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

const app = express();

const compression = require('compression');

const redis = require('ioredis');

const redisClient = new redis();



// Middleware

app.use(compression()); // Enable HTTP response compression

app.use(express.json()); // Middleware to parse incoming JSON requests



// Mock Data: Simulating database interaction
```

```
const users = [

{ id: 1, name: 'Alice' },

{ id: 2, name: 'Bob' }

];



// Controller: Get all users with caching

const getUsers = async (req, res, next) => {

try {

  // Check if users are already cached

  const cachedUsers = await redisClient.get('users');

  if (cachedUsers) {
    console.log('Cache hit');

    return res.status(200).json(JSON.parse(cachedUsers)); // Serve from cache

  }



  console.log('Cache miss');

  // Simulate database query

  const result = users;

  redisClient.setex('users', 3600, JSON.stringify(result)); // Cache for 1 hour

  res.status(200).json(result);

} catch (error) {

  next(error); // Pass errors to the global error handler
```

```
}

};



// Controller: Get a specific user by ID

const getUserById = async (req, res, next) => {

try {

  const user = users.find(u => u.id === parseInt(req.params.id));

  if (!user) {

    return res.status(404).json({ message: 'User not found' });

  }

  res.status(200).json(user);
```

```
} catch (error) {
  next(error); // Pass errors to the global error handler
}
};

// Routes
app.get('/users', getUsers);
app.get('/users/:id', getUserById);

// Global Error Handling Middleware
app.use((err, req, res, next) => {
```

```
console.error(err.stack);

res.status(500).json({ message: 'Something went wrong!', error: err.message });

});



// Start the server

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Server running on http://localhost:${PORT}`);

});
```