INTERNATIONAL UNIVERSITY
VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
School of Computer Science and Engineering
-----*-----**

PROJECT REPORT

AIBIO Website

Web application development (IT093IU)
Semester 1 - Academic year 2024-2025
Course by Dr. Nguyen Van Sinh

Contents

CONTRIBUTION TABLE

No.	Full Name	Student's ID	Task	Contribution
1	Trần Thanh Nguyên(Leader)	ITCSIU21093	Backend, Frontend,DesignSystem	100%
2	Nguyễn Đức Quốc Anh	ITITWE20018	UML Diagram,	90%

ABSTRACT

Artificial Intelligence (AI) is reshaping drug discovery and development by enabling faster, cost-effective, and more precise approaches. Leveraging machine learning, deep learning, and natural language processing, AI analyzes complex biological data to identify drug targets, predict efficacy and toxicity, and design novel compounds. It also accelerates processes like virtual screening and drug repurposing, significantly reducing development timelines.

This project focuses on leveraging Artificial Intelligence (AI) to streamline and enhance various stages of drug development by applying advanced computational techniques in molecular biology. Key components of the project include:

1. **Search:** AI-powered algorithms are employed to navigate and analyze extensive biological and chemical databases efficiently. This enables rapid identification of relevant compounds, biological targets, and associated scientific literature.
2. **Property Prediction:** Machine learning models are utilized to predict key molecular properties such as pharmacokinetics, toxicity, solubility, and binding affinity, helping to prioritize promising drug candidates.
3. **Visualization:** Advanced AI tools generate intuitive visualizations of molecular structures, interactions, and pathways. These visual representations aid researchers in understanding complex biological processes and drug mechanisms.

4. **Molecule Generation:** AI-driven generative models, such as deep generative networks, are used to design novel molecules with desirable therapeutic properties. These models explore vast chemical spaces efficiently, offering innovative drug candidates.
5. **Molecule Description:** Natural language processing (NLP) techniques are applied to annotate and describe molecular structures and their potential functions. This bridges the gap between computational predictions and biological insights.
6. **Document Retrieval:** AI-enhanced search engines retrieve relevant scientific documents, patents, and regulatory guidelines, streamlining the process of knowledge acquisition and ensuring compliance.

By integrating these functionalities, the project aims to accelerate drug discovery and development while reducing costs and enhancing accuracy, ultimately contributing to more effective and personalized therapies.

I INTRODUCTION

A. Objectives

The primary objective of this project is to explore and implement Artificial Intelligence (AI) applications in the field of drug development to address the growing complexities and challenges of modern pharmaceutical research. Specifically, the project aims to achieve the following:

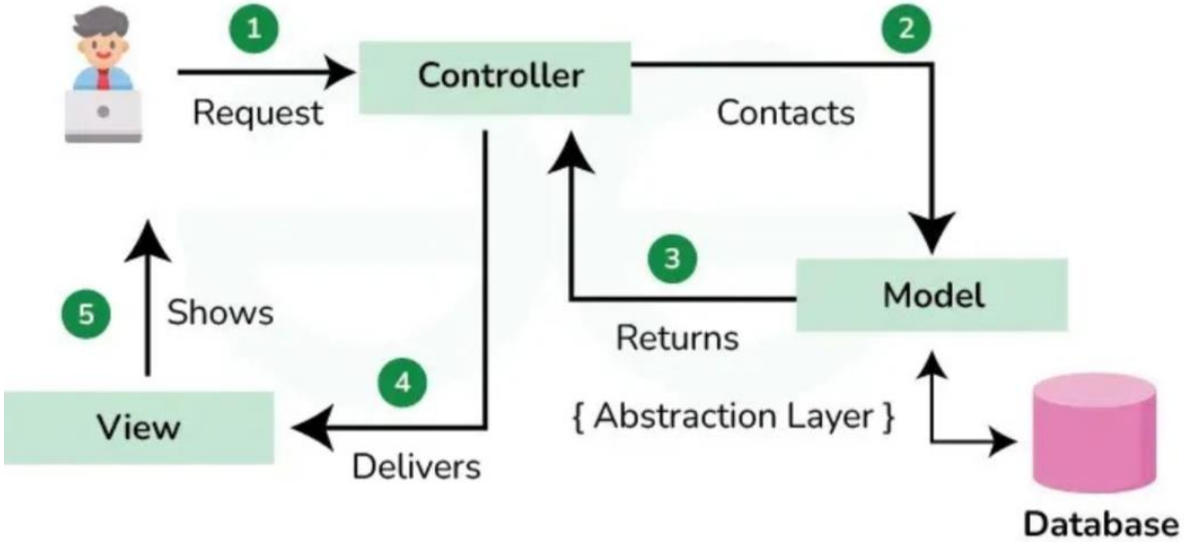
1. Accelerate Drug Discovery Processes: Utilize AI algorithms to analyze large-scale biological and chemical data, identifying novel drug targets and lead compounds more efficiently than traditional methods.
2. Enhance Predictive Accuracy: Develop and validate AI models capable of predicting critical molecular properties, including efficacy, toxicity, and pharmacokinetics, thereby reducing the risk of late-stage failures.
3. Facilitate Molecule Design: Leverage AI-driven generative models to design and optimize new molecular structures with desired therapeutic properties, expanding the scope of chemical space exploration.
4. Improve Data Utilization: Employ natural language processing (NLP) and machine learning techniques to extract, organize, and integrate information from diverse data sources, such as scientific literature, databases, and patents.
5. Enable Visualization and Interpretation: Create intuitive visualizations for molecular interactions, pathways, and other complex biological systems to support decision-making and hypothesis generation.
6. Streamline Regulatory Compliance: Use AI tools to retrieve and analyze regulatory documents and guidelines, ensuring adherence to compliance standards and facilitating smoother approval processes.

7. Promote Interdisciplinary Collaboration: Foster integration between computational biologists, chemists, and AI specialists to drive innovation and maximize the utility of AI in drug development.
8. Reduce Costs and Time: Shorten development timelines and lower financial investments by automating labor-intensive tasks and focusing resources on the most promising drug candidates.

By achieving these objectives, the project aims to transform the landscape of drug development, making it faster, more cost-effective, and increasingly aligned with precision medicine principles.

B. ENVIRONMENT

Since this is a web-based product, the project is conducted using some Web Design and Programming Language under the model of MVC



The implementation of this project relies on a robust combination of programming languages, frameworks, and tools to ensure functionality, scalability, and ease of use. Key tools include:

1. Python Language: A versatile programming language used for data analysis, machine learning model development, and integration of AI-driven applications.
2. SQLAlchemy: A powerful database toolkit and Object Relational Mapper (ORM) for efficient data storage, retrieval, and management.
3. Flask Framework: A lightweight web framework that supports the development of web-based applications and APIs to enable user interaction and data accessibility.
4. HTML: Markup language used to structure and display content on web applications, facilitating a user-friendly interface.
5. CSS: Stylesheet language for designing visually appealing and responsive web pages, enhancing the user experience.

C. DEVELOPMENT PROCESS

Our team has chosen to adapt the agile method and the cycle of machine learning project into our project to ensure flexibility, iterative progress, and continuous improvement. By combining these approaches, we can remain responsive to changing requirements and integrate feedback from stakeholders at every stage.

The agile methodology will allow us to break down the project into manageable tasks, enabling faster delivery of results and easier adjustment of priorities.

Additionally, the machine learning project cycle will guide us through crucial phases such as data collection, preprocessing, model training, and evaluation. This structured approach will help ensure that our models are optimized for performance and can evolve as we gather more data and insights. By iterating through these cycles, we aim to continuously enhance the system's accuracy and effectiveness, ultimately delivering a more robust solution.

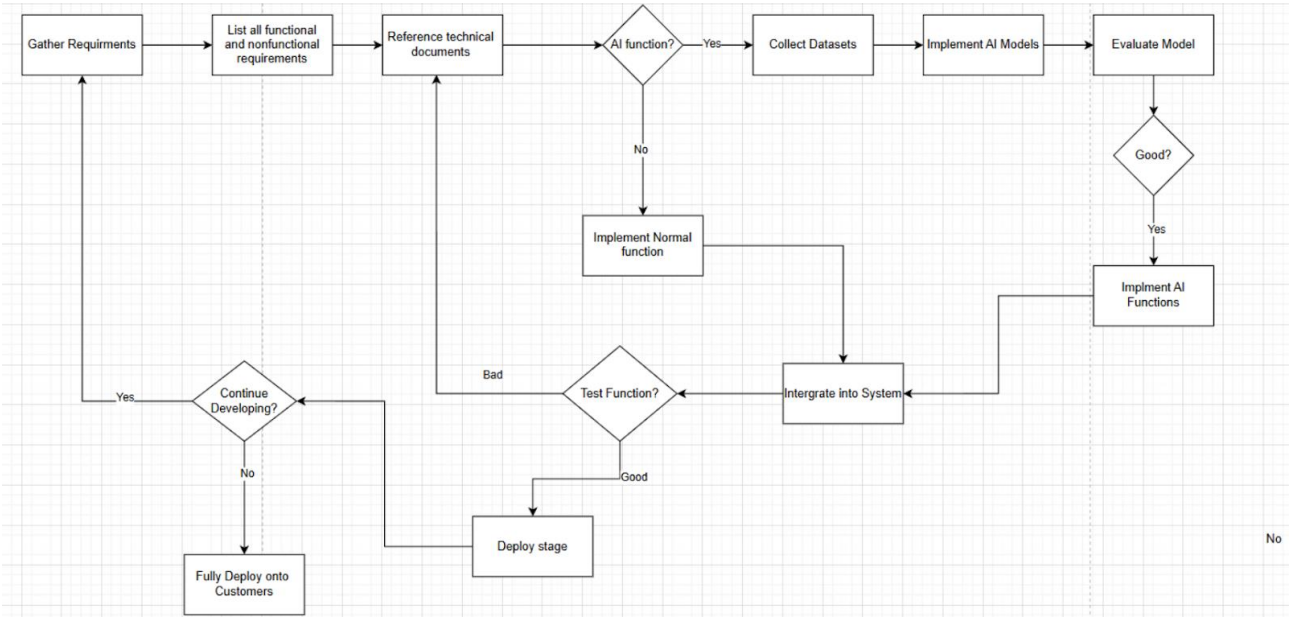


Figure 1.1 System Development Workflow

Project Development Process

Gather Requirements: Identify functional and non-functional requirements for the system.

1. **Reference Technical Documents:** Consult relevant technical documents to ensure alignment with standards and practices.
2. **AI Function?** Determine if the system requires AI functionality.
3. **If Yes (AI Function):**
    - **Collect Datasets:** Gather data for AI training and testing.
    - **Implement AI Models:** Develop and train AI models.
    - **Evaluate Model:** Assess the AI model's performance.

4. **If No (No AI Function):** Develop the system's core functionality without AI components.
5. **Test Function:** Thoroughly test the system's functionality.
6. **Integrate into System:** Integrate tested components (AI or non-AI) into the overall system.
7. **Deploy Stage:** Deploy the system in a controlled environment for validation.
8. **Fully Deploy to Customers:** If satisfactory, fully deploy the system to customers.

Additional Notes:

- The process is iterative, with a decision point to continue development.
- If testing fails, the system may need rework.
- If the AI model is deemed suitable, it will be integrated.

D. CUSOMER REQUIREMENTS

1. Functional:

User Management:

- Allow user registration with secure authentication.
- Provide login/logout functionality.
- Role-based access control to ensure appropriate access for researchers, administrators, and other stakeholders.

Data Integration and Management:

- Support for uploading, storing, and managing large datasets (e.g., chemical, biological, and pharmaceutical data).
- Data validation mechanisms to ensure input data quality and consistency.
- Enable efficient querying of the database for specific molecule properties, interactions, or datasets.

AI-Driven Functionality:

- Develop and deploy AI models for predicting molecular properties such as efficacy, toxicity, and pharmacokinetics.
- Implement generative models to design new molecular structures based on user-defined criteria.
- Include natural language processing (NLP) for extracting and processing information from scientific literature and regulatory documents.

Visualization Tools:

- Provide tools for visualizing molecular structures, pathways, and interactions.
- Allow users to view predictive outcomes and molecular designs through dynamic and interactive interfaces.

Compliance Tools:

- Automate compliance checking with regulatory standards using AI-based analysis.
- Generate reports and summaries to support regulatory approval processes.

Collaboration Features:

- Enable team collaboration through shared data access and comment functionality.
- Provide version control for datasets and analysis results.

System Feedback:

- Notify users of errors, such as invalid inputs or system failures, with clear and actionable messages.
- Provide progress updates for long-running tasks, such as training AI models or processing large datasets.

2. Non-Functional

Performance:

- Ensure the system processes AI-driven predictions within a maximum of 45 seconds per query.
- Allow high throughput for handling multiple users and data queries simultaneously.

Scalability:

- Support an increasing number of users, datasets, and AI models without significant degradation in performance.
- Design the system to scale both vertically (adding more resources) and horizontally (distributing workloads).

Reliability:

- Provide a fault-tolerant infrastructure to minimize downtime.
- Ensure database backups are conducted regularly to prevent data loss.

Security:

- Implement end-to-end encryption for data in transit and at rest.
- Securely store sensitive user data, such as passwords and confidential research findings, using robust encryption methods.

Usability:

- Ensure the user interface is intuitive, with minimal learning curve for researchers.
- Provide detailed documentation and training resources for system use.

Accessibility:

- Ensure compliance with accessibility standards (e.g., WCAG) for users with disabilities.
- Make the system available on various devices, including desktops, tablets, and mobile phones.

Interoperability:

- Integrate with existing pharmaceutical and research tools, such as chemical libraries, modeling software, and external APIs.
- Ensure compatibility with common data formats (e.g., CSV).

Maintainability:

- Develop modular components to simplify updates and debugging.
- Provide an automated testing framework to ensure system stability during development cycles.

Ethical Compliance:

- Ensure that AI models are explainable and their predictions can be interpreted by researchers.
- Avoid biases in data processing and AI predictions to ensure fairness and accuracy.

Deployment:

- Enable smooth deployment to both development and production environments.
- Ensure that updates and upgrades are implemented with minimal disruption to users.

II. REQUIREMENT ANALYSIS AND DESIGN

This section provides an overview of the requirement analysis and design process, outlining the approach for the future implementation of the project. Based on the requirements specification, we will implement each function, addressing both the conditions and the functional and non-functional requirements provided by the customers. Throughout the implementation phase, we will continually revise and update the system to ensure we stay aligned with project progress and maintain up-to-date versions.

1. REQUIREMENT ANALYSIS

USE CASE DIAGRAM

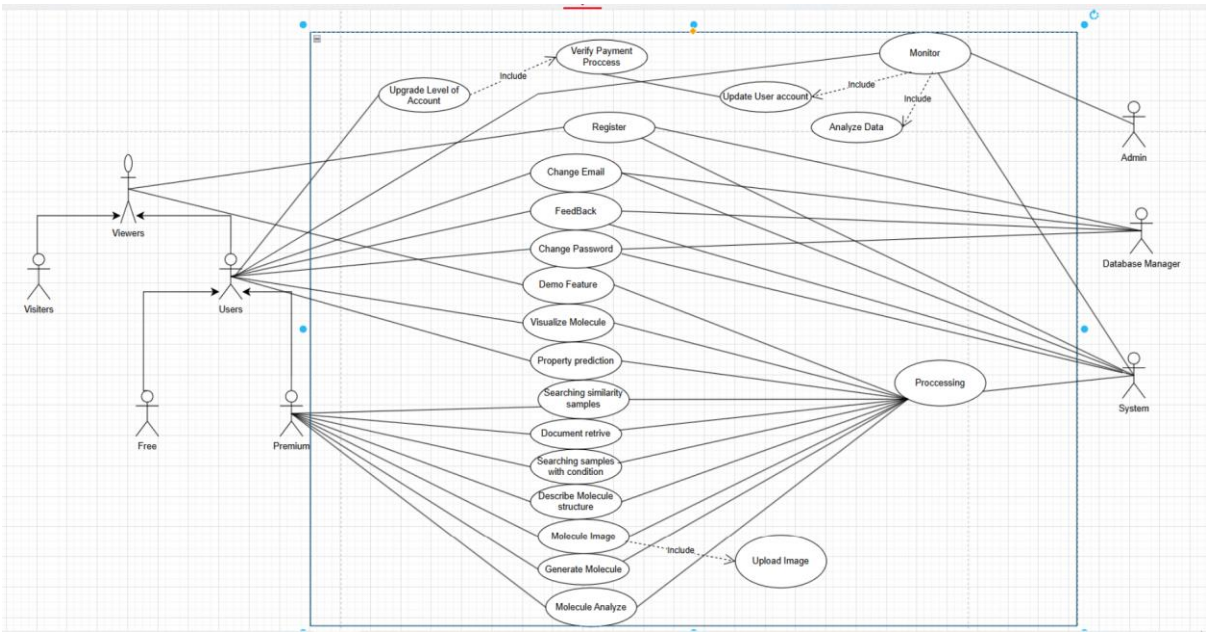


Figure 1.2 USE CASE DIAGRAM

Use Case 1: Register

- Primary Actor: Visiters
- Secondary Actors: Database Manager,System
- Identifier: UC1
- Name: Register

Inputs:

User's registration details, including:

1. User ID
2. Full name
3. Email address
4. Password (and confirmation)

Outputs:

A new user account is created in the database.

Users can access to User main page

Basic Flow:

1. Visitor accesses the Register page.
   1.1. The system displays a registration form with required fields.

2. Visitor fills out the form with valid details.
   2.1. The system validates the inputs for correctness (e.g., email format, password match).
3. Visitor submits the registration form.
   3.1. System checks if the User ID is unique.
   3.2. Database Manager creates a new account and stores it in the database.

Alternative Flows:

- Invalid Input:
  1. Visitor: Enters invalid or incomplete information.
  2. System: Displays an error message indicating the specific issue (e.g., "Invalid email format", "Passwords do not match").
- Duplicate User ID:
  1. Visitor: Attempts to register with an existing User ID.
  2. System: Displays an error message indicating the User ID is already taken.

Preconditions:

- The User ID not already exist .
- The user accesses the registration page.

Postconditions:

- A new account is successfully created and stored in the database.
- The user can now log in to the system.

Use Case 2: Login

- Identifier: UC2
- Name: Login
- Primary Actor: User
- Secondary Actors: Database Manager,System

Basic Flow:

1. User: Accesses the login page.
2. User: Enters valid login credentials.
3. System: Validates the input.
4. Database Manager: Checks the credentials against the database.
5. System: If the credentials are valid:
   Notify to system to generate a session token to track Users.
   Redirects the user to the main page.

Preconditions:

- The user has a registered account.

- The login page is accessible.

Postconditions:

- The user is successfully authenticated and logged in.
- The user is redirected to the appropriate main page.

Alternative Flow:

Invalid Credentials:

- User: Enters invalid login credentials (e.g., wrong User ID or password).
- System: Displays an error message indicating the login failed due to invalid credentials.

Use Case 3: Predict Molecule Properties

- Identifier: UC3
- Name: Predict Molecule Properties
- Primary Actor: Member Researchers
- Secondary Actor: System

Preconditions:

- The user has access to the main page.
- The prediction model is deployed and functional.

Postconditions:

- The user receives the predicted molecule properties.
- An error message is displayed if the prediction fails or the input is invalid.

Basic Flow:

1. User: Clicks the "Predict" button on user main page.
2. System: Move user to Predict page.
3. User: Enters the molecule representation (e.g., SMILES).
4. System: Processes the input and predicts the molecule properties.
5. System: Return the result to Predict page

Alternative Flows:

Invalid Input:

- User: Enters an invalid molecule representation.
- System: Displays an error message indicating the input is invalid.

Prediction Failure:

- System: Fails to predict properties due to technical issues or model limitations.
- System: Displays an error message indicating the prediction failed.

Preconditions:

- The user has access to the main page.

- The prediction model is deployed and functional.

Postconditions:

- The user receives the predicted molecule properties.
- An error message is displayed if the prediction fails or the input is invalid.

Use Case 4: Visualize Molecule

- Identifier: UC4
- Name: Visualize Molecule
- Primary Actor: User
- Secondary Actor: System

Preconditions:

- The user has access to the main page.
- The visualization model is deployed and functional.

Postconditions:

- The user receives the visualized molecule image.
- An error message is displayed if the visualization fails or the input is invalid.

Basic Flow:

1. User: Selects the "Visualize" function on the main page.
2. System: Redirected to Visualize page.
3. User: Enters the molecule representation (e.g., SMILES).
4. User Submits the input.
5. System: Converts the molecule representation into an image.

Alternative Flows:

-Invalid Input:

- User: Enters an invalid molecule representation (e.g., incorrect format).
- System: Displays an error message indicating the input is invalid.

-Visualization Failure:

- System: Fails to convert the molecule into an image due to technical issues or model limitations.
- System: Displays an error message indicating the visualization failed.

Preconditions:

- The user has access to the main page.
- The visualization model is deployed and functional.

Postconditions:

- The user receives the visualized molecule image.
- An error message is displayed if the visualization fails or the input is invalid.

Use Case 5: Generate Molecule

- Identifier: UC5
- Name: Generate Molecule
- Primary Actor: User
- Secondary Actor: System

Basic Flow:

1. User: Clicks the "Generate" button on the user main page
2. System: Checks the User 's access level.
3. System: If User has sufficient access:
    - Renders User to the molecule generation page.
4. User: Enters the desired molecule description.
5. User: Submits the input.
6. System: Processes the molecule description and generates the molecule.
7. System: Displays the generated molecule to the user.

Alternative Flows:

- Insufficient Level Access:
    1. System: Denies access and displays an error message to the User.
- Invalid Input:
    1. User: Enters an invalid molecule description.
    2. System: Displays an error message indicating the input is invalid.
- Generation Failure:
    1. System: Fails to generate the molecule due to technical issues or model limitations.
    2. System: Displays an error message indicating the generation failed.

Preconditions:

- The user has access to the main page.
- The molecule generation model is deployed and functional.

Postconditions:

- The user receives the generated molecule.
- An error message is displayed if the generation fails or the input is invalid.

Use Case 6: Describe Molecule

- Identifier: UC6

- Name: Describe Molecule
- Primary Actor: User
- Secondary Actor: System

Basic Flow:

1. User: Accesses the main page.
2. User: Clicks the "Describe" button.
3. System: Checks the User 's access level.
4. System: If the User has sufficient access:
    - Renders user to the molecule description page.
5. System: Enters the molecule representation (e.g., SMILES).
6. User: Submits the input.
7. System: Processes the molecule representation and generates a description.
8. System: Displays the generated description to the User.

Alternative Flows:

- Insufficient Level Access:
    1. System: Denies access and displays an error message to the User.
- Invalid Input:
    1. User: Enters an invalid molecule representation.
    2. System: Displays an error message indicating the input is invalid.
- Description Failure:
    1. User: Fails to generate a description due to technical issues or model limitations.
    2. User: Displays an error message indicating the description failed.

Preconditions:

- The User has access to the main page.
- The description model is deployed and functional.

Postconditions:

- The User receives the generated molecule description.
- An error message is displayed if the description fails or the input is invalid.

Use Case 7: Describe Molecule from Image

- Identifier: UC7
- Name: Describe Molecule from Image
- Primary Actor: User

- Secondary Actor: System

Basic Flow:

1. User: Accesses the main page.
2. User: Clicks the "Upload Molecule Image" button.
3. System: Checks the User 's access level.
4. System: If the User has sufficient access:
    - Renders User to the image upload page.
5. User: Uploads the molecule image.
6. System: Processes the uploaded image to extract molecule information.
7. System: Generates a description of the molecule.
8. System: Displays the generated description to the user.

Alternative Flows:

- Insufficient Level Access:
    1. System: Denies access and displays an error message to the user.
- Invalid Image:
    1. User: Uploads an invalid or unsupported image format.
    2. System: Displays an error message indicating the image is invalid.
- Description Failure:
    1. User: Fails to generate a description due to technical issues or model limitations.
    2. System: Displays an error message indicating the description failed.

Preconditions:

- The user has access to the main page.
- The image processing and description models are deployed and functional.

Postconditions:

- The user receives the generated molecule description.
- An error message is displayed if the image processing or description generation fails or the input is invalid.

Use Case 8: Search Similar Molecules

- Identifier: UC8
- Name: Search Similar Molecules
- Primary Actor: User
- Secondary Actor: System

Basic Flow:

1. User: Accesses the main page.
2. System: Clicks the "Search" button.
3. User: Checks the user's access level.
4. System: If the user has sufficient access:
    - Renders user to the search page.
5. User: Enters the molecule representation (e.g., SMILES).
6. System: Submits the input.
7. User: Searches for similar molecules based on the input molecule.
8. System: Displays the search results to the user.

Alternative Flows:

- Insufficient Level Access:
    1. System: Denies access and displays an error message to the user.
- Invalid Input:
    1. User: Enters an invalid molecule representation.
    2. System: Displays an error message indicating the input is invalid.

Preconditions:

- The user has access to the main page.
- The similarity search model is deployed and functional.

Postconditions:

- The user receives the search results for similar molecules.
- An error message is displayed if the search fails or the input is invalid.

Use Case 9: Conditional Search for Molecules

- Identifier: UC9
- Name: Conditional Search for Molecules
- Primary Actor: User
- Secondary Actor: System

Basic Flow:

1. User: Accesses the main page.
2. Models: Clicks the "Conditional Search" button.
3. User: Checks the user's access level.
4. System: If the user has sufficient access:
    - Renders to the conditional search page.
5. User: Sets up search conditions (e.g., molecular properties, substructures).
6. User: Submits to the search conditions.
7. System: Processes the search conditions and searches for matching molecules.

8. System: Displays the search results to the user.

Alternative Flows:

- Insufficient Level Access:
    1. System: Denies access and displays an error message to User.
- Invalid Conditions:
    1. User: Enters invalid or inconsistent search conditions.
    2. System: Displays an error message indicating the conditions are invalid.
- Search Failure:
    1. System: Fails to find matching molecules due to technical issues or model limitations.
    2. System: Displays an error message indicating the search failed.

Preconditions:

- The user has access to the main page.
- The search engine is deployed and functional.

Postconditions:

- The user receives the search results for molecules matching the specified conditions.
- An error message is displayed if the search fails or the input is invalid.

Use Case 10: Document Retrieval

- Identifier: UC10
- Name: Document Retrieval
- Primary Actor: User
- Secondary Actor: System

Basic Flow:

1. User: Accesses the main page.
2. User: Clicks the "Document Retrieve" button.
3. System: Checks the user's access level.
4. System: If the user has sufficient access:
    - Renders the document retrieval page.
5. User: Enters the search keywords or topics.
6. System: Submits the input.
7. System: Sends the search query to the ARIV API.
8. ARIV API: Processes the search query and retrieves relevant documents.
9. ARIV API: Returns the search results to the System.
10. System: Displays the search results to the user.

Alternative Flows:

- Insufficient Level Access:
    1. System: Denies access and displays an error message to the user.
- Invalid Input:
    1. User: Enters invalid or incomplete search keywords or topics.
    2. System: Displays an error message indicating the input is invalid.
- Search Failure:
    1. ARIV API: Fails to find relevant documents due to technical issues or database limitations.
    2. System: Displays an error message indicating the search failed.

Preconditions:

- The user has access to the main page.
- The ARIV API is accessible and functional.

Postconditions:

- The user receives the search results for relevant documents.
- An error message is displayed if the search fails or the input is invalid.

Use Case 11: Molecule Analysis

- Identifier: UC11
- Name: Molecule Analysis
- Primary Actor: User
- Secondary Actor: System

Basic Flow:

1. User: Clicks the "Analyze" button on the main page.
2. System: Checks the user's access level.
3. System: If the user has sufficient access:
    - Renders the molecule analysis page.
4. User: Enters the molecule representation (e.g., SMILES).
5. User: Submits the input.
6. System: Processes the molecule representation and performs analysis.
7. System: Displays the analysis results to the user.

Alternative Flows:

- Insufficient Level Access:
    1. System: Denies access and displays an error message to the User.
- Invalid Input:

1. User: Enters an invalid molecule representation.
2. System: Displays an error message indicating the input is invalid.

- Analysis Failure:
    1. System: Fails to analyze the molecule due to technical issues or model limitations.
    2. System: Displays an error message indicating the analysis failed.

Preconditions:

- The User has access to the main page.
- The analysis models are deployed and functional.

Postconditions:

- The User receives the analysis results for the molecule.
- An error message is displayed if the analysis fails or the input is invalid.

Use Case 12: Change Email

- Identifier: UC12
- Name: Change Email
- Primary Actor: User
- Secondary Actor: Database Manager, System

Basic Flow:

1. User: Accesses the main page.
2. User: Clicks the "Change Email" button.
3. System: Redirects the user to the change email page.
4. User: Fills in the change email form.
5. User: Submits the form.
6. System: Processes the change email request.
7. System:Request Database Manager to update
8. Database Manager: Updates the user's email address in the database.
9. System: Notifies the user of the successful change.

Alternative Flows:

- Invalid Email:
    1. User: Enters an invalid email address.
    2. System: Displays an error message indicating the email is invalid.
- Email Update Failure:
    1. User: Fails to update the user's email address in the database.

2. System: Displays an error message indicating the update failed.

Preconditions:

- The User has an existing account.
- The User is logged in.

Postconditions:

- The User 's email address is successfully updated.
- The User is notified of the successful change.
- An error message is displayed if the update fails or the input is invalid.

Use Case 13: Change Password

- Identifier: UC13
- Name: Change Password
- Primary Actor: User
- Secondary Actor: System, Database Manager

Basic Flow:

1. User: Accesses the main page.
2. User: Clicks the "Change Email" button.
3. System: Redirects the user to the change email page.
4. User: Fills in the change password form.
5. User: Submits the form.
6. System: Processes the change password request.
7. System:Request Database Manager to update
8. Database Manager: Updates the user's password in the database.
9. System: Notifies the user of the successful change.

Alternative Flows:

- Invalid Current Password:
    1. Member researcher: Enters an incorrect current password.
    2. System: Displays an error message indicating the current password is incorrect.
- Password Update Failure:
    1. System: Fails to update the user's password in the database.
    2. System: Displays an error message indicating the update failed.

Preconditions:

- The user has an existing account.
- The user is logged in.

Postconditions:

- The user 's password is successfully updated.
- The user is notified of the successful change.
- An error message is displayed if the update fails or the input is invalid.

Use Case 14: Update Account

Identifier: UC14

Name: Update Account

Primary Actor: User

Secondary Actor: System,Database Manager

Basic Flow:

1. User: Accesses the main page.
2. User: Clicks the "Update" button.
3. System: Redirects the user to the update profile page.
4. User: Selects the information to update (e.g., profile picture, personal information).
5. System: Redirects the User to the appropriate update page.
6. User: Updates the selected information.
7. System: Processes the update request.
8. Database Manager: Updates the user 's information in the database.
9. System: Notifies the User of the successful update.

Alternative Flows:

- Invalid Input:
    1. User: Enters invalid or incomplete information.
    2. System: Displays an error message indicating the input is invalid.
- Update Failure:
    1. System: Fails to update the user's information in the database.
    2. System: Displays an error message indicating the update failed.

Preconditions:

- The user has an existing account.

- The user is logged in.

Postconditions:

- The user 's information is successfully updated.
- The user is able to access higher level in case of successful update.
- An error message is displayed if the update fails or the input is invalid.

Use Case 15: Provide Feedback

- Identifier: UC15
- Name: Provide Feedback
- Primary Actor: User
- Secondary Actor: System, Database Manager

Basic Flow:

1. User: Clicks the "Feedback" button on the user main page.
2. System: Redirects researcher to the feedback page.
3. User: Fills in the feedback form.
4. User: Submits the feedback.
5. System: Processes the feedback.
6. Database Manager: Stores the feedback in the database.
7. Database Manager: Notifies the user of the successful submission.

Alternative Flows:

- Invalid Input:
    1. User: Enters invalid or incomplete feedback.
    2. System: Displays an error message indicating the input is invalid.
- Feedback Submission Failure:
    1. Database Manager: Fails to store the feedback in the database.
    2. System: Displays an error message indicating the submission failed.

Preconditions:

- The user has an account.
- The user is logged in.

Postconditions:

- The user 's feedback is successfully stored.
- The user is notified of the successful submission.
- An error message is displayed if the submission fails or the input is invalid.

Use Case 16: Admin Manage User Information

- Identifier: UC16
- Name: Admin Manage User Information
- Primary Actor: Admin
- Secondary Actor: System, Database

Basic Flow:

1. System: Retrieves all user information from the database.
2. System: Displays the admin's information to the admin.
3. Admin: Selects a admin to view or edit.
4. System: Displays the selected researcher 's details.
5. Admin: Makes changes to the researcher 's information (e.g., update role, delete account).
6. System: Processes the changes and updates the database.
7. System: Notifies the admin of the successful changes.

Alternative Flows:

- Insufficient Permissions:
    1. Admin: Attempts to perform an action without sufficient permissions.
    2. System: Denies access and displays an error message.
- Database Error:
    1. System: Fails to retrieve or update user information from the database.
    2. System: Displays an error message indicating the database operation failed.

Preconditions:

- The admin is logged in.
- The admin has sufficient permissions to manage users.

Postconditions:

- Researcher's information is displayed to the admin.
- Changes to researcher information are successfully applied.
- Error messages are displayed for failed operations.

Use Case 17: Admin Monitor Model Records

- Identifier: UC17
- Name: Admin Monitor Model Records
- Primary Actor: Admin

- Secondary Actor: System, Database Manager

Basic Flow:

1. Admin: Accesses the admin page.
2. Admin: Clicks the "Manage Models" button.
3. System: Queries the database for all model records.
4. Database Manager: Returns the model records to the system.
5. System: Displays the model records to the admin.
6. Admin: Selects a model record to view or edit.
7. System: Displays the details of the selected model record.

Alternative Flows:

- Insufficient Permissions:
    1. Admin: Attempts to perform an action without sufficient permissions.
    2. System: Denies access and displays an error message.
- Database Error:
    1. Database Manager: Fails to retrieve or update model records from the database.
    2. Database Manager: Displays an error message indicating the database operation failed.

Preconditions:

- The admin is logged in.
- The admin has sufficient permissions to manage models.

Postconditions:

- Model records are displayed to the admin.
- Error messages are displayed for failed operations.

Use Case 18: Admin Monitor User Experience

- Identifier: UC18
- Name: Admin Monitor User Experience
- Primary Actor: Admin
- Secondary Actor: System, Database Manager

Basic Flow:

1. Admin: Accesses the admin page.
2. Admin: Clicks the "Manage User Experiment" button.
3. System: Queries the database for researcher’s experiment data.

4. Database Manager: Returns the researcher’s experiment data to the system.
5. System: Displays the researcher’s experiment data to the admin.
6. Admin: Selects a researcher’s experiment to view or analyze.
7. System: Displays the details of the selected researcher’s experiment.

Alternative Flows:

- Insufficient Permissions:
    1. Admin: Attempts to perform an action without sufficient permissions.
    2. System: Denies access and displays an error message.
- Database Error:
    1. Database Manager: Fails to retrieve researcher’s experiment data from the database.
    2. Database Manager: Displays an error message indicating the database operation failed.

Preconditions:

- The admin is logged in.
- The admin has sufficient permissions to access user experiment data.

Postconditions:

- User’s experiment data is displayed to the admin.
- The admin can analyze and interpret the data.
- Error messages are displayed for failed operations.

Use Case 19: Admin Update User Level

- Identifier: UC19
- Name: Admin Update User Level
- Primary Actor: Admin
- Secondary Actor: System, Database Manager

Basic Flow:

1. Admin: Accesses the Admin page.
2. Admin: Clicks the "Manage User" button.
3. System: Retrieves all user data from the database.
4. System: Displays the user data to the Admin.
5. Admin: Clicks the "Update User" button for a specific user.
6. System: Displays the selected user's details.
7. Admin: Enters the new user level.
8. Database Manager: Updates the user's level in the database.
9. System: Notifies the Admin of the successful update.

Alternative Flows:

- Insufficient Permissions:
    1. Admin: Attempts to update a user level without sufficient privileges.
    2. System: Displays an error message and denies the action.
- Database Error:
    1. Database Manager: Fails to retrieve or update user data from the database.
    2. Database Manager: Displays an error message indicating the database operation failed.
- Invalid Input:
    1. Admin: Enters invalid user level information.
    2. System: Displays an error message indicating invalid input.

Preconditions:

- The Admin is logged in.
- The Admin has sufficient permissions to update user levels.

Postconditions:

- The user's level is successfully updated in the database.
- The Admin is notified of the successful update.
- Error messages are displayed for failed operations.

Use Case 20: Analyze User Data

- Identifier: UC20
- Name: Analyze User Data
- Primary Actor: Admin
- Secondary Actors: System, Database Manager

Basic Flow:

1. Admin: Accesses the Admin page.
2. Admin: Clicks the "Manage Users" button.
3. System: Redirects the Admin to the User Management page.
4. Admin: Clicks the "Analyze Data" button.
5. System: Requests user data from the Database.
6. Database Manager: Retrieves the requested user data.
7. Database Manager: Returns the user data to the System.
8. System: Processes the retrieved user data.
9. System: Displays the analyzed data (e.g., reports, charts) to the Admin.

Alternative Flows:

- Insufficient Permissions:
    1. Admin: Attempts to analyze user data without sufficient privileges.
    2. System: Displays an error message and denies the action.
- Database Error:
    1. Database Manager: Fails to retrieve or process user data from the database.
    2. Database Manager: Displays an error message indicating the database operation failed.
- Data Processing Error:
    1. System: Fails to process the retrieved user data due to errors in the data or the analysis algorithm.
    2. System: Displays an error message indicating the data processing failed.

Preconditions:

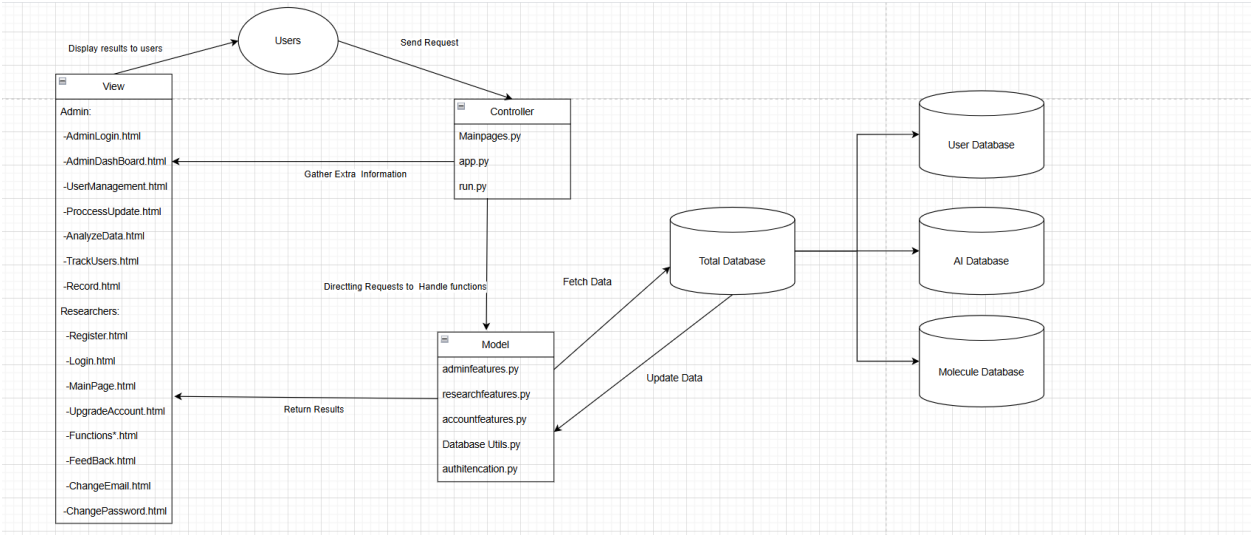
- The Admin is logged in.
- The Admin has sufficient permissions to analyze user data.

Postconditions:

- The system successfully analyzes the user data.
- The analyzed data is displayed to the Admin.
- Error messages are displayed for failed operations.

2. DESIGN

1. System Architecture

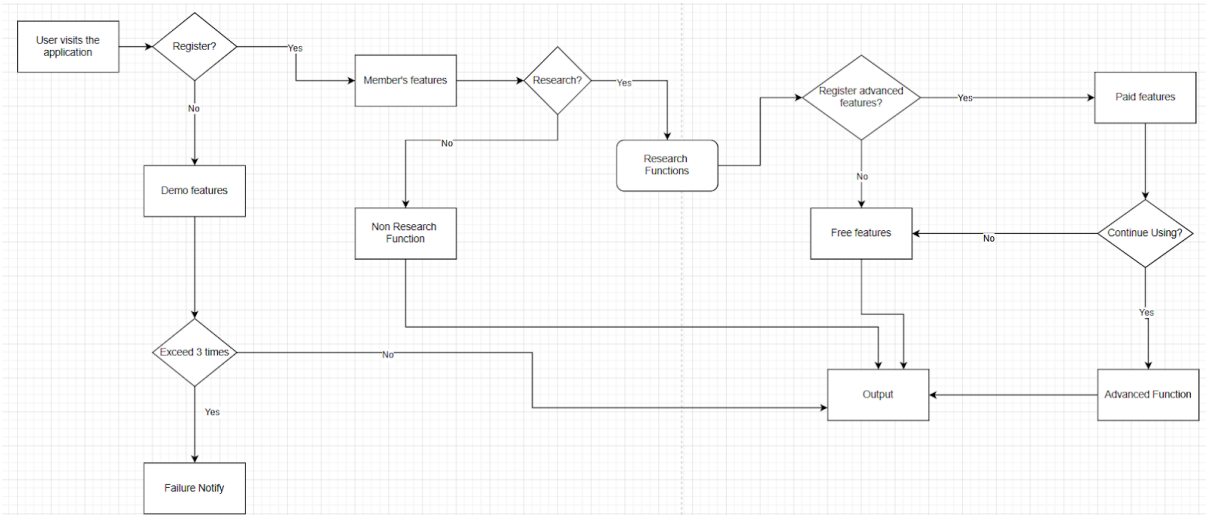


The system is designed using a Model-View-Controller (MVC) architecture, which effectively separates the application into three distinct components, ensuring improved modularity and scalability:

- **Users**: End-users, including administrators and researchers, interact with the system through the views. Administrators are responsible for managing the system, while researchers and other users access their accounts and provide feedback.
- **Views**: The views component is responsible for the user interface, displaying relevant information and collecting input from users. It communicates with the model to process user requests.
- **Controller**: Serving as an intermediary between the views and models, the controller handles user requests, processes them, and updates the views with the appropriate data.
- **Model**: The model manages the application's logic and interacts with the database to retrieve, update, and validate data. Key models include:
    - adminfeatures.py
    - researchfeatures.py
    - accountfeatures.py
    - databasemanager.py
- **Database**: The system uses three integrated databases to store persistent data: User, AI, and Molecule databases. These databases are accessed through a central Total Database, which ensures data consistency and integrity.

2.Work Flow Diagram

2.1 User Flow Diagram:

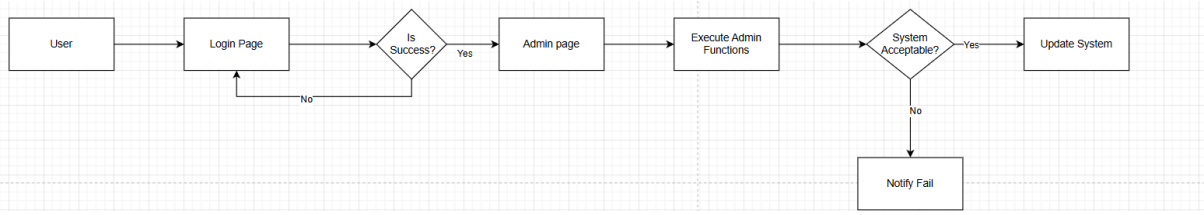


1. **Start: User Visits the Application**:
    - The flow begins when a user accesses the application.
2. **Decision: Register?**:
    - **Yes**: The user registers and gains access to **Member's Features**.
    - **No**: The user is directed to **Demo Features**.
3. **Demo Features**:
    - Users can explore the application with limited functionality.
    - **Limit**: Users are allowed up to 3 attempts. If they exceed this limit:
        - They receive a **Failure Notification**, likely prompting them to register or stop using the application.
4. **Member's Features**:
    - Registered users are directed to the next decision: **Research?**
    - This decision determines whether the user wants to use features related to research.
5. **Decision: Research?**:
    - **Yes**: The user accesses **Research Functions**, tailored for users requiring research-specific tools.
    - **No**: The user accesses **Non-Research Functions**, which include general features available to all registered users.
6. **Advanced Feature Options**:
    - Users in both research and non-research functions are prompted with the option to **Register for Advanced Features**.
        - **Yes**: They gain access to **Paid Features** (premium services or tools).
        - **No**: They remain limited to **Free Features**.
7. **Paid Features**:
    - Users with access to paid features are provided the option to **Continue Using** these features.
        - **Yes**: They continue with the **Advanced Functionality**.

- **No**: They are directed to the **Output** stage or default features.

8. **Output**:
    - Regardless of the pathway (demo, free, or advanced), the system generates an output based on the features accessed.

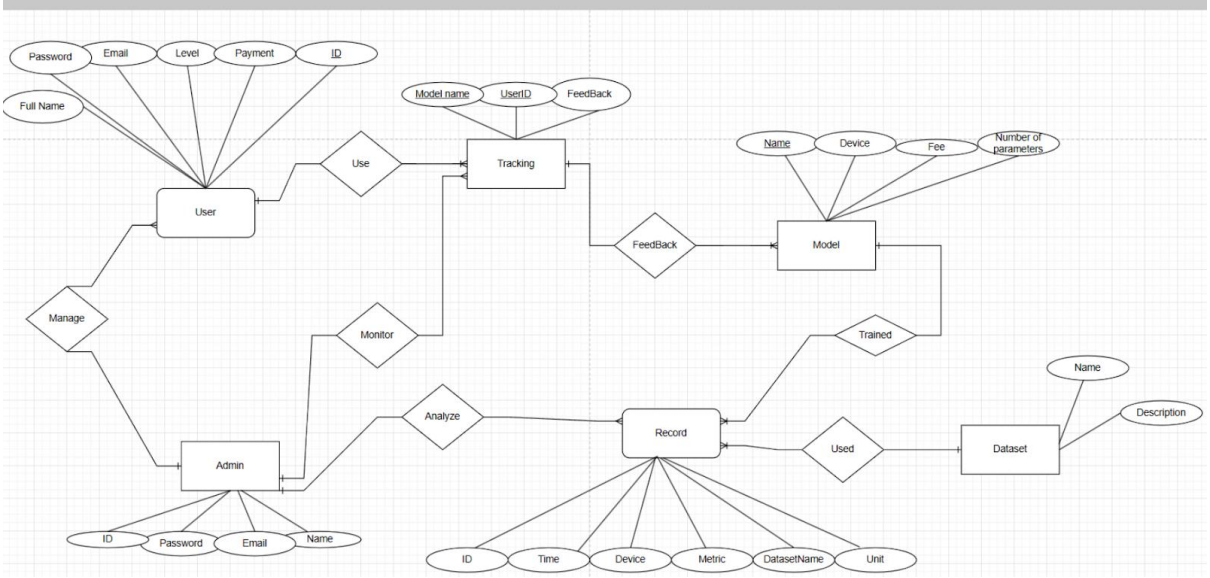
2.2 Admin Workflow



1. **User Access**:
    - The process begins when an admin (or a user with administrative rights) initiates access to the system.
2. **Login Page**:
    - The admin is directed to a **Login Page**, where they must enter their credentials to proceed.
3. **Decision: Is Success?**:
    - The system validates the login attempt:
        - **Yes**: Successful login redirects the admin to the **Admin Page**.
        - **No**: The admin is required to retry the login process.
4. **Admin Page**:
    - Once logged in, the admin gains access to the **Admin Page**, which serves as the control center for administrative actions.
5. **Execute Admin Functions**:
    - From the Admin Page, the admin can perform various **Admin Functions**. These could include:
        - Managing user accounts.
        - Monitoring system metrics.
        - Reviewing logs or analytics.
        - Making configuration changes.
6. **Decision: System Acceptable?**:
    - After performing their administrative functions, the admin evaluates the system's status:
        - **Yes**: If the system meets requirements or functions as expected, the admin proceeds to **Update System** if necessary.
        - **No**: If the system is not acceptable (e.g., errors or issues are detected), a **Notify Fail** action is triggered.
7. **Update System**:
    - The admin updates the system to address pending tasks, implement changes, or apply fixes.
8. **Notify Fail**:

- If the system status is unacceptable, a notification is sent (e.g., via email, logs, or alerts) to inform relevant parties of the failure or issue. Further actions may need to be initiated to resolve the problem.

3 Entity-Relationship Diagram (ERD)



Entities and Attributes:

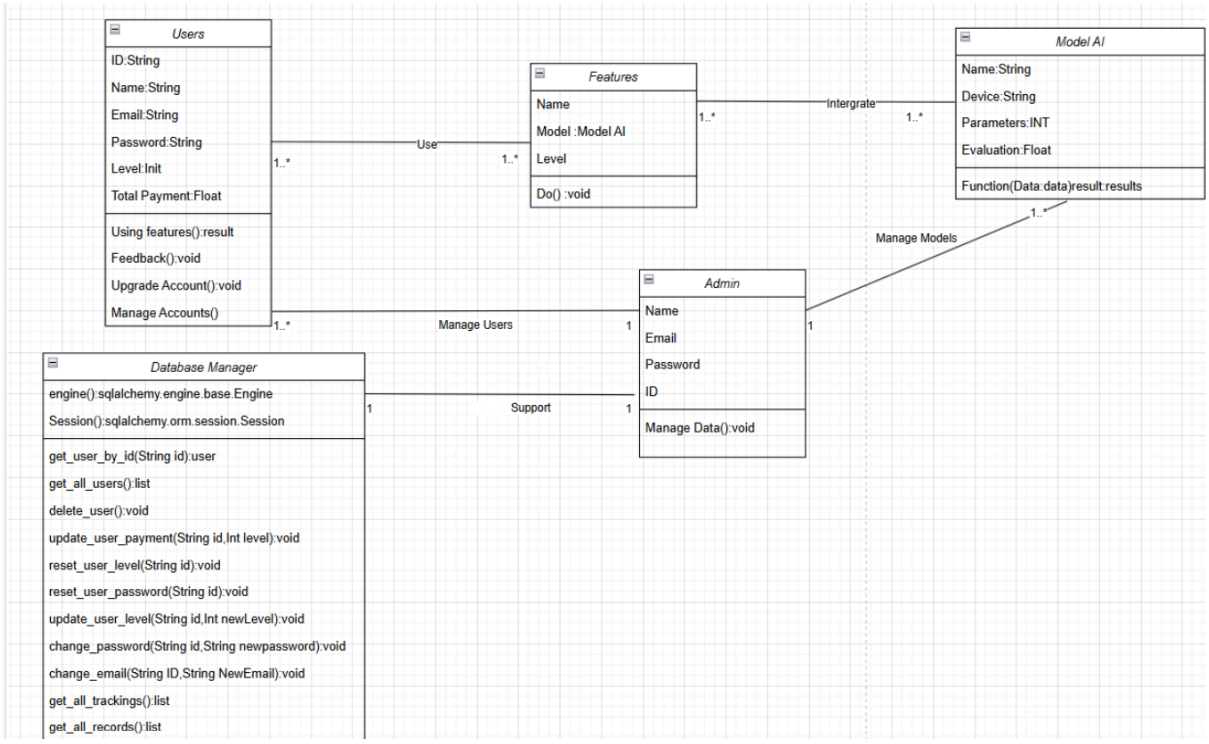
1. **User**:
   - **Attributes**: Full Name, Password, Email, Level, Payment, ID
   - Represents individuals using the system. Interacts with models, and their activities are tracked by admins.
2. **Admin**:
   - **Attributes**: ID, Password, Email, Name
   - Manages users and oversees their activities.
3. **Tracking**:
   - **Attributes**: Model Name, UserID, Feedback
   - Logs user interactions with models, including feedback.
4. **Model**:
   - **Attributes**: Name, Device, Fee, Number of Parameters
   - Represents computational models used for training or monitoring.
5. **Record**:
   - **Attributes**: ID, Time, Device, Metric, Dataset Name, Unit
   - Logs activities, metrics, and datasets in the system.
6. **Dataset**:
   - **Attributes**: Name, Description
   - Datasets used to train or evaluate models.

Relationships and Interactions:

1. **User and Tracking**: Users' interactions with the system and feedback on models are logged in the Tracking entity.
2. **Admin and User**: Admins manage users and control their activities.
3. **Admin and Monitoring**: Admins monitor user actions, models, and records for performance and error detection.
4. **Model and Feedback**: Models are improved based on feedback from Tracking.
5. **Model and Dataset**: Models are trained with datasets.
6. **Record and Analyze**: Records are analyzed for system performance or user behavior.
7. **Dataset and Record**: Datasets are linked to records that log their usage and evaluation metrics.

4 Class Diagram

4.1 Overview System



1. **Users Class**

Attributes:

- ID (String), Name (String), Email (String), Password (String), Level (Int), Total Payment (Float)

Methods:

- Using features(): Use features and get results.
- Feedback(): Submit feedback.
- Upgrade Account(): Upgrade account.
- Manage Accounts(): Manage account details.

Relationships:

- Users can use multiple features (1..* with Features).
- Managed by Admin (1..* with Admin).

2. **Features Class**

Attributes:

- Name, Model (Model AI), Level

Methods:

- Do(): Execute feature functionality.

Relationships:

- Linked to multiple AI models (1..* with Model AI).
- Used by multiple users (1..* with Users).

3. **Admin Class**

Attributes:

- Name, Email, Password, ID

Methods:

- Manage Data(): Manage models, users, and system data.

Relationships:

- Manages users (1..* with Users).
- Manages models (1..* with Model AI).
- Supported by Database Manager.

4. **Model AI Class**

Attributes:

- Name, Device, Parameters (Int), Evaluation (Float)

Methods:

- Function(Data): Processes data and returns results.

Relationships:

- Integrated with multiple features (1..* with Features).
- Managed by admins (1..* with Admin).

5. Database Manager Class

Attributes:

- Engine, Session

Methods:

- Various methods to manage users, records, and data.

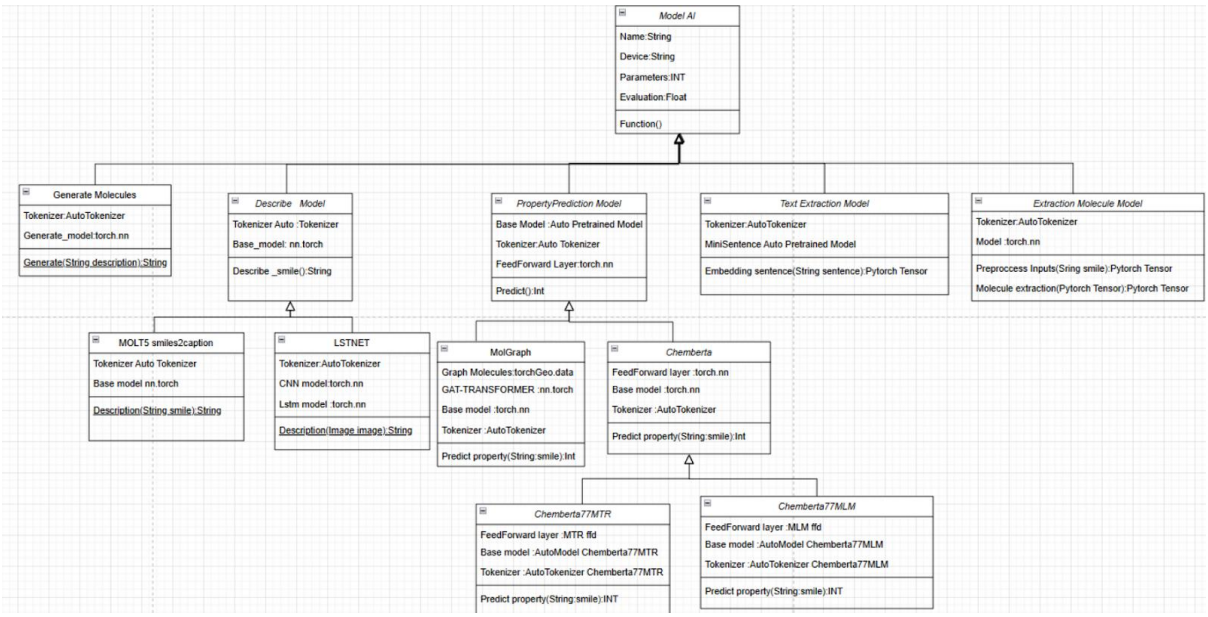
Relationships:

- Supports Admin in managing data (1 with Admin).

Relationships Overview

- **Users and Features**: Users access features based on their level.
- **Features and Model AI**: Features integrate with AI models.
- **Admin and Users**: Admins manage users.
- **Admin and Model AI**: Admins oversee models.
- **Database Manager**: Supports data management for admins.

4.2 AI Models



1. **Model AI**:
    - **Attributes**:

        - Name: Name of the AI model.
        - Device: Device the model is deployed on.
        - Parameters: Number of model parameters.
        - Evaluation: Model evaluation or performance score.
    - **Method**:
        - Function(): Base function representing the general functionality of the model.
    - **Purpose**:
        - This serves as the **base class** for all specialized AI models.
2. **Generate Molecules**:
    - **Attributes**:
        - Tokenizer: AutoTokenizer used for processing input data.
        - Generate_model: PyTorch neural network model.
    - **Method**:
        - Generate(String description): Generates molecules based on a given description.
    - **Purpose**:
        - Focuses on generating molecules using textual or other forms of input.
3. **Describe Model**:
    - **Attributes**:
        - Tokenizer: AutoTokenizer used for input processing.
        - Base_model: Base neural network (PyTorch).
    - **Method**:
        - Describe_(String smile): Describes molecular structures in terms of SMILES (Simplified Molecular Input Line Entry System).
    - **Purpose**:
        - Handles molecular description tasks, particularly for SMILES structures.
4. **Property Prediction Model**:
    - **Attributes**:
        - Base Model: Pretrained model for predictions.
        - Tokenizer: AutoTokenizer used for preprocessing.
        - FeedForward Layer: Neural network layer for property prediction.
    - **Method**:
        - Predict(): Predicts molecular properties.
    - **Purpose**:
        - Predicts molecular properties like solubility, toxicity, or stability.
5. **Text Extraction Model**:
    - **Attributes**:
        - Tokenizer: AutoTokenizer for processing input text.
        - MiniSentence: Pretrained sentence-level model.
    - **Method**:
        - Embedding Sentence(String sentence): Converts input sentences into PyTorch tensor embeddings.

- **Purpose**:
  - Handles text extraction and embedding generation for tasks like feature engineering or text-based predictions.

6. **Extraction Molecule Model**:
   - **Attributes**:
     - Tokenizer: AutoTokenizer for processing molecular SMILES.
     - Model: PyTorch neural network model.
   - **Methods**:
     - Preprocess Inputs(String smile): Preprocesses molecular input into a PyTorch tensor.
     - Molecule Extraction(PyTorch Tensor): Extracts molecular features from the tensor.
   - **Purpose**:
     - Processes molecular data and extracts features for downstream tasks.

Specialized Classes (Derived from Parent Classes)

1. **MOLT5 Smiles2Caption**:
   - Inherits from **Describe Model**.
   - **Attributes**:
     - Tokenizer: AutoTokenizer for SMILES processing.
     - Base_model: PyTorch base model.
   - **Method**:
     - Description(String smile): Converts SMILES molecular representations into descriptive captions.
2. **LSTNET**:
   - Inherits from **Describe Model**.
   - **Attributes**:
     - Tokenizer: AutoTokenizer for text input.
     - CNN Model: Convolutional Neural Network for feature extraction.
     - LSTM Model: Long Short-Term Memory network for sequence learning.
   - **Method**:
     - Description(Image image): Describes molecular images using CNN-LSTM architecture.
3. **MolGraph**:
   - Inherits from **Property Prediction Model**.
   - **Attributes**:
     - Graph Molecules: Uses torchGeo for graph-based molecular data.
     - GAT-Transformer: Graph Attention Transformer for relational learning.
     - Base Model: PyTorch neural network.
   - **Method**:

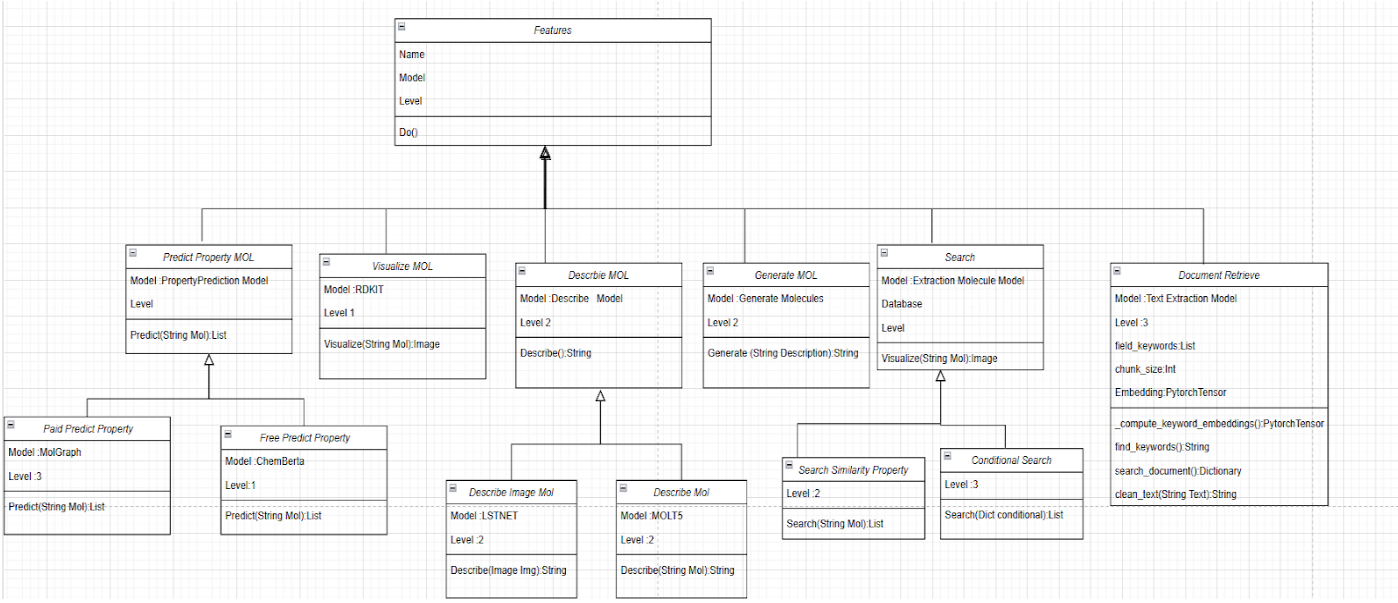
        - Predict Property(String smile): Predicts properties from molecular graphs.
4. **Chemberta**:
    - Inherits from **Property Prediction Model**.
    - **Attributes**:
        - FeedForward Layer: PyTorch layer for predictions.
        - Base Model: Pretrained BERT-based model.
    - **Method**:
        - Predict Property(String smile): Predicts properties for SMILES representations.
5. **Chemberta77MTR**:
    - Inherits from **Chemberta**.
    - **Attributes**:
        - FeedForward Layer: Multi-task learning feedforward layer.
        - Base Model: Chemberta77 pretrained model.
    - **Method**:
        - Predict Property(String smile): Similar to Chemberta with additional multi-task learning capabilities.
6. **Chemberta77MLM**:
    - Inherits from **Chemberta**.
    - **Attributes**:
        - FeedForward Layer: Specialized for masked language modeling.
        - Base Model: Chemberta77MLM pretrained model.
    - **Method**:
        - Predict Property(String smile): Predicts properties with masked molecular input learning.

Purpose of the System

This architecture is designed for:

1. **Molecule Generation**:
    - Generate new molecular structures based on text or other input.
2. **Molecule Description**:
    - Provide meaningful descriptions of molecular structures, particularly for SMILES data.
3. **Property Prediction**:
    - Predict molecular properties like stability, toxicity, or solubility using specialized models.
4. **Text Embedding and Extraction**:
    - Generate embeddings for textual inputs to enhance molecular understanding.
5. **Feature Extraction**:
    - Extract features from molecules for use in downstream machine learning tasks.

4.3 Research Features



- **Attributes**:
  - Name: Name of the feature.
  - Model: The underlying AI or processing model used to implement the feature.
  - Level: The required user access level for utilizing the feature.
- **Methods**:
  - Do(): Abstract method for executing the feature.

Subclasses and Specializations

1. **Predict Property MOL**
   - **Purpose**:
     - Predict molecular properties like solubility, stability, or reactivity.
   - **Attributes**:
     - Model: Uses the PropertyPredictionModel class.
     - Level: Defines the access level required to use the feature.
   - **Method**:
     - Predict(String Mol): Takes a molecular string (SMILES) as input and predicts properties.
   - **Subclasses**:
     - **Paid Predict Property**:
       - Model: MolGraph.
       - Level: 3 (advanced users or paid access).
     - **Free Predict Property**:
       - Model: ChemBerta.
       - Level: 1 (available to general users).
2. **Visualize MOL**
   - **Purpose**:
     - Provides visual representations of molecular structures.

- **Attributes**:
  - Model: Uses RDKit, a popular toolkit for molecular visualization and cheminformatics.
  - Level: 1 (accessible to all users).
- **Method**:
  - Visualize(String Mol): Converts molecular SMILES to an image representation.

3. **Describe MOL**
   - **Purpose**:
     - Describes molecular features, either textually or as captions.
   - **Attributes**:
     - Model: Uses the Describe Model class.
     - Level: 2 (mid-level access).
   - **Method**:
     - Describe(String Mol): Converts molecular input into descriptive text.
   - **Subclasses**:
     - **Describe Image MOL**:
       - Model: LSTNET.
       - Designed to describe molecular images.
     - **Describe MolT5**:
       - Model: MolT5.
       - Generates descriptive captions for molecular structures.
4. **Generate MOL**
   - **Purpose**:
     - Generates new molecular structures.
   - **Attributes**:
     - Model: Uses Generate Molecules.
     - Level: 2 (mid-level access).
   - **Method**:
     - Generate(String Description): Creates molecular structures based on input descriptions.
5. **Search**
   - **Purpose**:
     - Searches for specific molecules or properties in the database.
   - **Attributes**:
     - Model: Uses the Extraction Molecule Model.
     - Database: Indicates the database used for the search.
     - Level: Variable based on the search complexity.
   - **Subclasses**:
     - **Search Similarity Property**:
       - Level: 2.
       - Searches molecules based on similarity.
     - **Conditional Search**:
       - Level: 3 (advanced search).

        - Accepts specific conditions for more refined search results.
6. **Document Retrieve**
    - **Purpose**:
        - Retrieves and processes documents related to molecular properties or research.
    - **Attributes**:
        - Model: Uses Text Extraction Model.
        - Level: 3 (advanced access).
        - field_keywords: List of keywords to filter search results.
        - chunk_size: Defines the document chunk size for processing.
        - Embedding: Converts document text into PyTorch tensor embeddings.
    - **Methods**:
        - compute_keyword_embeddings(): Computes embeddings for a list of keywords.
        - search_document(): Searches a document for relevant information based on keywords.
        - clean_text(String Text): Cleans and preprocesses input text for further analysis.

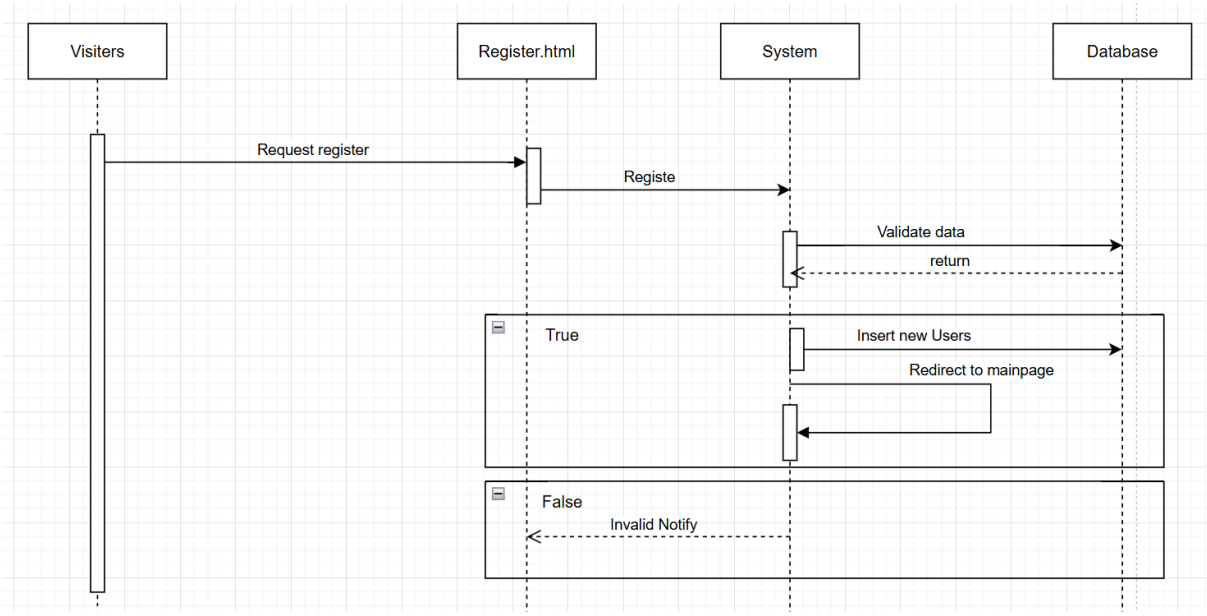
Key Relationships

1. **Hierarchy and Inheritance**:
    - The Features class is the parent, while its subclasses represent specialized functionalities.
    - Subclasses inherit core attributes like Name, Model, and Level from Features.
2. **Access Levels**:
    - Level 1: General access (e.g., free property prediction, visualization).
    - Level 2: Mid-level access (e.g., molecular description, generation, similarity search).
    - Level 3: Advanced or paid access (e.g., paid prediction, conditional search, document retrieval).
3. **Modular Design**:
    - Each subclass uses specific models (e.g., RDKit, MolGraph, LSTNET) tailored to its function, enabling modularity and scalability.

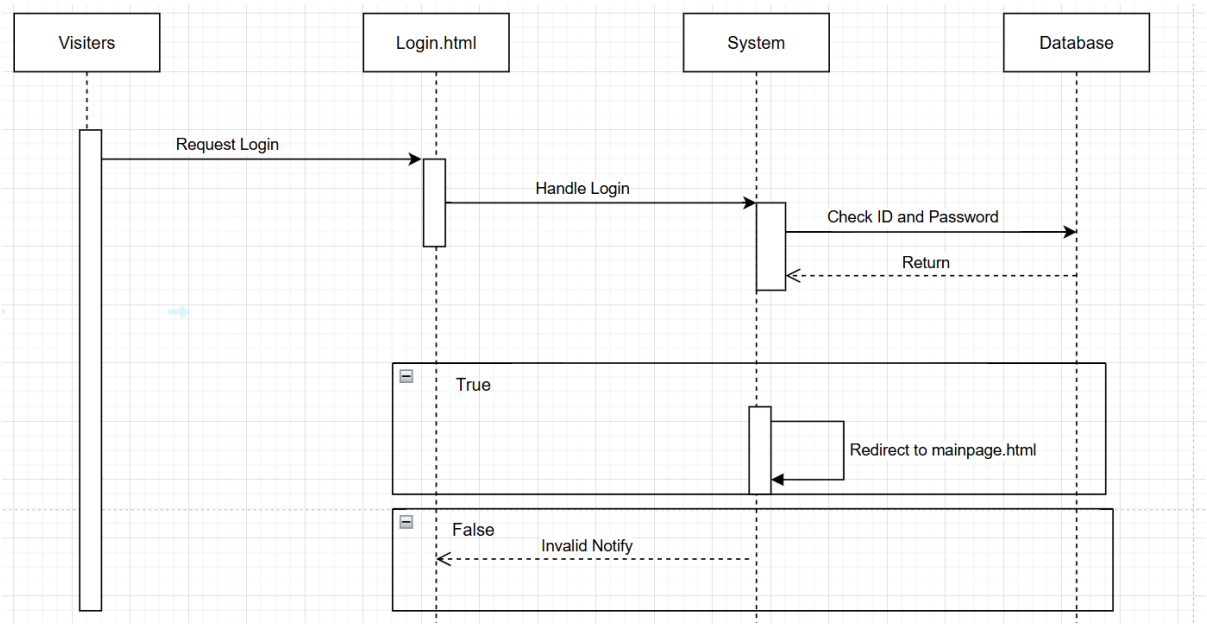
5 Sequence Diagrams

1 User's features

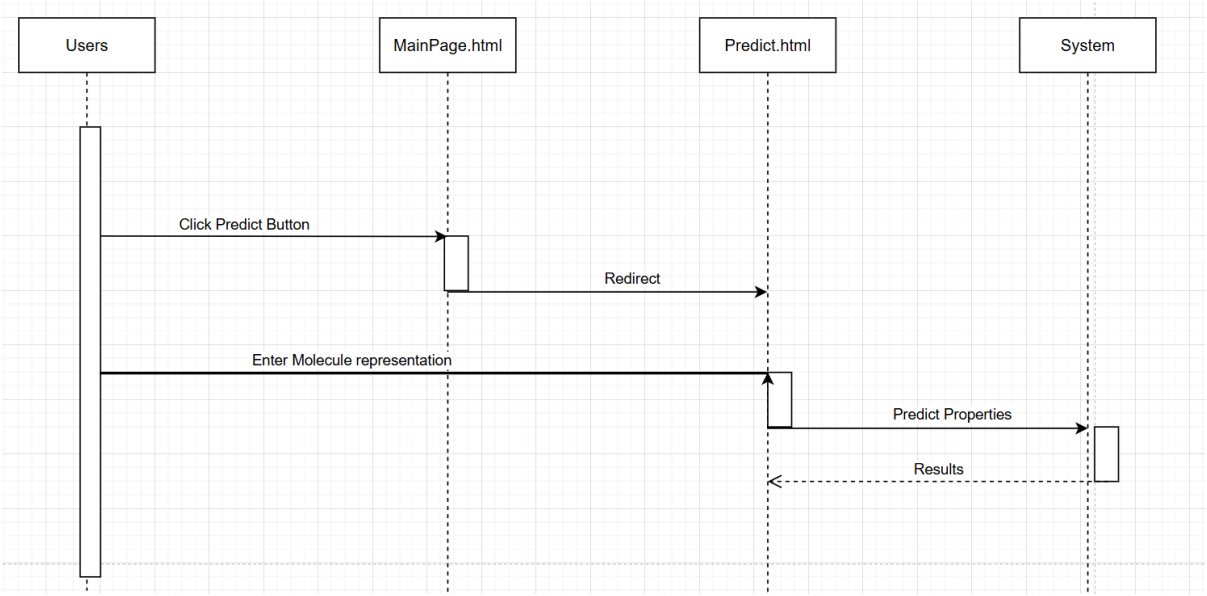
Use Case 1: Register



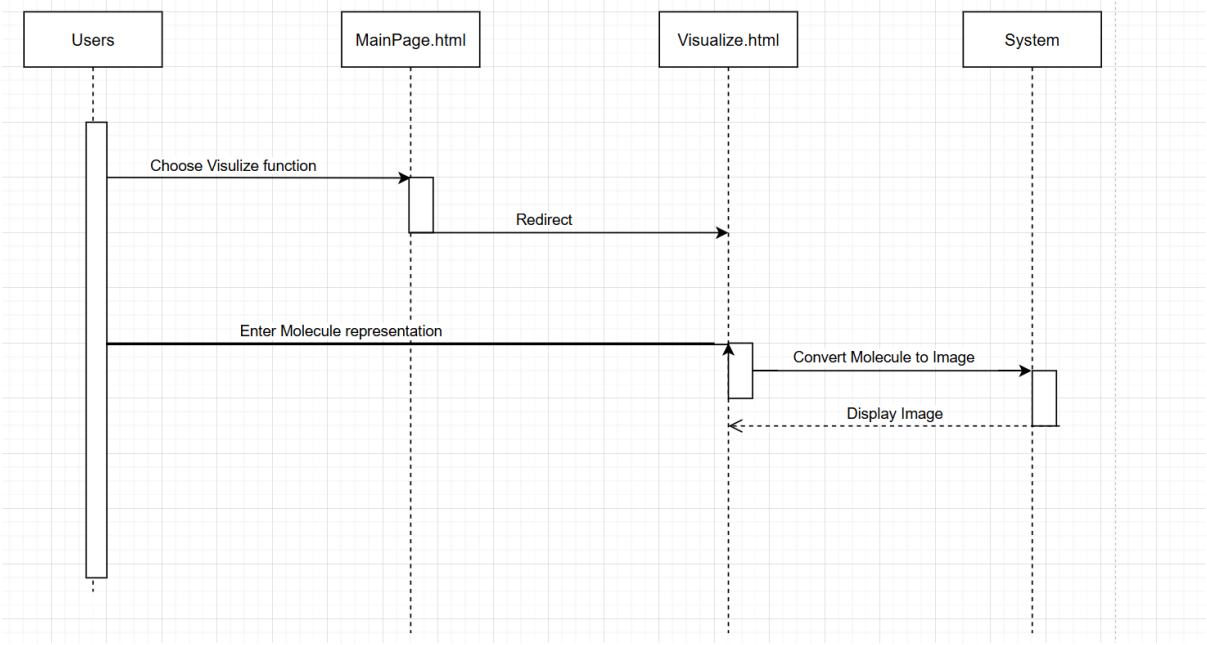
Use Case 2: Login



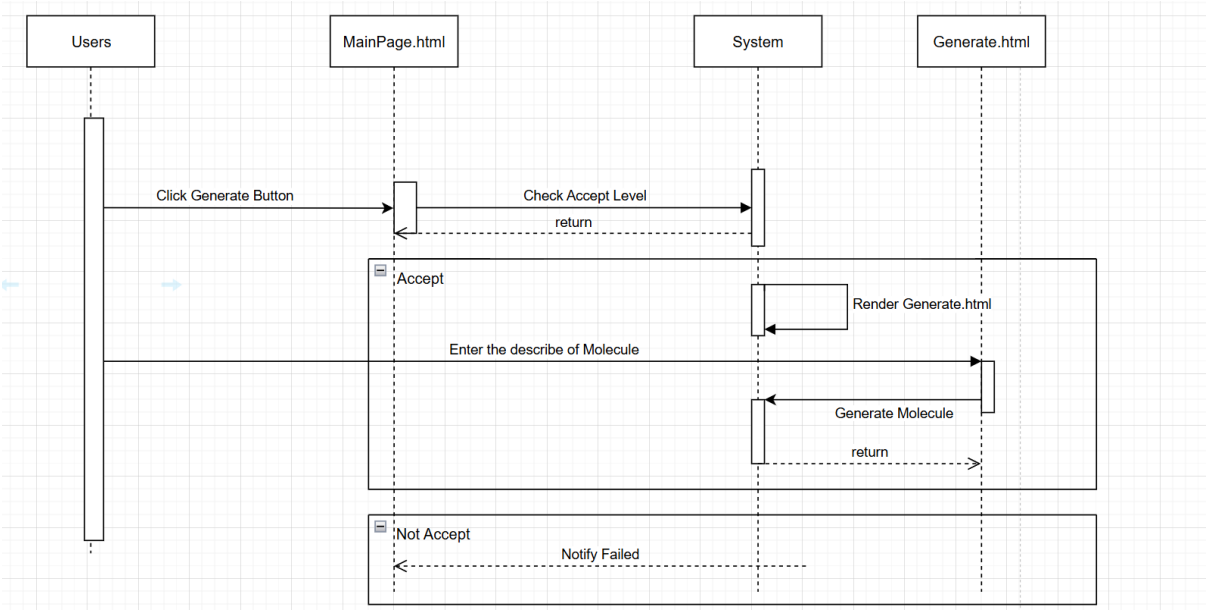
Use case 3 Using Predict features



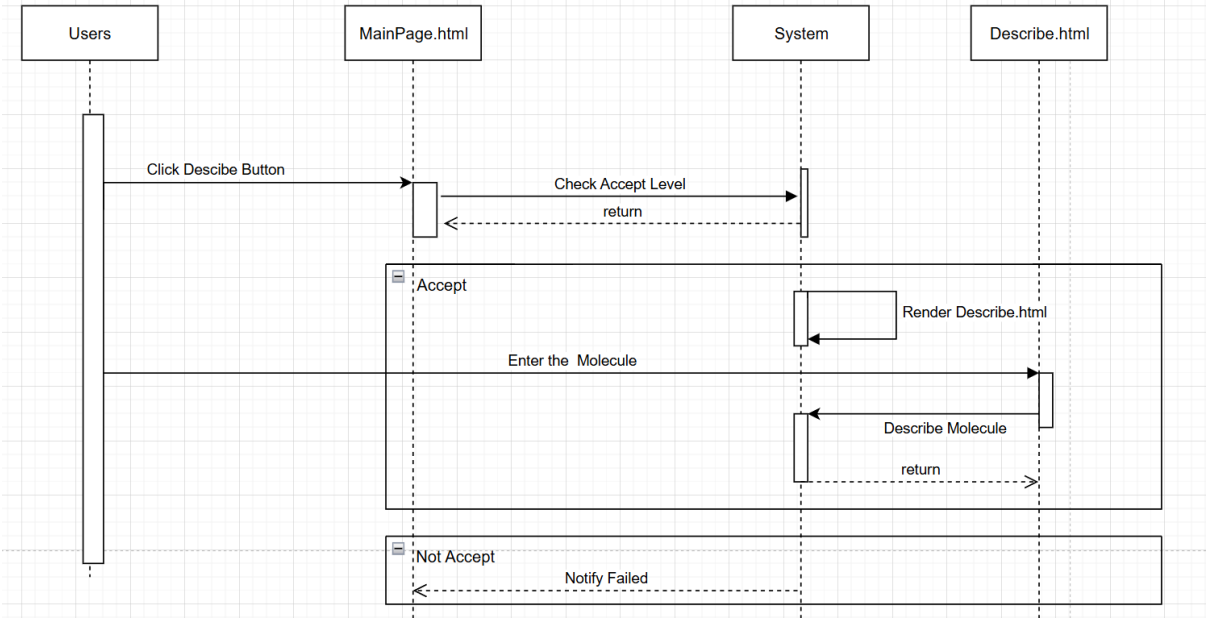
Use case 4 Using Visualize features



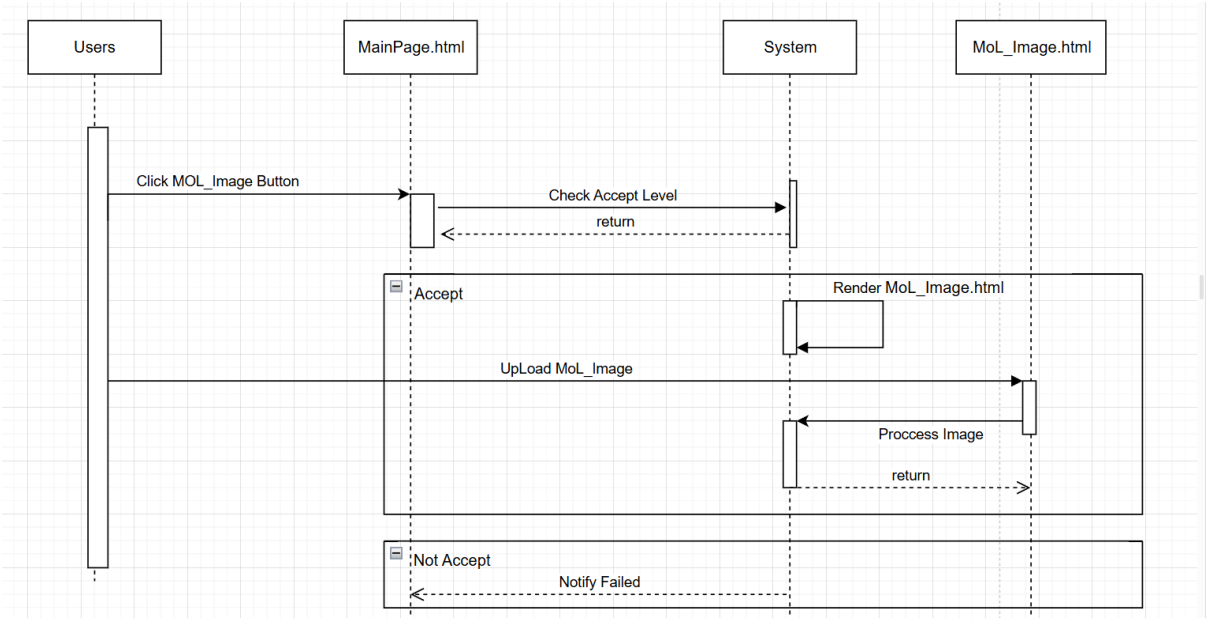
Use case 5 Using Generate Molecule features



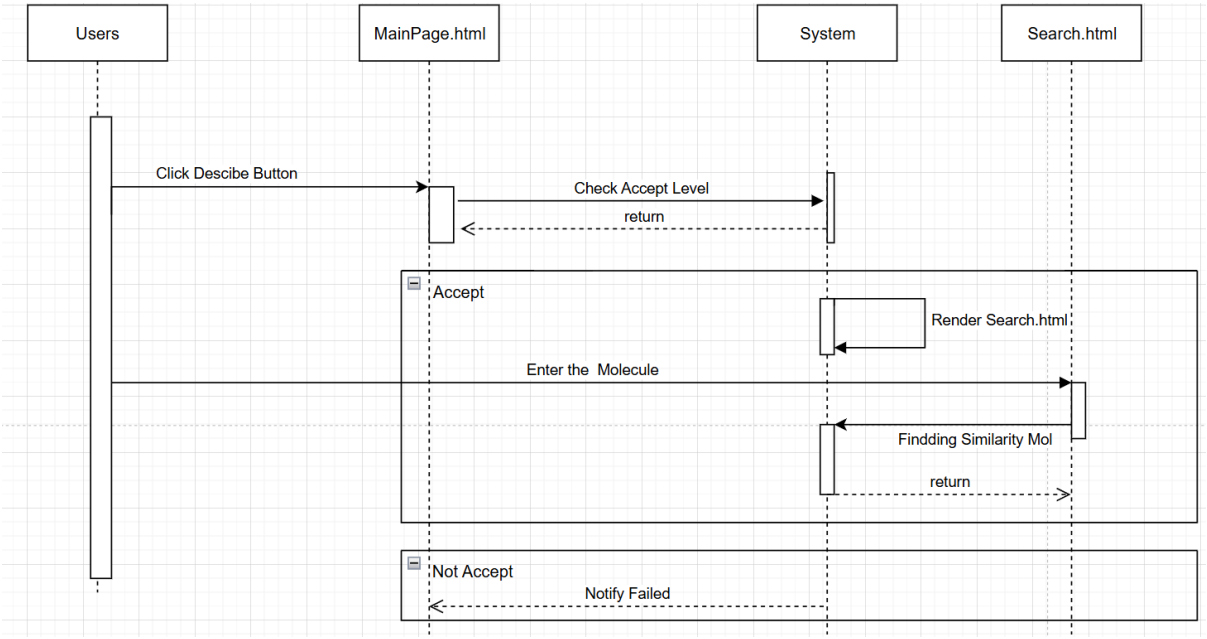
Use case 6 Using Describe Molecule feature



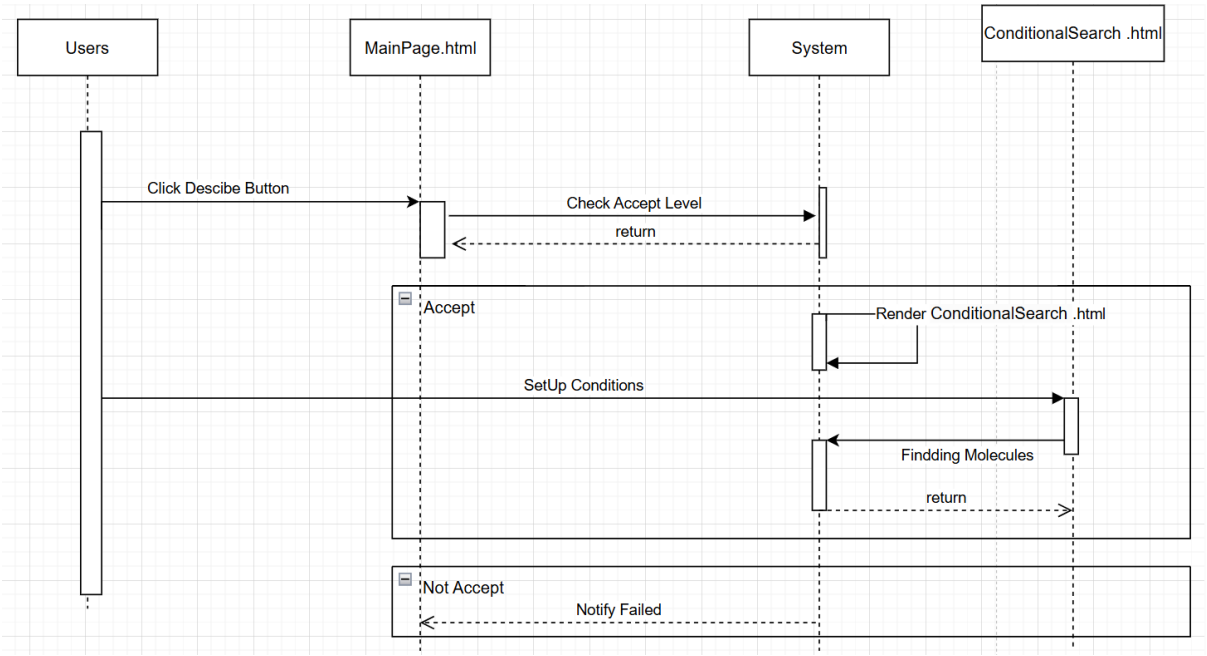
Use case 7 Using Molecule Image features



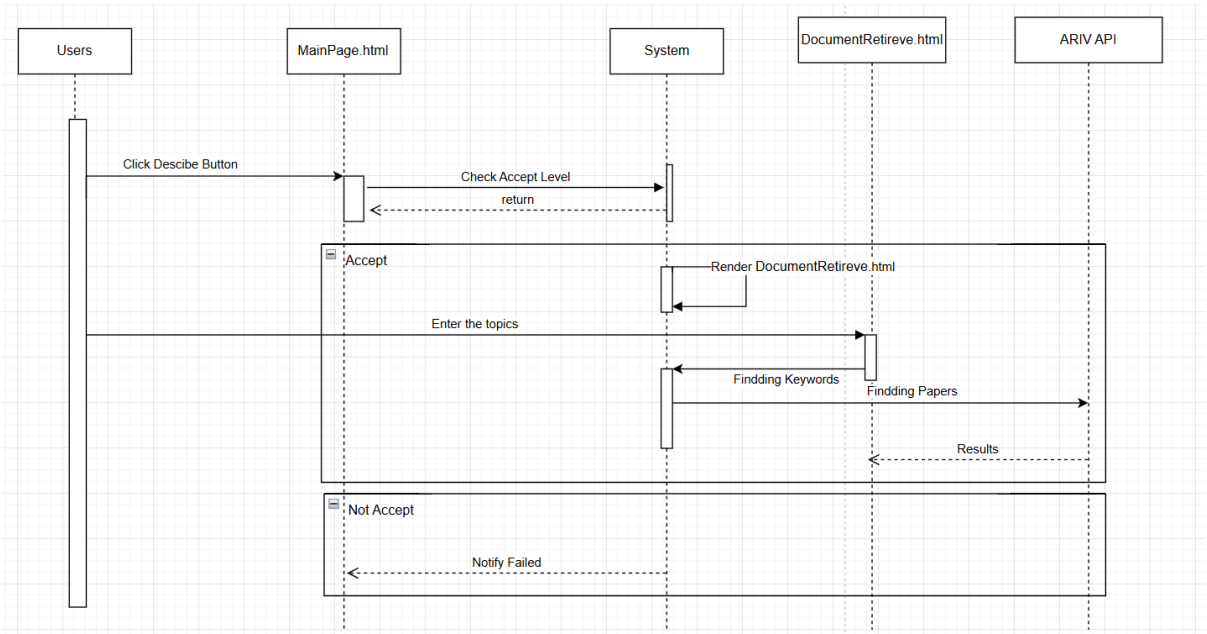
Use case 8 Using Search Molecule features



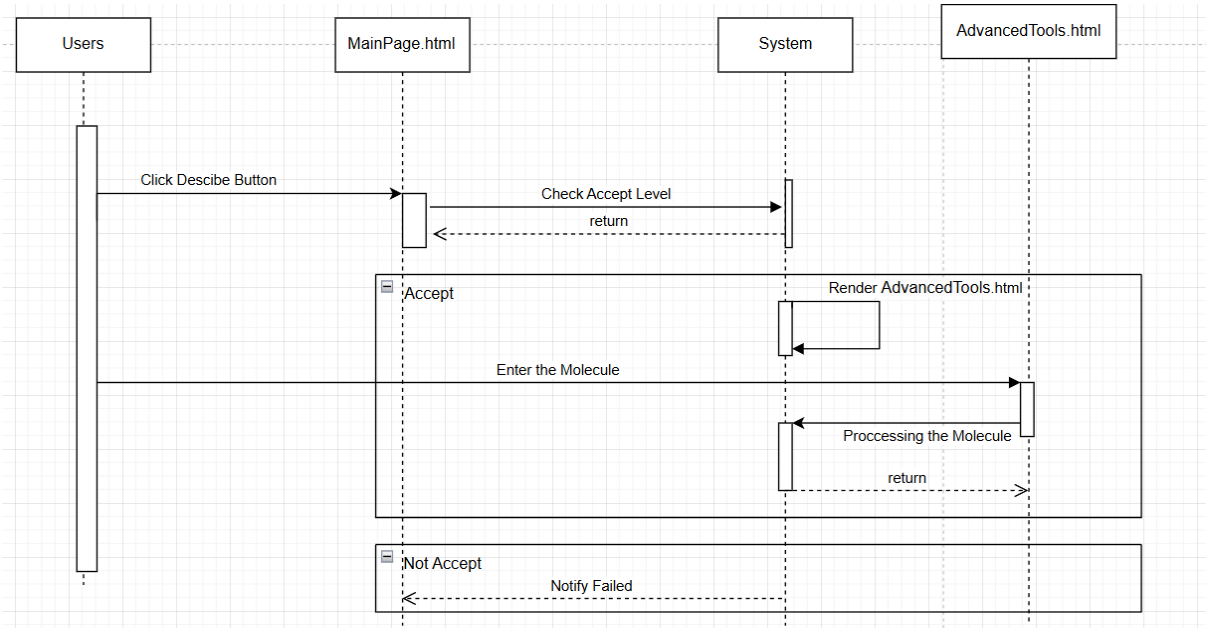
Use case 9 Using Search Molecule with Condition features



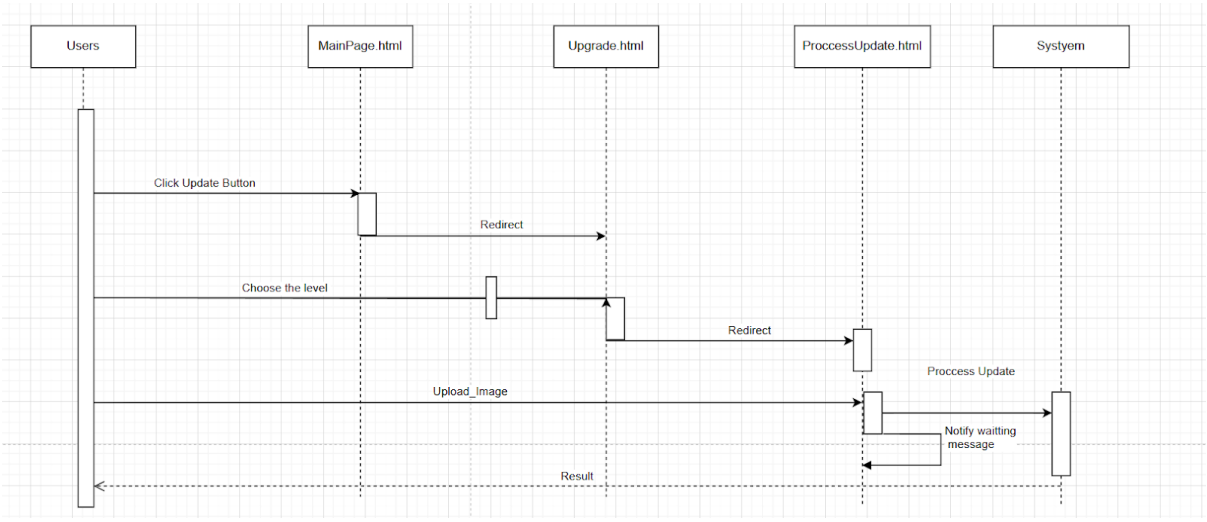
Use case 10 Using Search Document features



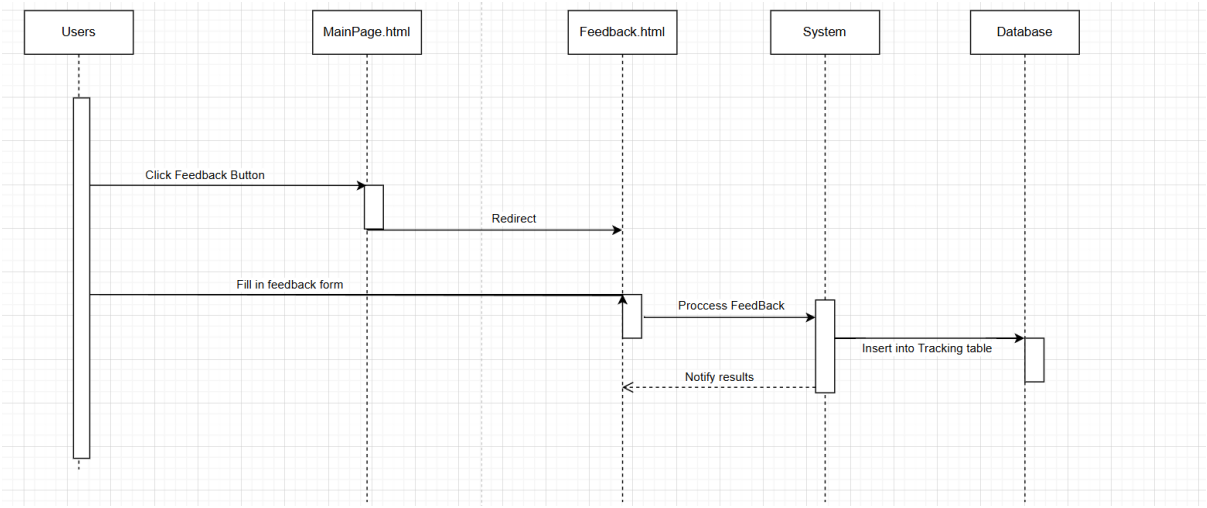
Use case 11 Using Advanced Analysis features



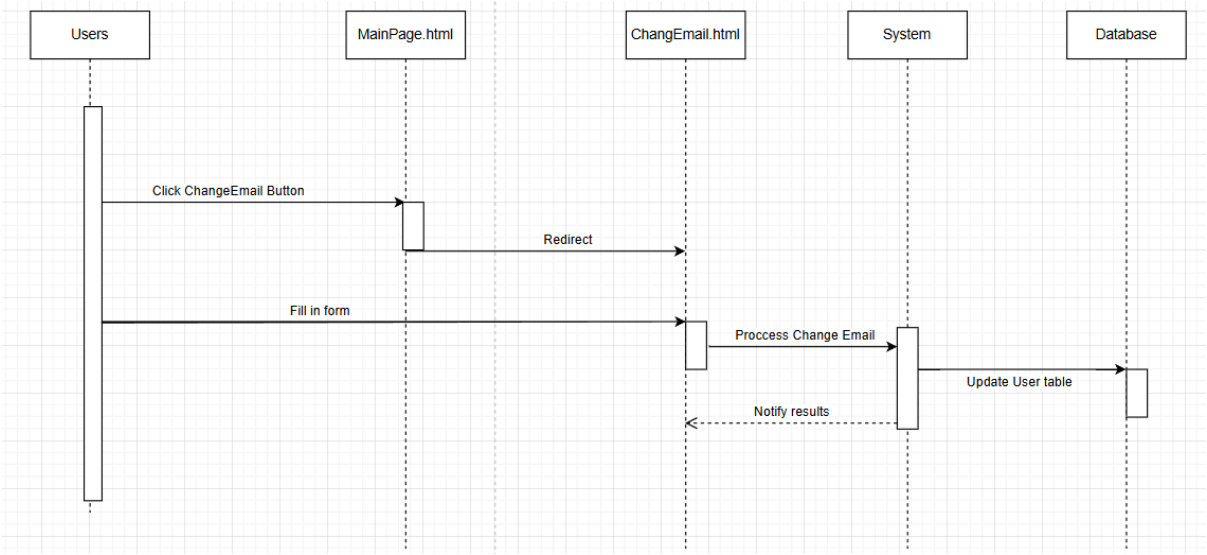
Use case 12 Update Account Level



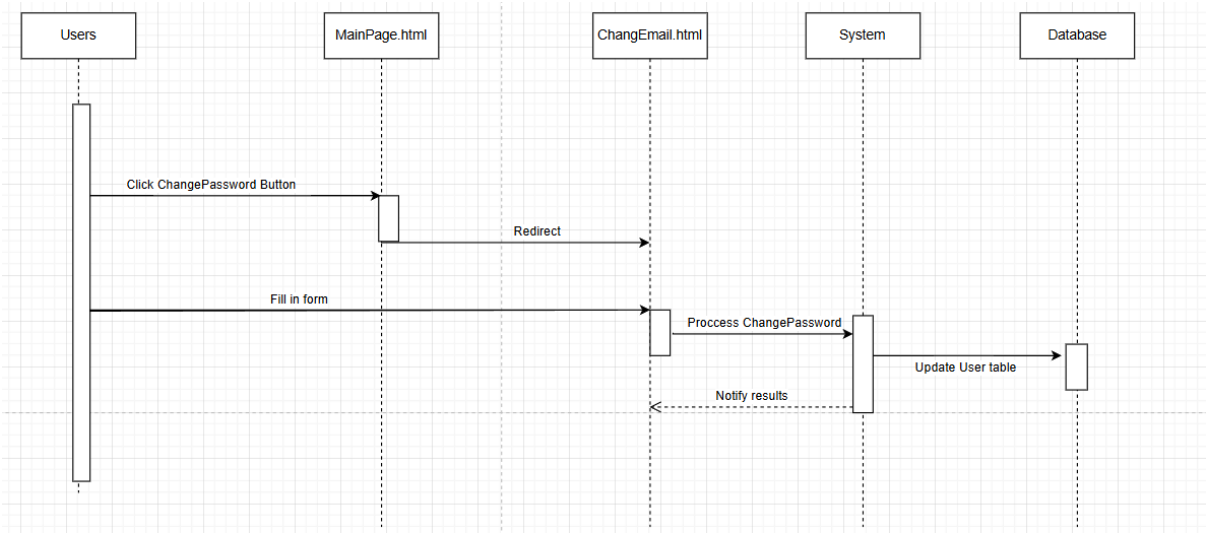
Use case 13 Feedback Features



Use case 14 Change Email

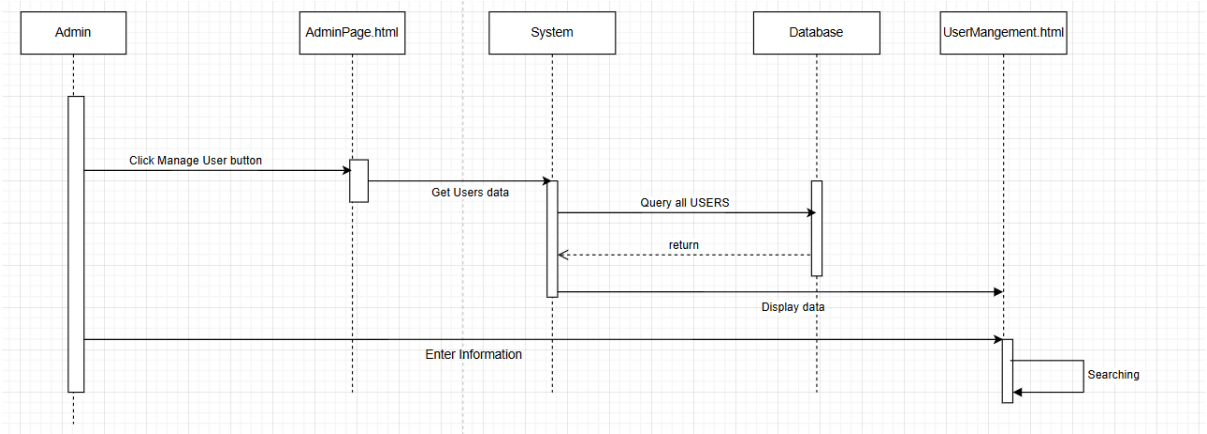


Use case 15 Change Password

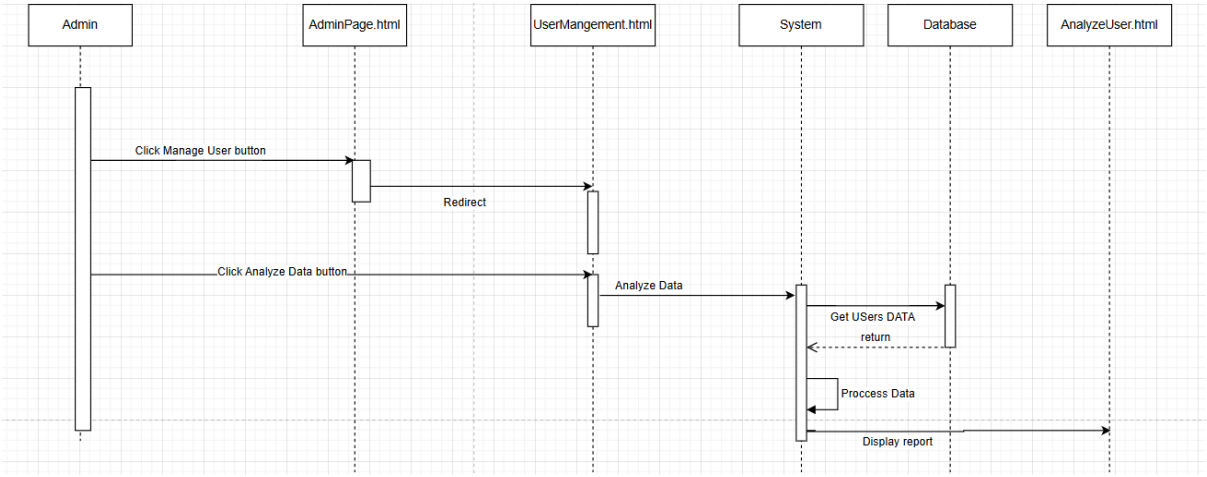


2 Admin's Features

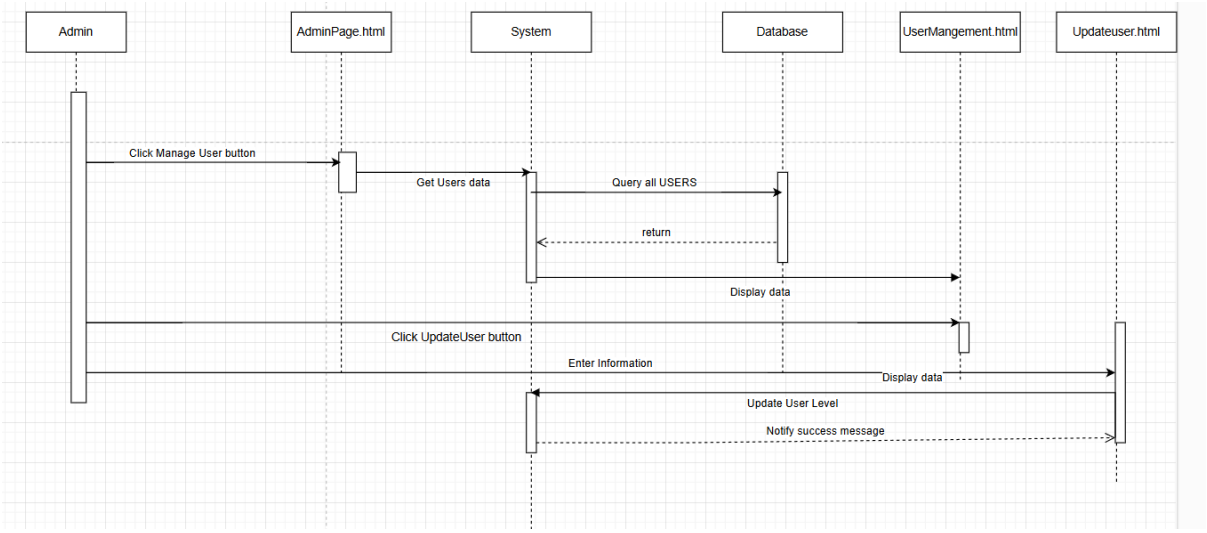
Use Case 1 Manage User data



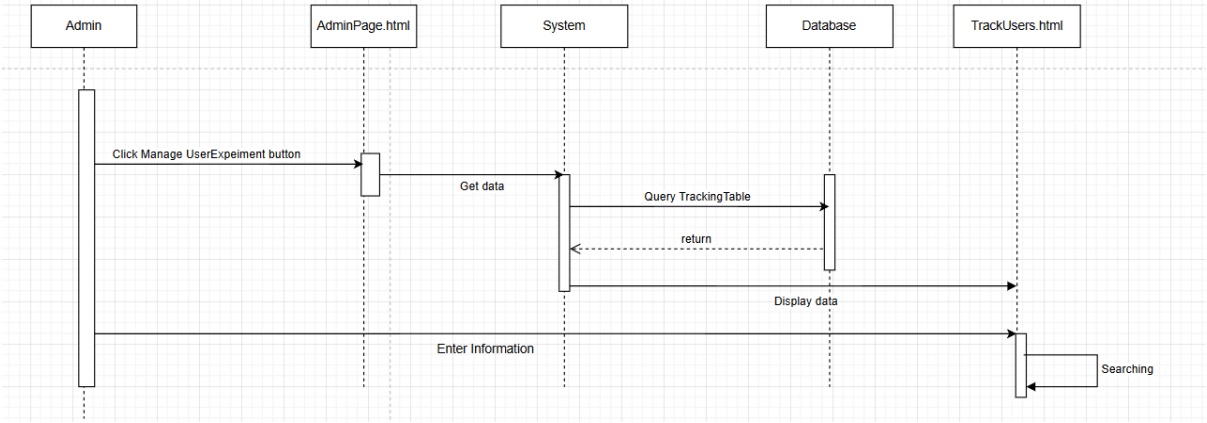
Use Case 2 Analyze User data



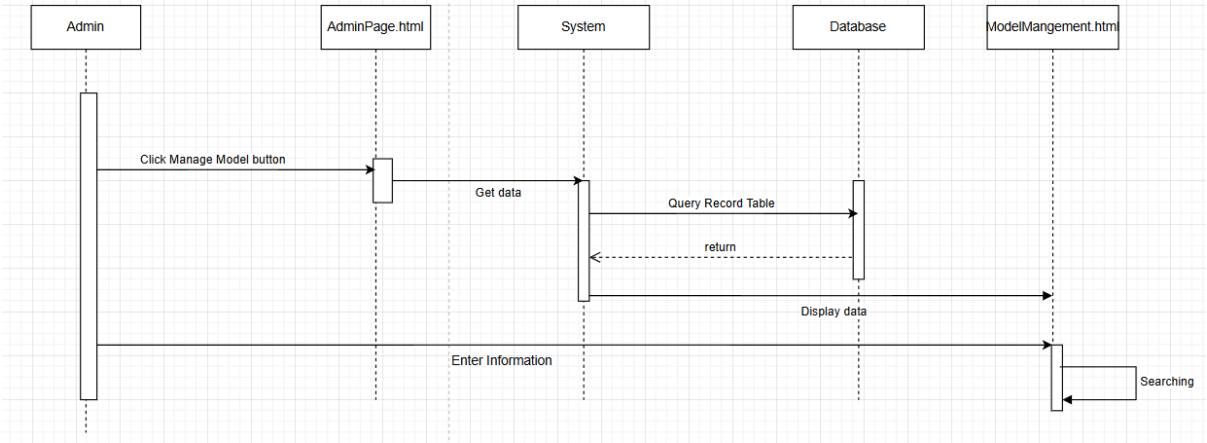
Use Case 3 Process User update



Use Case 4 Manage User's experience



Use Case 5 Manage Models in platform

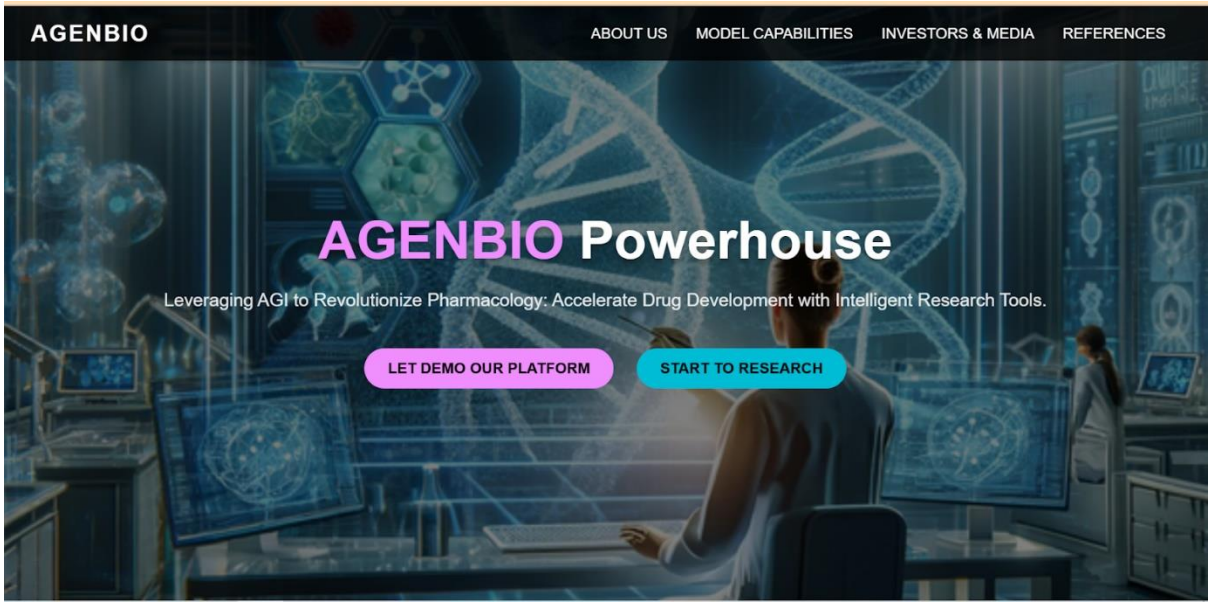


III Implementation

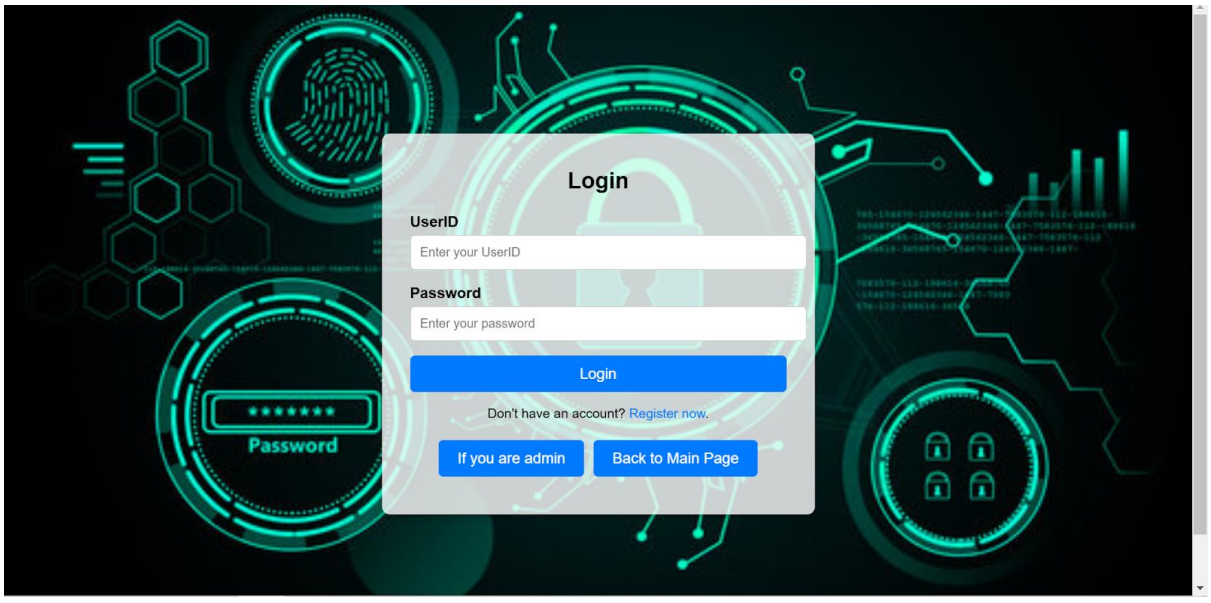
1 User features

1.1. Login into an account

Step 1: Go to the Main page via the link:



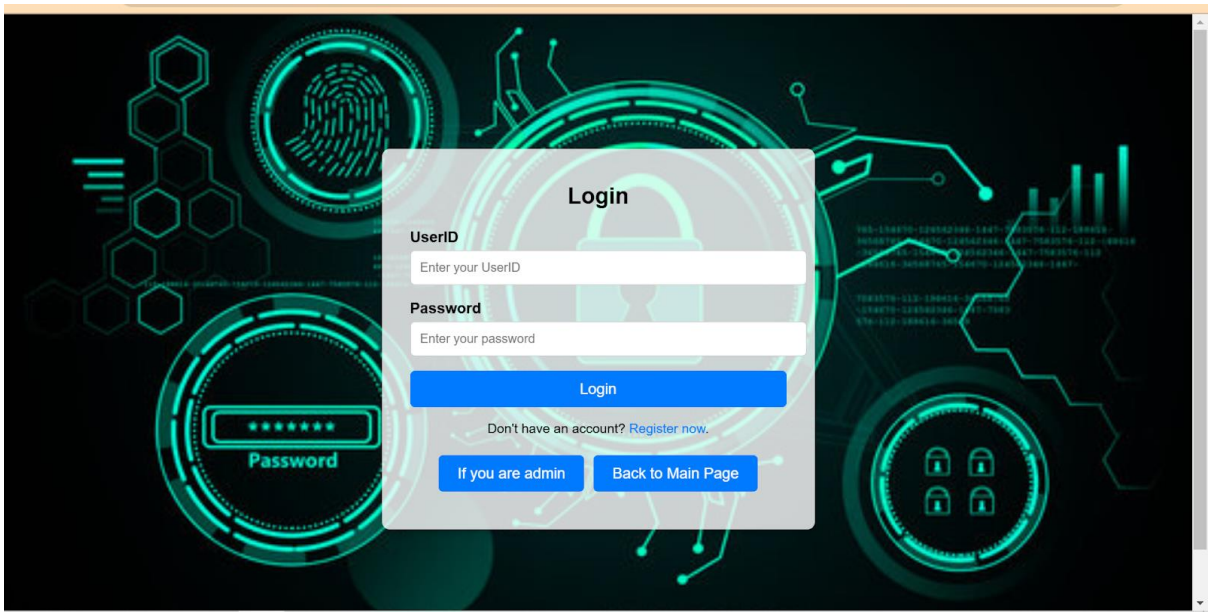
Step 2 Click on Start to research button from the homepage, the login page will appear.



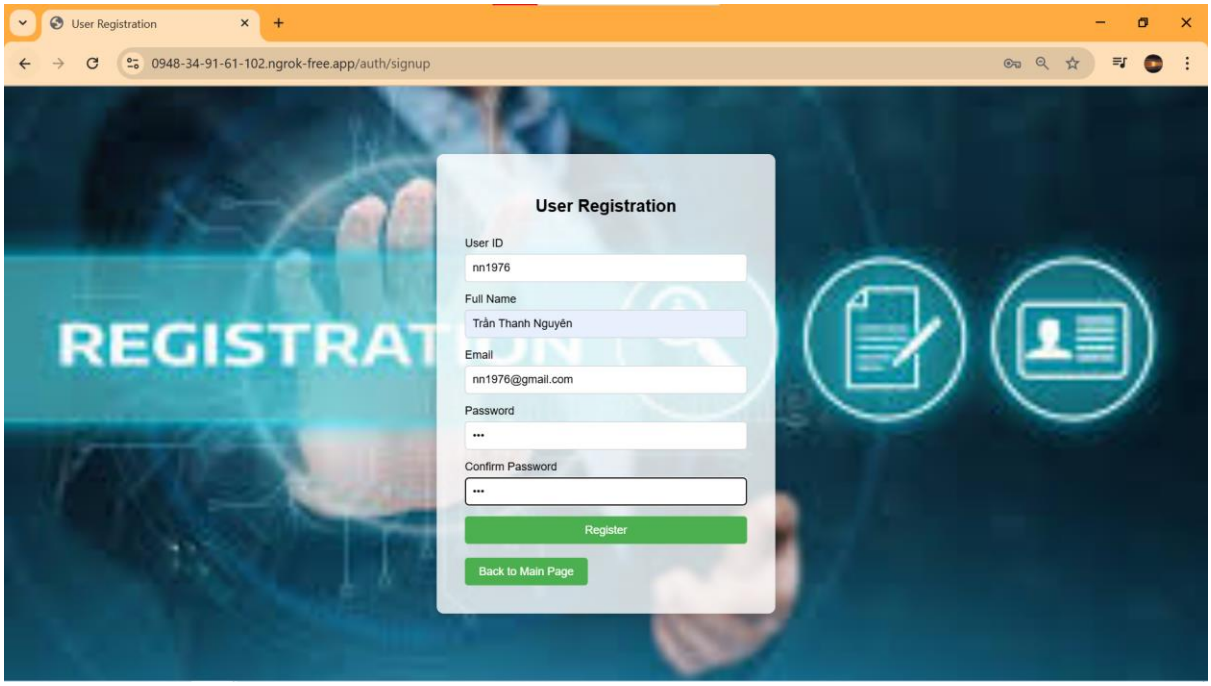
A successful login dialog will appear and the customer will be redirected to the homepage.

1.2. Register a new account

Step 1: In the login page, click Register now link



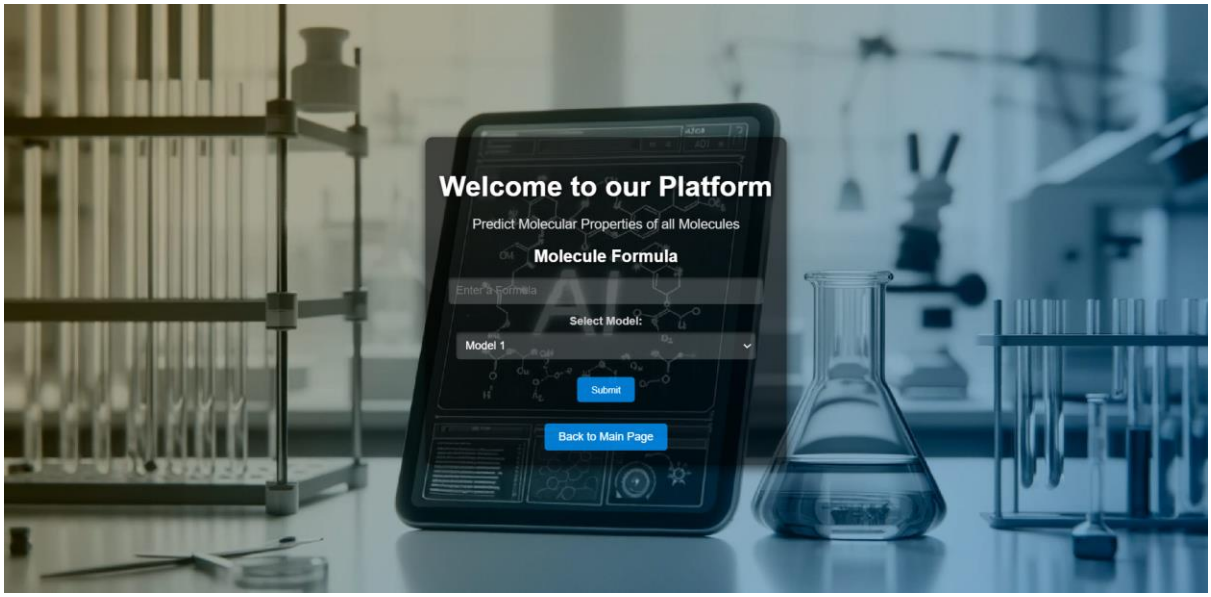
Step 2: In the Register page,fill in the register Form



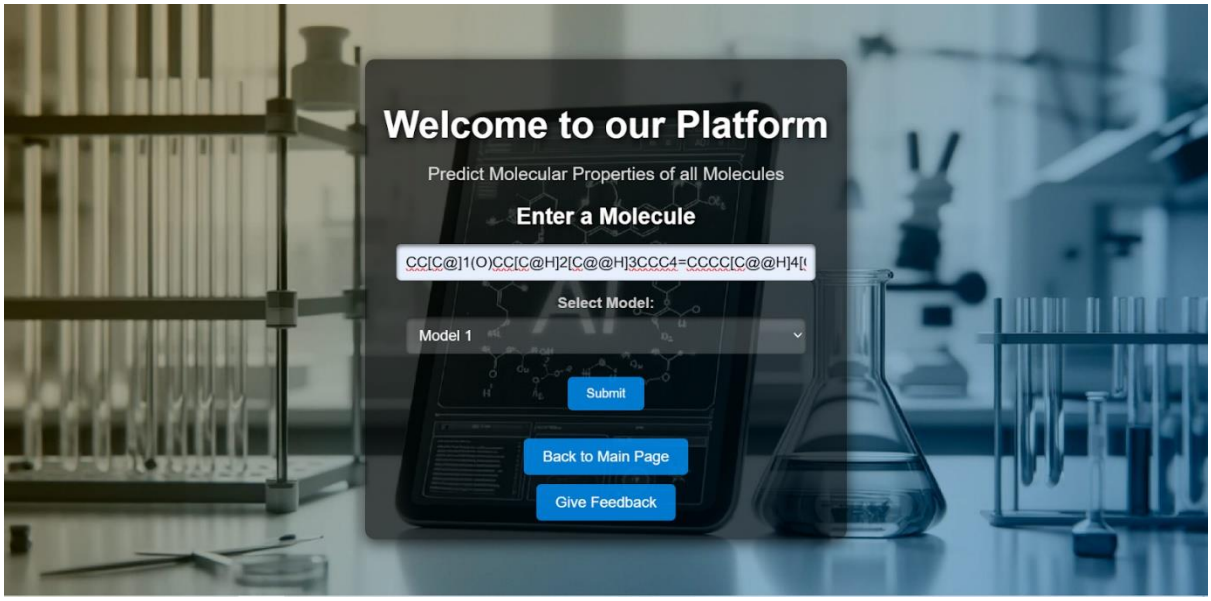
Step 3: When register successfully ,the users will be redirect into user mainpage

1.3.Using Predict Properties

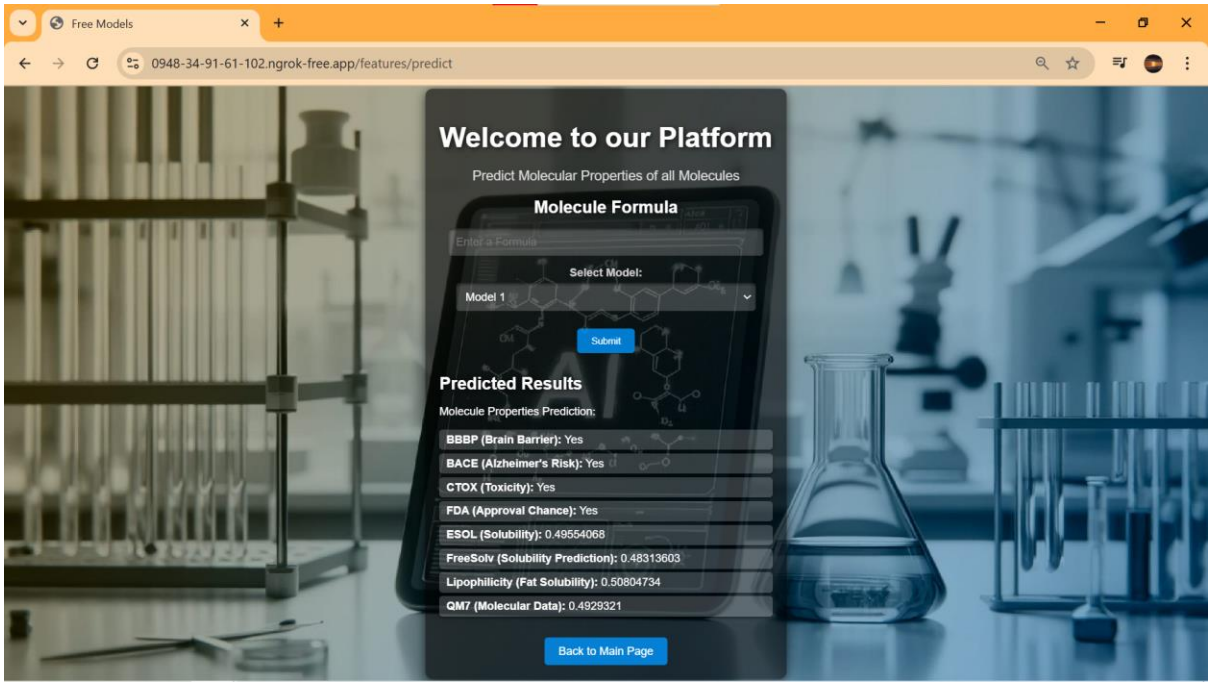
Step 1 Click Predict Properties button on main page,Users will be move to



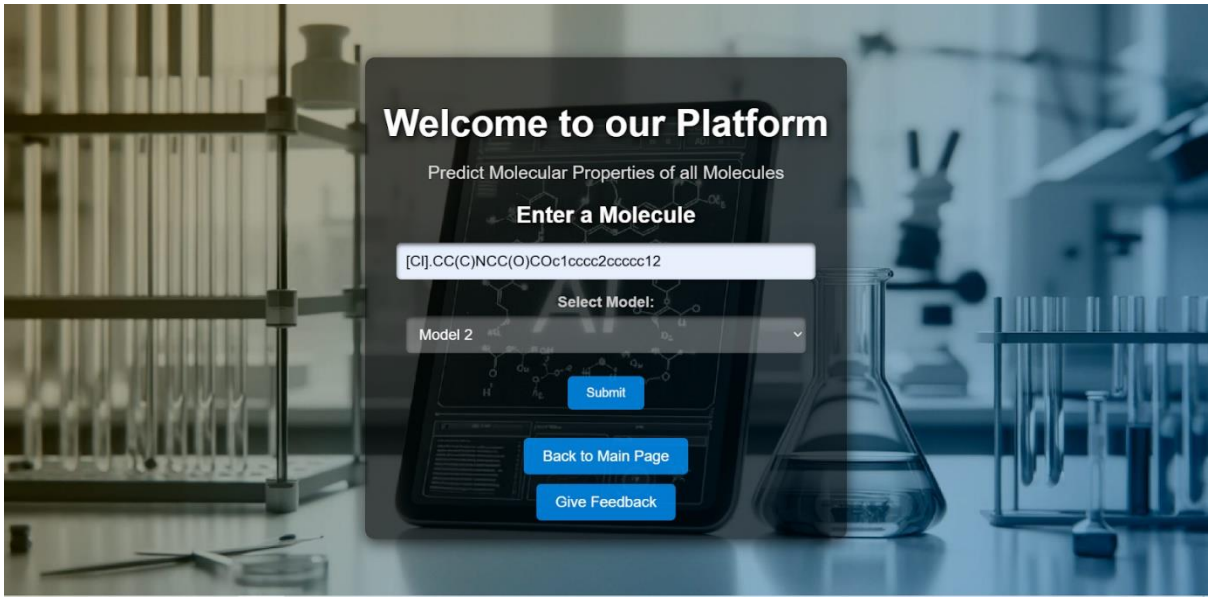
Step 2 :Users enter a Molecule formula and select Model they want to use

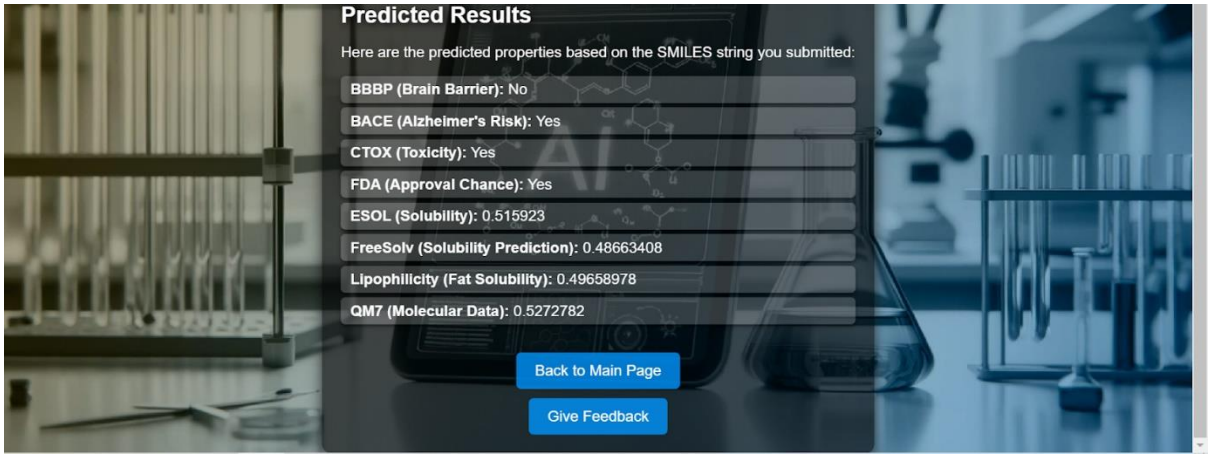


Step 3 :Click On Submit button ,the result displayed as



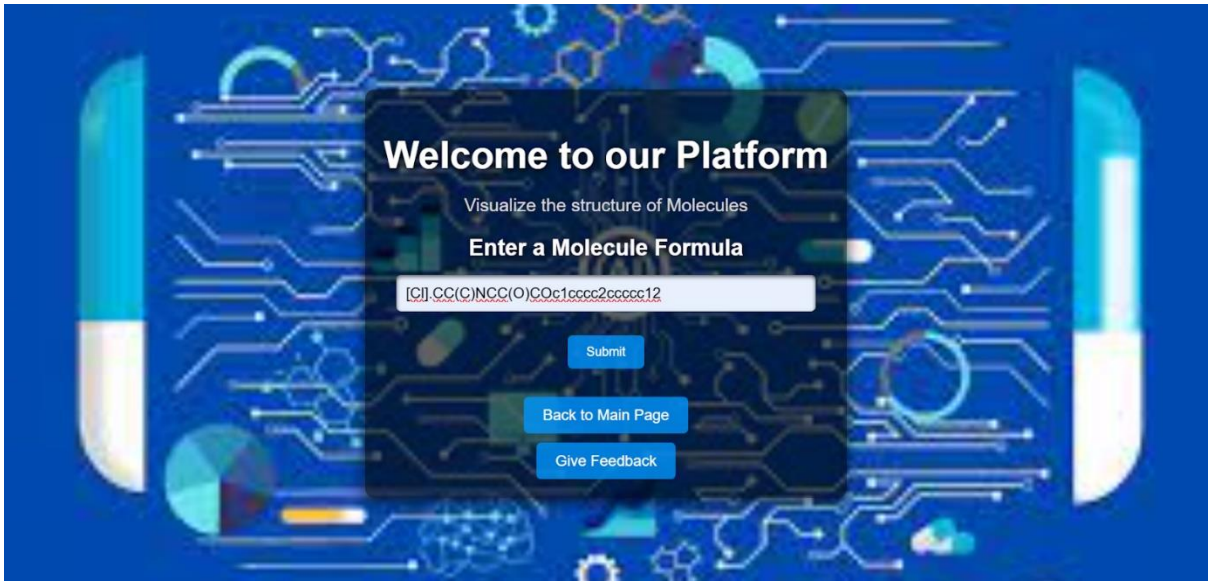
Note:Users can use either model 1 or model 2 to predict:



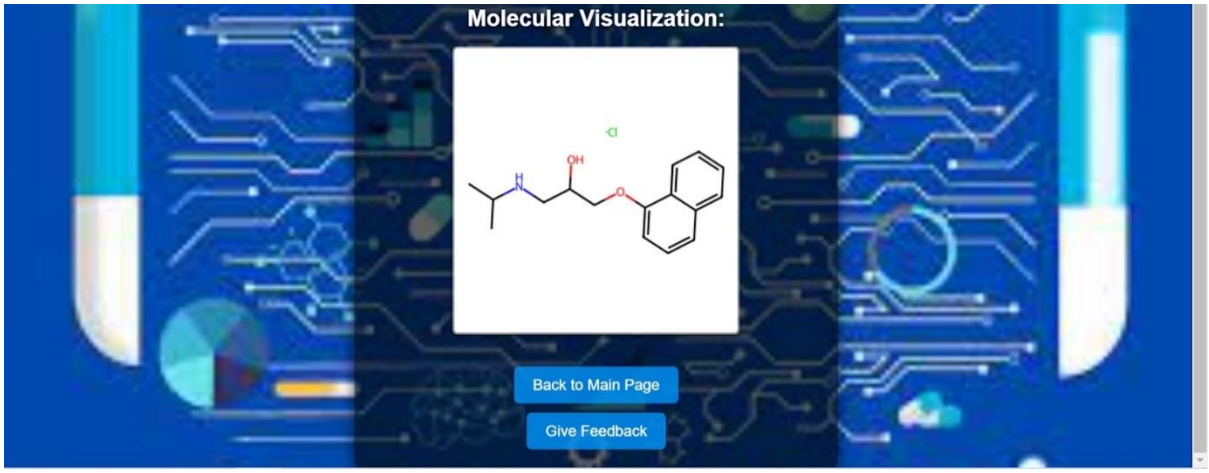


1.4.Visualize the Molecule structure

Step 1 :Click Molecule Visualization button on main page



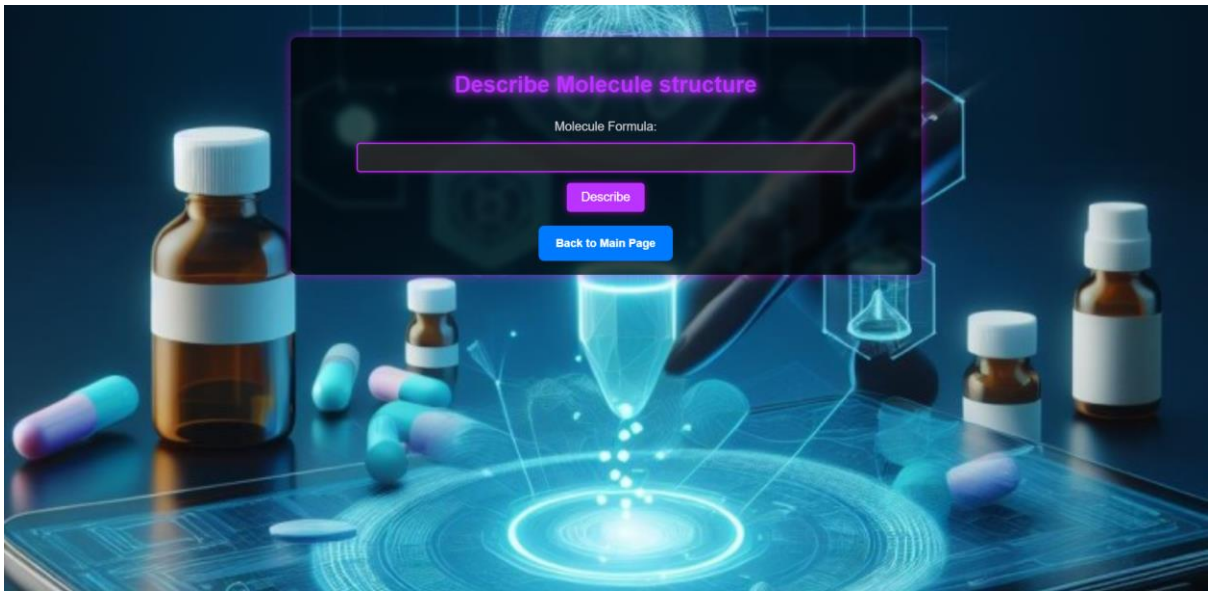
The result will be



1.5.Describe the Molecule structure

Step 1 :Click Describe Molecule button on the main page

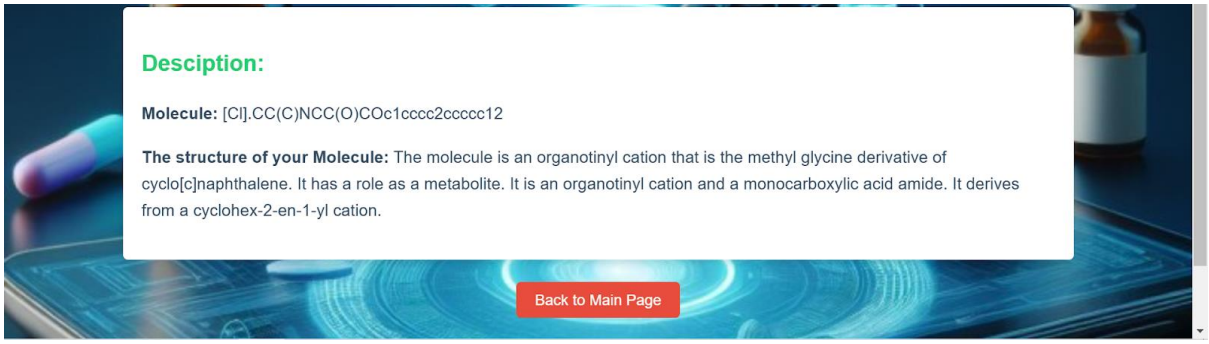
Step 2 :Enter the formula of Molecule that you want to describe



Note

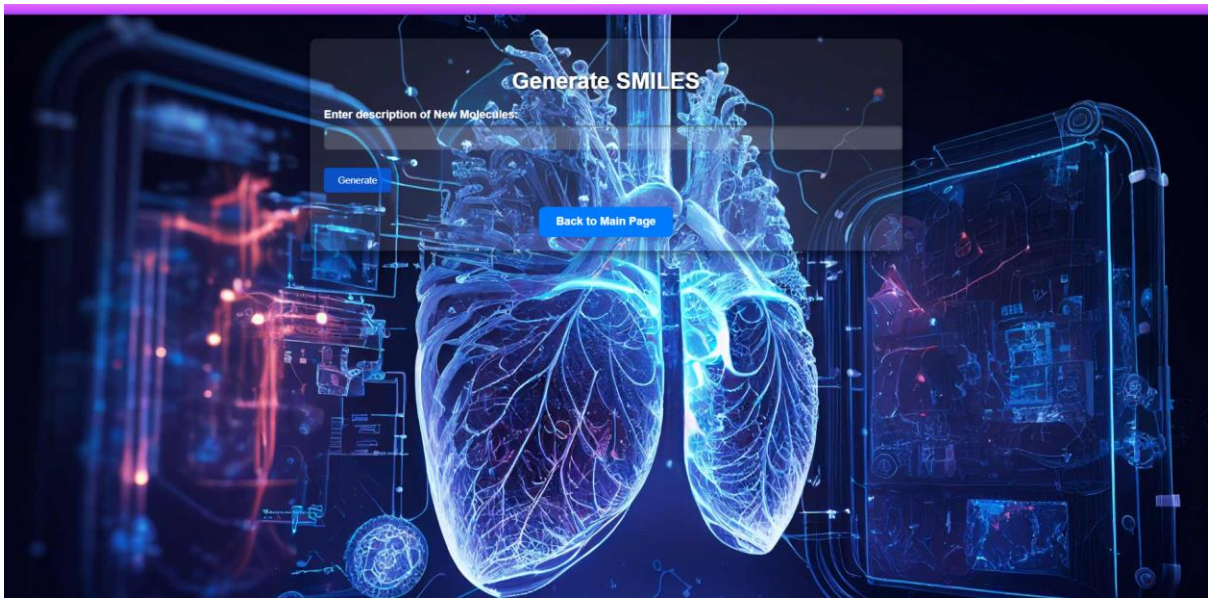
The request will be handled in about 45 seconds up to one and a half minutes depending on the current device.

The result will be

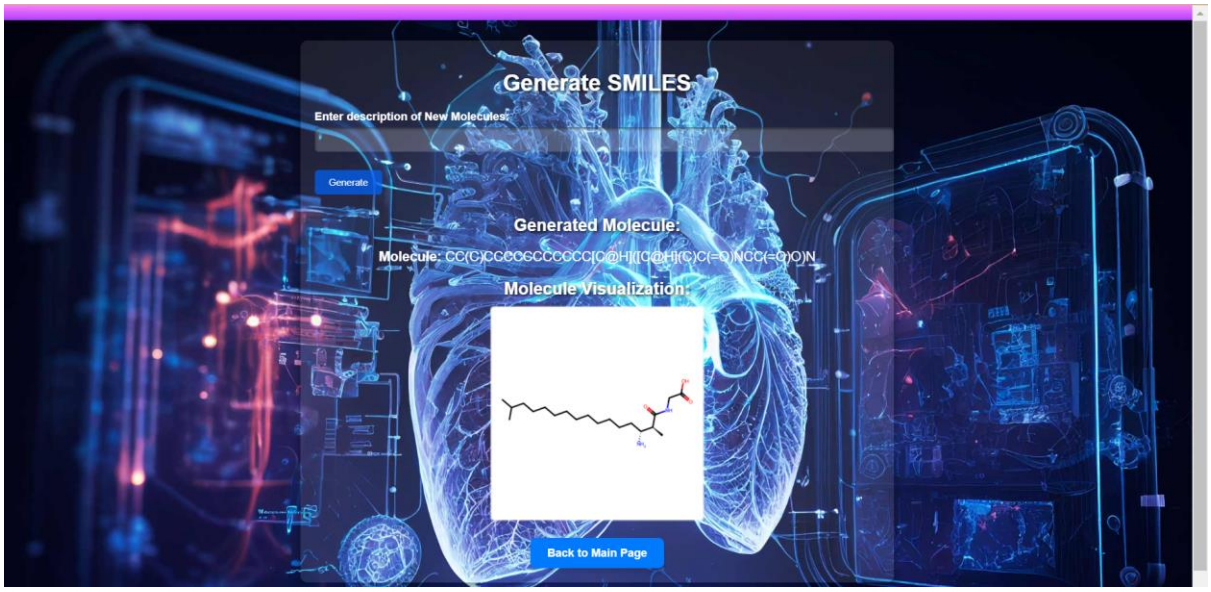


1.6.Generate the new Molecule from User’s description

Step 1 :Click Molecule Generation button on the main page
Step 2 :Enter the formula of Molecule that you want to generate

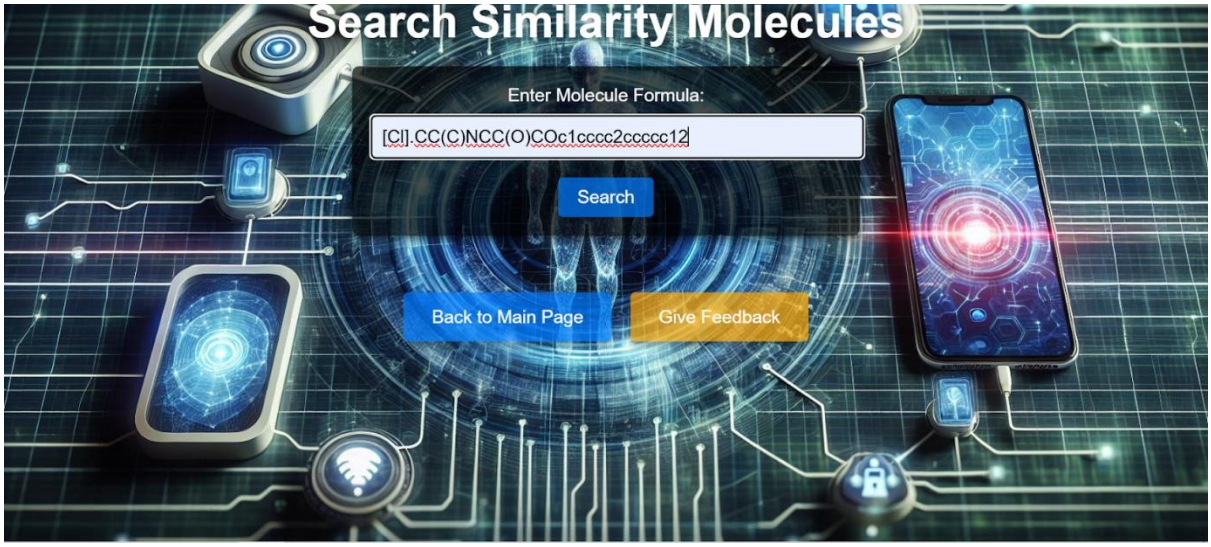


Note
The request will be handled in about 45 seconds up to one and a half minutes depending on the current device.
The result will be

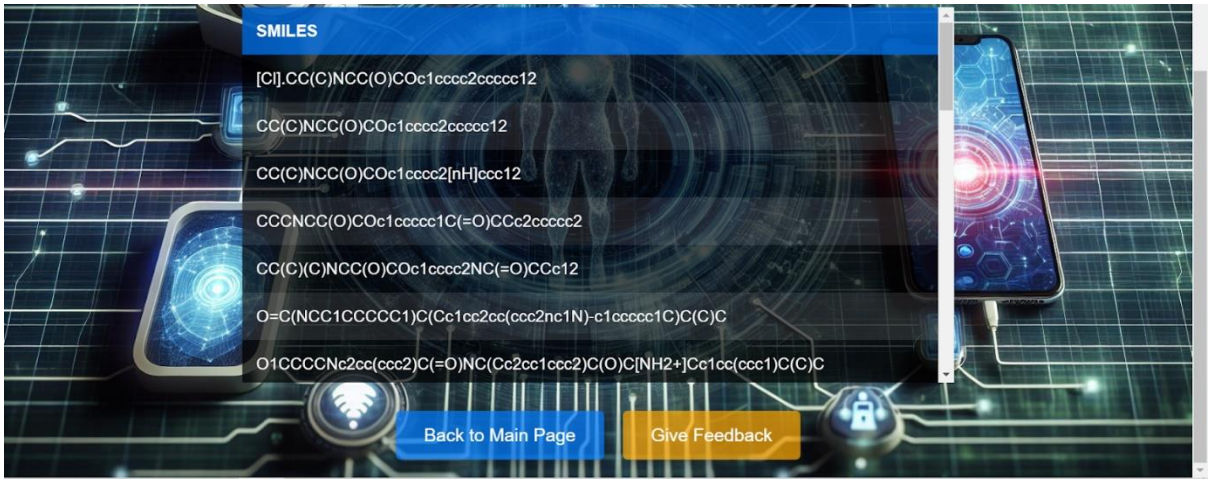


1.7.Searching Molecules from similarity property

Step 1 :Click Molecules searchingbutton on the main page
Step 2 :Enter Formula of Root Molecule



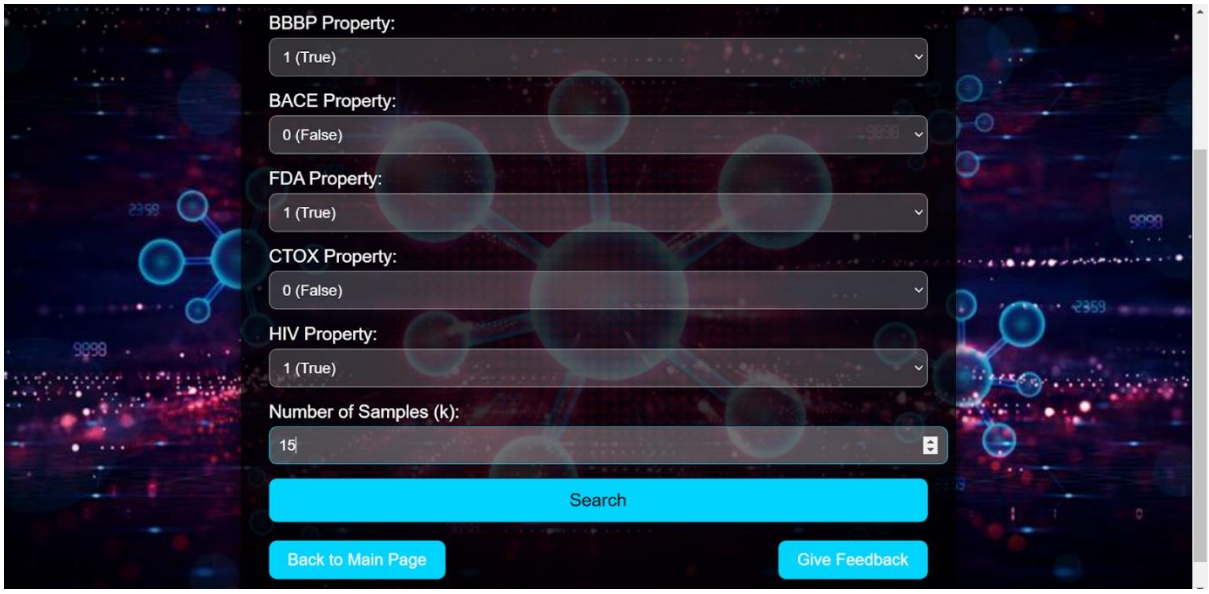
The result will be



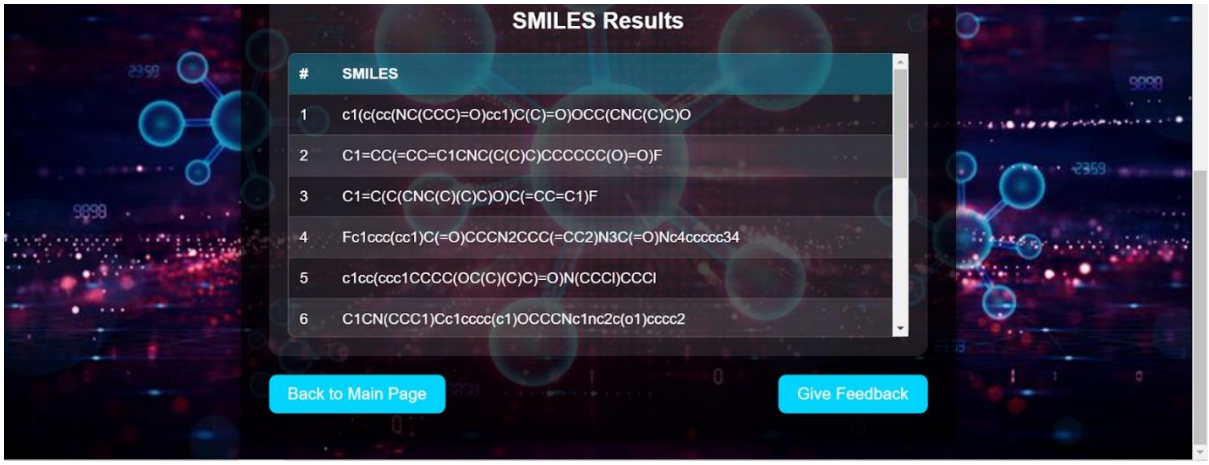
1.8.Searching Molecules with user's conditions

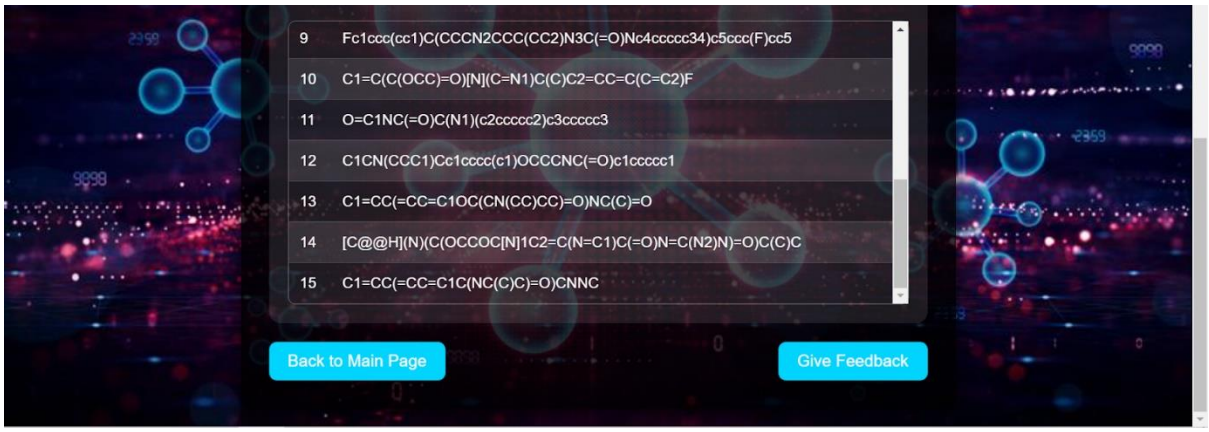
Step 1 :Click Conditional searching button on the main page

Step 2 :Setup conditions for searching



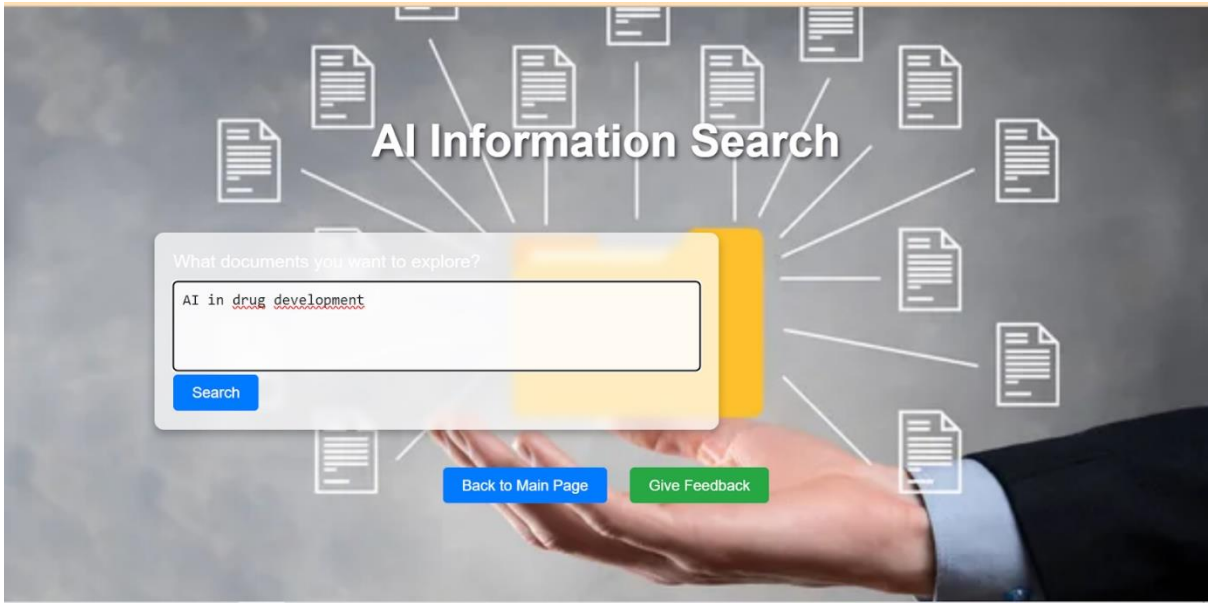
The result will be



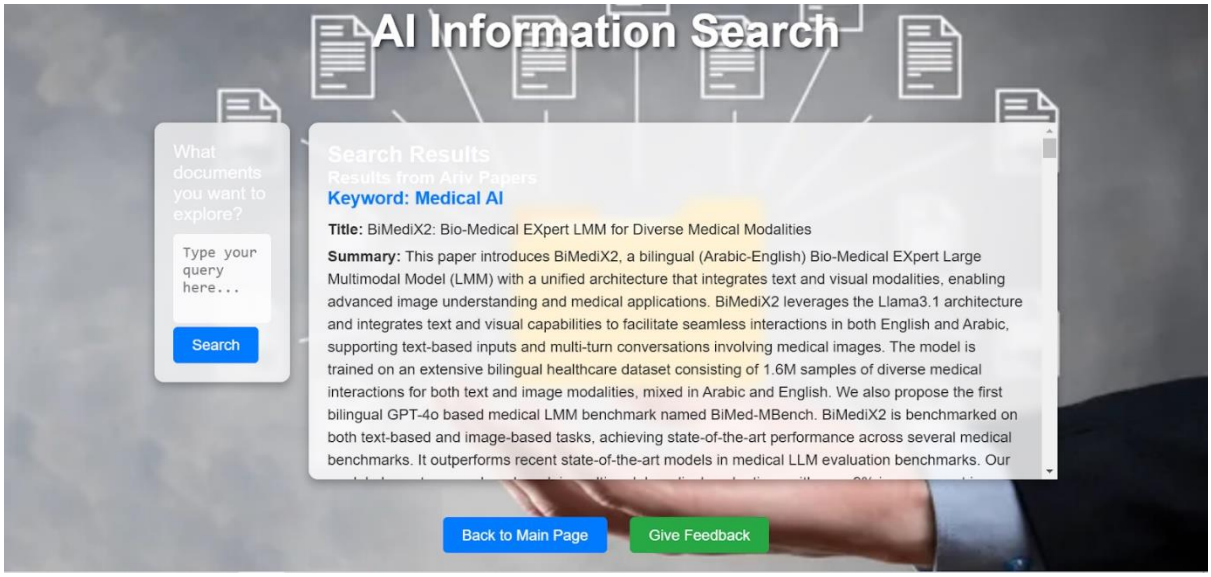


1.9.Retrieve Documents to serve for User's research

Step 1 :Click retrieve documents button on the main page
Step 2 :Enter what type of documents they want to find

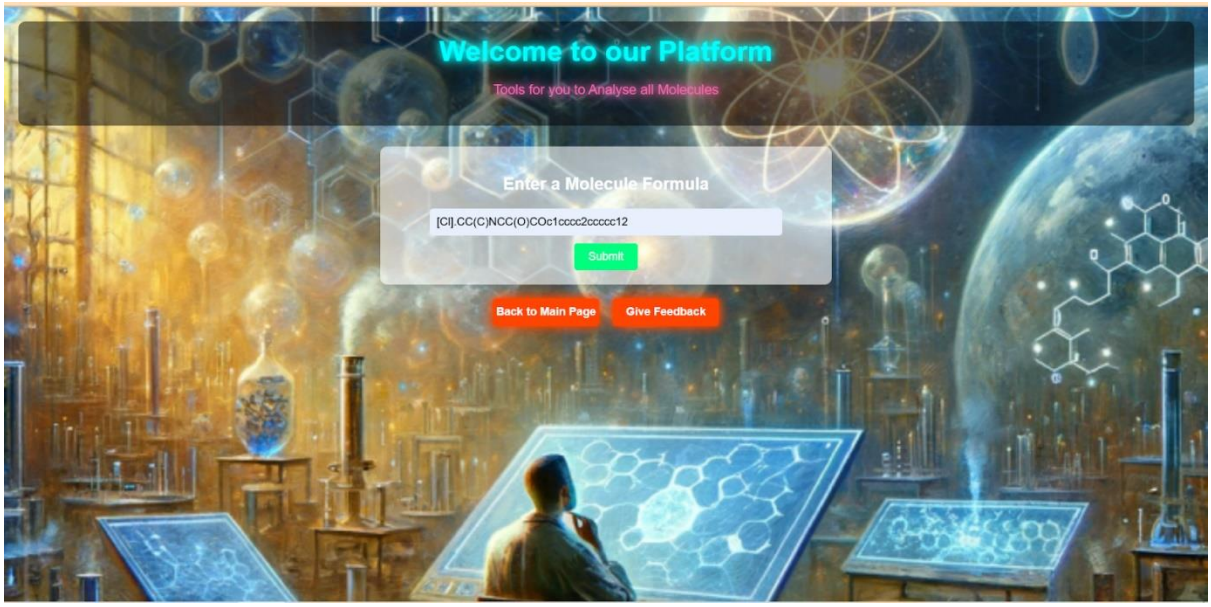


The result will be

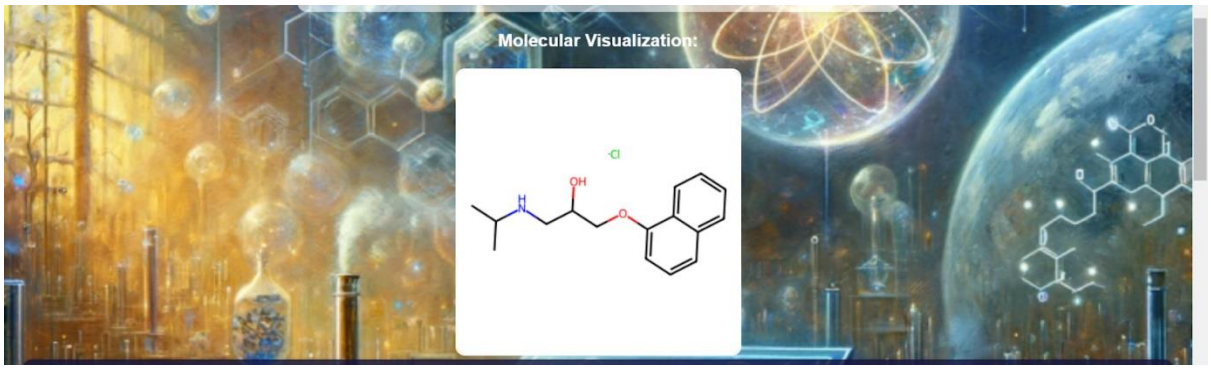


1.10.Advanced Molecule Analysis

Step 1 :Click Molecule Researching button on the main page
Step 2 :Enter Molecule Formula

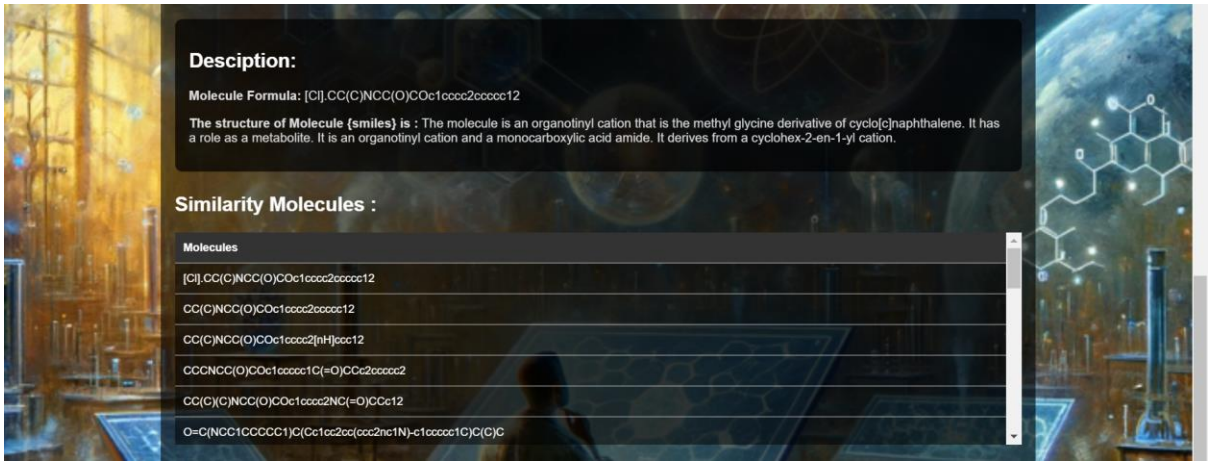


The result will be



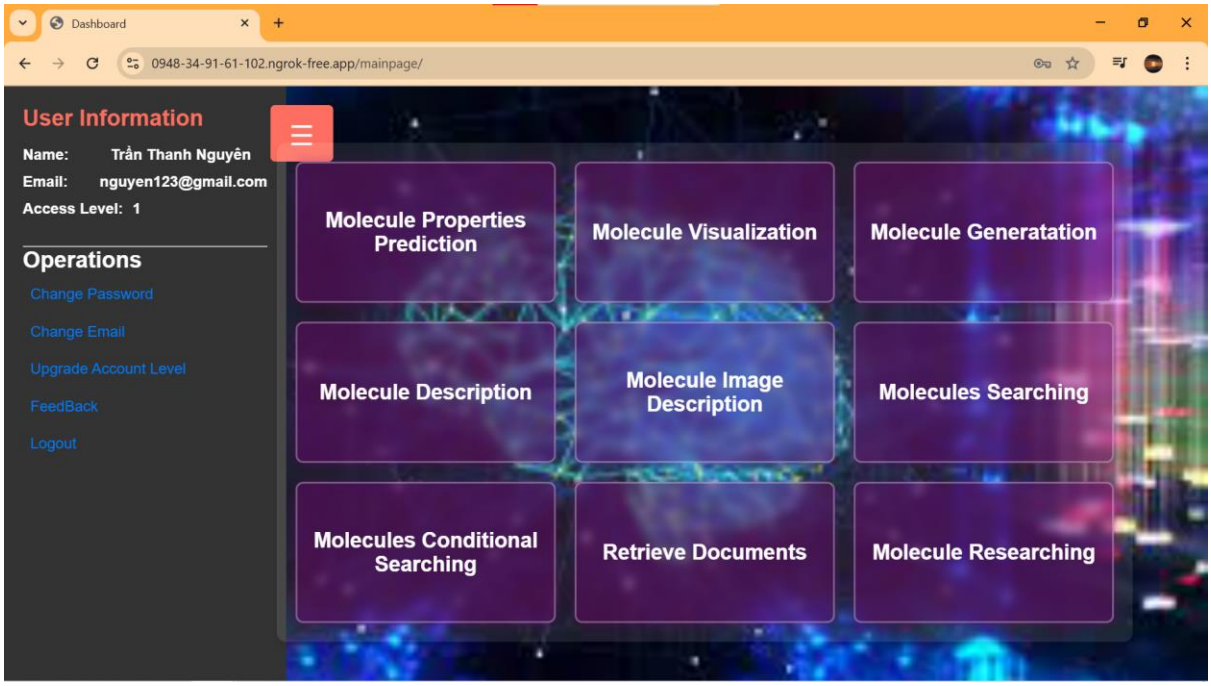
Predictions:

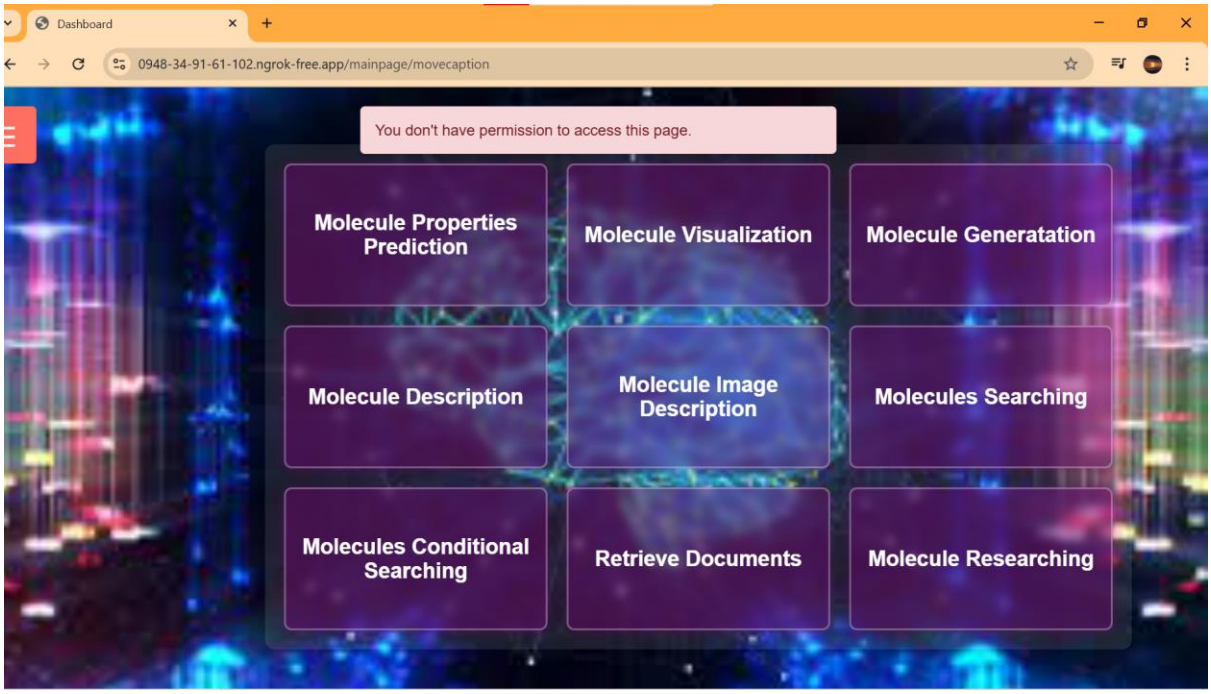
Property	Prediction
Blood-Brain Barrier (BBBP):	Yes
Alzheimer's Risk (BACE):	Yes
Toxicity (CTOX):	No
FDA Approval Chance (FDA):	Yes
Solubility (ESOL):	-0.5232252478599548
Solubility Prediction (FreeSolv):	-1.7088125944137573
Fat Solubility (Lipophilicity):	1.4405909776687622
Molecular Data (QM7):	-1768.4312744140625
HIV Inhibitor Prediction (HIV):	No
Electrostatic Potential 1 (E1CC2):	-0.09425874799489975
Electrostatic Potential 2 (E2CC2):	-0.8125718235969543
Energy Calculation 1 (E1PBE0):	-0.7432737946510315
Energy Calculation 2 (E2PBE0):	-0.5449671149253845
Energy Calculation 1 (E1CAM):	-0.3495720624923706
Energy Calculation 2 (E2CAM):	-0.85101318359375



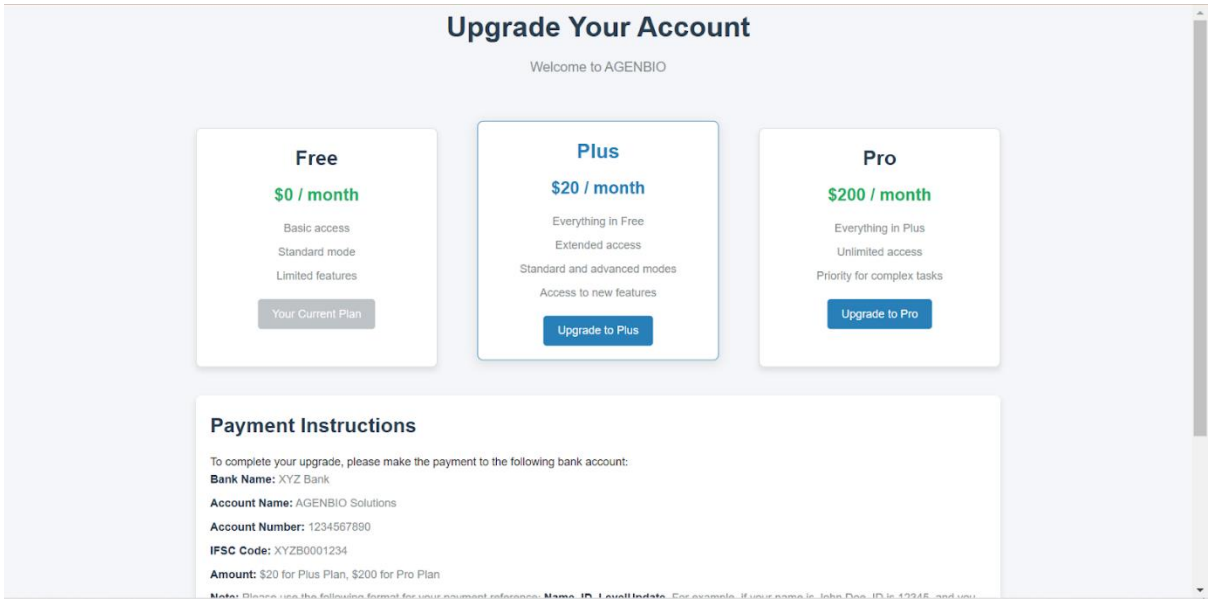
1.11 Upgrade Account Level

By default,users are in level 1 and they are accessible only to Predict and Visualize Features.To access wider features they need to upgrade their account

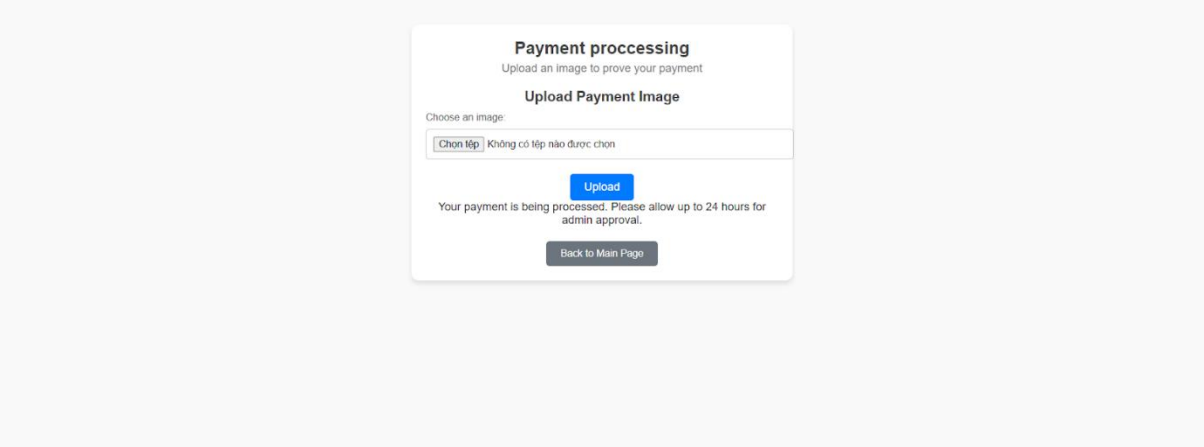




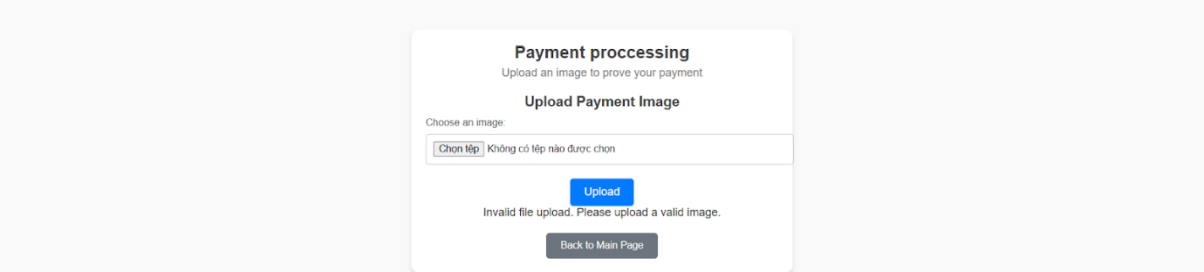
Step1 On the sidebar of mainpage ,click onto update account level button



Step 2 click onto level they want to upgrade(in this case click to Pro),then move to the page to prove your Payment



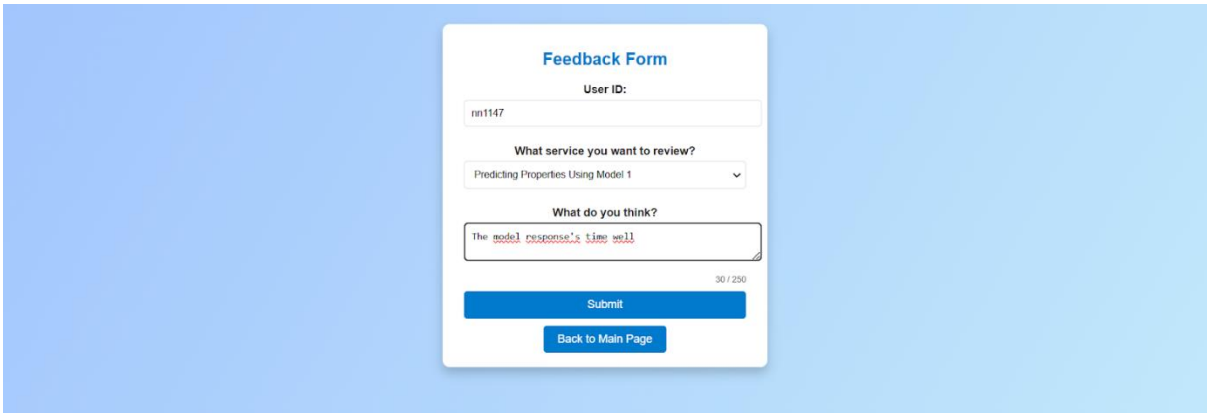
If users uploaded a acceptable image ,they will be received the waiting notify,Otherwise



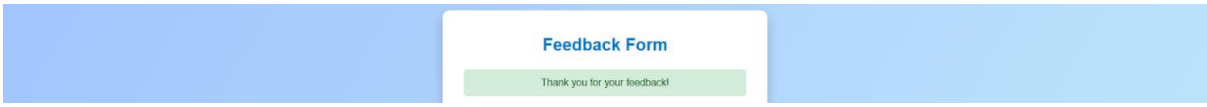
When Admin accept and update their level they will be able to access to features correspond to this level

1.12 Feedback

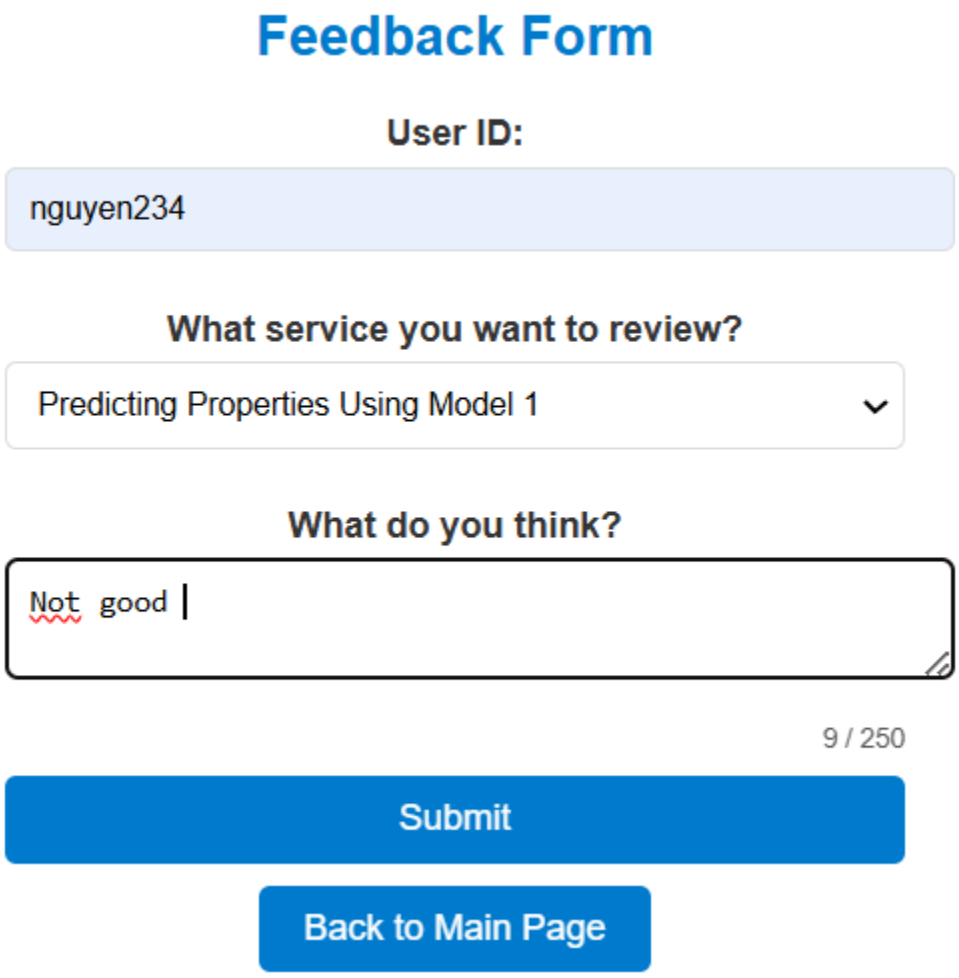
Step 1 ,Open sidebar on web main page,then click to Feedback button
Step 2,They are moved to fill in the feedback form

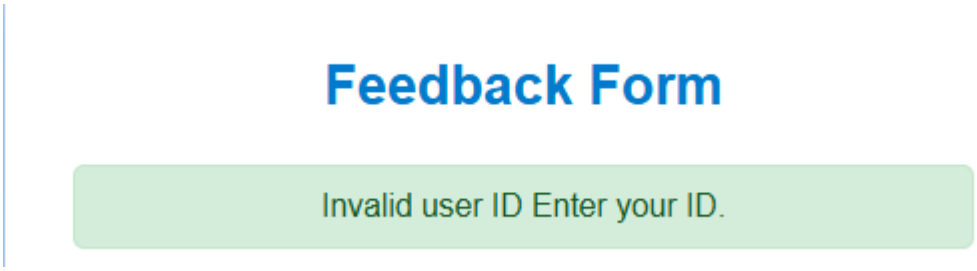


If they enter valid information the successful notify will be



Otherwise(in case they do not enter right UserID)

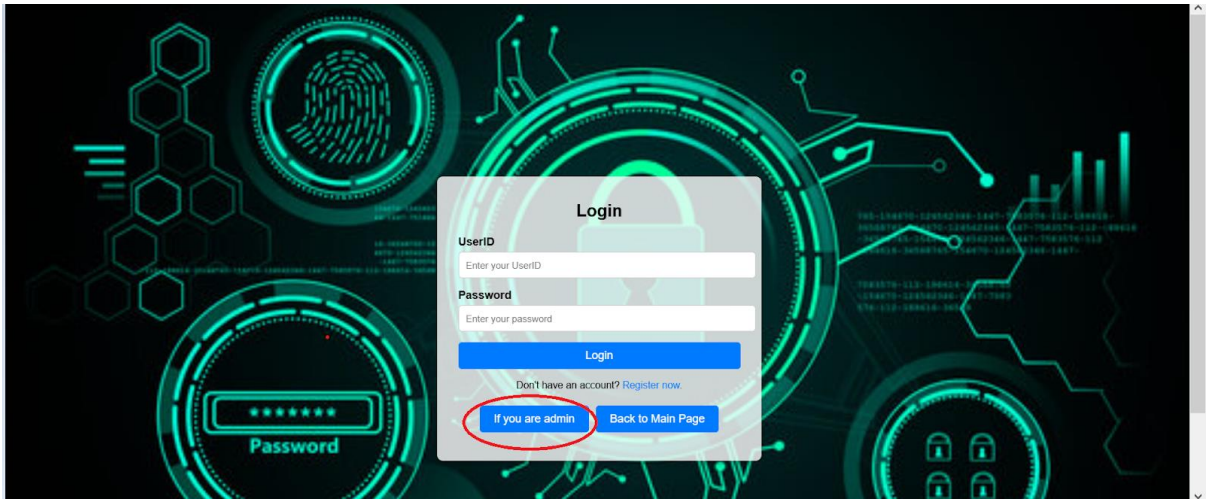




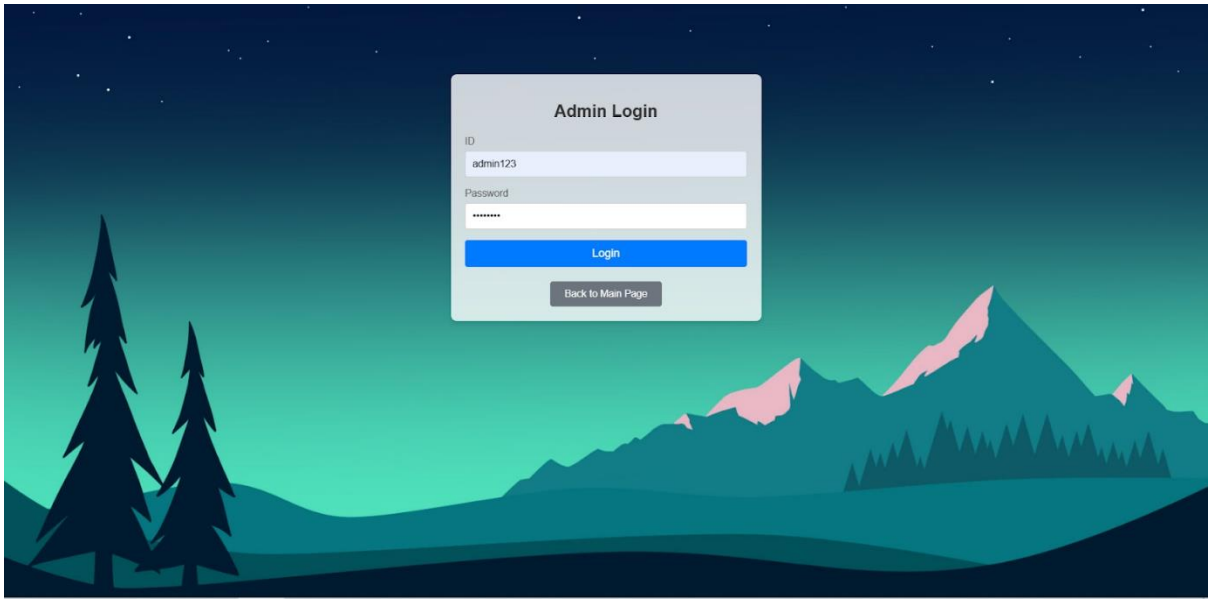
2 Admin use case

2.1 Admin Login

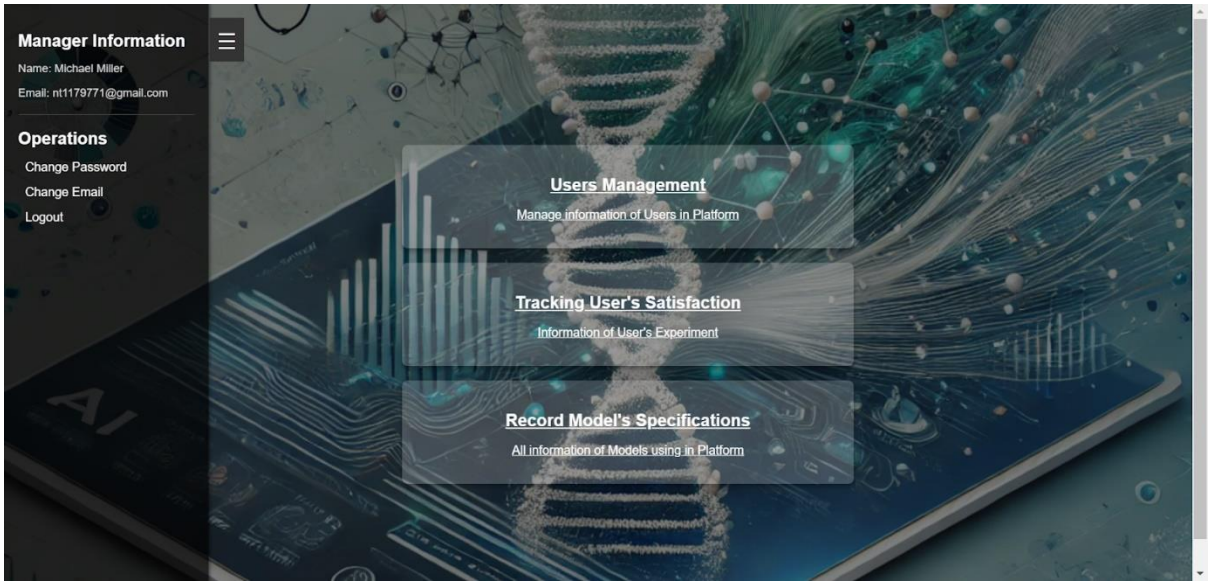
Step 1 On the Login page



Step 2 Enter the information



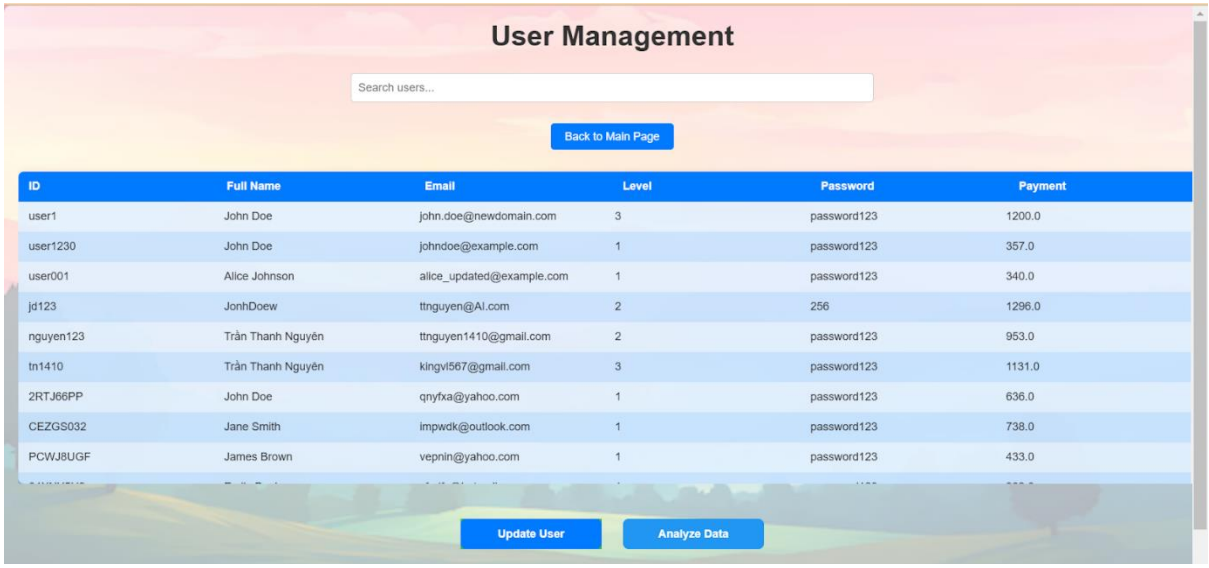
If they enter right information they will be moved to



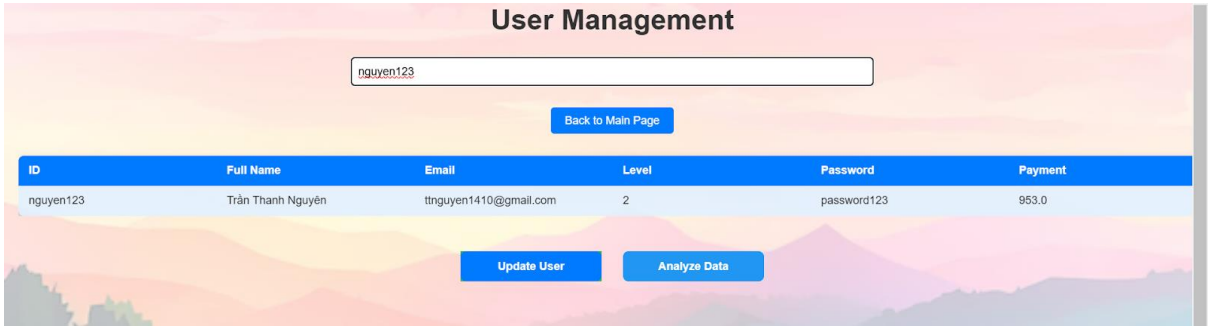
2.2 User's Management

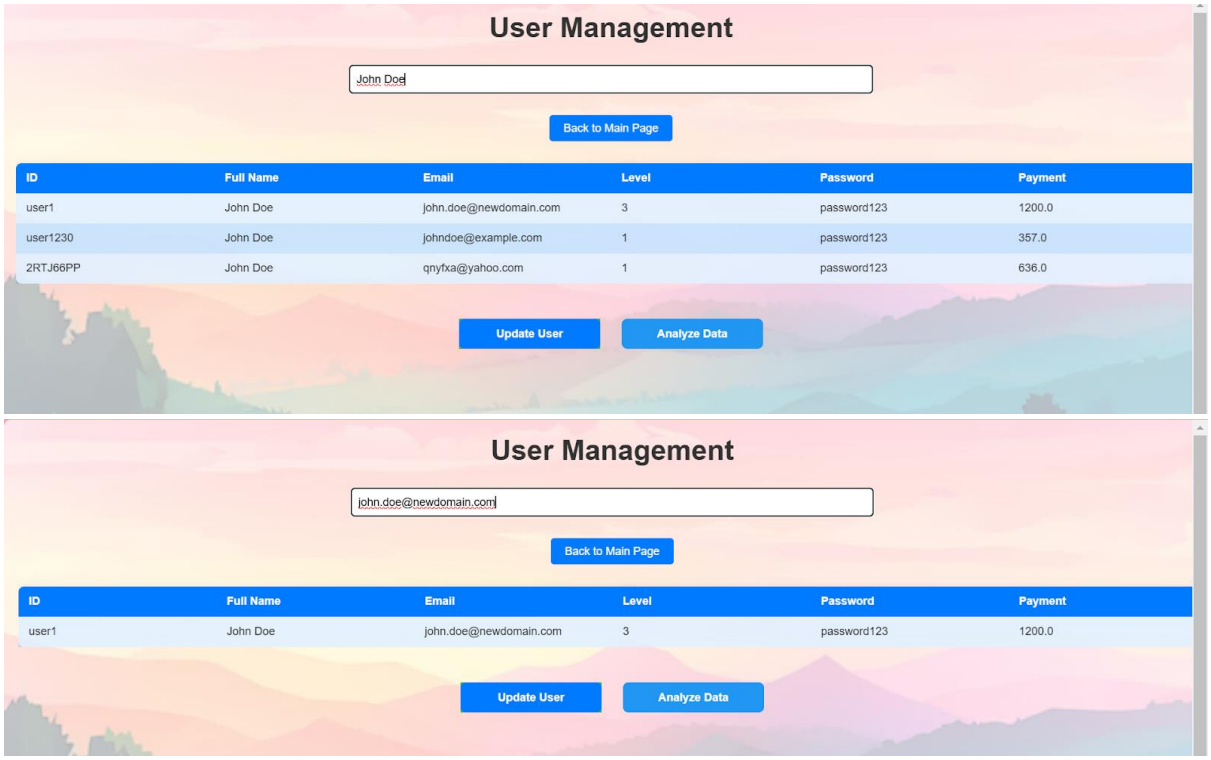
Step 1 Click User Management on the main page

They will be moved to User Management page where displayed all information of users



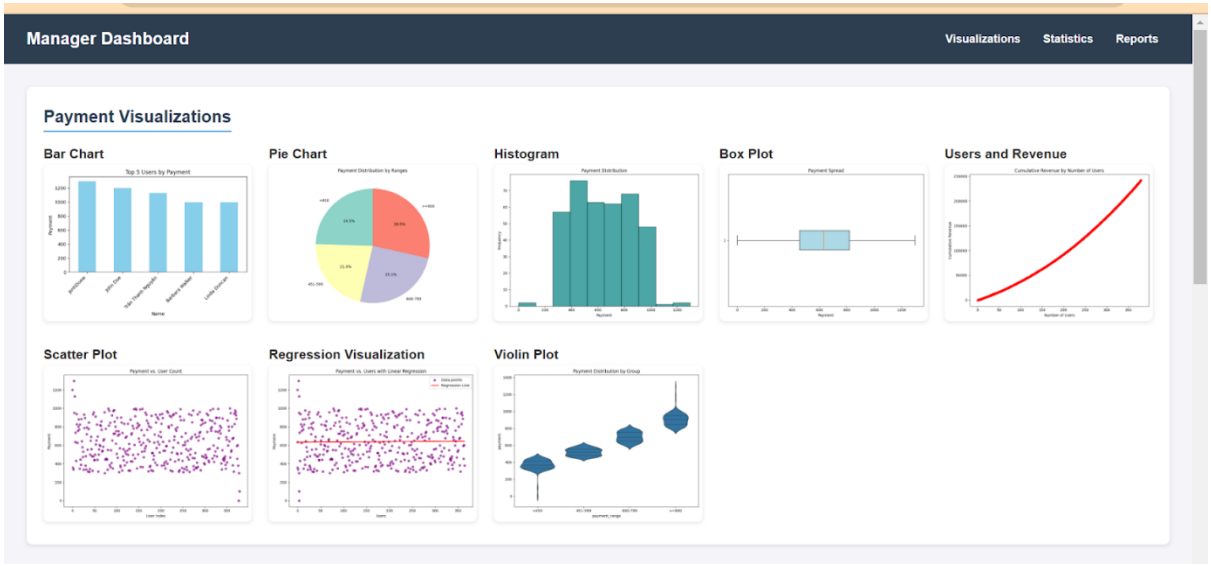
Step 2 .Enter the name,ID ,Email of users to find

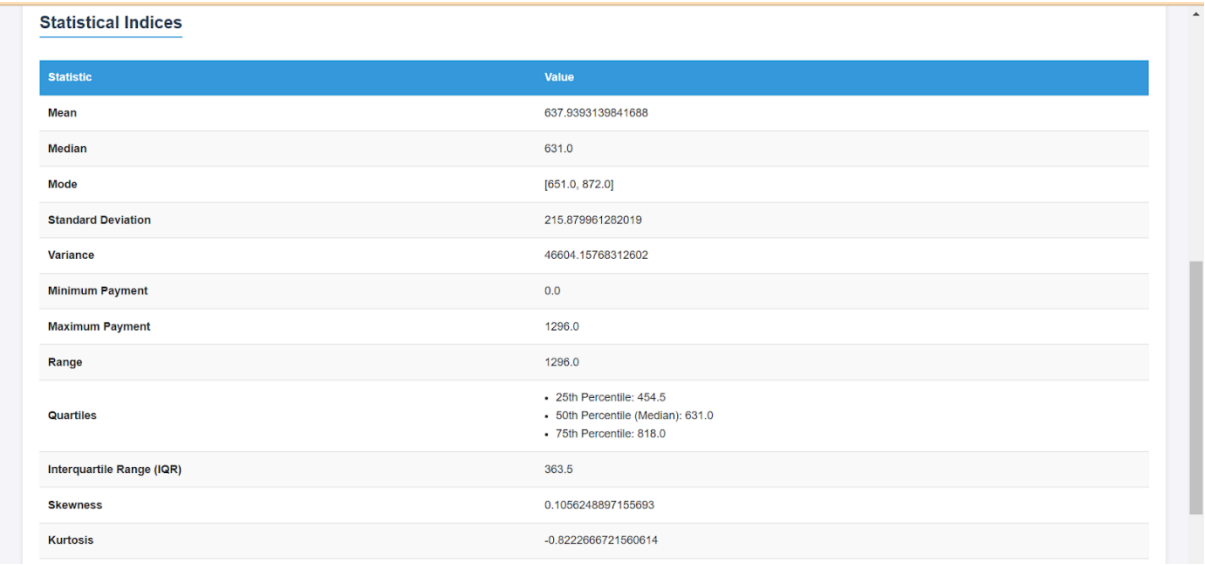




2.2.1 Analyze data of user

Step 1 click Analyze data on User Management Page
The result look like





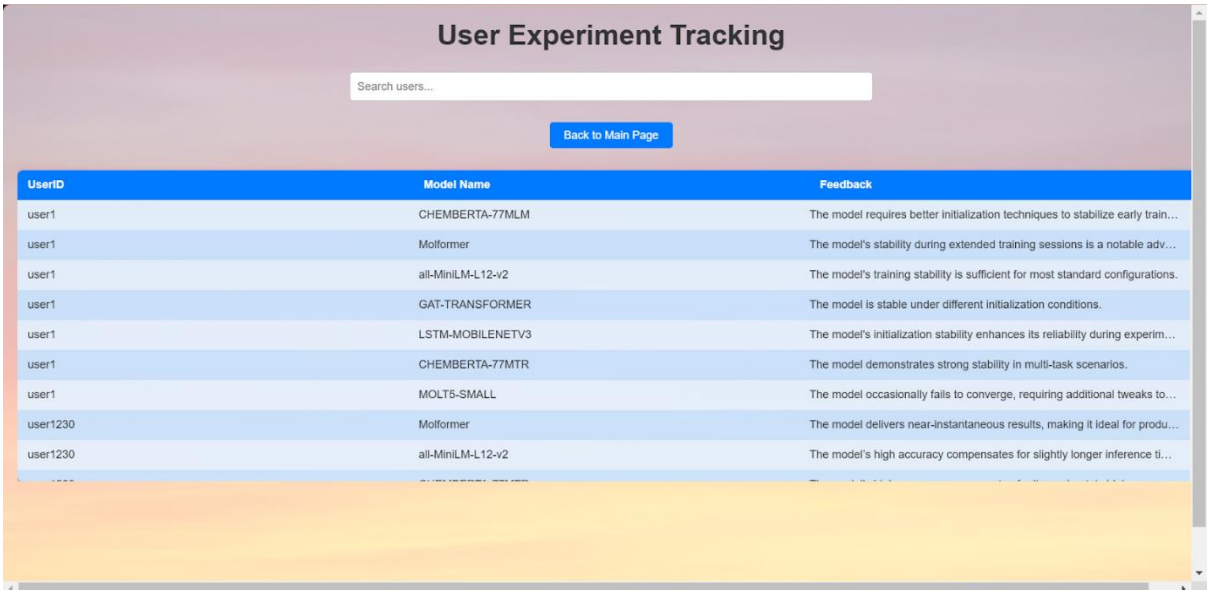
Statistical Indices

Statistic	Value
Mean	637.9393139841688
Median	631.0
Mode	[651.0, 872.0]
Standard Deviation	215.879961282019
Variance	46604.15768312602
Minimum Payment	0.0
Maximum Payment	1296.0
Range	1296.0
Quartiles	• 25th Percentile: 454.5 • 50th Percentile (Median): 631.0 • 75th Percentile: 818.0
Interquartile Range (IQR)	363.5
Skewness	0.1056248897155693
Kurtosis	-0.8222666721560614

There are essential statics and analysis of data for admin to understand the capacity of the Website

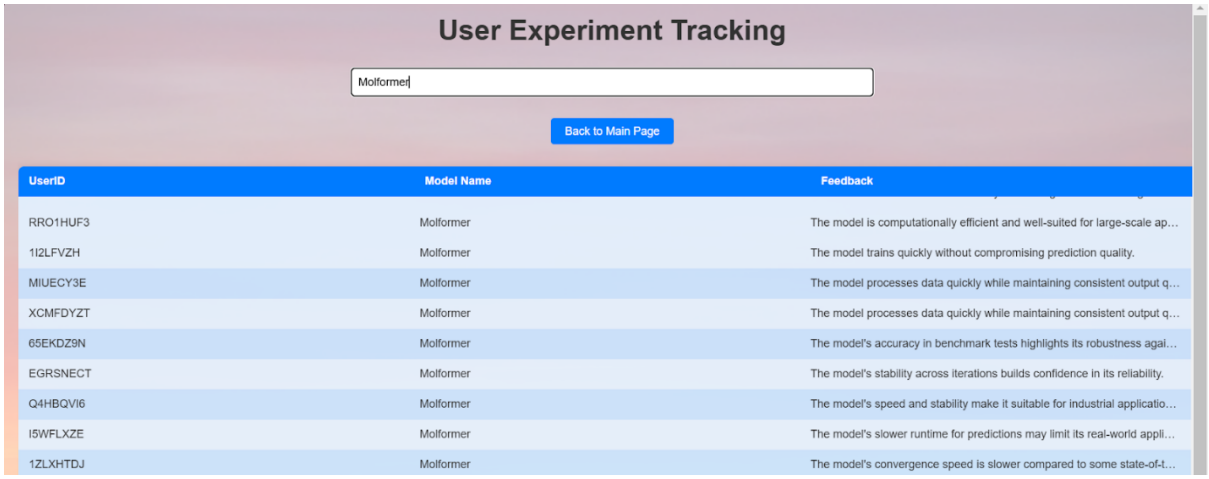
2.3 User's satisfaction tracking

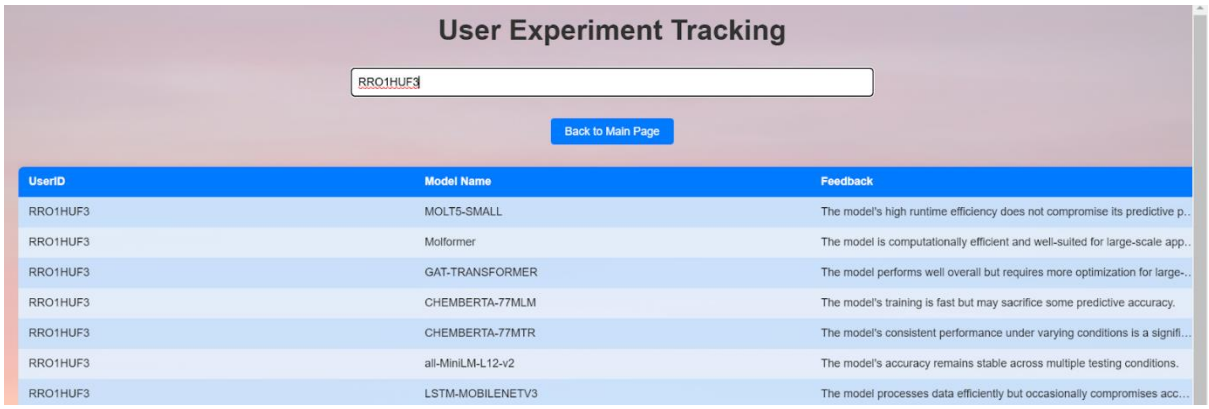
Step 1 On the Admin Main page click Tracking Users Satisfaction button



There are place to store all feedback of users relate to the Model they have used in these Platform

Step 2 Admin can filter User ,Model by enter their name on Search bar ,even though it is designed for user information





2.4 Model's Monitors

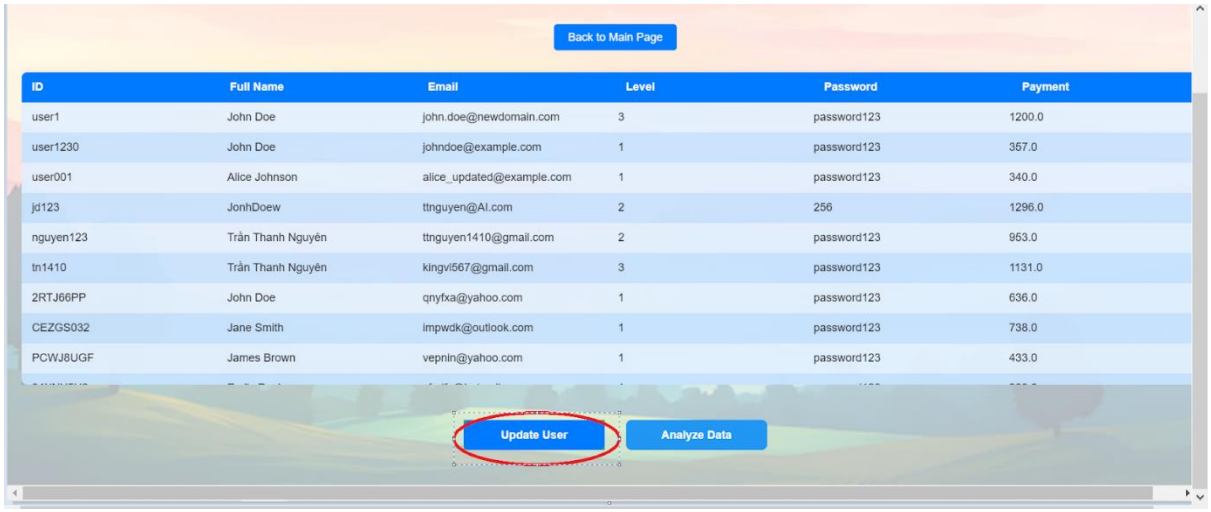
Step 1 Click on Record Model specification button on the Admin main page

Step 2 Filter model

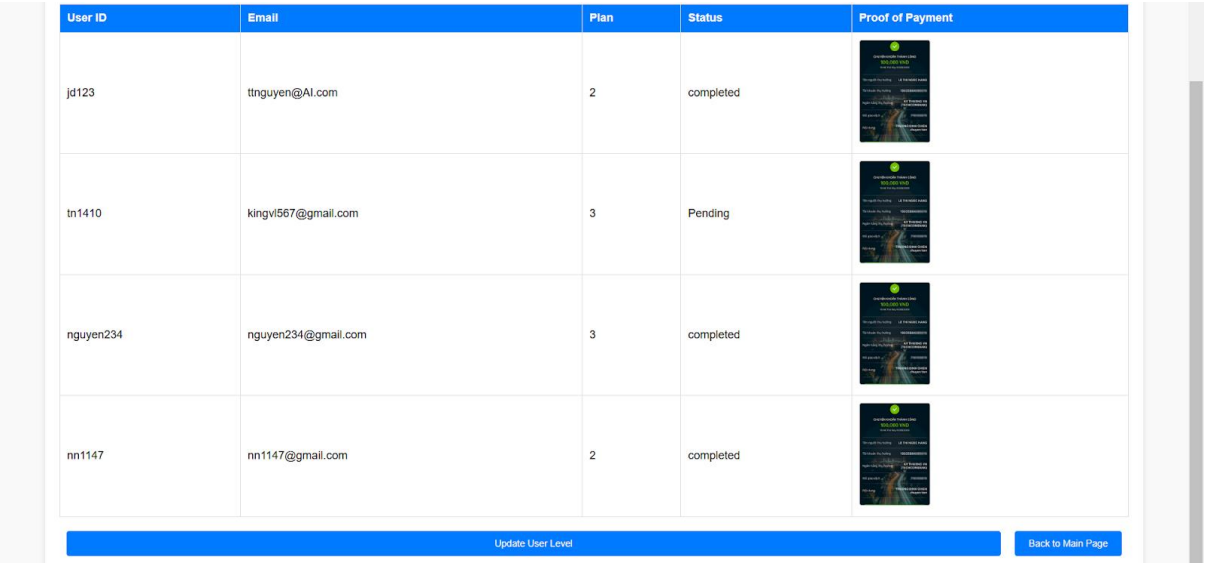
This page is designed for Admin understand the components of each Model using in platform to have a broader viewpoint to suggest innovation to higher level people such as Managers or Developers team

2.5 Users Update Level process

Step 1 On user Management page,click on Update User



Step 2 Check the information

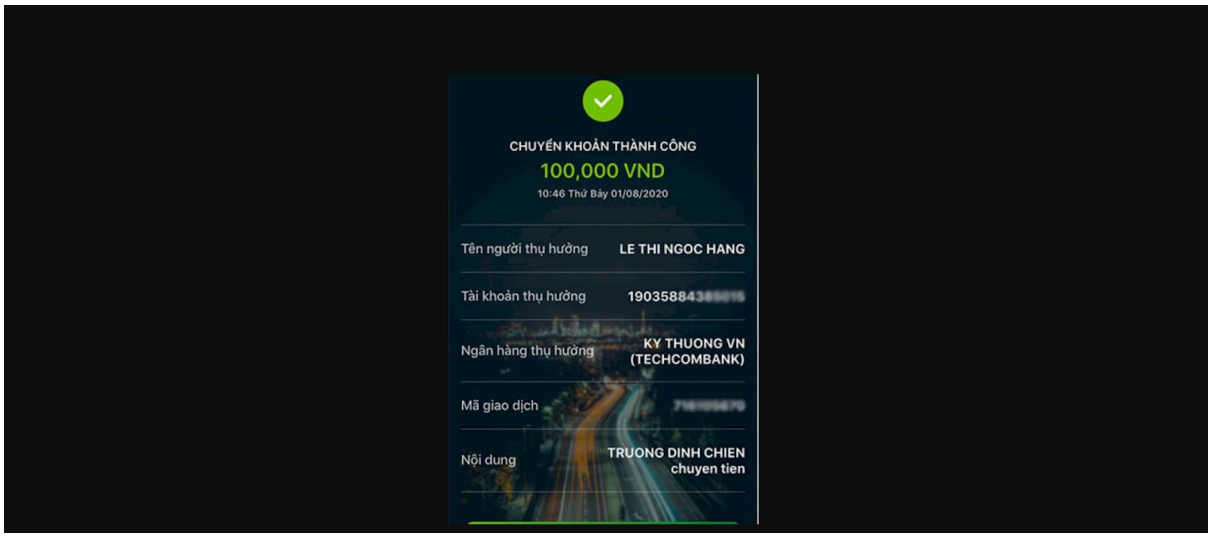


User ID	Email	Plan	Status	Proof of Payment
jd123	ttnguyen@AI.com	2	completed	
tn1410	kingvl567@gmail.com	3	Pending	
nguyen234	nguyen234@gmail.com	3	completed	
nn1147	nn1147@gmail.com	2	completed	

Update User Level | Back to Main Page

Here Admin need to considered cases which have pending status

Step 3 Click onto the image if they want to visualize more clearly



Step 4 Update Level of users

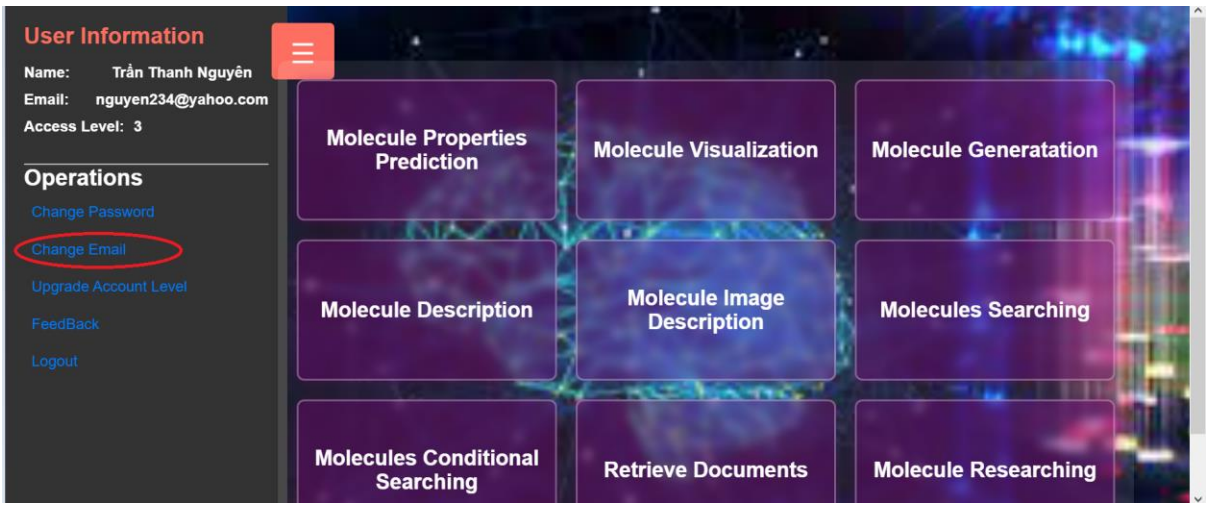
Result

User ID	Email	Plan	Status	Proof of Payment
jd123	ttnguyen@AI.com	2	completed	
tn1410	kingvl567@gmail.com	3	completed	
nguyen234	nguyen234@gmail.com	3	completed	
nn1147	nn1147@gmail.com	2	completed	

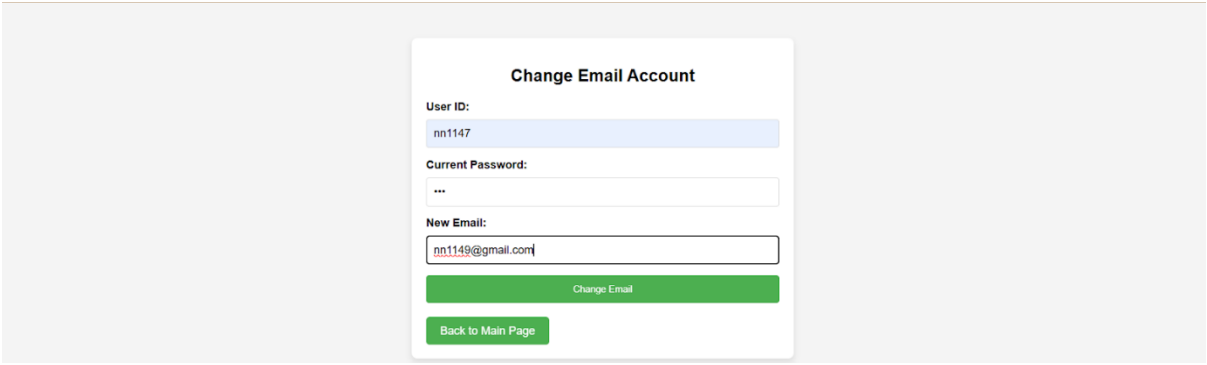
3 Account utilization

3.1 Change Email

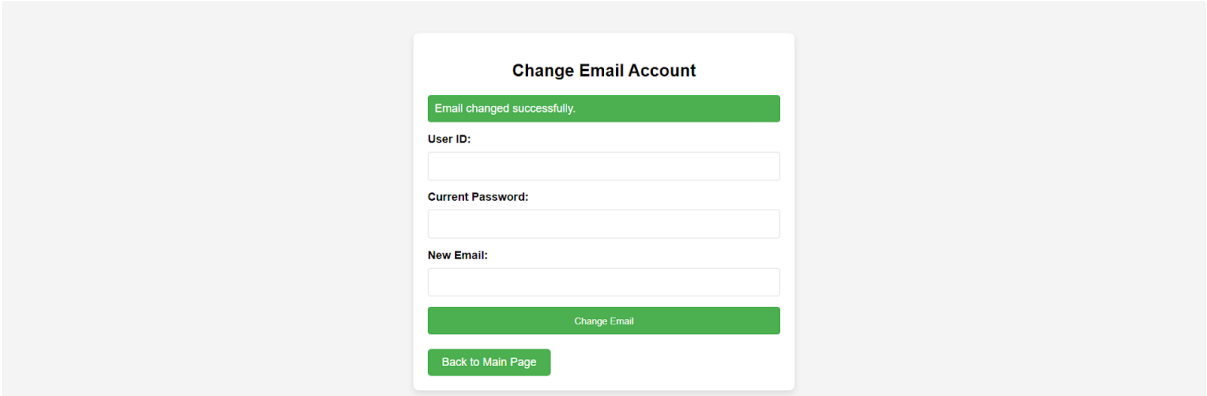
Step 1 click on



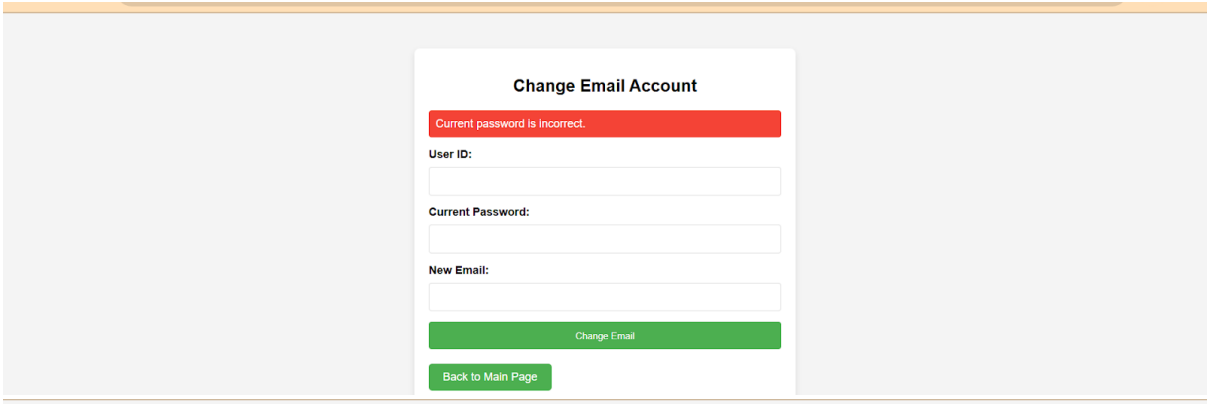
Step 2 Fill in the form

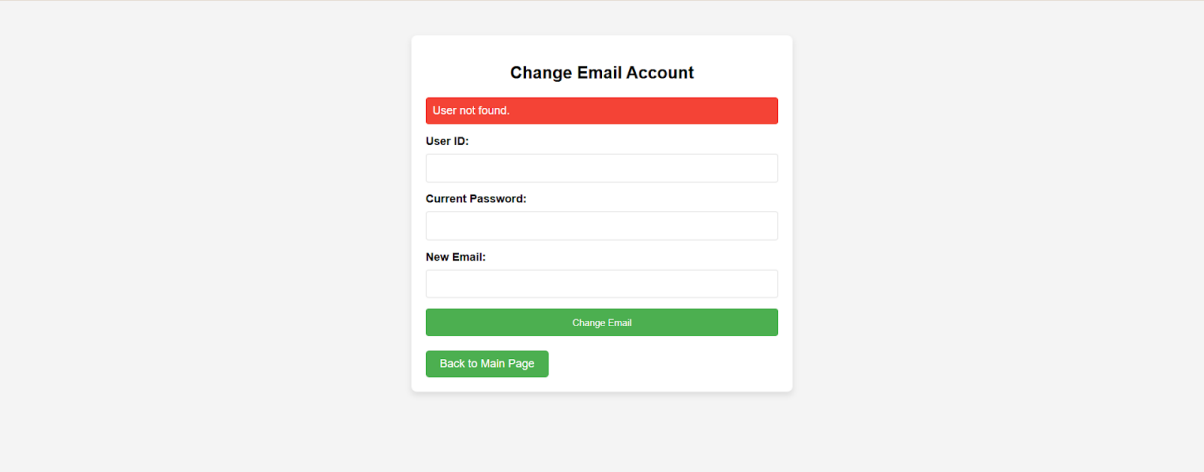


Step 3



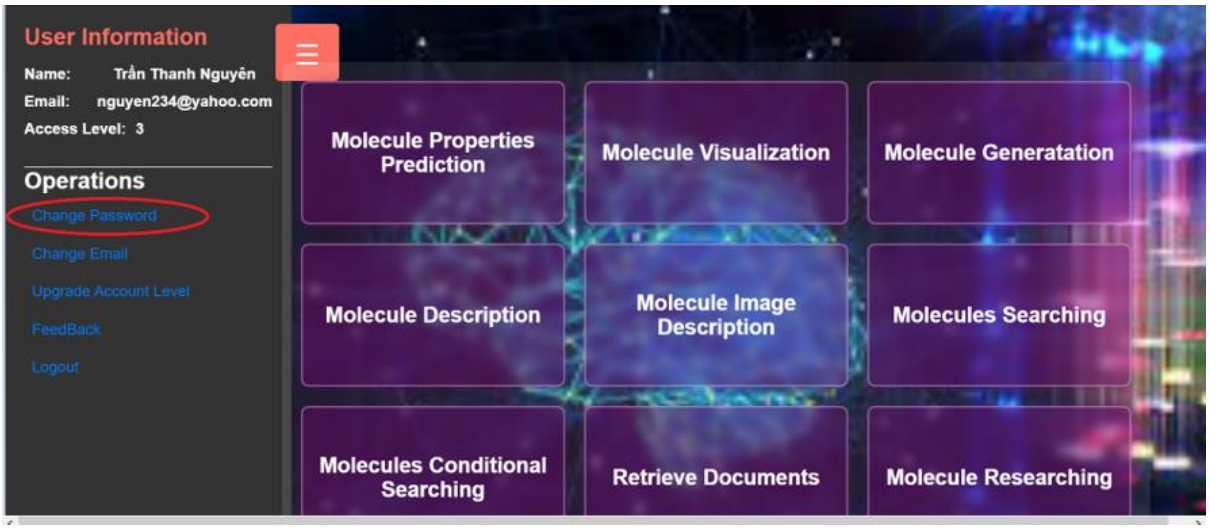
Otherwise if they enter invalid ID or current Password



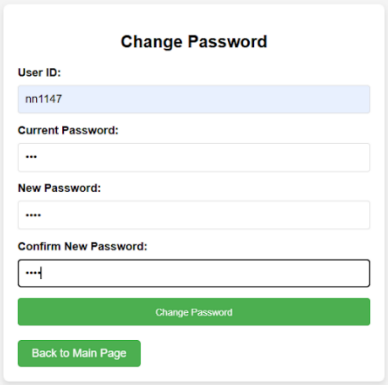


3.2 Change Password

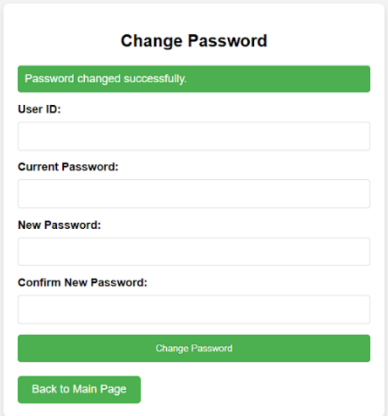
Step 1 :Click on



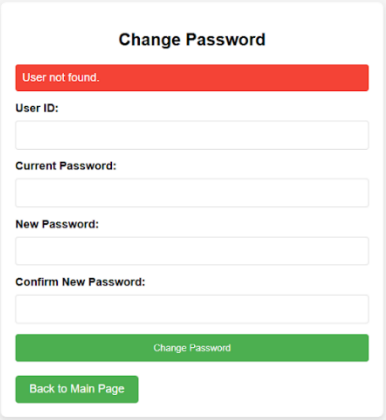
Step 2 :Enter the information

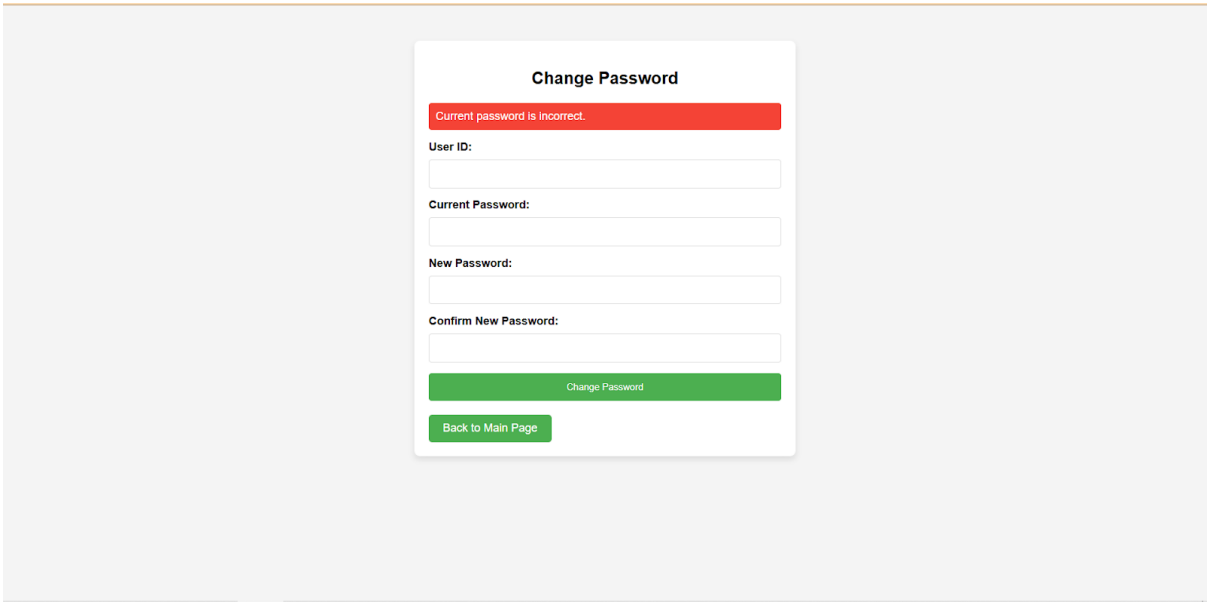


Result



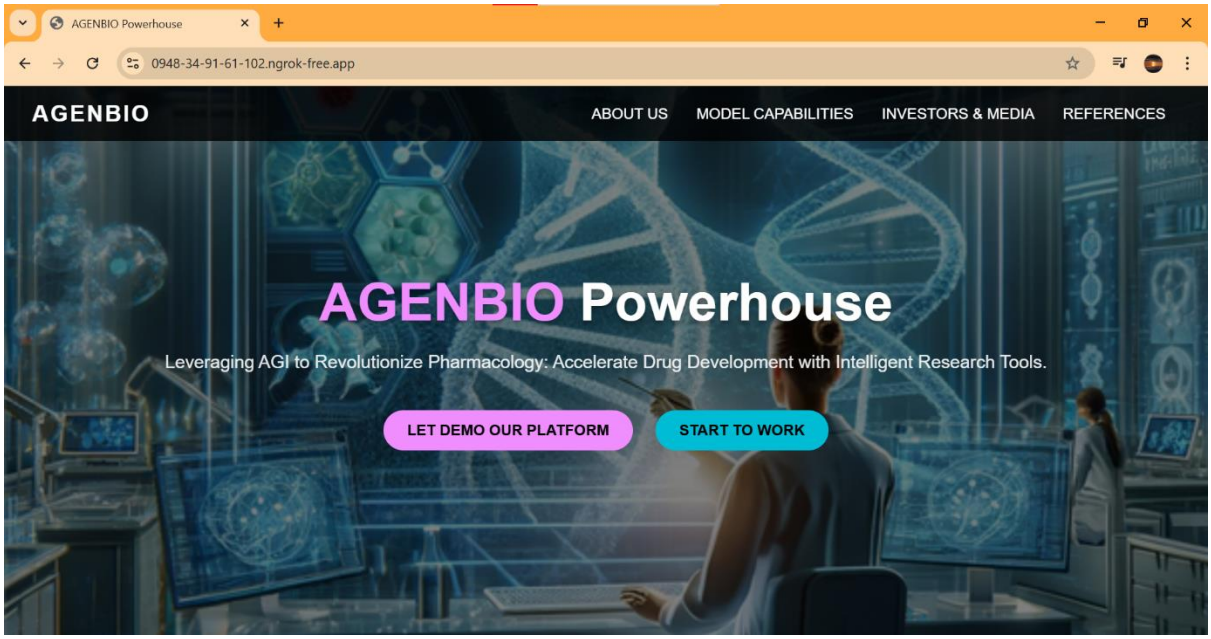
If invalid information input(ID,Current Password)
Otherwise if they enter invalid ID or current Password

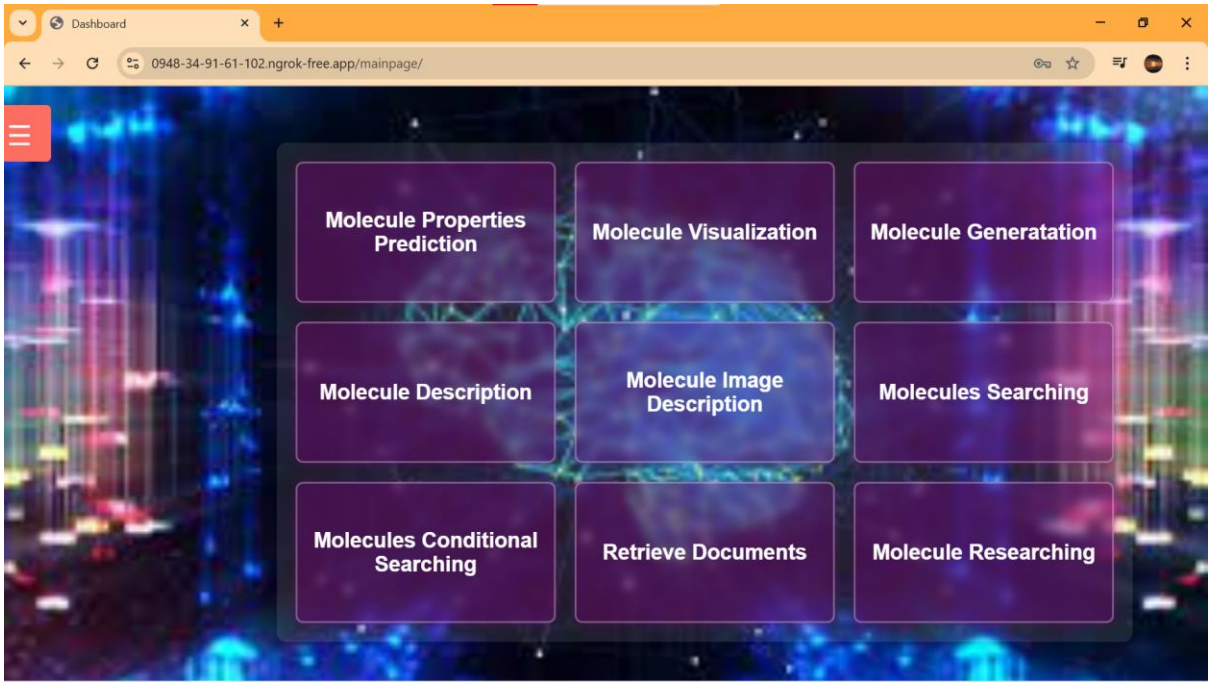


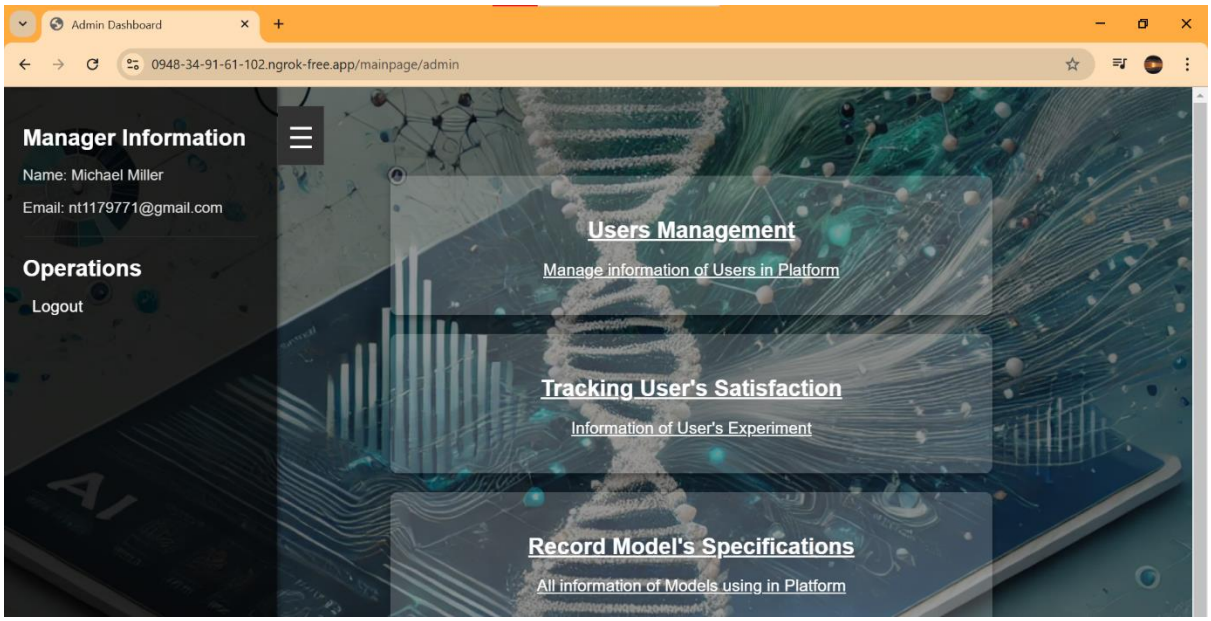


4 Internet Deployment

In this project, we selected the **Ngrok** platform to deploy our application on the internet for several reasons. Ngrok is user-friendly and seamlessly integrates with Google Colab, which provides a robust virtual environment and multiple components necessary for running and testing applications efficiently. Additionally, its compatibility with various devices enhances its utility in diverse deployment scenarios.







IV. Discussion

4.1 Fully Implemented Website Application Features

The AIBIO app project encompasses a comprehensive set of features tailored for website functionality, including:

1. **CRUD Features for Users:**
    - The app provides Create, Read, Update, and Delete (CRUD) capabilities, enabling users to manage their data efficiently.

2. **Administrative Management Abilities:**
    - An admin panel is integrated into the application, allowing administrators to oversee and manage the system effectively.
3. **Access Control Based on Payment Levels:**
    - User access is segmented based on their subscription or payment level. This ensures tailored user experiences and appropriate feature availability according to account tiers.
4. **Database Storage:**
    - All data generated or modified within the app is stored securely in a robust and scalable database system.

These features collectively enhance the functionality, usability, and security of the AIBIO website application, making it a fully operational platform.

4.2 Integration of AI Model

The AIBIO app integrates advanced Artificial Intelligence (AI) models to enhance its core functionalities, utilizing deep learning (DL) and machine learning (ML) techniques for robust performance across various tasks. Key aspects of AI model integration include:

1. **Document Retrieval**:
   The app employs DL and ML algorithms to efficiently retrieve relevant documents such as scientific articles, patents, and regulatory guidelines. Natural Language Processing (NLP) models are used for semantic search and contextual ranking, ensuring that users access the most pertinent and up-to-date information.
2. **Handling Diverse Input Types**:
   AI models are designed to process a wide range of molecular input types, including chemical structures, SMILES strings, and molecular images. This flexibility ensures the app can handle diverse datasets, accommodating the varied needs of researchers in drug development.
3. **Task Versatility**:
   AI integration supports multiple computational tasks crucial for drug development:
    - **Prediction Tasks**: Classification and regression models predict key molecular properties such as toxicity, pharmacokinetics, solubility, and binding affinity.
    - **Generation Tasks**: Generative models, such as deep generative networks, create novel molecular structures with optimized therapeutic properties.
    - **Description Tasks**: NLP techniques are used to annotate and describe molecular functions, bridging computational insights with biological applications.
    - **Searching Task**: Given the vast molecular space (over a billion samples), the app provides faster, more flexible search capabilities for researchers, helping pharmacists efficiently locate relevant molecules.

4. **Molecular Image Processing**:
   Advanced image recognition models process and analyze molecular images, enabling feature extraction and property prediction from visual data. This capability enhances the interpretation of chemical diagrams and molecular pathways, providing valuable insights into molecular behavior.

This AI-driven integration ensures that the AIBIO app is a powerful tool for drug development, offering fast, accurate, and flexible solutions for researchers and pharmacists.

4.3 Enhancing AI Features for Optimal Performance

To elevate the AIBIO app's AI capabilities, we incorporate cutting-edge architectures and methodologies designed to boost prediction accuracy, processing efficiency, and result relevance. The following strategies focus on optimizing pretrained models to ensure superior performance across core functionalities:

1. **Property Prediction**
   To refine the accuracy of molecular property predictions, we integrate a **Graph Neural Network (GNN)** with the pretrained **ChemBERTa** model. The GNN enhances ChemBERTa's ability to understand and interpret molecular graphs, such as atomic connections and bond types, by applying graph-based representations. This hybrid model dramatically improves prediction accuracy for critical molecular properties, including solubility, toxicity, and binding affinity—key factors in drug development.
2. **Molecular Image Processing**
   When it comes to analyzing molecular images, we deploy the **LSTNet architecture**, which combines **Long Short-Term Networks (LSTNet)** with a pretrained **Convolutional Neural Network (CNN)**. This powerful combination excels at extracting both spatial and temporal features from complex molecular diagrams. With its deep learning capabilities, LSTNet delivers fast and accurate image-based molecular analysis, offering researchers rapid insights with remarkable reliability.
3. **Document Retrieval**
   To enhance the speed and accuracy of document retrieval without the computational overhead of large-scale models, we implement a **two-stage approach**:
   - **Keyword Extraction**: A lightweight NLP model extracts essential terms and phrases from scientific documents and stores them in an organized database.
   - **Similarity Matching**: When a user submits a query, the model compares the extracted keywords to those in the database, measuring similarity using **cosine similarity**. This strategy strikes the perfect balance between efficiency and relevance, ensuring quick and accurate access to scientific papers without taxing computational resources.
4. **Molecule Similarity Search**
   For molecule similarity searches, the **MolFormer pretrained model** takes center stage. MolFormer excels at encoding molecular representations into high-dimensional

vectors, capturing the intricate chemical properties that define each molecule. The database is organized by chemical properties, and similarity searches are conducted within each segment. By consolidating top matches across segments, this approach ensures precise identification of molecules, even across vast and diverse chemical spaces.

5. **Conditional Molecule Search by Chemical Properties**
   For more tailored searches, we enable molecule filtering based on specific **chemical properties**. Each property group is assigned unique feature vectors, allowing the system to sequentially narrow down results by filtering molecules step by step. This approach offers **exceptional precision and scalability**, ensuring users can efficiently search large datasets while maintaining highly relevant results.

By integrating these sophisticated enhancements, the AIBIO app takes its capabilities to the next level. With cutting-edge models and innovative AI architectures, the app not only improves accuracy in property predictions, molecular analysis, and document retrieval but also provides researchers with powerful, intuitive tools for drug development. The result is a seamless, high-performance experience that guarantees satisfaction and delivers meaningful insights in a fraction of the time.

5 Conclusion

Learning Outcomes

Through the development and implementation of this project, we have gained valuable insights and practical experience in the following areas:

- **Building a Complete Web Application**: We successfully developed a fully functional website using the Flask framework and SQLAlchemy for database management.
- **AI Integration**: We effectively integrated AI models into the application to enhance its functionality and deliver intelligent features.
- **Deployment**: The application was deployed on Ngrok for customer demonstrations, showcasing its capabilities in a real-world environment.

Limitations

Despite these achievements, there are areas requiring further improvement:

1. **Feature Expansion**:
   - Implement advanced authentication mechanisms, such as OAuth 2.0, enabling users to log in using Gmail, Facebook, or other third-party accounts.
   - Automate the handling of user requirements across features currently managed manually by customers and admins. This could be achieved by leveraging

advanced AI technologies, such as Computer Vision and NLP models, to enhance the user experience.

2. **AI Model Optimization**:
    - Improve the performance of existing AI models to enhance their capacity, reduce response times, and increase prediction accuracy.
    - Enhance overall user satisfaction by refining AI-driven features and ensuring their reliability and responsiveness.

Overall Assessment

This project has provided a comprehensive learning experience, from web development and AI integration to deployment. While the current application demonstrates strong functionality, addressing the mentioned limitations and incorporating additional features will significantly enhance its utility and user experience. Moving forward, we are committed to refining the application to better meet user needs and deliver a seamless, high-performance solution.