package.json

```json
{
  "name": "wellness-backend",
  "version": "1.0.0",
  "description": "Backend API for Wellness Questionnaire",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.0.0",
    "express": "^4.18.2",
    "mongoose": "^7.0.0",
    "morgan": "^1.10.0",
    "express-validator": "^6.15.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.22"
  }
}
```

.env.example

```
PORT=4000
MONGO_URI=mongodb://localhost:27017/wellness_db
```

server.js

```javascript
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
const morgan = require('morgan');
const cors = require('cors');

const questionnaireRoutes = require('./routes/questionnaire');

const app = express();
app.use(express.json());
app.use(cors());
app.use(morgan('dev'));
```

```
// Routes
app.use('/api/questionnaire', questionnaireRoutes);

// Health
app.get('/api/health', (req, res) => res.json({ ok: true, ts: new Date().toISOString() }));

const PORT = process.env.PORT || 4000;
const MONGO_URI = process.env.MONGO_URI;

mongoose
  .connect(MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => {
    console.log('Connected to MongoDB');
    app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
  })
  .catch((err) => {
    console.error('MongoDB connection error:', err);
    process.exit(1);
  });
```

models/Response.js

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

const ResponseSchema = new mongoose.Schema({
  stage: { type: String, required: true },
  region: { type: String, required: true },
  sleepHours: { type: Number, required: true, default: 0 },
  appetite: { type: Boolean, required: true },
  mood: { type: Boolean, required: true },
  support: { type: Boolean, required: true }, // true = adequate support
  history: { type: Boolean, required: true },
  resultLabel: { type: String, required: true },
  score: { type: Number, required: true },
  createdAt: { type: Date, default: Date.now }
});

module.exports = mongoose.model('Response', ResponseSchema);
```

routes/questionnaire.js

```
const express = require('express');
const { body, validationResult } = require('express-validator');
const Response = require('../models/Response');

const router = express.Router();

router.post(
  '/submit',
  [
    body('stage').isString().notEmpty(),
    body('region').isString().notEmpty(),
    body('sleepHours').isNumeric(),
    body('appetite').isBoolean(),
    body('mood').isBoolean(),
    body('support').isBoolean(),
    body('history').isBoolean()
  ],
  async (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) return res.status(400).json({ errors: errors.array() });

    const { stage, region, sleepHours, appetite, mood, support, history } = req.body;

    // Scoring logic (server-side mirror of Flutter)
    // appetite true => +1, mood true => +1, lack of support => +1, history true => +1
    let score = 0;
    if (appetite === true) score++;
    if (mood === true) score++;
    if (support === false) score++; // lack of support increases risk
    if (history === true) score++;

    const resultLabel = score >= 2 ? 'Possible PPD Risk' : 'Low Risk';

    try {
      const resp = new Response({
        stage,
        region,
        sleepHours,
        appetite,
        mood,
        support,
        history,
        resultLabel,
        score
      });
      const saved = await resp.save();
      return res.status(201).json({ success: true, data: saved });
    } catch (err) {
      console.error('Save error', err);
      return res.status(500).json({ success: false, message: 'Server error' });
    }
  }
);

/**
 * GET /api/questionnaire/history
 * Optional query params: ?limit=20
 */
router.get('/history', async (req, res) => {
  const limit = Math.min(parseInt(req.query.limit) || 50, 500);
  try {
    const items = await Response.find().sort({ createdAt: -1 }).limit(limit);
    return res.json({ success: true, count: items.length, data: items });
  } catch (err) {
    console.error(err);
    return res.status(500).json({ success: false, message: 'Server error' });
  }
});

/**
 * GET /api/questionnaire/:id
 */
router.get('/:id', async (req, res) => {
  try {
    const item = await Response.findById(req.params.id);
    if (!item) return res.status(404).json({ success: false, message: 'Not found' });
    return res.json({ success: true, data: item });
  } catch (err) {
    console.error(err);
    return res.status(500).json({ success: false, message: 'Server error' });
  }
});

module.exports = router;
```