

**Q1:** We can use the SDLC (Software Development Life Cycle) methodology for the given ~~been~~ project, because according to the project's requirements, SDLC seems to be the most suitable choice.

SDLC is suitable here because —

- ① It ensures data confidentiality
- ② The time and budget is limited
- ③ The project requirements are pre-defined, meaning a proper SRS document is present.

Since it is also a large-scale project eliminating the chances of rapid changes, SDLC is the best development method to be used.

**Q2:** An organization might have developed a technically perfect system but there is a lot that could have gone wrong here, discouraging the employees to use it. Key reasons can be —

- ① No proper reinforcement or coordination.
- ② Not understanding the needs of the employees.
- ③ Hierarchical imbalance (Higher ups not considering the challenges of the worker class)



Q3: An organization should consider replacing a system instead of maintaining it when maintaining costs become greater than building a completely new software from the ground up. The organization might also consider replacing legacy systems with the new system to incorporate modern technology to their software.

Q4: According to the given scenario there seems to be the SDLC phase 'Planning' that was likely ignored. Since the project was cancelled after being two years into development, apparently the developers or the stakeholders undermined the time required.

Q5: SDLC.

Two features of SDLC —

- ① Pre-defined SRS document
- ② Suitable for long-term projects  
(e.g. Govt. related projects)





19-01-2026

## Chapter II

### Modeling Systems Graphically

#### #Context-Level Data Flow Diagram (DFD)

↑  
first of type

Mid er por : Chapter 7

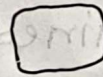
Level 0, Level 1, ...

{ Other dfd inherits  
from context-level dfd }

CL DFD → only contains 1 process

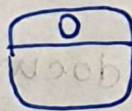
↳ needs 3 symbols to draw

① Process → which transform the data

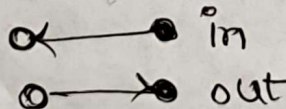


'Round Rect'

→ named after the 'System'



② Data flow (in/out)



③ Entity



(Process গ্রাহক বাচ্ থেকে input নিয়ে বা output নিয়ে)  
(Source/Destination of Data)  
→ System এর বাইরের কিছু (Customer/People)



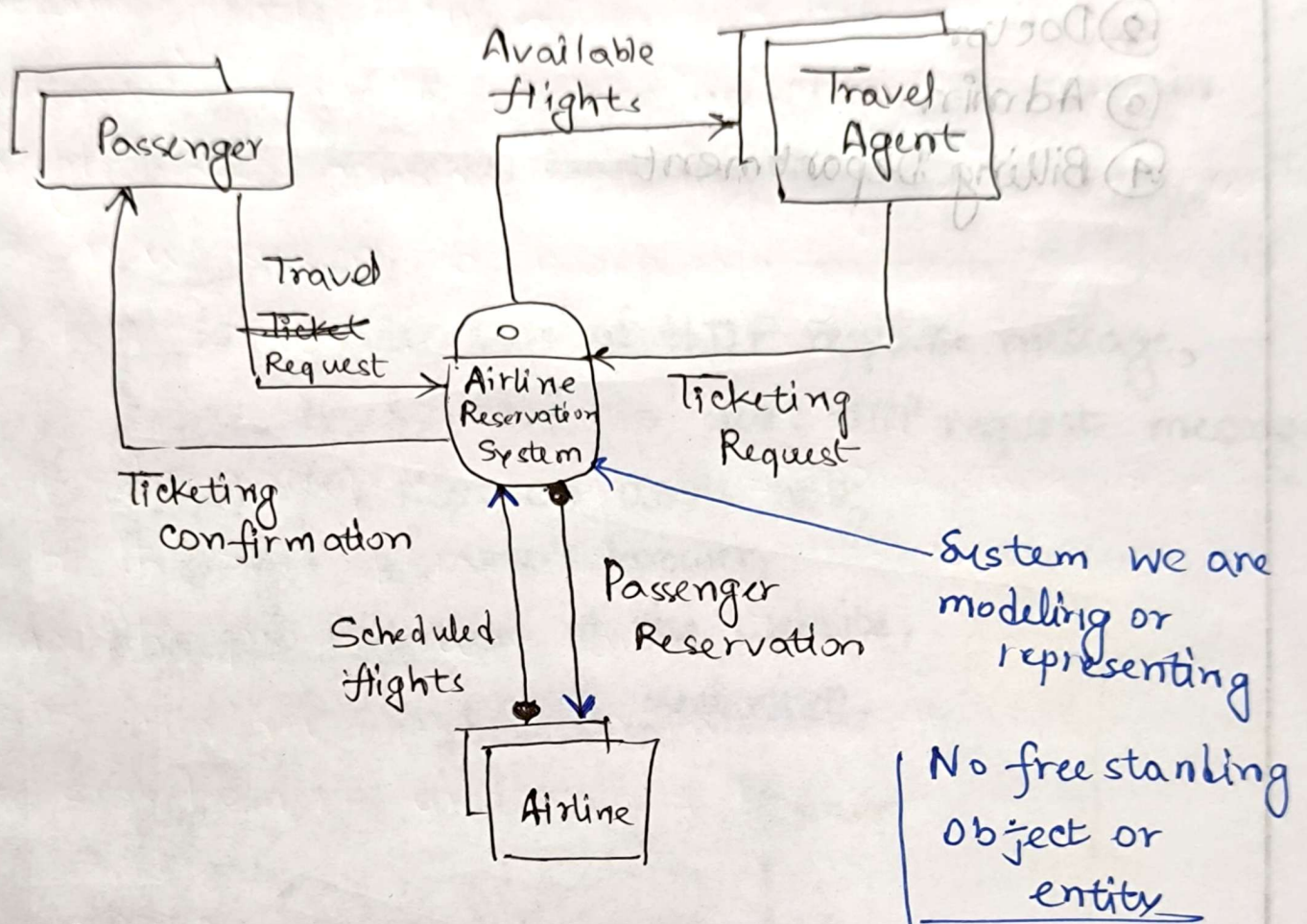


fig: Context-level DFD of Airline Reservation System.

Two entities cannot be connected together

Entities can have multiple in/outs

Some entities might not have an 'in' or 'out' it's okay.

But every process should have at least 1 in and 1 out.

Data flow (in/out) should be named.



## 1. Entities

- ① Patient
- ② Doctor
- ③ Admin
- ④ Billing Department