E - LEARNING website

(ASK ME A QUESTION page)

Team members

VENU

BALA

SRI HARSHINI

SAI THANMAI

INTRODUCTION:

Project Overview:

The "Ask a Question" page is a crucial feature of our e-learning platform, designed to enhance student engagement by providing a streamlined method for students to seek help and clarification on course materials. This feature aims to bridge the gap between students and educators, ensuring that learning continues smoothly outside the traditional classroom setting.

Importance:

By facilitating real-time interaction and providing a repository of past questions, the "Ask a Question" page supports continuous learning and addresses individual student needs efficiently.

Features and Functionality:

<u>Core Features:</u>

1. Question/Complaint Submission Form: Allows students to submit their queries directly on the platform.

2. Redirect to Live Meetings: Provides an option to join live meetings (Zoom/Google Meet) for more detailed discussions.

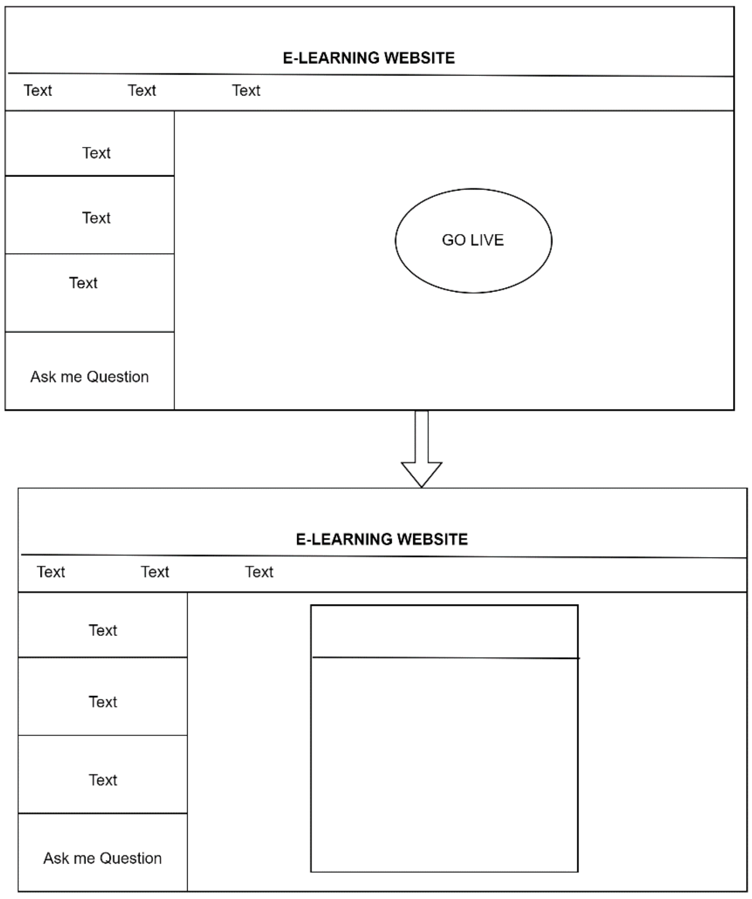
3. User Detail Capture: Collects and stores user information to tailor support and track interactions.

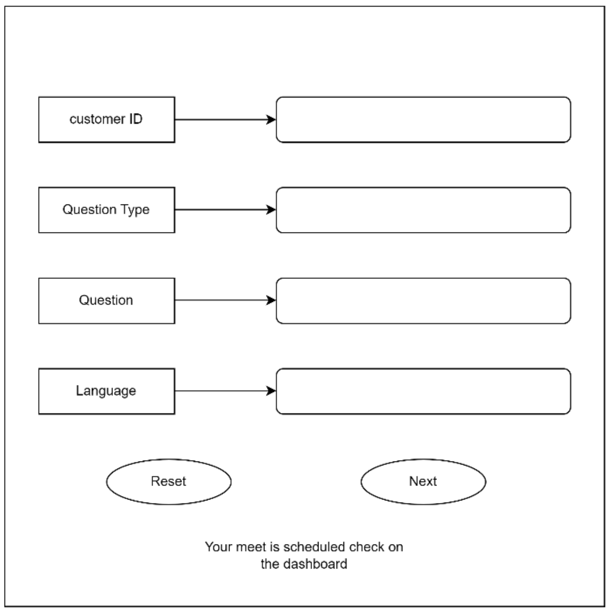
Advanced Features:

1. Automatic Meeting Link Generation: Automatically generates meeting links for live sessions.

2. Repository of Questions and Answers: Maintains a searchable database of previously asked questions and their answers.

BLOCK DIAGRAM





Technical Specifications:

Front-End Technologies:

1. HTML: Structure of the web page.
2. CSS: Styling and layout.
3. JavaScript: Interactivity and dynamic content handling.

Back-End Technologies:

1. Java: Server-side logic and processing.
2. MongoDB: Non-relational database for storing Q&A data.
3. MySQL: Relational database for user information.
4. Postman: API testing and integration tool.

Endpoints:

1. Question Management:
   - POST /api/questions: Submit a new question.
   - GET /api/questions: Retrieve all questions.
   - GET /api/questions/:id: Retrieve a specific question by ID.

     Required Fields:
      - complaintID (auto-generated random number)
      - questionID (auto-generated number)
      - questionTopic: Topic of the question
      - description: Detailed description of the question
      - meetingLink: Auto-generated Zoom/Google Meet link
      - userName: Name of the user submitting the question

2. Answer Management(optional):

- POST /api/answers: Submit a new answer.
- GET /api/answers: Retrieve all answers.
- GET /api/answers/:id: Retrieve a specific answer by ID.

3. Meeting Management:
   - POST /api/meetings: Schedule a new meeting.

     Required Fields:
     - meetingID: Unique identifier for the meeting
     - meetingLink: Generated link for the meeting
     - questionID: Associated question ID
     - timestamp: Date and time of the meeting
   - GET /api/meetings: Retrieve all meetings.
   - GET /api/meetings/:id: Retrieve a specific meeting by ID.

Database Design:

<u>Database Schema:</u>

1. Structure:

- Tables for users, questions, answers(optional), and meeting link.
- Relationships between users and their questions, questions and answers, and users and meetings.

2. Data Capturing:

- User Details: Name, email, course information.
- Question Details:
  - complaintID: Auto-generated unique identifier
  - questionID: Auto-generated unique identifier
  - questionTopic: Topic of the question
  - description: Detailed description
  - meetingLink: Auto-generated meeting link
  - userName: Name of the user
- Meeting Details:
  - meetingID: Unique identifier for the meeting
  - meetingLink: Generated link
  - timestamp: Date and time of the meeting

<u>Schema Design:</u>

1. Questions Table:

- complaintID: Primary Key
- questionID: Unique Identifier
- questionTopic: String
- description: String
- meetingLink: String

- userName: String

- timestamp: DateTime

2. Answers Table(optional):

- answerID: Primary Key

- questionID: Foreign Key

- answerText: String

- timestamp: DateTime

3. Meetings Table:

- meetingID: Primary Key

- meetingLink: String

- questionID: Foreign Key

- timestamp: DateTime

User Interface Design:

Design Principles:

1. User-Friendly Design: Simple and intuitive interface to encourage usage.

2. Accessibility: Ensuring the platform is accessible to all users, including those with disabilities.

Wireframes and Mockups:

1. Sample Screens:

- Question Submission Form: Fields for entering questionTopic, description, and auto-generated meetingLink.

- Real-Time Q&A Section: Display of live questions and answers.

- Meeting Redirection Interface: Button/link to join a live meeting.

Implementation Details:

<u>Development Process:</u>

1. Step-by-Step Guide:
   - Setting up the project environment.
   - Developing the front-end interface.
   - Implementing back-end logic and database integration.
   - Integrating front-end with back-end.
2. Front-End and Back-End Integration:
   - Connecting the user interface with server-side functionality.
   - Handling API requests and responses.

Testing and Deployment:

<u>Testing Procedures:</u>

1. Unit Testing: Testing individual components for functionality.
2. Integration Testing: Ensuring different parts of the system work together seamlessly.

<u>Deployment Steps:</u>

1. Setting Up the Environment: Configuring servers and deployment tools.
2. Deploying the Application: Moving the application from development to production.

FUTURE ENHANCEMENTS:

POTENTIAL IMPROVEMENTS

1.ADDITIONAL FEATURES: Adding more functionalities based on user feedback along with those parts where we can improve more and more things at a particular function.

For example:

Advanced Search and Filtering:

- Integrate a search bar allowing students to find existing questions and answers based on keywords or topics.
- Implement filters to categorize questions by course, topic, difficulty level, or even unanswered questions.

Flexibility of Time slots:

- Allowing the users to select their own time slots or make a request for Time period giving them the opportunity to be more flexible.

Advanced Analytics and Reporting:

- Track student inquiries and identify common pain points to improve course content or teaching methods.
- Analyse question trends to understand areas where students need more support.
- Generate reports for educators to gauge student comprehension and identify knowledge gaps.

Community Building Features:

- Allow students to answer each other's questions, fostering peer-to-peer learning and knowledge sharing.
- Integrate a gamification system where students earn points or badges for asking insightful questions and providing helpful answers.
- Implement discussion threads where students can delve deeper into specific topics sparked by a question.

AI-powered Assistance:

- Introduce a chatbot that can answer basic student queries using Natural Language Processing (NLP). This can help with frequently asked questions and reduce educator workload.
- Consider a feature where the system suggests similar questions or relevant course materials based on the student's query.

2.PERFORMANCE OTIMIZATIONS:

Advanced Load Balancing: Implementing more sophisticated load balancing algorithms and strategies, such as global server load balancing (GSLB), to better distribute traffic across multiple data centers.

Real-time Analytics: Integrating real-time analytics to monitor application performance and user interactions, allowing for immediate adjustments and optimizations.

CONCLUSION:

In conclusion, creating an "Ask Me a Question" page with G-meet link generation for a student learning platform using Spring Boot offers a dynamic solution for student-educator interaction. This system leverages the strengths of Spring Boot's backend capabilities for data processing and API integration with Google Meet. By implementing JPA and a database, you can store student questions and potentially manage time slots for G-meet sessions.

The user-friendly frontend, likely built with HTML, CSS, and potentially JavaScript, allows students to submit questions and receive G-meet links for personalized support. This setup ensures a seamless and intuitive user experience, making it easier for students to get the help they need in a timely manner. Additionally, the platform can be designed to send email notifications to both students and educators, confirming the creation of the G-meet session and providing the necessary details.

We can further enhance this by implementing the other functionalities we have mentioned in the Future Enhancements, which can significantly improve the overall functionality and user engagement of the platform.

By utilizing Spring Boot's powerful ecosystem, this project not only facilitates efficient backend operations but also opens up possibilities for future enhancements and scalability. This robust solution aims to bridge the gap between students and educators, fostering a more interactive and supportive learning environment.