User Story: Change Name for Verified Badges

📌 Title: Rename Verified Badges Section for Clarity

As a user of the bitLabs application,
I want the navbar section label to clearly indicate that it contains both skill-based tests and achievement badges,
so that I can easily understand the purpose of the section without having to explore it first.

📋 Detailed Description

In the current bitLabs application, the navbar contains a section labeled “Verified Badges.” This section includes:

- Skill-based tests that users can attempt.
- Badges awarded upon successfully passing those tests.

The label “Verified Badges” creates confusion for end users, as it only emphasizes badges while tests are also a core part of the component. Users often don’t understand what is inside this section until they navigate into it, resulting in poor discoverability and reduced clarity.

The application should:

- Update the component name/label to a clearer, more intuitive term that reflects both tests and badges.
- Ensure the new label communicates both aspects (testing and achievement).
- Improve navigation clarity and enhance overall user experience by reducing ambiguity.

✅ Acceptance Criteria

- The section label is renamed to a more descriptive and intuitive term.
- The new label is updated consistently across the application (navbar, breadcrumbs, documentation, etc.).
- End users can easily identify that this section contains both tests and badges.
- The updated label is reflected in all relevant UI components without breaking existing functionality.
- User feedback confirms improved clarity and discoverability after the change.

Subtasks Breakdown

1. **Requirement & Design**
    - Conduct user research to identify intuitive label options (e.g., “Skills & Badges,” “Test and Achieve”).

    - Validate proposed labels with stakeholders and a sample user group.
    - Update UI wireframes to reflect the new label in the navbar and related components.
    - Document the new label in the application's style guide and documentation.
2. **Frontend Development (React Web & Mobile)**
    - Update the navbar component to display the new label across web and mobile platforms.
    - Ensure the label is consistently updated in breadcrumbs, tooltips, and other UI elements.
    - Implement responsive design to accommodate the new label length, if necessary.
    - Test cross-platform consistency (React web and React Native mobile).
3. **Testing (QA)**
    - Verify the new label appears correctly in the navbar, breadcrumbs, and documentation.
    - Test that the updated label does not break existing navigation or functionality.
    - Validate that users can easily understand the section's purpose (e.g., via usability testing).
    - Perform regression testing to ensure no unintended side effects.
    - Negative testing: Ensure the app handles edge cases (e.g., long label names) gracefully.
4. **Deployment & Monitoring**
    - Deploy the updated label across production environments.
    - Monitor user feedback and engagement metrics to assess the impact of the label change.
    - Set up analytics to track clicks on the renamed section for discoverability improvements.
    - Configure alerts for any issues related to the updated UI components.

User Story: Update AI-Based Interview Preparation

📌 Title: Dedicated AI Interview Preparation Dashboard Card

As a user of the bitLabs application, I want to access AI-based interview preparation from a dedicated dashboard card instead of a chatbot, so that I clearly understand its purpose and receive personalized, profile-based responses that are relevant to me.

📋 Detailed Description

Currently, the AI-based interview preparation is implemented as a chatbot widget on the webpage. Many users misinterpret this chatbot as an FAQ/help bot for the application rather than a tool for interview preparation. This confusion leads to low engagement with the AI interview feature and poor discoverability. Additionally, the chatbot interface does not effectively leverage the user's profile, resulting in generic and less relevant responses.

The application should:

- Remove the existing chatbot widget from the webpage.
- Introduce a dedicated card on the user dashboard to highlight AI Interview Preparation.
- Ensure the new model:
    - Provides a clear entry point on the dashboard, avoiding confusion with FAQs.
    - Generates AI responses personalized based on the user's profile (skills, job preferences, history, etc.).
    - Offers an improved UI experience with a card-based layout that aligns with other dashboard components.
    - Increases user awareness and engagement with the interview preparation feature.

✅ Acceptance Criteria

- The chatbot widget is removed from the webpage.
- A new AI Interview Preparation card is added to the user dashboard.
- The card clearly communicates its purpose (e.g., "Prepare for Interviews with AI").
- AI responses leverage user profile data to provide tailored guidance.
- The card design is consistent with other dashboard components in terms of style and layout.
- User feedback confirms improved clarity and engagement with the feature.
- The feature is accessible and functional across web and mobile platforms.

Subtasks Breakdown

1. **Requirement & Design**
    - Analyze user feedback to confirm confusion with the current chatbot widget.

- Design a new dashboard card for AI Interview Preparation, including title, description, and call-to-action.
- Define personalization logic for AI responses based on user profile data (e.g., skills, job preferences).
- Update UI wireframes to integrate the new card into the dashboard layout.
- Document API requirements for fetching user profile data and generating tailored AI responses.

2. **Backend Development (Spring Boot)**

    - Remove backend services or APIs supporting the chatbot widget.
    - Develop a new service to generate personalized AI interview responses based on user profile data.
    - Create REST APIs to deliver AI responses to the frontend dashboard card.
    - Ensure secure access to user profile data (e.g., skills, job history) for personalization.
    - Optimize backend performance to handle AI response generation efficiently.
3. **Frontend Development (React Web & Mobile)**

    - Remove the chatbot widget from the webpage across web and mobile platforms.
    - Implement a new AI Interview Preparation card on the user dashboard.
    - Ensure the card design aligns with existing dashboard components (e.g., consistent fonts, colors, and layout).
    - Develop an interactive UI for users to engage with AI responses (e.g., question prompts, response display).
    - Ensure cross-platform consistency (React web and React Native mobile).
4. **Testing (QA)**

    - Verify the chatbot widget is fully removed and does not appear on any pages.
    - Test the new dashboard card for visibility, clarity, and functionality.
    - Validate that AI responses are personalized based on user profile data.
    - Ensure the card design is consistent with other dashboard components.
    - Perform usability testing to confirm improved user awareness and engagement.
    - Negative testing: Ensure the app handles missing profile data or API failures gracefully.
5. **Deployment & Monitoring**

    - Deploy the updated dashboard with the new AI Interview Preparation card.
    - Configure monitoring to track user engagement with the new card (e.g., clicks, time spent).
    - Set up analytics to measure the effectiveness of personalized AI responses.
    - Monitor for any issues related to the removal of the chatbot widget.
    - Collect user feedback post-deployment to assess clarity and satisfaction.

Blog Component Description for bitLabs Jobs App

Title: Technology Trends Blog Section

As a candidate using the bitLabs Jobs App, I want to access a dedicated blog section with regularly updated posts about trending technologies, AI advancements, and in-demand skills, so that I stay informed, enhance my learning, and engage more with the platform.

As an admin, I want to review and approve AI-generated blog content before publication, so that I can ensure high-quality, relevant, and accurate technology-focused content for users.

Detailed Description

The bitLabs Jobs App will introduce a new Blog Component to deliver engaging, technology-focused content to candidates. The blogs will cover the latest trends in technology, AI advancements, and in-demand skills relevant to job profiles, complementing the app's existing features (tests, badges, interview preparation, and push notifications for trending technologies). The content will be automatically generated by integrating multiple AI content generation models and news APIs to ensure diversity and reliability. An admin review workflow will ensure content quality before publication. The blog section will be accessible via a dedicated UI component, seamlessly integrated into the app's dashboard.

Goals

1. Increase User Engagement: Provide candidates with insightful blogs to keep them informed and encourage regular interaction with the app.
2. Automated Content Generation: Leverage multiple AI models (e.g., GPT-based models) and news APIs (e.g., TechCrunch, The Verge, or Google News with tech filters) to generate and aggregate relevant blog content.
3. Consistent Publishing: Publish two high-quality blogs per week, focusing on trending technologies, AI, and skills.
4. Admin Oversight: Implement a review workflow where admins can preview, approve, or reject AI-generated blogs to maintain quality and relevance.
5. Seamless UI Integration: Ensure the blog section is easily accessible from the dashboard with clear navigation and a user-friendly design.

Acceptance Criteria

1. **Blog Component Accessibility:**
    - A dedicated "Technology Blogs" section is added to the app, accessible via a clear navigation link from the dashboard.
    - The section is distinct from tests, badges, and interview preparation features to avoid confusion.
2. **Content Generation:**
    - Blogs are automatically generated using multiple AI content generation models and aggregated from reliable news APIs (not reliant on a single source).

    - Content focuses on technology trends, AI advancements, and in-demand skills relevant to job profiles.
    - Two blogs are generated and published per week (e.g., every Monday and Thursday).
3. **Admin Review Workflow:**
    - Admins can access a preview of each AI-generated blog in an admin panel.
    - Options to approve/finalize or reject blogs are available.
    - Only admin-approved blogs are visible to users.
4. **UI/UX Design:**
    - Blogs are displayed in a clean, readable format with title, publication date, tags (e.g., AI, Cloud Computing, Skills), and full content.
    - Responsive design ensures consistency across web and mobile platforms (React web and React Native mobile).
5. **Publishing Workflow:**
    - Automated generation of blogs occurs twice weekly, followed by admin review and approval before publishing.
    - Approved blogs are immediately visible in the user-facing blog section.
6. **User Experience:**
    - The blog section is clearly labeled as “Technology Blogs” to align with the app’s theme of delivering tech-focused updates.
    - Blogs complement daily push notifications by providing deeper insights into trending topics.

Subtasks Breakdown

1. **Requirement & Design**

- Identify reliable news APIs (e.g., TechCrunch, The Verge, Google News with tech filters) and AI content generation models (e.g., Grok, GPT-based APIs).
- Define filtering mechanisms to ensure content relevance (keywords: AI, machine learning, cloud computing, skills, etc.).
- Design a database schema to store blog content (e.g., title, content, tags, date, status: draft/approved/rejected).
- Create UI wireframes for the blog section (user-facing) and admin panel (review workflow).
- Document API endpoints for blog generation, admin review, and frontend integration.

2. Backend Development (Spring Boot)

- Implement a service to fetch and aggregate tech news from multiple APIs and generate blog content using AI models.
- Apply filtering logic to ensure relevance to technology, AI, and skills.
- Store generated blogs in the database with status tracking (draft/approved/rejected).
- Create a scheduler (e.g., Cron Job) to trigger blog generation twice per week.
- Develop REST APIs for:
    - Fetching blog content for the frontend.
    - Admin panel to preview, approve, or reject blogs.
- Ensure secure admin authentication for the review workflow.

3. Frontend (React Web & Mobile)

- Build a “Technology Blogs” section with a card/list layout (title, date, tags, short summary, “Read More” link).
- Implement navigation from the dashboard to the blog section.
- Create an admin panel UI for blog preview, approval, and rejection.
- Ensure cross-platform consistency (React for web, React Native for mobile).
- Integrate with push notification system to optionally notify users of new blog posts (if not muted).

4. Testing (QA)

- Verify blog content is relevant to technology, AI, and skills.
- Test automatic generation of two blogs per week.
- Validate admin review workflow (preview, approve, reject).
- Ensure only approved blogs are visible to users.
- Test navigation from dashboard to blog section.
- Verify responsive UI across web and mobile.
- Perform negative testing (e.g., handle API failures gracefully without crashing).

5. Deployment & Monitoring

- Deploy blog component and admin panel with existing app infrastructure.
- Configure monitoring for blog generation and publishing workflow (e.g., alerts for failed API calls or generation issues).
- Track user engagement metrics (e.g., blog views, time spent, click-through rates).
- Ensure integration with Firebase Cloud Messaging for optional blog post notifications.

User Story (bitLabs Jobs App – Push Notifications on Trending Technologies)

Title: Push Notifications for Trending Technologies & Skills

As a candidate using bitLabs Jobs app, I want to receive daily push notifications about rending technologies, AI advancements, and in-demand skills related to my job profile, so that I stay updated, learn new skills, and engage more with the application.

Detailed Description

The application should deliver curated technology news/updates to candidates. The news should come from reliable resources, either through subscriptions or filtering mechanisms, ensuring that only relevant technology-related content is shown.
Candidates should receive:

- A daily notification with a short headline/summary.
- A UI section in the app where detailed news/posts are displayed.
- An option to mute/unmute these notifications from their app settings.

Acceptance Criteria

1. Candidates receive daily notifications with technology-related updates.
2. Notifications contain short, concise text (headline + short description).
3. News displayed in-app is relevant to technology, AI, and skills (not general news).
4. Users can mute/unmute notifications in app settings.
5. If muted, users should not receive any push notifications until they unmute.
6. Content should refresh daily (at least once in 24 hours).
7. Notifications should be clickable, leading the user to the detailed news page.

Subtasks Breakdown

1. Requirement & Design

- Define reliable sources for trending technology news (e.g., APIs, RSS feeds, subscriptions).
- Decide filtering mechanism (keywords, categories, AI-based filtering).
- Create DB schema to store fetched news items.
- Design notification workflow (Scheduler/Trigger).
- Create UI wireframe for daily news section.
- Document API endpoints or RSS feeds for integration.

2. Backend Development (Spring Boot)

- Create a service to fetch daily tech news from identified resources.
- Implement filtering logic (only tech, AI, trending skills).

- Store curated news in the database (daily batch job).
- Implement scheduled trigger (e.g., Cron Job) to push notifications once per day.
- Provide REST APIs for the frontend to fetch news content.
- Add API for updating user notification preferences (mute/unmute).

3. Frontend (React Web & Mobile)

- Create news feed section (cards/list with title, short summary, link).
- Implement push notification UI on mobile (React Native / Expo / Firebase Cloud Messaging).
- Create mute/unmute toggle in settings.
- Show indicator if notifications are muted.
- Ensure cross-platform consistency (web + mobile).

4. Testing (QA)

- Verify only technology-related news is displayed.
- Verify push notifications arrive at the correct time.
- Verify mute/unmute works correctly across sessions.
- Test notification click → navigates to correct news detail page.
- Verify UI is responsive (mobile/web).
- Negative testing: when API fails, app should not crash.

5. Deployment & Monitoring

- Configure Firebase Cloud Messaging (FCM) for push delivery.
- Add monitoring/alerts for failed notifications.
- Track candidate engagement metrics (open rate, clicks).

Hackathon Module Description for bitLabs Jobs App

Title: Hackathon Module for Recruiters and Candidates

As a recruiter using the bitLabs Jobs App, I want to create and manage hackathons with detailed instructions and GitHub submission links, so that I can evaluate candidates based on real-world coding challenges and engage with talent effectively.

As a candidate, I want to discover and participate in hackathons tailored to my skills and interests, access clear instructions, and submit my solutions via GitHub, so that I can practice, showcase my abilities, and connect with recruiters.

Detailed Description

The bitLabs Jobs App will introduce a new Hackathon Module to enhance engagement and provide a structured platform for recruiters to host different hackathon challenges and for candidates to demonstrate their technical skills and problem solving skills. Recruiters can create hackathons with detailed specifications, including GitHub links for submissions, while candidates can discover relevant hackathons, register, and submit solutions. The module aims to foster community building, increase platform interaction, and align with the app's focus on trending technologies and skills.

Goals

1. Recruiter Empowerment: Enable recruiters to create and manage hackathons with clear instructions, evaluation criteria, and GitHub-based submissions to assess candidate skills.
2. Candidate Engagement: Provide candidates with a personalized list of hackathons matching their skills, interests, and job preferences, encouraging participation and skill demonstration.
3. Structured Workflow: Ensure a seamless process for hackathon creation, registration, participation, and submission, with secure data storage and clear UI navigation.
4. Platform Integration: Maintain consistency with the app's technology-focused theme and existing features, enhancing overall user engagement.

Acceptance Criteria

1. **Recruiter Side:**
    - Recruiters can create hackathons with details including title, description, start/end dates, required skills/technologies, instructions, evaluation criteria, and a GitHub repository link for submissions.
    - Hackathon data is securely stored in the database.
    - Recruiters can upload or define a documentation template/guide to assist candidates.
    - Instructions and submission processes are clearly published within the hackathon details.

2. **Candidate Side:**
    - Candidates see a personalized list of hackathons filtered by their profile (skills, interests, job preferences).
    - Candidates can register/apply for hackathons with a single click.
    - Hackathon details include clear instructions, rules, GitHub submission link, and documentation guide.
    - Candidates can submit solutions via the provided GitHub link and upload required documentation.
    - Submission status (e.g., applied, submitted, pending review) is visible in the candidate dashboard.
3. **General:**
    - The Hackathon Module is accessible from both recruiter and candidate dashboards with clear navigation.
    - UI/UX is consistent with the app's existing design (React web and React Native mobile).
    - Data (hackathon details, submissions) is securely persisted in the backend.
    - The module supports the app's theme of technology trends and skills, aligning with features like push notifications and blogs.

Subtasks Breakdown

1. **Requirement & Design**

- Define database schema to store hackathon data (e.g., title, description, dates, skills, GitHub link, documentation, status).
- Design filtering logic to match hackathons to candidate profiles (based on skills, interests, job preferences).
- Create UI wireframes for:
    - Recruiter dashboard: Hackathon creation and management.
    - Candidate dashboard: Hackathon discovery, registration, and submission.
- Document REST API endpoints for hackathon creation, registration, and submission.
- Specify integration with GitHub for solution submissions (e.g., validate repo links).

2. Backend Development (Spring Boot)

- Implement a service to allow recruiters to create and manage hackathons, storing details in the database.
- Develop logic to filter and recommend hackathons to candidates based on their profiles.
- Create REST APIs for:
    - Hackathon creation and management (recruiter).
    - Hackathon listing and registration (candidate).
    - Submission handling (GitHub link validation, documentation upload).
- Integrate with GitHub API to verify submission links.
- Ensure secure storage of sensitive data (e.g., candidate submissions).
- Implement status tracking for submissions (applied, submitted, pending review).

3. Frontend (React Web & Mobile)

- Build a “Hackathons” section in the candidate dashboard displaying a personalized list of hackathons (title, description, skills, dates).
- Implement hackathon registration and submission UI (GitHub link input, documentation upload).
- Create a recruiter dashboard section for hackathon creation (form with fields for title, description, dates, skills, GitHub link, documentation).
- Display submission status in the candidate dashboard.
- Ensure responsive design and cross-platform consistency (React for web, React Native for mobile).
- Integrate with push notification system to optionally notify candidates of new hackathons or submission updates (if not muted).

4. Testing (QA)

- Verify recruiters can create and manage hackathons with all required details.
- Test candidate hackathon discovery (personalized filtering based on skills/interests).
- Validate registration, submission process (GitHub link, documentation), and status updates.
- Ensure UI consistency across web and mobile platforms.
- Test navigation from dashboards to the Hackathon Module.
- Perform negative testing (e.g., invalid GitHub links, missing documentation, API failures).
- Verify integration with existing features (e.g., notifications, blogs).

5. Deployment & Monitoring

- Deploy the Hackathon Module with existing app infrastructure.
- Configure monitoring for hackathon creation, registration, and submission processes (e.g., alerts for failed API calls or invalid GitHub links).
- Track engagement metrics (e.g., hackathon registrations, submission rates, recruiter interactions).
- Ensure integration with Firebase Cloud Messaging for optional hackathon-related notifications.

1-Minute Video Engagement Module Description for bitLabs Jobs App

Title: Enhanced 1-Minute Video Engagement Module

As a candidate using the bitLabs Jobs App, I want to browse and watch 1-minute skill-based videos through an intuitive, visually appealing UI, so that I can quickly learn about trending technologies and skills relevant to my profile.

As a mentor, I want to record, edit, and upload 1-minute videos, so that I can contribute fresh, high-quality content to help candidates stay updated on technology trends.

As a developer, I want the backend to efficiently fetch and display videos based on user profiles, ensuring fast, seamless, and scalable performance for an optimal user experience.

Detailed Description

The bitLabs Jobs App will enhance its existing 1-Minute Video Engagement Proof of Concept (POC) into a fully integrated module to boost user engagement with short, skill-focused videos. The module will improve the UI/UX for video browsing and playback, introduce a mentor-driven content creation workflow, and optimize backend logic for faster, more relevant video delivery. Videos will continue to be stored in AWS S3 and linked in the database with tags and technologies, aligning with the app's focus on trending technologies, AI advancements, and in-demand skills. The module ensures personalized content delivery, seamless mentor contributions, and a scalable backend to enhance the overall user experience.

Goals

1. Improved User Experience: Redesign the video listing and playback UI for intuitive navigation, better filtering by tags/technologies, and clear progress tracking.
2. Mentor-Driven Content: Enable mentors to record, edit, and upload 1-minute videos, ensuring fresh, high-quality content aligned with trending technologies.
3. Backend Optimization: Enhance video fetching and database queries for faster retrieval, accurate skill-based matching, and smooth handling of watched videos.
4. Personalized Engagement: Deliver videos tailored to user profiles, with fallback to general videos when no skill match exists, maintaining alignment with the app's technology-focused theme.

Acceptance Criteria

1. **UI/UX Improvements:**
    - The video listing and playback UI is redesigned for a modern, user-friendly experience (e.g., card-based layout with thumbnails, titles, tags, and play button).
    - Users can filter videos by tags/technologies and track progress (e.g., watched/unwatched status).
    - UI is responsive and consistent across web (React) and mobile (React Native) platforms.
2. **Mentor-Recorded Videos:**
    - Mentors can record 1-minute videos through an in-app or external tool and upload them.

    - A workflow allows mentors to edit videos (e.g., trim, add captions) before publishing.
    - Edited videos are uploaded to AWS S3 and mapped in the database with relevant tags/technologies**.**
3. **Backend Enhancements:**
    - Video fetching logic is optimized for fast retrieval from AWS S3.
    - Database queries efficiently match videos to user profile skills and tags.
    - Watched videos automatically move down the list, prioritizing unwatched content.
    - Fallback logic displays general technology-focused videos when no skill match is found.
4. **General:**
    - The module integrates seamlessly with the app's dashboard, accessible via clear navigation (e.g., under "Skills & Achievements" or a dedicated section).
    - Data (video metadata, tags, watched status) is securely stored in the backend.
    - The module aligns with the app's theme of delivering tech-focused content, complementing push notifications, blogs, and hackathons.
    - QA testing ensures smooth UI/UX, backend performance, and mentor upload workflow.

Subtasks Breakdown

1. Requirement & Design

- Define UI wireframes for the redesigned video listing and playback interface (e.g., card layout, filters, progress indicators).
- Design a mentor workflow for recording, editing, and uploading videos (e.g., integration with a video editor or simple in-app tools).
- Update the database schema to support video metadata (e.g., title, tags, technologies, watched status, mentor details).
- Specify AWS S3 storage configuration for secure video uploads.
- Document REST API endpoints for video fetching, mentor uploads, and user interaction tracking.
- Define fallback logic for general video display when no skill match exists.

2. Backend Development (Spring Boot)

- Optimize video fetching logic for faster retrieval from AWS S3.
- Enhance database queries to match videos with user profile skills and tags efficiently.
- Implement logic to track watched videos and reorder the list (push watched videos down).
- Develop a fallback mechanism to serve general technology videos when no skill match is found.
- Create a service for mentors to upload videos to S3 and store metadata in the database.
- Implement REST APIs for:
    - Fetching video lists (filtered by skills/tags).

    - Tracking watched status and updating the list order.
    - Mentor video uploads and metadata management.
- Ensure secure handling of video data and user interactions.

3. Frontend (React Web & Mobile)

- Redesign the video listing UI with a card-based or list-based layout (e.g., thumbnail, title, tags, play button).
- Implement video playback with smooth controls (play, pause, seek) and progress tracking (watched/unwatched).
- Add filtering options by tags/technologies for personalized browsing.
- Create a mentor interface for recording/uploading videos (e.g., file picker or integration with a recording tool).
- Implement an editing interface for mentors (e.g., basic trimming, captions) or integrate with an external editor.
- Ensure cross-platform consistency (React for web, React Native for mobile).
- Integrate with push notifications to optionally alert users about new videos (if not muted).

4. Testing (QA)

- Verify the redesigned UI/UX is intuitive, responsive, and consistent across web and mobile.
- Test video playback for smooth performance and accurate progress tracking.
- Validate mentor video recording, editing, and upload workflow.
- Ensure videos are correctly stored in AWS S3 and mapped in the database with tags/technologies.
- Test backend logic for optimized video fetching and skill-based matching.
- Verify watched videos move down the list and general videos display as a fallback.
- Perform negative testing (e.g., handle S3 failures, invalid video formats, or API errors).

5. Deployment & Monitoring

- Deploy the enhanced video module with existing app infrastructure.
- Configure AWS S3 for secure and scalable video storage.
- Set up monitoring for video fetching, upload processes, and playback performance (e.g., alerts for S3 or API failures).
- Track engagement metrics (e.g., video views, completion rates, filter usage).
- Ensure integration with Firebase Cloud Messaging for optional video-related notifications.

Mentorship Connect Meet Module

Title: Mentorship Connect Meet with Google Meet Integration

As a mentor using the bitLabs Jobs App, I want to schedule live mentorship sessions with Google Meet links and share them with candidates, so that I can provide training and guidance on specific technologies in an interactive format.

As a candidate, I want to discover and register for mentorship sessions, receive a Google Meet link via email, and be auto-admitted as a registered participant, so that I can learn, clarify doubts, and engage directly with mentors.

As a system, I want to integrate with Google Calendar and Google Meet to manage session scheduling, ensure only registered users are invited and auto-admitted, and maintain security and engagement.

Detailed Description

The bitLabs Jobs App will introduce a Mentorship Connect Meet Module to enable direct, live interactions between mentors and candidates through Google Meet. Mentors can schedule sessions focused on specific technologies, generating Google Meet links and integrating with Google Calendar for seamless event management. Candidates can discover and register for sessions via the app's dashboard, receiving email confirmations with meeting links and gaining auto-admission to ensure secure access.

Goals

1. Mentor-Candidate Interaction: Enable mentors to schedule and conduct live sessions, fostering direct training and engagement on trending technologies and skills.
2. User Accessibility: Allow candidates to easily discover, register for, and join mentorship sessions relevant to their profiles.
3. Seamless Integration: Leverage Google Meet and Google Calendar for secure, automated session management and attendee control.
4. Enhanced Engagement: Increase platform interaction by offering live Q&A and training, complementing the app's focus on skill development and tech trends.

Acceptance Criteria

1. **Mentor Scheduling:**
    - Mentors can schedule sessions with details including title, description, technology focus, date, and time.
    - A Google Meet link is generated and stored in the database with session details.
    - The session is automatically created as an event in Google Calendar.
2. **Candidate Experience:**
    - Available mentorship sessions are displayed in the candidate dashboard, filtered by technology or user profile (skills/interests).
    - Candidates can register for sessions with a single click, with registrations stored in the database.
    - Registered candidates receive an email confirmation containing the Google Meet link.
3. **System Integration:**

    - Google Calendar events are created with only registered users added as attendees.
    - Only registered users are auto-admitted to Google Meet sessions; others cannot join directly with the link.
4. **General:**
    - The Mentorship Connect Meet Module is accessible from both mentor and candidate dashboards with clear navigation.
    - UI/UX is consistent with the app's existing design (React web and React Native mobile).
    - Data (session details, registrations) is securely stored in the backend.
    - The module aligns with the app's theme of delivering tech-focused learning, integrating with features like "Skills & Achievements," blogs, and hackathons.
    - QA testing ensures seamless scheduling, registration, email delivery, and auto-admission workflows.

Subtasks Breakdown

1. Requirement & Design

- Define database schema to store mentorship session data (e.g., title, description, technology, date, time, Google Meet link, registered users).
- Design UI wireframes for:
    - Mentor dashboard: Session creation form.
    - Candidate dashboard: Session listing and registration interface.
- Specify integration with Google Meet and Google Calendar APIs for link generation and event management.
- Define security measures to ensure only registered users are auto-admitted to sessions.
- Document REST API endpoints for session scheduling, registration, and email notifications.
- Design email template for registration confirmations with Google Meet links.

2. Backend Development (Spring Boot)

- Implement a service for mentors to schedule sessions, storing details in the database.
- Integrate Google Meet API to generate unique meeting links for each session.
- Integrate Google Calendar API to create events and add registered users as attendees.
- Develop logic to filter and display sessions based on candidate profiles (skills, interests, job preferences).
- Create REST APIs for:
    - Session creation and management (mentor).
    - Session listing and registration (candidate).
    - Email notifications with Google Meet links.
- Implement security to restrict Google Meet access to registered users only.
- Store registration data securely in the database.

3. Frontend (React Web & Mobile)

- Build a “Mentorship Connect” section in the candidate dashboard displaying available sessions (title, description, technology, date, time).
- Implement a registration button for candidates to join sessions.
- Create a mentor interface for scheduling sessions (form with fields for title, description, technology, date, time).
- Display confirmation of session registration and link access in the candidate dashboard.
- Ensure responsive design and cross-platform consistency (React for web, React Native for mobile).
- Integrate with push notifications to optionally alert candidates about new sessions or registration confirmations (if not muted).

4. Testing (QA)

- Verify mentors can schedule sessions with all required details and generate Google Meet links.
- Test Google Calendar integration (event creation, attendee addition).
- Validate candidate session discovery and registration process.
- Ensure email confirmations with Google Meet links are delivered correctly.
- Test auto-admission for registered users and access restrictions for non-registered users.
- Verify UI consistency across web and mobile platforms.
- Perform negative testing (e.g., handle invalid dates, API failures, or unauthorized access attempts).
- Ensure integration with existing features (e.g., notifications, “Skills & Achievements”).

5. Deployment & Monitoring

- Deploy the Mentorship Connect Meet Module with existing app infrastructure.
- Configure Google Meet and Google Calendar API integrations for secure and scalable performance.
- Set up monitoring for session scheduling, registration, and email delivery (e.g., alerts for API failures or undelivered emails).