

Backend:**backend/controllers/userController.js**

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

// @desc   Auth user & get token
// @route  POST /api/users/auth
// @access Public
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');
  }
});

// @desc   Register a new user
// @route  POST /api/users
// @access Public
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');
  }
});

// @desc   Logout user / clear cookie
// @route  POST /api/users/logout
// @access Public
const logoutUser = (req, res) => {
  res.cookie('jwt', '', {
    httpOnly: true,
    expires: new Date(0),
  });
  res.status(200).json({ message: 'Logged out successfully' });
};

// @desc   Get user profile
// @route  GET /api/users/profile
// @access Private
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');
  }
});
```

```
});

// @desc   Update user profile
// @route   PUT /api/users/profile
// @access  Private
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');
  }
});

export {
  authUser,
  registerUser,
  logoutUser,
  getUserProfile,
  updateUserProfile,
};
```

backend/config/db.js

```
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);  
    }  
  };  
};
```

```
export default connectDB;
```

backend/middleware/authMiddleware.js

```
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');  
  }  
});  
  
export { protect };
```

backend/models/userModel.js

```
import mongoose from 'mongoose';  
import bcrypt from 'bcryptjs';  
  
const userSchema = mongoose.Schema(  
  {
```

```
name: {
  type: String,
  required: true,
},
email: {
  type: String,
  required: true,
  unique: true,
},
password: {
  type: String,
  required: true,
},
},
{
  timestamps: true,
}
);

// Match user entered password to hashed password in database
userSchema.methods.matchPassword = async function (enteredPassword) {
  return await bcrypt.compare(enteredPassword, this.password);
};

// Encrypt password using bcrypt
userSchema.pre('save', async function (next) {
  if (!this.isModified('password')) {
    next();
  }

  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
});

const User = mongoose.model('User', userSchema);

export default User;
```

backend/routes/userRoutes.js

```
import express from 'express';
import {
  authUser,
  registerUser,
  logoutUser,
  getUserProfile,
  updateUserProfile,
} from '../controllers/userController.js';
import { protect } from '../middleware/authMiddleware.js';
```

```
const router = express.Router();
```

```
router.post('/', registerUser);
router.post('/auth', authUser);
router.post('/logout', logoutUser);
router
  .route('/profile')
  .get(protect, getUserProfile)
  .put(protect, updateUserProfile);
```

```
export default router;
```

backend/server.js

```
import path from 'path';
import express from 'express';
import dotenv from 'dotenv';
dotenv.config();
import connectDB from './config/db.js';
import cookieParser from 'cookie-parser';
import { notFound, errorHandler } from './middleware/errorMiddleware.js';
import userRoutes from './routes/userRoutes.js';
```

```
const port = process.env.PORT || 5000;
```

```
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.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;
```

frontend/src/screens/LoginScreen.jsx

```
import { useState, useEffect } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { Form, Button, Row, Col } from 'react-bootstrap';
import FormContainer from '../components/FormContainer';
import { useDispatch, useSelector } from 'react-redux';
import { useLoginMutation } from '../slices/usersApiSlice';
import { setCredentials } from '../slices/authSlice';
import { toast } from 'react-toastify';
import Loader from '../components/Loader';
```

```
const LoginScreen = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
```

```
const dispatch = useDispatch();
const navigate = useNavigate();

const [login, { isLoading }] = useLoginMutation();

const { userInfo } = useSelector((state) => state.auth);

useEffect(() => {
  if (userInfo) {
    navigate('/');
  }
}, [navigate, userInfo]);

const submitHandler = async (e) => {
  e.preventDefault();
  try {
    const res = await login({ email, password }).unwrap();
    dispatch(setCredentials({ ...res }));
    navigate('/');
  } catch (err) {
    toast.error(err?.data?.message || err.error);
  }
};

return (
  <FormContainer>
    <h1>Sign In</h1>

    <Form onSubmit={submitHandler}>
      <Form.Group className='my-2' controlId='email'>
        <Form.Label>Email Address</Form.Label>
        <Form.Control
          type='email'
          placeholder='Enter email'
          value={email}
          onChange={(e) => setEmail(e.target.value)}
        ></Form.Control>
      </Form.Group>

      <Form.Group className='my-2' controlId='password'>
        <Form.Label>Password</Form.Label>
        <Form.Control
          type='password'
```



```
        placeholder='Enter password'
        value={password}
        onChange={(e) => setPassword(e.target.value)}
      ></Form.Control>
    </Form.Group>

    <Button
      disabled={isLoading}
      type='submit'
      variant='primary'
      className='mt-3'
    >
      Sign In
    </Button>
  </Form>

  {isLoading && <Loader />}

  <Row className='py-3'>
    <Col>
      New Customer? <Link to='/register'>Register</Link>
    </Col>
  </Row>
</FormContainer>
);
};

export default LoginScreen;
```

frontend/src/screens/RegisterScreen.jsx

```
import { useState, useEffect } from 'react';
import { Form, Button, Row, Col } from 'react-bootstrap';
import FormContainer from '../components/FormContainer';
import Loader from '../components/Loader';
// import { Link, useLocation, useNavigate } from 'react-router-dom';
import { Link, useNavigate } from 'react-router-dom';
import { useDispatch, useSelector } from 'react-redux';
import { useRegisterMutation } from '../slices/usersApiSlice';
import { setCredentials } from '../slices/authSlice';
import { toast } from 'react-toastify';

const RegisterScreen = () => {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
```

```
const [password, setPassword] = useState("");
const [confirmPassword, setConfirmPassword] = useState("");

const dispatch = useDispatch();
const navigate = useNavigate();

const [register, { isLoading }] = useRegisterMutation();

const { userInfo } = useSelector((state) => state.auth);

useEffect(() => {
  if (userInfo) {
    navigate('/');
  }
}, [navigate, userInfo]);

const submitHandler = async (e) => {
  e.preventDefault();

  if (password !== confirmPassword) {
    toast.error('Passwords do not match');
  } else {
    try {
      const res = await register({ name, email, password }).unwrap();
      dispatch(setCredentials({ ...res }));
      navigate('/');
    } catch (err) {
      toast.error(err?.data?.message || err.error);
    }
  }
};

return (
  <FormContainer>
    <h1>Register</h1>
    <Form onSubmit={submitHandler}>
      <Form.Group className='my-2' controlId='name'>
        <Form.Label>Name</Form.Label>
        <Form.Control
          type='name'
          placeholder='Enter name'
          value={name}
          onChange={(e) => setName(e.target.value)}
        ></Form.Control>
      </Form.Group>
    </FormContainer>
  )
);
```

```
<Form.Group className='my-2' controlId='email'>
  <Form.Label>Email Address</Form.Label>
  <Form.Control
    type='email'
    placeholder='Enter email'
    value={email}
    onChange={(e) => setEmail(e.target.value)}
  ></Form.Control>
</Form.Group>

<Form.Group className='my-2' controlId='password'>
  <Form.Label>Password</Form.Label>
  <Form.Control
    type='password'
    placeholder='Enter password'
    value={password}
    onChange={(e) => setPassword(e.target.value)}
  ></Form.Control>
</Form.Group>
<Form.Group className='my-2' controlId='confirmPassword'>
  <Form.Label>Confirm Password</Form.Label>
  <Form.Control
    type='password'
    placeholder='Confirm password'
    value={confirmPassword}
    onChange={(e) => setConfirmPassword(e.target.value)}
  ></Form.Control>
</Form.Group>

<Button type='submit' variant='primary' className='mt-3'>
  Register
</Button>

{isLoading && <Loader />}
</Form>

<Row className='py-3'>
  <Col>
    Already have an account? <Link to={` /login`} >Login</Link>
  </Col>
</Row>
</FormContainer>
);
```

```
};
```

```
export default RegisterScreen;
```

Screenshots: