Project Report

Group2

Group Members: Zijian Chen, Yichun Gong, Xueming Feng, Honghao Zeng.

Project Name: Blood Bank

Project Description:

### Donor

1. Upload Information

2. Respond to donate request

3. Donate blood

4. View donation history

### Requestor

1. Upload Information

2. Make blood request

3. Check request status

4. View request history

### Administrator

1. Check user information

2. Remove user

3. Modify user

4. Check donation history

5. Remove donation history

6. Check all blood types

7. Add blood type

### Mobile Client

1. Register new user

2. Provide login for donor and requestor

### Website Client

1. Provide login for administrator

Functional requirement:

1. Donor account system:
2. The system should let donor create, login and change account (user id and password)
3. The system should let donor upload his/her personal information
4. The system should let donor declares intents to donate.
5. The system should display the donation history to the donor.
6. The system should take into account donor’s donation history when this donor declares intents to donate.
7. The system will notify the appropriate donor when new request appears.
8. The system will also notify the donor if he/she is chosen to donate blood.
9. Requester account system:
10. The system should let requester donor create, login and change account (user id and password)
11. The system should let requester upload his/her personal information
12. The system should let requester upload his/her detailed request information.
13. The system will notify the requester when it find the appropriate donor.
14. The system will display the status of the request.

Non-Functional requirements:

1. Operational Requirements
   1. An admin web interface should be provided to manage users.
   2. System should have user friendly UI.
   3. System should be able to connect to backend database.
   4. System should handle invalid input on both backend and frontend.

2. Performance requirements:

1. System should allow more then 100 users online at the same time.

2. System should be available 24/7.

3. Security Requirement

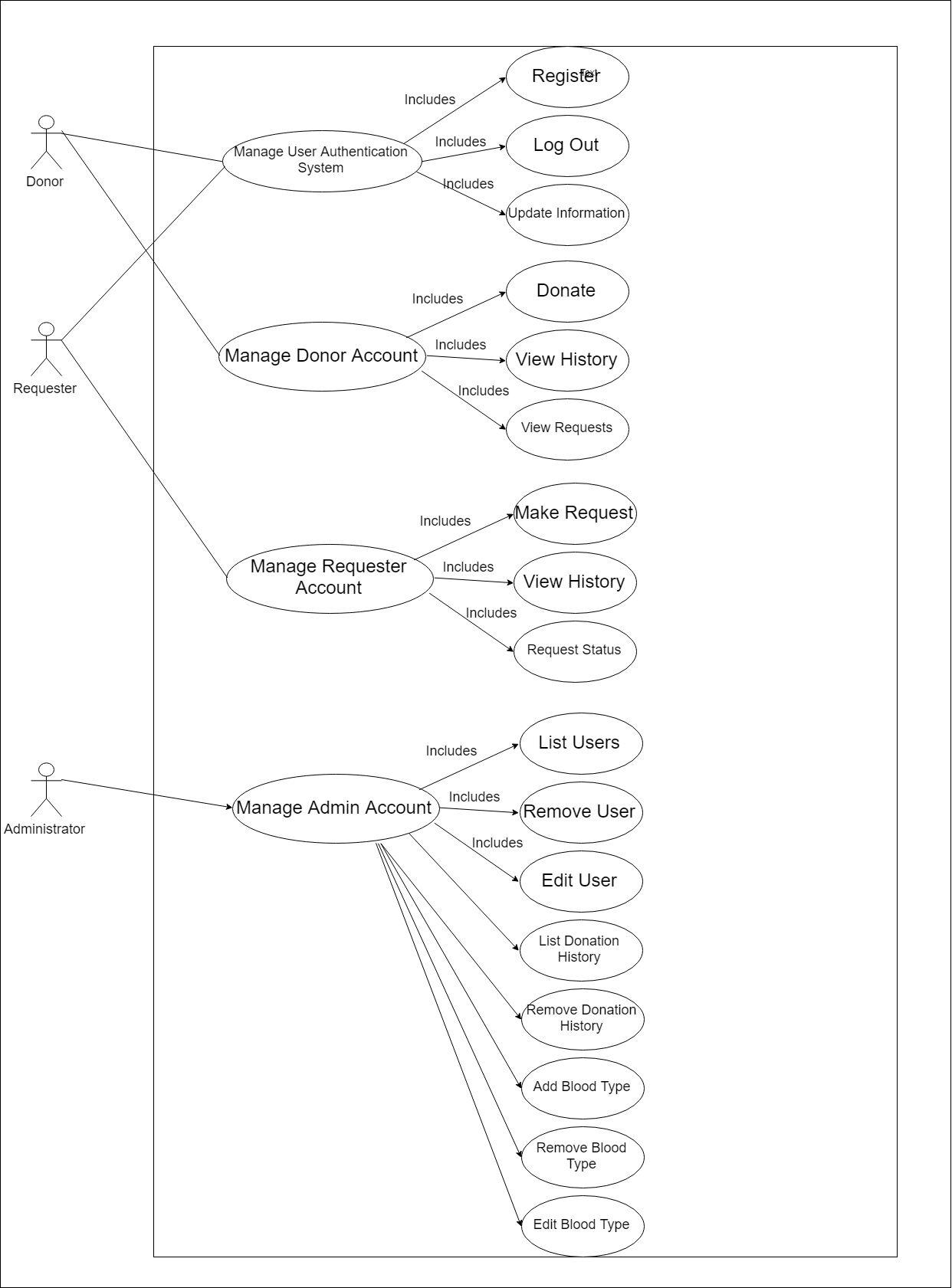
1. Backend should check for invalid input.

2. Backend should handle database error and should filter input for potential abuses (e.g. SQL inject, XSS, CSRF)

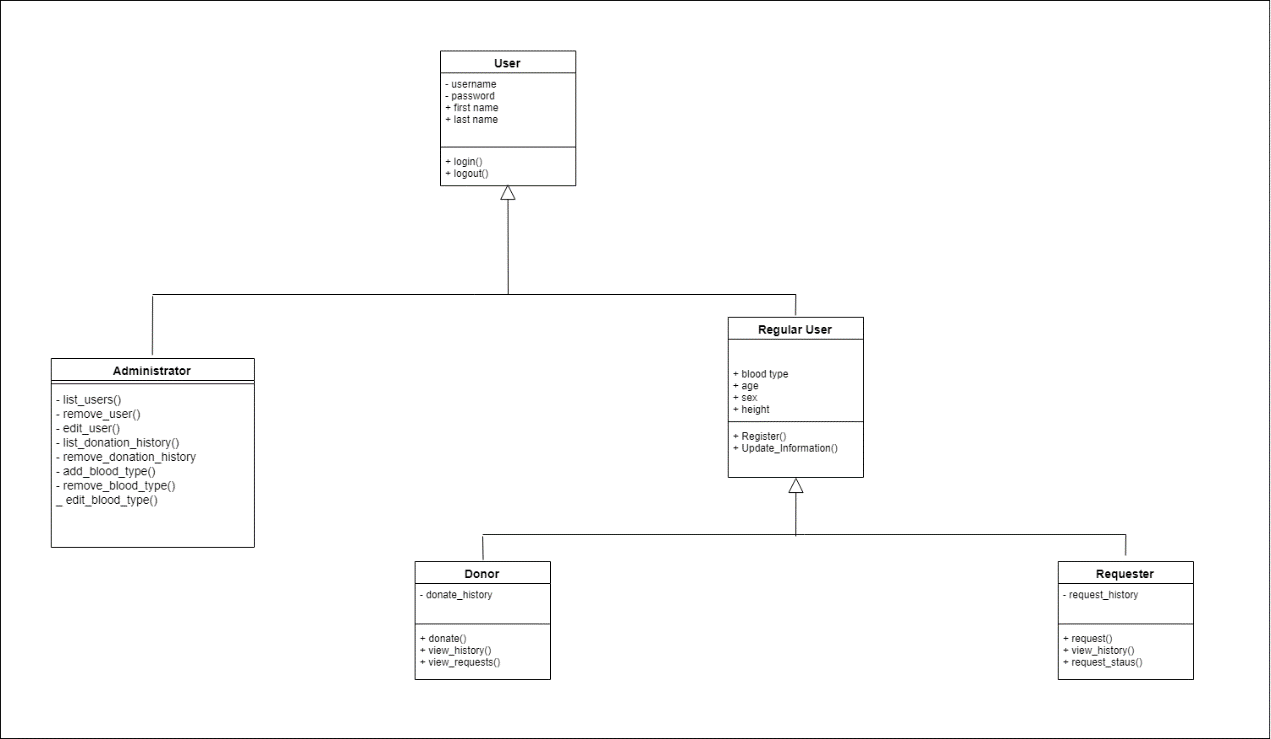
3. User should not be able to access the services that they don't have access to. (e.g. Donor should not be allow to use Requester's API)

4. User's password should be stored in hashed format, not plaintext.

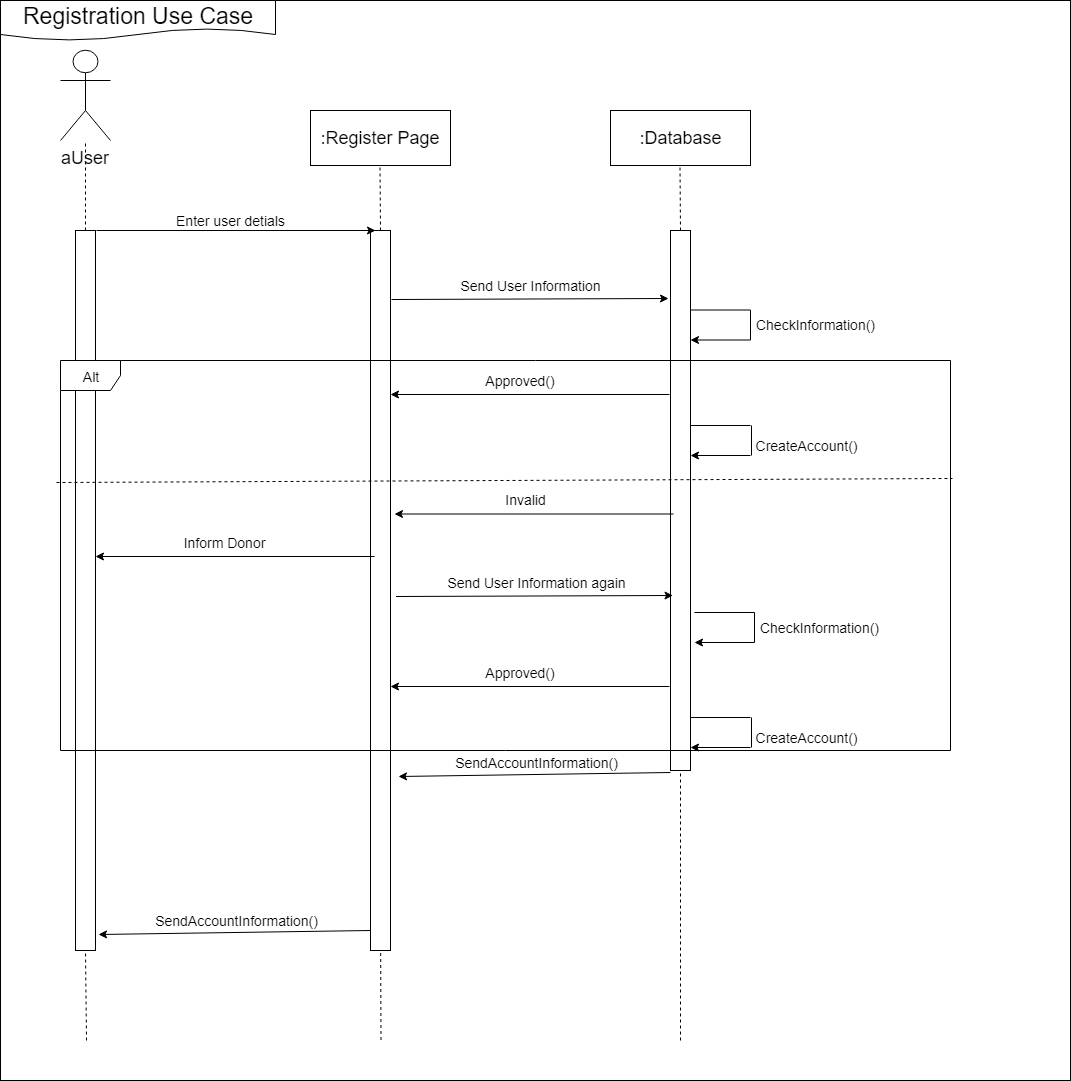
Use Case Diagram:

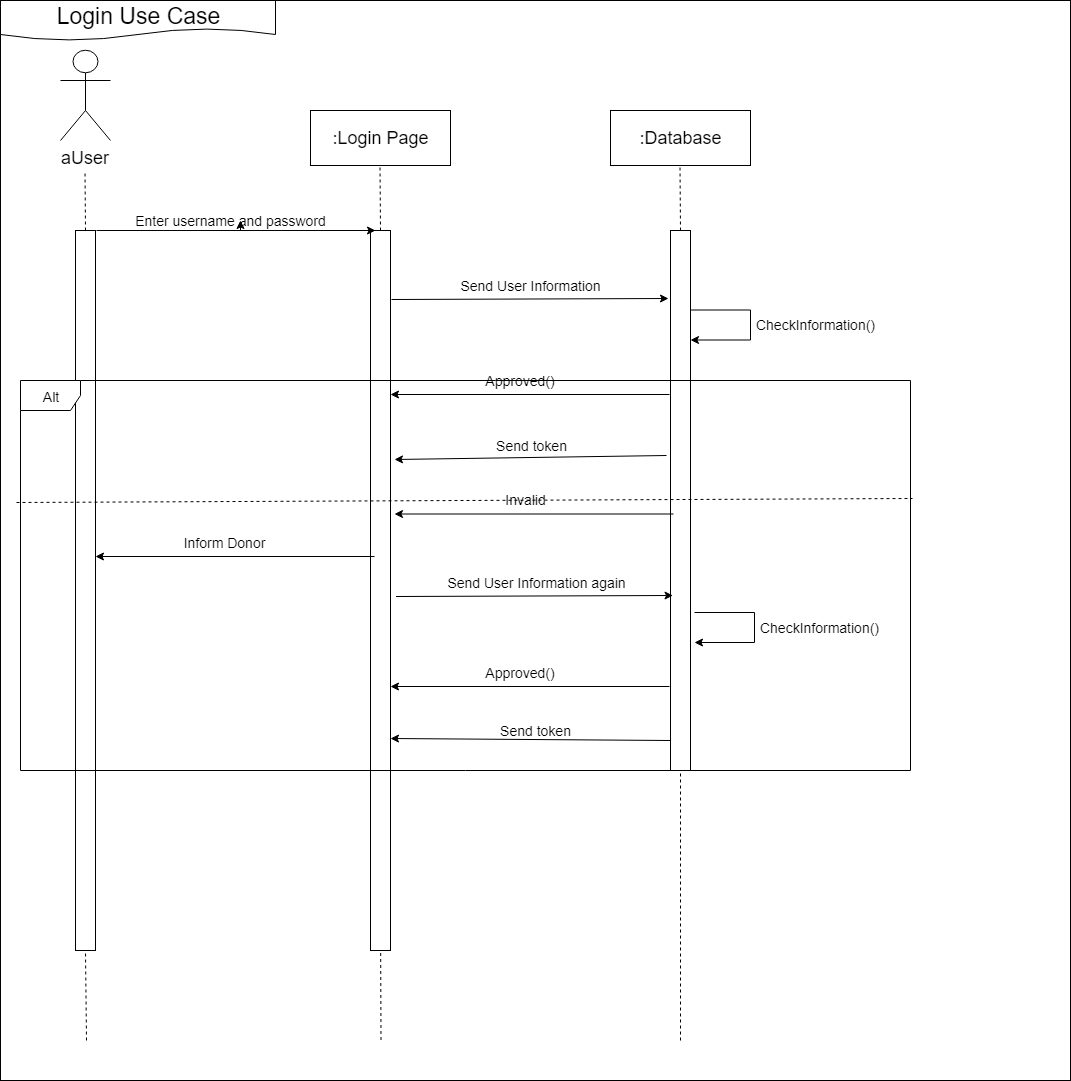


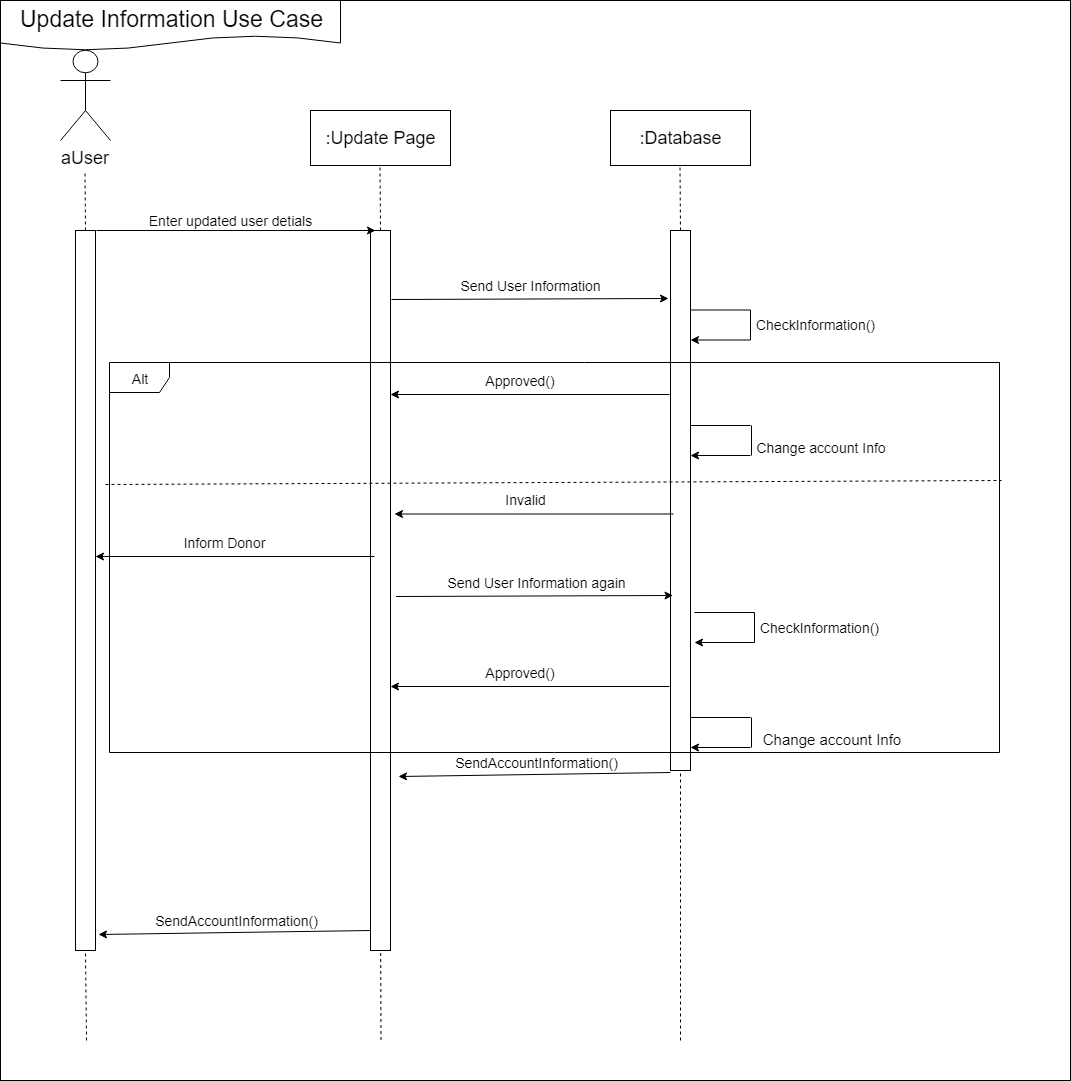
Class diagram

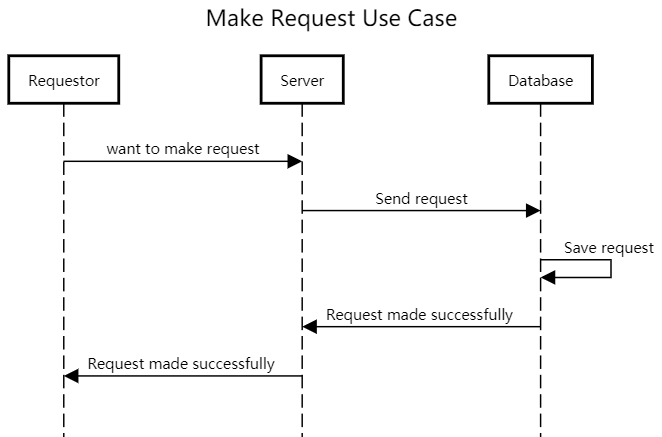


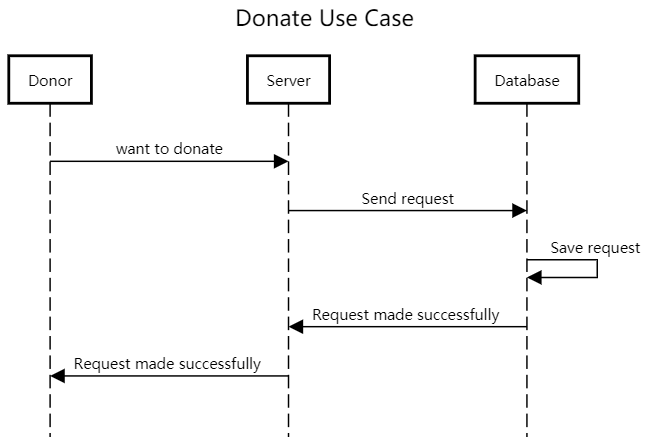
sequence diagrams

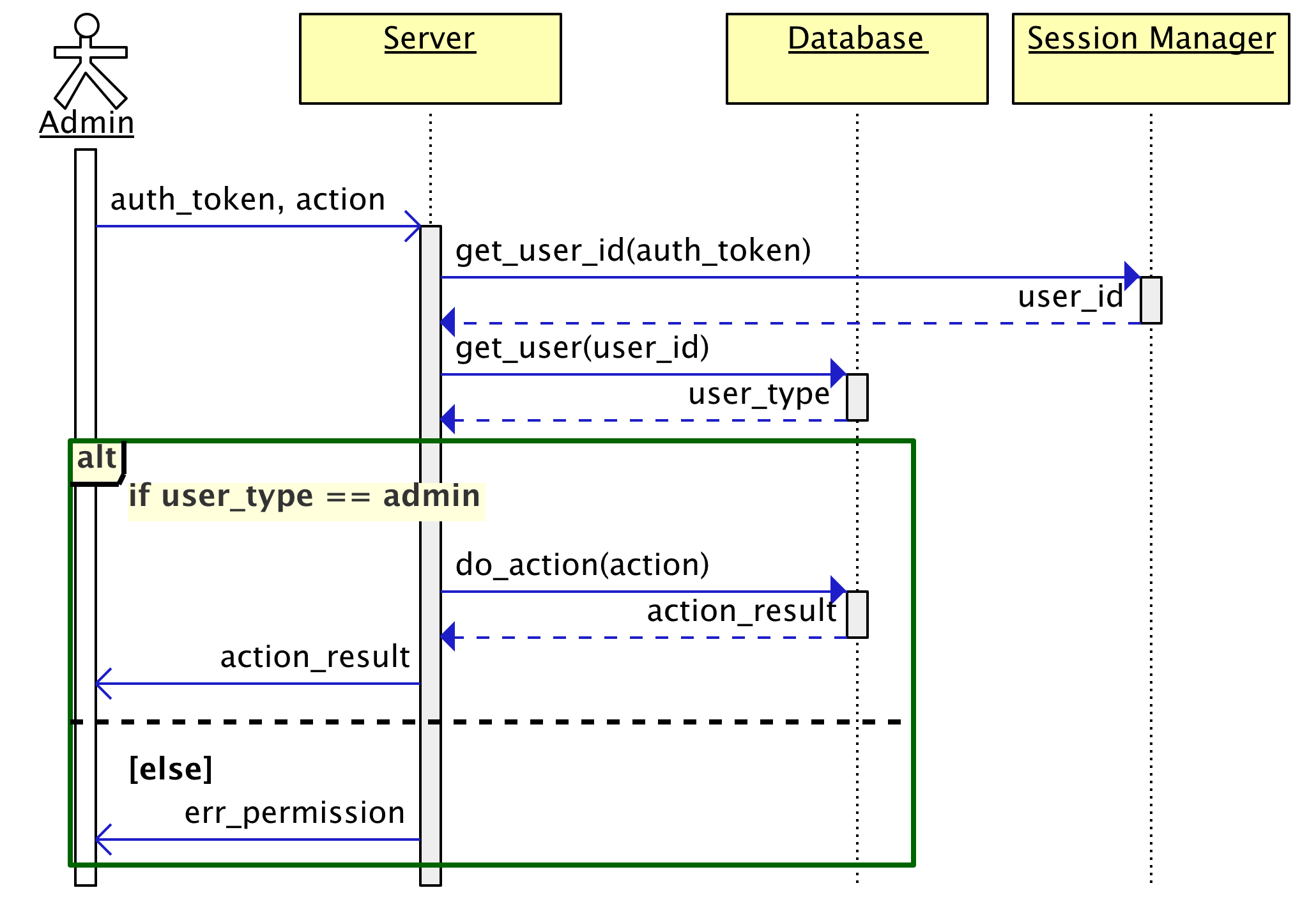


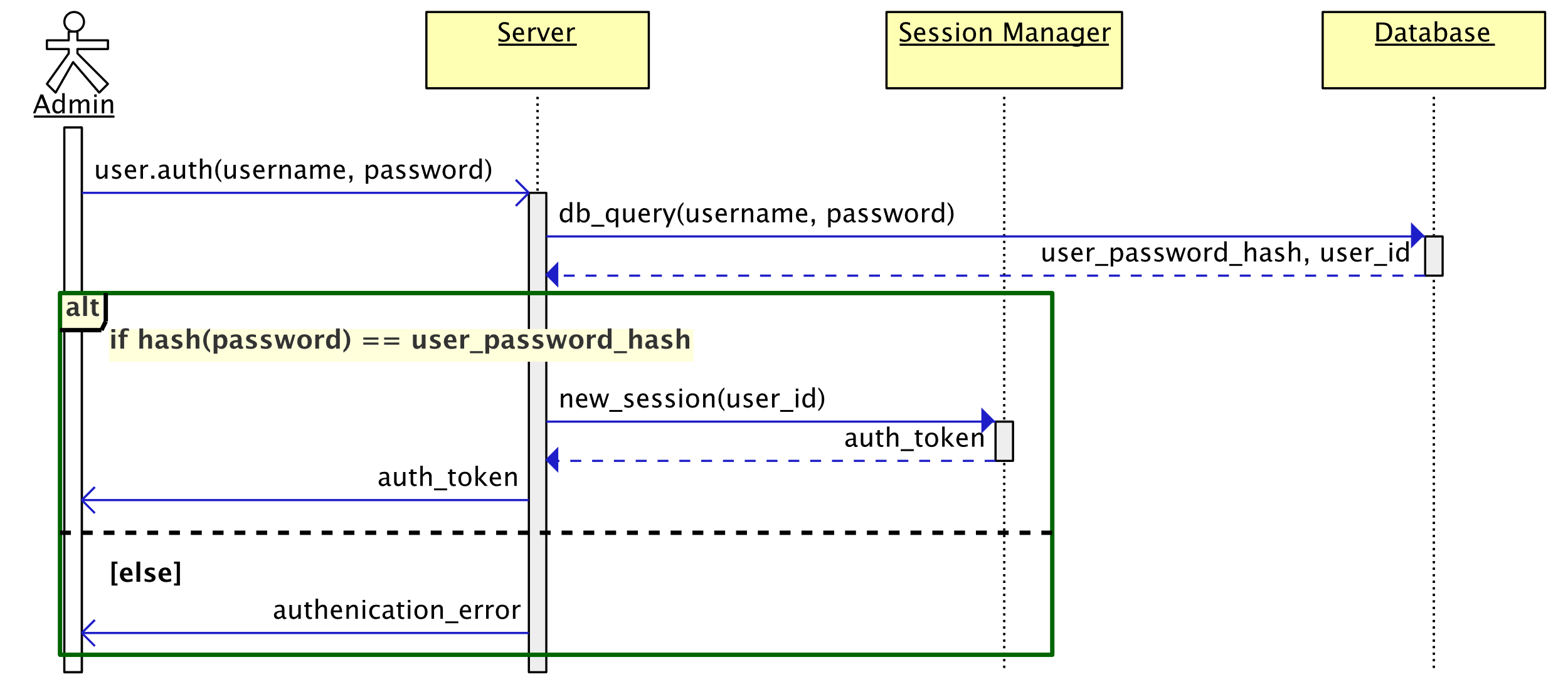


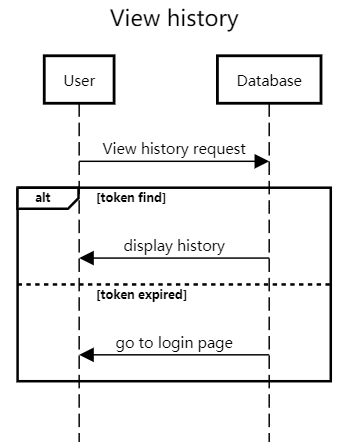




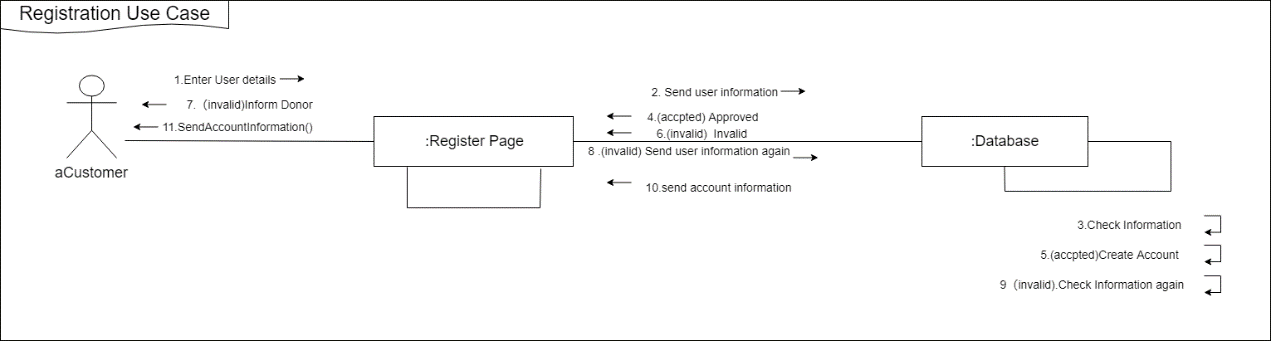




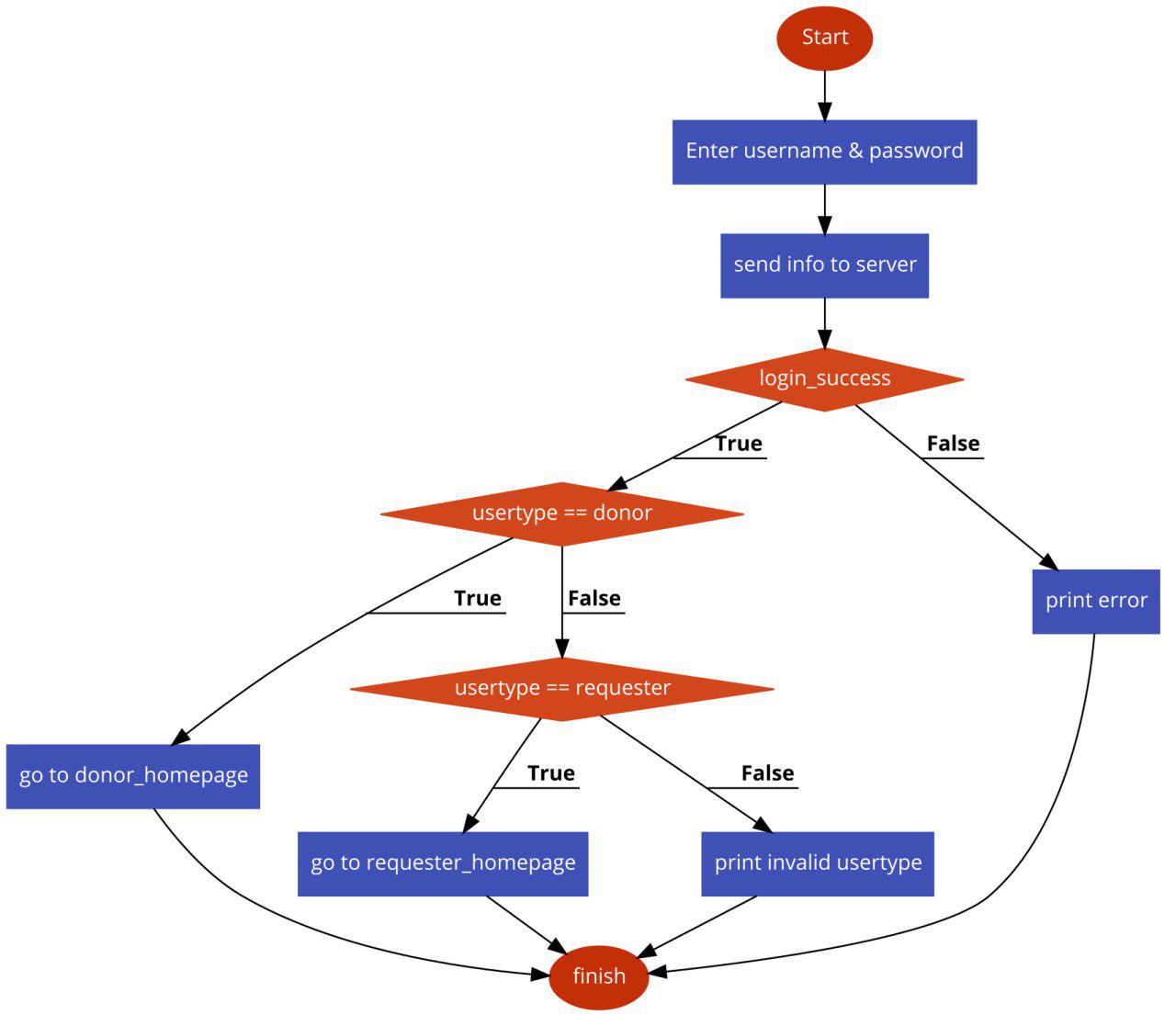


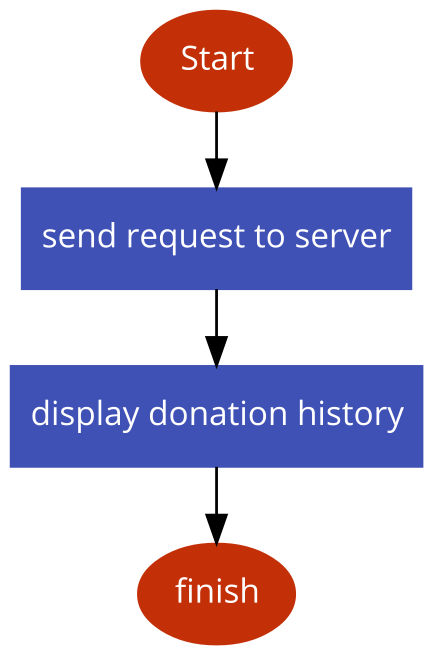
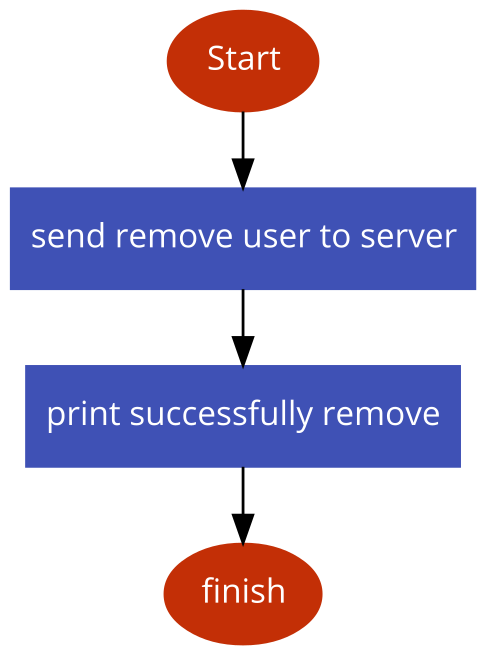
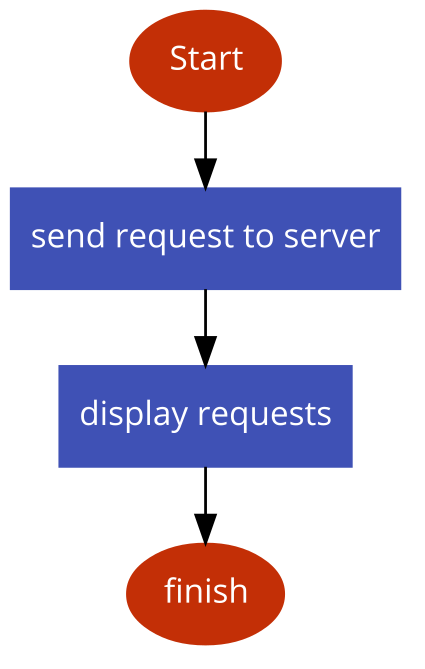
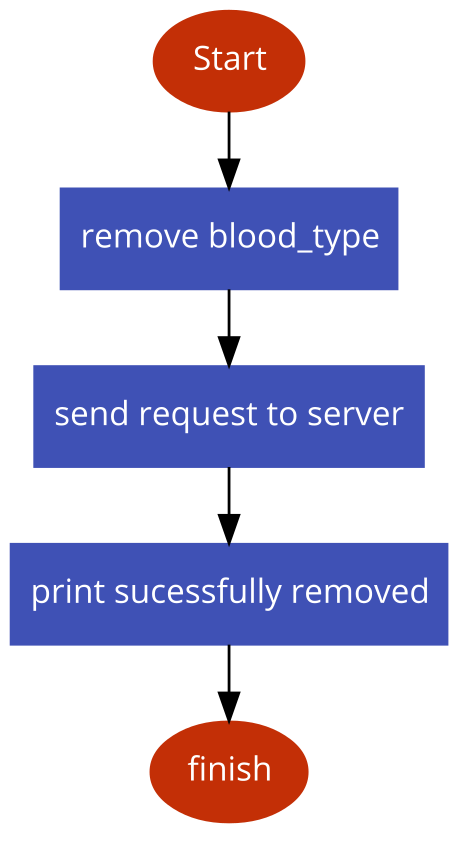
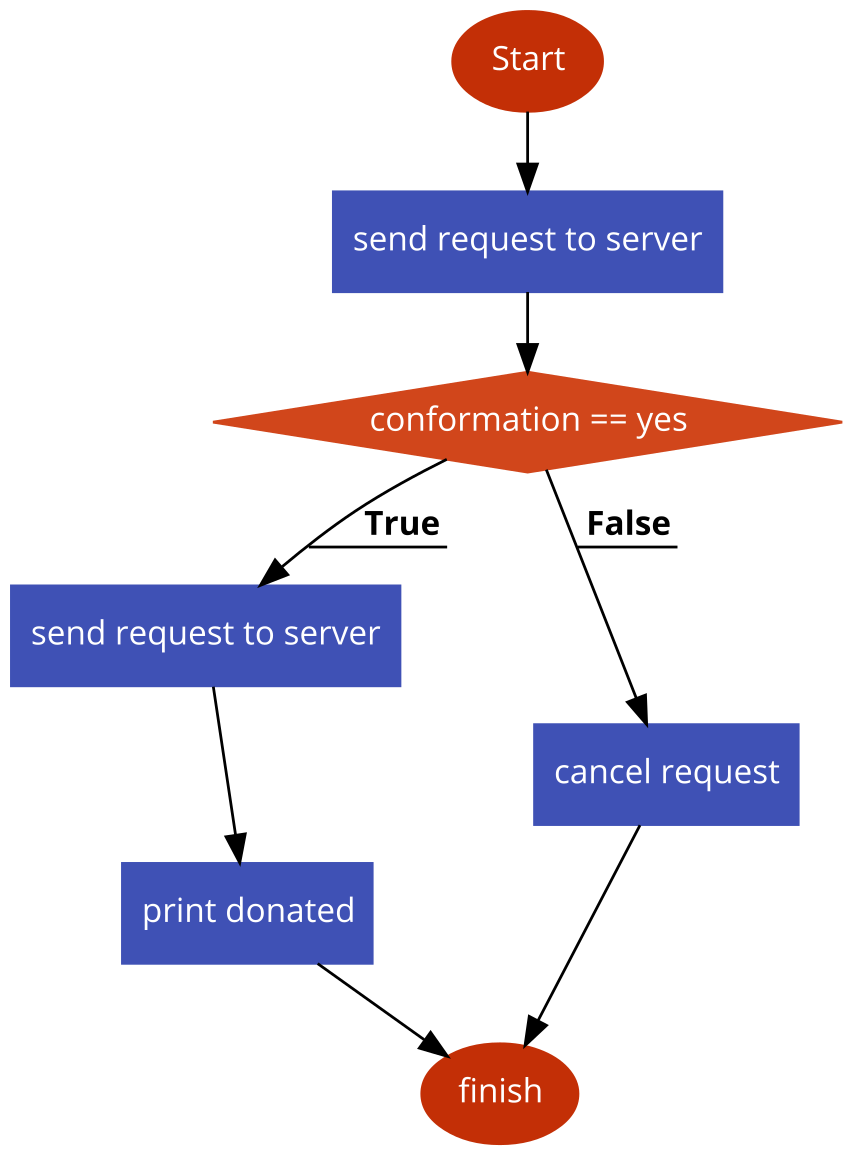
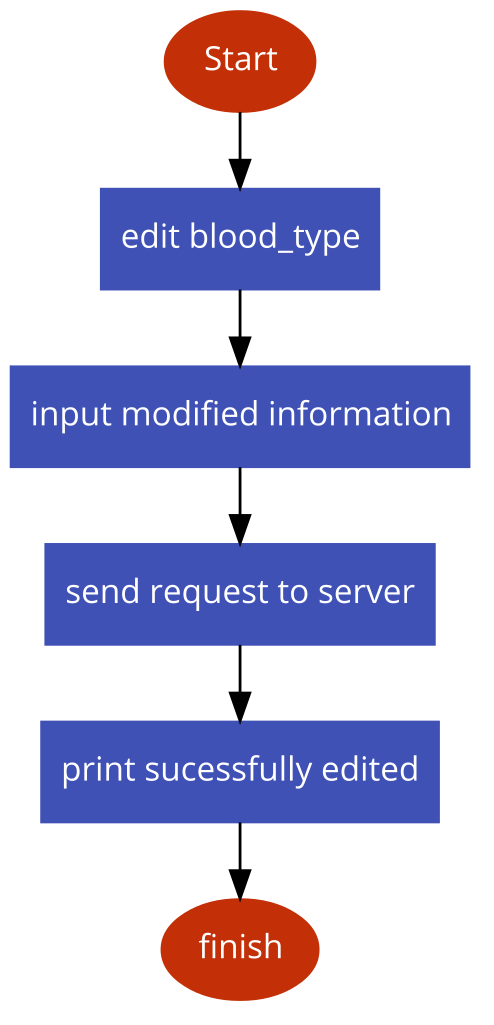
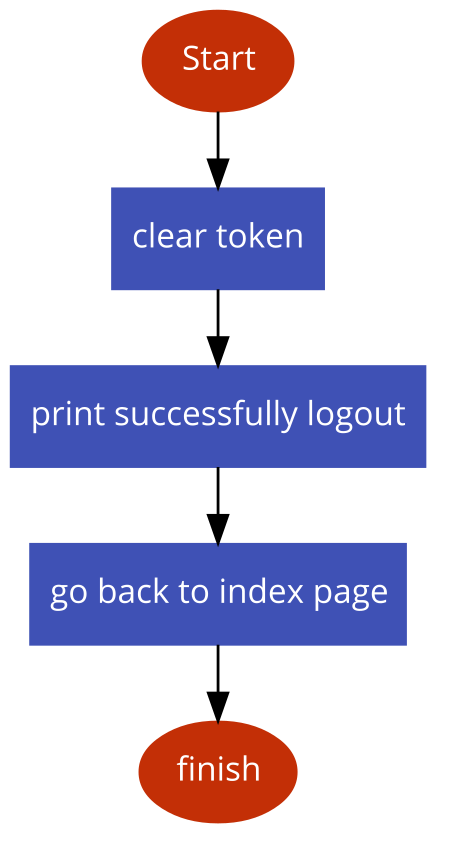
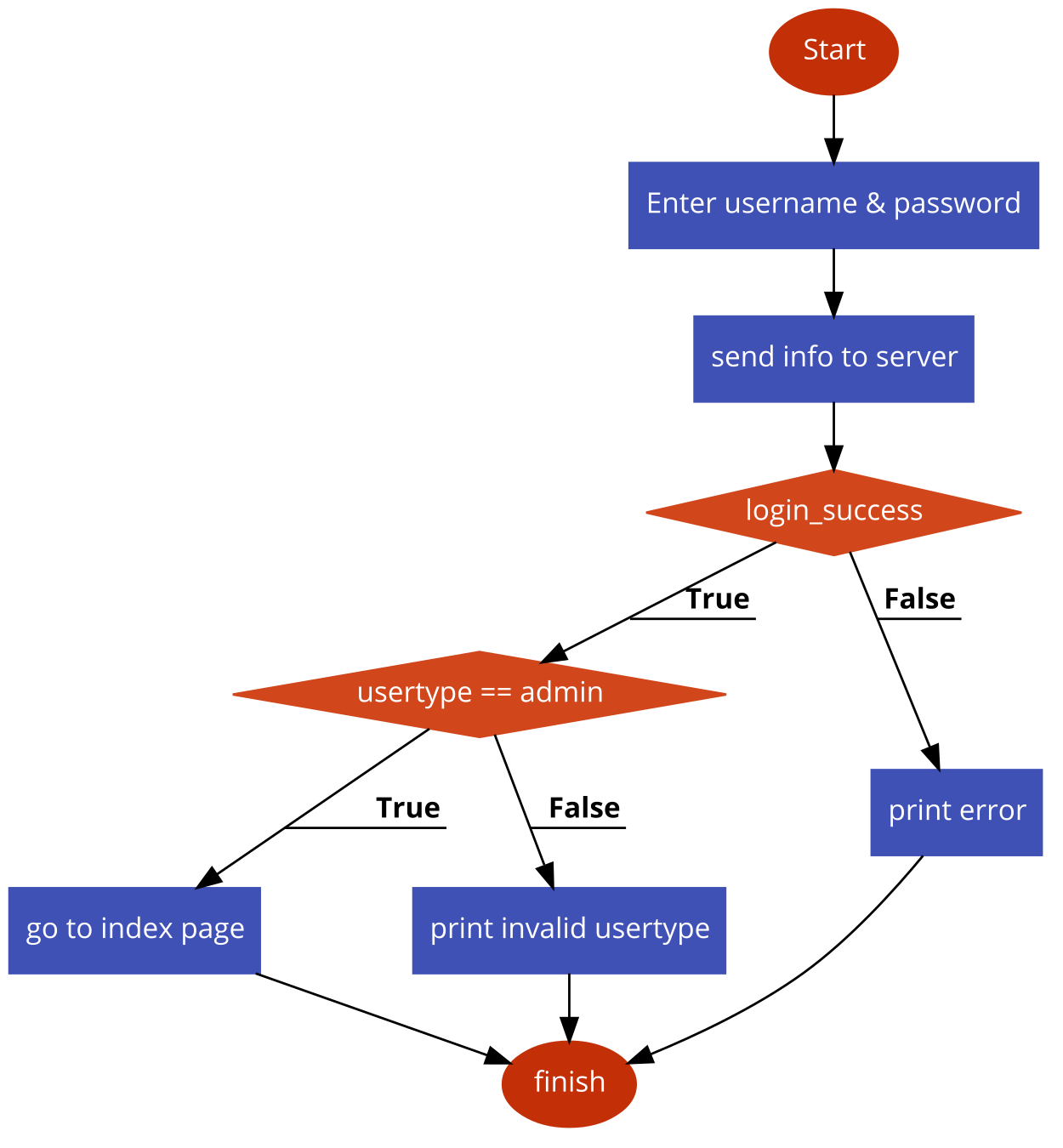
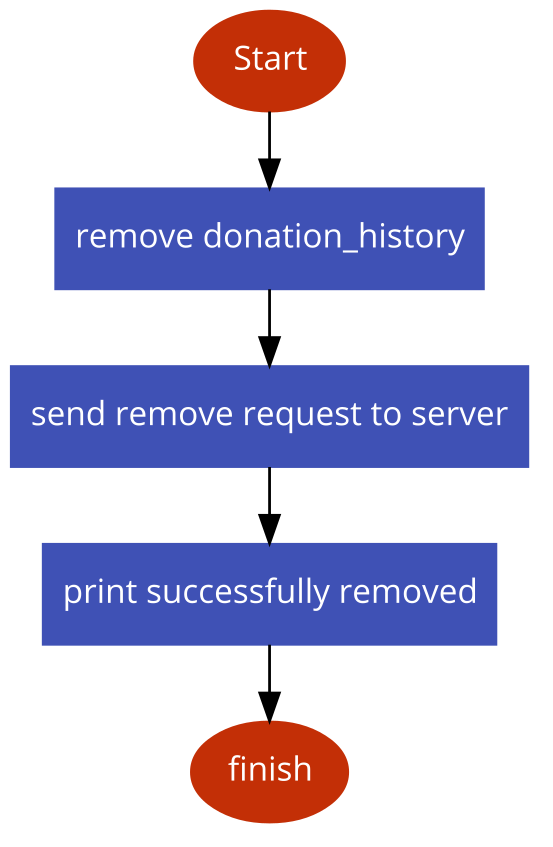
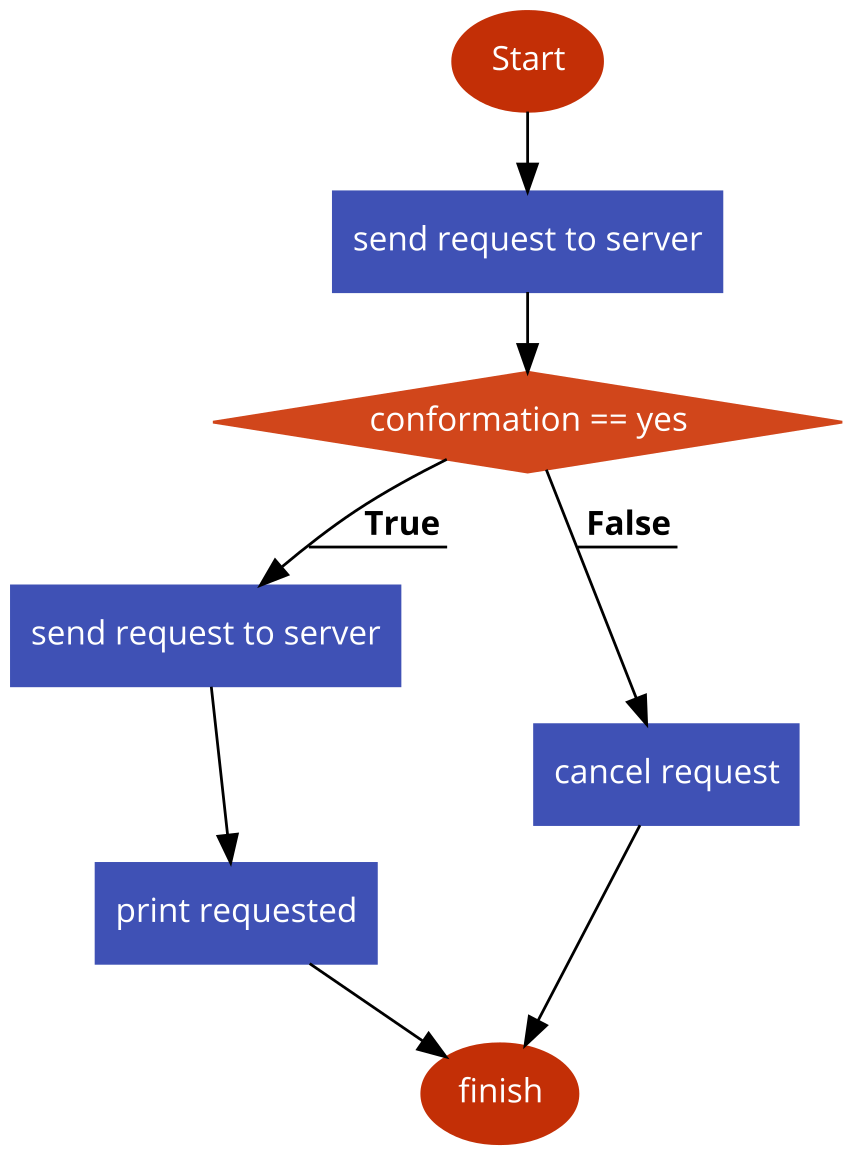
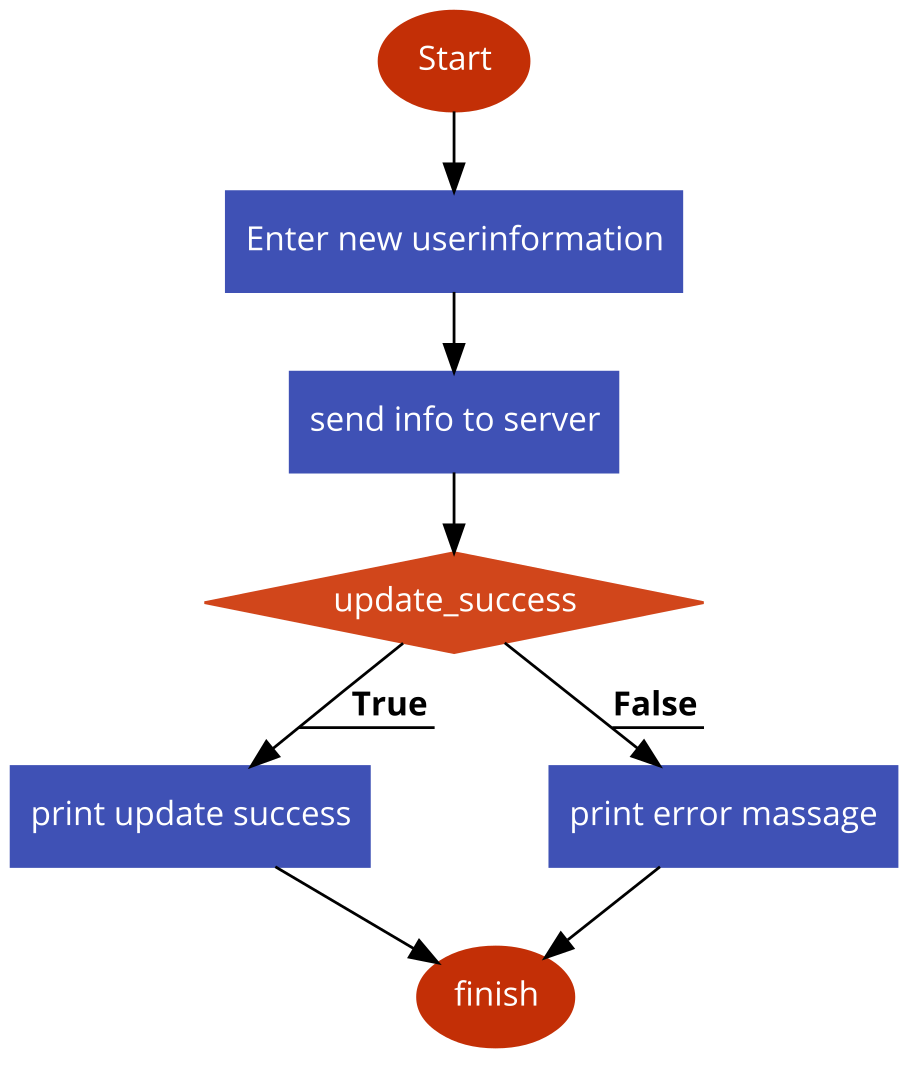
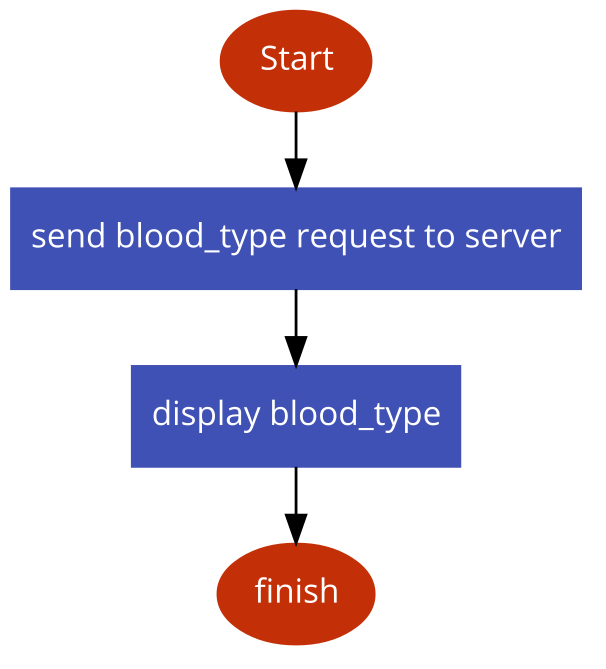
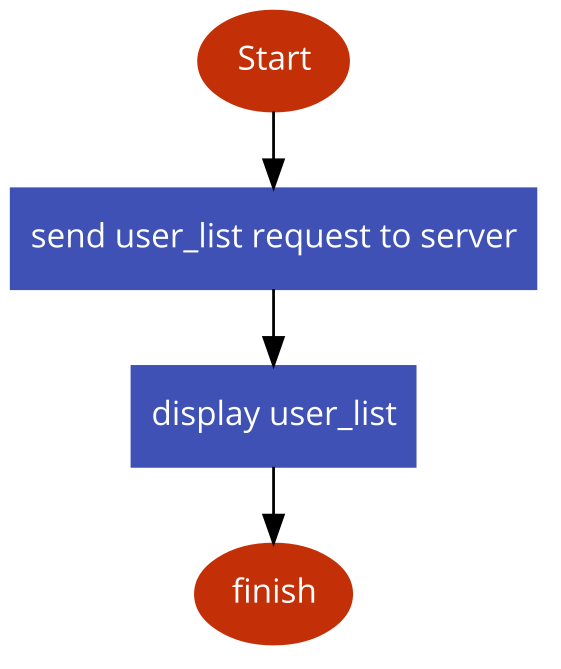
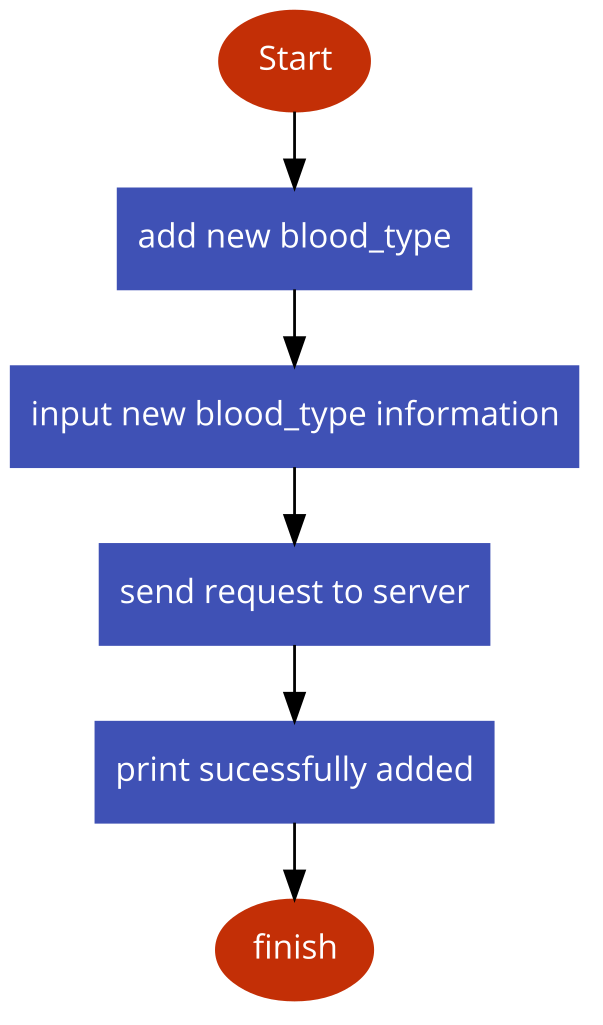
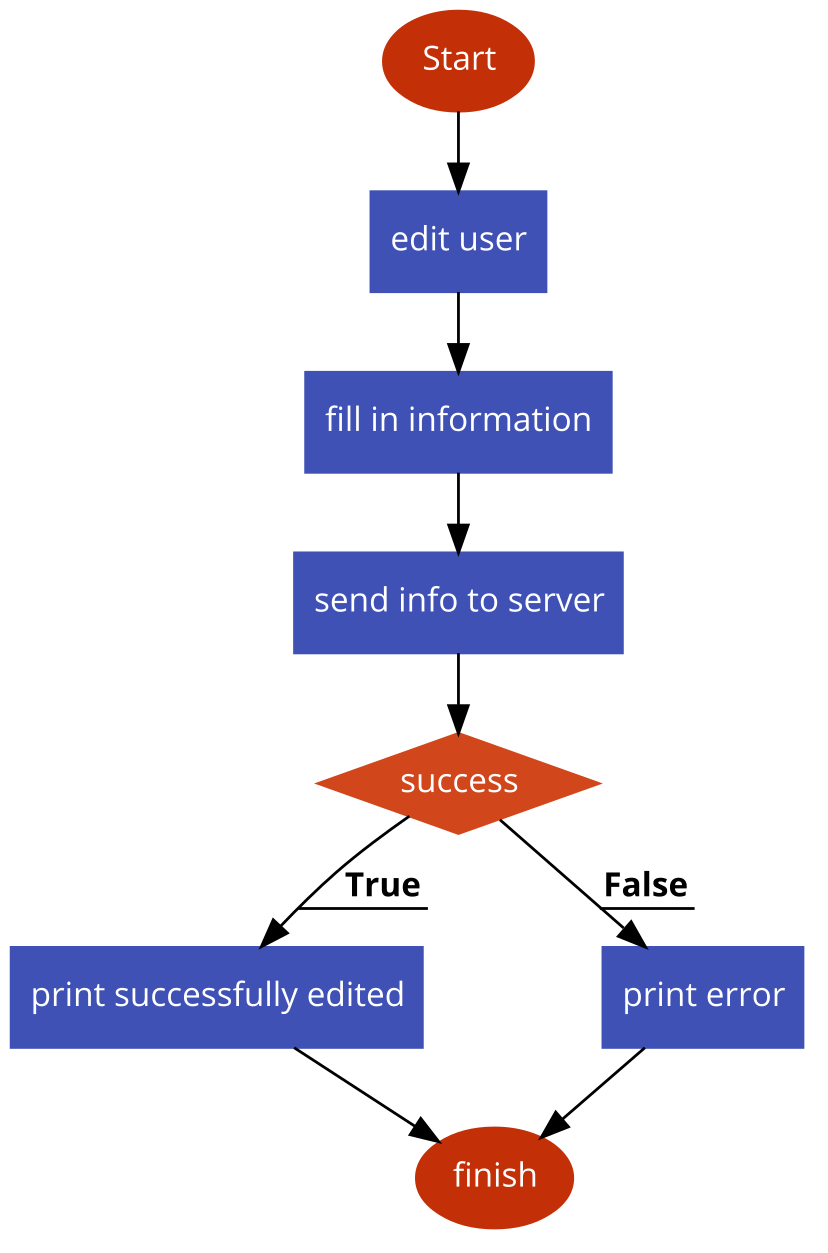


communication diagrams



State Diagram





CRC Cards

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: Administrator | ID:2 | | Type: Concrete, Domain |
| Description: An individual that can manage user information | | | Associated use case: 4 |
| Responsibilities  Check, remove and edit all the users.  Check and remove history.  Check, add and remove blood type. | | Collaborators | |

|  |
| --- |
| Attributes:  Username  Password  First Name  Last Name |
| Relationships:  Generalization: User  Aggregation:  Other Association: |

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: Regular User | ID:1 | | Type: Concrete, Domain |
| Description: An individual that can use donor or requestor’s function | | | Associated use case: 1,2,3 |
| Responsibilities  Register  Upload Information | | Collaborators | |

|  |
| --- |
| Attributes:  Blood Type  Age  Sex  Height |
| Relationships:  Generalization: User  Aggregation: Donor, Requestor  Other Association: |

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: User | ID:3 | | Type: Concrete, Domain |
| Description: An individual that can use the blood bank app | | | Associated use case: 1 |
| Responsibilities  Login  Log out | | Collaborators | |

|  |
| --- |
| Attributes:  Username  Password  First Name  Last Name |
| Relationships:  Generalization:  Aggregation: Administrator, Regular User  Other Association: |

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: Donor | ID:4 | | Type: Concrete, Domain |
| Description: An individual that can donate blood | | | Associated use case: 1,2 |
| Responsibilities  Donate  View History of donation  View requests | | Collaborators  Requestor | |

|  |
| --- |
| Attributes:  Donation History |
| Relationships:  Generalization: Regular User  Aggregation:  Other Association: |

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: Requestor | ID:5 | | Type: Concrete, Domain |
| Description: An individual that can request blood | | | Associated use case: 1,3 |
| Responsibilities  Make request  View requests history  Check request history | | Collaborators  Donor | |

|  |
| --- |
| Attributes:  Request History |
| Relationships:  Generalization: Regular User  Aggregation:  Other Association: |

**Backend Method Specification: /donor/request/list**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name: /donor/request/list | | Class Name: Donor | | | ID: 1 |
| Clients (Consumers): Donor | | | | | |
| Associated Use Cases: List Current Requests | | | | | |
| Description of Responsibilities: List active blood requests that match the donor's blood type. | | | | | |
| Argument Received: Token | | | | | |
| Type of Value Returned: array of blood\_request if success (ok: true), otherwise an error message will be set. | | | | | |
| Pre-Conditions: User is logged in and has valid token, user is a donor. | | | | | |
| Post-Conditions: None | | | | | |
| Method Name: /donor/request/list | | | | Class Name: Donor | ID: 1 |
| Contract ID: 1 | | | | Programmer: Honghao Zeng | Date Due: |
| Programming language: ECMAScript | | | | | |
| Triggers/Events: Donor requested list of requestests | | | | | |
| Arguments Received & Data Type: | Notes: | | | | |
| Token (string) | User token returned by user/auth | | | | |
| Message Sent & Argument Passed: | | | | Data Type: | Notes: |
| getUserInfo().blood\_type | | | | int | Get donors bloodtype |
| getRequests().filter() | | | | bool | Filter requests by blood type |
| Argument Returned: | | | Notes: | | |
| blood\_request[] | | | List of blood requests. | | |
| Algorithm Specification:  session = session.get(token)  if (!token) return "not logged in"  user = db.getUser(session)  if (user.type != 'DONOR') return "not donor"  blood\_type = user.blood\_type  requests = db.getRequests(blood\_type)  return requests; | | | | | |