LAB 09

Chat-app backend

- CRUD operations on the Login/Signup Pages and If possible than also on some other important key feature(s) of your project.

Models

```
const mongoose = require('mongoose');

const MessageSchema = new mongoose.Schema({
 user: { type: String, required: true },
 text: { type: String, required: true },
 timestamp: { type: Date, default: Date.now }
});

module.exports = mongoose.model('Message', MessageSchema);
```

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
 username: { type: String, required: true, unique: true },
 password: { type: String, required: true },
 email: { type: String, required: true, unique: true },
 firstName: { type: String },
 lastName: { type: String },
 createdAt: { type: Date, default: Date.now },
 profilePicture: { type: String },
 bio: { type: String },

 // Fields specific to period tracking
 cycleLength: { type: Number, default: 28 }, // Average cycle length in
days
 periodLength: { type: Number, default: 5 }, // Average period length in
days
 lastPeriodDate: { type: Date }, // Date of the last period
 ovulationDate: { type: Date }, // Estimated ovulation date
 notes: [{ // User can store notes related to their cycle
   date: { type: Date },
   content: { type: String }
```

```
 }],
});

// Indexing for faster searching by username or email
userSchema.index({ username: 1, email: 1 });

module.exports = mongoose.model('User', userSchema);
```

Routers

```
const express = require('express');
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const router = express.Router();

// User model
const User = require('../models/User'); // Adjust the path according to
your project structure

// Register
router.post('/register', async (req, res) => {
   // Log the incoming request body
   console.log(req.body);

   const { username, password, email, firstName, lastName, cycleLength,
periodLength } = req.body;

   // Input validation
   if (!username || !password || !email) {
     return res.status(400).json({ message: 'Username, password, and email
are required' });
   }

   try {
     const hashedPassword = await bcrypt.hash(password, 10);

     const user = new User({
       username,
       password: hashedPassword,
       email,
       firstName,
       lastName,
       cycleLength,
       periodLength
```

```
    });

    await user.save();
    res.status(201).json({ message: 'User registered', user });
  } catch (err) {
    // Log the error for debugging
    console.error('Error during registration:', err);
    res.status(400).json({ message: 'Registration failed', error:
err.message });
  }
});


// Login
router.post('/login', async (req, res) => {
 const { username, password } = req.body;

 const user = await User.findOne({ username });
 if (!user || !(await bcrypt.compare(password, user.password))) {
   return res.status(401).send('Invalid credentials');
 }

 const token = jwt.sign({ id: user._id }, 'your_jwt_secret');
 res.json({ token, user: { username: user.username, email: user.email,
firstName: user.firstName, lastName: user.lastName } });
});

module.exports = router;
```

```
const express = require('express');
const router = express.Router();
const Message = require('../models/Message');


// Get all messages
router.get('/', async (req, res) => {
 try {
   console.log('GET /messages');
   const messages = await Message.find();
   res.json(messages);
 } catch (err) {
   res.status(500).json({ message: err.message });
 }
});

// Post a new message
```

```
router.post('/', async (req, res) => {
 console.log('POST /messages');
 const { user, text } = req.body;
 const newMessage = new Message({ user, text });

 try {
   const savedMessage = await newMessage.save();
   res.status(201).json(savedMessage);
 } catch (err) {
   res.status(400).json({ message: err.message });
 }
});

module.exports = router;
```

```
const express = require('express');
const mongoose = require('mongoose');
const http = require('http');
const socketIo = require('socket.io');
const cors = require('cors');
const app = express();
const server = http.createServer(app);
const io = socketIo(server);
const port = 3000;
const mongoURI = 'mongodb://localhost:27017/chat-backend';

// MongoDB connection
mongoose.connect(mongoURI, { useNewUrlParser: true, useUnifiedTopology:
true })
 .then(() => console.log('MongoDB connected'))
 .catch(err => console.log('MongoDB connection error:', err));

// Middleware
app.use(cors());
app.use(express.json());

// Routes
const messageRoutes = require('./routes/messages');
const authRoutes = require('./routes/auth'); // Import auth routes
app.use('/api/messages', messageRoutes);
app.use('/api/auth', authRoutes); // Use auth routes

// Socket.IO connection
io.on('connection', (socket) => {
 console.log('New client connected');
 socket.emit('message', { text: 'Welcome to the chat!' });
```

```
 socket.on('message', (msg) => {
   io.emit('message', msg);
 });

 socket.on('disconnect', () => {
   console.log('Client disconnected');
 });
});

// Basic route
app.get('/', (req, res) => {
 res.send('Chat backend server is running');
});

// Listen on all interfaces
server.listen(port, '0.0.0.0', () => {
 console.log(`Server is running on http://0.0.0.0:${port}`);
});
```

Chat_screen.dart

```
                import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:socket_io_client/socket_io_client.dart' as IO;
import 'dart:convert';
import 'package:sdp/services/auth_service.dart'; // Import your
AuthService here
import 'auth_screen.dart'; // Import AuthScreen for navigation

class ChatScreen extends StatefulWidget {
 static const String id = 'chat_screen';

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

class _ChatScreenState extends State<ChatScreen> {
 final TextEditingController _messageController = TextEditingController();
 final List<String> _messages = [];
 late IO.Socket _socket;

 @override
 void initState() {
   super.initState();
```

```
  _checkAuthentication();
}

Future<void> _checkAuthentication() async {
  final authService = AuthService();
  bool isLoggedIn = await authService.isLoggedIn();
  if (!isLoggedIn) {
    Navigator.pushReplacementNamed(context, AuthScreen.id);
  } else {
    _socket = IO.io('http://10.10.30.100:3000', <String, dynamic>{
      'transports': ['websocket'],
      'autoConnect': false,
    });

    _socket.connect();

    _socket.on('message', (data) {
      setState(() {
        _messages.add(data['text']);
      });
    });

    _socket.on('connect_error', (error) {
      print('Connection Error: $error');
    });
  }
}

@override
void dispose() {
  _socket.disconnect();
  super.dispose();
}

void _sendMessage(String message) {
  if (_socket.connected) {
    _socket.emit('message', {'text': message});
    _messageController.clear();
  } else {
    print('Socket is not connected');
  }
}

Widget _buildMessageBubble(String message) {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 4.0, horizontal: 10.0),
    child: Align(
```

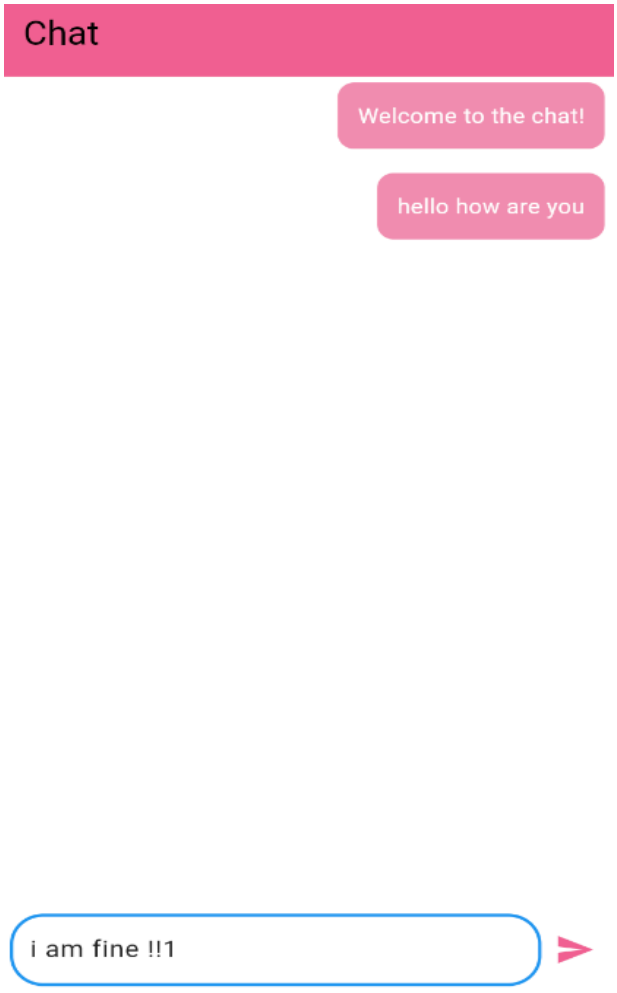
```
      alignment: Alignment.centerRight,
      child: Container(
        padding: EdgeInsets.all(12.0),
        decoration: BoxDecoration(
          color: Colors.pink[200],
          borderRadius: BorderRadius.circular(10.0),
        ),
        child: Text(
          message,
          style: TextStyle(color: Colors.white),
        ),
      ),
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Chat'),
      backgroundColor: Colors.pink[300],
    ),
    body: Column(
      children: [
        Expanded(
          child: ListView.separated(
            itemCount: _messages.length,
            itemBuilder: (context, index) {
              return _buildMessageBubble(_messages[index]);
            },
            separatorBuilder: (context, index) => SizedBox(height: 8.0),
          ),
        ),
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Row(
            children: [
              Expanded(
                child: TextField(
                  controller: _messageController,
                  decoration: InputDecoration(
                    hintText: 'Enter your message',
                    border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(20.0),
                      borderSide: BorderSide(color: Colors.pink[200]!),
                    ),
```

```
                    filled: true,
                    fillColor: Colors.white,
                  ),
                ),
              ),
              IconButton(
                icon: Icon(Icons.send, color: Colors.pink[300]),
                onPressed: () {
                  final message = _messageController.text;
                  if (message.isNotEmpty) {
                    _sendMessage(message);
                  }
                },
              ),
            ],
          ),
        ),
      ],
    ),
  );
 }
}
```

ChatCard.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/cupertino.dart';
import '../screens/chat_screen.dart';  // Import the chat screen

class ChatCard extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return GestureDetector(
     onTap: () {
       Navigator.pushNamed(context, ChatScreen.id);  // Navigate to chat
screen
     },
     child: Container(
       width: 200,
       margin: EdgeInsets.all(8.0),
       padding: EdgeInsets.all(16.0),
       decoration: BoxDecoration(
         color: Colors.pink[100],
         borderRadius: BorderRadius.circular(12),
         boxShadow: [
```

```
          BoxShadow(
            color: Colors.grey.withOpacity(0.3),
            spreadRadius: 3,
            blurRadius: 5,
            offset: Offset(0, 3),
          ),
        ],
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(Icons.chat, size: 48, color: Colors.blue[700]),
          SizedBox(height: 10),
          Text(
            'Girls_Chat_Only',
            style: TextStyle(
              fontSize: 18,
              fontWeight: FontWeight.bold,
              color: Colors.pink[900],
            ),
          ),
          SizedBox(height: 8),
          Text(
            'Track your period and get insights.',
            textAlign: TextAlign.center,
            style: TextStyle(
              fontSize: 14,
              color: Colors.pink[800],
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```



Authentication

LOGIN.dart

```
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'package:flutter_secure_storage/flutter_secure_storage.dart';

class AuthService {
 final storage = FlutterSecureStorage();

 // Check if the user is logged in by reading the token from secure
storage
 Future<bool> isLoggedIn() async {
```

```
  String? token = await storage.read(key: 'auth_token');
  return token != null;
}

// Save the token to secure storage
Future<void> saveToken(String token) async {
  await storage.write(key: 'auth_token', value: token);
}

// Logout by deleting the token from secure storage
Future<void> logout() async {
  await storage.delete(key: 'auth_token');
}

// Login function to authenticate the user
Future<String?> login(String username, String password) async {
  final response = await http.post(
    Uri.parse('http://192.168.1.3:3000/api/auth/login'), // Your server's
login URL
    headers: {
      'Content-Type': 'application/json', // Set Content-Type to JSON
    },
    body: json.encode({
      'username': username,
      'password': password,
    }),
  );

  if (response.statusCode == 200) {
    final data = json.decode(response.body);
    String token = data['token']; // Assumes that the response contains a
'token' field
    await saveToken(token);
    return token;
  } else {
    return null; // Return null if login fails
  }
}

// Registration function to create a new user account
Future<bool> register(
    String username,
    String password,
    String email,
    String firstName,
    String lastName,
    int cycleLength,
```

```
    int periodLength,
    ) async {
  final response = await http.post(
    Uri.parse('http://10.10.30.100:3000/api/auth/register'), // Your
server's registration URL
    headers: {
      'Content-Type': 'application/json', // Set Content-Type to JSON
    },
    body: json.encode({
      'username': username,
      'password': password,
      'email': email,
      'firstName': firstName,
      'lastName': lastName,
      'cycleLength': cycleLength,
      'periodLength': periodLength,
    }),
  );

  return response.statusCode == 201; // Returns true if registration is
successful
 }
}
```

Auth_screen.dart

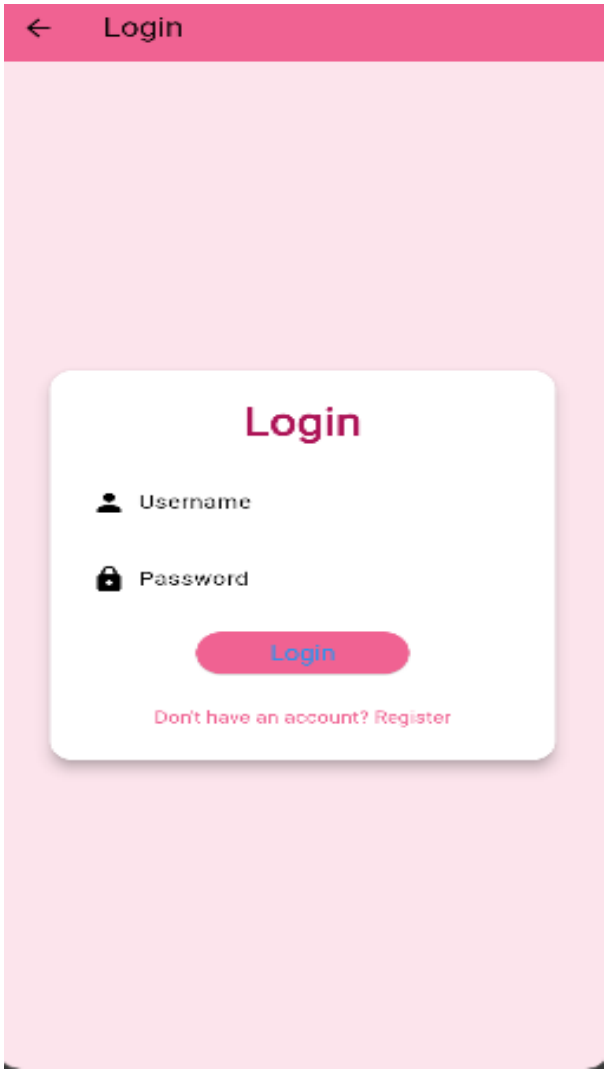
```
import 'package:flutter/material.dart';
import 'package:sdp/services/auth_service.dart'; // Adjust according to
your structure
import 'chat_screen.dart'; // Import ChatScreen for navigation

class AuthScreen extends StatefulWidget {
 static const String id = 'auth_screen';

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

class _AuthScreenState extends State<AuthScreen> {
 final TextEditingController _usernameController =
TextEditingController();
 final TextEditingController _passwordController =
TextEditingController();
 final TextEditingController _emailController = TextEditingController();
```

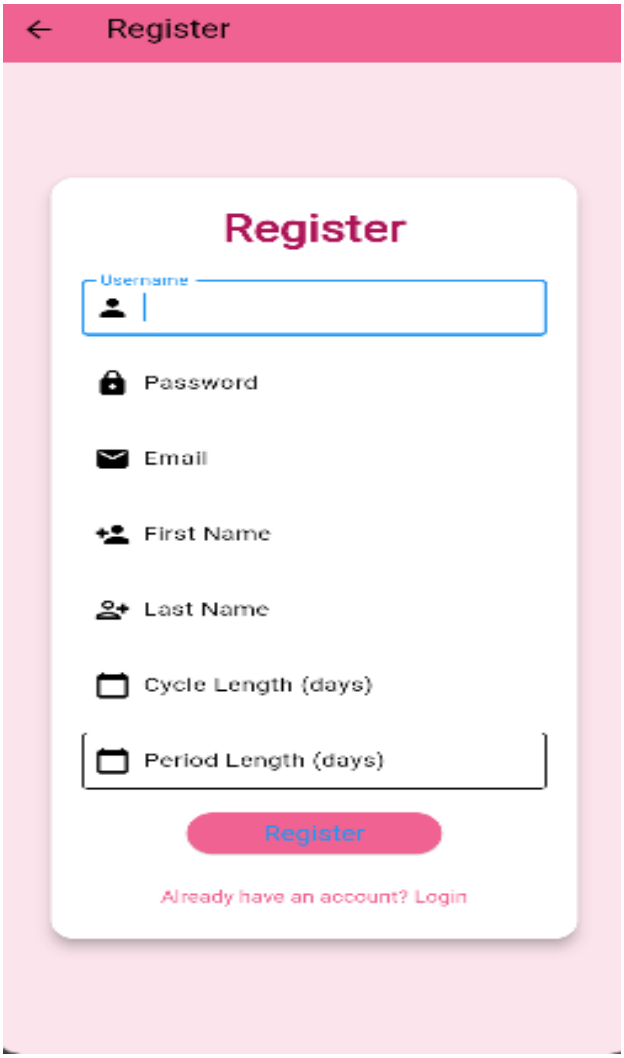
```
 final TextEditingController _firstNameController =
TextEditingController();
 final TextEditingController _lastNameController =
TextEditingController();
 final TextEditingController _cycleLengthController =
TextEditingController();
 final TextEditingController _periodLengthController =
TextEditingController();

 bool _isLogin = true;

 void _submit() async {
   final authService = AuthService();
   if (_isLogin) {
     final token = await authService.login(
       _usernameController.text,
       _passwordController.text,
     );
     if (token != null) {
       Navigator.pushReplacementNamed(context, ChatScreen.id);
     } else {
       _showError('Invalid credentials');
     }
   } else {
     final success = await authService.register(
       _usernameController.text,
       _passwordController.text,
       _emailController.text,
       _firstNameController.text,
       _lastNameController.text,
       int.tryParse(_cycleLengthController.text) ?? 0,
       int.tryParse(_periodLengthController.text) ?? 0,
     );
     if (success) {
       _showSuccess('Registration successful! Please log in.');
       setState(() {
         _isLogin = true;
       });
     } else {
       _showError('Registration failed');
     }
   }
 }

 void _showError(String message) {
   ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:
Text(message)));
```

```
 }

 void _showSuccess(String message) {
   ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:
Text(message)));
 }

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     backgroundColor: Colors.pink[50],
     appBar: AppBar(
       title: Text(_isLogin ? 'Login' : 'Register'),
       backgroundColor: Colors.pink[300],
     ),
     body: Padding(
       padding: const EdgeInsets.all(16.0),
       child: Center(
         child: SingleChildScrollView(
           child: Card(
             elevation: 8,
             shape: RoundedRectangleBorder(
               borderRadius: BorderRadius.circular(16),
             ),
             child: Padding(
               padding: const EdgeInsets.all(20.0),
               child: Column(
                 mainAxisAlignment: MainAxisAlignment.center,
                 children: [
                   Text(
                     _isLogin ? 'Login' : 'Register',
                     style: TextStyle(fontSize: 32, fontWeight:
FontWeight.bold, color: Colors.pink[800]),
                   ),
                   SizedBox(height: 20),
                   _buildTextField(_usernameController, 'Username',
Icons.person),
                   SizedBox(height: 16),
                   _buildTextField(_passwordController, 'Password',
Icons.lock, obscureText: true),
                   if (!_isLogin) ...[
                     SizedBox(height: 16),
                     _buildTextField(_emailController, 'Email',
Icons.email),
                     SizedBox(height: 16),
                     _buildTextField(_firstNameController, 'First Name',
Icons.person_add),
```

```
                    SizedBox(height: 16),
                    _buildTextField(_lastNameController, 'Last Name',
Icons.person_add_alt),
                    SizedBox(height: 16),
                    _buildTextField(_cycleLengthController, 'Cycle Length
(days)', Icons.calendar_today, keyboardType: TextInputType.number),
                    SizedBox(height: 16),
                    _buildTextField(_periodLengthController, 'Period
Length (days)', Icons.calendar_today, keyboardType: TextInputType.number),
                  ],
                  SizedBox(height: 20),
                  ElevatedButton(
                    onPressed: _submit,
                    child: Text(_isLogin ? 'Login' : 'Register'),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.pink[300],
                      padding: EdgeInsets.symmetric(horizontal: 50,
vertical: 15),
                      textStyle: TextStyle(fontSize: 18),
                    ),
                  ),
                  SizedBox(height: 20),
                  TextButton(
                    onPressed: () {
                      setState(() {
                        _isLogin = !_isLogin;
                      });
                    },
                    child: Text(_isLogin
                        ? 'Don\'t have an account? Register'
                        : 'Already have an account? Login'),
                    style: TextButton.styleFrom(
                      foregroundColor: Colors.pink[300],
                    ),
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
  Widget _buildTextField(TextEditingController controller, String label,
IconData icon, {bool obscureText = false, TextInputType keyboardType =
TextInputType.text}) {
```

```
    return TextField(
      controller: controller,
      obscureText: obscureText,
      keyboardType: keyboardType,
      decoration: InputDecoration(
        labelText: label,
        border: OutlineInputBorder(),
        prefixIcon: Icon(icon),
      ),
    );
  }
}
```





Period_Tracking.dart

```
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

class PeriodCard extends StatefulWidget {
 final Function(DateTime) onDateSelected;  // Callback to pass the
selected date

 PeriodCard({required this.onDateSelected});

 @override
 _PeriodCardState createState() => _PeriodCardState();
}
```

```dart
class _PeriodCardState extends State<PeriodCard> {
 DateTime? _selectedDate;
 String? _formattedDate;
 final TextEditingController _nameController = TextEditingController();

 void _selectDate(BuildContext context) async {
   final DateTime? picked = await showDatePicker(
     context: context,
     initialDate: DateTime.now(),
     firstDate: DateTime(2000),
     lastDate: DateTime(2101),
   );
   if (picked != null && picked != _selectedDate) {
     setState(() {
       _selectedDate = picked;
       _formattedDate = DateFormat('yyyy-MM-dd').format(_selectedDate!);
     });
   }
 }
 void _showInputPopup(BuildContext context) {
   showDialog(
     context: context,
     builder: (BuildContext context) {
       return AlertDialog(
         title: Text('Enter Details'),
         content: Column(
           mainAxisSize: MainAxisSize.min,
           children: [
             TextField(
               controller: _nameController,
               decoration: InputDecoration(labelText: 'Your Name'),
             ),
             SizedBox(height: 10),
             ElevatedButton(
               onPressed: () => _selectDate(context),
               child: Text(_formattedDate ?? 'Select Last Period Date'),
             ),
           ],
         ),
         actions: [
           TextButton(
             onPressed: () {
               Navigator.of(context).pop();
             },
             child: Text('Cancel'),
           ),
```

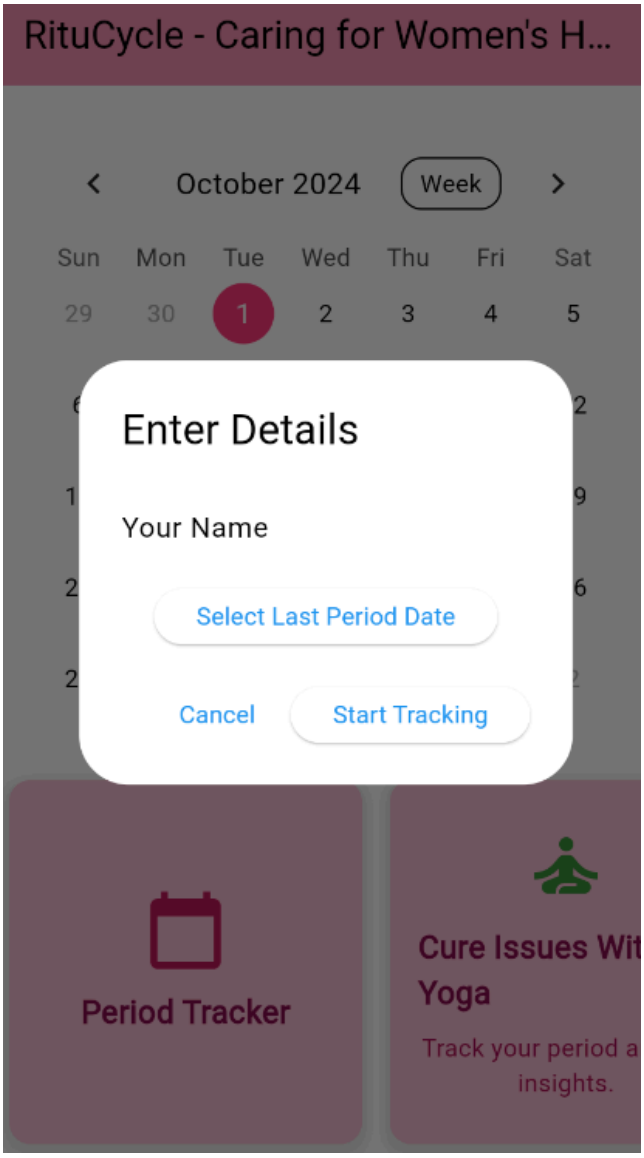
```
          ElevatedButton(
            onPressed: () {
              if (_nameController.text.isNotEmpty && _selectedDate !=
null) {
                widget.onDateSelected(_selectedDate!);  // Pass the date
back to HomeScreen
                Navigator.of(context).pop();
              }
            },
            child: Text('Start Tracking'),
          ),
        ],
      );
    },
  );
 }

 @override
 Widget build(BuildContext context) {
   return GestureDetector(
     onTap: () {
       _showInputPopup(context);
     },
     child: Container(
       width: 200,
       margin: EdgeInsets.all(8.0),
       padding: EdgeInsets.all(16.0),
       decoration: BoxDecoration(
         color: Colors.pink[100],
         borderRadius: BorderRadius.circular(12),
         boxShadow: [
           BoxShadow(
             color: Colors.grey.withOpacity(0.3),
             spreadRadius: 3,
             blurRadius: 5,
             offset: Offset(0, 3),
           ),
         ],
       ),
       child: Column(
         mainAxisAlignment: MainAxisAlignment.center,
         children: [
           Icon(Icons.calendar_today, size: 48, color: Colors.pink[700]),
           SizedBox(height: 10),
           Text(
             'Period Tracker',
             style: TextStyle(
```

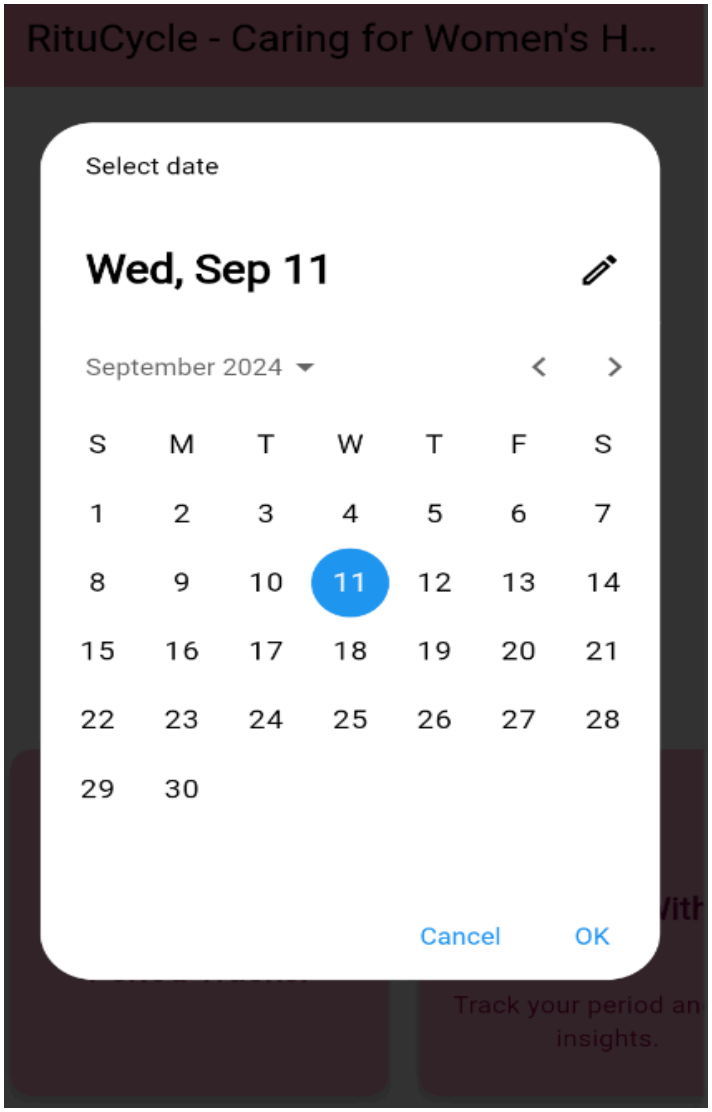
```
                fontSize: 18,
                fontWeight: FontWeight.bold,
                color: Colors.pink[900],
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

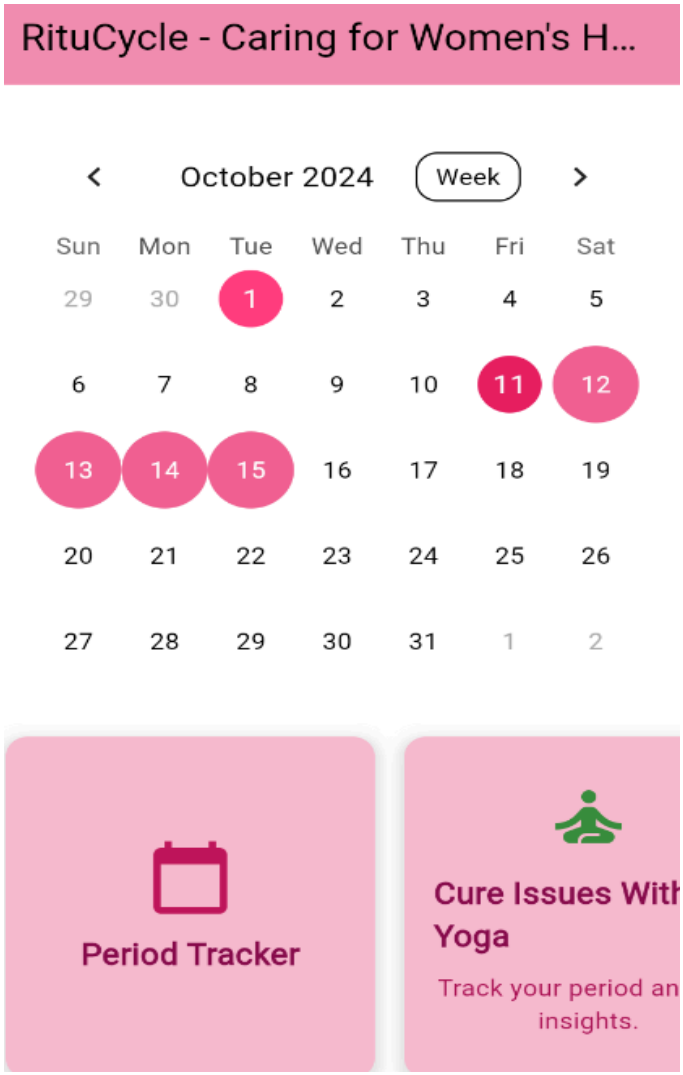
```
import 'package:flutter/material.dart';
import 'package:table_calendar/table_calendar.dart';

class CalendarWidget extends StatefulWidget {
 final List<DateTime> highlightedDates; // Add highlighted dates parameter

 CalendarWidget({required this.highlightedDates});

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

class _CalendarWidgetState extends State<CalendarWidget> {
 CalendarFormat _calendarFormat = CalendarFormat.month;
 DateTime _selectedDay = DateTime.now();
 DateTime _focusedDay = DateTime.now();

 @override
 Widget build(BuildContext context) {
   return Padding(
     padding: const EdgeInsets.all(16.0),
     child: TableCalendar(
       firstDay: DateTime.utc(2010, 10, 16),
       lastDay: DateTime.utc(2030, 3, 14),
       focusedDay: _focusedDay,
       selectedDayPredicate: (day) => isSameDay(_selectedDay, day),
       calendarFormat: _calendarFormat,
       availableCalendarFormats: const {
         CalendarFormat.month: 'Month',
         CalendarFormat.week: 'Week',
       },
       onDaySelected: (selectedDay, focusedDay) {
         setState(() {
           _selectedDay = selectedDay;
```

```
            _focusedDay = focusedDay;
          });
        },
        onFormatChanged: (format) {
          setState(() {
            _calendarFormat = format;
          });
        },
        headerStyle: HeaderStyle(
          formatButtonVisible: true,
          titleCentered: true,
        ),
        calendarStyle: CalendarStyle(
          todayDecoration: BoxDecoration(
            color: Colors.pinkAccent,
            shape: BoxShape.circle,
          ),
          selectedDecoration: BoxDecoration(
            color: Colors.pink,
            shape: BoxShape.circle,
          ),
        ),
        calendarBuilders: CalendarBuilders(
          defaultBuilder: (context, day, focusedDay) {
            final isHighlighted =
widget.highlightedDates.any((highlightedDate) =>
            highlightedDate.year == day.year &&
                highlightedDate.month == day.month &&
                highlightedDate.day == day.day);
            return Container(
              decoration: BoxDecoration(
                color: isHighlighted ? Colors.pink[300] : null,
                shape: BoxShape.circle,
              ),
              child: Center(
                child: Text(
                  '${day.day}',
                  style: TextStyle(
                    color: isHighlighted ? Colors.white : null,
                  ),
                ),
              ),
            );
          },
        ),
      ),
    );
```

```
  }
}
```


RituCycle - Caring for Women's H...
October 2024
Week
Sun
Mon
Tue
Wed
Thu
Fri
Sat
29
30
1
2
3
4
5
Enter Details
Your Name
Select Last Period Date
Cancel
Start Tracking
Period Tracker
Cure Issues Wit
Yoga
Track your period a
insights.


RituCycle - Caring for Women's H...
Select date
Wed, Sep 11
September 2024
S M T W T F S
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
Cancel
OK
Track your period an
insights.


RituCycle - Caring for Women's H...
October 2024
Week
Sun Mon Tue Wed Thu Fri Sat
29 30 1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31 1 2
Period Tracker
Cure Issues With
Yoga
Track your period an
insights.