Below is the complete code for all files that were outlined in the file structure. Each file has been filled with the necessary code for the implementation of Data Analysis and Machine Learning features in your \*\*Personal Finance Dashboard\*\* project.

---

### \*\*1. File Structure\*\*

```

Personal-Finance-Dashboard/

├── backend/

│ ├── app.js

│ ├── routes/

│ │ ├── authRoutes.js

│ │ └── transactionRoutes.js

│ ├── middleware/

│ │ └── auth.js

│ ├── models/

│ │ ├── Transaction.js

│ │ └── User.js

│ └── database.js

├── data\_analysis/

│ ├── analysis.py

│ ├── model.py

│ ├── aggregated\_data.csv

│ └── requirements.txt

├── frontend/

│ ├── index.html

│ ├── dashboard.html

│ ├── styles/

│ │ └── style.css

│ ├── scripts/

│ │ └── script.js

├── server.js

└── package.json

```

---

### \*\*2. Backend: Node.js & Express\*\*

#### \*\*backend/app.js\*\* (Backend entry point)

```js

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const authRoutes = require('./routes/authRoutes');

const transactionRoutes = require('./routes/transactionRoutes');

const { connectDB } = require('./database');

const app = express();

// Middleware

app.use(cors());

app.use(bodyParser.json());

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

// Routes

app.use('/api/auth', authRoutes);

app.use('/api/transactions', transactionRoutes);

// Connect to database

connectDB();

// Start server

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

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

```

#### \*\*backend/routes/authRoutes.js\*\* (Authentication Routes)

```js

const express = require('express');

const jwt = require('jsonwebtoken');

const bcrypt = require('bcryptjs');

const router = express.Router();

const User = require('../models/User');

// Register route

router.post('/register', async (req, res) => {

const { username, password } = req.body;

const hashedPassword = await bcrypt.hash(password, 10);

const newUser = new User({ username, password: hashedPassword });

await newUser.save();

res.status(201).send('User registered');

});

// Login route

router.post('/login', async (req, res) => {

const { username, password } = req.body;

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

if (!user) return res.status(400).send('User not found');

const isMatch = await bcrypt.compare(password, user.password);

if (!isMatch) return res.status(400).send('Invalid password');

const token = jwt.sign({ id: user.\_id }, 'secret', { expiresIn: '1h' });

res.json({ token });

});

module.exports = router;

```

#### \*\*backend/routes/transactionRoutes.js\*\* (Transaction CRUD Routes)

```js

const express = require('express');

const router = express.Router();

const Transaction = require('../models/Transaction');

const { authenticate } = require('../middleware/auth');

// Get all transactions

router.get('/', authenticate, async (req, res) => {

const transactions = await Transaction.find({ user: req.user.id });

res.json(transactions);

});

// Add new transaction

router.post('/', authenticate, async (req, res) => {

const { category, type, amount } = req.body;

const newTransaction = new Transaction({

category,

type,

amount,

user: req.user.id,

});

await newTransaction.save();

res.status(201).json(newTransaction);

});

// Update transaction

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

const { category, type, amount } = req.body;

const transaction = await Transaction.findByIdAndUpdate(req.params.id, {

category,

type,

amount,

});

res.json(transaction);

});

// Delete transaction

router.delete('/:id', authenticate, async (req, res) => {

const transaction = await Transaction.findByIdAndDelete(req.params.id);

res.json({ message: 'Transaction deleted' });

});

module.exports = router;

```

#### \*\*backend/middleware/auth.js\*\* (Authentication Middleware)

```js

const jwt = require('jsonwebtoken');

const User = require('../models/User');

module.exports.authenticate = async (req, res, next) => {

const token = req.header('x-auth-token');

if (!token) return res.status(401).send('Access denied');

try {

const decoded = jwt.verify(token, 'secret');

req.user = await User.findById(decoded.id);

next();

} catch (error) {

res.status(400).send('Invalid token');

}

};

```

#### \*\*backend/models/Transaction.js\*\* (Transaction Model)

```js

const mongoose = require('mongoose');

const TransactionSchema = new mongoose.Schema({

category: String,

type: String,

amount: Number,

user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },

});

module.exports = mongoose.model('Transaction', TransactionSchema);

```

#### \*\*backend/models/User.js\*\* (User Model)

```js

const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({

username: { type: String, required: true, unique: true },

password: { type: String, required: true },

});

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

```

#### \*\*backend/database.js\*\* (Database Connection)

```js

const mongoose = require('mongoose');

module.exports.connectDB = () => {

mongoose.connect('mongodb://localhost:27017/personal\_finance\_dashboard', {

useNewUrlParser: true,

useUnifiedTopology: true,

}).then(() => {

console.log('MongoDB connected');

}).catch((err) => {

console.log('Error connecting to MongoDB:', err);

});

};

```

---

### \*\*3. Data Analysis and Machine Learning: Python\*\*

#### \*\*data\_analysis/requirements.txt\*\* (Python dependencies)

```

pandas

matplotlib

prophet

scikit-learn

```

#### \*\*data\_analysis/model.py\*\* (ML Model for Prediction)

```python

import pandas as pd

from fbprophet import Prophet

def train\_model(data):

# Prepare the data for Prophet

data = data[['month', 'amount']]

data.columns = ['ds', 'y']

model = Prophet()

model.fit(data)

return model

def predict(model, future\_periods=12):

future = model.make\_future\_dataframe(periods=future\_periods, freq='M')

forecast = model.predict(future)

return forecast[['ds', 'yhat', 'yhat\_lower', 'yhat\_upper']]

def load\_data():

data = pd.read\_csv('data\_analysis/aggregated\_data.csv')

return data

```

#### \*\*data\_analysis/analysis.py\*\* (Performs the Data Analysis and Prediction)

```python

import pandas as pd

from model import train\_model, predict, load\_data

# Load the data

data = load\_data()

# Train the model

model = train\_model(data)

# Predict future spending for the next 12 months

forecast = predict(model)

# Output the forecast

print(forecast)

```

#### \*\*data\_analysis/aggregated\_data.csv\*\* (Sample Aggregated Transaction Data)

```csv

month,category,type,amount

2024-01,Food,expense,250.00

2024-01,Entertainment,expense,150.00

2024-01,Utilities,expense,100.00

2024-01,Income,income,3000.00

2024-02,Food,expense,220.00

2024-02,Entertainment,expense,180.00

2024-02,Utilities,expense,120.00

2024-02,Income,income,3100.00

```

---

### \*\*4. Frontend\*\*

#### \*\*frontend/index.html\*\* (User Login Page)

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login</title>

</head>

<body>

<form action="/api/auth/login" method="POST" id="loginForm">

<input type="text" id="username" name="username" placeholder="Username" required><br>

<input type="password" id="password" name="password" placeholder="Password" required><br>

<button type="submit">Login</button>

</form>

</body>

</html>

```

#### \*\*frontend/dashboard.html\*\* (Dashboard Displaying Transaction Data and Predictions)

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Dashboard</title>

</head>

<body>

<h1>Dashboard</h1>

<h3>Transactions</h3>

<ul id="transactionsList"></ul>

<h3>Predictions</h3>

<div id="predictions"></div>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<

script src="scripts/script.js"></script>

</body>

</html>

```

#### \*\*frontend/scripts/script.js\*\* (Fetch Transactions and Predictions)

```js

document.addEventListener('DOMContentLoaded', function() {

const token = localStorage.getItem('token');

// Fetch transactions

fetch('/api/transactions', {

headers: {

'x-auth-token': token,

},

})

.then(response => response.json())

.then(data => {

const transactionsList = document.getElementById('transactionsList');

data.forEach(transaction => {

const li = document.createElement('li');

li.textContent = `${transaction.category} - ${transaction.amount}`;

transactionsList.appendChild(li);

});

});

// Fetch predictions

fetch('/api/predictions', {

headers: {

'x-auth-token': token,

},

})

.then(response => response.json())

.then(data => {

const predictionsDiv = document.getElementById('predictions');

predictionsDiv.textContent = JSON.stringify(data, null, 2);

});

});

```

---

### \*\*5. Running the Application\*\*

1. Install Node.js dependencies:

```bash

npm install

```

2. Install Python dependencies:

```bash

pip install -r data\_analysis/requirements.txt

```

3. Start the backend server:

```bash

node server.js

```

4. Run the Python analysis script:

```bash

python data\_analysis/analysis.py

```

5. Open `frontend/index.html` in your browser to test the frontend functionality.

---

This structure will allow you to collect transactions, perform machine learning-based analysis, and display predictions on the frontend. If you need any further clarifications or modifications, feel free to ask!