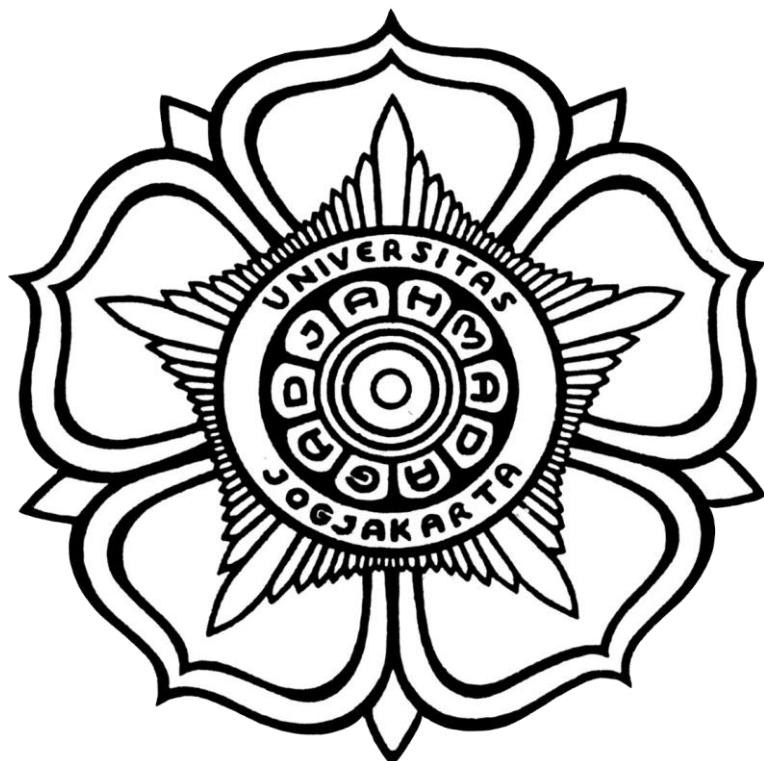


LAPORAN PROJECT AKHIR

PRAKTIKUM TIK



Disusun oleh :

- | | |
|---------------------------------|----------------------|
| 1. Muhammad Faris Alkautsar | (19/441202/SV/16554) |
| 2. Tania Desnatasari | (19/441212/SV/16564) |
| 3. Bramantyo Wahyu Kuncoro | (19/447113/SV/16832) |
| 4. Muhammad Risang Danarsantika | (19/447119/SV/16838) |
| 5. Zaenal Muttaqin | (19/447129/SV/16848) |

ARM 1

**DEPARTEMEN TEKNIK MESIN SEKOLAH VOKASI
UNIVERSITAS GADJAH MADA
YOGYAKARTA**

2021

PROJECT 6

SOAL

Soal 6

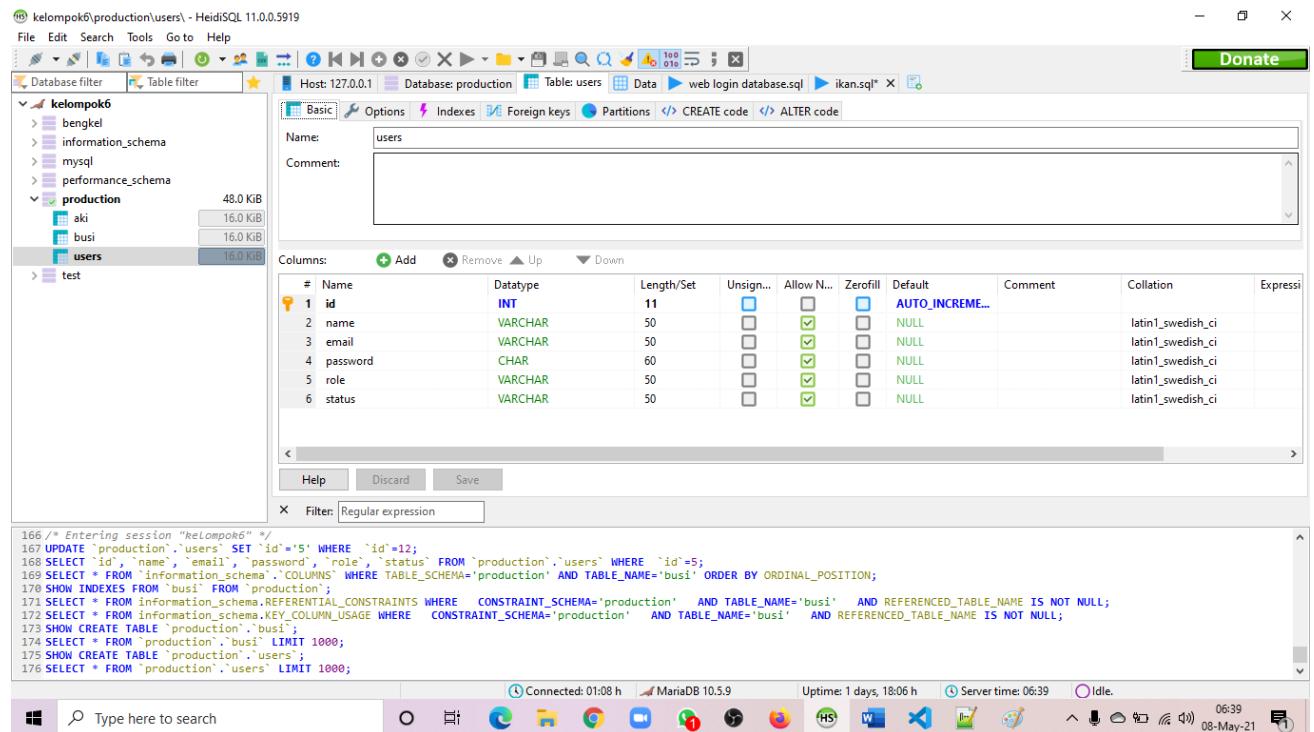
Dashboard Production Control System

- input product from plant to database (automatic by system)
- proses bisnis Production Control : monitoring production whether in performance or under perfomance based on goal target production
- tabel hrs ada kolom datetime/timestamp (utk pencatatan waktu/tanggal inputan data)
- report production vs goal target + chart

TAHAPAN PROSES

1. Membuat database menggunakan MySQL yang berisikan tabel users, busi, dan aki.

Table users



The screenshot shows the HeidiSQL interface for MySQL version 11.0.0.5919. The left sidebar shows the database structure with 'kelompok6' as the current schema. Inside 'kelompok6', there are several databases: bengkel, information_schema, mysql, performance_schema, and production. Within the production database, there are three tables: aki, busi, and users. The 'users' table is currently selected. The main pane displays the table's structure:

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default	Comment	Collation	Expressi
1	id	INT	11	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT		latin1_swedish_ci	
2	name	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	
3	email	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	
4	password	CHAR	60	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	
5	role	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	
6	status	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	

Below the table structure, there are buttons for Help, Discard, and Save. The bottom of the window shows the MySQL command history:

```
166 /* Entering session "kelompok6" */
167 UPDATE `production`.`users` SET `id`='5' WHERE `id`='12';
168 SELECT `id`, `name`, `email`, `password`, `role`, `status` FROM `production`.`users` WHERE `id`=5;
169 SELECT * FROM `information_schema`.`COLUMNS` WHERE TABLE_SCHEMA='production' AND TABLE_NAME='busi' ORDER BY ORDINAL_POSITION;
170 SHOW INDEXES FROM `busi` FROM `production`;
171 SELECT * FROM `information_schema.REFERENTIAL_CONSTRAINTS` WHERE CONSTRAINT_SCHEMA='production' AND TABLE_NAME='busi' AND REFERENCED_TABLE_NAME IS NOT NULL;
172 SELECT * FROM `information_schema.KEY_COLUMN_USAGE` WHERE CONSTRAINT_SCHEMA='production' AND TABLE_NAME='busi' AND REFERENCED_TABLE_NAME IS NOT NULL;
173 SHOW CREATE TABLE `production`.`busi`;
174 SELECT * FROM `production`.`busi` LIMIT 1000;
175 SHOW CREATE TABLE `production`.`users`;
176 SELECT * FROM `production`.`users` LIMIT 1000;
```

The taskbar at the bottom of the screen shows various application icons, and the system tray indicates the date and time as 06:39 08-May-21.

Table Busi

The screenshot shows the HeidiSQL interface for the 'kelompok6\production\busi' database. The left sidebar lists databases: kelompok6, bengkel, information_schema, mysql, performance_schema, production (selected), aki, busi (selected), users, and test. The right panel displays the 'Basic' tab for the 'busi' table. The table has 5 columns: id (INT, primary key, auto-increment, not null), tanggal (VARCHAR 255), target (VARCHAR 255), aktual (VARCHAR 255), and status (VARCHAR 255). The SQL pane at the bottom contains various MySQL queries related to the 'busi' table.

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default	Comment	Collation	Expressi
1	id	INT	11	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT		latin1_swedish_ci	
2	tanggal	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	
3	target	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	
4	aktual	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	
5	status	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	

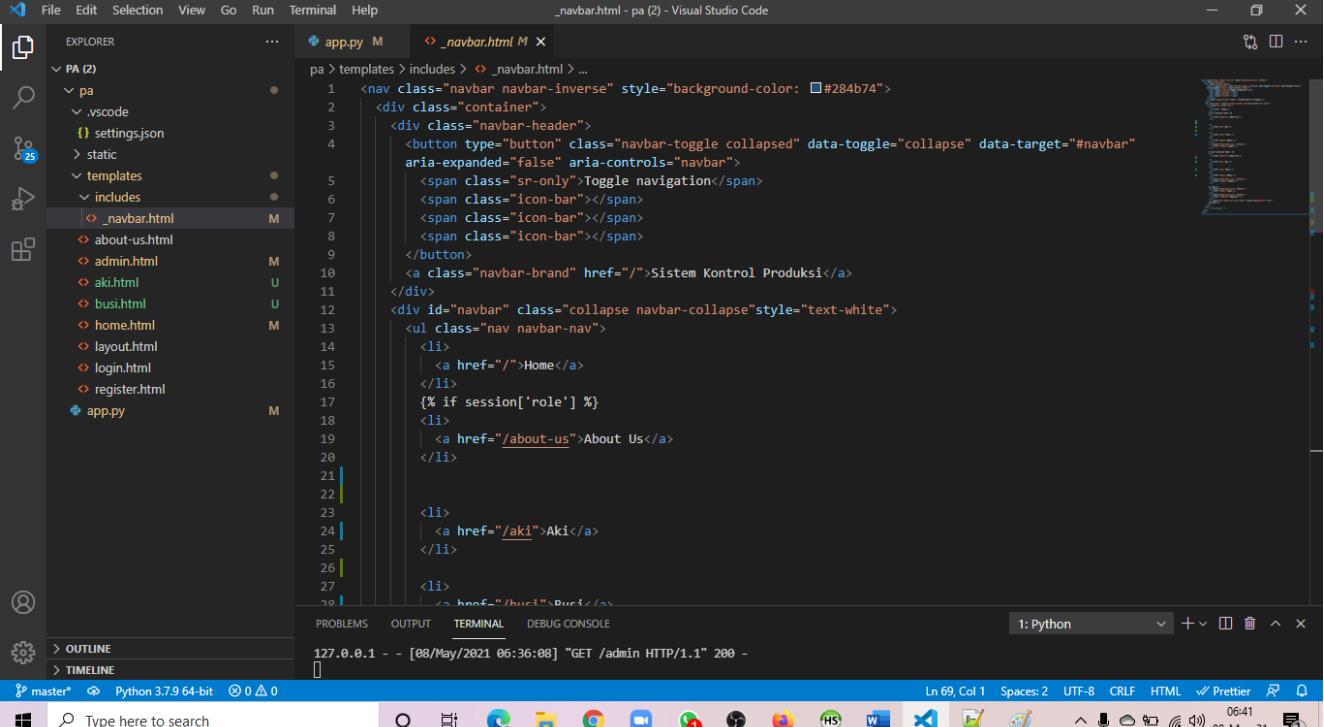
Table Aki

The screenshot shows the HeidiSQL interface for the 'kelompok6\production\aki' database. The left sidebar lists databases: kelompok6, bengkel, information_schema, mysql, performance_schema, production (selected), aki (selected), busi, users, and test. The right panel displays the 'Basic' tab for the 'aki' table. The table has 5 columns: id (INT, primary key, auto-increment, not null), tanggal (VARCHAR 255), target (VARCHAR 255), aktual (VARCHAR 255), and status (VARCHAR 255). The SQL pane at the bottom contains various MySQL queries related to the 'aki' table.

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default	Comment	Collation	Expressi
1	id	INT	11	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT		latin1_swedish_ci	
2	tanggal	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	
3	target	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	
4	aktual	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	
5	status	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		latin1_swedish_ci	

2. Selanjutnya membuat front end, back end, dan simulator web html melalui Visual Studio Code.

_navbar.html

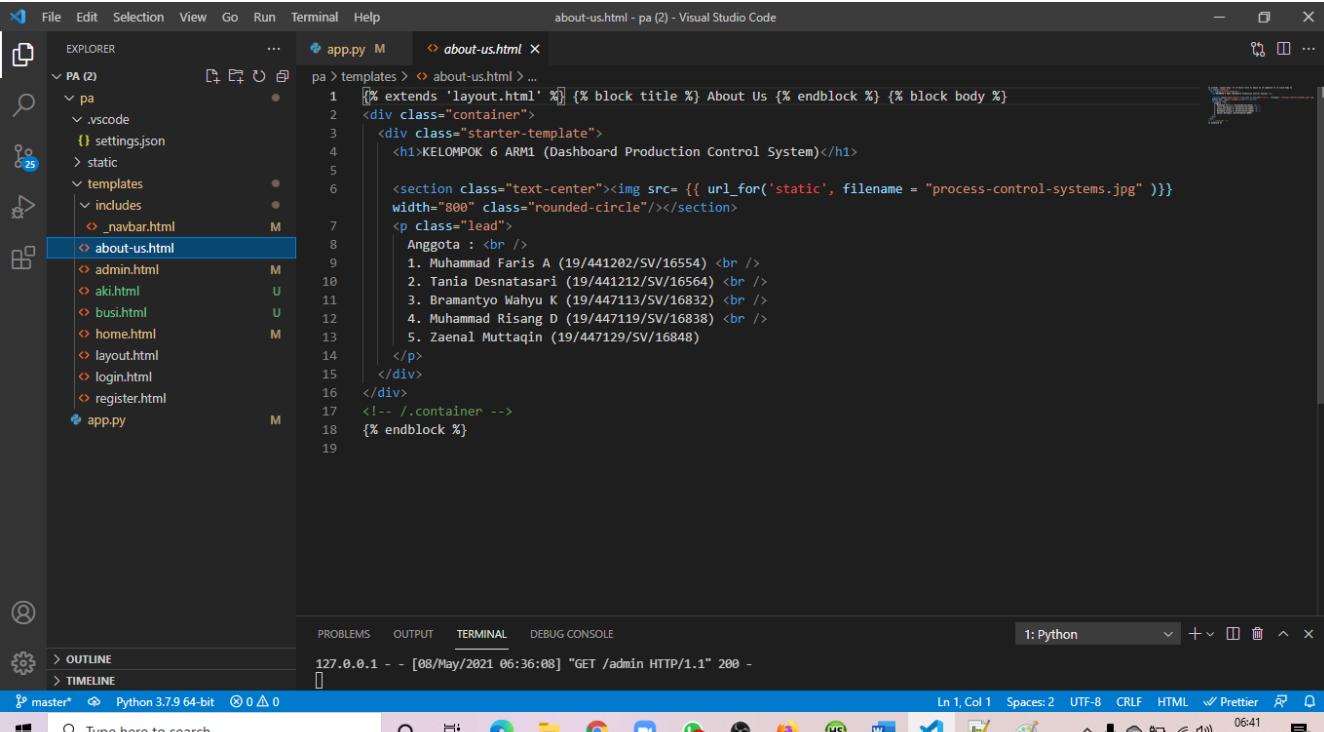


The screenshot shows the Visual Studio Code interface with the file `_navbar.html` open in the editor. The code defines a navigation bar with a brand link, user session logic, and links to other pages like `/about-us`, `/aki`, and `/busi`. The terminal at the bottom shows a successful HTTP request to `127.0.0.1`.

```
<nav class="navbar navbar-inverse" style="background-color: #284b74">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="/">Sistem Kontrol Produksi</a>
    </div>
    <div id="navbar" class="collapse navbar-collapse" style="text-align: center">
      <ul class="nav navbar-nav">
        <li>
          <a href="/">Home</a>
        </li>
        {% if session['role'] %}
        <li>
          <a href="/about-us">About Us</a>
        </li>
        <li>
          <a href="/aki">Aki</a>
        </li>
        <li>
          <a href="/busi">Busi</a>
        </li>
      </ul>
    </div>
  </div>

```

about-us.html



The screenshot shows the Visual Studio Code interface with the file `about-us.html` open in the editor. It extends the `layout.html` template and displays a list of five team members with their names and student IDs. The terminal at the bottom shows a successful HTTP request to `127.0.0.1`.

```
{% extends 'layout.html' %} {% block title %} About Us {% endblock %} {% block body %}
  <div class="container">
    <div class="starter-template">
      <h1>KELOMPOK 6 ARM1 (Dashboard Production Control System)</h1>
      <section class="text-center"><img src= "{{ url_for('static', filename = "process-control-systems.jpg") }}" width="800" class="rounded-circle"/></section>
      <p class="lead">
        Anggota : <br />
        1. Muhammad Faris A (19/441202/SV/16554) <br />
        2. Tania Desnatasari (19/441212/SV/16564) <br />
        3. Bramantyo Wahyu K (19/447113/SV/16832) <br />
        4. Muhammad Risang D (19/447119/SV/16838) <br />
        5. Zaenal Muttaqin (19/447129/SV/16848)
      </p>
    </div>
  </div>

```

admin.html

The screenshot shows the Visual Studio Code interface with the admin.html file open in the editor. The code is a template for displaying user data:

```
pa > templates > admin.html > div.container > div.starter-template > table.table-striped > thead > tr
1  {% extends 'layout.html' %} {% block title %} Admin {% endblock %} {% block body %}
2
3  <div class="container">
4    <div class="starter-template">
5      {% if session['name'] %}
6        <h1>Data Users</h1>
7
8        <table class="table table-striped">
9          <thead>
10         <tr>
11           <td>Id</td>
12           <td>Name</td>
13           <td>Email</td>
14           <td>Password</td>
15           <td>Role</td>
16           <td>Status</td>
17           <td>Aksi</td>
18         </tr>
19       </thead>
20
21       <tbody>
22         {% for row in users %}
23
24           <tr>
25             <td>{{row.id}}</td>
26             <td>{{ row.name }}</td>
27             <td>{{ row.password }}</td>
28             <td>{{ row.email }}</td>
29             <td>{{ row.role }}</td>
30           </tr>
31
32       </tbody>
33     </table>
34   </div>
35 </div>
```

The terminal at the bottom shows a successful HTTP request:

```
127.0.0.1 - - [08/May/2021 06:36:08] "GET /admin HTTP/1.1" 200 -
```

busi.html

The screenshot shows the Visual Studio Code interface with the busi.html file open in the editor. The code handles date range input fields:

```
pa > templates > busi.html > ...
14  <input type='text' class="form-control" name="startDate" />
15  <span class="input-group-addon">
16    | <span class="glyphicon glyphicon-calendar"></span>
17  </span>
18 </div>
19 <div class="starter-template col-md-6 mb-4" style="padding-top: 0rem;">
20
21   <p>Input Tanggal Akhir :</p>
22   <div class="input-group date" id='datepicker2'>
23     <input type='text' class="form-control" name="endDate" />
24     <span class="input-group-addon">
25       | <span class="glyphicon glyphicon-calendar"></span>
26     </span>
27   </div>
28 </div>
29 <div class="row">
30   <div class="col-md-12 mb-4">
31     <div style="padding-top: 2rem; padding-bottom: 2rem;">
32       <button type="submit" class="btn btn-success btn-lg">
33         Lihat Data
34       </button>
35     </div>
36   </div>
37 </div>
38 <div class="row">
39   <div id="columnchart_material" style="width: 100%; height: 500px;"></div>
40 </div>
```

The terminal at the bottom shows a successful HTTP request:

```
127.0.0.1 - - [08/May/2021 06:36:08] "GET /admin HTTP/1.1" 200 -
```

aki.html

The screenshot shows the Visual Studio Code interface with the file 'aki.html' open in the editor. The code is an HTML form for product data entry, featuring two date input fields ('start Date' and 'end Date') with accompanying Bootstrap datepickers.

```
<div class="starter-template col-md-6 mb-4">
<h1>Data Produk Aki</h1>
<div class="row">
<div class="starter-template col-md-6 mb-4" style="padding-top: 0rem;">
<p>Input Tanggal Awal :</p>
<div class="input-group date" id='datetimepicker1'>
<input type="text" class="form-control" name="startDate" />
<span class="input-group-addon">
<span class="glyphicon glyphicon-calendar"></span>
</span>
</div>
</div>
<div class="starter-template col-md-6 mb-4" style="padding-top: 0rem;">
<p>Input Tanggal Akhir :</p>
<div class="input-group date" id='datetimepicker2'>
<input type="text" class="form-control" name="endDate" />
<span class="input-group-addon">
<span class="glyphicon glyphicon-calendar"></span>
</span>
</div>
</div>
<div class="row">
<div class="col-md-12 mb-4">
<div style="padding-top: 2rem; padding-bottom: 2rem;">
<button type="submit" class="btn btn-success btn-lg">
```

home.html

The screenshot shows the Visual Studio Code interface with the file 'home.html' open in the editor. The code is an HTML page for the main landing page, featuring a jumbotron with a message and three columns of text.

```
<% extends 'layout.html' %> {%
    block title %} Home {%
    endblock %} {%
    block body %}
    <!-- Main jumbotron for a primary marketing message or call to action -->
    <div class="jumbotron" style="background-color: #rgb(138, 158, 224)">
        <div class="container">
            <div class="row">
                <div class="col-md-6">{%
                    if session['name'] %} Login Berhasil! Selamat Datang Kakak {{ session['name'] }} {%
                    } {%
                    else %} Silahkan Login atau Register Terlebih Dahulu {%
                        endif %}</div>
            </div>
            <h1 class="text-center">Hello, Selamat Datang di Sistem Kontrol Produksi Kelompok 6 ARM1</h1>
            <section class="text-center"><img src= "{{ url_for('static', filename = "bc.jpg" )}} width="500" class="rounded-circle"/></section>
        </div>
    </div>
    <div class="container" style="background-color: #rgb(198, 199, 199)">
        <!-- Example row of columns -->
        <div class="row">
            <div class="col-md-4">
                <h2>Amanah</h2>
                <p>Kami menjaga dan melindungi apa yang sudah diamanahkan pelanggan pada kami serta melaksanakan tugas dengan sebaik-baiknya.</p>
            </div>
            <div class="col-md-4">
                <h2>Jujur</h2>
                <p>Kami mengutamakan selalu mengutamakan kesesuaian sikap antara perkataan yang diucapkan dan perbuatan yang dilakukan.</p>
            </div>
            <div class="col-md-4">
                <h2>Berkualitas</h2>
                <p>Kami selalu menyediakan produk yang berkualitas tinggi dan memperbaiki kualitas secara terus-menerus untuk
```

layout.html

The screenshot shows the Visual Studio Code interface with the layout.html file open in the editor. The file contains HTML and JavaScript code for a layout template. It includes meta tags, a title, and links to Bootstrap and jQuery libraries. A script block at the bottom initializes four datepickers on the page.

```
<!DOCTYPE html>
</html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>{% block title %} {% endblock %} {% block body %} {% endblock %}</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" type="text/css" media="screen" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datetimepicker/4.17.47/css/bootstrap-datetimepicker.css" />
</head>
<body style="padding-bottom: 5rem;">
    {% include 'includes/_navbar.html' %} {% block body %} {% endblock %}
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="https://code.jquery.com/jquery-3.6.0.slim.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.24.0/moment.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datetimepicker/4.17.47/js/bootstrap-datetimepicker.min.js"></script>
    <div id="columnchart_material" style="width: 800px; height: 500px; padding-left: 18.5rem;"></div>
    <script type="text/javascript">
        $(function() {
            $('#datetimepicker1').datetimepicker();
            $('#datetimepicker2').datetimepicker();
            $('#datetimepicker3').datetimepicker();
            $('#datetimepicker4').datetimepicker();
        });
    </script>
</body>
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

127.0.0.1 - - [08/May/2021 06:36:08] "GET /admin HTTP/1.1" 200 -

master* Python 3.7.9 64-bit @ 0 △ 0

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF HTML ✘ Prettier ⌂ 06:44 08-May-21

login.html

The screenshot shows the Visual Studio Code interface with the login.html file open in the editor. The file extends the layout.html base and contains HTML for a login form. It includes a container div, a messages block, and a form with email and password fields.

```
{% extends 'layout.html' %} {% block title %} Login {% endblock %} {% block body %}<div class="container">
    {% with messages = get_flashed_messages() %}
        {% if messages %}
            {% for message in messages %}
                {{ message }}
            {% endfor %}
        {% endif %}
    {% endwith %}
<div class="row">
    <div class="col-md-6 col-md-offset-3">
        <form action="/login" method="POST">
            <div class="form-group">
                <label>Email:</label>
                <input type="email" class="form-control" name="email" />
            </div>
            <div class="form-group">
                <label>Password:</label>
                <input type="password" class="form-control" name="password" />
                <input type="hidden" name="status" value="aktif" />
            </div>
        </form>
    </div>
</div>
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

127.0.0.1 - - [08/May/2021 06:36:08] "GET /admin HTTP/1.1" 200 -

master* Python 3.7.9 64-bit @ 0 △ 0

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF HTML ✘ Prettier ⌂ 06:44 08-May-21

register.html

The screenshot shows the Visual Studio Code interface with the register.html file open in the editor. The file contains HTML code for a registration form. The code includes a container row, several form groups for name, email, and password, and a dropdown for job selection. The file is part of a project named 'pa'.

```
<% extends 'layout.html' %> {# block title #} Register {# endblock #} {# block body #}</div>
<div class="row">
<div class="col-md-6 col-md-offset-3">
<form action="/register" method="POST">
<div class="form-group">
<label>Name:</label>
<input type="text" class="form-control" name="name" />
</div>

<div class="form-group">
<label>Email:</label>
<input type="email" class="form-control" name="email" />
</div>

<div class="form-group">
<label>Password:</label>
<input type="password" class="form-control" name="password" />
</div>

<div class="mb-3">
<label for="Select" class="form-label" style="padding-bottom: 2rem">Jabatan :</label>
<select id="Select" class="form-select text-left" name="select">
<option value="Admin">Admin</option>
<option value="Manager">Manager</option>
</select>
</div>
```

app.py

The screenshot shows the Visual Studio Code interface with the app.py file open in the editor. The file contains Python code for a Flask application. It imports random, Flask, MySQLdb, bcrypt, json, and datetime. It configures the MySQL database and creates an Flask app. It defines routes for the home page and a login endpoint. The login endpoint handles POST requests by extracting email, status, and password from the form, encoding the password, and executing a MySQL query to select users by email. The file is part of a project named 'pa'.

```
import random
from flask import Flask, render_template, request, redirect, url_for, session, flash
from flask_mysqldb import MySQL, MySQLdb
import bcrypt
import json
import datetime

app = Flask(__name__)
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'risss'
app.config['MYSQL_DB'] = 'production'
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'
mysql = MySQL(app)

@app.route('/')
def home():
    return render_template("home.html")

@app.route('/login', methods=['GET', "POST"])
def login():
    if request.method == 'POST':
        email = request.form['email']
        status = request.form['status']
        password = request.form['password'].encode('utf-8')

        curl = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        curl.execute("SELECT * FROM users WHERE email=%s", (email,))
```

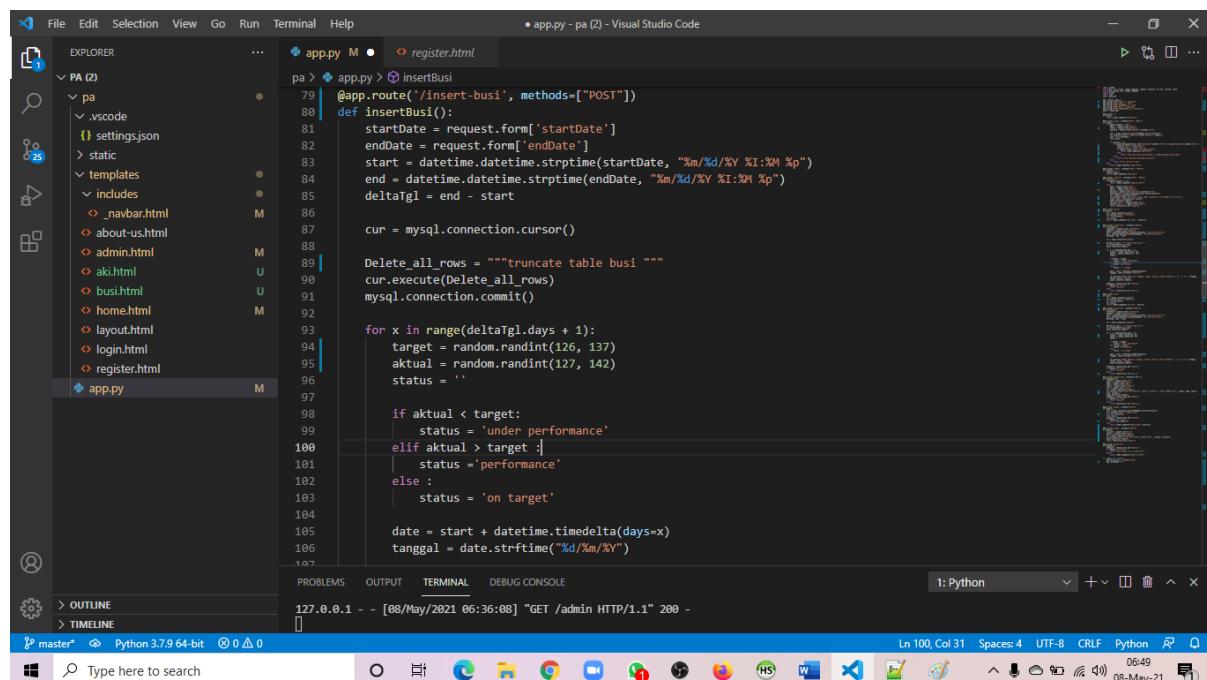
✚ Membuat kodingan simulator

```
for x in range(deltaTgl.days + 1):
    target = random.randint(126, 137)
    aktual = random.randint(127, 142)
    status = ''

    if aktual < target:
        status = 'under performance'
    elif aktual > target :
        status ='performance'
    else :
        status = 'on target'

    date = start + datetime.timedelta(days=x)
    tanggal = date.strftime("%d/%m/%Y")
```

✚ Menggabungkan kodingan simulator dengan back end



The screenshot shows the Visual Studio Code interface with the file `app.py` open. The code integrates the simulator logic from the previous snippet into the existing backend logic. The code is as follows:

```
@app.route('/insert-busi', methods=["POST"])
def insertBusi():
    startDate = request.form['startDate']
    endDate = request.form['endDate']
    start = datetime.datetime.strptime(startDate, "%m/%d/%Y %I:%M %p")
    end = datetime.datetime.strptime(endDate, "%m/%d/%Y %I:%M %p")
    deltaTgl = end - start

    cur = mysql.connection.cursor()

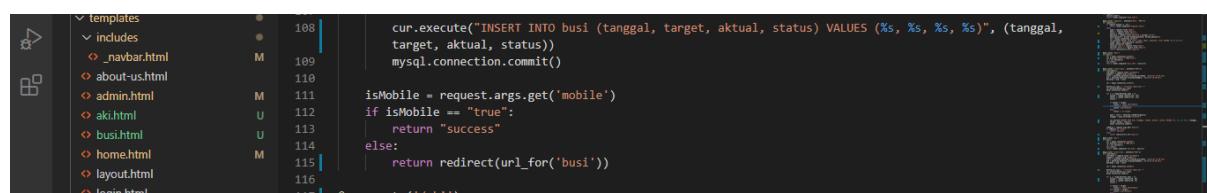
    Delete_all_rows = """truncate table busi """
    cur.execute(Delete_all_rows)
    mysql.connection.commit()

    for x in range(deltaTgl.days + 1):
        target = random.randint(126, 137)
        aktual = random.randint(127, 142)
        status = ''

        if aktual < target:
            status = 'under performance'
        elif aktual > target :
            status ='performance'
        else :
            status = 'on target'

        date = start + datetime.timedelta(days=x)
        tanggal = date.strftime("%d/%m/%Y")
```

The interface includes the Explorer sidebar showing project files like `pa`, `templates`, and `static`. The bottom status bar shows the Python environment and the current date and time.



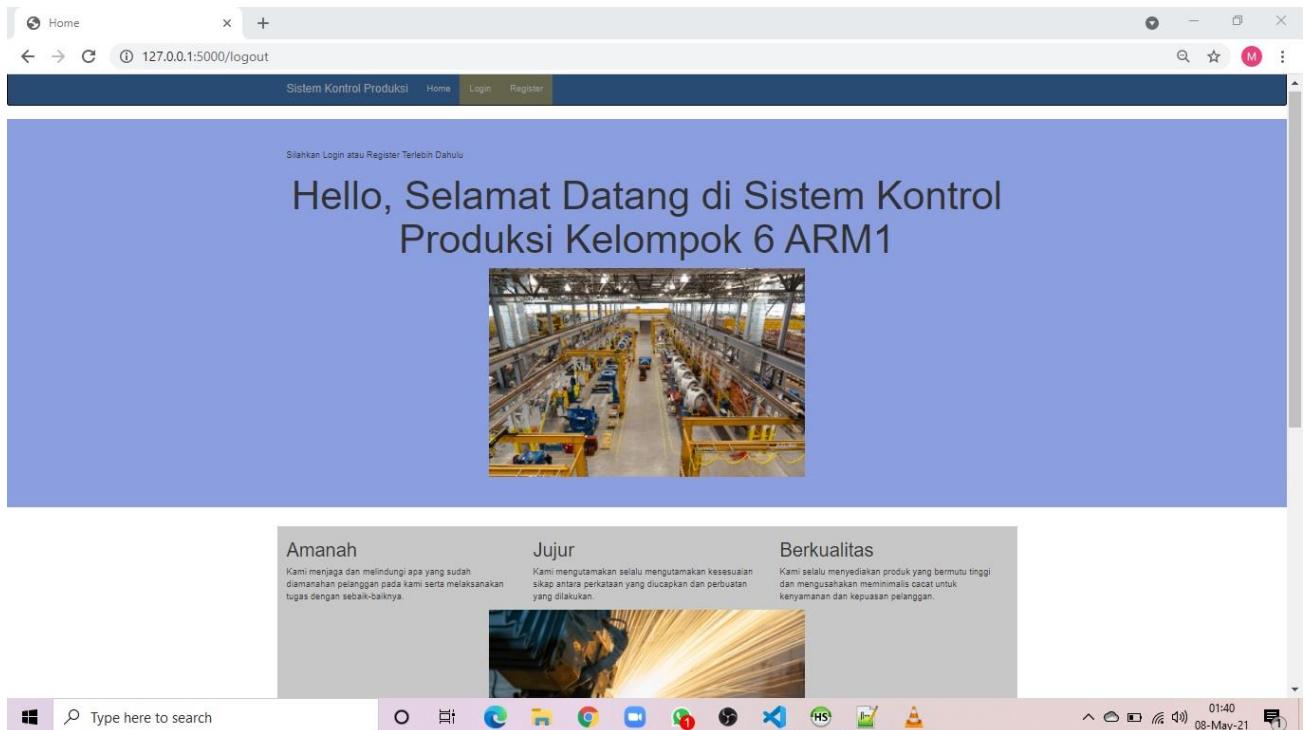
The screenshot shows the Visual Studio Code interface with the file `busi.html` open in the templates folder. The code integrates the simulator logic into the template logic. The code is as follows:

```
cur.execute("INSERT INTO busi (tanggal, target, aktual, status) VALUES (%s, %s, %s, %s)", (tanggal,
target, aktual, status))
mysql.connection.commit()

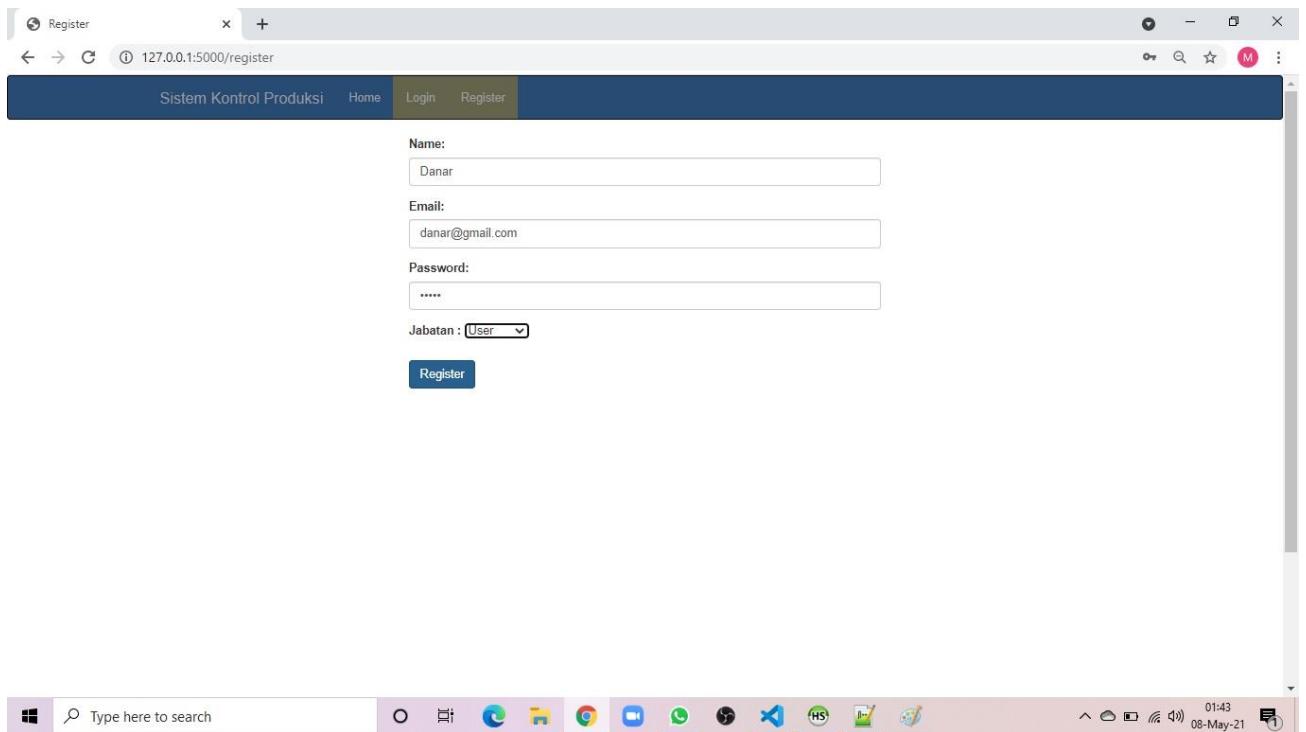
isMobile = request.args.get('mobile')
if isMobile == "true":
    return "success"
else:
    return redirect(url_for('busi'))
```

Tampilan WEB

- + Tampilan pada **home** sebelum login



- + Tampilan pada **register**



⊕ Tampilan pada **login** jika akun sudah di approve oleh admin

registrasi anda berhasil! Silahkan Login...

Email:
danar@gmail.com

Password:
.....

Login

⊕ Tampilan pada **home** setelah login

Login Berhasil! Selamat Datang Kakak Danar

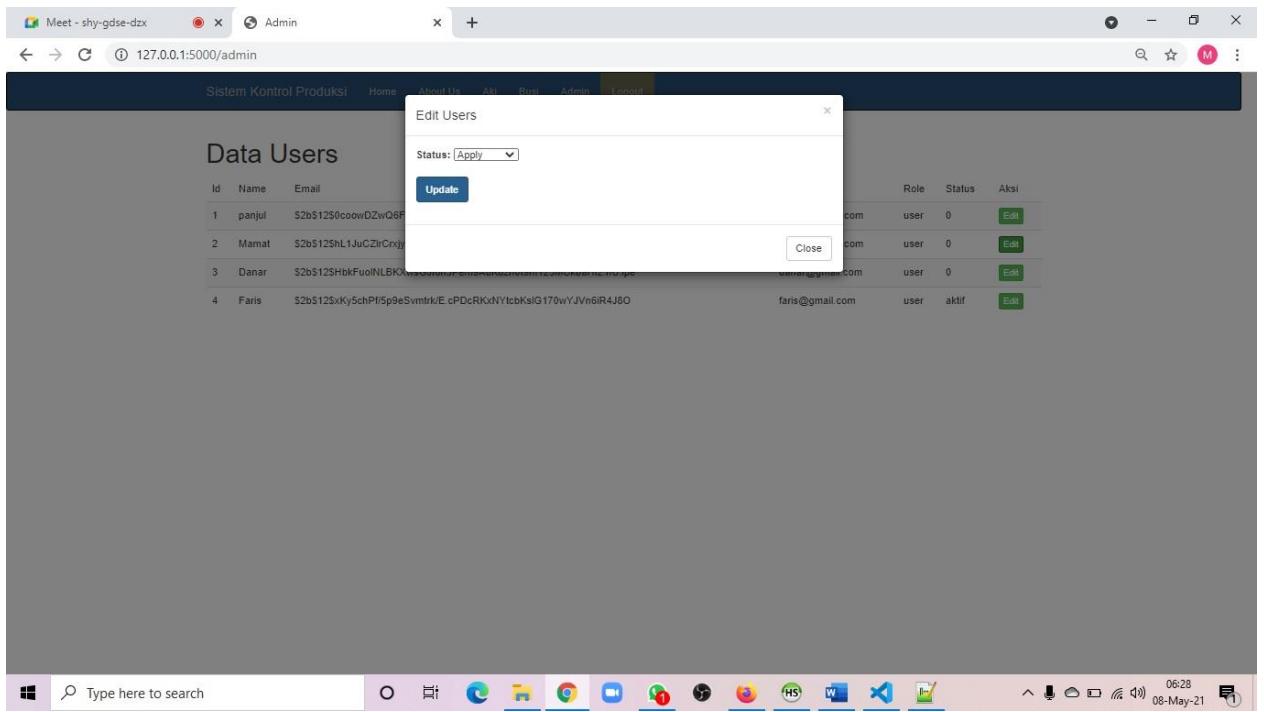
Hello, Selamat Datang di Sistem Kontrol Produksi Kelompok 6 ARM1

Amanah
Kami menjaga dan melindungi apa yang sudah diamanahkan pelanggan pada kami serta melaksanakan tugas dengan sebaik-baiknya.

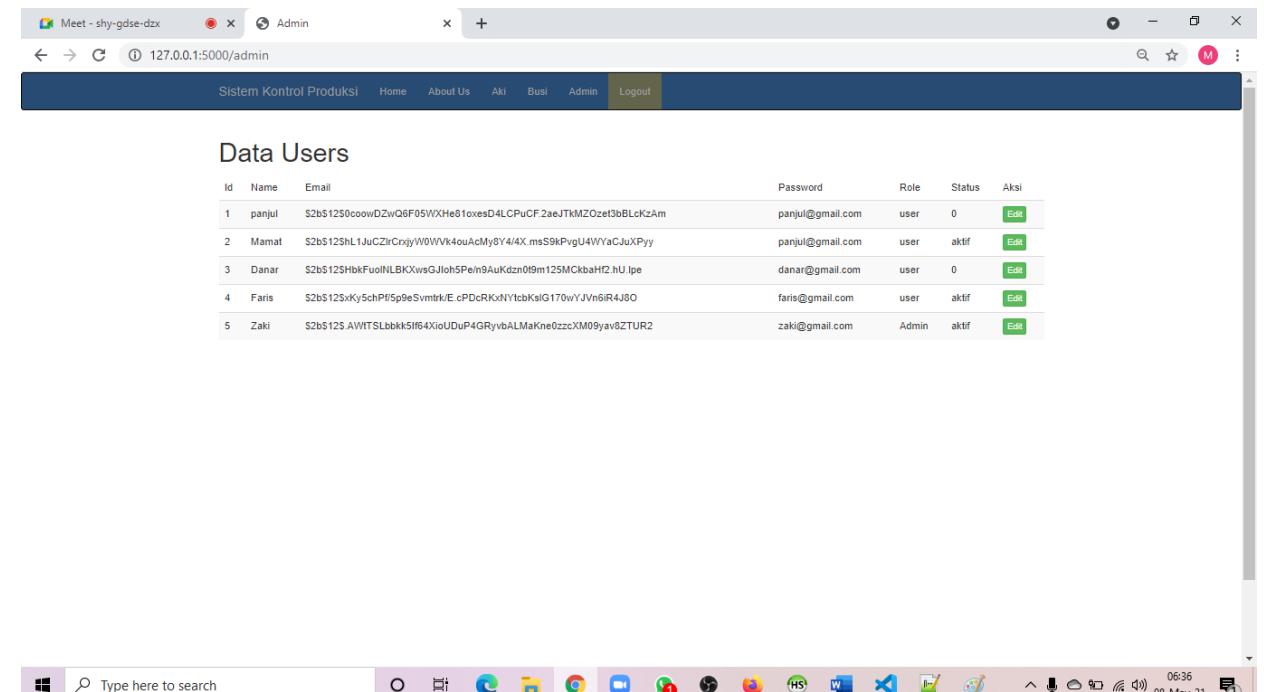
Jujur
Kami mengutamakan selalu mengulamakan kesesuaian sikap antara perkataan yang diucapkan dan perbuatan yang dilakukan.

Berkualitas
Kami selalu menyediakan produk yang bermutu tinggi dan mengusahakan meminimalis cacat untuk kenyamanan dan kepuasan pelanggan.

Tampilan pada admin



The screenshot shows a web application interface for managing user data. A modal dialog titled "Edit Users" is open in the center. It contains a dropdown menu set to "Apply" and a single button labeled "Update". In the background, there is a table titled "Data Users" with columns: Id, Name, Email, Password, Role, Status, and Aksi (Action). The table lists five users: panjul, Mamat, Danar, Faris, and Zaki. Each row has an "Edit" button under the "Aksi" column. The browser's address bar shows the URL "127.0.0.1:5000/admin". The system navigation bar includes links for Home, About Us, Aki, Busi, Admin, and Logout. The status bar at the bottom right indicates the date and time as "08-May-21 06:28".



The screenshot shows the same web application interface after an update. The "Edit Users" modal is no longer visible. The "Data Users" table now shows six rows, including the previously listed users and a new entry for "Zaki". The "Aksi" column for Zaki shows an "Edit" button. The browser's address bar, system navigation bar, and status bar are identical to the previous screenshot.

Data ter-input masuk ke dalam database

```

158 SHOW PROCEDURE STATUS WHERE `Db`='production';
159 SHOW TRIGGERS FROM `production`;
160 SELECT * FROM information_schema.EVENTS AS `EVENT_NAME` FROM information_schema.EVENTS WHERE `EVENT_SCHEMA`='production';
161 SELECT * FROM information_schema.COLUMNS WHERE TABLE_SCHEMA='production' AND TABLE_NAME='users' ORDER BY ORDINAL_POSITION;
162 SHOW INDEXES FROM users FROM `production`;
163 SELECT * FROM information_schema.REFERENTIAL_CONSTRAINTS WHERE CONSTRAINT_SCHEMA='production' AND TABLE_NAME='users' AND REFERENCED_TABLE_NAME IS NOT NULL;
164 SELECT * FROM information_schema.KEY_COLUMN_USAGE WHERE CONSTRAINT_SCHEMA='production' AND TABLE_NAME='users' AND REFERENCED_TABLE_NAME IS NOT NULL;
165 SELECT * FROM `production`.`users` LIMIT 1000;
166 /* Entering session "kelompok6" */
167 UPDATE `production`.`users` SET `id`=5 WHERE `id`=12;
168 SELECT `id`, `name`, `email`, `password`, `role`, `status` FROM `production`.`users` WHERE `id`=5;

```

Connected: 01:04 h MariaDB 10.5.9 Uptime: 1 days, 18:01 h Server time: 06:35 Idle. 0635 08-May-21

Tampilan pada about us

Anggota :

1. Muhammad Faris A (19/441202/SV/16554)
2. Tania Desnatasari (19/441212/SV/16564)
3. Bramantyo Wahyu K (19/447113/SV/16832)
4. Muhammad Risang D (19/447119/SV/16838)
5. Zaenal Muttaqin (19/447129/SV/16848)

⊕ Tampilan pada Busi

Sistem Kontrol Produksi

Monitoring Produksi

Home About Us Aki Busi Logout

Data Produksi Busi

Input Tanggal Awal : 05/01/2021 3:04 AM

Input Tanggal Akhir : 05/09/2021 3:04 AM

Lihat Data

Diagram Batang Sistem Monitoring Busi

Tanggal	Jumlah Produksi Busi	Target Produksi Busi	Status Produksi
01/05/2021	135	133	performance
02/05/2021	136	135	performance
03/05/2021	140	132	performance
04/05/2021	138	135	performance
05/05/2021	129	134	under performance
06/05/2021	130	127	performance
07/05/2021	141	135	performance
08/05/2021	139	137	performance
09/05/2021	135	128	performance

Aktual
Target

08-May-21 03:04

⊕ Tampilan pada aki

Sistem Kontrol Produksi

Monitoring Produksi

Home About Us Aki Busi Logout

Data Produksi Aki

Input Tanggal Awal : 05/01/2021 2:14 AM

Input Tanggal Akhir : 05/09/2021 2:14 AM

Lihat Data

Diagram Batang Sistem Monitoring Produksi Aki

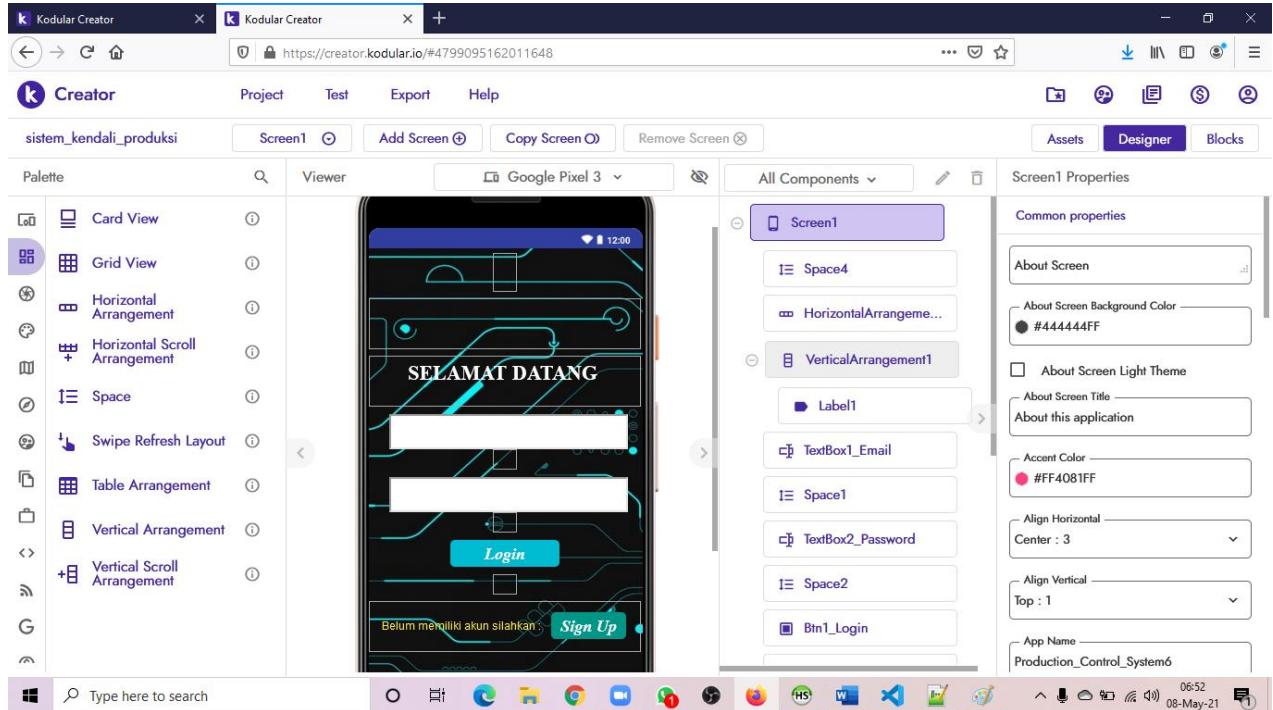
Tanggal	Jumlah Produksi Aki	Target Produksi Aki	Status Produksi
01/05/2021	60	50	performance
02/05/2021	58	55	performance
03/05/2021	67	51	performance
04/05/2021	54	52	performance
05/05/2021	67	58	performance
06/05/2021	54	58	under performance
07/05/2021	68	58	performance
08/05/2021	61	59	performance
09/05/2021	51	51	on target

Aktual
Target

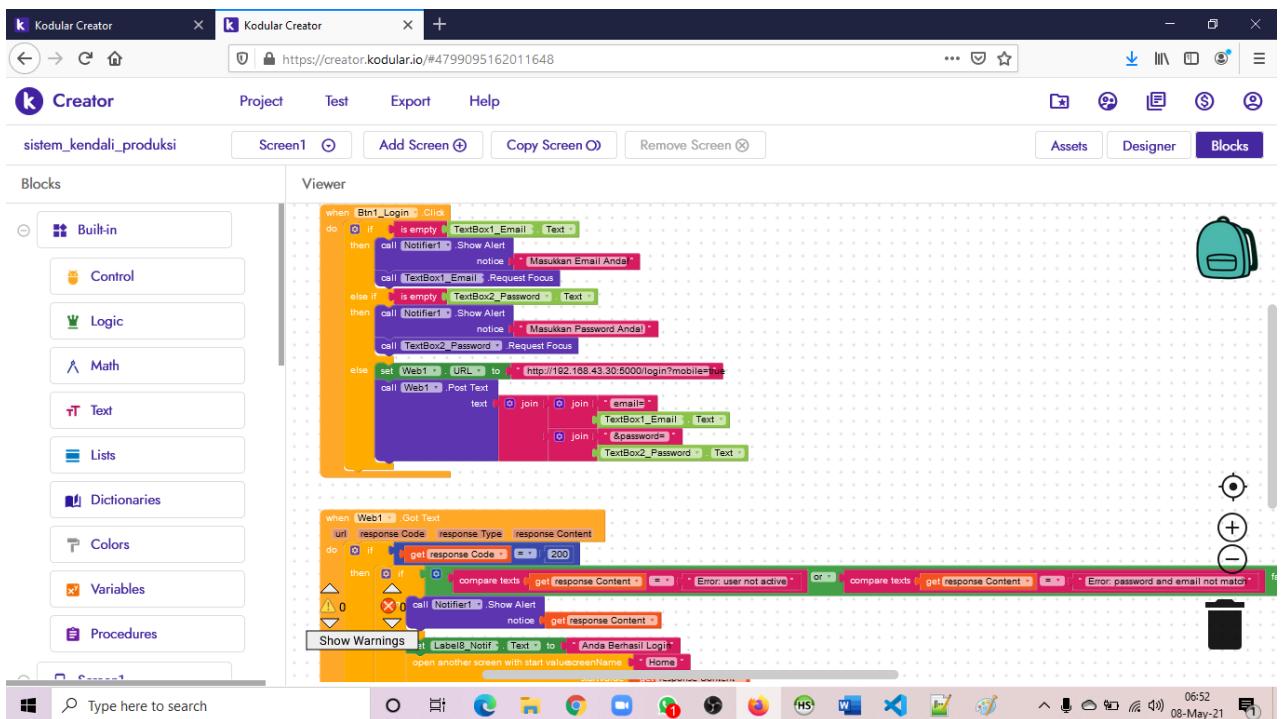
08-May-21 02:14

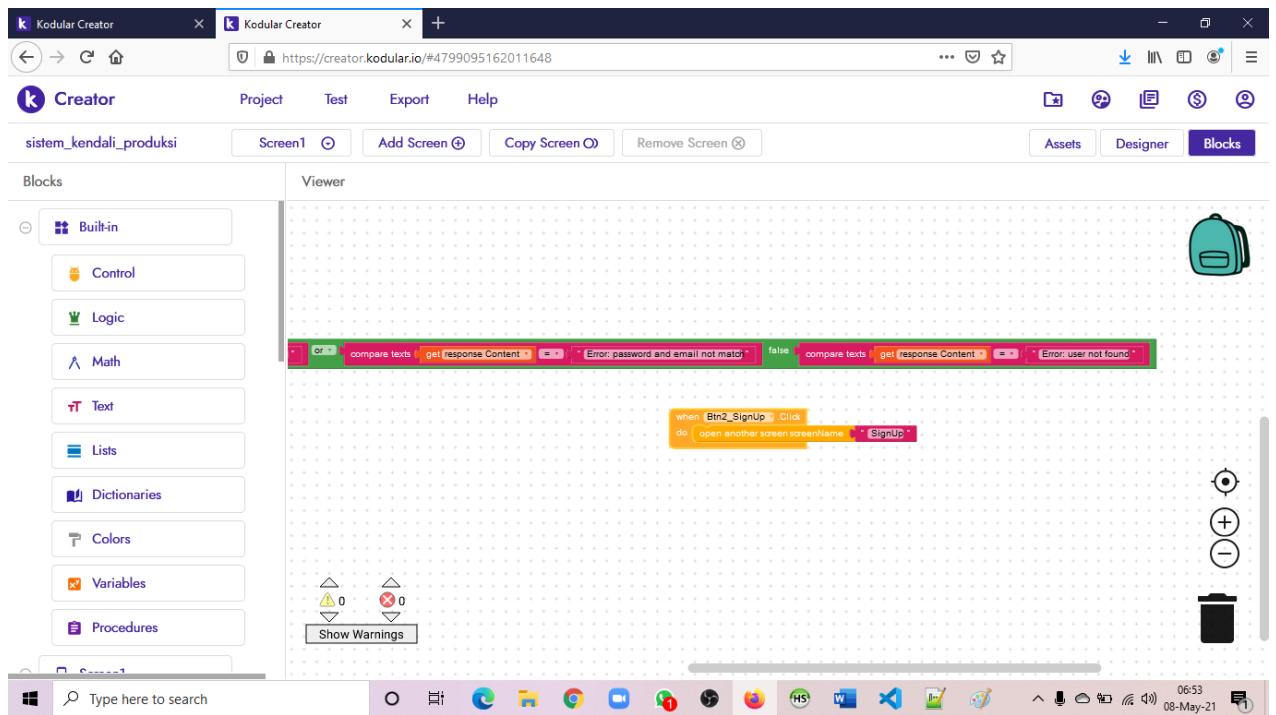
3. Membuat mobile apps dengan kodular

⊕ Desain screen 1 (login)

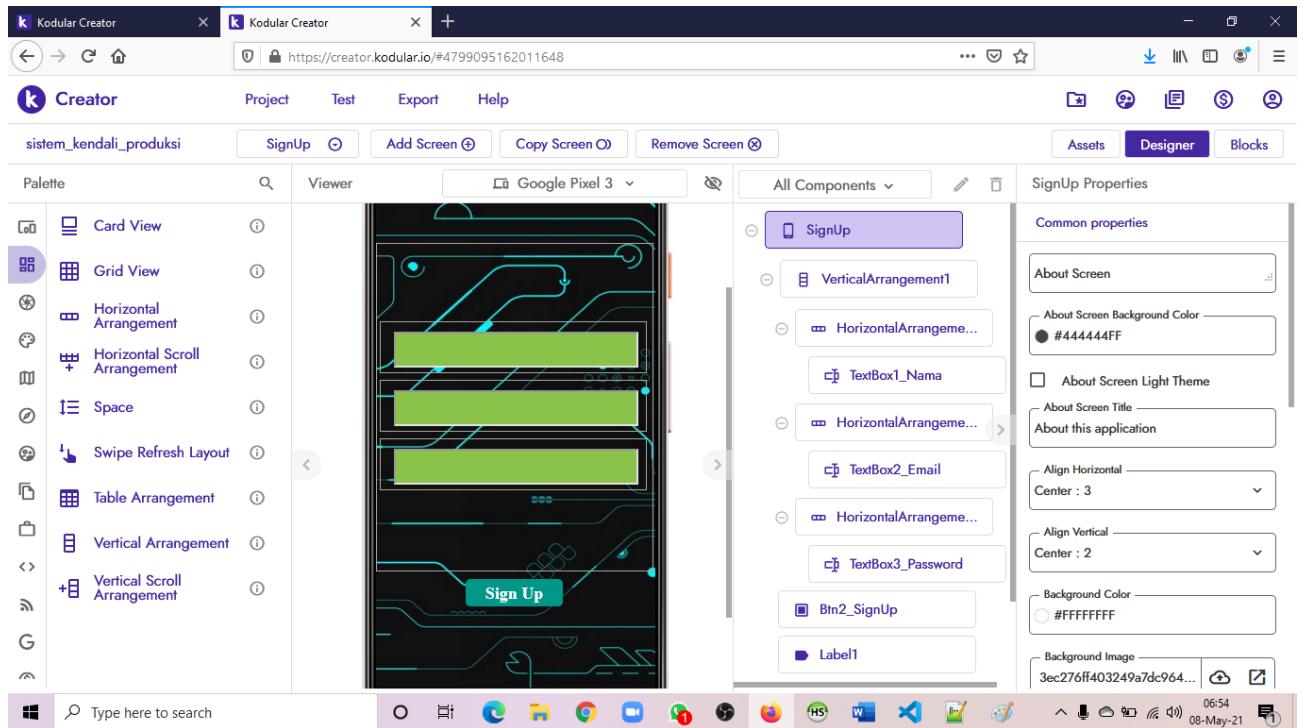


⊕ Blocks pada screen 1

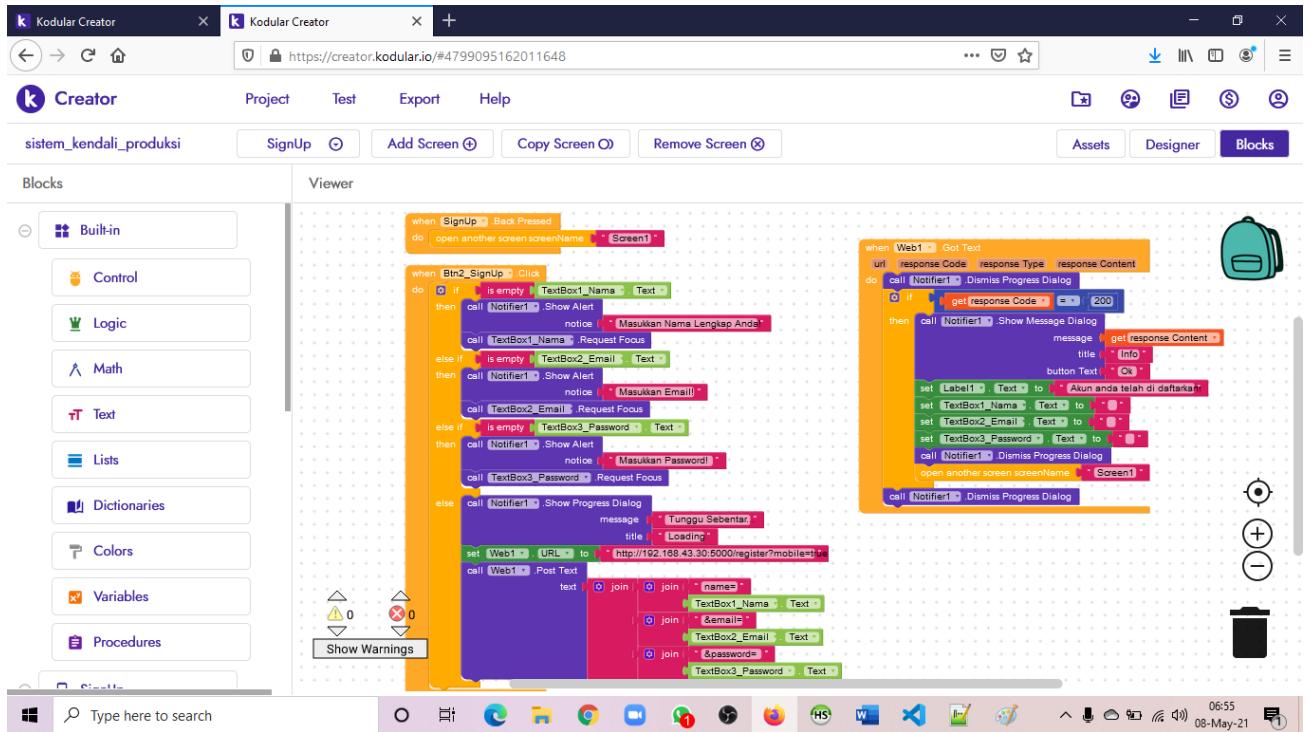




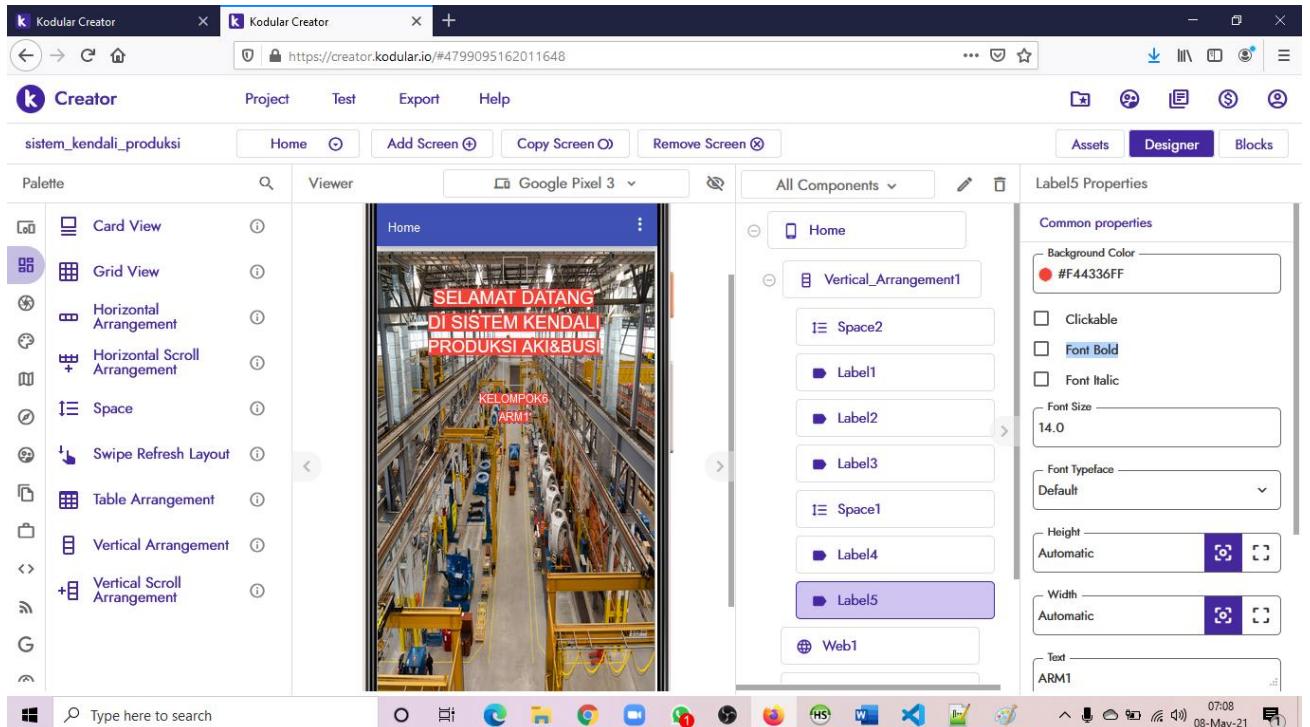
➊ Desain screen signup



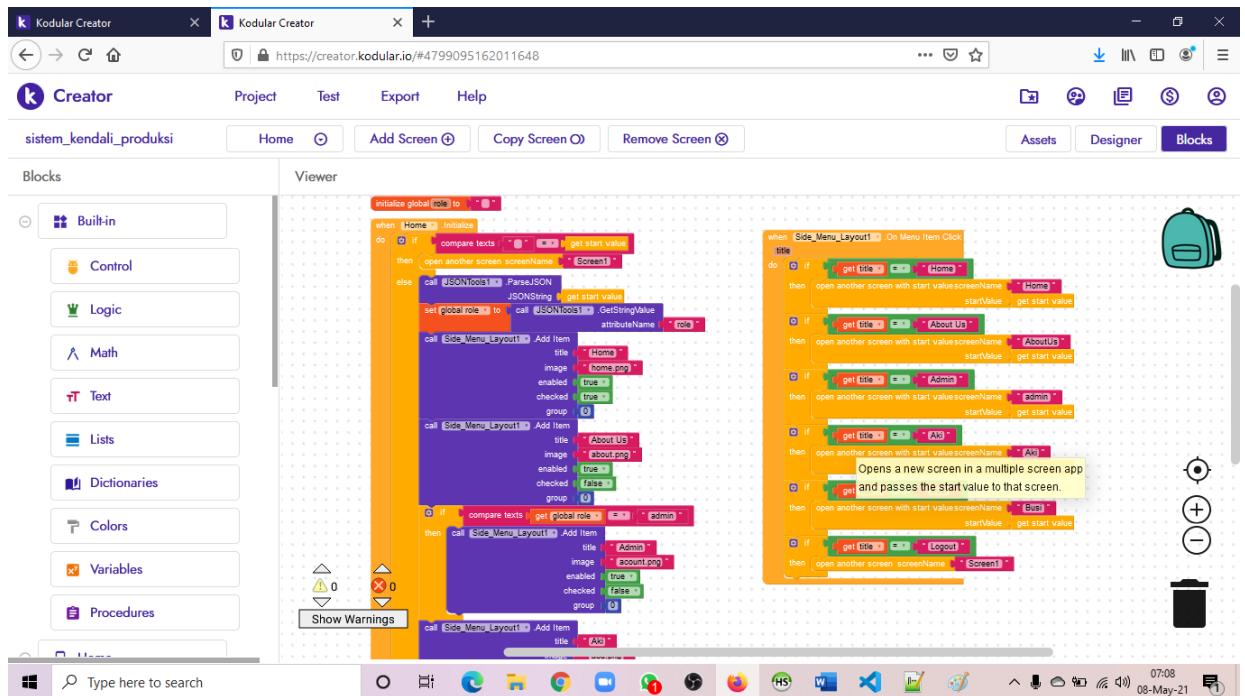
Blocks pada screen signup



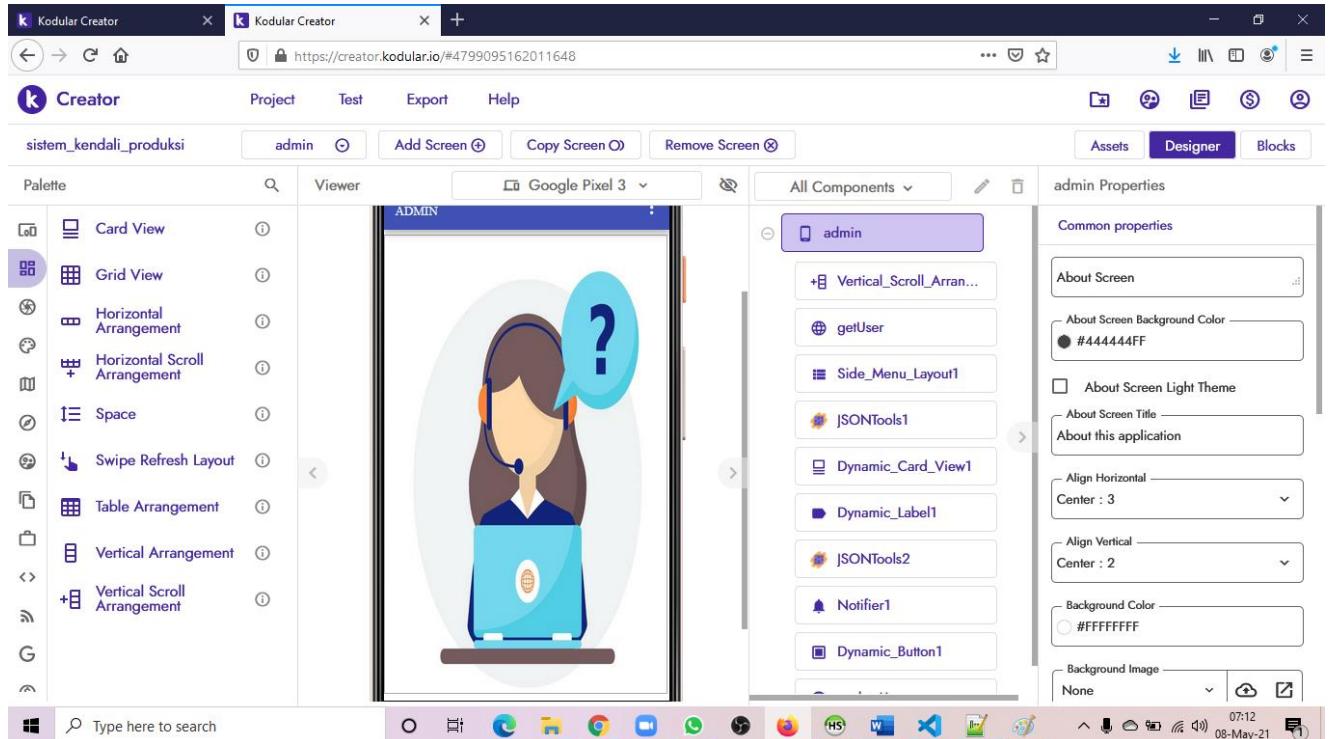
Desain screen home dengan menyantumkan sidebar



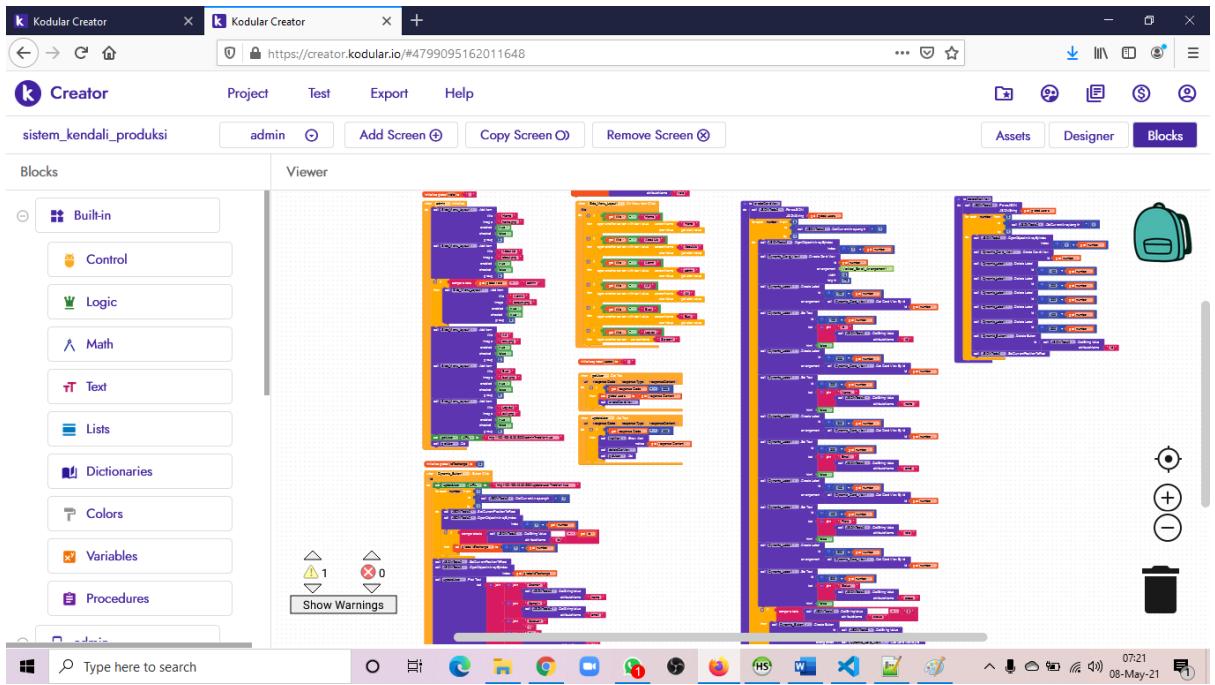
Blocks pada screen home dengan sidebar



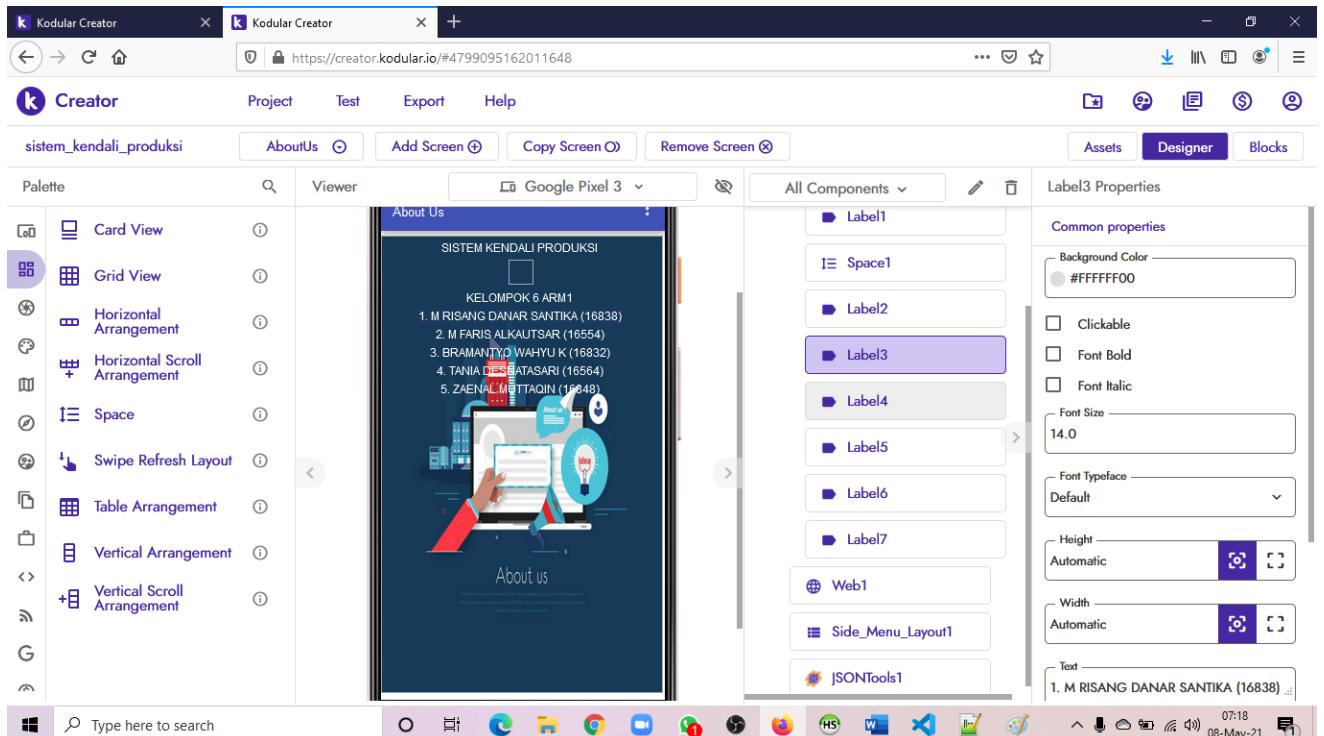
Desain screen admin dengan menyantumkan sidebar



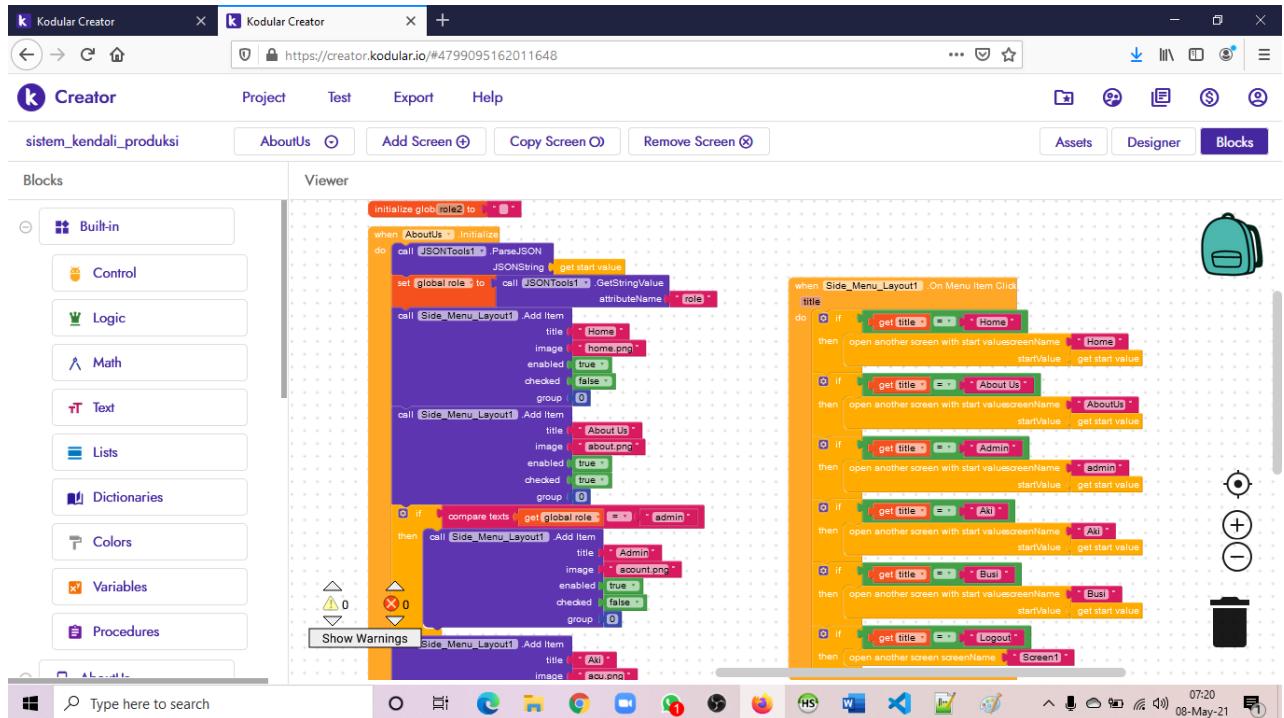
Blocks pada screen admin dengan sidebar



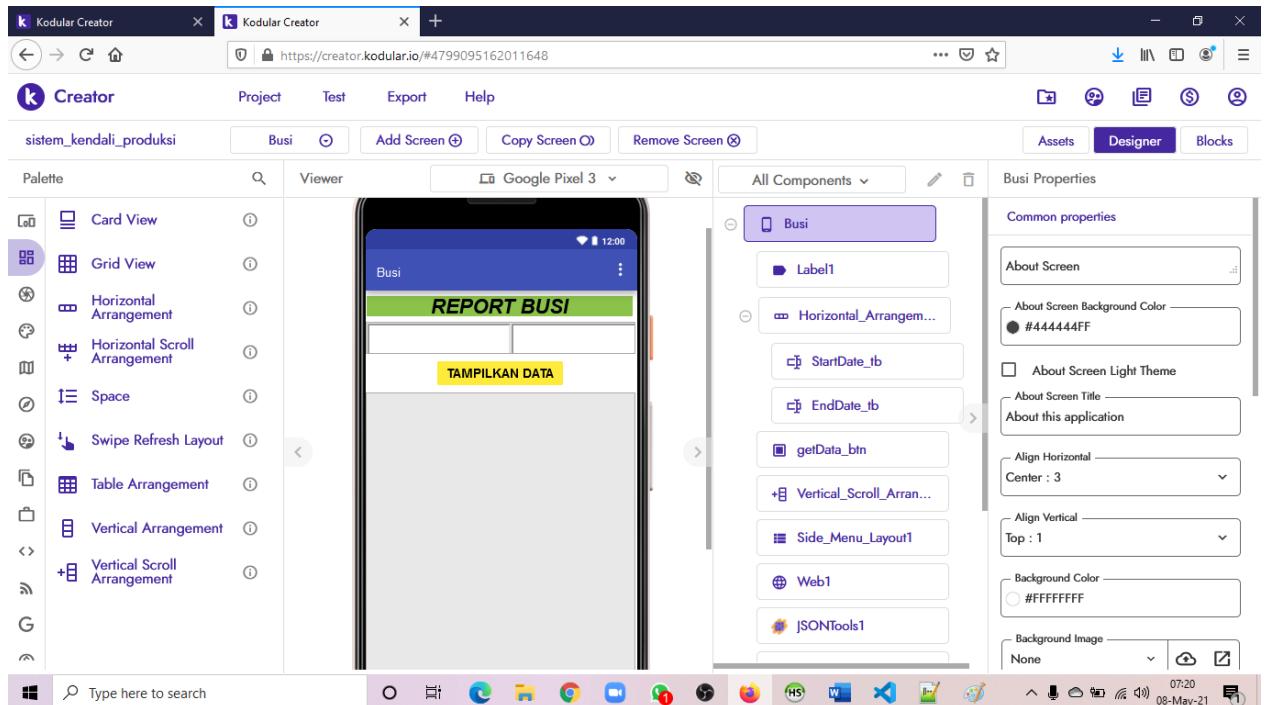
Desain screen about us dengan menyantumkan sidebar



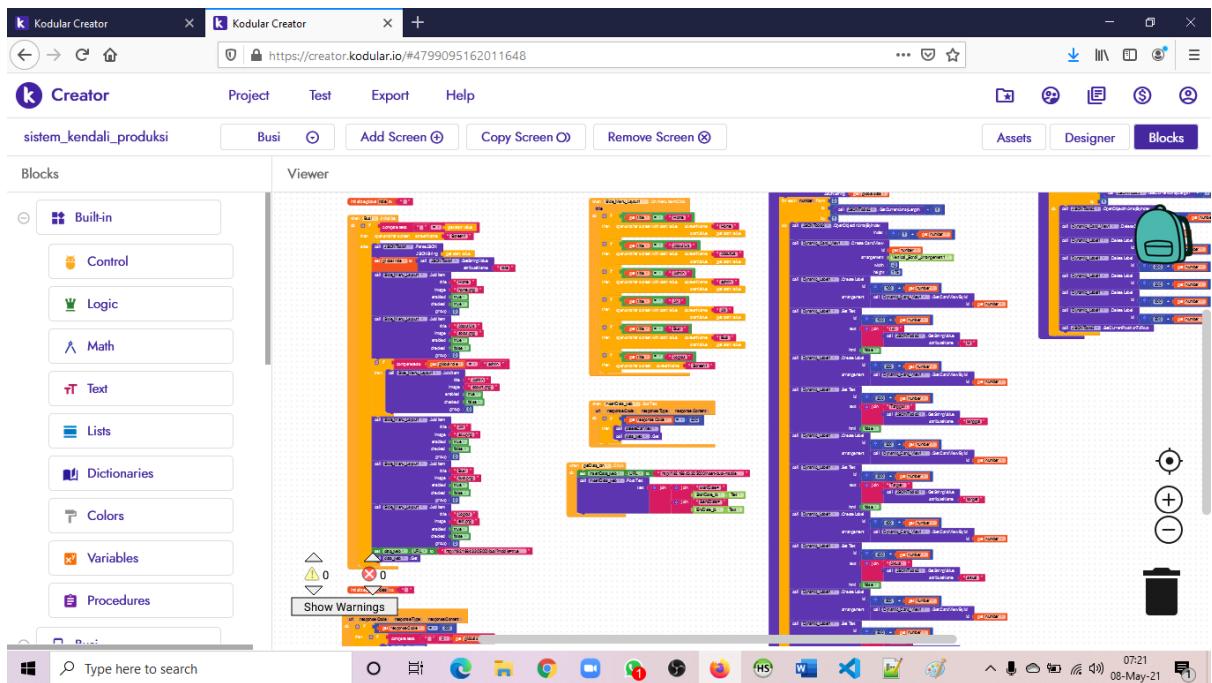
Blocks pada screen about us dengan sidebar



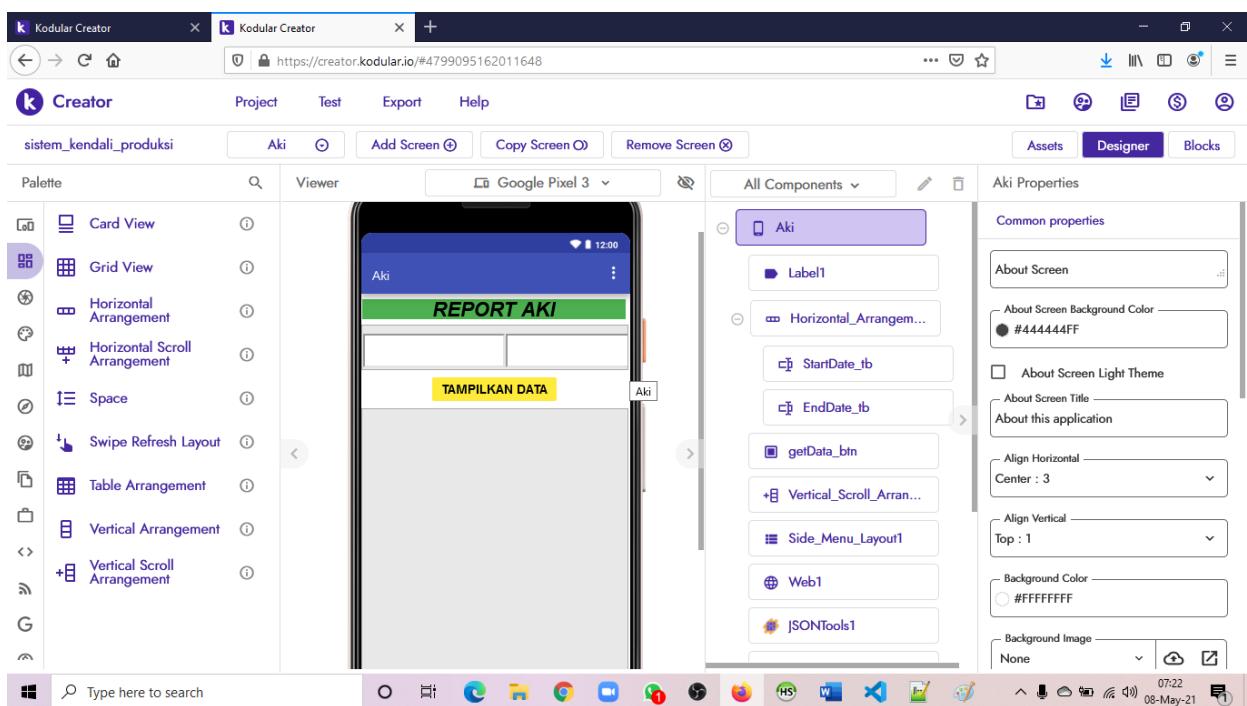
Desain screen busi dengan menyantumkan sidebar



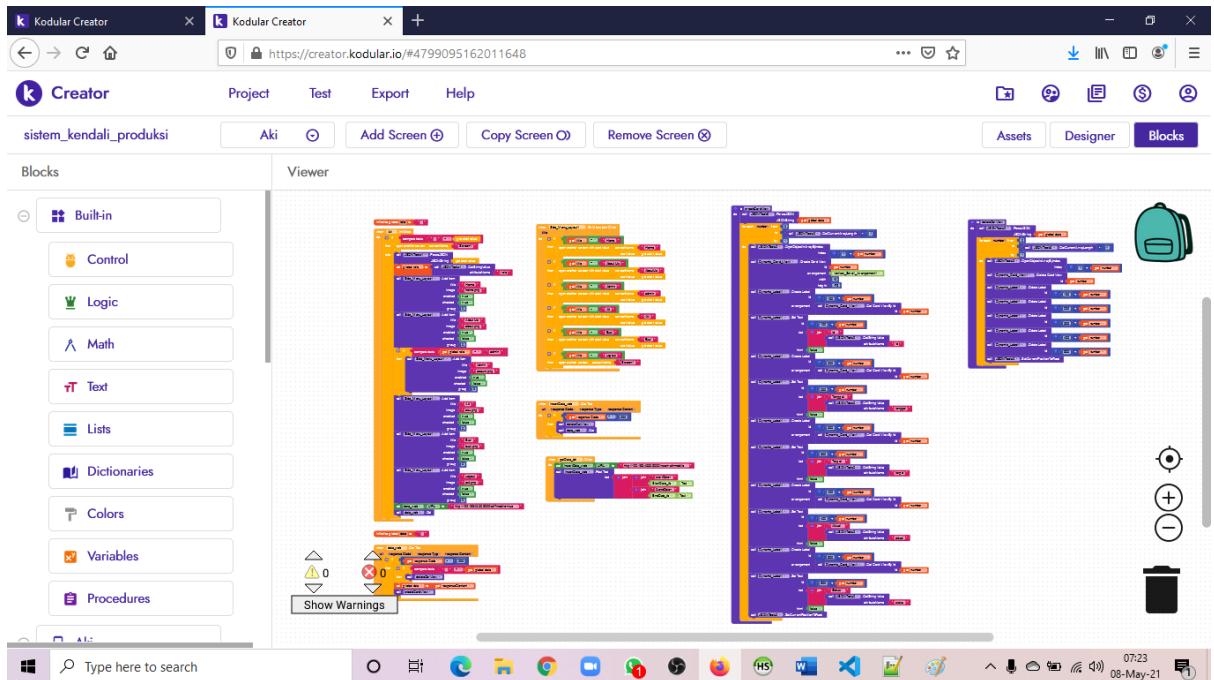
Blocks pada screen busi dengan sidebar



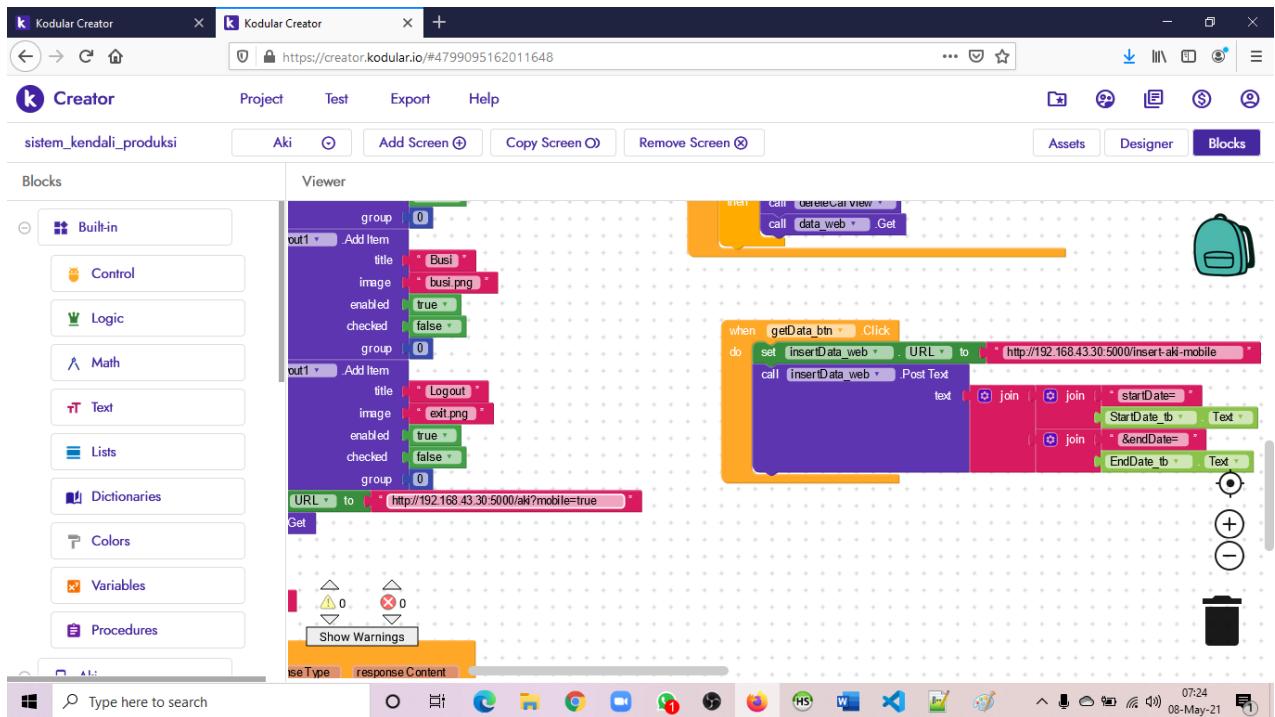
Desain screen aki dengan menyantumkan sidebar



Blocks pada screen aki dengan sidebar



4. Menghubungkan kodingan vscode dengan kodular.

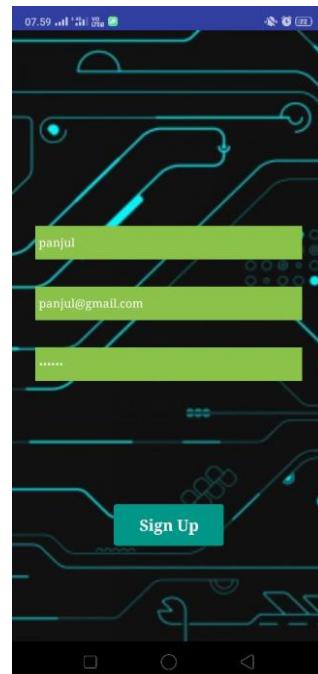


Tampilan pada mobile app

➊ Tampilan screen 1 (login)



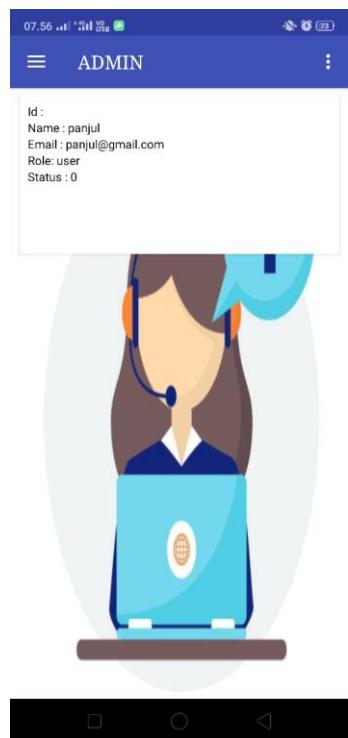
➋ Tampilan screen signup



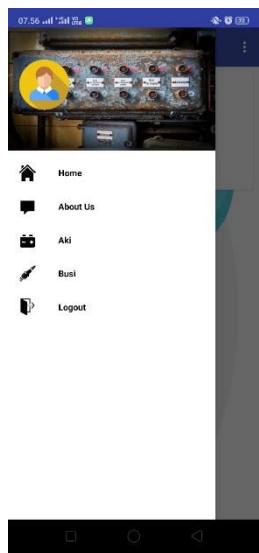
■ Tampilan screen home



■ Tampilan screen admin



✍ Contoh sidebar



✍ Tampilan screen about us

