VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY

UNIVERSITY OF SCIENCE

Faculty of Information Technology

CHAT BOT - PA3

SOFTWARE ARCHITECTURE DOCUMENT

by

Phan Thế Anh 22127021

Trần Quốc Huy 22127165

Trần Gia Khang 22127181

Võ Anh Quân 22127352

Nguyễn Hoàng Sang 22127361

Phan Văn Tài 22127372

Under the supervision of

Teacher: **Nguyễn Minh Huy**

Assistant Teacher: **Phạm Hoàng Hải**

Assistant Teacher: **Ngô Ngọc Đăng Khoa**

NOVEMBER 28th 2024

<Chat Bot Application>
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Table of contents

Revision History..... 3
1. Introduction..... 6
1.1 Purpose..... 6
1.2 Scope..... 6
1.3 Definitions, Acronyms, and Abbreviations..... 6
1.4 References..... 6
- MVC:..... 6
+ https://www.geeksforgeeks.org/mvc-design-pattern/..... 6
1.5 Overview..... 7
2. Architectural Goals and Constraints..... 7
2.1 Goals..... 7
2.2 Constraints..... 7
3. Use Case Model..... 8
3.1 Authentication and Authorization:..... 8
3.2 Chat with AI Agents:..... 9
3.3 Create and manage AI BOT:..... 10
3.4 Knowledge Data Management:..... 11
3.5 Prompt Management:..... 12
3.6 Account Management:..... 13
3.7 Photo Chating:..... 14
3.8 Email with AI Agents:..... 14
4. Logical View..... 15
4.1 Authentication and Authorization:..... 15
4.2 Chat with AI model..... 16
Responsibility:..... 16
AIChatManager Features:..... 18
ChatThread Features:..... 18
AIModel Features:..... 19
4.3 AI BOT Management:..... 19
4.4. knowledge dataset management..... 21
4.5 Prompt management..... 23
4.6.1 Upgrade Account to Pro:..... 24
4.6.2 Monetization:..... 24
4.7 AI Image Q&A:..... 27
4.8 email with ai agent..... 28
AIEmailManager Features:..... 30

5. Deployment..... 31
6. Implementation View..... 31

1. Introduction

The purpose of this document is to provide a detailed description of the software architecture for the AI chatbot website. This architecture document is essential to guide the development and ensure consistency, scalability, and maintainability throughout the project lifecycle.

1.1 Purpose

The AI chatbot website is designed to offer seamless communication between users and an AI-Model system capable of understanding and responding to queries effectively.

1.2 Scope

The AI chatbot website includes:

- A user-friendly interface .
- Integration with AI models for natural language understanding and generation.
- Secure and scalable deployment .

The platform is intended for businesses, educators, and individuals seeking efficient AI-driven communication.

1.3 Definitions, Acronyms, and Abbreviations

- **AI**: Artificial Intelligence
- **API**: Application Programming Interface
- **HTTPS**: Hypertext Transfer Protocol Secure

1.4 References

- **MVC:**
  - https://www.geeksforgeeks.org/mvc-design-pattern/
- **Class diagram:**
  - https://www.lucidchart.com/pages/uml-class-diagram
  - Relationship in class diagram, https://votanlanh.wordpress.com/2017/08/08/mot-so-quan-he-giua-cac-class-trong-uml/

1.5 Overview

This document outlines the architectural goals, use-case models, logical view, deployment, and implementation strategies for the AI chatbot website. The architecture emphasizes scalability, security, and maintainability to meet current and future requirements.

2. Architectural Goals and Constraints

2.1 Goals

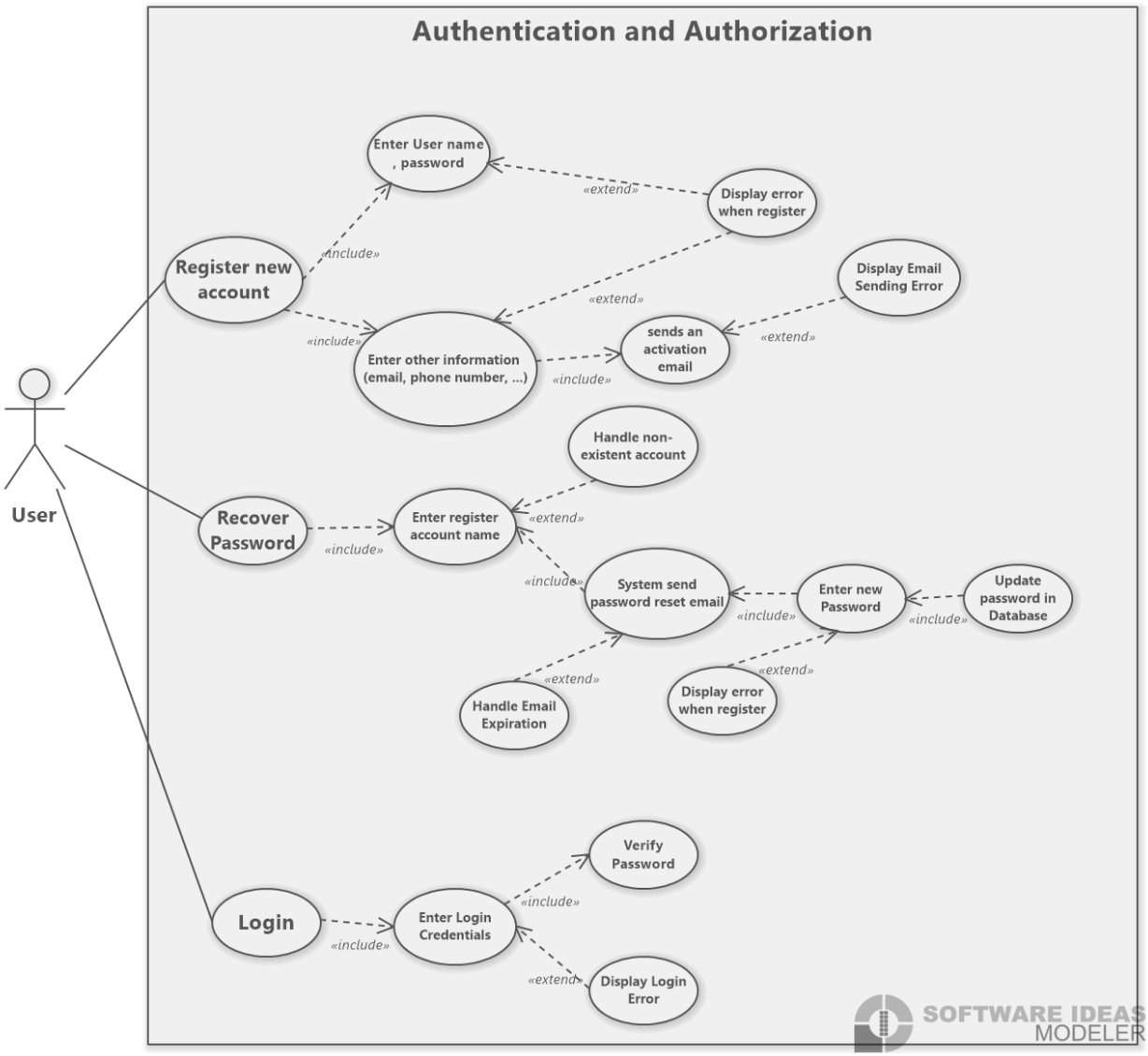
- **Scalability**: Ensure the chatbot can handle increasing user loads without degradation in performance.
- **Security**: Protect user data and interactions through authentication mechanisms and encrypted communication.
- **Performance**: Deliver responses within an acceptable time .
- **Accessibility**: Provide a responsive design that works across web and mobile platforms.

2.2 Constraints

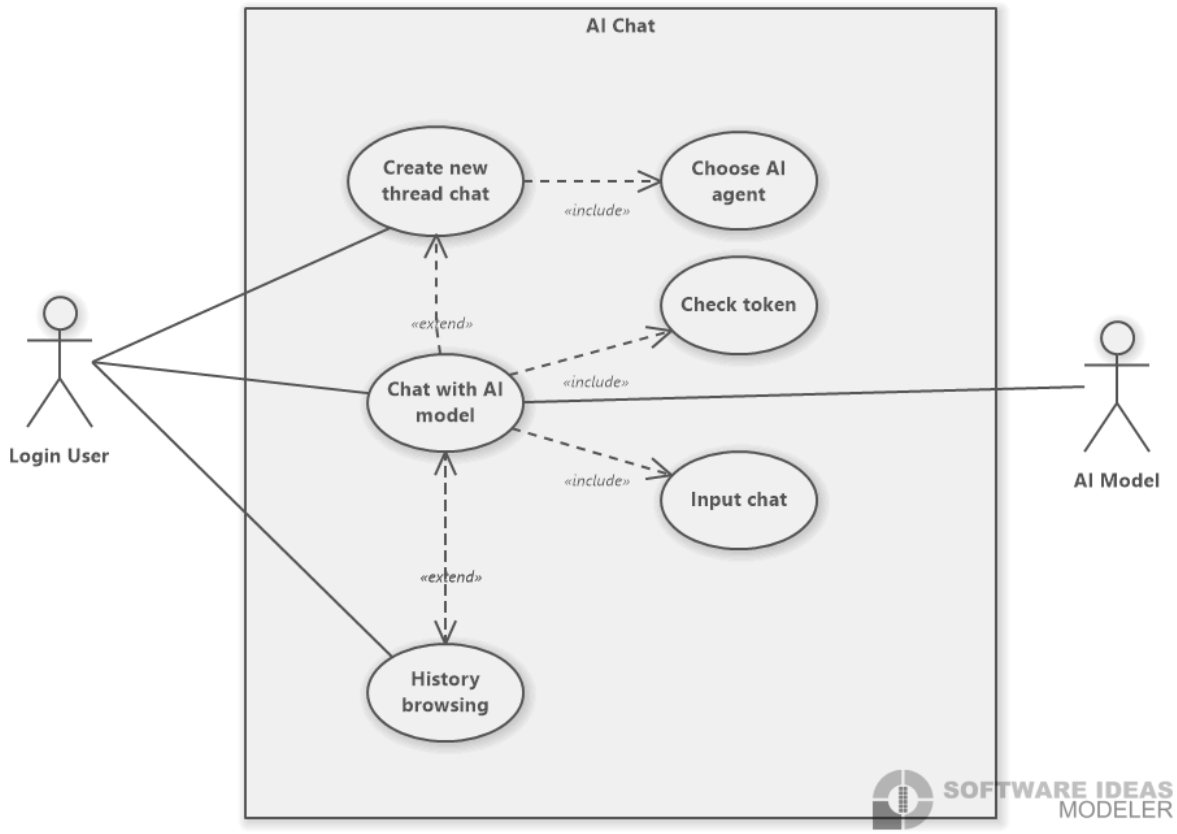
- **Programming Language**: The system must be developed in HTML, CSS, Javascript with frameworks such as Express, Bootstrap,..
- **AI Model Integration**: The chatbot must integrate with APIs like OpenAI or similar services.
- **Development Schedule**: The project must meet a tight timeline, with deliverables phased over a six-month period.

3. Use Case Model

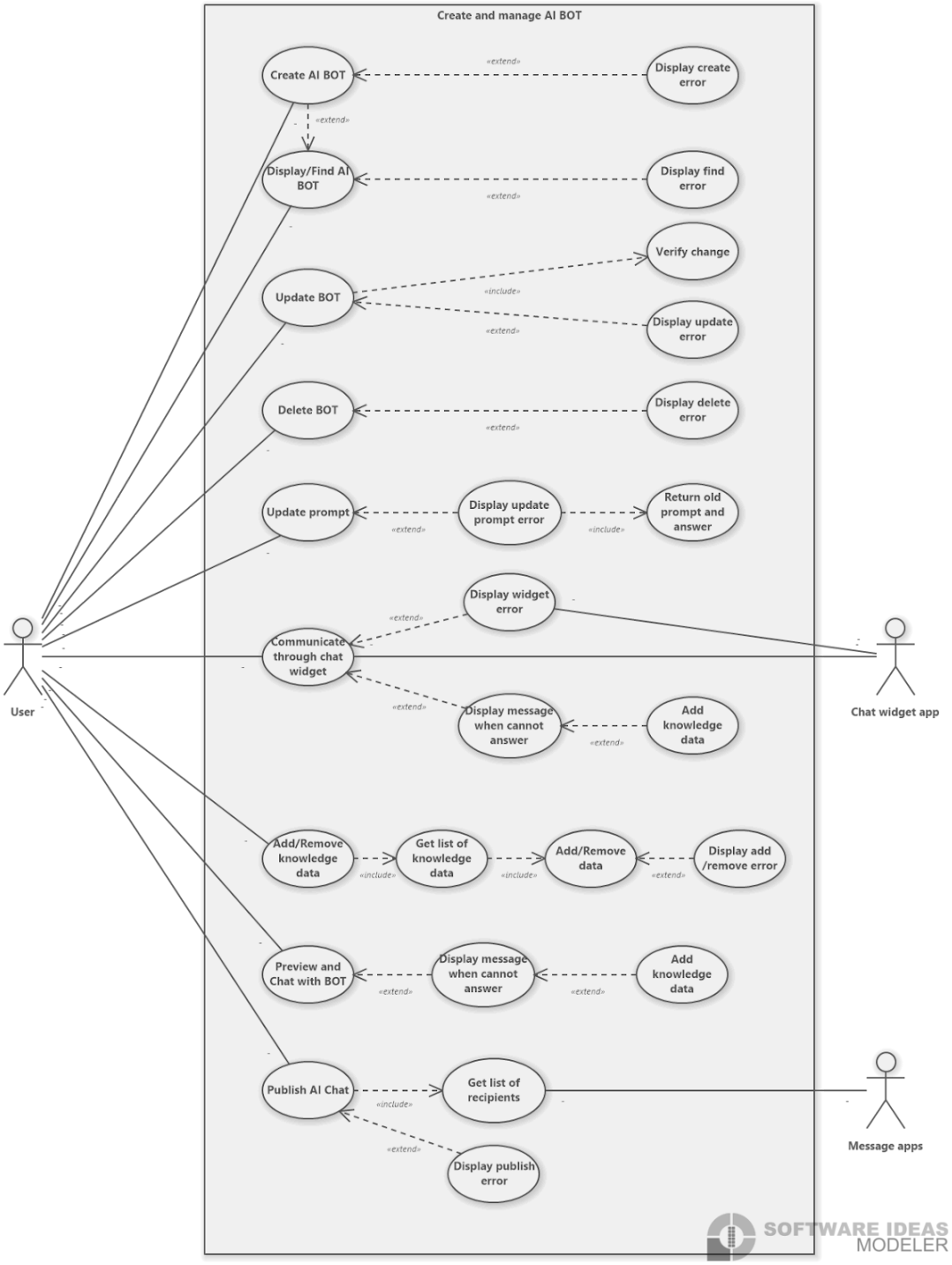
3.1 Authentication and Authorization:



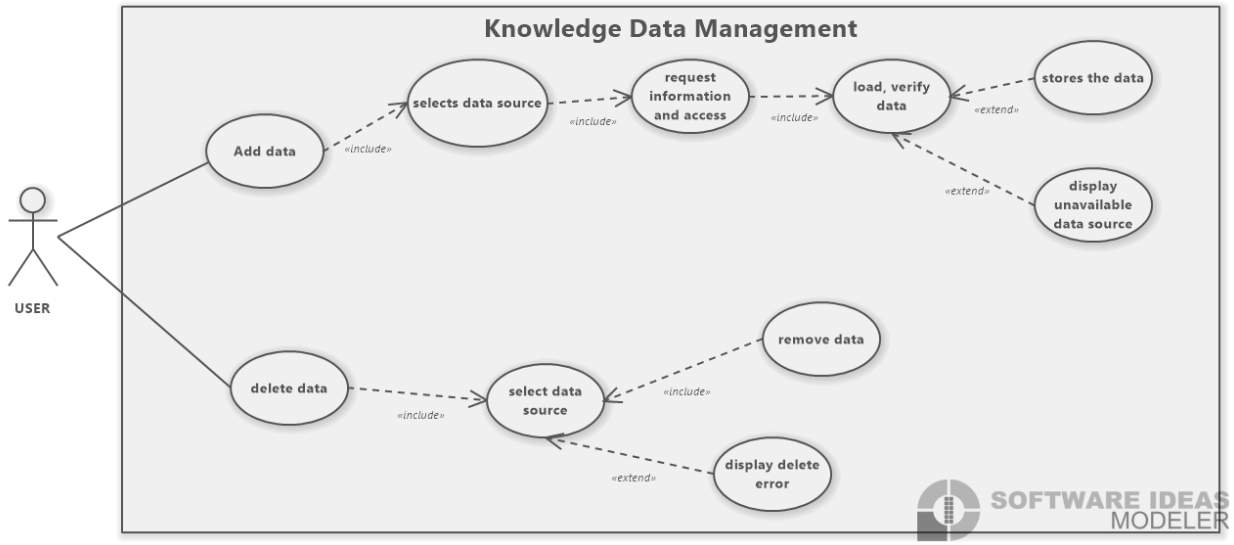
3.2 Chat with AI Agents:



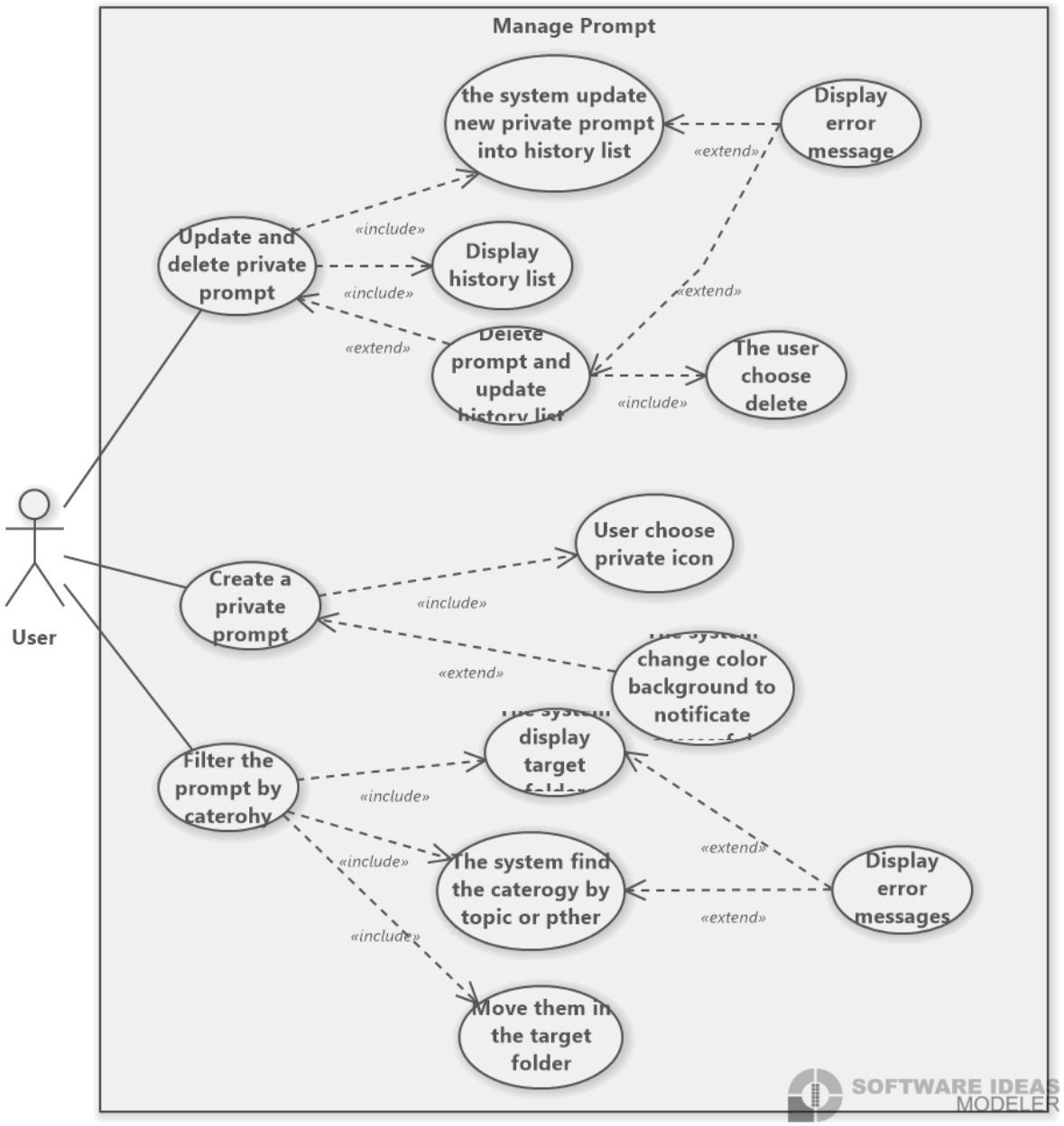
3.3 Create and manage AI BOT:



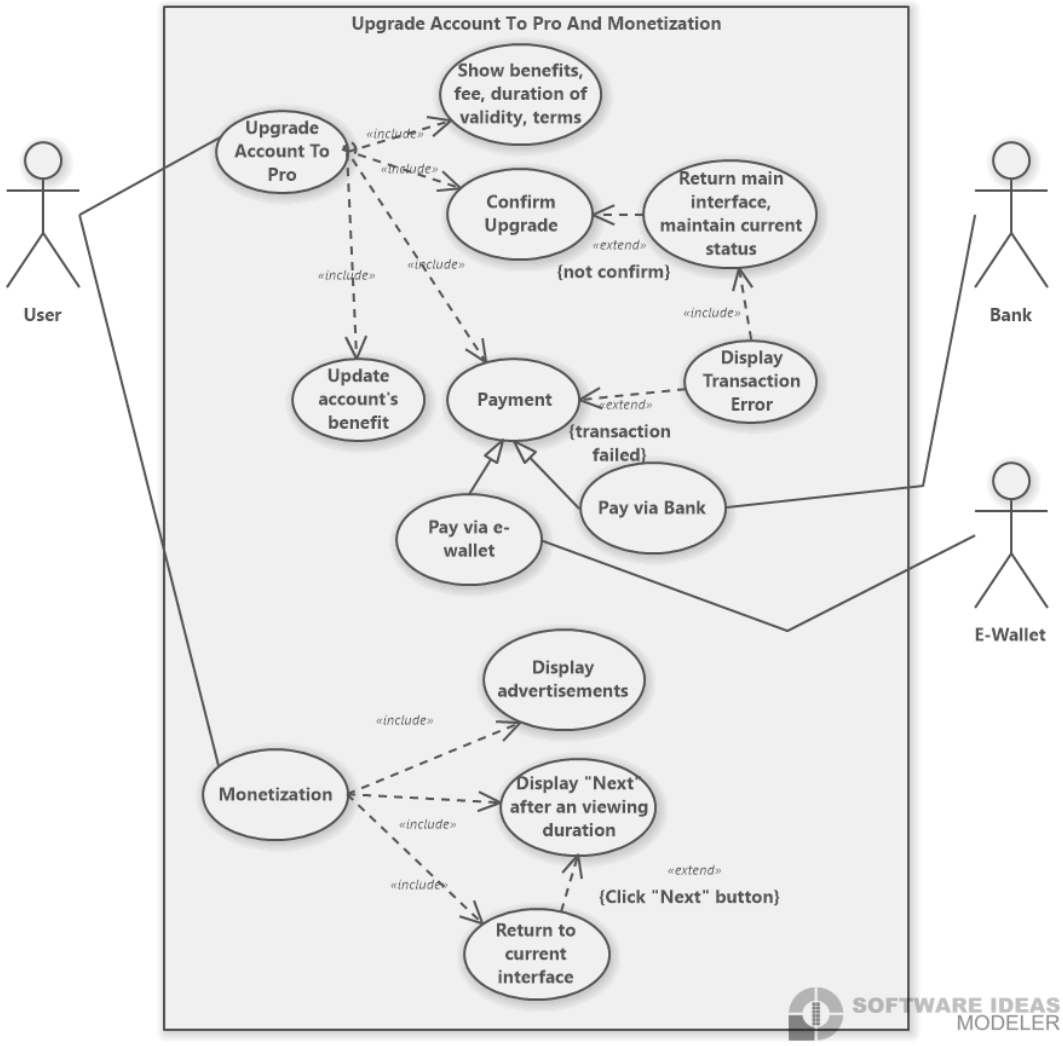
3.4 Knowledge Data Management:



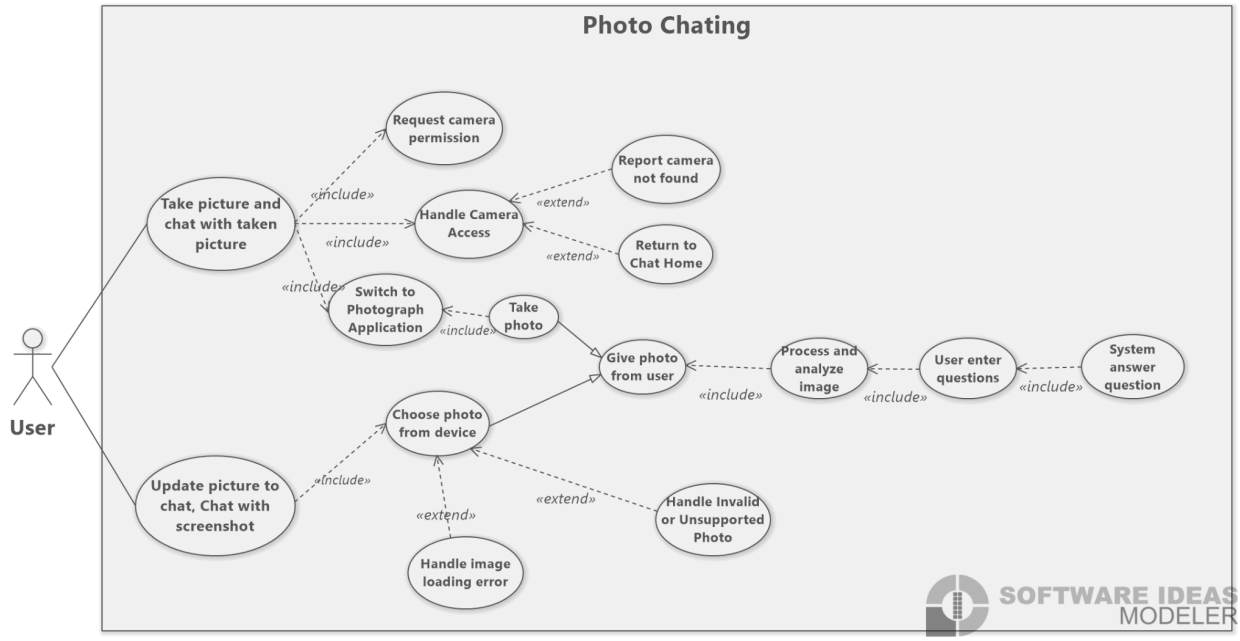
3.5 Prompt Management:



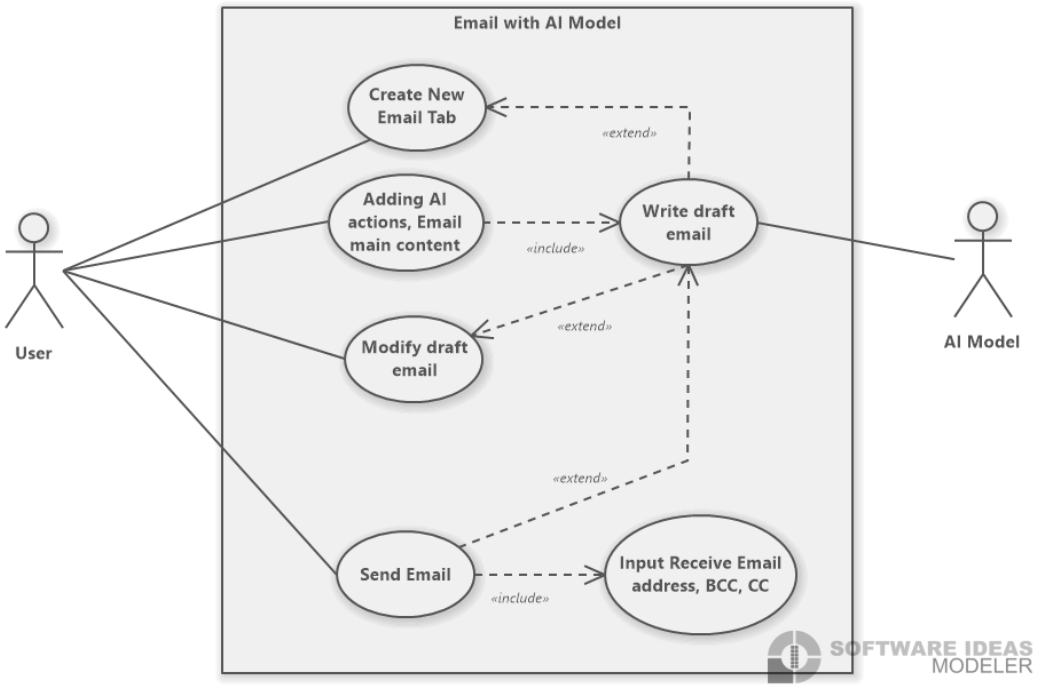
3.6 Account Management:



3.7 Photo Chating:



3.8 Email with AI Agents:



4. Logical View

4.1 Authentication and Authorization:

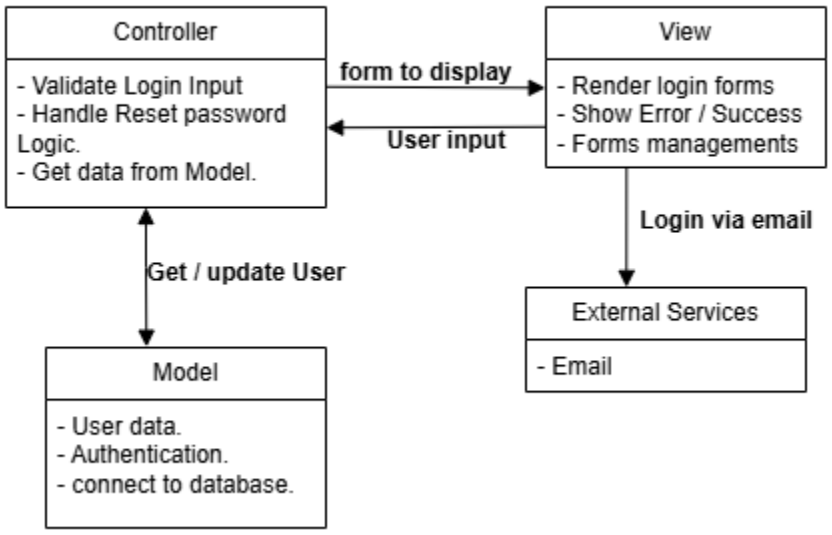
Responsibilities:

- Manage user registration, login, and logout.
- Validate user credentials.
- Handle Google Login and email-based account activation.
- Provide secure access tokens for authenticated users.

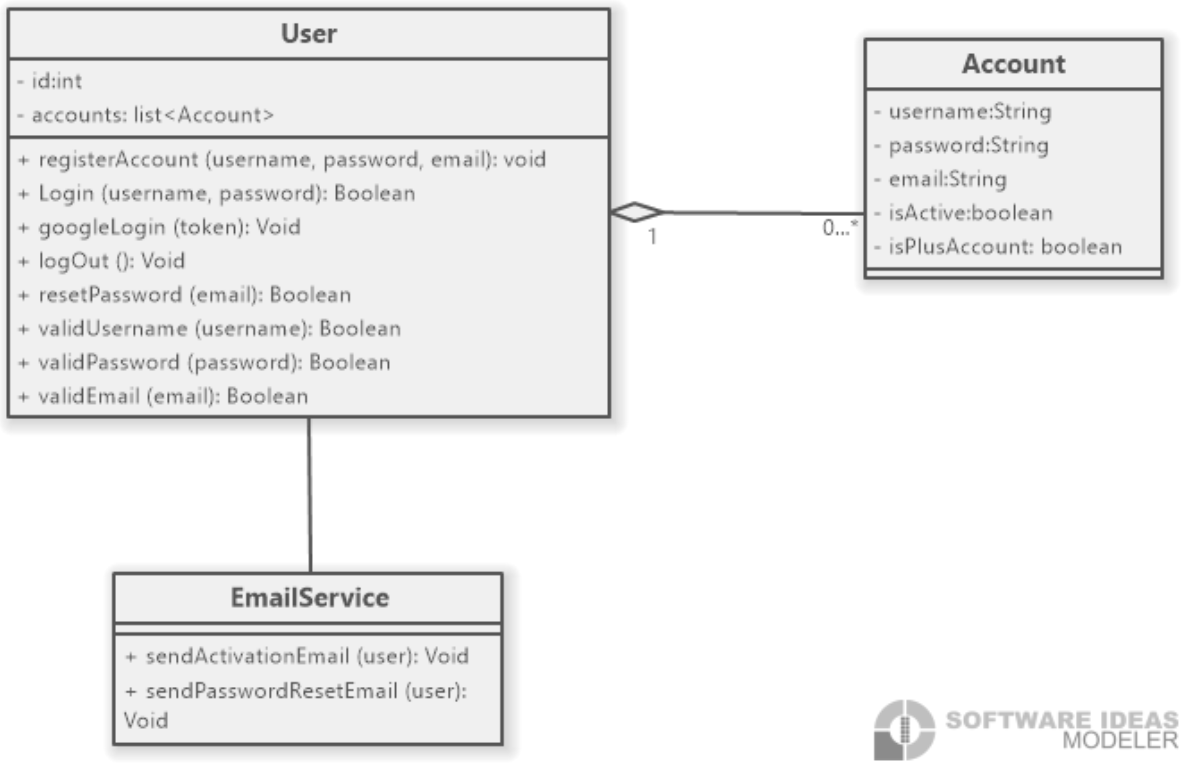
Connections:

- Interacts with AuthenticationManager to verify credentials.
- Uses EmailService to send activation or reset password emails.
- Updates the User database table.

a) MVC diagram:



b) Class diagram:

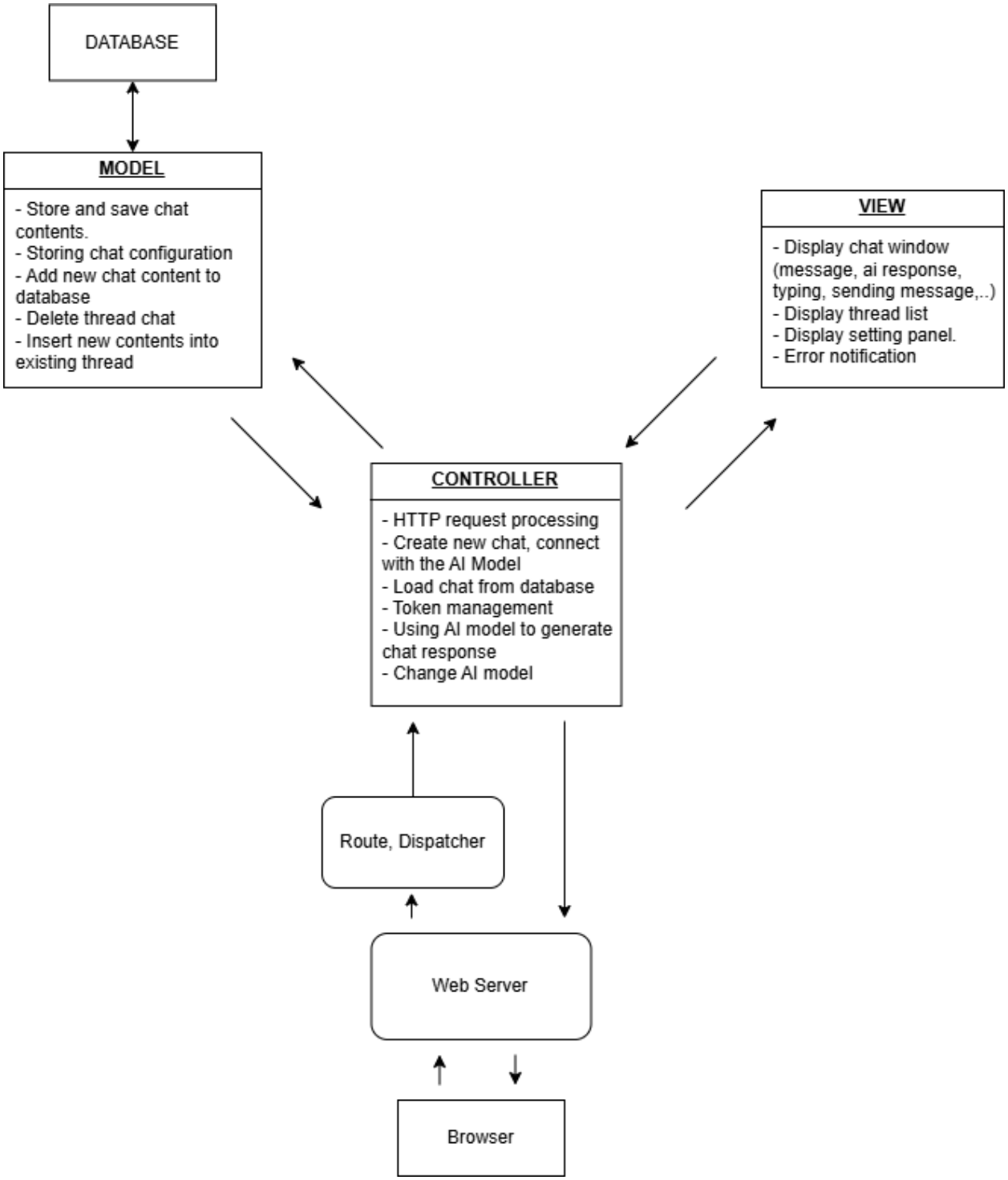


4.2 Chat with AI model

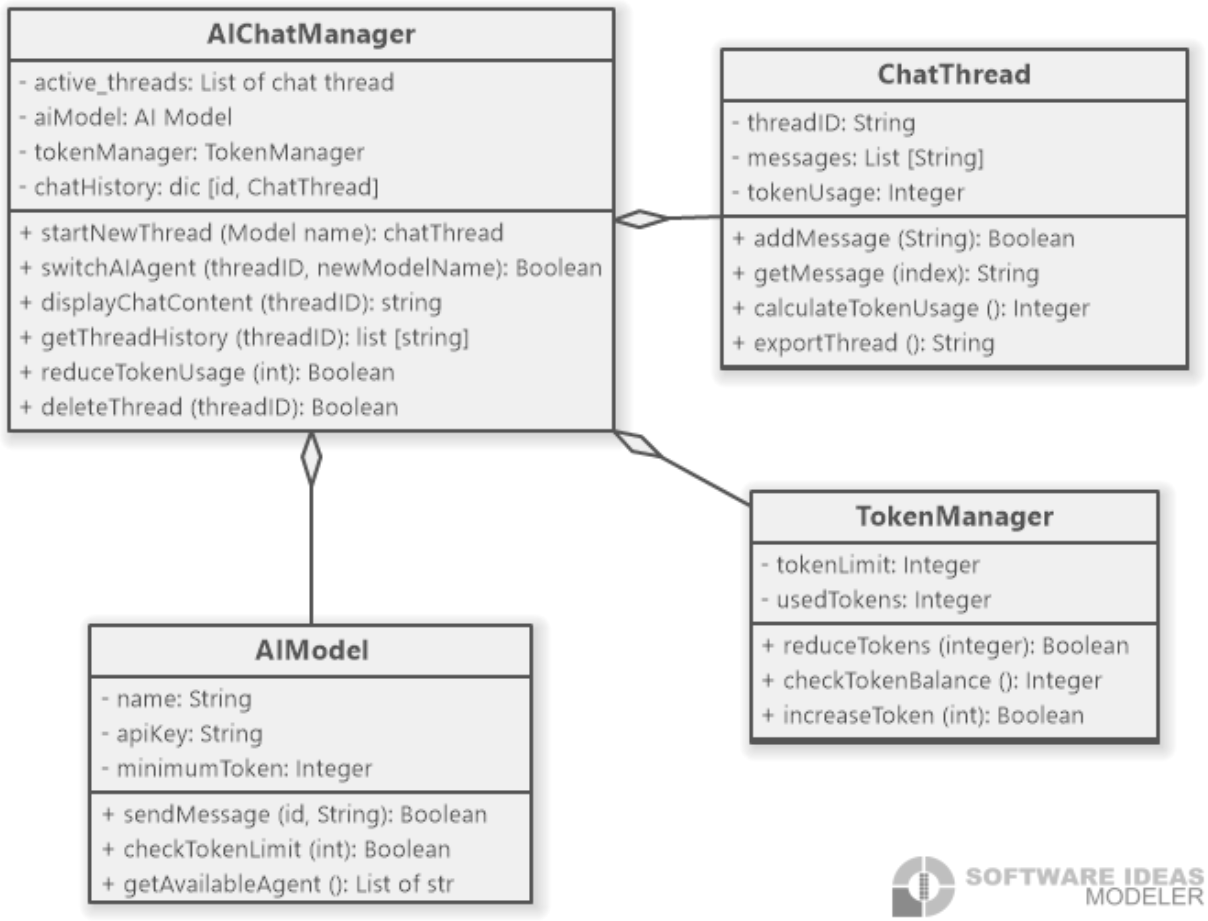
Responsibility:

The system ensures efficient communication with AI by managing chat threads, optimizing token usage, and offering advanced functionalities such as conversation history and the ability to switch between different AI models.

a) MVC Diagram:


DATABASE
MODEL
- Store and save chat contents.
- Storing chat configuration
- Add new chat content to database
- Delete thread chat
- Insert new contents into existing thread
VIEW
- Display chat window (message, ai response, typing, sending message,..)
- Display thread list
- Display setting panel.
- Error notification
CONTROLLER
- HTTP request processing
- Create new chat, connect with the AI Model
- Load chat from database
- Token management
- Using AI model to generate chat response
- Change AI model
Route, Dispatcher
Web Server
Browser

b) Class Diagram:



c) Classes and features:

AIChatManager Features:

- Start New Thread: Creates a new chat thread for interaction with the AI model.
- Switch AI Agent: Allows switching between different AI agents.
- Display Chat Content: Retrieves and displays the full chat content of a specific thread.
- Get Thread History: Provides a history of messages from a specific chat thread.
- Reduce Token Usage: Optimizes or limits token usage for cost and performance efficiency.
- Delete Thread: Deletes an existing chat thread and its associated messages.

ChatThread Features:

- Add Message: Appends a message to the thread, either from the user or the AI.
- Get Message: Retrieves a specific message from the thread based on its index.

- Calculate Token Usage: Computes the total token usage for the thread, useful for managing API costs.
- Export Thread: Allows exporting the entire chat thread in a specific format (e.g., plain text, JSON).

AIModel Features:

- Send Message: Handles sending a user message to the AI and receiving a response.
- Check Token Limit: Validates if a message can be sent within the token limits.
- Get Available Agent: Lists all available AI agents that can be used in the chat.

TokenManager Features:

- Reduce Tokens: Deducts a specific number of tokens from the token balance.
- Check Token Balance: Checks the remaining token balance for usage monitoring.
- Increase Token: Adds more tokens to the balance when needed.

4.3 AI BOT Management:

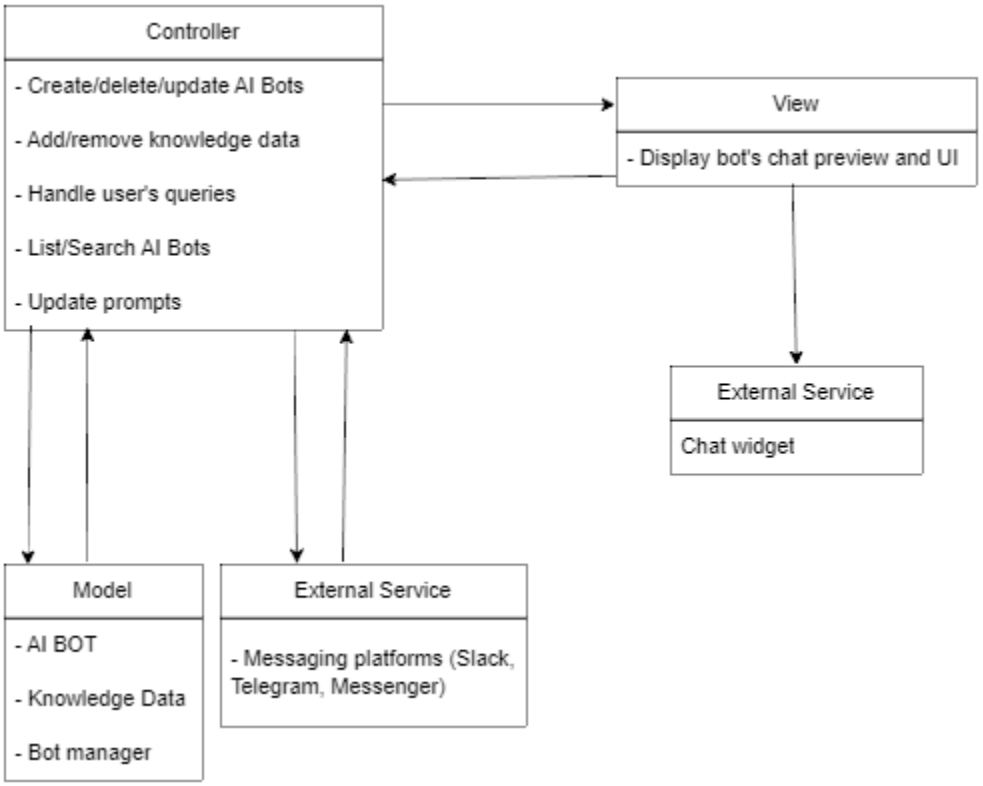
Responsibilities:

- Manage AI BOTs.
- Manage Knowledge Data.
- Handle user's queries.
- Publish bot's chat.

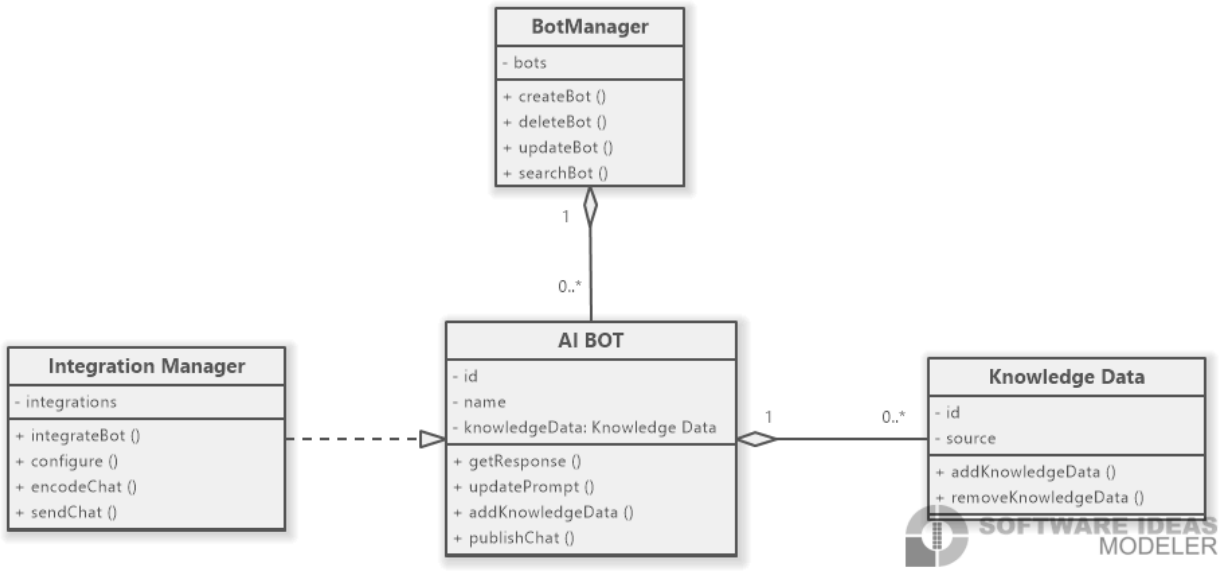
Connections:

- View and controller's interactions:
  - View sends user's input to Controller.
  - Controller responses accordingly based on user's input.
- Controller and Model's interactions:
  - Controller receives requests from user and updates the Model accordingly.
  - The Model can notify the Controller of changes to AI Bot data or knowledge data.
- View and External Service's interactions:
  - View sends UI and chat preview to chat widget application.
- Controller and External Service's interactions:
  - Messaging platforms send recipient lists to Controller.
  - Controller encodes and sends bot's chat to messaging platforms.

a) MVC diagram:



b) Class diagram:



4.4. knowledge dataset management

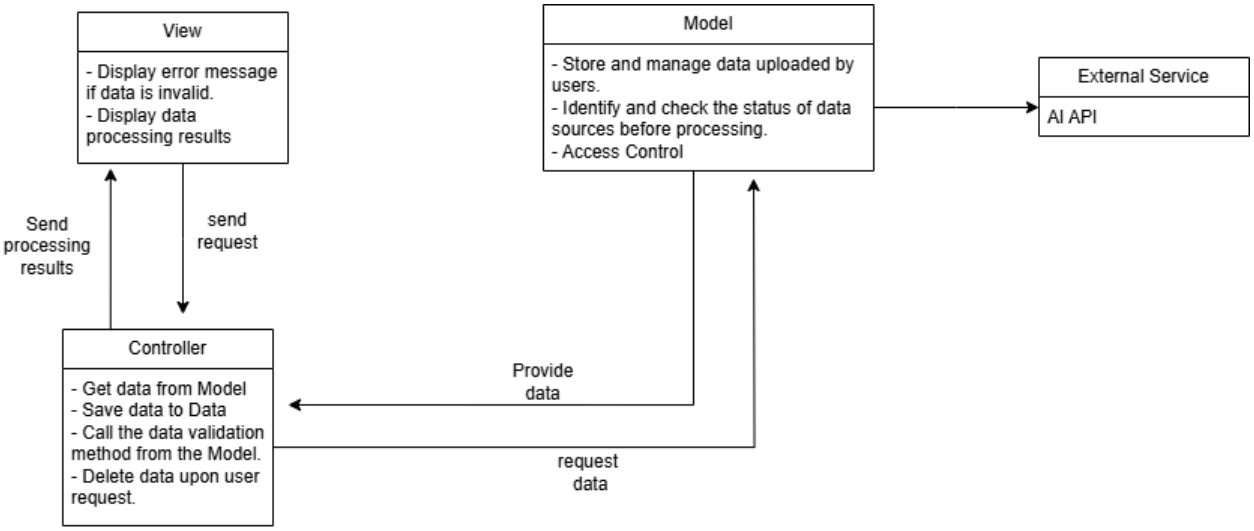
Responsibilities:

- Manage user actions
- Manage data source information and status
- data manipulation
- Represent data, provide actions

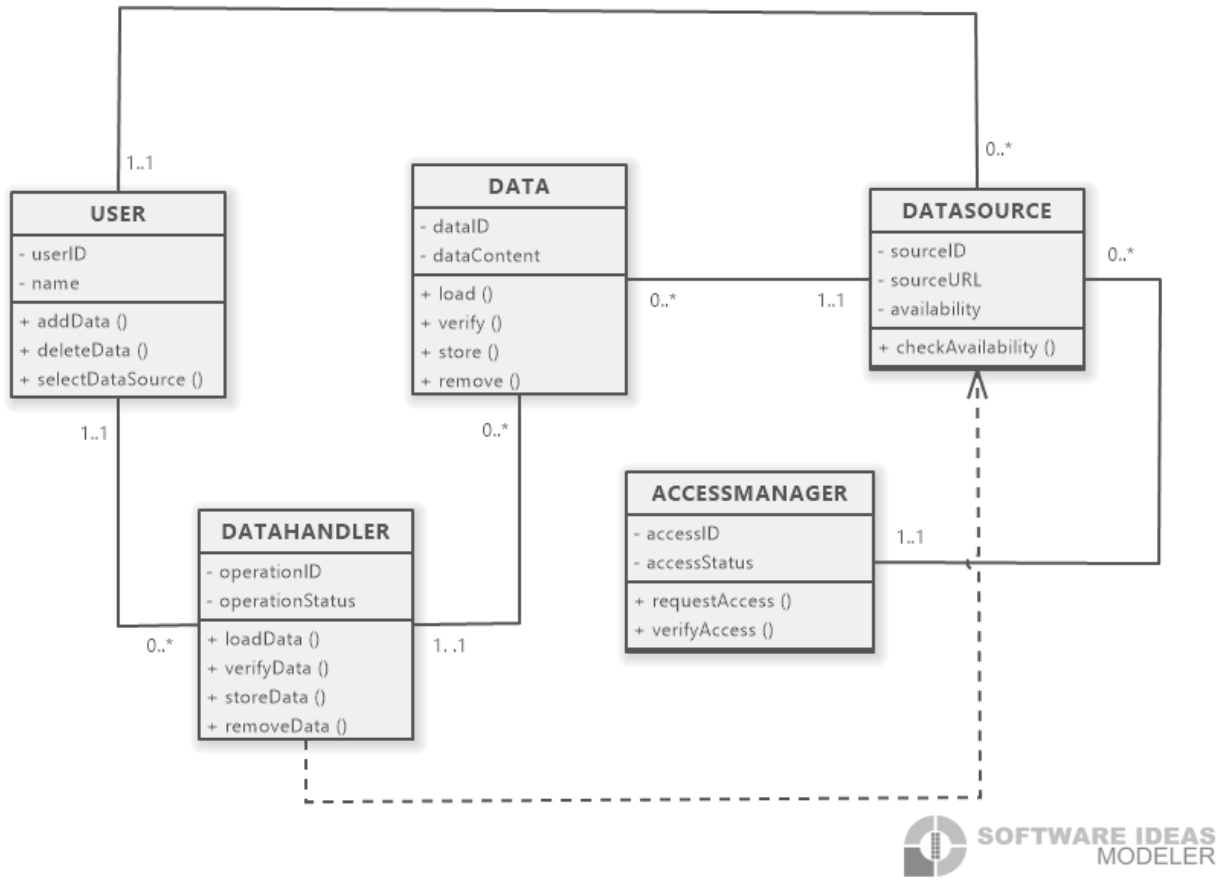
Connections:

- User and DataHandler’s interactions
  - User sends requests to DataHandler to manage data operations.
  - DataHandler processes these requests and interacts with Data as required.
- DataHandler and Data’s interactions
  - DataHandler interacts with Data to perform specific actions.
  - DataHandler manages multiple Data objects during batch operations.
- User and DataSource’s interactions
  - User selects a DataSource to perform data operations.
  - User depends on DataSource availability to proceed.
- DataHandler and DataSource’s interactions
  - DataHandler retrieves data from the selected DataSource.
  - DataHandler checks the DataSource’s availability before performing operations.
- AccessManager and DataSource’s interactions
  - AccessManager verifies access to DataSource.
  - AccessManager grants or denies access based on permissions.
- DataSource and Data’s interactions
  - DataSource stores and manages multiple Data objects.
  - Data depends on the DataSource for its lifecycle.

a) MVC diagram:

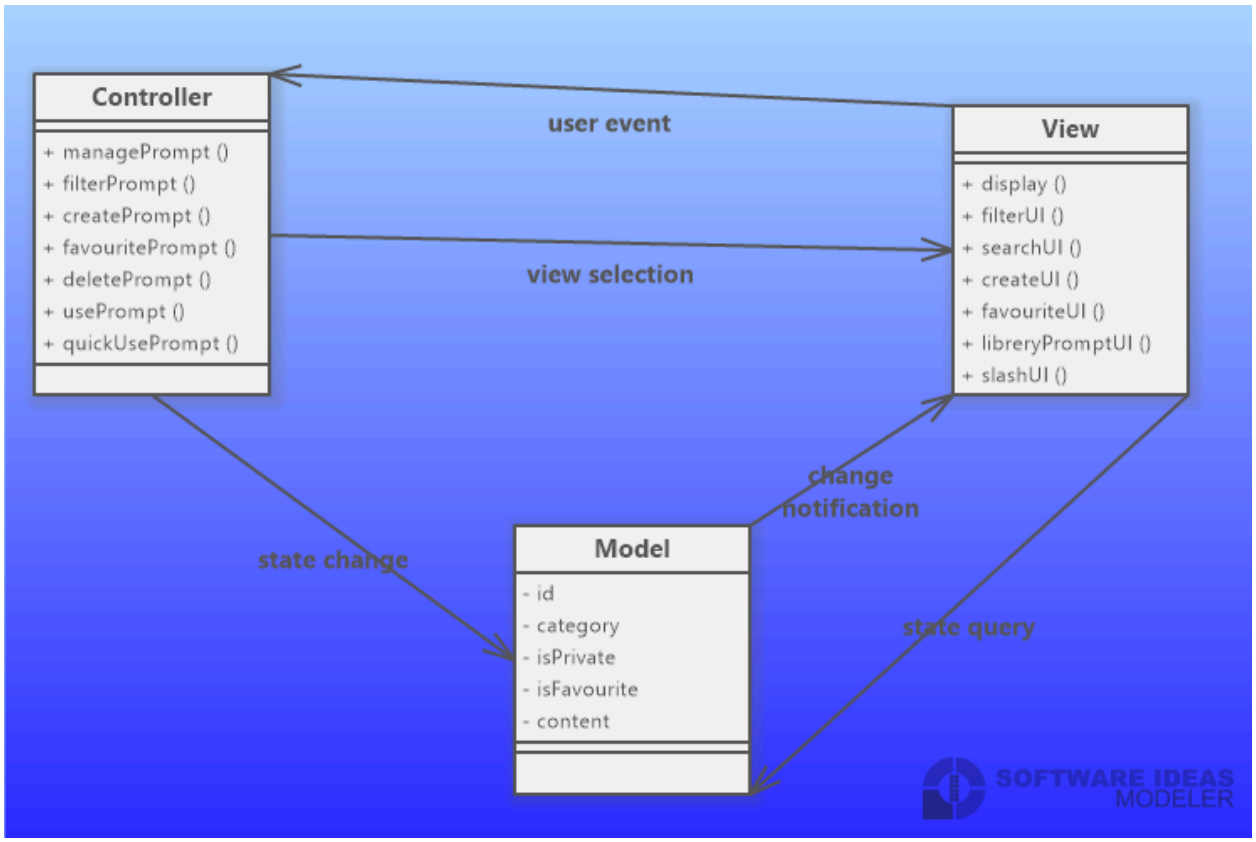


b) Class diagram:

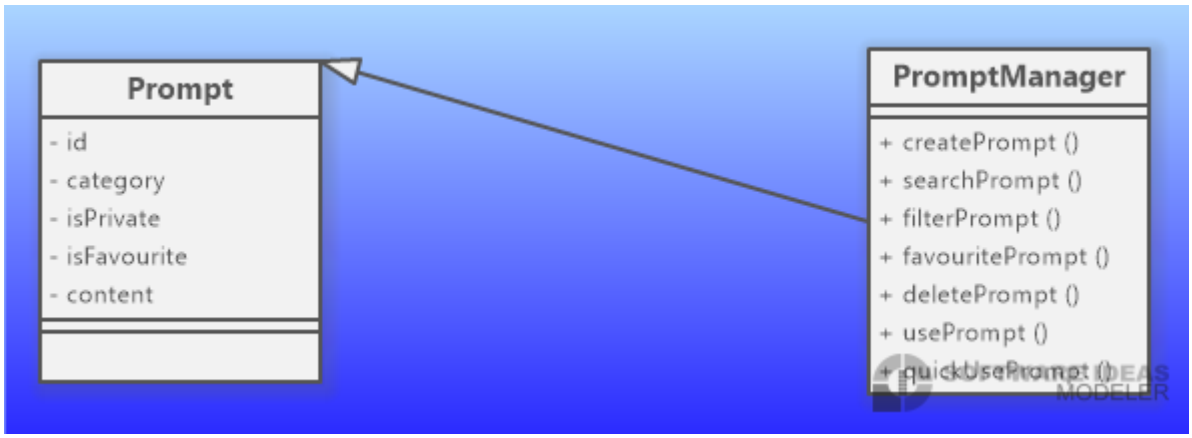


4.5 Prompt management

a) MVC diagram:



b) Class diagram:



4.6 Upgrade Account to Pro and Monetization:

4.6.1 Upgrade Account to Pro:

Allows users to enhance their account status to a Pro level, unlocking advanced features and benefits.

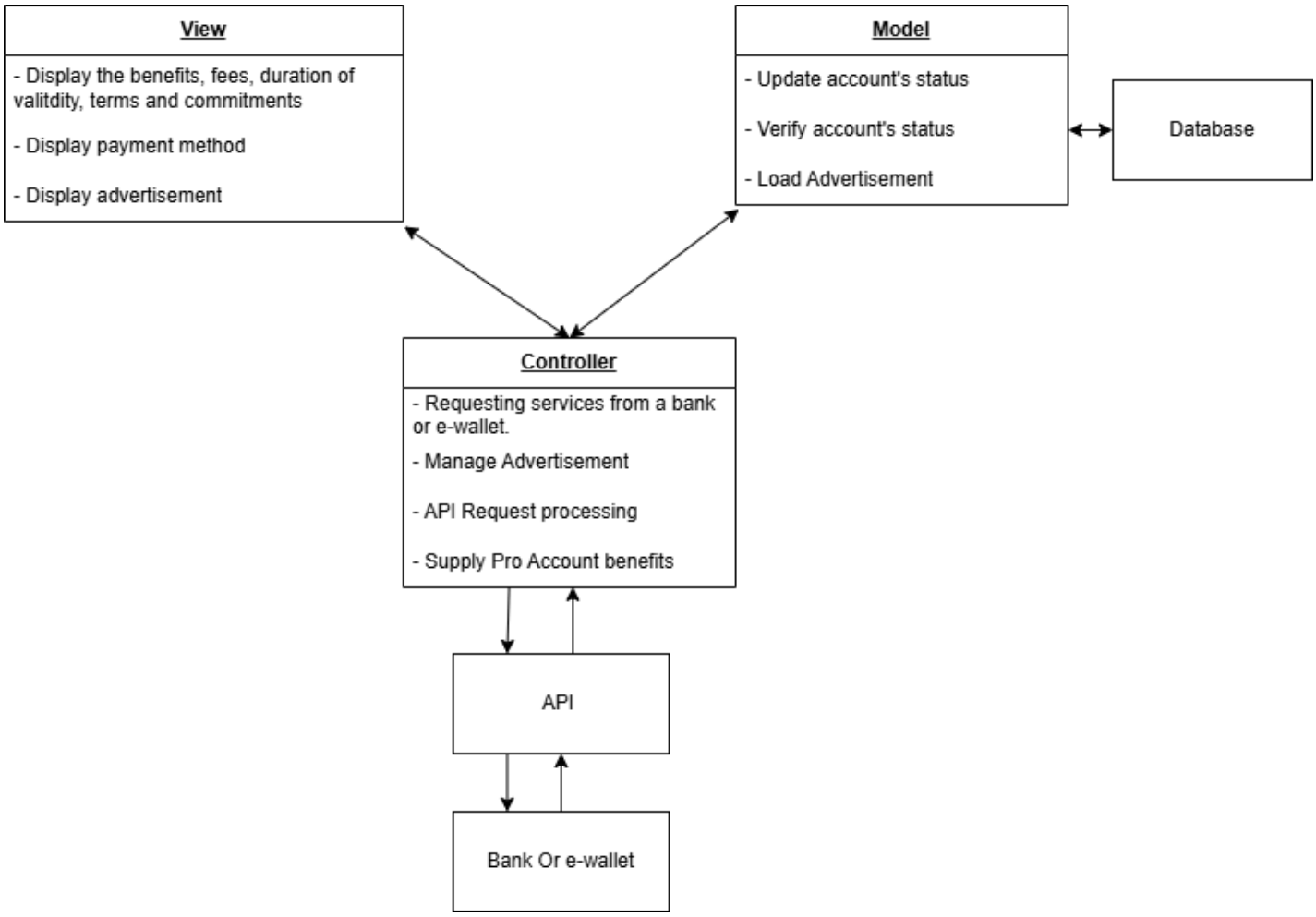
- **Responsibilities:**
  - Processes payment for Pro upgrades.
  - Updates the user's account tier in the system.
  - Enables Pro-exclusive services or features such as increased storage, advanced analytics, or priority support.
- **Services Provided to Other Components:**
  - Notifies the billing component to manage recurring payments if applicable.
  - Updates the user profile component with new permissions or features.
  - Integrates with notification systems to inform the user about the successful upgrade and associated benefits.

4.6.2 Monetization:

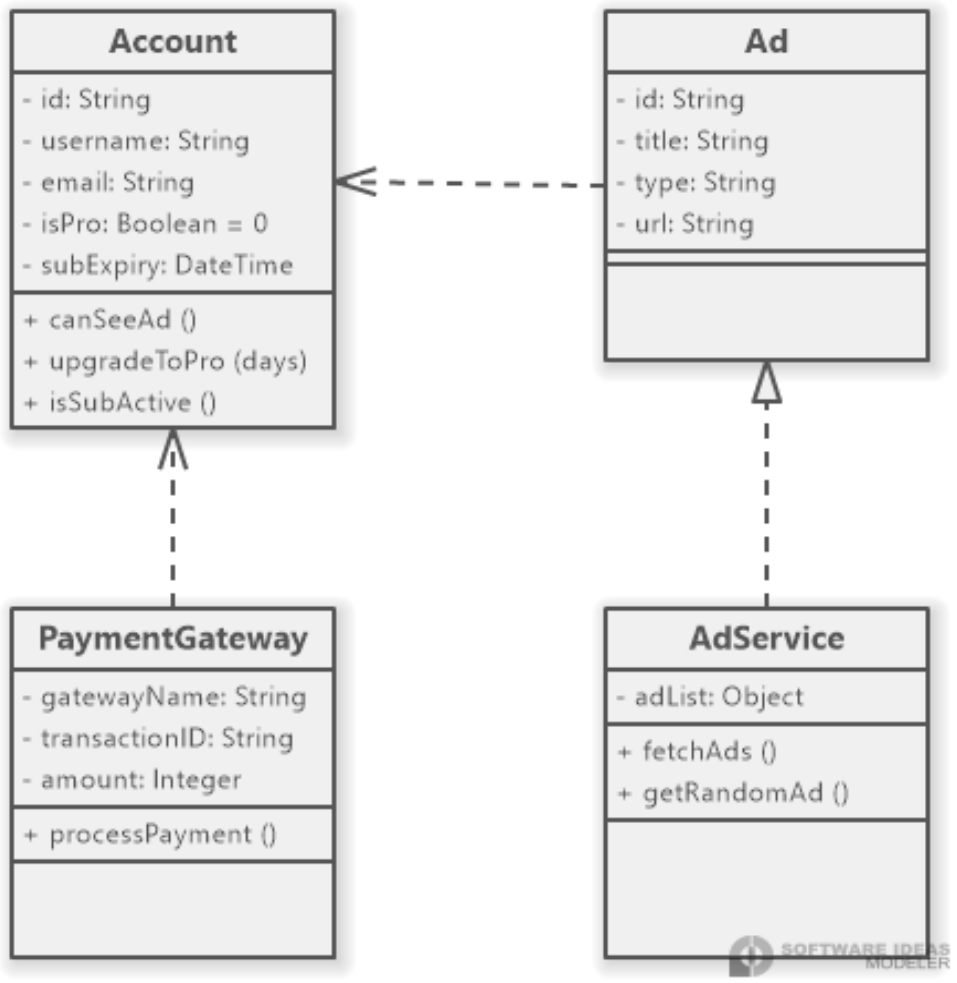
Enables users to generate revenue from their content or services through integrated monetization tools.

- **Responsibilities:**
  - Manages revenue-sharing agreements.
  - Provides analytics for earnings and activity.

a) MVC diagram:



b) Class diagram:



4.7 AI Image Q&A:

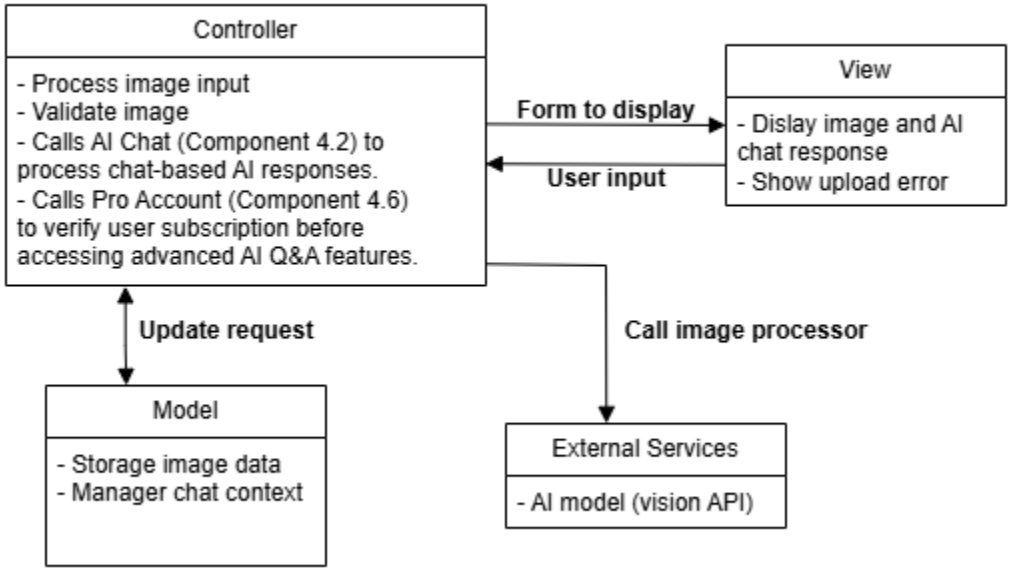
Responsibilities:

- Process uploaded or captured images for chat interactions.
- Extract image data to facilitate AI responses.

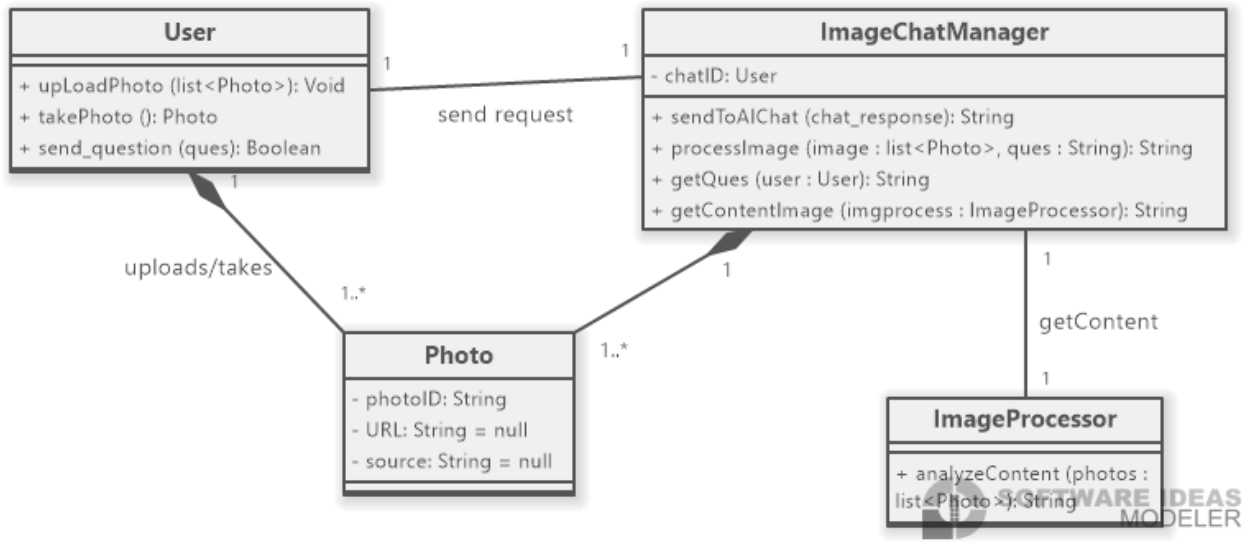
Connections to other components:

- **AI Chat (Component 4.2)**:
- The **AI Image Q&A Controller** sends chat context and processed image data to the **AI Chat component** for generating AI responses.
- Example: After analyzing an image, the response is passed to the chat interface for seamless user interaction.
- **Pro Account and Monetization (Component 4.6)**:
- Before allowing image-based Q&A, the **AI Image Q&A Controller** checks the user's account subscription level.
- Example: Free users might be restricted to a limited number of image Q&A sessions, while Pro users have access to unlimited sessions.
- **Knowledge Dataset Management (Component 4.4)**:
- The processed image data can be sent to the **Knowledge Dataset Management** for adding new knowledge or improving existing datasets.
- Example: When users upload educational images (e.g., diagrams), these can be stored and used for training AI bots.

a) MVC diagram:



b) Class diagram:

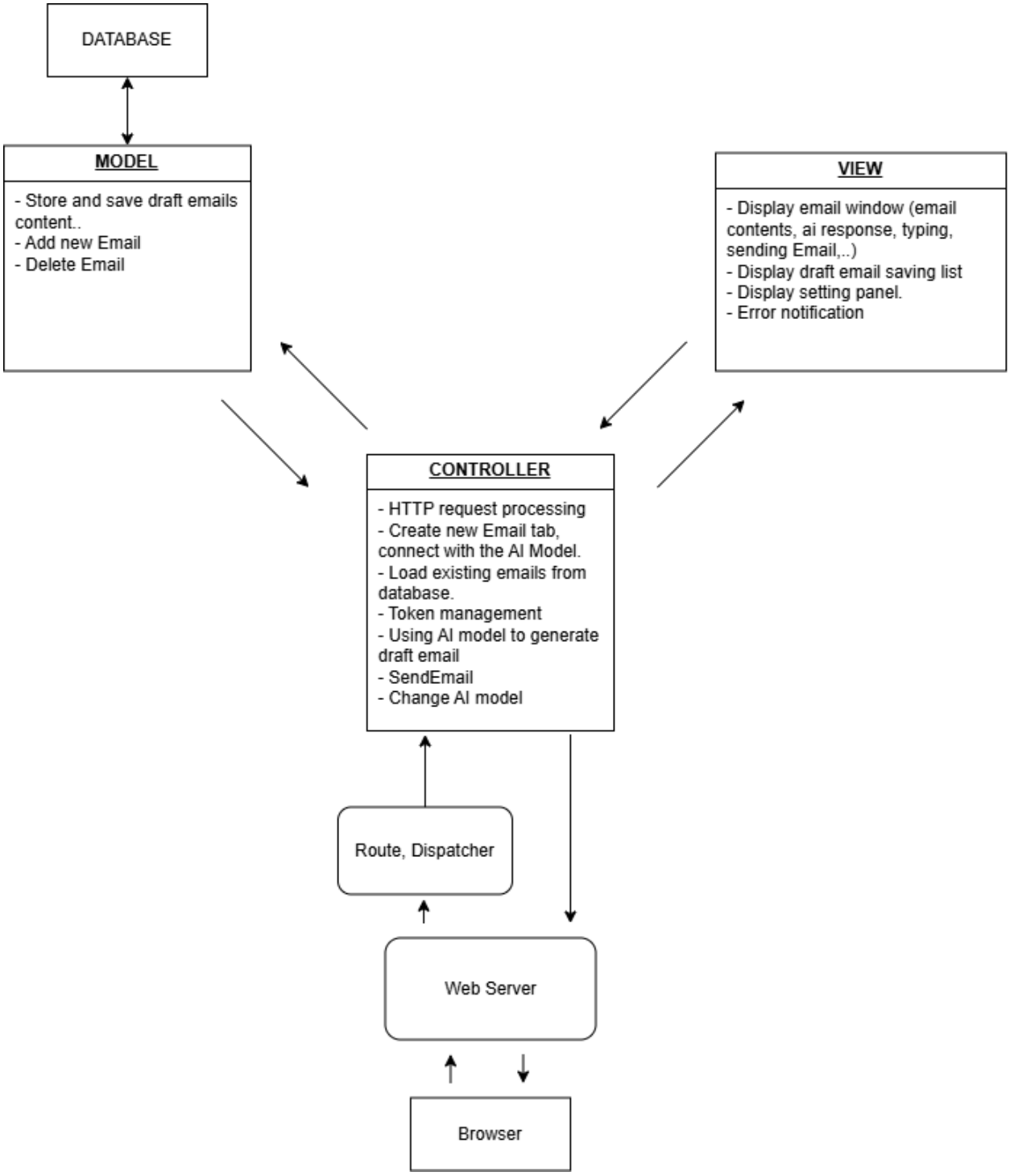


4.8 email with ai agent

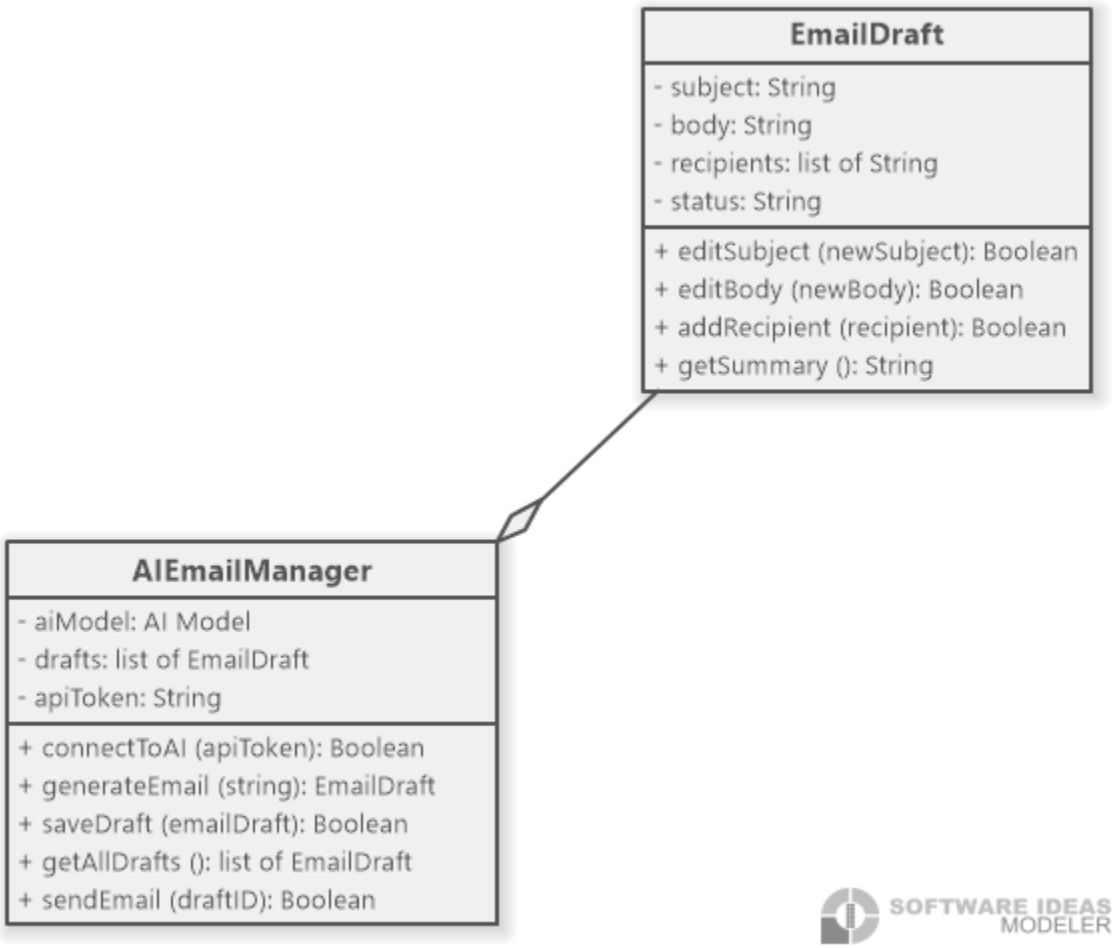
Responsibility:

Create email tab, utilizing the AI model to generate email drafts based on user-selected actions.

a) MVC diagram:


DATABASE
MODEL
- Store and save draft emails content..
- Add new Email
- Delete Email
VIEW
- Display email window (email contents, ai response, typing, sending Email,..)
- Display draft email saving list
- Display setting panel.
- Error notification
CONTROLLER
- HTTP request processing
- Create new Email tab, connect with the AI Model.
- Load existing emails from database.
- Token management
- Using AI model to generate draft email
- SendEmail
- Change AI model
Route, Dispatcher
Web Server
Browser

b) Class diagram:



c) Class and Features:

AIEmailManager Features:

- Connect to AI: Establishes a connection with an AI model using an API token for authentication.
- Generate Email: Creates a new email draft
- Save Draft: Saves an email draft to the internal draft list for future use or editing.
- Get All Drafts: Retrieves all stored email drafts as a list.
- Send Email: Sends an email based on a selected draft

EmailDraft Features:

- Edit Subject: Allows editing of the email subject line.
- Edit Body: Enables modifications to the email body content.

- Add Recipient: Adds a recipient email address to the list of recipients.
- Get Summary: Provides a brief summary of the email, including its subject, recipient count, and a preview of the body text.

5. Deployment

6. Implementation View