Assumptions:

- **Contact** entity must have first and last names. They cannot be null. *Middle name*, however, is optional and can be null.
- It is assumed that there can be contacts who may have same combination of first and last names. They need not necessarily be unique.
- **Address** entity must have all the values. None of the address components can have null values. For example, an address that does not have a *city* is invalid in this design.
- **Zip code** of an address must always be exactly 5 digits. Alphabets and digits whose length is not equal to 5 are not accepted as zip codes.
- **Address** of address entity can have at most 200 characters only.
- Specific preset values are not provided to the user for the *type* of address, or phone. User is expected to enter the *type* manually. This is considered because this design can accommodate any combination of values for *type* rather than preset values.
- **Phone** entity must have both *area code* and *number*. Null values are not allowed for these components of phone numbers. Combined, area code and number must be only 10 digits long.
- **Area code** of a phone number must have exactly 3 digits. Alphabets or digits whose length is not equal to 3 are not allowed.
- **Phone number** of phone entity must have exactly 7 digits. Alphabets or digits whose length is not equal to 7 are not allowed.
- In the **Date** entity, it is assumed that even the future dates can be added. This is assumed after considering the flexibility in terms of what types of events can be added. For example, it is possible to add the expected important meeting date for some event that is related to the contact being added.
- When a contact is deleted from the primary table, all of its children referring to this contact are deleted as well. This is achieved using **cascaded delete** option.

System Architecture:

- The Contact App is a native Android application that runs only on Android devices. For full functionality, the recommended OS version of Android is Oreo and above. The app runs both on QEMU Android Emulator and physical device.
- The top layer, UI and front end portion of the application, is developed using Java 8 and Android SDK 26. Other third party libraries used as a part of the development are Google's GSON, and Retrofit API. Most of the validations are done in front end by adhering to the Android's recommended view model pattern. On a high level, the input of this layer comes from the end user in the form of UI interaction and the output of this layer is a serialised JSON object wrapped in a HTTP request which is sent to server side application.

- The second layer, server side application, is purely developed using Node JS. The role of this layer is to listen for incoming HTTP requests sent by the layer above and handle them based on the HTTP Methods such as POST, GET, and DELETE. Core logic includes validating the JSON data and inserting it into the database in such a way that the atomicity of database is maintained. For example, the addresses and other secondary data of contact is inserted only after inserting the primary data in the contact table. This layer communicates with the database server to perform any of the CRUD operations.
- The bottom most layer is the database server and the database itself. The database server in this case is the local server (localhost). Any database related requests are executed and the output (or error) is sent to the second layer. This layer worries only about the data and not the semantics of it. For example, it only validates the schema constraints but does not check if the value actually makes sense in the reality.
- The completed HTTP request is sent back to the front end application in the form of HTTP response containing JSON object. In the front end, this JSON is deserialised into POJO which is used to update the UI.

Types of HTTP requests:

- To insert or update a contact, HTTP POST request is sent with a serialised JSON object. Structure of JSON used for this design is described below. HTTP Response does not contain any data. **Code 200** means the insert is successful.
- To fetch contacts, HTTP GET request is sent with a query parameter “search_criteria”. Response contains serialised **JSON data of list of contacts**.
- To delete contacts, HTTP DELETE request is sent with a query parameter “contact_id”. Contact with this ID is deleted from the contact table which will result in the cascaded delete of all the contact’s secondary details as well. HTTP Response does not contain any data. **Code 200** means the delete is successful.

Structure of JSON:


```
{
  "nameData": {
    "fname": "Carlita",
    "mname": "Gussi",
    "lname": "Bindin"
  },
  "addressData": [
    {
      "_id": 13949,
      "contactId": 123,
      "addressType": "work",
      "address": "07 Butternut Avenue",
      "city": "Allen",
      "state": "Texas",
```

```
      "zipcode": 75005
    }
  ],
  "phoneData": [
    {
      "_id": 16493,
      "contactId": 123,
      "phoneType": "home",
      "areaCode": 841,
      "number": 3897758
    }
  ],
  "dateData": [
    {
      "_id": 4729,
      "contactId": 123,
      "dateType": "Birthday",
      "date": "1980-03-14"
    }
  ]
}
```