Group j
Object Oriented Programming coursework

BABIRYE HOPE MUSOKE 2023/DCS/DAY/1623G
ASELE JOYCE EVERLINE 2023/DCS/DAY/1353
NAKATO FLORENCE RM 2023/DCS/DAY/1465
ATULA RONALD 2023/DCS/DAY/1140G
AJIGA AKRAM 2023/DCS/DAY/1343

1. **Backend (Java - Spring Boot)**

Project Structure

- GroupJAgriApp (Root)
  - src/main/java/com/groupjagriapp
    - controller
    - service
    - repository
    - model
    - security
    - exception
  - src/main/resources
    - application.properties
  - pom.xml

pom.xml (Dependencies)

xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt</artifactId>
        <version>0.9.1</version>
    </dependency>
    <dependency>
        <groupId>org.postgresql</groupId>
```

```
            <artifactId>postgresql</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
            <groupId>com.stripe</groupId>
            <artifactId>stripe-java</artifactId>
            <version>22.3.0</version>
        </dependency>
</dependencies>
```

application.propertiesConfiguration)

properties

```
spring.datasource.url=jdbc:postgresql://localhost:5432/groupjagriapp
spring.datasource.username=yourUsername
spring.datasource.password=yourPassword
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

jwt.secret=your_secret_key
jwt.expiration=86400000    # 1 day
```

Model (User.java)

java

```
package com.groupjagriapp.model;

import javax.persistence.*;
import javax.validation.constraints.NotBlank;

@Entity
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
    @NotBlank(message = "Username is mandatory")
    private String username;

    @NotBlank(message = "Password is mandatory")
    private String password;

    @NotBlank(message = "Role is mandatory")
    private String role; // 'Admin', 'Seller', 'Buyer'

    // Getters and Setters

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }
}
```

Model (Product.java)
java

```
package com.groupjagriapp.model;

import javax.persistence.*;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotBlank;

@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long productID;

    @NotBlank(message = "Product name is mandatory")
    private String name;

    @NotBlank(message = "Category is mandatory")
    private String category;

    @Min(value = 0, message = "Price must be positive")
    private Double price;

    @Min(value = 0, message = "Quantity must be positive")
    private Integer quantity;

    // Getters and Setters

    public Long getProductID() {
        return productID;
    }

    public void setProductID(Long productID) {
        this.productID = productID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
```

```
        this.name = name;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public Integer getQuantity() {
        return quantity;
    }

    public void setQuantity(Integer quantity) {
        this.quantity = quantity;
    }
}
```

Repository (UserRepository.java)

java

```
package com.groupjagriapp.repository;

import com.groupjagriapp.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

Repository (ProductRepository.java)

java

```
package com.groupjagriapp.repository;
```

```
import com.groupjagriapp.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface ProductRepository extends JpaRepository<Product, Long> {
    List<Product> findByCategory(String category);
}
```

Service (UserService.java)

java

```
package com.groupjagriapp.service;

import com.groupjagriapp.model.User;
import com.groupjagriapp.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class UserService {

    @Autowired
    private UserRepository userRepository;

    public User findByUsername(String username) {
        return userRepository.findByUsername(username);
    }

    public void saveUser(User user) {
        userRepository.save(user);
    }
}
```

Service (ProductService.java)

java

```
package com.groupjagriapp.service;

import com.groupjagriapp.model.Product;
import com.groupjagriapp.repository.ProductRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
```

```
@Service
public class ProductService {

    @Autowired
    private ProductRepository productRepository;

    public List<Product> getAllProducts() {
        return productRepository.findAll();
    }

    public Product addProduct(Product product) {
        return productRepository.save(product);
    }

    public Product updateProduct(Long id, Product productDetails) {
        return productRepository.findById(id)
                .map(product -> {
                    product.setName(productDetails.getName());
                    product.setPrice(productDetails.getPrice());
                    product.setQuantity(productDetails.getQuantity());
                    return productRepository.save(product);
                }).orElse(null);
    }

    public void deleteProduct(Long id) {
        productRepository.deleteById(id);
    }
}
```

Security (JwtTokenProvider.java)
java

```
package com.groupjagriapp.security;

import io.jsonwebtoken.*;
import org.springframework.stereotype.Component;
import java.util.Date;

@Component
public class JwtTokenProvider {

    private String secret = "your_secret_key";
    private long expiration = 86400000; // 1 day
```

```
    public String generateToken(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new Date())
                .setExpiration(new Date(System.currentTimeMillis() + expiration))
                .signWith(SignatureAlgorithm.HS512, secret)
                .compact();
    }

    public boolean validateToken(String token) {
        try {
            Jwts.parser().setSigningKey(secret).parseClaimsJws(token);
            return true;
        } catch (JwtException | IllegalArgumentException e) {
            return false;
        }
    }

    public String getUsernameFromToken(String token) {
        return
Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody().getSubject();
    }
}
```

Security (SecurityConfig.java)
java

```
package com.groupjagriapp.security;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBui
lder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
public class SecurityConfig {
```

```
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.csrf().disable()
            .authorizeRequests()
            .antMatchers("/api/auth/**").permitAll() // Allow authentication endpoints
            .anyRequest().authenticated()
            .and()
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);

        return http.build();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

Controller (AuthController.java)

java

```
package com.groupjagriapp.controller;

import com.groupjagriapp.model.User;
import com.groupjagriapp.security.JwtTokenProvider;
import com.groupjagriapp.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/auth")
public class AuthController {

    @Autowired
    private UserService userService;

    @Autowired
    private JwtTokenProvider jwtTokenProvider;

    @Autowired
    private PasswordEncoder passwordEncoder;
```

```
    @PostMapping("/signup")
    public ResponseEntity<String> signup(@Validated @RequestBody User user) {
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        userService.saveUser(user);
        return ResponseEntity.ok("User registered successfully!");
    }

    @PostMapping("/login")
    public ResponseEntity<String> login(@RequestBody User user) {
        User foundUser = userService.findByUsername(user.getUsername());
        if (foundUser != null && passwordEncoder.matches(user.getPassword(),
foundUser.getPassword())) {
            String token = jwtTokenProvider.generateToken(foundUser.getUsername());
            return ResponseEntity.ok(token);
        } else {
            return ResponseEntity.status(401).body("Invalid credentials");
        }
    }
}
```

Controller (ProductController.java)

java

```
package com.groupjagriapp.controller;

import com.groupjagriapp.model.Product;
import com.groupjagriapp.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import javax.validation.Valid;
import java.util.List;

@RestController
@RequestMapping("/api/products")
public class ProductController {

    @Autowired
    private ProductService productService;

    @GetMapping
    public List<Product> getAllProducts() {
        return productService.getAllProducts();
    }
```

```
    @PostMapping
    public Product addProduct(@Valid @RequestBody Product product) {
        return productService.addProduct(product);
    }

    @PutMapping("/{id}")
    public Product updateProduct(@PathVariable Long id, @Valid @RequestBody Product product) {
        return productService.updateProduct(id, product);
    }

    @DeleteMapping("/{id}")
    public void deleteProduct(@PathVariable Long id) {
        productService.deleteProduct(id);
    }
}
```

Exception Handling (CustomExceptionHandler.java)

java

```
package com.groupjagriapp.exception;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice
public class CustomExceptionHandler {

    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> handleException(Exception e) {
        return new ResponseEntity<>(e.getMessage(), HttpStatus.BAD_REQUEST);
    }
}
```

2. **Frontend (React) Updates**

Project Structure

- src/
  - components/
    - ProductList.js

- ProductForm.js
- Login.js
- Signup.js
- Payment.js

- App.js
-index.js
-services/ (Add Axios service for API calls)

nstall Dependencies
Run the following command to install Material-UI and Axios:
bash
npm install @mui/material @emotion/react @emotion/styled axios jwt-decode

Axios Service (api.js)
javascript

```
import axios from 'axios';

const api = axios.create({
    baseURL: 'http://localhost:8080/api',
});

// Add a request interceptor to include the token in headers
api.interceptors.request.use(config => {
    const token = localStorage.getItem('token');
    if (token) {
        config.headers['Authorization'] = `Bearer ${token}`;
    }
    return config;
}, error => {
    return Promise.reject(error);
});

export default api;
```

Login Component (Login.js)
javascript

```
import React, { useState } from 'react';
import api from '../services/api';
import { TextField, Button, Typography } from '@mui/material';

const Login = () => {
    const [user, setUser] = useState({ username: '', password: '' });
    const [error, setError] = useState('');
```

```
    const handleSubmit = (event) => {
        event.preventDefault();
        api.post('/auth/login', user)
            .then(response => {
                localStorage.setItem('token', response.data);
                window.location.href = '/'; // Redirect after successful login
            })
            .catch(error => {
                setError('Invalid credentials');
            });
    };

    return (
        <form onSubmit={handleSubmit}>
            <Typography variant="h4">Login</Typography>
            <TextField label="Username" variant="outlined" fullWidth
                value={user.username} onChange={(e) => setUser({ ...user, username:
e.target.value })} />
            <TextField label="Password" type="password" variant="outlined" fullWidth
                value={user.password} onChange={(e) => setUser({ ...user, password:
e.target.value })} />
            {error && <Typography color="error">{error}</Typography>}
            <Button type="submit" variant="contained" color="primary">Login</Button>
        </form>
    );
};

export default Login;
`Signup Component (Signup.js)
javascript
import React, { useState } from 'react';
import api from '../services/api';
import { TextField, Button, Typography } from '@mui/material';

const Signup = () => {
    const [user, setUser] = useState({ username: '', password: '', role: 'Buyer' });
    const [success, setSuccess] = useState('');

    const handleSubmit = (event) => {
        event.preventDefault();
        api.post('/auth/signup', user)
            .then(() => {
                setSuccess('User registered successfully!');
```

```
            })
            .catch(error => {
                console.error('Error during signup:', error);
            });
    };

    return (
        <form onSubmit={handleSubmit}>
            <Typography variant="h4">Signup</Typography>
            <TextField label="Username" variant="outlined" fullWidth
                value={user.username} onChange={(e) => setUser({ ...user, username:
e.target.value })} />
            <TextField label="Password" type="password" variant="outlined" fullWidth
                value={user.password} onChange={(e) => setUser({ ...user, password:
e.target.value })} />
            <TextField label="Role" variant="outlined" fullWidth
                value={user.role} onChange={(e) => setUser({ ...user, role: e.target.value })}
/>
            {success && <Typography color="success">{success}</Typography>}
            <Button type="submit" variant="contained" color="primary">Signup</Button>
        </form>
    );
};

export default Signup;
```

Product List Component (ProductList.js)

```
javascript
import React, { useEffect, useState } from 'react';
import api from '../services/api';
import { Button, Typography } from '@mui/material';

const ProductList = () => {
    const [products, setProducts] = useState([]);

    useEffect(() => {
        api.get('/products')
            .then(response => {
                setProducts(response.data);
            })
            .catch(error => {
                console.error('Error fetching products:', error);
            });
    }, []);
```

```
    return (
        <div>
            <Typography variant="h4">Product List</Typography>
            <ul>
                {products.map(product => (
                    <li key={product.productID}>
                        <Typography>{product.name} - {product.category} - ${product.price}</Typography>
                    </li>
                ))}
            </ul>
            <Button variant="contained" color="primary" href="/add-product">Add Product</Button>
        </div>
    );
};

export default ProductList;
```

Product Form Component (ProductForm.js)

javascript

```
import React, { useState } from 'react';
import api from '../services/api';
import { TextField, Button, Typography } from '@mui/material';

const ProductForm = () => {
    const [product, setProduct] = useState({ name: '', category: '', price: 0, quantity: 0 });
    const [success, setSuccess] = useState('');

    const handleSubmit = (event) => {
        event.preventDefault();
        api.post('/products', product)
            .then(() => {
                setSuccess('Product added successfully');
            })
            .catch(error => {
                console.error('Error adding product:', error);
            });
    };

    return (
        <form onSubmit={handleSubmit}>
```

```
                <Typography variant="h4">Add Product</Typography>
                <TextField label="Product Name" variant="outlined" fullWidth
                    value={product.name} onChange={(e) => setProduct({ ...product, name:
e.target.value })} />
                <TextField label="Category" variant="outlined" fullWidth
                    value={product.category} onChange={(e) => setProduct({ ...product, category:
e.target.value })} />
                <TextField label="Price" type="number" variant="outlined" fullWidth
                    value={product.price} onChange={(e) => setProduct({ ...product, price:
parseFloat(e.target.value) })} />
                <TextField label="Quantity" type="number" variant="outlined" fullWidth
                    value={product.quantity} onChange={(e) => setProduct({ ...product, quantity:
parseInt(e.target.value, 10) })} />
                {success && <Typography color="success">{success}</Typography>}
                <Button type="submit" variant="contained" color="primary">Add
Product</Button>
            </form>
        );
};

export default ProductForm;
```

App Component (App.js)

javascript

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import ProductList from './components/ProductList';
import ProductForm from './components/ProductForm';
import Login from './components/Login';
import Signup from './components/Signup';

function App() {
    return (
        <Router>
            <Switch>
                <Route exact path="/" component={ProductList} />
                <Route path="/add-product" component={ProductForm} />
                <Route path="/login" component={Login} />
                <Route path="/signup" component={Signup} />
            </Switch>
        </Router>
    );
}
```

```
export default App;
```

3. **Payment Integration (Using Stripe)**
Backend Payment Controller (PaymentController.java)
java

```
package com.groupjagriapp.controller;

import com.stripe.Stripe;
import com.stripe.model.PaymentIntent;
import com.stripe.param.PaymentIntentCreateParams;
import org.springframework.web.bind.annotation.*;

import java.util.Map;

@RestController
@RequestMapping("/api/payments")
public class PaymentController {

    public PaymentController() {
        Stripe.apiKey = "your_stripe_secret_key"; // Set your Stripe secret key
    }

    @Post
Mapping
    public String createPaymentIntent(@RequestBody Map<String, Object> request) {
        PaymentIntentCreateParams params = PaymentIntentCreateParams.builder()
                .setAmount((Long) request.get("amount")) // Amount in cents
                .setCurrency("usd")
                .build();

        try {
            PaymentIntent paymentIntent = PaymentIntent.create(params);
            return paymentIntent.toJson();
        } catch (Exception e) {
            return "Error: " + e.getMessage();
        }
    }
}
```

Frontend Payment Component (Payment.js)
javascript

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

```
import api from '../services/api';
import { TextField, Button, Typography } from '@mui/material';

const Payment = () => {
    const [amount, setAmount] = useState(0);
    const [success, setSuccess] = useState('');

    const handlePayment = () => {
        api.post('/payments', { amount: amount * 100 }) // Amount in cents
            .then(response => {
                setSuccess('Payment successful!');
            })
            .catch(error => {
                console.error('Payment error:', error);
            });
    };

    return (
        <div>
            <Typography variant="h4">Payment</Typography>
            <TextField
                type="number"
                placeholder="Enter amount"
                value={amount}
                onChange={(e) => setAmount(e.target.value)}
            />
            <Button onClick={handlePayment} variant="contained" color="primary">Pay</Button>
            {success && <Typography color="success">{success}</Typography>}
        </div>
    );
};

export default Payment;
```

4. **Docker Compose File**
`docker-compose.yml` file to run both frontend and backend.

```
yaml
version: '3'
services:
  backend:
    build: ./backend
```

```
    ports:
      - "8080:8080"
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://db:5432/groupjagriapp
      SPRING_DATASOURCE_USERNAME: yourUsername
      SPRING_DATASOURCE_PASSWORD: yourPassword

frontend:
    build: ./frontend
    ports:
      - "3000:3000"
```