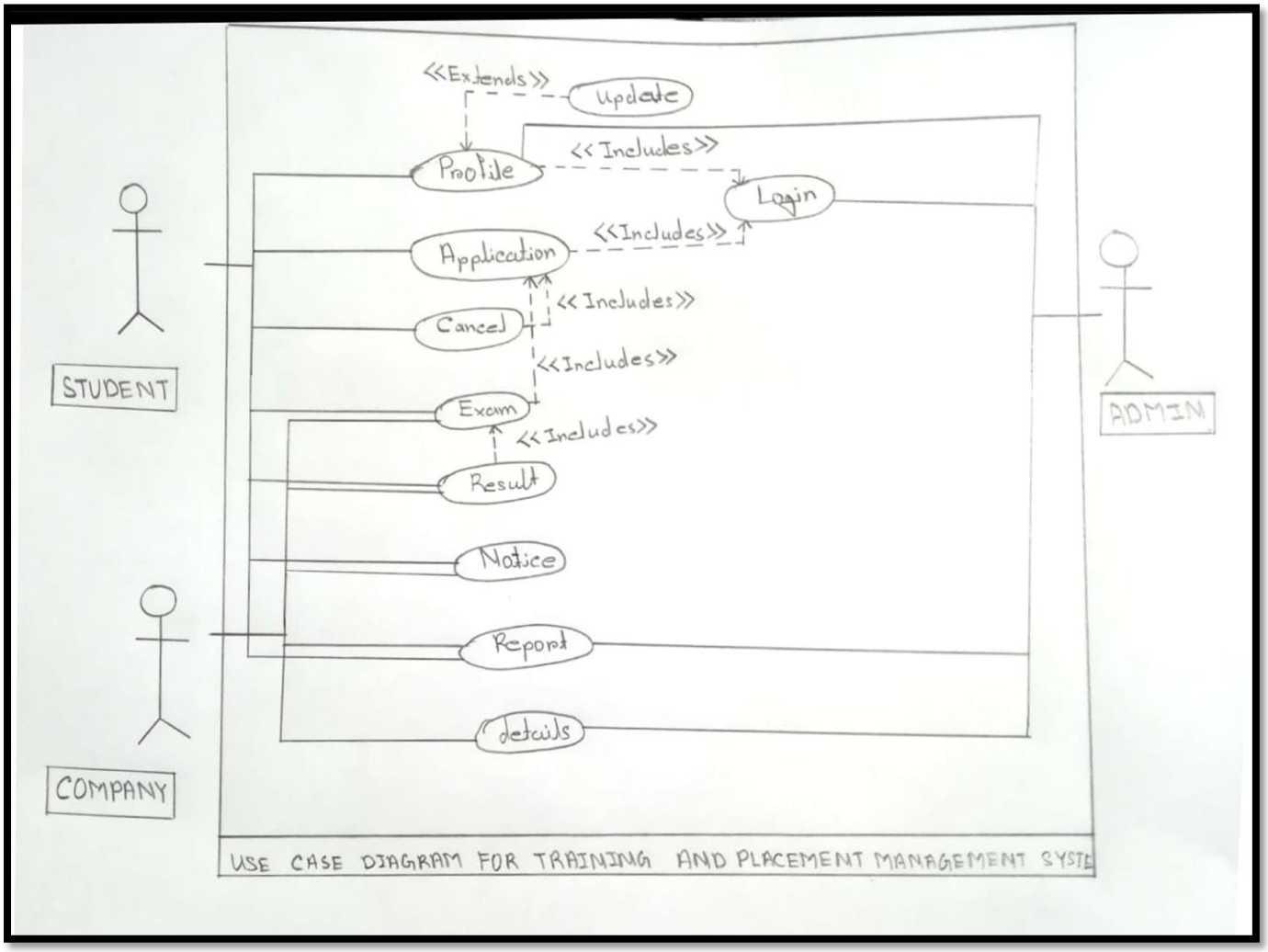
Training and Placement System Use Case Diagram



Discussion

Here, we learnt how to draw a Use Case Diagram marking the functions that were used. We used <<includes>> that is an implicit function and <<extends>> that is an explicit function.

Here we are making a Use Case Diagram for our project Training and placement system which has three actors that are Student, Company and Admin and we have shown how the process flows.

In our project students provide their profile details. If any issues come in their profile <<extends>> so they can update their profile details for filling their profile details. First of all<<includes>> they need to login in our system.

For applying any job student needs to fill application form <<includes>> login. For cancel <<includes>> application. Application <<includes>> exam when date is schedule by company. Exam <<includes>> result when company release their selected candidate. Companies can update their information in notice which students can access. Students and company both can report their respective issues in report section which maintain by admin. Companies will see each candidates details which will be provided by admin

Questionnaires

Q. Explain the significance of <<include>> and <> relations in use case diagram.

Include-

In UML modeling, an include relationship is a relationship in which one use case (the base use case) includes the functionality of another use case (the inclusion use case). The include relationship supports the reuse of functionality in a use-case model.

● When the base use case is executed, the included use case is executed every time.

● The base use case required the completion of the included use case in order to be completed.

● The result of the behavior that the inclusion use case specifies, not the behavior itself, is important to the base use case.

Extend –

In UML modeling, you can use an extended relationship to specify that one use case (extension) extends the behavior of another use case (base). This type of relationship reveals details about a system or application that are typically hidden in a use case. The extended relationship specifies that the incorporation of the extension use case is dependent on what happens when the base use case executes. The extension use case owns the extended relationship. You can specify several extended relationships for a single base use case.

● A part of a use case that is optional system behavior.

● A subflow is executed only under certain conditions.

● A set of behavior segments that may be inserted in a base use case.

● When the base use case is executed, the extended use case is executed only sometimes.

● The extended use case will happen only when certain criteria are met.