Initialize Node.js Project

1. Create a new directory for your project and navigate into it:

mkdir expense-tracker

cd expense-tracker

2. Initialize a new Node.js project:

npm init -y

3. Install necessary dependencies for the backend:

npm install express mongoose bcryptjs jsonwebtoken

4. Install development dependencies:

npm install --save-dev nodemon

Set Up Front-End Framework

You can choose between React, Vue.js, or Angular. Here, we will use React.

1. Create a new React application:

npx create-react-app client

2. Navigate into the client directory:

cd client

3. Install necessary dependencies for React:

npm install axios react-router-dom

Set Up MongoDB

You can either use a local MongoDB instance or a cloud-based solution like MongoDB Atlas.

1. Install MongoDB:

npm install mongodb

High-Level Design

Architecture Overview

The application will consist of three main components:

1. **Front-end (React)**: For the user interface.
2. **Back-end (Node.js, Express)**: For handling API requests.
3. **Database (MongoDB)**: For storing user and expense data.

Data Flow

1. **User Registration/Login**:

→User registers/logs in through the front-end.

→Front-end sends a request to the back-end.

→Back-end processes the request, interacts with the database, and returns a response.

2. **Expense Management**:

→User adds/edits/deletes/view expenses through the front-end.

→Front-end sends a request to the back-end.

→Back-end processes the request, interacts with the database, and returns a response.

3. **Summary and Insights**:

→User requests a summary view.

→Front-end sends a request to the back-end.

→Back-end processes the request, aggregates data, and returns a response.

Project Structure

The project structure will be as follows:

```
expense-tracker/
|-- client/
|   |-- public/
|   |-- src/
|       |-- components/
|       |-- pages/
|       |-- App.js
|       |-- index.js
|-- server/
|   |-- models/
|   |-- routes/
|   |-- controllers/
|   |-- index.js
|-- package.json
|-- README.md
```

Implementation

Back-End (Node.js, Express)

1. **User Model**:

```
// server/models/User.js
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');

const UserSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});

UserSchema.pre('save', async function(next) {
  if (!this.isModified('password')) return next();
  this.password = await bcrypt.hash(this.password, 10);
  next();
});

const User = mongoose.model('User', UserSchema);
module.exports = User;
```

2. **Expense Model**:

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

const ExpenseSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  date: { type: Date, required: true },
  amount: { type: Number, required: true },
  category: { type: String, required: true },
  description: { type: String }
});

const Expense = mongoose.model('Expense', ExpenseSchema);
```

```
module.exports = Expense;
```

3. **User Routes**:

```
// server/routes/userRoutes.js
const express = require('express');
const User = require('../models/User');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const router = express.Router();

router.post('/register', async (req, res) => {
  try {
    const { username, email, password } = req.body;
    const user = new User({ username, email, password });
    await user.save();
    res.status(201).send('User registered successfully');
  } catch (error) {
    res.status(400).send(error.message);
  }
});

router.post('/login', async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await User.findOne({ email });
    if (!user || !await bcrypt.compare(password, user.password)) {
      return res.status(401).send('Invalid credentials');
    }
    const token = jwt.sign({ userId: user._id }, 'secretKey', { expiresIn: '1h' });
    res.json({ token });
  } catch (error) {
    res.status(400).send(error.message);
```

```
  }
});

module.exports = router;
```

4. **Expense Routes**:

```
// server/routes/expenseRoutes.js
const express = require('express');
const Expense = require('../models/Expense');
const router = express.Router();

router.post('/', async (req, res) => {
  try {
    const { userId, date, amount, category, description } = req.body;
    const expense = new Expense({ userId, date, amount, category, description });
    await expense.save();
    res.status(201).send('Expense added successfully');
  } catch (error) {
    res.status(400).send(error.message);
  }
});

router.get('/:userId', async (req, res) => {
  try {
    const expenses = await Expense.find({ userId: req.params.userId });
    res.json(expenses);
  } catch (error) {
    res.status(400).send(error.message);
  }
});

router.put('/:id', async (req, res) => {
```

```
  try {
    const expense = await Expense.findByIdAndUpdate(req.params.id, req.body, { new: true });
    res.json(expense);
  } catch (error) {
    res.status(400).send(error.message);
  }
});

router.delete('/:id', async (req, res) => {
  try {
    await Expense.findByIdAndDelete(req.params.id);
    res.send('Expense deleted successfully');
  } catch (error) {
    res.status(400).send(error.message);
  }
});

module.exports = router;
```

5. **Server Setup**:

```
// server/index.js
const express = require('express');
const mongoose = require('mongoose');
const userRoutes = require('./routes/userRoutes');
const expenseRoutes = require('./routes/expenseRoutes');

const app = express();

mongoose.connect('mongodb://localhost:27017/expense-tracker', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
  useCreateIndex: true,
```

```
  useFindAndModify: false
});

app.use(express.json());

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

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Front-End (React)

1. **App Component**:

```
// client/src/App.js
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import Register from './pages/Register';
import Login from './pages/Login';
import Dashboard from './pages/Dashboard';

function App() {
  return (
    <Router>
      <Switch>
        <Route path="/register" component={Register} />
        <Route path="/login" component={Login} />
        <Route path="/dashboard" component={Dashboard} />
      </Switch>
    </Router>
  );
}
```

```
export default App;
```

2. **Register Page**:

```
// client/src/pages/Register.js
import React, { useState } from 'react';
import axios from 'axios';

function Register() {
  const [username, setUsername] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleRegister = async () => {
    try {
      await axios.post('/api/users/register', { username, email, password });
      alert('User registered successfully');
    } catch (error) {
      alert(error.response.data);
    }
  };

  return (
    <div>
      <h2>Register</h2>
      <input type="text" placeholder="Username" value={username} onChange={(e) =>
setUsername(e.target.value)} />
      <input type="email" placeholder="Email" value={email} onChange={(e) =>
setEmail(e.target.value)} />
      <input type="password" placeholder="Password" value={password} onChange={(e) =>
setPassword(e.target.value)} />
      <button onClick={handleRegister}>Register</button>
    </div>  );
}
```