Design Document

CS - 6360.003

saivikas Meda

sxm190011

Assumptions:

- **Contact** entity must have first and last name. This are mandatory fields; user can't add contact with this fields being null. However, middle name is optional and can be null. Once the contact being saved into database user cannot change/ update name (first, middle and last).
- It is assumed that there can be to contacts having same first and last name. they need not necessarily be unique.
- **Phone entity** both area code and phone number are mandatory while adding the number. If any of this fields is left empty and alert message will be displayed to the user.
- **Area Code** entity must be exactly 3digits long. User cannot enter more than 3digits as logic restricts it and any number less than 3 will not allowed to save into database. User can only enter digits as number keypad is displayed.
- **Phone Number** entity must be exactly 7digits long. User cannot enter more than 7 digits and any number less than 7 will not be allowed to save into database. User can enter only digits as numbers keypad is displayed.
- **Address entity** at least one of this fields (Street, Apt, City, State, Zip code) needs to be filled in order to save the contact into database. The other fields can be filled by clicking on modify button later. street can be at most 40 characters only. zip code can be at most 10 characters only. apartment number is an optional text field. even null or empty string is a valid input.
- **Date** entity, it is assumed that even future dates can be selected. The selection of dates depends on the event. If the event is in future and future date needs to be entered.
- When a contact is deleted from the primary table (Contact table), all of its children referring to this contact are deleted as well. This is done by using **cascade delete feature**.

System Architecture:

- The DB_Project App is a native iOS application that runs only on iOS devices. The application runs on but physical device and iPhone simulator.
- The top layer, UI and front end of the application, is developed using Swift 4 and XCode (iOS SDK) and Cocoa touch framework. Important libraries used as part of application are UIKIT (developing the UIComponents) and UserNotifications (for sending notifications to the user). Most of the validations are done on front end either by logic or UIComponenets adhering to iOS's view model architecture. The input from end user is taken through the UIComponents (text fields, buttons, tables) and ouput of this layer is sent a JSON object using JSON.serialisation and wrapped in a **HTTP request** which is sent to server side (Backend) of application.
- The Second layer, Server side application, is purely developed using Node JS. The main part of this layer is to listen for incoming HTTP requests sent by the layer above this (front end) and handle them based on the **HTTP** methods such as "**POST**", "**GET**", "**DELETE**". The data transferred as JSON objects between this layers. The JSON objects are validated and inserted into the database. First the Contact table is updated and **Contact_id** is fetched and other tables (Address, Phone, Date) are updated. This way the atomicity of database is maintained. This layer perform any of the CRUD operations by communicating with the database server.
- The bottom layer is the MySQL database server and database. The database server is hosted on the local server (localhost). Any errors occurred during inserting data into database are sent to the second layer. Second layer handles the error sent by database server. This layer doesn't validate any data that is sent to it, it only raises error when the data doesn't match with the semantic of it. For Instance, If the address (street) field max characters are 40, if a string of 50 characters are sent, it will result in error. This layer doesn't check weather data to relevant to the field or not.
- The completed HTTP request is sent back to first layer (front end) of the application in the form of HTTP response containing JSON object. In front end, we deserialize the JSON object and update the UI.

Types of HTTP requests:

- To insert or Update a contact, **HTTP POST** request is sent with a serialized JSON object. HTTP response does not contain any data. Code 200 means the insert is successful. The insert and update are differentiated by URL String. for update request we send query parameter which contain contact_id.
- To fetch contacts, **HTTP GET** request is sent with query parameter "search". HTTP request contains serialized JSON data of list of contacts that satisfies the search parameters.
- To delete contacts, HTTP DELETE request is sent with query parameter "contact_id". Contact with this contact_id is deleted from the contact table which result in the cascade delete of all contact related details in other tables (Address, phone, date). HTTP response doesn't contain any data.

Structure of JSON Object:

{

"contact_id" : 1,

"fname" : saivikas

"mname": ""

"lname": "Meda"

"phone" : [{"area" : "788", "phonenumber":"6788768", "type":"Home"},

{"area" :"999", "phonenumber":"1231231", "type": "Cell"}]

"address": [{"Street": "7815 Mccallum", "apt:"14206", "city":"dallas", "state":"Texas", "zipcode":"75252","type":"Home"}]

"date" : ["type":"Birthday", "date":"1995-06-25"]

}