Product Design

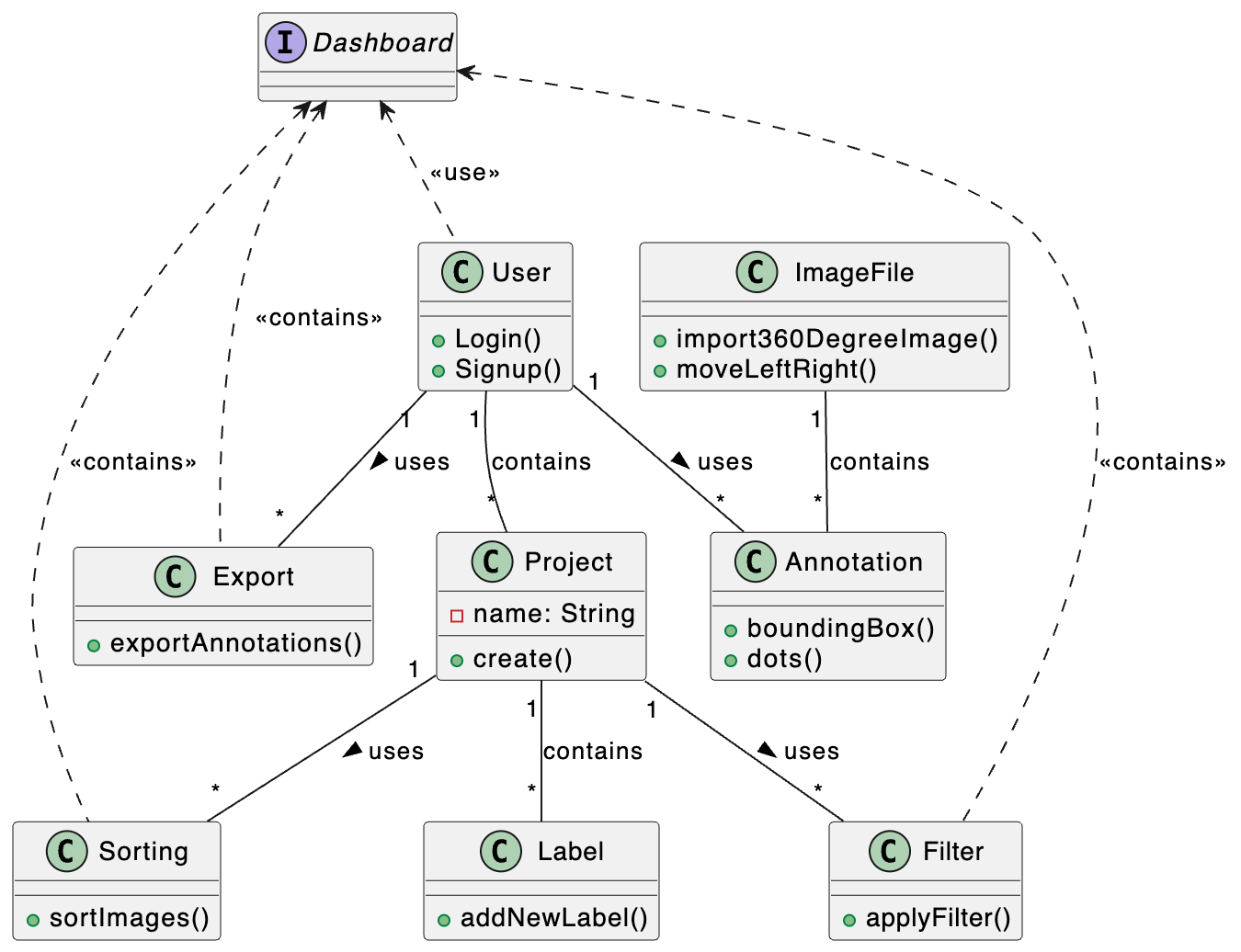
|  |  |
| --- | --- |
| **Team** | **Team-3**  Karan Nijhawan  Sanchit Jalan  Sreeja Palle  Manda Vaishnavi Reddy |

# Design Model

|  |  |
| --- | --- |
| User | Class state   * Maintains user information such as username, password, and user preferences.   Class behavior   * logging in * logging out * updating user information * managing user preferences. |
| Image | Class state   * *Information about the image file, such as file name, size, and format.* * *Metadata related to the image, such as creation date, uploader, etc.* * *Navigation controls state (e.g., current view angle, zoom level).*   Class behavior   * *Methods to import/upload image files.* * *Functionality to navigate the image using controls provided.* * *Functionality to save annotation data.* * *Export methods to save the annotated image with labels.* |
| Annotation | Class state   * *Metadata related to annotations (e.g., creation date, author).* * *Position and size information for the annotated regions.*   Class behavior   * *Methods to create, edit, and delete annotations.* * *Methods to retrieve annotation data associated with an image.* |
| Label | Class state   * *Labels or tags assigned to a project for categorization or classification purposes.* * *Metadata related to labels (e.g., creation date, color-coding).*   Class behavior   * *Methods to assign, remove, and modify labels.* * *Add Color for each label.* * *Methods to retrieve images or annotations based on specific labels.* |
| Project | Class state   * Project details such as name, description, and associated images.   Class behavior   * *Functionality to display uploaded images and navigate between them.* * *Interface methods to interact with user inputs and controls.* * *Sort and Filter options* |
| Dashboard | Class state   * Contains Project list and other navigations   Class behavior   * *Create Project option* * *Acts as an interface for other functionalities* |

.

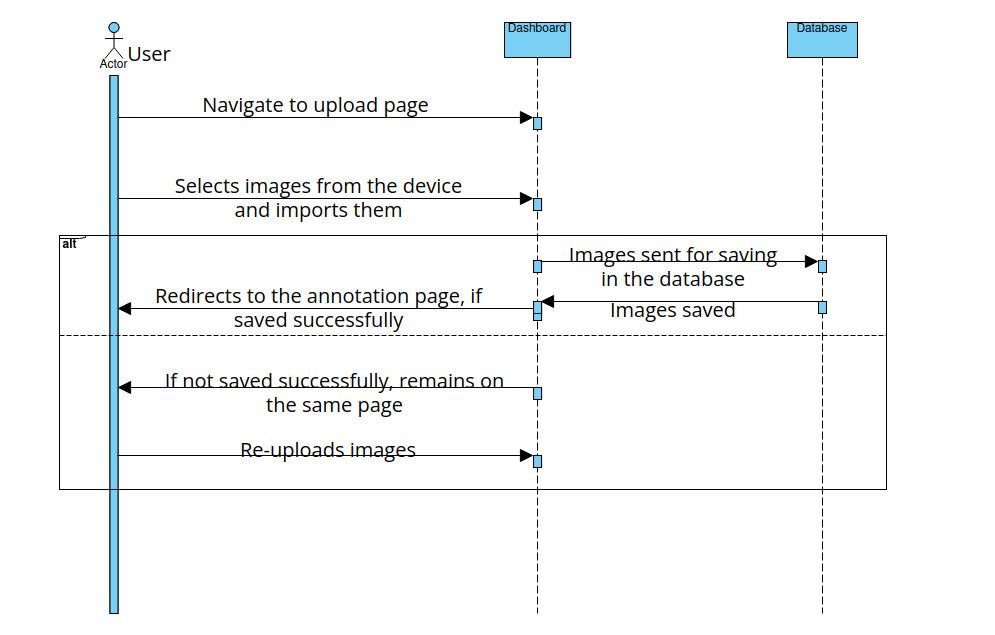
**UML Class Diagram**

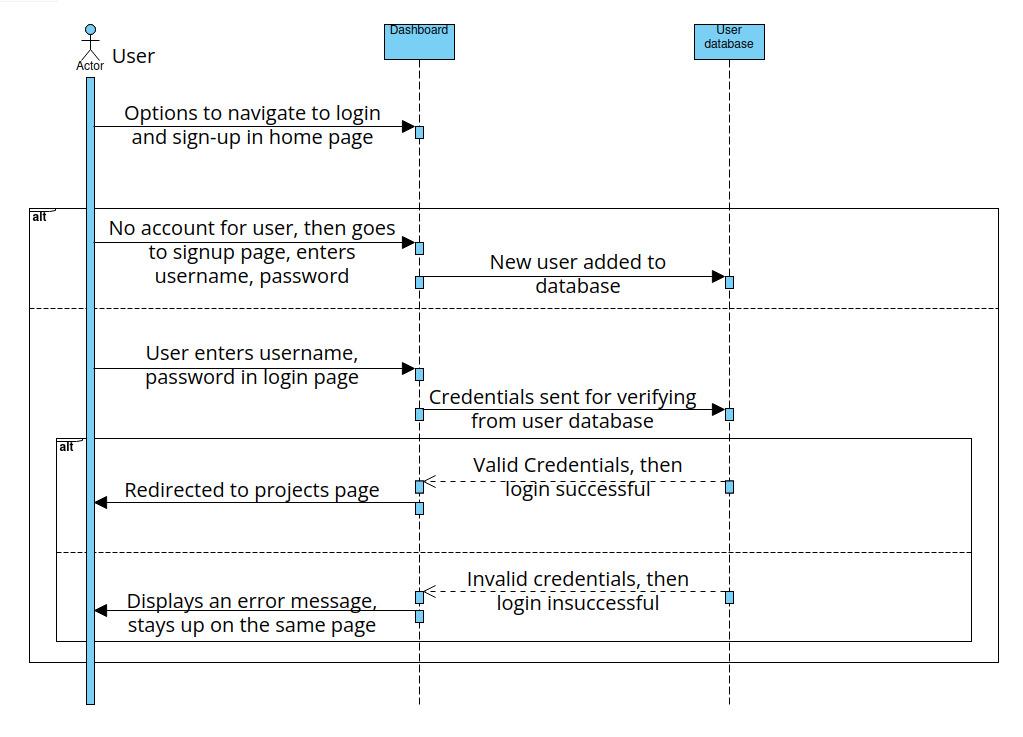


# Sequence Diagram(s)

# 1.

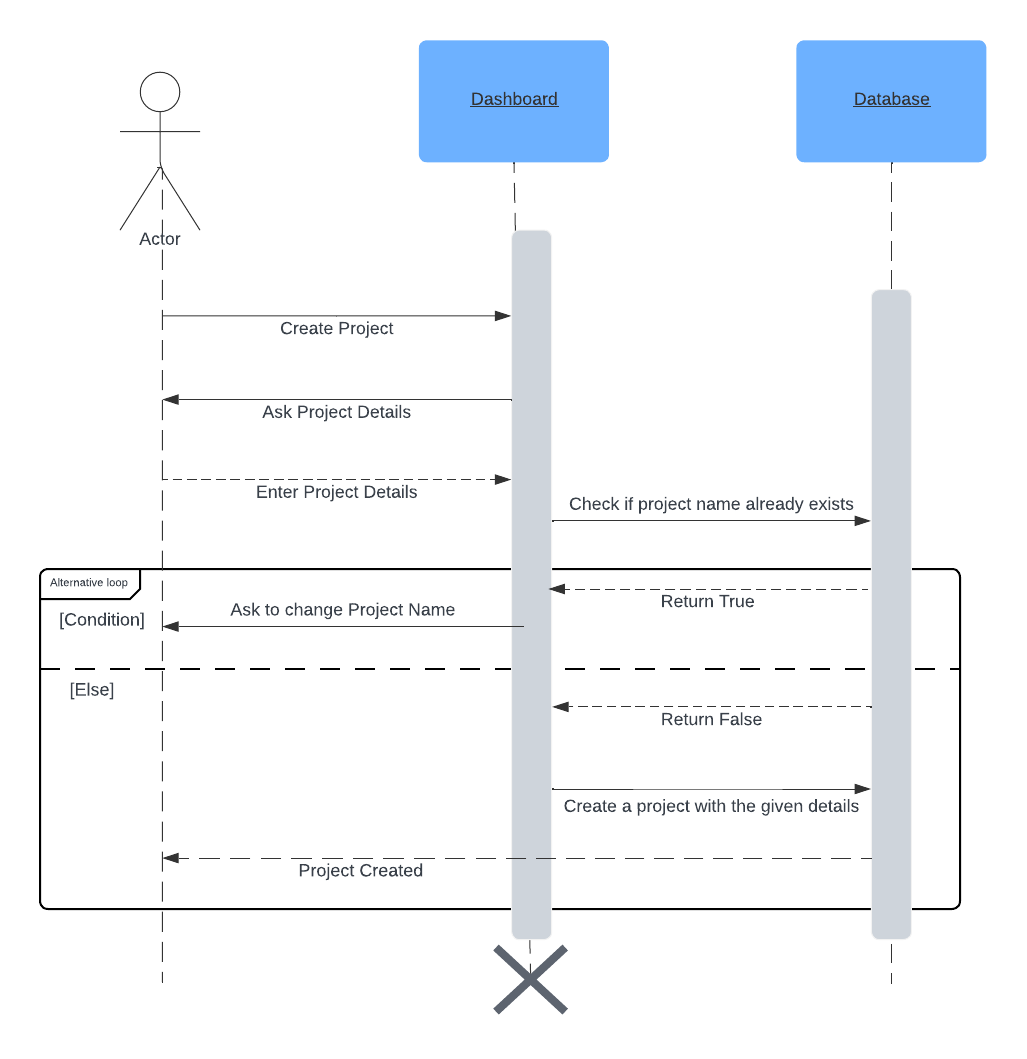
# 

2.

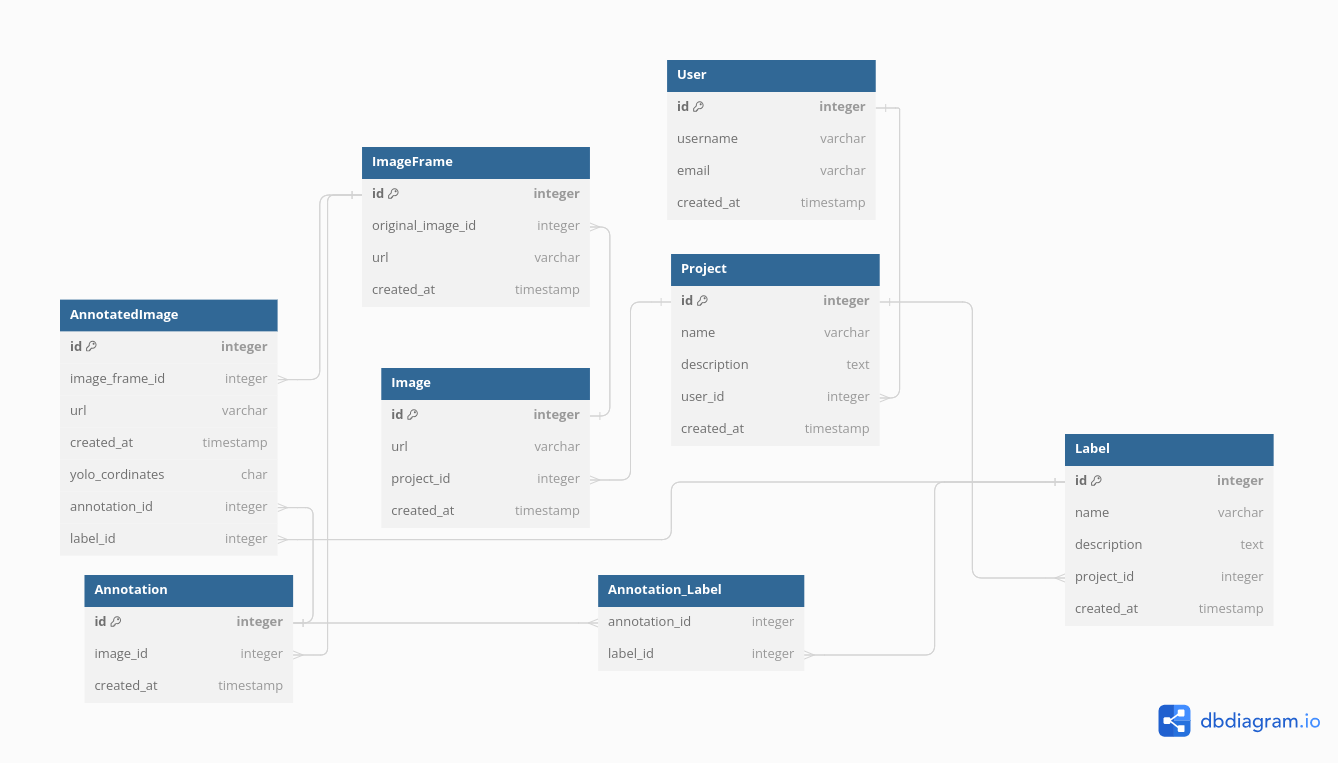


3.

4.



Database Design



**Design Rationale**

### **Redesign of the Database**

Alternatives Considered:

* Maintaining Existing Database Structure: Could involve patching existing conflicts and issues, but might not adequately address underlying structural deficiencies.

Strengths of Chosen Approach (Integration of Annotations and Labels):

* Improved Data Integrity: By integrating annotations and labels directly into the database model, the system ensures a more cohesive and structured approach to data categorization.
* Streamlined Operations: With annotations and labels as integral parts of the database schema, operations such as querying and updating related data become more efficient and accurate.

Weaknesses:

* Increased Complexity: Integrating annotations and labels into the database model may introduce additional complexity.

Rationale for Choice: The decision to integrate annotations and labels into the database model was driven by the need to resolve inherent conflicts and inefficiencies in the previous implementation. By doing so, the system aims to improve operational efficiency and enable more accurate and structured data categorization, ultimately enhancing the overall functionality and usability of the database.

### Redesign of the User Interface (UI)

Alternatives Considered:

* Minimal UI Changes: Could involve making incremental adjustments to the existing UI without a complete overhaul, potentially minimizing disruption but limiting the scope for significant improvements.
* User Testing and Iterative Design: Could involve gathering user feedback and iteratively refining the UI based on user preferences and pain points, ensuring a user-centric approach but potentially requiring more time and resources.

Strengths of Chosen Approach (Revamping UI):

* Alignment with Modern Design Principles: By revamping the UI, the system ensures alignment with modern design standards, enhancing aesthetics and usability.
* Improved User Experience: A more intuitive and user-friendly interface facilitates seamless navigation and interaction with the database, leading to higher user satisfaction and productivity.

Weaknesses:

* Potential Disruption: A complete UI overhaul may temporarily disrupt user workflows.
* Resource Intensive: Redesigning the UI can be time-consuming and resource-intensive.

Rationale for Choice: The decision to redesign the UI was motivated by the desire to create a more intuitive and user-friendly interface that aligns with modern design principles. By prioritizing usability and accessibility, the updated UI aims to enhance the overall user experience and facilitate seamless interaction with the database, ultimately improving user satisfaction and productivity.

### Customization of URLs for Effective Communication

Alternatives Considered:

* Using Default URLs: Could involve sticking with default URLs generated by the framework, which might not always be intuitive or optimized for the specific needs of the application.
* Manual URL Configuration: Could involve manually configuring URLs for each endpoint, providing maximum control but potentially leading to inconsistencies and maintenance challenges.

Strengths of Chosen Approach (Customized URLs):

* Improved Readability and Predictability: Customized URLs are designed to be more human-readable and predictable, making it easier for developers to understand and work with them.
* Enhanced Communication between Frontend and Backend: By customizing URLs to facilitate smooth handling of GET and POST requests, the system ensures effective communication between the frontend and backend systems, improving overall system performance.

Weaknesses:

* Initial Setup Overhead: Customizing URLs may require additional setup time and effort, particularly if the application has a large number of endpoints.

Rationale for Choice: The decision to customize various URLs was made to ensure a smooth experience with GET and POST requests, facilitating effective communication between the frontend and backend systems. By prioritizing readability, predictability, and efficient request handling, the system aims to streamline operations and improve overall system performance and maintainability.