->AI Integrated , Dynamic Quiz App <-

About :

This project is an **AI-based online quiz web application** developed using **HTML, CSS, Bootstrap, JavaScript, and Flask**. The system allows users to register and log in, enter a topic of their choice, and select a difficulty level (Easy, Medium, or Hard). Based on the user input, the application dynamically generates multiple-choice questions using AI.

The quiz evaluates user responses, calculates the final score, and stores quiz results in a database for future reference. The application is designed with a simple and responsive user interface and follows a modular backend structure, making it easy to maintain, scale, and deploy for public use.

Tech use :

For FrontEnd :- HTML, CSS, JS, BOOTSTRAP

For BackEnd :- FLASK OR DJANGO

For DataBase :- SQLITE OR MYSQL

AI Used :- OpenAI / gpt 4.0 mini

BASIC FLOW :

In frontend first home page where it include about the application and a login /signup page .

Login/singup page include name gmail and name password ect .

In second section user topic input and difficulty level and a generate button , it will generate a 10 question.

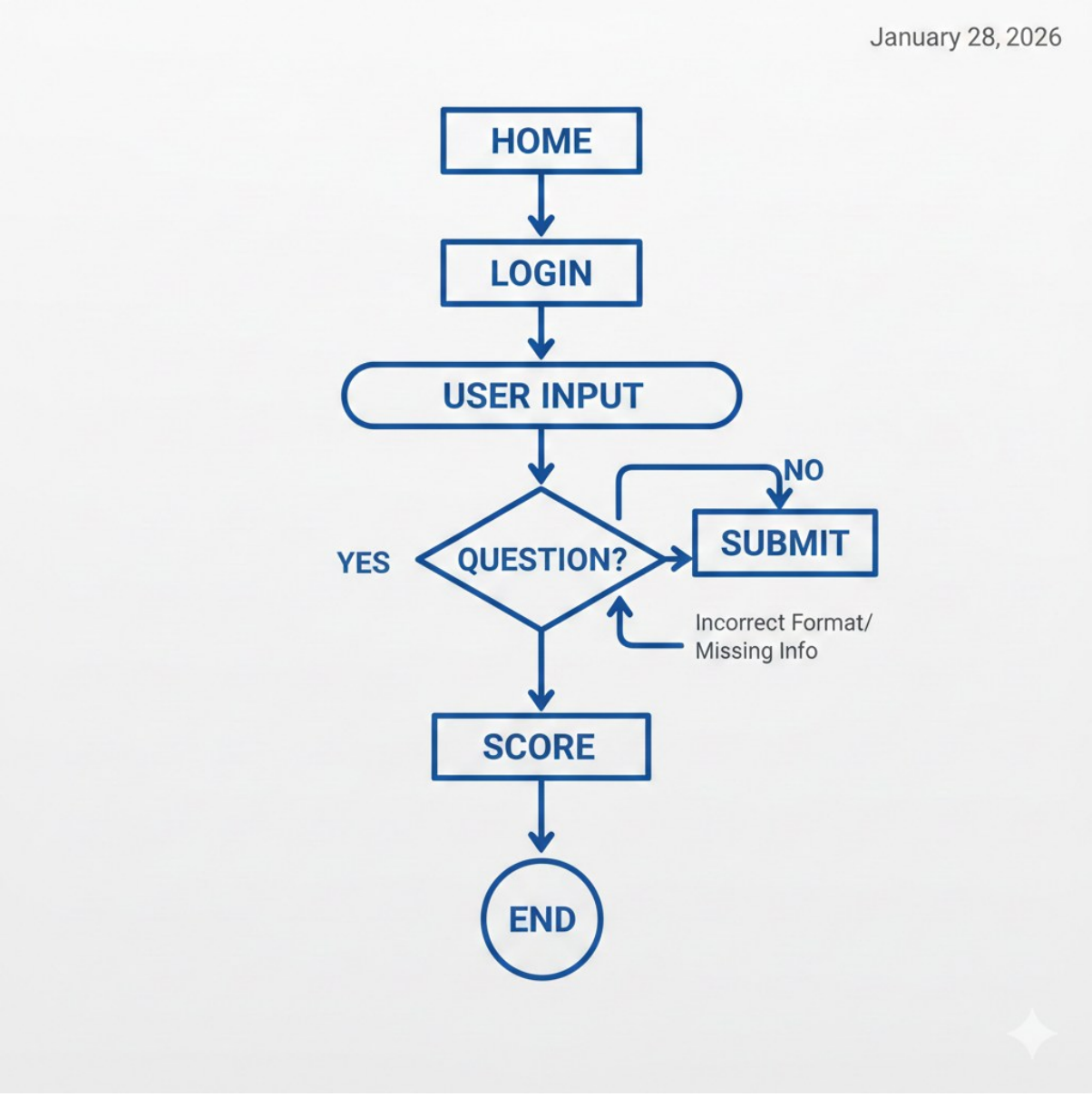
After sekecting answer submit button .

In third page first it will display score and the correction of wrong answer

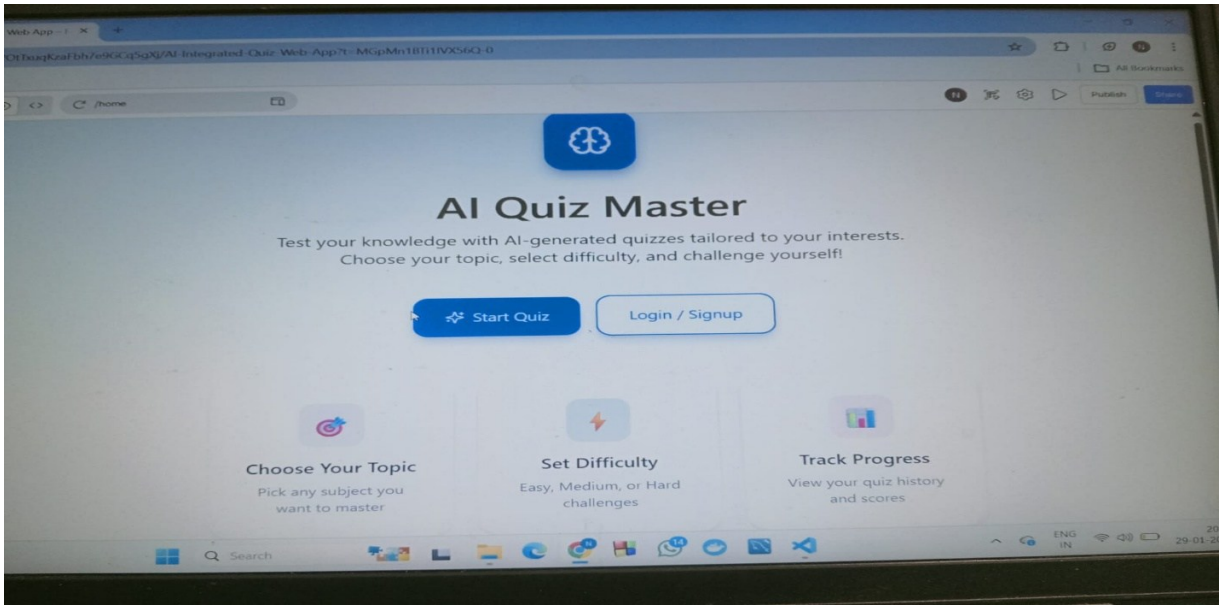
In Backend we use flask or Django where we use our logic , it will connected to data base for getting details enter by user and history

Database include Two table one visible for user other for admin

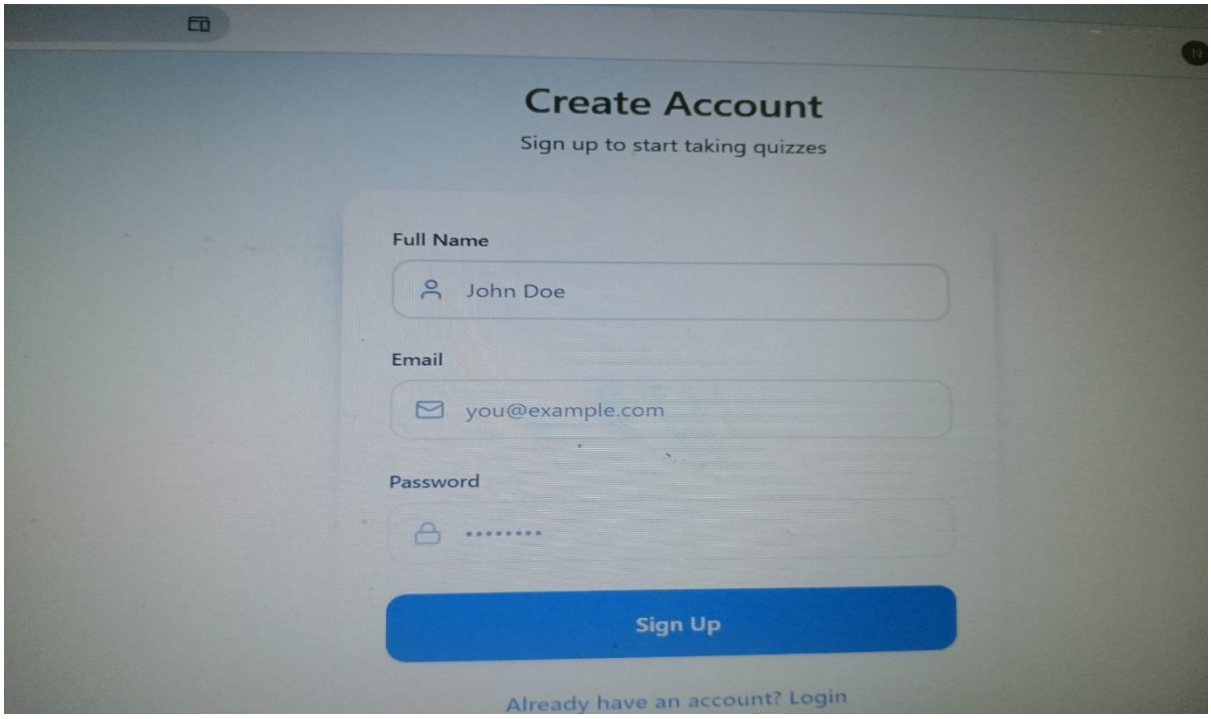
For admin it show the detail about user , in user level it shows topic ,Score and date of test in history

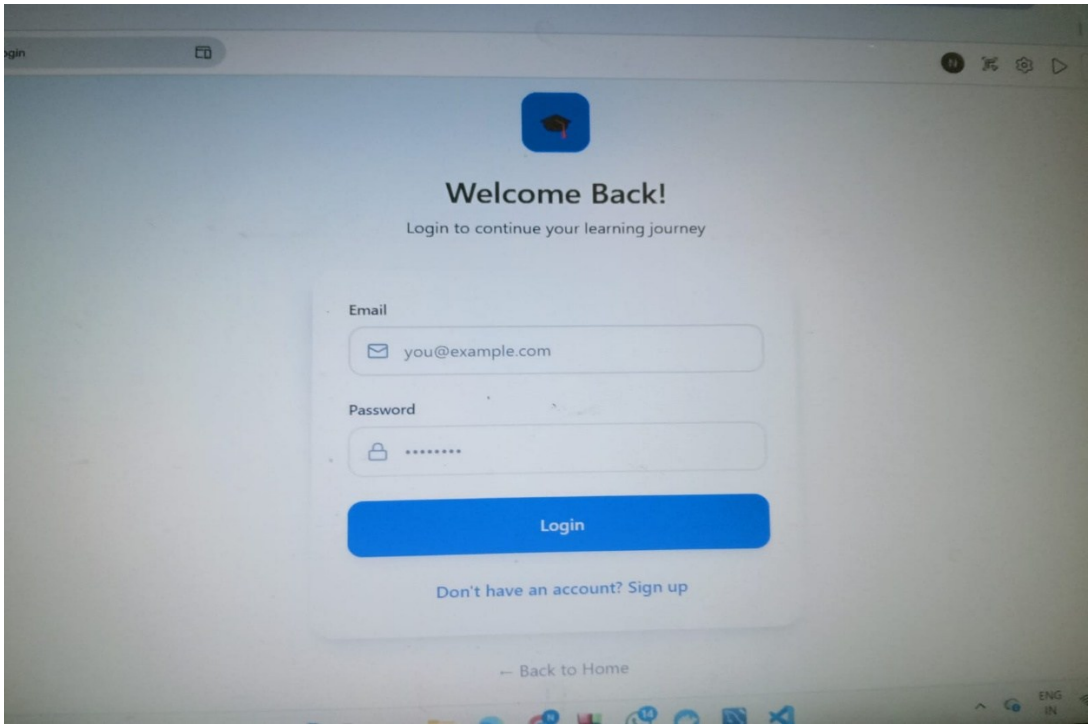


Front End

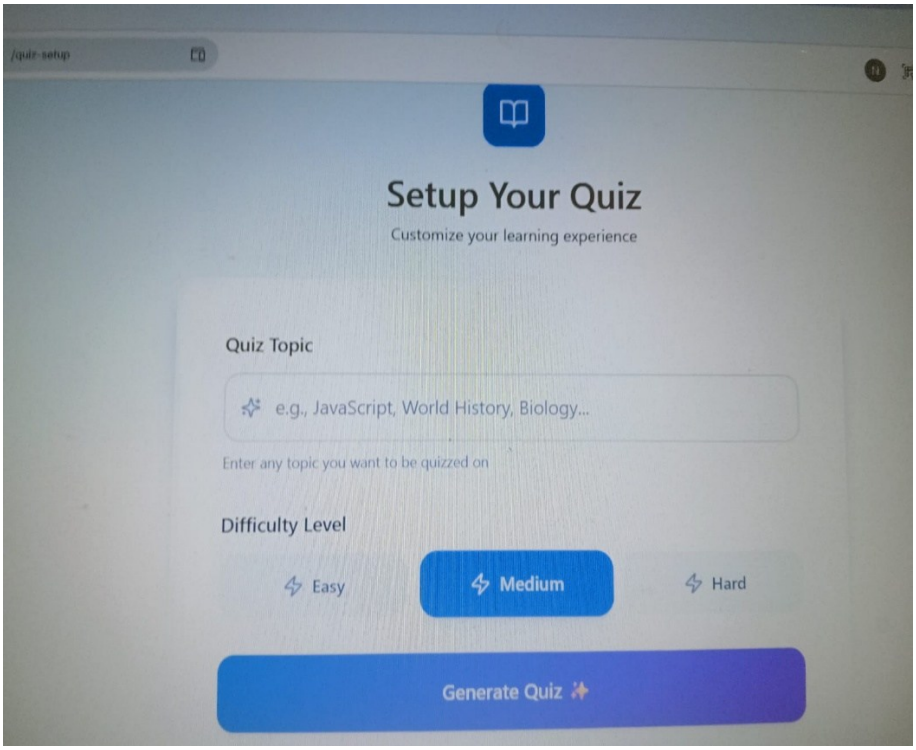


About section | start quiz | Login / Signup system





Create and signup



Quiz test page

Back End

 Python – core backend language

 Web Framework – Flask

 Database ORM – to talk to database easily

 Authentication system – login, roles (user/admin)

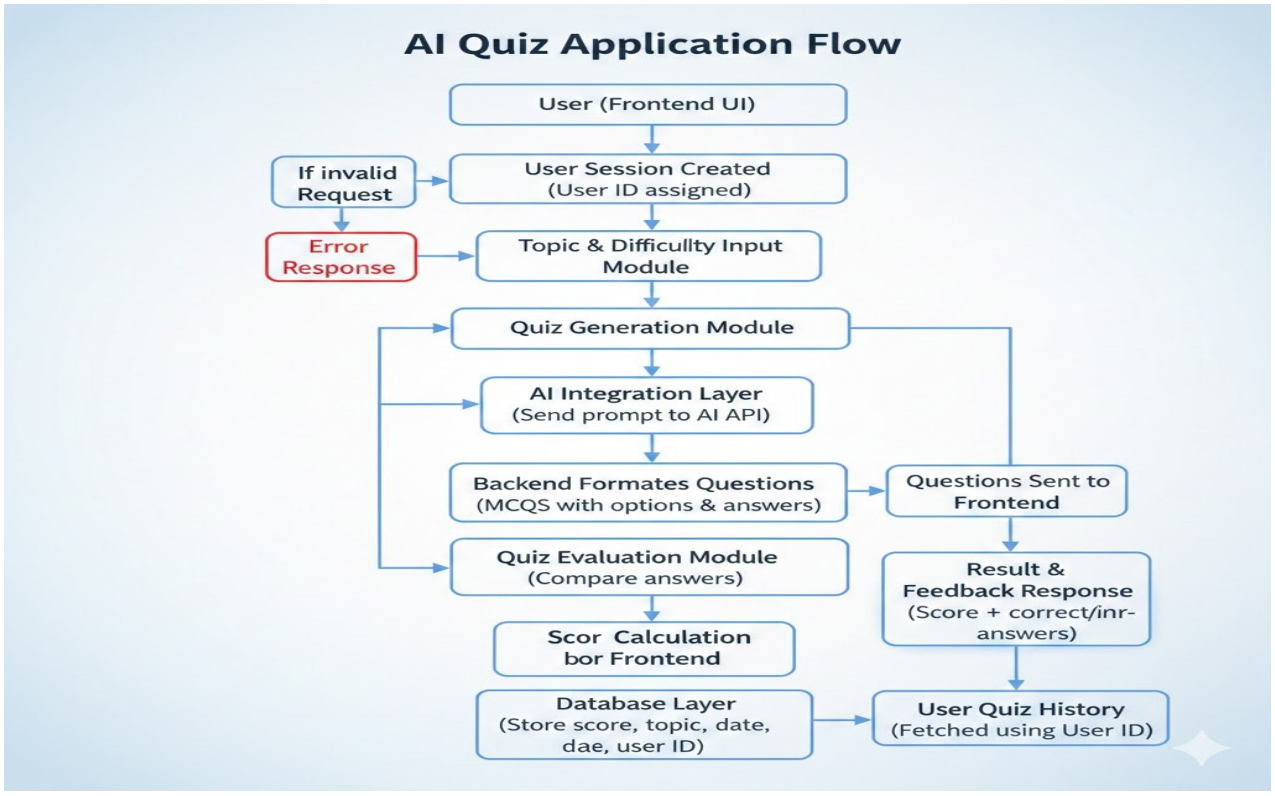
 Session management – to keep users logged in

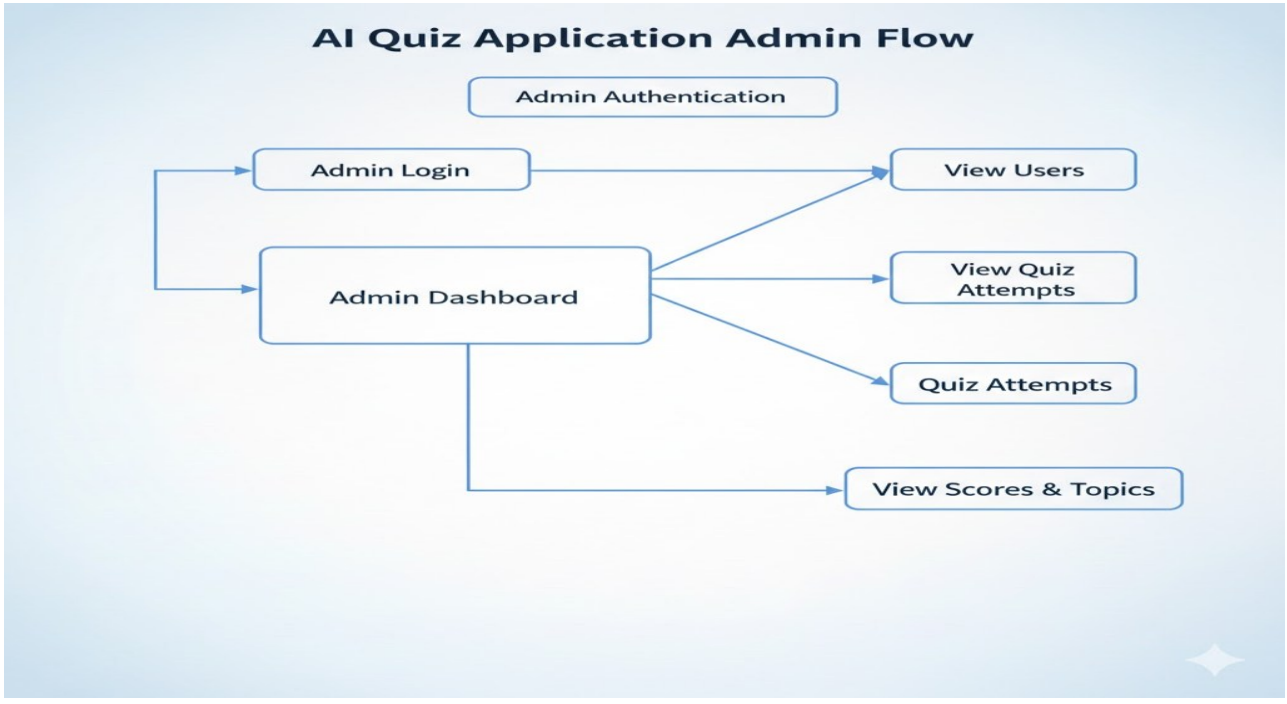
 AI API integration – to generate quiz questions

 Deployment server – to make it public

- ➔ The backend of the AI Integrated Dynamic Quiz App is developed using Flask to handle application logic, user authentication , quiz processing and database management.
- ➔ The backend manages user registration and login, ensuring that each user has a unique account and secure session. After login, the system tracks users using their unique user id , which allows storing and retrieving individual quiz histories, scores, selected topics, and test dates from the datebase.
- ➔ AI integration is handled in the backend, where user-selected topics and difficulty levels are sent to the AI API (OpenAI / GPT-4.0 mini).
- ➔ Once the quiz is submitted, the backend evaluates user responses, calculates the final score, stores results in the database, and returns detailed feedback, including correct and incorrect answers.
- ➔ An **admin module** is also managed in the backend, allowing admins to view user details, quiz attempts, and performance analytics, while normal users can only access their own data.

The backend follows a **modular and scalable architecture**, making it easy to maintain, extend, and deploy for real-world use.


AI Quiz Application Flow
User (Frontend UI)
If invalid Request
User Session Created (User ID assigned)
Error Response
Topic & Difficullty Input Module
Quiz Generation Module
AI Integration Layer (Send prompt to AI API)
Backend Formates Questions (MCQS with options & answers)
Questions Sent to Frontend
Quiz Evaluation Module (Compare answers)
Result & Feedback Response (Score + correct/inr-answers)
Scor Calculation bor Frontend
Database Layer (Store score, topic, date, dae, user ID)
User Quiz History (Fetched using User ID)


AI Quiz Application Admin Flow
Admin Authentication
Admin Login
View Users
Admin Dashboard
View Quiz Attempts
Quiz Attempts
View Scores & Topics

DataBase System

The data base of this application have two tables linked with each other one table store user information and other quiz history .

User info table:

It store user details , every user have their unique id , if user create new account it store with new unique id . If user login the user name , gmail , password are checked in database if found it check its id then it connect to another table.

. User ID (unique)

. Name

. Email

. Password

. Role (User / Admin)

. Account creation date

Quiz history table:

Quiz table have details of user attempts it store with user id it include id , date of attempt , Topic , Score ,level it like history of a particular user . When user login into system db check id and select those attempt from db having same db.

. User ID (linked to Users table)

. Topic

. Difficulty level

. Score

. Date & time of quiz

How User Data Is Kept Separate (Very Important)

- When a user logs in, the backend creates a **session** using their **User ID**
- Every quiz attempt is stored with that **User ID**
- When a user views history:
    - Backend fetches data **only where user_id = logged-in user**
- So users can **only see their own quiz history**

👉 This is how **data separation** is achieved.

How Data Is Visible for Admin

🔐 **Admin Access Logic**

- Admin is also stored in the **Users table**
- Admin has a special **role = admin**
- When admin logs in:
    - Backend checks role
    - Grants access to admin dashboard

IN SHA ALLAH!, IT GONNA BE DONE BEFORE OR IN RAMZAN