**Description of the software design and modeling**

1. Requirement gathering: The first step in designing a student community web application is to gather requirements from stakeholders, such as students, faculty, and administrators. This would involve identifying the key features and functionalities required, such as the ability to join groups, view and create posts, and comment on posts.

2. Use case modeling: Once the requirements have been identified, the next step is to create a use case diagram to model the interactions between the user and the system. This would involve identifying the main use cases, such as "Group chat messages", "Create Posts", "Create posts", "Interact with posts".

3.System architecture design: The system architecture design for a student community web application would typically involve identifying the components of the system, such as the user interface, database, and server-side logic. The system would be designed to be scalable and able to handle large volumes of users and data.

* Presentation layer: This layer is responsible for presenting the user interface to the user. It includes components such as HTML, CSS, and PHP, as well as libraries and frameworks such as Laravel. The presentation layer communicates with the other layers via APIs or web services.
* Application layer: This layer contains the business logic of the application. It includes components such as controllers, services, and models, and is responsible for processing user requests and generating responses. The application layer communicates with the data layer to retrieve or update data as needed.
* Data layer: This layer is responsible for managing the data used by the application. It includes components such as a database, data access layer, and ORM (Object-Relational Mapping) framework. The data layer communicates with the application layer to provide data as needed.
* Infrastructure layer: This layer provides the infrastructure needed to run the application. It includes components such as web servers, load balancers, and firewalls. The infrastructure layer is responsible for ensuring the availability, scalability, and security of the application.

4. Data modeling: The data required by the system would need to be modeled, such as student profiles, group information, and posts. This would involve identifying the data entities, their relationships, and the attributes they would require.

5. User interface design: The user interface design for the student community web application would need to be intuitive and user-friendly, with features such as a dashboard to easily navigate to different areas of the application. The design would also need to be responsive to support different screen sizes and devices.

6. Implementation: The student community web application would need to be implemented using a suitable programming language, we are going to use PHP in the Laravel framework. The implementation would need to follow best practices and standards to ensure the system is reliable, secure, and maintainable.

7. Testing and debugging: The system would need to be tested thoroughly to ensure that it meets the requirements and is free of bugs and errors. Automated testing tools and manual testing would be used to achieve this.

8. Deployment: Once testing is complete, the student community web application would need to be deployed to a production environment. This would involve configuring the application server, database, and any other required components.

Below we have built a use case diagram which is a visual representation of the behavior of the system in user's perspective. Just like this diagram, we can build many others to give an idea of how the system works.

