# **Design Specification Document (DSD) of ReviveMyRide.com**

Group 7 Team Information:

| Name | Title | Email | Phone number |
| --- | --- | --- | --- |
| Mohammed Maheen Khan | Team Leader/Project Manager | Mohakhan@clarku.edu | 857 395 8775 |
| Bakhsinderdeep Singh | Frontend Developer | BakhsinderdeepSing@clarku.edu | 7744189097 |
| Jun Wang | Backend Developer | JunWang2@clarku.edu | 9179913784 |
| Vanzel Vinay Dsilva | Quality Assurance Tester | VDsilva@clarku.edu | 508 774 8682 |
| Venkatesh Seeram | UI/UX designer | vseeram@clarku.edu | 5086850371 |

## 

**Table of contents****2**

[1. Introduction](#_i9hd4blsp5ub) 3

[1.1 Purpose](#_5a8uv2uvznuf) 3

[1.2 Scope](#_tonqbazifvyq) 4

[2. Architectural Design](#_o7b3pyf637za) 6

[2.1 Overview of System Architecture](#_fphtz3hjiuqb) 6

[2.2 High-Level Components](#_hzaa3tgvnmf5) 7

[2.3 Data Flow Diagrams](#_l3r96fvqc0rb) 7

[2.4 System Dependencies](#_knce6s6uopei) 8

[2.5 Technology Stack](#_y9hjrp4jooey) 8

[3. Component Design](#_mptpy8a4vd50) 9

[3.1 User Registration](#_nspxs79mne6m) 9

[3.2 Posting and Listing Car Parts for Sale](#_1f720op914i6) 9

[3.2.1 Part Posting Component:](#_73k32h1gnmw4) 9

[3.2.2 Listing Display Component:](#_m7j3tayfzyw9) 9

[3.3 Search and Filtering Capabilities](#_kokhw0vdl02j) 9

[3.3.1 Search Algorithm:](#_hma7f3i6408j) 9

[3.3.2 Filtering Component:](#_u5qgib1avp61) 9

[3.4 Online Purchase Functionality](#_f45095xqz9pv) 10

[3.4.1 Payment Gateway Integration:](#_ck158jhyde6e) 10

[3.4.2 Transaction Processing Component:](#_oaplda1fol1y) 10

[4. Database Design :](#_2cvms46uopvg) 10

[4.1 Database Schema](#_9b6lphp004yy) 10

[4.2 Data Relationships and Integrity](#_7hu2gfnbiosd) 11

[4.3 Indexing and Optimization](#_nczn21k6hqcz) 11

[5. User Interface Design](#_do63f7b1n1jb) 11

[5.1 Overview of UI Architecture](#_ti1f5qp3iy1o) 11

[5.2 Wireframes and Mockups](#_6wjnyqenwih9) 11

[5.3 Responsiveness across Devices](#_1qqkjyromyh8) 11

[6. Security Design](#_3ey949g61c98) 12

[6.1 Authentication](#_ocqgg41ytzc3) 12

[6.2 Secure Communication](#_v01jkkaynnvh) 12

[6.3 Handling Sensitive Information](#_s8b6wxmnsc82) 12

[7.1 Load Balancing](#_ks3pgqrlfluu) 12

[7.2 Caching Strategies](#_isn081tuk3rc) 12

[8. Deployment Architecture](#_fy0btfal1ou6) 13

[8.1 Hosting Environment](#_hygo8od5qoh4) 13

[8.2 Deployment Diagram:](#_7mh1dxaqy6cg) 13

[8.3 Scalability Plan](#_s0rxricj9yua) 14

[9. Testing and Quality Assurance](#_9d3eswjldxhr) 15

[9.1 Test Scenarios](#_eh37uzqss91n) 15

[9.2 Quality Assurance Processes](#_j00dr3men1fq) 15

[10. Version Control](#_g18rscd1h3pf) 15

[11. Conclusion 15](#_g18rscd1h3pf)

## 

## **1. Introduction**

### **1.1 Purpose**

### In today’s world, there are plenty of technical stacks we can use to build online commerce platforms. After careful and thorough analysis of Clarkton Company’s target user scenario, we decided to focus the design of ReviveMyRide.com on the smooth, convenient and secure user experience, robustness and performance of the system. We leverage state-of-art technologies to achieve this goal. The overall design of ReviveMyRide.com can be seen as three parts, namely, React for user interface, Firebase to take care of all the backend , Cloud fire store for data storage. On the frontend, we handle functionalities related to user registration, car parts listing, search and filtering, online purchase , part listing, handle search and user profiles. On the backend, we provide services like user management, process payment, send notifications and run administrative tasks.

### **1.2 Scope**

The scope of the ReviveMyRide.com project encompasses various aspects of its design, architecture, and functional components. Based on the outlined design principles and technology stack described, the scope includes:

Design Principles and Architecture Focus:

User Experience: Focus on delivering a smooth, convenient, and secure user experience for the automotive parts marketplace.

Robustness and Performance: Emphasize the reliability and performance of the system to ensure efficiency even under heavy loads.

Technological Stack:

Frontend (UI): Utilizing React for building the user interface, ensuring responsiveness and interactivity.

Backend Services: Employing Firebase to handle various backend functionalities, such as user management, payment processing, notifications, and administrative tasks.

Data Storage: Leveraging Cloud Firestore for scalable and flexible data storage.

Frontend Functionalities:

User Registration: Implementing a user registration system to onboard new users.

Car Parts Listing: Facilitating the listing of automotive parts for sale on the platform.

Search and Filtering: Providing robust search and filtering capabilities for users to find specific parts efficiently.

Online Purchase: Enabling secure transactions for users to buy automotive parts.

User Profiles: Allowing users to manage their profiles and track their activities on the platform.

Backend Services:

User Management: Handling user data, authentication, and authorization.

Payment Processing: Managing secure payment transactions for purchases.

Notifications: Implementing a system to send notifications to users for various events or updates.

Additional Considerations:

Security Measures: Ensuring robust security measures are in place for user data, transactions, and communication.

Scalability: Designing the system with scalability in mind to accommodate potential growth in user base and data volume.

Testing and Quality Assurance: Implementing comprehensive testing strategies to ensure reliability, functionality, and security.

Compliance and Regulations: Adhering to relevant industry standards, data protection regulations, and compliance requirements.

Continuous Improvement: Establishing a framework for continuous updates, improvements, and maintenance of the platform based on user feedback and evolving technological trends.

## **2. Architectural Design**

Design Architecture :



### **2.1 Overview of System Architecture**

The system will follow a three-tier architecture:

● Frontend: React for a dynamic and responsive user interface.

● Backend: Firebase to take care of server-side logic

● Database: Cloud fire store for data storage and user management.

### **2.2 High-Level Components**

Frontend Components:

● User Login and Registration

● Car Parts Listing

● Search and Filters

● Online Purchase

● Post an Ad

● Backend Components (Comprises both server , Authentication and Database management for user):

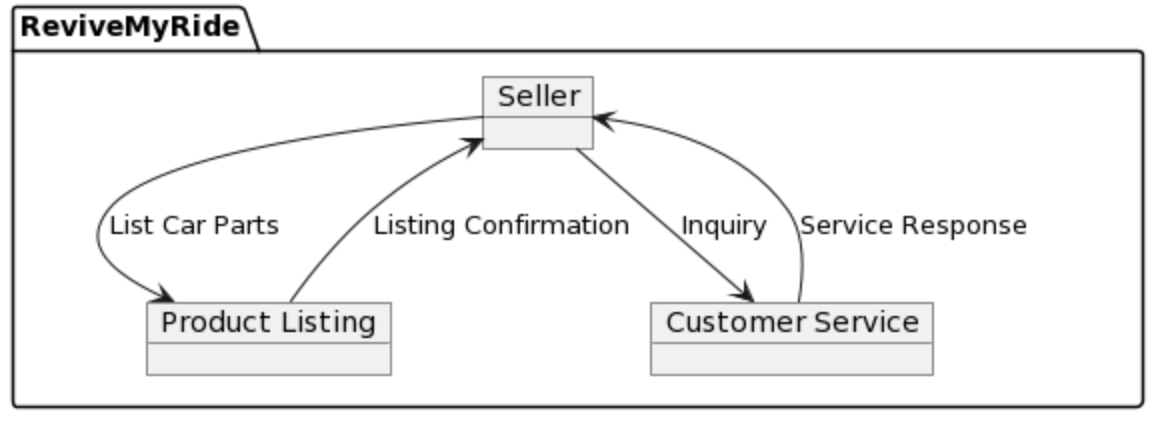
● User Profiles

● User Management

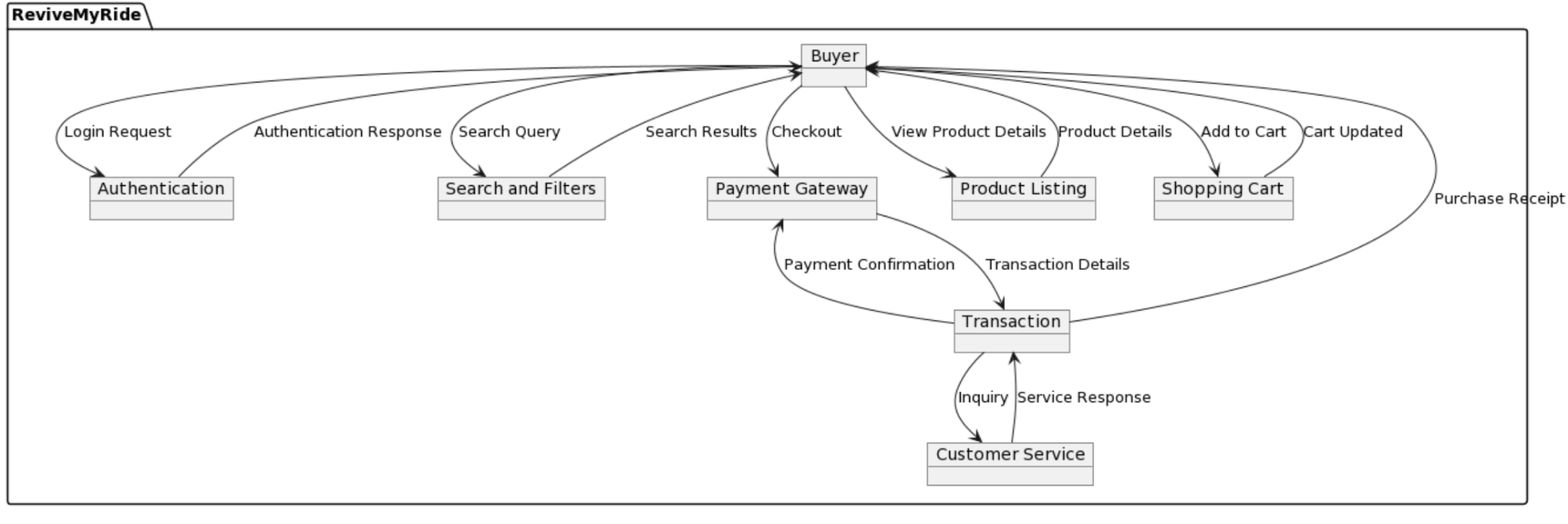
● Payment Processing

### **2.3 Data Flow Diagrams**

Data flow for sellers:



Data flow for buyers:



### **2.4 System Dependencies**

● Node.js (v14.x)

● React (v17.x)

● React Hooks

● Firebase

● Other relevant dependencies

### **2.5 Technology Stack**

● Frontend:

● React.js

● React Hooks for state management

● Backend:

● Firebase

● Cloud Firestore for database

● Deployment:

● Firebase Hosting

## **3. Component Design**

## **3.1 User Registration**

## **3.1. User Registration Component:**

## User Registration: Call Firebase Auth’s createUserWithEmailAndPassword method to register a new user whose data will be updated, stored automatically in firebase to facilitate proper user login validation.

## Validation: Implement server-side validation for user inputs (e.g., username uniqueness, password strength) and client-side validation for a smoother user experience.

## Error Handling: Utilize HTTP status codes (e.g., 400 for bad request) along with descriptive error messages in JSON format to communicate errors back to the user.

## 

## **3.2 Posting and Listing Car Parts for Sale**

## **3.2.1 Part Posting Component:**

## Image Upload: Ability to handle image uploads, ensuring image size limits and formats for better user experience.

## Description: Implement a rich text editor for detailed part descriptions allowing basic formatting for better readability.

## Price Setting: Use appropriate input validation to ensure only valid price formats are accepted.

## **3.2.2 Listing Display Component:**

## Responsive Design: Employ responsive design principles using CSS frameworks (like Bootstrap) for optimal display on various devices.

## Pagination: Implement pagination for listing pages to manage and display a large number of items effectively.

## **3.3 Search and Filtering Capabilities**

## **3.3.1 Search Algorithm:**

## Full-Text Search: Implement full-text search capabilities for efficient keyword-based searches via React.js.

## Indexing: Implement proper indexing on searchable fields for faster query execution.

## **3.3.2 Filtering Component:**

## Dynamic Filters: Create dynamic filtering options based on available attributes of car parts.

## **3.4 Online Purchase Functionality**

## **3.4.1 Payment Gateway Integration:**

## Secure Communication: Ensure HTTPS protocol usage during payment transactions to encrypt data in transit using stripe.

## Error Handling: Handle payment failures gracefully, providing clear instructions for users to retry

## **3.4.2 Transaction Processing Component:**

## Transaction Security: Utilize transactional integrity within the firebase to maintain consistency in case of transactional failures.

## Confirmation Email: Send detailed order summaries via email after successful transactions.

### ***4. Database Design :***



#### *4.1 Database Schema :*

### **User Collection Schema:**

{

"uid": String,

"email": String,

"password": String

}

### **Product Collection Schema:**

{

“Id: Number

"category": String,

"description": String,

"imageUrl": String,

"name": String,

"price": Number

}

### **Cart Collection Schema:**

{

"cartId": String,

"uid": String,

"cartTotalPrice": Number,

"products": [

{

"productId": ObjectId,

"productName": String,

"totalProductPrice": Number,

"quantity": Number,

:price”: : Number

},

// Additional product entries as needed

totalPrice

totalQuantity

]

}

### **Order Collection Schema:**

{

"orderId": String,

"cartId": String,

"orderPrice": Number,

"paymentMethod": String,

"products": [

{

"productId": ObjectId,

"productName": String,

"totalProductPrice": Number,

"quantity": Number

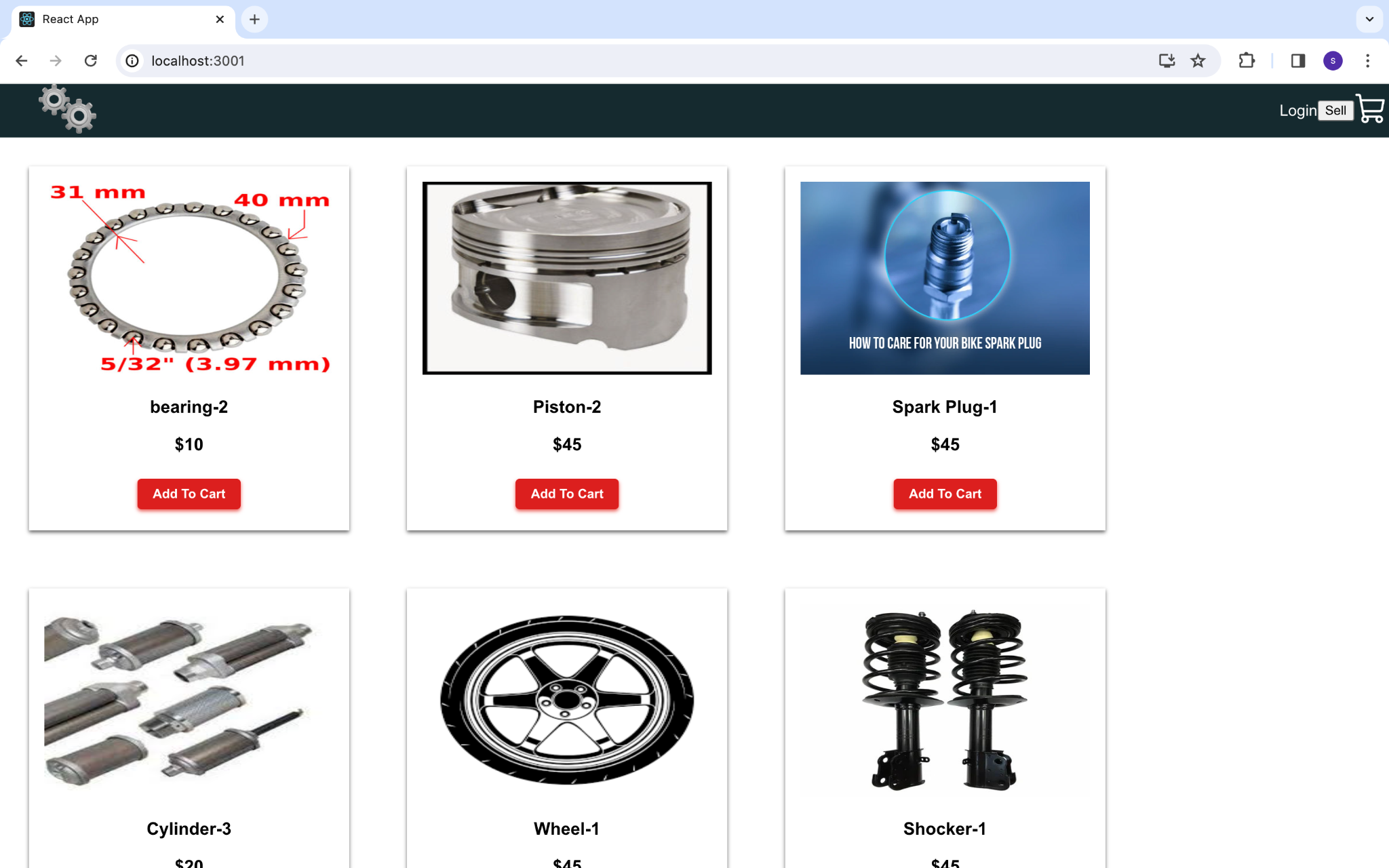
},

// Additional product entries as needed

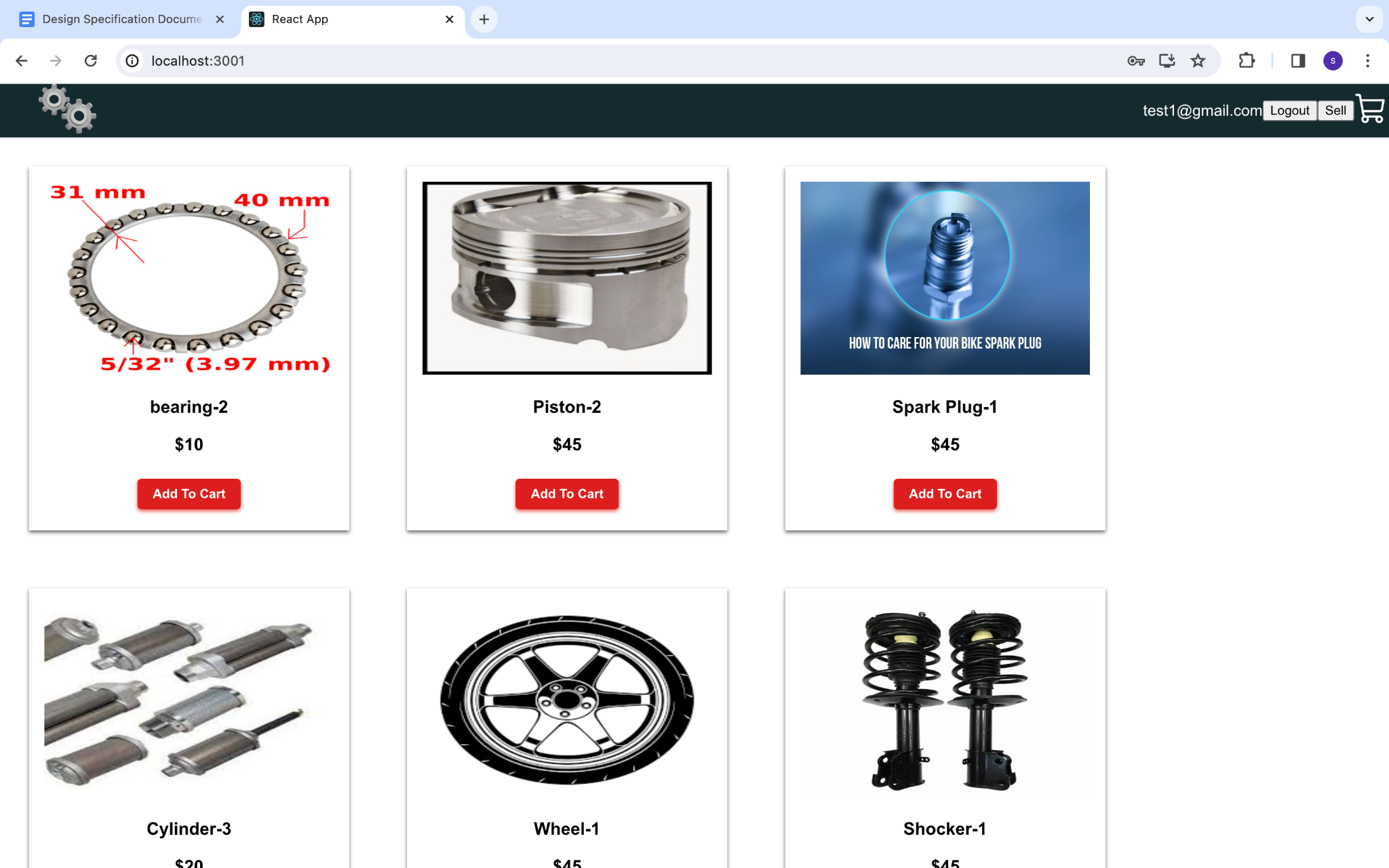
}

### ***5. User Interface Design*** :

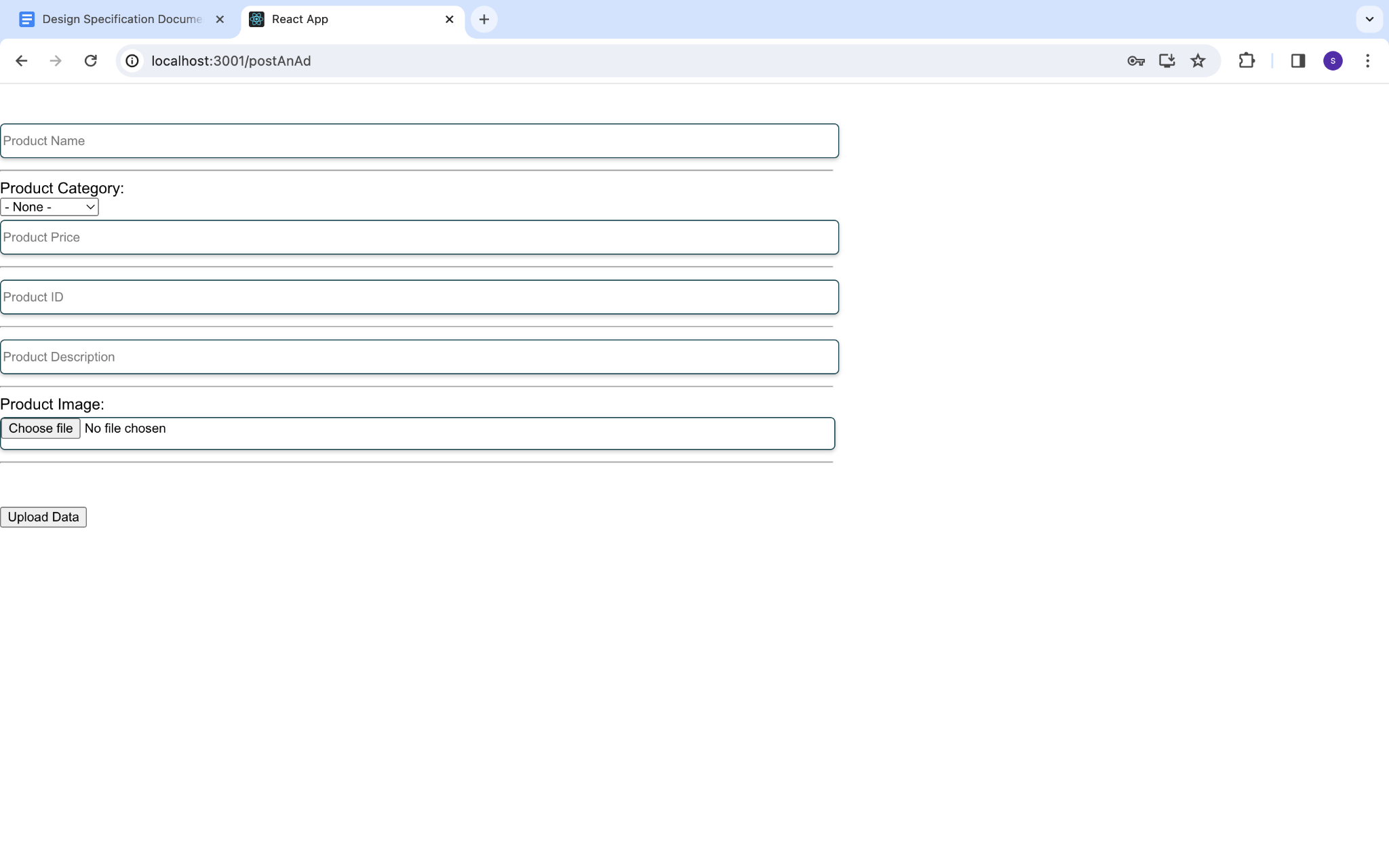
HomePage:



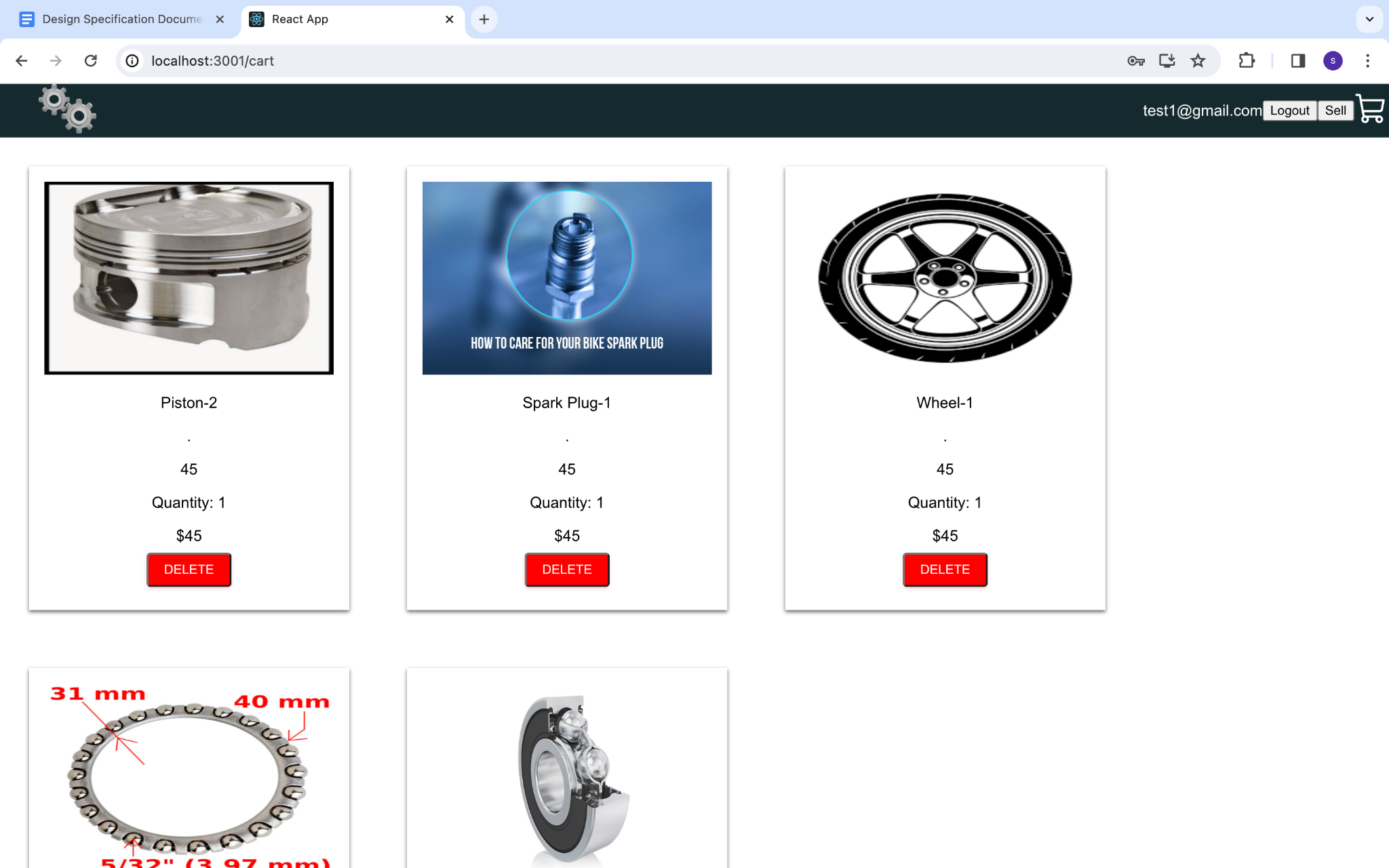
HomePage AfterLogin :



PostAdd Page:



CartPage :



#### *5.1 Overview of UI Architecture*

* *UI Design Patterns:* Component based design pattern using react framework

#### *5.3 Responsiveness across Devices*

* *Responsive Design:* website is compatible to both web browser and mobile browser

***6. Security Design***

#### *6.1 Authentication*

*User Authentication: Firebase Auth's methods like createUserWithEmailAndPassword , signInWithEmailAndPassword are utilized for user registration and login. The user credentials will be sent over to Firebase’s Authentication API and it creates a token and sends it over to the client on successful registration or login respectively.*

#### *6.2 Secure Communication*

* *API Security: Ensure secure communication between frontend and Firebase backend using HTTPS.*
  + *Firebase Security Rules: When customers utilize firebase , Google is generally responsible for data processing under GDPR(EU General Data Protection Regulation) and processes its data on their behalf . It also operates as a service provider under CCPA/CPRA(California Consumer Privacy Act/California Privacy Rights Act) ensuring security to restrict unauthorized access.*

#### *6.3 Handling Sensitive Information*

* *Password Handling: Each firestore object’s data and any metadata associated is encrypted via the 256-bit Advanced Encryption standard..*
  + *Data Encryption: Firestore automatically encrypts all of its data before its being written to the disk.No setup or configuration needed to change the way a user accesses the service*

*.****7. Performance Design***

#### *7.1 Load Balancing*

* *Firebase Load Distribution: As Cloud firestore supports traffic to the database, there will be no issues regarding load balancing .*

#### *7.2 Caching Strategies*

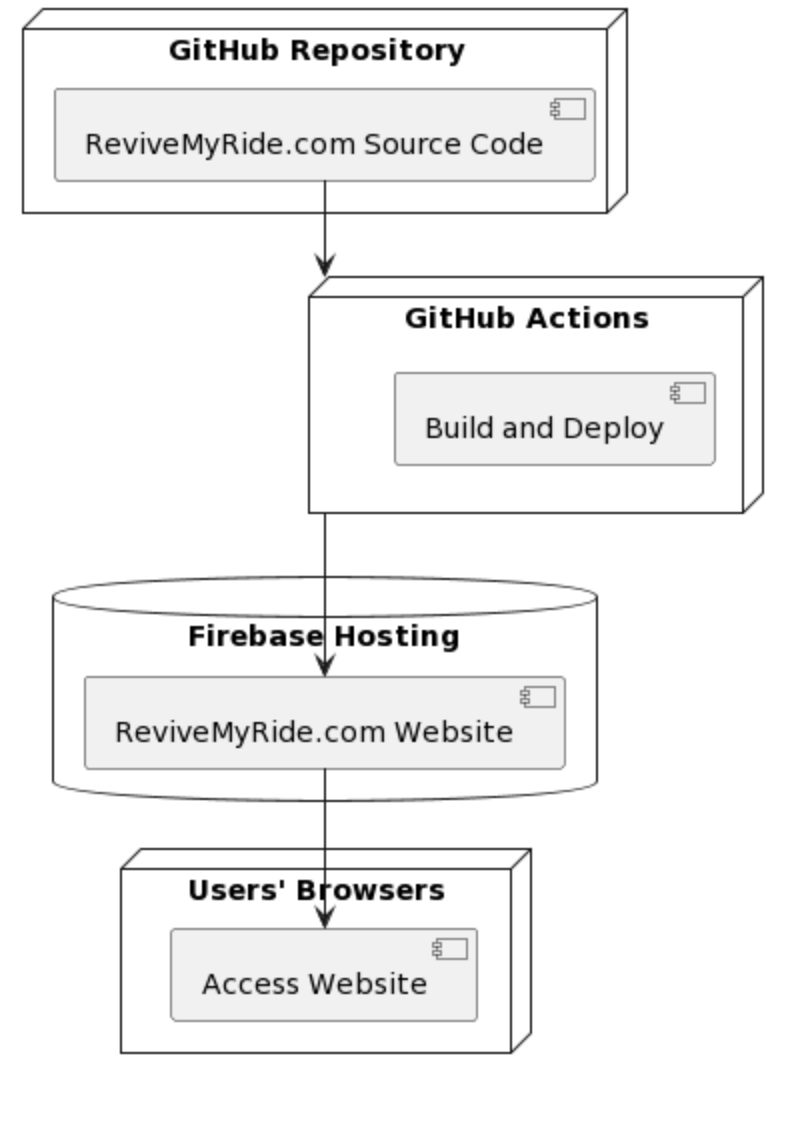
* *Caching Techniques: To Achieve Dynamic caching, proper cache control headers are set .*
  + *Firebase Caching: Any static content will be automatically cached on the global CDN that Firebase uses.*

### ***8. Deployment Architecture***

#### *8.1 Hosting Environment*

* *Firebase Hosting: It is a fully-managed service to serve static, dynamic content as well as microservices. It is backed by SSD storage and global CDN. Zero configuration SSL is already a built-in feature to this ensuring content is delivered securely.*

#### *8.2 Deployment Diagram:*



#### *8.3 Scalability Plan*

* *Scalability Strategies:*

*Firebase Scalability: Firebase Hosting uses a global CDN to serve your content, which can help improve the performance of your app or website. Additionally, Firebase Hosting is designed to scale quickly and automatically as your traffic increases, which can be helpful if you have a lot of dynamic content or need to scale quickly.*

### ***9. Testing and Quality Assurance***

#### *9.1 Test Scenarios*:

* On click of login go to login page
* User Login success : go to home page
* User Login Failure : go the login page again with error pop up
* If clicked on cart page without login redirect to login page
* Add Product to listing : Product will be displayed in home page under respective category
* Purchase : When a product is purchased it should be removed from the category listing.

#### *9.2 Quality Assurance Processes*

* *Code Reviews: Code review should be done by team lead before merge.*

*Continuous Integration* *Continuous Deployment :* Using github actions for *Continuous Integration*

### ***10. Version Control***

* *Version Control System: git*
* *Branching Model: develop branch to hold develop code, main to have finalized code*

***11. Conclusion*** :

In conclusion, the Design Specification Document outlines a comprehensive plan for the development and implementation of ReviveMyRide.com. It provides a roadmap for creating a secure, robust, and user-friendly automotive parts marketplace, embracing modern technologies and best practices. The document serves as a guide for the development team, ensuring consistency and efficiency throughout the project lifecycle.

#### 