BAB V
IMPLEMENTASI DAN PENGUJIAN SISTEM

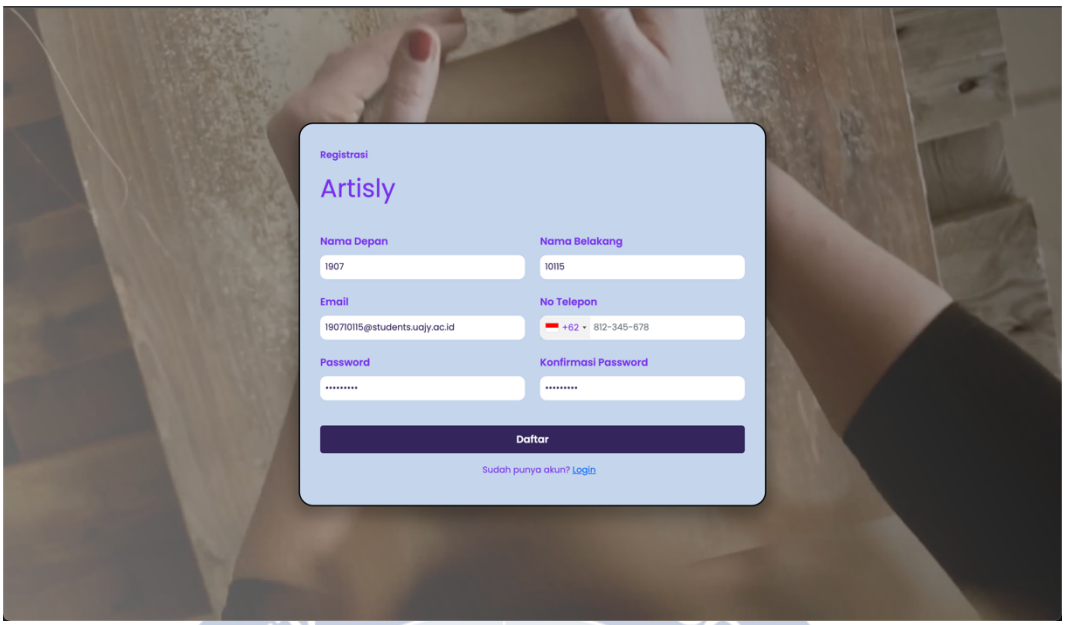
Setelah membahas perancangan dari sistem pada, Bab ini akan membahas implementasi dan pengujian dari sistem informasi *e-commerce* berbasis *website.* Bab ini akan memaparkan kode sumber implementasi dari sistem, serta hasil pengujian pengguna menggunakan metode *blackbox testing.* Bab ini akan menyertakan tangkapan layar dari implementasi antarmuka sistem serta tangkapan layar kode sumber yang penting untuk menjalankan fungsionalitas dari halaman yang dibahas.

A. Implementasi Sistem Implementasi Antarmuka

Gambar 5.1 adalah tampilan antarmuka dari halaman registrasi yang digunakan pengguna untuk mendaftar ke dalam sistem. *Frontend* akan mengirimkan seluruh data pengguna yang diisi di *form* saat pengguna menekan tombol “Daftar” ke *backend* melalui *REST API Call*.

Gambar 5.2 menunjukkan bahwa *presentation layer* merupakan salah satu bagian terpenting bagi keamanan sistem. Pada fungsi registrasi dan fungsi-fungsi lainnya, *presentation layer* memiliki fungsi untuk melakukan validasi data dan memastikan bahwa seluruh *input* dari pengguna sudah sesuai dengan format yang diharapkan dan tidak mengandung hal-hal yang bersifat jahat sebagai bagian dari strategi keamanan berlapis yang dimiliki oleh sistem. Validasi yang baik bisa mengurangi risiko serangan berjenis *injection* seperti *XSS Injection* dan *SQL Injection.*

Pada Gambar 5.3, terdapat potongan kode *service layer* yang menjalankan logika seperti memastikan bahwa alamat *email* dan nomor telepon yang digunakan belum terdaftar pada *database.* Pengecekan tersebut dilakukan agar *backend* memiliki informasi lebih banyak dan bisa merespon dengan dua *exception* berbeda jika salah satu dari alamat *email* atau nomor telepon yang dimasukan oleh pengguna sudah digunakan oleh pengguna lain.



Gambar 5.1 Halaman Registrasi

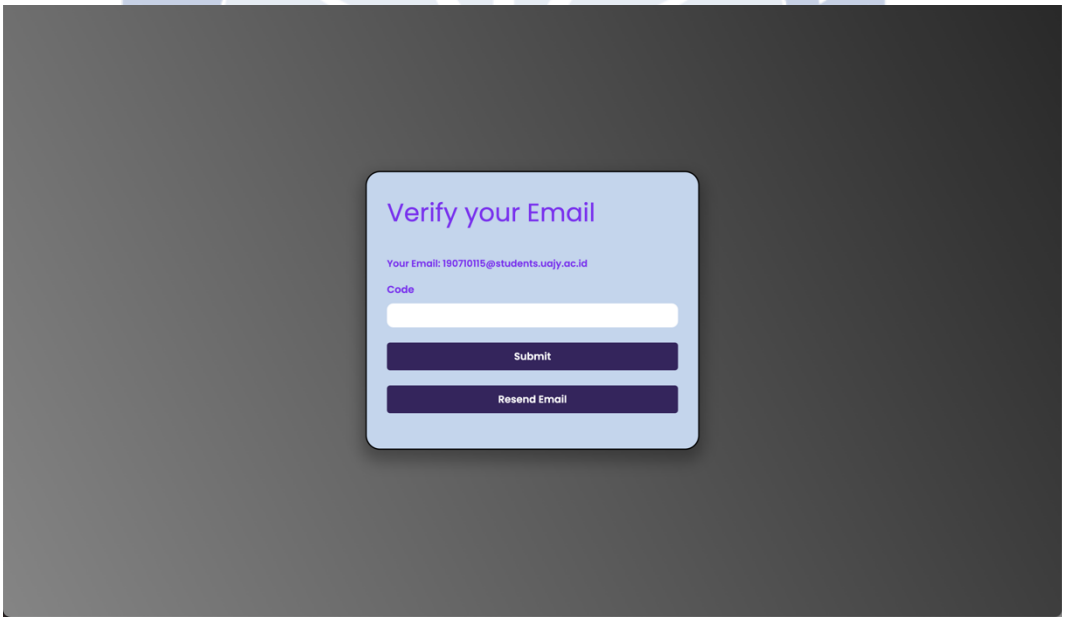
```java
@Override
public void handle(Context context) {
    InputFilter.validateEmail( field: "email", ParamField.FORM, context);
    InputFilter.validateName( field: "first_name", ParamField.FORM, context);
    InputFilter.validateName( field: "last_name", ParamField.FORM, context);
    InputFilter.validatePassword( field: "password", ParamField.FORM, context);
    InputFilter.validatePassword( field: "confirm_password", ParamField.FORM, context);
    InputFilter.validatePhoneNumber( field: "phone_number", ParamField.FORM, context);

    if (context.attribute( key: "hasErrors") != null) {
        StandarizedResponses.invalidParameter(context);
        return;
    }

    String email = context.formParam( key: "email");
    String firstName = context.formParam( key: "first_name");
    String lastName = context.formParam( key: "last_name");
    char[] password = context.formParam( key: "password").toCharArray();
    char[] confirmPassword = context.formParam( key: "confirm_password").toCharArray();
    String phoneNumber = context.formParam( key: "phone_number");

    if (!Arrays.equals(password, confirmPassword)) {
        ErrorContainer errorContainer = new ErrorContainer( parameter: "confirm_password",  error: "Passwords do not match");
        if (context.attribute( key: "errors") == null)
        {
            context.attribute("errors", new ArrayList<ErrorContainer>());
            context.attribute("hasErrors", true);
        }
        ArrayList<ErrorContainer> errors = context.attribute( key: "errors");
        errors.add(errorContainer);
        StandarizedResponses.invalidParameter(context);
        return;
    }

    try
    {
        User user = UserService.register(email, password, firstName, lastName, phoneNumber);
        StandarizedResponses.success(
                context,
                 message: "REGISTER_SUCCESS",
                 customMessage: "Successfully registered",
                 name: "registered_user",
```

Gambar 5.2 Potongan Kode Registrasi

```
2 usages   EdgarG4m3r *
public static User register(String email, char[] password, String firstName, String lastName, String phoneNumber) throws SQLExcep
{
    try(Connection connection = Artisly.instance.getMySQL().getConnection())
    {
        Optional<User> optionalUser = CRUDSUser.readByEmail(connection, email);
        Optional<User> optionalUser2 = CRUDSUser.readByPhoneNumber(connection, phoneNumber);
        if (optionalUser.isPresent())
        {
            throw new EmailTakenException("That email is already taken");
        }
        if (optionalUser2.isPresent())
        {
            throw new PhoneNumberTakenException("That phone number is already taken");
        }

        //Uses Argon2 and encode it with UTF-8
        String hashedPassword = HashEngine.hashPassword(password);
        EmailService.queueEmail(
                email,
                subject: "Registration Successful",
                body: "Account registration successful, you will receive a verification code shortly!"
        );
        return CRUDSUser.create(connection, email, hashedPassword, firstName, lastName, phoneNumber);
    }
}
```

Gambar 5.3 Potongan Kode Registrasi(2)

Gambar 5.4 adalah tampilan dari antarmuka halaman verifikasi *email* yang digunakan untuk melakukan verifikasi *email* saat melakukan registrasi. Gambar 5.5 adalah potongan kode *presentation layer* dari operasi verifikasi *email*. Gambar 5.6 merupakan potongan kode *service layer* dari operasi verifikasi *email.* Saat pengguna melakukan verifikasi alamat *email*, sistem akan melakukan pengecekan tujuan dari verifikasi. Ada dua jenis verifikasi berdasarkan tujuannya yaitu verifikasi *email* pertama kali saat proses registrasi dan yang kedua adalah verifiaksi *email* saat pengguna mencoba *login* dari *IP Address* yang belum dikenal oleh sistem sebagai *IP Address trusted* milik pengguna. Seluruh kode verifikasi disimpan dalam Redis untuk mempercepat performa sistem dan mengurangi *traffic* ke *database* MySQL yang memiliki performa jauh lebih rendah dibandingkan dengan Redis. Selain performa, Redis juga memiliki fungsi *expire* yang sangat cocok digunakan untuk menyimpan informasi yang memiliki waktu valid, seperti kode verifikasi yang diharapkan hanya dapat digunakan sampai dengan 15 menit setelah dibuat oleh sistem.



Gambar 5.4 Halaman Verifikasi *Email*

```java
public void handle(Context context) {
    InputFilter.validateEmail( field: "email", ParamField.QUERY, context);
    InputFilter.validateVerificationCode( field: "code", ParamField.QUERY, context);

    if (context.attribute( key: "hasErrors") != null) {
        StandarizedResponses.invalidParameter(context);
        return;
    }

    String email = context.queryParam( key: "email");
    String code = context.queryParam( key: "code");

    try {
        boolean result = UserService.verifyEmail(email, code);
        if (result)
        {
            StandarizedResponses.success(
                    context,
                     message: "EMAIL_VERIFIED",
                     customMessage: "Email verified, Silahkan login kembali"
            );
        }
        else
        {
            context.redirect( s: "https://artisly.net/login", HttpStatus.TEMPORARY_REDIRECT);
        }
    } catch (UserNotFoundException e) {
        StandarizedResponses.generalFailure(context,  code: 404,  message: "USER_NOT_FOUND_EXCEPTION", e.getMessage());
    } catch (EmailAlreadyVerified e) {
        StandarizedResponses.generalFailure(context,  code: 409,  message: "EMAIL_ALREADY_VERIFIED", e.getMessage());
    } catch (SQLException e) {
        e.printStackTrace();
        StandarizedResponses.generalFailure(context,  code: 500,  message: "SQL_EXCEPTION",  friendlyMessage: "Terjadi kesalahan saat mengirim kode verifikasi,
    } catch (InvalidVerificationCodeException e) {
        StandarizedResponses.generalFailure(context,  code: 401,  message: "INVALID_VERIFICATION_CODE_EXCEPTION", e.getMessage());
    }
}
```

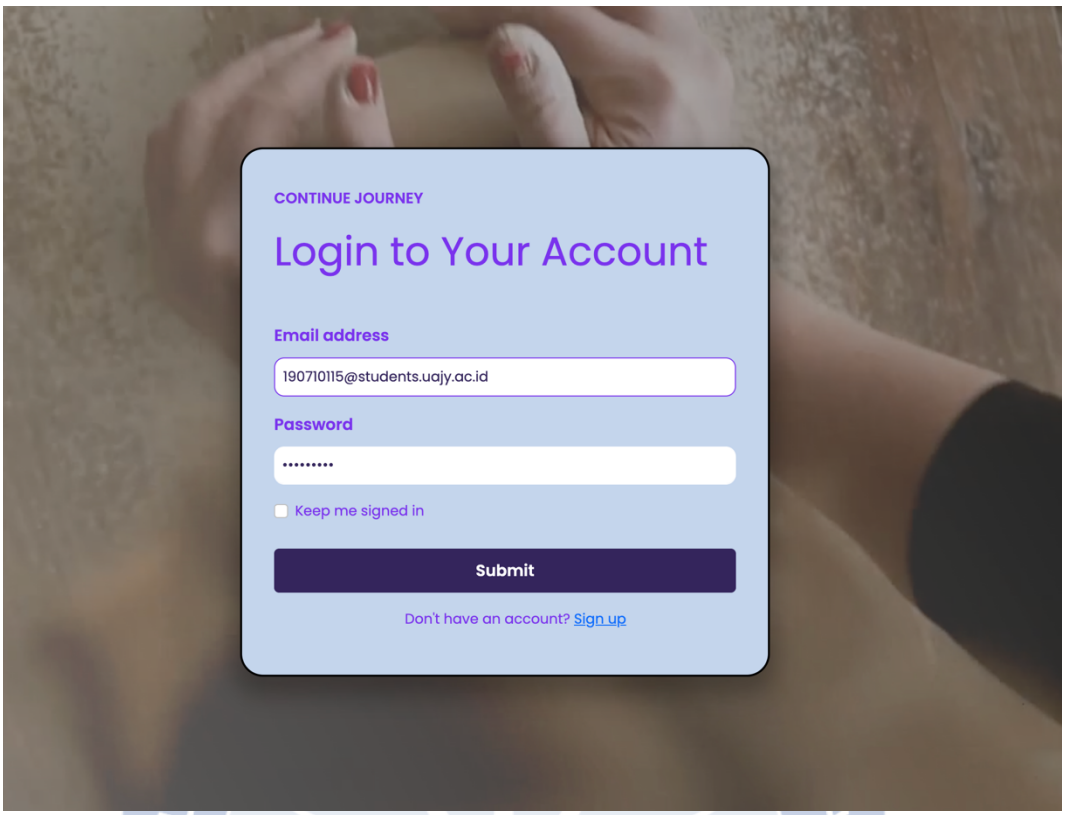
Gambar 5.5 Potongan Kode Verifikasi *Email*

```java
public static boolean verifyEmail(String email, String code) throws SQLException, InvalidVerificationCodeException, UserNotFoundException,
{
    try(Jedis jedis = Artisly.instance.getRedis().getJedis().getResource())
    {

        if (jedis.exists( key: "user_ip_awaiting_verification:" + code))
        {
            String emailIp = jedis.get("user_ip_awaiting_verification:" + code);
            String emailInRedis = emailIp.split( regex: ":")[0];
            String ip = emailIp.split( regex: ":")[1];

            if (!emailInRedis.equalsIgnoreCase(email))
            {
                throw new InvalidVerificationCodeException("Invalid verification code, if you haven't received a code yet, please request one");
            }

            jedis.del( key: "user_ip_awaiting_verification:" + code);
            try(Connection connection = Artisly.instance.getMySQL().getConnection())
            {
                Optional<User> optionalUser = CRUDSUser.readByEmail(connection, email);
                if (optionalUser.isEmpty())
                {
                    throw new UserNotFoundException("We couldn't find a user with that email");
                }
                User user = optionalUser.get();

                jedis.set("user_ip:" + user.id() + ":" + ip, "true");
                jedis.expire( key: "user_ip:" + user.id() + ":" + ip,  seconds: 60 * 60 * 24 * 30);

                EmailService.queueEmail(
                        user.email(),
                         subject: "Trusted IP Added",
                         body: "IP " + ip + " has been added to your trusted IP list for 30 days"
                );
            }
            return false;
        }
        else
        {
            if (!jedis.exists( key: "verificationCode:" + email))
```

Gambar 5.6 Potongan Kode Verifikasi *Email*(2)

Gambar 5.7 adalah tampilan dari antarmuka halaman *login* yang digunakan untuk melakukan masuk ke dalam sistem. Potongan kode *service layer* yang memproses aksi *login* dari pengguna terdapat Gambar 5.8. Fungsi *login* pada sistem ini merupakan fungsi yang paling penting dalam keamanan dari sistem. Fungsi ini yang bertanggung jawab terhadap *authorization* seluruh pengguna *platform*. Jika fungsi ini tidak dapat berfungsi seperti yang diharapkan, maka keamanan dari sistem akan terancam. Oleh karena itu, fungsi *login* juga memiliki sistem *rate limiting* yang memblokir percobaan *login* akun tertentu selama 1 menit jika pengguna memasukkan kata sandi yang salah lebih dari 5 kali dalam satu menit.

Fungsi *rate limiting* dan penggunaan Argon2 adalah strategi keamanan sistem untuk melindungi dari serangan *bruteforce*. Selain itu, fungsi *login* akan meminta pengguna untuk melakukan verifikasi *email* dengan mengakses *link* yang dikirimkan kepada pengguna melalui alamat *email* jika tindakan *login* berhasil tetapi dilakukan menggunakan *IP Address* yang belum pernah digunakan oleh pengguna. Setelah selesai melakukan verifikasi *email* lagi, *IP Address* tersebut akan di *whitelist* oleh sistem selama 30 hari agar pengguna tidak perlu melakukan verifikasi *email* pada *IP Address* tersebut. Selain itu, setiap kali ada tindakan *login* yang berhasil dan verifikasi *email* yang berhasil, sistem akan mengirimkan *email* pemberitahuan *login* berhasil dan *email* pemberitahuan bahwa *IP Address* tertentu berhasil ditambahkan ke *whitelist* pengguna. *IP Address* yang sudah di *whitelist* hanya akan berlaku pada akun pengguna tertentu saja, sehingga *IP Address* yang sudah berhasil digunakan pada akun pengguna lain tidak dapat digunakan di akun lainnya.

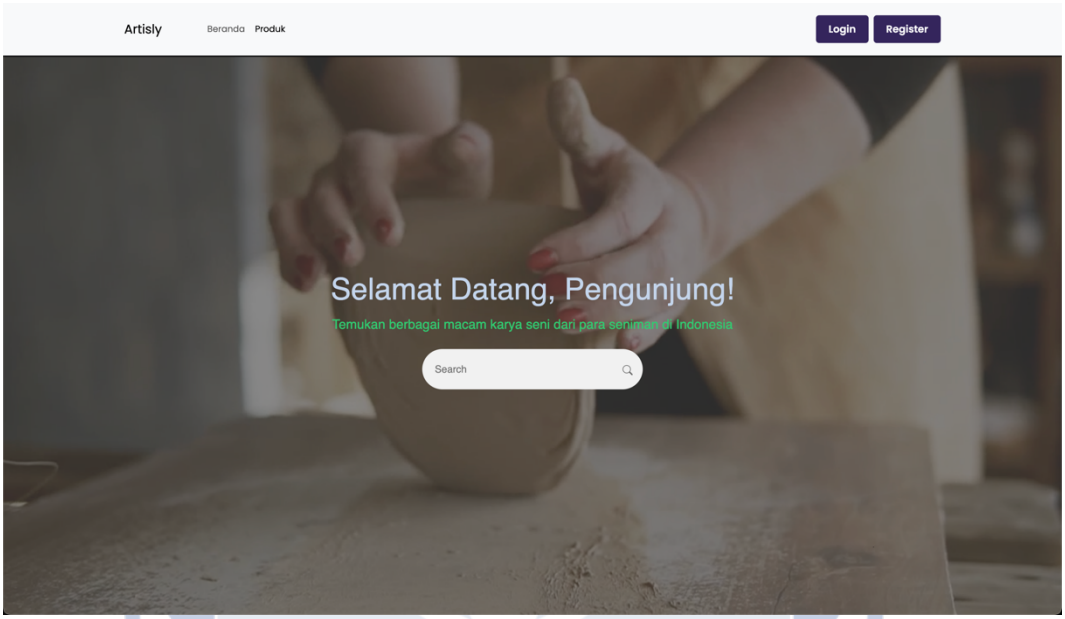


Gambar 5.7 Halaman *Login*

```
User user = optionalUser.get();
if (user.banned())
{
    throw new UserBannedException("Akun anda telah dibanned!");
}

try(Jedis jedis = Artisly.instance.getRedis().getJedis().getResource())
{
    int attempt;
    String key = "login_attempt:" + user.id();
    String attempts = jedis.get(key);
    if (attempts == null)
    {
        attempt = 1;
    }
    else
    {
        attempt = Integer.parseInt(attempts);
    }

    if (attempt >= 5)
    {
        throw new UserBannedException("Percobaan login diblokir sementara! Silahkan coba lagi dalam 1 menit.");
    }
}
if (!HashEngine.verifyPassword(user.password(), password))
{
    try(Jedis jedis = Artisly.instance.getRedis().getJedis().getResource())
    {
        String key = "login_attempt:" + user.id();
        String attempts = jedis.get(key);
        if (attempts == null)
        {
            jedis.set(key, "1");
            jedis.expire(key, seconds: 60);
        }
        else
        {
            int attemptCount = Integer.parseInt(attempts);
            if (attemptCount >= 5)
            {
                throw new UserBannedException("Percobaan login diblokir sementara! Silahkan coba lagi dalam 1 menit.");
```

Gambar 5.8 Potongan Kode *Login*

Gambar 5.9 adalah tampilan dari antarmuka halaman beranda yang mengandung *form* pencarian. Halaman ini hanya digunakan sebagai *landing page* dari sistem dan tidak memiliki logika di sisi *backend*. Formulir pencarian berjalan di *client-side* yang mengarahkan pengguna ke halaman produk ditambah dengan kata kunci yang dimasukkan oleh pengguna.



Gambar 5.9 Halaman Beranda

Gambar 5.10 adalah tampilan dari antarmuka halaman profil yang mengandung *form* perubahan profil pengguna dan *form* penggantian kata sandi. Gambar 5.11 adalah potongan kode dari *presentation layer* yang dihubungi oleh *frontend* untuk mengambil informasi profil pengguna.

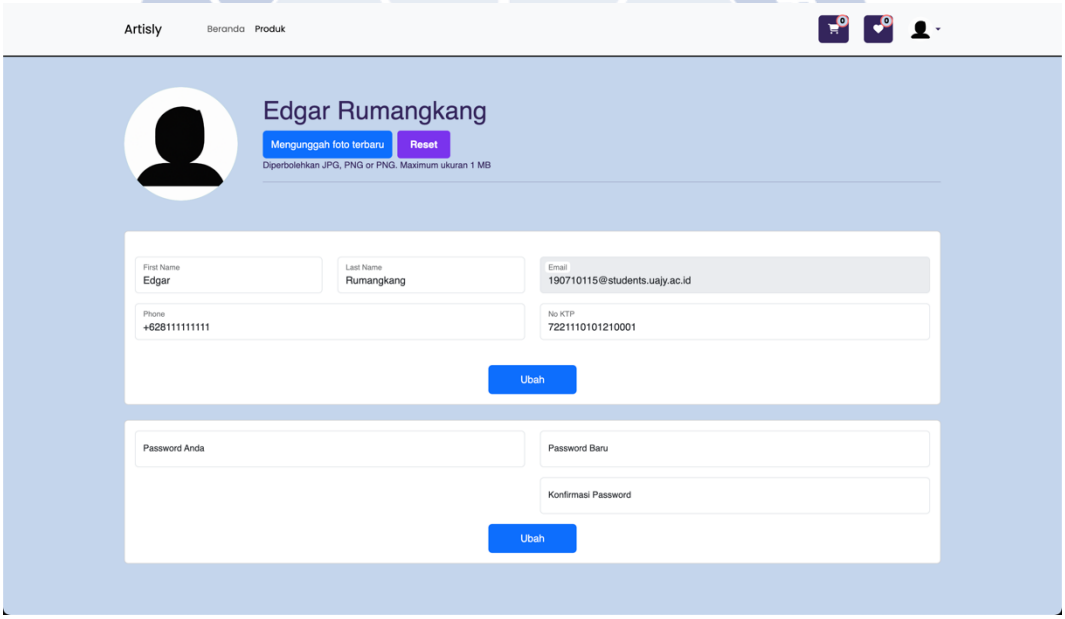
Gambar 5.12 adalah gambar potongan kode *presentation layer* yang menerima permintaan perubahan profil pengguna yang kemudian diteruskan kepada *service layer* pada Gambar 5.13 yang kemudian memanggil fungsi pada *data layer* untuk menyimpan data ke dalam *database* seperti pada potongan kode yang terdapat pada Gambar 5.14.

Gambar 5.12 menunjukan beberapa komponen dari sistem seperti penggunaan UserSessionContainer yang merupakan bagian penting dari sistem *authorization*. Setiap kali pengguna berhasil login, maka sistem akan membuat *token* yang kemudian disimpan di dalam *database* Redis dan juga disimpan di *cookie* pengguna. Setiap kali *backend* ingin melakukan *authorization* seperti saat pengguna mengakses *API Endpoint* yang dilindungi, maka *backend* akan mengambil *object* UserSessionContainer yang didapatkan dari Redis yang berisi dua *object* yaitu User yang merupakan *object* yang mengandung informasi pengguna, dan *token* pengguna. Jika terjadi masalah saat sistem melakukan pemrosesan token untuk proses *authorization*, maka UserSessionContainer akan menjadi *null* sehingga *backend* dapat berhenti memproses permintaan.

Gambar 5.13 adalah *service layer* dari perubahan profil pengguna, dalam hal ini, *service layer* melakukan beberapa validasi seperti keberadaan pengguna di dalam *database*, dan yang terpenting adalah sistem akan mencegah pengguna menghapus nomor KTP dari sistem. Selain itu, sistem akan meminta kata sandi pengguna setiap perubahan profil sebagai salah satu strategi keamanan untuk mencegah serangan *Cross-Site Request Forgery (CSRF).*

Gambar 5.15 adalah gambar potongan kode dari *presentation layer* yang menerima permintaan perubahan kata sandi pengguna yang kemudian diteruskan kepada *service layer* yang ada pada Gambar 5.16.

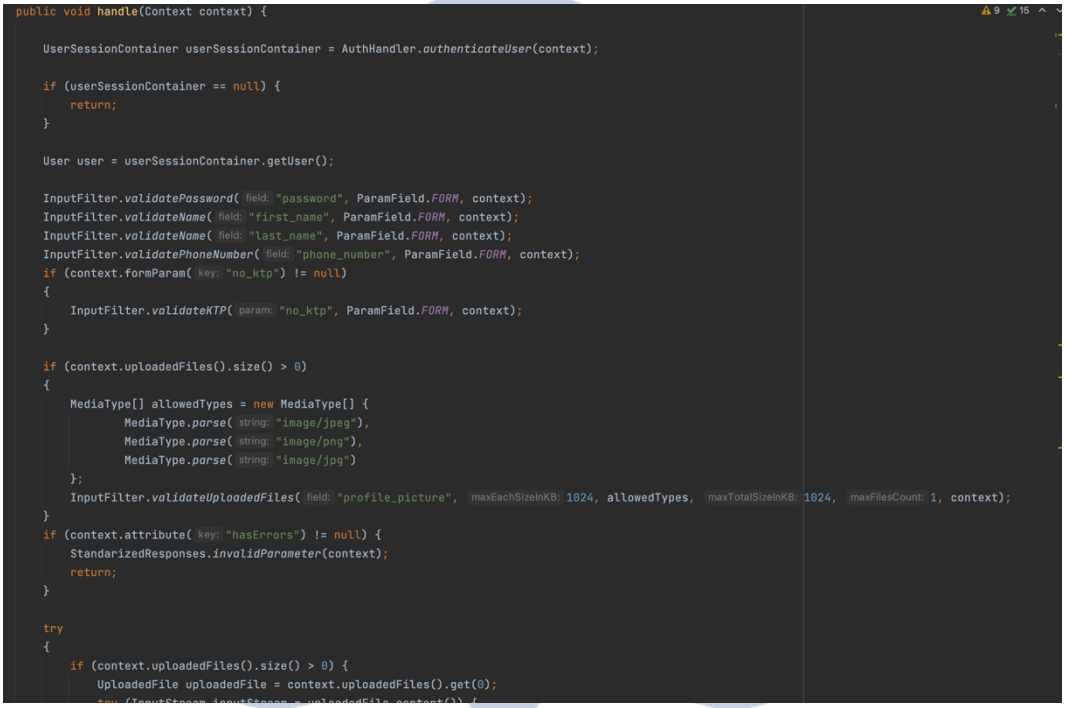
Gambar 5.16 menunjukkan *service layer* dari penggantian kata sandi. Sistem menggunakan Argon2 sebagai algoritma yang digunakan untuk mengenkripsi kata sandi pengguna. Argon2 dipilih dalam penelitian ini karena Argon2 memiliki beberapa kelebihan dari algoritma *password-hashing* lainnya seperti kompleksitas waktu yang dapat diatur, kompleksitas *memory* yang dapat diatur, dan derajat paralelisme yang dapat diatur. Dalam penelitian ini, konfigurasi Argon2 membuat setiap operasi *hashing* memakan waktu kira-kira 230 milidetik sampai dengan 280 milidetik. Waktu tersebut tidak terlalu lama untuk membuat pengguna tidak nyaman, tapi sangat signifikan untuk melindungi kata sandi pengguna dari serangan *offline brute-force* karena setiap percobaan kombinasi membutuhkan kira-kira 230 milidetik sampai dengan 280 milidetik yang akan memperlambat serangan secara signifikan. Selain itu, Argon2 memiliki *salt* yang mencegah serangan *rainbow attack*.



Gambar 5.10 Halaman Profil

```
2 usages  EdgarG4m3r
public class Profile implements APIHandler {

    EdgarG4m3r
    @Override
    public void handle(Context context) {
        UserSessionContainer userSessionContainer = AuthHandler.authenticateUser(context);
        if (userSessionContainer == null) {
            return;
        }

        User user = userSessionContainer.getUser();
        JSONObject userJSON = user.toJSON();
        StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Berhasil mendapatkan profil.", name: "user", userJSON);

    }
}
```

Gambar 5.11 Potongan Kode Mengambil Profil



```
public void handle(Context context) {

    UserSessionContainer userSessionContainer = AuthHandler.authenticateUser(context);

    if (userSessionContainer == null) {
        return;
    }

    User user = userSessionContainer.getUser();

    InputFilter.validatePassword( field: "password", ParamField.FORM, context);
    InputFilter.validateName( field: "first_name", ParamField.FORM, context);
    InputFilter.validateName( field: "last_name", ParamField.FORM, context);
    InputFilter.validatePhoneNumber( field: "phone_number", ParamField.FORM, context);
    if (context.formParam( key: "no_ktp") != null)
    {
        InputFilter.validateKTP( param: "no_ktp", ParamField.FORM, context);
    }

    if (context.uploadedFiles().size() > 0)
    {
        MediaType[] allowedTypes = new MediaType[] {
                MediaType.parse( string: "image/jpeg"),
                MediaType.parse( string: "image/png"),
                MediaType.parse( string: "image/jpg")
        };
        InputFilter.validateUploadedFiles( field: "profile_picture", maxEachSizeInKB: 1024, allowedTypes, maxTotalSizeInKB: 1024, maxFilesCount: 1, context);
    }
    if (context.attribute( key: "hasErrors") != null) {
        StandarizedResponses.invalidParameter(context);
        return;
    }

    try
    {
        if (context.uploadedFiles().size() > 0) {
            UploadedFile uploadedFile = context.uploadedFiles().get(0);
```

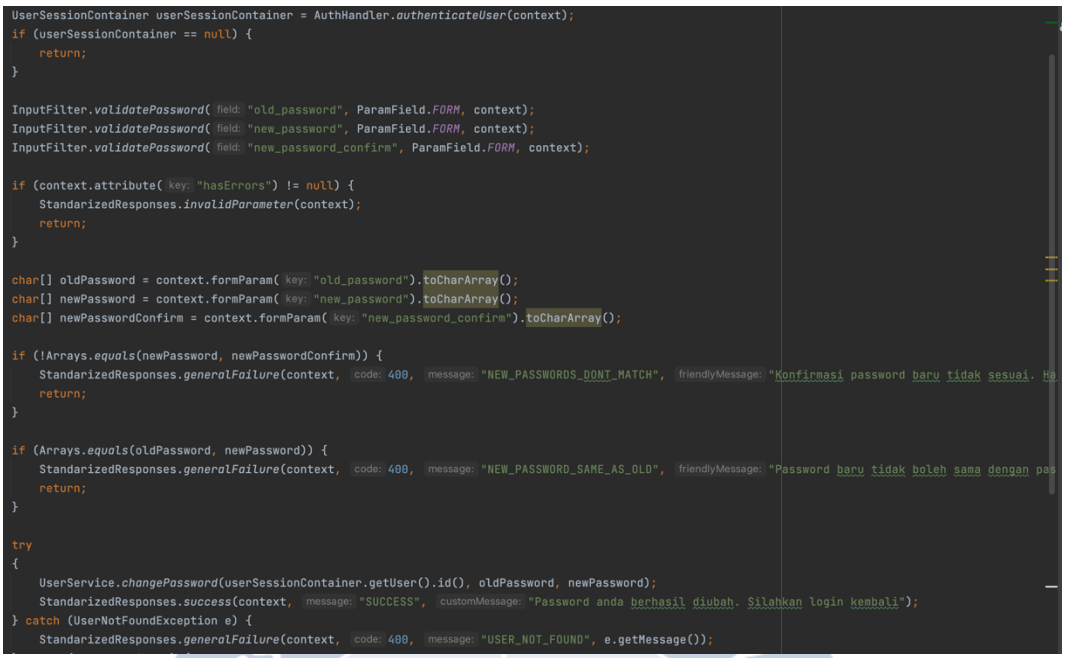
Gambar 5.12 Potongan Kode Perubahan Profil

```
 * @throws UserNotFoundException
 * @throws InvalidCredentialsException
 * @throws IOException
 */
2 usages   EdgarG4m3r
public static boolean changeProfile(UUID userId, char[] password, Optional<String> firstName, Optional<String> lastName, Optional<String> phoneNumber, Op
{
    try(Connection connection = Artisly.instance.getMySQL().getConnection())
    {
        Optional<User> optionalUser = CRUDSUser.readById(connection, userId);
        if (optionalUser.isEmpty())
        {
            throw new UserNotFoundException("We couldn't find a user with that email");
        }
        User user = optionalUser.get();
        if (!user.nomorKTP().isEmpty())
        {
            if (noKTP.isEmpty())
            {
                throw new AccountNotVerifiedException("No KTP tidak dapat dihapus");
            }
        }
        if (!HashEngine.verifyPassword(user.password(), password))
        {
            throw new InvalidCredentialsException("Invalid password");
        }
        if (profilePicture.isPresent())
        {
            Artisly.instance.getMediaService().uploadProfilePicture(userId, profilePicture.get(), new Tika().detect(profilePicture.get()));
        }
        return CRUDSUser.update(connection, user.id(), user.email(), user.password(), firstName.orElse(user.firstName()), lastName.orElse(user.lastName())
    }
}
```

Gambar 5.13 Potongan Kode Perubahan Profil(2)

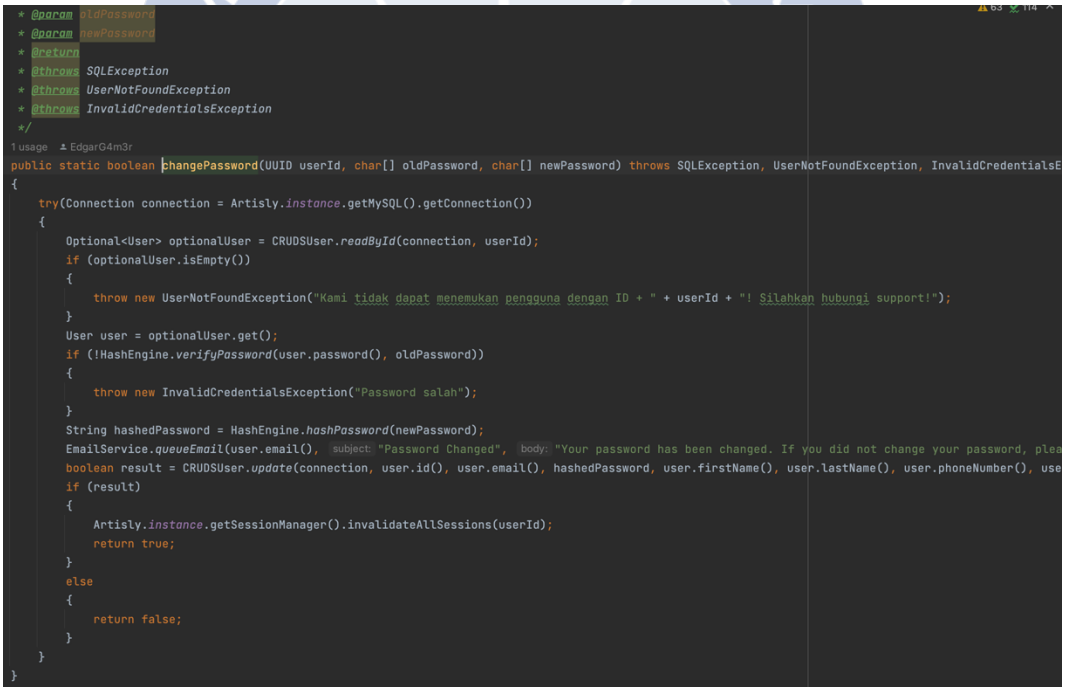
```
 EdgarG4m3r *
public static boolean update(Connection connection, UUID userId, String userEmail, String userPassword, String userFirstName, String userLastName, Strin
    String query = "UPDATE users SET user_email = ?, user_password = ?, user_first_name = ?, user_last_name = ?, user_phone_number = ?, user_banned = ?,
    try(PreparedStatement ps = connection.prepareStatement(query)) {
        ps.setString( parameterIndex: 1, userEmail);
        ps.setString( parameterIndex: 2, userPassword);
        ps.setString( parameterIndex: 3, userFirstName);
        ps.setString( parameterIndex: 4, userLastName);
        ps.setString( parameterIndex: 5, userPhoneNumber);
        ps.setBoolean( parameterIndex: 6, userBanned);
        ps.setDate( parameterIndex: 7, Date.valueOf(userCreated));
        ps.setDate( parameterIndex: 8, Date.valueOf(userUpdated));
        if(userEmailVerified.isPresent()) {
            ps.setDate( parameterIndex: 9, Date.valueOf(userEmailVerified.get()));
        } else {
            ps.setNull( parameterIndex: 9, Types.DATE);
        }
        ps.setString( parameterIndex: 10, nomorKTP.orElse( other: null));
        ps.setBoolean( parameterIndex: 11, admin);
        ps.setString( parameterIndex: 12, userId.toString());
        removeFromCache(new User(userId, userEmail, userPassword, userFirstName, userLastName, userPhoneNumber, userBanned, userCreated, userUpdated, us
        return ps.executeUpdate() > 0;
    }
}
```

Gambar 5.14 Potongan Kode Perubahan Profil(3)



```
UserSessionContainer userSessionContainer = AuthHandler.authenticateUser(context);
if (userSessionContainer == null) {
    return;
}

InputFilter.validatePassword( field: "old_password", ParamField.FORM, context);
InputFilter.validatePassword( field: "new_password", ParamField.FORM, context);
InputFilter.validatePassword( field: "new_password_confirm", ParamField.FORM, context);

if (context.attribute( key: "hasErrors") != null) {
    StandarizedResponses.invalidParameter(context);
    return;
}

char[] oldPassword = context.formParam( key: "old_password").toCharArray();
char[] newPassword = context.formParam( key: "new_password").toCharArray();
char[] newPasswordConfirm = context.formParam( key: "new_password_confirm").toCharArray();

if (!Arrays.equals(newPassword, newPasswordConfirm)) {
    StandarizedResponses.generalFailure(context,  code: 400,  message: "NEW_PASSWORDS_DONT_MATCH",  friendlyMessage: "Konfirmasi password baru tidak sesuai. Ha
    return;
}

if (Arrays.equals(oldPassword, newPassword)) {
    StandarizedResponses.generalFailure(context,  code: 400,  message: "NEW_PASSWORD_SAME_AS_OLD",  friendlyMessage: "Password baru tidak boleh sama dengan pas
    return;
}

try
{
    UserService.changePassword(userSessionContainer.getUser().id(), oldPassword, newPassword);
    StandarizedResponses.success(context,  message: "SUCCESS",  customMessage: "Password anda berhasil diubah. Silahkan login kembali");
} catch (UserNotFoundException e) {
    StandarizedResponses.generalFailure(context,  code: 400,  message: "USER_NOT_FOUND", e.getMessage());
```

Gambar 5.15 Potongan Kode Penggantian Kata Sandi

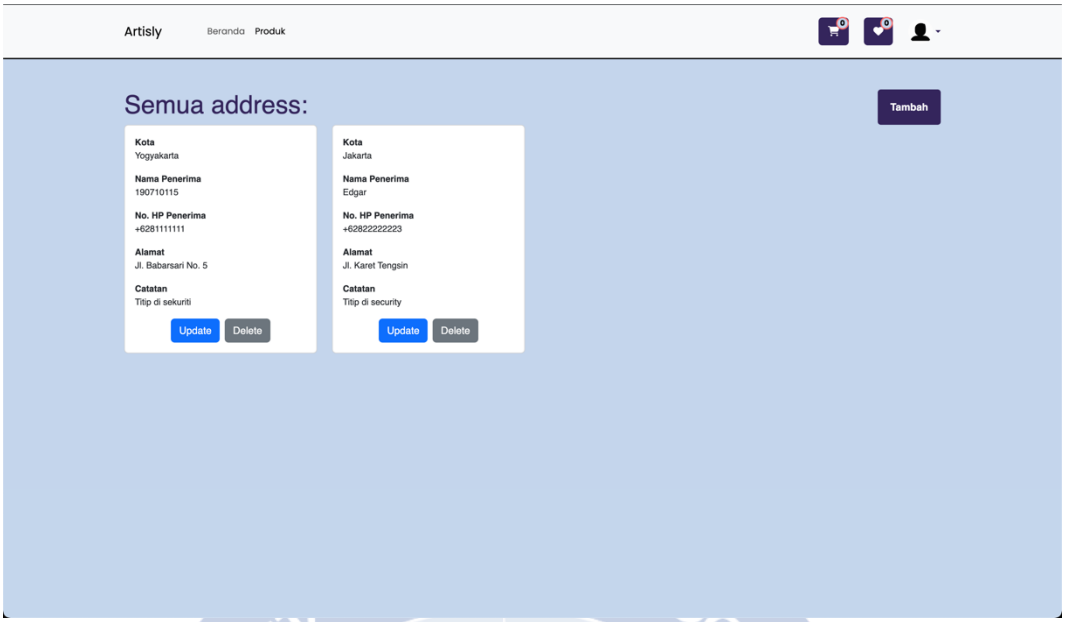


```
 * @param oldPassword
 * @param newPassword
 * @return
 * @throws SQLException
 * @throws UserNotFoundException
 * @throws InvalidCredentialsException
 */
1 usage   EdgarG4m3r
public static boolean changePassword(UUID userId, char[] oldPassword, char[] newPassword) throws SQLException, UserNotFoundException, InvalidCredentialsE
{
    try(Connection connection = Artisly.instance.getMySQL().getConnection())
    {
        Optional<User> optionalUser = CRUDSUser.readById(connection, userId);
        if (optionalUser.isEmpty())
        {
            throw new UserNotFoundException("Kami tidak dapat menemukan pengguna dengan ID + " + userId + "! Silahkan hubungi support!");
        }
        User user = optionalUser.get();
        if (!HashEngine.verifyPassword(user.password(), oldPassword))
        {
            throw new InvalidCredentialsException("Password salah");
        }
        String hashedPassword = HashEngine.hashPassword(newPassword);
        EmailService.queueEmail(user.email(),  subject: "Password Changed",  body: "Your password has been changed. If you did not change your password, plea
        boolean result = CRUDSUser.update(connection, user.id(), user.email(), hashedPassword, user.firstName(), user.lastName(), user.phoneNumber(), use
        if (result)
        {
            Artisly.instance.getSessionManager().invalidateAllSessions(userId);
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

Gambar 5.16 Potongan Kode Penggantian Kata Sandi(2)

Gambar 5.17 adalah tampilan dari antarmuka halaman alamat yang menunjukkan seluruh alamat pengguna dan mengandung *form* penambahan dan perubahan alamat. Gambar 5.18 adalah potongan kode dari *presentation layer* untuk mengambil daftar alamat, Gambar 5.19 adalah potongan kode yang menjalankan permintaan pembuatan alamat, Gambar 5.20 adalah potongan kode yang menjalankan permintaan perubahan alamat, Gambar 5.21 adalah potongan kode yang menerima permintaan untuk menghapus alamat. Gambar 5.22 adalah implementasi antarmuka dari *form* pembuatan dan perubahan alamat. Fitur penyimpanan alamat merupakan salah satu strategi sistem untuk meningkatkan kenyamanan pengguna agar pengguna tidak perlu mengetikan alamat setiap kali melakukan pesanan yang baru.

Pada Gambar 5.18, terlihat bahwa sistem akan mencoba mencari *enum* kota dari data yang tersimpan dari dalam *database* tetapi sistem akan meloncati alamat jika tidak dapat menemukan kota. Hal ini akan sangat jarang terjadi dalam *production* karena sistem juga akan memvalidasi kebenaran kota sebelum dimasukkan ke dalam *database*. Saat pembuatan alamat, sistem akan memastikan bahwa pengguna hanya memiliki maksimal 10 alamat seperti terlihat pada Gambar 5.19. Hal ini untuk mencegah pengguna melakukan serangan *flooding* kepada sistem dengan cara menambahkan banyak alamat dengan tujuan untuk mengganggu fungsi normal dari *database*.



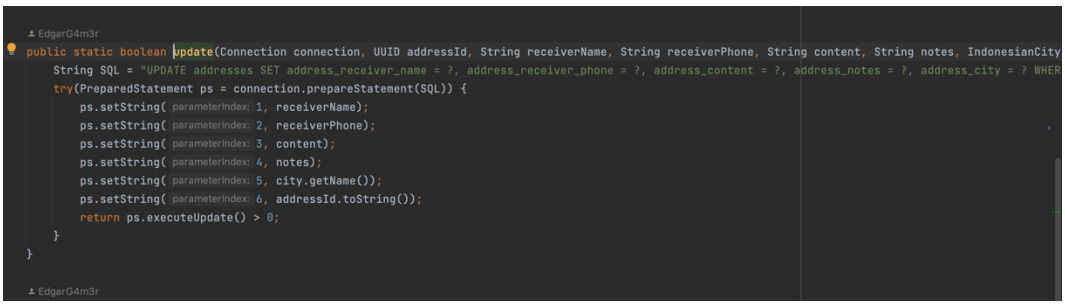
Gambar 5.17 Halaman Alamat

```
public static List<Address> readByUserId(Connection connection, UUID userId) throws SQLException {
    String SQL = "SELECT * FROM addresses WHERE user_id = ?";
    List<Address> addresses = new ArrayList<>();
    try(PreparedStatement ps = connection.prepareStatement(SQL)) {
        ps.setString( parameterIndex: 1, userId.toString());
        try(var resultSet ResultSet = ps.executeQuery()) {
            while (resultSet.next()) {
                IndonesianCity city;
                try {
                    city = IndonesianCity.valueOf(resultSet.getString( columnLabel: "address_city").toUpperCase().replaceAll( regex: " ", replacement: "_"));
                } catch (Exception e) {
                    continue;
                }
                Address address = new Address(
                        UUID.fromString(resultSet.getString( columnLabel: "address_id")),
                        UUID.fromString(resultSet.getString( columnLabel: "user_id")),
                        resultSet.getString( columnLabel: "address_receiver_name"),
                        resultSet.getString( columnLabel: "address_receiver_phone"),
                        resultSet.getString( columnLabel: "address_content"),
                        resultSet.getString( columnLabel: "address_notes"),
                        city
                );
                addresses.add(address);
            }
        }
    }
    return addresses;
}
```

Gambar 5.18 Potongan Kode Pengambilan Daftar Alamat

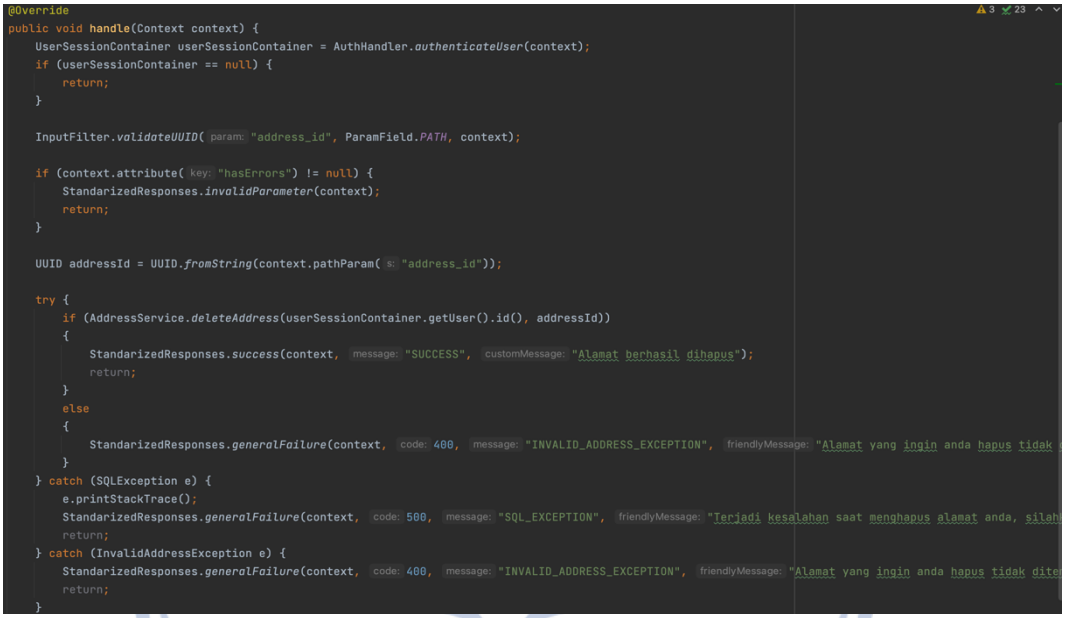
```
private static final int MAX_ADDRESS_COUNT = 10;

1 usage  EdgarG4m3r
public static Address createAddress(UUID userId, String receiverName, String receiverPhone, String content, String note, IndonesianCity city) throws SQLExc
    try(Connection connection = Artisly.instance.getMySQL().getConnection())
    {
        List<Address> addresses = CRUDSAddress.readByUserId(connection, userId);
        if (addresses.size() >= MAX_ADDRESS_COUNT) {
            throw new AddressMaximumCountException("Anda hanya dapat memiliki maksimal " + MAX_ADDRESS_COUNT + " alamat");
        }

        Address address = CRUDSAddress.create(connection, userId, receiverName, receiverPhone, content, note, city);
        return address;
    }
}
```

Gambar 5.19 Potongan Kode Pembuatan Alamat



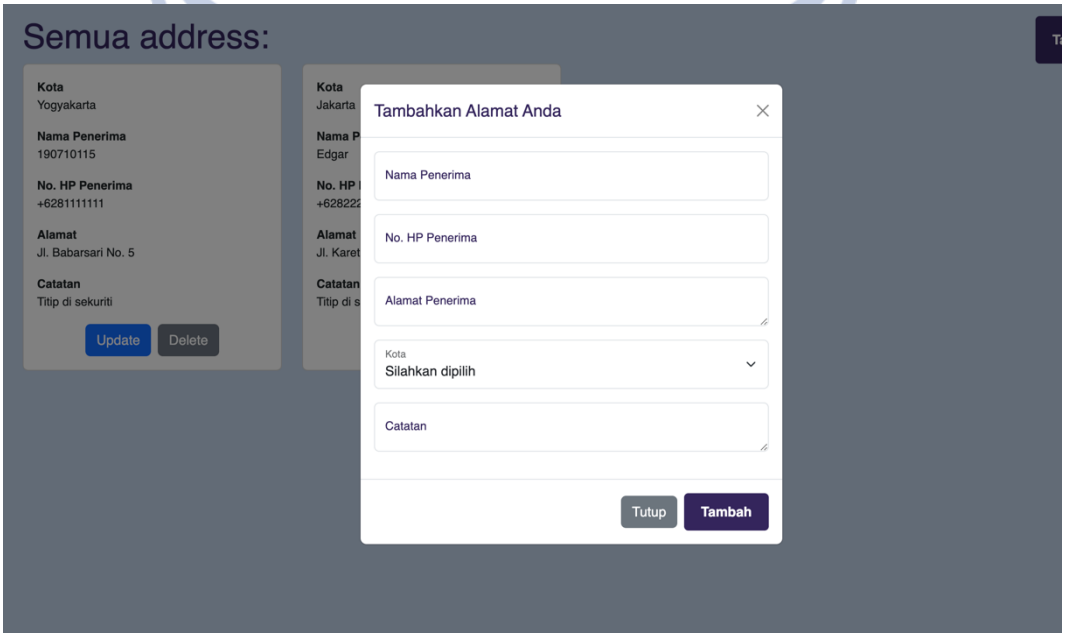
```
EdgarG4m3r
public static boolean update(Connection connection, UUID addressId, String receiverName, String receiverPhone, String content, String notes, IndonesianCity
    String SQL = "UPDATE addresses SET address_receiver_name = ?, address_receiver_phone = ?, address_content = ?, address_notes = ?, address_city = ? WHER
    try(PreparedStatement ps = connection.prepareStatement(SQL)) {
        ps.setString( parameterIndex: 1, receiverName);
        ps.setString( parameterIndex: 2, receiverPhone);
        ps.setString( parameterIndex: 3, content);
        ps.setString( parameterIndex: 4, notes);
        ps.setString( parameterIndex: 5, city.getName());
        ps.setString( parameterIndex: 6, addressId.toString());
        return ps.executeUpdate() > 0;
    }
}
EdgarG4m3r
```

Gambar 5.20 Potongan Kode Perubahan Alamat



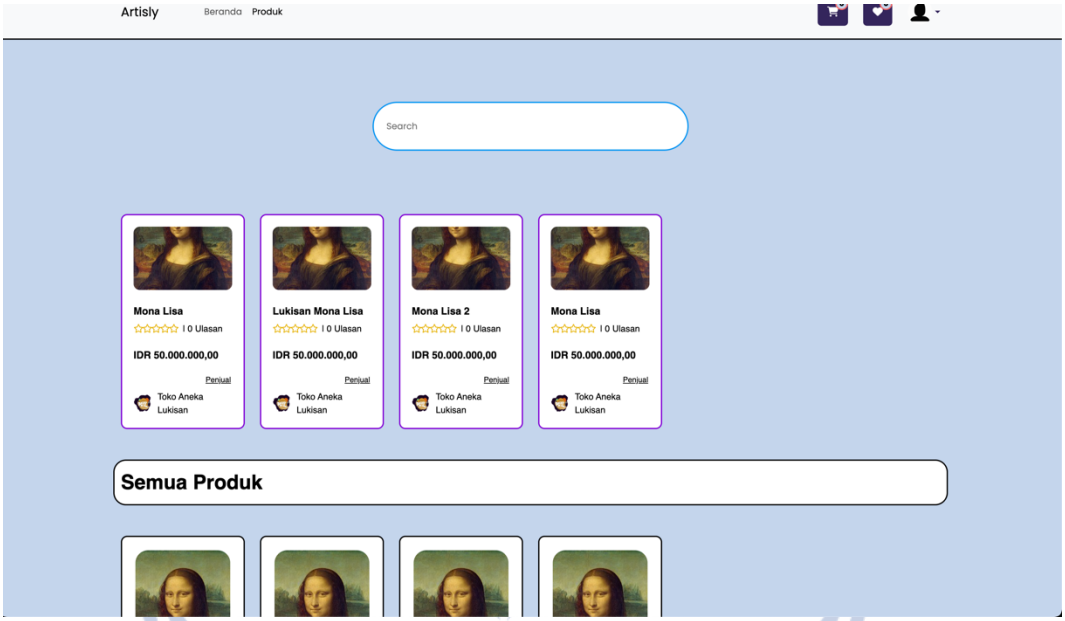
```
@Override
public void handle(Context context) {
    UserSessionContainer userSessionContainer = AuthHandler.authenticateUser(context);
    if (userSessionContainer == null) {
        return;
    }

    InputFilter.validateUUID( param: "address_id", ParamField.PATH, context);

    if (context.attribute( key: "hasErrors") != null) {
        StandarizedResponses.invalidParameter(context);
        return;
    }

    UUID addressId = UUID.fromString(context.pathParam( s: "address_id"));

    try {
        if (AddressService.deleteAddress(userSessionContainer.getUser().id(), addressId))
        {
            StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Alamat berhasil dihapus");
            return;
        }
        else
        {
            StandarizedResponses.generalFailure(context, code: 400, message: "INVALID_ADDRESS_EXCEPTION", friendlyMessage: "Alamat yang ingin anda hapus tidak
        }
    } catch (SQLException e) {
        e.printStackTrace();
        StandarizedResponses.generalFailure(context, code: 500, message: "SQL_EXCEPTION", friendlyMessage: "Terjadi kesalahan saat menghapus alamat anda, silah
        return;
    } catch (InvalidAddressException e) {
        StandarizedResponses.generalFailure(context, code: 400, message: "INVALID_ADDRESS_EXCEPTION", friendlyMessage: "Alamat yang ingin anda hapus tidak dite
        return;
    }
```

Gambar 5.21 Potongan Kode Penghapusan Alamat

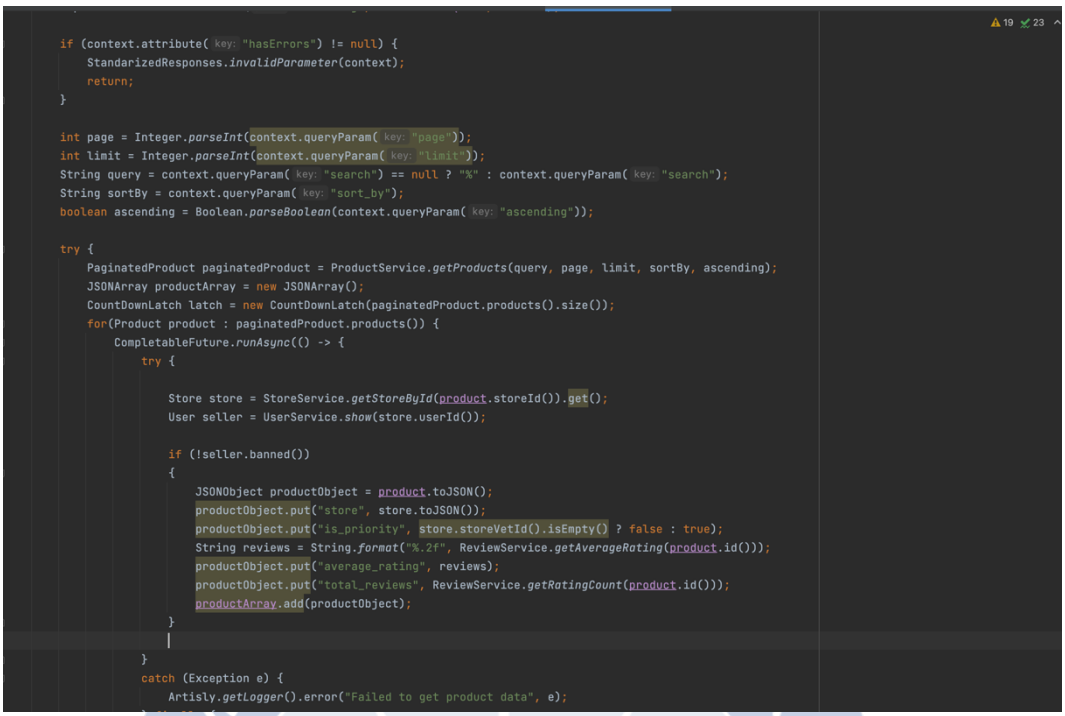


Gambar 5.22 Formulir Penambahan atau Perubahan Alamat

Gambar 5.23 adalah tampilan dari antarmuka halaman produk yang menunjukkan seluruh produk atau produk yang mengandung kata kunci tertentu. Halaman produk akan mengirimkan permintaan ke *backend* sebanyak 2 kali untuk mengambil daftar produk biasa atau campuran dan untuk mengambil daftar produk prioritas yaitu produk dari toko yang sudah "terverifikasi". Gambar 5.25 adalah potongan kode yang memproses pengambilan produk biasa dan Gambar 5.25 adalah potongan kode yang memproses pengambilan produk prioritas.

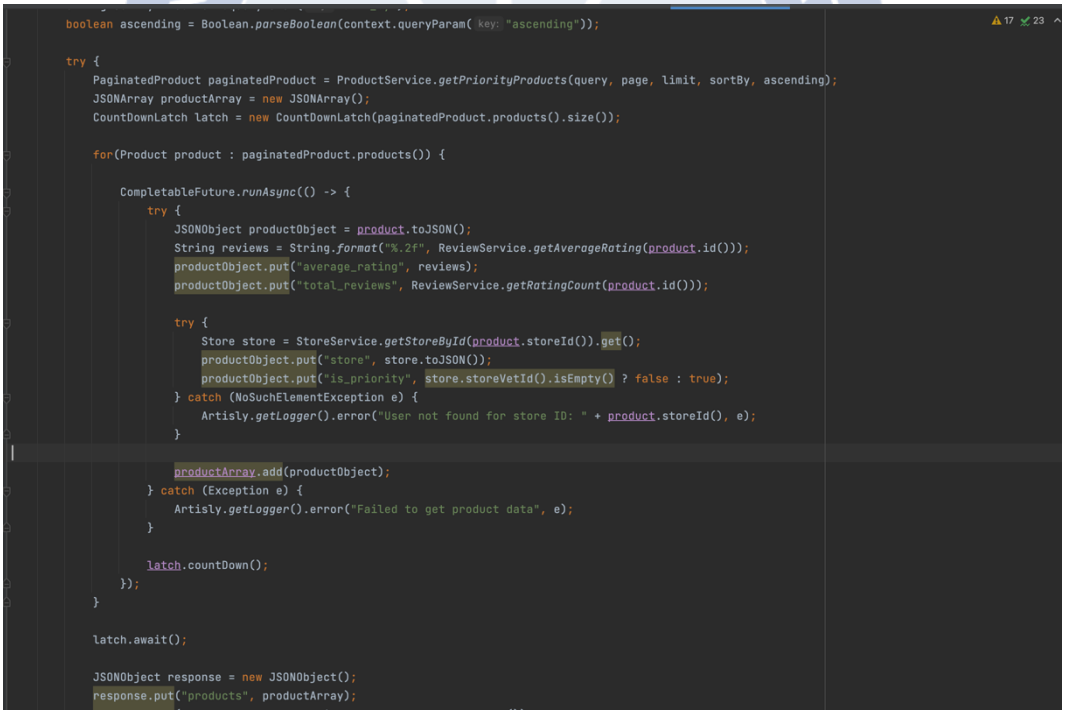


Gambar 5.23 Halaman Produk



```
if (context.attribute( key: "hasErrors") != null) {
    StandarizedResponses.invalidParameter(context);
    return;
}

int page = Integer.parseInt(context.queryParam( key: "page"));
int limit = Integer.parseInt(context.queryParam( key: "limit"));
String query = context.queryParam( key: "search") == null ? "%" : context.queryParam( key: "search");
String sortBy = context.queryParam( key: "sort_by");
boolean ascending = Boolean.parseBoolean(context.queryParam( key: "ascending"));

try {
    PaginatedProduct paginatedProduct = ProductService.getProducts(query, page, limit, sortBy, ascending);
    JSONArray productArray = new JSONArray();
    CountDownLatch latch = new CountDownLatch(paginatedProduct.products().size());
    for(Product product : paginatedProduct.products()) {
        CompletableFuture.runAsync(() -> {
            try {

                Store store = StoreService.getStoreById(product.storeId()).get();
                User seller = UserService.show(store.userId());

                if (!seller.banned())
                {
                    JSONObject productObject = product.toJSON();
                    productObject.put("store", store.toJSON());
                    productObject.put("is_priority", store.storeVetId().isEmpty() ? false : true);
                    String reviews = String.format("%.2f", ReviewService.getAverageRating(product.id()));
                    productObject.put("average_rating", reviews);
                    productObject.put("total_reviews", ReviewService.getRatingCount(product.id()));
                    productArray.add(productObject);
                }
            }
            catch (Exception e) {
                Artisly.getLogger().error("Failed to get product data", e);
```

Gambar 5.24 Potongan Kode Pengambilan Produk Biasa

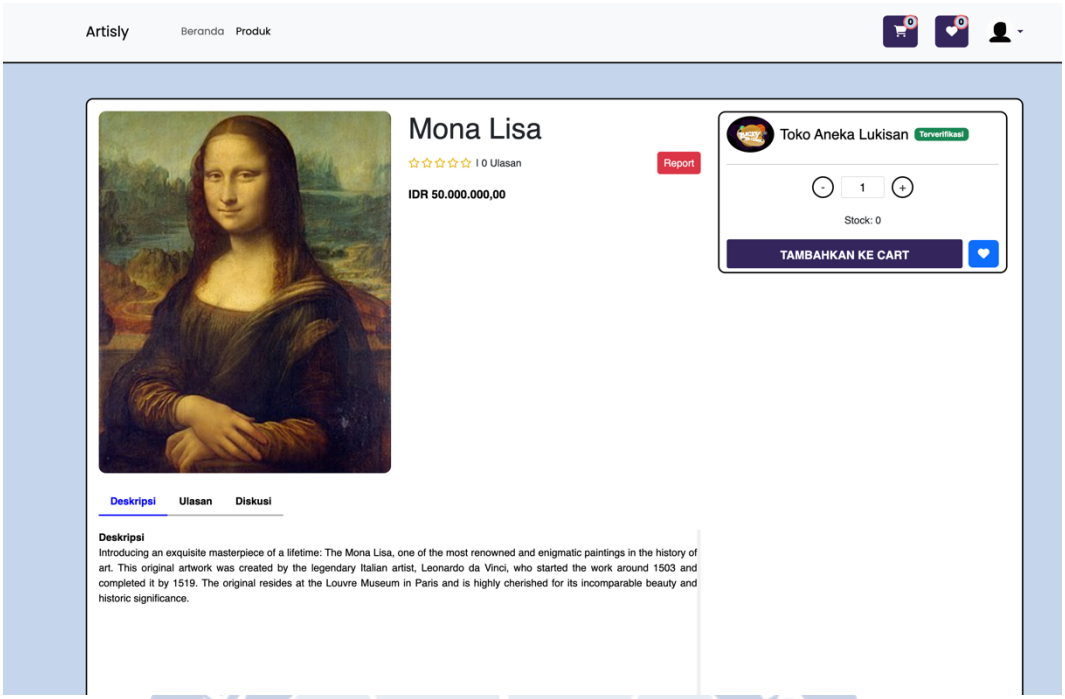


```
boolean ascending = Boolean.parseBoolean(context.queryParam( key: "ascending"));

try {
    PaginatedProduct paginatedProduct = ProductService.getPriorityProducts(query, page, limit, sortBy, ascending);
    JSONArray productArray = new JSONArray();
    CountDownLatch latch = new CountDownLatch(paginatedProduct.products().size());

    for(Product product : paginatedProduct.products()) {

        CompletableFuture.runAsync(() -> {
            try {
                JSONObject productObject = product.toJSON();
                String reviews = String.format("%.2f", ReviewService.getAverageRating(product.id()));
                productObject.put("average_rating", reviews);
                productObject.put("total_reviews", ReviewService.getRatingCount(product.id()));

                try {
                    Store store = StoreService.getStoreById(product.storeId()).get();
                    productObject.put("store", store.toJSON());
                    productObject.put("is_priority", store.storeVetId().isEmpty() ? false : true);
                } catch (NoSuchElementException e) {
                    Artisly.getLogger().error("User not found for store ID: " + product.storeId(), e);
                }

                productArray.add(productObject);
            } catch (Exception e) {
                Artisly.getLogger().error("Failed to get product data", e);
            }

            latch.countDown();
        });
    }

    latch.await();

    JSONObject response = new JSONObject();
    response.put("products", productArray);
```

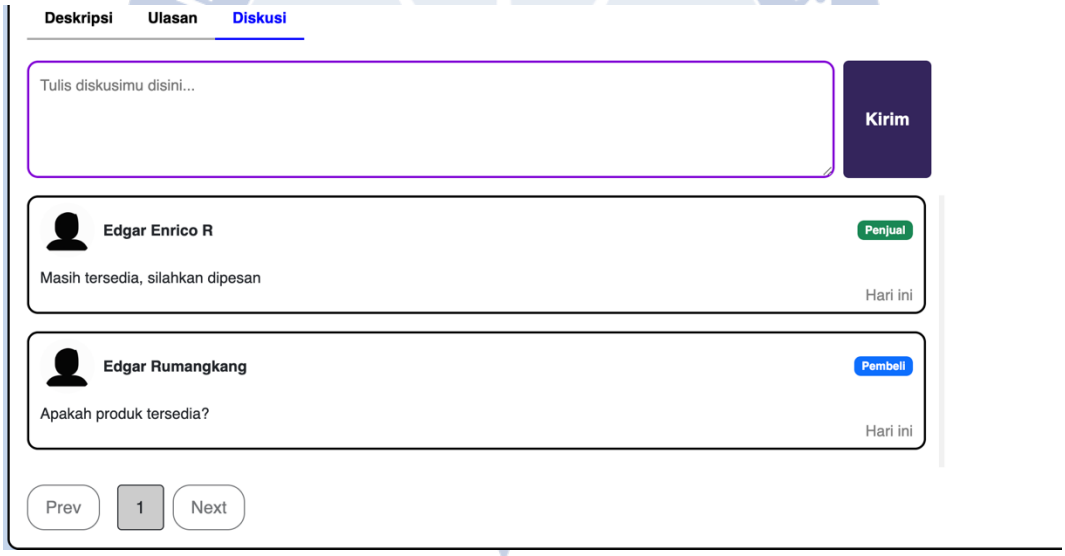
Gambar 5.25 Potongan Kode Pengambilan Produk Prioritas

Potongan kode pengambilan produk biasa dan pengambilan produk prioritas pada Gambar 5.24 dan Gambar 5.25 memiliki kode yang sangat mirip dengan perbedaan di fungsi yang digunakan untuk mendapatkan *object* PaginatedProduct. Pada fungsi pengambilan produk biasa, *object* PaginatedProduct diambil dari fungsi *getProducts* sedangkan pada fungsi pengambilan produk prioritas menggunakan fungsi *getPriorityProducts*. Setelah daftar produk berhasil diambil, maka sistem akan mengambil rata-rata ulasan dari setiap produk dan informasi toko dari produk. Untuk mempercepat operasi, maka sistem menggunakan konsep *multi-threading* di mana pengambilan rata-rata ulasan dan informasi toko setiap produk dilakukan secara *concurrent* untuk menghemat waktu. Hal tersebut dilakukan karena setiap operasi pengambilan ulasan dan pengambilan detail toko adalah operasi yang membutuhkan waktu yang relatif lama untuk diselesaikan karena sistem harus mengambil data dari *database* beberapa kali. Penggunaan *multi-thread* menghindari pemanggilan yang *sequential*.

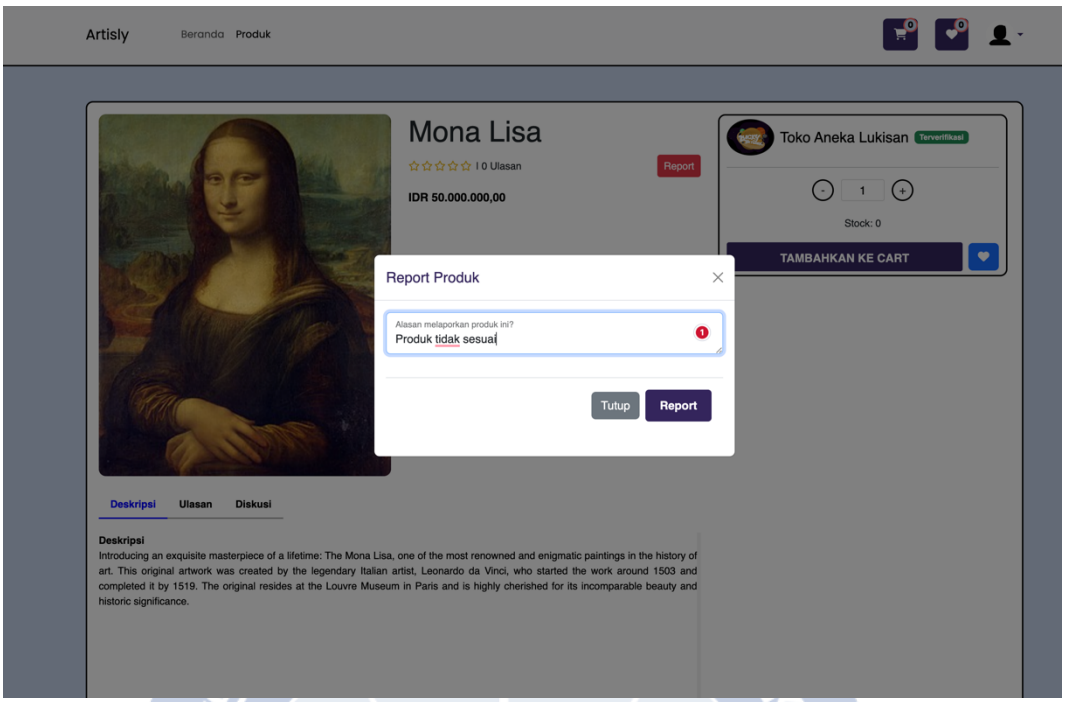
Gambar 5.26 adalah tampilan dari antarmuka halaman detail produk yang berisi informasi produk, daftar ulasan produk, daftar diskusi dan *form* pembuatan diskusi, *form* pelaporan produk, serta *form* penambahan produk ke dalam keranjang. Gambar 5.27 menunjukkan bagian diskusi serta tampilan *form* pembuatan diskusi. Gambar 5.28 menunjukkan implementasi antarmuka *form* pelaporan produk. Untuk menampilkan halaman detail produk, *frontend* juga mengirimkan 4 *request* yaitu *request* untuk mendapatkan detail produk, *request* untuk mendapatkan daftar ulasan produk, *request* untuk mendapatkan informasi penjual atau toko, dan *request* untuk mendapatkan daftar diskusi. Gambar 5.29 merupakan potongan kode yang menangani permintaan detail produk, Gambar 5.30 merupakan potongan kode yang menangani permintaan daftar ulasan produk, Gambar 5.31 merupakan potongan kode yang menangani permintaan detail toko, Gambar 5.32 adalah potongan kode yang menangani permintaan diskusi produk, dan Gambar 5.33 adalah potongan kode yang menangani pelaporan produk dan toko.



Gambar 5.26 Halaman Detail Produk



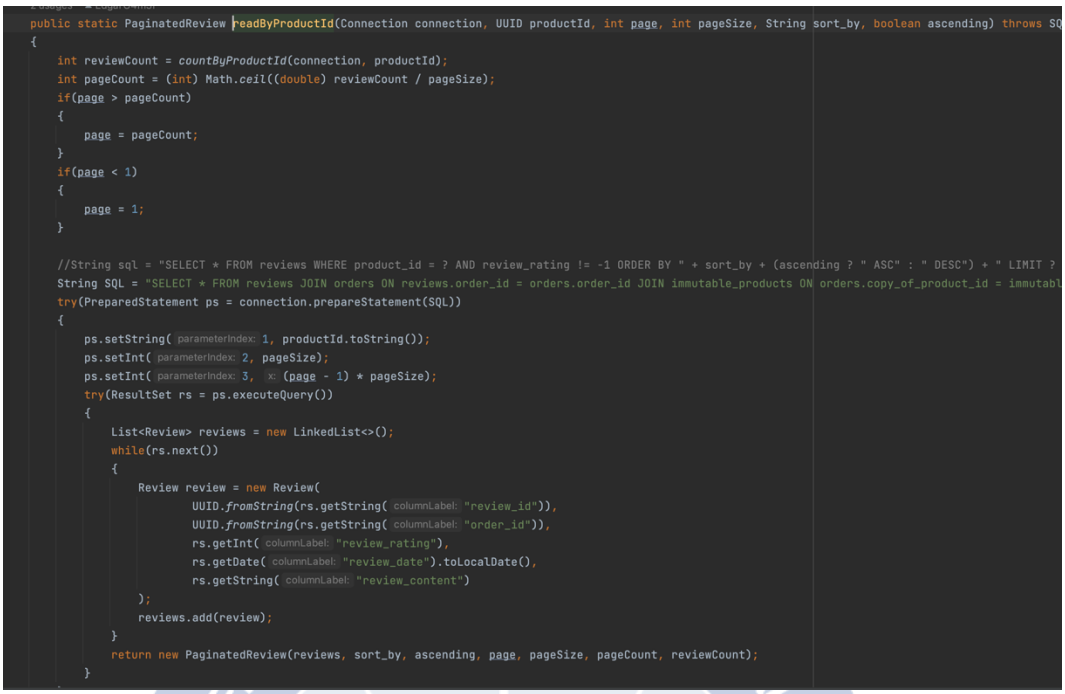
Gambar 5.27 Halaman Detail Produk(2)



Gambar 5.28 *Form* Pelaporan Produk

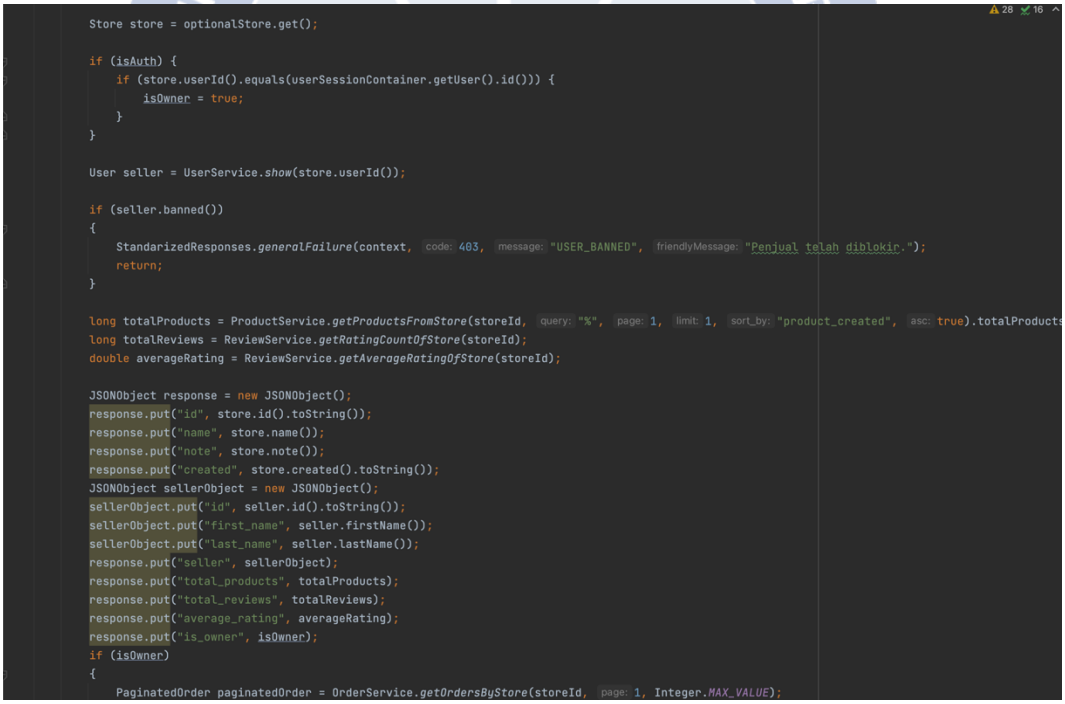
```
3 usages  EdgarG4m3r
public static Optional<Product> readByProductId(Connection connection, UUID productId) throws SQLException {
    String sql = "SELECT * FROM products WHERE product_id = ?";
    try(PreparedStatement ps = connection.prepareStatement(sql)) {
        ps.setString( parameterIndex: 1, productId.toString());
        try(ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                Product product = new Product(
                        UUID.fromString(rs.getString( columnLabel: "product_id")),
                        UUID.fromString(rs.getString( columnLabel: "store_id")),
                        UUID.fromString(rs.getString( columnLabel: "category_id")),
                        rs.getString( columnLabel: "product_name"),
                        rs.getString( columnLabel: "product_description"),
                        rs.getInt( columnLabel: "product_stock"),
                        rs.getDouble( columnLabel: "product_price"),
                        rs.getDate( columnLabel: "product_created").toLocalDate(),
                        rs.getDate( columnLabel: "product_updated").toLocalDate()
                );
                return Optional.of(product);
            }
        }
    }
    return Optional.empty();
}
```

Gambar 5.29 Potongan Kode Pengambilan Detail Produk



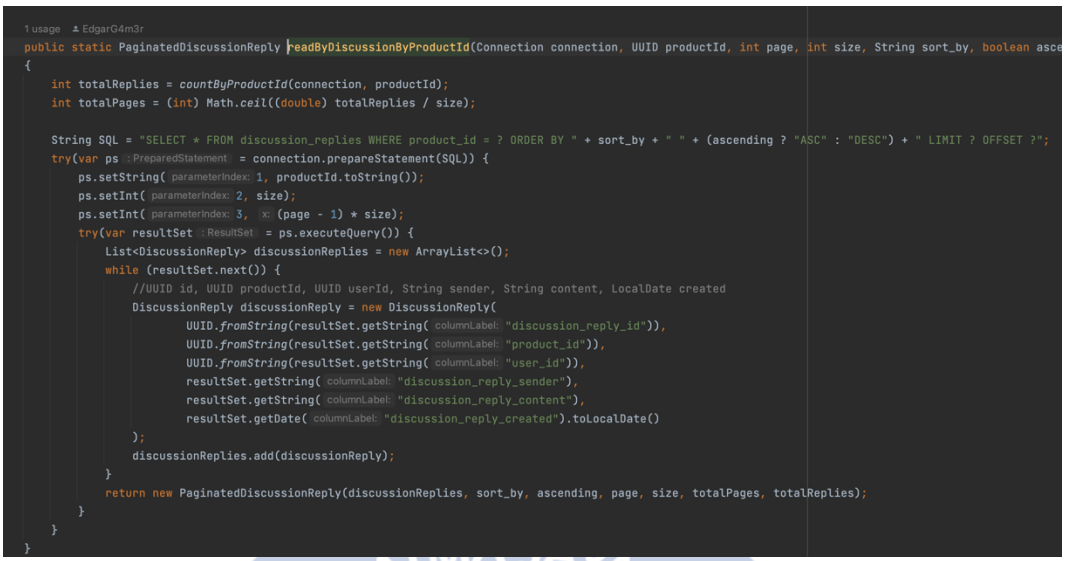
```
public static PaginatedReview readByProductId(Connection connection, UUID productId, int page, int pageSize, String sort_by, boolean ascending) throws SQ
{
    int reviewCount = countByProductId(connection, productId);
    int pageCount = (int) Math.ceil((double) reviewCount / pageSize);
    if(page > pageCount)
    {
        page = pageCount;
    }
    if(page < 1)
    {
        page = 1;
    }

    //String sql = "SELECT * FROM reviews WHERE product_id = ? AND review_rating != -1 ORDER BY " + sort_by + (ascending ? " ASC" : " DESC") + " LIMIT ?
    String SQL = "SELECT * FROM reviews JOIN orders ON reviews.order_id = orders.order_id JOIN immutable_products ON orders.copy_of_product_id = immutabl
    try(PreparedStatement ps = connection.prepareStatement(SQL))
    {
        ps.setString( parameterIndex: 1, productId.toString());
        ps.setInt( parameterIndex: 2, pageSize);
        ps.setInt( parameterIndex: 3, x: (page - 1) * pageSize);
        try(ResultSet rs = ps.executeQuery())
        {
            List<Review> reviews = new LinkedList<>();
            while(rs.next())
            {
                Review review = new Review(
                        UUID.fromString(rs.getString( columnLabel: "review_id")),
                        UUID.fromString(rs.getString( columnLabel: "order_id")),
                        rs.getInt( columnLabel: "review_rating"),
                        rs.getDate( columnLabel: "review_date").toLocalDate(),
                        rs.getString( columnLabel: "review_content")
                );
                reviews.add(review);
            }
            return new PaginatedReview(reviews, sort_by, ascending, page, pageSize, pageCount, reviewCount);
        }
```

Gambar 5.30 Potongan Kode Pengambilan Daftar Ulasan Produk



```
    Store store = optionalStore.get();

    if (isAuth) {
        if (store.userId().equals(userSessionContainer.getUser().id())) {
            isOwner = true;
        }
    }

    User seller = UserService.show(store.userId());

    if (seller.banned())
    {
        StandarizedResponses.generalFailure(context, code: 403, message: "USER_BANNED", friendlyMessage: "Penjual telah diblokir.");
        return;
    }

    long totalProducts = ProductService.getProductsFromStore(storeId, query: "%", page: 1, limit: 1, sort_by: "product_created", asc: true).totalProducts
    long totalReviews = ReviewService.getRatingCountOfStore(storeId);
    double averageRating = ReviewService.getAverageRatingOfStore(storeId);

    JSONObject response = new JSONObject();
    response.put("id", store.id().toString());
    response.put("name", store.name());
    response.put("note", store.note());
    response.put("created", store.created().toString());
    JSONObject sellerObject = new JSONObject();
    sellerObject.put("id", seller.id().toString());
    sellerObject.put("first_name", seller.firstName());
    sellerObject.put("last_name", seller.lastName());
    response.put("seller", sellerObject);
    response.put("total_products", totalProducts);
    response.put("total_reviews", totalReviews);
    response.put("average_rating", averageRating);
    response.put("is_owner", isOwner);
    if (isOwner)
    {
        PaginatedOrder paginatedOrder = OrderService.getOrdersByStore(storeId, page: 1, Integer.MAX_VALUE);
```

Gambar 5.31 Potongan Kode Pengambilan Detail Toko



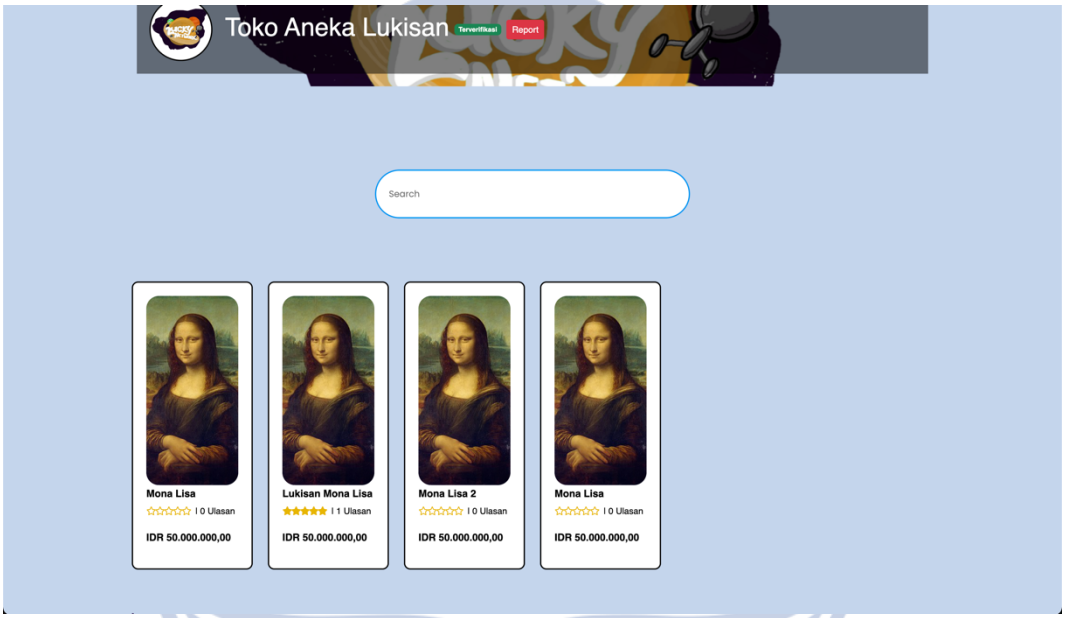
```
1 usage   EdgarG4m3r
public static PaginatedDiscussionReply readByDiscussionByProductId(Connection connection, UUID productId, int page, int size, String sort_by, boolean asce
{
    int totalReplies = countByProductId(connection, productId);
    int totalPages = (int) Math.ceil((double) totalReplies / size);

    String SQL = "SELECT * FROM discussion_replies WHERE product_id = ? ORDER BY " + sort_by + " " + (ascending ? "ASC" : "DESC") + " LIMIT ? OFFSET ?";
    try(var ps :PreparedStatement = connection.prepareStatement(SQL)) {
        ps.setString( parameterIndex: 1, productId.toString());
        ps.setInt( parameterIndex: 2, size);
        ps.setInt( parameterIndex: 3, x: (page - 1) * size);
        try(var resultSet :ResultSet = ps.executeQuery()) {
            List<DiscussionReply> discussionReplies = new ArrayList<>();
            while (resultSet.next()) {
                //UUID id, UUID productId, UUID userId, String sender, String content, LocalDate created
                DiscussionReply discussionReply = new DiscussionReply(
                        UUID.fromString(resultSet.getString( columnLabel: "discussion_reply_id")),
                        UUID.fromString(resultSet.getString( columnLabel: "product_id")),
                        UUID.fromString(resultSet.getString( columnLabel: "user_id")),
                        resultSet.getString( columnLabel: "discussion_reply_sender"),
                        resultSet.getString( columnLabel: "discussion_reply_content"),
                        resultSet.getDate( columnLabel: "discussion_reply_created").toLocalDate()
                );
                discussionReplies.add(discussionReply);
            }
            return new PaginatedDiscussionReply(discussionReplies, sort_by, ascending, page, size, totalPages, totalReplies);
        }
    }
}
```

Gambar 5.32 Potongan Kode Pengambilan Daftar Diskusi Produk

```
1 usage   EdgarG4m3r *
public static void createReport(UUID reportedStore, UUID reportedBy, Optional<UUID> reportedProduct, String reportReason) throws SQLException, ProductNo
    try(Connection connection = Artisly.instance.getMySQL().getConnection()) {
        connection.setAutoCommit(false);
        if (reportedProduct.isPresent()) {
            ImmutableProduct immutableProduct = ProductService.copyProduct(reportedProduct.get());
            Artisly.instance.getMediaService().copyProductImagesToImmutable(reportedProduct.get(), immutableProduct.id());
            CRUDSStoreReport.create(connection, reportedStore, reportedBy, Optional.of(immutableProduct.id()), reportReason);
            return;
        }
        CRUDSStoreReport.create(connection, reportedStore, reportedBy, immutableProductId: Optional.empty(), reportReason);
        connection.commit();
        return;
    }
}
```

Gambar 5.33 Potongan Pembuatan Laporan Aduan Produk dan Toko

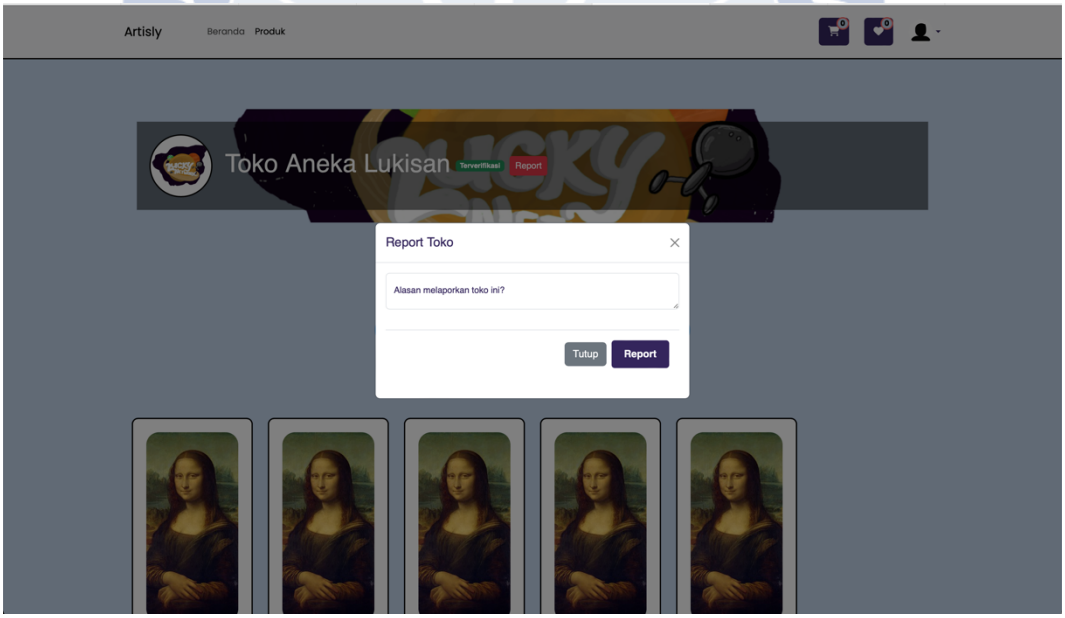
Gambar 5.34 adalah tampilan dari antarmuka halaman detail toko yang menampilkan *form* pelaporan toko, daftar produk yang dijual oleh toko, serta detail toko. Gambar 5.35 menunjukan potongan kode yang menampilkan produk-produk yang ada dalam toko. Gambar 5.33 menunjukkan potongan kode untuk menangani laporan toko. Potongan kode untuk mengambil daftar produk yang dijual oleh toko terdapat pada Gambar 5.36. Gambar 5.31 merupakan potongan kode yang digunakan untuk menampilkan detail toko. Gambar 5.36 menampilkan implementasi antarmuka *form* pelaporan toko.



Gambar 5.34 Halaman Detail Toko

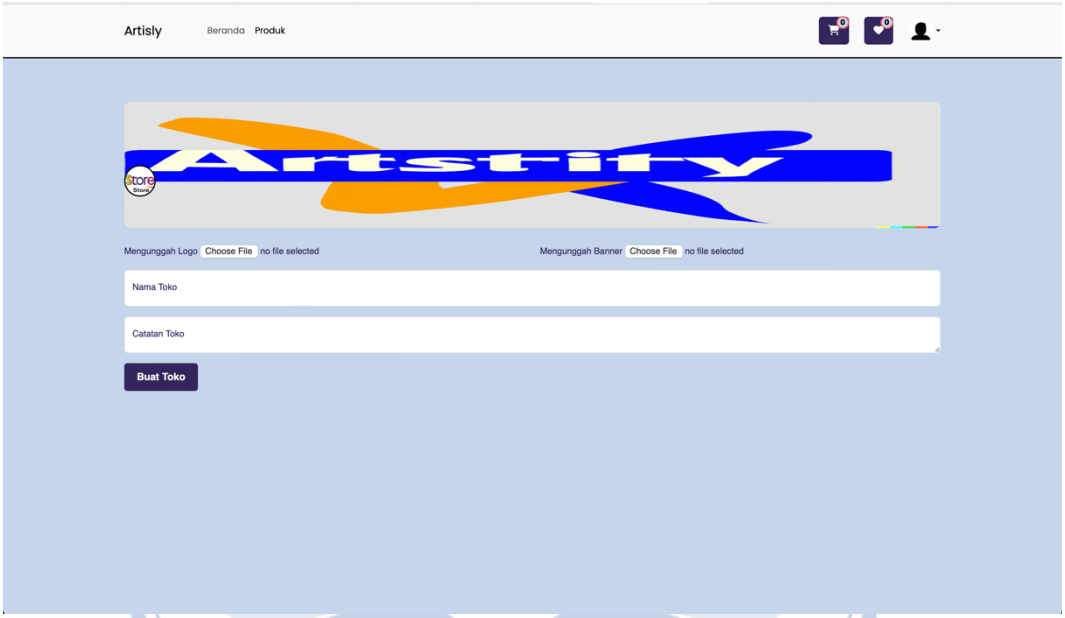
```
public static PaginatedProduct searchByStore(Connection connection, UUID storeId, String query, int page, int size, String sort_by, boolean as
    if (query.equalsIgnoreCase( anotherString: "%"))
    {
        return readByStoreId(connection, storeId, page, size, sort_by, ascending);
    }

    int totalProducts = getProductCountByStoreId(connection, storeId, query);
    int totalPages = (int) Math.ceil((double) totalProducts / size);
    int offset = (page - 1) * size;

    String sortOrder = ascending ? "ASC" : "DESC";

    String sortByColumn;
    switch (sort_by) {
        case "review":
            sortByColumn = "avg_review";
            break;
        case "price":
            sortByColumn = "p.product_price";
            break;
        default:
            sortByColumn = "p.product_created";
    }


    String SQL = "SELECT p.product_id, p.store_id, p.category_id, p.product_name, p.product_description, p.product_stock, p.product_price, p.product_cre
            "FROM products p " +
            "LEFT JOIN immutable_products ip ON p.product_id = ip.product_id " +
            "LEFT JOIN orders o ON ip.immutable_product_id = o.copy_of_product_id " +
            "LEFT JOIN reviews r ON o.order_id = r.order_id " +
            "WHERE p.store_id = ? " +
            "AND (p.product_name LIKE ? OR p.product_description LIKE ?) " +
            "GROUP BY p.product_id " +
            "ORDER BY " + sortByColumn + " " + sortOrder + " " +
            "LIMIT ? OFFSET ?";

    try(PreparedStatement ps = connection.prepareStatement(SQL)) {
```

Gambar 5.35 Potongan Kode Pengambilan Daftar Produk Toko



Gambar 5.36 *Form* Pelaporan Toko

Gambar 5.37 adalah tampilan dari antarmuka halaman buka toko yang menampilkan *form* pembukaan toko. Gambar 5.38 menampilkan potongan kode pada *presentation layer* pembukaan toko yang bertanggung jawab atas validasi data dan menyimpan media dari toko, Gambar 5.39 menampilkan potongan kode *service layer* yang melakukan pengecekan lebih lanjut terhadap pengguna sebelum membuka toko.



Gambar 5.37 Halaman Pembukaan Toko

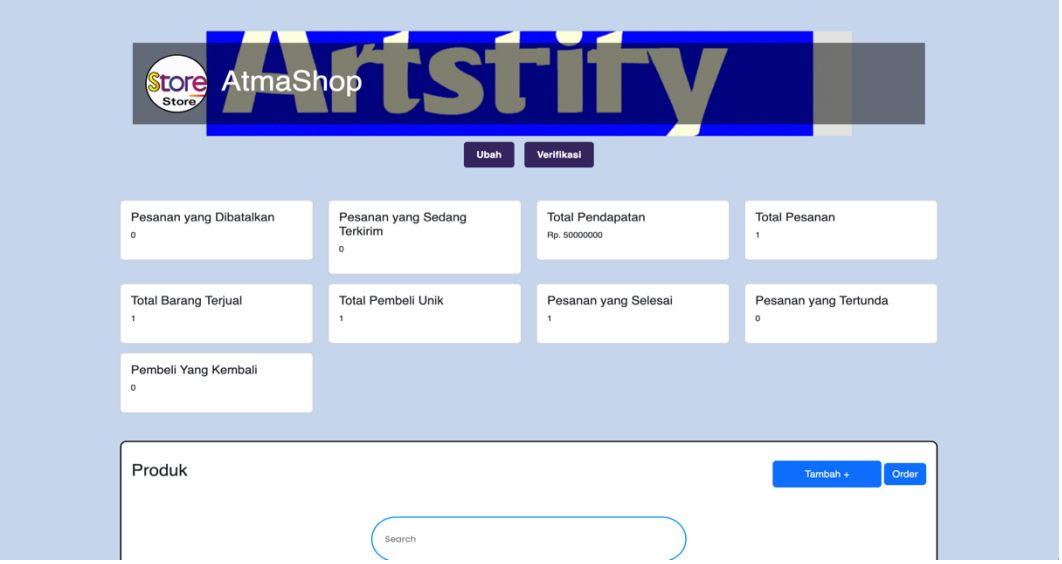
```
@Override
public void handle(Context context) {
    UserSessionContainer userSessionContainer = AuthHandler.authenticateUser(context);
    if (userSessionContainer == null) {
        return;
    }

    InputFilter.validateString( param: "store_name", ParamField.FORM, context, length: 50);
    InputFilter.validateString( param: "store_note", ParamField.FORM, context, length: 500);
    MediaType[] allowedTypes = new MediaType[] {
            MediaType.image( type: "png"),
            MediaType.image( type: "jpeg"),
            MediaType.image( type: "jpg")
    };
    InputFilter.validateUploadedFiles( field: "store_logo", maxEachSizeInKB: 5 * 1024, allowedTypes, maxTotalSizeInKB: 5 * 1024, maxFilesCount: 1, context);
    InputFilter.validateUploadedFiles( field: "store_banner", maxEachSizeInKB: 10 * 1024, allowedTypes, maxTotalSizeInKB: 10 * 1024, maxFilesCount: 1, context);

    if (context.attribute( key: "hasErrors") != null) {
        StandarizedResponses.invalidParameter(context);
        return;
    }

    String storeName = context.formParam( key: "store_name");
    String storeNote = context.formParam( key: "store_note");

    try
    {
        StoreService.createStore(userSessionContainer.getUser().id(), storeName, storeNote);
        Store store = StoreService.getStore(userSessionContainer.getUser().id()).get();
        try(InputStream is = context.uploadedFile( fileName: "store_logo").content()) {
            Artisly.instance.getMediaService().uploadStoreLogo(store.id(), is, new Tika().detect(is));

        } catch (IOException e) {
            StandarizedResponses.generalFailure(context, code: 500, message: "IO_EXCEPTION", friendlyMessage: "Terjadi kesalahan saat mengunggah gambar");
            return;
        }
        try(InputStream is = context.uploadedFile( fileName: "store_banner").content()) {
```

Gambar 5.38 Potongan Kode Pembukaan Toko

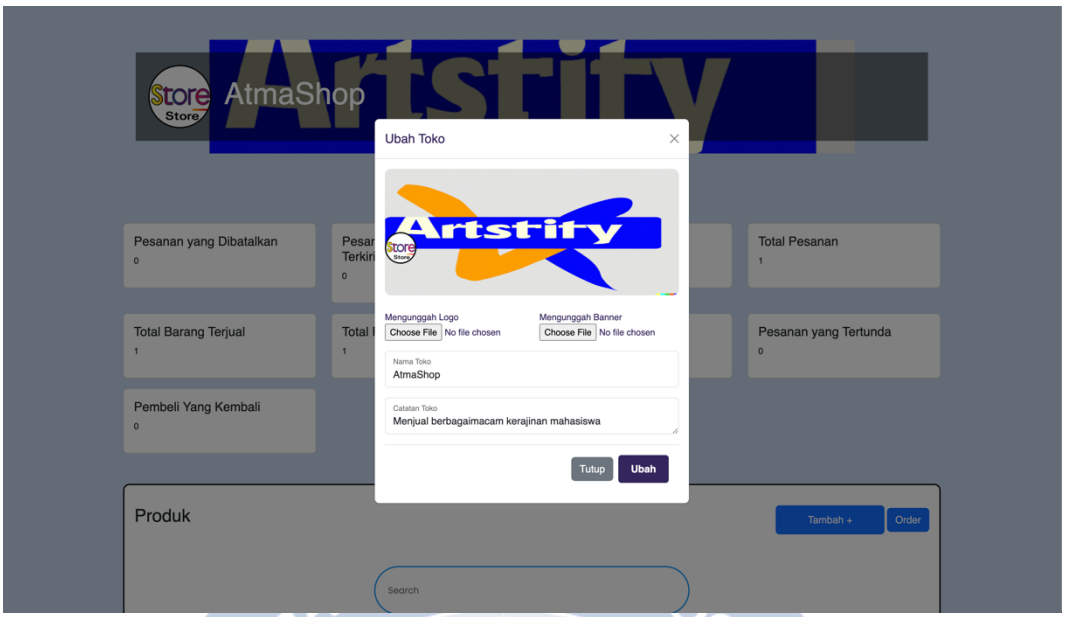
```java
EdgarG4m3r *
public class StoreService {

    1 usage   EdgarG4m3r *
    public static void createStore(UUID userId, String storeName, String storeNote) throws SQLException, UserNotFoundException, EmailNotVerifiedException, A
    {
        try(Connection connection = Artisly.instance.getMySQL().getConnection())
        {
            Optional<User> userOptional = CRUDSUser.readById(connection, userId);
            if(userOptional.isEmpty())
            {
                throw new UserNotFoundException("Akun tidak ditemukan");
            }

            User user = userOptional.get();
            if(user.emailVerified().isEmpty())
            {
                throw new EmailNotVerifiedException("Email belum terverifikasi");
            }

            if(user.nomorKTP().isEmpty())
            {
                throw new AccountNotVerifiedException("Harap masukan nomor KTP");
            }

            Optional<Store> storeOptional = CRUDSStore.readByUserId(connection, userId);
            if(storeOptional.isPresent())
            {
                throw new AlreadyHaveStoreException("Anda sudah memiliki toko");
            }

            CRUDSStore.create(connection, storeName, storeNote, userId);


        }
    }
```

Gambar 5.39 Potongan Kode Pembukaan Toko(2)

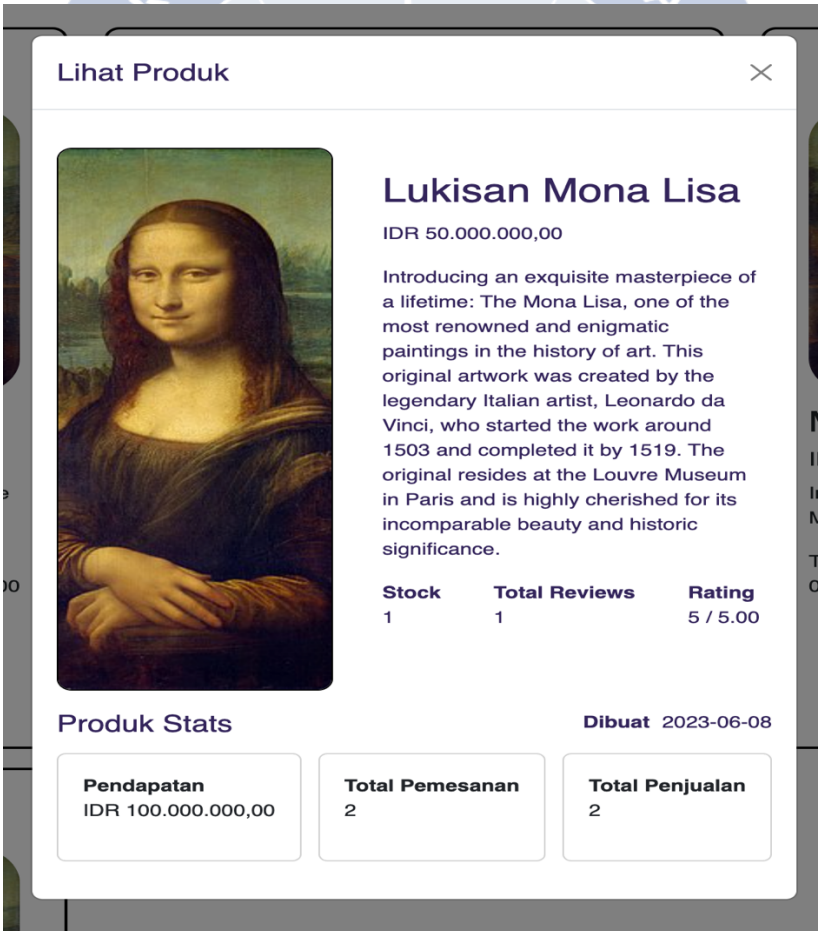
Gambar 5.40 adalah tampilan dari antarmuka halaman toko anda yang menampilkan statistik toko, produk-produk toko, *form* perubahan toko, *form* penambahan dan perubahan produk, dan *form* permohonan verifikasi toko. Untuk menampilkan halaman toko anda, *frontend* menghubungi *backend* sebanyak dua kali untuk mengambil data detail toko yang juga mengandung data statistik toko, dan untuk mengambil daftar produk-produk toko. Gambar 5.35 merupakan potongan kode yang melayani pengambilan daftar produk suatu toko, Gambar 5.31 merupakan potongan kode yang melayani pengambilan detail toko, Gambar 5.41 adalah tampilan dari antarmuka *form* perubahan toko, Gambar 5.42 adalah gambar tampilan dari antarmuka *modal* detail produk, Gambar 5.43 adalah tampilan dari antarmuka *form* penambahan dan perubahan produk di mana *frontend* akan mengisi seluruh *field* dengan informasi produk yang sudah ada jika ingin melakukan perubahan produk dan akan mengosongkan seluruh *field* jika pengguna menekan tombol pembuatan produk baru, Gambar 5.44 adalah tampilan dari antarmuka *form* permohonan verifikasi toko, Gambar 5.45 merupakan gambar potongan kode yang melayani permintaan perubahan toko, Gambar 5.46 merupakan gambar potongan kode yang melayani penambahan produk, Gambar 5.47 merupakan gambar potongan kode yang melayani permintaan perubahan produk, dan Gambar 5.48 merupakan potongan kode yang melayani permintaan untuk pembuatan permohonan verifikasi toko pada *backend*.



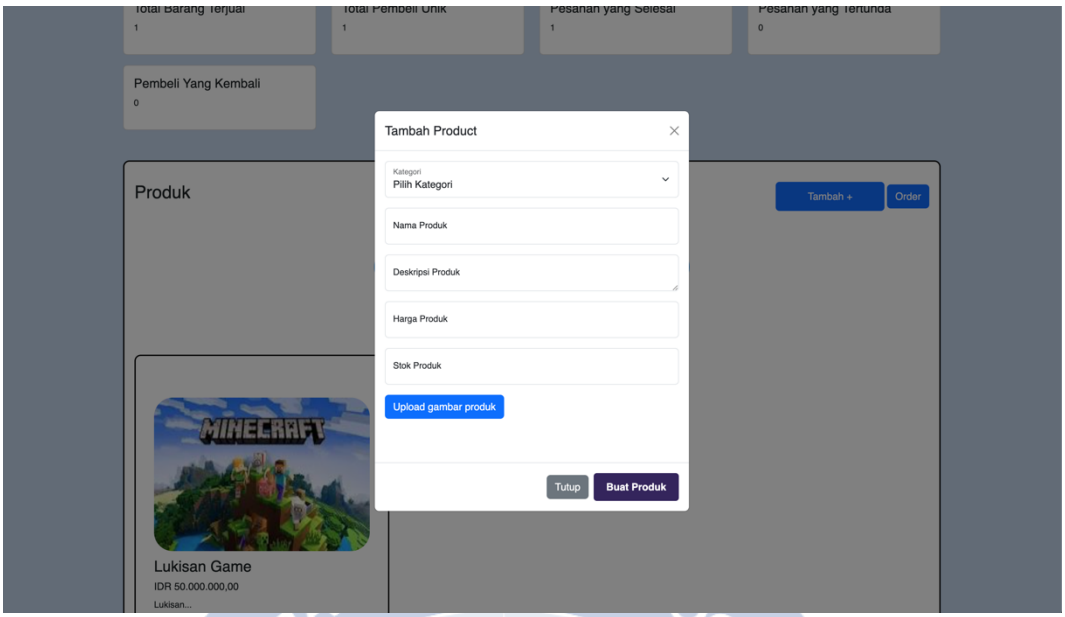
Gambar 5.40 Halaman Toko Anda



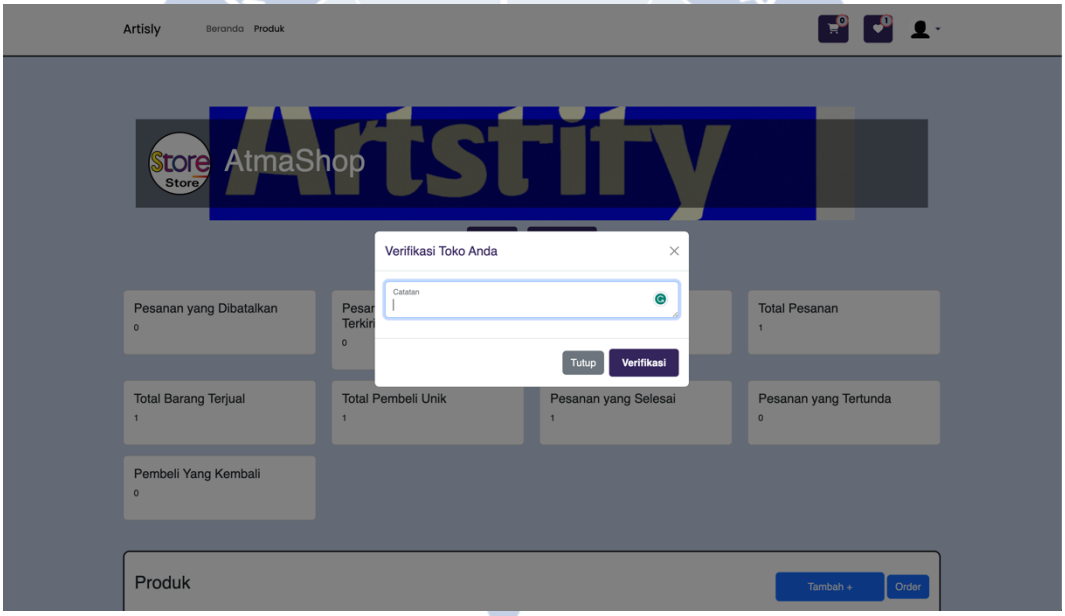
Gambar 5.41 *Form* Perubahan Toko



Gambar 5.42 *Modal* Detail Produk



Gambar 5.43 *Form* Penambahan dan Perubahan Produk



Gambar 5.44 *Form* Permohonan Verifikasi Toko

```java
public static void updateStore(UUID userId, String storeName, String storeNote) throws SQLException, UserNotFoundException, EmailNotVerifiedException, Stor
{
    try(Connection connection = Artisly.instance.getMySQL().getConnection()) {
        //Connection connection, UUID storeId, String storeName, String storeNote,

        Optional<User> userOptional = CRUDSUser.readById(connection, userId);
        if (userOptional.isEmpty()) {
            throw new UserNotFoundException("Akun tidak ditemukan");
        }

        User user = userOptional.get();
        if (user.emailVerified().isEmpty()) {
            throw new EmailNotVerifiedException("Email belum terverifikasi");
        }

        if (user.nomorKTP().isEmpty()) {
            throw new AccountNotVerifiedException("Harap masukan nomor KTP");
        }

        Optional<Store> storeOptional = CRUDSStore.readByUserId(connection, userId);
        if (storeOptional.isEmpty()) {
            throw new StoreNotExist("Anda belum memiliki toko");
        }

        Store store = storeOptional.get();
        CRUDSStore.update(connection, store.id(), storeName, storeNote, store.storeVetId());
    }
}
```

Suggested plugins AWS Toolkit, Protocol Buffers

Gambar 5.45 Potongan Kode Perubahan Toko

```java
1 usage  ↥ EdgarG4m3r
public static Product createProduct(UUID storeId, UUID categoryId, String productName, String productDescription, double productPrice, int productStock) thr
    try(Connection connection = Artisly.instance.getMySQL().getConnection())
    {
        Optional<Category> category = CategoryService.getCategory(categoryId);
        if (category.isEmpty())
        {
            throw new CategoryNotExist("Kategori tidak ditemukan. Harap cek kembali kategori yang anda masukkan");
        }

        Optional<Store> store = StoreService.getStoreById(storeId);

        if (store.isEmpty())
        {
            throw new StoreNotExist("Toko tidak ditemukan. Harap cek kembali toko yang anda masukkan");
        }

        return CRUDSProduct.create(connection, storeId, categoryId, productName, productDescription, productPrice, productStock);
    }
}
```

Gambar 5.46 Potongan Kode Penambahan Produk

```java
1 usage  ↥ EdgarG4m3r
public static boolean updateProductOfUser(UUID userId, UUID productId, UUID categoryId, String productName, String productDescription, double productPri
    try(Connection connection = Artisly.instance.getMySQL().getConnection())
    {
        Optional<Category> category = CategoryService.getCategory(categoryId);
        if (category.isEmpty())
        {
            throw new CategoryNotExist("Kategori tidak ditemukan. Harap cek kembali kategori yang anda masukkan");
        }

        Optional<Product> product = getProduct(productId);
        if (product.isEmpty())
        {
            throw new ProductNotExist("Produk tidak ditemukan. Harap cek kembali produk yang anda masukkan");
        }

        Optional<Store> store = StoreService.getStore(userId);
        if (store.isEmpty())
        {
            throw new StoreNotExist("Anda tidak memiliki toko. Silahkan buat toko terlebih dahulu");
        }

        if (product.get().storeId().compareTo(store.get().id()) != 0)
        {
            throw new ProductNotExist("Produk tidak ditemukan. Harap cek kembali produk yang anda masukkan");
        }

        return CRUDSProduct.update(connection, productId, categoryId, productName, productDescription, productPrice, productStock);
    }
```

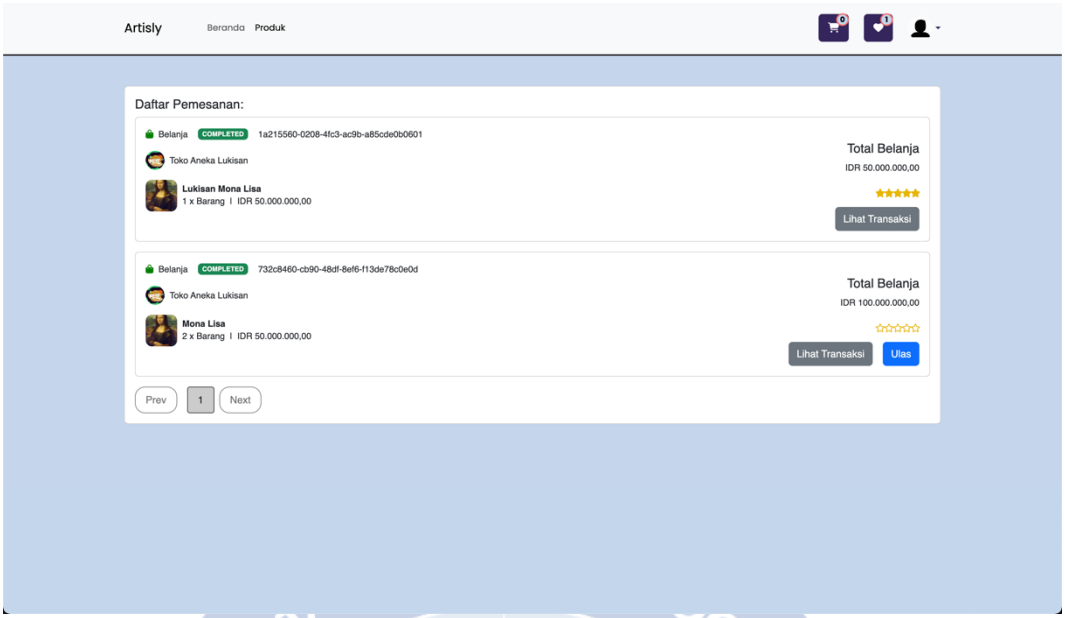
Gambar 5.47 Potongan Kode Perubahan Produk

```
public static void requestVet(UUID userId, String note) throws AlreadyHaveVetRequest, SQLException, StoreNotExist, RequirementNotMet {
    try (Connection connection = Artisly.instance.getMySQL().getConnection())
    {
        Optional<Store> storeOptional = CRUDSStore.readByUserId(connection, userId);
        if(!storeOptional.isPresent())
        {
            throw new StoreNotExist("Toko tidak ditemukan");
        }

        if (ReviewService.getAverageRatingOfStore(storeOptional.get().id()) < 4.0)
        {
            throw new RequirementNotMet("Toko harus memiliki rating minimal 4.0");
        }

        PaginatedOrder paginatedOrder = OrderService.getOrdersByStore(storeOptional.get().id(), page: 1, Integer.MAX_VALUE);
        List<Order> allTimeOrders = paginatedOrder.orders();
        int totalCompletedOrders = 0;
        for (Order order : allTimeOrders)
        {
            if (order.orderStatus().equals(OrderStatus.COMPLETED))
            {
                totalCompletedOrders++;
            }
        }

        if (totalCompletedOrders < 10)
        {
            throw new RequirementNotMet("Toko harus memiliki minimal 10 pesanan yang telah selesai");
        }

        Optional<StoreVet> storeVetOptional = CRUDSStoreVet.readByStoreId(connection, storeOptional.get().id());
        if(storeVetOptional.isPresent())
        {
            throw new AlreadyHaveVetRequest("Toko sudah memiliki permintaan vet");
        }

        CRUDSStoreVet.create(connection, userId, storeOptional.get().id(), note);
    }
```

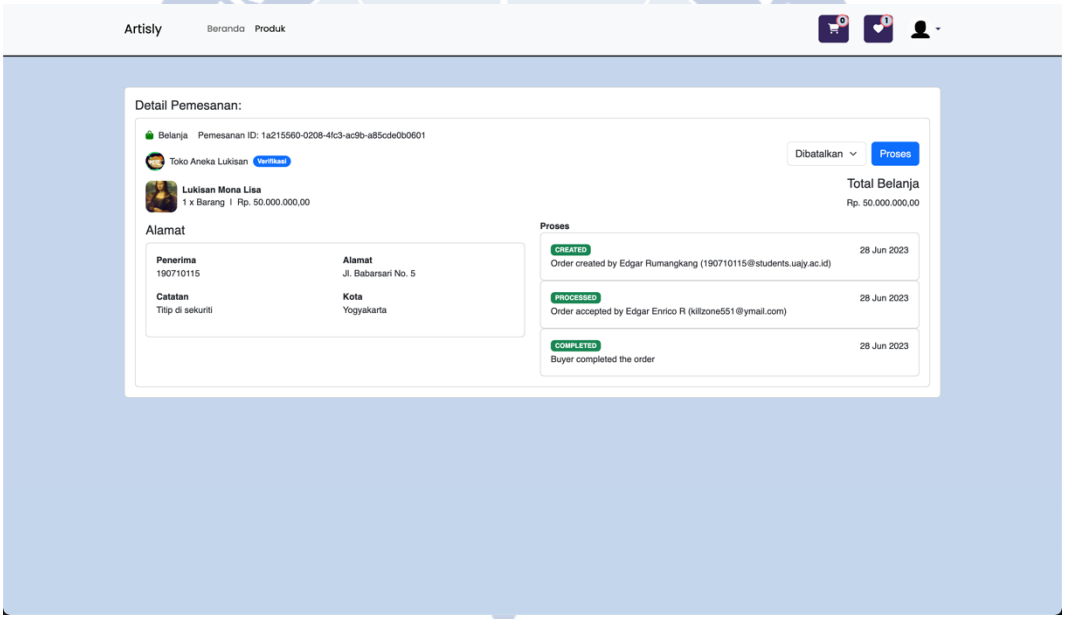
Gambar 5.48 Potongan Kode Pembuatan Permintaan Verifikasi Toko

Gambar 5.49 adalah tampilan dari antarmuka halaman riwayat pemesanan yang menampilkan daftar pesanan, Gambar 5.50 adalah tampilan dari antarmuka halaman detail pemesanan, *form* pembuatan ulasan yang terdapat pada Gambar 5.51, Gambar 5.52 menampilkan potongan kode yang dipanggil oleh *frontend* untuk mengambil daftar pesanan, Gambar 5.53 menampilkan potongan kode *backend* yang melayani permintaan pengambilan detail pesanan, Gambar 5.54 merupakan potongan kode yang melayani permintaan yang dikirim oleh *frontend* saat pengguna ingin membuat suatu ulasan, dan Gambar 5.55 merupakan potongan kode yang melayani permintaan dari *form* perubahan status pesanan.

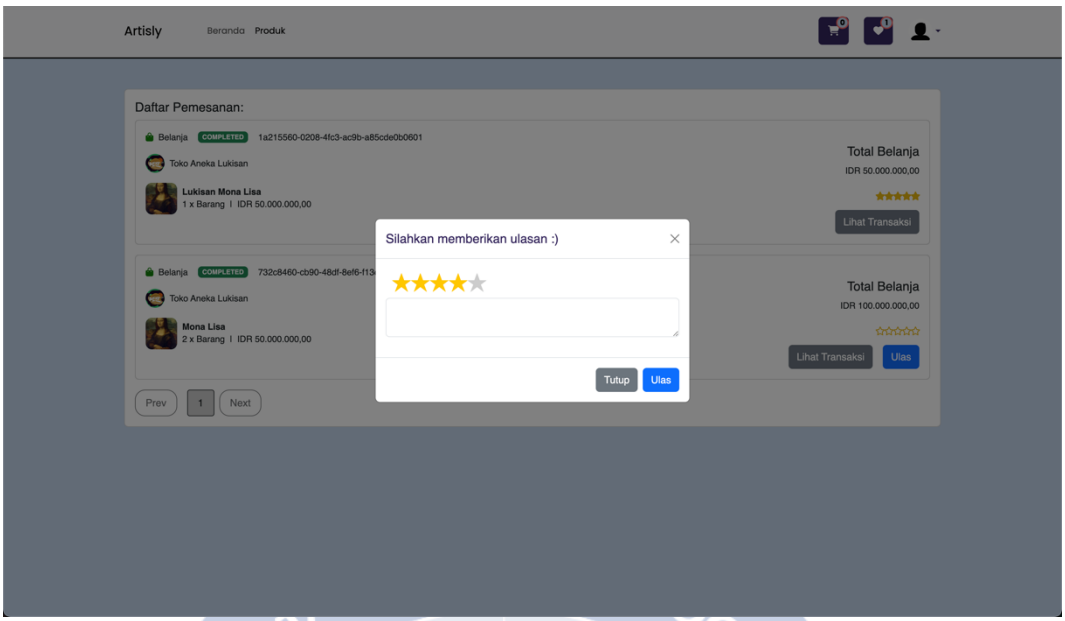
Dalam operasi yang mengandung dengan pemesanan produk seperti pada Gambar 5.55, sistem menggunakan *object* ImmutableProduct dan ImmutableAddress yaitu sebuah *object* yang merupakan *snapshot* dari produk atau alamat tertentu yang mengandung seluruh informasi dari produk atau alamat yang asli. *Object* ImmutableProduct dan ImmutableAddress tidak dapat diubah oleh pengguna dan berfungsi untuk menjaga konsistensi pemesanan. Hal ini untuk menghindari hal yang tidak terduga yang dapat muncul jika pengguna menghapus atau mengubah alamat atau pembeli mengubah atau menghapus produk setelah melakukan pemesanan terjadi. Selain itu, ImmutableProduct juga digunakan untuk melindungi pengguna dari percobaan penipuan jika penjual memutuskan untuk mengubah informasi produk setelah pemesanan terjadi. ImmutableProduct juga dibuat oleh sistem pada saat pengguna melaporkan suatu produk tertentu.



Gambar 5.49 Halaman Riwayat Pemesanan



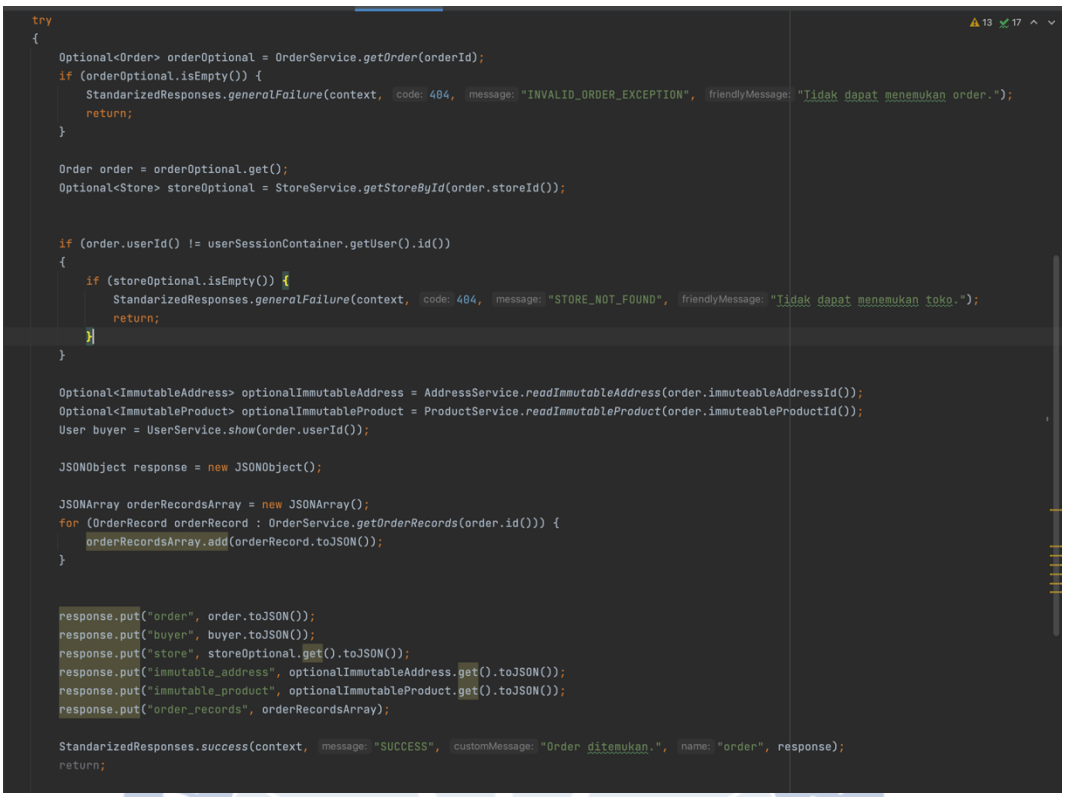
Gambar 5.50 Halaman Detail Pemesanan



Gambar 5.51 *Form* Pembuatan Ulasan

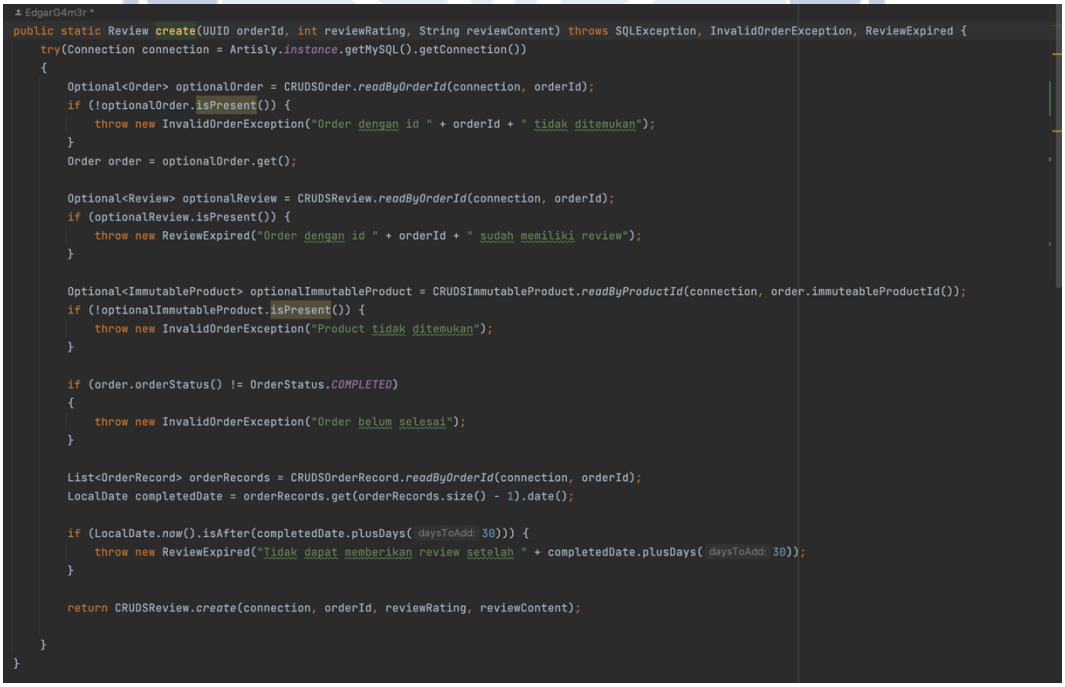
```
try
{
    PaginatedOrder userOrders;
    if (view.equals("user")) {
        userOrders = OrderService.getOrders(userSessionContainer.getUser().id(), page, limit);
    }
    else
    {
        Optional<Store> store = StoreService.getStore(userSessionContainer.getUser().id());
        if (store.isEmpty()) {
            StandarizedResponses.generalFailure(context, code: 404, message: "STORE_NOT_FOUND", friendlyMessage: "Tidak dapat menemukan toko.");
            return;
        }
        userOrders = OrderService.getOrdersByStore(store.get().id(), page, limit);
    }
    JSONArray ordersArray = new JSONArray();
    for (Order order : userOrders.orders())
    {
        ImmutableAddress immutableAddress = AddressService.readImmutableAddress(order.immuteableAddressId()).get();
        ImmutableProduct immutableProduct = ProductService.readImmutableProduct(order.immuteableProductId()).get();
        User buyer = UserService.show(order.userId());
        Store store = StoreService.getStoreById(order.storeId()).get();

        JSONObject entry = new JSONObject();
        entry.put("order", order.toJSON());
        entry.put("address", immutableAddress.toJSON());
        entry.put("product", immutableProduct.toJSON());
        entry.put("buyer", buyer.toJSONRestricted());
        entry.put("store", store.toJSON());
        if (view.equals("user")) {
            Optional<Review> review = ReviewService.getReviewByOrderId(order.id());
            JSONObject reviewObject = new JSONObject();
            if (review.isPresent()) {
                reviewObject.put("data", review.get().toJSON());
                reviewObject.put("reviewed", true);
                reviewObject.put("can_review", false);
            }
            else
            {
                reviewObject.put("data", null);
                reviewObject.put("reviewed", false);
```

Gambar 5.52 Potongan Kode Riwayat Pemesanan



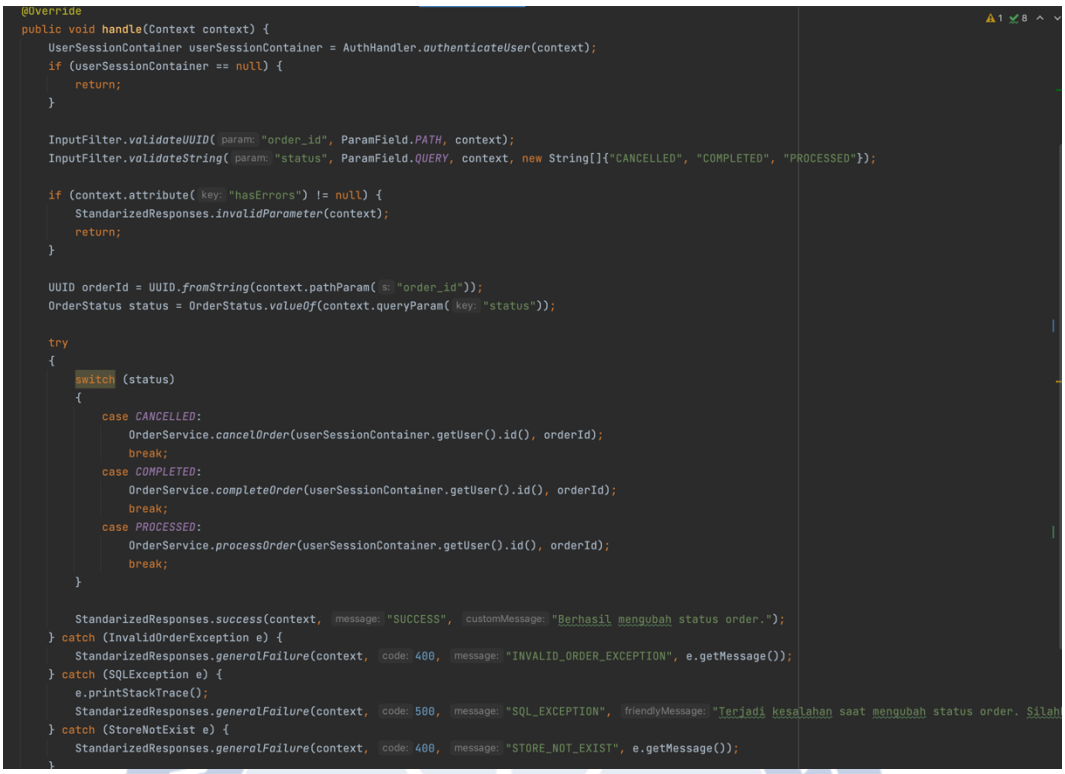
```
try
{
    Optional<Order> orderOptional = OrderService.getOrder(orderId);
    if (orderOptional.isEmpty()) {
        StandarizedResponses.generalFailure(context, code: 404, message: "INVALID_ORDER_EXCEPTION", friendlyMessage: "Tidak dapat menemukan order.");
        return;
    }

    Order order = orderOptional.get();
    Optional<Store> storeOptional = StoreService.getStoreById(order.storeId());


    if (order.userId() != userSessionContainer.getUser().id())
    {
        if (storeOptional.isEmpty()) {
            StandarizedResponses.generalFailure(context, code: 404, message: "STORE_NOT_FOUND", friendlyMessage: "Tidak dapat menemukan toko.");
            return;
        }
    }

    Optional<ImmutableAddress> optionalImmutableAddress = AddressService.readImmutableAddress(order.immuteableAddressId());
    Optional<ImmutableProduct> optionalImmutableProduct = ProductService.readImmutableProduct(order.immuteableProductId());
    User buyer = UserService.show(order.userId());

    JSONObject response = new JSONObject();

    JSONArray orderRecordsArray = new JSONArray();
    for (OrderRecord orderRecord : OrderService.getOrderRecords(order.id())) {
        orderRecordsArray.add(orderRecord.toJSON());
    }


    response.put("order", order.toJSON());
    response.put("buyer", buyer.toJSON());
    response.put("store", storeOptional.get().toJSON());
    response.put("immutable_address", optionalImmutableAddress.get().toJSON());
    response.put("immutable_product", optionalImmutableProduct.get().toJSON());
    response.put("order_records", orderRecordsArray);

    StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Order ditemukan.", name: "order", response);
    return;
```

Gambar 5.53 Potongan Kode Detail Pemesanan



```
EdgarG4m3r *
public static Review create(UUID orderId, int reviewRating, String reviewContent) throws SQLException, InvalidOrderException, ReviewExpired {
    try(Connection connection = Artisly.instance.getMySQL().getConnection())
    {
        Optional<Order> optionalOrder = CRUDSOrder.readByOrderId(connection, orderId);
        if (!optionalOrder.isPresent()) {
            throw new InvalidOrderException("Order dengan id " + orderId + " tidak ditemukan");
        }
        Order order = optionalOrder.get();

        Optional<Review> optionalReview = CRUDSReview.readByOrderId(connection, orderId);
        if (optionalReview.isPresent()) {
            throw new ReviewExpired("Order dengan id " + orderId + " sudah memiliki review");
        }

        Optional<ImmutableProduct> optionalImmutableProduct = CRUDSImmutableProduct.readByProductId(connection, order.immuteableProductId());
        if (!optionalImmutableProduct.isPresent()) {
            throw new InvalidOrderException("Product tidak ditemukan");
        }

        if (order.orderStatus() != OrderStatus.COMPLETED)
        {
            throw new InvalidOrderException("Order belum selesai");
        }

        List<OrderRecord> orderRecords = CRUDSOrderRecord.readByOrderId(connection, orderId);
        LocalDate completedDate = orderRecords.get(orderRecords.size() - 1).date();

        if (LocalDate.now().isAfter(completedDate.plusDays( daysToAdd: 30))) {
            throw new ReviewExpired("Tidak dapat memberikan review setelah " + completedDate.plusDays( daysToAdd: 30));
        }

        return CRUDSReview.create(connection, orderId, reviewRating, reviewContent);

    }
}
```

Gambar 5.54 Potongan Kode Pembuatan Ulasan



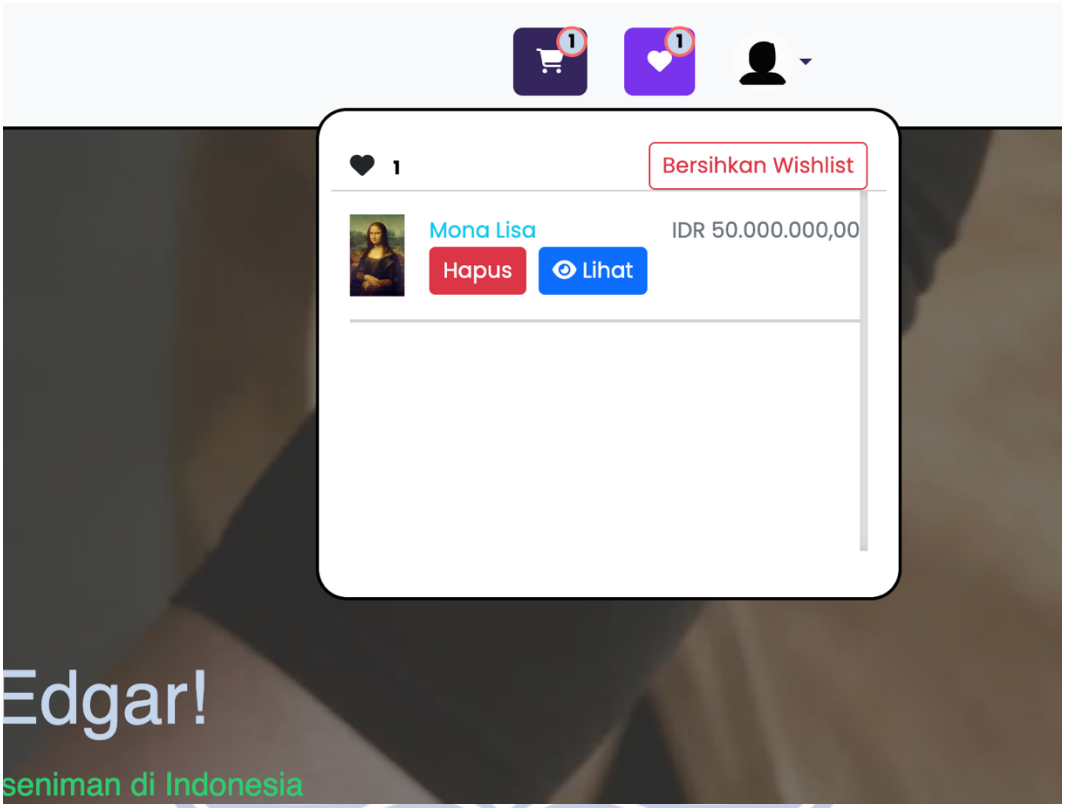
```
@Override
public void handle(Context context) {
    UserSessionContainer userSessionContainer = AuthHandler.authenticateUser(context);
    if (userSessionContainer == null) {
        return;
    }

    InputFilter.validateUUID( param: "order_id", ParamField.PATH, context);
    InputFilter.validateString( param: "status", ParamField.QUERY, context, new String[]{"CANCELLED", "COMPLETED", "PROCESSED"});

    if (context.attribute( key: "hasErrors") != null) {
        StandarizedResponses.invalidParameter(context);
        return;
    }

    UUID orderId = UUID.fromString(context.pathParam( s: "order_id"));
    OrderStatus status = OrderStatus.valueOf(context.queryParam( key: "status"));

    try
    {
        switch (status)
        {
            case CANCELLED:
                OrderService.cancelOrder(userSessionContainer.getUser().id(), orderId);
                break;
            case COMPLETED:
                OrderService.completeOrder(userSessionContainer.getUser().id(), orderId);
                break;
            case PROCESSED:
                OrderService.processOrder(userSessionContainer.getUser().id(), orderId);
                break;
        }

        StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Berhasil mengubah status order.");
    } catch (InvalidOrderException e) {
        StandarizedResponses.generalFailure(context, code: 400, message: "INVALID_ORDER_EXCEPTION", e.getMessage());
    } catch (SQLException e) {
        e.printStackTrace();
        StandarizedResponses.generalFailure(context, code: 500, message: "SQL_EXCEPTION", friendlyMessage: "Terjadi kesalahan saat mengubah status order. Silah
    } catch (StoreNotExist e) {
        StandarizedResponses.generalFailure(context, code: 400, message: "STORE_NOT_EXIST", e.getMessage());
    }
```

Gambar 5.55 Potongan Kode Perubahan Status Pemesanan

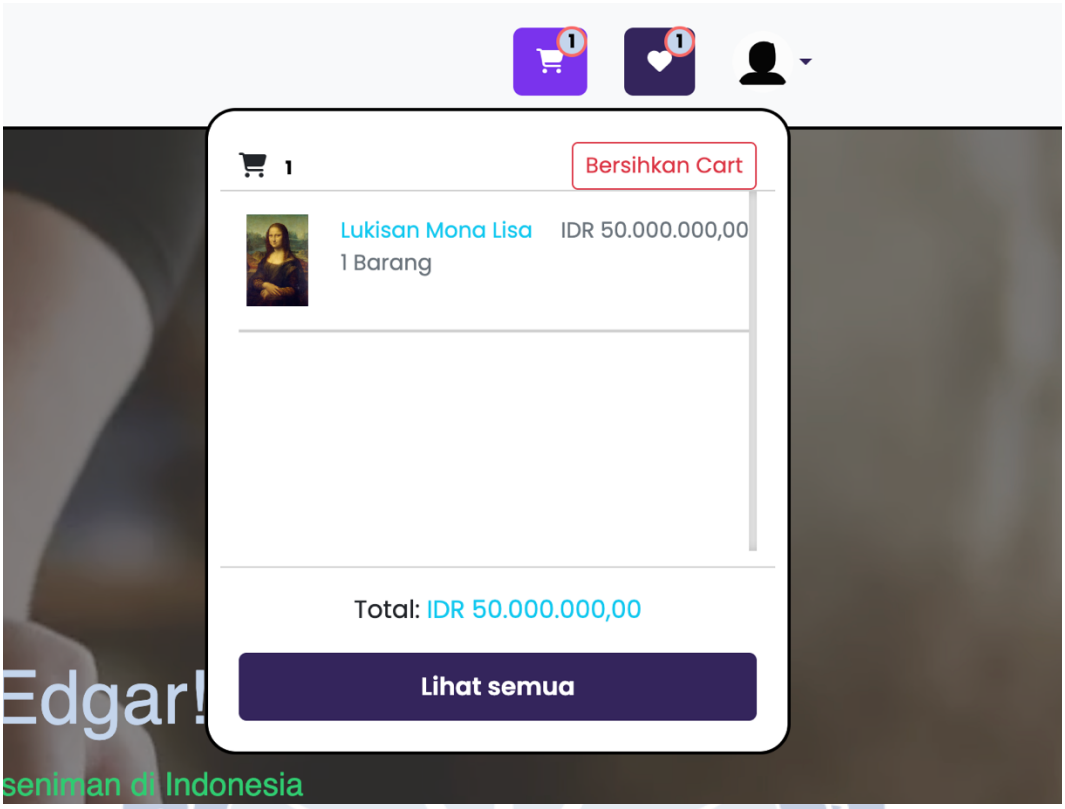
Gambar 5.56 adalah gambar implementasi antarmuka dari *dropdown wishlist* yang dapat digunakan oleh pengguna untuk menyimpan produk-produk yang ingin dilihat kembali di masa mendatang. Selain itu, Gambar 5.57 menunjukan antarmuka dari *dropdown cart* atau keranjang belanja yang digunakan oleh pengguna untuk mengisi produk yang ingin dibeli. Pada Gambar 5.58 terdapat potongan kode yang mengambil daftar produk yang ada dalam *wishlist* pengguna dari *database*, Gambar 5.59 menunjukan potongan kode yang mengambil daftar produk yang ada dalam *cart* pengguna dari *database*.

Sistem mengimplementasikan fitur *cart* dan *wishlist* pada *header website* sehingga *frontend* akan selalu mencoba mengambil informasi mengenai isi *cart* dan *wishlist* pengguna setiap kali pengguna membuka halaman baru. Hal ini membuat kedua fitur harus memiliki performa *read* yang tinggi sehingga data *cart* dan *wishlist* pengguna ditempatkan pada *database* Redis untuk mempercepat waktu akses seperti yang terlihat pada Gambar 5.59. Selain itu, sistem juga menggunakan konsep *multi-threading* untuk mempersingkat waktu pengambilan informasi detail

produk dari *database* dalam kode *wishlist* dan *cart* seperti yang terlihat pada Gambar 5.58 jika pengguna mengakses informasi detail dari *cart* dan *wishlist* yang mengharuskan sistem untuk menunjukkan informasi seperti nama dan gambar produk. Jika pengguna tidak membuka *dropdown*, maka sistem tidak perlu menghubungi *database* MySQL untuk mendapatkan informasi detail produk.



Gambar 5.56 *Dropdown Wishlist*



Gambar 5.57 *Dropdown Cart*

```java
@Override
public void handle(Context context) {
    UserSessionContainer userSessionContainer = AuthHandler.authenticateUser(context);
    if (userSessionContainer == null) {
        return;
    }

    Wishlist wishlist = WishlistManager.getWishlist(userSessionContainer.getUser().id());
    JSONObject wishlistJson = wishlist.toJSON();
    wishlistJson.put("items_in_cart", wishlist.productId().size());
    List<Product> products = new ArrayList<>();
    CountDownLatch latch = new CountDownLatch(wishlist.productId().size());
    for (UUID productId : wishlist.productId()) {
        CompletableFuture.runAsync(() -> {
            try {
                Optional<Product> product = ProductService.getProduct(productId);
                if (product.isPresent()) {
                    products.add(product.get());
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
            latch.countDown();
        });
    }

    try {
        latch.await();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    JSONArray productsJson = new JSONArray();
    for (Product product : products) {
        productsJson.add(product.toJSON());
    }
    wishlistJson.put("products", productsJson);

    StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Berhasil mendapatkan wishlist.", name: "wishlist", wishlistJson);
```

Gambar 5.58 Potongan Kode *Wishlist*

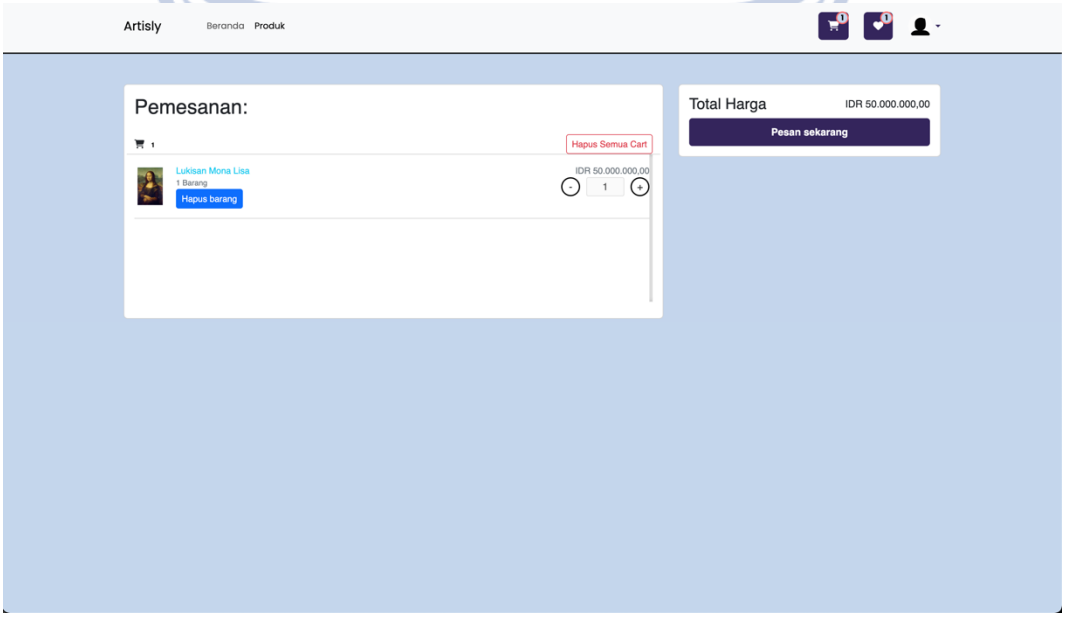
```java
public static Cart getCart(UUID userId)
{
    try(Jedis jedis = Artisly.instance.getRedis().getJedis().getResource()) {
        String cartKey = "cart::" + userId.toString();
        Map<String, String> cartData = jedis.hgetAll(cartKey);
        Map<UUID, Integer> result = new HashMap<>();

        for (Map.Entry<String, String> entry : cartData.entrySet()) {
            UUID productId = UUID.fromString(entry.getKey());
            int quantity = Integer.parseInt(entry.getValue());
            result.put(productId, quantity);
        }

        jedis.expire(cartKey, (int) TimeUnit.DAYS.toSeconds(CART_EXPIRATION_DAYS));
        Cart cart = new Cart(userId, result);
        return cart;
    }
}
```

Gambar 5.59 Potongan Kode *Cart*

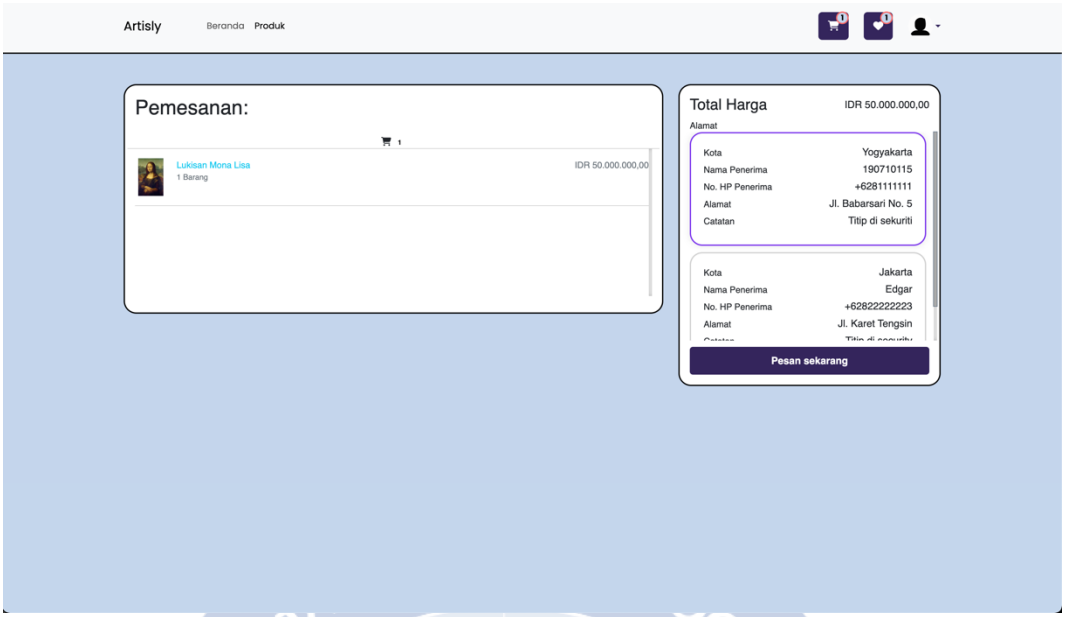
Gambar 5.60 adalah gambar implementasi antarmuka dari halaman pemesanan dan Gambar 5.61 adalah implementasi antarmuka dari halaman pemilihan alamat. Halaman pemesanan merupakan halaman yang menunjukkan seluruh produk yang ada dalam *cart* pengguna, dan halaman konfirmasi pesanan adalah halaman di mana pengguna dapat memilih alamat melalui *form* pemilihan alamat dan membuat pesanan. Gambar 5.62 adalah potongan kode untuk pembuatan pesanan.

Kode pada Gambar 5.62 merupakan bagian terpenting dalam sistem ini. Kode tersebut merupakan implementasi dari fungsi transaksi atau pembuatan pesanan. Sebelum membuat pesanan, sistem akan mengambil koneksi dari *database* dan mematikan *auto commit,* sistem melakukan beberapa cek seperti mengambil alamat pengguna, mengambil isi *cart* pengguna, menghapus produk yang sudah tidak memiliki *stock*, dan memastikan bahwa *cart* pengguna memiliki isi. Setelah itu, daftar produk dari keranjang belanja diproses satu per satu.

Setiap produk dianggap sebagai setiap pesanan tersendiri, sehingga untuk setiap produk, sistem akan membuat *object* ImmutableAddress dan ImmutableProduct yang tidak dapat diubah, dan menyalin gambar produk. Selanjutnya, sistem akan mengirimkan pemberitahuan kepada pembeli dan penjual melalui *email* dan membersihkan produk-produk yang sudah dipesan dari *cart* pengguna seperti yang terlihat pada Gambar 5.62.



Gambar 5.60 Halaman Pemesanan

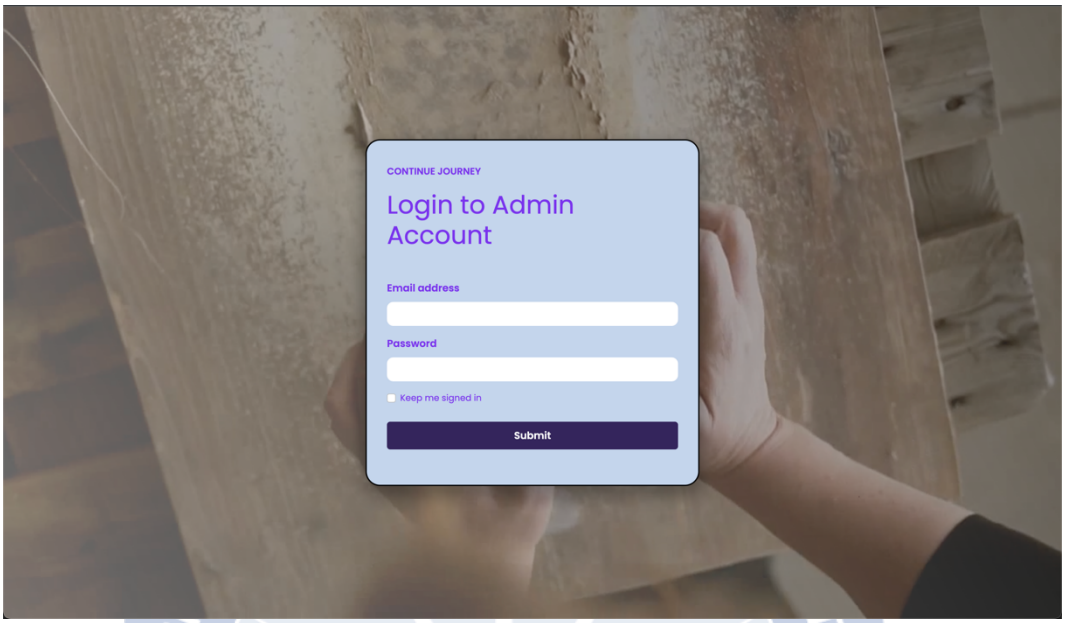


Gambar 5.61 Halaman Pemilihan Alamat

```
try(Connection connection = Artisly.instance.getMySQL().getConnection())
{
    connection.setAutoCommit(false);
    User buyer = CRUDSUser.readById(connection, userId).get();
    Cart cart = CartService.getCart(userId);
    Map<Product, Integer> productList = cart.getFullProductList();
    productList.entrySet().removeIf(entry -> entry.getKey().stock() < entry.getValue());

    if (productList.size() == 0)
        throw new InvalidOrderException("Tidak ada produk yang bisa dipesan dari cart");

    for (Map.Entry<Product, Integer> entry : productList.entrySet()) {
        Product product = entry.getKey();
        int quantity = entry.getValue();

        try {
            ImmutableAddress immutableAddress = CRUDSImmutableAddress.create(connection, userId, addressId, address.receiverName(), address.receiverPhone(), addre
            ImmutableProduct immutableProduct = CRUDSImmutableProduct.create(connection, product);
            Artisly.instance.getMediaService().copyProductImagesToImmutable(product.id(), immutableProduct.id());
            Order order = CRUDSOrder.create(connection, product.storeId(), userId, immutableProduct.id(), quantity, orderPrice: product.price() * quantity, immutabl
            CRUDSOrderRecord.create(connection, order.id(), OrderStatus.CREATED, Optional.of( value: "Order created by " + buyer.firstName() + " " + buyer.lastName

            Optional<Store> optionalStore = CRUDSStore.readById(connection, product.storeId());
            if (optionalStore.isPresent()) {
                Store store = optionalStore.get();
                User storeOwner = CRUDSUser.readById(connection, store.userId()).get();
                EmailService.queueEmail(storeOwner.email(), subject: "New order #" + order.id(), body: "You have a new order #" + order.id().toString().split( regex:
                byte[] outputStream = generatePdf(order.id(), buyerName: buyer.firstName() + " " + buyer.lastName(), buyer.email(), sellerName: storeOwner.firstName
                EmailService.queueEmail(buyer.email(), subject: "Order #" + order.id() + " created", body: "Your order #" + order.id().toString().split( regex: "-")[
            }
            CartService.removeProductFromCart(userId, product.id().toString());
        } catch (SQLException e) {
            connection.rollback();
            throw e;
        } catch (DocumentException e) {
            e.printStackTrace();
            throw new InvalidOrderException("Failed to generate PDF");
        }
    }
    connection.commit();
```

Gambar 5.62 Potongan Kode Pembuatan Pesanan

Gambar 5.63 merupakan implementasi dari antarmuka halaman *login admin* yang digunakan untuk masuk ke dalam *dashboard admin*. Gambar 5.8 merupakan potongan kode *login*.

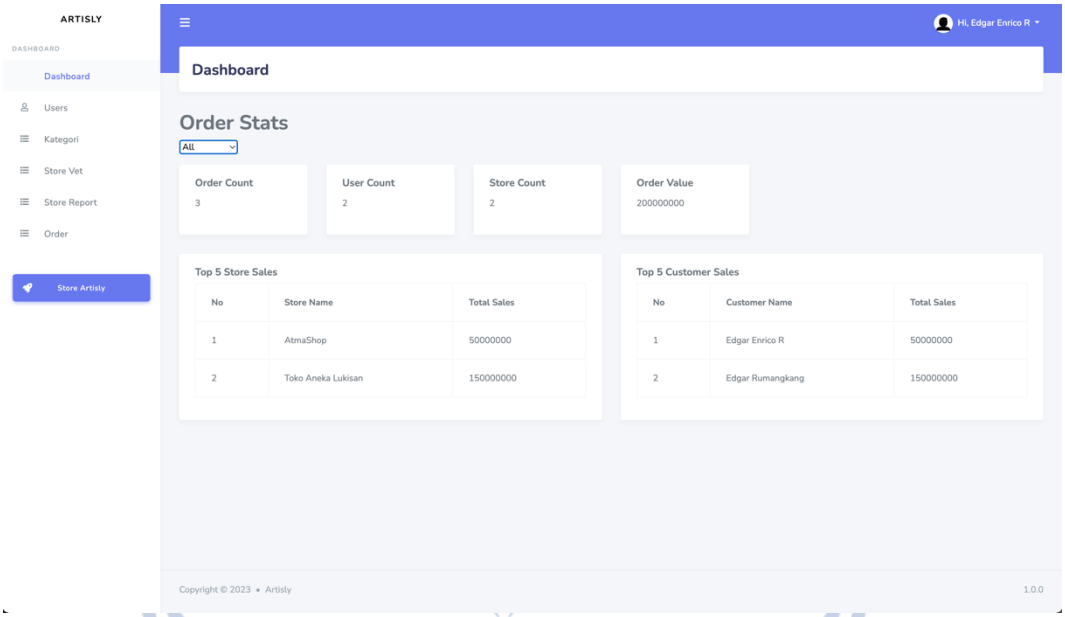


Gambar 5.63 Halaman *Login Admin*

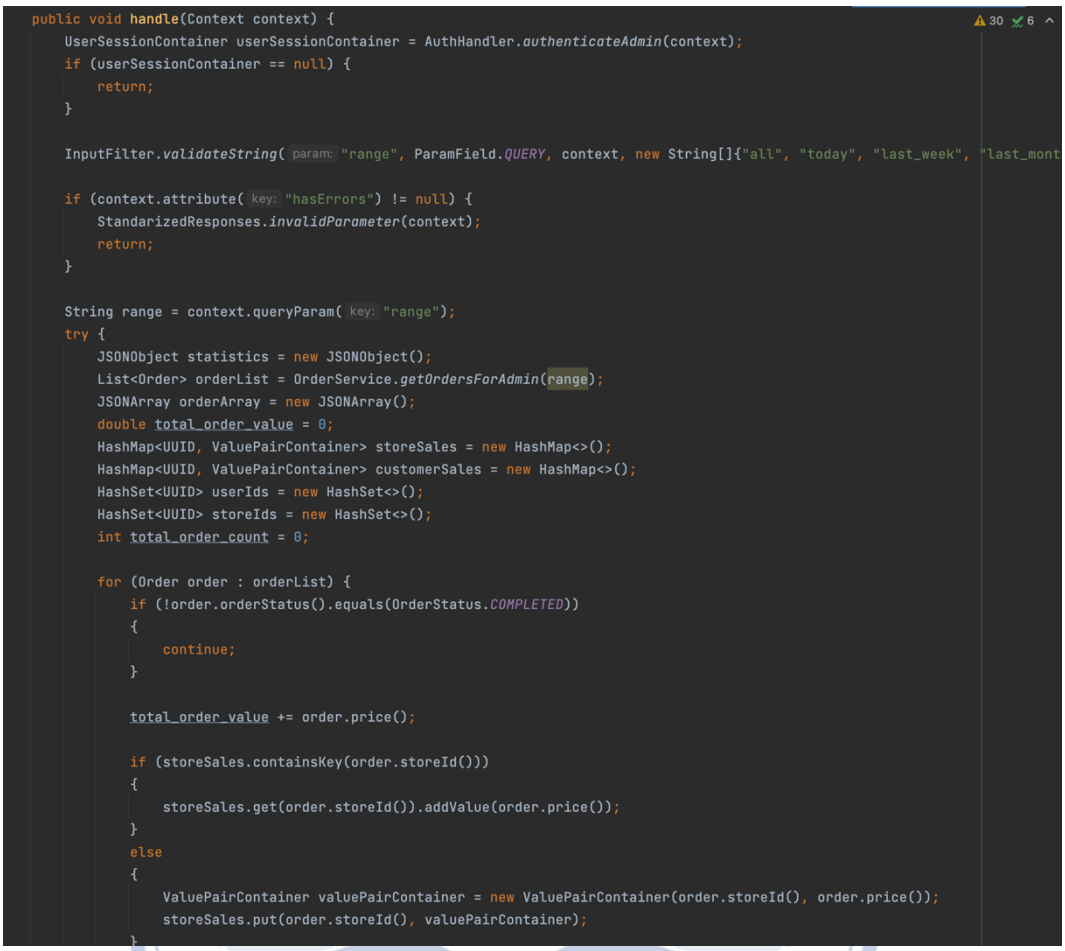
Gambar 5.64 merupakan implementasi dari antarmuka halaman utama *dashboard admin* yang menunjukkan statistik penjualan di *platform*. Untuk menghasilkan statistik tersebut, *frontend* menghubungi *API endpoint* bagian *orders* untuk menghasilkan laporan tersebut yang kemudian ditampilkan pada *frontend*. Statistik tersebut menunjukkan jumlah pesanan yang dibuat, jumlah pengguna unik yang melakukan pembelian, jumlah toko unik yang melakukan penjualan, dan nilai pemesanan dalam rupiah yang diselesaikan dalam rentang waktu yang dipilih. Selain itu, sistem juga menampilkan *top 5* toko dan pembeli dengan nilai *sales* terbanyak. Potongan kode dari *API endpoint index orders for admin* yang digunakan untuk menghasilkan laporan tersebut terdapat pada Gambar 5.65.

Potongan kode pada Gambar 5.65 adalah kode dari *presentation layer* yang bekerja untuk menampilkan statistik *platform* dengan cara mengambil seluruh order sesuai dengan rentang waktu tertentu, kemudian melakukan *looping* untuk mencari jumlah pembeli yang unik, jumlah toko yang menjual, dan statistik pemesanan lainnya.

Presentation layer pada Gambat 5.65 juga bertanggung jawab untuk menentukan jumlah pembeli dengan pembelian terbanyak dan jumlah toko dengan penjualan terbanyak. Untuk menemukan statistik ini, diperlukan algoritma yang menggabungkan seluruh pembelian pengguna tertentu dan penjualan toko tertentu dan melakukan *sort* data pembelian pengguna dan penjualan toko dengan algoritma *mergesort*. Untuk mempercepat operasi, *sort* data pembelian pengguna dan data penjualan toko dilakukan secara *parallel.*



Gambar 5.64 Halaman Utama *Dashboard Admin*



```
public void handle(Context context) {
    UserSessionContainer userSessionContainer = AuthHandler.authenticateAdmin(context);
    if (userSessionContainer == null) {
        return;
    }

    InputFilter.validateString( param: "range", ParamField.QUERY, context, new String[]{"all", "today", "last_week", "last_mont

    if (context.attribute( key: "hasErrors") != null) {
        StandarizedResponses.invalidParameter(context);
        return;
    }

    String range = context.queryParam( key: "range");
    try {
        JSONObject statistics = new JSONObject();
        List<Order> orderList = OrderService.getOrdersForAdmin(range);
        JSONArray orderArray = new JSONArray();
        double total_order_value = 0;
        HashMap<UUID, ValuePairContainer> storeSales = new HashMap<>();
        HashMap<UUID, ValuePairContainer> customerSales = new HashMap<>();
        HashSet<UUID> userIds = new HashSet<>();
        HashSet<UUID> storeIds = new HashSet<>();
        int total_order_count = 0;

        for (Order order : orderList) {
            if (!order.orderStatus().equals(OrderStatus.COMPLETED))
            {
                continue;
            }

            total_order_value += order.price();

            if (storeSales.containsKey(order.storeId()))
            {
                storeSales.get(order.storeId()).addValue(order.price());
            }
            else
            {
                ValuePairContainer valuePairContainer = new ValuePairContainer(order.storeId(), order.price());
                storeSales.put(order.storeId(), valuePairContainer);
            }
```

Gambar 5.65 Potongan Kode *Index Order Admin*

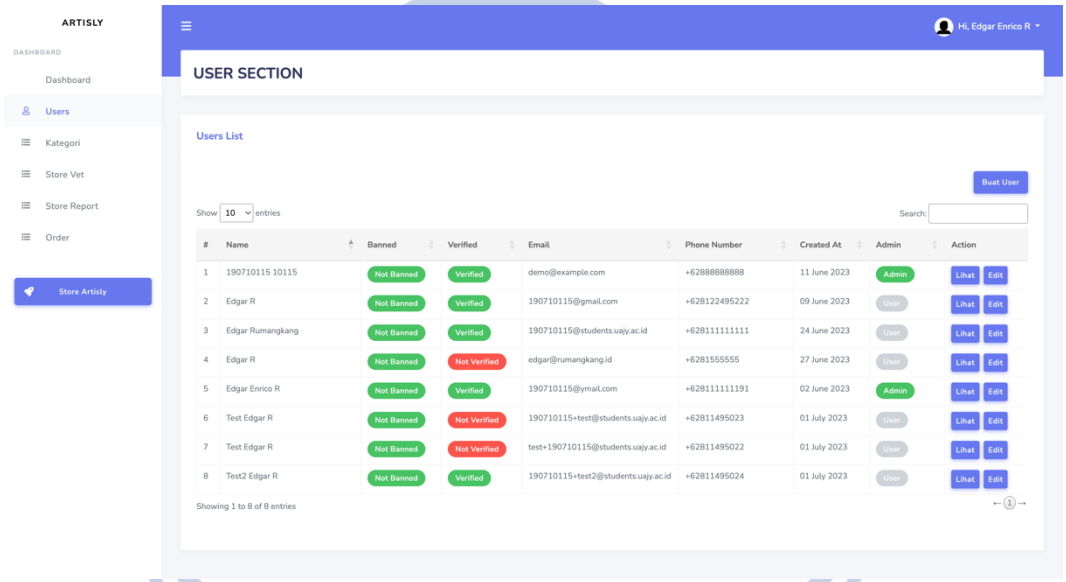
Gambar 5.66 merupakan halaman *user section* di mana halaman ini dapat digunakan oleh pengelola sistem untuk mengelola pengguna. Dalam halaman ini, ditampilkan seluruh pengguna yang sudah terdaftar di sistem. Selain itu, terdapat *form* pembuatan pengguna dan *form* perubahan pengguna. Potongan kode yang melayani permintaan pendaftaran pengguna seperti pada Gambar 5.67.

Pada *form* pembuatan pengguna yang terdapat pada Gambar 5.68, *admin* bisa mendaftarkan pengguna baru dengan memasukkan informasi pengguna. Sistem akan mengirimkan *email* yang berisi *password* akun pengguna kepada *email* pengguna. *Password* tidak akan ditampilkan oleh *admin* untuk alasan keamanan dan untuk menjaga integritas sistem. Potongan kode yang melayani registrasi pengguna oleh *admin* dapat dilihat pada Gambar 5.69.

Pada *form* perubahan pengguna yang terdapat pada Gambar 5.70, *admin* dapat mengubah atribut pengguna serta memblokir pengguna atau memberikan status

admin kepada pengguna agar pengguna dapat *login* ke *dashboard admin*. Untuk menjaga integritas sistem, maka *admin* tidak diperbolehkan untuk mengubah *email*, kata sandi, dan nomor KTP. Selain itu, *admin* tidak bisa mengubah status verifikasi *email* pengguna. Potongan kode yang melayani perubahan atribut pengguna dapat dilihat pada Gambar 5.71. Pada kode perubahan atribut pengguna, sistem akan mengirimkan *email* jika *admin* mengubah atribut *banned* dari pengguna.

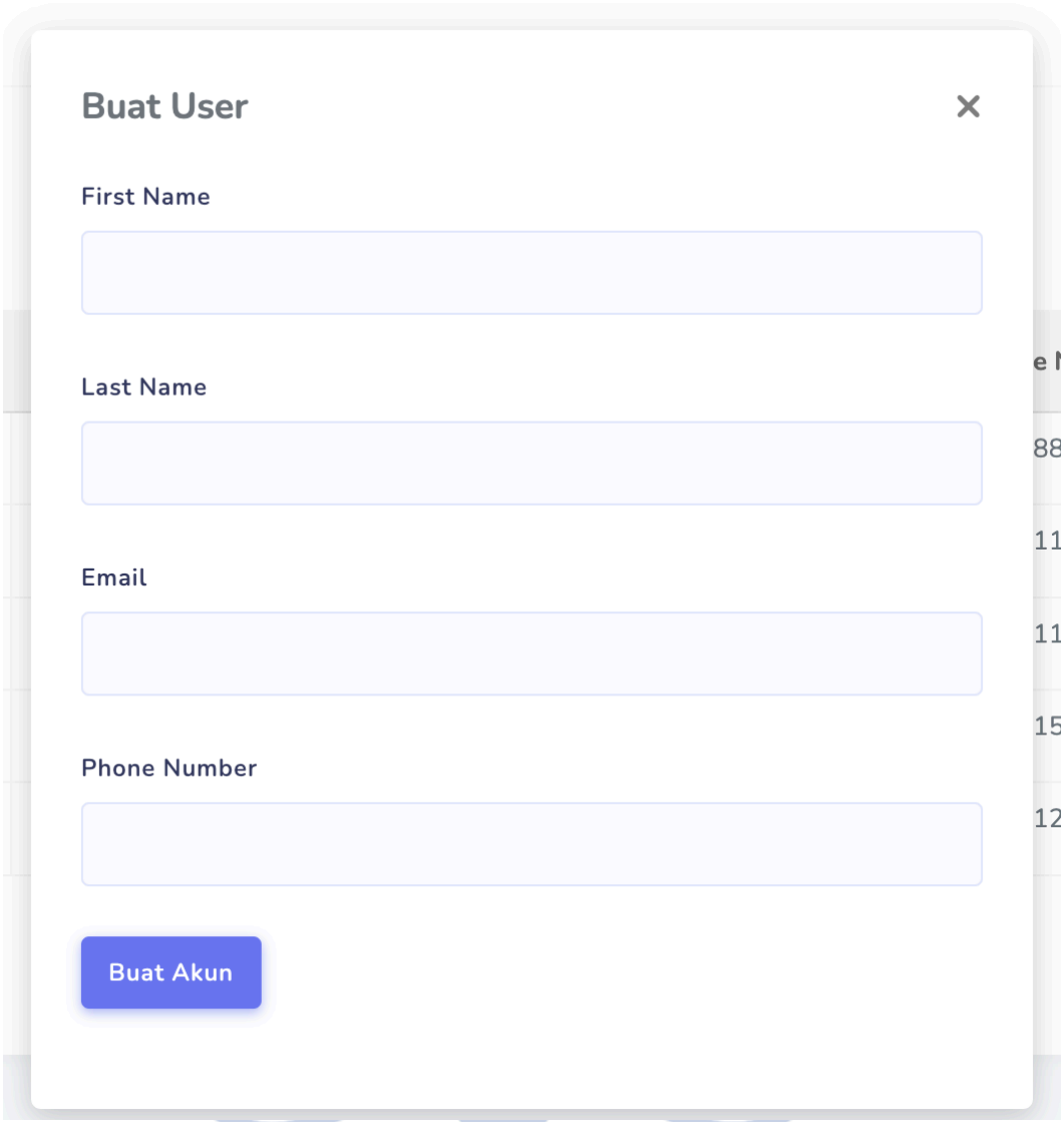
Selain melihat daftar pengguna, *admin* juga dapat melihat detail pengguna melalui *modal* yang menunjukkan informasi detail pengguna pada Gambar 5.72.



Gambar 5.66 Halaman *User Section*

```
public class Index implements APIHandler {
    EdgarG4m3r
    @Override
    public void handle(Context context) {

        UserSessionContainer userSessionContainer = AuthHandler.authenticateAdmin(context);
        if (userSessionContainer == null) {
            return;
        }

        try
        {
            List<User> users = UserService.index();
            JSONArray usersArray = new JSONArray();
            for (User user : users)
            {
                usersArray.add(user.toJSON());
            }

            StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Berhasil mengambil index user", name: "response",
        }
        catch (SQLException e)
        {
            e.printStackTrace();
            StandarizedResponses.generalFailure(context, code: 500, message: "SQL_EXCEPTION", friendlyMessage: "Failed to retrieve users"
        }
    }
}
```

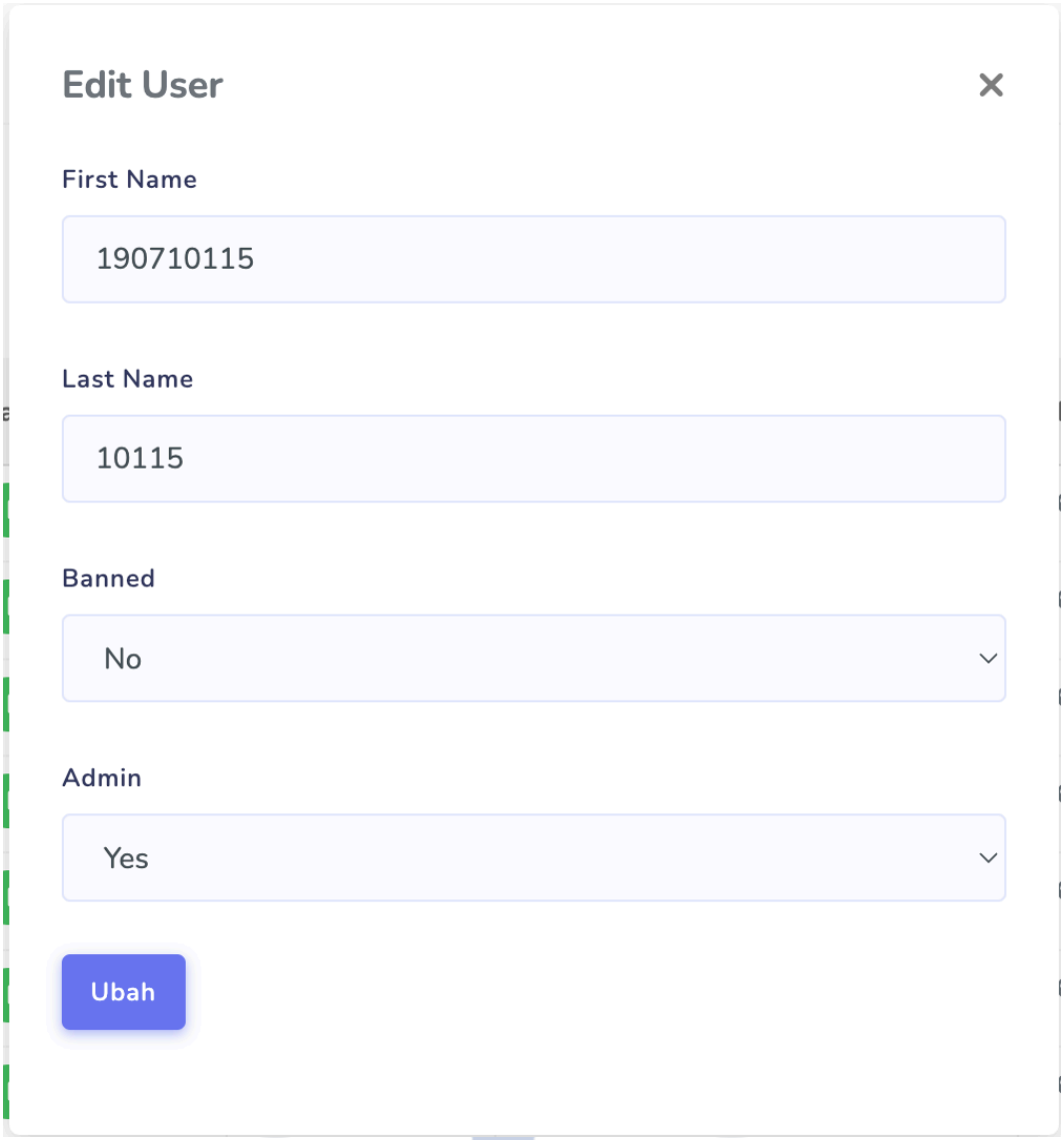
Gambar 5.67 Potongan Kode *User Section*



Gambar 5.68 *Form* Pembuatan Pengguna

```
1 usage   EdgarG4m3r *
public static void registerAdmin(String email, String firstName, String lastName, String phoneNumber) throws SQLException, EmailT
    char[] password = HashEngine.randomPassword();
    register(email, password, firstName, lastName, phoneNumber);

    EmailService.queueEmail(email,
            subject: "New Password",
            body: "Your account was registered on Artisly by an Administrator. Please change your password immediately. New Passw

    try
    {
        requestVerificationCode(email);
    }
    catch (Exception e)
    {
        //ignored, because user can request verification code later and all of the exceptions are user related.
    }

}
```

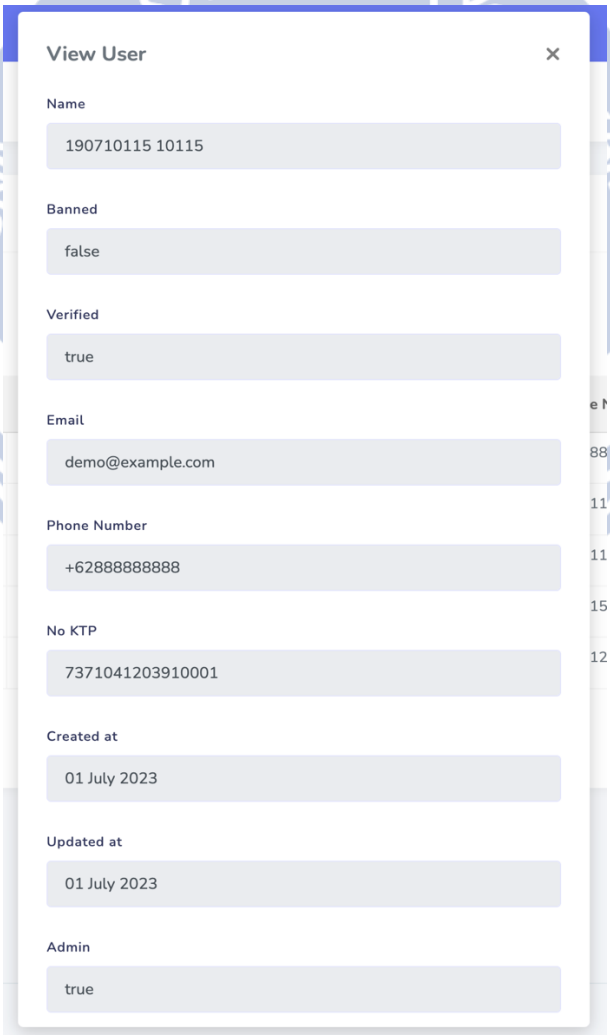
Gambar 5.69 Potongan Kode Pembuatan Pengguna



Gambar 5.70 *Form* Perubahan Pengguna

```
1 usage   1 EdgarG4m3r
public static boolean editUser(UUID userId, String firstName, String lastName, boolean banned, boolean admin) throws SQLException,
{
    try(Connection connection = Artisly.instance.getMySQL().getConnection())
    {
        Optional<User> optionalUser = CRUDSUser.readById(connection, userId);
        if (optionalUser.isEmpty())
        {
            throw new UserNotFoundException("Tidak dapat menemukan pengguna dengan ID + " + userId + "! Harap coba lagi!");
        }
        User user = optionalUser.get();
        //check if user is receiving a ban
        if (banned && !user.banned())
        {
            Artisly.instance.getSessionManager().invalidateAllSessions(userId);
            EmailService.queueEmail(user.email(),  subject: "Account Banned",  body: "Your account has been banned. If you believe thi
        }
        //check if user is getting unbanned
        else if (!banned && user.banned()) {
            EmailService.queueEmail(user.email(),  subject: "Account Unbanned",  body: "Your account has been unbanned. You may now lo
        }
        return CRUDSUser.update(connection, user.id(), user.email(), user.password(), firstName, lastName, user.phoneNumber(), ban
    }
}
```

Gambar 5.71 Potongan Kode Perubahan Pengguna

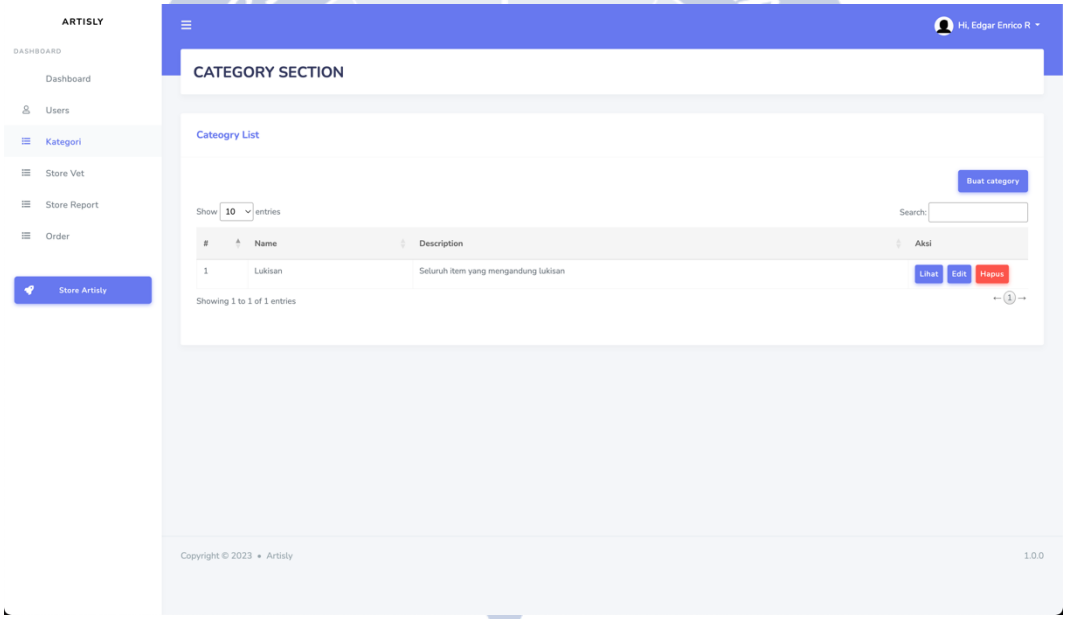


Gambar 5.72 *Modal* Detail Pengguna

Gambar 5.73 merupakan tampilan dari halaman *category section* di mana *admin* dapat mengelola kategori dalam *platform*. Data yang ditampilkan pada halaman kategori diambil dari *endpoint index category* yang ada pada Gambar 5.74 yang merupakan potongan kode *index category*.

Gambar 5.75 merupakan tampilan dari *form* pembuatan kategori. *Form* ini berfungsi untuk membuat kategori baru pada *platform*. Saat pembuatan kategori, *frontend* akan memanggil *endpoint* dari *create category* yang terdapat pada Gambar 5.76

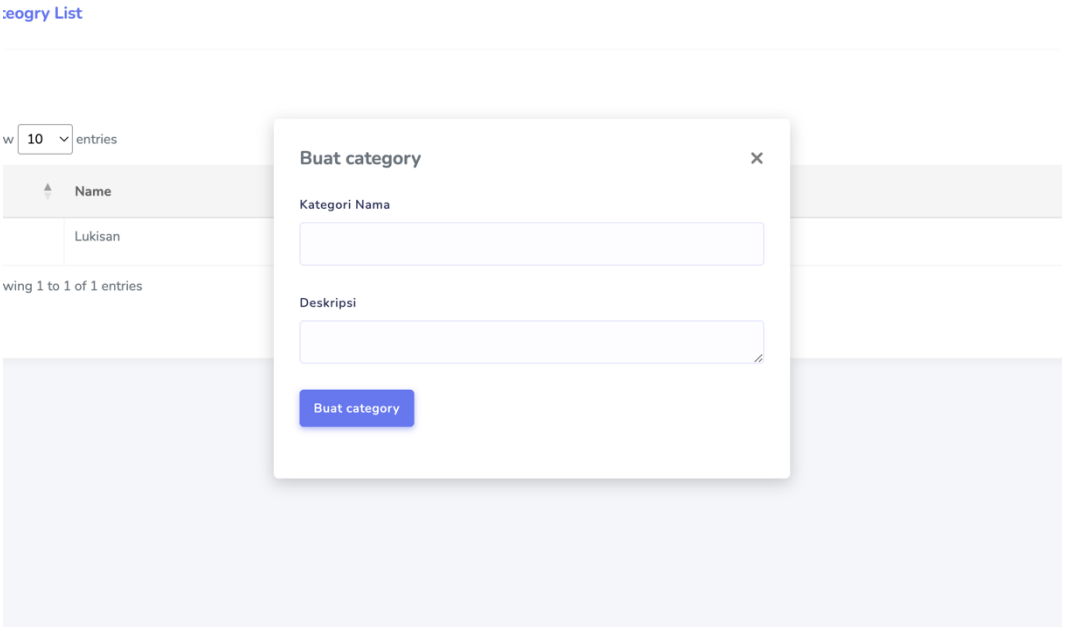
Gambar 5.77 merupakan tampilan dari *form* perubahan kategori. Saat *form* ini ditampilkan pada *admin*, sistem akan mengisi informasi kategori yang sudah ada pada *field* form. Saat perubahan kategori, *frontend* akan menghubungi *endpoint* dari *edit category* yang terdapat pada Gambar 5.78.



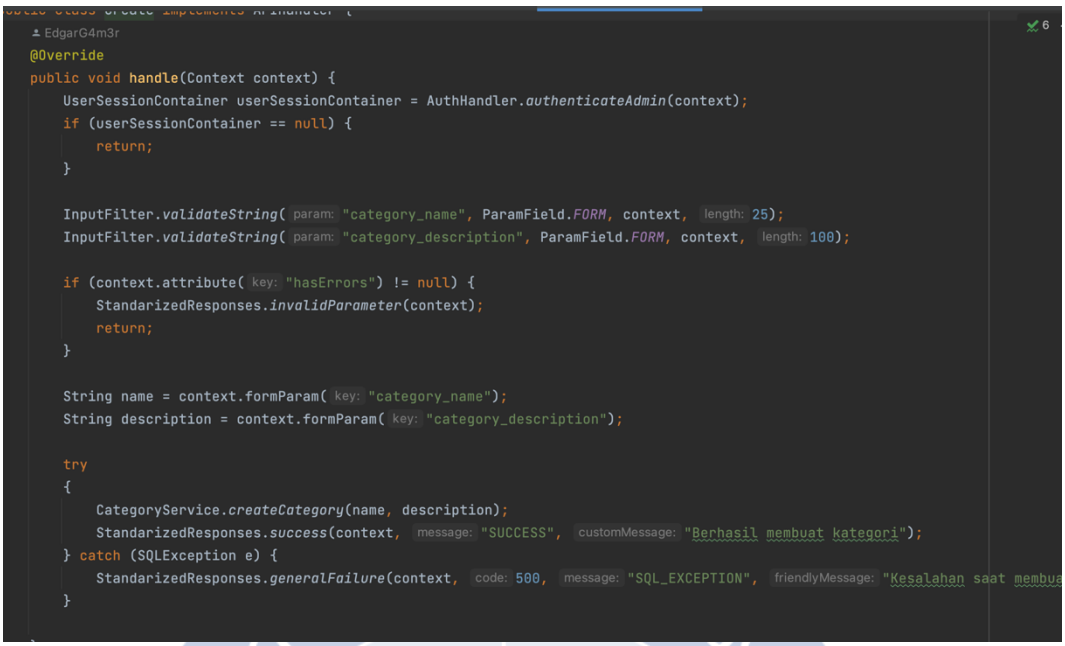
Gambar 5.73 Halaman *Category Section*

```java
@Override
public void handle(Context context) {
    UserSessionContainer userSessionContainer = AuthHandler.authenticateAdmin(context);
    if (userSessionContainer == null) {
        return;
    }

    try
    {
        JSONArray response = new JSONArray();
        List<Category> categoryList = CategoryService.getCategories();
        for (Category category : categoryList)
        {
            JSONObject categoryObject = new JSONObject();
            categoryObject.put("id", category.id());
            categoryObject.put("name", category.name());
            categoryObject.put("description", category.description());
            response.add(categoryObject);
        }
        StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Berhasil mengambil data kategori", name: "data"
    }
    catch (SQLException e)
    {
        e.printStackTrace();
        StandarizedResponses.generalFailure(context, code: 500, message: "SQL_EXCEPTION", friendlyMessage: "Kesalahan saat mengambi
    }

}
```

Gambar 5.74 Potongan Kode *Category Section*

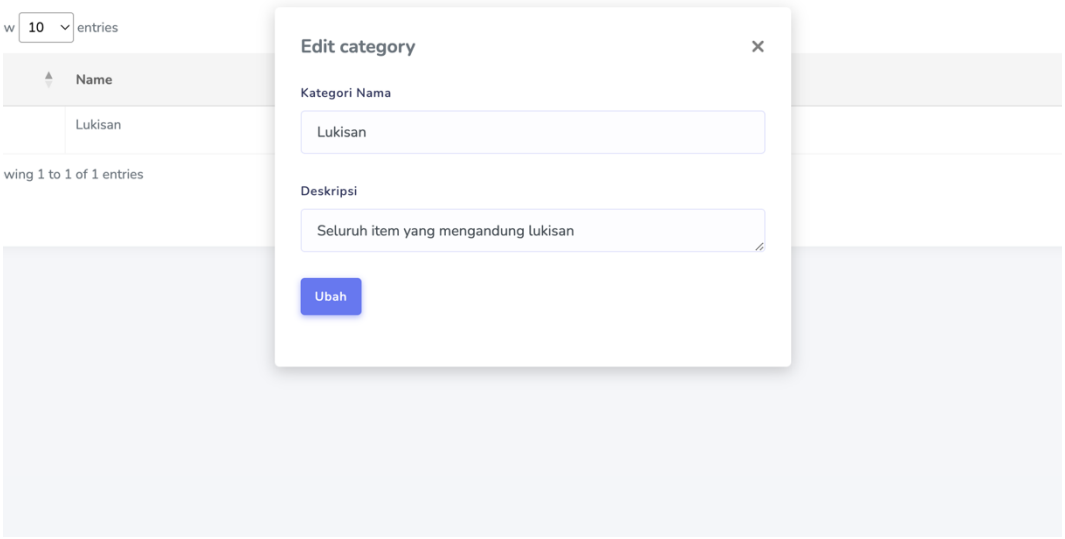


Gambar 5.75 *Form* Pembuatan Kategori



```
EdgarG4m3r
@Override
public void handle(Context context) {
    UserSessionContainer userSessionContainer = AuthHandler.authenticateAdmin(context);
    if (userSessionContainer == null) {
        return;
    }

    InputFilter.validateString( param: "category_name", ParamField.FORM, context, length: 25);
    InputFilter.validateString( param: "category_description", ParamField.FORM, context, length: 100);

    if (context.attribute( key: "hasErrors") != null) {
        StandarizedResponses.invalidParameter(context);
        return;
    }

    String name = context.formParam( key: "category_name");
    String description = context.formParam( key: "category_description");

    try
    {
        CategoryService.createCategory(name, description);
        StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Berhasil membuat kategori");
    } catch (SQLException e) {
        StandarizedResponses.generalFailure(context, code: 500, message: "SQL_EXCEPTION", friendlyMessage: "Kesalahan saat membua
    }
```

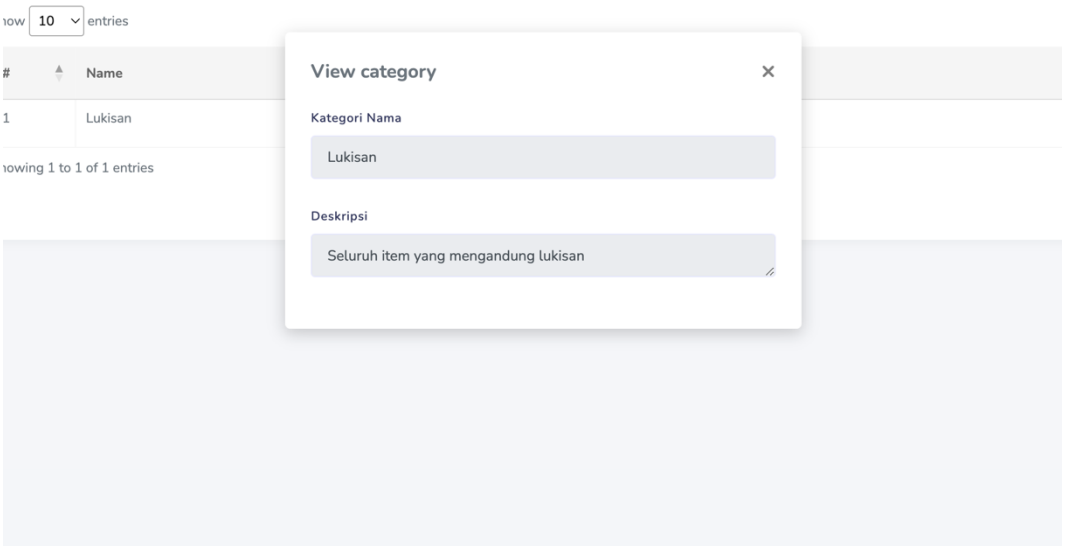
Gambar 5.76 Potongan Kode Pembuatan Kategori



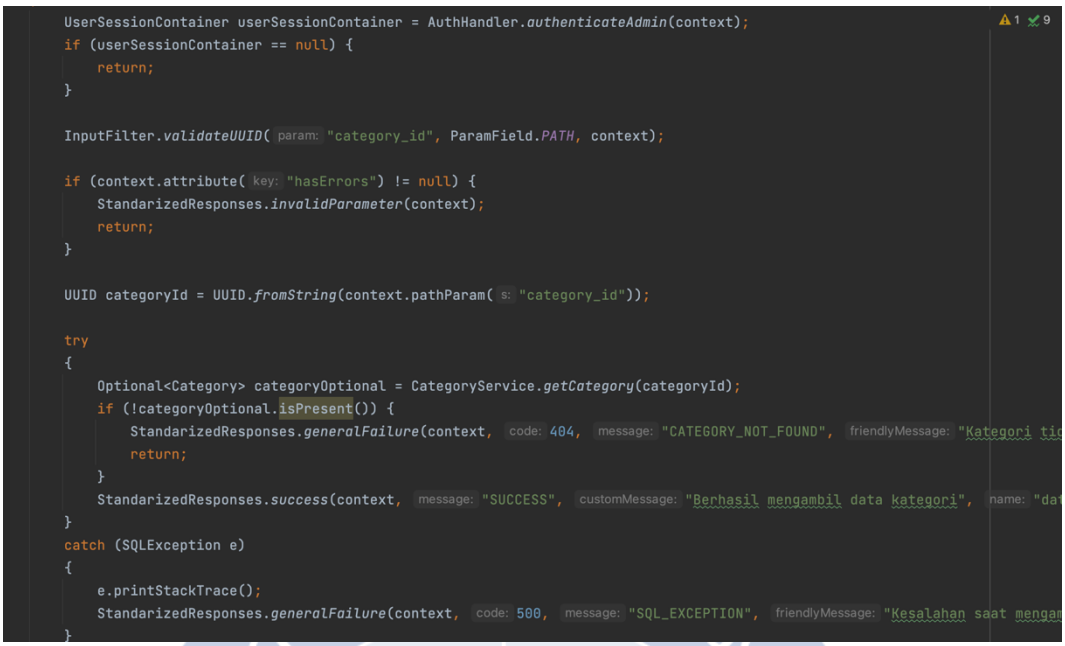
Gambar 5.77 *Form* Perubahan Kategori

```
InputFilter.validateUUID( param: "category_id", ParamField.PATH, context);

if (context.attribute( key: "hasErrors") != null) {
    StandarizedResponses.invalidParameter(context);
    return;
}

UUID categoryId = UUID.fromString(context.pathParam( s: "category_id"));
String name = context.formParam( key: "category_name");
String description = context.formParam( key: "category_description");

try
{
    boolean result = CategoryService.updateCategory(categoryId, name, description);
    if (result)
    {
        StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Berhasil mengupdate kategori!");
    }
    else
    {
        StandarizedResponses.generalFailure(context, code: 400, message: "FAILED", friendlyMessage: "Gagal mengupdate kategori!"
    }

} catch (SQLException e) {
    StandarizedResponses.generalFailure(context, code: 500, message: "SQL_EXCEPTION", friendlyMessage: "Gagal saat mengupdate ka
} catch (CategoryNotExist e) {
    StandarizedResponses.generalFailure(context, code: 500, message: "CATEGORY_NOT_EXIST", e.getMessage());
```

Gambar 5.78 Potongan Kode Perubahan Kategori



Gambar 5.79 *Modal* Informasi Detail Kategori

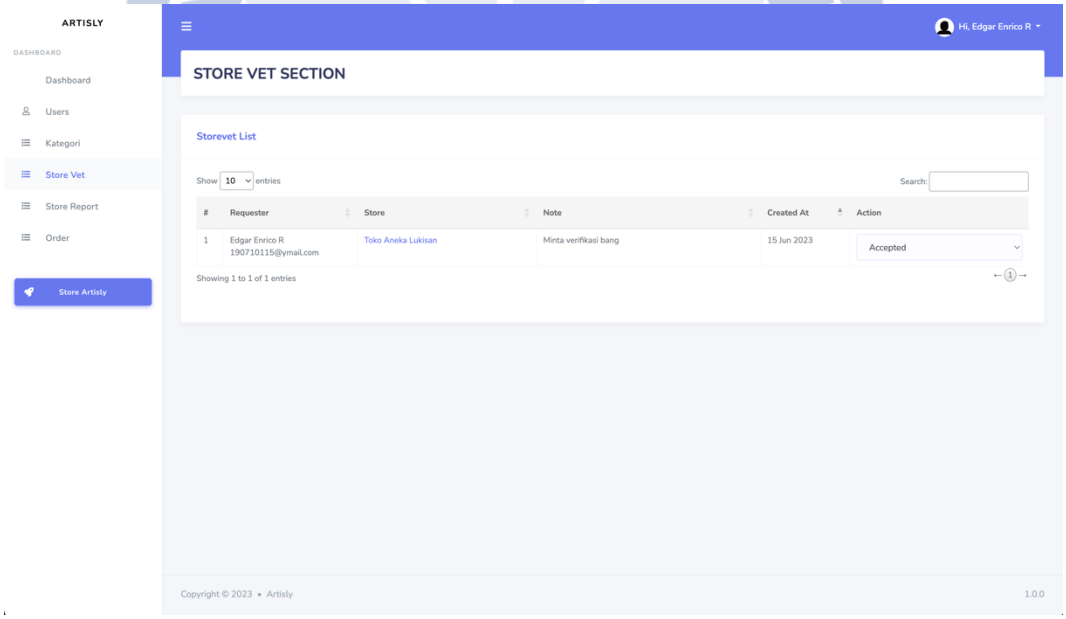


```
UserSessionContainer userSessionContainer = AuthHandler.authenticateAdmin(context);
if (userSessionContainer == null) {
    return;
}

InputFilter.validateUUID( param: "category_id", ParamField.PATH, context);

if (context.attribute( key: "hasErrors") != null) {
    StandarizedResponses.invalidParameter(context);
    return;
}

UUID categoryId = UUID.fromString(context.pathParam( s: "category_id"));

try
{
    Optional<Category> categoryOptional = CategoryService.getCategory(categoryId);
    if (!categoryOptional.isPresent()) {
        StandarizedResponses.generalFailure(context, code: 404, message: "CATEGORY_NOT_FOUND", friendlyMessage: "Kategori tid
        return;
    }
    StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Berhasil mengambil data kategori", name: "da
}
catch (SQLException e)
{
    e.printStackTrace();
    StandarizedResponses.generalFailure(context, code: 500, message: "SQL_EXCEPTION", friendlyMessage: "Kesalahan saat mengam
}
```

Gambar 5.80 Potongan Kode Detail Kategori

Gambar 5.81 merupakan tampilan dari halaman *store vet section* di mana *admin* dapat mengelola permintaan verifikasi toko. Untuk menampilkan seluruh permintaan verifikasi toko, *frontend* memanggil *endpoint index store vet* yang terdapat pada Gambar 5.82. *Backend* akan mengambil seluruh informasi kategori dari dalam *database* kemudian mengirimkan seluruh kategori pada *frontend* menggunakan format *JSON Array*.

Untuk mengubah status permohonan verifikasi toko, *frontend* menghubungi *endpoint edit store vet* untuk mengubah status permohonan menjadi diterima atau ditolak. Potongan kode perubahan status permohonan verifikasi toko terdapat pada Gambar 5.83. Pada saat status permohonan verifikasi toko diterima, maka sistem akan mengubah status toko menjadi "Terverifikasi" dan juga akan mengirimkan *email* kepada penjual. Saat status permohonan ditolak, sistem akan menghapus permohonan verifikasi toko dari dalam *database* dan mengirimkan *email* kepada penjual bahwa permohonan verifikasi ditolak.



Gambar 5.81 Halaman *Store Vet Section*

```java
public void handle(Context context) {
    UserSessionContainer userSessionContainer = AuthHandler.authenticateAdmin(context);
    if (userSessionContainer == null) {
        return;
    }

    try
    {
        List<StoreVet> storeVets = StoreVetService.getStoreVetRequestForAdmins();
        JSONArray storeVetArray = new JSONArray();
        for (StoreVet storeVet : storeVets)
        {
            try
            {
                JSONObject storeVetEntry = new JSONObject();
                storeVetEntry.put("vet", storeVet.toJSON());
                User requester = UserService.show(storeVet.userId());
                Store store = StoreService.getStoreById(storeVet.storeId()).get();
                storeVetEntry.put("requester", requester.toJSON());
                storeVetEntry.put("store", store.toJSON());

                storeVetArray.add(storeVetEntry);
            }
            catch (Exception e)
            {
                continue;
            }
        }

        StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Berhasil mengambil data vet", name: "response",
        return;
    } catch (SQLException e) {
        e.printStackTrace();
        StandarizedResponses.generalFailure(context, code: 500, message: "SQL_EXCEPTION", friendlyMessage: "Terjadi kesalahan pada
    }
```

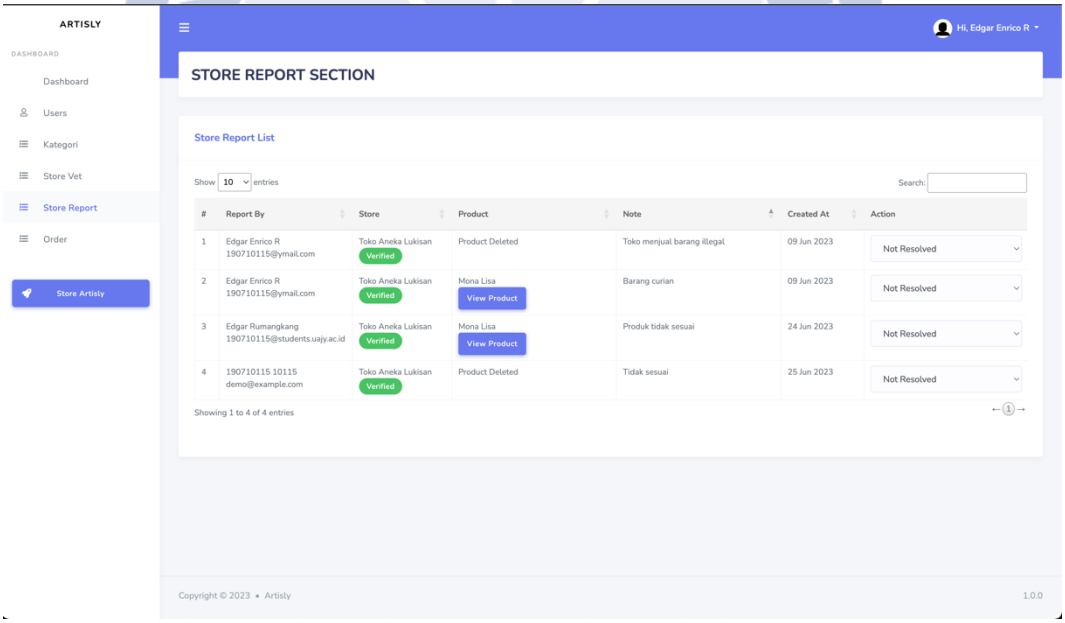
Gambar 5.82 Potongan Kode *Store Vet Section*

```java
public void handle(Context context) {
    UserSessionContainer userSessionContainer = AuthHandler.authenticateAdmin(context);
    if (userSessionContainer == null) {
        return;
    }

    InputFilter.validateUUID( param: "vet_id", ParamField.PATH, context);
    InputFilter.validateBoolean( parm: "accepted", ParamField.FORM, context);

    if (context.attribute( key: "hasErrors") != null) {
        StandarizedResponses.invalidParameter(context);
        return;
    }

    UUID vetId = UUID.fromString(context.pathParam( s: "vet_id"));
    boolean accepted = Boolean.parseBoolean(context.formParam( key: "accepted"));

    try
    {
        if (accepted)
        {
            StoreVetService.acceptVetRequest(vetId);
        }
        else
        {
            StoreVetService.rejectVetRequest(vetId);
        }
        StandarizedResponses.success(context, message: "SUCCESS", customMessage: "Berhasil mengubah status verifikasi");
    }
    catch (SQLException e)
    {
        e.printStackTrace();
        StandarizedResponses.generalFailure(context, code: 500, message: "SQL_EXCEPTION", friendlyMessage: "Gagal saat mengubah sta
    } catch (StoreNotExist e) {
        e.printStackTrace();
        StandarizedResponses.generalFailure(context, code: 404, message: "STORE_NOT_EXIST", friendlyMessage: "Toko tidak ditemukan"
    }
}
```

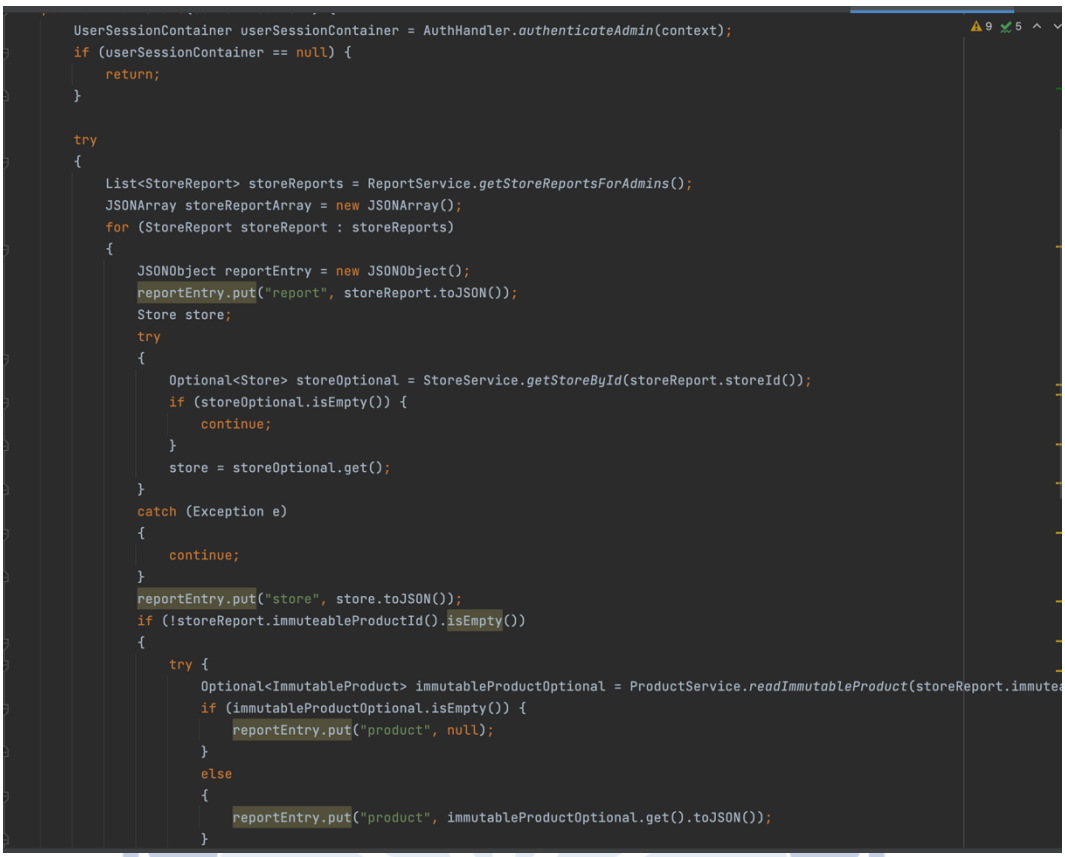
Gambar 5.83 Potongan Kode Perubahan Status Verifikasi Toko

Gambar 5.84 merupakan tampilan dari halaman *store report section* di mana *admin* dapat mengelola laporan aduan toko dan produk. Untuk menampilkan seluruh laporan aduan toko dan produk, *frontend* memanggil *endpoint index store report* yang terdapat pada Gambar 5.85. *Backend* akan mengambil seluruh laporan yang tersimpan dari dalam *database* kemudian mengambil seluruh informasi ImmutableProduct dari *database* sebelum mengirimkan seluruh laporan aduan pada *frontend* menggunakan format *JSON Array*.

Untuk mengubah status laporan aduan, *frontend* menghubungi *endpoint edit storereport* untuk mengubah status laporan aduan menjadi diterima atau ditolak. Potongan kode perubahan status laporan aduan terdapat pada Gambar 5.86. Jika *admin* menerima status laporan, maka penjual akan dikirimkan peringatan melalui *email.* Jika *admin* menolak laporan, maka pelapor akan dikirimkan *email* pemberitahuan bahwa laporan aduan sudah diperiksa tapi belum ada tindakan yang dapat diambil oleh *admin.*

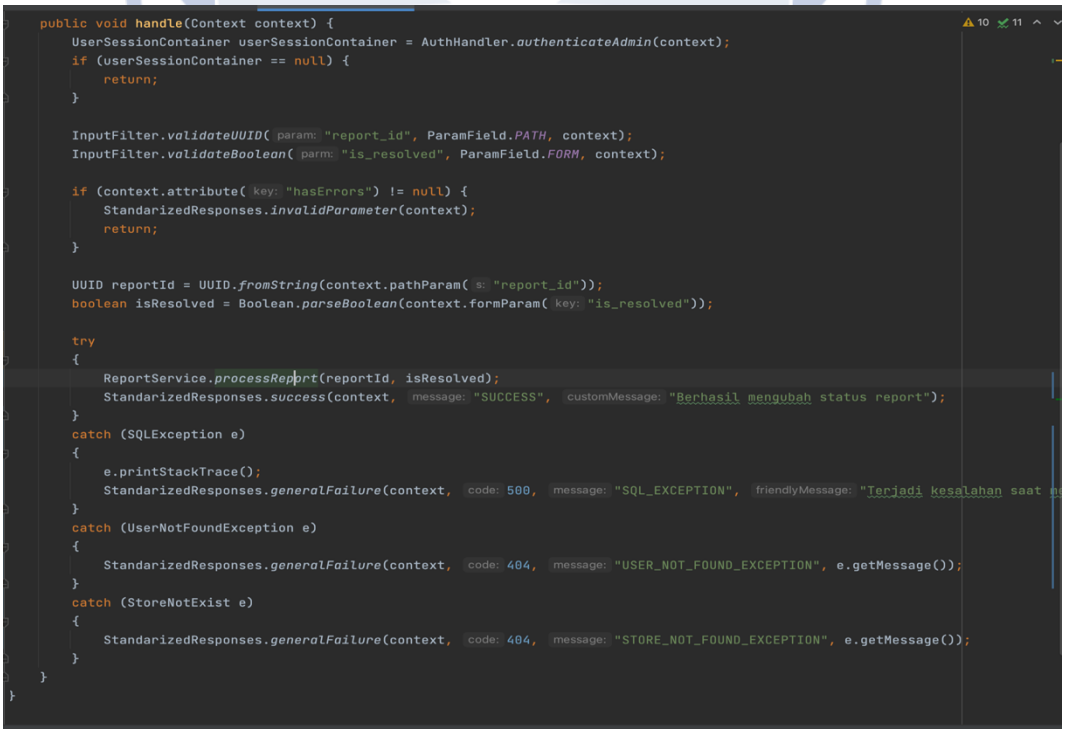


Gambar 5.84 Halaman *Store Report Section*



```
UserSessionContainer userSessionContainer = AuthHandler.authenticateAdmin(context);
if (userSessionContainer == null) {
    return;
}

try
{
    List<StoreReport> storeReports = ReportService.getStoreReportsForAdmins();
    JSONArray storeReportArray = new JSONArray();
    for (StoreReport storeReport : storeReports)
    {
        JSONObject reportEntry = new JSONObject();
        reportEntry.put("report", storeReport.toJSON());
        Store store;
        try
        {
            Optional<Store> storeOptional = StoreService.getStoreById(storeReport.storeId());
            if (storeOptional.isEmpty()) {
                continue;
            }
            store = storeOptional.get();
        }
        catch (Exception e)
        {
            continue;
        }
        reportEntry.put("store", store.toJSON());
        if (!storeReport.immuteableProductId().isEmpty())
        {
            try {
                Optional<ImmutableProduct> immutableProductOptional = ProductService.readImmutableProduct(storeReport.immute
                if (immutableProductOptional.isEmpty()) {
                    reportEntry.put("product", null);
                }
                else
                {
                    reportEntry.put("product", immutableProductOptional.get().toJSON());
                }
```

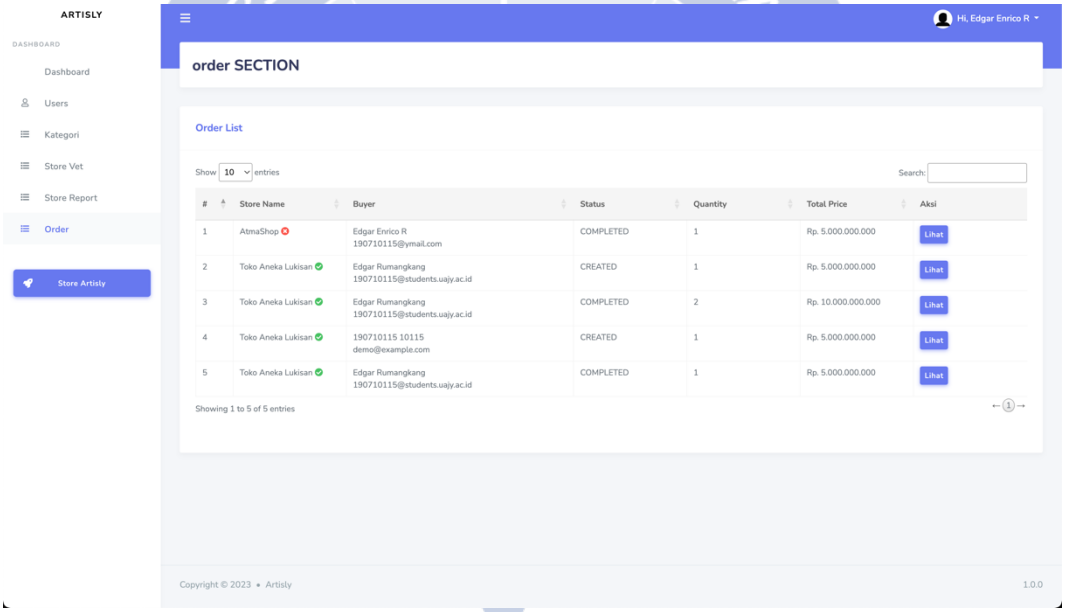
Gambar 5.85 Potongan Kode *Store Report Section*



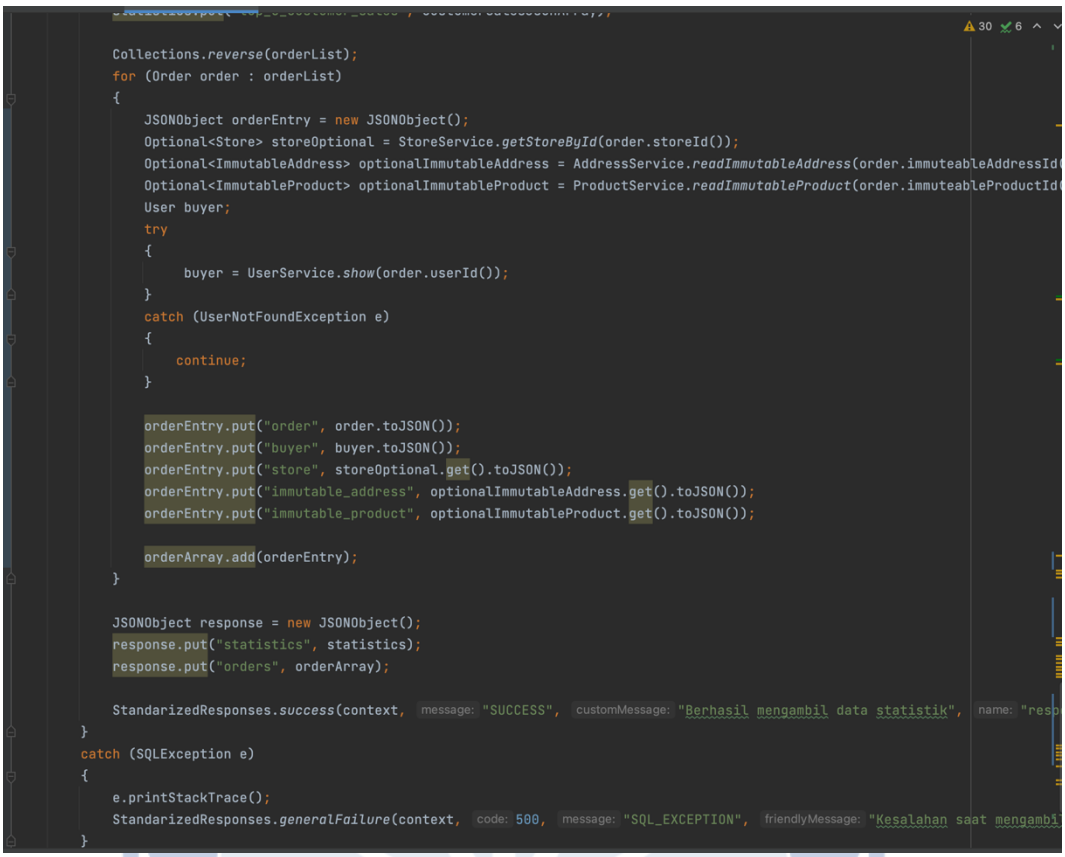
```
public void handle(Context context) {
    UserSessionContainer userSessionContainer = AuthHandler.authenticateAdmin(context);
    if (userSessionContainer == null) {
        return;
    }

    InputFilter.validateUUID( param: "report_id", ParamField.PATH, context);
    InputFilter.validateBoolean( parm: "is_resolved", ParamField.FORM, context);

    if (context.attribute( key: "hasErrors") != null) {
        StandarizedResponses.invalidParameter(context);
        return;
    }

    UUID reportId = UUID.fromString(context.pathParam( s: "report_id"));
    boolean isResolved = Boolean.parseBoolean(context.formParam( key: "is_resolved"));

    try
    {
        ReportService.processReport(reportId, isResolved);
        StandarizedResponses.success(context,  message: "SUCCESS",  customMessage: "Berhasil mengubah status report");
    }
    catch (SQLException e)
    {
        e.printStackTrace();
        StandarizedResponses.generalFailure(context,  code: 500,  message: "SQL_EXCEPTION",  friendlyMessage: "Terjadi kesalahan saat m
    }
    catch (UserNotFoundException e)
    {
        StandarizedResponses.generalFailure(context,  code: 404,  message: "USER_NOT_FOUND_EXCEPTION", e.getMessage());
    }
    catch (StoreNotExist e)
    {
        StandarizedResponses.generalFailure(context,  code: 404,  message: "STORE_NOT_FOUND_EXCEPTION", e.getMessage());
    }
}
}
```

Gambar 5.86 Potongan Kode Perubahan Status Laporan Aduan

Gambar 5.87 merupakan tampilan dari halaman *order section* di mana *admin* dapat melihat seluruh pesanan yang dibuat di dalam *platform*. Untuk menampilkan seluruh, *frontend* memanggil *endpoint index order* yang terdapat pada Gambar 5.88. *Backend* akan mengambil seluruh pesanan dari dalam *database* kemudian mengirimkan informasi tersebut kepada *frontend* menggunakan format *JSON Array*. Karena *endpoint* tersebut juga digunakan oleh halaman utama *dashboard admin* untuk menampilkan statistik, maka informasi yang dikirim dari *backend* juga mengandung statistik *platform* tetapi *frontend* tidak akan membaca informasi tersebut. Alasan kenapa kedua fungsionalitas tersebut menggunakan satu *endpoint* karena kedua fungsionalitas tersebut memerlukan informasi yang sama dari *database* sehingga menggabungkan kedua fungsionalitas tersebut dalam satu *endpoint* merupakan keputusan untuk menghindari kode yang *redundant*.



Gambar 5.87 Halaman *Order Section*



```
        Collections.reverse(orderList);
        for (Order order : orderList)
        {
            JSONObject orderEntry = new JSONObject();
            Optional<Store> storeOptional = StoreService.getStoreById(order.storeId());
            Optional<ImmutableAddress> optionalImmutableAddress = AddressService.readImmutableAddress(order.immuteableAddressId
            Optional<ImmutableProduct> optionalImmutableProduct = ProductService.readImmutableProduct(order.immuteableProductId
            User buyer;
            try
            {
                buyer = UserService.show(order.userId());
            }
            catch (UserNotFoundException e)
            {
                continue;
            }

            orderEntry.put("order", order.toJSON());
            orderEntry.put("buyer", buyer.toJSON());
            orderEntry.put("store", storeOptional.get().toJSON());
            orderEntry.put("immutable_address", optionalImmutableAddress.get().toJSON());
            orderEntry.put("immutable_product", optionalImmutableProduct.get().toJSON());

            orderArray.add(orderEntry);
        }

        JSONObject response = new JSONObject();
        response.put("statistics", statistics);
        response.put("orders", orderArray);

        StandarizedResponses.success(context,  message: "SUCCESS",  customMessage: "Berhasil mengambil data statistik",  name: "resp
    }
    catch (SQLException e)
    {
        e.printStackTrace();
        StandarizedResponses.generalFailure(context,  code: 500,  message: "SQL_EXCEPTION",  friendlyMessage: "Kesalahan saat mengambi
    }
```

Gambar 5.88 Potongan Kode *Order Section*

B. Pengujian Fungsionalitas Perangkat Lunak

Pengujian perangkat lunak aplikasi e-commerce dilakukan dengan metode *black box*. Tabel 5.1 mencatat hasil pengujian yang melibatkan pengguna, penjual, dan admin di *website* utama dan website admin. Fokusnya adalah *output* yang diharapkan dari fungsi-fungsi aplikasi. Metode ini memastikan bahwa aplikasi e-commerce beroperasi dengan baik dan memberikan hasil yang diinginkan.

Tabel 5.1 Hasil Pengujian Fungsionalitas

Deskripsi	Prosedur Pengujian	Masukkan	Keluaran yang Diharapkan	Hasil Keluaran	Kesimpulan
Pengujian fungsi *registrasi*	- Masuk ke halaman registrasi - Menekan tombol registrasi	- Nama Depan: kosong - Nama Belakang: kosong - Alamat *Email*: kosong - *Password*: kosong - Konfirmasi *Password*: kosong - Nomor Telepon: kosong	Sistem memberi peringatan	Sistem memberi peringatan	Handal
Pengujian fungsi *registrasi*	- Masuk ke halaman registrasi	- Nama Depan: demo - Nama Belakang: demo - Alamat *Email*: demo@example.com - *Password*: demo	Sistem memberi peringatan	Sistem memberi peringatan	Handal

	- Isi formulir registrasi - Menekan tombol registrasi	- Konfirmasi *Password*: demo - Nomor Telepon: +62800000000			
Pengujian fungsi *registrasi*	- Masuk ke halaman registrasi - Isi formulir registrasi - Menekan tombol registrasi	- Nama Depan: Edgar - Nama Belakang: R - Alamat *Email*: 190710115@students.uajy.ac.id - *Password*: 190710115 - Konfirmasi *Password*: 190710115	Sistem mendaftarkan pengguna dan mengirimkan kode verifikasi *email*.	Sistem mendaftarkan pengguna dan mengirimkan kode verifikasi *email*.	Handal
Pengujian fungsi verifikasi *email*	- Selesaikan registrasi - Masukan kode verifikasi *email* - Menekan tombol verifikasi	- Kode verifikasi: 476170300	Sistem memberikan peringatan.	Sistem memberikan peringatan.	Handal

Pengujian fungsi verifikasi *email*	- Selesaikan registrasi - Masukan kode verifikasi *email* - Menekan tombol verifikasi	- Kode verifikasi: 47617030	Sistem berhasil memverifikasi *email.*	Sistem berhasil memverifikasi *email.*	Handal
Pengujian fungsi *login*	- Masuk ke halaman *login* - Menekan tombol *login*	- Alamat *Email*: kosong - *Password*: kosong	Sistem memberikan peringatan.	Sistem memberikan peringatan.	Handal
Pengujian fungsi *login*	- Masuk ke halaman *login* - Mengisi formulir *login* - Menekan tombol *login*	- Alamat *Email*: 190710115 - *Password*: 190710115	Sistem memberikan peringatan.	Sistem memberikan peringatan.	Handal

Pengujian fungsi *login*	- Masuk ke halaman *login* - Mengisi formulir *login* - Menekan tombol *login*	- Alamat *Email*: 190710115@students.uajy.ac.id - *Password*: password_salah	Sistem memberikan peringatan.	Sistem memberikan peringatan.	Handal
Pengujian fungsi *login*	- Masuk ke halaman *login* - Mengisi formulir *login* - Menekan tombol *login*	- Alamat *Email*: 190710115@students.uajy.ac.id - *Password*: 190710115	Berhasil masuk ke dalam sistem dan sistem mengirimkan pemberitahuan *login* melalui *email*.	Berhasil masuk ke dalam sistem dan sistem mengirimkan pemberitahuan *login* melalui *email*.	Handal
Pengujian fungsi menelusuri produk	- Masuk ke halaman produk	- Kata kunci: kosong	Berhasil menampilkan semua produk	Berhasil menampilkan semua produk.	Handal

Pengujian fungsi menelusuri produk (dengan pencarian)	- Masuk ke halaman produk - Masukkan kata kunci pada *search bar*	- Kata kunci: mona	Sistem menampilkan produk yang mengandung kata kunci “mona”.	Sistem menampilkan produk yang mengandung kata kunci “mona”.	Handal
Pengujian fungsi menelusuri produk (dengan pencarian)	- Masuk ke halaman produk - Masukkan kata kunci pada *search bar*	- Kata kunci: test	Sistem tidak menampilkan produk apapun.	Sistem tidak menampilkan produk apapun.	Handal
Pengujian fungsi melihat detail produk	- Menekan produk yang ingin dilihat - Masuk ke halaman detail produk	Tidak ada masukan	Sistem menampilkan informasi detail produk.	Sistem menampilkan informasi detail produk.	Handal

Pengujian fungsi pelaporan produk	- Masuk ke halaman detail produk - Menekan tombol pelaporan produk - Menekan tombol konfirmasi	- Alasan pelaporan: kosong	Sistem menampilkan peringatan.	Sistem menampilkan peringatan.	Handal
Pengujian fungsi pelaporan produk	- Masuk ke halaman detail produk - Menekan tombol pelaporan produk - Masukkan alasan pelaporan - Menekan tombol konfirmasi	- Alasan pelaporan: Produk tidak sesuai	Sistem berhasil menyimpan laporan.	Sistem berhasil menyimpan laporan.	Handal
Pengujian fungsi penambahan	- Masuk ke halaman detail produk	- Kuantitas: 10000	Sistem menampilkan peringatan.	Sistem menampilkan peringatan.	Handal

produk dalam keranjang	- Masukkan kuantitas produk - Menekan tombol penambahan produk				
Pengujian fungsi *logout*	- Menekan tombol *logout*	Tidak ada masukan	Sistem mengeluarkan pengguna dari akun.	Sistem mengeluarkan pengguna dari akun.	Handal
Pengujian fungsi menambahkan produk dalam *wishlist*	- Masuk ke halaman detail produk	Tidak ada masukan	Sistem menambahkan produk ke dalam *wishlist* pengguna.	Sistem menambahkan produk ke dalam *wishlist* pengguna.	Handal
Pengujian fungsi melihat produk dalam *wishlist*	- Menekan tombol *wishlist* akun pada *header*	Tidak ada masukan	Sistem menampilkan seluruh produk yang ada dalam *wishlist* pengguna.	Sistem menampilkan seluruh produk yang ada	Handal

				dalam *wishlist* pengguna,	
Pengujian fungsi menghapus produk dalam *wishlist*	- Menekan tombol *wishlist akun* pada *header* - Menekan tombol hapus produk dalam *dropdown wishlist*	Tidak ada masukan	Sistem menghapus produk yang dipilih dari dalam *wishlist.*	Sistem menghapus produk yang dipilih dari dalam *wishlist.*	Handal
Pengujian fungsi melihat produk dalam keranjang	- Menekan tombol *cart* pada *header*	Tidak ada masukan	Sistem menampilkan seluruh produk yang ada dalam keranjang pengguna	Sistem menampilkan seluruh produk yang ada dalam keranjang pengguna	Handal

Pengujian fungsi mengubah keranjang	- Masuk ke halaman pemesanan - Mengubah kuantitas produk	- Kuantitas produk: 2000	Sistem menampilkan peringatan stok tidak cukup	Sistem menampilkan peringatan stok tidak cukup	Handal
Pengujian fungsi mengubah keranjang	- Masuk ke halaman pemesanan - Mengubah kuantitas produk	- Kuantitas produk: 2	Sistem berhasil mengubah kuantitas produk pada keranjang	Sistem berhasil mengubah kuantitas produk pada keranjang	Handal
Pengujian fungsi melakukan transaksi	- Masuk ke halaman pemesanan - Menekan tombol pemesanan - Menekan tombol pesan sekarang	- Kota: kosong - Nama Penerima: kosong - No. HP Penerima: kosong - Alamat: kosong - Catatan: kosong	Sistem menampilkan peringatan	Sistem menampilkan peringatan	Handal

Pengujian fungsi melakukan transaksi	- Masuk ke halaman pemesanan - Menekan tombol pemesanan - Menekan tombol pesan sekarang	- Kota: Yogyakarta - Nama Penerima: 190710115 - No. HP Penerima: +6281111111 - Alamat: Jl. Babarsari No. 5 - Catatan: Titip di sekuriti	Sistem berhasil membuat pesanan dan mengirimkan pemberitahuan kepada penjual dan pembeli	Sistem berhasil membuat pesanan dan mengirimkan pemberitahuan kepada penjual dan pembeli	Handal
Pengujian fungsi melihat riwayat pesanan pengguna	- Menekan tombol riwayat pesanan pada *dropdown* pada *header.* - Masuk ke halaman riwayat pemesanan	Tidak ada masukan	Sistem berhasil menampilkan riwayat pemesanan.	Sistem berhasil menampilkan riwayat pemesanan.	Handal
Pengujian fungsi melihat riwayat pesanan penjual	- Menekan tombol *order* pada halaman toko penjual.	Tidak ada masukan	Sistem berhasil menampilkan riwayat pemesanan.	Sistem berhasil menampilkan riwayat pemesanan.	Handal

	- Masuk ke halaman riwayat pemesanan				
Pengujian fungsi mengubah status transaksi dari pembeli	- Menekan tombol riwayat pesanan pada *dropdown* pada *header.* - Masuk ke halaman riwayat pemesanan - Memilih status transaksi	- Status transaksi: Dibatalkan	Sistem berhasil mengubah status pesanan / transaksi.	Sistem berhasil mengubah status pesanan / transaksi.	Handal
Pengujian fungsi mengubah status transaksi dari penjual	- Menekan tombol riwayat pesanan pada *dropdown* pada *header.* - Masuk ke halaman riwayat pemesanan	- Status transaksi: Dibatalkan	Sistem berhasil mengubah status pesanan / transaksi.	Sistem berhasil mengubah status pesanan / transaksi.	Handal

	- Memilih status transaksi				
Pengujian fungsi mengubah status transaksi dari penjual	- Menekan tombol riwayat pesanan pada *dropdown* pada *header.* - Masuk ke halaman riwayat pemesanan - Memilih status transaksi	- Status transaksi: Dikirim	Sistem berhasil mengubah status pesanan / transaksi.	Sistem berhasil mengubah status pesanan / transaksi.	Handal
Pengujian fungsi mengubah status transaksi dari pembeli	- Menekan tombol riwayat pesanan pada *dropdown* pada *header.* - Masuk ke halaman riwayat pemesanan	- Status transaksi: Selesai	Sistem berhasil mengubah status pesanan / transaksi.	Sistem berhasil mengubah status pesanan / transaksi.	Handal

	- Memilih status transaksi				
Pengujian fungsi pembukaan toko	- Masuk ke halaman buka toko - Isi formulir pembukaan toko - Mekekan tombol pembukaan toko	- Logo: *file* gambar berakhiran .png - Banner: *file* gambar berakhiran .png - Nama toko: AtmaShop - Catatan toko: Menjual berbagaimacam kerajinan mahasiswa	Sistem berhasil membuat toko.	Sistem berhasil membuat toko.	Handal
Pengujian fungsi pembukaan toko	- Masuk ke halaman buka toko - Isi formulir pembukaan toko - Mekekan tombol pembukaan toko	- Logo: kosong - Banner: kosong - Nama toko: kosong - Catatan toko: kosong	Sistem menampilkan peringatan.	Sistem menampilkan peringatan.	Handal
Pengujian fungsi pelaporan toko	- Masuk ke halaman detail toko	- Alasan pelaporan: kosong	Sistem menampilkan peringatan	Sistem menampilkan peringatan	Handal

	- Menekan tombol lapor toko - Menekan tombol konfirmasi				
Pengujian fungsi pelaporan toko	- Masuk ke halaman detail toko - Menekan tombol lapor toko - Mengisi formulir pelaporan toko - Menekan tombol konfirmasi	- Alasan pelaporan: Tidak sesuai aturan	Sistem berhasil menyimpan laporan	Sistem berhasil menyimpan laporan	Handal
Pengujian fungsi melihat profil	- Menekan tombol profil pada *header* - Masuk ke halaman profil pengguna	Tidak ada masukan	Sistem berhasil menampilkan profil pengguna	Sistem berhasil menampilkan profil pengguna	Handal

Pengujian fungsi mengubah profil	- Menekan tombol profil pada *header* - Masuk ke halaman profil pengguna - Mengubah informasi yang sudah ada pada formulir perubahan profil - Menekan tombol perubahan profil - Memasukkan konfirmasi *password* - Menekan tombol konfirmasi	- Nama depan: tidak berubah - Nama belakang: Rumangkang - Nomor telepon: tidak berubah - Nomor KTP: tidak berubah	Sistem berhasil mengubah informasi profil	Sistem berhasil mengubah informasi profil	Handal

Pengujian fungsi penggantian kata sandi	- Menekan tombol profil pada *header* - Masuk ke halaman profil pengguna - Memasukkan kata sandi saat ini, kata sandi baru, dan konfirmasi kata sandi pada formulir penggantian kata sandi - Menekan tombol konfirmasi	- Kata sandi saat ini: bukan_190710115 - Kata sandi baru: demo - Konfirmasi kata sandi baru: demo	Sistem menampilkan peringatan	Sistem menampilkan peringatan	Handal
Pengujian fungsi penggantian kata sandi	- Menekan tombol profil pada *header* - Masuk ke halaman profil pengguna	- Kata sandi saat ini: 190710115 - Kata sandi baru: demo - Konfirmasi kata sandi baru: 190710115	Sistem menampilkan peringatan	Sistem menampilkan peringatan	Handal

	- Memasukkan kata sandi saat ini, kata sandi baru, dan konfirmasi kata sandi pada formulir penggantian kata sandi - Menekan tombol konfirmasi				
Pengujian fungsi penggantian kata sandi	- Menekan tombol profil pada *header* - Masuk ke halaman profil pengguna - Memasukkan kata sandi saat ini, kata sandi baru, dan konfirmasi kata sandi pada formulir	- Kata sandi saat ini: 190710115 - Kata sandi baru: demo - Konfirmasi kata sandi baru: demo	Sistem berhasil mengubah kata sandi akun	Sistem berhasil mengubah kata sandi akun	Handal

	penggantian kata sandi - Menekan tombol konfirmasi				
Pengujian fungsi melihat alamat	- Menekan tombol alamat pada *header* - Masuk ke halaman alamat	Tidak ada masukan	Sistem berhasil menampilkan alamat pengguna	Sistem berhasil menampilkan alamat pengguna	Handal
Pengujian fungsi menambahkan alamat	- Menekan tombol alamat pada *header* - Masuk ke halaman alamat - Menekan tombol penambahan alamat - Mengisi formulir penambahan alamat	- Nama penerima: Edgar - No telepon penerima: +62811111111 - Alamat penerima: Jl. Babarsari No. 4 - Kota: Yogyakarta - Catatan: Titip di security	Sistem berhasil menambahkan alamat.	Sistem berhasil menambahkan alamat.	Handal

	- Menekan tombol konfirmasi				
Pengujian fungsi mengubah alamat	- Menekan tombol alamat pada *header* - Masuk ke halaman alamat - Menekan tombol *update* alamat - Mengisi formulir perubahan alamat - Menekan tombol konfirmasi	- Nama penerima: tidak berubah - No telepon penerima: +62822222223 - Alamat penerima: Jl. Karet Tengsin - Kota: Jakarta - Catatan: Titip di resepsionis	Sistem berhasil mengubah alamat.	Sistem berhasil mengubah alamat.	Handal
Pengujian fungsi menghapus alamat	- Menekan tombol alamat pada *header* - Masuk ke halaman alamat	Tidak ada masukan	Sistem berhasil menghapus alamat.	Sistem berhasil menghapus alamat.	Handal

	- Menekan tombol penghapusan alamat				
Pengujian fungsi menghapus produk	- Masuk ke halaman toko - Menekan tombol penghapusan produk	Tidak ada masukan	Sistem berhasil menghapus produk.	Sistem berhasil menghapus produk.	Handal
Pengujian fungsi mengubah produk	- Masuk ke halaman toko - Menekan tombol pengubahan produk - Mengisi formulir perubahan produk - Menekan tombol konfirmasi	- Kategori: tidak berubah - Nama produk: Lukisan Mona Lisa - Deskripsi produk: tidak berubah - Harga produk: tidak berubah - Stok produk: 2	Sistem berhasil mengubah produk.	Sistem berhasil mengubah produk.	Handal
Pengujian fungsi menambahkan produk	- Masuk ke halaman toko	- Kategori: Lukisan - Nama produk: Mona Lisa - Deskripsi produk: Lukisan mona lisa	Sistem berhasil menambahkan produk.	Sistem berhasil menambahkan produk.	Handal

	- Menekan tombol penambahan produk - Mengisi formulir penambahan produk - Menekan tombol konfirmasi	- Harga produk: 50000000 - Stok produk: 1 - Gambar produk: *file* berakhiran .png			
Pengujian fungsi mengubah toko	- Masuk ke halaman toko - Menekan tombol perubahan toko - Memasukan informasi baru - Menekan tombol konfirmasi	- Logo: *file* berakhiran .png - Banner produk: *file* berakhiran .png - Nama toko: Aneka lukisan - Catatan toko: Catatan	Sistem berhasil mengubah toko.	Sistem berhasil mengubah toko.	Handal
Pengujian fungsi melihat statistik produk	- Masuk ke *dashboard admin*	Tidak ada masukan	Sistem berhasil menampilkan statistik *platform*.	Sistem berhasil menampilkan	Handal

				statistik *platform*.	
Pengujian fungsi melihat pengguna	- Masuk ke *dashboard admin* - Masuk ke halaman *users*	Tidak ada masukan	Sistem berhasil menampilkan seluruh pengguna.	Sistem berhasil menampilkan seluruh pengguna.	Handal
Pengujian fungsi mengubah pengguna	- Masuk ke *dashboard admin* - Masuk ke halaman *users* - Menekan tombol perubahan pengguna - Memasukan informasi baru - Menekan tombol konfirmasi	- Nama depan: tidak berubah - Nama belakang: 190710115 - Banned: tidak berubah - Admin: tidak berubah	Sistem berhasil mengubah informasi pengguna.	Sistem berhasil mengubah informasi pengguna.	Handal

Pengujian fungsi melihat kategori	- Masuk ke *dashboard admin* - Masuk ke halaman kategori	Tidak ada masukan	Sistem berhasil menampilkan seluruh kategori.	Sistem berhasil menampilkan seluruh kategori.	Handal
Pengujian fungsi membuat kategori	- Masuk ke *dashboard admin* - Masuk ke halaman kategori - Menekan tombol pembuatan kategori - Mengisi informasi kategori baru - Menekan tombol konfirmasi	- Nama kategori: Tas - Deskripsi: Tas tradisional	Sistem berhasil membuat kategori.	Sistem berhasil membuat kategori.	Handal
Pengujian fungsi mengubah kategori	- Masuk ke *dashboard admin*	- Nama kategori: Tas daun - Deskripsi: Tas tradisional	Sistem berhasil mengubah kategori.	Sistem berhasil mengubah kategori.	Handal

	- Masuk ke halaman kategori - Menekan tombol pengubahan kategori - Mengisi informasi kategori baru - Menekan tombol konfirmasi				
Pengujian fungsi menghapus kategori	- Masuk ke *dashboard admin* - Masuk ke halaman kategori - Menekan tombol penghapusan kategori - Menekan tombol konfirmasi	Tidak ada masukkan.	Sistem berhasil menghapus kategori.	Sistem berhasil menghapus kategori.	Handal

Pengujian fungsi mengirim permintaan verifikasi toko	- Masuk ke halaman toko - Menekan tombol verifikasi toko - Memasukan catatan permohonan - Menekan tombol konfirmasi	- Alasan: Ingin melakukan verifikasi toko	Sistem berhasil mengirim permohonan verifikasi toko.	Sistem berhasil mengirim permohonan verifikasi toko.	Handal
Pengujian fungsi penambahan pengguna	- Masuk ke *dashboard admin* - Masuk ke halaman *users* - Menekan tombol penambahan pengguna - Memasukan informasi baru	- Nama depan: 0115 - Nama belakang: 190710115 - Alamat *email*: 190710115@students.uajy.ac.id - Nomor telepon: +62811110422	Sistem memberikan peringatan	Sistem memberikan peringatan	Handal

	- Menekan tombol konfirmasi				
Pengujian fungsi penambahan pengguna	- Masuk ke *dashboard admin* - Masuk ke halaman *users* - Menekan tombol penambahan pengguna - Memasukan informasi baru - Menekan tombol konfirmasi	- Nama depan: 0115 - Nama belakang: 190710115 - Alamat *email*: edgar@rumangkang.id - Nomor telepon: +62811110422	Sistem mendaftarkan akun pada sistem dan mengirimkan kata sandi melalui *email*.	Sistem mendaftarkan akun pada sistem dan mengirimkan kata sandi melalui *email*.	Handal
Pengujian fungsi melihat permohonan verifikasi toko	- Masuk ke *dashboard admin*	Tidak ada masukan	Sistem berhasil menampilkan seluruh permohonan verifikasi toko.	Sistem berhasil menampilkan seluruh	Handal

	- Masuk ke halaman verifikasi toko / *store vet*			permohonan verifikasi toko.	
Pengujian fungsi melihat laporan aduan toko dan produk	- Masuk ke *dashboard admin* - Masuk ke halaman laporan toko / *store report*	Tidak ada masukan	Sistem berhasil menampilkan seluruh laporan toko dan produk.	Sistem berhasil menampilkan seluruh laporan toko dan produk.	Handal
Pengujian fungsi melihat seluruh pesanan	- Masuk ke *dashboard admin* - Masuk ke halaman pesanan / *orders*	Tidak ada masukan	Sistem berhasil menampilkan seluruh pesanan yang terjadi di dalam *platform*.	Sistem berhasil menampilkan seluruh pesanan yang terjadi di dalam *platform*.	Handal

Pengujian fungsi penambahan pengguna	- Masuk ke *dashboard admin* - Masuk ke halaman *users* - Menekan tombol penambahan pengguna - Memasukan informasi baru - Menekan tombol konfirmasi	- Nama depan: kosong - Nama belakang: kosong - Alamat *email*: kosong - Nomor telepon: kosong	Sistem memberikan peringatan.	Sistem memberikan peringatan.	Handal
Pengujian fungsi mengubah status verifikasi toko	- Masuk ke *dashboard admin* - Masuk ke halaman *users*	- Nama depan: kosong - Nama belakang: kosong - Alamat *email*: kosong - Nomor telepon: kosong	Sistem memberikan peringatan.	Sistem memberikan peringatan.	Handal

	- Menekan tombol penambahan pengguna - Memasukan informasi baru - Menekan tombol konfirmasi				
Pengujian fungsi mengubah status laporan aduan toko dan produk	- Masuk ke *dashboard admin* - Masuk ke halaman laporan toko / *store report* - Memilih status baru	- Status: Resolved	Sistem berhasil mengubah status laporan aduan toko dan atau produk.	Sistem berhasil mengubah status laporan aduan toko dan atau produk.	Handal
Pengujian fungsi melihat	- Masuk ke *dashboard admin*	- Status: Accepted	Sistem berhasil menyimpn status	Sistem berhasil menyimpn status	Handal

permohonan verifikasi toko	- Masuk ke halaman verifikasi toko / *store vet* - Memilih status baru		verifikasi toko yang baru.	verifikasi toko yang baru.	

C. Hasil Pengujian Terhadap Pengguna

Pengujian aplikasi dilakukan untuk memastikan sistem informasi yang dihasilkan oleh penelitian sudah sesuai dengan tujuan penelitian. Pengujian aplikasi dilakukan dengan metode *blackbox testing* di mana setiap *role* dalam sistem diuji oleh Pengguna, Penjual, atau Admin yang akan memberikan masukan setelah selesai melakukan pengujian. Setiap penguji telah melakukan pengujian menggunakan sistem informasi yang sudah di-deploy dengan *database* yang sudah dipenuhi oleh data *dummy.*

Pengujian *role* Pengguna dilakukan oleh 33 orang. Setiap orang menggunakan sistem secara normal mengeksplorasi seluruh fitur-fitur dari sistem informasi kecuali. Khusus *role* Pengguna, sudah diinstruksikan agar tidak melakukan pembukaan toko. Setelah selesai menguji, setiap Pengguna diberikan kuesioner. Setiap responden sebelumnya sudah pernah melakukan pembelanjaan secara *online* di *platform e-commerce* dan 15% dari responden sudah pernah melakukan penjualan secara *online.*

Pengujian *role* Penjual dan *Admin* dilakukan oleh satu *tester* yang sudah memiliki pengalaman dalam menjual barang secara daring di *platform e-commerce* lain sebelumnya. Penguji merupakan salah satu penjual karya seni berupa gambar atau ilustrasi secara daring. Setelah selesai melakukan pengujian, wawancara untuk mendapatkan masukan dilakukan secara daring melalui panggilan telepon *Voice Over IP*.

Tabel 5.2 merupakan hasil wawancara yang dilakukan untuk pengujian *role* Penjual.

Tabel 5.2 Hasil Wawancara Penjual

No.	Pertanyaan	Jawaban
1	Apakah sistem mudah digunakan untuk menjual?	*Flow* nya mudah digunakan, instruksi-instruksinya jelas dan tidak rumit.

2	Apakah sistem sudah memenuhi kebutuhan penjual untuk melakukan aktivitas penjualan?	W*ebsite* sudah bisa untuk transaksi jual beli sudah memiliki fitur-fitur seperti *website* jual beli lain.
3	Apakah sistem dapat membantu menjual barang seni dan kerajinan tangan?	*Website* nya bisa membantu *banget* kalau untuk penjual-penjual kecil ya. Saat ini *website-website* lain tidak ada tempat khusus untuk penjual kecil seperti kami.
4	Apakah ada kendala saat menggunakan sistem sebagai penjual?	Tidak ada, saat menggunakan *website* nya lancar.
5	Apakah puas dengan fitur-fitur yang ada dalam sistem sebagai penjual?	Dengan fitur *website* nya puas karena sudah bisa melakukan kegiatan jual beli dan juga *website* nya punya verifikasi toko yang bisa *tempatin* produk kita menjadi prioritas untuk toko-toko yang sudah besar dan *trusted.*
6	Apakah sistem nyaman digunakan sebagai penjual?	*Website* nyaman digunakan, fitur-fiturnya mudah dipahami dan kebutuhan untuk penjualan sudah terpenuhi.

7	Apakah fitur-fitur keamanan *platform* sudah cukup untuk mendukung aktivitas penjualan?	Sejauh ini *website* nya sudah aman dengan fitur *login* yang mengharuskan verifikasi *email* kalau menggunakan jaringan lain, sangat mempersulit *hacker.*

Tabel 5.3 merupakan hasil wawancara yang dilakukan untuk pengujian *role admin*.

Tabel 5.3 Hasil Wawancara *Admin*

No	Pertanyaan	Jawaban
1	Apakah fitur dari sistem sudah sesuai dengan kebutuhan pengelolaan *e-commerce*?	Fitur *website* nya sudah cukup lumayan untuk kegiatan administratif, terutama kita sebagai pengurus bisa melihat statistik transaksi itu sangat bagus dan juga fitur pemblokiran orang itu sangat berguna.
2	Apakah sistem mudah dipahami?	*Website* nya bersih dan kalau dipahami sangat mudah karena sangat jelas penggunaannya
3	Apakah fitur keamanan sistem sudah cukup untuk keperluan moderasi *platform*?	Kalau keamanan *website* sepertinya sudah cukup aman, ada beberapa fitur keamanan yang sangat bagus *kayak* harus meng-klik *link* kalau coba *login* dari *wifi* lain bisa memblokir

		hacker dan pengurus *website* bisa memblokir orang lain.
4	Apakah ada kesalahan seperti *bug* saat menggunakan sistem sebagai *admin*?	*Web*site nya lancar digunakan. Kalau *error-error gitu* tidak ada saat saya gunakan.
5	Apakah puas dengan fitur-fitur yang ada dalam sistem sebagai *admin*?	Fitur untuk pengurus *website* sangat memuaskan, dan juga tidak susah untuk digunakan.
6	Apakah fitur administrasi atau moderasi *platform* nyaman digunakan?	Fitur-fitur yang ada sejauh ini cukup nyaman digunakan.

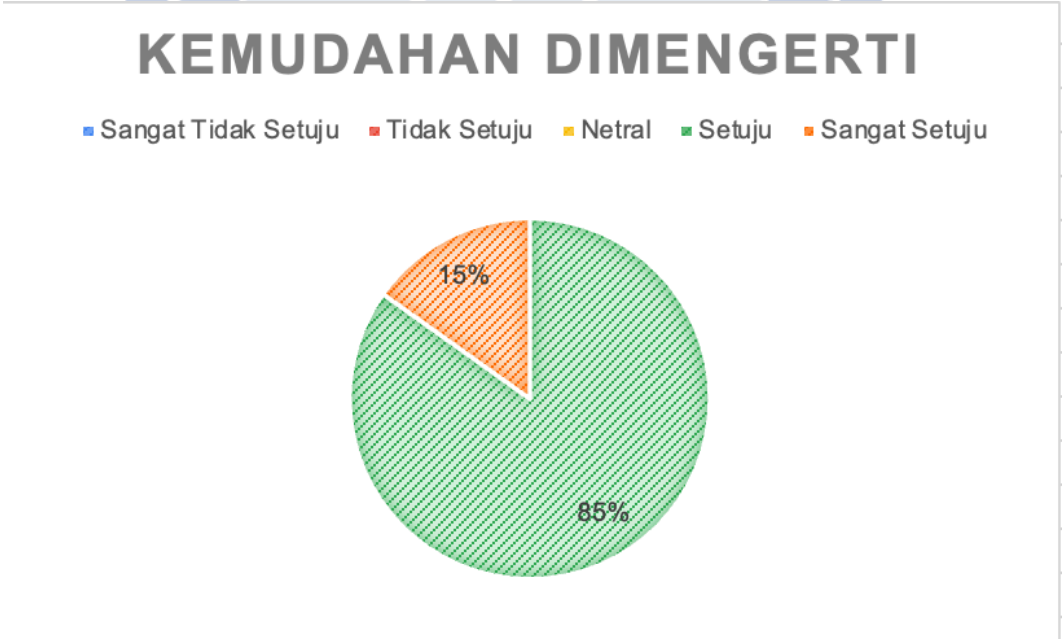
Tabel 5.4 merupakan hasil kuesioner pengetesan sistem menggunakan *role* pengguna atau pembeli.

Tabel 5.4 Hasil Kuesioner Pengguna/Pembeli

No	Pertanyaan	STS	TS	N	S	SS
1	Tampilan dan fitur-fitur sistem mudah dimengerti	0	0	0	28	5
2	Sistem nyaman digunakan	0	0	0	2	31
3	Sistem memiliki fitur yang lengkap untuk sebuah *platform e-commerce* untuk penjualan barang seni dan kerajinan tangan	0	0	0	8	25

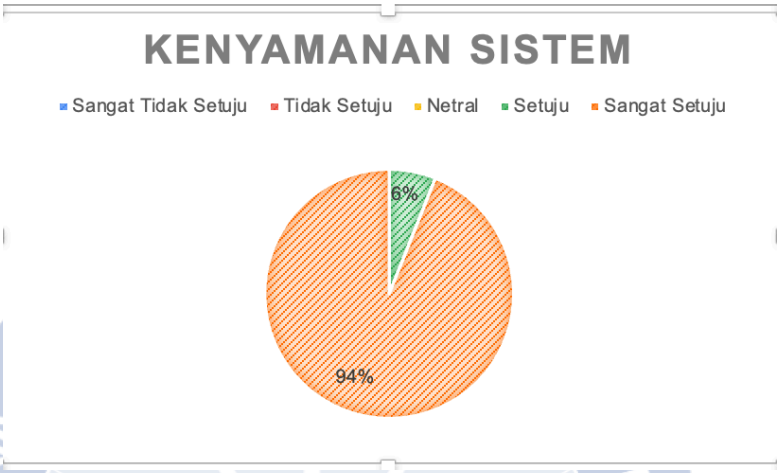
4	Sistem berjalan dengan lancar tanpa kesalahan atau *bug* saat *testing*	0	0	0	8	25
5	Sistem memiliki performa yang cepat	0	0	0	5	28
6	Sistem memiliki fitur keamanan yang cukup untuk *platform e-commerce*	0	0	0	5	28

Gambar 5.89 adalah grafik berupa *pie chart* yang menunjukkan jawaban dari pertanyaan "Tampilan dan fitur-fitur sistem mudah dimengerti". Dari 33 orang, 85% dari *tester* menjawab Setuju, dan 15% menjawab Sangat Setuju. Hal ini menunjukan bahwa sistem mudah dimengerti menurut responden. Tingkat kerumitan penggunaan *platform* merupakan salah satu faktor yang dapat mempengaruhi kenyamanan pengguna.



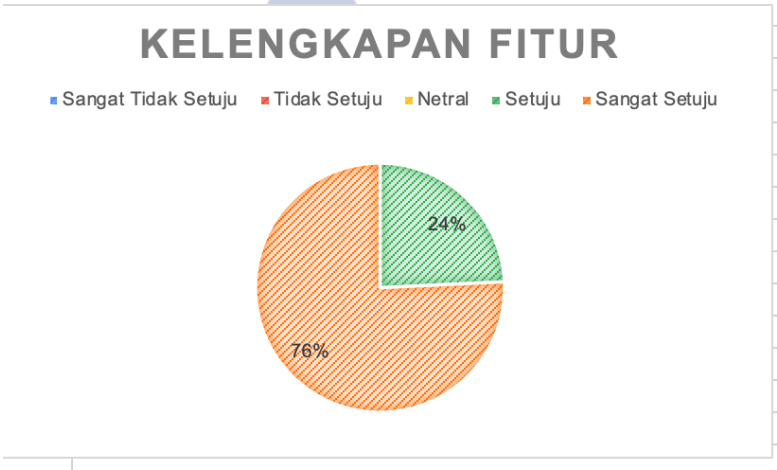
Gambar 5.89 Grafik Jawaban Pertanyaan Pertama

Gambar 5.90 adalah grafik berupa *pie chart* yang menunjukkan jawaban dari pertanyaan "Sistem nyaman digunakan". Dari 33 orang, 94% dari *tester* menjawab Sangat Setuju, dan 6% menjawab Sangat Setuju. Hal ini menunjukan bahwa sistem nyaman digunakan menurut responden sehingga salah satu tujuan dari penelitian sudah terpenuhi.



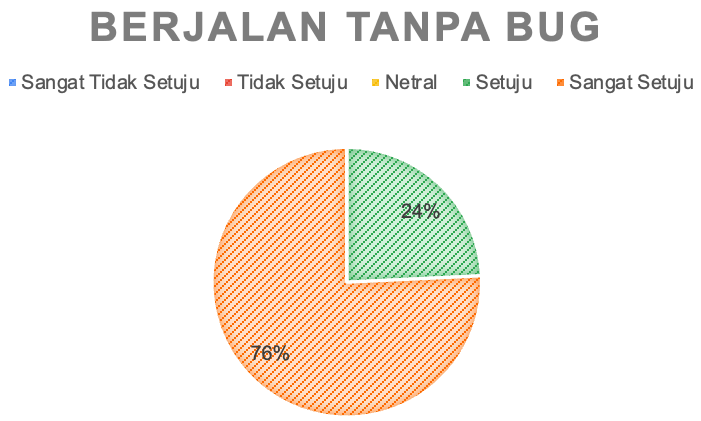
Gambar 5.90 Grafik Jawaban Pertanyaan Kedua

Gambar 5.91 adalah grafik berupa *pie chart* yang menunjukkan jawaban dari pertanyaan "Sistem memiliki fitur yang lengkap untuk sebuah *platform e-commerce* untuk penjualan barang seni dan kerajinan tangan". Dari 33 orang, 76% dari *tester* menjawab Sangat Setuju, dan 24% menjawab Sangat Setuju..



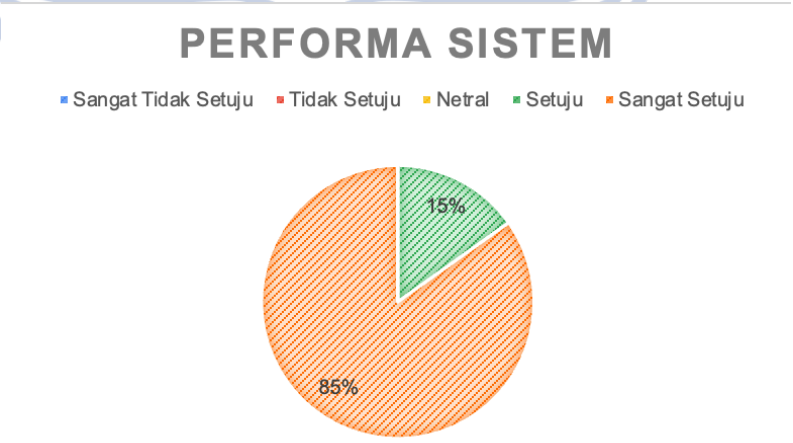
Gambar 5.91 Grafik Jawaban Pertanyaan Ketiga

Gambar 5.92 adalah grafik berupa *pie chart* yang menunjukkan jawaban dari pertanyaan "Sistem berjalan dengan lancar tanpa kesalahan atau *bug* saat *testing"*. Dari 33 orang, 76% dari *tester* menjawab Sangat Setuju, dan 24% menjawab Setuju.



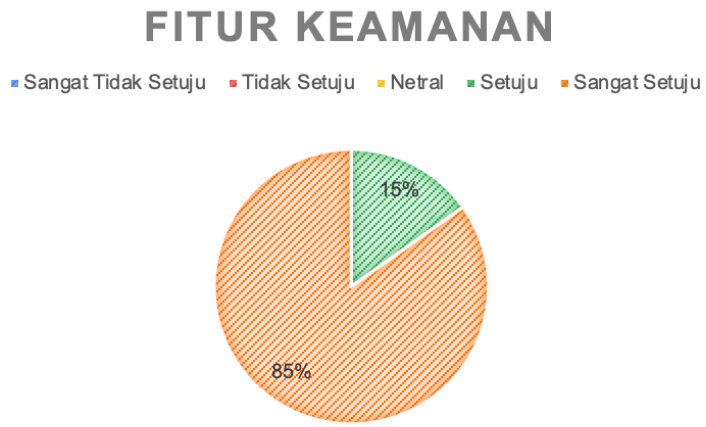
Gambar 5.92 Grafik Jawaban Pertanyaan Keempat

Gambar 5.93 adalah grafik berupa *pie chart* yang menunjukkan jawaban dari pertanyaan "Sistem memiliki performa yang cepat". Dari 33 orang, 85% dari *tester* menjawab Sangat Setuju, dan 15% menjawab Setuju.



Gambar 5.93 Grafik Jawaban Pertanyaan Kelima

Gambar 5.94 adalah grafik berupa *pie chart* yang menunjukkan jawaban pertanyaan "Sistem memiliki fitur keamanan yang cukup untuk *platform e-commerce"*. Dari 33 orang, 85% dari *tester* menjawab Sangat Setuju, dan 15% menjawab Setuju. Dapat disimpulkan bahwa salah satu tujuan dari penelitian sudah terpenuhi.



Gambar 5.94 Grafik Jawaban Pertanyaan Keenam

Setelah dilakukan pengujian, ada beberapa kelebihan dan kekurangan yang dimiliki oleh sistem informasi yang dihasilkan oleh penelitian ini. Kelebihan dari sistem informasi yang dihasilkan adalah:

1. *Admin* dapat melakukan pemblokiran pengguna sehingga mempermudah proses penegakan peraturan situs.
2. Penjual dapat melihat statistik produk dan statistik toko, sehingga mempermudah proses pengumpulan data penjualan.
3. Pengguna merasakan performa sistem yang tinggi sehingga membuat sistem nyaman digunakan.
4. Pengguna merasa aman dengan fitur pemblokiran *login* otomatis, notifikasi *login* yang dikirim ke *email*, dan fitur verifikasi *email* jika *login* dilakukan menggunakan *IP Address* lain.

Namun, sistem juga memiliki beberapa kelemahan yang ditemukan oleh *tester* saat melakukan percobaan sistem:

1. Tidak ada fitur diskon sehingga Penjual dan *Admin* kesulitan untuk melakukan kegiatan promosi.
2. *Admin* belum bisa melakukan pengelolaan produk Penjual sehingga *Admin* tidak dapat langsung menghapus produk yang melanggar aturan, tetapi harus memblokir akun Penjual untuk menutup toko.
3. Tidak ada tampilan berdasarkan kategori sehingga Pengguna kesulitan untuk melihat produk dengan kategori tertentu saja.