MERN Stack Developer Assessment: AI-Powered Finance Tracker

Assignment Overview

Build a full-stack finance tracker application that uses AI to parse bank statements and display financial insights in a dashboard format. This is a 24-hour take-home assignment designed to evaluate your technical skills, problem-solving abilities, and capacity to deliver a working product under time constraints.

Core Requirements

Functional Requirements

1. **File Upload System**
    - Allow users to upload bank statements (PDF, CSV, or text formats)
    - Support multiple file formats commonly used by banks
    - Implement file validation and error handling
2. **AI-Powered Statement Parsing**
    - Use AI/ML services to extract transaction data from uploaded statements
    - Identify and categorize transactions (expenses, income, transfers, etc.)
    - Extract key information: date, amount, description, merchant/source
3. **Dashboard Interface**
    - Display parsed transactions in a clean, organized table
    - Show financial summary cards (total income, total expenses, net balance)
    - Implement basic categorization of expenses (food, utilities, entertainment, etc.)
    - Include simple data visualizations (charts/graphs)
4. **Data Management**
    - Store parsed transaction data in MongoDB
    - Implement basic CRUD operations for transactions
    - Allow users to edit/correct AI-parsed data

Technical Stack Requirements

- **Frontend**: React.js with modern hooks and functional components

- **Backend**: Node.js with Express.js
- **Database**: MongoDB
- **AI Integration**: Choose from OpenAI GPT, Claude, Gemini any APIs
- **Styling**: Your choice (CSS, Styled Components, Material-UI, Tailwind, etc.)

Deliverables

1. Working Application

- Fully functional MERN stack application
- Deployed and accessible via URL (Vercel, Netlify, Heroku, or similar)
- Source code repository on GitHub with clear README

2. Documentation

- **README.md** with:
    - Setup and installation instructions
    - API documentation
    - Description of AI service used and integration approach
    - Known limitations and potential improvements
    - Time breakdown of development phases

3. Demo Materials

- Screen recording (2-3 minutes) demonstrating core functionality
- At least 2 sample bank statements for testing (can be mock data)

Evaluation Criteria

Technical Excellence (30%)

- **Code Quality**: Clean, readable, well-structured code with proper commenting
- **Architecture**: Logical separation of concerns, proper folder structure
- **Error Handling**: Robust error handling and user feedback
- **Security**: Basic security practices (input validation, secure file uploads)

AI Integration & Innovation (25%)

- **Effectiveness**: How well the AI parsing works with real-world bank statement formats
- **Implementation**: Quality of AI service integration and prompt engineering
- **Accuracy**: Precision of transaction extraction and categorization
- **Fallback Handling**: How the system handles AI parsing failures

User Experience (20%)

- **Interface Design**: Intuitive, clean, and responsive UI/UX
- **Performance**: Fast loading times and smooth interactions
- **Mobile Responsiveness**: Works well on different screen sizes
- **User Feedback**: Clear loading states, success/error messages

Problem-Solving & Creativity (15%)

- **Creative Solutions**: Innovative approaches to common problems
- **Feature Completeness**: How much functionality delivered within time constraints
- **Edge Case Handling**: Consideration of various bank statement formats and edge cases
- **Value-Added Features**: Any additional features that enhance user experience

Documentation & Communication (10%)

- **Code Documentation**: Clear comments and documentation
- **README Quality**: Comprehensive setup instructions and project description
- **Communication**: Clear explanation of technical decisions and trade-offs

Bonus Points

- **Advanced Categorization**: Smart expense categorization with machine learning
- **Data Visualization**: Interactive charts showing spending patterns over time
- **Export Functionality**: Ability to export data to CSV/PDF
- **Authentication**: User registration and login system
- **Bulk Processing**: Handle multiple statements at once
- **Transaction Matching**: Detect and handle duplicate transactions
- **Budget Tracking**: Basic budgeting features with alerts

Technical Guidelines

AI Usage Policy

- **Encouraged**: Use AI tools for code generation, debugging, and problem-solving
- **Document**: Clearly mention which AI tools you used and how
- **Original Work**: Ensure you understand and can explain all code submitted
- **Learning**: Use this as an opportunity to learn new technologies/approaches

Submission Requirements

Deliverable Format

1. **GitHub Repository**: Public repo with all source code
2. **Live Demo**: Deployed application with working URL
3. **Demo Video**: 2-3 minute screen recording showing functionality
4. **Email Summary**: Brief email with links and key highlights

Submission Deadline

Submit all deliverables within exactly 24 hours of receiving this assignment. Late submissions will be penalized.

Questions and Support

- For technical clarifications, email your questions within the first 4 hours
- No implementation help will be provided
- Focus on delivering a working MVP rather than a perfect solution

Final Notes

Remember, this is not just about completing all requirements—we're looking for:

- **Quality over quantity**: A well-built core feature is better than many broken features
- **Practical decision-making**: Smart trade-offs given the time constraint
- **Professional approach**: How you would handle a real-world client project
- **Growth mindset**: Willingness to learn and adapt to new challenges

Good luck! We're excited to see your creative and technical solutions to this challenging problem.