1. Feature Description
   1. **Requirements**

* A chat bot interface to interact with the user, which shows different subscriptions of educational courses and their discounted prices based on user’s performance in tests.
* User should be able to add liked courses to cart.
* User should be able to remove courses from cart.
* User should be able to complete hassle free payment at checkout.
* User should be able to view his/her order history.
* Admin should be able to list his/her courses on the platform, along with its pictures, subject, topic, description and price.
* After the payment is complete from user’s side, the vendor should receive the money in his account.
  1. **Detailed description of the Feature**

The web app contains following sections:

* Recommendation Screen
* Checkout Screen
  + 1. **Recommendation Screen**

This is the first screen which appears when the user completes test. This screen is the main place where all the user-chatbot interaction happens. This screen contains a chat bot which welcomes the user to the app. The chatbot shows the overall result and topic wise result.

Based upon the score of test taken by the user, the chat bot suggests various courses.

The user can choose the course of his choice from the list. User can either add it to cart or purchase instantly.

* + 1. **Cart Window**

This is a small modal window which shows the items currently in the user’s cart.

* + 1. **Checkout screen**

In this screen, users can make final adjustments in their cart and confirm all the courses they added in the cart. After that they can proceed for payment.

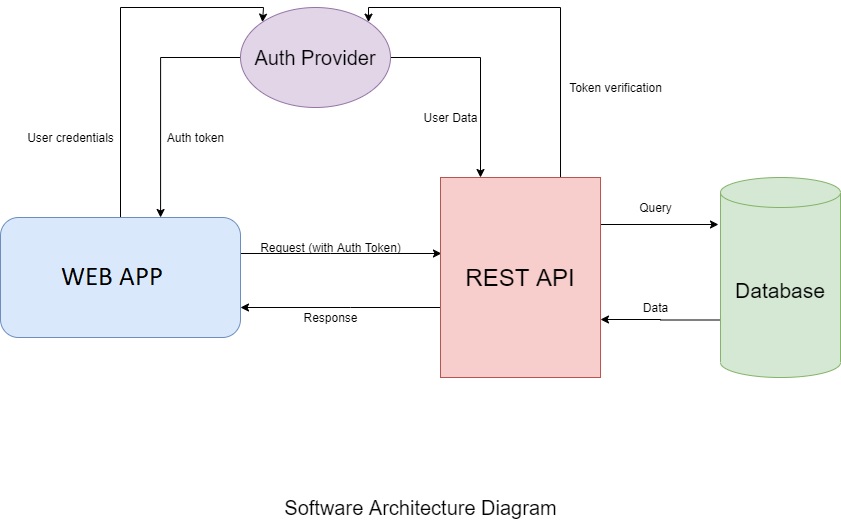
1. Design and Specification
   1. **Software Architecture**

The web app will form the frontend (client-side) of the application, which can be accessed from any web browser.

The web app will communicate to a RESTful web service or API which will form the backend (server-side) of the application. This RESTful web service contains all the business logic which is critical for the application to serve its purpose.

The server-side REST API will talk to a cloud hosted database and provide our frontend web app with all the requested data.

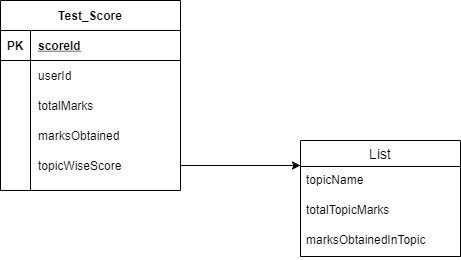
For authenticating the requests, the client-side PWA will get a token from authentication server (Firebase) and attach it as a header to all the HTTP requests to the server. The REST API will then communicate with the authentication server (Firebase) to verify the token and hence validate the incoming requests



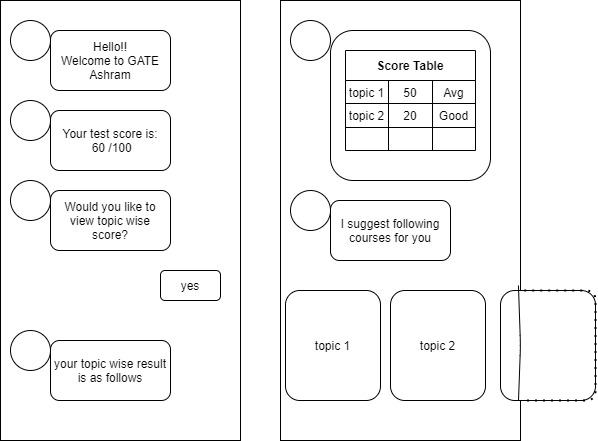
* + 1. User Interface:

The user interface consists of various components and widgets which make the complete app. The data management has been implemented by using a centralized data store (Redux Store).

* Every screen is built out of a reusable screen component.
* Every screen consists of many widgets which are further built from even smaller components.
* The home screen consists of a chat window component, which is further divided into chat message widget and inside chat message widget we have subject list, topic list and course list widgets.
* Every course list is made up of individual course widgets, which contain course title component, image component, price component as well as add-to-cart and buy-now button components.
* The cart widget consists of multiple cart-item widgets.
* The checkout screen consists of multiple checkout-item widgets.
* The profile screen consists of profile card widget and order history widget, which further comprises of multiple order-card widgets.
* Every widget talks to the centralized data store via actions. This ensures the accuracy and consistency of data is maintained.
  + 1. REST API Services
* Subject Service:
  + This service provides a list of all the subjects of courses which are available.
* Topic Service:
  + This service provides a list of all the topics of courses which are available under a particular subject.
* Courses List Service:
  + This service provides a list of all the courses which are available under a subject and topic.
* Course Service:
  + Provides all the details about a particular course.
  + Provides price of a particular course.
  + Provides images (URLs) of a particular course.
* User Service:
  + Gets the profile details of a particular user.
  + Creates a new User in the database after login process.
* Cart Service:
  + Gets all the courses in the cart of a particular user.
  + Adds a new course in the cart of a particular user.
  + Removes an existing course from the cart of a particular user.
* Payment Service:
  + Performs the transaction of money using stripe payments API.
* Orders Service:
  + Provided a list of all the past orders of a particular user.
  + Adds a new order into the order history of a particular user after the payment is successful.
    1. Database Design



* + 1. Frontend Design



* 1. **Sequence Diagram**
  2. **Technology Specification**

2.3.1 **Frontend**

* HTML 5
* CSS 3/SCSS
* JavaScript
* Angular
* Redux
* Firebase

**2.3.2 Backend**

* Node.js
* Express.js
* Firebase-Admin
* Stripe
  + 1. **Database**
* MongoDB (Atlas)
  + 1. **Version Control**
* Git
* GitHub
  1. **Deployment Specification**
* The frontend Web App is deployed as a standalone Single Page Application (SPA) on Heroku. Automated deployment is available, which means whenever there is a change pushed to the GitHub repository, the change is deployed in production by Heroku.
* The backend RESTful API is deployed separately on AWS cloud server provided by Heroku. The deployment process is taken care of by Heroku CLI (Command Line Application).
* The cloud hosted Database is provided by MongoDB Atlas. It is configured to be accessible to only specific users after proper authentication.

1. Testing
   1. **System Testing**

System testing validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications.

* 1. **Test Cases**

Test Cases help guide the tester through a sequence of steps to validate whether a software application is free from bugs and is working as required by the end user. Typically, test cases for a given module or part of an application, are grouped into a test suite. More often than not, a test session will include many test cases as there will usually be more than one specific scenario to be tested.

* + 1. Test Case 1

Title: Course list

Description: After taking the test, the user should be shown a list of courses from the specific subjects and topics.

Precondition: User should be on the Home screen.

Assumption: Supported Browser is used.

Test Steps:

* Take a test.

Expected Result: The chatbot shows a list of courses from the chosen type, category and brand.

* + 1. Test Case 2

Title: Cart

Description: After adding a course to the cart, the course must appear in the cart window as well as checkout screen.

Precondition: User should have fetched a list of courses as described in Test Case 1.

Assumption: Supported Browser is used.

Test Steps:

* Click on “Add To Cart” button on the course card.
* Open the cart window.
* Go to Checkout Screen.

Expected Result: The selected course is present in the cart window as well as the Checkout Screen.