Sarthak Pandit(C24CS1001)

Siddharth Ranka(C24CS1002)

Vansh Agarwal(B23CS1077)

Seema(B20CS064)

Notes on what we have implemented through the cards.

1. Stakeholders:
   In this project, we identified all important stakeholders. These include the Admin, who will manage users, who will log in and access services, and the Security Manager, who ensures data safety.
   The team (which is also a stakeholder) worked together to prepare diagrams, documents, and the system design. Internally, we divided responsibilities within the team — like who will make the class diagram, component diagram, SRS, etc.
   We represented the responsibilities clearly in the diagrams — for example, Admin manages users and Security Manager interacts with the database for authentication.
   However, we didn't formally set up meetings with all stakeholders (like getting official feedback from a professor during work), so there's partial stakeholder involvement.

2. Opportunity:
   We clearly understood the opportunity or problem — the need for a secure user management system where the Admin can log in, manage users, and the whole system has proper authentication and security.
   This opportunity was identified from the assignment itself, and we (the team) discussed it in detail.
   We proposed a solution — a system with components like Admin, Authentication Manager, Security Manager, and Database.
   The system's value is clear: it automates user management, adds security, and reduces manual errors.
   However, since this is an academic project, we didn't calculate actual business value (like cost saved or time saved in real life).

3. Software System:

We built a complete architectural overview using the Component Diagram, which shows how different components like Admin, Authentication Manager, and Database interact.
We also created a Class Diagram, showing the internal design — classes like Admin, User, Security Manager, Database, and their attributes, methods, and relationships.
We prepared a Use Case Diagram to show what actions users (Admin/User) can perform, and we prepared an Activity Diagram to show the login flow.
This makes the system architecture, flow, and features very clear.

4. Work:

The work for this project was planned by the team. Each member took different responsibilities — some worked on diagrams, others on the SRS document, and we combined efforts to finalize everything.
The tasks were identified and prioritized — diagrams first, then SRS, then final adjustments.
We kept reviewing progress internally and fixed mistakes when needed (like updating relationships in class diagram).
This shows that the work was prepared, started, and kept under control.

5. Way of Working:

The team followed a basic way of working — we discussed the approach in the beginning (what diagrams are needed, who will do what).
We used simple tools like drawing tools (Draw.io) to make diagrams.
We also communicated regularly through group chats to share updates and solve doubts.
This way of working was agreed within the team, though it was not formally documented.

6. Team:

The Team was formed on assigned roles and worked together.
Responsibilities were divided effectively on their interest and skill.
Regular communication helped in avoiding errors and ensuring timely progress.

7. Requirements:

Our team created a detailed Software Requirement Specification (SRS) document to properly explain everything the system needs to do. In the SRS, we listed functional requirements, which describe the important actions the system must perform — like allowing users to log in and giving the admin the ability to manage users by adding, editing, or deleting their details. We also included non-functional requirements, which describe the overall quality and performance expectations — for example, the system should be secure, protecting user passwords, and should respond quickly to user actions. To ensure our work was consistent, we made sure that all diagrams (like class diagram, component diagram, and use case diagram) matched these requirements, so that the design and the requirements were always aligned. This made it easier for the whole team to stay clear on what the system should do and how it should work.