EXPERIMENT-1

BOOK BANK SYSTEM

1. Problem Statement

The Book Bank System facilitates the lending of books and magazines to registered members. It manages the purchase of new titles, ensures popular titles are available in multiple copies, and removes outdated or damaged materials. Members can reserve unavailable items and receive notifications when they become available. The system enables easy creation, updating, and deletion of information related to titles, members, loans, and reservations.

2. Preparation of Software Requirement Specification Document

User Characteristics:

- **Students**: Individuals who wish to borrow books and submit information to the database.
- **Librarians**: Staff members with privileges to add new titles and approve book reservations.

System Modules:

- **User Registration/Login**: Secure registration for students and librarians, allowing access through an online form.
- **Book Bank Management**: A database containing all available books and magazines, enabling addition of new titles, deletion of damaged ones, and updating of book information.
- **Operations**: Students and librarians perform operations such as adding, deleting, updating, and viewing book details through the system's interface.

3. Specific Requirements

Functional Requirements:

- **User Registration**:
  - Students and librarians can register by providing personal details.
- **Book Management**:
  - Add new books with details (book number, title, author, edition, publisher).

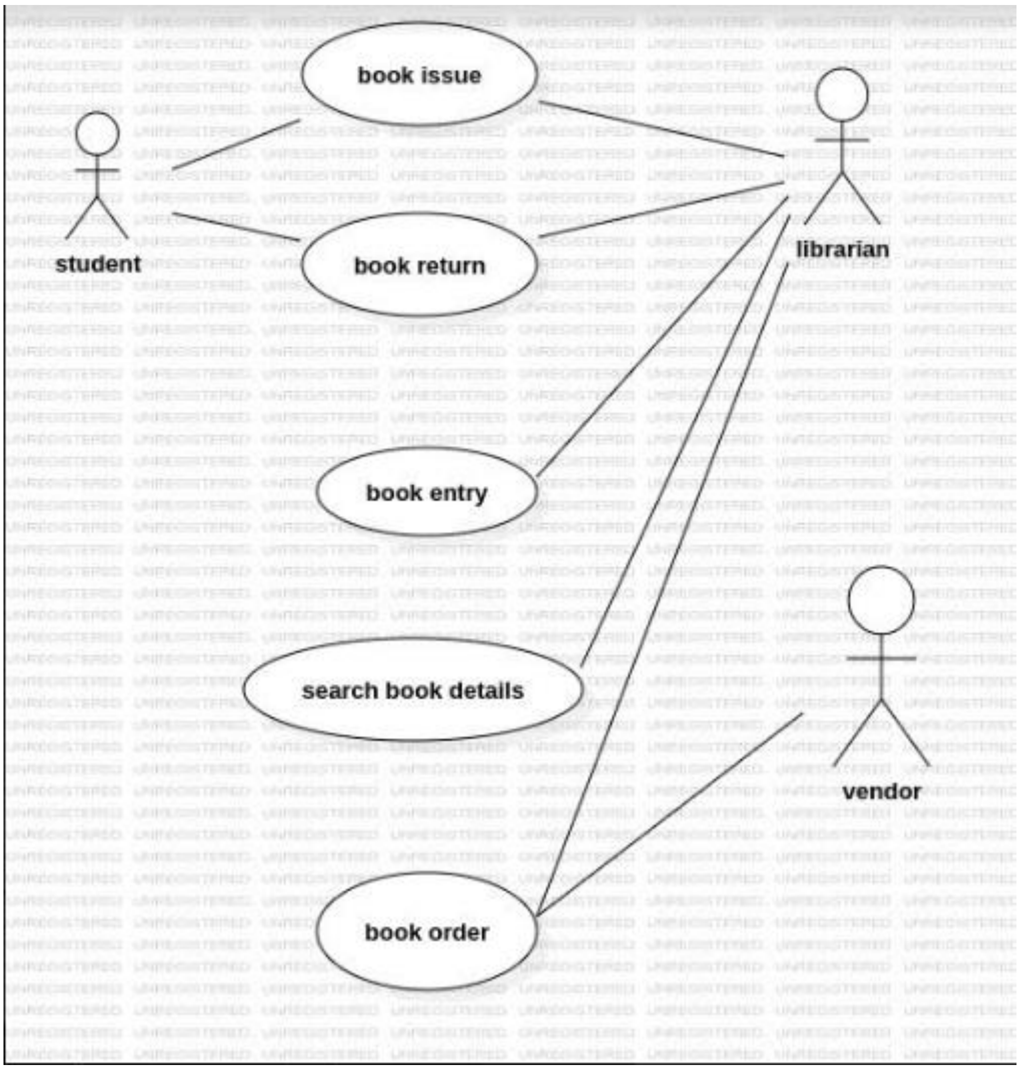
  - Delete books that are damaged or outdated.
  - Update existing book information.
- **Reservation System**:
  - Members can reserve books that are currently unavailable and receive notifications upon their return or purchase.
- **User Operations**:
  - Students can view book details and manage their loans.
  - Librarians can approve reservations and manage the book inventory.

Non-Functional Requirements:

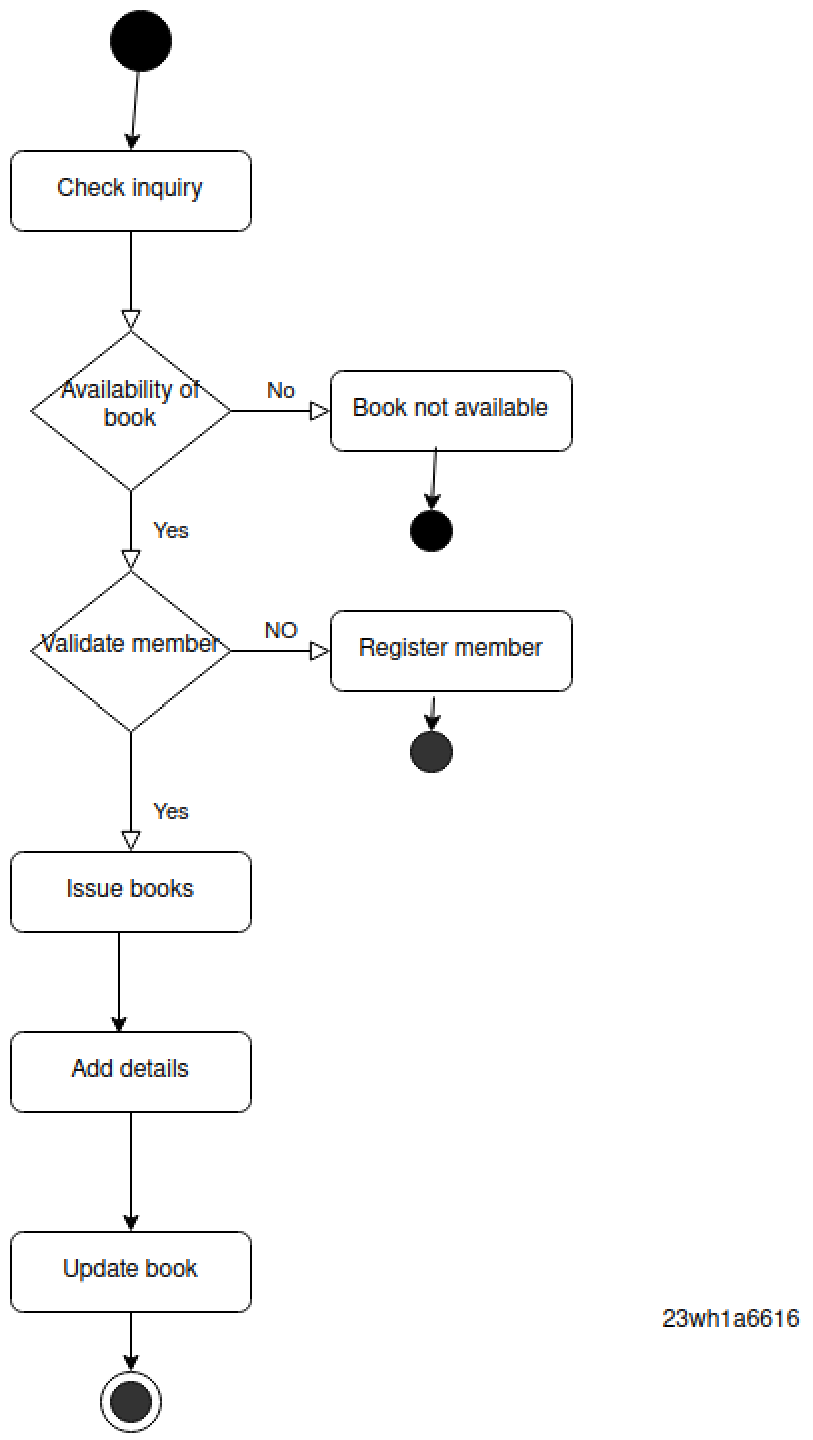
- **Privacy**: Maintain user privacy through secure management of usernames and passwords.
- **Portability**: The system should be installable and executable across various platforms (e.g., web, mobile).
- **Performance**: The system should handle multiple simultaneous users without performance degradation.

UML DIAGRAMS:

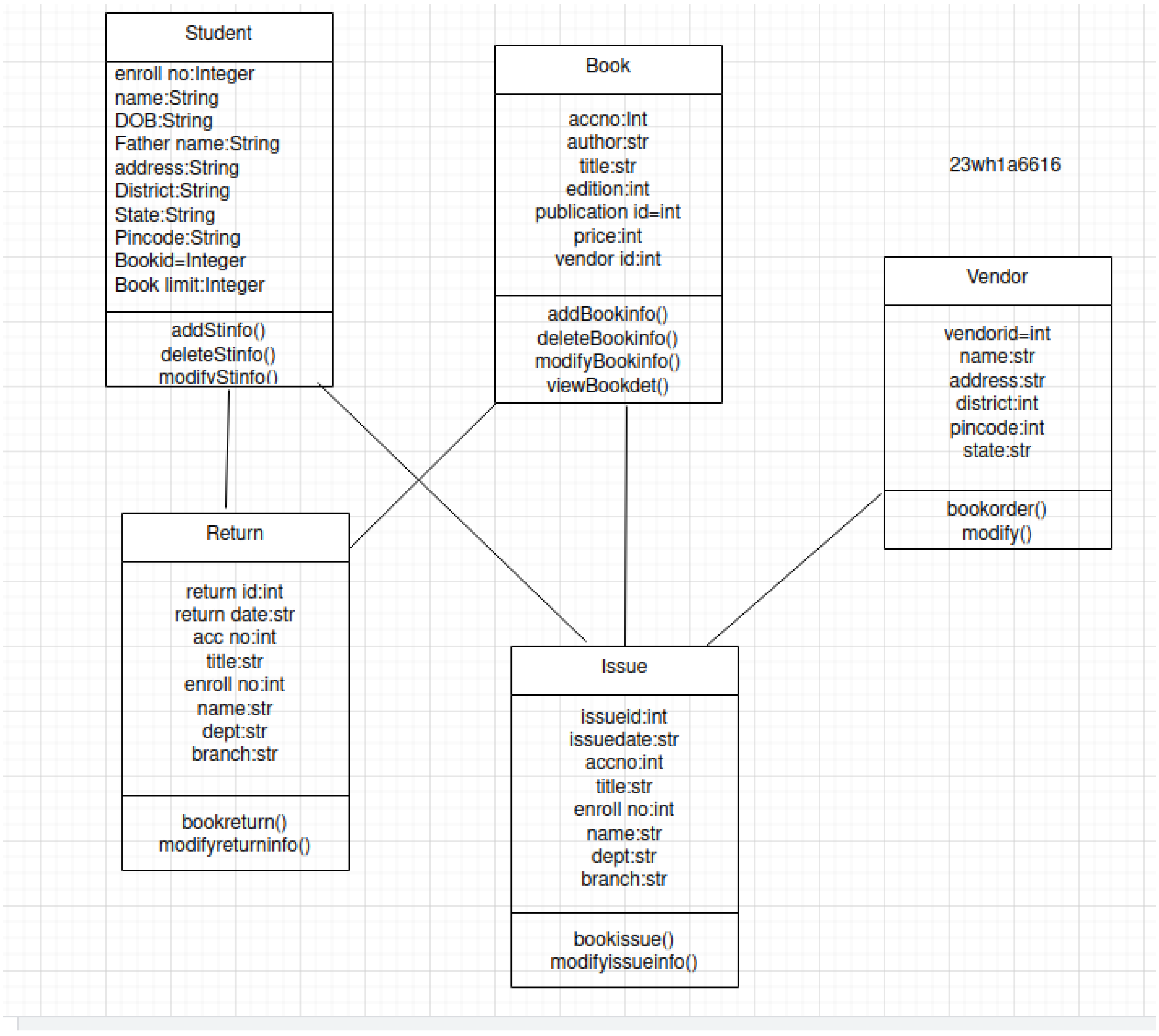
USE-CASE DIAGRAM: A use case diagram visually represents the interactions between users (actors) and a system. It outlines the system's functionalities through "use cases," depicting what the system should do from the user's perspective. Actors can be human users or other systems that interact with the primary system. The diagram helps identify system requirements and user interactions clearly and concisely. It serves as a valuable communication tool between stakeholders and developers.



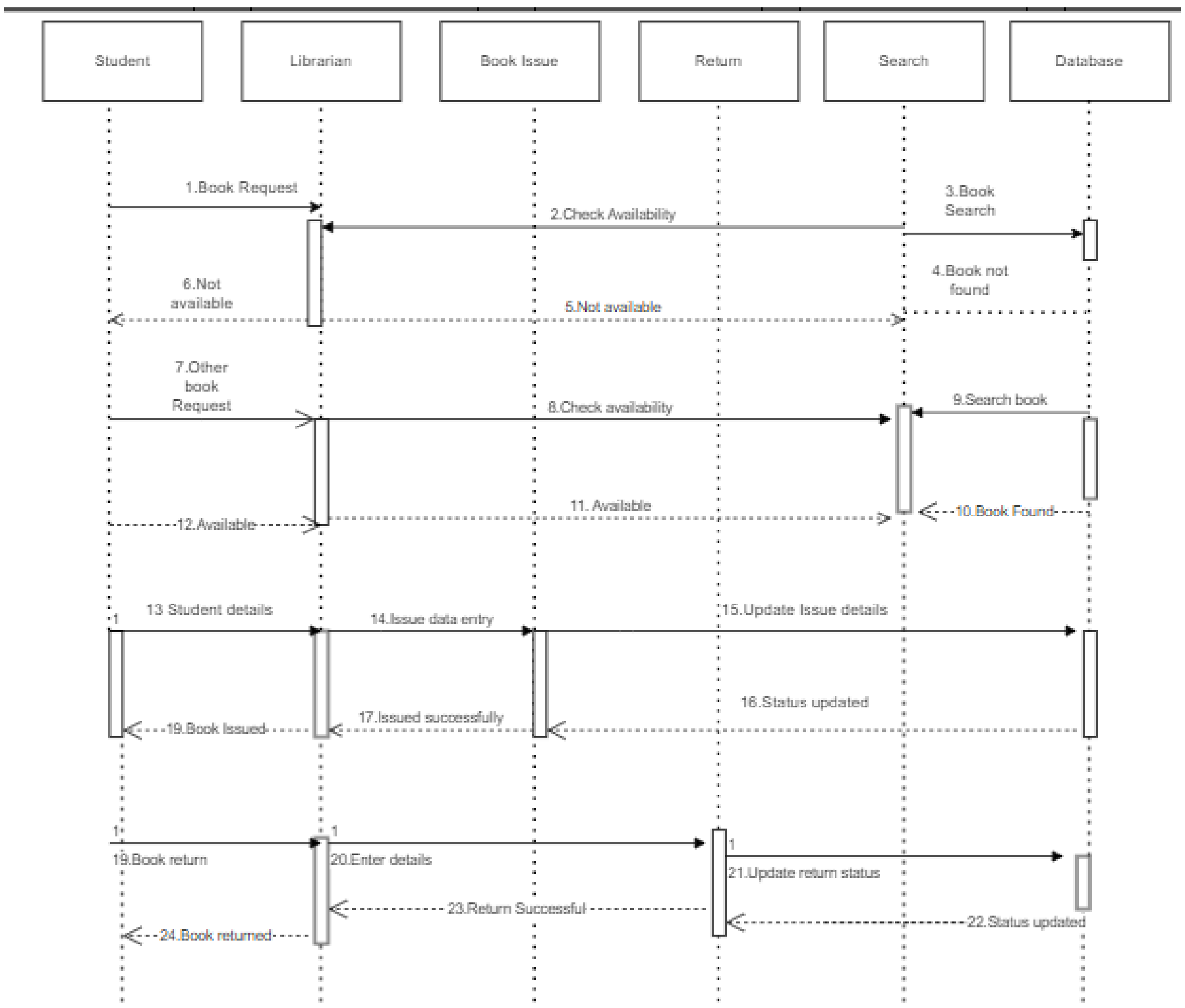
ACTIVITY DIAGRAM: An activity diagram is a type of UML (Unified Modeling Language) diagram that illustrates the flow of activities within a system or process. It showcases the sequence of actions, decisions, and parallel processes involved in completing a task. Each activity is represented by a rounded rectangle, while arrows indicate the flow from one activity to another. Decision points are depicted as diamonds, showing branching paths based on conditions. Activity diagrams are useful for modeling workflows, clarifying processes, and identifying potential bottlenecks or inefficiencies.



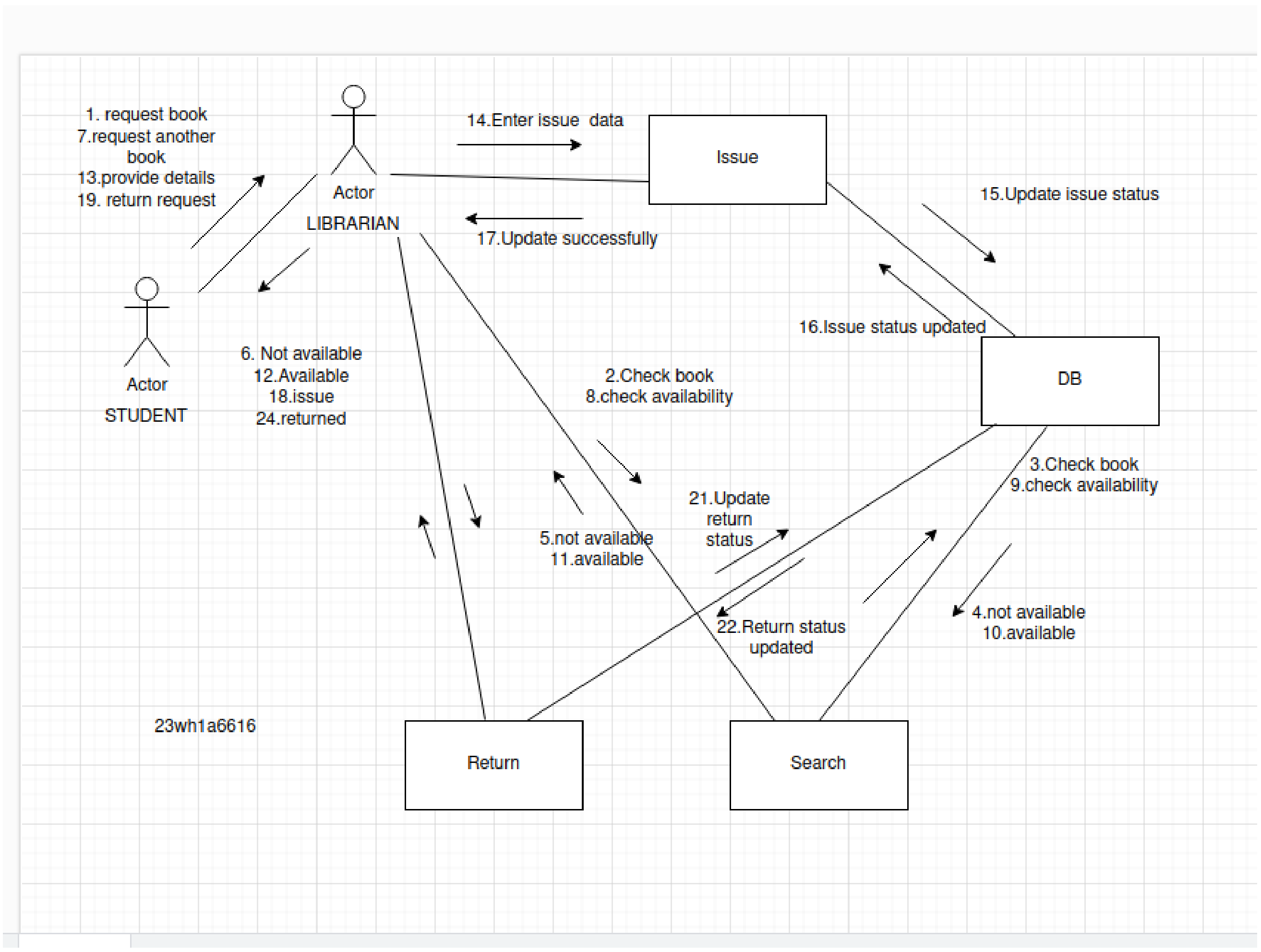
CLASS DIAGRAM: A class diagram is a static structure diagram in UML (Unified Modeling Language) that represents the classes within a system and their relationships. It shows the attributes and methods of each class, providing a blueprint for the system's structure. Classes are depicted as rectangles, with compartments for the class name, attributes, and methods. Relationships between classes, such as inheritance, association, and aggregation, are illustrated with lines and arrows. Class diagrams are essential for understanding the system's architecture and guiding the development process.



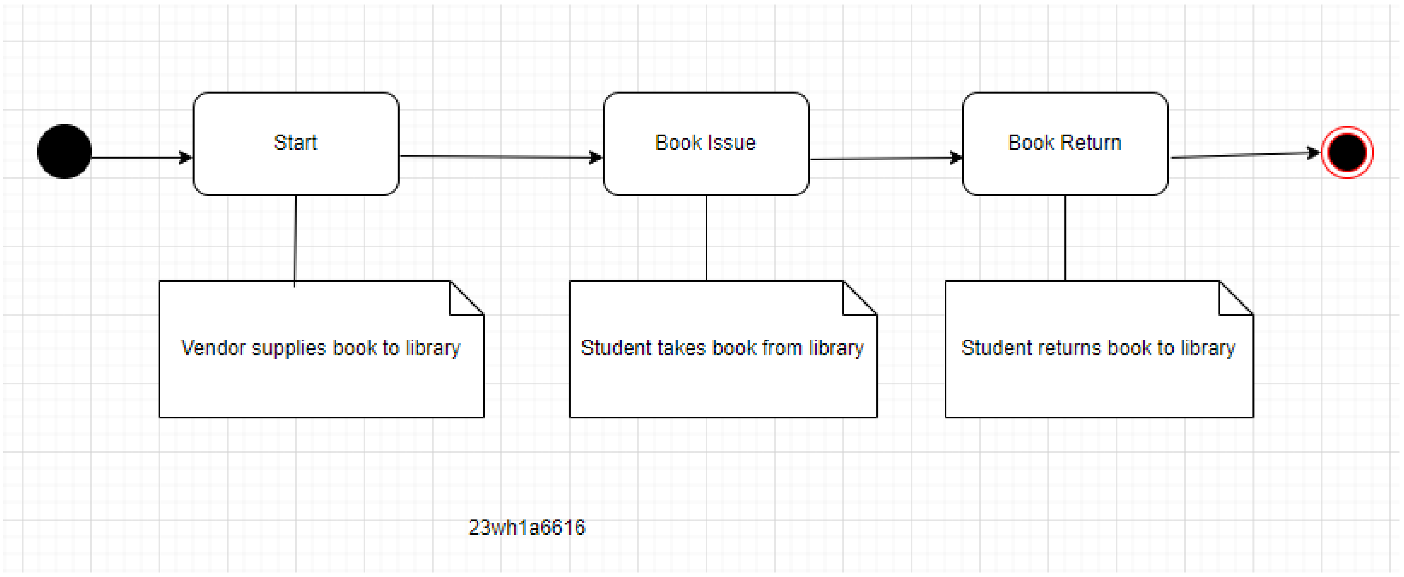
SEQUENCE DIAGRAM: A sequence diagram is a type of UML diagram that illustrates how objects interact in a particular scenario of a use case over time. It shows the sequence of messages exchanged between objects, represented by vertical lifelines, with horizontal arrows indicating the messages or calls sent between them. The timeline runs from top to bottom, capturing the order of interactions and any concurrent processes. Sequence diagrams help clarify the flow of control and data in a system, making them valuable for understanding complex interactions and identifying potential issues.



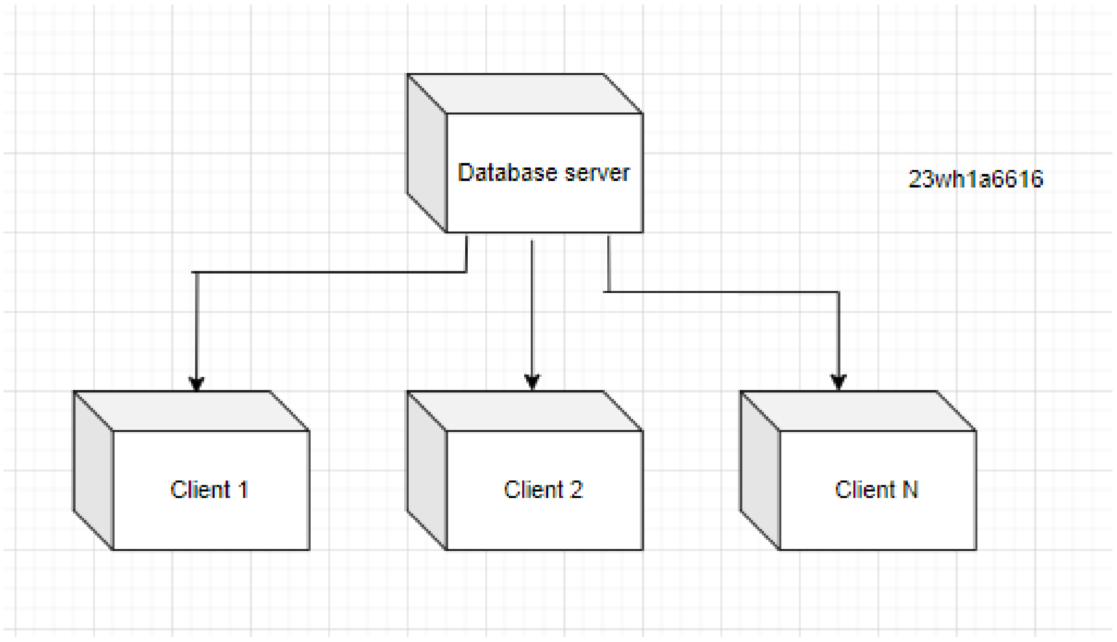
COLLABORATION DIAGRAM: A collaboration diagram, also known as a communication diagram, is a type of UML diagram that emphasizes the relationships and interactions between objects in a system. It shows how objects collaborate to fulfill a specific task or use case, using numbered messages to indicate the order of interactions. Objects are represented as rectangles, and their relationships are illustrated with connecting lines. This diagram focuses on the structural organization of the system rather than the temporal aspects, making it useful for understanding how different components work together to achieve a goal.



STATE CHART DIAGRAM: A state chart diagram is a visual representation of an object's various states and the transitions between them. It consists of rounded rectangles that denote different states, while arrows connect these states to illustrate how they change in response to events. The diagram includes an initial state marked by a filled circle and a final state indicated by a concentric circle. Each transition is labeled with events or actions that trigger the change from one state to another, providing a clear overview of the object's behavior throughout its lifecycle.


Start
Book Issue
Book Return
Vendor supplies book to library
Student takes book from library
Student returns book to library
23wh1a6616

DEPLOYMENT DIAGRAM: Deployment is the process of making a software application available for use in a live environment. It begins with preparing and testing the code, followed by setting up the necessary infrastructure. The application is then moved from development to production using various deployment methods. After deployment, monitoring is crucial to ensure performance and address any issues. Ongoing maintenance and updates are also necessary to keep the application secure and functioning optimally.



EXPERIMENT-2

PASSPORT AUTOMATION SYSTEM

1. Problem Statement

The Passport Automation System facilitates the application, processing, and issuance of passports for citizens. It allows applicants to submit their information online, upload necessary documents, track the status of their applications, and receive notifications upon completion. The system also provides administrative tools for officials to manage applications, verify documents, and ensure compliance with regulatory requirements.

2. Preparation of Software Requirement Specification Document

User Characteristics:

- **Applicants**: Citizens applying for a passport who need to fill out forms and upload documents.
- **Officials**: Government personnel with privileges to review applications, approve or reject requests, and manage the overall system.

System Modules:

- **User Registration/Login**: Secure registration and login for applicants and officials, including identity verification.
- **Application Management**: Applicants can create, edit, and submit passport applications. They can upload required documents and track application status.
- **Verification and Approval**: Officials can review submitted applications, verify documents, approve or reject applications, and manage issuance of passports.
- **Notifications**: Automated notifications to applicants regarding application status and passport readiness.

Operations:

- Applicants and officials perform various operations through a user-friendly web interface, including submitting applications, reviewing documents, and managing user accounts.

3. Specific Requirements

Functional Requirements:

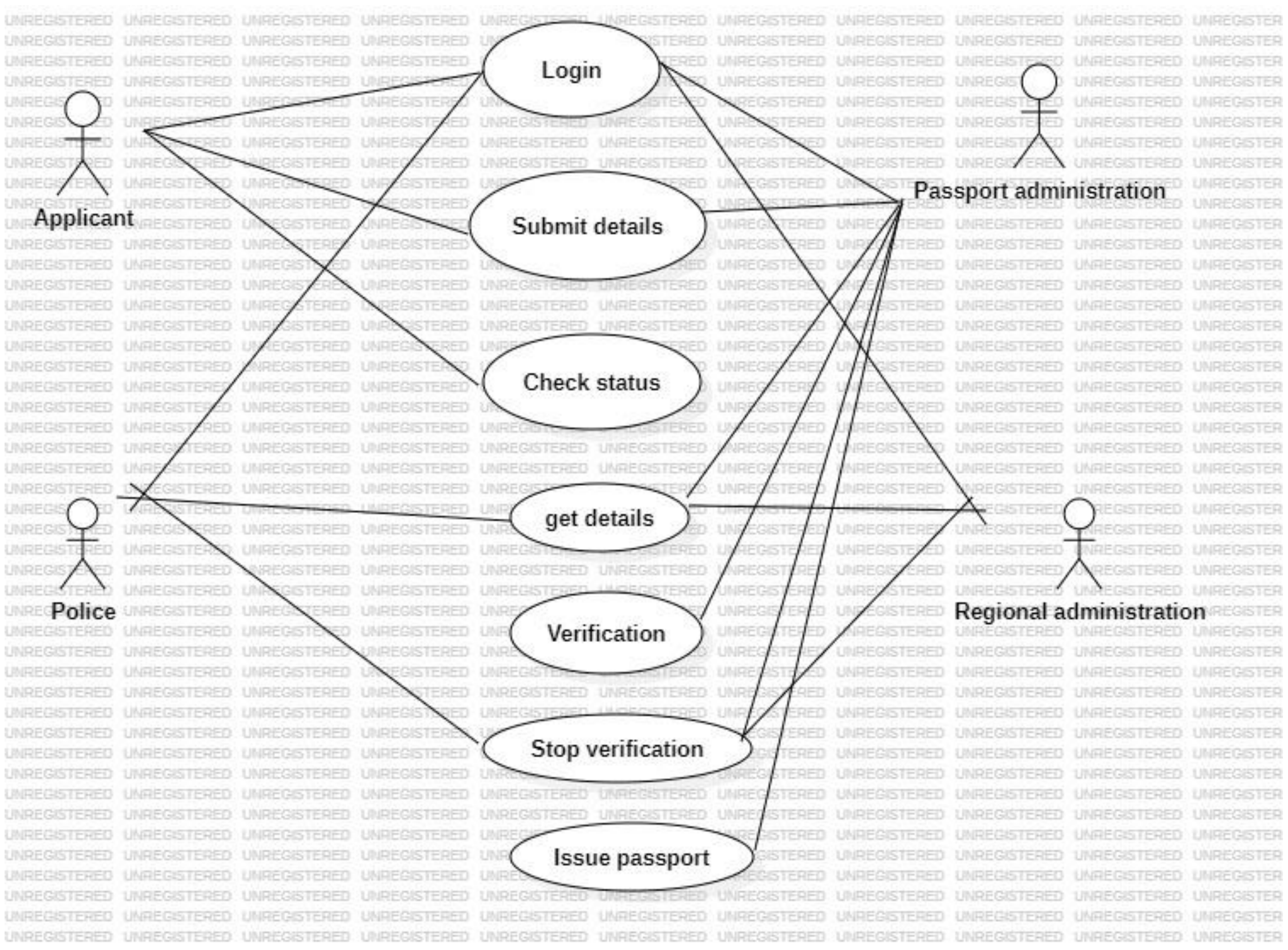
- **User Registration**: Applicants and officials can register securely, providing personal details and authentication.
- **Application Submission**:
  - Applicants can fill out the online passport application form.
  - Upload necessary documents (e.g., ID proof, photographs).
- **Application Tracking**: Applicants can view the status of their application in real-time.
- **Administrative Tools**: Officials can:
  - View and manage all applications.
  - Approve or reject applications based on compliance.
  - Issue notifications to applicants.

Non-Functional Requirements:

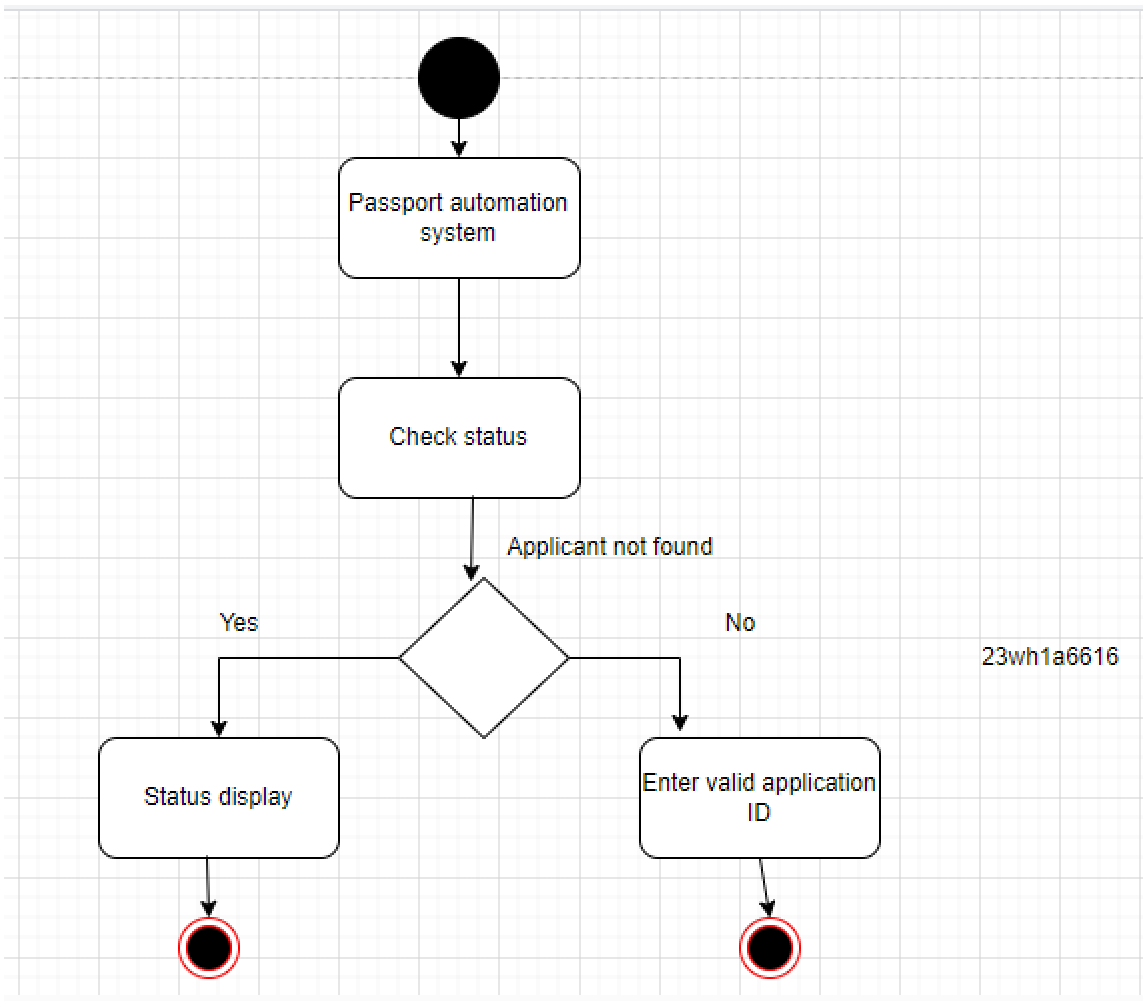
- **Security**: Ensure secure handling of personal information and documents through encryption and access control.
- **Usability**: The interface must be user-friendly and accessible, accommodating users with varying levels of tech proficiency.
- **Performance**: The system should handle multiple concurrent users without degradation of performance.
- **Availability**: The system should be available 24/7 with minimal downtime for maintenance.

UML DIAGRAMS:

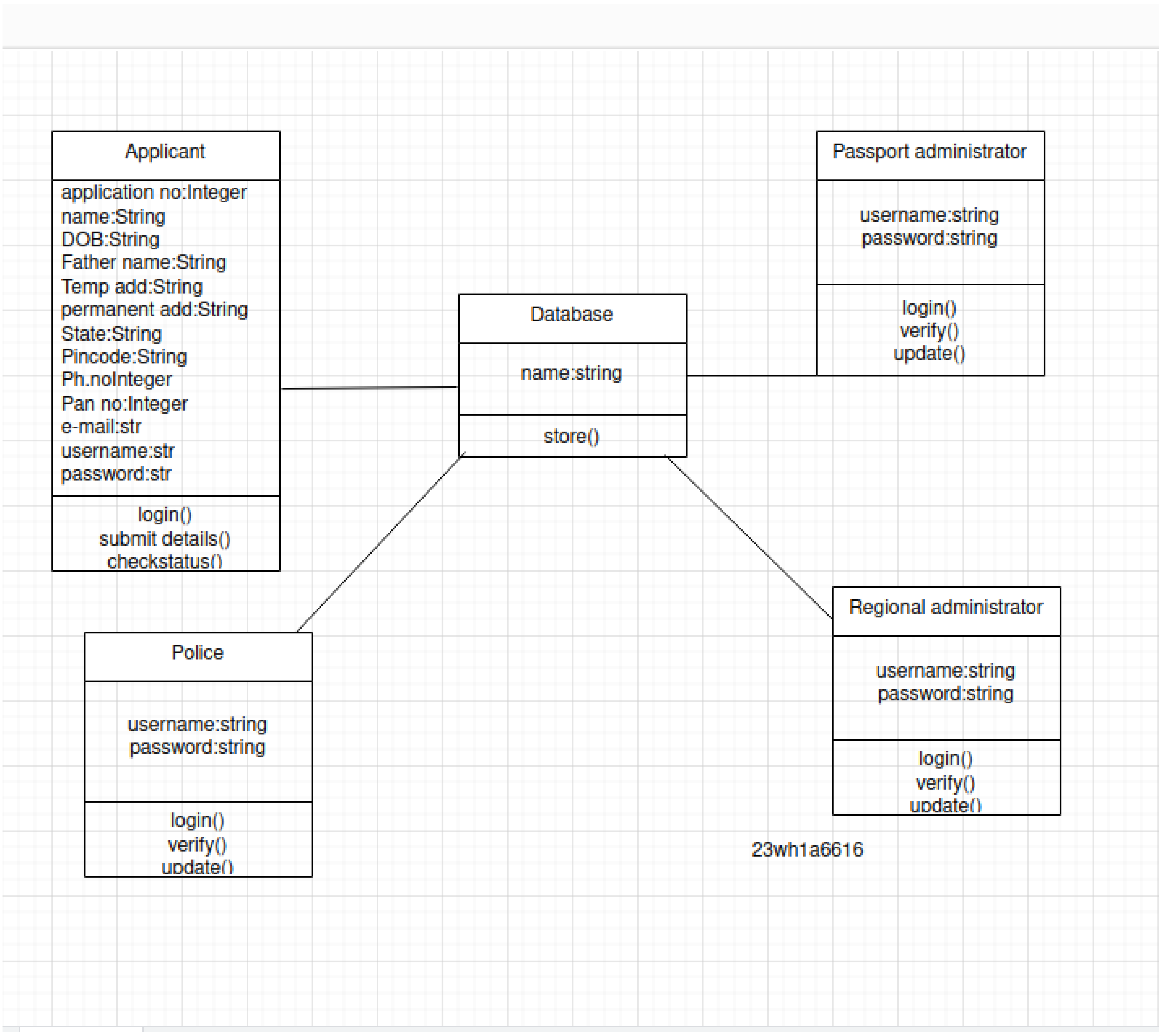
USE CASE DIAGRAM: A use case diagram visually represents the interactions between users (actors) and a system. It outlines the system's functionalities through "use cases," depicting what the system should do from the user's perspective. Actors can be human users or other systems that interact with the primary system. The diagram helps identify system requirements and user interactions clearly and concisely. It serves as a valuable communication tool between stakeholders and developers.



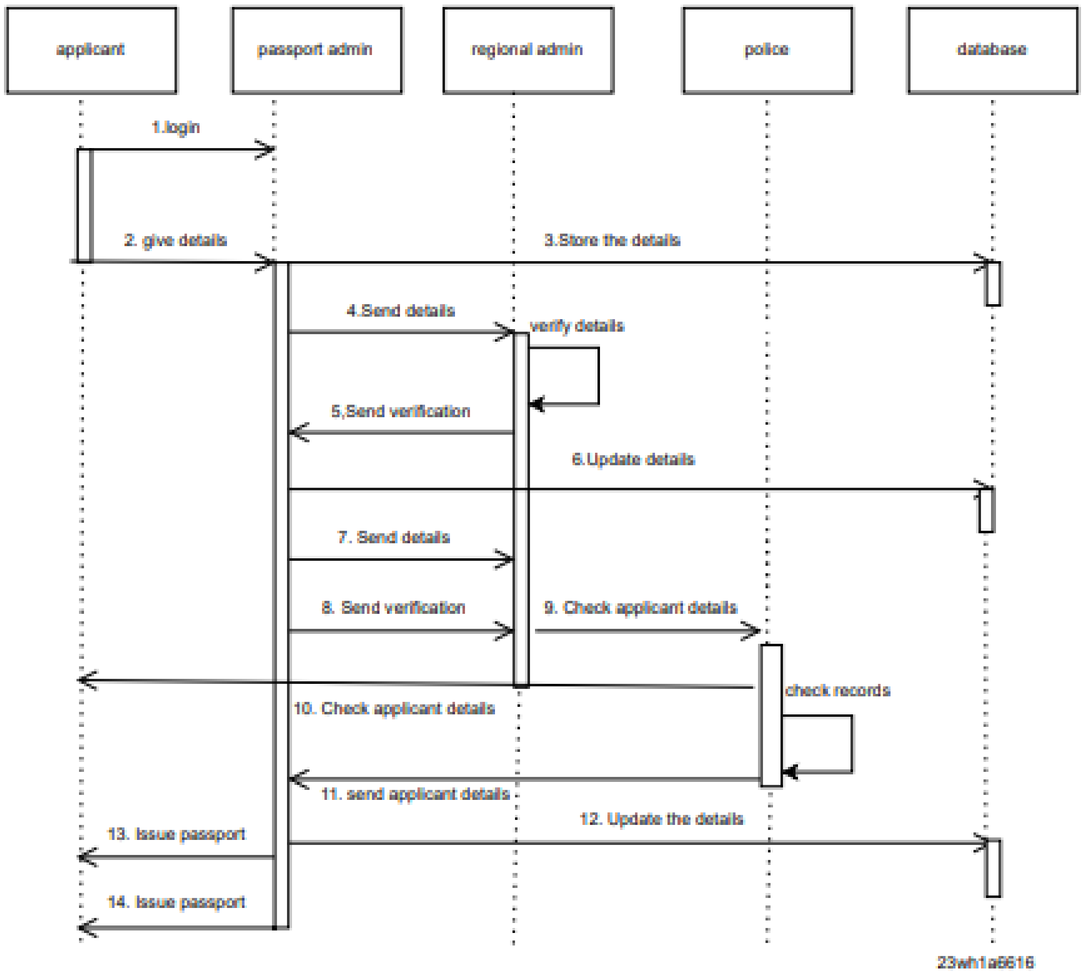
ACTIVITY DIAGRAM: An activity diagram is a type of UML (Unified Modeling Language) diagram that illustrates the flow of activities within a system or process. It showcases the sequence of actions, decisions, and parallel processes involved in completing a task. Each activity is represented by a rounded rectangle, while arrows indicate the flow from one activity to another. Decision points are depicted as diamonds, showing branching paths based on conditions. Activity diagrams are useful for modeling workflows, clarifying processes, and identifying potential bottlenecks or inefficiencies.



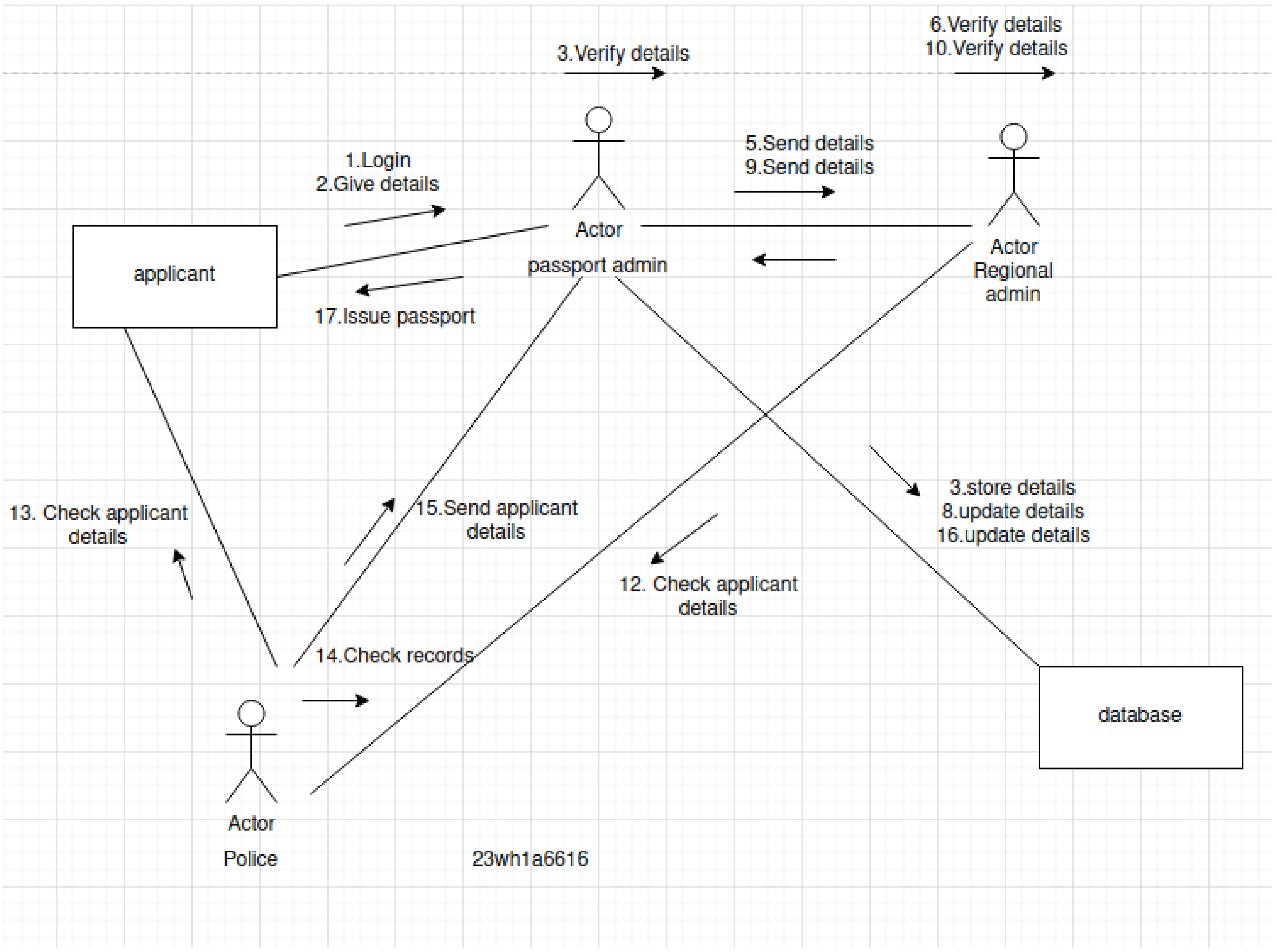
CLASS DIAGRAM: A class diagram is a static structure diagram in UML (Unified Modeling Language) that represents the classes within a system and their relationships. It shows the attributes and methods of each class, providing a blueprint for the system's structure. Classes are depicted as rectangles, with compartments for the class name, attributes, and methods. Relationships between classes, such as inheritance, association, and aggregation, are illustrated with lines and arrows. Class diagrams are essential for understanding the system's architecture and guiding the development process.



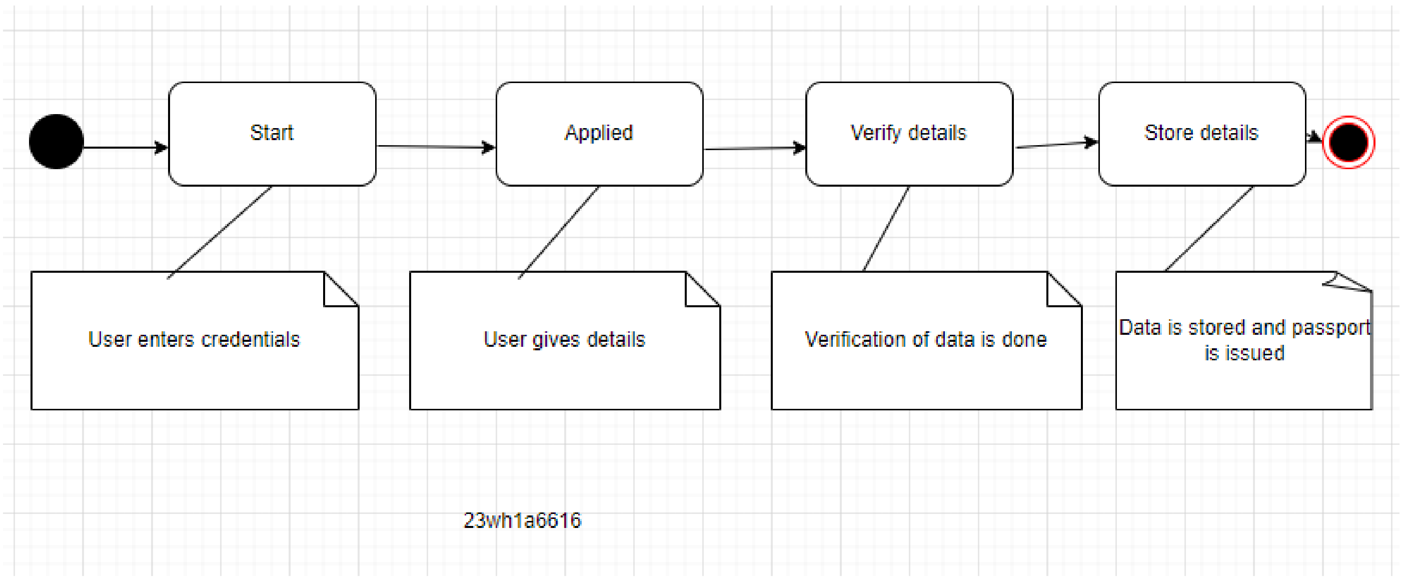
SEQUENCE DIAGRAM: A sequence diagram is a type of UML diagram that illustrates how objects interact in a particular scenario of a use case over time. It shows the sequence of messages exchanged between objects, represented by vertical lifelines, with horizontal arrows indicating the messages or calls sent between them. The timeline runs from top to bottom, capturing the order of interactions and any concurrent processes. Sequence diagrams help clarify the flow of control and data in a system, making them valuable for understanding complex interactions and identifying potential issues.



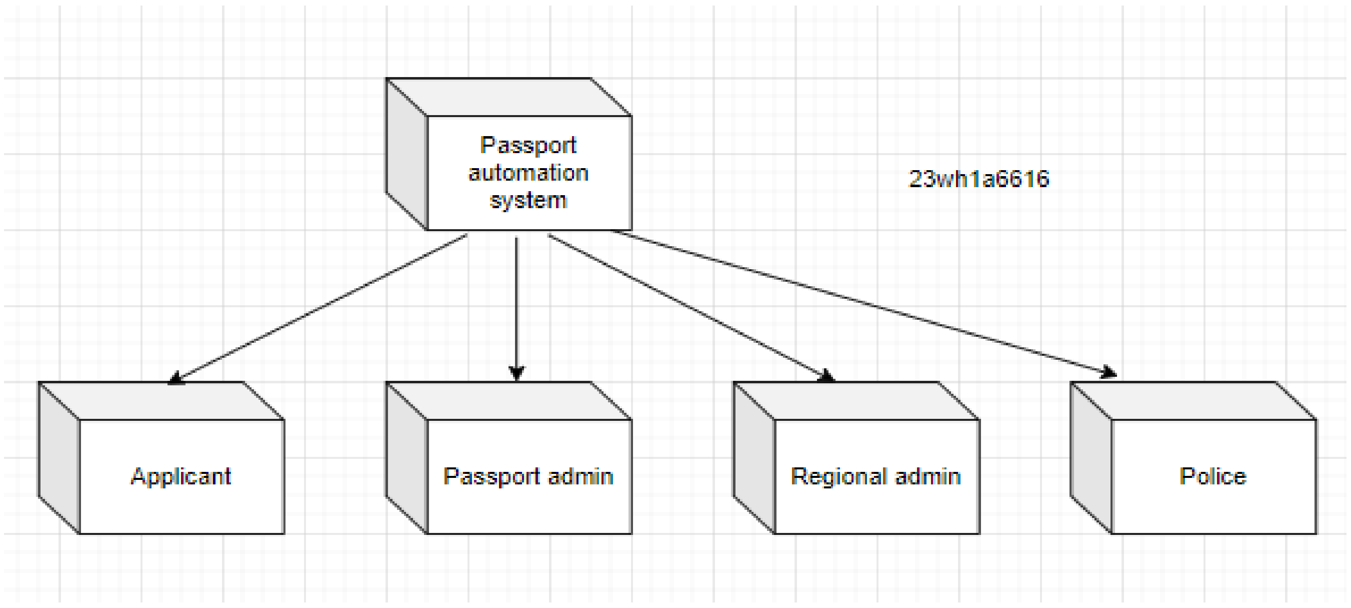
COLLABORATION DIAGRAM: A collaboration diagram, also known as a communication diagram, is a type of UML diagram that emphasizes the relationships and interactions between objects in a system. It shows how objects collaborate to fulfill a specific task or use case, using numbered messages to indicate the order of interactions. Objects are represented as rectangles, and their relationships are illustrated with connecting lines. This diagram focuses on the structural organization of the system rather than the temporal aspects, making it useful for understanding how different components work together to achieve a goal.



STATE CHART DIAGRAM: A state chart diagram is a visual representation of an object's various states and the transitions between them. It consists of rounded rectangles that denote different states, while arrows connect these states to illustrate how they change in response to events. The diagram includes an initial state marked by a filled circle and a final state indicated by a concentric circle. Each transition is labeled with events or actions that trigger the change from one state to another, providing a clear overview of the object's behavior throughout its lifecycle.



DEPLOYMENT DIAGRAM: Deployment is the process of making a software application available for use in a live environment. It begins with preparing and testing the code, followed by setting up the necessary infrastructure. The application is then moved from development to production using various deployment methods. After deployment, monitoring is crucial to ensure performance and address any issues. Ongoing maintenance and updates are also necessary to keep the application secure and functioning optimally.


Passport automation system
23wh1a6616
Applicant
Passport admin
Regional admin
Police