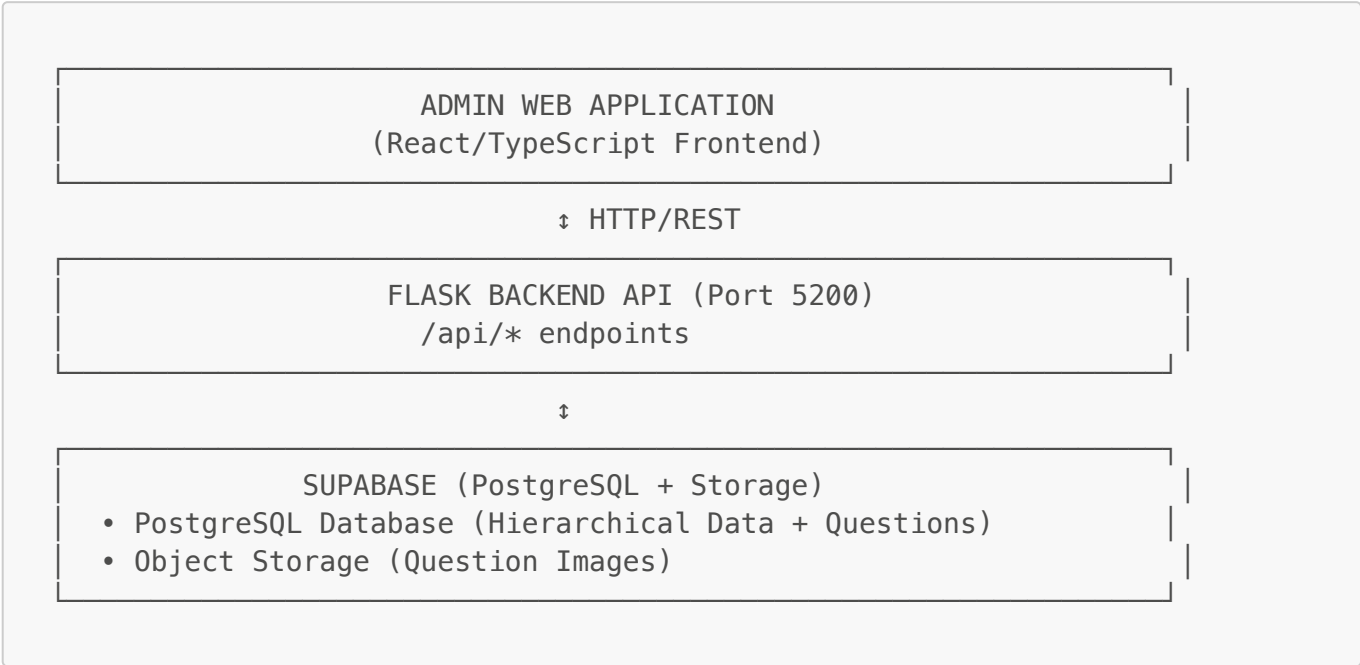
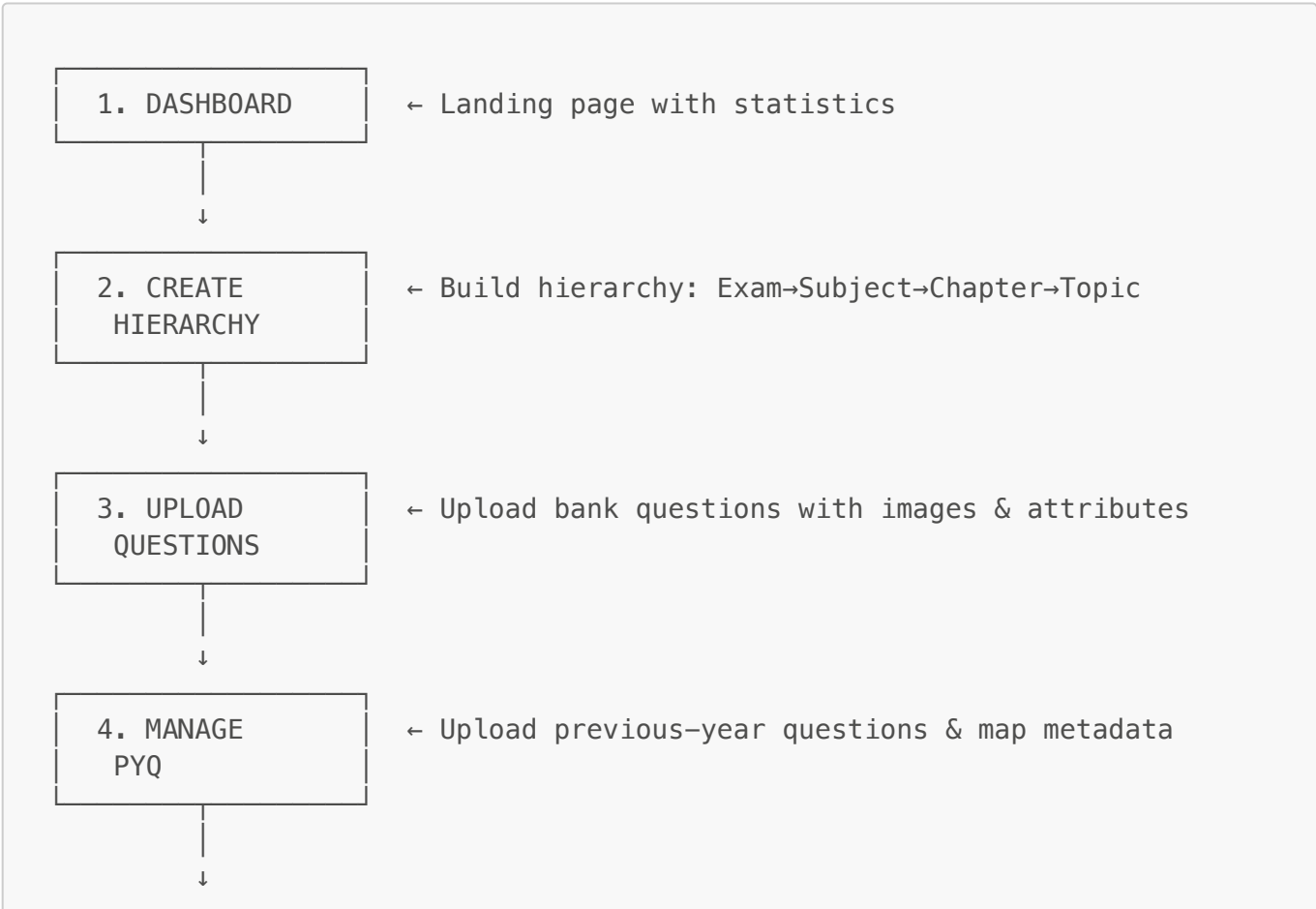


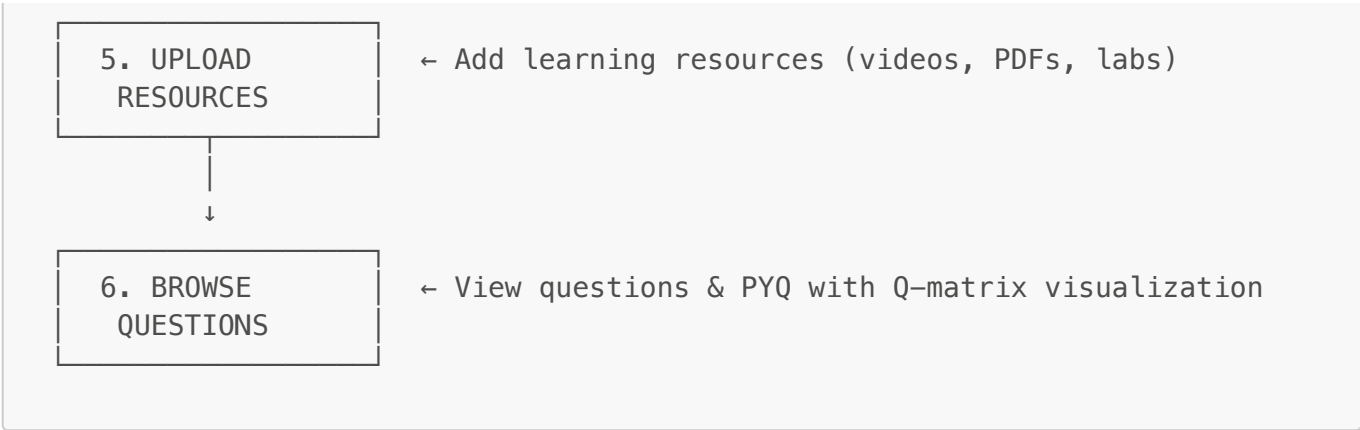
Admin Material Upload Web App - Architecture Overview

Application Structure



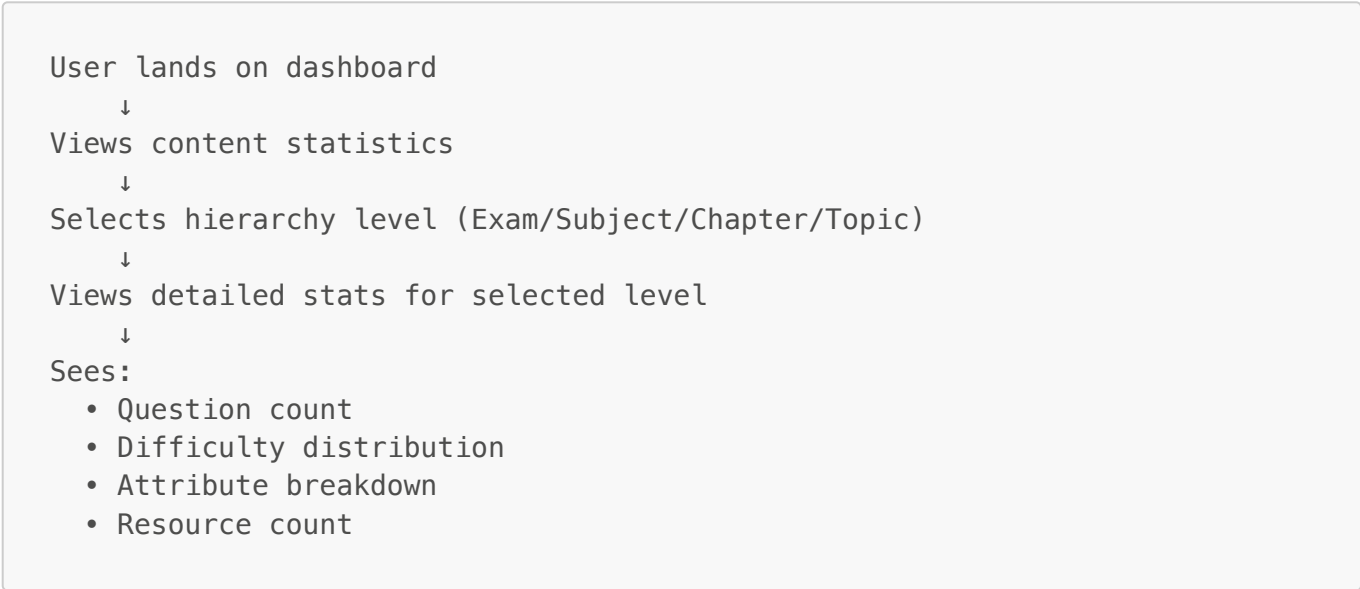
5-Page Application Flow



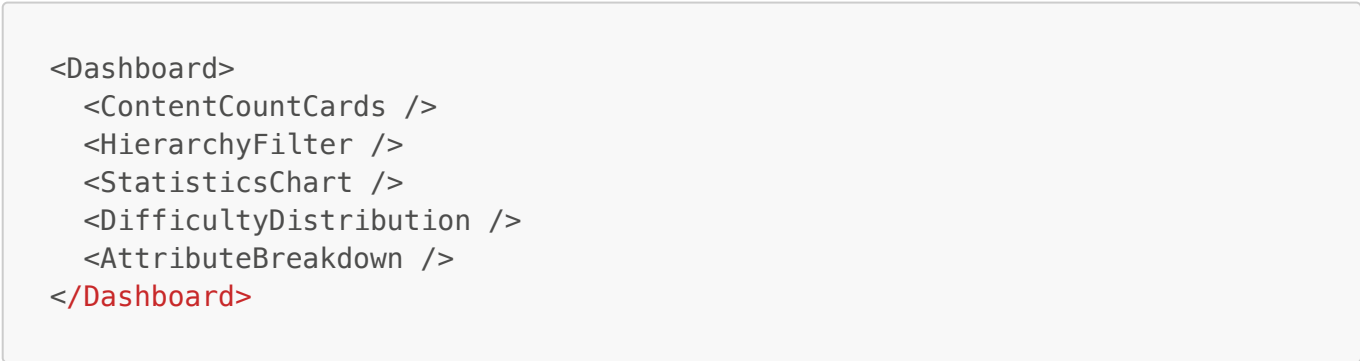


Page 1: Dashboard

User Flow



Key Components



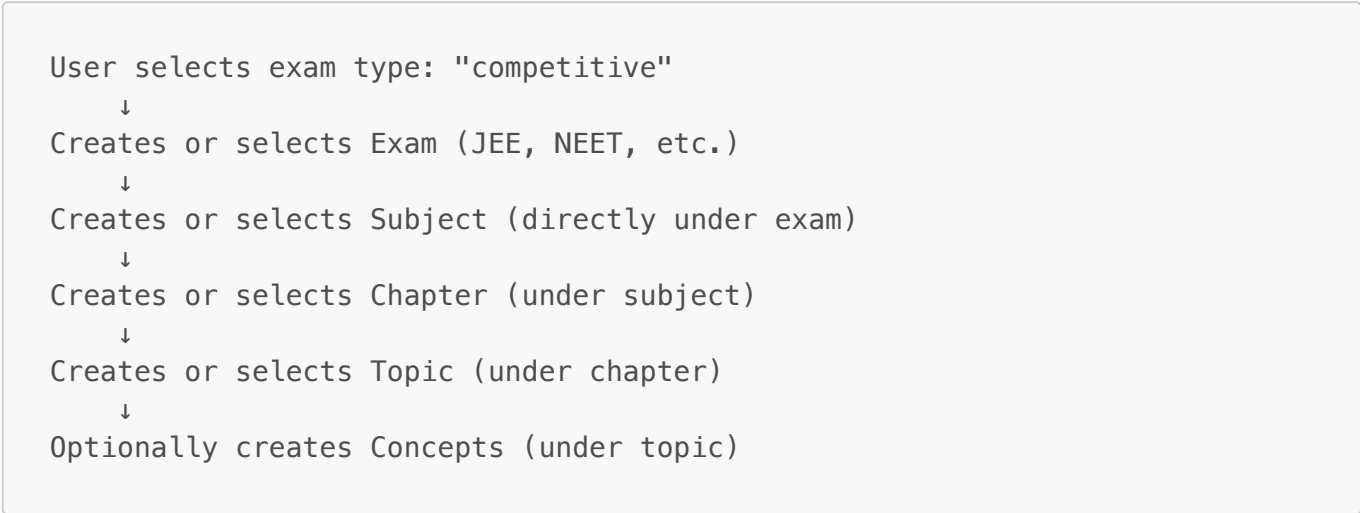
API Calls



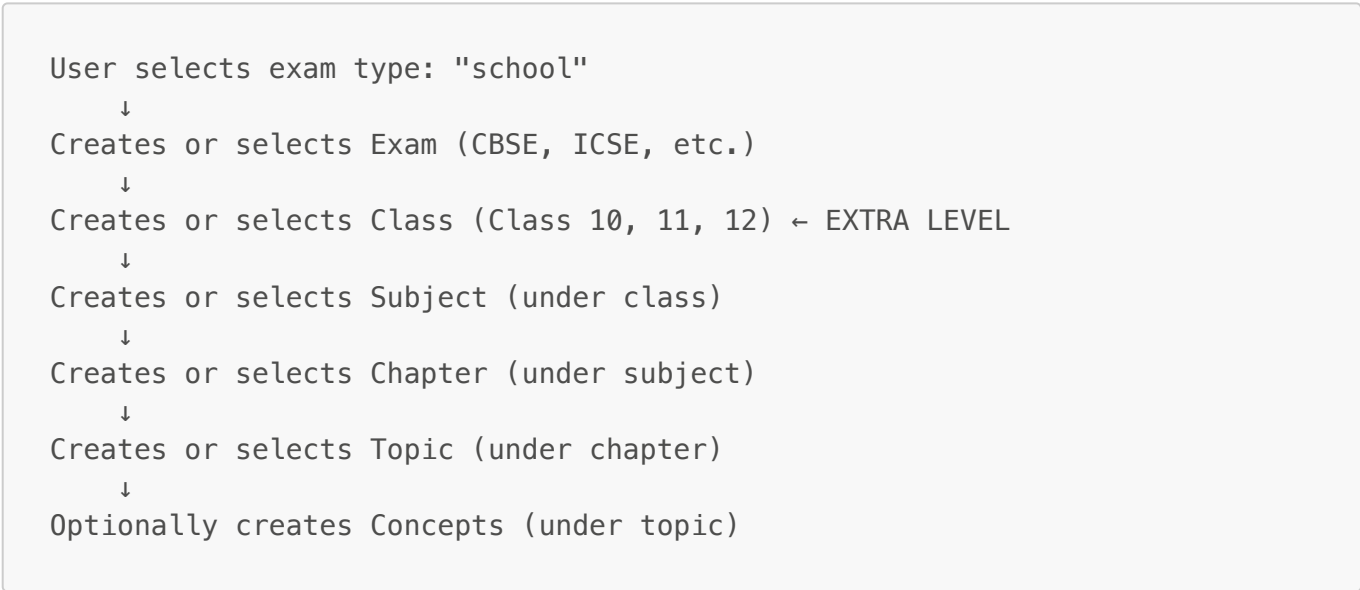
Page 2: Hierarchical Creation

User Flow

Competitive Exam Path:

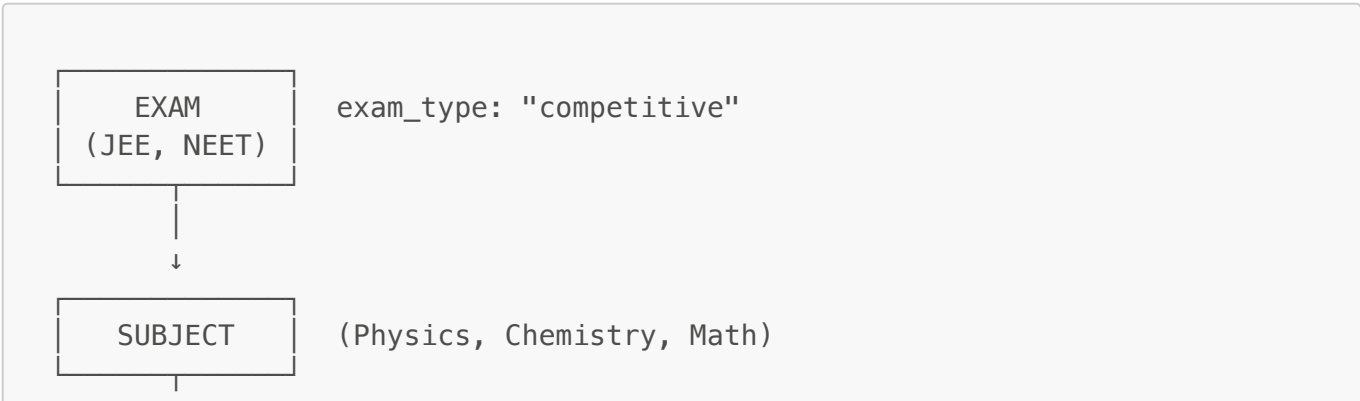


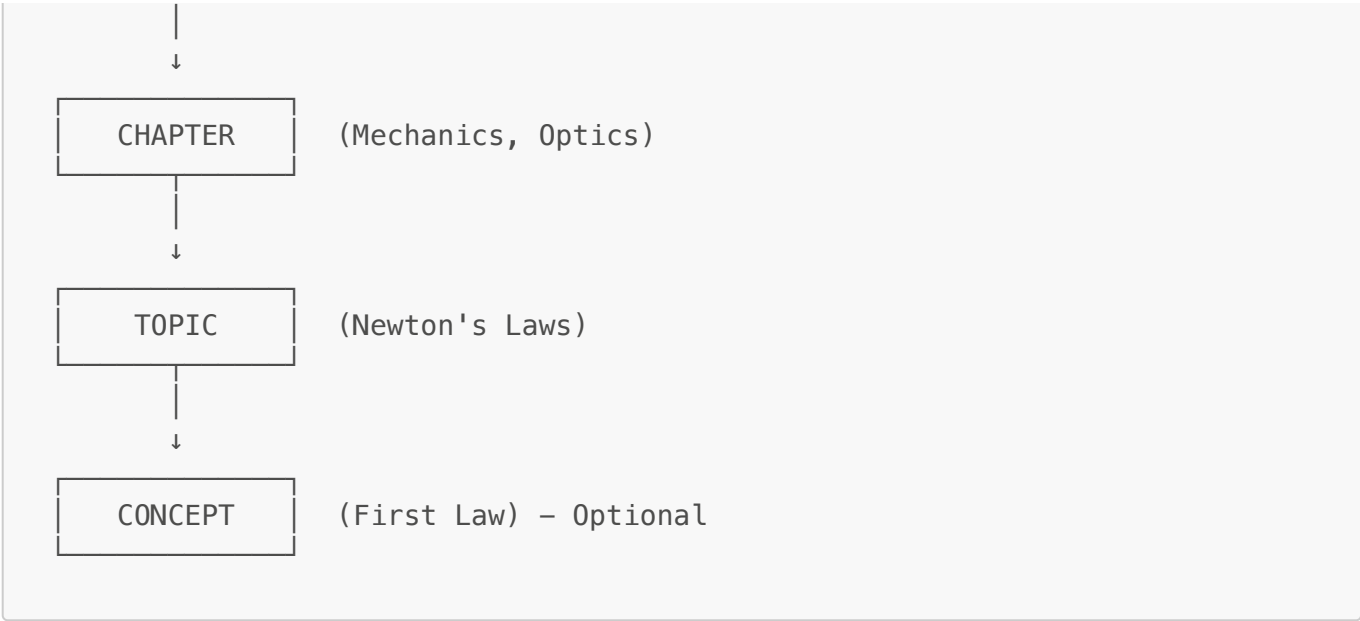
School Exam Path:



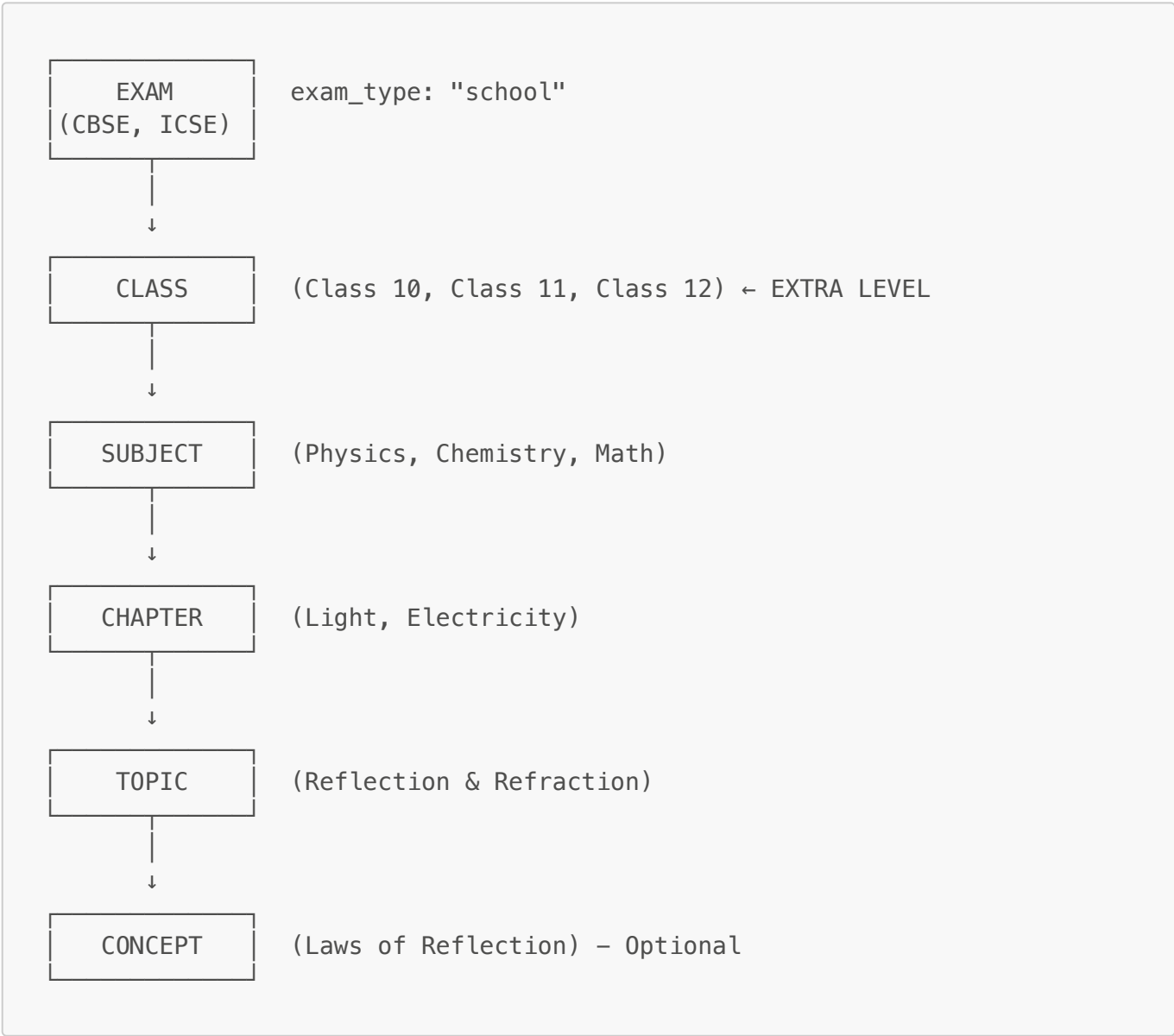
Hierarchy Structure

Path 1: Competitive Exam (JEE, NEET, CAT, etc.)





Path 2: School Exam (CBSE, ICSE, State Board, etc.)



Key Components

```
<HierarchyCreator>
  <ExamTypeSelector />  { /* competitive or school */}
  <CascadingDropdowns>
    <ExamSelector />
      {examType === 'school' && <ClassSelector />}  { /* Conditional */}
    <SubjectSelector />
    <ChapterSelector />
    <TopicSelector />
    <ConceptSelector />
  </CascadingDropdowns>
  <CreateNewForm />
  <HierarchyTreeView />
</HierarchyCreator>
```

API Calls

```
POST /api/exams
POST /api/classes (school exams only)
POST /api/subjects (with exam_id OR class_id)
POST /api/chapters
POST /api/topics
POST /api/concepts
GET /api/hierarchy/tree
GET /api/hierarchy/classes?exam_id={id} (school exams)
GET /api/hierarchy/subjects?exam_id={id} (competitive)
GET /api/hierarchy/subjects?class_id={id} (school)
GET /api/hierarchy/chapters?subject_id={id}
```

Page 3: Question Upload Service

User Flow

```
User navigates hierarchy to select topic
↓
Chooses upload mode (Single/Batch)
↓
Enters question details:
• Question text
• 4 options (A, B, C, D)
• Correct answer
• 3PL parameters (difficulty, discrimination, guessing)
↓
Uploads question image (optional)
↓
Uploads option images (optional)
↓
Selects/creates attributes
```

↓
Saves question

Question Structure

```
{
  "content": "Question text here",
  "options": {
    "A": "First option",
    "B": "Second option",
    "C": "Third option",
    "D": "Fourth option"
  },
  "correct_answer": "A",
  "difficulty": 0.5,
  "discrimination": 1.2,
  "guessing": 0.25,
  "attributes": [
    { "attribute_id": "attr-1", "value": true },
    { "attribute_id": "attr-2", "value": true }
  ]
}
```

Image Upload Flow

```
Question Created (ID: question-123)
↓
Upload Question Image → /api/questions/123/image
↓
Upload Option Images → /api/questions/123/options/images
  (Sends: option_A, option_B, option_C, option_D files)
↓
Images stored in Supabase Storage
↓
URLs returned and saved in question metadata
```

Key Components

```
<QuestionUpload>
  <HierarchyNavigator />
  <UploadModeSelector mode="single|batch" />

  <SingleQuestionForm>
    <QuestionTextInput />
    <QuestionImageUpload />
    <OptionsInput>
```

```

    <OptionField key="A" withImage />
    <OptionField key="B" withImage />
    <OptionField key="C" withImage />
    <OptionField key="D" withImage />
  </OptionsInput>
  <CorrectAnswerSelector />
  <ThreePLParameters>
    <DifficultySlider min={-3} max={3} />
    <DiscriminationSlider min={0} max={3} />
    <GuessingInput default={0.25} />
  </ThreePLParameters>
  <AttributeSelector>
    <ExistingAttributes />
    <CreateNewAttribute />
  </AttributeSelector>
</SingleQuestionForm>

<BatchQuestionForm>
  <QuestionList />
  <AddQuestionButton />
</BatchQuestionForm>

<QuestionPreview />
<SubmitButton />
</QuestionUpload>

```

API Calls

```

GET /api/topic/{id}/attributes
POST /api/questions/create-with-attributes
POST /api/questions/batch
POST /api/questions/{id}/image
POST /api/questions/{id}/options/images
PUT /api/questions/{id}
DELETE /api/questions/{id}

```

Page 4: PYQ Upload & Session Management

User Flow

```

Select hierarchy (Exam → Subject → Chapter → Topic)
↓
Choose ingest mode (Single | Bulk JSON | Excel import)
↓
Enter PYQ question, options, and correct answer
↓
Attach metadata (year, session, marks, tags, source, difficulty)
↓

```

Assign existing topic attributes or create new ones

↓

Preview and upload → immediate availability for practice sessions

Key Components

```
<PYQUploadPage>
  <HierarchyNavigator />
  <IngestModeTabs modes={['single', 'bulk', 'excel']} />
  <MetadataForm>
    <YearSessionFields />
    <MarksAndTiming />
    <SourceTagsInput />
    <QuestionTypeSelector />
  </MetadataForm>
  <QuestionContentForm />
  <AttributeSelector allowCreate />
  <PreviewPanel />
  <UploadActions />
  <RecentUploadsTable />
</PYQUploadPage>
```

API Calls

POST /api/pyq/upload/single	# single-question ingest
POST /api/pyq/upload/bulk	# JSON batch ingest
POST /api/pyq/upload/excel	# spreadsheet ingest
GET /api/pyq/filters/options	# populate metadata selectors
GET /api/pyq/search	# list PYQ for hierarchy/metadata filters
POST /api/pyq/session/create	# optional quick practice sessions

Data Integrations

- **Supabase Tables:**
 - **questions** (stores PYQ question body)
 - **pyq_metadata** (stores year/session/source metadata)
 - **q_matrix** (links PYQ to topic attributes)
- **Fallback Cache:** local in-memory store mirrors uploads when Supabase is unavailable so the admin flow continues without error.

UX Considerations

- Surface metadata defaults based on last upload (e.g., auto-fill year/session).
- Warn if a PYQ already exists for the same year/paper/question_number combination.
- Provide quick links to launch a practice session filtered to the current hierarchy and newly uploaded PYQ set.

Page 5: Resource Upload Service

User Flow

```
User navigates to topic
↓
Selects resource type:
🎥 Video | 📄 PDF | 🖋️ Virtual Lab | 🧑‍🔬 3D Model
🌟 Animation | 🖼️ Image | 🎮 Interactive | 📰 Article | ⚙️ Simulation
↓
Enters resource details:
• Title
• Description
• URL
• Thumbnail URL (optional)
• Duration (for videos/animations)
• File size
• Metadata (platform, quality, etc.)
↓
Saves resource
↓
Views all resources for topic
```

Resource Types & Examples

```
VIDEO:
- YouTube: https://youtube.com/watch?v=...
- Vimeo: https://vimeo.com/...
- Metadata: { platform: "YouTube", quality: "1080p", creator: "Khan Academy" }

PDF:
- Study guides, textbooks
- Metadata: { pages: 25, language: "English" }

VIRTUAL_LAB:
- PhET simulations: https://phet.colorado.edu/...
- Labster: https://labster.com/...
- Metadata: { platform: "PhET", experiments: [...] }

3D_MODEL:
- GLTF/GLB files
- Metadata: { format: "GLTF", polygons: 50000, animated: true }

ANIMATION:
- MP4, GIF animations
- Metadata: { format: "MP4", fps: 60, loopable: true }
```

Key Components

```
<ResourceUpload>
  <TopicNavigator />
  <ResourceTypeSelector />

  <ResourceForm>
    <TitleInput />
    <DescriptionInput />
    <URLInput />
    <ThumbnailURLInput />
    <DurationInput show={type === 'video' || type === 'animation'} />
    <FileSizeInput />
    <MetadataForm>
      {/* Dynamic fields based on resource type */}
      <PlatformInput />
      <QualityInput />
      <FormatInput />
    </MetadataForm>
    <OrderIndexInput />
  </ResourceForm>

  <ExistingResourcesList>
    <ResourceCard />
    <EditButton />
    <DeleteButton />
  </ExistingResourcesList>

  <BulkUploadButton />
</ResourceUpload>
```

API Calls

```
POST /api/topics/{id}/resources
POST /api/topics/{id}/resources/bulk
GET /api/topics/{id}/resources
GET /api/topics/{id}/resources?resource_type=video
GET /api/chapters/{id}/resources
PUT /api/resources/{id}
DELETE /api/resources/{id}
```

Page 6: Enhanced Question Fetching & Display

User Flow

```
User selects hierarchy level (Exam/Subject/Chapter/Topic)
↓
Views questions at that level
↓
```

Sees for each question:

- Question text & images
- Options with images
- Correct answer
- 3PL parameters
- Q-Matrix visualization
- Attributes tested

↓

Can filter by:

- Difficulty range
- Text search
- Specific attributes

↓

Can edit or delete questions

Q-Matrix Visualization

What is Q-Matrix?

- Maps questions to cognitive attributes
- Binary matrix: 1 = question tests attribute, 0 = doesn't test

Example:

Attributes for "Newton's Laws" topic:

- [0] Understanding First Law
- [1] Understanding Second Law
- [2] Understanding Third Law
- [3] Applying Forces
- [4] Problem Solving

Question: "A book rests on a table. What keeps it stationary?"

Q-Vector: [1, 0, 0, 0, 0]

- ✓ Tests "Understanding First Law"
- Doesn't test others

Question: "Calculate force on 5kg mass with 10m/s^2 acceleration"

Q-Vector: [0, 1, 0, 1, 1]

- ✓ Tests "Understanding Second Law"
- ✓ Tests "Applying Forces"
- ✓ Tests "Problem Solving"

Visual Representation

Question #42

[Edit] [Delete]

[Question Image]

What is Newton's First Law of Motion?

- ☐ A. Law of Inertia ✓
- ☐ B. $F = ma$
- ☐ C. Action-Reaction
- ☐ D. None of the above

Difficulty: 0.50 | Discrimination: 1.20 | Guess: 0.25

Q-Matrix: Tests 2 of 5 attributes

- | |
|--|
| <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Understanding First Law <input checked="" type="checkbox"/> Applying Newton's Laws <input type="checkbox"/> Understanding Second Law <input type="checkbox"/> Problem Solving <input type="checkbox"/> Advanced Applications |
|--|

Key Components

```
<QuestionBrowser>
  <HierarchySelector />
  <ContentTypeToggle options={['bank', 'pyq', 'combined']} />
  <FilterPanel>
    <SearchInput />
    <DifficultyRangeFilter />
    <AttributeFilter />
    <PYQMetadataFilter visibleWhen={['pyq', 'combined']} />
  </FilterPanel>

  <QuestionList>
    <QuestionCard>
      <QuestionHeader />
      <QuestionContent>
        <QuestionImage />
        <QuestionText />
        <PYQMetadataSummary visibleWhen="pyq" />
      </QuestionContent>
      <OptionsDisplay>
        <Option key="A" withImage />
        <Option key="B" withImage />
        <Option key="C" withImage />
        <Option key="D" withImage />
      </OptionsDisplay>
      <ThreePLDisplay visibleWhen="bank">
        <Difficulty />
        <Discrimination />
        <Guessing />
      </ThreePLDisplay>
      <QMatrixDisplay />
    </QuestionCard>
  </QuestionList>
</QuestionBrowser>
```

```

    <ActionButton>
      <EditButton disabledWhen="pyq" />
      <DeleteButton disabledWhen="pyq" />
      <LaunchPracticeSession />
    </ActionButton>
  </QuestionCard>
</QuestionList>

<Pagination />
</QuestionBrowser>

```

API Calls

```

GET /api/hierarchy/{level}/{id}/questions/enhanced?page=1&page_size=20
GET /api/hierarchy/{level}/{id}/questions
GET /api/search/questions?topic_id={id}&difficulty_min=0.5
GET /api/pyq/search?topic_id={id}&year={year}&source={source}
POST /api/pyq/session/create
POST /api/pyq/session/{session_id}/submit
GET /api/item-bank/{level}/{id}

```

Enhanced Question Response Format

```

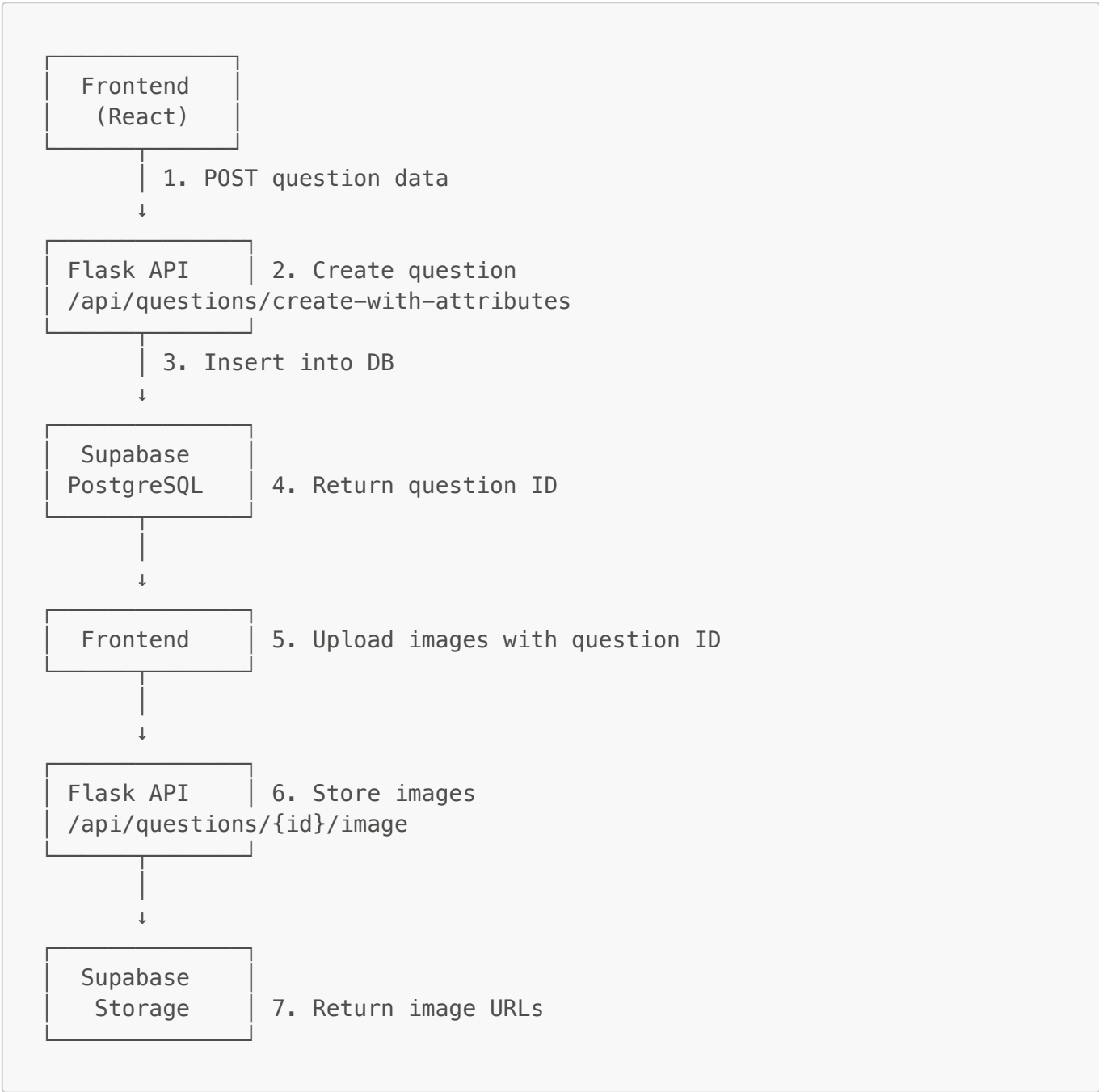
{
  "level": "topic",
  "level_id": "topic-uuid-1",
  "total_questions": 150,
  "attributes": [
    { "id": "attr-1", "name": "Understanding First Law", ... },
    { "id": "attr-2", "name": "Applying Forces", ... }
  ],
  "questions": [
    {
      "id": "q-1",
      "content": "Question text",
      "options": { "A": "...", "B": "...", "C": "...", "D": "..." },
      "correct_answer": "A",
      "difficulty": 0.5,
      "discrimination": 1.2,
      "guessing": 0.25,
      "q_vector": [1, 1, 0, 0],
      "attribute_count": 2,
      "question_image_url": "https://...",
      "option_images": { "A": "https://...", "B": "https://..." }
    }
  ],
  "pagination": {
    "page": 1,
    "page_size": 20,

```

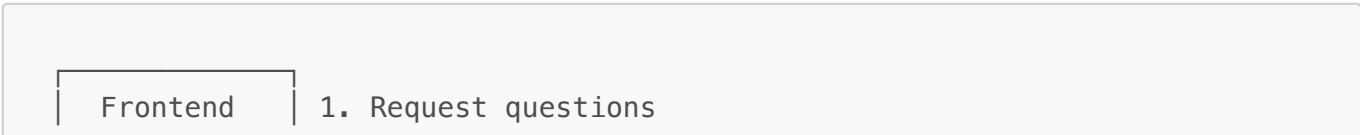
```
    "total": 150,  
    "total_pages": 8,  
    "has_more": true  
  }  
}
```

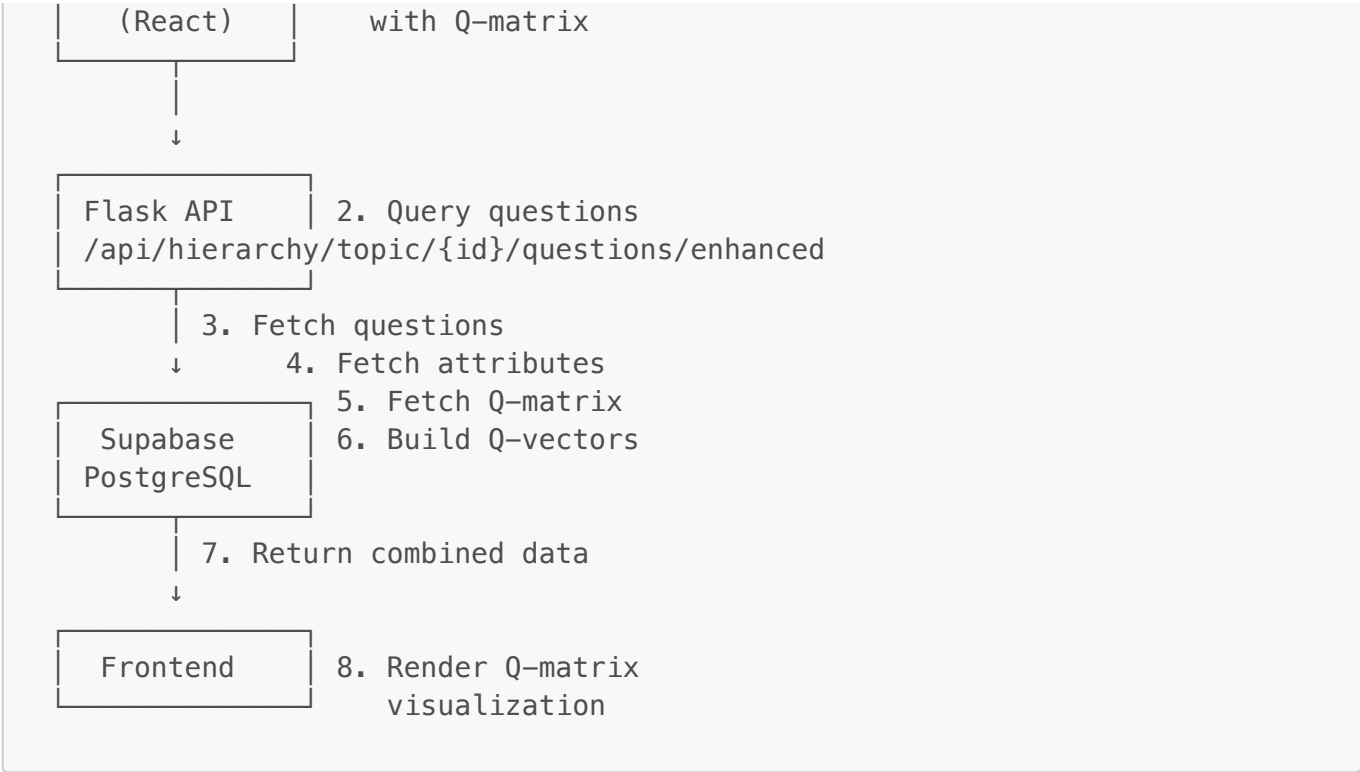
Data Flow Architecture

Question Upload Flow



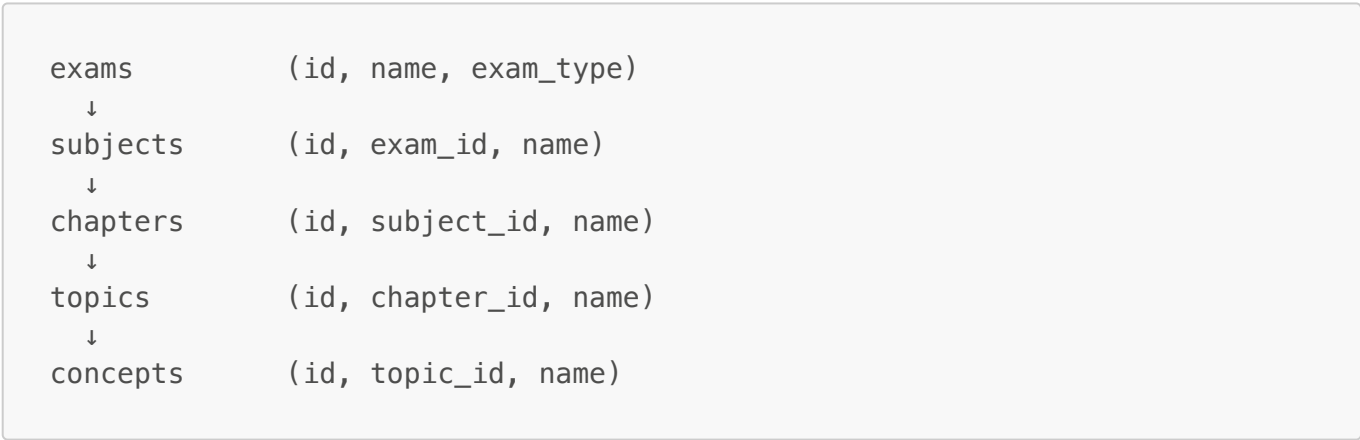
Question Fetching with Q-Matrix Flow



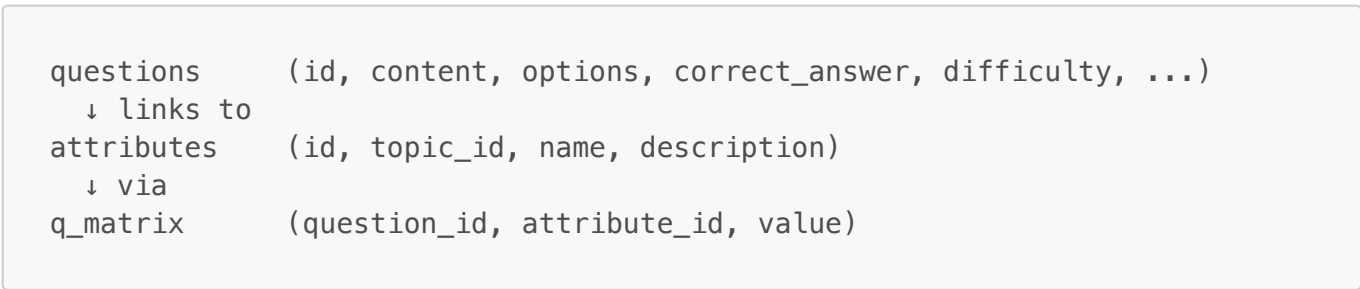


Database Schema (Key Tables)

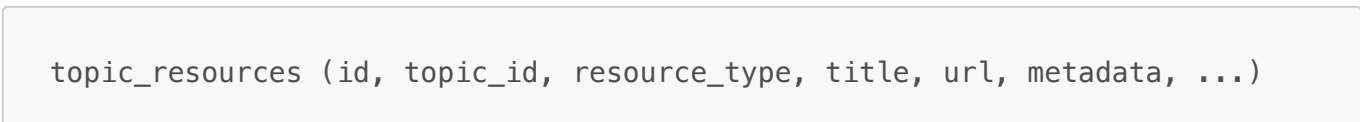
Hierarchy Tables



Question Tables



Resource Table



Technology Stack

Backend

- **Framework:** Flask (Python)
- **Database:** PostgreSQL (via Supabase)
- **Storage:** Supabase Object Storage
- **Port:** 5200

Frontend (Recommended)

- **Framework:** React or Next.js
- **Language:** TypeScript
- **UI Library:** Material-UI / Ant Design / Chakra UI
- **State:** React Query / Redux Toolkit
- **Forms:** React Hook Form
- **File Upload:** react-dropzone

API

- **Protocol:** REST
- **Format:** JSON
- **Images:** multipart/form-data

Security Considerations

Current Implementation

- No authentication (add before production)
- CORS enabled for development
- File upload size limits
- Input validation on backend

Recommendations for Production

- Add JWT authentication
- Role-based access control (admin, teacher, student)
- Rate limiting
- Input sanitization
- SQL injection prevention (using ORM)
- File type validation
- Virus scanning for uploads

Performance Optimization

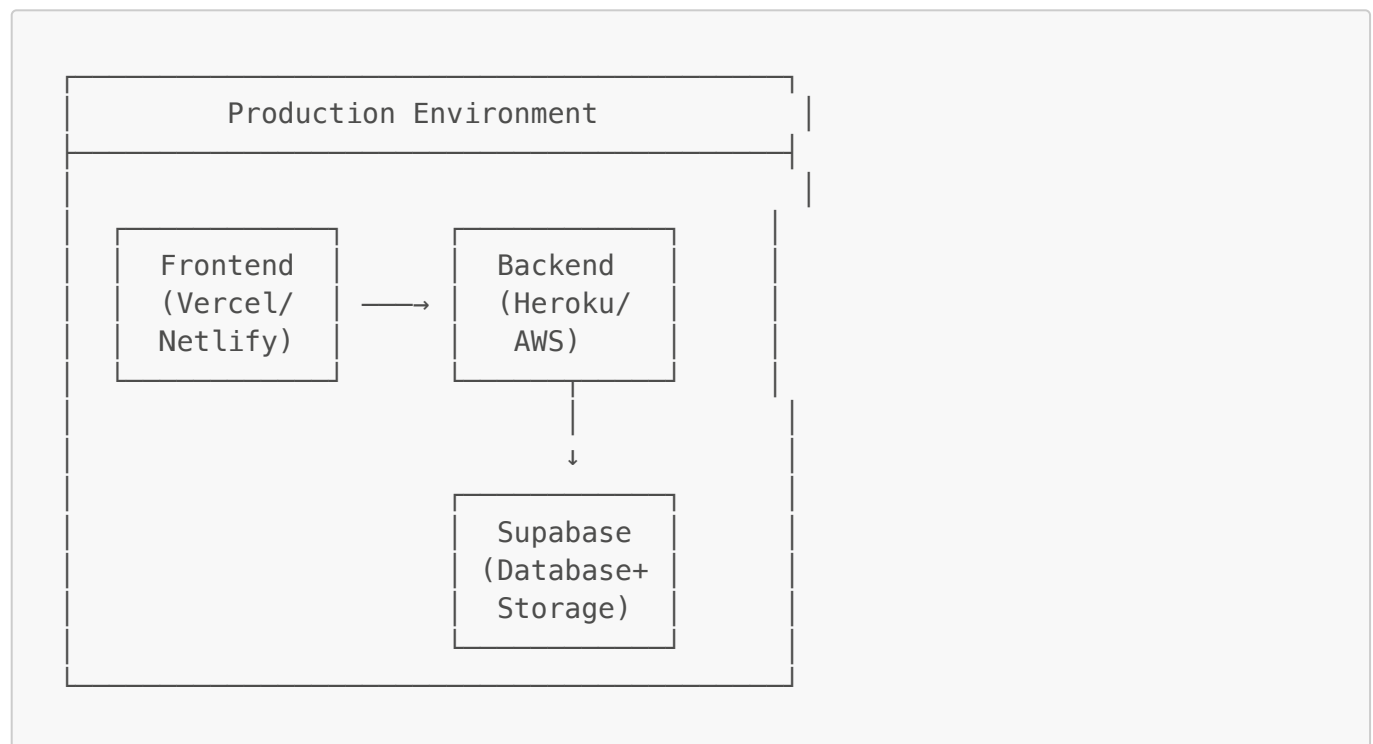
Backend

- Database indexes on foreign keys
- Pagination for large result sets
- Caching for frequently accessed data
- Connection pooling

Frontend

- Lazy loading for images
- Virtual scrolling for long lists
- Debounced search inputs
- Optimistic UI updates
- React.memo for expensive components

Deployment Architecture



Getting Started

1. Backend Setup

```
cd its-question-service
python -m app.main
# Server runs on http://localhost:5200
```

2. Frontend Setup

```
npx create-react-app admin-app --template typescript
cd admin-app
npm install @mui/material axios react-query
npm start
```

3. API Configuration






```
// src/api/config.ts
export const API_BASE_URL = 'http://localhost:5200/api';
```

4. Start Building

- Follow FRONTEND_ADMIN_APP_GUIDE.md
- Use API_QUICK_REFERENCE.md for endpoints
- Copy provided component examples

Summary

This admin application provides a complete solution for:

-  Managing educational content hierarchy
-  Uploading questions with images and attributes
-  Adding learning resources to topics
-  Viewing questions with Q-matrix analysis
-  Comprehensive statistics and filtering

All documentation is complete and ready to use!

Refer to:

- [FRONTEND_ADMIN_APP_GUIDE.md](#) - Complete guide
- [API_QUICK_REFERENCE.md](#) - Quick reference
- [README_FRONTEND_DOCS.md](#) - Documentation index