TICKETING PORTAL

PRESENTED BY:

- Teja Swaroop Reddy(TL)
- Santhoshini.S
- Tejas.G
- Sridevi.M

TECHNOLOGIES USED:

- .NET Core Web App (MVC)
- Entity Framework
- .NET Core Web API
- C#
- HTML
- CSS

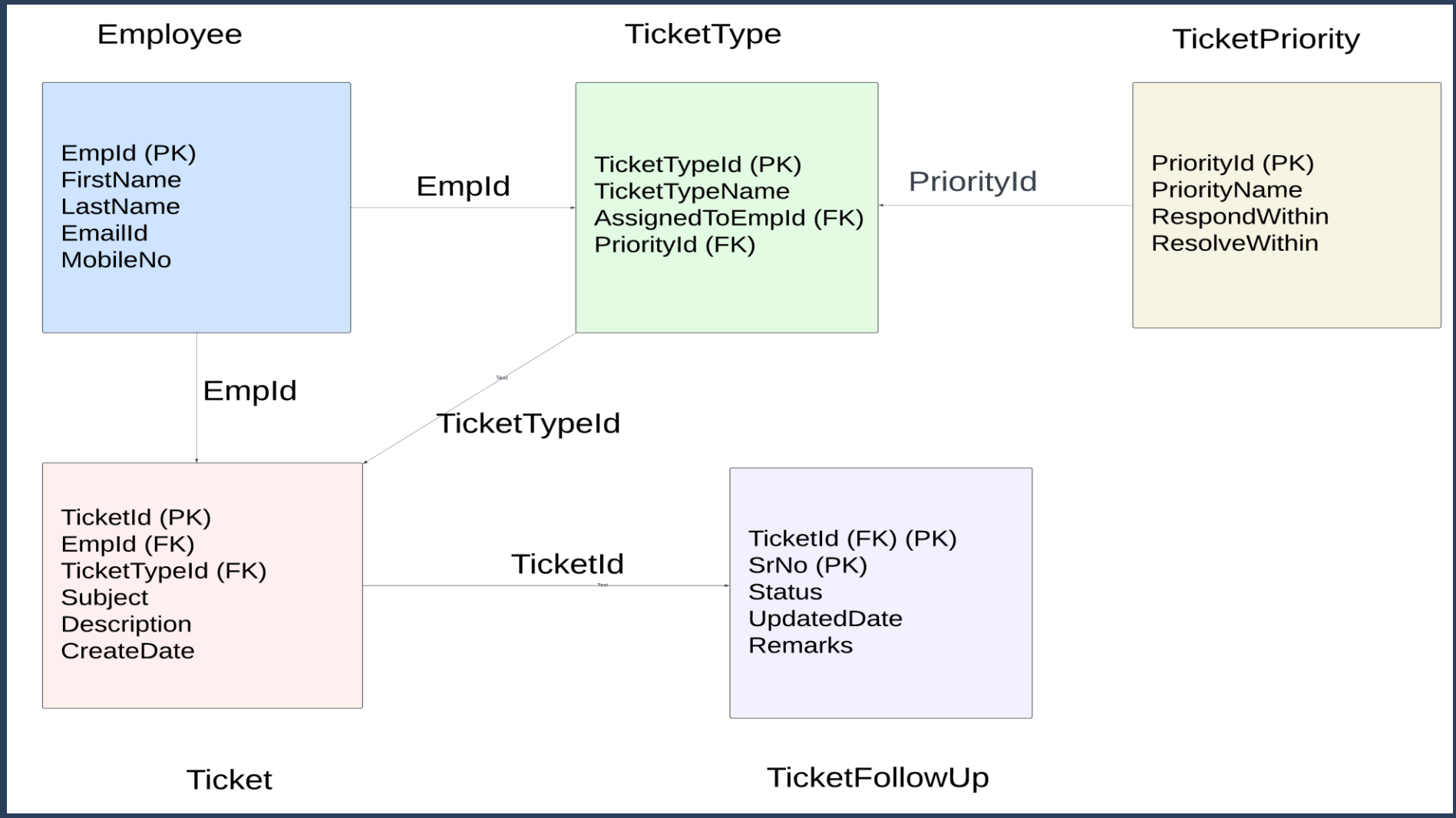
WHAT IS TICKET?

In smaller companies, whenever employees seek IT support, they can simply walk up to their internal IT team and get their issues fixed. But as organizations grow in size, managing employee issues and internal IT service requests tends to become cumbersome. Emails work when companies are small, but their speed and simplicity don't cope well with the large number of requests that are being raised in a big organization.

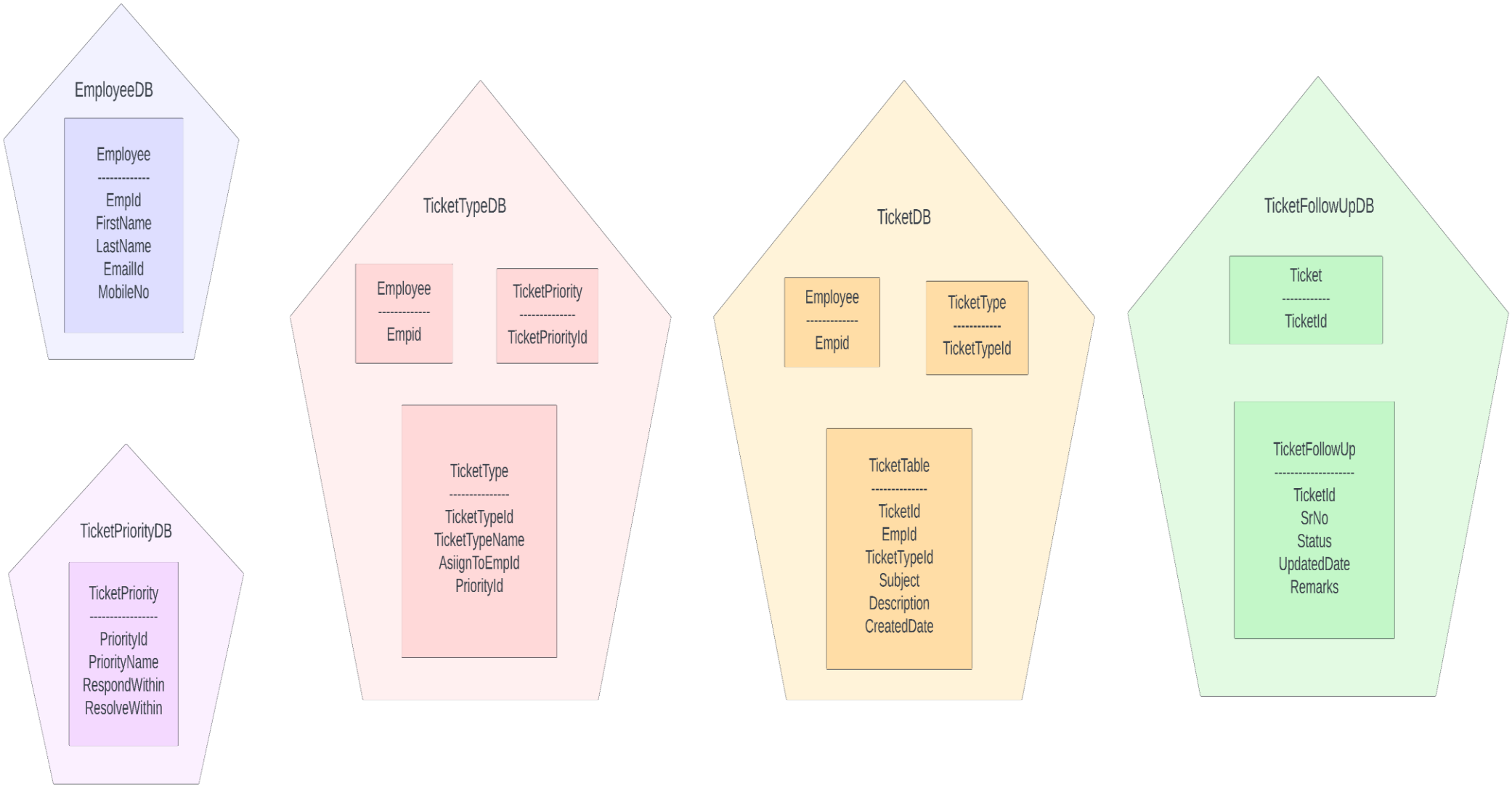
INTRODUCTION:

A ticket, in a helpdesk ticketing system, acts as a documentation of a particular problem, its current status, and other associated information. Raised by the end users of an organization whenever they encounter an event that interrupts their workflow, these tickets are routed to a ticketing software where they are categorized, prioritized, and assigned to different agents according to the organizational norms. The agents then analyze these tickets and suggest appropriate fixes or workarounds and resolve the issue. As a central repository of all these tickets, an IT Ticketing Software helps in providing the context of the issue history and its resolution.

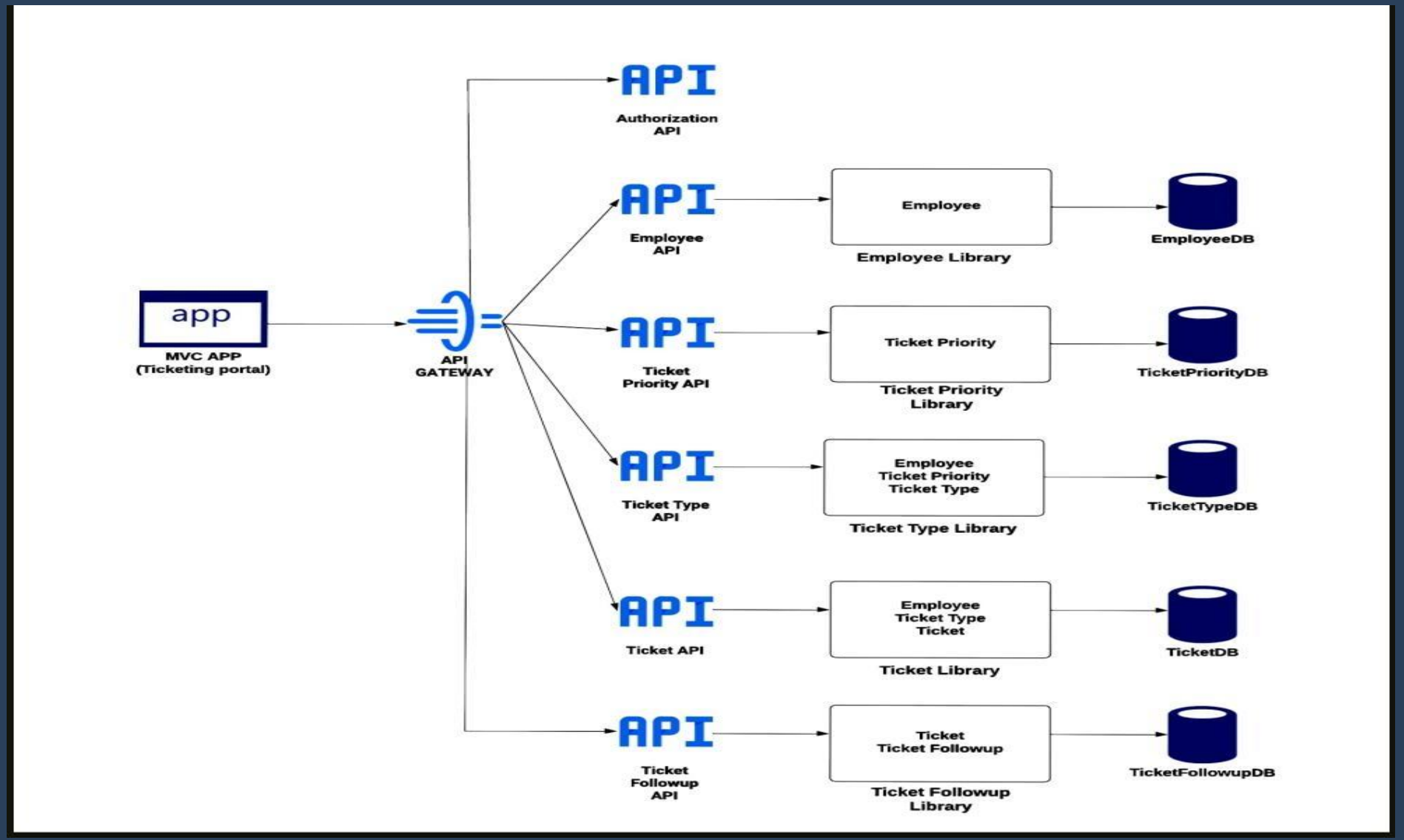
ER DIAGRAM:


Employee
TicketType
TicketPriority
EmpId (PK)
FirstName
LastName
EmailId
MobileNo
EmpId
TicketTypeId (PK)
TicketTypeName
AssignedToEmpId (FK)
PriorityId (FK)
PriorityId
PriorityId (PK)
PriorityName
RespondWithin
ResolveWithin
EmpId
TicketTypeId
TicketId (PK)
EmpId (FK)
TicketTypeId (FK)
Subject
Description
CreateDate
TicketId
TicketId (FK) (PK)
SrNo (PK)
Status
UpdatedDate
Remarks
Ticket
TicketFollowUp

DATABASE CONNECTIONS:



TICKETING PORTAL MICROSERVICES:



MODULES:

- Employee
- Ticket Priority
- Ticket Type
- Ticket
- Ticket Follow Up

EMPLOYEE:

- Purpose: Handles HTTP requests and responses for Employee Operations
- Index (GET): Returns all Employees
- Details (GET): Fetches employee details by ID
- Create (POST): Adds a new employee and communicates with external services
- Edit (PUT): Updates employee details
- Delete (DELETE): Deletes an employee and communicates with external services

EMPLOYEE:

- Integration with External APIs
- Authentication: Requests token from the Auth API
- Communication with TicketType and Ticket APIs
- Handles operations related to employee associations with Tickets
- Flow of Data:

User → API Gateway → EmployeeController(HTTP Request) → Repository Methods → EF Context → Database

TICKET PRIORITY:

- Purpose: Handles HTTP requests and responses for TicketPriority Operations
- Index (GET): Returns all Ticket Priorities
- Details (GET): Fetches Ticket Priority details by Priority Id
- Create (POST): Adds a new Ticket Priority and communicates with external services
- Edit (PUT): Updates employee details
- Delete (DELETE): Deletes an employee and communicates with external services

TICKET PRIORITY:

- Integration with External APIs
- Authentication: Requests token from the Auth API
- Communication with TicketType API
- Handles operations related to TicketPriority associations with Tickets
- Flow of Data:

  User → API Gateway → TicketTypeController (HTTP Request) → Repository Methods → EF Context → Database

TICKET TYPE:

- Purpose: Handles HTTP requests and responses for TicketType Operations
- Index (GET): Returns all Ticket Types
- Details (GET): Fetches Ticket Type details by ticket type Id
- Create (POST): Adds a new Ticket Type and communicates with external services
- Edit (PUT): Updates Ticket Types details
- Delete (DELETE): Deletes an employee and communicates with external services

TICKET TYPE:

- Integration with External APIs
- Authentication: Requests token from the Auth API
- Communication with Ticket API
- Handles operations related to TicketType associations with Tickets
- Flow of Data:
  User → API Gateway → TicketTypeController (HTTP Request) → Repository Methods → EF Context → Database

TICKET:

- Purpose: Handles HTTP requests and responses for Ticket Operations
- Index (GET): Returns all tickets
- Details (GET): Fetches ticket details by Ticket Id
- Create (POST): Adds new ticket and communicates with external services
- Edit (PUT): Updates Ticket details
- Delete (DELETE):Deletes a ticket and communicates with external services

TICKET:

- Integration with External APIs
- Authentication: Requests token from the Auth API
- Communication with TicketFollowUp API
- Handles operations related to Ticket associations with Tickets
- Flow of Data:

  User → API Gateway → TicketController(HTTP Request) → Repository Methods → EF Context → Database

TICKET FOLLOWUP:

- Purpose: Handles HTTP requests and responses for TicketFollowUp Operations
- Index (GET): Return all Ticket Follow ups
- Details (GET): Fetches Ticket follow up details by ticket id and serial no
- Create (POST): Adds new ticket follow up and communicates with external services
- Edit (PUT): Updates Ticket Follow up details
- Delete (DELETE):Deletes a ticket follow up and communicates with external services

TICKET FOLLOWUP:

- Integration with External APIs
- Authentication: Requests token from the Auth API
- Handles operations related to Ticket associations with Tickets
- Flow of Data:
  User → API Gateway → TicketController(HTTP Request) → Repository Methods → EF Context → Database

TEAM COLLABORATION:

- Teja Swaroop Reddy(TL)
  Ticket Followup, Authorization API, Helper class
- Santhoshini.S
  Employee, TicketPriority, ApiGateway
- Tejas.G
  Ticket Type, Frontend
- Sridevi.M
  Ticket, Frontend

CONCLUSION:

- We were successfully able to build the Ticketing Portal project, providing the interface for the employees to create and manage the tickets.
- The integration of key features such as User Authentication and API Validations has significantly improved the efficiency.

THANK YOU

Finally, we would like to thank Mr. S.N.Rao sir for your valuable guidance .