Backend (Express.js with MongoDB)

1. Setup:

mkdir backend
cd backend
npm init -y
npm install express mongoose body-parser bcryptjs jsonwebtoken

2. Server Setup (server.js):

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

const app = express();

app.use(bodyParser.json());

// MongoDB Connection
mongoose.connect('mongodb://localhost:27017/myapp', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

const User = mongoose.model('User', {
  username: String,
  password: String,
});

const JWT_SECRET = 'your_secret_key';

// Middleware to authenticate user token
const authenticateToken = (req, res, next) => {
  const token = req.header('Authorization');
  if (!token) return res.status(401).json({ message: 'Access Denied' });

  jwt.verify(token, JWT_SECRET, (err, user) => {
    if (err) return res.status(403).json({ message: 'Invalid Token' });
    req.user = user;
    next();
```

```
  });
};

// User Registration
app.post('/register', async (req, res) => {
  try {
    const hashedPassword = await bcrypt.hash(req.body.password, 10);
    const user = new User({
      username: req.body.username,
      password: hashedPassword,
    });
    await user.save();
    res.status(201).send();
  } catch {
    res.status(500).send();
  }
});

// User Login
app.post('/login', async (req, res) => {
  const user = await User.findOne({ username: req.body.username });
  if (!user) return res.status(400).json({ message: 'User not found' });

  const validPassword = await bcrypt.compare(req.body.password, user.password);
  if (!validPassword) return res.status(400).json({ message: 'Invalid Password' });

  const token = jwt.sign({ username: user.username }, JWT_SECRET);
  res.json({ token });
});

// Protected Route
app.get('/protected', authenticateToken, (req, res) => {
  res.json({ message: 'Welcome to protected route' });
});

const PORT = 3000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Frontend (Svelte):

1. Setup:

```
npx degit sveltejs/template frontend
cd frontend
```

```
npm install
```

2. User Registration Component (Register.svelte):

```
<script>
  let username = '';
  let password = '';

  const register = async () => {
    const response = await fetch('http://localhost:3000/register', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ username, password }),
    });
    if (response.ok) {
      alert('Registration successful');
    } else {
      alert('Registration failed');
    }
  };
</script>

<h2>Register</h2>

<form on:submit|preventDefault={register}>
  <label>
    Username:
    <input type="text" bind:value={username}>
  </label>
  <label>
    Password:
    <input type="password" bind:value={password}>
  </label>
  <button type="submit">Register</button>
</form>
```

3. User Login Component (Login.svelte):

```
<script>
  let username = '';
  let password = '';

  const login = async () => {
    const response = await fetch('http://localhost:3000/login', {
```

```
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ username, password }),
    });
    if (response.ok) {
      const { token } = await response.json();
      localStorage.setItem('token', token);
      alert('Login successful');
    } else {
      alert('Login failed');
    }
  };
</script>

<h2>Login</h2>

<form on:submit|preventDefault={login}>
  <label>
    Username:
    <input type="text" bind:value={username}>
  </label>
  <label>
    Password:
    <input type="password" bind:value={password}>
  </label>
  <button type="submit">Login</button>
</form>
```