127.0.0.1:27017';

const app = express();

app.use(express.urlencoded({

extended: true

}));

app.use(express.json());

app.use(cors());

app.get('/products', (req, res)=> {

mongoClient.connect(connectionString, function(err, clientObject){

if(!err){

const dbo = clientObject.db('ishopdb');

dbo.collection('tblproducts').find({}).toArray(function(err, documents){

if(!err){

res.send(documents);

}

})

}

})

});

app.get('/products/:id', (req, res)=> {

mongoClient.connect(connectionString, function(err, clientObject){

if(!err){

const dbo = clientObject.db('ishopdb');

dbo.collection('tblproducts').find({id:parseInt(req.params.id)}).toArray(function(err, documents){

if(!err){

res.send(documents);

}

})

}

});

});

app.get('/categories', (req, res)=> {

mongoClient.connect(connectionString, function(err, clientObject){

if(!err){

const dbo = clientObject.db('ishopdb');

dbo.collection('tblcategories').find({}).toArray(function(err, documents){

if(!err){

res.send(documents);

}

})

}

});

});

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

mongoClient.connect(connectionString, function(err, clientObject){

if(!err){

const dbo = clientObject.db('ishopdb');

dbo.collection('tblproducts').find({category:req.params.category}).toArray(function(err, documents){

if(!err){

res.send(documents);

}

})

}

});

});

app.get('/getcustomers', (req, res)=>{

mongoClient.connect(connectionString, function(err, clientObject){

if(!err){

const dbo = clientObject.db('ishopdb');

dbo.collection('tblcustomers').find({}).toArray(function(err, documents){

if(!err){

res.send(documents);

}

})

}

});

});

app.get('/getadmin', (req, res)=>{

mongoClient.connect(connectionString, function(err, clientObject){

if(!err){

const dbo = clientObject.db('ishopdb');

dbo.collection('tbladmin').find({}).toArray(function(err, documents){

if(!err){

res.send(documents);

}

})

}

});

});

app.post('/adminregister', (req, res)=>{

mongoClient.connect(connectionString, function(err, clientObject){

if(!err){

const dbo = clientObject.db('ishopdb');

var data = {

UserId:req.body.UserId,

FirstName:req.body.FirstName,

LastName: req.body.LastName,

Password: req.body.Password

}

dbo.collection('tbladmin').insertOne(data, function(err, result){

if(!err) {

console.log('Record Inserted');

}

})

}

});

})

app.post('/customerregister', (req, res)=>{

mongoClient.connect(connectionString, function(err, clientObject){

if(!err){

const dbo = clientObject.db('ishopdb');

var data = {

UserId: req.body.UserId,

FirstName: req.body.FirstName,

LastName: req.body.LastName,

DateOfBirth: new Date(req.body.DateOfBirth),

Email: req.body.Email,

Gender: req.body.Gender,

Address: req.body.Address,

PostalCode: req.body.PostalCode,

State:req.body.State,

Country: req.body.Country,

Mobile: req.body.Mobile,

Password: req.body.Password

}

dbo.collection('tblcustomers').insertOne(data, function(err, result){

if(!err) {

console.log('Record Inserted');

}

})

}

});

})

app.listen(8080);

console.log(`Server Listening : http://127.0.0.1:8080`);

**API Requests**

|  |  |  |
| --- | --- | --- |
| **Http Verb** | **Request** | **Description** |
| GET | /products | Returns all products data |
| GET | /products/1 | Returns specific product details by id |
| GET | /categories | Returns all categories |
| GET | /categories/electronics | Returns products belong to specific category. |
| GET | /getcustomers | Returns all customers data. |
| GET | /getadmin | Returns all admin data. |
| POST | /adminregister | Inserts data into admin table. |
| POST | /customerregister | Inserts data into customers table. |

**Front End React Application**

* **Create a new React Application**> npx create-react-app ishop-project
* **Install the following libraries**> npm install formik --save   
  > npm install yup --save   
  > npm install axios --save   
  > npm install react-router-dom --save   
  > npm install react-cookie –save   
  > npm install bootstrap --save   
  > npm install bootstrap-icons --save

**IndexComponent.js**

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

import axios from 'axios';

import { BrowserRouter as Router, Route, Link, Switch, Redirect, useParams, useHistory } from 'react-router-dom';

import { useFormik } from 'formik';

import { useCookies } from 'react-cookie';

const RegisterAdmin = () => {

const [adminUsers, setAdminUsers] = useState([]);

const [msg, setMessage] = useState('');

const history = useHistory();

useEffect(() => {

axios.get('http://127.0.0.1:8080/getadmin')

.then(res=> {

setAdminUsers(res.data);

})

}, []);

const formik = useFormik({

initialValues: {

UserId: '',

FirstName: '',

LastName: '',

Password: ''

},

onSubmit: values => {

axios.post('http://127.0.0.1:8080/adminregister', values);

alert('Registered Successfully..');

history.push('/adminlogin');

}

})

function VerifyUserId(e){

for(var user of adminUsers)

{

if(user.UserId===e.target.value) {

setMessage('User Name Taken : Try Another');

break;

} else {

setMessage('User Name Available');

}

}

}

return(

<div className="w-50">

<form onSubmit={formik.handleSubmit} >

<h3>Admin Register</h3>

<dl>

<dt>User Id</dt>

<dd>

<input type="text" onKeyUp={VerifyUserId} onChange={formik.handleChange} name="UserId" value={formik.values.UserId} />

</dd>

<dd>{msg}</dd>

<dt>First Name</dt>

<dd>

<input type="text" onChange={formik.handleChange} name="FirstName" value={formik.values.FirstName} />

</dd>

<dt>Last Name</dt>

<dd>

<input type="text" onChange={formik.handleChange} name="LastName" value={formik.values.LastName} />

</dd>

<dt>Password</dt>

<dd>

<input type="password" onChange={formik.handleChange} name="Password" value={formik.values.Password} />

</dd>

</dl>

<button className="btn btn-primary">Register</button>

<div className="mt-2">

Existing User <Link to="/adminlogin" className="btn btn-link"> Login</Link>

</div>

</form>

</div>

)

}

const LoginComponent = () => {

const [users, setUsers] = useState([]);

const [msg, setMsg] = useState('');

const [userdetails, setUserDetails] = useState({UserId:'', Password:''});

const history = useHistory();

const [cookies, setCookie] = useCookies(['userid']);

useEffect(() => {

axios.get('http://127.0.0.1:8080/getadmin')

.then(res=>{

setUsers(res.data);

})

}, []);

function handleUserId(e){

setUserDetails({

UserId: e.target.value,

Password: userdetails.Password

})

}

function handlePassword(e) {

setUserDetails({

UserId: userdetails.UserId,

Password: e.target.value

})

}

function handleLogin(){

for(var user of users)

{

if(user.UserId==userdetails.UserId && user.Password==userdetails.Password){

setCookie('userid', userdetails.UserId, {path:'/'})

history.push('/admindashboard');

} else {

setMsg('Invalid Credentials - Try Again');

}

}

}

return(

<div className="w-50">

<h2>Admin Login</h2>

<dl>

<dt>User Id</dt>

<dd><input type="text" onChange={handleUserId} /></dd>

<dt>Password</dt>

<dd><input type="password" onChange={handlePassword} /></dd>

</dl>

<button onClick={handleLogin} >Login</button>

<h3>{msg}</h3>

<div>

New User ? <Link to="/adminregister">Register</Link>

</div>

</div>

)

}

const AdminDashBoard = () => {

const [cookies, setCookie, removeCookie] = useCookies(['userid']);

const [username, setUserName] = useState('');

const history = useHistory();

useEffect(() => {

if(cookies.userid==undefined) {

history.push('/adminlogin');

} else {

setUserName(cookies.userid);

}

}, []);

function handleLogout(){

removeCookie('userid');

history.push('/adminlogin');

}

return(

<div>

<h2>Admin Dashboard - {cookies.userid} Signed In </h2>

<button onClick={handleLogout} className="btn btn-link">Logout</button>

</div>

)

}

export default class IndexComponent extends React.Component

{

render(){

return(

<div className="container-fluid">

<Router>

<header className="bg-danger text-white text-center p-2 mt-2">

<h2><span className="bi bi-cart4"></span> I-Shop </h2>

</header>

<div className="mt-2 row">

<div className="col-3">

<div className="mt-2">

<Link className="btn btn-danger w-100" to="/adminregister" >

Admin Register

</Link>

</div>

<div className="mt-2">

<Link className="btn btn-danger w-100" to="/adminlogin" >

Admin Login

</Link>

</div>

<div className="mt-2">

<Link className="btn btn-danger w-100" to="/admindashboard" >

Admin Dashboard

</Link>

</div>

</div>

<div className="col-9">

<Switch>

<Route exact path="/adminregister" >

<RegisterAdmin />

</Route>

<Route exact path="/adminlogin">

<LoginComponent />

</Route>

<Route exact path="/admindashboard" >

<AdminDashBoard />

</Route>

</Switch>

</div>

</div>

</Router>

</div>

)

}

}

