Cohort A - Subjective question examples

Question 1: Application Architecture and Tech Stack Planning (~35 minutes)

Scenario:
You have been approached by a local fitness studio chain (5 locations) to build a comprehensive booking and membership management system. The studio owner has the following requirements:

- Members should be able to browse class schedules across all locations, book classes, and cancel bookings
- Each location has different class types, instructors, and time slots
- Members need profiles showing their booking history, membership tier (Basic, Premium, Elite), and expiry dates
- Studio managers at each location should be able to create/edit classes, view attendance, and manage capacity limits
- The main administrator needs a dashboard showing revenue metrics, popular classes, and member retention statistics
- The system should send automated email reminders 24 hours before booked classes
- Mobile responsiveness is critical as most bookings happen on phones
- Budget is moderate, and they want to launch within 6 weeks

Your Task:
Design a complete technical architecture for this application. Your response should include:

1. **Platform Selection and Justification**: Choose the appropriate platform(s) from the tools covered in the course (Webflow, Bubble, Lovable, or a combination). Provide detailed reasoning for your choice, explaining why alternatives would be less suitable for this specific use case.
2. **Data Schema Design**: Create a detailed data model showing all necessary tables (data types), their fields with appropriate field types, and the relationships between them. Explain your reasoning for how you've structured the data and any design decisions you made to handle the multi-location complexity.
3. **User Flow and Permissions Architecture**: Describe how you would implement the three different user roles (members, location managers, main administrator) and what permissions

each would have. Map out the critical user journeys for at least two personas.
4. **Integration Strategy**: Identify what third-party integrations or tools would be needed (payment processing, email automation, etc.) and explain how you would connect them to your chosen platform.
5. **Scalability and Risk Assessment**: Discuss at least three potential technical challenges or limitations you might encounter with your chosen approach, and how you would mitigate them.

Deliverable: A comprehensive written document (file upload recommended) with diagrams where appropriate. You may sketch data models, flowcharts, or architecture diagrams by hand and include photos, or create them using any tool you prefer.

Ideal Solution:

A strong answer would demonstrate the following:

Platform Choice: Bubble would likely be the most appropriate choice here because this requires a full-stack application with complex user roles, dynamic data relationships, and workflow automation. The student should explain that while Webflow excels at marketing sites, it lacks native database and user authentication capabilities. Lovable could work but might be overkill for this scope, and the AI-generated code might require more technical maintenance than the client wants. The answer should show understanding of each platform's strengths and constraints.

Data Schema: The data model should include at minimum these tables with thoughtful relationships:

- Users (with role field: member/manager/admin, linked to membership tier)
- Membership Tiers (Basic/Premium/Elite with benefits)
- Locations (name, address, manager assigned)
- Class Types (yoga, spin, HIIT, etc.)
- Classes (specific instances with date, time, instructor, location, capacity)
- Bookings (junction table linking Users and Classes, with status field)
- Instructors (linked to locations)

The student should explain one-to-many and many-to-many relationships, such as how one User can have many Bookings, but one Class can also have many Bookings. They should recognize the need for capacity management and status tracking.

Permissions Architecture: The answer should detail role-based access control where members can only see/modify their own bookings, location managers can manage classes at their assigned location only, and the admin has global access. Critical user journeys should map the complete flow

from browsing classes through booking confirmation for a member, and from class creation through viewing attendance for a manager.

Integration Strategy: Should identify needs for payment gateway (Stripe), email automation (SendGrid or native Bubble email), and possibly calendar integration. The student should explain whether to use Bubble's API Connector, Zapier/Make, or native plugins, demonstrating understanding of when each approach is appropriate.

Scalability Assessment: Thoughtful answers might address concerns like Bubble's performance with large data sets as the member base grows, the need for database optimization through proper indexing, potential need for Redis caching for frequently-accessed class schedules, managing real-time booking conflicts when multiple users book the last spot simultaneously, and the cost scaling of the chosen platform as usage increases. Strong students will propose specific mitigation strategies like implementing database searches efficiently, using scheduled workflows for non-urgent tasks, or architecting for a potential future migration path.

The answer should be 4-6 pages of detailed written analysis, showing systems thinking rather than just feature lists.

Question 2: Data Operations and Workflow Engineering (~40 minutes)

Scenario:
You're building a collaborative project management tool for creative agencies. The tool needs to support the following workflow:

A creative agency works on multiple client projects simultaneously. Each project goes through stages: Brief → Concept → Design → Review → Revision → Final → Delivered. Each project has multiple deliverables (logo, website, marketing materials, etc.). Team members have different roles (Creative Director, Designer, Copywriter, Account Manager). When a deliverable moves to "Review" stage, the assigned Account Manager should receive a notification, and the client should get an automated email with a secure link to view the deliverable. If the client requests revisions, a new task should automatically be created and assigned to the original designer. The Creative Director needs a real-time dashboard showing all projects, which are at risk of missing deadlines, and each team member's current workload.

Your Task:

1. **Comprehensive Data Model**: Design the complete database schema for this system. Identify all necessary tables, fields (with specific field types), and relationships. Pay special attention to how you'll track project stages, deliverable status, team member assignments, and revision history. Explain your reasoning for complex decisions like whether to use status fields versus separate tables, or how to model the relationship between projects, deliverables, and team members.
2. **CRUD Operations Mapping**: For each of the following user actions, describe the complete data operations that need to occur in the database:
    - Creating a new project with three deliverables
    - Moving a deliverable from "Design" to "Review" stage
    - A client requesting revisions on a deliverable
    - Deleting a team member who has been assigned to active projects
3. **Workflow Automation Design**: Describe in detail how you would implement the automated workflows mentioned in the scenario (notifications, email sending, automatic task creation). For a visual platform like Bubble, explain the trigger conditions, actions in sequence, and what data would be passed between steps. Consider edge cases like what happens if an Account Manager isn't assigned yet, or if multiple revisions are requested simultaneously.
4. **Dashboard Data Logic**: Explain how you would calculate and display the following on the Creative Director's dashboard:
    - Projects at risk (approaching deadline with incomplete deliverables)
    - Team member workload (how many active deliverables each person is assigned to)
    - Project health scoring based on multiple factors

    Describe what database searches or filters you'd need to perform, and whether you'd calculate these in real-time or cache them.
5. **Data Integrity and Edge Cases**: Identify at least four potential data integrity issues or edge cases in this system and explain how your data model and workflows would prevent or handle them.

Deliverable: Written document with clear sections for each component. Include visual diagrams of your data model and workflow sequences where helpful.

Ideal Solution:

Data Model: A comprehensive schema should include:

Tables:

- Projects (name, client, start_date, deadline, status, creative_director_assigned)

- Deliverables (name, description, project_relationship, current_stage, assigned_designer, assigned_copywriter, due_date)
- Stages (this could be a static option set or separate table: Brief, Concept, Design, Review, Revision, Final, Delivered)
- Team_Members (name, role, email, active_status)
- Clients (company_name, contact_person, email)
- Revisions (deliverable_relationship, requested_by, requested_date, notes, resolved_status, assigned_to)
- Stage_History (deliverable_relationship, from_stage, to_stage, changed_date, changed_by) - for audit trail
- Assignments (junction table linking Team_Members to Deliverables with role specification)

The student should explain why they chose to track revisions separately versus as status changes, how they handle many-to-many relationships between deliverables and team members, and why audit trails matter for project management.

CRUD Mapping: For creating a new project with three deliverables, the answer should outline the step-by-step database writes: first creating the Project record, then three Deliverable records each linked to the Project, creating initial Stage_History entries for each deliverable, and potentially creating Assignment records linking team members to each deliverable.

For moving a deliverable to "Review", they should describe: updating the Deliverable's current_stage field, creating a Stage_History record, searching for the Account Manager assigned to this project's client, triggering a notification workflow to that Account Manager, and sending an email to the Client with dynamically generated secure link.

For client revision requests, they should describe creating a new Revision record, updating the Deliverable status back to "Revision" stage, creating a new Assignment linking the original designer to this revision work, and triggering notifications.

For deleting a team member, they should identify the challenge (what happens to their assignments?) and propose solutions like reassignment workflows or soft-delete with an "inactive" status instead of true deletion.

Workflow Automation: Strong answers will show understanding of conditional logic and sequential processing. For the Review stage trigger, they should explain: When Deliverable's current_stage changes to "Review", search for Account_Manager assigned to this Deliverable's Project, if found then create Notification record and send in-app notification, search for Client linked to this Project, generate secure viewing URL (perhaps using a unique token), compose email with deliverable details

and link, send email. They should address edge cases like missing Account Manager assignments with fallback notifications to Creative Director.

Dashboard Logic: For at-risk projects, they should describe a search query that finds Projects where deadline is within 7 days AND there exists at least one Deliverable where current_stage is not "Delivered". For team workload, they should explain counting active Deliverable assignments per Team_Member where status is not Final or Delivered. They should discuss whether to calculate these in real-time (slower but always current) or use scheduled workflows to cache counts (faster dashboard loading but potentially stale data), and justify their choice.

Data Integrity: Good edge cases to address include:

- What if a team member is deleted mid-project? (Solution: prevent deletion if active assignments exist, or require reassignment first)
- What if multiple users try to change the same deliverable status simultaneously? (Solution: timestamp-based conflict resolution or last-write-wins with notification)
- What if a revision is requested but the original designer has left the company? (Solution: automatic reassignment to Creative Director or another team member of same role)
- What if a client email bounces? (Solution: flag the Client record and notify Account Manager)

The answer should be 5-7 pages showing deep understanding of relational data design and workflow automation beyond just listing features.

Question 3: Comparative Analysis and Critical Decision-Making (~50 minutes)

You are a technical consultant, and three different clients have approached you. Each has a different project with specific requirements, constraints, and goals. Your expertise in visual development platforms is critical to their success.

Client A - SaaS Startup:
A startup wants to build an MVP (Minimum Viable Product) for a SaaS tool that helps remote teams coordinate work schedules across timezones. They need user authentication, a database to store team members and schedules, ability to integrate with Google Calendar, payment processing for subscriptions (3 tiers: Free, Pro, Enterprise), and a clean responsive interface. They have $5,000 budget, need to launch in 4 weeks to pitch to investors, and their founding team has no technical background but they're willing to learn and iterate quickly based on user feedback.

Client B - Enterprise Marketing Site:
A well-established law firm with 50 attorneys wants to completely redesign their website. They need pixel-perfect design matching their premium brand, attorney profile pages (pulling from a database of 50+ lawyers with photos, bios, specializations), a blog/news section they can update regularly, a complex practice area section with filterable case studies, contact forms that integrate with their existing Salesforce CRM, and excellent SEO. They have a $30,000 budget, need launch in 8 weeks, and will hire a marketing team member to maintain content long-term. Page load speed and professional appearance on all devices is critical.

Client C - Internal Business Tool:
A manufacturing company wants to digitize their inventory and equipment maintenance tracking. Currently they use Excel spreadsheets and paper forms. They need a system where warehouse staff can scan equipment QR codes on tablets, log maintenance performed, track part inventory levels, set automated reminders for scheduled maintenance, and generate reports for management. It will have 25 daily users (warehouse staff and managers), needs to work reliably offline in warehouse areas with poor connectivity, and must integrate with their existing email system. They have $10,000 budget, flexible timeline of 12 weeks, and an IT manager who can handle basic technical tasks but not full programming.

Your Task:

For each client, provide a comprehensive recommendation that includes:

1. **Platform Selection with Detailed Justification**: Choose the most appropriate platform(s) from those covered in the course (Webflow, Bubble, Lovable, Replit, Cursor, or a combination). Provide specific reasoning why this platform best matches their requirements, budget, timeline, and technical capabilities. Also explicitly explain why the other platforms would be unsuitable for this particular client.
2. **Implementation Approach**: Outline the high-level development strategy - what you would build first, how you would structure the project phases, and how you would handle their timeline constraints.
3. **Risk Assessment and Mitigation**: Identify the top 3 risks specific to this client and your chosen platform, and propose concrete mitigation strategies.
4. **Long-term Viability Analysis**: Discuss the sustainability of your solution - what happens 6 months, 1 year, or 2 years down the line? Address concerns like platform lock-in, scaling limitations, maintenance requirements, and potential migration needs.
5. **Alternative Approach**: For at least one client, describe a secondary approach using a different platform or combination of platforms. Explain the trade-offs between your primary recommendation and this alternative.

Deliverable: A detailed consulting report addressing all three clients. This should read like professional technical consulting documentation.

Ideal Solution:

Client A - SaaS Startup:

Platform Choice: Bubble is the optimal choice. The student should explain that Bubble natively supports user authentication, has a robust database with proper relational structures, includes API connections for Google Calendar integration, has Stripe payment plugins for subscription handling, and can handle all the full-stack requirements of a SaaS MVP. The responsive engine allows for good UI across devices.

They should explain why alternatives fall short: Webflow lacks native user authentication and database capabilities for this complex application logic; Lovable could work but the AI-generated code would require technical knowledge to maintain and customize that the non-technical team lacks; Cursor/Replit would require actual coding which contradicts their "no technical background" constraint.

Implementation: Phase 1 (week 1-2) would focus on core user flows and authentication, basic schedule data model, and simple interface. Phase 2 (week 2-3) adds Google Calendar sync and timezone logic. Phase 3 (week 3-4) implements payment subscription tiers and polishes UI for investor demo. This phased approach gives them something demonstrable early even if all features aren't complete.

Risks: (1) Google Calendar API integration might be complex - mitigation: use Zapier as intermediary layer if direct API proves difficult. (2) Bubble's learning curve for non-technical founders - mitigation: recommend Bubble's extensive tutorials and consider 10 hours of consultant time to teach them. (3) Performance issues if they scale to thousands of users - mitigation: architect database efficiently from start, use proper indexing, and acknowledge that beyond 10,000 users they may need to migrate.

Long-term: Bubble is viable for 1-2 years of growth. Once they have proven product-market fit and 5,000+ users, they should plan for a migration to custom code, but this is actually ideal for an MVP strategy. Monthly costs will scale with usage (~$30-300/month depending on growth). Platform lock-in is a risk, but the alternative of custom development would prevent them from launching in 4 weeks at all.

Client B - Law Firm Website:

Platform Choice: Webflow is the clear winner. Students should explain that Webflow excels at pixel-perfect design with complete visual control, has a powerful CMS for attorney profiles and blog posts, allows custom code for Salesforce integration, produces clean SEO-friendly code, and generates fast-loading production sites. The visual editor gives the marketing team member an intuitive way to maintain content without developer dependency.

Why alternatives fail: Bubble focuses on application logic over design precision and produces slower-loading sites; Lovable/Cursor would require coding expertise they don't have; Replit is for prototyping not production websites.

Implementation: Weeks 1-2: Design system and template creation. Weeks 3-4: Attorney profile CMS structure and individual pages. Weeks 5-6: Practice areas, case studies with filtering logic, blog setup. Weeks 7-8: Salesforce integration (custom code), SEO optimization, final QA and launch. This timeline is tight but achievable with focused execution.

Risks: (1) Salesforce integration complexity - mitigation: hire a Webflow specialist for the integration component (budget allows). (2) CMS limitations for very complex filtering - mitigation: supplement with custom JavaScript if needed. (3) Client revision requests extending timeline - mitigation: clearly defined approval gates and change order process.

Long-term: This solution should last 3-5+ years with regular content updates. Webflow's hosting is reliable and scales well. No migration needed unless they want to add complex application features later (unlikely for a law firm site). Monthly cost is predictable (~$50-200 depending on hosting tier). The marketing team can manage content indefinitely.

Alternative Approach: Could use Cursor to build a custom Next.js site with better performance and no platform fees, but this would require ongoing developer maintenance that contradicts their "marketing team member" maintenance plan. The tradeoff is performance and long-term cost versus ease of content management and timeline.

Client C - Manufacturing Inventory Tool:

Platform Choice: This is the trickiest scenario. Students should recognize that offline functionality is a major constraint that most visual platforms handle poorly. Bubble has limited offline capabilities. The best approach is likely a **hybrid: Bubble for the web-based management dashboard and reports** combined with **a Progressive Web App (PWA) built in Lovable or with Cursor** for the tablet-based warehouse scanning functionality that works offline and syncs when connected.

Alternatively, they could recommend Bubble alone with the explicit caveat that offline mode means data is cached locally in the browser but might have sync conflicts, or they could suggest this is a

case where the client actually needs custom development and visual platforms aren't ideal - which shows mature critical thinking.

Implementation: If choosing hybrid approach: Weeks 1-4: Build core database and management dashboard in Bubble. Weeks 5-8: Develop offline-capable tablet scanning interface with Lovable, implementing local storage and sync logic. Weeks 9-12: Integration testing, QR code scanning testing, deployment, and training.

Risks: (1) Offline sync conflicts when multiple warehouse staff edit same equipment - mitigation: timestamp-based conflict resolution and clear UI indicators. (2) QR scanner not working reliably on all tablet browsers - mitigation: extensive device testing early, possibly native app wrapper. (3) Email integration complexity - mitigation: use Zapier/Make for scheduled maintenance reminders.

Long-term: This solution has sustainability concerns. The hybrid approach creates maintenance complexity. After 12-18 months, if the system proves valuable, they should invest in native mobile app development. The IT manager can handle Bubble maintenance but might struggle with the Lovable/custom code portions.

The answer should show sophisticated analysis that acknowledges when visual development platforms have genuine limitations and when hybrid or alternative approaches are necessary. A truly excellent answer might even recommend that Client C's requirements make this a borderline case where traditional development might be more sustainable, while still providing a visual-development approach that could work.

The full answer should be 6-8 pages showing nuanced understanding of platform capabilities, constraints, business context, and technical tradeoffs.

Question 4: Integration Architecture and Production Deployment Strategy (~55 minutes)

Scenario:
You have built a multi-platform solution for an e-learning startup. The architecture consists of:

- **Marketing website** (Webflow): Public-facing site with course catalog, instructor bios, blog, and lead capture forms
- **Learning application** (Bubble): Student dashboard where users access courses, track progress, submit assignments, and communicate with instructors

- **Content management** (Airtable): Database storing course content, lesson videos, quizzes, student records, and instructor information
- **Video hosting** (Vimeo): All course videos are hosted here
- **Payment processing** (Stripe): Handles course purchases and subscription management
- **Email communication** (SendGrid): Sends automated emails for course reminders, assignment deadlines, and marketing

The system currently works in development, but now you need to deploy it to production and ensure all components work together reliably.

Your Task:

1. **Integration Architecture Documentation**: Create a comprehensive map of how all these components connect and communicate with each other. For each integration point, specify:
   - What triggers the communication (user action, scheduled workflow, webhook, etc.)
   - What data is being passed between systems
   - Which integration method you would use (native API connector, Zapier/Make, webhooks, embedded code)
   - What happens if the integration fails

   For example, when a user purchases a course on the Webflow site, trace the complete data flow through every system.
2. **Deployment Strategy and Checklist**: Develop a detailed deployment plan for moving this entire ecosystem from development to production. Address:
   - Order of deployment (which components go live first and why)
   - Environment configuration (development vs. production settings for each platform)
   - DNS and domain configuration strategy
   - SSL certificates and secure communication
   - How you'll handle the transition without breaking existing test data or user access
   - Rollback plan if something goes wrong during deployment
3. **Testing Protocol**: Design a comprehensive testing protocol that verifies all integrations work correctly in production. Include:
   - Specific test cases for critical user journeys
   - How you'll test payment flows without charging real money
   - How you'll verify email delivery
   - Performance and load testing considerations
   - Mobile responsiveness verification across key flows
4. **Monitoring and Maintenance Plan**: Propose an ongoing monitoring strategy for production, including:
   - What key metrics you would track for system health

- How you would detect integration failures or performance degradation
- Backup and data recovery procedures
- Version control strategy for visual platforms
- Who needs access to which systems and what permission levels

5. **Scaling and Optimization Analysis**: Looking 6 months ahead when the startup has 1,000 active students instead of 100, identify:
   - Potential bottlenecks in your current architecture
   - Which integrations might fail or slow down at scale
   - What optimizations you would implement proactively
   - At what point you might need to re-architect components
   - Cost scaling implications across all platforms

Deliverable: A detailed technical operations document suitable for handoff to a technical operations person. Include diagrams, checklists, and specific configuration details.

Ideal Solution:

Integration Architecture:

A comprehensive answer would map all integration touchpoints. For example:

Course Purchase Flow:
User submits payment on Webflow → Form triggers Zapier webhook → Zapier creates Stripe checkout session → On successful payment, Stripe webhook fires → Zapier receives webhook → Zapier creates student record in Bubble database AND creates record in Airtable AND enrolls user in course → Bubble workflow triggers SendGrid email via API with welcome message and login credentials.

The student should identify that Webflow doesn't have native database, so form submissions need to flow through Zapier. Bubble can use API Connector for Stripe webhooks and SendGrid. Airtable could sync with Bubble via API or Zapier depending on data volume and real-time requirements. Vimeo videos would be embedded in Bubble pages using Vimeo's embed codes with privacy settings configured.

For failure handling, they should propose: retry logic in Zapier (built-in), error notifications to admin email, manual reconciliation process for failed webhook payments, and logging of all integration attempts.

Course Progress Tracking:
Student watches video in Bubble → On video completion, Bubble updates progress record in own database → Daily scheduled workflow syncs completion data to Airtable for reporting → If student completes all lessons, workflow triggers SendGrid certificate email.

Deployment Strategy:

A strong answer would propose this sequence:

1. **Airtable first** (configure production base, migrate essential data, set up API keys)
2. **Bubble application** (deploy to live, configure production database, set environment variables for API keys, test authentication)
3. **Stripe configuration** (switch from test mode to live mode, update webhooks to production URLs)
4. **SendGrid setup** (verify domain for email sending, configure production templates)
5. **Zapier workflows** (clone development zaps, update with production API keys and URLs, test each individually)
6. **Webflow last** (publish to custom domain, update form actions to production Zapier URLs, configure SSL)

They should explain why this order matters: dependent systems need foundation systems deployed first. Each platform needs specific production configurations:

- Bubble: environment variables for API keys, live database separated from development, Bubble's "live" version published, custom domain configured
- Webflow: custom domain DNS pointed to Webflow, SSL auto-generated by Webflow
- Stripe: switch to live publishable and secret keys, update webhook endpoints to production URLs
- SendGrid: domain authentication (SPF, DKIM records), production API key

For transition without disruption: deploy outside business hours, maintain parallel development environment, use database backup before migration, test with small user group first (soft launch).

Rollback plan: maintain development environment accessible, have database backups, ability to revert Bubble to previous version, keep old Zapier zaps paused not deleted.

Testing Protocol:

Test cases should include complete end-to-end scenarios:

1. *Course Purchase Journey:* Create test account → Browse course on Webflow → Click purchase → Complete Stripe checkout with test card (4242 4242 4242 4242) → Verify student record created in Bubble → Verify enrollment in Airtable → Verify welcome email received → Log into Bubble → Access course content → Verify video plays → Mark lesson complete → Verify progress saved.
2. *Instructor Assignment Upload:* Instructor logs into Bubble → Uploads assignment → Verify student receives email notification → Student submits assignment → Instructor receives notification → Grades assignment → Student receives grade notification.

For payment testing, use Stripe test mode with test card numbers. For email, use real email addresses of test team or email testing tools like Mailtrap. For performance, use browser developer tools to check page load times, ensure all pages load under 3 seconds. Test on mobile devices (iPhone, Android) and tablets, verify responsive design works.

Load testing considerations: Bubble has usage limits on lower tier plans, Zapier has task limits, consider what happens with 100 simultaneous users. May need to upgrade plans or optimize database searches.

Monitoring Plan:

Key metrics to track:

- Uptime monitoring for Webflow, Bubble (use UptimeRobot or Pingdom)
- Zapier task success rate (check Zapier dashboard daily)
- Stripe payment success vs. failure rate
- SendGrid email delivery rate and bounce rate
- Bubble database capacity usage
- Airtable API usage limits
- Student login frequency and course completion rates

Integration failure detection:

- Zapier sends error emails automatically
- Set up Bubble workflows to alert admin if critical processes fail
- Monitor Stripe dashboard for webhook failures
- Check SendGrid for blocked emails

Backup procedures:

- Bubble: export database weekly (CSV backups)
- Airtable: use automated backup tool or API scripts

- Webflow: download site backup monthly
- Git repository for any custom code

Version control:

- Bubble: use built-in version history and restore points before major changes
- Webflow: backup before publishing major changes
- Airtable: duplicate base before schema changes
- Document all integration configuration changes

Access control:

- Bubble: Admin (full access), instructors (content management), students (limited to own data)
- Webflow: Designer access for marketing team, no student access
- Airtable: view-only for instructors, edit for content team
- Stripe: admin only
- SendGrid: admin and marketing team

Scaling Analysis:

Bottlenecks at 1,000 students:

1. *Bubble database performance:* Searches become slower with more records. Mitigation: implement database indexing, optimize searches to filter on indexed fields, consider Bubble's capacity plan upgrades.
2. *Zapier task limits:* Free tier is 100 tasks/month. At 1,000 students, course purchases alone could exceed this. Mitigation: upgrade to paid Zapier plan ($20-50/month) or reduce Zapier dependency by using Bubble's API Connector for direct integrations.
3. *Video bandwidth:* Vimeo's basic plan has bandwidth limits. 1,000 students watching hours of video could exceed limits. Mitigation: upgrade Vimeo plan or migrate to Cloudflare Stream or AWS S3 + CloudFront.
4. *Email sending limits:* SendGrid free tier limits daily emails. Mitigation: upgrade to paid SendGrid plan before hitting limits.
5. *Airtable API rate limits:* 5 requests per second. If Bubble hits this during peak usage, requests fail. Mitigation: implement request queuing in Bubble, batch updates, or cache data locally in Bubble database.

Proactive optimizations:

- Set up database indexing in Bubble from day one
- Implement lazy loading for course lists (pagination)

- Cache frequently accessed data in Bubble rather than API calls to Airtable
- Optimize images and assets for faster loading
- Consider CDN for static assets

Re-architecture trigger points:

- When Bubble workload units consistently hit limits (>10,000 students)
- When Airtable becomes limiting factor (Airtable not designed for production apps at scale, migrate to proper database)
- When custom features require coding beyond visual platform capabilities

Cost scaling:

- Bubble: $29/month currently \rightarrow$119-349/month at scale
- Zapier: $0 \rightarrow$20-50/month
- Vimeo: $20/month \rightarrow$75-240/month
- Airtable: $20/user/month for team features
- SendGrid: $0 \rightarrow$15-50/month
- Stripe: 2.9% + 30¢ per transaction (scales with revenue, actually good)
- Total: ~$100/month now \rightarrow$ 500-800/month at 1,000 students

The answer should be 7-9 pages showing sophisticated understanding of production operations, not just development.