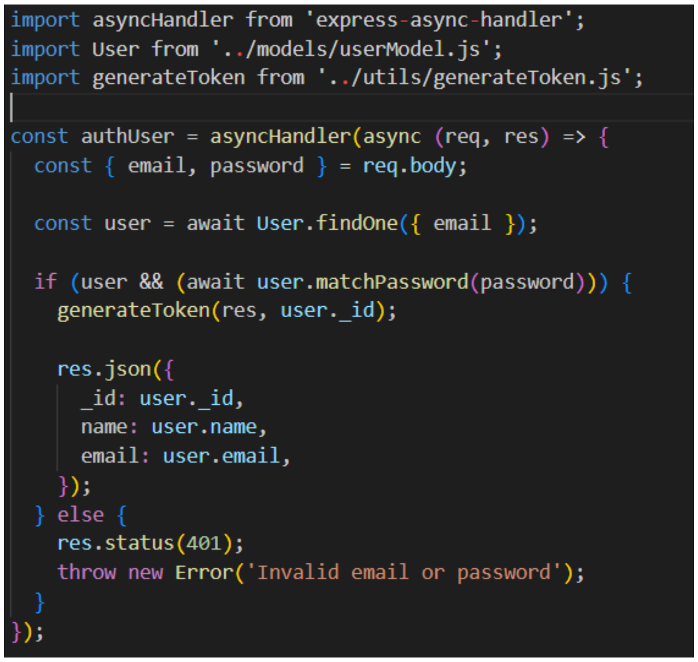
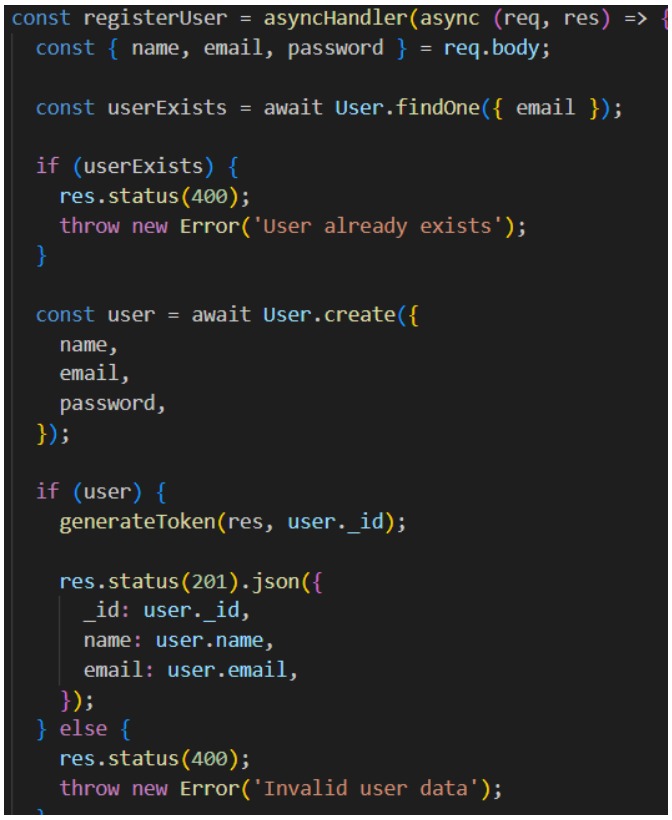
Taniya Nagdev **D15B / 50**

Experiment 5

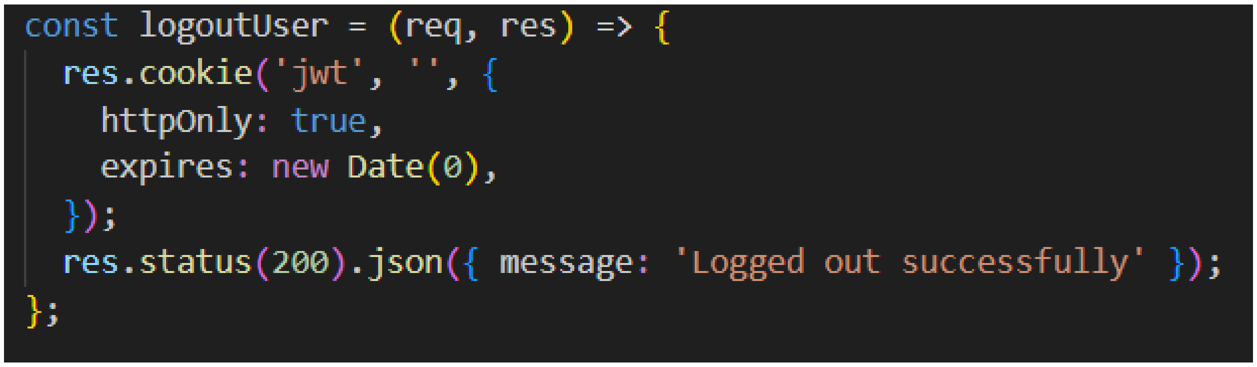
Source code:



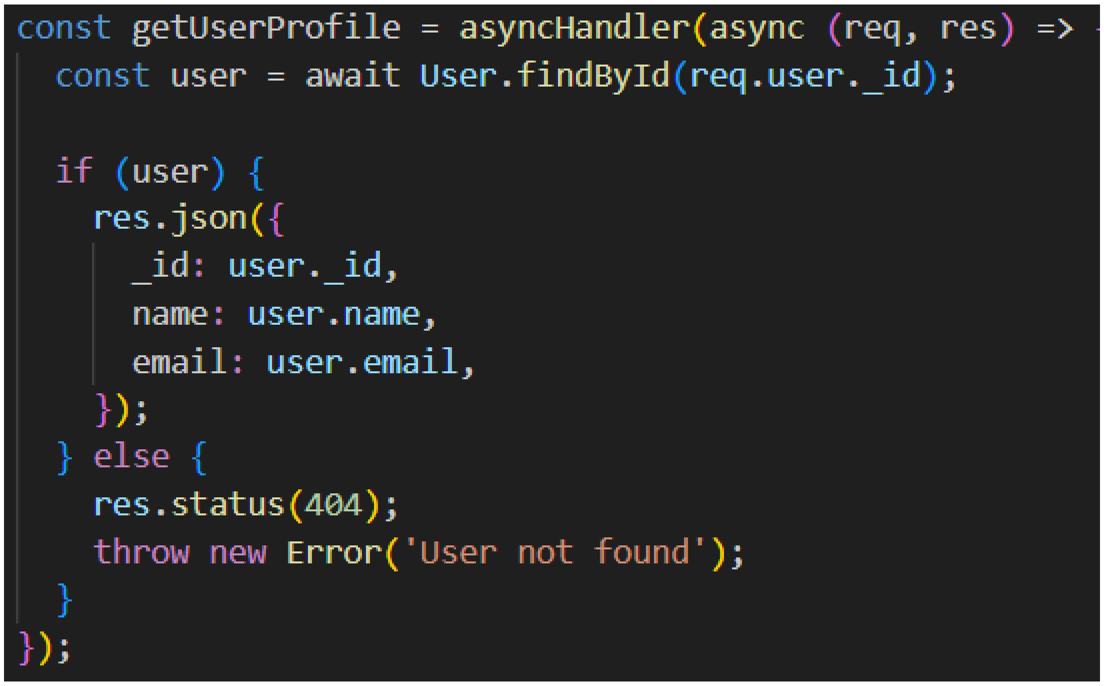
```
import asyncHandler from 'express-async-handler';
import User from '../models/userModel.js';
import generateToken from '../utils/generateToken.js';

const authUser = asyncHandler(async (req, res) => {
  const { email, password } = req.body;

  const user = await User.findOne({ email });

  if (user && (await user.matchPassword(password))) {
    generateToken(res, user._id);

    res.json({
      _id: user._id,
      name: user.name,
      email: user.email,
    });
  } else {
    res.status(401);
    throw new Error('Invalid email or password');
  }
});
```



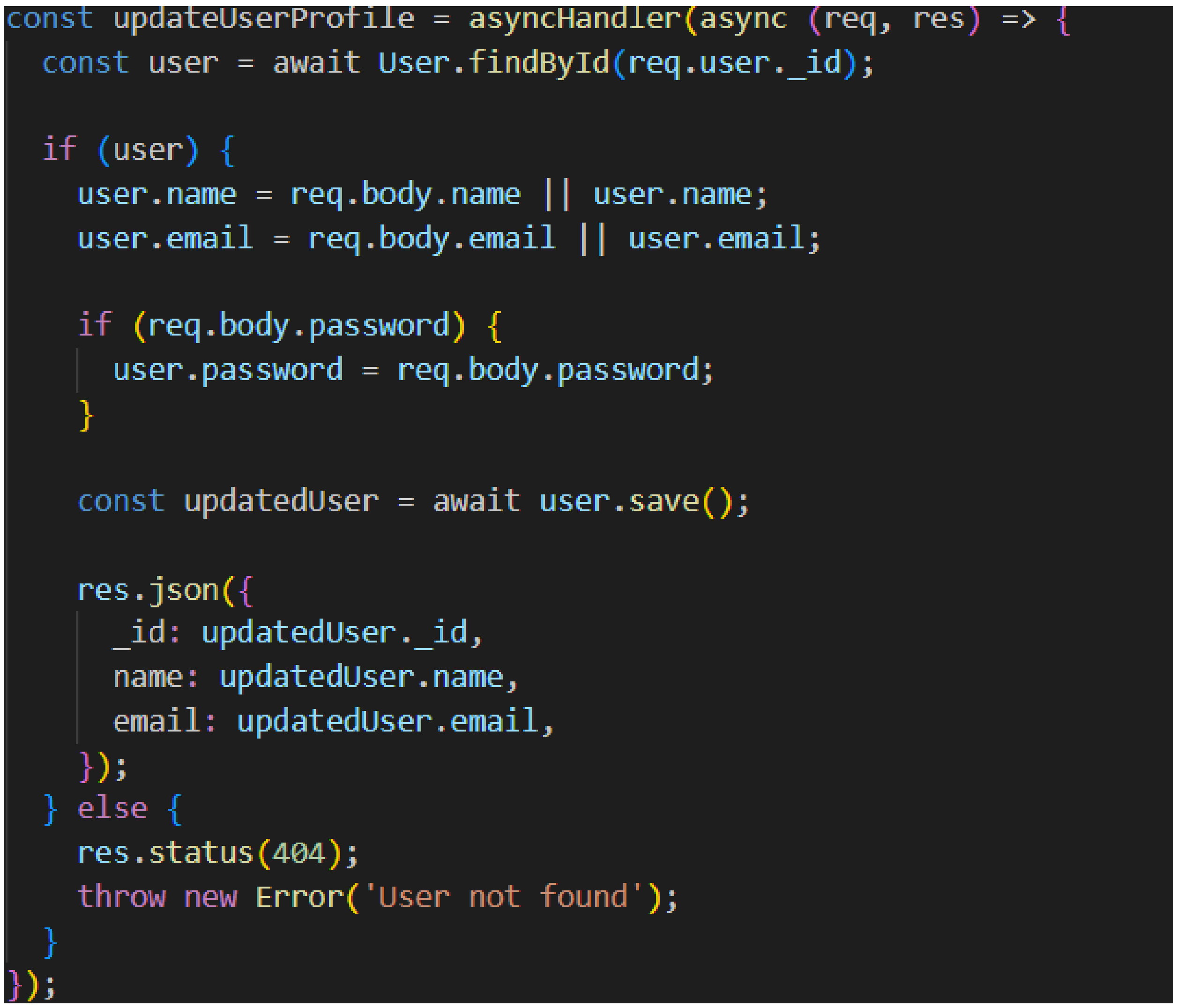
```
const registerUser = asyncHandler(async (req, res) => {
  const { name, email, password } = req.body;

  const userExists = await User.findOne({ email });

  if (userExists) {
    res.status(400);
    throw new Error('User already exists');
  }

  const user = await User.create({
    name,
    email,
    password,
  });

  if (user) {
    generateToken(res, user._id);

    res.status(201).json({
      _id: user._id,
      name: user.name,
      email: user.email,
    });
  } else {
    res.status(400);
    throw new Error('Invalid user data');
```



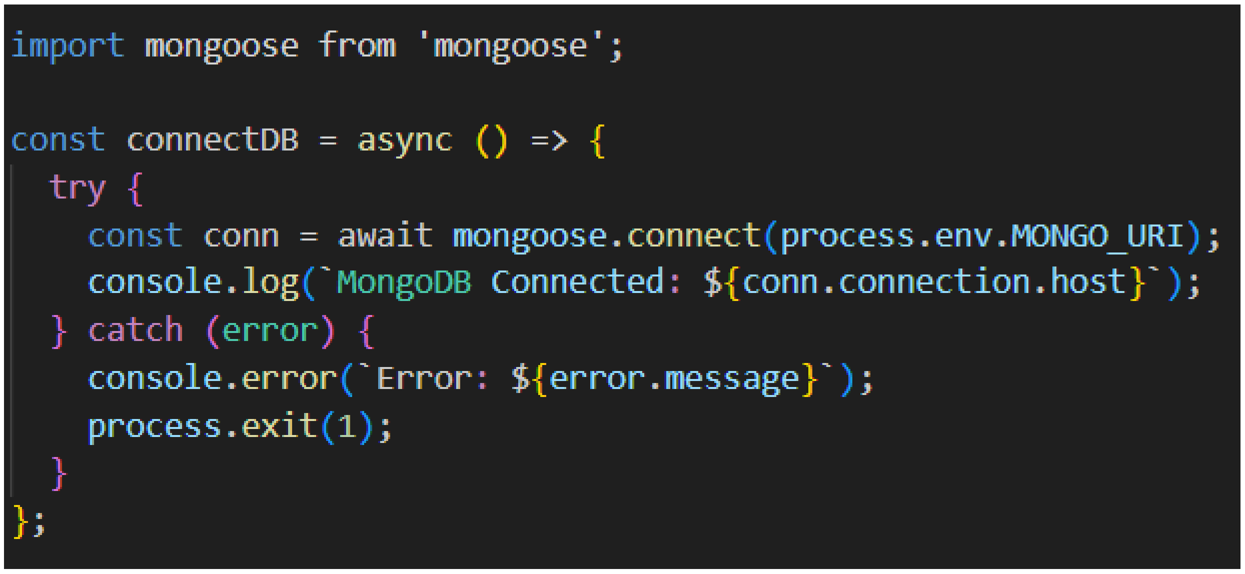
```
const logoutUser = (req, res) => {
  res.cookie('jwt', '', {
    httpOnly: true,
    expires: new Date(0),
  });
  res.status(200).json({ message: 'Logged out successfully' });
};
```



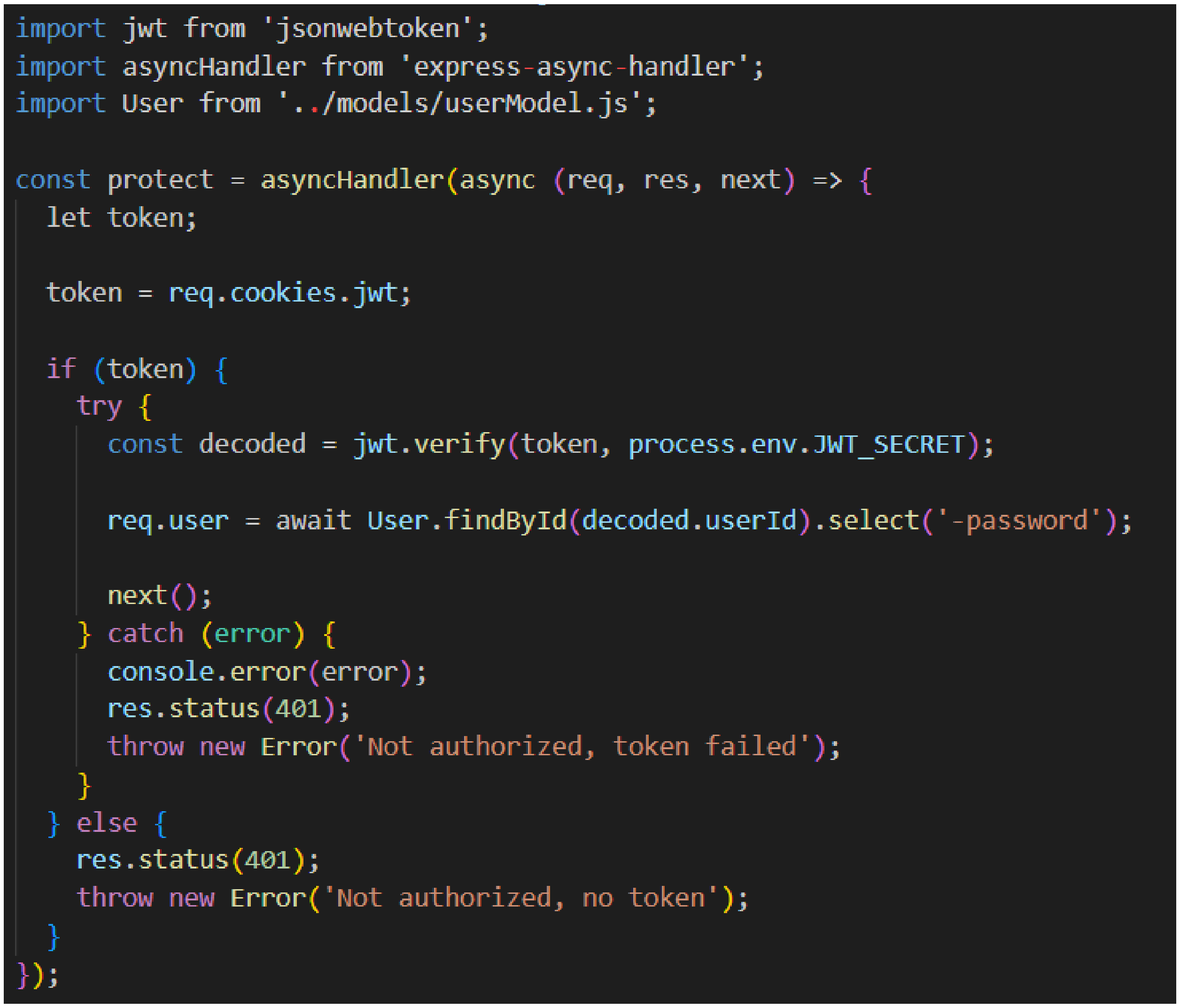
```
const getUserProfile = asyncHandler(async (req, res) =>
  const user = await User.findById(req.user._id);

  if (user) {
    res.json({
      _id: user._id,
      name: user.name,
      email: user.email,
    });
  } else {
    res.status(404);
    throw new Error('User not found');
  }
});
```



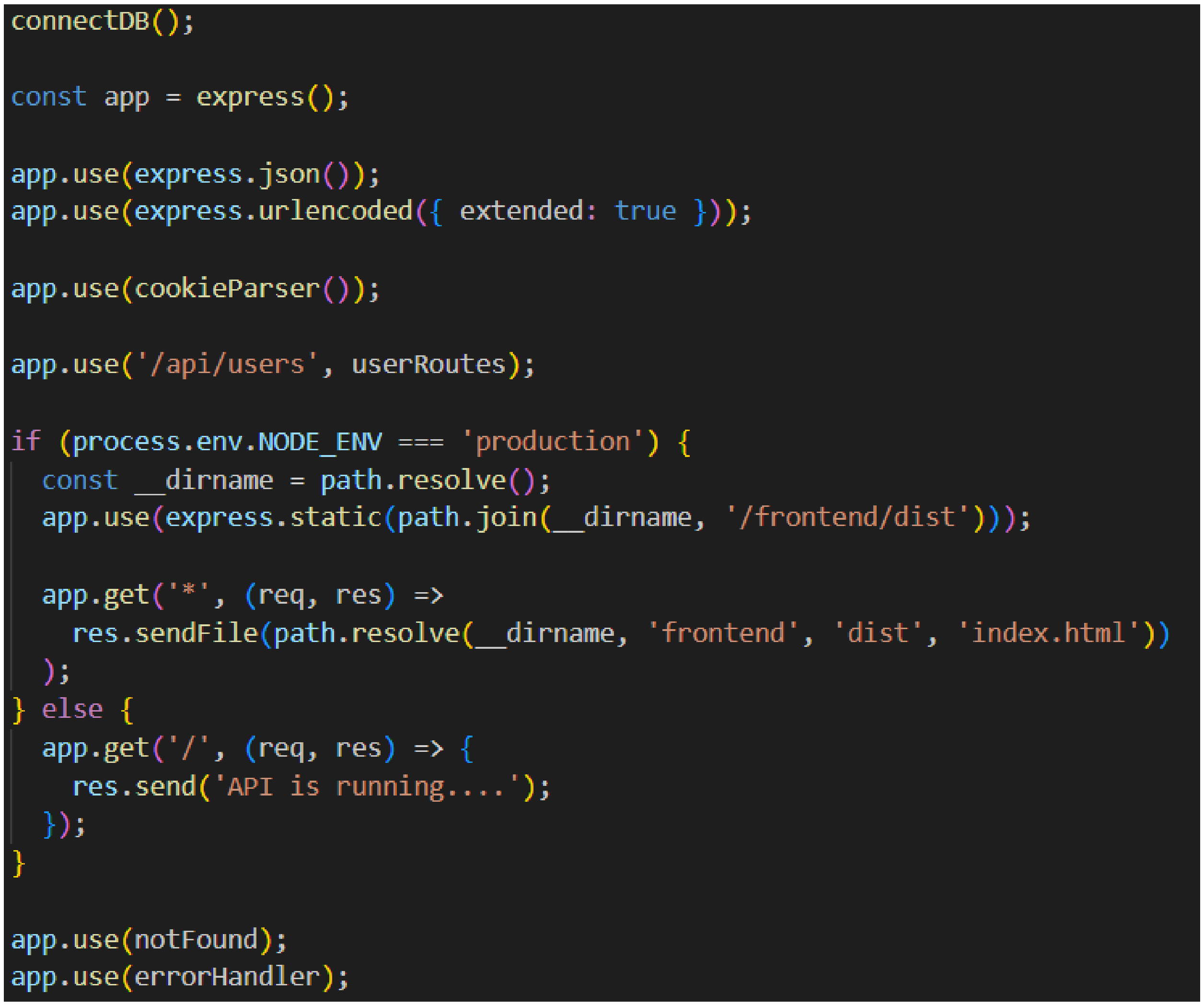
```
const updateUserProfile = asyncHandler(async (req, res) => {
  const user = await User.findById(req.user._id);

  if (user) {
    user.name = req.body.name || user.name;
    user.email = req.body.email || user.email;

    if (req.body.password) {
      user.password = req.body.password;
    }

    const updatedUser = await user.save();

    res.json({
      _id: updatedUser._id,
      name: updatedUser.name,
      email: updatedUser.email,
    });
  } else {
    res.status(404);
    throw new Error('User not found');
  }
});
```



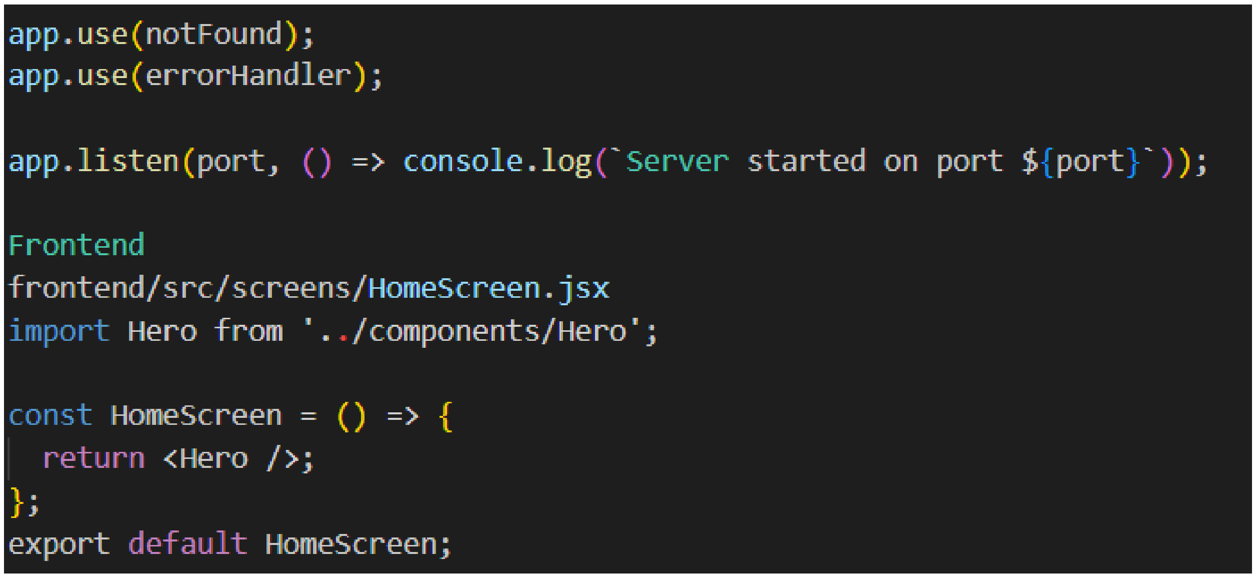
```
import mongoose from 'mongoose';

const connectDB = async () => {
  try {
    const conn = await mongoose.connect(process.env.MONGO_URI);
    console.log(`MongoDB Connected: ${conn.connection.host}`);
  } catch (error) {
    console.error(`Error: ${error.message}`);
    process.exit(1);
  }
};
```



```
import jwt from 'jsonwebtoken';
import asyncHandler from 'express-async-handler';
import User from '../models/userModel.js';

const protect = asyncHandler(async (req, res, next) => {
  let token;

  token = req.cookies.jwt;

  if (token) {
    try {
      const decoded = jwt.verify(token, process.env.JWT_SECRET);

      req.user = await User.findById(decoded.userId).select('-password');

      next();
    } catch (error) {
      console.error(error);
      res.status(401);
      throw new Error('Not authorized, token failed');
    }
  } else {
    res.status(401);
    throw new Error('Not authorized, no token');
  }
});
```

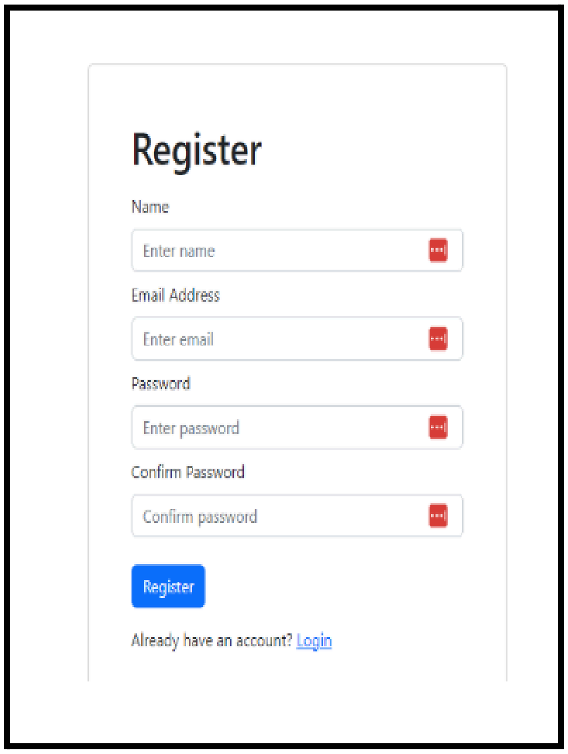


```
connectDB();

const app = express();

app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.use(cookieParser());

app.use('/api/users', userRoutes);

if (process.env.NODE_ENV === 'production') {
  const __dirname = path.resolve();
  app.use(express.static(path.join(__dirname, '/frontend/dist')));

  app.get('*', (req, res) =>
    res.sendFile(path.resolve(__dirname, 'frontend', 'dist', 'index.html'))
  );
} else {
  app.get('/', (req, res) => {
    res.send('API is running....');
  });
}

app.use(notFound);
app.use(errorHandler);
```



```
app.use(notFound);
app.use(errorHandler);

app.listen(port, () => console.log(`Server started on port ${port}`));

Frontend
frontend/src/screens/HomeScreen.jsx
import Hero from '../components/Hero';

const HomeScreen = () => {
  return <Hero />;
};
export default HomeScreen;
```

Output:



Register

Name

Enter name

Email Address

Enter email

Password

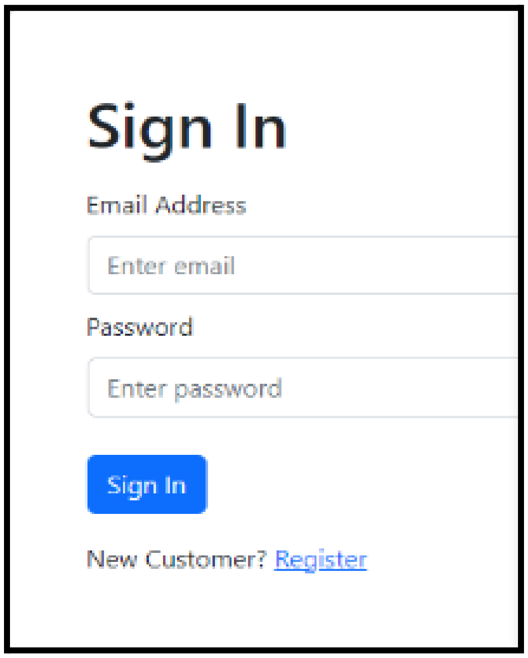
Enter password

Confirm Password

Confirm password

Register

Already have an account? Login



Sign In

Email Address

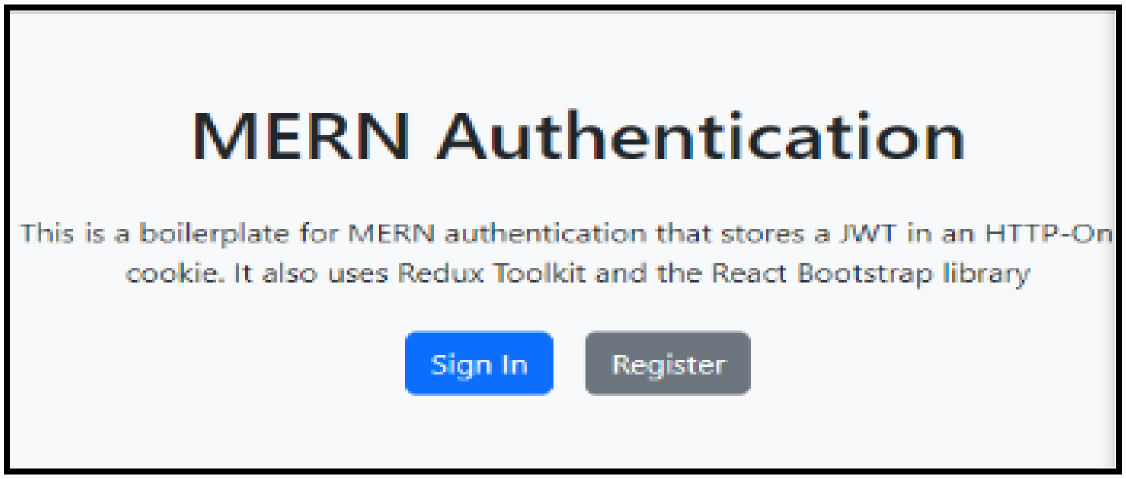
Enter email

Password

Enter password

Sign In

New Customer? Register



MERN Authentication

This is a boilerplate for MERN authentication that stores a JWT in an HTTP-On
cookie. It also uses Redux Toolkit and the React Bootstrap library