Hacettepe University

Computer Engineering Department

BBM479 Project Proposal Report

Project Details

Title	LLM-Based Toxicity Advisor	
Short Description (max. 200 words)	This project addresses the high attrition rate in drug discovery caused by unforeseen toxicity. While computational QSAR/ADMET models can predict risks early, their output is often a set of complex numerical scores that are difficult for chemists to interpret, creating a significant bottleneck. Our solution is a web-based platform that bridges this "interpretability gap." It integrates a battery of validated ADMET prediction models with a powerful Large Language Model (LLM). When a user submits a compound, the system generates raw toxicity predictions and then tasks the LLM to act as an expert toxicologist. The LLM synthesizes this data into a concise, actionable report, complete with a risk summary, an explanatory narrative, and recommendations for subsequent experiments. By translating complex data into clear decisions, our platform empowers researchers to identify and mitigate risks earlier, accelerating the drug discovery pipeline and enabling a more efficient allocation of resources.	
Supervisor	Hacer YALIM KELEŞ	
Technical and Scientific Difficulty	( ) Easy ( ) Mediocre (X) Challenging	
External Support	( ) Yes (X) No	
If yes,	**Type**	**Details**
	( ) Company Funding / Support	Company name: Amount:
	( ) TÜBİTAK Project Fund	Type: Amount:
	( ) Other Fund	Source : Amount:

Group Members

	Full Name	Student ID
1	Kadir Çağrı KUZAYTEPE	2210356036

2	Mehmet Baki ŞAHİN	2210765030

Project Summary (/ 20 Points)

Explain the project in summary, including your motivation to do the project, your solution plan in short and your expected outcome and impact. You have to summarize your project between 200-500 words.

Our project is dedicated to addressing a critical challenge in early-stage drug discovery and chemical development: the high cost and failure rate associated with the late-stage identification of adverse toxicological profiles. The motivation behind this initiative is to de-risk the development process by providing researchers with a powerful, intuitive, and data-driven tool for early toxicity assessment. By flagging potentially hazardous candidates sooner, we can help focus resources on more promising avenues, accelerating the overall discovery pipeline.

Our solution is an integrated decision-support platform that synergizes the predictive power of quantitative structure-activity relationship (QSAR) models with the analytical capabilities of Large Language Models (LLMs). The system is built on a microservice architecture. When a user submits a candidate molecule through the web interface, our Node.js backend orchestrates a complex workflow. It leverages services to fetch standardized chemical data from public databases like PubChem and dispatches analysis requests to a dedicated Python-based ADMET (Absorption, Distribution, Metabolism, Excretion, and Toxicity) prediction service. This service runs computational models to generate raw scientific data on the molecule's likely behavior. The core innovation of our project is the final step: these complex, structured data points are then processed by an LLM, which synthesizes them into a coherent, human-readable analysis.

The expected outcome is to provide chemists and pharmacologists with concise, actionable intelligence for each compound. This includes a summary report detailing potential risks, predictive toxicity scores, notes on the model's confidence and uncertainty, and, most importantly, intelligent recommendations for the next crucial experimental validation steps. The ultimate impact of this project will be to empower scientists to make faster, more informed decisions, significantly reducing the financial and temporal costs of drug development and enhancing the safety of novel chemical entities.

Problem Definition and Literature Review (/ 20 Points)

Define your problem as clearly as possible. Explain your inputs, your context, your outputs and your limitations. Try to use a scientific language as much as possible. Where necessary use citations to existing literature to create context and clarify the problem. Equations, flow charts, etc. are welcome.

1. Problem Definition

The contemporary pharmaceutical research and development (R&D) pipeline is characterized by prohibitively high costs and a low probability of success. A significant driver of this inefficiency is the high attrition rate of drug candidates, with a substantial portion of failures in

late-preclinical and clinical stages being attributed to unforeseen toxicity (DiMasi, Grabowski, & Hansen, 2016). The manifestation of adverse effects, such as hepatotoxicity, cardiotoxicity, or mutagenicity, after significant investment has been made, represents a major financial and ethical burden.

The core problem this project addresses is the "interpretability gap" between high-throughput *in silico* toxicological predictions and their practical application in medicinal chemistry decision-making. While numerous Quantitative Structure-Activity Relationship (QSAR) models exist to predict ADMET (Absorption, Distribution, Metabolism, Excretion, Toxicity) properties, their outputs are typically numerical scores, probabilities, or classifications (e.g., "Class I toxicant"). This raw data lacks the contextual narrative required by chemists and project teams to make informed decisions, such as how to prioritize analogs, which specific liabilities to address through structural modification, or what subsequent *in vitro* or *in vivo* experiments to conduct. Our project aims to bridge this gap by creating a system that not only predicts toxicity but also synthesizes these predictions into an actionable, evidence-based narrative report.

---

2. Context: The Paradigm of *In Silico* ADMET Prediction

To mitigate late-stage failures, the "fail early, fail cheap" paradigm has been widely adopted, promoting the use of computational methods at the earliest stages of discovery. Among these, QSAR modeling is a cornerstone. A QSAR model is a mathematical regression or classification model that correlates quantitative measures of chemical structure (molecular descriptors) with a biological or toxicological activity.

The general form of a linear QSAR model can be expressed as:

$$\text{Activity} = \beta_0 + \Sigma(\beta_i * D_i) + \varepsilon$$

Where:

- **Activity** is the dependent variable, often expressed as log(1/C), where C is the concentration required to elicit a specific biological effect (e.g., IC_{50}, LD_{50}).
- **Di** are the molecular descriptors (independent variables), which can be constitutional, topological, quantum-chemical, etc.
- **βi** are the regression coefficients determined from a training set of molecules with known activities.
- **ε** is the error term.

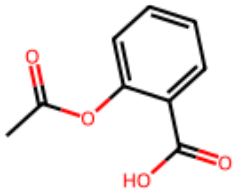
Our system leverages a battery of pre-existing, validated QSAR models for various critical toxicological endpoints (e.g., Ames mutagenicity, hERG inhibition, hepatotoxicity). The challenge, however, is that a single compound will be evaluated against dozens of such models, generating a high-dimensional output vector of uncorrelated data points that is difficult to interpret holistically. This is the context into which our LLM-based synthesis layer is introduced.

---

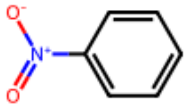
3. System Architecture and Data Flow

The platform operates via a microservice architecture designed to decouple tasks for scalability and maintainability. The data flow is visualized below. It seems there was an issue generating the image. Here is the text formatted correctly. I will describe the intended visual.

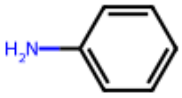
(Intended Image: A 2x2 grid of chemical structures. Top-left: Aspirin, labeled "Example Input". Top-right: Nitrobenzene, labeled "Mutagenicity Alert". Bottom-left: Aniline, labeled "Mutagenicity Alert". Bottom-right: Amiodarone, labeled "hERG Inhibitor".)



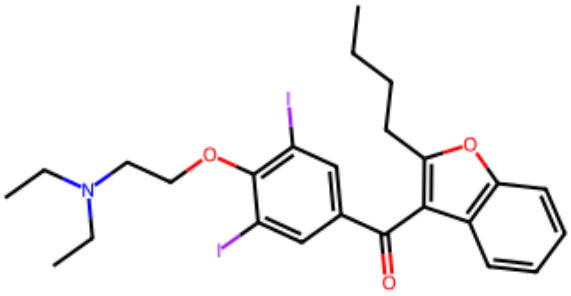
Aspirin (Example Input)



Nitrobenzene (Mutagenicity Alert)



Aniline (Mutagenicity Alert)



Amiodarone (hERG Inhibitor)

4. System Inputs

The primary input to the system is the chemical structure of a candidate compound. This can be provided in several standard formats:

- SMILES (Simplified Molecular-Input Line-Entry System) string.
- IUPAC Name.
- Other common identifiers like CAS Number or an internal compound ID.

The backend canonicalizes this input, typically by resolving it to a standardized SMILES string and 2D/3D structure file via the PubChem service, ensuring consistent input for all downstream predictive models.

---

5. System Outputs

The system is designed to produce a multi-faceted, structured output that goes beyond simple numerical scores. For each input compound, the output includes:

1. **Executive Summary:** A top-level natural language paragraph summarizing the compound's overall toxicological profile and highlighting the most significant predicted liabilities.
2. **Quantitative Risk Profile:** A detailed breakdown of predictions for each toxicological endpoint, including:
    - The predicted value (e.g., pIC_{50} for hERG).
    - A classification (e.g., "High Risk," "Low Risk").
    - The model's confidence score or applicability domain assessment, indicating how reliable the prediction is for the given structure.
3. **Evidence-Based Narrative:** A section where the LLM explains the predictions in scientific context. For example, it might state: "The model predicts a high probability of Ames mutagenicity. This is likely due to the presence of an aromatic nitro group, which is a well-known structural alert for DNA reactivity (Ashby & Tennant, 1991)."
4. **Actionable Recommendations:** A prioritized list of suggested next steps. This is the key decision-support component. Examples include:
    - "High risk of cardiotoxicity predicted. Recommend immediate *in vitro* hERG patch-clamp assay for confirmation."
    - "Potential for hepatotoxicity noted. Suggest monitoring liver enzyme biomarkers in initial *in vivo* rodent studies."
    - "Structural alert for mutagenicity identified at the aniline moiety. Recommend synthesis of analogs where this group is masked or replaced."

---

6. Limitations

It is critical to acknowledge the inherent limitations of this approach:

1. **Predictive, Not Definitive:** All *in silico* predictions are probabilistic and subject to false positives and negatives. They are intended to guide and prioritize experimentation, not replace it.
2. **Applicability Domain (AD):** The accuracy of any QSAR model is confined to its AD. For novel chemical scaffolds that are dissimilar to the model's training set, predictions may be unreliable. The system must transparently communicate when a compound falls outside the AD.
3. **Mechanistic Insight:** While the system can identify structural alerts, QSAR models are often "black boxes" and do not elucidate the precise biological mechanism of toxicity. The LLM can hypothesize mechanisms based on literature, but this remains an inference, not a direct output.
4. **Data Quality and Availability:** The performance of the underlying QSAR models is fundamentally limited by the quality and scope of the public and private training data available. For rare or complex toxicological endpoints, robust models may not exist.

Solution Plan (/ 20 Points)

Explain the potential paths to solution. You should propose at least one solid plan to attack the problem. Dissect your plan into steps and clearly identify the inputs and outputs of each step. You are not expected to provide the technical details of each step. Provide a weekly timeline/Gantt chart displaying the relevant weeks for each step. er of the project will contribute by assigning members to steps. If you are assigning more than one member to a step, explain their specific role and how the work will be divided among them.

Our Roadmap for Bringing This Project to Life

As we thought about how to build this project, we had three main paths in front of us. We evaluated the pros and cons of each:

1. The All-in-One (Monolithic) Approach This meant building everything into one single, massive application, probably using a Python framework like Django or FastAPI. The web frontend, QSAR predictions, and LLM (Large Language Model) calls would all live in the same codebase.

- **The upside:** It's the simplest to set up initially. No need for extra protocols for services to talk to each other.
- **The downside:** As it grows, maintenance and scaling become a nightmare. For example, a CPU-intensive QSAR prediction could lock up the entire website. We'd also lose the chance to use the best technology for each job (e.g., Python is great for ML, but Node.js might be more efficient for web requests).

2. The Fully Serverless Approach Here, we'd design each component as a separate cloud function (like AWS Lambda or Google Cloud Functions). An API gateway would trigger the main analysis function, which would then call other functions for prediction and reporting.

- **The upside:** Incredibly scalable and cost-effective (you pay for what you use). No server management headaches.
- **The downside:** As things get complex, it can turn into what we call a "Lambda-lith"—a structure that's hard to manage. We risk vendor lock-in, and "cold start" latency issues could hurt the user experience.

3. The Microservice Approach (Our Choice) This path, which also best fits the project's existing structure, involves building each core function (prediction, reporting, web) as its own independent, standalone service. These services will communicate over clear protocols like REST APIs.

- **The upside:** Services can be developed, deployed, and scaled independently. It lets us use the best language for each job (Python for ML, Node.js for the backend). Most importantly, it's highly resilient. Even if the ADMET predictor fails, the rest of the application stays up.
- **The downside:** The initial setup is a bit more complex. It requires careful API design and a plan for how services will find each other (service discovery).

Our Game Plan: The Microservice Path

We've broken this project down into a five-step plan to get to a functional prototype:

Step 1: Build the "Science Engine" (The ADMET Prediction Service) Our first goal is to create a reliable, standalone scientific engine.

We'll take our critical toxicological endpoints (Ames mutagenicity, hERG inhibition, etc.) and our trained QSAR model files. We'll use Flask or FastAPI to build a simple REST API around these models. This service will have one job: to accept a **SMILES** (chemical fingerprint) string at an endpoint like `/predict`, calculate the molecular descriptors, run the predictions, and return the raw **JSON** data (e.g., `{"hERG_pIC50": 5.2, "ames_mutagenicity_prob": 0.85}`). Finally, we'll package this entire service into a **Docker container** to make it portable.

Step 2: Develop the "Orchestra Conductor" (The Main Backend Service) This service will act as the application's central nervous system.

When a user makes a request from the UI with a chemical name like "Aspirin," the request will hit our main backend endpoint, `/api/analyze`. The backend's job is, in order:

1. Take that name and query an external service (like PubChem) to find its correct SMILES string.
2. Send that SMILES string to our ADMET Prediction Service (from Step 1).
3. Receive the raw numerical predictions and gather them for the next step.

Step 3: Bring in "The Translator" (The LLM Synthesis Layer) This is where the innovative part begins. We'll translate raw data into "meaningful" insight.

The backend service will send a specially designed **prompt** to an LLM (like the Gemini API), along with all the raw data, instructing it to act as an "expert computational toxicologist." The LLM will use this data and the instructions to generate the relevant sections of the report (introduction, toxicology summary, conclusion, etc.). Our backend will receive this ready-to-read text from the LLM and package it into a final JSON object.

Step 4: Build "The Showcase" (The Frontend User Interface) We'll build the part the user sees and interacts with—the website.

We'll design a clean interface with a simple input form, an "Analyze" button, and a results area. When the user clicks the button, the UI will call the `/api/analyze` endpoint in the background, show a loading indicator, and then render the final, human-readable report from Step 3 in a clean, well-formatted way.

Step 5: End-to-End Testing and Validation Finally, we'll make sure the whole thing works robustly, both technically and scientifically.

We'll define a validation set of 10-20 compounds with well-known toxicological profiles. We'll run each one through the system and critically compare the generated reports against existing scientific literature. This comparison will help us refine the "prompt" (from Step 3) to improve accuracy and reduce LLM "hallucinations." We'll also conduct basic user experience testing to make sure the system is easy to use.

Project Timeline

We've set a 7-week timeline to develop this prototype.

Gantt Chart for Design Project

Methodology (/ 20 Points)

Explain the methodology you will use in each of the steps you have described under your solution plan. Here, you are expected to give more technical details about each solution step. Also explain how each member of the project will contribute by assigning members to steps. If you are assigning more than one member to a step, explain their specific role and how the work will be divided among them.

Methodology

Our project will be executed using a microservice-based methodology that aligns with our chosen solution path. This approach allows for technological specialization in each service and promotes a decoupled, scalable, and resilient architecture. Each step of the solution plan will be implemented with specific technologies and a clear division of labor between team members.

Step 1: Building the ADMET Prediction Service ("Science Engine")

This foundational step involves creating a robust, containerized microservice dedicated to performing all scientific computations.

- **Technical Methodology:**
  - **Framework:** A RESTful API will be developed using **Python 3.11** and the **FastAPI** framework. This provides a high-performance, asynchronous server gateway for our models. The primary endpoint will be `/predict`, as defined in `admet/main.py`.
  - **Core Libraries:**
    - **admet-ai:** This library will be used as the primary tool for generating machine learning-based predictions for various ADMET endpoints.
    - **RDKit:** This cheminformatics library is crucial for validating and canonicalizing SMILES strings, calculating molecular descriptors (e.g., MolWt, LogP), and identifying structural alerts (PAINS, Brenk) as implemented in `admet/analysis.py`.
  - **Data Enrichment:** The service will enrich the predictions with real-world experimental data by querying public databases. This is handled by functions in `admet/queries.py` which interface with **PubChem** (via `pubchempy`) and **ChEMBL** (via `chembl_webresource_client`).
  - **Asynchronous Task Handling:** The service is designed to be called by a worker process. The `admet/worker.py` script listens to a **RabbitMQ** message queue (`admet_tasks`). Upon receiving a task, this worker makes an HTTP request to its own local FastAPI server. This isolates the long-running scientific computation from the message queue listener, preventing timeouts and improving stability.
  - **Containerization:** The entire Python service, including all its scientific dependencies, will be containerized using **Docker**. The `admet/Dockerfile` utilizes `micromamba` to install complex dependencies like RDKit efficiently, ensuring a reproducible and portable environment.
- **Team Contribution:**
  - **Member Assigned:** Kadir Çağrı KUZAYTEPE
  - **Role:** Kadir will be the primary developer for the entire ADMET service. His responsibilities include implementing the FastAPI endpoints, integrating the `admet-ai` and `RDKit` libraries into the analysis pipeline (`admet/pipeline.py`), developing the RabbitMQ worker logic, and writing and maintaining the service's Dockerfile.

Step 2: Developing the Main Backend Service ("Orchestra Conductor")

This central service will manage the application's overall workflow, from handling user requests to orchestrating communication between all other services.

- **Technical Methodology:**

    - **Framework:** The backend will be built on **Node.js** using the **Express.js** framework, as configured in backend/server.js. It will serve as the main API gateway for the frontend and will also serve the static frontend files.
    - **Orchestration Logic:** The core workflow orchestration is defined in backend/src/handlers/chatHandler.js. Upon receiving a request at /api/chat, this handler will:
        - Parse the user's input to identify a chemical entity.
        - Resolve the entity to a SMILES string using backend/src/services/pubchemService.js.
        - Dispatch the analysis task to the ADMET service via the RabbitMQ message queue, using the logic in backend/src/services/queueService.js.
    - **Asynchronous Communication:**
        - **RabbitMQ (amqplib):** The backend will publish task messages to the admet_tasks queue for the Python workers to consume.
        - **WebSockets (ws):** A WebSocket server, defined in backend/src/services/notificationService.js, will be used to send the final, processed results back to the client in real-time once the analysis is complete. This avoids long-polling and provides a responsive user experience.
    - **Caching:** A **Redis** client (backend/src/services/redisService.js) will be heavily utilized to cache results from external API calls (e.g., PubChem queries) and, most importantly, the final LLM-generated reports to reduce latency and redundant processing on subsequent identical requests.
- **Team Contribution:**
    - **Member Assigned:** Kadir Çağrı KUZAYTEPE
    - **Role:** Kadir will be responsible for developing the Node.js backend, setting up the Express server, implementing the API routes, and managing the microservice communication layer (RabbitMQ publisher and WebSocket server). He will ensure that the backend correctly orchestrates the entire analysis workflow.

Step 3: Implementing the LLM Synthesis Layer ("The Translator")

This logical layer within the backend is responsible for transforming raw scientific data into human-readable, actionable insights.

- **Technical Methodology:**
    - **LLM Integration:** This layer will be implemented within the Node.js backend, primarily in the /api/task-complete endpoint handler in backend/server.js.
    - **Prompt Engineering:** A carefully engineered system prompt, stored in backend/src/utils/constants.js (admetContextPrompt), will instruct the LLM to act as an expert toxicologist.
    - **API Service:** The backend/src/services/llmService.js will handle all interactions with the Large Language Model. It is configured to use the **OpenRouter API**, allowing for flexibility in model selection (e.g., GPT, Gemini). This service

includes a robust retry mechanism with exponential backoff to handle potential API rate limits or transient errors.
    - **Data-to-Text Synthesis:** When the backend receives the raw JSON data from the ADMET worker, it will format this data into a structured text block. This block, along with the system prompt, will be sent to the LLM to generate the final narrative report. The generated report is then cached in Redis before being sent to the user via WebSocket.
- **Team Contribution:**
    - **Member Assigned:** Mehmet Baki ŞAHİN
    - **Role:** Mehmet will be responsible for the LLM integration. His primary tasks are prompt engineering and data formatting. He will design, test, and refine the prompts sent to the LLM to ensure the generated reports are accurate, coherent, and scientifically sound. He will also implement the logic for caching the final reports in `backend/server.js`.

Step 4: Building the Frontend User Interface ("The Showcase")

This step involves creating a dynamic, intuitive, and responsive web interface for the user.

- **Technical Methodology:**
    - **Technology Stack:** The frontend will be developed using modern, framework-less **Vanilla JavaScript (ES6 Modules)**, **HTML5**, and **CSS3**. This approach provides maximum control and performance. The code is modularized into services, components, and utilities as seen in the `frontend/js` directory.
    - **Core Components:**
        - **`main.js`:** The application's entry point, responsible for initializing all components and managing the overall state.
        - **`ui.js`:** Manages all primary DOM interactions, including switching between the welcome screen and the chat interface, and handling UI state changes.
        - **`molecule.js`:** A custom-built component that uses the **HTML5 Canvas API** to parse SMILES strings and render 2D molecular structures. It includes interactive features like panning and zooming.
    - **API Communication:** The `frontend/js/services/api.js` service handles all `fetch` requests to the backend. For asynchronous ADMET tasks, it receives a `sessionId` and establishes a **WebSocket** connection to listen for the final report.
    - **Styling:** A modular CSS approach is used, with styles organized into `base`, `layouts`, and `components`. **CSS Custom Properties (variables)** are used for theming and maintaining consistency, as defined in `frontend/css/base/variables.css`.
- **Team Contribution:**
    - **Member Assigned:** Mehmet Baki ŞAHİN
    - **Role:** Mehmet will be the sole developer for the entire frontend. His responsibilities include building the HTML structure, designing the user interface

with modular CSS, and implementing all client-side interactivity and logic in JavaScript. This includes developing the custom molecule renderer and managing the WebSocket communication for real-time updates.

Step 5: End-to-End Testing and Validation

This final step ensures the technical robustness and scientific validity of the entire system.

- **Technical Methodology:**
  - **Technical Validation:** A suite of integration tests will be performed to ensure seamless data flow between the services: Frontend -> Backend -> RabbitMQ -> ADMET Worker -> Backend -> WebSocket -> Frontend. Docker Compose will be used to create an isolated environment for this testing.
  - **Scientific Validation:** We will compile a validation set of 10-20 well-characterized compounds (e.g., Aspirin, Paracetamol, Amiodarone) with known toxicological profiles. Each compound will be processed by the system. The LLM-generated reports will be critically evaluated against established scientific literature to assess their accuracy.
  - **Prompt Iteration:** The results from the scientific validation will be used to iteratively refine the LLM prompt in `backend/src/utils/constants.js` to minimize "hallucinations" and improve the quality and relevance of the generated insights.
  - **User Experience (UX) Testing:** Informal UX testing sessions will be conducted to gather feedback on the platform's usability, clarity of the reports, and the intuitiveness of the molecule drawing tool.
- **Team Contribution:**
  - **Members Assigned:** Kadir Çağrı KUZAYTEPE and Mehmet Baki ŞAHİN
  - **Division of Labor:**
    - **Kadir** will be responsible for the **technical validation**. He will set up the testing environment and write scripts to automate the testing of the microservice communication and ensure the reliability of the data pipeline.
    - **Mehmet** will lead the **scientific and UX validation**. He will be responsible for selecting the validation compounds, comparing the generated reports against literature, refining the LLM prompts based on the findings, and conducting the user feedback sessions.

Outcome and Impact (/ 20 Points)

Explain the expected outcome of your project. If it is a software product, try to include example screen designs, if it is a hardware product, try to provide detailed technical specifications, if it is

research output try to explain the outcome's contribution to the field. Also, explain the potential impacts of your results. These may be how the result will be used in real life, how it will change an existing process, or where it will be published, etc.

Outcome and Impact

Expected Outcome: A Decision-Support Software for Modern Chemists

The primary outcome of this project will be a fully functional, web-based software product. This platform is designed as an intelligent decision-support tool for medicinal chemists, pharmacologists, and researchers involved in early-stage drug discovery. The final product will deliver a seamless user experience, translating complex computational data into clear, actionable intelligence.

The key features and outputs of the software will include:

1. **Intuitive User Interface:** A clean, modern web interface allowing users to submit a chemical compound by its name (e.g., "Aspirin") or its SMILES string. The platform will also feature an interactive 2D molecule viewer that visualizes the compound's structure.
2. **Comprehensive and Structured Analysis Report:** For each submitted compound, the system will generate a multi-part report that goes far beyond raw numerical data. The report will be structured as follows:
    - **Executive Summary:** A top-level paragraph, generated by the LLM, that summarizes the compound's overall risk profile and highlights the most critical predicted liabilities.
    - **Visual Risk Profile:** An interactive radar chart that provides an at-a-glance visualization of the compound's performance across key toxicity endpoints (e.g., Ames, hERG, DILI).
    - **Detailed Predictions:** A quantitative breakdown of predictions from the underlying ADMET models, presented in a clear, tabular format.
    - **Evidence-Based Narrative:** An LLM-synthesized explanation of the predictions in a scientific context, linking structural features of the molecule to potential toxicological risks.
    - **Actionable Recommendations:** A prioritized list of suggested next steps for experimental validation, helping researchers make informed decisions on whether to advance, modify, or deprioritize a compound.

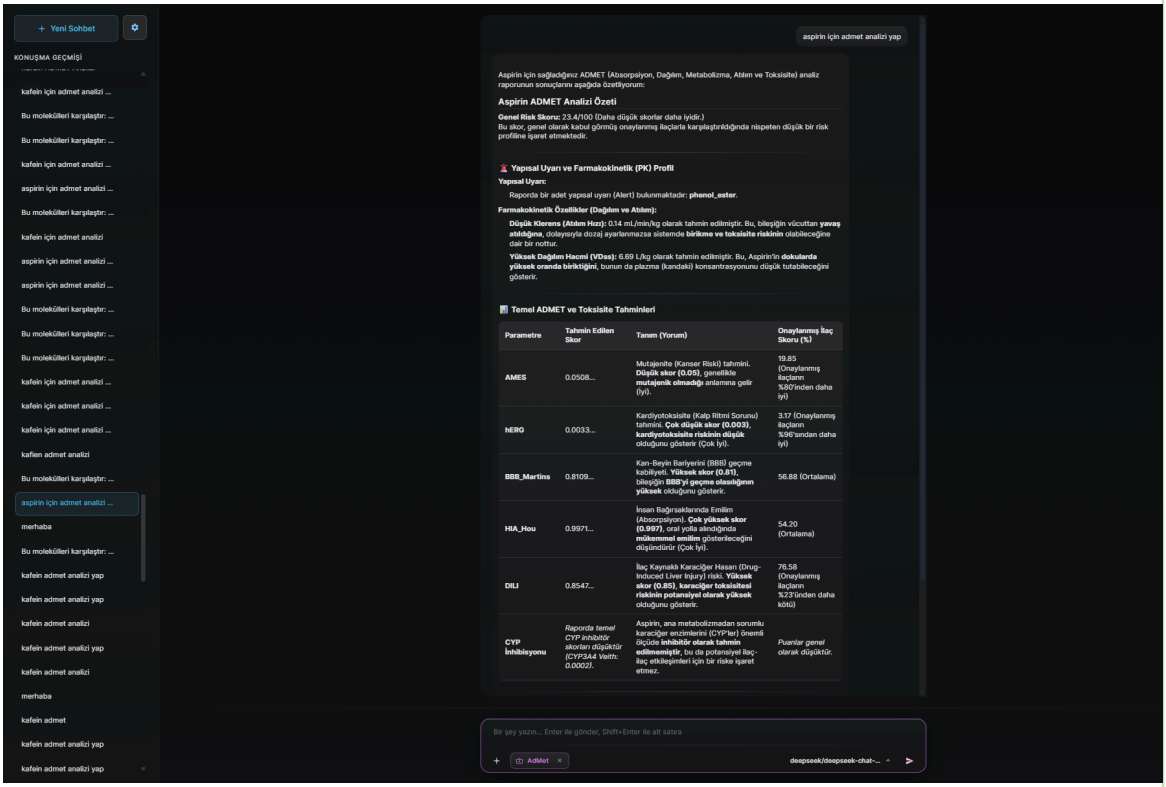
Example Screen Designs:

Below are conceptual designs for the key screens of the platform.

- **1. Interactive Molecule Drawing and Analysis Screen:** This screen functions as an intelligent, conversational interface for molecular visualization. The user can make a request in natural language (e.g., "draw caffeine" or "show me the structure of aspirin"). The system leverages an LLM to interpret the request, resolves the molecule's structure, and renders it directly in the interactive 2D viewer. Subsequently, key physicochemical

properties for the drawn molecule are automatically fetched from public databases and displayed in an adjacent panel.

- **2. Final Analysis Report Screen:** This is the primary output screen where the user sees the synthesized report for a single molecule. It integrates data visualization with LLM-generated text to provide a comprehensive overview.



Potential Impact: Transforming the Drug Discovery Workflow

The impact of this project extends beyond being a simple software tool; it aims to fundamentally change and improve an existing critical process in pharmaceutical R&D.

- **Real-World Application:** The platform is designed for direct use in real-world scenarios by:
    - **Medicinal Chemists** in pharmaceutical companies and academic labs to rapidly screen new chemical designs and prioritize which compounds to synthesize.
    - **Toxicologists and Pharmacologists** to gain initial insights into a compound's potential liabilities before committing to costly and time-consuming *in vitro* or *in vivo* studies.
    - **Educational Institutions** as a teaching tool to help students understand the connection between chemical structure and toxicological outcomes.
- **Changing an Existing Process:** The platform's primary impact is bridging the "interpretability gap" that currently slows down drug discovery.
    - **Accelerating Decision-Making:** Traditionally, interpreting raw ADMET data requires significant expertise and time. Our platform automates this synthesis, allowing a chemist to go from a molecular idea to an expert-level risk assessment in minutes, not days.
    - **Implementing the "Fail Early, Fail Cheap" Paradigm:** By providing clear warnings about potential toxicity at the earliest design stage, the tool empowers

research teams to deprioritize high-risk compounds before investing significant resources. This directly reduces the high attrition rates that plague the industry.
    - **Democratizing Expertise:** The platform encapsulates the reasoning of an expert toxicologist, making high-level insights accessible to researchers who may not have deep expertise in every specific area of ADMET profiling. This is particularly valuable for smaller biotech companies, startups, and academic labs with limited resources.
- **Contribution to the Field and Publication Potential:**
    - **Scientific Contribution:** This project represents a novel application of Large Language Models in the cheminformatics space, demonstrating their utility not just for text generation, but for scientific data synthesis and decision support.
    - **Publication and Dissemination:** Upon completion and validation, the project's methodology and results would be suitable for publication in a peer-reviewed journal focusing on computational chemistry, bioinformatics, or drug discovery (e.g., *Journal of Chemical Information and Modeling*). The platform itself could be presented at relevant scientific conferences. Furthermore, by releasing the code as an open-source project on platforms like GitHub, we can contribute a valuable tool to the global scientific community.