3. **LAMPIRAN** SOURCE CODE

1. Login_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/cupertino.dart';
import 'package:abadinursery/services/api_service.dart';
import 'package:abadinursery/models/user_model.dart';
import '../services/session_manager.dart';
import 'package:abadinursery/main.dart';

import 'register_page.dart';
class LoginPage extends StatefulWidget {
  const LoginPage({super.key});

  @override
  LoginPageState createState() => LoginPageState();
}

class LoginPageState extends State<LoginPage> {
  final TextEditingController _usernameController =
TextEditingController();
  final TextEditingController _passwordController =
TextEditingController();
  bool _isLoading = false;
  bool _passwordVisible = false;

  void _login() async {
    String username = _usernameController.text.trim();
    String password = _passwordController.text.trim();

    if (username.isEmpty || password.isEmpty) {
      _showCupertinoDialog("Ups", "Username dan Password harus
diisi.");
      return;
    }

    setState(() {
      _isLoading = true;
    });

    try {
      final response = await ApiService.login(username, password);

      setState(() {
        _isLoading = false;
      });

      if (response != null && response.containsKey('token')) {
        await SessionManager.saveAccessToken(response['token']);
        final user = User.fromJson(response['user']);

        if (!mounted) return;

        Navigator.pushReplacement(
          context,
          MaterialPageRoute(
            builder: (context) => MainScreen(user: user),
          ),
```

```
        );
      } else {
        _showCupertinoDialog(
            "Error", "Gagal login. Silakan cek username dan
password Anda.");
      }
    } catch (e) {
      if (!mounted) return;

      setState(() {
        _isLoading = false;
      });
      _showCupertinoDialog(
          "Error", "Terjadi kesalahan saat login. Silakan coba
lagi.");
    }
  }

  void _showCupertinoDialog(String title, String message) {
    if (!mounted) return;
    showCupertinoDialog(
      context: context,
      builder: (BuildContext context) {
        return CupertinoAlertDialog(
          title: Text(title),
          content: Text(message),
          actions: [
            CupertinoDialogAction(
              isDefaultAction: true,
              child: const Text('OK'),
              onPressed: () {
                Navigator.of(context).pop();
              },
            ),
          ],
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        backgroundColor: Colors.white,
        body: Padding(
          padding: const EdgeInsets.symmetric(vertical: 10,
horizontal: 20),
          child: SingleChildScrollView(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                const SizedBox(
                  height: 30,
                ),
                const Text(
                  "Masuk",
                  style: TextStyle(fontWeight: FontWeight.bold,
fontSize: 25),
```

```
                ),
                Image.asset(
                  "assets/images/login.jpg",
                  height: 250,
                  width: double.infinity,
                ),
                const Text(
                  "Masukkan Username dan Kata Sandi Anda untuk
Masuk",
                  style: TextStyle(fontSize: 12),
                ),
                const SizedBox(
                  height: 20,
                ),
                const Text(
                  "Username",
                  style: TextStyle(fontSize: 12),
                ),
                Container(
                  decoration: BoxDecoration(
                      border: Border.all(
                        color: Colors.black12,
                      ),
                      color: Colors.grey[100],
                      borderRadius:
                          const
BorderRadius.all(Radius.circular(10))),
                  child: TextField(
                    controller: _usernameController,
                    decoration: const InputDecoration(
                        border: InputBorder.none,
                        hintText: 'Masukkan Username',
                        contentPadding: EdgeInsets.all(10)),
                  ),
                ),
                const SizedBox(
                  height: 20,
                ),
                const Text(
                  "Kata Sandi",
                  style: TextStyle(fontSize: 12),
                ),
                Container(
                  decoration: BoxDecoration(
                      border: Border.all(
                        color: Colors.black12,
                      ),
                      color: Colors.grey[100],
                      borderRadius:
                          const
BorderRadius.all(Radius.circular(10))),
                  child: TextField(
                    controller: _passwordController,
                    obscureText: !_passwordVisible,
                    decoration: InputDecoration(
                      border: InputBorder.none,
                      hintText: 'Masukkan Password',
                      contentPadding: const EdgeInsets.all(10),
                      suffixIcon: IconButton(
```

```
                        icon: Icon(
                          _passwordVisible
                              ? Icons.visibility
                              : Icons.visibility_off,
                        ),
                        onPressed: () {
                          setState(() {
                            _passwordVisible = !_passwordVisible;
                          });
                        },
                      ),
                    ),
                  ),
                ),
                const SizedBox(
                  height: 20,
                ),
                MaterialButton(
                  color: Theme.of(context).primaryColor,
                  height: 25,
                  minWidth: double.maxFinite,
                  padding: const EdgeInsets.symmetric(vertical:
10),
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(5.0),
                      side: BorderSide(color:
Theme.of(context).primaryColor)),
                  onPressed: _isLoading ? null : _login,
                  child: const Text(
                    "Masuk",
                    style: TextStyle(color: Colors.white,
fontSize: 20),
                  ),
                ),
                const SizedBox(
                  height: 8,
                ),
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    const Text(
                      "Belum punya akun? ",
                      style: TextStyle(
                          fontWeight: FontWeight.normal,
                          fontSize: 14,
                          color: Colors.grey),
                    ),
                    InkWell(
                      onTap: () {
                        Navigator.push(
                            context,
                            MaterialPageRoute(
                                builder: (_) => const
RegisterPage()));
                      },
                      child: Text(
                        "Daftar ",
                        style: TextStyle(
                            fontWeight: FontWeight.bold,
```

```
                            fontSize: 14,
                            color:
Theme.of(context).primaryColor),
                        ),
                      ),
                    ],
                  )
                ],
              ),
            ),
          ),
        ),
      );
  }
}
```

2. Auth.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/boots
trap.min.css">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style2.css') }}">
    <title>Otentikasi Abadi Green Nursery</title>
</head>
<body>
    <div class="container" id="container">
        <div class="form-container sign-in-container" {% if
form_type != 'login' %}style="display:none;"{% endif %}>
            <form method="POST" action="{{
url_for('auth.login_view') }}">
                <h1>Masuk</h1>
                <span>Masukkan kredensial akun di sini</span>
                <input type="text" class="form-control"
name="username" placeholder="Username" required>
                <input type="password" class="form-control"
name="password" placeholder="Password" required>
                <div class="form-group remember-me">
                    <!-- <input type="checkbox" name="remember"
id="remember">
                    <label for="remember">Ingat Saya</label> -->
                </div>
                <button type="submit"
class="button">Masuk</button>
                <div class="text-center mt-3">
                    <!-- <a href="#" id="signUpLink">Daftar</a> --
>
                </div>
                {% with messages =
get_flashed_messages(with_categories=true) %}
```

```
                    {% if messages %}
                        <div class="mt-3">
                            {% for category, message in messages
%}
                                <div class="alert alert-{{
category }}">{{ message }}</div>
                            {% endfor %}
                        </div>
                    {% endif %}
                {% endwith %}
            </form>
        </div>
        <div class="form-container sign-up-container" {% if
form_type != 'register' %}style="display:none;"{% endif %}>
            <form method="POST" action="{{
url_for('auth.register_view') }}">
                <h1>Daftarkan Akun</h1>
                <span>Masukkan informasi tentang kamu di
sini</span>
                <input type="text" class="form-control"
name="username" placeholder="Username" required>
                <input type="text" class="form-control"
name="nama_lengkap" placeholder="Nama Lengkap" required>
                <input type="password" class="form-control"
name="password" placeholder="Password" required>
                <button type="submit"
class="button">Daftar</button>
                <div class="text-center mt-3">
                    <!-- <a href="#" id="signInLink">Sudah punya
akun? Masuk di sini</a> -->
                </div>
                {% with messages =
get_flashed_messages(with_categories=true) %}
                    {% if messages %}
                        <div class="mt-3">
                            {% for category, message in messages
%}
                                <div class="alert alert-{{
category }}">{{ message }}</div>
                            {% endfor %}
                        </div>
                    {% endif %}
                {% endwith %}
            </form>
        </div>
        <div class="overlay-container">
            <div class="overlay">
                <div class="overlay-panel overlay-left">
                    <h1>Selamat datang kembali!</h1>
                    <p>Agar terus terkoneksi dengan kami, harap
masukkan kredensial akun Anda.</p>
                    <button class="ghost button"
id="signIn">Masuk</button>
                </div>
                <div class="overlay-panel overlay-right">
                    <h1>Hello, Sahabat Tanaman Hias!</h1>
                    <p>Masukkan kredensial akun di sini untuk
terus terhubung dengan kami</p>
```

```
                </div>
            </div>
        </div>
    </div>
    <script>
        const signUpButton = document.getElementById('signUp');
        const signInButton = document.getElementById('signIn');
        const container = document.getElementById('container');
        const signUpLink = document.getElementById('signUpLink');
        const signInLink = document.getElementById('signInLink');

        signUpButton.addEventListener('click', () => {
            container.classList.add('right-panel-active');
            document.querySelector('.sign-in-
container').style.display = 'none';
            document.querySelector('.sign-up-
container').style.display = 'block';
        });

        signInButton.addEventListener('click', () => {
            container.classList.remove('right-panel-active');
            document.querySelector('.sign-up-
container').style.display = 'none';
            document.querySelector('.sign-in-
container').style.display = 'block';
        });

        signUpLink.addEventListener('click', (e) => {
            e.preventDefault();
            container.classList.add('right-panel-active');
            document.querySelector('.sign-in-
container').style.display = 'none';
            document.querySelector('.sign-up-
container').style.display = 'block';
        });

        signInLink.addEventListener('click', (e) => {
            e.preventDefault();
            container.classList.remove('right-panel-active');
            document.querySelector('.sign-up-
container').style.display = 'none';
            document.querySelector('.sign-in-
container').style.display = 'block';
        });
    </script>
</body>
</html>
```

3. Data_preprocessing.py

```
import pandas as pd
from extensions import mysql

def load_csv_data(filepath):
    data = pd.read_csv(filepath)
    data['waktu'] = pd.to_datetime(data['waktu'], format='%Y%m')
    return data
```

```
def load_db_data():
    query = """
    SELECT nama_tanaman, jenis_tanaman, total_quantity AS
jumlah_tersewa,
           CONCAT(tahun, LPAD(bulan, 2, '0')) AS waktu
    FROM aggregated_bookings
    """
    cur = mysql.connection.cursor()
    cur.execute(query)
    result = cur.fetchall()
    cur.close()

    db_data = pd.DataFrame(result, columns=['nama_tanaman',
'jenis_tanaman', 'jumlah_tersewa', 'waktu'])
    db_data['waktu'] = pd.to_datetime(db_data['waktu'],
format='%Y%m')
    return db_data

def combine_data(csv_data, db_data):
    combined_data = pd.concat([csv_data, db_data],
ignore_index=True)
    return combined_data
```

4. Arima_model.py

```
import pandas as pd

from statsmodels.tsa.arima.model import ARIMA

def train_arima(data):
    grouped_data = data.groupby('jenis_tanaman')
    forecasts_arima = {}

    best_params_per_tanaman_arima = {
        'Anggrek': {'p': 4, 'd': 1, 'q': 3},
        'Koleksi Kita': {'p': 4, 'd': 1, 'q': 1},
        'Tanaman Besar': {'p': 1, 'd': 1, 'q': 4}
    }

    for name, group in grouped_data:
        if name not in best_params_per_tanaman_arima:
            continue
        group = group.sort_values(by='waktu')
        group.set_index('waktu', inplace=True)
        train_data = group['jumlah_tersewa']

        params = best_params_per_tanaman_arima[name]
        model_arima = ARIMA(train_data, order=(params['p'],
params['d'], params['q']))
        model_fit_arima = model_arima.fit()
        forecast_arima = model_fit_arima.forecast(steps=12)
        forecasts_arima[name] = forecast_arima

    return forecasts_arima
```

5. TES_model.py

```
import pandas as pd
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

```
def train_tes(data):
    grouped_data = data.groupby('jenis_tanaman')
    forecasts_tes = {}

    best_params_per_tanaman_tes = {
        'Anggrek': {'trend': 'add', 'seasonal': 'add', 'alpha':
0.2, 'beta': 0.2, 'gamma': 0.2},
        'Koleksi Kita': {'trend': 'mul', 'seasonal': 'add',
'alpha': 0.2, 'beta': 0.4, 'gamma': 0.4},
        'Tanaman Besar': {'trend': 'add', 'seasonal': 'add',
'alpha': 0.4, 'beta': 0.6, 'gamma': 0.2}
    }

    for name, group in grouped_data:
        if name not in best_params_per_tanaman_tes:
            continue
        group = group.sort_values(by='waktu')
        group.set_index('waktu', inplace=True)
        train_data = group['jumlah_tersewa']

        params = best_params_per_tanaman_tes[name]
        model_tes = ExponentialSmoothing(train_data,
trend=params['trend'], seasonal=params['seasonal'],
seasonal_periods=12)
        model_fit_tes =
model_tes.fit(smoothing_level=params['alpha'],
smoothing_trend=params['beta'],
smoothing_seasonal=params['gamma'])
        forecast_tes = model_fit_tes.forecast(steps=12)
        forecasts_tes[name] = forecast_tes

    return forecasts_tes
```

6. Prediction_routes.py

```
from flask import Blueprint, jsonify
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from extensions import mysql
from datetime import datetime

prediction_blueprint = Blueprint('prediction', __name__)

@prediction_blueprint.route('/predict')
def predict():
    csv_data = pd.read_csv('data/dataset_fix.csv')
    csv_data['waktu'] = pd.to_datetime(csv_data['waktu'],
format='%Y-%m')

    cur = mysql.connection.cursor()
    cur.execute("SELECT nama_tanaman, jenis_tanaman, quantity,
bulan, tahun FROM aggregated_bookings")
    aggregated_data = cur.fetchall()
    cur.close()

    aggregated_df = pd.DataFrame(aggregated_data,
columns=['nama_tanaman', 'jenis_tanaman', 'total_quantity',
'bulan', 'tahun'])
```

```
    print("Aggregated Data from DB:")
    print(aggregated_df.head())

    aggregated_df['waktu'] =
pd.to_datetime(aggregated_df['tahun'].astype(str) + '-' +
aggregated_df['bulan'].astype(str) + '-01')

    csv_data = csv_data.rename(columns={'jumlah_tersewa':
'total_quantity'})
    combined_data = pd.concat([csv_data, aggregated_df[['waktu',
'jenis_tanaman', 'nama_tanaman', 'total_quantity']]],
ignore_index=True)
    print("Combined Data:")
    print(combined_data.head())

    grouped_data = combined_data.groupby('jenis_tanaman')

    forecasts_arima = {}
    forecasts_tes = {}

    arima_params = {
        'Anggrek': {'order': (4, 1, 3)},
        'Koleksi Kita': {'order': (4, 1, 1)},
        'Tanaman Besar': {'order': (1, 1, 4)},
    }
    tes_params = {
        'Anggrek': {'trend': 'add', 'seasonal': 'add',
'seasonal_periods': 12, 'alpha': 0.2, 'beta': 0.2, 'gamma': 0.2},
        'Koleksi Kita': {'trend': 'mul', 'seasonal': 'add',
'seasonal_periods': 12, 'alpha': 0.2, 'beta': 0.4, 'gamma': 0.4},
        'Tanaman Besar': {'trend': 'add', 'seasonal': 'add',
'seasonal_periods': 12, 'alpha': 0.4, 'beta': 0.6, 'gamma': 0.2},
    }

    default_arima_params = {'order': (0, 1, 1)}
    default_tes_params = {'trend': 'add', 'seasonal': 'add',
'seasonal_periods': 12, 'alpha': 0.3, 'beta': 0.3, 'gamma': 0.3}

    for name, group in grouped_data:
        print(f"Processing group: {name}")
        group = group.sort_values(by='waktu')
        group = group.drop_duplicates(subset='waktu',
keep='first')
        group.set_index('waktu', inplace=True)
        group = group.asfreq('MS')

        if len(group) < 12:
            print(f"Skipping {name} due to insufficient data")
            continue

        train_data = group['total_quantity'].iloc[:-12]
        test_data = group['total_quantity'].iloc[-12:]

        if len(train_data) == 0 or len(test_data) == 0:
            print(f"Skipping {name} due to empty train or test
data")
            continue

        try:
```

```
            arima_param = arima_params.get(name,
default_arima_params)
            model_arima = ARIMA(train_data,
order=arima_param['order'])
            model_fit_arima = model_arima.fit()
            forecast_arima = model_fit_arima.forecast(steps=12)
            forecast_arima.index =
pd.date_range(start=train_data.index[-1], periods=12, freq='MS')
            forecasts_arima[name] = forecast_arima.tolist()
        except Exception as e:
            print(f"Error fitting ARIMA for {name}: {e}")
            continue

        try:
            tes_param = tes_params.get(name, default_tes_params)
            model_tes = ExponentialSmoothing(train_data, **{k: v
for k, v in tes_param.items() if k not in ['alpha', 'beta',
'gamma']})
            model_fit_tes =
model_tes.fit(smoothing_level=tes_param['alpha'],
smoothing_trend=tes_param['beta'],
smoothing_seasonal=tes_param['gamma'])
            forecast_tes = model_fit_tes.forecast(steps=12)
            forecast_tes.index =
pd.date_range(start=train_data.index[-1], periods=12, freq='MS')
            forecasts_tes[name] = forecast_tes.tolist()
        except Exception as e:
            print(f"Error fitting TES for {name}: {e}")
            continue

    return jsonify({'arima': forecasts_arima, 'tes':
forecasts_tes})
```