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 ( ) 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:

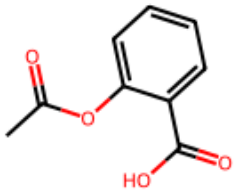
- **Activity** is the dependent variable, often expressed as log(1/C), where C is the concentration required to elicit a specific biological effect (e.g., IC_{50}, LD_{50}).
- **Di** are the molecular descriptors (independent variables), which can be constitutional, topological, quantum-chemical, etc.
- **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.

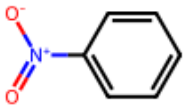
3. System Architecture and Data Flow

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.

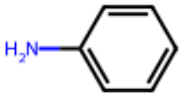
(Intended Image: A 2x2 grid of chemical structures. Top-left: Aspirin, labeled "Example Input". Top-right: Nitrobenzene, labeled "Mutagenicity Alert". Bottom-left: Aniline, labeled "Mutagenicity Alert". Bottom-right: Amiodarone, labeled "hERG Inhibitor".)



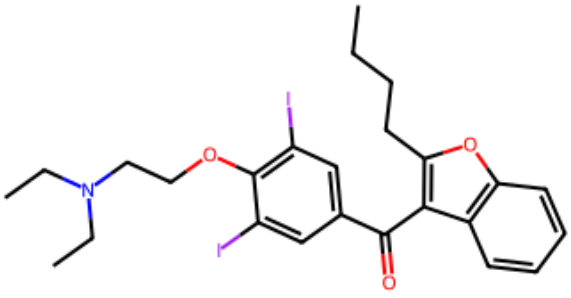
Aspirin (Example Input)



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.

Fill.

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.

>Fill.