Sakka Mohamad-Mario 1231B
Al-Khalidy Essam 1231B

DEPRESSION SYMPTOMS DETECTION

DESCRIPTION

1. REAL-TIME CHAT

- Users can engage in conversations with an AI assistant, which is designed to:
  - Analyze user inputs.
  - Identify symptoms of depression based on their responses.
  - Provide empathetic and engaging responses tailored to the user's current and past conversations.

2. GENERAL MOOD AND SYMPTOM TRACKING

- **General Mood Score**:
  - Computed based on weighted analysis of depression symptoms, ranging from `0%` (no signs of depression) to `100%` (high likelihood of depression).

- **Symptom Scores**:
  - Individual symptoms such as anxiety, sadness, irritability, etc., are scored per conversation session.
  - These scores are saved and visualized to track trends over time.

3. DATA VISUALIZATION

- **General Mood Graph**:
  - Displays the trend of the user's general mood scores across multiple sessions.

- **Symptom Graphs**:
  - Plots each symptom's score over time, allowing users to identify patterns and improvements in specific areas.

- **Interactive Visualization**:
  - Graphs include tooltips, markers, and labels for better interpretability.

4. PRIVACY

- **Federated**:
  - Data is saved locally, instead of being stored on a remote server

- **Anonymity**:
  - It is not associated to any user, name or real person.

Sakka Mohamad-Mario 1231B
Al-Khalidy Essam 1231B

TECHNICAL ARCHITECTURE

DATA MANAGER & PROCESSOR

- **Built with Node.js**:
  - Handles real-time communication using **Socket.IO (TCP)**.
  - Processes data storage and retrieval using **file-based JSON local storage**.
- **Key Functionalities**:
  1. **Start Chat**:
     - Generates initial questions based on past conversations or starts a generic conversation if no history exists.
  2. **Chat Analysis**:
     - Analyzes user responses using ChatGPT and extracts scores for depression symptoms and general mood.
     - Summarizes each conversation and updates the summary of all past interactions at the end of each conversation.
  3. **Data Storage**:
     - Stores user conversations, scores, and summaries in a structured JSON format.

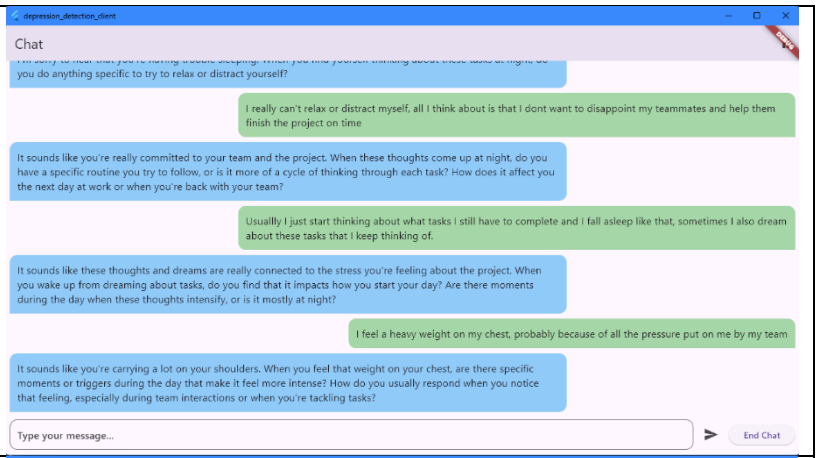
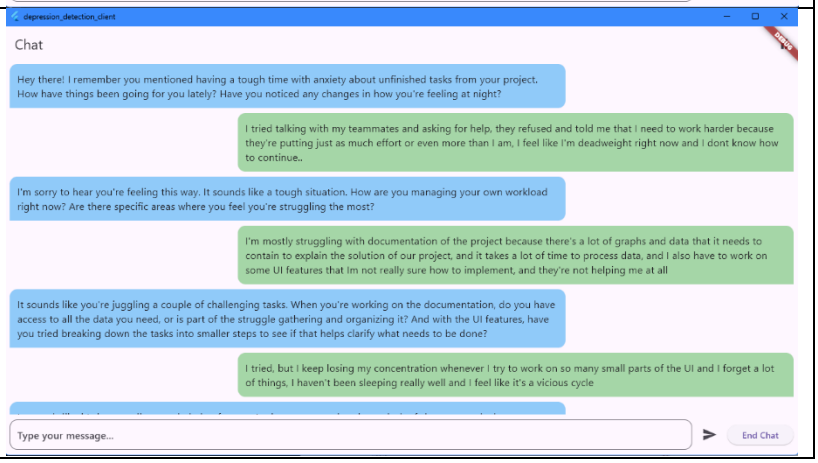
UI (FLUTTER)

- **Built with Flutter**:
  - Provides a seamless and interactive user interface.
  - Handles chat interaction, and data visualization.
- **Key Features**:
  1. **Chat Interface**:
     - Real-time chat bubbles with text wrapping and auto scrolling.
     - Supports starting and ending conversations dynamically.
  2. **Data Visualization**:
     - Fetches data on depression symptoms from Node client and the general mood scores across conversations (0% to 100%), higher scores indicate higher probability of depression.
     - Uses Syncfusion charts to display graphs for symptoms and general mood.

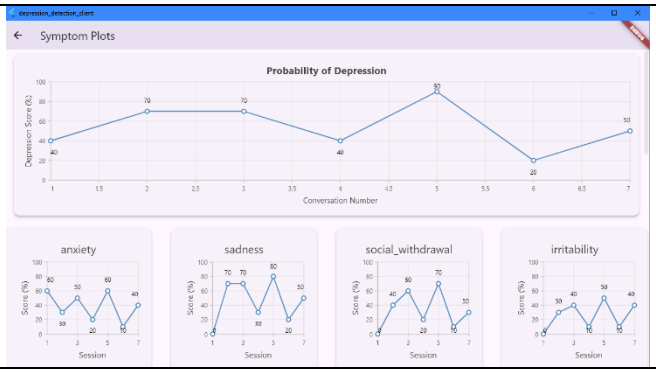
Sakka Mohamad-Mario 1231B
Al-Khalidy Essam 1231B

EXAMPLES (SCREENSHOTS)

1. CHATS

Starting a chat	
Picking up the conversation after some time	

2. VISUALIZATION OF DATA

First Set	

Sakka Mohamad-Mario 1231B
Al-Khalidy Essam 1231B

Second Set	
Third Set	

3. DATA STORAGE

appData.json × server.js

data > appData.json > [ ] conversations


```
2     "conversations": [
3       {
4         "chat": [
21        {
23          "content": "Usuallly I just start thinking about what tasks I still have to complete and I fall asleep like that, sometimes I also dream about these tasks that I keep thinking of
24        },
25        {
26          "role": "assistant",
27          "content": "It sounds like these thoughts and dreams are really connected to the stress you're feeling about the project. When you wake up from dreaming about tasks, do you find
28        },
29        {
30          "role": "user",
31          "content": "I feel a heavy weight on my chest, probably because of all the pressure put on me by my team"
32        },
33        {
34          "role": "assistant",
35          "content": "It sounds like you're carrying a lot on your shoulders. When you feel that weight on your chest, are there specific moments or triggers during the day that make it fe
36        }
37      ],
38      "finalScores": {
39        "anxiety": 60,
40        "sadness": 0,
41        "social_withdrawal": 0,
42        "irritability": 0,
43        "sleep_disturbance": 70,
44        "appetite_disturbance": 0,
45        "reckless_behavior": 0,
46        "reduced_productivity": 0,
47        "loss_of_interest": 0,
48        "physical_complaints": 0,
49        "memory_issues": 0,
50        "fear_of_separation": 0,
51        "sensitivity_to_rejection": 0,
52        "physical_heaviness": 50,
53        "concentration_difficulty": 0,
54        "fatigue": 0,
55        "worthlessness": 0,
56        "suicidal_ideation": 0
```

Sakka Mohamad-Mario 1231B
Al-Khalidy Essam 1231B

4. CODE SAMPLES

UI Flutter

chat_screen.dart

```
}

void _initializeSocket() {
  final String socketUrl = dotenv.env['SOCKET_URL'] ?? 'http://localh
  _socket = IO.io(
    socketUrl,
    IO.OptionBuilder().setTransports(['websocket']).build(),
  );

  _socket.onConnect((_) {
    print('Connected to server');
  });

  _socket.on('message', (data) {
    setState(() {
      _messages.add({'role': data['role'], 'content': data['content']
      _scrollToBottom();
    });
  });

  _socket.onDisconnect((_) {
    print('Disconnected from server');
  });
}

@override
void dispose() {
  _socket.dispose();
  _messageController.dispose();
  _scrollController.dispose();
  super.dispose();
}
```

chat_screen.dart

```
Widget _buildChatBubble(Map<String, String> message, bool isUser) {
  return Align(
    alignment: isUser ? Alignment.centerRight : Alignment.centerLeft,
    child: Container(
      constraints: BoxConstraints(
        maxWidth: MediaQuery.of(context).size.width * 0.7, // Limit bubble width
      ),  // BoxConstraints
      margin: const EdgeInsets.symmetric(
        vertical: 4.0,
        horizontal: 8.0,
      ),  // EdgeInsets.symmetric
      padding: const EdgeInsets.all(12.0),
      decoration: BoxDecoration(
        color: isUser ? Colors.green[300] : Colors.blue[200],
        borderRadius: BorderRadius.only(
          topLeft: const Radius.circular(12.0),
          topRight: const Radius.circular(12.0),
          bottomLeft: isUser ? const Radius.circular(12.0) : const Radius.circular(0),
          bottomRight: isUser ? const Radius.circular(0) : const Radius.circular(12.0),
        ),  // BorderRadius.only
      ),  // BoxDecoration
      child: Text(
        message['content'] ?? '',
        style: const TextStyle(fontSize: 16.0),
        softWrap: true, // Allow text to wrap
        overflow: TextOverflow.visible, // Ensure long text doesn't truncate
      ),  // Text
    ),  // Container
  );  // Align
}
```

flutter_ui · main

chat_screen.dart · plot_screen.dart

```
Future<void> _fetchSymptomScores() async {
  try {
    final String apiUrl = dotenv.env['API_URL'] ?? 'http://localhost:3000';
    final response = await http.get(Uri.parse('$apiUrl/fetch-scores'));

    if (response.statusCode == 200) {
      final List<dynamic> data = json.decode(response.body);
      final Map<String, List<double>> symptomData = {};
      final List<double> generalMoodScores = []; // List for mood scores

      for (var conversation in data) {
        // Update symptom scores
        conversation['finalScores'].forEach((symptom, score) {
          symptomData.putIfAbsent(symptom, () => []).add(score.toDouble());
        });

        // Add general mood score
        if (conversation['generalMoodScore'] != null) {
          generalMoodScores.add( conversation['generalMoodScore'].toDouble());
        }
      }

      setState(() {
        _symptomData = symptomData;
        _generalMoodScores = generalMoodScores;
        _isLoading = false;
      });
    } else {
      setState(() {
        _isLoading = false;
      });
```

chat_screen.dart · plot_screen.dart

```
Widget _buildSymptomGraph(String symptom, List<double> data) {
  return Container(
    height: 40, // Reduced height
    width: 40,
    padding: const EdgeInsets.all(8.0),
    child: SfCartesianChart(
      title: ChartTitle(
          text: symptom, textStyle: const TextStyle(fontSize: 18)),  // ChartTitle
      primaryXAxis: NumericAxis(
        title: AxisTitle(
            text: 'Session', textStyle: const TextStyle(fontSize: 16)),  // AxisTitle
        minimum: 1,

      ),  // NumericAxis
      primaryYAxis: NumericAxis(
        title: AxisTitle(
            text: 'Score (%)', textStyle: const TextStyle(fontSize: 16)),  // AxisTitle
        maximum: 100,
      ),  // NumericAxis
      tooltipBehavior: TooltipBehavior(enable: true),
      series: <LineSeries<double, int>>[
        LineSeries<double, int>(
          dataSource: data,
          xValueMapper: (value, index) => index + 1,
          yValueMapper: (value, _) => value,
          markerSettings: const MarkerSettings(isVisible: true),
          name: symptom,
          dataLabelSettings: const DataLabelSettings(isVisible: true),
        ),  // LineSeries
      ],  // <LineSeries<double, int>>[]
    ),  // SfCartesianChart
  );  // Container
```

Node Client

server.js

server.js > io.on("connection") callback > socket.on("message") callb

```
app.get("/fetch-scores", (req, res) => {
  }
});

// Socket.IO for real-time chat
io.on("connection", (socket) => {
  console.log("User connected via WebSocket");

  socket.on("message", async (data) => {
    const action = data.action;

    switch (action) {
      case "start":
        await handleStart(socket);
        break;
      case "chat":
        await handleChat(socket, data.message);
        break;
      case "end":
        await handleEnd(socket);
        break;
      default:
        socket.emit("error", "Invalid action.");
    }        You, 15 hours ago • Initial commit
  });

  socket.on("disconnect", () => {
    console.log("User disconnected from WebSocket");
  });
});
```

server.js · schemas.js

schemas.js > default_final_scores

```
export const scoresSchema = {
};

export const default_final_scores = {
  anxiety: 0,
  sadness: 0,
  social_withdrawal: 0,
  irritability: 0,
  sleep_disturbance: 0,
  appetite_disturbance: 0,
  reckless_behavior: 0,
  reduced_productivity: 0,
  loss_of_interest: 0,        You, 15 hours ago • Initial
  physical_complaints: 0,
  memory_issues: 0,
  fear_of_separation: 0,
  sensitivity_to_rejection: 0,
  physical_heaviness: 0,
  concentration_difficulty: 0,
  fatigue: 0,
  worthlessness: 0,
  suicidal_ideation: 0,
};

export const summarySchema = {
  name: "conversation_summary_schema",
  schema: {
    type: "object",
    properties: {
      summary: {
        type: "string",
        description: "Summary of past conversations.",
      },
    },
    required: ["summary"],
  },
};
```

Sakka Mohamad-Mario 1231B
Al-Khalidy Essam 1231B

JS server.js

JS server.js > handleChat > response

```javascript
// Function: Handle Start Chat
async function handleStart(socket) {
  try {
    const data = readJsonFile(dataFilePath, {
      username: "",
      conversations: [],
      summary_of_past_conversations: "",
    });

    // Use the existing summary from appData.json
    const existingSummary = data.summary_of_past_conversations;

    // Determine the initial prompt
    const initialPrompt = existingSummary
      ? `Based on this summary of past conversations: "${existingSummary}", start a ne
        Be engaging and empathetic. Don't talk in the third person; use "you" or "you
      : `No prior summary is available. Start a new conversation by asking the user ab
        Be empathetic and engaging. Avoid referencing any past conversations.`;

    // Generate the initial response from ChatGPT
    const response = await chatWithGPT(initialPrompt);

    // Add a new conversation entry
    const newConversation = {
      chat: [],
      finalScores: {},
      generalMoodScore: 0,
    };
    data.conversations.push(newConversation);
    writeJsonFile(dataFilePath, data);

    // Send the assistant's first message
    socket.emit("message", { role: "assistant", content: response });
  } catch (error) {
    console.error("Error starting chat:", error.message);
    socket.emit("error", "Failed to start chat.");
  }
}

// Function: Handle Chat Message
async function handleChat(socket, userMessage) {
  try {
    const data = readJsonFile(dataFilePath, { conversations: [] });
    const currentConversation =
      data.conversations[data.conversations.length - 1];
```

JS server.js JS schemas.js

JS schemas.js > default_final_scores

You, 15 hours ago | 1 author (You)

```javascript
export const scoresSchema = {
  name: "symptom_scores_schema",
  schema: {
    type: "object",
    properties: {
      finalScores: {
        type: "object",
        description:
          "Scores for various symptoms, between 0 and 100!!!, it shows how much the user is experiencing the sy
        properties: {
          anxiety: { type: "number", minimum: 0, maximum: 100 },
          sadness: { type: "number", minimum: 0, maximum: 100 },
          social_withdrawal: { type: "number", minimum: 0, maximum: 100 },
          irritability: { type: "number", minimum: 0, maximum: 100 },
          sleep_disturbance: { type: "number", minimum: 0, maximum: 100 },
          appetite_disturbance: { type: "number", minimum: 0, maximum: 100 },
          reckless_behavior: { type: "number", minimum: 0, maximum: 100 },
          reduced_productivity: { type: "number", minimum: 0, maximum: 100 },
          loss_of_interest: { type: "number", minimum: 0, maximum: 100 },
          physical_complaints: { type: "number", minimum: 0, maximum: 100 },
          memory_issues: { type: "number", minimum: 0, maximum: 100 },
          fear_of_separation: { type: "number", minimum: 0, maximum: 100 },
          sensitivity_to_rejection: {
            type: "number",
            minimum: 0,
            maximum: 100,
          },
          physical_heaviness: { type: "number", minimum: 0, maximum: 100 },
          concentration_difficulty: {
            type: "number",
            minimum: 0,
            maximum: 100,
          },
          fatigue: { type: "number", minimum: 0, maximum: 100 },
          worthlessness: { type: "number", minimum: 0, maximum: 100 },
          suicidal_ideation: { type: "number", minimum: 0, maximum: 100 },
        },
        required: [
```