1. Problem ----> Abstraction ----> One Algo (can be applied to traverse, build, react frontend)
2. Note specific details of the App:
    - UI / UX (inputs, cards, tables, etc)
    - Train, Flight, Bus
3. Train stop A ----> Train stop B
4. We train and use AI to study the API documentation to and understand what columns we need.
5. Input API doc ---> Output JSON
6. API process called Airline Access Library ------> makes the API call to the respective API
    - Algo is used to build and parse the data
    - JSON is used to store the data.
7. Study the API doc manually ----> train the AI giving it a sample JSON -----> AI now understands the nature and columns involved and through the prompts we trained it with.
8. Now this AI training can be applied to all the different APIs to generate the **JSON PARAMETERS (query parameters) AND PAYLOAD FOR THE URL REQUEST TO THE API** whether it is for trains, planes, or busses.
9. Now we have the database configuration -----> Node Table and Edges Table
10. Now when we want to generate the query to get the respective route and nodes involved and display the route details to the user:
    - AI generates and pre-prepares all the graphql queries based on the fixed columns from the table and keep them ready on the frontend.
    - When we need a route, the requirements are matched to the query based on the query name and the request is sent to the backend, thus automating the process.
    - To prevent sql injecton the columns to be extracted are double checked before querying.
11. Unified graph microservice ----> uses AAL to get the data from the API and then performs the graph algorithms to traverse it.
12. UI Frontend -----> Backend Node Server ------> Internally runs **Postgraphile** to convert the **Graphql Query** to SQL ----> Database to read or mutate.
13. In order to get the latest travel data which keeps updating, we have two approaches:
    - POLLING - Keep calling the API and check for latest data
    - EVENT Subscription - Any time the data updates, it triggers an event to get the fresh data from the API server.
14. There is a difference between development and deployment servers. In development we dont need to worry about multiple servers, but when we deploy we need to configure the microservies and their distribution based on demand.
15. UI --------> | API Gateway | ---- { Load Balancers } ----> Node Backend ----- { Load Balancers } -----> Multiple Databases
    - API Gateway ensures security and accesses any risks involved before transmitting to the backend.
16. Now to ensure that when we are making api calls, querying from databases, and more, we can run into problems such as:

- Too many requests made to the server which overburdens and slows it down
- 429 Response

17. To handle these huge number of requests:
    - There is an Application Control Flow process followed
    - For every 1000 requests per minute, only 50% percentage of requests are processed at random with a probability of 1/2.
    - The API requests that were rejected are made to wait for a little longer and then catered to.
    - DynamoDB stored procedures are used to handle the processes for the API requests that were successful.