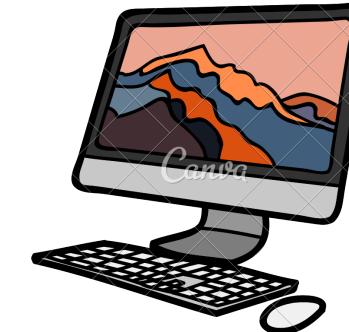


IMPERIAL



AI-enabled Personalised Learning Platform

Yin Xian Liang (Sofia)



Personalised learning, empowered by technology and community, that transforms education.

Supervisors: Dr. Konstantinos Gkoutzis, Dr. Ovidiu Serban



Objectives

Provide a **user-based** support platform for the sharing of learning resources.

Build a **unique learner model** based on learner's profile and learning data.

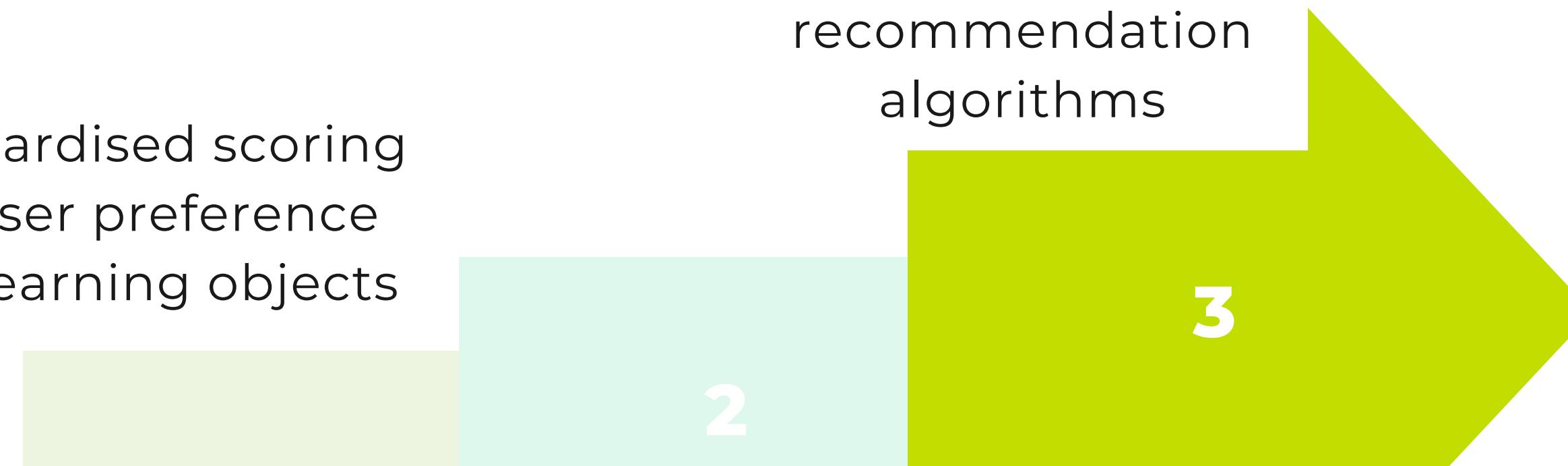
Engineer **appropriate prompts** to promise quality of **auto-generated content**.

Customise selection of learning content to boost user satisfaction and motivation.



Contributions

Standardised scoring
for user preference
and learning objects



Comparison study of
recommendation
algorithms

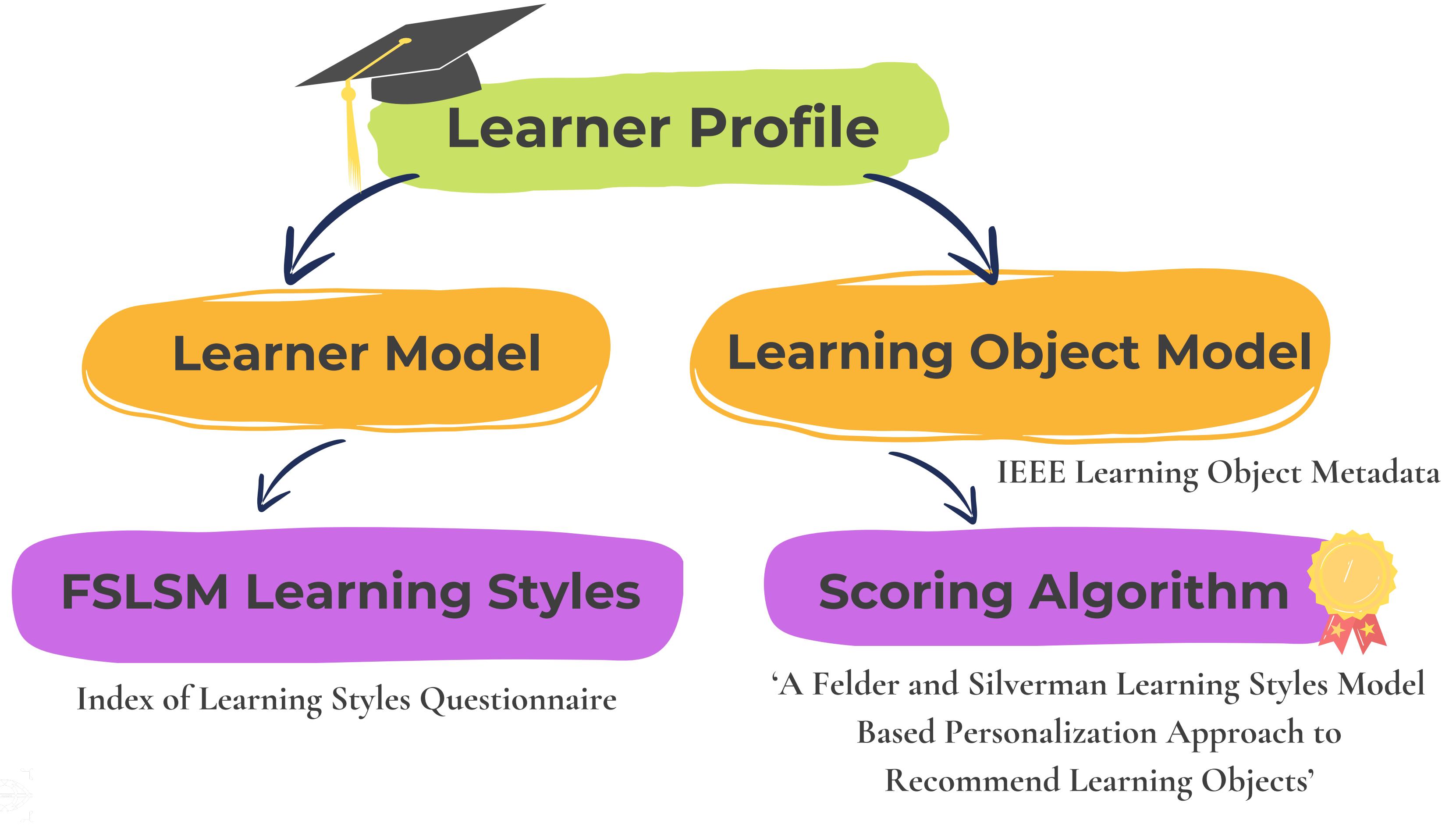
Optimal
metaprompting
strategy

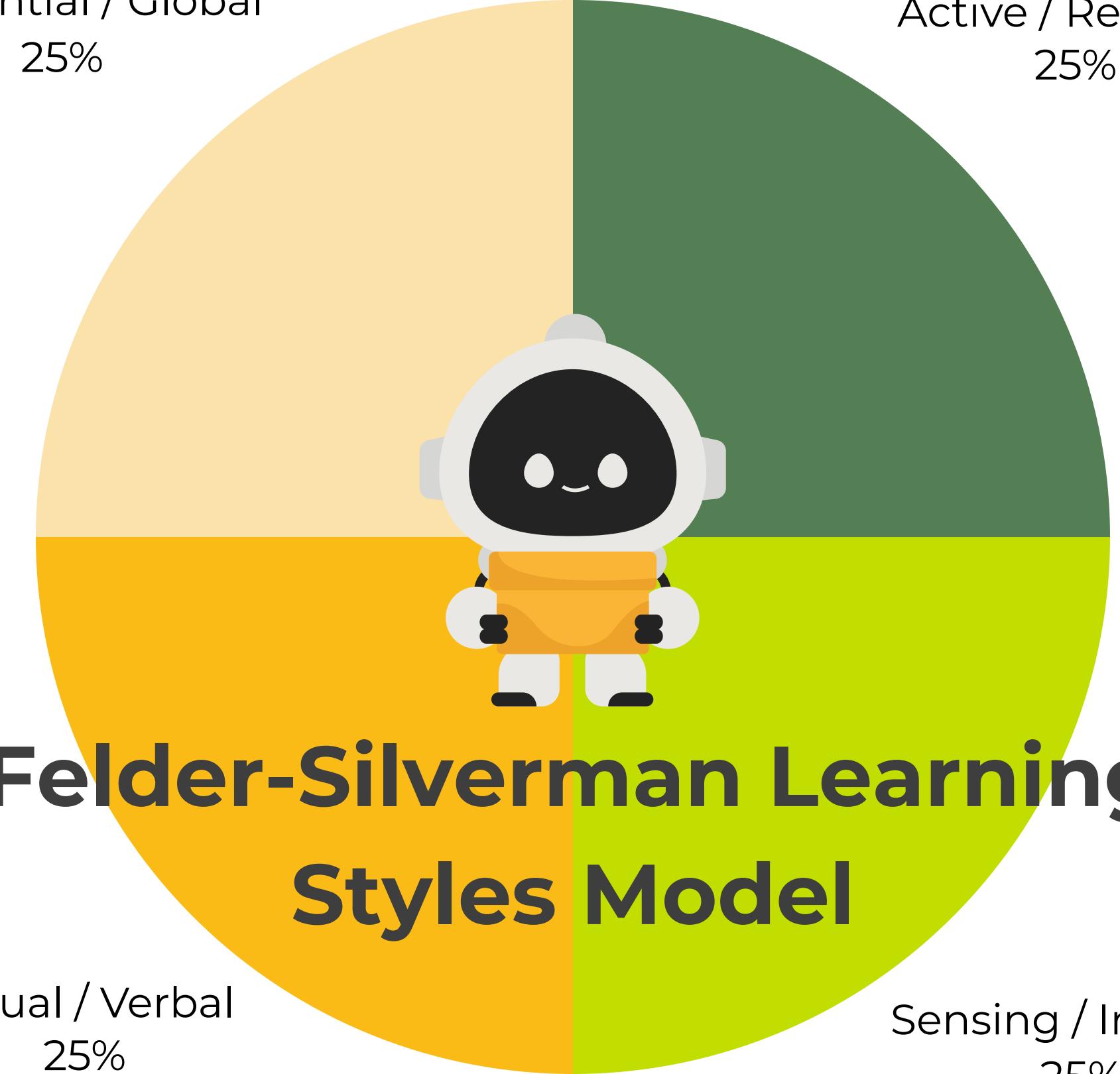


Learner Profile



Main aspect of customisation that offers adapted presentation of learning materials





Sequential / Global

25%

Active / Reflexive

25%



Felder-Silverman Learning Styles Model

Visual / Verbal

25%

Sensing / Intuitive

25%

III

Index of Learning Styles Questionnaire



- 44 Questions
- 4 dimensions, -11 to +11
- normalised from 0 to 1
- vector of 8 real numbers

```
learningPreferences: {  
    active: 0.63636363  
    reflexive: 0.36363636  
    sensing: 0.36363636  
    intuitive: 0.63636363  
    visual: 0.90909090  
    verbal: 0.09090909  
    sequential: 0.09090909  
    global: 0.90909090 }
```

Learning Object

```
_id: ObjectId('6669a7b54b1d16676220b010')
▶ general : Object
▶ lifecycle : Object
▶ technical : Object
▼ educational : Object
  interactivityType : "expositive"
  learningResourceType : "narrative text"
  interactivityLevel : "low"
  context : "higher education"
  difficulty : ""
▶ content : Object
▼ score : Object
  act : 0
  ref : 1
  vis : 0
  ver : 1
  sen : 0
  int : 1
  seq : 1
  glo : 0
  --v : 0
  createdAt : 2024-06-12T13:50:45.531+00:00
  updatedAt : 2024-06-12T13:53:43.079+00:00
  ▶ content : Object
    text : "We will now attempt to make the guessing game more user friendly.
    ...
    link : ""
  ▶ audio : Object
  ▶ questionnaire : Object
    ▶ openAI : Array (empty)
    ▶ vertexAI : Array (empty)
  ▶ exercise : Object
  ▶ glossary : Object
  ▶ challenge : Object
  ▶ transcript : Object
  ▶ description : Object
  embed : false
  aiGenerated : false
  image : ""
  video : "
```

Figure 3.3: The further categorisation of learning object content

Figure 3.2: Example of a learning object stored in the project database

IEEE Learning Object Metadata Standard

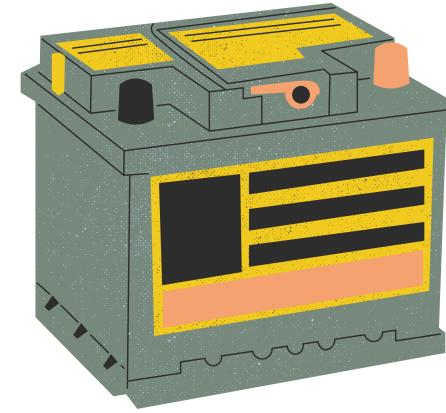
Nr.	IEEE LOM Standard	Value Space
4.1	format	MIME types based on IANA (e.g., video/mpeg, text/html)
5.1	interactivity type	active, expositive, mixed
5.2	learning resource type	exercise, simulation, questionnaire, diagram, figure, graph, index, slide, table, narrative text, exam, experiment, problem statement, self assessment
5.7	difficulty	very easy, easy, medium, difficult, very difficult

'A Felder Silverman Learning Styles Model Based Personalization Approach to Recommend Learning Objects'

0 = Active, 1 = Reflexive

Learning Resource Type value	Active/Reflective dimension	Sensing/Intuitive dimension	Visual/Verbal dimension	Sequential/Global dimension
Exercise	0	0	0	0
Simulation	0	0	0	0
Questionnaire	0	1	0	0
Diagram	1	0	0	0
Figure	1	0	0	0
Graph	1	0	0	0
Index	1	0	0	1
Slide	1	1	0	0
Table	1	0	0	0
Narrative text	1	1	1	0
Exam	0	1	0	0
Experiment	0	0	0	0
Problem statement	0	1	0	0
Self assessment	0	1	0	0
Lecture	1	1	0	1

Modified lecture to be visual and global based on user ratings



Automated Content Generation



A curated bank of lesson content that offers a variety of materials, providing an optimal learning experience to diverse learners.

Comparison of Large Language Models (LLMs)

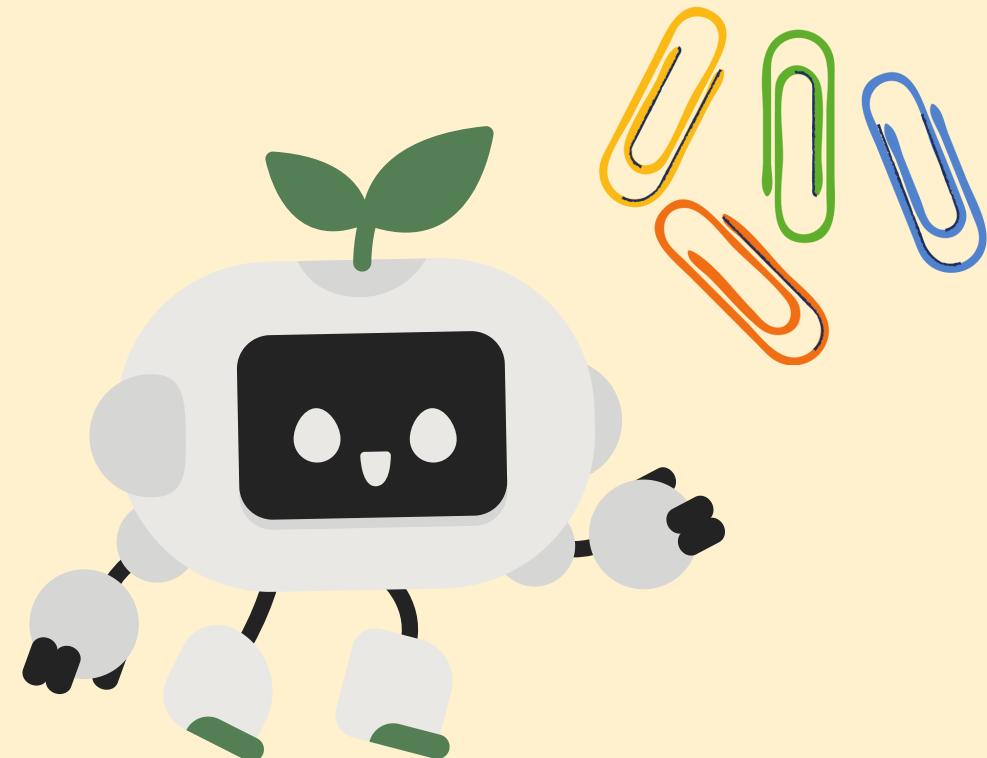
Model	Context window	Modality		Price (<=128K context window)	
Gemini-1.5-Pro	Top quality up to 2M tokens	Input	Image	\$0.001315 / image	
			Video	\$0.001315 / second	
Gemini-1.0-Pro	32K tokens	Input	Text	\$0.00125 / 1k characters	
			Audio	\$0.000125 / second	
Text-to-Speech		Output	Text	\$0.00375 / 1k characters	
Speech-to-Text		Input	Image	\$0.0025 / image	
			Video	\$0.002 / second	
		Output	Text	\$0.000125 / 1k characters	
			Text	\$0.000375 / 1k characters	
		Input	Text		
			Audio	\$0.000016 / byte	
		Output	Image		
			Text	\$0.024 / min	

Table 4.1: VertexAI models API requests pricing table as of 14th June 2024 [53] [54] [55]

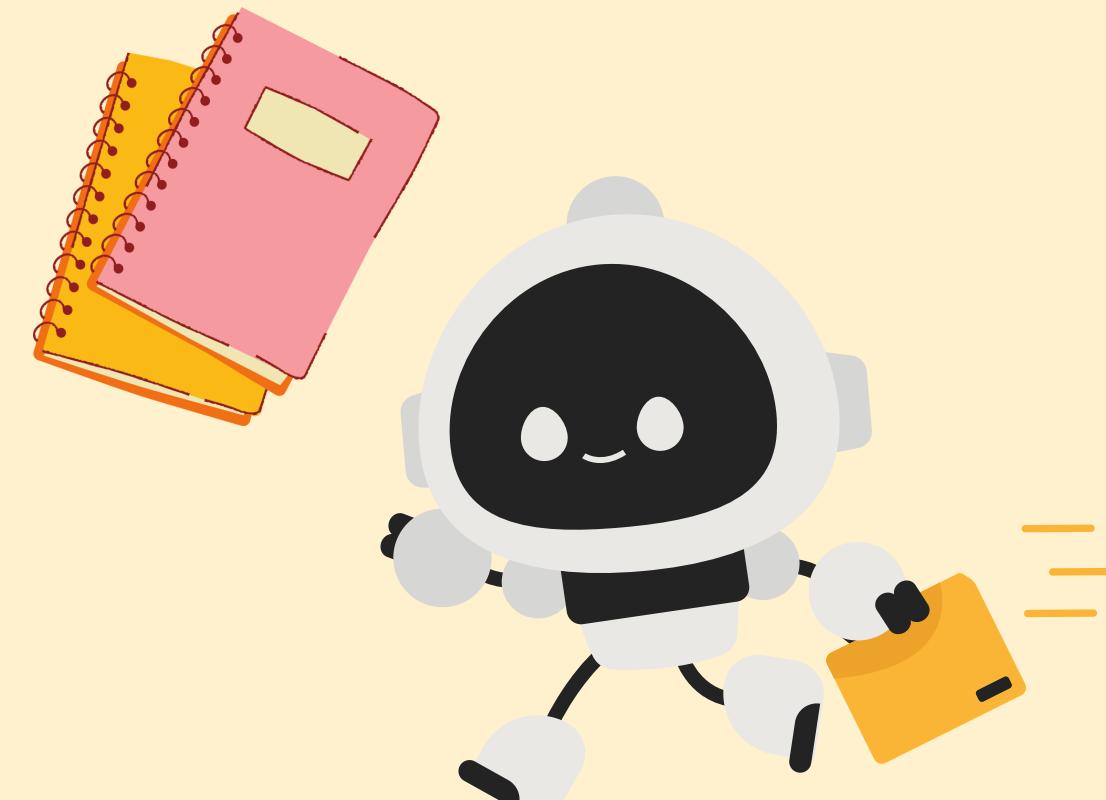
Model	Context window	Modality		Price (<=128K context window)
GPT-4o	128K tokens	Input	Image	\$0.003825 / 1024 x 1024 px
			Text	\$0.005 / 1K tokens
GPT-3.5	16,385 tokens	Output	Text	\$0.015 / 1K tokens
			Text	\$0.0005 / 1K tokens
TTS-1		Output	Text	\$0.0015 / 1K tokens
			Text	
Whisper-1		Input	Text	
			Audio	\$0.015 / 1K tokens
Dall-E-3		Input	Text	
			Image	\$0.040 / 1024px x 1024px

Table 4.2: OpenAI models API requests pricing table as of 14th June 2024 [56]

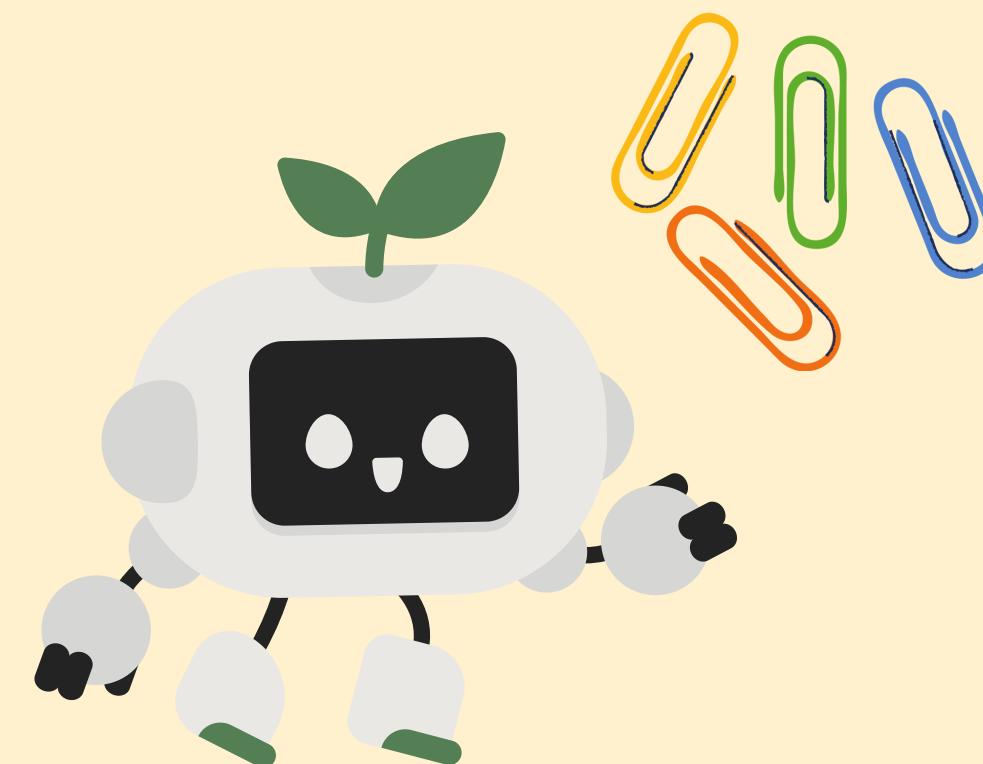
Component-wise Content Generation



Lesson-wise Content Generation



Component-wise Content Generation



Transitioning same content from one medium to another

Text-to-Podcast

Increases digestibility, enabling visual learners to engage with a body of text



Google Cloud Text-to-Speech



OpenAI's TTS



= It's all about instructing the computer!

Summary:

Programming ≠ coding

Programming ⊃ coding

Programming involves:

1. Understanding and formulating the problem
2. Designing an algorithm to solve the problem
3. Implementing the algorithm

Machine code is what the computer understands, but us writing machine code will be tedious. High-level languages have been developed to help make programming easier.

Translators convert high-level languages to low-level code:

1. **Compilers** pre-translate codes into a set of low-level instructions, which is then executed.
2. **Interpreters** translate and execute code line by line, on the fly.

Programming languages need to be unambiguous.

Spend more time thinking about the problem and developing a clear algorithm before coding.





It's all about instructing the computer!

This lecture transcript is generated by AI.

Lecture Notes on Programming

Introduction to Programming

- **Initial Thoughts on Programming:**
- Common misconception: Programming is just typing code.
- Reality: Programming involves much more than coding; it's a creative process.

Key Aspects of Programming

- **Understanding the Problem:**
- Take a task or a problem.
- Dissect and understand its components.
- Formulate and define the problem clearly.

- **Designing an Algorithm:**
- An algorithm is a sequence of instructions to solve the problem.
- Create a detailed plan (algorithm) to address the problem.

- **Implementation (Coding):**
- Write a program to implement the algorithm.
- A program is an algorithm expressed in a language a computer can understand.

Video Transcript

Rewrite transcript in format of lecture notes

It's all about instructing the computer!



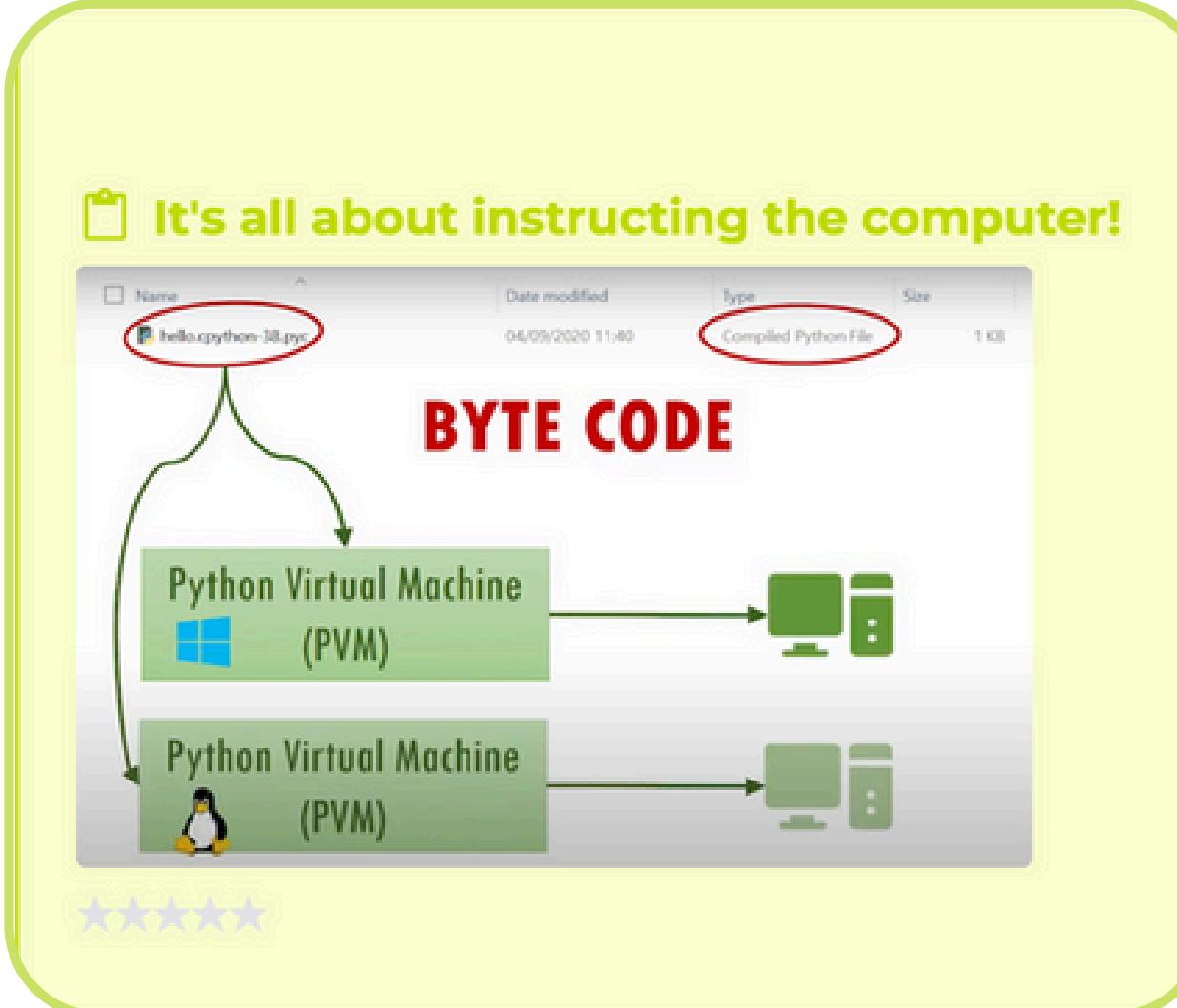
Watch on YouTube

★★★★★

Slide-to-Lecture Notes



Write alternative lecture notes based on the diagram



It's all about instructing the computer!

This slide description is generated by AI.

Python Execution Model

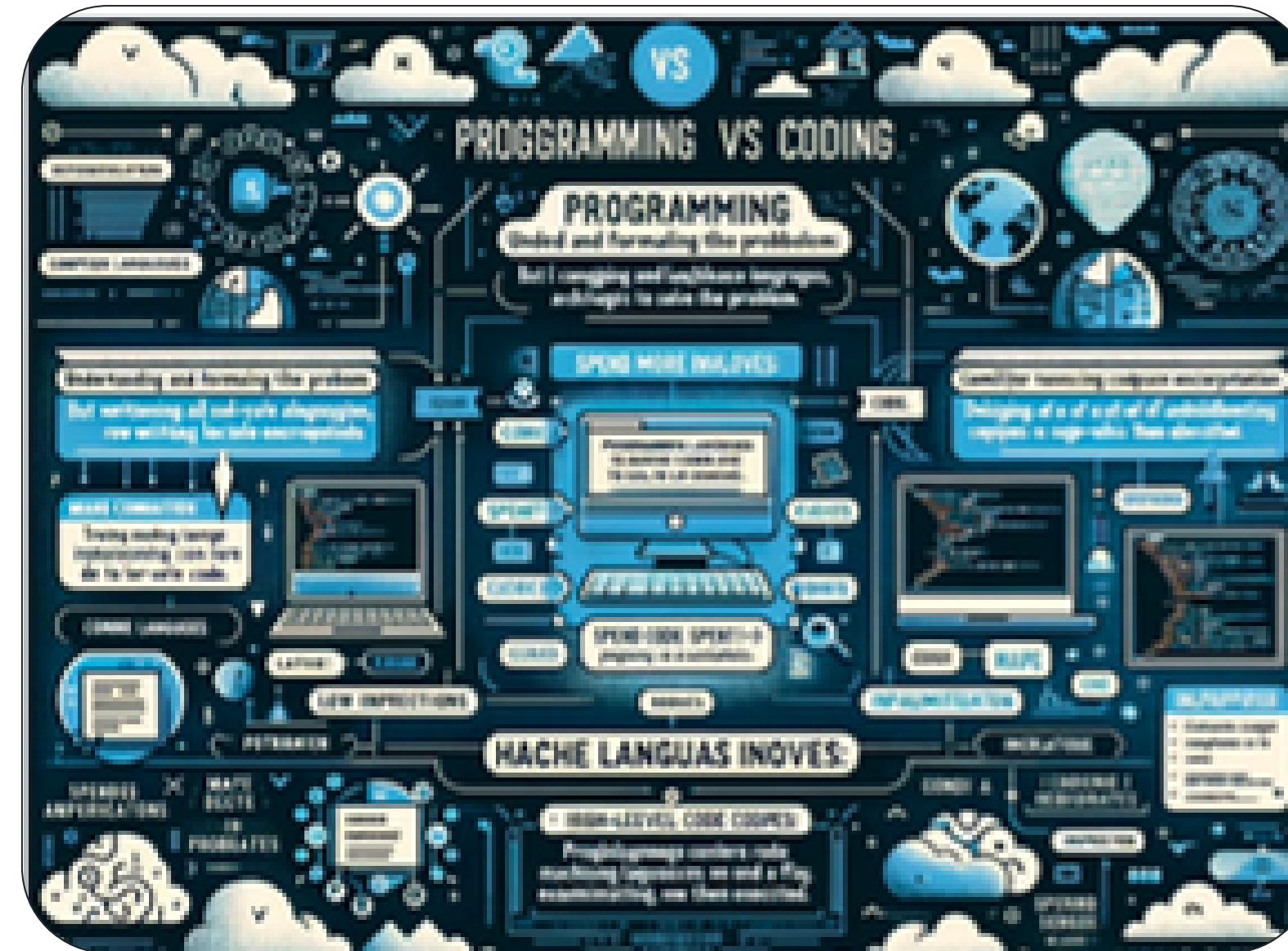
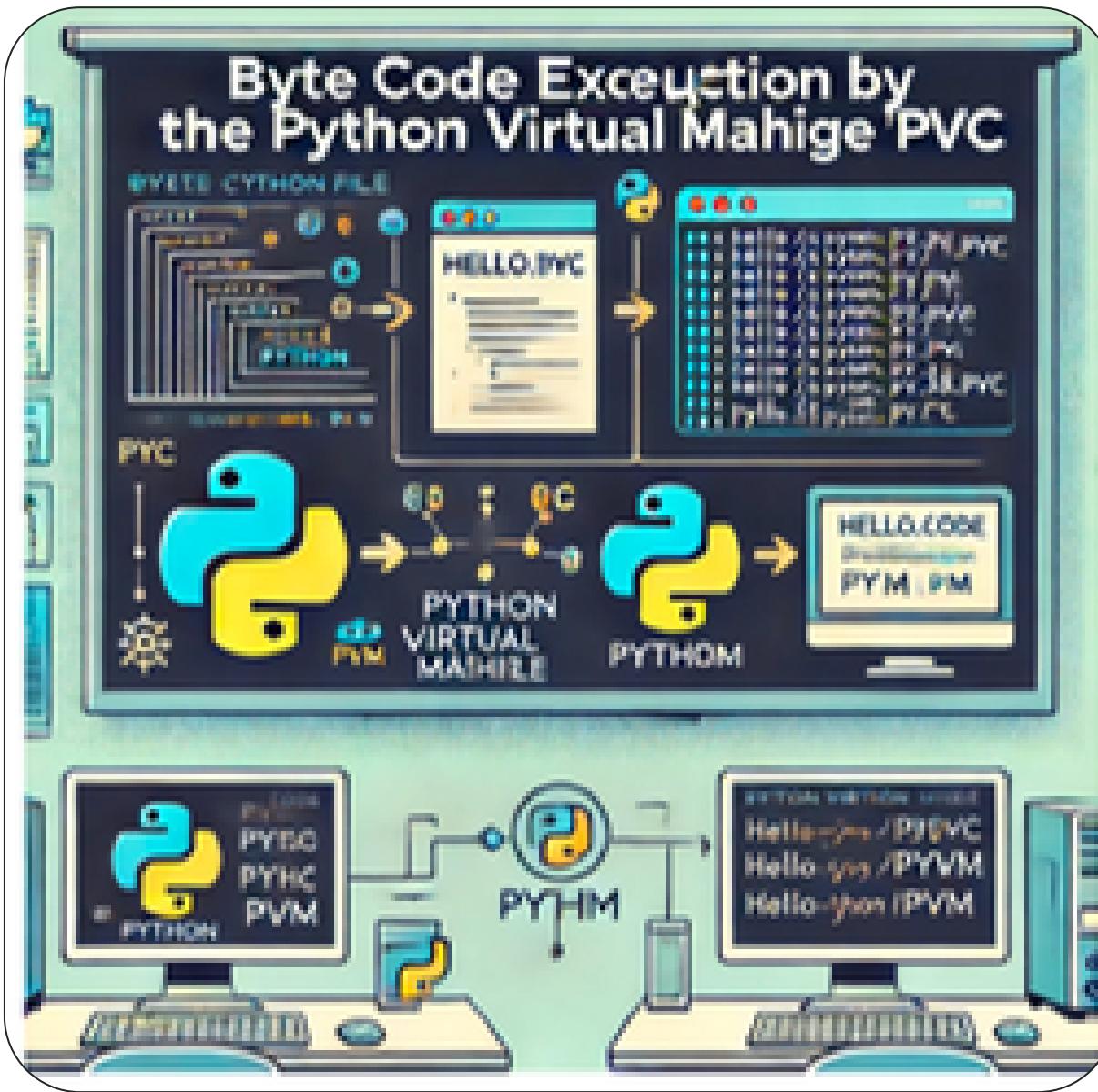
Python code is not directly executed by the operating system. Instead, it undergoes a two-step process:

- Compilation to Byte Code:**
 - When a Python program ('.py' file) is executed, it is first compiled into byte code.
 - Byte code is a low-level, platform-independent representation of the code.
 - The compiled byte code is saved in a '.pyc' file (as shown in the image with 'hello.cpython-38.pyc').
- Execution by Python Virtual Machine (PVM):**
 - The byte code is then executed by the Python Virtual Machine (PVM).
 - The PVM is an interpreter that reads and executes the byte code instructions.
 - The PVM is platform-specific, meaning there are different PVMs for Windows, Linux, macOS, etc.



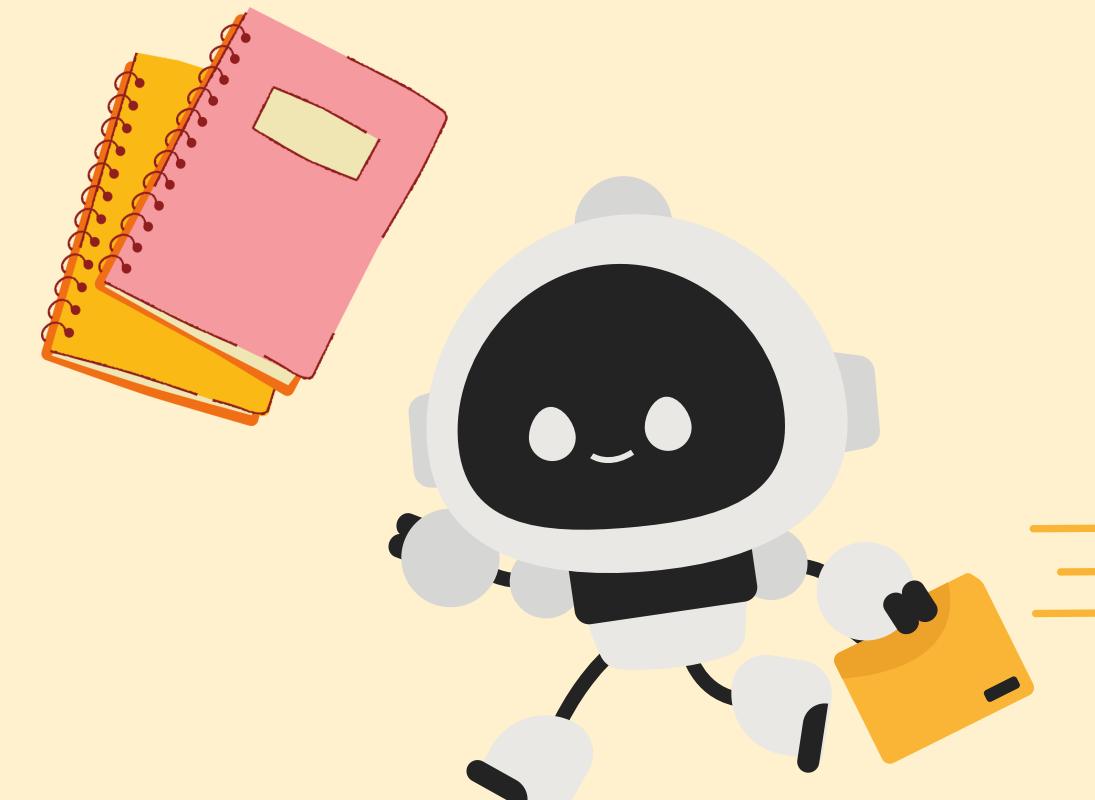
Lecture Notes-to-Slides

Struggled to present coherent text or effectively represent key message of text through relevant images



Generate more
interactive elements
for active learning

Lesson-wise Content Generation



Multiple Choice Questions ✨

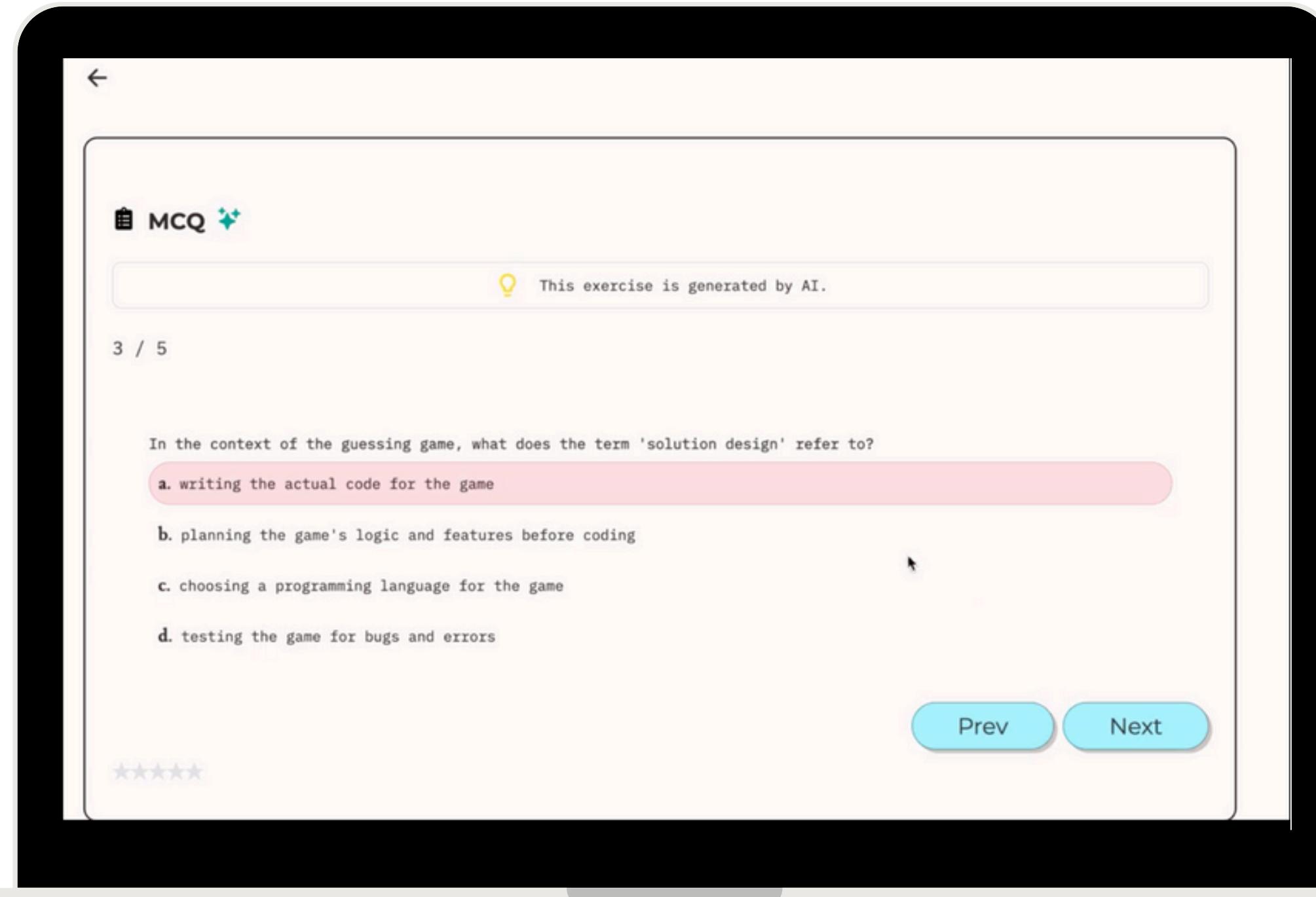
A multi-stage approach

01 Paraphrase Generation

02 Keyword Extraction

03 Question Generation

04 Distractor Generation



Multi-stage Prompting

MCQ Generation

01 Paraphrase Generation

02 Keyword Extraction

03 Question Generation

04 Distractor Generation

Paraphrase the given context \${lessonText}

Extract the keywords from the given context
\${paraphraseGeneration}

Role:

You are a computer science university lecturer.

Task:

Create a list of 5 one-sentence multiple choice questions based on the paraphrased content
\${paraphraseGeneration} and correct answers
\${keywordExtraction}, that \${levelPrompt}.

Meta Prompting

Structuring an effective prompt

01 Role
Prompting

02 Direct Task
Specification

03 Input Data

04 Expected
Outcome

Paraphrase the given context \${lessonText}

Extract the keywords from the given context
\${paraphraseGeneration}

Role:

You are a computer science university lecturer.

Task:

Create a list of 5 one-sentence multiple choice
questions based on the paraphrased content
\${paraphraseGeneration} and correct answers
\${keywordExtraction}, that \${levelPrompt}.

JSON Formatting

Prompt Output Formatting

Output:

Format the response as a parsable json array
for the MCQs as follows:
[{ "question" : "...",
 "choices" : [{
 "text": "...",
 "value": 1 (if correct) 0 (if wrong)}],
 ...] } ...]

Bloom's Taxonomy

Test Different Levels of Understanding

The image shows a mobile application interface with a dark background. In the center, there is a white modal window with rounded corners. At the top of the modal, it says "How much do you know about 'Intro to programming'" with a small "X" icon in the top right corner. Below this, there is a rating scale consisting of five yellow stars, with the third star being filled. At the bottom of the modal is a blue "Submit" button. Behind the modal, there are two main cards. The left card has a brown background and features the text "Design a guessing game" in orange, with a small blue icon of a person's head and shoulders below it. The right card also has a brown background and features the text "Intro to programming" in orange, with a small blue icon of a computer monitor below it. At the bottom of the screen, there is a light gray navigation bar with a small blue button containing a white plus sign on the right side.

Design
a guessing game

Intro
to programming

How much do you know about 'Intro to programming'

Submit

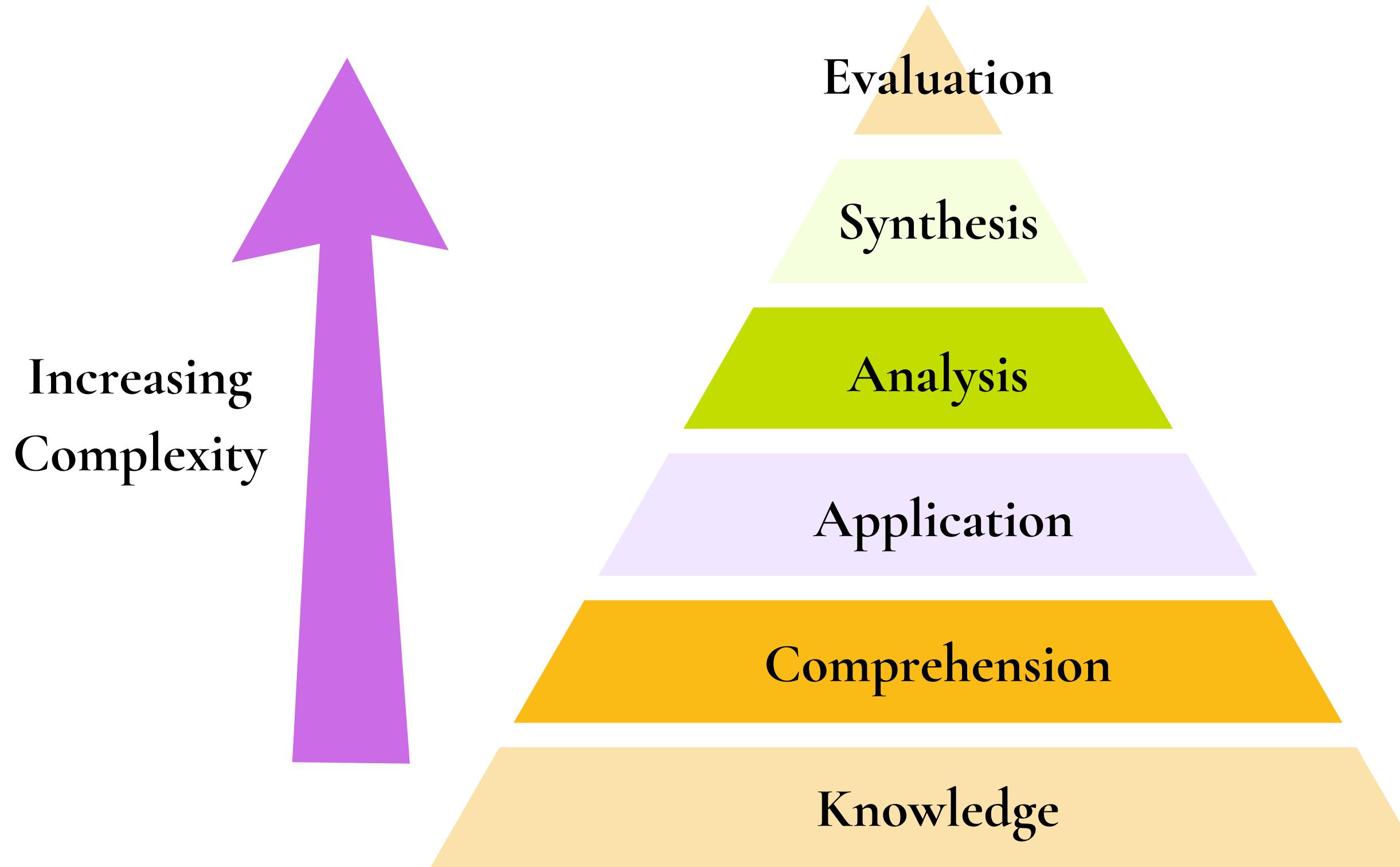
Let's dive straight into designing a game... because life is too short to waste!

Really... what exactly is programming?

+

Bloom's Taxonomy

Test Different Levels of Understanding

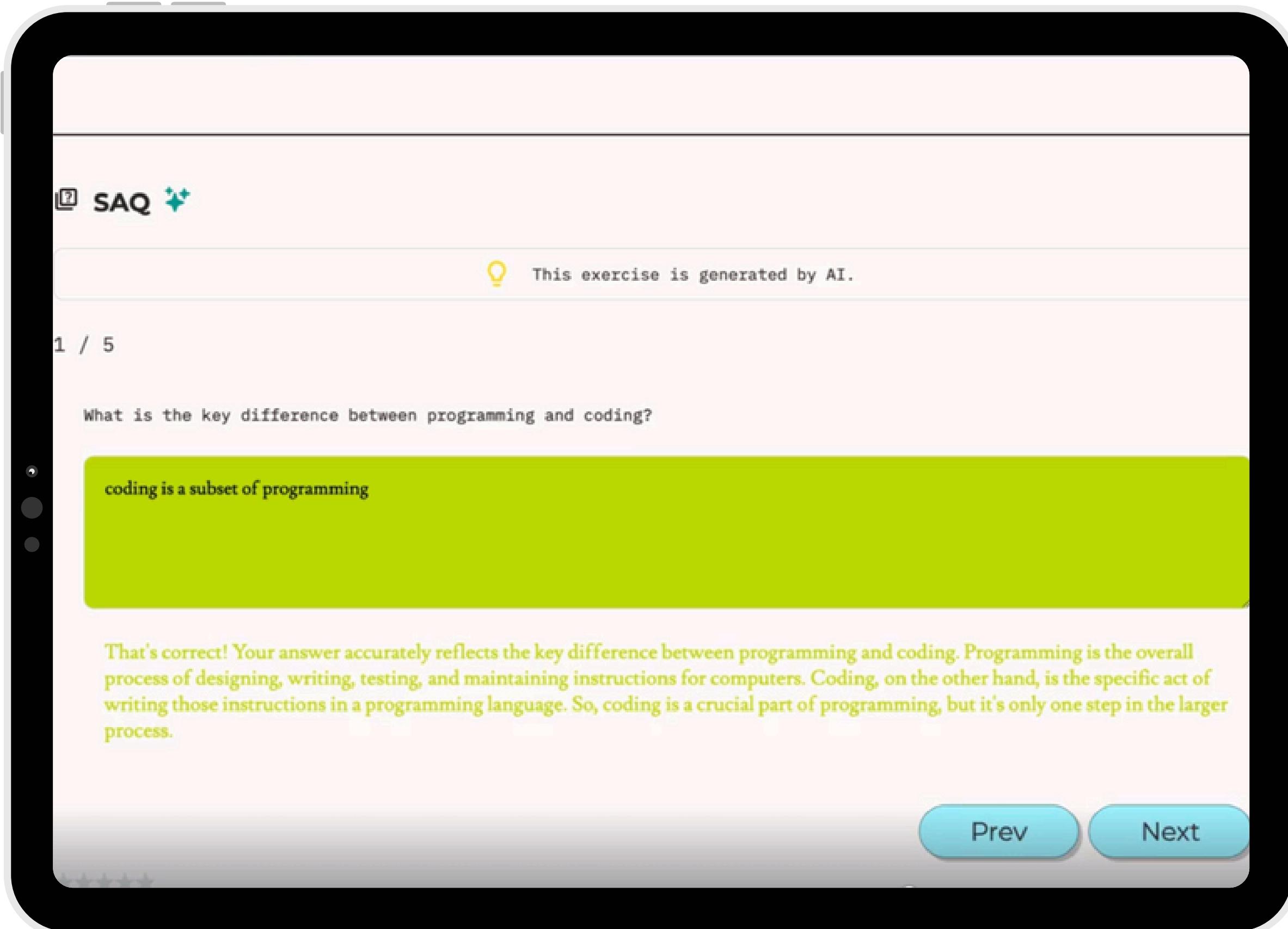
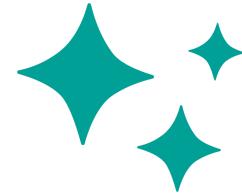


Bloom's Taxonomy

Test Different Levels of Understanding

IEEE LOM Difficulty Level	Level Prompt
very easy	assess recall of core concepts through memory of terminology, facts, setting, methods
easy	assess comprehension by translation of attained knowledge to similar instances
medium	assess application of knowledge to new situations, problem-solving, and analysis
hard	assess evaluation of knowledge, judgement , and dissection, through argumentative exercises
very hard	assess creation of new knowledge, synthesis of information, and design of new patterns, by producing purposeful outcomes by rearranging patterns

Question & Answer

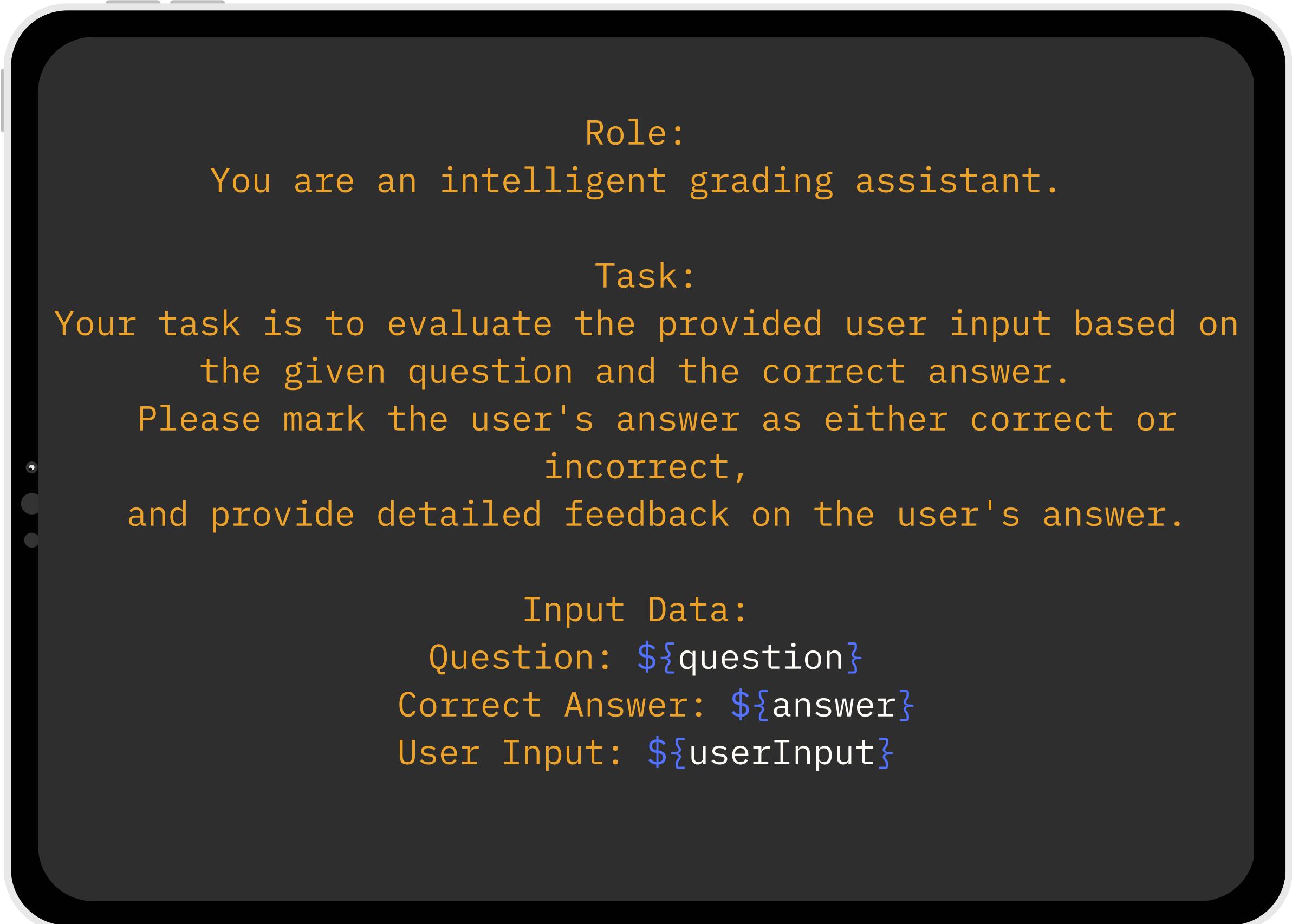


A smartphone screen showing a Q&A application. The top bar has three dots. The main area shows a section titled "SAQ" with a teal star icon. Below it, a yellow lightbulb icon indicates "This exercise is generated by AI." A progress bar shows "1 / 5". The question "What is the key difference between programming and coding?" is displayed. A green button contains the answer "coding is a subset of programming". At the bottom, a yellow text box provides feedback: "That's correct! Your answer accurately reflects the key difference between programming and coding. Programming is the overall process of designing, writing, testing, and maintaining instructions for computers. Coding, on the other hand, is the specific act of writing those instructions in a programming language. So, coding is a crucial part of programming, but it's only one step in the larger process." Navigation buttons "Prev" and "Next" are at the bottom right.

01 Open-ended questions

02 AI-generated feedback
(Based on answers aligned with lesson)

Intelligent Grading Assistant



01 Role Prompting

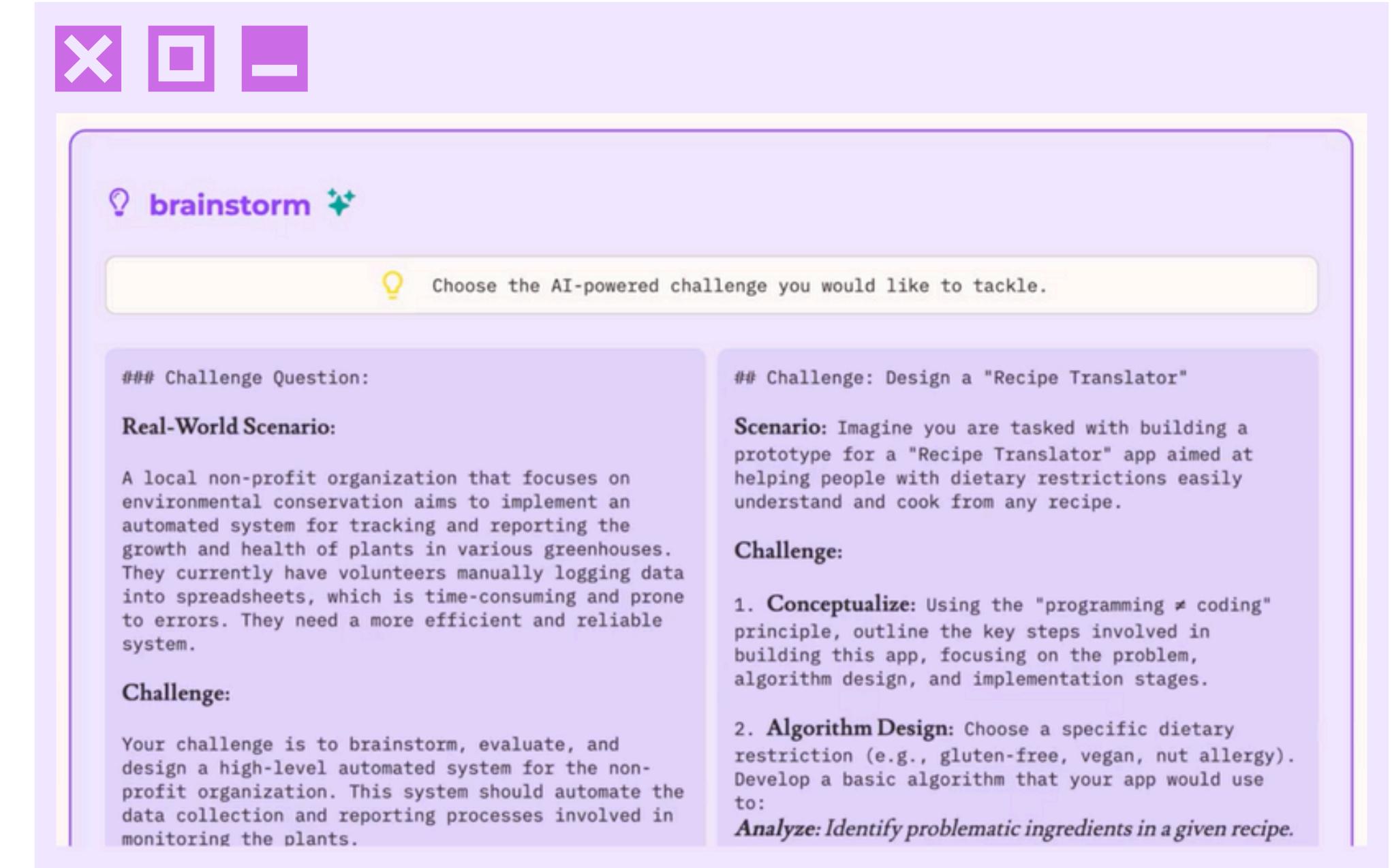
02 Direct Task Specification

03 Input Data

04 Expected Outcome

Challenge ✨

Brainstorm, Evaluate, Create



X □ -

💡 brainstorm ✨

💡 Choose the AI-powered challenge you would like to tackle.

Challenge Question:

Real-World Scenario:

A local non-profit organization that focuses on environmental conservation aims to implement an automated system for tracking and reporting the growth and health of plants in various greenhouses. They currently have volunteers manually logging data into spreadsheets, which is time-consuming and prone to errors. They need a more efficient and reliable system.

Challenge:

Your challenge is to brainstorm, evaluate, and design a high-level automated system for the non-profit organization. This system should automate the data collection and reporting processes involved in monitoring the plants.

Challenge: Design a "Recipe Translator"

Scenario: Imagine you are tasked with building a prototype for a "Recipe Translator" app aimed at helping people with dietary restrictions easily understand and cook from any recipe.

Challenge:

- Conceptualize:** Using the "programming × coding" principle, outline the key steps involved in building this app, focusing on the problem, algorithm design, and implementation stages.
- Algorithm Design:** Choose a specific dietary restriction (e.g., gluten-free, vegan, nut allergy). Develop a basic algorithm that your app would use to:
Analyze: Identify problematic ingredients in a given recipe.

Glossary

Table Form / Index Cards

Glossary

This glossary is generated by AI.

Programming	The comprehensive process of problem-solving using computers, encompassing problem understanding, algorithm design, and implementation.
Coding	The act of writing instructions in a programming language to be executed by a computer. It is a subset of programming.
Algorithm	A step-by-step procedure or formula for solving a problem. It represents the logic behind a program.
Machine Code	Low-level instructions that a computer's processor can directly understand and execute.
High-Level Language	A programming language designed to be human-readable and easier to use than machine code. Examples include Python, Java, and C++.
Compiler	A translator that converts high-level code into machine code before execution. The resulting machine code can be executed multiple times without recompilation.
Interpreter	A translator that converts and executes high-level code line by line, on the fly. It does not create a separate executable file.

★★★★★

Glossary

This glossary is generated by AI.

Programming

1/7



Learning Object Recommendation



Personalise display based on initialised learning preferences and user ratings.

Content-based Filtering

Predict ratings of new learning object by clustering previously rated learning objects by same user

K-means Clustering

Hamming Distance

Get Nearest Cluster

Cosine Similarity

Get Top-N in Nearest Cluster

Pearson Correlation

Predict New Learning Object Rating

Pearson Correlation

Collaborative Filtering

Predict ratings of new learning object based on ratings by other users on platform that share similar learning preferences.

K-means Clustering

Euclidean Distance

Get Nearest Cluster

Cosine Similarity

Get Top-N in Nearest Cluster

Pearson Correlation

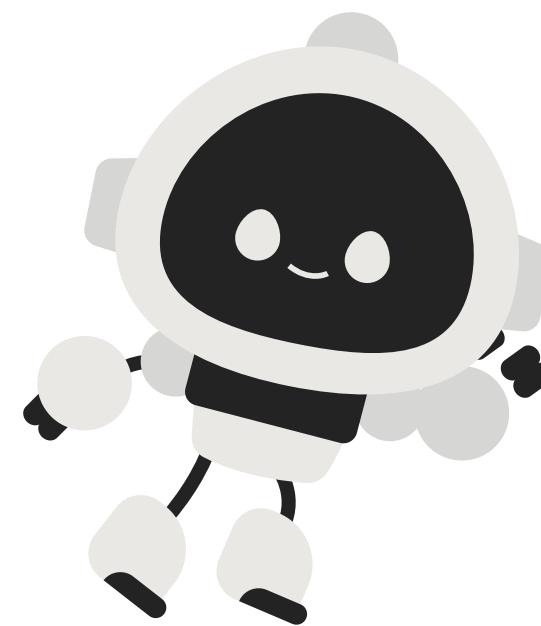
Predict New Learning Object Rating

Pearson Correlation

Hybrid Filtering

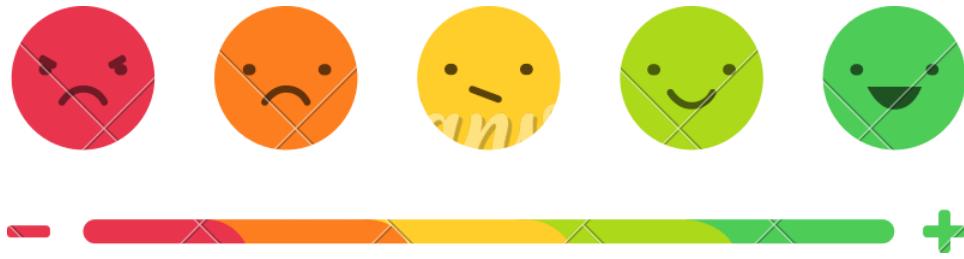
Predict ratings of new learning object based on both previous user ratings and learning preferences of other learners.

```
if (ratingsForLO.length === 0 && allRatingsByUser.length === 0) {  
    predictedRating = predictInitialRating(userLS, loScore);  
} else if (ratingsForLO.length === 0) {  
    predictedRating = await calculateContentPrediction(allRatingsByUser, loScore, accessToken);  
} else if (allRatingsByUser.length === 0) {  
    predictedRating = calculateCollaborativePrediction(nearestClusterLS, userLS, ratingsForLO);  
} else {  
    const r1 = calculateCollaborativePrediction(nearestClusterLS, userLS, ratingsForLO);  
    const r3 = await calculateContentPrediction(allRatingsByUser, loScore, accessToken);  
    predictedRating = weight * r1 + (1 - weight) * r3;  
}
```



Recommendation & Sequencing

- Exercise/Example-first
- Filter LOs above median rating

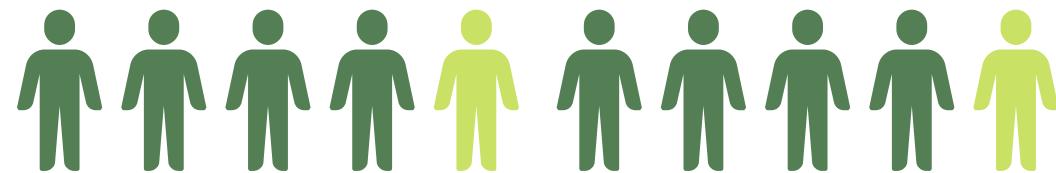


Evaluation



Verify **effectiveness** and **accuracy** of **filtering** algorithms, and assess **benefit** of **AI-generated** content.

Evaluation Setup

10 test subjects


Imperial College London

diverse courses and learning styles

20 learning objects per lesson

Test Subjects	1	2	3	4	5	6	7	8	9	10
Active	0.182	0.318	0.636	0.818	0.545	0.182	0.636	0.182	0.364	0.000
Sensing	0.636	0.636	0.909	0.909	0.636	1.000	0.364	0.909	0.545	0.364
Visual	0.636	0.955	0.909	0.636	0.727	0.818	0.909	0.636	0.818	0.455
Sequential	0.455	0.455	0.409	0.727	0.636	0.727	0.091	0.636	0.636	0.182

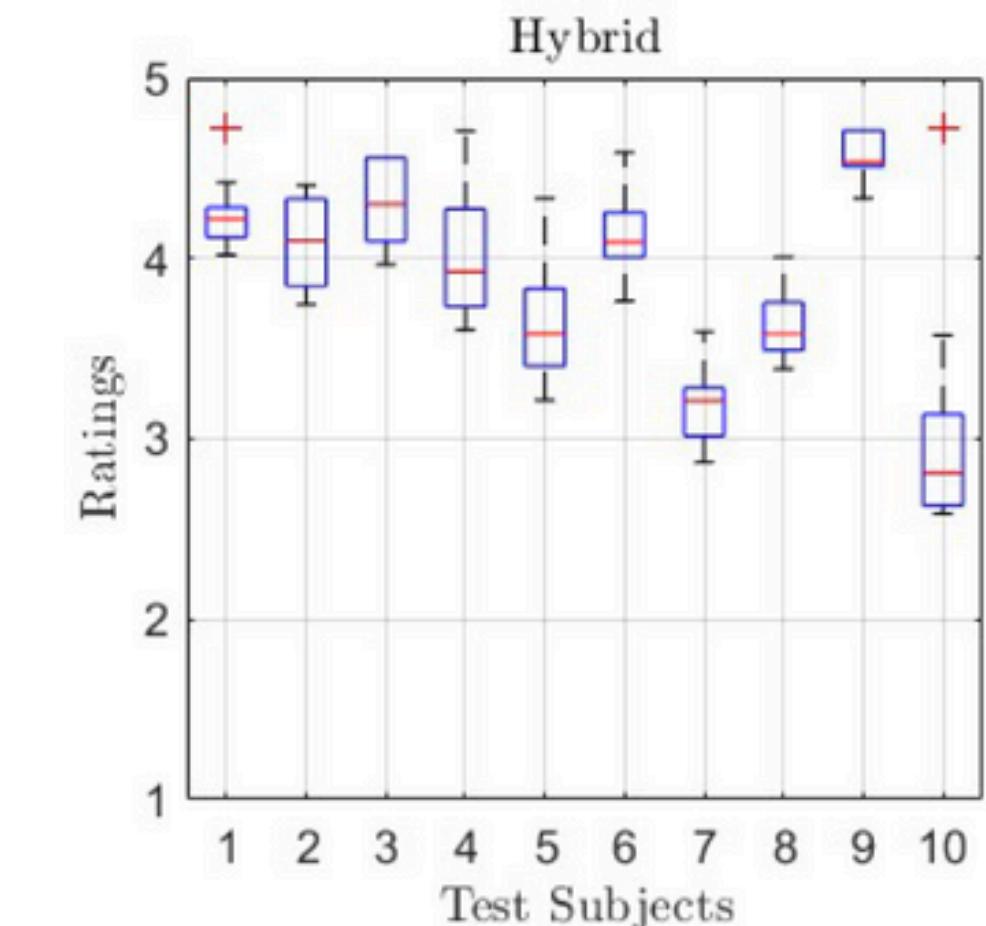
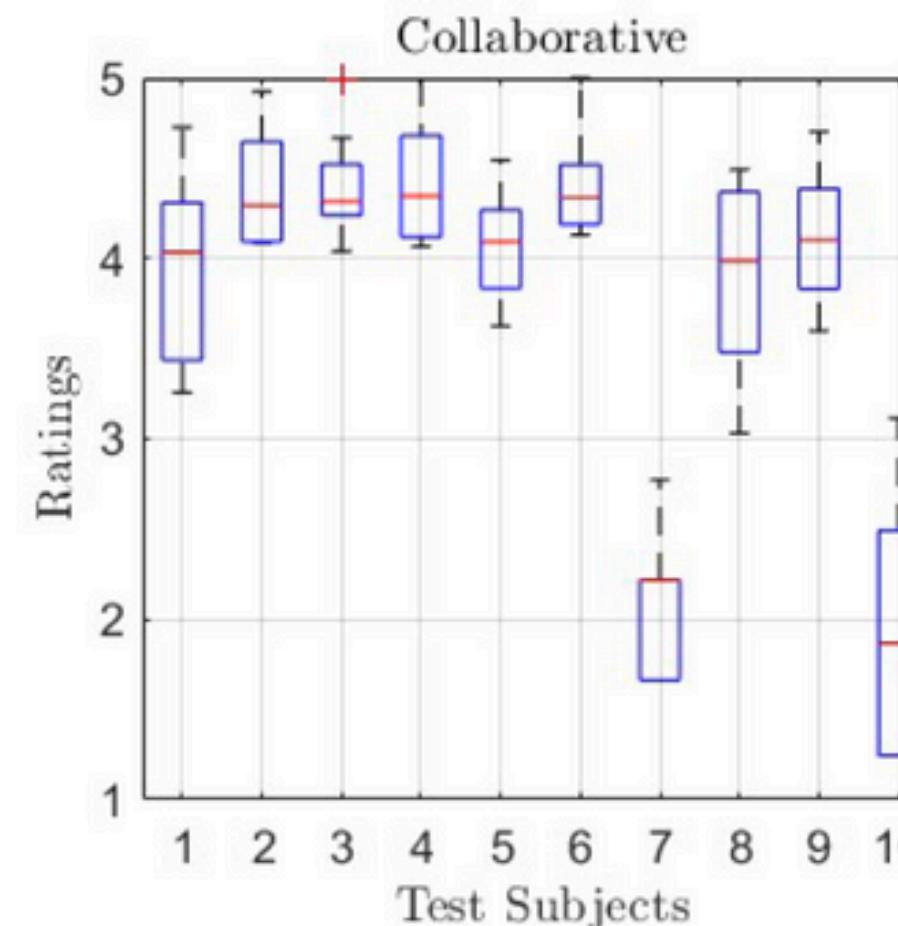
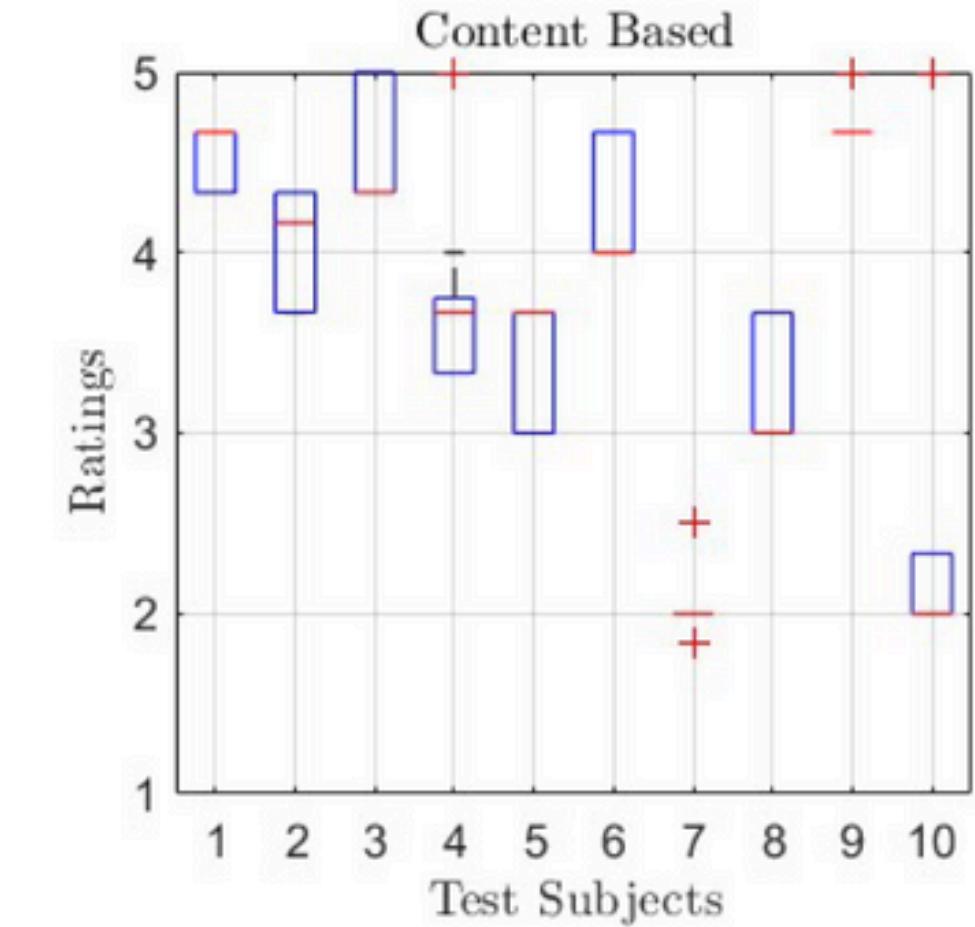
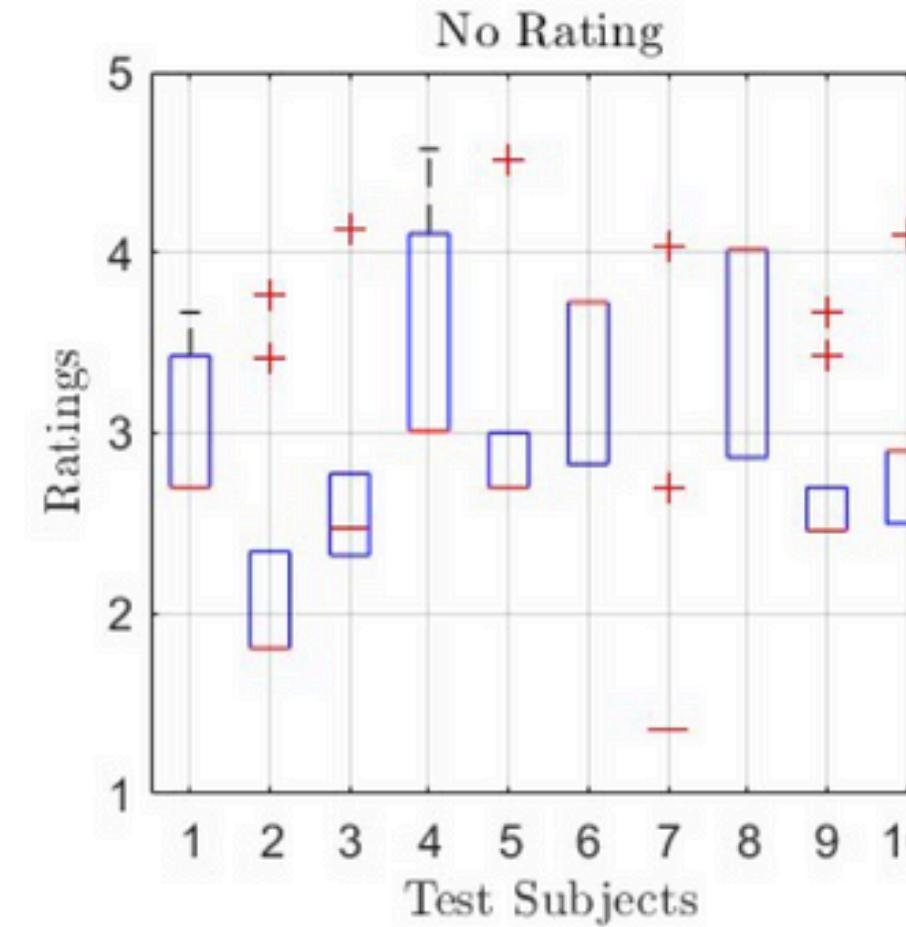
Table 6.1: Learning Preferences of the Test Subjects

Note: The scores range from 0 to 1 for each FSLSM dimension. E.g: 0 indicates least active and most reflexive, 1 indicates least reflexive and most active (opposite polarity).

Comparison of Overall

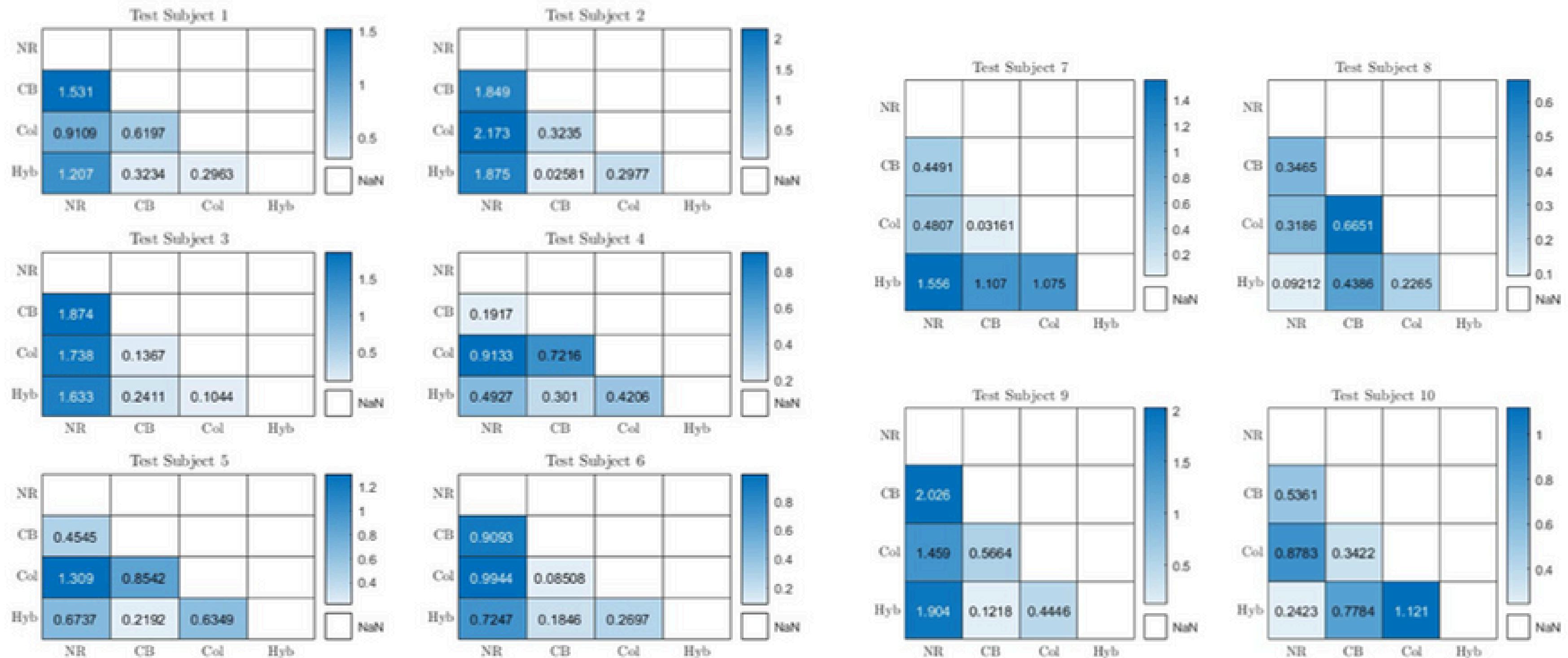
Ratings

Learning objects above median
normalised by min-max
normalisation



Effectiveness of Recommendation Algorithms

Tukey's Honest Significant Difference (HSD) Test for all 10 test subjects



Mean Squared Error (MSE)

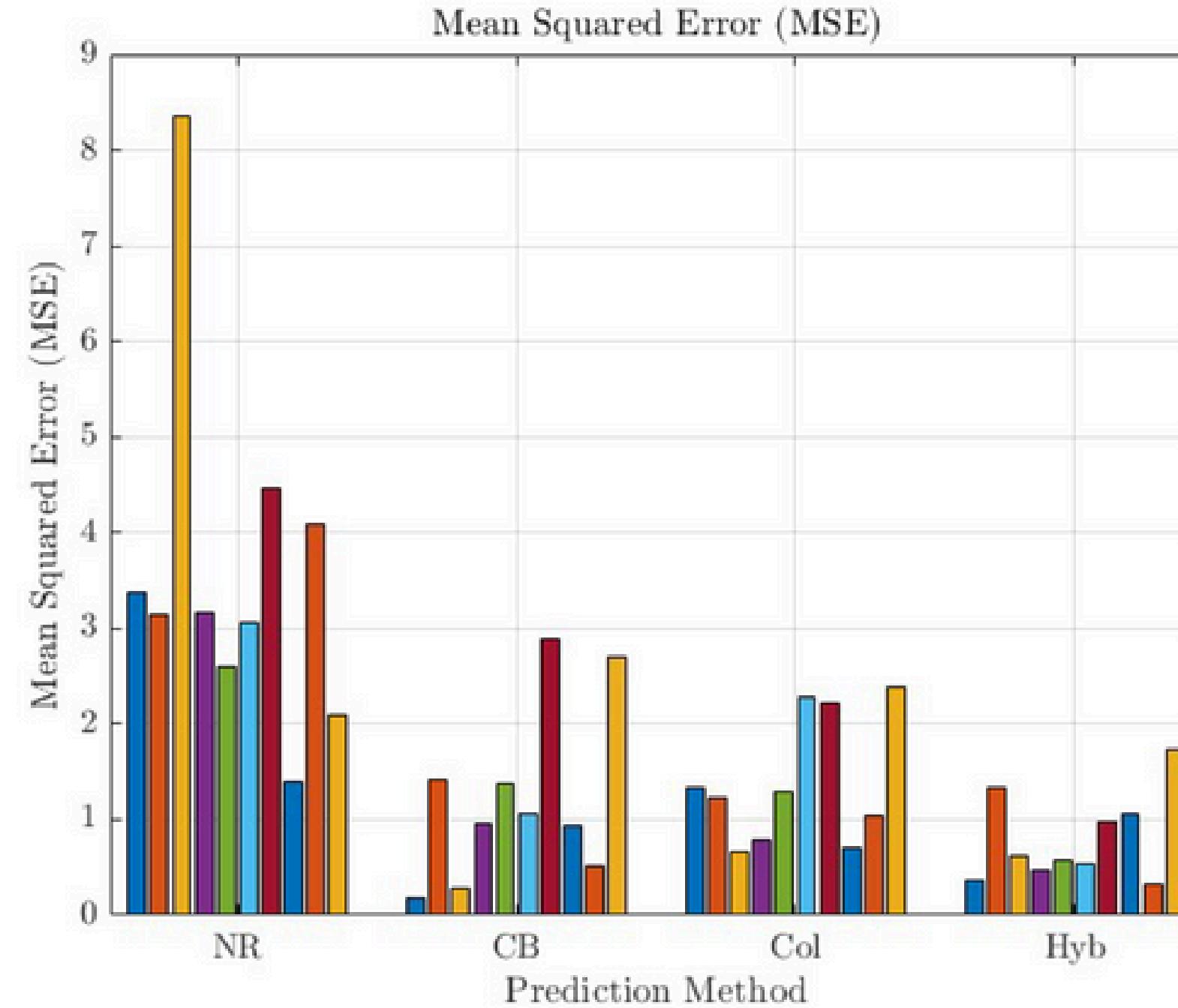


Figure 6.3: Mean squared errors between predicted ratings and actual ratings of recommended learning objects across four recommendation algorithms (No Rating, Content-Based, Collaborative, Hybrid).

The Ratio
7 : 2

AI-generated Content

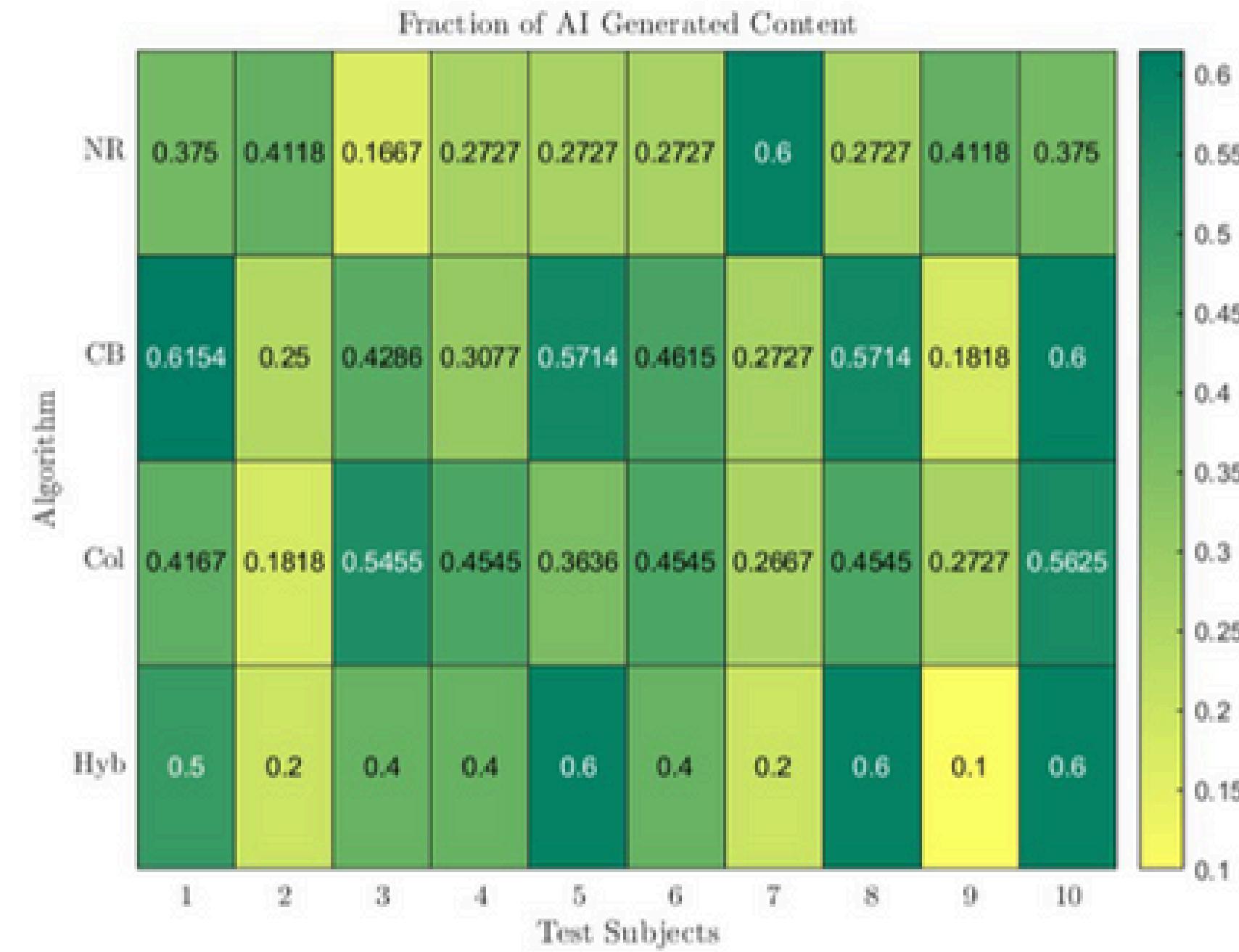


Figure 6.4: Fractions of AI-generated content in learning object recommendations for each test subject across four algorithms.



Conclusion



This project uncovered an effective personalised learning strategy using a **fully algorithmic recommendation technique** and enhanced the learning experience with AI through **automated content generation**.

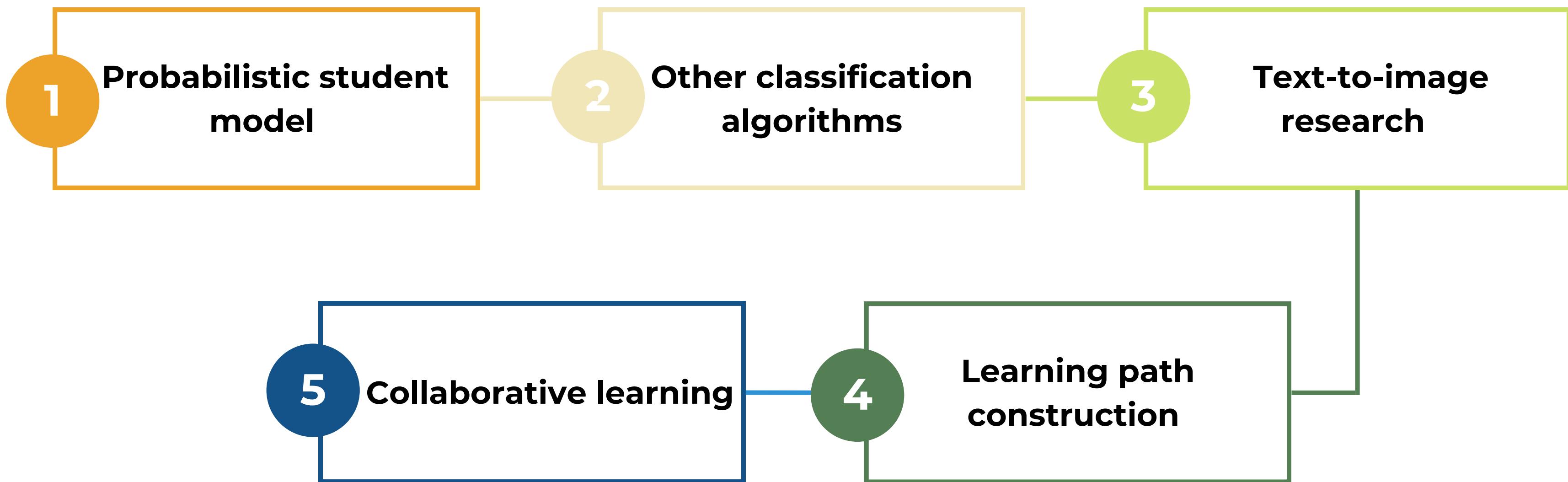
Findings

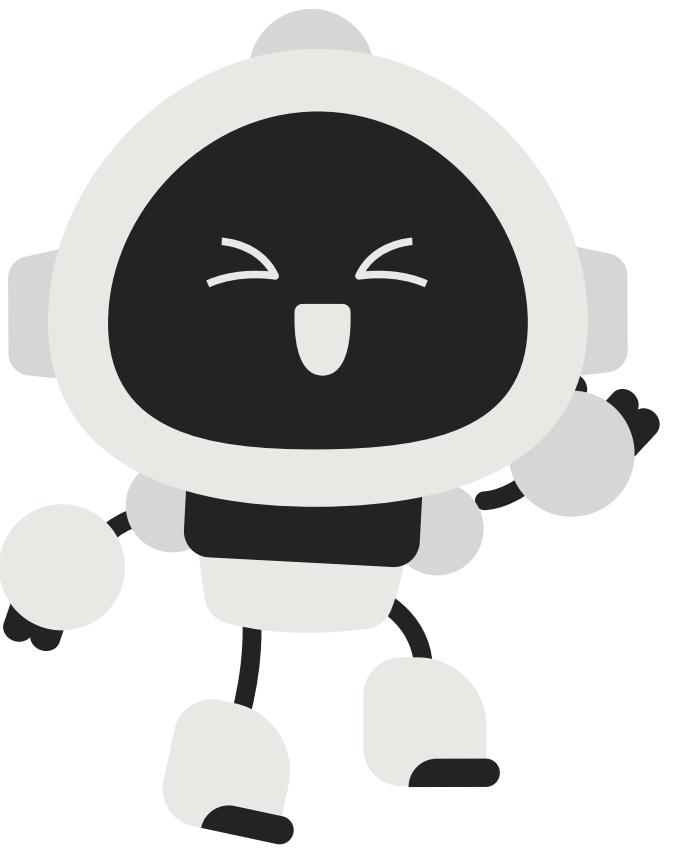
LLMs by OpenAI and VertexAI each have their **own strengths** in generating a diverse set of learning objects that cater to **various learning styles**

Hybrid filtering learning object recommendation ($\alpha = 0.5$) provides best personalisation, with **highest** and **most accurate** overall ratings

AI-generated content is **effective** in catering to diverse learning styles

Future Works





thank you!