MileStone 2 Report

Karim Alaa Mohey Eldeen	19016172
Louay Magdy Abd El halim	19016195
Mohamed Magdy Elsayed	19016479
Youhanna Yousri Fawzy	19016899
Youssef Saber Saaid	19016924
Youssef Magdy Helmy	19016937

- Functional Requirements:

1. Finding Institutions

Function	Finds institutions that has blood bags of certain types.
Description	Given the recipient blood type, finds all the institutions that has blood bags with compatible types.
Inputs	Recipient blood type.
Source	Drop down list of blood types in the frontend.
Outputs	List of institutions (name, location, distance, number of available blood bags of the compatible type) sorted by distance.
Action	- Get compatible blood types - Get the institutions. - Get user location. - Calculate distance between user and each institution. - Sort the institution by distance (nearby first).
Requirements	- DB Connection established and had institution table. - Location permission enabled on the user phone (sorting and showing distance will not occur if not provided).

2. User Requests for Blood

Function	Creates a request for a certain blood type
Description	Creates a user request for a blood type - Validating inputs in the blood request - Checking if there is a duplicate request from the same user, within the same institution for the same blood type - Creating post record in the DB - Notifying users → the next milestone

	- Deleting the post once required number of bags is collected or the expiry date has come
Inputs	- User email from the token sent on sign in - Institution ID - Required Blood Group, with RHD - Required number of bags - Creation date and expiry date
Source	Request form in the frontend, text fields are used to input the above fields
Outputs	- Creating a record for the post in the DB
Action	- Validating input data such as: • Expiry date is after creation date • Required blood type is valid - If input is valid, the DB state is checked for a duplicate post and if there is no duplicate post, a record is created in the DB. And users of compatible blood types are got to be notified - There is a background operation deleting posts from time to time if the expiry date has come
Requirements	DB Connection established and has post table
Side-effects	Adding new record to the DB state and creating new post if input is valid, and there is not another similar post

3. Institutions campaigns (events)

Function	Creating events for blood donation
Description	The institution creates an event for blood donation • <u>Fields to collect:</u> - Start and end date - Institution info.: name, location, working hours • Saving the event to the DB • Deleting post once expiry date has come
Inputs	- Start and end dates - Working hours of institutions - Name of institution and its location

Source	event form in the frontend, text fields are used to input the email and password.
Outputs	- Event record in the DB created - Event tab to access all the nearby events
Action	The application takes the required fields from the front end and creates a record for it in the dB
Requirements	Database connection established and has event table.
Side Effects	Adding some record in the DB

4. Transaction Management

Function	The institution is required to manage 3 kinds of transactions: - User – User transactions - User – Institution transactions (Donation) - Institution – User transactions (Asking for Blood)
Description	The institution is required to manage its related transactions: - The user – user transactions at the institution - The donation transactions - Asking-for-blood transactions • It should collect info. About users (registered in the app.) • It should decrement the required bags for related posts • It should keep track of its collected bags of each type
Inputs	Some form at the frontend specifying the type of transactions to manage it
Outputs	- The required bags for posts are decremented - Info. About users and institutions stored in the DB
Destination	The DB is updated with info. As mentioned, to be used in the next milestone for doing statistical analysis
Action	- Saving the transactions to the DB. Provided an ER diagram for it

- Non-Functional Requirements:
  - Security
    - To send a request to the backend, the user has to send his jwt token, that was issued by the backend, and his email is extracted from this token.
  - Collecting info. for statistical analysis

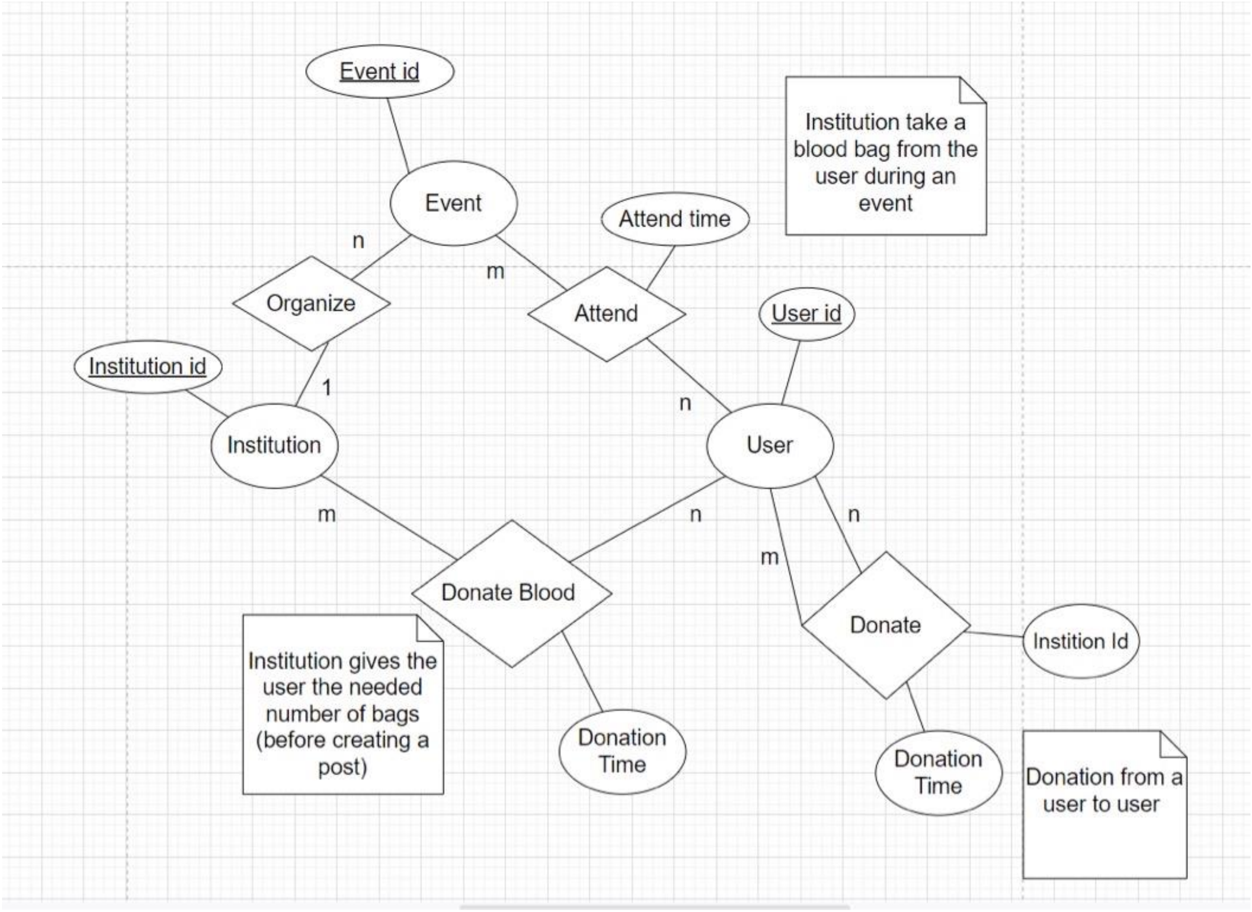
- Diagrams and UMLs
  - Schema of Post table

Post ID	User id **F**	Institution id **F**	bloodtype	Required bags	Creation time	Expiry time

  - Schema of event table

Event ID	Institution Id **F**	Start date	End date	Institution name	Institution location	Work. hours

  - ER diagram for institution, e



Institution take a blood bag from the user during an event

Institution gives the user the needed number of bags (before creating a post)

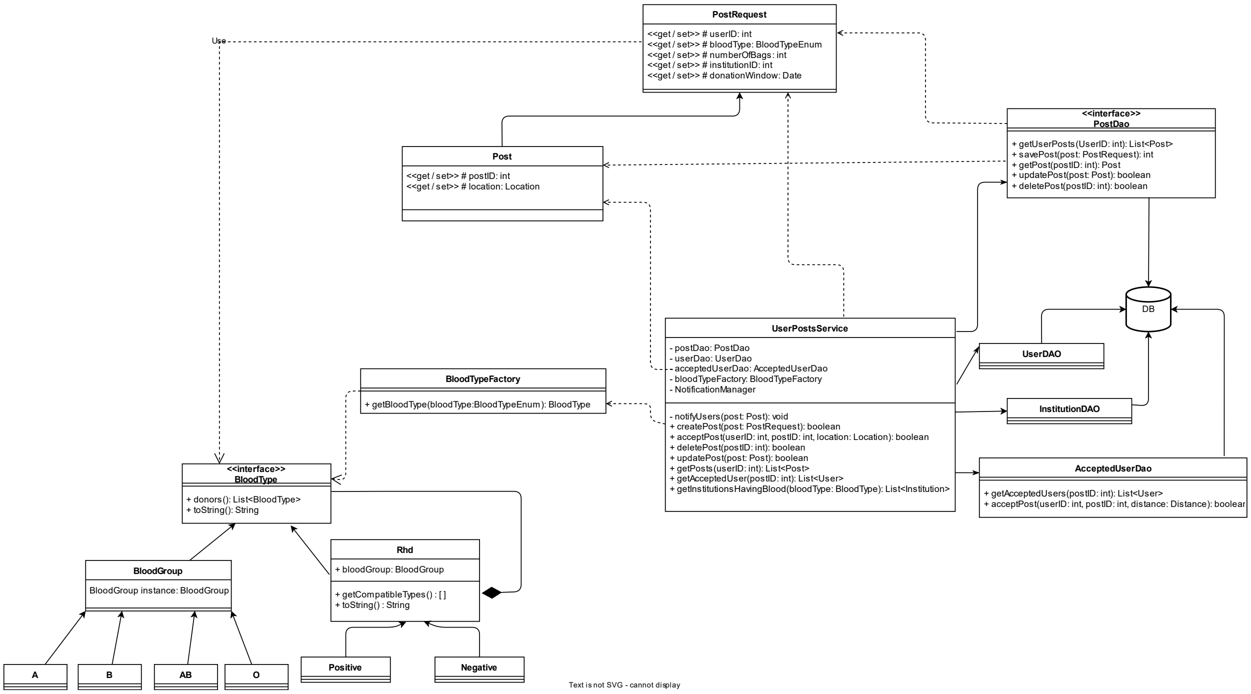
Donation from a user to user

- Updated User Request class Diagram

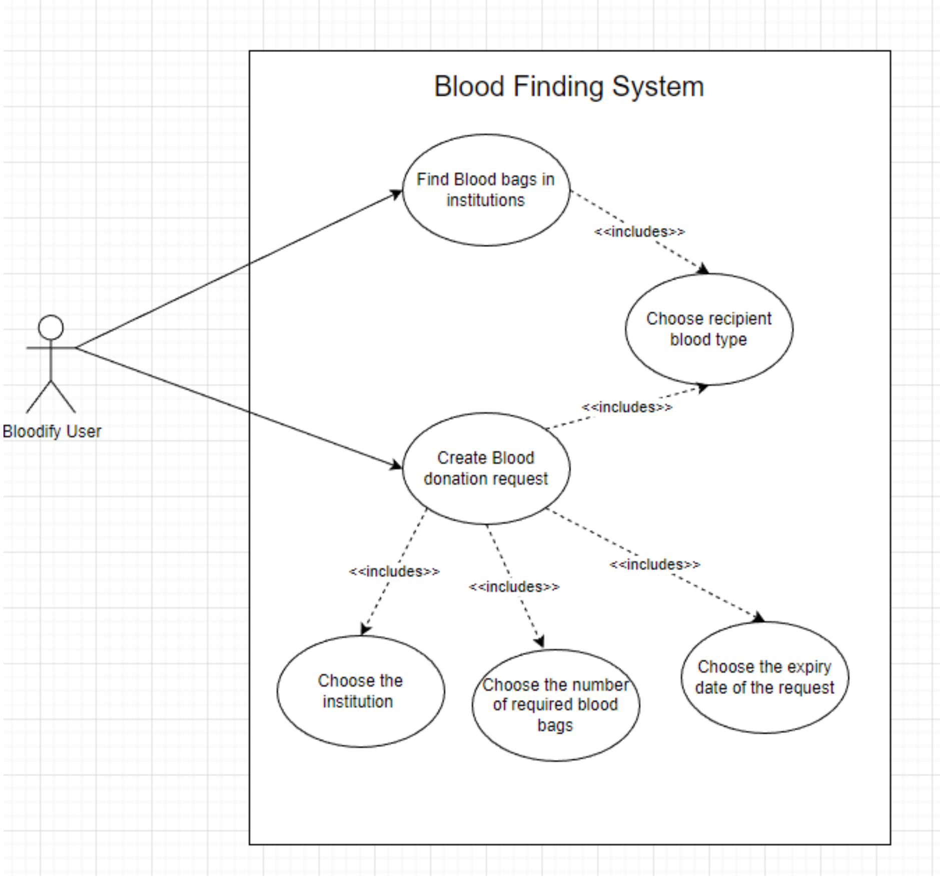
The only updated part is the blood hierarchy

- Using the RHD class as a wrapper instead of wrappee to facilitate querying the DB
- Using Blood Group, Blood type factory as Singleton class

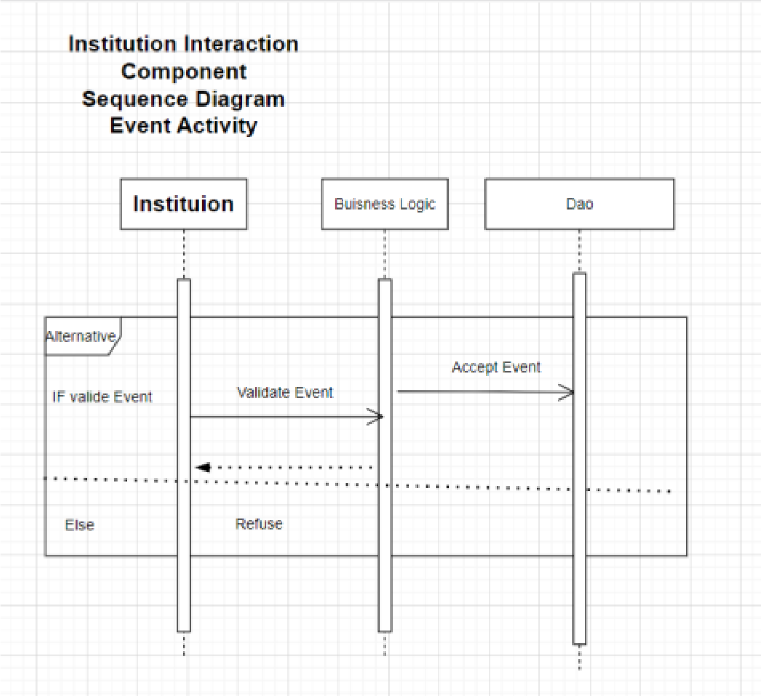
Post Component...

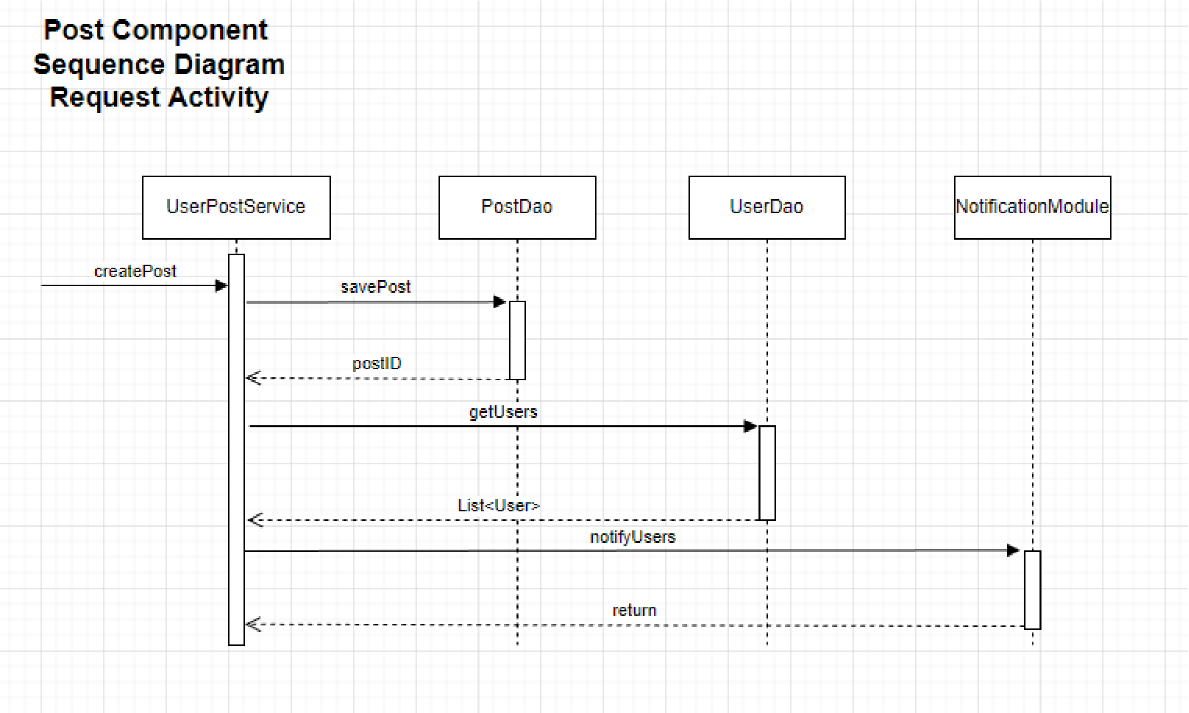

PostRequest
<<get / set>> # userID: int
<<get / set>> # bloodType: BloodTypeEnum
<<get / set>> # numberOfBags: int
<<get / set>> # institutionID: int
<<get / set>> # donationWindow: Date
Use
<<interface>>
PostDao
+ getUserPosts(UserID: int): List<Post>
+ savePost(post: PostRequest): int
+ getPost(postID: int): Post
+ updatePost(post: Post): boolean
+ deletePost(postID: int): boolean
Post
<<get / set>> # postID: int
<<get / set>> # location: Location
DB
UserPostsService
- postDao: PostDao
- userDao: UserDao
- acceptedUserDao: AcceptedUserDao
- bloodTypeFactory: BloodTypeFactory
- NotificationManager
- notifyUsers(post: Post): void
+ createPost(post: PostRequest): boolean
+ acceptPost(userID: int, postID: int, location: Location): boolean
+ deletePost(postID: int): boolean
+ updatePost(post: Post): boolean
+ getPosts(userID: int): List<Post>
+ getAcceptedUser(postID: int): List<User>
+ getInstitutionsHavingBlood(bloodType: BloodType): List<Institution>
UserDAO
InstitutionDAO
AcceptedUserDao
+ getAcceptedUsers(postID: int): List<User>
+ acceptPost(userID: int, postID: int, distance: Distance): boolear
BloodTypeFactory
+ getBloodType(bloodType:BloodTypeEnum): BloodType
<<interface>>
BloodType
+ donors(): List<BloodType>
+ toString(): String
BloodGroup
BloodGroup instance: BloodGroup
Rhd
+ bloodGroup: BloodGroup
+ getCompatibleTypes() : [ ]
+ toString() : String
A
B
AB
O
Positive
Negative
Text is not SVG - cannot display

- <u>Blood requests and Search for Use case diagrams</u>

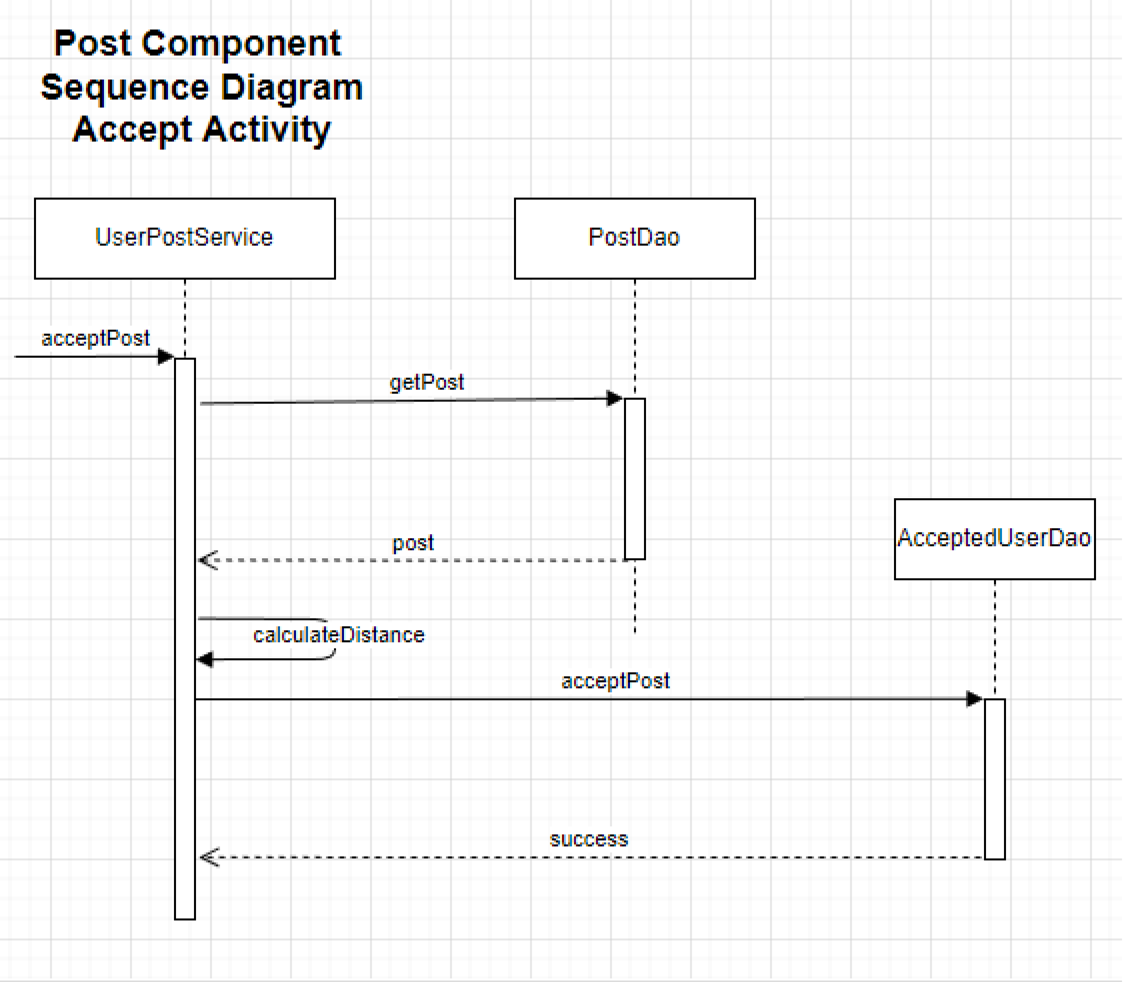

Blood Finding System
Find Blood bags in institutions
<<includes>>
Choose recipient blood type
<<includes>>
Bloodify User
Create Blood donation request
<<includes>>
<<includes>>
<<includes>>
Choose the institution
Choose the number of required blood bags
Choose the expiry date of the request

- <u>Sequence and Activity diagrams</u>



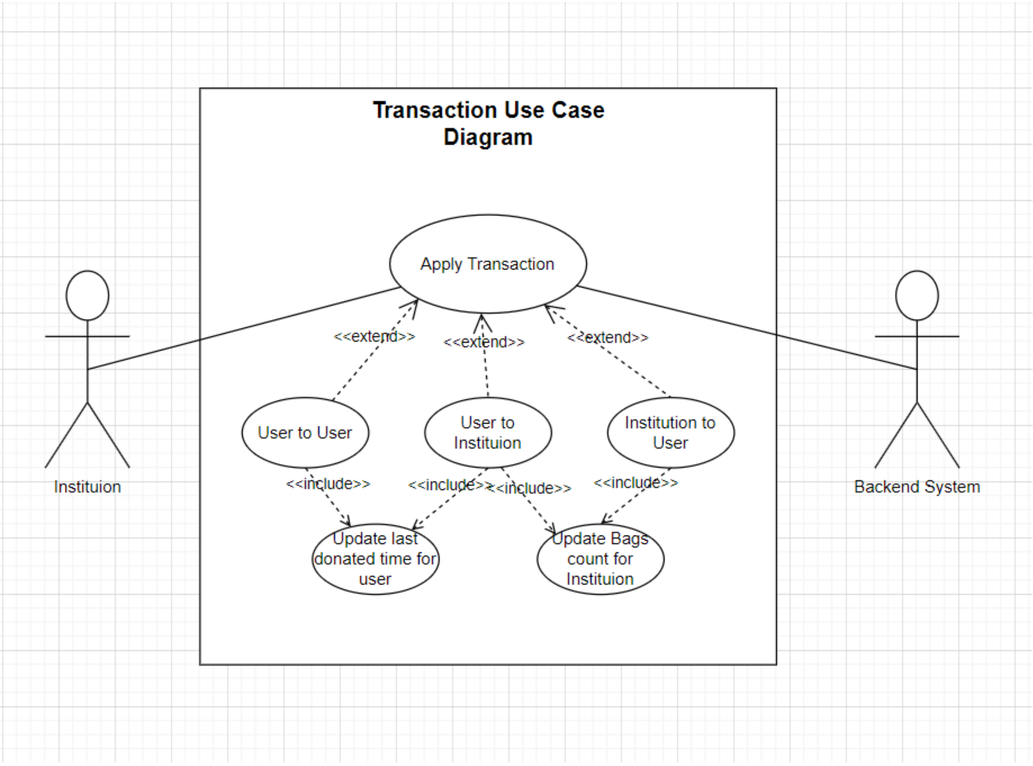


Post Component Sequence Diagram Request Activity

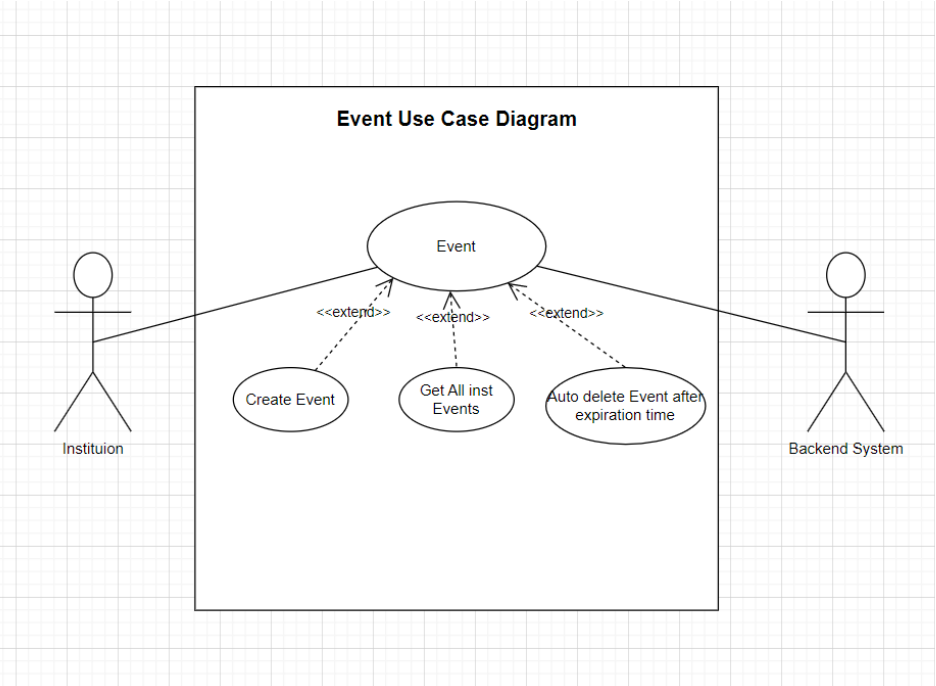


Post Component Sequence Diagram Accept Activity

- <u>Transaction use case diagram:</u>



- <u>Event use case diagram:</u>


Event Use Case Diagram
Event
<<extend>>
<<extend>>
<<extend>>
Create Event
Get All inst Events
Auto delete Event after expiration time
Instituion
Backend System