XML Parser Design Document

1. Objective - Build a MERN stack application for uploading, parsing, storing, and reporting on XML files. The app will allow users to upload XML files, extract credit report data, save it in MongoDB, and display comprehensive reports in a React frontend.

2. High-level architecture –

- Frontend:
  - React (with React Router for navigation)
  - Fetches/upload data via RESTful API
- Backend:
  - Node.js + Express
  - Handles file upload, XML parsing, data storage, and data retrieval
- Database:
  - MongoDB (via Mongoose ODM)
  - Stores extracted credit report data
- File Upload:

  According to the application use-case there is no need for storing the XML files permanently. As soon as the user uploads it, parse the data inside the XML file and then delete it, and also the file size limit is 2MB.

  - Handeled using the multer.memoryStorage with automatic clean up mechanism to delete the file after it is parsed. Parsing is fast as data is present in RAM directly. May impact performance when very large number of users but ideal for current use case.

  Alternatives –

  - multer.diskStorage() – suitable for larger files with manual clean up mechanism
  - using cloud storage
- XML Parsing:
  - xml2js for parsing XML to JS objects

3. Data Model (MongoDB Schema)

- Credit report Schema -

  - Basic Details -

- name:String (Full name of applicant)
- mobilePhone: String
- pan: String
- creditScore: Number

- Report Summary –
  - totalAccounts: Number
  - activeAccounts: Number
  - closedAccounts: Number
  - currentBalance: Number
  - securedAmount: Number
  - unsecuredAmount: Number
  - last7DaysEnquiries: Number
- CreditAccounts: Array of objects (one user can have multiple credit accounts)
  - accountNumber: String
  - bankName: String
  - currentBalance: Number
  - overdueAmount: Number
  - address: String
- createdAt: Date (auto)

4. API Specification

4.1 File Upload

- Endpoint: `POST /api/v1/reports`
- Request: Multipart/form-data, field name `file`, accepts `.xml`
- Response:
  - `200 OK` with upload confirmation and parsed data summary
  - `400 Bad Request` if no file/invalid format
  - `500 Internal Server Error` on parse/persist errors

4.2 Data Retrieval

- Endpoint:`GET /api/v1/reports`
- Request: None
- Response: Array of credit report summaries

- Endpoint: `GET /api/v1/reports/:id`
- Request: Credit report ID as params
- Response: Detailed report data

5. UI/UX Design

- Upload Page: (navigation path - /)
  - File input for XML
  - Displays upload status
- Reports List Page: (navigation path - /reports)
  - Displays all uploaded reports
  - Shows name, score, and link to details
- Report Details Page: (navigation path - /reports/{id})
  - Sections for Basic Details, Report Summary, Credit Accounts
  - Table for account info

6. Error Handling & Validation

- Backend:
  - Validate file type (xml) /size (2MB)
  - Graceful error responses (HTTP status + message)
  - Logging (console)

- Frontend:
  - Show error messages for upload failures
  - Validate file before upload (should be xml and less than 2MB)

7. Testing Strategy

- Backend:

        - Unit tests for XML parsing logic

        - Integration tests for upload and retrieval endpoints (Jest)

    - Frontend:

        - Render tests for UI components (React Testing Library + Vitest)

        - Upload form interaction tests

8. Deployment & Hosting

    - Local:

        - MongoDB running locally

        - Backend: Node.js server on port 5000

        - Frontend: React app on port 5173

    - Production:

        - Environment variables for MongoDB URI, PORT, VITE_API_URL

        - Separate deployment for frontend/backend with frontend on vercel and backend on render or some other platform.

9. README & Documentation

    - Setup instructions for backend and frontend

    - API endpoint documentation

    - Testing instructions

    - Schema explanation

    - Features of the application

10. Future Enhancements

    - User authentication for uploads.

    - Pagination/filtering for reports.

    - Export reports to PDF/CSV.

    - Cloud storage for file upload and it’s temporary storage.