Architecture document

Application

The Learning Roadmap System employs a modular approach to ensure scalability, flexibility, and efficient resource management, using principles from microservices, event-driven, and serverless architectures. These modern architectures collectively enhance the performance, availability, and scalability of the application, aligning it with best practices for cloud-native, distributed applications.

Microservices

The application adopts a microservices architecture, where individual features or services—such as user authentication, task management, chatbot

interactions, and data storage—are developed, deployed, and managed independently. Each microservice performs a specific function and communicates with other services through APIs, making the overall system more modular and maintainable. For example:

- **User Authentication Service:** Manages sign-in, login, and user session management.
- **To-Do List Service:** Manages the creation, updating, and tracking of tasks.
- **Chatbot Service:** Supports interactions with the chatbot, providing reminders, tips, and user feedback.
- **Analytics Service:** Gathers and processes data from the user's study patterns to offer personalized insights in the dashboard.

Event-Driven Architecture

An event-driven architecture enables the Study Planner application to respond in real time to specific events or triggers. This architecture model is especially useful for triggering notifications, updating task statuses, and sending reminders based on user interactions and scheduled events. For example:

- **Progress Tracking:** As the user marks tasks as completed, events are logged to update the dashboard and analytics, allowing users to see up-to-date progress.
- **Chatbot Interactions:** When users request assistance or tips from the chatbot, an event triggers a response, creating a conversational experience.

Serverless Architecture

The application also incorporates serverless architecture principles for specific components, enabling functions to execute on-demand without the need to

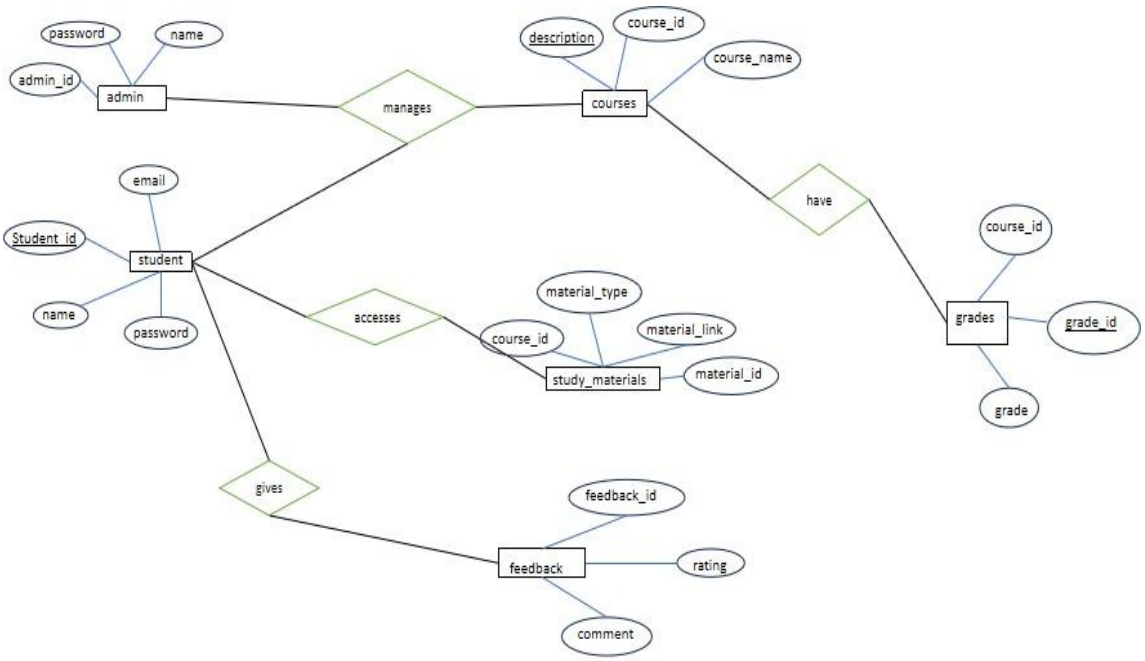
manage server infrastructure. Firebase, which supports serverless functions, is utilized to manage backend tasks such as database queries, authentication, and data synchronization. Key examples of serverless functions in this application include:

- **User Authentication Functions:** Firebase Authentication handles user sign-in and session management securely, without the need for dedicated servers.

- **Data Storage and Syncing:** The to-do lists, progress data, and chatbot interactions are stored and retrieved from Firebase's serverless database, allowing real-time data sync without a traditional backend server.

Database

ER Diagram



Schema Design

Students(student_id, name, email, password, last_login) Courses(course_id, course_name, description, teacher_id)

To-Do List(task_id, student_id, task_title, task_description, due_date, status)

Study Materials(material_id, course_id, material_type, material_link) Grades(grade_id, student_id, course_id, grade, date)

3. Data Exchange Contract

3.1 Frequency of Data Exchanges

The data exchange between various services (microservices, database, external systems) occurs at different intervals:

Real-time exchanges for tasks like chatbot responses, grade analysis, and to-do list updates.

Daily updates for student performance analysis and progress reporting.

3.2 Data Sets

Student Data: Includes student profiles, to-do lists, and grade records. Course Data: Includes study materials, course details, and subject breakdowns.

Grade Data: Includes historical and recent grades for each course the student is enrolled in, used for focus area analysis.

Chatbot Interaction Data: Includes the queries students ask and the chatbot's responses, stored for future reference and improvement.

3.3 Mode of Exchanges (API, File, Queue, etc.)

API Exchanges:

REST APIs are used for data exchange between frontend and backend

services (e.g., retrieving to-do lists, study materials, and chatbot responses using Groq APIs).

Queue-based Exchanges:

Event-driven components (e.g., grade analysis) use message queues to handle asynchronous processing of data.

File-based Exchanges:

Exporting progress reports, study plans, and grades can be done via downloadable CSV or PDF files.