Project Structure

```
wellness-tracker-backend/
├── controllers/
│   └── activityController.js
├── models/
│   └── ActivityLog.js
├── redisClient.js
├── routes/
│   └── activityRoutes.js
├── server.js
├── .env
├── package.json
└── README.md
```

Server Setup (server.js)

```
// server.js
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
require('dotenv').config();

const activityRoutes = require('./routes/activity');
const deviceSyncService = require('./services/deviceSyncService');

const app = express();
const PORT = process.env.PORT || 3000;

// Middleware
app.use(bodyParser.json());

// Routes
app.use('/activity', activityRoutes);

// Sync device data (can be scheduled or triggered manually)
app.post('/sync-device-data', async (req, res) => {
 try {
   await deviceSyncService.syncDeviceDataFromAPI();
   res.status(200).json({ message: 'Device data synced successfully!' });
 } catch (error) {
   res.status(500).json({ error: 'Failed to sync device data' });
 }
});

// Connect to MongoDB
mongoose.connect(process.env.MONGODB_URI, {
 useNewUrlParser: true,
 useUnifiedTopology: true,
})
 .then(() => console.log('MongoDB connected!'))
 .catch(err => console.log(err));

// Start the server
app.listen(PORT, () => {
 console.log(`Server running on port ${PORT}`);
});
```

MongoDB Model (models/ActivityLog.js)

```
// models/ActivityLog.js
const mongoose = require('mongoose');

const activityLogSchema = new mongoose.Schema({
 user_id: { type: String, required: true },
 date: { type: String, required: true },
 hydration_liters: { type: Number, required: true },
 sleep_hours: { type: Number, required: true },
 exercise_minutes: { type: Number, required: true },
 meditation_minutes: { type: Number, required: true },
 source: { type: String, default: 'manual' }, // 'manual' or 'device'
});

module.exports = mongoose.model('ActivityLog', activityLogSchema);
```

Redis Client Setup (redisClient.js)

```
// redisClient.js
const { createClient } = require('@redis/client');

// Create Redis client
const client = createClient({
 url: 'redis://localhost:6379', // Use the correct URL for your Redis server
});

client.connect().catch((err) => {
 console.error('Redis connection error:', err);
});

client.on('connect', () => {
 console.log('Connected to Redis');
});

client.on('error', (err) => {
 console.error('Redis client error:', err);
});

module.exports = client;
```

Controller Implementation (Controllers/activityController.js)

```
// controllers/activityController.js
const redisClient = require('../redisClient'); // Redis client
const ActivityLog = require('../models/ActivityLog');

// POST /activity - Log wellness activity
const logActivity = async (req, res) => {
 const { user_id, date, hydration_liters, sleep_hours, exercise_minutes,
meditation_minutes, source } = req.body;

 try {
   const newActivityLog = new ActivityLog({
     user_id,
     date,
     hydration_liters,
     sleep_hours,
     exercise_minutes,
     meditation_minutes,
     source: source || 'manual',
   });

   await newActivityLog.save();
   res.status(201).json(newActivityLog);
 } catch (err) {
   console.error('Error logging activity:', err);
   res.status(500).json({ error: 'Error logging activity' });
 }
};

// GET /activity/:userId - Get activity logs for a user
const getActivityLogs = async (req, res) => {
 const { userId } = req.params;
 const { start, end } = req.query;

 // Construct the cache key based on userId and date range
 const cacheKey = `activity_logs:${userId}:${start}:${end}`;

 try {
   // 1. Check Redis for cached data
   const cachedData = await redisClient.get(cacheKey);
```

```
    if (cachedData) {
      // 2. If cache hit, return the cached data
      console.log('Cache hit');
      return res.status(200).json(JSON.parse(cachedData)); // Redis stores data as
string, so parse it
    }

    // 3. If no cache, query MongoDB
    console.log('Cache miss');
    const query = { user_id: userId };
    if (start && end) {
      query.date = { $gte: start, $lte: end }; // Filter by date range if provided
    }

    const activityLogs = await ActivityLog.find(query);

    if (!activityLogs.length) {
      return res.status(404).json({ message: 'No activity logs found for this user in
the specified range' });
    }

    // 4. Cache the data in Redis (with an expiration time)
    await redisClient.setEx(cacheKey, 3600, JSON.stringify(activityLogs)); // Cache for
1 hour

    // 5. Return the data from MongoDB
    return res.status(200).json(activityLogs);
  } catch (err) {
    console.error('Error retrieving activity logs:', err);
    res.status(500).json({ error: 'Error retrieving activity logs' });
  }
};

module.exports = { logActivity, getActivityLogs };
```

Routes (routes/activity.js)

```
// routes/activity.js
const express = require('express');
const router = express.Router();
const { logActivity, getActivityLogs } = require('../controllers/activityController');

// POST /activity - Log wellness activity
router.post('/', logActivity);

// GET /activity/:userId - Retrieve logs for a user by date range
router.get('/:userId', getActivityLogs);

module.exports = router;
```