



# Implementation Plan for FCDM-Based Adaptive Question Bank

## Introduction:

This plan outlines a step-by-step approach to convert a large PDF-based question bank (high school/SAT-level math and data reasoning questions) into a dynamic, adaptive assessment system. We leverage **Fuzzy Cognitive Diagnosis Models (FCDM)** for skill profiling and a **rule-based adaptive testing** engine for delivering the *next-best* questions. Each step identifies key tasks, appropriate tools/technologies, and best practices to ensure the system is scalable and maintainable.

## Step 1: Extracting and Structuring Questions from PDFs using LLMs

Extracting questions from PDFs is the first critical step. We will use a combination of PDF parsing tools and Large Language Model (LLM) assistance to digitize and structure the content:

- 1. PDF Text and Math Extraction:** Use PDF parsing libraries (e.g. **PyMuPDF** or **PDFPlumber**) to extract raw text. If the PDFs contain mathematical notation or diagrams, integrate an OCR tool like **Mathpix** or Tesseract to capture equations and figures as LaTeX or image references. This ensures all math symbols and expressions are accurately retrieved.

2. **Segmenting Questions:** Identify question boundaries, options, and any solution/explanation text. This can be done by analyzing formatting patterns (e.g. questions often start with numbers or new lines). For complex layouts, train an LLM or use prompt-engineering with a model like **GPT-4** to recognize and label question components (question stem, answer choices, correct answer, etc.) in the extracted text.
3. **LLM-driven Structuring:** Feed the raw text (in chunks to fit token limits) to an LLM with instructions to output structured data (for example, JSON with fields: *question\_text*, *options*, *correct\_answer*, *explanation*, etc.). The LLM's language understanding can infer structure even when PDF formatting is inconsistent.
4. **Validation and Cleanup:** After LLM processing, validate the structured output. Ensure each question has the correct number of options, all mathematical notation is properly formatted (e.g. as LaTeX for the digital platform), and no content is missing. Spot-check a sample of questions manually to verify accuracy. Develop scripts to catch common errors (like garbled characters or mis-numbered choices).

*Tools & Best Practices:* Use a **hybrid human-AI approach** – the LLM greatly accelerates structuring, but human oversight is needed for quality control. Utilize version control (Git) for the question data files (JSON or XML), so changes or extractions can be tracked. This step can be scaled by processing PDFs in parallel and using cloud-based OCR/LLM services if needed. Keep an eye on any sensitive data in prompts (avoid sending answer keys if not necessary) when using external LLM APIs.

## Step 2: Annotating Questions with Metadata

Once questions are structured, each item must be enriched with metadata for effective organization and adaptivity. The following metadata tags will be applied: **chapter**, **topic**, **sub-topic**, **concept tags**, **difficulty level**, **question type**, and **cognitive skill level**. We will implement a semi-automated tagging process:

- **Define Taxonomy:** First, define a taxonomy of subjects → chapters → topics → sub-topics, and a list of fine-grained **concepts/skills** relevant to the curriculum. For cognitive skills, decide on a framework (for example, Bloom's taxonomy levels such as *Knowledge, Comprehension, Application*, etc.). Difficulty can be classified into tiers (e.g. easy, medium, hard) based on expert judgment or empirical data (past performance if available).
- **LLM-Assisted Tagging:** Leverage an LLM to suggest tags for each question. The LLM can be prompted with the question text to output likely topic/concept and cognitive skill. For instance, *"This question involves interpreting a data graph, likely tagged under Data Analysis > Interpreting Charts; cognitive skill: Analysis."* An AI-based metadata generator can achieve high accuracy for structured tags like these – prior research on semi-automatic item tagging showed around *78% accuracy for cognitive level and over 90% for content, type, and difficulty tags* using automated methods [cse.iitb.ac.in](https://cse.iitb.ac.in) . We expect modern LLMs to match or exceed this performance.
- **Rule-Based Classification:** For certain metadata, simple rules or keyword matching may assist. For example, if a question contains an integral symbol, it likely falls under calculus; if it mentions "median" or "mean," it's a statistics concept. These rules can augment the LLM suggestions for reliability.
- **Human Review and Refinement:** Have subject matter experts review the LLM-generated tags for a sample of questions. This is critical for maintaining accuracy and educator trust. Experts can correct misclassifications and refine the tagging guidelines for the AI on subsequent passes (iteratively improving prompts or model fine-tuning).
- **Metadata Structure:** Store the annotations in a database or as part of the question object (the digital platform's question bank likely supports custom tags or fields). Each question thus will have fields like `{chapter: "", topic: "", subtopic: "", concepts: [], difficulty: "", q_type: "", cognitive_level: ""}` .

*Tools & Best Practices:* Consider using a specialized **annotation tool or interface** for experts to quickly validate tags (perhaps a simple web app showing question + AI-suggested tags with edit options). This can speed up review. Maintain a **master list of concepts and skills** – essentially a mini knowledge ontology – to ensure consistent naming (e.g., one concept shouldn't be tagged as "Coordinate Geometry" on one question and "Analytic Geometry" on another if they are meant to be the same). Periodically re-evaluate the tagging model against new questions to ensure it remains accurate. Automating the metadata generation, with human-in-the-loop, will dramatically speed up the organization of the question repository [cse.iitb.ac.in](https://cse.iitb.ac.in) .

## Step 3: Mapping Concepts and Skills to a Knowledge Structure (for FCDM)

With each question tagged by concept/skill, we create a **knowledge structure or graph** that will underpin the Fuzzy Cognitive Diagnosis Model. This involves mapping how concepts relate to each other and to assessment items:

- **Knowledge Graph Construction:** Represent the domain knowledge as a directed graph or network. Nodes represent **concepts/skills** (e.g., "*Pythagorean Theorem*", "*Box-and-Whisker Plot Interpretation*"), and edges can represent prerequisite relationships or topic hierarchy (e.g., "*linear equations*" might be a prerequisite for "*quadratic equations*"). This graph encodes the learning structure students navigate.
- **Q-Matrix Definition:** In cognitive diagnostic terms, define the Q-matrix, which is a mapping of questions to the concepts they assess [github.com](https://github.com) . For each question in the bank, mark which concept(s) it covers. Most high school questions target a primary concept and maybe secondary ones. This matrix will be used by the FCDM to connect student responses to skill mastery estimates. (For example, Question 42 might map to concepts {Algebraic Manipulation, Linear Functions} indicating it requires both skills.)

- **Fuzzy Cognitive Diagnosis Model Setup:** The FCDM extends traditional Cognitive Diagnosis Models (CDMs) by allowing continuous or “fuzzy” proficiency levels instead of binary mastery. Using the Q-matrix and student performance data, FCDM will infer a proficiency score for each concept per student. We will integrate an existing implementation (such as the **EduCDM** library’s FuzzyCDF model) to leverage tested algorithms [github.com](https://github.com) [github.com](https://github.com) . These models take as input the question-skill relationships and student answers (correct/incorrect), and output a profile of skill mastery probabilities.
- **Aligning with Graph:** Ensure the knowledge graph’s prerequisites inform the FCDM interpretation. If a student has low mastery in a prerequisite concept, FCDM can propagate that fuzzily to related higher concepts. Conversely, high performance in an advanced concept can boost the estimate of underlying basics (to some extent). Fine-tune the model parameters or priors based on expert knowledge (e.g., some concepts might be given more weight if they are fundamental).
- **Continuous Updating:** The knowledge structure should not be static; as new questions or concepts are introduced, update the graph and Q-matrix. The system must accommodate adding or refining skills over time. For scaling, store the knowledge structure in a graph database like **Neo4j** or as structured data that can be easily updated, and design the FCDM integration to pull the latest structure for its calculations.

*Tools & Best Practices:* Use visualization tools (Gephi or Neo4j Bloom) to visualize the concept graph for sanity checks – it helps to spot if any concept is orphaned or if any dependency cycle exists. An accurate knowledge model is *crucial*, as it drives mastery estimates and adaptivity [arxiv.org](https://arxiv.org) [arxiv.org](https://arxiv.org) . We should involve educators to verify the prerequisite links (e.g., confirm that “fractions” is indeed prerequisite to “linear equations” etc.). This collaborative modeling ensures the knowledge graph remains pedagogically sound. When deploying FCDM, initialize it with reasonable default parameters; as student usage grows, calibrate the model using response data (the model can learn item difficulties or slip/guess probabilities to improve diagnostic accuracy).

## Step 4: Designing a Rule-Based Adaptive Testing Framework

With questions ready and a diagnostic model in place, we design the **adaptive testing logic** that selects the optimal next question based on a student's performance in real time. Unlike pure psychometric adaptivity (IRT-based), we will use a *rule-based engine* informed by the FCDM skill profile and expert rules. Key design elements:

- **Initial Assessment and Seeding:** When a student begins, the system may start with a medium-difficulty question in a core topic to gauge level. Alternatively, a short entry quiz covering each major domain can initialize the student's skill profile (FCDM can use these to set starting mastery levels).
- **Next-Best Question Selection:** After each question response, apply rules to pick the next question:
  - *Concept targeting:* Emphasize weaker areas – for example, if FCDM indicates the student's lowest proficiency is in *Geometry*, the next question should address a geometry concept the student is "ready to learn" (not too easy, but a bit challenging).
  - *Difficulty adjustment:* If a student answers correctly with high confidence (or quickly), consider increasing difficulty next time (harder question in the same concept or a linked, more advanced concept). If the student answers incorrectly, the next question should be easier or target a prerequisite concept.
  - *Prerequisite logic:* If a student is struggling in a concept, branch to an easier question on a **prerequisite skill**. This echoes knowledge space theory, where failing a concept suggests checking understanding of its prerequisites aleks.com. Conversely, if they excel, the engine can skip ahead to more advanced topics that build on the current concept.
  - *Coverage and variety:* Ensure the rules also cover broad topic coverage to avoid drilling too narrowly. For instance, after a series of algebra questions, the rules might direct the next question to a different domain (geometry or data analysis) to keep the assessment comprehensive, then loop back to weaker topics.
- **Rule Implementation:** Use a simple rules engine or conditional logic in the platform's backend. Pseudocode example:

pseudo

 Copy

```

if last_answer.correct == false:
    if concept_mastery[last_question.concept] < 0.5:
        next_question = pickQuestion(prerequisite_of(last_question.concept),
    else:
        next_question = pickQuestion(last_question.concept, difficulty="easy")
else: # last answer correct
    if concept_mastery[last_question.concept] > 0.8:
        next_question = pickQuestion(next_concept(last_question.concept), difficulty="hard")
    else:
        next_question = pickQuestion(last_question.concept, difficulty="medium")

```



This illustrates using the concept mastery estimates ( `concept_mastery` from FCDM) and simple thresholds to decide. The function `next_concept()` might choose a more advanced concept or a related topic if mastery is solid. The `pickQuestion(concept, difficulty)` will retrieve an unused question matching the criteria.

- **Preventing Loops or Stagnation:** Include rules to avoid repetitive loops (e.g., not to give the same concept three times in a row even if it's weak, to prevent frustration). Also define an **exit condition** for the test: it could be a fixed number of questions or dynamic (e.g., when confidence in all major concepts mastery reaches a threshold or the improvement plateaus). For practice mode, it could be open-ended until the student stops.
- **Logging and Adjustments:** Log every question served and the reason (rule) it was chosen. This will help in later analysis to refine the rules. Over time, we might discover certain rules work better than others, or we may incorporate more of the FCDM probabilities directly (for example, selecting the question that maximally reduces uncertainty in the student's knowledge state). We should plan periodic reviews of these rules based on student outcome data.

*Tools & Best Practices:* If the digital platform allows scripting or has an API, implement the selection logic in that environment (for instance, some platforms might support **Python** or **JavaScript** hooks for custom adaptive quizzes). If not, consider a middleware approach: the platform sends the student's answer to an external service (our adaptive engine) which then returns the next question ID to present. A **rules engine** like **Drools** or a lightweight Python-based rules evaluator can help manage complex rule sets more transparently. To scale, ensure the next-question selection is efficient (the question bank can be indexed by concept and difficulty for quick lookup). Also, thoroughly test the adaptive flow with simulated student profiles (e.g., a high-performer, a low-performer, someone with uneven skills) to ensure the rules lead to sensible question sequences in all cases.

## Step 5: Feedback Mechanisms (Periodic and Real-Time)

Effective learning requires timely feedback. We will incorporate both **real-time feedback** during the adaptive session and **periodic diagnostic feedback** as the student progresses:

- **Immediate Feedback on Answers:** After each question, provide the student with feedback on that item. At minimum, indicate whether they were correct or incorrect. Since the platform supports math content, we can also display a step-by-step solution or hint for the problem (this could be authored manually or generated with an LLM and then verified). This immediate feedback helps learning – the student can see mistakes and correct misconceptions on the fly.
- **Skill Mastery Updates:** As the student answers questions, update an on-screen proficiency dashboard. For example, show a list of major topics with a bar or percentage indicating current estimated mastery (as computed by FCDM, which updates after each response). This gives real-time insight such as *"Algebra: 80% mastered, Geometry: 50% mastered"*. However, use caution in presentation – because these are estimates, we might bucket them into qualitative levels (e.g. "Mastered / Developing / Needs Improvement") to avoid false precision.



- **Periodic Diagnostics:** After a set of questions or at logical breakpoints (e.g., after covering a domain or after 5 questions), present a brief diagnostic summary. For instance: *"You have demonstrated strong skills in Quadratic Equations and Linear Functions. You struggled with Rational Expressions. Upcoming questions will focus on improving that area."* This narrative can be generated from the FCDM profile (taking the lowest concepts and highest concepts and giving encouraging or corrective statements). The system can also suggest taking a short break or reviewing a concept if fatigue or repeated errors are detected (for student well-being).
- **Summative Feedback:** At the end of an adaptive test or practice session, produce a comprehensive report. This report should include the **knowledge state** of the student – essentially what the student can confidently do and what they are ready to learn next aleks.com. For example: a mastered skills list (perhaps "Can solve linear equations, factor basic polynomials...") and a "ready to learn" list (skills just beyond current mastery for the student, which could be studied next) aleks.com. This approach follows best practices in cognitive diagnostic assessment, where results are given in terms of skills rather than just a score.
- **Engagement and Remediation:** Use the adaptive nature of the system for feedback as well – if a student continually errs on a concept, the platform could recommend a specific video or textbook section to review, or even switch to a brief instructional mode on that concept before more questions. Incorporating links to learning resources in the feedback loop can turn assessment into a learning opportunity.

*Tools & Best Practices:* Ensure the platform's UI can display dynamic feedback – many LMS or quiz systems have provisions for per-question feedback and post-quiz reports. If not built-in, a custom front-end might be needed. Keep feedback constructive and focused on controllable aspects (effort, strategies) to encourage a growth mindset. Also, make use of the **fuzzy nature of FCDM outputs**: because the model gives a continuous score, you can trigger feedback when a concept mastery crosses a threshold (e.g., once a student achieves  $>0.8$  in a skill, acknowledge it: "You've mastered this concept!" which can motivate them). Regularly update feedback content as the curriculum or common student errors evolve; this maintenance ensures feedback stays relevant and effective.

## Step 6: Integration with the Digital Platform

Integrating all these components into the existing digital assessment platform is crucial for a seamless user experience. The platform is math-capable, which likely means it supports LaTeX or MathML for formulas and perhaps interactive items. Our integration strategy covers content import, adaptive logic integration, and data flow:

- **Content Ingestion:** Import the structured questions (from Step 1) with their metadata (Step 2) into the platform's question bank. Many modern assessment systems support bulk import via formats like **QTI (Question and Test Interoperability)** or CSV/XML with custom fields. We will write a converter to transform our question JSON into the platform's required format, including all tags. If direct database import is possible (and safe), that's an alternative for large-scale upload. Retain the association between question IDs in the platform and our internal mapping (needed for the adaptive engine to reference the right items).
- **Math Content Handling:** Ensure that all mathematical notation is correctly rendered. If the platform uses MathJax or similar, verify that the LaTeX in questions is displaying properly. For any images (graphs, geometric figures), upload those into the platform's media storage and ensure the links in the questions point correctly. This step is mostly one-time, but whenever new questions are added, repeat the process and double-check formatting.
- **Adaptive Engine Integration:** The rule-based adaptive logic (Step 4) and FCDM calculations could run on an external service or possibly within the platform if it allows custom code. Two integration patterns are possible:
  - *In-Platform Scripting:* If the platform has an API or scripting plugin, embed our adaptive algorithm there. For example, some platforms allow JavaScript in questions or have a workflow for adaptive assessments. We would use the student's response data to call our selection function which then dynamically loads the next question.

- *External Service via API:* Alternatively, implement the adaptivity in a separate service (a web server running our Python/Java logic). After each question submission, the platform makes an API call to this service with the student's response and current state. The service, in turn, replies with the next question's identifier and any feedback to display. The platform then uses this info to present the next question to the student. This requires that the platform be somewhat flexible (some assessment systems support a "remote adaptive test" through LTI or custom integration).
- **Data Synchronization:** The student response data (correct/incorrect, time taken, etc.) should feed into the FCDM model in real-time. If running internally, the model can update in memory. If external, the service should maintain session state or consult a shared database. Also, record the final skill estimates and performance summary back into the platform's records (e.g., as part of the grade book or a learning analytics dashboard). This might involve using the platform's APIs to push summary results or storing them in a linked learning record store.
- **User Management and Enrollment:** Leverage the platform's user accounts and class structures. Ensure that when a student launches the adaptive test, their identity is passed to our system so we load the correct profile (especially important if a student takes multiple sessions over time – we want to remember their previous mastery state). Integration might use standards like **LTI (Learning Tools Interoperability)** if our adaptive module is an external tool, so that single sign-on and grade return are handled smoothly.
- **Testing and Sandbox:** Before full deployment, test the integration in a sandbox environment with dummy students. Check that the handoff between platform and adaptive logic is seamless (no significant delays or glitches when moving to the next question), and that all data (responses, scores, mastery updates) are correctly captured. Also test failure cases (e.g., if the adaptive service is down or a question fails to load) to ensure the system can fail gracefully (perhaps fall back to a linear test or notify the user).

*Tools & Best Practices:* Use the platform's **developer documentation** extensively to guide this integration. If the platform provides analytics, integrate our data to that so instructors can see the rich diagnostic info. For maintainability, containerize any external services (using Docker/Kubernetes), which helps in scaling horizontally if many concurrent students are using the adaptive test. Make sure to comply with data privacy policies – student performance data should be protected and only used for the intended diagnostic purposes.

## Scalability and Maintenance Considerations

Building this system is an ongoing process. Below are best practices to ensure the solution scales to large numbers of questions and students, and remains up-to-date:

- **Content Scaling:** As the question bank grows (potentially thousands of questions), maintain an organized content repository. Use unique IDs and modular files (e.g., one file per topic or per chapter) to ease updates. Automate the PDF extraction and tagging pipeline for new additions – for example, if new questions come in PDF, run them through the same LLM-assisted process and then have them reviewed. This minimizes manual effort for future expansions.
- **Knowledge Graph Evolution:** Treat the knowledge graph as a living entity. Curriculum changes or new insights might require adding or redefining concepts. Have a periodic review (perhaps annually or per semester) where educators can propose updates to the concept graph. Changes should be version-controlled, and when updating, consider the impact on FCDM (re-training or adjusting the model might be necessary if the set of skills changes). Designing the system to handle multiple versions of the knowledge structure can be useful (e.g., if different courses use slightly different skill maps).

- **Model Maintenance (FCDM):** Over time, accumulate a dataset of student responses. Use this to refine the FCDM parameters for better accuracy. Regularly evaluate the model's diagnostic predictions against real performance (e.g., does the model correctly identify weaknesses that correlate with exam results?). If the platform expands to include more diverse question types (like open-ended), assess if the FCDM needs adjustments or if new models (Neural CDM, etc.) should be incorporated. The modular design (EduCDM library use) allows swapping or upgrading the diagnosis model without overhauling the whole system.
- **Rule Tuning:** Monitor the adaptive testing outcomes. Analytics can show if, say, students are getting stuck in certain loops or if the first questions are often too hard or too easy. Use such data to tweak the rule thresholds or sequence. For instance, we might find the difficulty jump after two correct answers is too high and causing frustration – we can then moderate that rule. A/B testing different rule strategies in small user groups could identify what yields the best learning gains or engagement. Keeping the rule logic in a configurable format (like a JSON or YAML defining thresholds) can allow easy updates without code changes.
- **System Performance:** As usage grows (imagine hundreds of students taking adaptive tests simultaneously), ensure the backend can handle it. Optimize the critical path – for example, the next-question selection should be quick. If using an external service, horizontal scaling (multiple instances behind a load balancer) might be needed. Also, implement caching where possible (e.g., caching the FCDM computation for the last question to quickly update for the next question, rather than recalculating from scratch each time).
- **Platform Integration Maintenance:** Keep track of any updates to the digital platform (if it's a third-party product). Platform updates could affect our integration (APIs change, etc.), so allocate effort for occasional maintenance testing after platform upgrades. Establish a relationship with the platform vendor if possible, to stay informed of new features (perhaps the platform will introduce its own adaptive testing capabilities that we can leverage, or better support for knowledge graphs).

- **User Support and Training:** To scale to many instructors and students, provide documentation or training materials about the adaptive system. Teachers should understand how to interpret the diagnostic reports, and students should get a brief orientation on how the adaptive quiz works (for example, explain that skipping or guessing may affect their learning path, to encourage genuine effort). A well-informed user base will make the system more effective and reduce confusion.
- **Quality Assurance:** Finally, maintain a feedback loop where educators can flag any issue in questions (e.g., wrong answer key or bad tagging) and content can be quickly fixed. A robust item versioning system will allow edits without disrupting ongoing tests (e.g., changes take effect for new sessions). Regularly audit a sample of questions and their metadata for quality. Scaling up is not just about quantity but ensuring consistent quality across the item bank.

By following this implementation plan, we will transform static PDF questions into a rich, adaptive learning system. This system will diagnose student knowledge using FCDM's nuanced skill modeling and serve tailored questions through rule-based logic, all integrated seamlessly into the existing platform. The result is an efficient, personalized assessment experience: students answer fewer questions while the system gains a precise understanding of their abilities and provides targeted feedback for improvement

[sciencedirect.com](https://www.sciencedirect.com) [ivtg.nl](https://www.ivtg.nl) . Through careful planning and ongoing maintenance, the platform can continue to grow and adapt, ultimately leading to better learning outcomes and user engagement.

### Sources:

- Rekha Ramesh et al., "Semi-Automatic Generation of Metadata for Items in a Question Repository," IEEE T4E (2014). – Describes a system for tagging questions with cognitive level, question type, content, and difficulty, achieving high accuracy in automated tagging [cse.iitb.ac.in](https://www.cse.iitb.ac.in) .
- Steven Moore et al., "Automated Generation and Tagging of Knowledge Components from MCQs," L@S 2024. – Discusses the importance of mapping questions to Knowledge Components for adaptive learning and mastery estimation [arxiv.org](https://arxiv.org) [arxiv.org](https://arxiv.org) .
- Qi Liu et al., "Fuzzy Cognitive Diagnosis Framework," ACM TIST 2018. – Introduces FuzzyCDF as a continuous multi-dimensional model for student cognition, using Q-matrix mappings of items to skills [github.com](https://github.com) [github.com](https://github.com) .

- Jean-Claude Falmagne et al., *"The Assessment of Knowledge in Theory and in Practice,"* ALEKS Corporation (2008). – *Outlines knowledge space theory and the idea of identifying the set of problems a student can solve ("can do") and the set they are ready to learn next* [aleks.com](https://www.aleks.com) , which informs our adaptive feedback design.

## Citations

 **Semi-Automatic Generation of Metadata for Items in a Question Repository**

<https://www.cse.iitb.ac.in/~sri/papers/qRepository-t4e2014.pdf>

 **Semi-Automatic Generation of Metadata for Items in a Question Repository**

<https://www.cse.iitb.ac.in/~sri/papers/qRepository-t4e2014.pdf>

 **GitHub - bigdata-ustc/EduCDM: The Model Zoo of Cognitive Diagnosis Models,...**

<https://github.com/bigdata-ustc/EduCDM>

 **GitHub - bigdata-ustc/EduCDM: The Model Zoo of Cognitive Diagnosis Models,...**


<https://github.com/bigdata-ustc/EduCDM>

 **GitHub - bigdata-ustc/EduCDM: The Model Zoo of Cognitive Diagnosis Models,...**

<https://github.com/bigdata-ustc/EduCDM>

 **Unsupervised Automated Generation and Tagging of Knowledge Components F...**

<https://arxiv.org/pdf/2405.20526>

 **Unsupervised Automated Generation and Tagging of Knowledge Components F...**

<https://arxiv.org/pdf/2405.20526>

 **The Assessment of Knowledge in Theory and in Practice**

[https://www.aleks.com/about\\_aleks/Science\\_Behind\\_ALEKS.pdf](https://www.aleks.com/about_aleks/Science_Behind_ALEKS.pdf)

 **The Assessment of Knowledge in Theory and in Practice**


[https://www.aleks.com/about\\_aleks/Science\\_Behind\\_ALEKS.pdf](https://www.aleks.com/about_aleks/Science_Behind_ALEKS.pdf)

 **The Assessment of Knowledge in Theory and in Practice**

[https://www.aleks.com/about\\_aleks/Science\\_Behind\\_ALEKS.pdf](https://www.aleks.com/about_aleks/Science_Behind_ALEKS.pdf)

 **Measurement precision and user experience with adaptive versus ...**

<https://www.sciencedirect.com/science/article/pii/S0191886924001351>

 **FAQ - iVTG**

<https://ivtg.nl/toetsinformatie/faq/>

## All Sources



cse.iitb.ac



github



arxiv



aleks



sciencedirect



ivtg