Project Report

1. Project Description

The ApplianceRentalHub is an online platform that provides users with easy access to a variety of household and office appliances for temporary rental. This web application addresses the common problem of appliance unavailability in modern apartments and the inconveniences of communal laundry facilities and expensive laundromats. By offering a streamlined rental process, users can easily search for, rent, and manage their appliances without the hassle of long-term commitments.

The platform will also include features for delivery and pick-up options, flexible rental durations, and insurance choices, enhancing user convenience and satisfaction

2. WebApp Functions, Pages, & DB

Technologies:

- Python: For backend business logic to manipulate data and handle request calls.
    - Flask framework – For rendering frontend and routing apis calls
    - pymongo – For Mongo DB connection
- JavaScript: Form validations, Content manipulations, fetch api calls
- HTML5: Adding content structure to the app and design
- CSS3: Styling the whole web-app for User Experience and responsiveness.

Functionalities:

- User Registration & Login: Allow users to create accounts and log in securely.
- Appliance Browsing: Users can search and filter available appliances based on type, brand, and condition.
- Rental Agreement Management: Users can view and manage their rental agreements online.
- Delivery & Pick-Up Scheduling: Users can select their preferred method for receiving appliances.
- Payment Processing: Secure payment processing for rental fees and deposits.
- Maintenance Tracking: Rental agency staff can log maintenance records and view appliance conditions.
- Insurance Options: Users can opt for insurance coverage during the rental process.
- User Dashboard: A personalized area for users to view their rental history and ongoing rentals.

Pages and Files:

- hompage.html: Landing page featuring service overview, login menu, and appliance highlights.
- signup.html & register.js: User registration page to create a new account.
- login.html & login.js: User login page.
- admindashboard.html: Admin interface for managing appliance inventory and maintenance records.
- dashboard.html: User dashboard displaying rental history and account information

- payment.html & payment.js: Secure payment processing page.
- server_main.py: manages all the requests from the frontend and fetches and renders html pages.
- DBManager.py: hands the MongoDB connection and collections
- validator.py: validates the login and signup form data.

3. Application Setup

1. Extract the Zip file: ‘web-application.zip’
2. Navigate to the project folder "cd web-application"
3. Install python from the website: https://www.python.org/
4. Install required libraries of python my using the below command
   pip install -r requirements.txt
5. The Database is connected through MongoDB cloud cluster.
6. Run the service using the command:
   python server_main.py
7. Server will be running in the default port 5000 in localhost
8. Check the status by hitting the URL in the browser
   http://localhost:5000

(Optional)

I. To setup MongoDB locally
II. Install MongoDB compass from https://www.mongodb.com/products/tools/compass
III. Create a local connection string such as eg: mongodb://localhost:27017/ from the MongoDB compass
IV. Import the collections files (.json) shared in the project zip.
V. Replace the connection string of the DB in the file *dbManager.py* to access the data from local.

4. Sample DB Data

appliance:

```
{
  "_id": {
    "$oid": "6743c8fe07e7eb9b07cd2999"
  },
  "brand": "BOSCH",
  "model": "TRES-FG",
  "type": "Air Cooler",
  "condition": "New",
  "rental_rate": "55",
  "availability_status": "Available",
  "deposit_amount": "70",
  "serial_number": "QWERT",
  "features": [
    "Instant Cooling"
  ],
  "image_url": "https://www.bosch-industrial.com/ocsmedia/optimized/2000x2000/o522361v389_Air_Flux_6300_v1.png"
}
```

users:

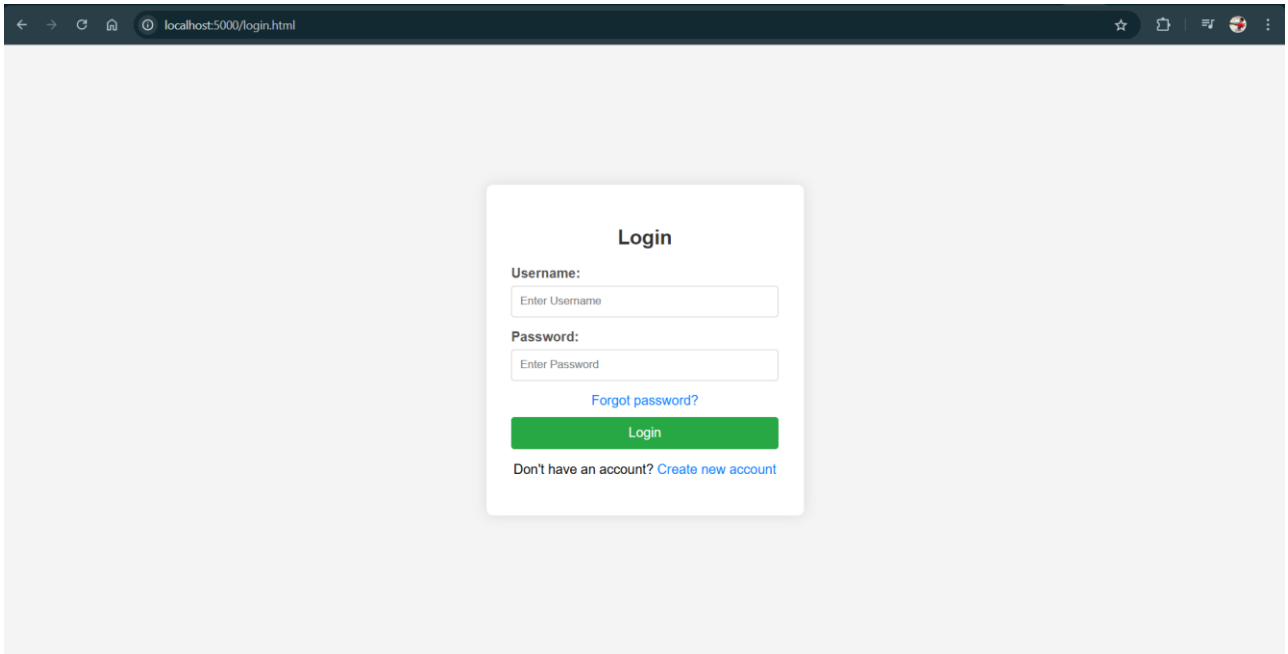
```
{
  "_id": {
    "$oid": "67477aac5d447b1a9dcb470e"
  },
  "firstname": "UserCheck",
```

```
            "lastname": "Out",
            "address": "7654W 109ST",
            "email": "uc@gmail.com",
            "phone": "999-999-9999",
            "ssn": "999-99-9999",
            "password": ""
          }
rentalAgreement:
          {
            "_id": {
              "$oid": "674a24d380fde5e3ed6af3f5"
            },
            "appliance_id": {
              "$oid": "6733d509adf627b2e93d4688"
            },
            "customer_id": {
              "$oid": "674751541cf4f83c8fb41593"
            },
            "rental_start_date": {
              "$date": "2024-11-30T00:00:00.000Z"
            },
            "rental_end_date": {
              "$date": "2024-12-07T00:00:00.000Z"
            },
            "quantity": 2,
            "rental_rate": 90,
            "deposit_amount": 180,
            "total_amount": 270,
            "insurance_status": "In Active",
            "return_status": "not returned",
            "damage_report": "none",
            "delivery_type": "delivery"
          }

Payment:

          {
            "_id": {
              "$oid": "674a24d380fde5e3ed6af3f6"
            },
            "agreement_id": {
              "$oid": "674a24d280fde5e3ed6af3f4"
            },
            "amount": 641,
            "payment_date": {
              "$date": "2024-11-29T20:32:19.213Z"
            },
            "status": "completed",
            "card_number": "4242-4242-4242-4242",
            "cvv": "111",
            "expired_date": "11/34",
            "name_on_card": "Hemanth",
            "zip_code": "12345",
            "card_type": "credit"
          }
customers:
          {
            "_id": {
```

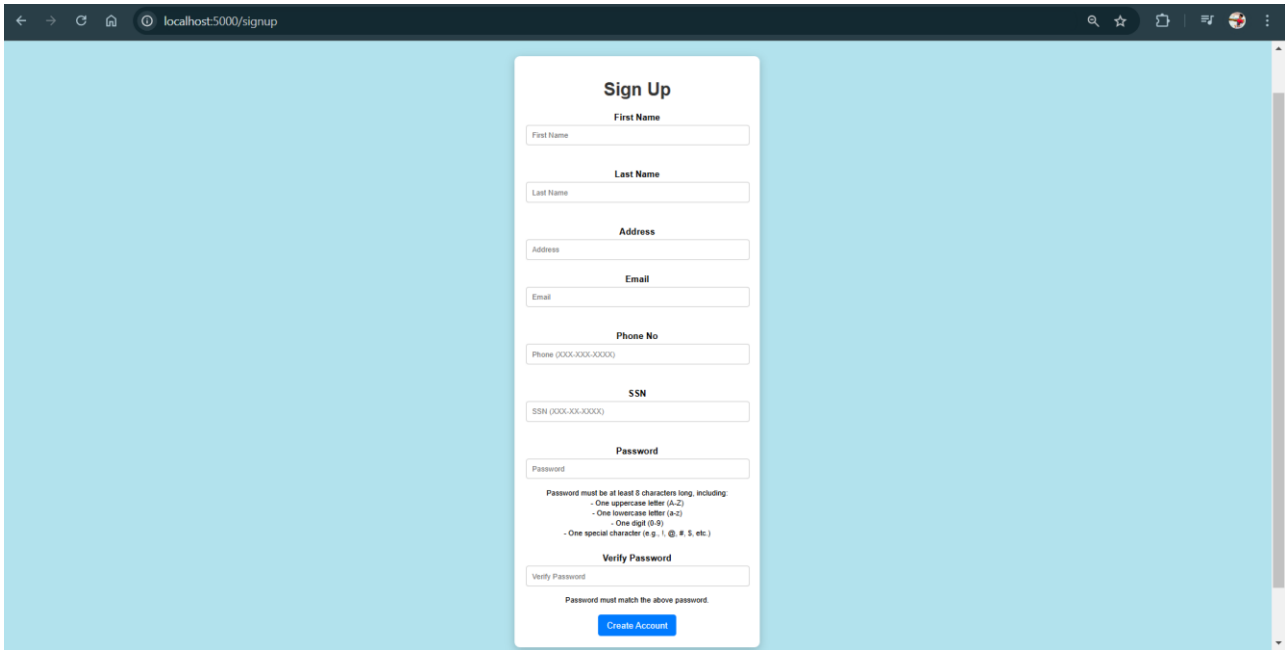
```
    "$oid": "67477af05d447b1a9dcb470f"
  },
  "customer_id": {
    "$oid": "67477aac5d447b1a9dcb470e"
  },
  "user_name": "UserCheck Out",
  "address": "7654W 109ST",
  "phone_number": "999-999-9999",
  "email": "uc@gmail.com",
  "rental_history": [
    {
      "appliance_id": "6743c8fe07e7eb9b07cd2999",
      "quantity": 5,
      "insurance": "Active"
    },
    {
      "appliance_id": "673feb047c7509e16282a019",
      "quantity": 6,
      "insurance": "InActive"
    }
  ]
}
```


5. User Interfaces and Forms

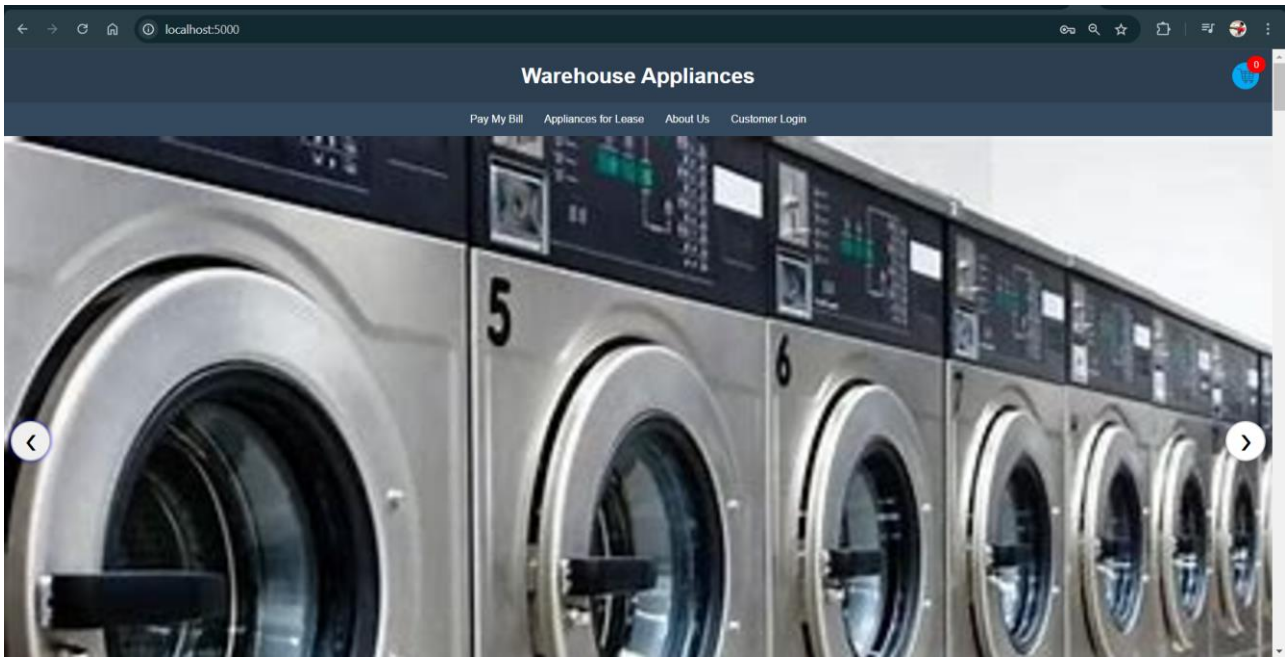
1. **Login page – login.html**

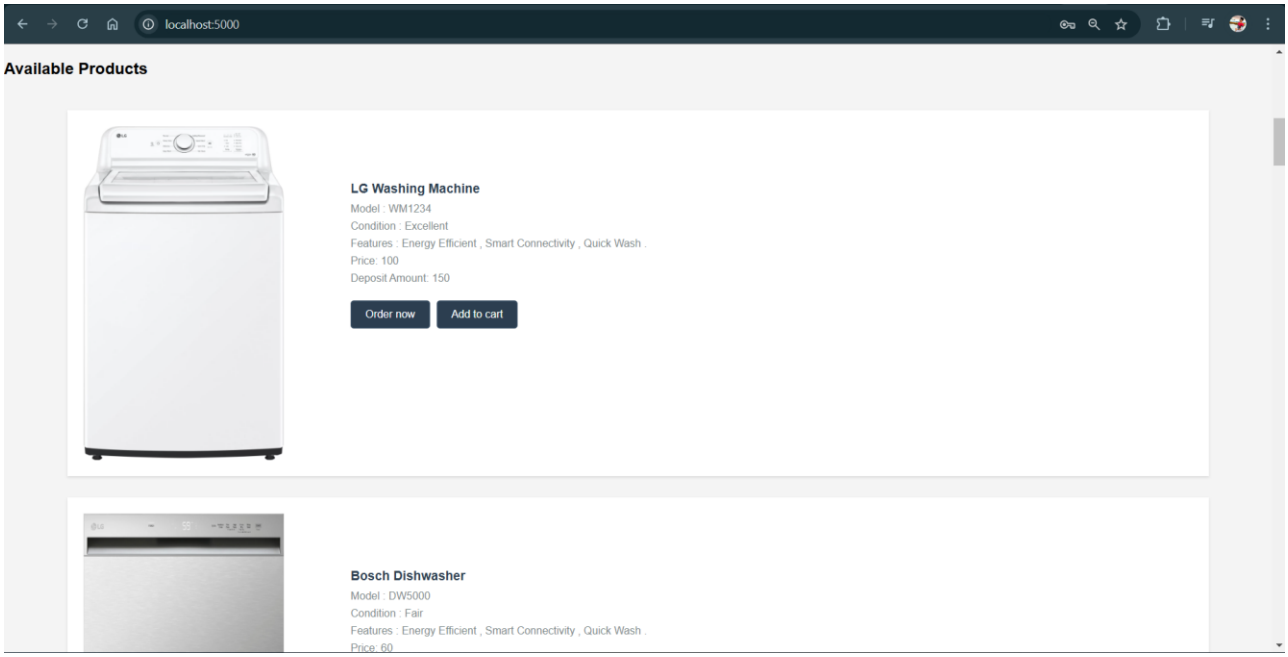


2. User Registration Page – Signup.html

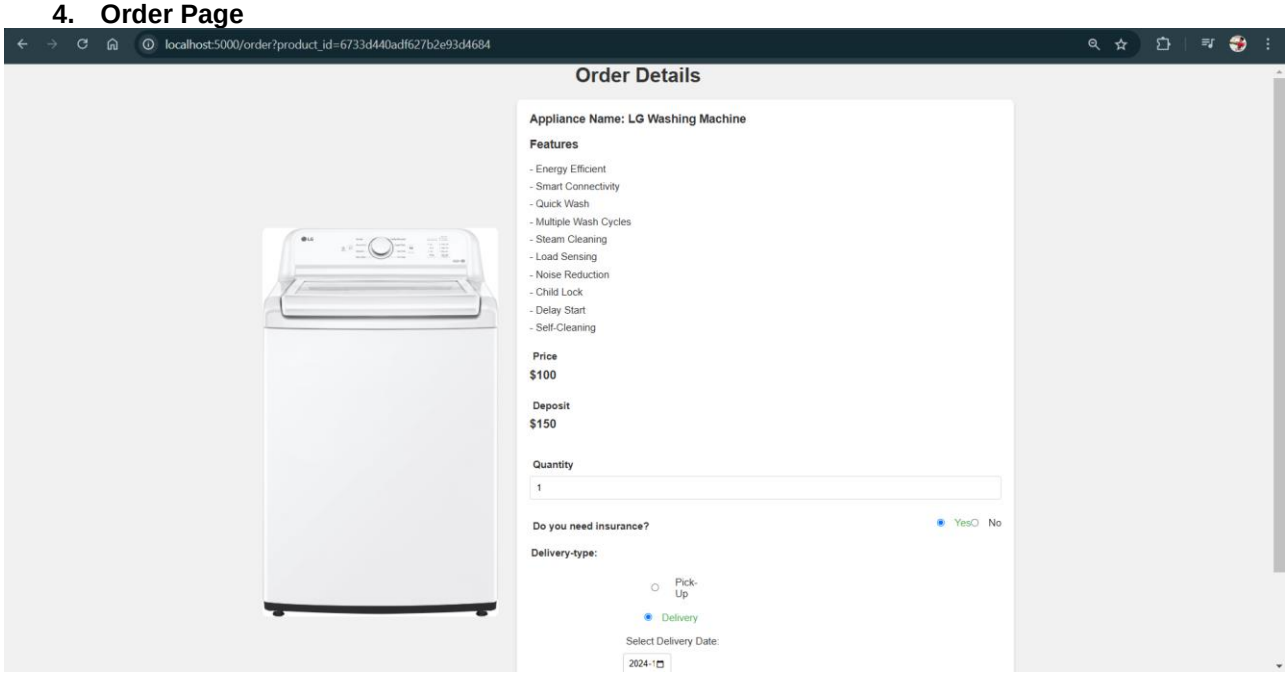


3. HomePage – homepage.html

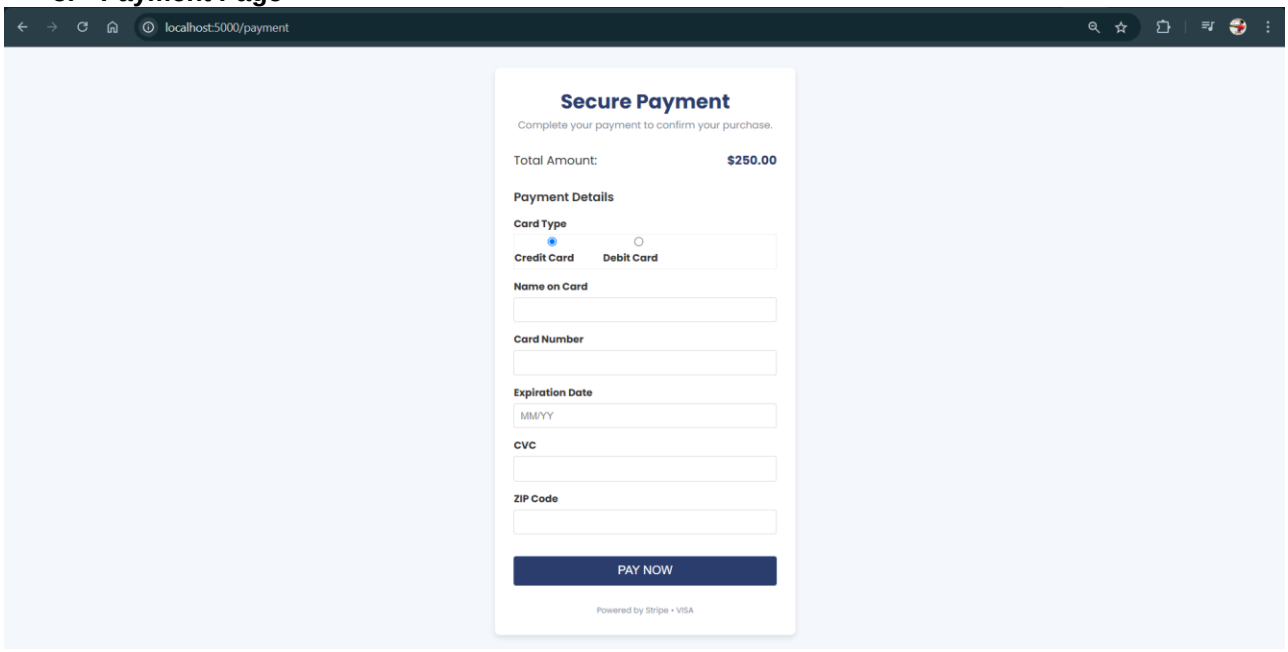



localhost:5000
Available Products
LG Washing Machine
Model : WM1234
Condition : Excellent
Features : Energy Efficient , Smart Connectivity , Quick Wash .
Price: 100
Deposit Amount: 150
Order now
Add to cart
Bosch Dishwasher
Model : DW5000
Condition : Fair
Features : Energy Efficient , Smart Connectivity , Quick Wash .
Price: 60

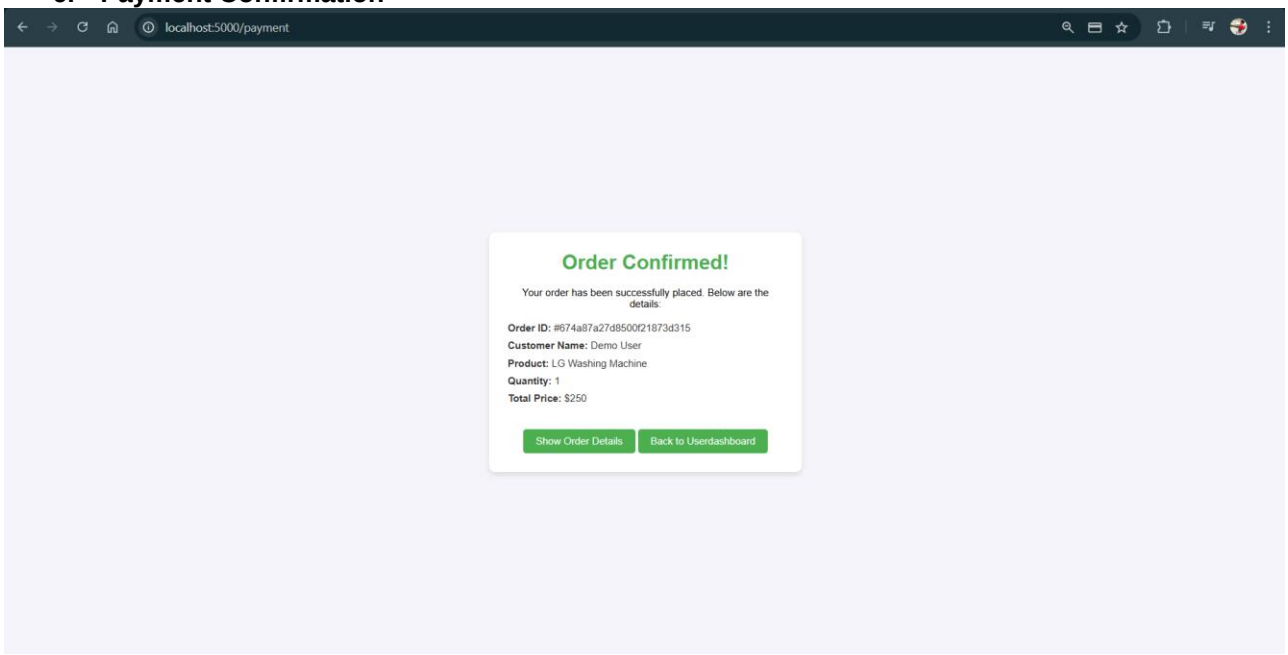
4. Order Page


localhost:5000/order?product_id=6733d440adf627b2e93d4684
Order Details
Appliance Name: LG Washing Machine
Features
- Energy Efficient
- Smart Connectivity
- Quick Wash
- Multiple Wash Cycles
- Steam Cleaning
- Load Sensing
- Noise Reduction
- Child Lock
- Delay Start
- Self-Cleaning
Price
$100
Deposit
$150
Quantity
1
Do you need insurance?
Yes
No
Delivery-type:
Pick-Up
Delivery
Select Delivery Date:

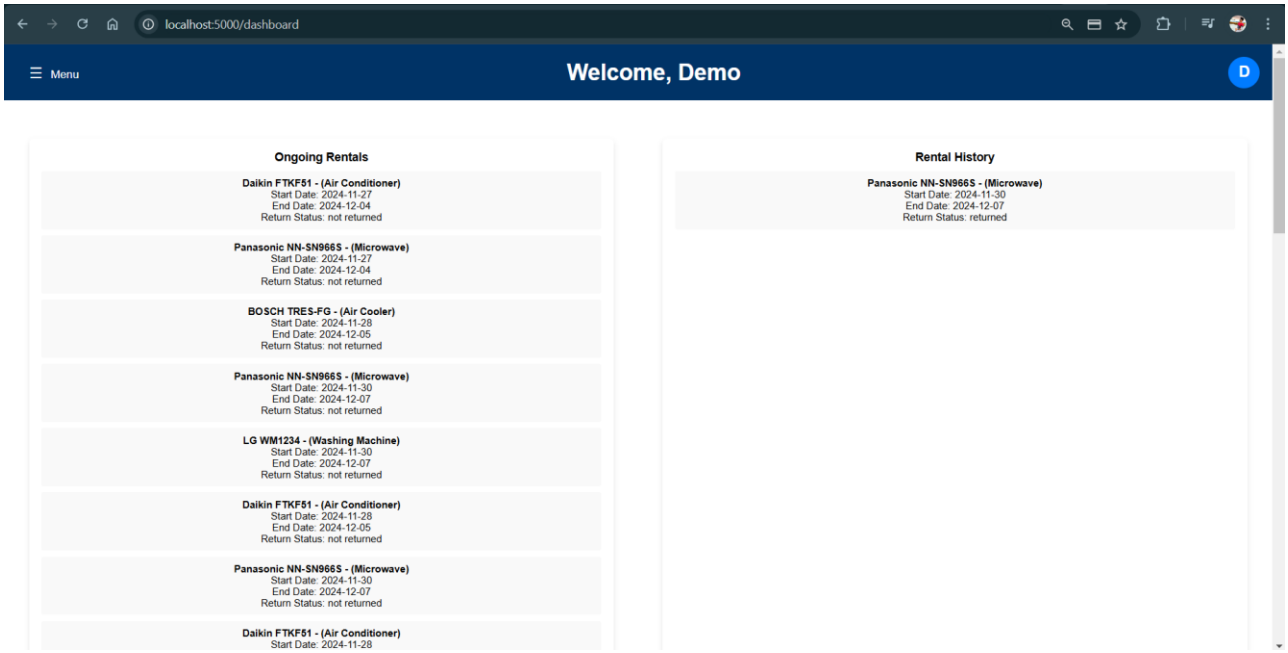
5. Payment Page



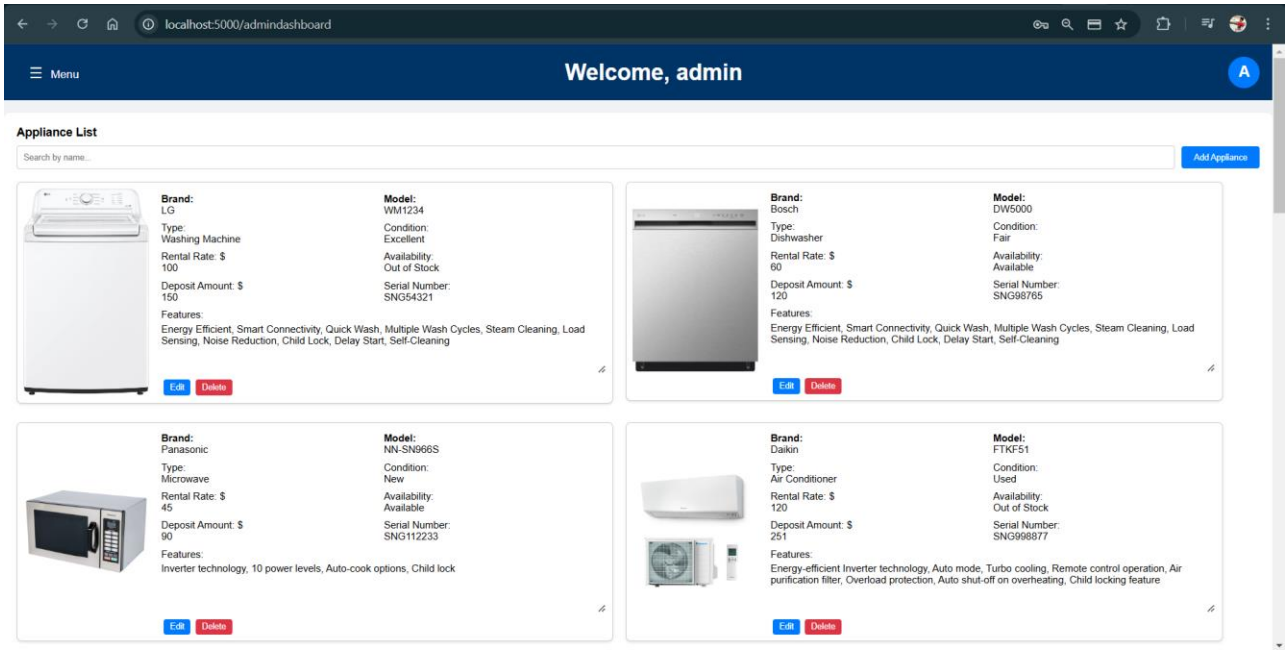
6. Payment Confirmation



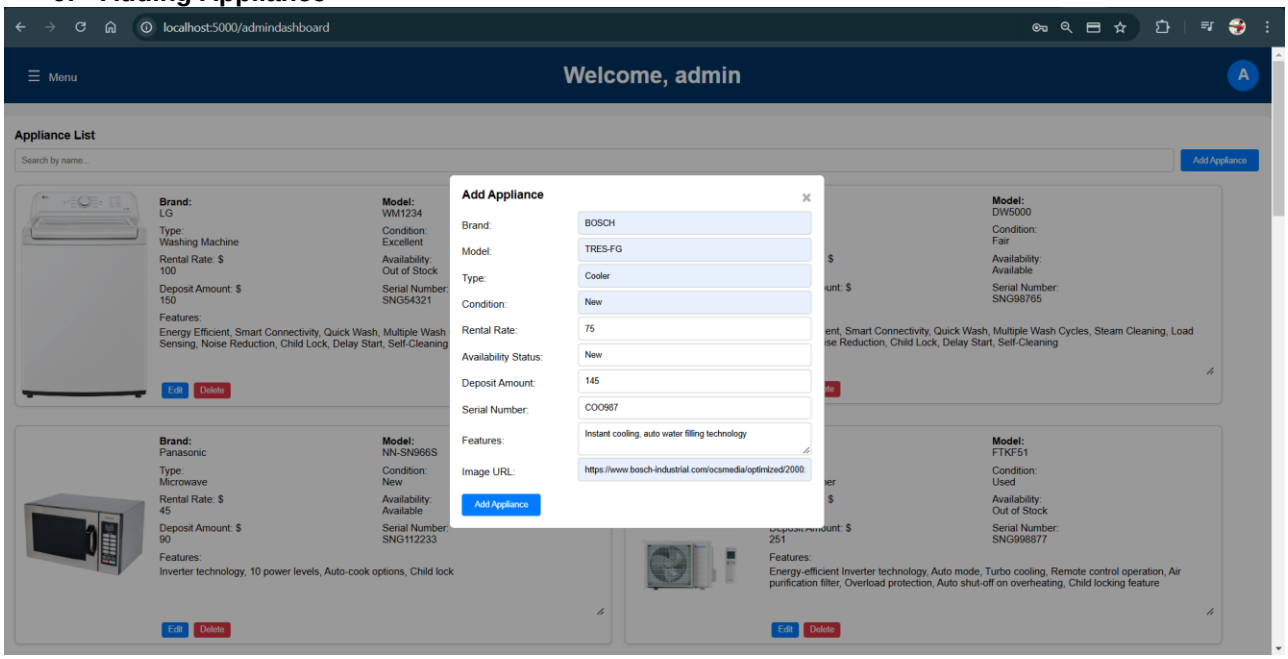
7. User Dashboard



8. Admin Dashboard



9. **Adding Appliance**



6. Source Code

https://github.com/KOTHAPALLIRAJASEKHARREDDY/Web-application

7. Live Application Link

https://web-application-tawny.vercel.app/

8. Contributions:

The project team will be divided into roles based on strengths and interests.

- **Student #1: Sai Kumar Pagalla**
  - Designing the database schema and manages all SQL queries for data retrieval and manipulation.
  - Will utilize HTML, CSS, and JavaScript for responsive design and user experience enhancements for login page.
- **Student #2: Rajasekhar Reddy Kothapalli**
  - Handling server-side programming, focusing on python flask framework for data processing, session management, and API routing.
  - Connecting MongoDB cloud cluster through pymongo and designing DBLayer for database operations.
  - Responsible for designing homepage, signup validations, order page routing post login, payment, and confirmation page design.
- **Student #3: Hemanth Rasabhathula**
  - Designing the User and Admin Dashboards, Cart page and data processing for backend api calls to make CRUD operations with MongoDB.
  - Responsible for storing data in local Storage and making the cart functionality persist across user session and go through payment module.

Conclusion

The ApplianceRentalHub is a rental Application which uses python for backend data logic and the JavaScript, HTML & CSS along with form validations and css manipulations to give a better user experience and intuitive visibility through out the application. For DB MongoDB is used with relational Normalization between collections and stores and retrieved using pymongo library provided by python.