📘 Detailed Project Documentation (Final Clean Version)

Transformer-Based Molecular Property Prediction & Visualization System

---

1. Introduction

This project is a full-stack molecular analysis system that accepts a **SMILES string**, predicts **19 quantum-chemical properties**, and visualizes the molecule using **2D and 3D renderings**. It also includes an interactive chemistry assistant powered by **Gemini-2.5-flash-lite**.

The system combines:

- A **Transformer regression model** trained on quantum-chemical data
- A **FastAPI backend** for prediction & AI chat
- A **React frontend** that displays properties, 2D diagrams, 3D models, and chatbot responses

The workflow is:

1. User enters SMILES
2. Backend predicts properties
3. Frontend displays results + visualizations
4. Optional conversation with chatbot

---

2. Model Overview (Transformer Regressor)

The core predictor is a **Transformer-based regression model**, not an RNN or CNN. It operates on **2048-bit Morgan fingerprints**, which represent local chemical environments of atoms.

Model Input

- 2048-dimensional Morgan fingerprint
- Radius = 2
- Binary bit vector representing substructures of the molecule

Model Architecture

The model structure is:

1. **Input projection layer**
    - Projects the 2048-d fingerprint → 256-dim embedding
2. **Positional embedding**
    - A trainable vector added to the embedding to give positional context
3. **Transformer Encoder (4 layers)**
    - Multi-head self-attention (8 heads)
    - Feedforward network (hidden size 1024)
    - Dropout for stability
4. **Output head (multi-layer MLP)**
    - 256 → 512 → 256 → **19-property output**

Output

The model predicts **19 quantum-chemical properties**, including:

- Rotational constants (A, B, C)
- Dipole moment
- Isotropic polarizability
- HOMO, LUMO, HOMO-LUMO gap
- Internal energies at multiple temperatures
- Enthalpy, free energy
- Heat capacity
- Atomization energies (various conditions)

These properties represent structural, electronic, and thermodynamic behavior of molecules.

---

3. Backend Architecture (FastAPI)

The backend implements three core functionalities:

1. **Property prediction**
2. **Molecule parsing and information extraction**
3. **Chemistry chatbot**

3.1 Prediction Endpoint (**POST /predict**)

Steps performed:

1. Parse SMILES using RDKit
2. Generate a **2048-bit Morgan fingerprint**
3. Apply input scaler
4. Run the Transformer model
5. Inverse-scale the output values
6. Compute confidence heuristics
7. Return JSON containing:
    - 19 predicted properties
    - Units
    - Confidence
    - Clean SMILES
    - Basic RDKit molecule info
        - formula
        - molecular weight
        - atom count
        - bond count
        - ring count
        - aromaticity

3.2 Chat Endpoint (POST /chat)

Uses **Gemini-2.5-flash-lite** to give:

- Explanations of predicted properties
- SMILES interpretation
- High-level chemistry concepts

It does NOT run predictions; it only answers questions.

3.3 Other Endpoints

- /properties → property names + units
- /health → confirms backend readiness

---

4. Frontend Architecture (React)

The frontend is responsible for input, prediction display, visualizations, and chatbot UI.

Top-level components:

✓ 1. Input Form

- Text box where user enters SMILES
- "Analyze" button triggers the prediction API
- Example SMILES presets
- Loading state & error messages

✓ 2. Prediction Display (ResultsCard)

Shows:

- All 19 property values
- Confidence score
- Units

- Molecule info (formula, MW, rings, aromaticity)

✓ 3. 2D Structure Viewer (CACTUS API)

Features:

- Fetches 2D PNG of the molecule
- Displays loading animation
- Provides download option
- Clean SMILES shown under the image
- Handles invalid SMILES gracefully

✓ 4. 3D Structure Viewer (3Dmol.js)

Features:

- Dynamically loads the 3Dmol.js library
- Fetches 3D SDF structure from PubChem
- Falls back to CACTUS if needed
- Renders:
    - Stick model
    - Sphere model
- Automatically rotates the molecule
- Reset view button
- Retrieves IUPAC name
- Handles missing 3D structures gracefully

✓ 5. Chemistry Chatbot

- Opens an interactive chat window
- Talks to Gemini-2.5-flash-lite
- Explains:

    - molecular properties
    - structural behavior
    - chemistry fundamentals
- Does not modify or influence predictions

---

5. Full User Flow

Step 1 — User Input

User enters a SMILES string and clicks **Analyze**.

Step 2 — Backend Prediction

FastAPI processes the SMILES:

- RDKit → fingerprint
- Transformer → property vector
- Post-process & return JSON

Step 3 — Frontend Displays Results

The UI shows:

- All predicted values
- Molecule metadata
- 2D structure
- 3D model

Step 4 — Optional Chat

The user can ask the chatbot:

- What does dipole moment mean?
- What does this prediction imply?

- What is the structure of benzene?

Chat is purely informational.

---

6. Error Handling

Backend errors handled:

- Invalid SMILES → “Invalid SMILES string”
- Missing model/scaler → service unavailable
- RDKit conversion failure
- Prediction errors

Frontend errors handled:

- Network failures
- API returning success: false
- CACTUS 2D image failures
- PubChem 3D model missing
- 3Dmol.js loading failure

All components display friendly UI messages.

---

7. Why This System Works Well

- **Transformer encoder** is good at modeling relationships between fingerprint bits.
- **Morgan fingerprints** capture chemical environments effectively.
- **FastAPI** ensures fast, clean prediction API responses.
- **React + Framer Motion** provides smooth visualization.
- **3Dmol.js + CACTUS + PubChem** gives reliable molecular rendering.

- **Gemini-2.5-flash-lite** gives natural-language chemistry help.

The combination results in a fully interactive, educational, and highly functional molecular analysis tool.

---

8. Conclusion

This project brings together deep learning, cheminformatics, visualization, and interactive AI into a unified system:

- SMILES → Transformer predictions
- Predictions → 2D/3D visualization
- Chatbot → chemistry explanations

The system is modular, extendable, and ready for deployment or publication.