IMPERIAL

MEng Individual Project

Final Report

Imperial College London

Department of Computing

Enhancing Puzzle-Solving with User-Tailored AI Assistance: A Study on Nonogram Learning Outcomes

Author:
Alexandra Neagu

Supervisor:
Dr. Nicole Salomons

Second Marker:
Dr. Antoine Cully

June 17, 2024

Abstract

Large Language Models (LLMs) have demonstrated remarkable capabilities in information sharing and fact-checking, distinguishing themselves from traditional search tools. By adapting conversational interfaces, LLMs have gained popularity as versatile personal assistants, capable of answering a wide range of user inquiries. However, their interactions primarily focus on delivering most probable answers based on their training data, and they do not distinguish between users differences in knowledge or preferences. Therefore, by integrating LLMs into systems that comprehend information regarding users' skills, goals and thought-process during a specific task, they could transform into the personalised AI assistants that current LLM products are marketed as.

This study explored how such AI assistants can be applied in educational settings to improve learning outcomes. It used a simple control environment (Nonogram puzzles) to investigate how to tailor an LLM (Meta's Llama 3 8b Instruct) through prompt engineering, without requiring further training, in order to provide personalised help based on the user's in-game progress and thought-process when solving the puzzles. Experiments were conducted with 20 novice Nonogram players to assess the improved learning outcomes brought by the tailored AI Assistant, including faster solving times and reduced uncertainty while completing the puzzles. The between-group designed experiment evaluated the impacts of a tailored hints versus randomised untailored hints onto the participants' learning process of Nonograms. The study demonstrated that participants preferred the tailored hints, finding them more useful.

Acknowledgements

I would like to thank my supervisor Dr. Nicole Salomons for her support throughout the development of the system and for her insightful feedback on designing the participant experiment.

Additionally, I would also like to thank all participants of my study for their time.

Contents

Chapter 1

Introduction

User interaction is at the heart of every product. Therefore, any new technological development should take into consideration how the user will interact with the system and how the system should be tailored to the user's skills and needs. With the recent advancements in artificial intelligence (AI), particularly the newly proven capabilities of Large Language Models (LLMs), it has become essential to adapt these models' advantageous functionalities to the users.

By considering such adaptations, LLMs can be tailored to each user, thus providing accurate information while aligning with the user's thought-process. For instance, imagine an LLM functioning as a personal tutor for a student learning mathematics. The model would not only provide correct answers but also understand the student's approach to problem-solving, identify areas where the student struggles, and offer targeted guidance. This level of customisation can transform an LLM from a generic searching and fact-checking tool into a customised assistant that supports individual learning paths.

Building on these ideas and opportunities, this study aims to explore how LLMs can be used to provide tailored hints and feedback based on the student's skill level on a given task. The task defined for this study is solving Nonogram puzzles. The system developed tracks the user's progress and returns tailored recommendations for next moves and strategies that aid the player complete the puzzles. This data is formulated into a knowledge base, which the LLM (Meta's Llama 3 8b Instruct [1]) uses to generate constructive hints that assist the user throughout the game.

1.1 Motivation

LLMs' effectiveness in knowledge sharing is already well-established. They respond to user prompts by generating answers based on their extensive knowledge base. However, despite their wide capabilities, LLMs also encounter many limitations such as hallucinations, where responses may contain erroneous information if the prompt references knowledge points not included in their training data. These limitations are particularly evident in reasoning tasks, such as solving mathematical problems or describing spatial relations between two objects on an unseen grid. Therefore, in application fields that require reasoning to aid the user in a task, general pre-trained LLMs are reasonably limited.

One such application is the use of LLM assistants to help guide students in their learning process. Tutoring is a critical component of education because it offers personalised attention and tailored instructions to the students' skills and goals. However, for an AI assistant to take on the task of tutoring, it must be aware of the students skill level, goals and learning styles. Currently, widely used AI assistants, such as ChatGPT [2] do not consider these personalised aspects. So, while they are able to convey information on various topics, they are not yet effective tutors. To fulfil the role of a tutor, LLMs would need to engage with students in a meaningful way, encouraging a deeper understanding and retention of knowledge. This could be done if the LLM adapts its responses to the student's skill level, and notices specific areas of difficulty, making learning more efficient and effective. Additionally, to ensure the quality and accuracy of LLM's responses, there must be a precise knowledge base for the LLM to rely on for comprehending the task it assists with.

This study aims to demonstrate how such AI assistants can be applied in educational contexts to improve learning outcomes. It uses a simple control environment (Nonogram puzzles) to develop ideas on how to customise a language model through prompt engineering, without requiring further training, to provide personalised help based on the user's progress within the puzzles, their localised focus relative to the puzzle grid, and their thought-process. Additionally, to ensure the quality and accuracy of the hints, the study aims to develop a precise knowledge base for the LLM to reference. Thus, the following hypotheses are the main focus of the study:

Hypothesis 1: *By utilising a Large Language Model to create hints and feedback customised to users' progress and spatial reasoning in a 2D grid puzzle game, we expect to see* ***faster completion times*** *compared to conventional learning methods without personalised assistance.*

Hypothesis 2: *By utilising a Large Language Model to create hints and feedback customised to users' progress and spatial reasoning in a 2D grid puzzle game, we expect to see* ***higher solving accuracy****, compared to conventional learning methods without personalised assistance.*

Hypothesis 3: *When learning how to solve a new puzzle game, people* ***prefer tailored hints*** *based on their progress.*

1.2 Objectives

By using an interactive Nonogram puzzle game (described in section 4.1), the study aims to compare the effectiveness of AI-generated hints when the system tracks the user's progress versus when it does not. Specifically, a between-group experiment is designed to investigate how the user's learning efficiency and accuracy differs with various approaches of AI assisted help (chapter 5). The customisation of these hints can range from generic Nonogram tips to tailored hints based on an analysis of the user's progress over time while completing the puzzle. Tailored hints can provide directional pointers to areas of the puzzle where the user's next best steps are located. Additionally, they can also offer conclusive pointers on specific grid cells to reconsider if the user is significantly struggling. This involves translating complex, hard-to-read game data into understandable instructions for the LLM, enabling it to produce informative and constructive hints. This methodology is detailed in section 4.2. Furthermore, the assisting system must also consider accessibility features, providing hints in both audio and text formats to accommodate various students' learning preferences.

The developed system will then be tested in a user trial involving various participants. This experiment will compare two control conditions: one group will experience a fully tailored AI assisting system, while the other group will interact with an assistant that is not aware of their in-game progress and understanding of the puzzle. The data gathered from this between-group experiment will be used to determine whether tailoring an AI assistant to the student's specific task and progress positively impacts their learning outcomes and user-experience compared to a non-tailored counterpart.

1.3 Ethical Considerations

There are a couple of ethical concerns regarding the project that have been considered. One concern is that the study includes collecting in-game participant data and participant questionnaire responses. The data collection procedure was designed to be safe and secure, storing the data in a private database, only accessible to the people in the research group. The data only served the purpose of this specific project, and any detail of the participants, including any personal data, game progress data, or survey responses is not be shared outside the scope of the presented work. An ethics checklist was completed, allowing this study to be clear about its intentions. The volunteers were explained these procedures before participating in the experiment and signed a consent form (found in Appendix F), permitting the study to use their data for further analysis.

Another point concerns the participants' involvement in the study. The research participants were volunteers (aged over 18) willing to participate in the study and could withdraw at any point during or after the experiment phase. To accept participants, the approval of the Ethical Team in the Department and of the Research Governance and Integrity Team was received to ensure the study proceeded with the correct steps.

Chapter 2

Background

This chapter introduces the background knowledge, fundamental ideas and best practices that this study is based on. The sections will also cover both technical fundamentals and implementations of Language Model technology.

2.1 Natural Language Processing & Large Language Models

Natural Language Processing (NLP) is a branch of artificial intelligence (AI) that concerns about allowing computers the ability to understand human language in various formats, such as spoken or written. A large area of NLP, that became a focus point in recent years, has been the development of Large Language Models (LLMs).

LLMs are Deep Learning models that are trained on vast amounts of data. Such models have shown immense capabilities in various NLP tasks, such as question answering, document summarization, machine translation, information retrieval, sentiment analysis, etc. Such capabilities are also not constrained by text alone, as current research in multi-modality allowed LLMs to also understand and generate images, videos, or voice recordings.

2.1.1 Behind LLMs: Transformers

Transformers are the main component that allowed LLMs to develop such multitude of abilities, as their development proved to boost the speed with which models can be trained. Hence, it allowed LLMs to scale more effectively with increasing amounts of data compared to traditional architectures (such as Recurrent Neural Networks and Convolutional Neural Networks).

Transformers are an architecture designed around the idea of attention and parallelisation [3], which processes the input sequence as a whole. Following the overview described by Alammar et al. [4], this can be done by having the input sequence be parsed into non-decomposing units called *tokens*. Such tokens can be formed by single words, part of words, or characters. Afterwards, tokens are processed into *embeddings*, which are the vector representations of these tokens capturing semantic information and contextual relationships in the language data.

The novelty of "attention" is that transformers allow for each word to develop a self-perspective of every other word's relevance to itself. It is done by having each word look at other positions in the input sequence (other words) and gather some information of relevance that will help to develop a better encoding for itself. Such information can point to the frequency of how often a word is followed by another (such as conjugating verbs and groups of positive adjectives), what noun is a pronoun referring to, or any other language characteristics. Hence, the idea of self-attention refers to the "egocentric focus of each token in a corpus" [5].

Additionally, transformers have been successful in transfer learning, where models pre-trained on large datasets can be fine-tuned for specific downstream tasks with smaller datasets, further leveraging the benefits of large-scale training. This is further detailed in subsection 2.1.3.

2.1.2 Data Processing and Datasets

In order to develop any AI model, a major challenge is the collection and preparation of data. Before being used for training in the shape of a dataset, the documents and data collected must be pre-processed through filtering, anonymising, and debiasing. This step is required to make sure the data is adequate for the task at hand. Such is the reason why LLMs are "large". A very vast amount of data must be gathered to develop a more comprehensive knowledge base. Such a database must show both breath and depth for the task at hand in order for the LLM to be capable of generating fluent, coherent, and relevant information.

Following best practices, the next steps in developing a dataset are to format the pre-processed data into the expected input format of the model for the decided NLP task. Table 2.1 below presents a couple of NLP tasks for LLMs and their respective dataset formats based on highly used datasets and benchmark datasets currently on the market (2024).

Question Answering (from SQuAD dataset [6])	`{`
	`  "question": "What is the capital of Japan?",`
	`  "context": "Japan, a country located in East Asia, has Tokyo as its capital.",`
	`  "answer": "Tokyo"`
	`}`

Text Generation and Summarization (from OpenOrca dataset [7])	`{`
	`  "system_prompt": "You are a helpful assistant, who always provides explanation. Think like you are answering to a five year old.",`
	`  "question": "What happens next in this paragraph? ...",`
	`  "response": "She dips the needle in ink ..."`
	`}`

Table 2.1: NLP Tasks and Expected Dataset Format

2.1.3 Tuning Language Models

There are various methods of tuning LLMs that specialises a model onto a specific knowledge base. There are supervised and unsupervised learning procedures that are used for tuning LLMs, for example fine-tuning and RAG respectively. Such methods require task specific datasets that would allow the LLMs to adapt themselves to similar prompts as seen in the datasets. This section is based on the comprehensive overview from Naveed et al. [8] and presents a concise summary of some points in the paper.

Fine tuning covers a range of processes for improving the performance of LLMs. For example, Transfer Learning [9] is one of the processes where a pre-trained model is further trained on a task-specific dataset to adapt it for that particular task or domain. Another example is Instruction-tuning [10], during which a pre-trained model is fine-tuned on instruction-shaped data to improve the model's response to user queries. Such processes are applied because LLMs require very large datasets to train effectively, however, real world data gathering is affected by data scarcity. Hence, pre-training a model generally then fine-tuning it to the domain-specific data allows the model to become highly more accurate.

Retrieval-augmented generation (RAG), first introduced by Lewis et al. [11], is a training and fine-tuning technique that relies on a retriever model to fetch relevant information from the documents forming the LLM knowledge base, and use them as context to the LLM to answer queries [8]. It succeeds in enhancing the accuracy and reliability of generative AI models by providing access to defined external sources (such as frequently changing documentations or domain-specific company documents). As it only replies on a constraint given knowledge base, RAG tuned models have a lower chance to 'hallucinate' incorrect or misleading information (more details in Section 2.1.6).

2.1.4 Prompt Engineering

Prompt Engineering is a topic of current active research which explores how user prompts can influence the model's responses and lower the chances of models hallucinating, thus positively impacting the model's performance [12]. Prompt Engineering involves writing effective inputs

(prompts) to obtain desired responses from language models. This involves techniques for wording prompts, structuring inputs(user or system prompts), and iterating on prompt designs to improve the model's performance on specific tasks. In contrast to fine-tuning and RAG, it is a cost-effective approach to optimising an LLM for specific tasks, as it does not require additional tuning and uses the published base model. However, the capabilities of the base model for inferencing are often limited compared to those of a fine-tuned model, which has been specifically adapted for a particular task.

Article [13] mentions sentences such as "This is very important to my career" [14] or "Let's think step by step" [15] that can greatly improve the quality of the LLMs response in attention to quality and logicality respectively. Therefore, a novel best practice has been introduced in the development of LLMs: system prompts. System prompts define a set of rules and initial instructions for the LLM to keep into consideration when generating answers during the user interaction. For example, article [16] presents the system prompt of ChatGPT [2](based on OpenAI's GPT-4 architecture). The prompt covers various instructions such as instructions for generating efficient python code and for ethical considerations when generating images using OpenAI's DALL·E system.

2.1.5 Language models for specific NLP tasks

Question Answering Models

Question Answering (QA) is an NLP task in which a model retrieves information from texts and documents in order to answer user questions. Common QA systems "allow a user to ask a question in everyday language and receive an answer quickly and succinctly, with sufficient context to validate the answer" [17]. Therefore, for LLMs to be accurate answer generators, current research focuses on the reliability and efficiency of their answer retrieval and generation [18] [19].

QA can also be used alongside multi-modality, by allowing models to base their answers on multiple types of inputs and knowledge bases, including text, images, speech, video, etc.

Conversational Models

Conversational language models facilitate meaningful and contextually relevant interactions with users. These models aim to comprehend user input, infer context, and generate appropriate responses that mimic human conversation. They find industry applications in chat-bots, virtual assistants, and customer service bots, such as OpenAI's ChatGPT [2], Amazon's assistant Q [20], and GitHub's Copilot Chat [21].

2.1.6 Current LLM Challenges

Hallucination

Hallucinations are one of the main challenge of LLMs. LLMs are considered to hallucinate when they generate erroneous information due to their knowledge base not being sufficient to generate an accurate answer. LLMs choose the next word by maximising the relevance with the previously generated tokens, thus building a sentence with the highest relevance between the tokens based on the training data provided. However, in the case of hallucinations, the LLM leads to generating sentences which are correct grammatically, but incorrect logically. This is the reason why evaluating LLMs' performance with technical and human metrics, such as model accuracy and subject relevance respectively, is very important.

Ji et al. [22] classifies hallucinations into two categories: "intrinsic", which refers to generated output conflicting with the source content, and "extrinsic", indicating generated output that cannot be verified from the source content. The nature of the source content varies based on the specific NLP task at hand. In the context of dialog generation, it would be the dialogue history or external knowledge sentences. Similarly, in question answering tasks, the source content may involve the retrieved documents and their selected content. Such distinction allows for an easier comprehension of the vast LLM characteristics that need to be evaluated (not only correctness and relevance, but also faithfulness and coherence, etc) to decide the performance of a model.

Spatial Reasoning Capabilities

Spatial reasoning capabilities of LLMs have been a significant area of research, especially in comparison to human spatial reasoning. Humans have an intuitive ability to perceive and describe spatial locations on an unseen grid using linguistic and visual representations, often using terms like "above," "below," "left," and "right" to refer to positions and changes in position [23]. Therefore, evaluations of LLM's spatial orientation throughout NLP tasks are often compared to such human abilities. However, such evaluation has revealed several challenges and limitations, including high rates of hallucinations in spatial relationships between objects in unseen grids.

To address these challenges, the SpartQA question-answering benchmark [24] was developed to evaluate LLM's spatial capabilities in tasks such as Choose-objects, Describe-objects, Find-all-relations, and Find-similar-objects within visual scenes. This widely used benchmark allows for evaluation of "the logical consistency, correctness, and the number of steps required"[24] for LLMs to answer a spatial related question.

Recent advancements have also adapted related NLP ideologies into spatial reasoning tasks, such as the idea of Chain-of-Thought (CoT) [25] as a multi-hop reasoning. Li et al. [26] evaluates GPT-4's multi-hop spatial reasoning proficiency using spatial stories and questions based on the StepGame [27] and SpartQA [24] benchmarks. The evaluation included eight spatial relations (top, down, left, right, top-left, top-right, down-left, and down-right) in the generation of spatial stories, with which LLMs are assessed on inter-object relations, such as relative locations. This has demonstrated the effectiveness of multi-hop spatial reasoning as CoT allows LLMs to divide their thought process into smaller tasks, basing their final reasoning on these intermediary steps.

Additionally, the novel Visualization-of-Thought (VoT) [28] approach introduces the idea of LLM's creating mental images during the spatial reasoning process by visualising their thoughts at each intermediate step in a Chain-of-Thought process. This multi-hop approach is described to generate mental images along the solving process of spatial reasoning tasks such as: natural language navigation (path planning), visual navigation, and visual tiling in 2D grid worlds. VoT's experimental results have been shown to improve performance across LLMs of various sizes, sometimes outperforming existing multimodal large language models (MLLMs) in these tasks.

Another approach of improving spatial reasoning capabilities is illustrated by Sharma et al. [29]. Spatial Prefix-Prompting is introduced as a process of prompt engineering that "instigates the model to first ponder a tangentially related spatial problem, and then use the 'knowledge gained' to answer an adjacent question"[29]. This study hypothesises the potential that knowledge transfer from simpler spatial tasks to more complex ones can improve overall LLM performance in spatial reasoning tasks.

2.2 Application Architecture

2.2.1 Unity

Unity is a cross-platform game engine that supports a variety of desktop, mobile, console, augmented reality, and virtual reality platforms. [30] With Unity, developers can create games in both 2D and 3D by using the provided scripting API in C# or the integrated interface for game design and functionality development. For this study, Unity was used to develop a game interface around the idea of solving Nonogram puzzles with the help of an AI Assistant.

2.2.2 Communication protocols

To provide reliable functionalities to an application based on a server-client system architecture, there is the need for a highly reliable communication protocol, such as HTTP. As presented on [31], Hypertext Transfer Protocol (HTTP) is an application-layer protocol widely adopted for web-based interactions between clients and servers. HTTP's key feature is the request-response model, within which a client opens a connection to make a request, then waits until it receives a response from the server. Additionally, HTTP is a stateless protocol, thus, no data (state) is stored between two requests in the server.

2.2.3 User-Centric AI Product Development

Any new technological development should take into consideration how the user will interact with the system and how the system should be tailored to the user's intentions and needs, thus encompassing the overall user experience (UX). Therefore, with the recent advancements of Large Language Models, it is required to adapt their focus to the users. By considering such adaptation, it is possible to tailor LLMs to each user, thus aiding in providing users with accurate information while grasping users' thought-process, intentions and understanding of the problem. This approach would allow for LLMs to be implemented in developing projects with more societal impact [32], such as personalised interactive assistants, instructors, or tutors. Therefore, by following the basics of co-creating innovation [33], such approach can enrich vital features of innovation, such as adaptability of LLM products towards their audience.

In order to understand user needs, there are various approaches for data collection, including user research through interviews and surveys, or analysing user interaction data to identify trends. Such data can facilitate a deeper understanding of the user experience, showcasing how users utilise the product's capabilities and providing insights in how much they trust the system and its outputs. For instance, Wang et al. [34] presents a conceptual framework for building human-centred explainable AI (XAI) [35], designed for use in human decision-making. The proposed methodologies in the study draw inspiration from philosophy and psychology, focusing on understanding patterns of human reasoning, decision-making processes, and cognitive factors influencing decision-making. Their framework underscores the importance of identifying how XAI can allow users to understand and trust the outputs and methodologies of "black box" AI models. This understanding, in turn, will support human reasoning and decision-making and also help mitigate errors or detect hallucinations in the AI generated answers.

XAI refers to the interpretability, transparency, accountability of AI models, aiming to unwrap the "black box" algorithms they are built on. Barredo Arrieta et al. [35] provides a comprehensive overview of AI models based on their explainability levels, out of which Deep Neural Networks, such as Convolutional Neural Networks and Recurrent Neural Networks, are categorised as non-transparent. Transparency implies that models can be simulated and have its algorithms and procedures decomposed for a human to understand. This could be done through graphical illustrations or human-readable rules. However, for non-transparent models, techniques such as post-hoc explainability aim to communicate how a developed model produces its predictions for any given input, thus encouraging human trust in outputs of such models. In this line of reasoning, Language Models would also need to be categorised as non-transparent, thus becoming crucial to develop XAI explainability techniques that unwrap their behaviours, limitations, and social impacts [36].

2.3 AI for Tutoring

2.3.1 GenAI for Feedback Generation and Knowledge Assessment

Providing information in the shape of feedback is a particular use of generative AI (GenAI). Previous research has been done in the field of Computing Education, using LLM-based tools and evaluating LLMs in computing classrooms [37]. Such tools can be used for sustaining peer-to-peer discussions on the educational material or code, exercises, and feedback generation.

By focusing on students who utilise LLMs for discussing and verifying their knowledge and thought process, such applications require the incorporation of fundamental feedback mechanisms in the form of dialogic conversations. This type of feedback is defined by Nicol et al. [38] as "a two-way process that involves coordinated teacher-student and peer-to-peer interaction as well as active learner engagement." Consequently, the introduction of LLMs in such context brings attention to adopting human-human interaction approaches within the fundamentals of knowledge sharing and teaching in human-computer interaction.

This study focuses on the feedback directed towards how well a task is being accomplished or performed, and how LLMs' generated task-oriented hints impact the users learning process. Following the framework of the Four Levels of Feedback defined by Hattie et al. [39], task-oriented feedback is a direct outcome of long- and short-term performance. Therefore, LLMs would have to tailor their hint generation on the available data on the user's performance, intentions, and prompts.

2.3.2 Existent Products

As of the present moment (June 2024), there exists various tools incorporating AI technologies to enhance tutoring methodologies. Below are a few examples that prominently focus on AI chatbot features. The descriptions below follow the official marketing publications, however due to their subscription based usage, they were not able to be tested and reviewed for critical commentary in this study.

Khanmigo

Khanmigo [40], introduced in March 2023, is an educational chatbot created by Khan Academy, a non-profit organisation dedicated to providing free online education. It is marketed as being able to play the role of a tutor, offering guidance on problem-solving, providing hints, collaboratively solving mathematical problems, and engaging in one-on-one debates on various topics. It is also promoted as an assistant for teachers assisting in lesson plan development and monitoring student activity and progress. Although, not much is publicly published regarding the technologies, procedures, and evaluation methods used for developing Khanmigo, it is known that Khanmigo is powered by GPT-4 and has access to all the learning materials already existent on the Khan Academy platform. As of December 2023, it was only available for purchase by residents of the United States over 18. Thus, a pilot testing phase started in the United States to gather real-world data of people's opinions and their use patterns [41].

AI Tutor

AI Tutor [42], developed in July 2023 by TutorOcean, is a learning tool that creates personalised support for students based on their level of education (from elementary to higher education). Aside from similar chatbot functionalities as Khanmigo, AI Tutor also provides personalised practice test and quizzes that help students test their knowledge of a subject. All in all, AI Tutor's features are closely similar to Khanmigo.

2.3.3 Educational Institutions' Policies on Generative AI

With the vast capabilities in providing people with information, LLMs have been introduced in many working sectors, with some sectors, such as education, being affected both positively and negatively. Whereas, LLMs have been used for improving people's task completion efficiency [43], problem-solving [44], and decision-making skills [45], their generative capabilities also introduced challenges in the procedures of plagiarism in students' work [46].

Therefore, many educational institutions have already taken action in covering such scenarios and defining new approaches of perceiving AI generated work. For instance, the Russell Group has published a policy [47] covering five 'principles' that propose an overall approach of UK research intensive Universities on generative AI tools. Those principles set an overview from the importance of AI-literacy to the preservation of academic integrity. However, each educational institution requires to detail out their own proposed policies. Imperial College London, for example, has established The Working Group on Artificial Intelligence Tools in Teaching and Assessment to explore the development, opportunities and implications of generative AI tools. Some points they are looking into include the impact of "allowing students to use generative AI to answer specific/complex questions" and an assessment of "the types of prompts students use to "chat" with the generative AI" in order to receive relevant and accurate information aids their studies [48]. Similarly, our study focuses on identifying the prompts that provide the most assistance to the participants learning a new task.

Chapter 3

System Design

This section will present the system design of the game interface and the AI Assistant server. Section 3.1 describes the deliverables and the expected system capabilities. Section 3.2 describes the high level system architecture and data flow within the system and mentions the subsections that will go in details about each feature.

3.1 Deliverables and Capabilities

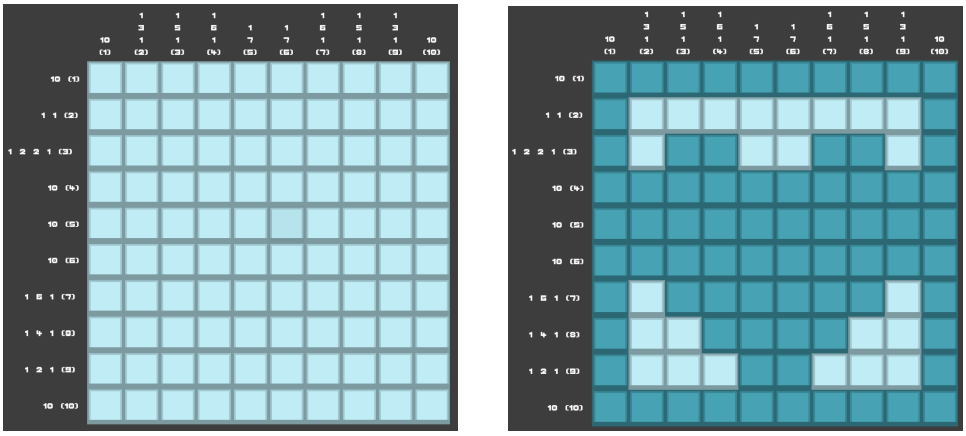
This project has been built on two main components: the Unity-based puzzle game and the AI Assistant backend server. The puzzle game provides an intuitive interface for users to complete several Nonogram puzzles. The backend server has the role of prompting the Large Language Model to generate hints and feedback for the users but also manipulate and store the user in-game data such as puzzle completion progress and UI interactions.

The goal of the system is to assist users solve a Nonogram more efficiently and think outside the box while guessing the meaning of the hidden image within the puzzle grid. Therefore, to successfully pass a level, users must both correctly fill in the Nonogram puzzle and also guess the meaning of the revealed image within a set time limit. The game features multiple levels with the same goals and time constraints. All in all, being developed with a participant study in mind, the system aims to provide insights into how user-tailored AI-assisted learning can improve puzzle-solving performance.

3.1.1 Nonograms

Nonograms are logic-based crossword puzzles, similar to Sudoku, where the player fills in a grid based on numerical clues to create a hidden image. The numerical clues indicate the length of each group of filled cells on that line, with each group separated by at least one empty cell. The Nonograms used in this study only have one unique solution. Thus, they can be solved by an experienced player or an automatic solver algorithm without any uncertain moves, as there is at least one definite path to the solution. In other words, each step in solving the puzzles can be definitively determined. Solving these puzzles requires deduction, pattern recognition, and strategic thinking skills, which are honed over time through experience and understanding of the game's rules and best strategies.

Nonograms were chosen as the control environment of the study because they are a classic example of constraint satisfaction problems, known to be NP-complete—meaning that, despite their complexity, there is no optimal algorithm for solving them in polynomial time. However, they are governed by a simple set of rules, making them accessible to human solvers who can develop effective, though not necessarily optimal, strategies. Thus, this study proposes the use of an AI assistance model to guide players and teach them Nonogram solving strategies by tailoring hints and feedback according to their current progress, the specific spatial area they are focusing on, and the game's set of rules.



(a) Nonogram initial blank state (b) Nonogram correctly solved

Figure 3.1: A sample Nonogram: the objective is to reveal a hidden image by filling in grid cells according to numerical clues provided along each row and column. The numerical clues indicate the length of each group of filled cells on that line. Each group is separated by at least one empty cell. A valid Nonogram only has one unique solution. Images taken from the study's game.

3.2 System Architecture

The high-level architecture of the system describes the overall flow of data from user interactions with the game interface to the server and back to the user. Diagram 3.2 illustrates the data types collected from the game and sent to the server for either storage or use in generating hints, which are then forwarded back to the user. The architecture can be divided in the following parts:

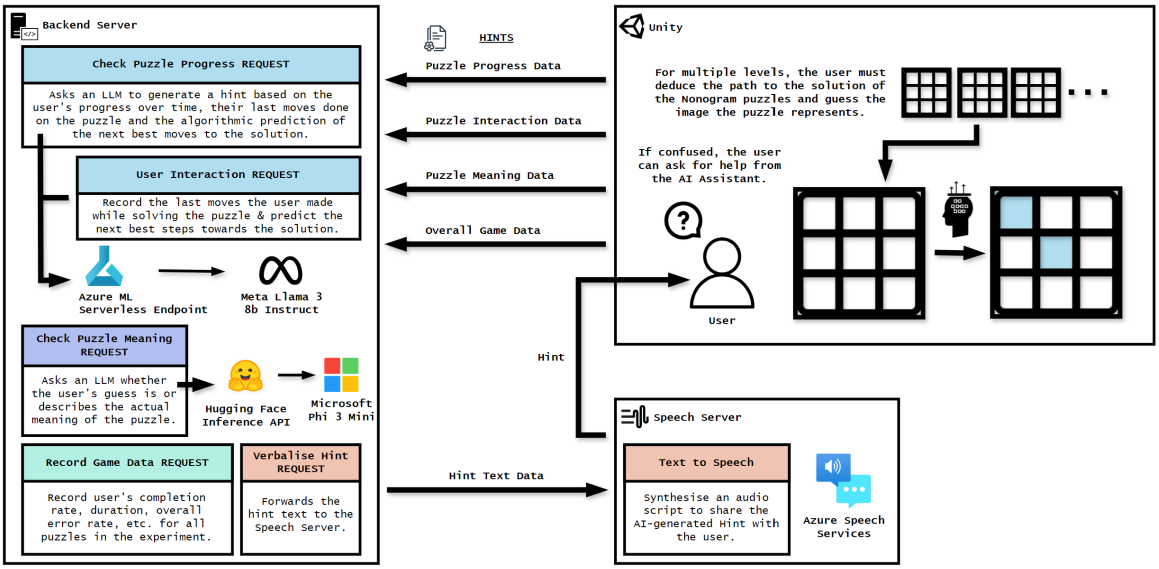


Figure 3.2: Data Flow between the Game and Server

The Game:

- On the right side of the diagram, the user aims to complete a level by both correctly filling in the Nonogram puzzle and guessing the meaning of the revealed image. Interactions, such as filling a grid cell or clicking one of the check buttons, are managed by Unity and forwarded over HTTP to the server. Details on the structure of the game UI and its capabilities are in section 4.1.
- Data gathered from the game includes *Puzzle Progress Data*, *Puzzle Interaction Data*, *Puzzle Meaning Data*, and *Overall Game Data*. Each data is detailed in section 4.1.3.

The Back-end Server:

- Each major feature of the server is accessed through distinct HTTP Requests that the game calls upon at specific times.
- One primary function of the server is to use the gathered in-game data and transform it into a knowledge base suitable for an LLM Assistant to understand and use for hints and feedback generation. This involves three main requests:
  - *User Interaction Request:* This request records the player's last move on the puzzle grid. It also calls a solver function to predict the best next steps for the player to take towards the solution. This data is then saved for further analysis of the user's progress, as explained in section 4.2.3.
  - *Puzzle Progress Request:* This request uses the user's in-game progress data and interaction data to generate a hint helpful for completing the puzzle. This function utilises Azure ML Services for prompting Meta's Llama 3 8b Instruct model through LLM Serverless API Endpoints. Details of this setup and its hint generation capabilities are described in sections 4.2.4 to 4.2.10.
  - *Verbalise Hint Request:* The generated hint is forwarded to the *Speech Server*, which synthesises an audio script from the hint text data using Azure AI Speech Services. More details on the setup is found in section 4.2.5.
- *Check Puzzle Meaning Request:* This request checks the puzzle meaning guess made by the user. This check verifies whether the user's guess correctly describes the solution meaning of the puzzle. This functions calls upon Hugging Face Inference API for prompting Microsoft's Phi 3 model through LLM Endpoints. More details are in section 4.2.2.
- *Record Game Data Request:* The server also records all user in-game data, including information about the user's completion rate, duration, and error rate for all levels throughout the game. This data is used for analysis succeeding the participants experiment presented in chapter 5.

Chapter 4

Technical Implementations

This section describes the details regarding the technical development of the two main components of the system: the puzzle game interface (section 4.1) and the AI Assistant server (section 4.2).

4.1 Puzzle Game Interface

The puzzle game is the only interface the users interact with throughout the participant trials. The game was developed in Unity, using its respective object-oriented scripting language C#. The goal of the game is for the experiment participants to complete several Nonogram puzzles and, when stuck or confused, ask for hints about what do to next. Hints differ depending on the progress the user has made and the level they are at, a more detailed overview can be found in section 4.2.8. Completing the levels implies filling in the Nonogram puzzle in order to fully respect the numerical clues and solution state, and also guessing the meaning of the revealed image correctly.

4.1.1 Game Scenes

The game is split in 5 main Scenes: *Main Menu*, *Levels Menu*, *Level Screen*, *Tutorial Screen*, and *Create Level Screen*. A game scene is a distinct screen or visual portion of the game. Each scene and its main roles are described below. Distinct visualisations of each scene can be found in Appendix A.

Main Menu: On this scene, the user can log in with their username by using an input box. The user has two main buttons on this scene: <Start the Game> and <Quit Game>. The start button will forward the user to the Levels Menu scene.

Levels Menu: On this scene, the user can select the level to complete. Level buttons are gradually made intractable as the user progresses through the game and completes each level. Each level is assigned to a specific Nonogram puzzle. More information on how the levels are loaded into the game is detailed in section 4.1.2.

Level Screen: On this scene, the main puzzle game is displayed, as presented in figure 4.1. On the left the Nonogram puzzle grid is displayed with its row and column clues. Below the grid, the user can see some more information about the level (such as puzzle grid size, time remaining to complete, and last clicked square in the grid). And on the right side of the screen, the user is also presented with the check buttons for the puzzle meaning and puzzle completion. The checks will either return an "affirmative" or "negative" animation if the verification passes or not. The user also has the <Hint> button and can request custom hints for that specific level.

Tutorial Screen: This scene is very similar to the Level Screen, however it contains an extra feature that provides users with a guide on the goal of the game and how to interact with the UI. This guide is accompanied by animations emphasising each of the UI components and their features.

Create Level Screen: This scene is only accessible by the developers as it allows for the creation of new levels in the game. Developers can set the puzzle grid size and meaning onto input boxes

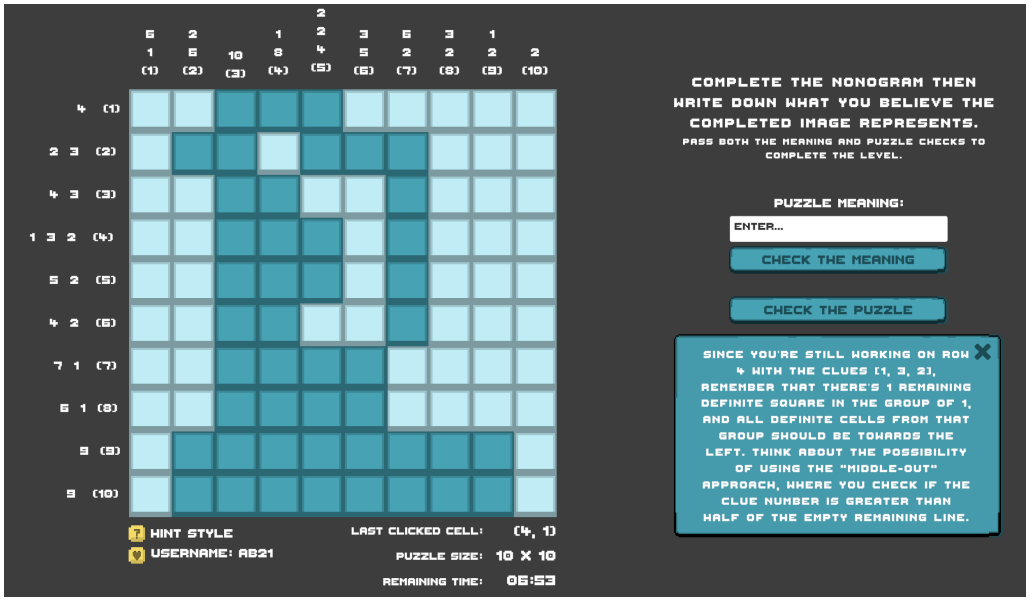


Figure 4.1: A partially completed Nonogram presented on the Level Screen of the Unity-based game. On the right side, a hint is presented to help the user continue completing the puzzle.

on this screen and directly set the solution state for the grid. Once the developer sets all the characteristics of the level, the level is saved in a JSON file, ready to be loaded as a new level in the game.

4.1.2 Game Logic

As the game is the only interface that the users view, the goal is to achieve a very user friendly and intuitive flow throughout the game. This section aims to present each feature and decision made to develop such an interface. Figure 4.1 shows an example level that is partially complete. The sections following refer to features contained by the *Level Screen*.

The Nonogram Levels

The Nonogram levels were developed with the idea of versatility, where any puzzle of any size can be integrated as a level in the game. A level can be easily created using the *Create Level Screen*, which provides the interface to manually shape the solution state of the puzzle grid and set its specific attributes. *Create Level Screen* allows for the newly created puzzle level data to be stored in a game file system as a JSON. Such a file will be referred to as the 'default level file' and it contains main attributes such as

```
{ puzzle size, solution grid state, and puzzle meaning solution }
```

The *puzzle size* indicates the number of rows and columns the Nonogram contains. The *grid state* represents the whole Nonogram grid with its cell states. It is formed of a 2D Map (List of Lists of Boolean variables), where the first element of the grid state signifies the first row and the first element of that row signifies a specific cell at row 1 and column 1. Each cell state is described by a boolean variable, where *True* means a filled cell and *False* means an empty or crossed cell. More information regarding the cell state and their characteristics is found in the following section 4.1.2. Additionally, a full view of a default level JSON can be found in Appendix B.

The level's deployment is based on loading and storing functions attached to the *Level Screen*. These functions use a custom defined class structure *GridData* to understand and manipulate the level specific JSON files in C#. This class contains all attributes described by the JSON file as below.

```
{ grid size, username, level, time spent until now on the level, user progress grid state,
  solution grid state, puzzle meaning solution, Nonogram puzzle and puzzle meaning
  completion flags }
```

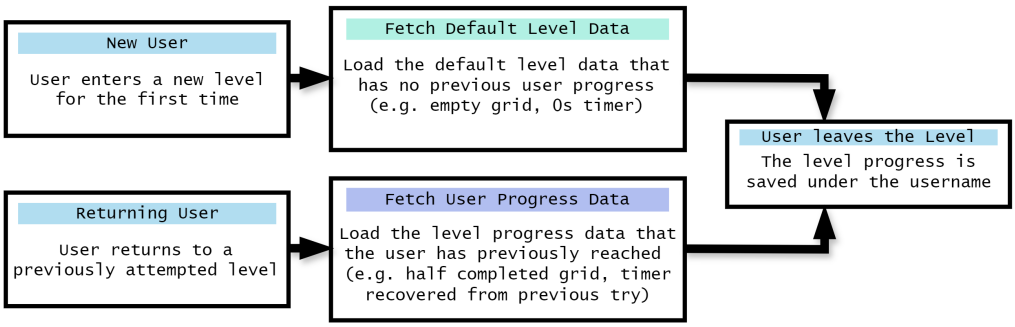


Figure 4.2: Process of loading a level file into game and storing the user progress into a file

Figure 4.2 describes the flow of how the level state data is maintained in the files. This is followed by further explanations below.

When a **new user starts a new level**, the puzzle will be loaded from the default level file, which contains all default values describing the level. For instance, the *username* is empty, *time spent* is 0.0 seconds, and the *grid state* and the *flags* are all set to *False*. However those variables will be updated by the user's interactions with the *Level Screen*. The rest of the attributes are kept as defined in the default level file (generated by the *Create Level Screen*).

Every time the **user leaves the *Level Screen*** and returns to the *Levels Menu*, the level state variables will be updated and stored under the user progress' folder in its respective file, following a naming scheme of *{username}_progress_level_{level}*. This folder contains all users' progress for each of the levels in the game. The progress is stored so that once a user returns to a previously attempted level, they will continue where they left off.

Moreover, during the experiment's procedure (described in chapter 5), the game limits users to one level at a time. Hence, users must either complete the current level or run out of time for the next level to become available. However, users can still leave a level at any point and have their progress saved to return later. The main limitation is that the user can only edit one level at a time. So, if they leave a level, they must return to the same level until the timer conditions are met (mentioned in section 4.1.2).

Interactions with the Nonogram Grid

The Grid is built out of n rows by m columns. Each line is formed of square cells. Each cell is an interactive button with 3 states: *empty*, *filled*, and *crossed*. Respecting the basics of Nonogram rules, the goal of the game is for the user to follow the numerical row and column clues to find all correct *filled* and *empty* cells that form a hidden image. Therefore, the *crossed* cells are only a visual aid introduced to the game UI for the user to mark the squares that they consider impossible to fill. But, to the system, the *crossed* cells are considered the same as the *empty* ones. Figure 4.3 shows the game interface presenting all cell states.

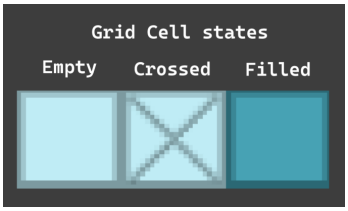


Figure 4.3: All states a grid cell can be under

A user can change the state of the cells in two main ways, by clicking on the square directly or by pressing the left mouse button and sliding over the cells (Click & Drag). The Click & Drag motion was implemented after the initial trials of the game's MVP. During those trials, it was noticed that experienced Nonogram players expected the slide motion, as they were used to other games on the market that have such a feature. Hence, following their recommendations and to

allow for most intuitive interactions, the Click & Drag was introduced. Figure 4.5 below shows the two interaction methods.

Figure 4.4: Main motions allowed to interact with the puzzle grid: Click and Click & Drag

Level Completion Requirements

For the user to complete the level shown on the *Level Screen*, they will have to first correctly fill in the Nonogram puzzle and then guess what the meaning of the revealed image is. Those two conditions can be checked by the player at any time throughout the level by clicking on the <Check the puzzle> and the <Check the meaning> buttons. The game will respond, through an animation, whether the condition checked has been passed. An affirmative response triggers an animation of a golden 'check' next to the specific button. Otherwise, a negative response triggers a 'cross' animation.

Puzzle Progress Check: At any point throughout the completion of the level, the user can check whether their current puzzle state is correct or not. By pressing the <Check the Puzzle> button on the right side of the *Level Screen*, the user will be notified through an animation if the grid matches the solution or it requires modifications.

The progress check is done locally by the game. By comparing every cell in the user's grid to the specific solution grid, the verification function will detect whether there are any inconsistencies between them. Thus finding the mistakes the user has made in their current grid state. If a mistake is found, the comparison is stopped and a negative return is provided, thus triggering the 'cross' animation.

Puzzle Meaning Check: Once a Nonogram is completed, the user can distinguish the revealed pixelated image. On the right side of the *Level Screen*, an input box is found where the users can type in their guess for the meaning of the puzzle and verify it by pressing the <Check the meaning> button. If the guess is correct, an animation will show the success, otherwise it will show the rejection of the guess, thus allowing the user to try again. Information on the function call processing the meaning check in the backend server is under section 4.2.2.

Level Timer

In order to limit the overall time duration of the game, the users have a constraint on how long they can spend on a single puzzle. Right under the Nonogram puzzle on the *Levels Screen*, a timer displays the minutes and seconds remaining to complete the level. This visual cue of the remaining time introduces time awareness and a sense of urgency for the participants throughout the experiment.

The timer can be stopped under two conditions. Firstly, when the user runs out of the time allocated for that level (8 minutes), the timer will stop counting down at 0.0. When this condition is reached, the user will be automatically sent back to the *Levels Menu* and will be given a warning notice that they have ran out of time and must move on to the next level. Secondly, the timer will stop when the user has successfully passed the two level completion checks: filled in the Nonogram and guessed the meaning of the revealed image correctly.

As all other level specific data, the time taken is also saved in the user progress JSON file. Hence, if at any point the user leaves the *Level Screen* with the level remaining uncompleted, then the duration time spend on that given level is remembered in the user's progress file. Thus, if the user returns to a previous level, the timer will continue counting down from the time remaining considering the variable loaded from the progress file.

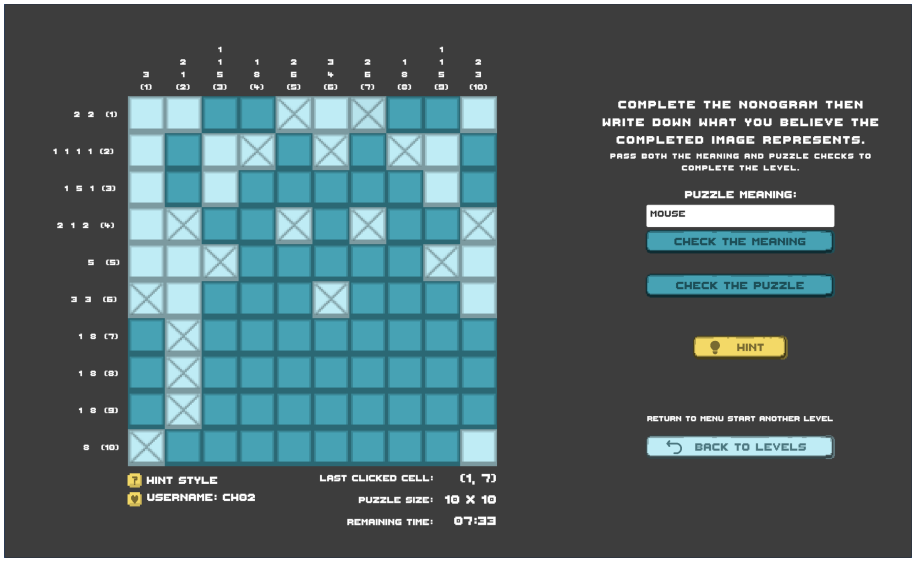


Figure 4.5: Level Screen example of the medium difficulty Mouse level. All components of this scene can be easily distinguished in this screenshot

User-Initiated Hint Request

At any point throughout the level, the player can press the <Hint> button on the right side of the *Level Screen* to request a progress tailored hint from the backend server. Further information regarding the types of hints that the player can receive and how they are developed is found in sections 4.2.8 and 4.2.9.

System-Initiated Hint Request Loop

Every 1.5 minutes the system takes the initiative and requests for a hint from the backend server. The loop starts whenever the *Level Screen* is opened regardless on the level loaded onto the screen. Such automatic request is the same as a user-initiated hint request mentioned above. This loop ensures that each user receives a minimum amount of assistance throughout the 8 minutes allocated per level. This feature was implemented after observing during MVP trials that participants often did not take the initiative to request help and spent excessive time trying to progress in the puzzle on their own. Therefore, to accommodate different user behaviours, the game supports both self-initiated and system-initiated hint requests.

Automatic Hint Time Buffer

To prevent hints from being requested too close together, a buffer of 20 seconds was introduced. After any hint is received (whether through a self-initiated or system-initiated request) a countdown begins to ensure that another automatic hint is not requested too soon. For instance, if a user might self-request a hint just a few moments before an automatic hint was scheduled, then with the buffer still counting down, the automatic hint is rejected and cannot be sent to the server. Without this buffer, the user might receive two consecutive hints with similar information, as the hints are based on nearly identical user progress. This mechanism avoids overwhelming the user with repetitive hints based and ensures a more spaced-out and manageable hint delivery system.

Hint Pop-up Display

The system aims to guide players and provide feedback on their puzzle progress. Therefore, hints generated by the backend server must be displayed intuitively and accessibly. Hints are returned in both audio and text formats to accommodate different user learning preferences and needs. The system firstly shares the hint verbally, allowing users to focus on the puzzle while listening. Afterwards, once the hint audio ends, the hint's transcript is displayed on the pop-up display in the bottom right corner of the *Level Screen*, ensuring users do not miss any crucial information that want to refer back to.

Regardless of how the hint request is triggered (whether by a user self-request or an automatic request from the server), the hint is first spoken aloud and then its transcript appears in the pop-up display.

Hidden Hint Style Toggle

As described in Chapter 5, the experiment that this game is used in will compare the learning impacts of the AI assistant on participants from two different control groups. One group will receive progress-tailored hints, while the other will not. Hence, in order to be able to easily change the game system between the two separate control groups, a discreet toggle is introduced in the game interface. This toggle allows easy switching between the two experimental conditions, by introducing different in-game data in the hint requests.

4.1.3 In-Game Data Gathering

This section describes the various data that is persistently shared with the backend server for storage and analysis. The request, data format, and purpose for each data type are detailed below.

All data transfer is done over HTTP requests, enabling a reliable client-server communication model where the client opens a connection with a request and awaits for the server's response. This protocol ensures reliable data exchanges between the game and server.

User Progress Data

The user's level progress data is gathered and sent to the server at every hint request, at both user- and system-initiated requests, and under both experimented control groups. The major distinction between the two control groups is the hint id, which will identity the hint style (tailored or untailored). The data forwarded contains:

```
{
    username, level, hint id, current grid state and puzzle grid solution, level
    meaning, completion state flags for puzzle and meaning
}
```

This information is used for calculating the user's progress and defining the class for the LLM generated hint. Further information regarding those steps is found under section 4.2.9.

User Interaction Data

Each interaction with the puzzle grid, such as clicking a cell, is recorded and sent to the server. The data forwarded contains:

```
{
    username, level, locations and states of the last 3 cells interacted with,
    current grid state and puzzle grid solution
}
```

This data is used to predict the next best steps for the user towards the completion of the puzzle. Further information as to how this data is used for analysing the prediction algorithm for the user's next best steps can be found in sections 4.2.4 - 4.2.10.

User Puzzle Meaning Guess

One of the checks the user has to pass in order to complete a level is to correctly guess the meaning of the image that the completed puzzle revealed. Thus the user inputs their guess in a text-box, which then forwards the text to the backend server for validation. The data sent contains:

```
{
    username, level, solution meaning, user's meaning guess
}
```

This data is then forwarded to the LLM server to check whether the user's guess is valid or describes something similar to the solution meaning. Further details on this process in the backend can be found in section 4.2.2

Overall Game Data

At the end of the game, when the user finishes all provided levels, the game UI will present a final notice confirming the end of the game and prompting the user to press the <End Game> button. This button gathers the overall user data from each level (such as completion flags and duration, etc) and forwards it to the server for storage and further analysis. The shared data contains:

```
for each level:
    {
        username, level, duration of completion in seconds, on-time completion
        flag, completion state flags for puzzle and meaning,
        number of hints used per level
    }
```

In the backend server, this data is appended with the user's calculated final progress per level. Further information on how this data is used in the final experiment analysis of the user's progress is explained in section 5.

4.2 Backend Server

This section details the features of the backend server for the developed system in this study. The backend server sets up a Flask app [49] with multiple routes allowing for the various requests called by the Unity-based game. Those requests include: *Puzzle Progress Request*, *Check Puzzle Meaning Request*, *User Interaction Request*, *Verbalise Hint Request*, *Record Game Data Request*. A short introduction to each request can be found back in the system architecture diagram in section 3.2.

Section 4.2.1 describes the procedures for storing all experiment-related in-game data into easily maintainable files for statistical analysis. Section 4.2.2 explains the system setup for the *Check Puzzle Meaning Request*; using an LLM to verify the users' guesses about the puzzle's hidden image meaning. Sections 4.2.3 and 4.2.4 present descriptions of the *User Interaction* and *Puzzle Progress Requests* and their roles. Section 4.2.7 presents the methodologies implemented to measure and track users' progress throughout the game levels. Section 4.2.8, 4.2.9, and 4.2.10 describe the hint hierarchy based on a progressive disclosure strategy, the use of an LLM to generate these hints, and the Nonogram solver algorithm used to develop the LLM's knowledge base. Section 4.2.11 discusses past challenges and lessons learned from previous approaches during the development of the AI assistant. Finally, section 4.2.5 details the setup for vocalising the hints, thus providing the user with both audio and text-based hints.

4.2.1 In-Game Data Storage

All in-game data is collected into multiple comma-separated value (CSV) files based on their purpose: *user progress data*, *meaning guess data*, *puzzle interaction data*, *overall game data*, and *audio data*. These data types are managed using a custom developed `CSVHandler` class. This class provides multiple functions to modify their respective CSV files, including appending new entries, updating existing entries, reading all entries, and retrieving specific entries based on an attribute.

Entries in the CSV files are initially created with empty attributes. Then throughout the pipeline of one request, the specific entry of interest is identified by filtering all entries by id and has its attribute updated.

Below is an enumeration of each attribute that the entries have depending on the CSV data file. The following sections will further explain each of the mentioned attribute and their role. Additionally, section 5 describes how the experiment-related in-game data is used for statistical analysis.

```
user progress data: { id, username,  level, hint class, hint response, next recommended
↪  steps, descriptive next steps, overall latency, hint generation latency, LLM model
↪  used, current and previous user progress percentage, hint session counter, timestamp }

meaning guess data: { id, username, level, solution meaning, user's meaning guess,
↪  approved flag, LLM model used, latency, timestamp }

puzzle interaction data: { id, username, level, last 3 cells the user interacted with and
↪  their state, current grid state, puzzle solution grid, user's selected next row and
↪  column, predicted next row and column, user's puzzle progress, timestamp }

overall game data: { id, username,  level, flags for puzzle and meaning completion, on
↪  time flag, duration spent on level, user's final progress on puzzle, number of hints
↪  used per level, timestamp }

audio data: { id, hint id, hint class, hint, audio generation latency, audio playback
↪  latency, timestamp }
```

4.2.2 Check Puzzle Meaning Request

For each level, the user will request to check their guess for the meaning of the revealed image. This request contains the user's guess and solution meaning. Those two variables are sent onto an LLM to verify whether they describe the same object or being. As this is not a very demanding task, a smaller and more efficient LLM is prompted: `Phi 3 mini 4k Instruct` model developed by Microsoft [50] and hosted on HuggingFace under a free-to-use Inference API [51]. The following section presents the prompt that is used:

"Under Answer section, write 'true' if the user guess is a synonym or describes something similar to the Solution; otherwise, return 'false'. Do not return an explanation. User guess: {guess}; Solution: {solution}; Answer: "

This prompt has proven reliable with all responses from the LLM to be accurate. However, to make sure the string response follows the appropriate expected format, an intermediate conversion step is done to remove all white-spaces in the response and make sure the string is all lower case. The response of 'true' or 'false', is then transformed into a boolean flag that is used by the game interface to trigger the corresponding animation for the user to see.

4.2.3 User Interaction Request

The User Interaction Request records the player's last three moves on the puzzle grid. Each move includes the location of a cell that the user interacted with and the state of the cell after the interaction, formatted as *(row, column, state)*.

In addition to recording player moves, this request calls upon the Nonogram solver to predict the best next steps for the player to take towards the solution. The data received from these interactions is stored as described in section 4.2.1 and contains empty attributes for the *user's selected next row and column* and *predicted next row and column*. Thus, after the prediction is made, the entry is updated with the respective next move prediction and the actual move made by the player.

This data is used to verify the accuracy of the prediction algorithm by checking if it predicts a relevant cell. Such cell should help towards completing the puzzle and must be in the area where the user was focusing at that point in time. Further details on how the received data is used for player move predictions can be found in section 4.2.10.

4.2.4 Nonogram Puzzle Progress Request

The primary feature of the backend server is generating hints for the user based on their progress on the puzzle. Therefore, the *Puzzle Progress Request* is the main and only functionality in the backend server that the user is calling themselves.

Diagram 4.6 describes the overall pipeline of the *Puzzle Progress Request* in generating a useful hint for the player.

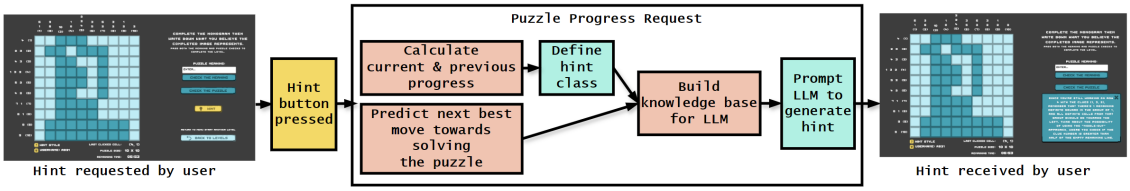


Figure 4.6: This request calculates the user's current and previous progress on the Nonogram to determine the appropriate hint class. The hint class hierarchy defines the level of information provided. Using the user's progress and recommended next steps, it builds a knowledge base for a Language Model to generate an encouraging and informative hint.

First, this request calls a function to calculate the user's current and previous progress on the Nonogram (section 4.2.7). It then uses this progress and its differences to specify the hint class that the user should receive (section 4.2.8). The hint class hierarchy determines the amount of information that one hint should provide the player with. Hence, after defining the hint class, specific information based on the user's level progress and recommended next steps (section 4.2.10) is modelled into a knowledge base, which is provided to a Language Model for generating an encouraging and informative hint (section 4.2.9).

Afterwards, the generated user-tailored hint is forwarded onto the *Verbalise Hint Request* to synthesise an audio version of the text response (section 4.2.5). Both which in itself will be forwarded back to the user through the game UI.

4.2.5 Verbalise Hint Request

All hints are shared with the user in both audio and text formats. After the player requests a hint, and the specific hint is generated by the LLM, the server sends the LLM's hint to a text-to-speech pipeline. This pipeline is hosted by a secondary flask server [49] that receives the transcript of the hint and uses the Azure Speech Services free tier [52] to synthesise it into an audio file. The audio is then played aloud to the player and the text transcript is sent back to the game to be displayed on the Hint Pop-up Display on the *Level Screen*.

4.2.6 Record Game Data Request

The server records all user in-game data, including information about each level's completion rate, duration of completion for each task in the level (puzzle and meaning guess tasks), and number of hints used for each level throughout the game. This data is used for analysis of the user's improvement in the Nonogram solving task and how much the system helped the user improve their problem-solving skills. This analysis succeeds the participants' experiment presented in chapter 5.

4.2.7 Level Progress Calculation

A method that accounts for both correctly and incorrectly filled cells is defined to quantify the progress of a user in solving a Nonogram puzzle. It provides a measure of how close the user is to completing the full puzzle correctly. Thus, the progress equation below depends on the number of correct cells and number of mistakes made by the player at a given time. It is as follows:

$$\text{Correct Cells} = T - W - M \tag{4.1}$$

$$\text{Current Progress} = \frac{\text{Correct Cells}}{T} = \frac{T - W - M}{T} \tag{4.2}$$

$$\text{Empty Grid Progress} = \frac{E}{T} \tag{4.3}$$

$$\begin{aligned} \text{Normalized Progress} &= \frac{\text{Current Progress} - \text{Empty Grid Progress}}{1 - \text{Empty Grid Progress}} \\ &= \frac{\left(\frac{T-W-M}{T}\right) - \left(\frac{E}{T}\right)}{1 - \left(\frac{E}{T}\right)} \end{aligned} \tag{4.4}$$

In the equations above, T represents the total number of cells in the puzzle, W represents the wrongly filled cells, M represents the missing cells that need to be filled as in the solution, and E represents the total empty cells in the solution grid.

Number of Correct Cells: This is the count of cells that the user has correctly filled according to the solution grid. This can be calculated by subtracting all types of mistakes from the total number of cells in the puzzle.

Number of Mistakes: This includes both missing filled cells (cells that should be filled but are not in the current puzzle) and wrongly filled cells (cells that are filled in the current puzzle but should not be).

Empty Grid Progress: This is the progress value that the an empty grid defines. In am pmty grid the user has made no changes, thus such value is supposed to be considered as 0 progress. However, equation 4.2 returns a non zero value for the initial empty grid, as some of the empty cells match the solution (as they are supposed to be empty) and are considered correct cells. Therefore, for the empty grid to be considered to have 0 progress, normalisation is needed.

Normalisation: Normalisation is crucial because it sets the initial empty grid to be considered to have 0 progress. Therefore, a relative progress to the initial empty grid follows the equation 4.4. Additionally, an inverted grid (all supposed to be empty cells are filled and vice versa) will have the maximal negative value, as all cells in the grid are considered mistakes. In such a grid, *Correct Cells* and *Current Progress* are 0, thus *Normalized Progress* only depends on the *Empty Grid Progress* and returns a negative value. By normalising the progress, we ensure that the value remains within a valid ± 1 range.

This approach and equation allow for effective tracking of the user's progress, offering a clear and quantitative measure that can guide the provision of hints throughout the puzzle-solving process.

4.2.8 Hint Hierarchy

The main feature of the system is to share customised hints to the user depending on their progress within the game and current level. Therefore, the primary development focus was on ensuring the hints are diverse, relevant, knowledgeable and constructive.

The system's hints are organised into four hierarchical levels: *General Hints*, *Directional Hints*, *Conclusive Hints*, and *Puzzle Meaning Hints*. This hierarchy is applied to both control groups in the experiment, although the knowledge base on which the hints are based on differs. Figure 4.7 illustrated the Hint Hierarchy, detailing each level, its definitions and an example.

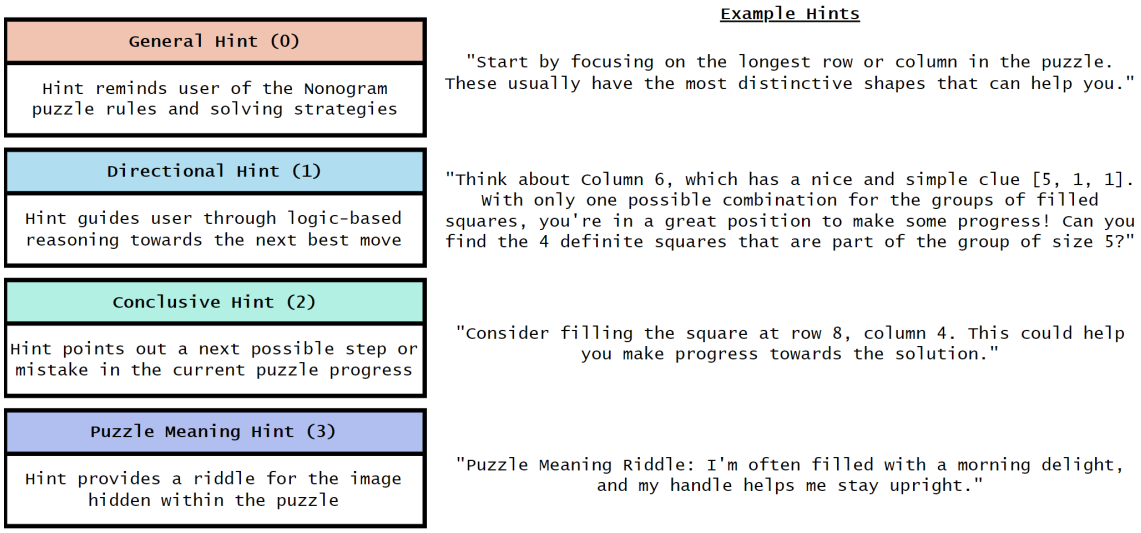


Figure 4.7: Hint Hierarchy defines the various hint classes by employing progressive disclosure. Hints are distinguished by their roles in helping the player and the knowledge they are based on.

The act of changing between hint classes is employing the strategy of progressive disclosure [53], where information or functionality is revealed incrementally as the user progresses through the task or in this study requests successive hints. This approach helps manage complexity by providing only the necessary information at each stage. Hence, with each hint requested, if the user's progress on the level has not increased, then progressively more information will be provided through the hints. Initially, hints will be very high-level, offering general rules of Nonograms and describe recommended locations to focus on, but not provide conclusive guidance. For example, an early hint can instruct the user to focus on a specific part of a row without mentioning the exact location of a mistake. This method ensures that users are encouraged to engage in critical thinking and problem-solving, receiving only the level of assistance they need at each stage depending on their progress, and gradually leading them to the solution if they continue to require help.

Figure 4.8 describes the approach for adjusting the *Hint Level* based on the user's progress and over time. The graph on the left shows the types of hints that the user will encounter depending on their current progress in a puzzle. For instance, if the user has just started the puzzle (progress $< 30\%$), they will only be presented with either general (0) and descriptive (1) hints. As they near the end of the puzzle, it is assumed that the user has understood the general rules of Nonograms. Hence, the system will not respond with such generalities, instead it will provide more detailed hints (such as directional (2) and conclusive (3)) that will help the user finish the last few steps in the level. Thus, the progress range defines a constraint on the possible hints that can be given at a specific time throughout the puzzle completion process. One special case is when the user has completed the Nonogram puzzle (progress 100%). In such a case, they will receive riddles pointing to the meaning of the revealed image, helping them pass the second check requirement of the level. All in all, the *Hint Level* increases if the user's progress on the level stagnates or deteriorates, resulting in more specific and detailed hints. Otherwise, if the *Hint Level* decreases, the system maintains the hints as vague yet relevant as possible. Additionally, the time graph on the right of Figure 4.8 illustrates an example of how *Hint Levels* change over time depending on the user's progress (e.g. in the 30-50% progress range, where all hint types can be provided).

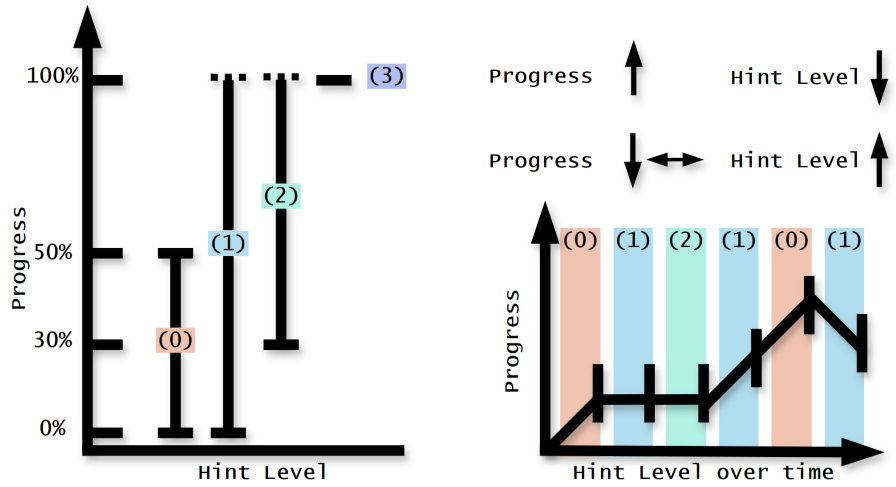


Figure 4.8: Hint Level Definition Strategy: defines how the class of the hints will change depending on the user's progress throughout the puzzle. Dotted lines represent non-inclusive ranges.

Additionally, as there are multiple levels to the game used in the controlled experiment, another constraint was added onto the general hints (0). General hints remind the user the rules and strategies in solving Nonogram puzzles, hence it is expected that towards the end of the game, the user has already understood and memorised such knowledge. Thus, general hints are not provided towards the end levels of the game (after level 3), in order for the system to be as helpful to the user as possible, and not seem repetitive or irrelevant.

4.2.9 Hint Generation

The generation of hints is done by a Large Language Model (LLM), which receives the information regarding the user's progress on a given level and uses it as the knowledge base for generating a hint. The system uses Meta's newly published `Llama 3 8b Instruct model` [1] with a *temperature* setting of 0.8, inferenced through the Microsoft Azure Machine Learning Serverless Endpoints [54]. To ensure a cost-effective use of the service and minimise the verbosity of the model's responses, each text generation task for distinct hint classes has a defined *max_tokens* parameter to limit the length of the generated responses. Hence, *General Hints* are limited to a length of 60 tokens, *Directional Hints* to 80 tokens, *Conclusive Hints* to 60 tokens, and *Meaning Hints* to 50 tokens. This limitation ensures that the hints are concise, relevant, and helpful without being overly long or distracting to the player. By keeping the hints brief, the system avoids overwhelming the user with lengthy unnecessary phrasings, thus maintaining close focus on the hint's information and enhancing the overall user experience.

Before the release of Llama 3 models (in Apr 2024), previous approaches to the hint generation used the `Llama 2 7b chat` model [55] with a *temperature* setting of 0.8. More details about the previous approaches and challenges encountered are mentioned in section 4.2.11.

Azure User Account Setup

In order to use any of the services provided by Microsoft Azure, a user with all necessary permissions had to be created on the cloud platform. This user was used throughout the development of the system in this study. The setup process involves assigning specific roles to the developer of the system, ensuring adherence to the principle of least privilege [56]. Below is a summary of the steps taken, where "user" refers to the system developer in this study:

Step 1: Full Development Access on Azure ML Workspace: Create a custom role based on the "Contributor Role". Grant permissions for actions necessary for ML Studio access and pay-as-you-go API endpoints. Assign the custom role to the user through the Azure portal.

Step 2: Allowing User to Create/Delete Azure ML Workspaces: Assign permissions for managing ML workspaces to the user, including permissions for storage accounts, key vaults, and operational insights.

Step 3: Allowing User to Manage Budgeting Alerts: Optionally, assign the role of "Cost Management Contributor" or "Cost Management Reader" to allow the user to manage budgets and alarms related to subscription usage of the allocated services.

Throughout the process, the developer's cloud user is assigned roles tailored to their specific needs, ensuring they have the necessary access without granting excessive permissions. Additionally, the administrators of the Azure subscription can monitor all changes and receive cost updates or alerts for the user's utilisation of the cloud services.

Azure LLM Inference through Serverless Endpoints

The system uses Azure Machine Learning's serverless endpoints to inference Meta's Llama 3 8b Instruct model under a pay-as-you-go subscription. This setup allows for scalable, cost-effective text generation by leveraging serverless technology. In this study, the text generation capability is used for generating knowledgeable and encouraging hints for the player to progress through the Nonogram levels. The inference process involves using the Azure REST Chat API [54], with requests formatted appropriately for Llama 3 using the LangChain Python package [57]. An example REST API request is found in Appendix C.1. This ensures that the model receives data in the expected format, enabling seamless interaction.

Additionally, to accurately measure the user experience, all latencies of hint generation are measured from the client (game). Section 4.3.2 presents all measured latencies and their average values throughout the entire experiment of this study.

Prompt Engineering for Hint Generation

Prompt Engineering approaches take advantage of models' capabilities and try to achieve a high performance by optimising the knowledge provided through the prompts. It involves writing effective inputs (prompts) to obtain desired responses from language models. This involves techniques for wording and structuring inputs (user or system prompts) and iterating on prompt designs to improve the model's performance on specific tasks.

Inferencing a chat-based LLM involves the writing of a user prompt and system prompt. Considering the conversational exchange as a chatroom, the user prompt is represented by the user's message in the chatroom and the system prompt is the chatbot's initial knowledge base, which is set by the administrators to provide information on its task and capabilities. However, for this game the communication with the model is done in the backend completely, with both the user and system prompts being written by the backend server. The server integrates the user's progress data from the level with a preset formatted message. Thus, the LLM can only provide responses based on the data the system shared about the user's progress.

Further information can be found below on each hint type generated by the Llama 3 8b model and the prompt engineering approaches that were used to achieve a reliable and accurate hint. The full prompts used for each hint type can be found in Appendix C.

General Hints are provided primarily in the beginning levels and early stages of the puzzle (when progress is less than 50%) or if the player has not interacted with the grid in any way (assuming that the player is confused on how to start the level). These hints describe the rules and best strategies to solve Nonograms. For instance interpreting numerical clues and starting with the largest groups first as they are the easiest to complete. For *General Hints*, the LLM takes advantage of its chat-based capabilities and it remembers previously generated hints to ensure different information is provided with each request, demonstrating an aspect of memory. Hence, the user prompt defined for this hint class is:

"I am a beginner and need some help. Give me a different hint than before."

An example of a *General Hint* is as follows:

```
"When starting a new row or column, it's often helpful to look for the row or
↪  column with the largest clue number first. This is because the larger the
↪  clue number, the more information you have to work with, and the easier it is
↪  to make progress."
```

Directional Hints constitute the majority of the hints a player might get, as they can be provided at any point regardless of the player's progress. These hints use a solver algorithm (section 4.2.10) to generate an accurate knowledge base to guide the player to a recommended area of the grid, such as a particular row or column, and focus on a specific group of filled cells on that line. This recommendation is based on the player's progress in the game and their area of focus. Additionally, it contains information regarding an entire row or column which the player should focus on next. Such information includes: the line Nonogram clue, the number of possible combinations that the line can be solved in, the number of remaining definite cells that are on that specific line and the size of the groups of filled cells that the definite cells are part of.

Afterwards, the information is organised into multiple variables to be incorporated into a system prompt specific to the LLM's task. This primary task is to help the player identify the definite cells than can be filled in the recommended row or column. Sometimes, the LLM also provides an explanation of the strategy that the solver algorithm is built upon. For example it encourages placing the groups of filled cells in all possible combinations on the given row or column in order to find the cells that preserve their state regardless of the combination tried (e.g. definite cells).

For *Directional Hints*, the LLM also takes advantage of its memory capabilities, and uses past provided hints as reference to not repeat the same information given before. Thus, it can rephrase a hint based on the same knowledge base, in order to help the player see a different perspective of the focus area in the puzzle grid. Thus the user prompt defined for this hint is:

"I am stuck when solving the puzzle, give me a hint on what to do next. If you have already given a hint about {line_ index} with the clues {line_ clue}, then that means I have not found all definite squares. Give me another hint with more information about it. Do not tell me about other lines."

Where the *line_ index* refers to a specific row or column, and the *line_ clue* specifies the numerical clues provided for that specific line.

An example of a *Directional Hint* is as follows:

```
"Check to Row 1 with clues [8, 1]. Since all definite cells from the group of 8
↪  should be towards the left of the row, and all definite cells are still left
↪  to find, think about how you can satisfy this condition. Remember that
↪  there's only one possible way for this row to be filled."
```

This hint could be provided in the following situation (illustrated in Figure 4.9). A player is solving a 10 by 10 puzzle and has last interacted with row 2 near the top of the grid. At this point the solver algorithm will analyse which other rows in that area are easiest to complete next. If row 1 is identified as the easiest to complete, the hint would provide information recommending it. Such information could include the column index of a specific cell on that row, the row clues [8, 1], or the number of remaining cells to be filled (9 in this case). Additionally, the hint might clarify that these cells correspond to the groups of size 8 and 1 within that row and where they are located in relation to the row. This allows the player to deduce that they can fill in all cells indicated by the clues, thereby completing row 1.

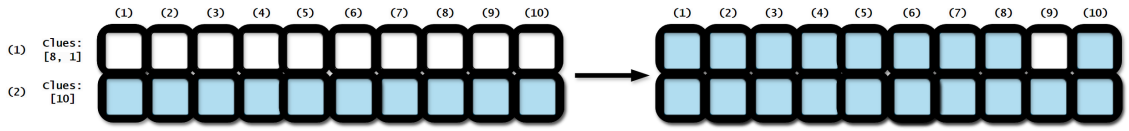


Figure 4.9: Section of 10 by 10 puzzle grid illustrating the actions a player can take by following a *Directional Hint*. The hint recommends the player to consider the remaining definite cells on row 1, which are in the vicinity of the area the player is focused on currently. The hint refers to the clues of the row: [8, 1] and confirms that there is only one possible way of completing that row. Blue cells signify filled states, and white ones are empty.

Conclusive Hints are only provided after the player has completed more than 30% of the puzzle. These hints assist by pinpointing the location (row and column) of a mistake, offering guidance such as, "Focus on location (x,y). It might help moving forward.". This approach aims to direct the player's attention to specific locations that may need correction. Thus, the LLM receives a list of mistakes from the current grid and has to tell the player to reconsider their locations. These mistakes are found by using a solver algorithm (section 4.2.10) that recommends the first two mistakes that are easiest to correct. This recommendation is based on the player's progress in the game and their area of focus, determined by their previous interactions with the grid.

The LLM in this situation, does not remember previous mistake locations as it was observed that it induced a high probability of hallucinations during the development process. Thus, the user prompt used for this hint class is simpler than the for the previous classes:

"I am stuck and need help. Give me a hint on what to do next."

An example of a *Conclusive Hint* is as follows:

```
"Take another look at the square at row 5 and column 2. It might hold a crucial
↪  clue to help you progress in the puzzle."
```

Puzzle Meaning Hints are provided when the player has successfully completed the Nonogram. These hints contain a riddle that points to the solution meaning of the image revealed by the completed Nonogram. The LLM in this case, does not remember the previous provided hint, as the task is reasonably general and the randomness induced by the model's parameters (e.g. $temperature = 0.8$) bring enough variety in the generated hints.

An example of a *Puzzle Meaning Hint* is as follows:

```
"He's a farm favorite, with feathers bright, and crows at the break of light."
```

4.2.10 Nonogram Solver Algorithm for accurate Hints

To generate relevant and accurate hints, the knowledge base for *Directional* and *Conclusive Hints* must rely on a logic-based algorithm designed to solve Nonogram puzzles. Therefore, a Nonogram solver and validator was adapted from Hennie de Harder's article "Solving Nonograms with 120 Lines of Code" [58]. This adaptation extends its capabilities beyond solving a Nonogram from scratch, enabling the provision of both tailored and untailored hints by recommending the next action based on the user's current progress and predicting definite cells in the user's focus area.

The `NonogramSolver` class contains all functions necessary for a) solving the nonogram, b) predicting the next best steps from a current progress grid, c) providing information on the recommended row or column to focus on, and d) provide a random step throughout the solving process. The first 3 capabilities are mainly used for generating user-tailored hints, however the last capability is specifically for generating untailored hints for the player. The class is initialised with several key attributes necessary for solving a nonogram puzzle, including lists for row and column clues (`ROWS_VALUES` and `COLS_VALUES`), grids to track progress (`PROGRESS_GRID` and `SOLUTION_GRID`) and a list of user's last interactions with the puzzle (`LAST_INTERACTIONS`). It also tracks completed rows and columns throughout the solving process with `rows_done` and `cols_done`.

The `PROGRESS_GRID` and `SOLUTION_GRID` define the current and goal states of the puzzle grid using a 2D list, where each element represents a grid cell state. The solver defines all uncertain cell locations as *initial* (0). By calling the `solve` or `recommend_next_action` functions, these initial cells are processed to determine their definite state, either *filled* (1) or *empty* (-1). Thus, the algorithm searches for both *filled* and *empty* definite cells, similar to a human player solving the Nonogram.

Solving the Nonogram

The `solve` method aims to find all the steps required to complete a puzzle from an empty grid state. It begins by generating all possible solutions for each row and column based on the numerical clues provided by the Nonogram. Next, it selects the row or column with the fewest solving combinations, as this is easier to work with. Using this line, the method identifies cells that have only one possible value throughout all of its solving combinations, known as *'definite cells' or 'definite squares'*. The process of identifying definite cells is shown in Figure 4.10. Definite squares have the same state

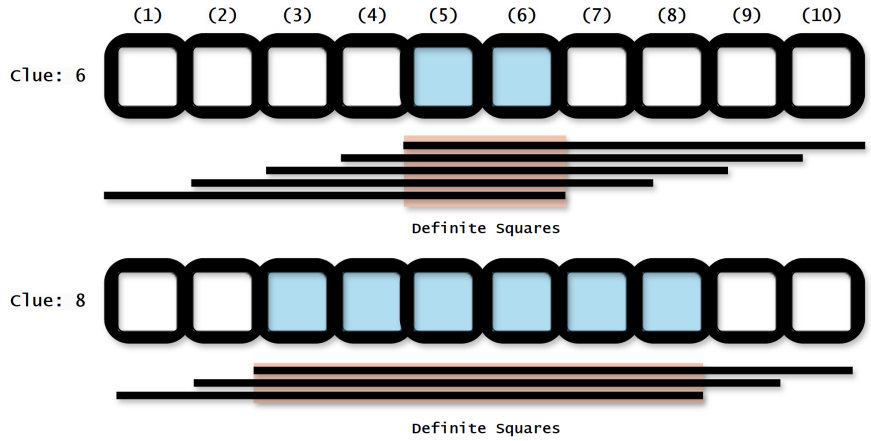


Figure 4.10: The Nonogram Solver searches for the rows or columns with the least number of possible combinations for the locations of the groups of filled cells. Once a row or column is parsed, it searches for the *definite squares*, that have the same state (filled or empty) throughout all the possible solving combinations (signified by the lines below the rows). This is an approach most experienced trial participants have explained to be using while completing Nonograms.

(filled or empty) throughout all the possible solving combinations of a line in the grid. Thus, regardless of the arrangement of the cells, the remaining definite cells are the best next steps for the player to take towards the completion of the puzzle. These definite cells are then used to update the board and to remove possible solutions of rows and columns that do not match the newly set cells. This process repeats until the puzzle is solved.

Next Best Steps Prediction

The `recommend_next_action` function suggests steps for the player to take throughout the solving process of the Nonogram. The next best step signifies the location of a grid cell and its respective goal state that is predicted to be the user's next step towards the finalisation of the puzzle. Hence, any reference of *steps* or *moves* are to be considered as grid cells to have their state changed to their goal state. The function accepts four parameters: `no_next_steps` (number of steps to recommend), `whole_line` (if set to True, returns all recommended steps for a whole row or column), `random_line` and `random_linewide_step` (if both are set to True, function returns all recommended steps for a row or column at a random point during the solving process).

Therefore, the `recommend_next_action` has 3 main capabilities depending on the arrangement of the parameter set. Firstly, the function can predict the next n best steps from a current progress grid. Secondly, the function can return information regarding a whole row or column where the recommended next steps are. Thirdly, the function can return the information regarding a row or column at a random point throughout the solving process of a puzzle.

Overall, for all capabilities, the function initialises by aligning the board state with the user's progress grid, and corrects any discrepancies with the solution grid. Similar to the `solve` function, it defines all possible solutions for each row and column based on the given grid. Then, in a while loop, it prioritises rows and columns with least number of solving combinations, identifies definite cells, and updates the board and other solving solutions throughout whole grid accordingly. The loop is running until the condition set by the specific capability is passed or the puzzle is solved. Following sections present each of those conditions for the specific capability of the `recommend_next_action` function:

I. Predict n next best steps: To predict n recommended steps, a counter keeps track of the found definite cells throughout the completion of the puzzle starting from the user's current progress grid. Therefore, this function capability returns n recommended steps regardless if they are from multiple lines in the grid. Algorithm 1 presents a high level preview of the function's approach for both predicting n next steps but also return all definite cells along the next recommended row or column as describes in feature II below.

Algorithm 1 Recommend Next Action in a Nonogram Puzzle

```
{Step 1: Initialize the board and correct errors}
board = correct_errors(progress_grid)
{Step 2: Define possible solutions for rows and columns & Eliminate inconsistencies}
possible_solutions = generate_possible_solutions(rows, cols)
clean_solutions(possible_solutions)
while not solved do
  {Step 3: Prioritize rows and columns for most appropriate recommendations}
  priority_indices = sort_by_priority(possible_solutions)
  {Step 4: Recommend next cell to fill in the prioritised rows and columns}
  for index in priority_indices do
    {Step 5: Identify cells with consistent values across all possibilities}
    consistent_cells = find_consistent_cells(possible_solutions, index)
    {Step 5: Record each Definite Cell as a next step to be done on that line}
    for cell in consistent_cells do
      if consistent_cells is available then
        print(f"Recommended cell:  row:  {row}, col:  {col}, val:  {val}")
        return row, col, val
      end if
    end for
  end for
  {Step 6: If no recommendation, check if puzzle is solved}
  if is_solved(board) then
    print("Puzzle solved.  No further recommendations.")
  end if
end while
```

II. Predict next recommended row or column: The `recommend_next_action` function can also recommend an entire row or column for the player to focus on next, if `whole_line` is set to true. The returned information includes all definite cells found in that row or column. Additionally, it provides further details gathered during the solving process, such as the number of possible solving combinations for the recommended row or column. For instance, in Figure 4.10, the row with a clue of 6 has 5 possible ways to be solved with 2 definite cells, while the row with a clue of 8 has only 3 combinations and 6 definite cells. In both cases, the middle cells in the row consistently preserve their states across all combinations, thus being the *definite squares* returned for that specific line.

III. Predict a random row or column: Finally, the `recommend_next_action` function is also used by the experiment control group receiving untailored hints. By setting `random_line` and `random_linewide_step` to True, the function returns all recommended steps for a row or column at a random point during the solving process. This means the algorithm does not track the user's current progress on the puzzle but instead solves the Nonogram from scratch and randomly selects one of the steps it applied during solving. Consequently, the player might receive information about a step they have already completed or a step far in the future, highlighting definite cells that are not yet certain to the player. The reasoning behind this capability is for players to experience randomised hints, similar to pre-set hints that purely return a complete random row or column in the puzzle, as seen in other Nonogram online games [59].

4.2.11 Previous Approaches for LLM Prompting

This section discusses previous approaches implemented during the system's development, along with the conclusions and lessons learned from them. All previous methods utilised the `Llama 2 7b chat` model and `Mixtral-8x7B-Instruct-v0.1` for understanding the knowledge representation necessary for hint generation. These approaches primarily relied on the spatial orientation capabilities of language models, which proved to be quite limited. Therefore, the latest version of the LLM prompting system incorporates improvements based on challenges overcome during the trials of previous approaches. The approaches are presented chronologically, highlighting the main challenges encountered and how they were resolved or adapted in the current implementation.

The earlier approaches were based on the hypothesis that if LLMs can understand tables and matrices, they might also comprehend spatial grids. However, a significant challenge encountered was that LLMs have very constrained spatial reasoning capabilities, particularly with 2D grids and maps. This limitation became evident when trying to apply LLMs to 2D crossword puzzles, such as Nonogram or Sudoku. As a result, the focus shifted towards leveraging LLMs for verbalising complex algorithms and knowledge bases that are difficult for humans to interpret directly. Consequently, algorithms were developed to generate the metadata that the LLM would use as a knowledge base, which could then be presented through natural language.

I. Using chat LLM models for all hint generations

Firstly, utilising chat versions of LLMs for hint generation presents certain challenges. One significant issue is that the model may attempt to continue or hallucinate a further conversation with the user, leading to the generation of fake questions and answers within its responses. This behaviour can strongly affect the accuracy of future responses, as the model incorporates these fabricated conversation points into its memory, resulting in a positive feedback loop, thus increasing hallucinations and further degrading overall performance.

To mitigate this, the latest version of the system limits the response length for each hint and defines a breaking character or term for the LLM to use when finishing its response to the user's prompt. For instance, the system only accepts completed sentences with an ending punctuation and it ignores anything generated after a new line. Additionally, all current system prompts (Appendix C) specify the expected response format, such as ensuring the LLM starts its response with the term *'Hint:'*. This way, if a hallucinated conversation is generated, it can be removed before the response is sent to the user and added to the conversation history.

II. Using whole matrix grid for observations

Secondly, one method tested involved passing the user progress and solution grids to the LLM to see if it could identify the differences and point them out. The expectation was for the LLM to compare the two matrix-shaped grids and detect cells with dissimilarities (filled versus empty states and vice versa). The LLM was then asked to use natural language to describe the overall location of the mistakes found relative to the grid (e.g., "Mistake found in top right corner of the grid"). However, the conclusion from this approach was that the LLM is not capable of understanding 2D grids and comparing them effectively.

To simplify the approach, the LLM was provided with the known mistake locations and the solution grid, and asked to make observations regarding the cells' locations. Despite this simplification, the same challenge was encountered. The LLM's limited spatial reasoning capabilities resulted in inaccurate directions and orientations relative to the grid/ For example, it incorrectly described a mistake at the bottom right as being in the "top left".

Another attempt involved providing the LLM all the information about the Nonogram puzzle in a table format, such as including the row and column indices or numerical clues along the solution grid, as seen in the example below. The LLM would also have the format of the grid and of the numerical clues explained in the system prompt. This approach would provide the LLM with the knowledge of the whole game and the appropriate solution of the puzzle. It was expected that having the mistake locations, it might detect trends in the solution grid and make out relations between the goal cell states. Thus, providing observations about the cell states in the area of the locations.

However, this approach overwhelmed the LLM with too much information to accurately respond to the task. It often confused row hints with the actual grid and mistook column hints for row hints, even when their purposes were explicitly mentioned in the system prompt. Other formats of the prompt were also tried (such as providing the row and column clues in separate lists and the solution grid as a table), but those too resulted in many inconsistencies in the responses. Thus, the low performance and accuracy led to the next approach on knowledge base representation.

```
Puzzle meaning: "coffee cup"
Grid:
    (         [1] [5, 1] [2, 7] [1, 4] [2, 7] [7] [2, 7] [5, 1] [1, 2, 1] [2, 1])
    [[1, 1, 1] 0 0 1 0 1 0 1 0 0 0]
    [[1, 1, 1] 0 0 1 0 1 0 1 0 0 0]
    [[0]       0 0 0 0 0 0 0 0 0 0]
    [[9]       0 1 1 1 1 1 1 1 1 1]
    [[2, 4, 1] 0 1 1 0 1 1 1 1 0 1]
    [[2, 5]    0 1 1 0 1 1 1 1 1 0]
    [[8]       0 1 1 1 1 1 1 1 1 0]
    [[7]       0 1 1 1 1 1 1 1 0 0]
    [[1, 5, 1] 1 0 1 1 1 1 1 0 0 1]
    [[8]       0 1 1 1 1 1 1 1 1 0]
```

Figure 4.11: Previous Method II: providing the LLM with all information about the Nonogram solution and detected user mistakes

III. From Grid to Language: Using gradual spatial descriptors

The third method involved using a step-by-step approach in creating spatial descriptors for the LLM to use as a knowledge base. It involved subdividing the large task into smaller, simpler tasks that the LLM can handle more effectively. This task-based pipeline allows the LLM to complete each sub-task sequentially, passing information from one layer to the next. Below are the steps forming the pipeline. Beginning steps are completed by algorithmic functions defined in the system, followed by the generative steps done by the LLM. The system prompts previously used for the LLM in these stages are under Appendix C.6.

1. **System:** Identify mistakes in the user's progress grid and return their locations.
2. **System:** Select one of the mistake coordinates at random and generate simple natural language localisation metadata relative to the grid (e.g., top row, bottom of grid, left side, right column, centre of grid, middle of row, row X, column Y).
3. **LLM:** Use the localisation metadata from Step 2 to reformulate and enrich the description, introducing randomness in the natural language localisation phrasing. (e.g. row towards top of the grid, bottom of column)
4. **LLM:** With the chosen mistake coordinates and localisation metadata from Step 2, return an observation regarding the solution cells in the surrounding area.
5. **LLM:** By knowing what a Nonogram is, use the responses from Steps 3 and 4 to generate a hint about the area of the grid where the mistake occurred. Such hint prompts the user to reconsider their choices and check the described area for mistakes.

By following such pipeline, the LLM is expected to consider the spatial reasoning rules outlined below when developing the observations on the area of the mistake. Such rules are referenced from Cohn et al. [60], a study which describes an alternative evaluation approach for LLM's consistency in spatial reasoning: dialectical evaluation. The study illustrates that by engaging in dialogue with the system, the consistency of the LLM's spatial knowledge base can be quantified. This approach was used throughout the development of Methodology III in order to find optimal prompts that would allow for the LLM to provide reassuring and robust responses for the sub-tasks. The study specific spatial rules defined are:

Inclusion and **exclusion** of a cell that has the wrong state (filled or empty) in relation to its surrounding states.

Part-hood signifying the grouping of cells of the same state (e.g., mistake part of consecutive cells filled in a row).

Permanence of the grid solution (e.g., surrounding cell states remain unchanged after correcting the mistake).

Therefore, the following decisions were taken for the models used at different stages in the pipeline:

STEP 3 and 4: In contrast to `Llama 2 7b chat`, an instruct model (`Mixtral 8x7B Instruct v0.1` hosted on the Hugging Face Inference API) proved to perform better by returning

conciser and more relevant responses on the tasks. This is expected because instruct models are optimised for tasks involving guidance, following instructions, or understanding queries framed in an instructional manner. Hence, by describing the task in the system prompt of those steps in the approach, they generate responses that are coherent, relevant, and aligned with the given instructions. For instance, if the localisation metadata indicates "left side of row 5" `Llama 2 7b chat` model would hallucinate and return "top-left of row 5". However the instruct model will return a synonym for "left of row", such as "west side of the row", or enrich the description, such as "towards the left margin of the row".

STEP 5: Llama 2 model is used for this step, due to it offering more versatility in expressing ideas in a conversational manner, making the hint sound constructive and encouraging, while keeping it relevant to the provided information.

Additionally, during the development of this knowledge generation approach, a recent paper by Sharma et al. [29] introducing the concept of Spatial Pre-Prompting was encountered. Its described methodology bears similarities to this approach, as it involves asking simple questions first to ensure the LLM possesses the necessary understanding to tackle more complex queries. Spatial Pre-Prompting however created a conversation pipeline with an LLM, thus allowing the LLM to depend on the memory of its previous simple requests and responses. This process of gradual questioning aligns with this approach's incremental provision of spatial information to provide the LLM with accurate description of grid-based puzzles and the user's predicted best next moves.

All in all, with the integration of the newly published Llama 3 Instruct model, the current system follows a similar approach of preparing a metadata knowledge base containing all information the LLM needs regarding the user's progress in the puzzle and the task at hand. However, due to the improved performance of Llama 3 compared to Llama 2, the pipeline is simplified to use a single main system prompt that contains all the necessary data. A more detailed discussion on the differences in performance between Llama 2 and Llama 3 models if found in the following section.

4.2.12 Discussion: Comparing & Evaluating Models' Performance

Overall, gathering human trial data is one of the main procedures to understand human-computer, and more specifically human-LLM interaction, and how well the LLM system performs to match the human standards of communication and assistance. This is because, the tasks that LLMs are developed to perform can be considered as social tasks that involve attributes such as hosting fluent conversations, retaining ethical ideologies, and presenting helpful demeanour. However, current research discusses the problem of such procedures not being enough for evaluating of LLMs [61].

Doshi-Velez et al. [61] discusses the importance of "contextual evaluation" in AI, where personal-context-dominating settings are considered the standard evaluation methods. This contrasts to the currently widely used evaluation procedures, such as benchmarks and user studies. "Contextual evaluation at task-time" is encouraged due to the inability of benchmarks to generalise the overall use-cases involved in an NLP task. For instance, Doshi-Velez et al. [61] mentions the impossibility of developing a benchmark that describes all possible types of documents that an LLM would be prompted to summarise. This is due to such NLP tasks requiring an immeasurably broad space of inputs. Therefore, by setting the evaluation to task-time, it could be possible to focus on the performance shown for the specifically given user task and then use various of those scenarios to develop a more general LLM evaluation metric. "The fact that correctness of an AI system output may be dominated by user and task specific considerations further questions how one might construct a procedure to meaningfully validate such an AI system in advance"[61]. Thus, with this perspective in mind, the following approach was implemented to evaluate the performance of the models in the study's specific Nonogram hint generation task (detailed in sections 4.3.3 and 4.3.4).

During this project's development, Meta released a new package of Llama models, including instruction fine-tuned versions, which were advertised to significantly outperform the earlier Llama 2 models. These new models were quickly made available in the Azure AI's model catalogue, allowing for their early integration into this project. However, due to limited official resources from Meta on the performance of the new Llama 3 models as of the time of writing (June 7, 2024), comparative scores published online by developers and a task-specific comparative test for this study's functionalities were used as the main basis for integrating the new Llama 3 model. The comparisons were made against the previously used `Llama 2 7b chat` and `Mixtral-8x7B-Instruct-v0.1` models.

To observe the performance differences between multiple models in the study's specific task of hint generation, a test was done throughout the testing and development of the system. The test included multiple comparisons of the hint quality generated based on the same or very similar system prompts, which indicate the task for the LLM to complete. During the comparison of performance, hallucinations and relevance to the provided knowledge base were the main points of reference. Llama 2's performance was used in the implementation of Methodology III (section 4.2.11) for reliable generation of hints based on the Nonogram puzzle data. This involved multiple prompting stages chained together to ensure the knowledge base was accurate for hint generation, as well as various steps and intermediate representations. In contrast, Llama 3, with its enhanced training to understand and execute user commands and requests, significantly simplified the process. Each hint class could be achieved with a single prompt, as opposed to the multiple chained prompts required for Llama 2.

Additionally, a consideration of widely used benchmarks was made. The official benchmark scores [62, 1] in table 4.1 were used as reasoning basis for which of the models to use. However, the scores are expected to not be strictly proportional to the performance in this study's task, thus they were used as mainly a reference point of the possible ranges of the model's abilities. The communication abilities can be referenced through the Chatbot Arena score [63] which can imply a more natural sounding hint, whereas the MMLU score [64] can imply a more relevant and accurate hint based on the knowledge base provided.

Task	**Model**	**Benchmark (score)**
Communication abilities	Llama 3 8b Instruct	Chatbot Arena (1153)
	Mixtral-8x7B-Instruct-v0.1	Chatbot Arena (1114)
	Llama 2 7b chat	Chatbot Arena (1037)
Reasoning and knowledge	Mixtral-8x7B-Instruct-v0.1	MMLU (71%)
	Llama 3 8b Instruct	MMLU (66.6%)
	Llama 2 7b chat	MMLU (45.3%)

Table 4.1: Comparing models used in previous approaches, before the release of Meta's Llama 3. Higher scores are better. Descending order based on published score on each benchmark.

Considering the comparisons made throughout the development process of this system and the published benchmark scores, the improved capabilities of `Llama 3 8b Instruct` model led to a simplified system design in contrast to using Methodology III's pipeline of `Llama 2 7b chat` and `Mixtral-8x7B-Instruct-v0.1` prompting. Additionally, `Llama 3 8b Instruct` also proven to provide a more natural and accurate hint generation process.

4.3 Technical Evaluation & Future Work

4.3.1 User Intuitive Design

One main requirements of the game interface design is to prioritise user-friendliness and clarity, ensuring players can navigate and interact with the system effortlessly. Key design features implemented that ensure this requirement include:

- **Guidance Through Notices and Animations:** To avoid user confusion or frustration, the system provides clear feedback at every stage. Animations and notices inform players of system states, such as progress checks, hint popups, time running out, and game completion. These visual cues help players understand what is happening in the background and shape their expectation of what will happen next.

  Additionally, the integration of the *Tutorial Level Screen* as the first level in the game allows users to understand the goal of the game and every interaction they can make with the system. As well as every response they can receive from the system. Hence, the system prioritises its transparency and the user's experience from the very beginning of the experiment.

- **Multiple Interaction Methods:** Various methods were considered for allowing users to interact with game features. For instance, the login feature allows users to connect to their local accounts and view their previous progress in the game levels. The login input box saves the entered username as the current user through various intuitive interactions. Despite providing a <Save> button (as seen in Figure A.2 in Appendix A), some users might not press it directly. Hence, the username was developed to get automatically saved by also pressing the <Enter> key or clicking anywhere outside the input box, such as the <Start a Game> button. This flexibility is also considered for the Puzzle Meaning Input box, where users can initiate the meaning check progress either by pressing <Enter> or clicking the <Check the meaning> button.

- **Accessibility Features:** Accessibility is also a crucial aspect of the game design. Therefore, hints are provided in both audio and text formats. This allows users to focus on the puzzle grid while listening to the audio and then refer back to the text transcript if they need to review the hint. This dual-format approach ensures that players can access and understand the hints effectively, regardless of their preferred learning style.

By incorporating these design elements, the system aims to create a seamless and enjoyable experience for users, minimising confusion and maximising engagement with the game. This can also be seen in the user's responses in the post-experiment questionnaire, where 50% of players mentioned that it was extremely easy to interact with the AI assistant, followed by 35% mentioning a somewhat easy interaction. Additionally more than 90% of the participants described their overall satisfaction with the system as high and very high.

However, the user study also presented constructive opinions on how to further improve the system. Here are a couple of ideas mentioned as **Future work**:

- Some participants mentioned asked for the ability to skip or stop the audio AI assistant, hence speeding up the audio session and receiving the text-formatted hint sooner.

- An idea of a hint style toggle could be implemented to provide the user's the choice of having only self-initiated hints or also receiving automatic system-initiated hints.

- The pre-questionnaire also presented a few opinions expecting the AI Assistant to provide hints when it detects the user to be stuck or when they make repetitive errors. This functionality would enhance the customisation of the assisting feature of the system, as not only the hints but also the time gap between automatic hints would depend on the user's progress.

- The untailored hints could try to predict the user's last interacted cell location based on their progress instead of providing completely random steps. Such predictions could try and match the user's progress on the puzzle with the algorithmic solver's progress, thus predicting the possible steps that the player could have done last. However, this should have a high error margin to induce enough randomness and create a feeling of mismatch between the user's progress and the hint provided in order to be considered untailored.

4.3.2 System Latency Measures

The system developed for this study offers an intuitive and interactive way for players to further develop their puzzle-solving skills, specifically in the context of Nonogram puzzles. However, to fully evaluate its user-centric nature and alignment with the study goals, it is essential to quantify the latency of the hint generation. Given the distributed nature of the system, it is needed to measure the delay between the user's request and the game's response to assess the quality of features impacting the user experience. This evaluation will provide insights into the system's performance and its effectiveness in delivering timely hints to the players.

Server Knowledge Metadata Formulation Latency

This latency quantifies the time required for the server to generate the necessary knowledge metadata based on the game data received (e.g. puzzle state) and used for the hint generation request. By using the data gathered through the experiments, the latencies for each hint class are $0.03s \pm 0.005$ for *General Hints*, $0.29s \pm 0.300$ for *Directional Hints*, $0.07s \pm 0.045$ for *Conclusive Hints* and $0.03s \pm 0.004$ for *Meaning Hints*. It is clearly seen that the *Directional Hints* induce the longest latencies before the prompting of the LLM, due to their longer knowledge base preparation pipeline explained in section 4.2.9.

Azure LLM Prompt Response Latency

This latency measures the time taken for the Azure AI (using `Llama 3 8b Instruct` as a Serverless API Endpoint) to process the formulated metadata and generate a response. This is the metric that brings the largest impact to the responsiveness of the hint generation system. However, it is important to note that this latency is intrinsic to the Azure AI cloud infrastructure and cannot be optimised in this developed system. Instead, it reflects the inherent processing time of the external AI service. Separate latencies were measured for each hint class due to the different lengths of responses requested. Hence, *General Hints* have a generation latency of $2.60s \pm 0.917$, *Directional Hints* $2.98s \pm 0.518$, *Conclusive Hints* $2.25s \pm 0.465$ and *Meaning Hints* $1.88s \pm 0.337$.

Those measurements match live analytics for the `Llama 3 8b Instruct` provided by Microsoft Azure presenting a median latency of $1.83s$ until the first token after an API request is received and a median latency of $5.1s$ to receive a 100 token response [65]. For this system, the average response length is of 55 tokens, hence, the system's results match the outsourced analysis data. This analysis is based on data gathered in the 14 days before the time of writing (07th June 2024).

Text to Speech Pipeline Latency

This latency measures the time taken to convert the generated hint text into speech. The audio generation time largely depends on the length of the text synthesised, therefore a separate metric was calculated for each hint class: *General Hints* have a TTS latency of $1.12s \pm 0.047$, *Directional Hints* $1.48s \pm 0.419$, *Conclusive Hints* $1.24s \pm 0.137$ and *Meaning Hints* $1.29s \pm 0.166$. Therefore, as *Directional Hints* are the longest responses (60 tokens maximum), this is reflected in their synthesising latency as well.

Overall User Hint Request Latency

This overall latency encompasses the total time from the user requesting a hint to receiving its audio and text formats. It is formed by the sum of all before mentioned latencies and provides an overall view of the system's responsiveness. Therefore, overall the user would experience a latency of $2.53s \pm 0.589$ on average during the use of the hint feature of the game.

The latency values in Table 4.2 represent the average time measured across all data gathered from participants throughout the running of the experiment to test the system's performance.

Latency Type	Average	Standard Deviation	Number of Requests
General Hint Request	2.63	0.923	30
Directional Hint Request	3.27	0.608	143
Conclusive Hint Request	2.32	0.490	128
Meaning Hint Request	1.91	0.335	37

Table 4.2: Latency Measurements (in seconds) for Overall User Hint Requests

Discussion: Overcoming Server Latency to Improve User Experience

To mitigate server latency and enhance the user experience, the system employs various strategies. One such approach is displaying notice messages onto the Hint Pop-up Display (section 4.1.2), that provide feedback on the request status, such as "Sending request to Nono AI" and "Waiting on a hint". This feedback loop helps manage user expectations during the waiting period. Additionally, another approach adapted was the implementation of the buffer period (section 4.1.2) between hint requests that ensures the system remains responsive and prevents the queuing of hint requests on the server side.

4.3.3 Benchmark for Hint Quality

The generation of hints for Nonogram puzzles is a specific task that the general LLMs have not been explicitly trained for, thus, there is substantially limited data that can be used to evaluate the performance of the system. Hence there is a necessity for the development of a synthetic data generator and benchmark to evaluate the accuracy and relevance of the LLM-generated hints.

To facilitate this, a simulation benchmark feature was introduced in the system. It aims to simulate human behaviour by attempting to complete the puzzle while making mistakes, requesting for hints and following the received pointers. Such human-like simulation will allow for a substantial amount of synthetic data to be generated, which can then be labelled by human evaluators to determine the quality of the hints. Quality for this study's task signifies whether the hint clearly presents the information provided by the backend knowledge base algorithm, and does not hallucinate.

The benchmark simulator uses a modified version of the algorithm from section 4.2.10 to try and solve the puzzle. This modified algorithm is guided by two main characteristics: the probability of making a mistake at a given moment in time and the probability of following the recommended actions suggested by the hint. The first probability aims to simulate instances where users make mistakes or take uncertain steps during the problem-solving process. For example, a player is not sure whether a cell is definite but they apply a trial and error approach. The latter probability aims to simulate the users comprehending the hints and implementing their recommended actions. However, it could also account for situations where users might ignore or fail to understand the hint's recommendations, thereby signifying their decision to follow or disregard the hint.

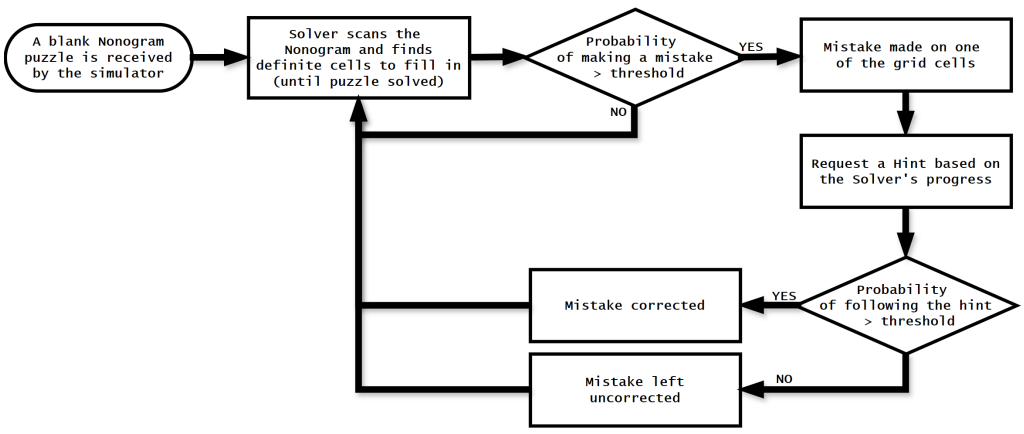


Figure 4.12: Benchmark Simulator Loop for solving a Nonogram puzzle with a chance to make a mistake and a probability of following the hint's recommendation to repair the mistake

The stages during the benchmark simulation loop are as follows:

1. The algorithm attempts to complete the puzzle step by step, but deliberately makes mistakes at certain points.
2. Upon a mistake made, the system requests a hint.
3. The hints generated during the process and their knowledge base are recorded for analysis.
4. The algorithm attempts to follow and implement the changes recommended by the hints.

The algorithm has been developed and is set to run based on the metrics gathered from the experiment explained in chapter 5. However, due to the experiment's data resulting in not being statistically significant and the time constraint of this project's outlines, the simulation benchmark development was stalled. Nevertheless, the accuracy was measured on the data-points gathered throughout the study's experiment and it shows an overall score of 92.9% accurate hints (out of a total of 155 hints). Values for each of the levels - except for the Levels 2 and 6 which did not provide hints - are shown in Table 4.4. The major factor impacting those scores was the hallucination within the *General Hints*, which sometimes referred to erroneous or nonexistent rules of Nonograms. For instance, despite clear explanations of the rules provided in the system prompt, one persistent hallucination was the incorrect recommendation that it is easier to start with shorter row or column clues.

	Level 1	**Level 3**	**Level 4**	**Level 5**	**Overall**
Hint Accuracy (%)	94.4	95.5	96.9	88.5	92.9

Table 4.3: The accuracy of the Hints generated for the participants throughout each level (except for the Levels 2 and 6 which did not provide hints)

Simulation Benchmark Future Work:

By analysing the experiment's data, it becomes possible to calculate the probability metrics for both characteristics of the benchmark simulation. Utilising real data from participants in each control group can allow for accurate simulations of real puzzle-solving tendencies and behaviours, thus also allowing to evaluate the accuracy and relevance of the hints generated by the system.

At the end of the simulation, human evaluators can assess the quality of the hints. They will have access to detailed in-game information about the player's mistake location and puzzle progress used to generate each hint. This allows them to review each hint and label them on whether their recommendations are reasonable or not. This includes the relevance and accuracy of hints. Evaluators will also assess whether the LLM provided helpful guidance or generated misleading hints due to hallucinations. All those characteristics could be labelled under one main YES/NO tag that provides an overview of the hint's reasonableness.

To ensure reliability and mitigate bias in the labelling approach, the evaluation process would involve:

- Two people labelling 60% of the hints each, with a 10% overlap to assess inter-coder reliability [66]. This overlap helps in determining how closely the evaluators' assessments align.
- Reviewing the consistency of the human evaluators' decisions to ensure that the benchmark provides a fair and accurate measure of hint quality.

This proposed approach could provide a comprehensive evaluation of hint quality, ensuring the hints are practical and helpful for players. The benchmark could help in further refining the hint generation system, making it more effective in assisting players in their learning-process.

4.3.4 Benchmark for Hint Effectiveness

In the previous section, an approach was described to measure the quality of the hints by either labelling the experiment data or by labelling generated synthetic data based on a simulation of human performance. However, this study focuses on the impact on learning outcomes, hence an evaluation of the effectiveness of the provided hints is necessary. Effectiveness in this context assesses whether the players' puzzle-solving behaviours are influenced by the generated hints. Solving behaviours considered include whether players interact with the part of the grid directed by the hint or whether they apply the strategy mentioned in the hint.

To achieve this, an evaluation functionality was developed to visualise the saved game-play of the participants in the experiment. This visualiser allows human evaluators to observe puzzle-solving behaviours of all participants across all levels in the game. The visualiser system and logistics are described in Figure 4.13.

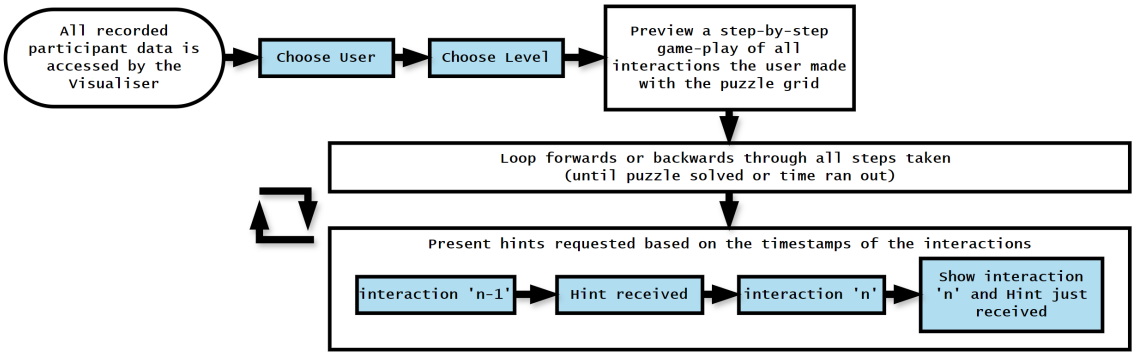


Figure 4.13: Benchmark Visualiser Loop for previewing all recorded steps participants have done throughout all levels of the experiment. The Visualiser allows evaluators to preview the puzzle-solving behaviours of the participants when receiving hints.

With such observations, evaluators assess whether the player has implemented the tips the hint provided in their next subsequent steps. If the player adapts their solving process according to the hint, then the hint is labelled as *effective*. However, if the player ignores the hint and continues working on a different area of the grid than the recommended row or column or they have already done the step before the hint was said out loud, then the hint is marked as *noneffective*. By respecting this strategy, the effectiveness measured on the data-points gathered throughout the study's experiment shows a score of 78.1% effective hints (out of a total of 155 hints). Values for each of the levels - except for the Levels 2 and 6 which did not provide hints - are shown in Table 4.4. A major factor impacting those scores was the difficulty of the levels. Harder levels posed greater challenges for users in correcting their grids to match the received hints, leading to confusion when trying to follow the hints. Comparably, in easier levels, particularly the tutorial Level 1, users were not fully accustomed to the game, the rules of Nonograms, and the hint formats, resulting in uncertainty about how to implement the received hints effectively.

	Level 1	**Level 3**	**Level 4**	**Level 5**	**Overall**
Hint Effectiveness (%)	83.3	86.4	81.3	68.9	78.1

Table 4.4: The effectiveness of the Hints generated for the participants throughout each level (except for the Levels 2 and 6 which did not provide hints)

Benchmark Future work:

As mentioned in the labelling process for evaluating the accuracy of the hints, the labelling process for the hint effectiveness can also be further improved by assigning multiple evaluators to assess the data with an overlap, thus ensuring inter-coder reliability [66]. Additionally, the visualisation functionality could be enhanced to allow for easier tracking of puzzle-solving behavioural trends. This could include integrating an automatic tracker that could analyse and unwrap the hint to identify the referenced area for the next best steps and detect whether following user interactions occur in that area of the grid.

4.3.5 Production Costs

The production costs for the system were carefully managed to ensure efficiency and affordability. Below is an overview of the cost structure:

- **Free API Calls for Puzzle Meaning Check:** The Puzzle Meaning Check feature utilises the HuggingFace Inference API for the Text Generation task (inferencing Microsoft's Phi 3 model). This feature offers free API calls for requests generating a maximum token number of 250. [67] However, due to the high demand for the servers globally, HuggingFace servers might return a server overload error. This is handled in the game as a warning animation that prompts the users to try again.
- **Pay-As-You-Go Serverless API Calls:** The system leverages Azure ML cloud services for the Llama 3 model, which operates on a pay-as-you-go subscription. Costs are incurred only based on the usage of the API calls (per tokens of requests and responses), ensuring that expenses limited to only the amount of usage. This is a cost-effective model that eliminates expenses when the LLM is not used or any up-front payments.
- **Local Game Data Storage:** The game itself runs locally on the player's device, and all game data is stored locally as well. This approach minimises reliance on external servers for storing data, reducing operational costs and potential security risks associated with data transfers.

By utilising free API calls where available and adopting a pay-as-you-go model for cloud services, the production costs are optimised to deliver a cost-effective solution for maintaining the system throughout the human trial of the study.

4.3.6 Autonomy of the System

The autonomy of the system refers to its ability to function independently, which in this study's context means being accessible to participants without the need for direct supervision.

Future work could involve making the game publicly available by publishing it or hosting it on a cloud server. This would enable broader testing that is not restricted by the supervisor's specific location or schedule. Hence, the experimentation stage could be opened to a larger pool of participants that may not be able to physically attend the study's experiments. By making the system accessible through a public domain name system (DNS), users could remotely access and interact with the game from anywhere. However, this would require setting up a secure virtual private network (VPN) for data gathering to ensure the privacy and security of participant data collected globally. The current system is capable of recording data from multiple users simultaneously, as it tracks requests by user ID. Therefore, the scalability of the current system allows for such future expansions in the experimentation procedure.

Chapter 5

Experiment

This chapter will formulate the experiment for testing the developed system and measuring its impact onto the participant's learning process. Section 5.1 provides an overview of the variables measured throughout the experiment and their role for this study. Section 5.2 describes the main lessons learned from the MVP and Trial tests throughout the development of the game and backend sever and the feature changes implemented henceforth. Following, the section 5.3 describes the overall experiment methodology including the overall participant's experience, details on the participant pool, and the main features of the experiment. Finally, the chapter wraps up with the evaluation section 5.4 that presents the results of the human trial and analysis of the data.

The project proposes the use of an AI assistance model (leveraging Large Language Models) to guide players during the solving process of Nonogram puzzles by tailoring hints and feedback according to their current progress, the specific spatial area they are focusing on, and the game's set of rules. Hence, the objective is to determine whether personalised AI assistance leads to improved learning outcomes compared to conventional puzzle-solving methods without personalised assistance. Therefore, this study requires a between-group experiment, during which the tailored and untailored AI generated hints can be tested on different control groups.

The formulated hypotheses for this study are:

Hypothesis 1: *By utilising a Large Language Model to create hints and feedback customised to users' progress and spatial reasoning in a 2D grid puzzle game, we expect to see* ***faster completion times*** *compared to conventional learning methods without personalised assistance.*

Hypothesis 2: *By utilising a Large Language Model to create hints and feedback customised to users' progress and spatial reasoning in a 2D grid puzzle game, we expect to see* ***higher solving accuracy****, compared to conventional learning methods without personalised assistance.*

Hypothesis 3: *When learning how to solve a new puzzle game, people* ***prefer tailored hints*** *based on their progress.*

This study's between-group human-computer interaction experiment compares the two dividing control conditions:

I. Tailored AI Assistant provides hints generated by a Tailored AI-Assistance System which tracks live user progress and spatial reasoning while solving a Nonogram puzzle. The algorithm for generating the knowledge base for the tailored process and the hint generation are explained in section 4.2.4 and 4.2.10.

II. Untailored AI Assistant provides hints generated by an Untailored AI-Assistance System, which is not aware of the user's progress, but provides randomised correct hints from within the solving process of a Nonogram puzzle. The untailored hints are generated by a randomised system that solves the blank Nonogram and returns a hint based on one of the actions it took at a random point in the solving process. Therefore, the untailored hint is correct but not relevant to the player, as it might refer to a step that the player has not yet reached or has already completed. The randomised algorithm is explained in section 4.2.10.

5.1 Experiment Variables

The variables of interest in this study are divided into independent and dependent variables as outlined below.

Independent Variables: The primary independent variable is the dividing condition between the participating groups of the experiment, which is the type of AI assistance provided. The effects of tailored AI assistance, which is customised to the user's current progress and spatial reasoning, will be compared against untailored AI assistance, which does not adapt to the user's individual skills.

As previously explained in section 4.10, the untailored hints are based on information provided by a randomised Nonogram solver. This solver functionality returns all recommended steps for a row or column at a random point during the solving process. In other words, the algorithm solves the nonogram from scratch and returns one of its line-wide recommendations at a random point throughout its puzzle completion process. This signifies that the untailored hint provided to the user is correct, however it refers to a wrong moment in time throughout the solving process of the Nonogram. Therefore, the participant might receive hints regarding steps they have previously done, or hints about steps that are far in the future for the player, so the recommended definite cells mentioned in the hint are not yet certain to the player. Hence such hints induce a sense of confusion to the player as the information shared is correct but it is not yet useful for them.

Dependent Variables: To determine the effectiveness of the personalised AI assistance in enhancing learning outcomes, problem-solving skills, and overall performance in a 2D grid puzzle game setting, dependent variables focused on learning characteristics are considered.

- **Efficiency:** This variable measures how quickly a task can be completed. It assesses whether users receiving tailored AI assistance gain more proficiency and therefore complete the puzzles faster compared to those receiving untailored assistance.
- **Accuracy:** This variable tracks how many errors are made throughout the task. It evaluates whether the tailored AI assistance helps users learn the puzzle solving-strategies more effectively and therefore make fewer errors compared to untailored assistance.
- **Usefulness:** This variable measures the participants' views on the usefulness and helpfulness of the received hints throughout the solving process of the Nonograms. It evaluates whether the tailored hints are preferred over the untailored hints.

Such evaluations are done by comparing the two control groups to evaluate the performance of participants throughout the game and their perceived user experience on the tested AI assistant.

5.2 Trial Testing

Several trial tests were conducted with two experienced and one novice players, aiming to evaluate the system's functionalities throughout the development process. For example tests were done on the game interface and the user-friendliness of the design, but they also helped gathering perceptions and opinions on the hints, and developing further ideas for creating the hint hierarchy.

One example of experience earned from these tests was the introduction of the click-and-drag motion, as explained in Figure 4.5. This interaction method was expected by the majority of the experienced players due to its intuitive use and effectiveness in game-play. Therefore, the functionality was integrated in the system.

Another example involved a trial with novice players on a series of puzzles to understand what kinds of hints they expected and needed throughout the puzzle completion process. This trial helped determine the hierarchy of hints (described in section 4.2.8) and ensure that the system provided useful guidance at different stages of puzzle-solving.

Additionally, the integration of the nonogram solver's logic into the hint knowledge base was also the outcome of the feedback from a trial test. The approach of finding definite cells by comparing all possible placements of groups of filled cells was highlighted by experienced trial participants as an effective method for solving Nonograms. More details on the solver's approach and its integration into the hints can be found in section 4.2.10.

5.3 Experiment Methodology

In order to set up the experiment stage, the study went through the sufficient approvals required by the Research Governance and Integrity Team from the Ethics Department at Imperial College London. The trial required participants to volunteer in a 1 hour long puzzle-solving task. The overall trial happened over a period of approximately 3 weeks. Experiments have been done is a silent room, with minimal to no distractions for the participants. The supervisor introduced the participant to the task through an induction stage, however, the participant was left alone to concentrate on the task until the end of the game. The following sections present the overall participants' experience.

5.3.1 The task

The participant needs to complete six Nonogram levels, increasing in difficulty as the game progresses. Figure 5.1 shows the flow of the game for every participant. After they log in, they can select the level to complete. The first level is a tutorial introducing the player to the game rules and interface and is not used for any analytics. The other levels are the sources of data for this study. The second and last levels provide no hints in order to test the user's puzzle-solving skills towards the beginning and end of the game. Whereas the rest of the levels allow participants to use unlimited amount of hints to help them complete the levels in the allocated 8 minutes per level.

The goal of a level is to successfully complete the Nonogram puzzle and to correctly guess what the meaning of the revealed image is. To complete the puzzle, the participants need to correctly fill all grid cells by following the numerical clues on the rows and columns. Thus the level is considered complete if the participant passes both requirements.

If the participant cannot pass both checks in the allocated time, they will be locked out of the level and receive a warning message: *"You ran out of time on the previous level! Don't worry, you can continue to the next level: i."*

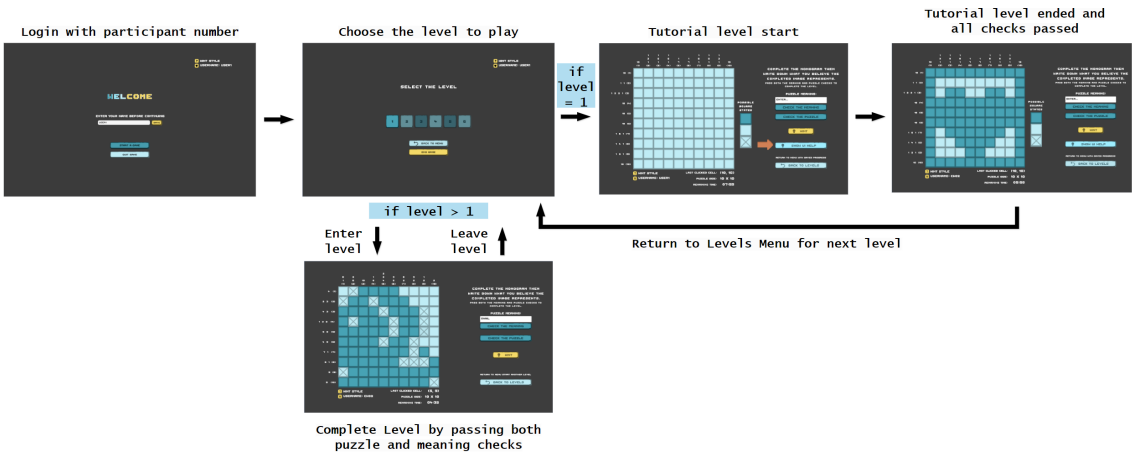


Figure 5.1: Flow of the experiment task that every participant underwent

As previously mentioned, the levels increase in difficulty as the game progresses, except for the Pre & Post test levels which are of the same difficulty (detailed in section 5.3.4). Images of all completed levels can be found in the Appendix A. The levels form the game difficulty structure in table 5.1 below:

Level	1 (Tutorial)	2 & 6 (Pre & Post Tests)	3	4	5
Difficulty	easy	easy-medium	easy	medium	hard
Level Meaning	heart	car	snail	mouse	rooster

Table 5.1: Difficulty and Meanings of the levels forming the structure of the game

5.3.2 Participants' experience

Participants were welcomed and briefed on the experiment's purpose, emphasising that their involvement would help determine the system's helpfulness in assisting them while learning how to solve Nonogram puzzles.

Experiment Overview: Participants were told that they would work through six Nonogram puzzles, each with an 8-minute time limit. They were provided with a simple definition of Nonograms and showed a picture of a completed one. Then they were told that the game system would help them progress throughout the game by providing audio hints and feedback, with transcripts also available on the screen. Participants were encouraged to request hints anytime needed, and were told that the system would also offer periodic hints automatically.

Pre-Game Stage: Participants were first asked to sign a consent form (found in Appendix F), permitting the study to use their data for further analysis. They were reminded that they could withdraw at any point, with all their data removed if necessary. Then, the volunteers are asked to fill out a pre-experiment form about their previous experiences with Nonograms and AI-assisted learning. More details on the questionnaire is found in Section 5.3.5.

Tutorial and Induction to the System: Participants that consider themselves novice Nonogram players were assured that a tutorial level was provided to explain the basic rules, game interface, and interactions they could make with the game. Additionally, assistance from the supervisor was available during this tutorial to confirm the interactions and goal of the game. However, no further contacts between the supervisor and the participant were to be made after the tutorial level until the end of the game. The supervisor confirmed that no recordings were to be made; only in-game data such as clicks and completion times would be collected throughout the experiment.

Game-Play: Participants logged in using a provided participant number and began with a tutorial level. They were accompanied by the supervisor throughout the tutorial level and were reassured that it did not count towards the final analysis. Reminders about the level's goal, timer per level, and hint availability were given at the end of the tutorial. Afterwards, the supervisor left the room and allowed the participants to continue with the game by themselves. The levels unlocked progressively as participants advanced throughout the game.

Post-Game Stage: Upon completing all the levels, participants would press the <End Game> button which would announce for them to call for the supervisor to return inside the room. The participants then filled out a post-experiment form about their experience with the AI Assistance System. More details on the questionnaire is found in Section 5.3.5. Finally, the participants were thanked for their contribution, and the experiment concluded with well-wishes for their day.

5.3.3 Participants

20 participants were recruited of which 15 were male, and 5 were female, most within the 18-25 age group. Most of the participants were students from a local university. Participants were randomly distributed between conditions: 10 participants (8 male, 2 female) were in the tailored condition, and 10 participants (7 male, 3 female) were in the untailored condition. 18 out of 20 participants reported having no prior experience with Nonograms or considered themselves beginners at the game, however 13 out of 20 participants mentioned occasionally or frequently playing puzzle, indicating a developed strategic puzzle-solving mindset.

Participants were mainly recruited through the word of mouth. Public notice messages were also sent out to students in multiple departments and posters were put up in public areas around the Imperial College South Kensington campus. The poster made as primary marketing material for gathering participants can be found in the Appendix E.

5.3.4 Pre & Post Test Levels

Levels 2 & 6 compose the defined Pre & Post Tests in the experiment. For these two levels, an identical easy-medium difficulty Nonogram is chosen, with the distinction that the latter level applies a linear transformation (a clockwise rotation of 90 degrees) to the puzzle grid. Figure 5.2 shows the differences between the two levels.

Using the same Nonogram puzzle allows for testing of the participant's skills at the beginning and end of the overall experiment process, enabling a comparison of their improvement in problem-solving skills. Additionally, these levels do not provide any hints (either self-initiated or automatic) to limit any discrepancies participants might experience with the same level, thus, measuring their true puzzle-solving skills at the two different points in time.

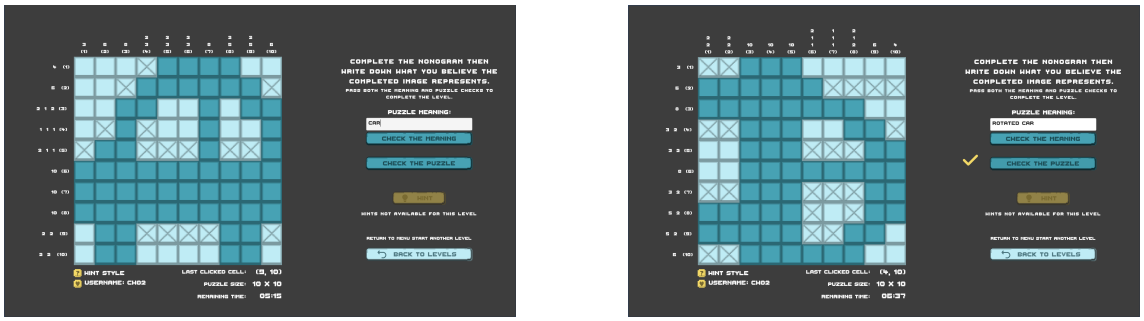


Figure 5.2: Pre & Post Test Levels: Level 2 *car* and Level 6 *invertedcar*

5.3.5 Questionnaires

There are two surveys the participants have to complete at the beginning and end of the experiment. The pre-questionnaire was developed to gather information regarding the participants' demographics and their prior experiences relevant to the study, such as their familiarity with Nonograms or other logic puzzles and AI-assisted learning. This survey is done before the participants play the game. Whereas, the post-questionnaire asks participants for their opinions on the AI-Assisted Puzzle Solving System after completing the experiment. The full questionnaires can be found in Appendix D.

The Pre-Experiment Questionnaire is completed by the participants right after the supervisor provides an overview of the experiment's procedure. At this point, participants have not seen the developed game UI nor been briefed on the Nonograms' basic rules or the goal of the game. This form collects participants' demographic information, such as age and gender, and inquires about their previous experiences with logic puzzles, Nonogram puzzles, and AI-assisted learning. At the end of the form, the participants are also given the option to describe their expectations of the system and the types of features and help they expect to receive from the AI Assistant.

The Post-Experiment Questionnaire is completed after participants have finished all levels within the game. The survey requests their ratings on the experience with the system. It requests participants to rate the system's functionalities on a Likert scale from 1 (Not at all) to 5 (Completely). Some questions include: how intuitive they found the AI Assistant, how useful and accurate the hints received were, how much they believe their puzzle solving skills have improved or how distracting or misleading the assistant was. Participants are also asked whether they encountered any errors and how much they still trusted the AI Assistant after encountering such erroneous situations. The survey ends with an open-ended question inviting participants to suggest any improvements to the system. This allows them to provide detailed feedback on their specific experience with the tested AI Assistant.

The results from the questionnaires and notable comments are discussed in section 5.4.3.

5.4 Experiment Evaluation

The evaluation of the study's hypotheses includes the collection and analysis of both in-game data and participants' responses from pre- & post-questionnaires. The following sections describe the methods used to evaluate each condition, the data collected, and outline the measurement criterion utilised to reject the study's null hypotheses.

5.4.1 Evaluating Puzzle Solving Behaviours

As the study follows a between-group design, two groups of participants took part in the experiment, each testing one of the control conditions (Tailored and Untailored AI Assistance). This section presents the evaluation of the study's hypotheses on the puzzle solving task and how the participants performed depending on the control group they were part of.

Evaluating Pre & Post Test Levels

This study aims to evaluate how the two types of AI Assistance impact the users' efficiency and accuracy in a puzzle-solving task. Therefore, for efficiency, the analysis compares the duration of puzzle-completion and, for accuracy, the evaluation considers the number of mistakes made throughout the puzzle solving process. This evaluation compares the data from the two Pre & Post test levels in order to observe the improvements participants of the two groups have gained. The Pre & Post levels were previously explained in section 5.3.4.

I. Puzzle Completion Duration: By comparing the duration taken by participants to complete the Pre & Post test levels in each condition group, it can be observed that, on average, the Tailored group has a shorter completion time for the Post level compared to the Untailored group. Figure 5.3 illustrates the participant data distributions for each of the Pre & Post test levels and control conditions. Overall, the Tailored group shows a more significant decrease in average completion times (`X` on the box chart). It is important to note that regardless of the level, participants have a maximum of 8 minutes (480 seconds) to complete the level. If they run out of time, the level is considered failed, and the maximum time recorded is 480 seconds. Therefore, it seems that some participants under both conditions still failed to complete either of the levels.

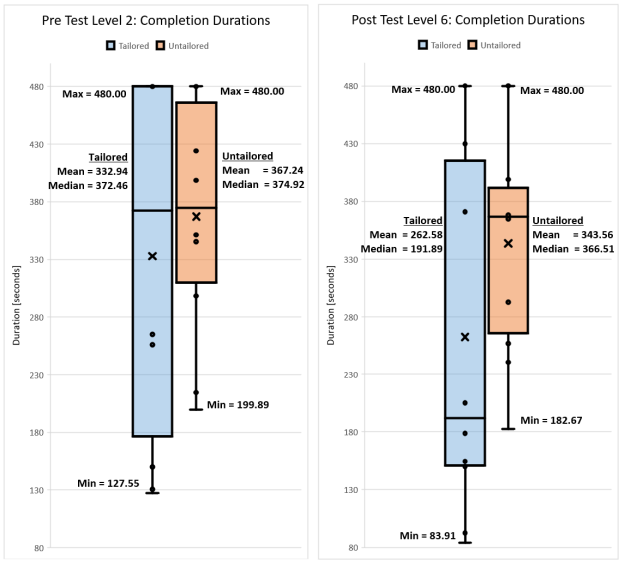
To observe the effects of AI Assistance under the two conditions, the evaluation considers the completion times for both Pre & Post test levels for each participant and analyses the average improvements, as shown in figure 5.4. The box and whiskers chart presents the distribution of improvements under the two control conditions. It is observed that the Tailored group presents lower mean and median completion times. However, the difference between its first (Q1) and third (Q3) quartiles is larger than that for the Untailored group, which could signify a larger variation in skill levels among the participants, resulting in a distribution with higher standard deviation. Therefore, the participants selected for the Untailored group seems to have had a more similar skill level before the experiment. However, their improvement rate varies much more than for the tailored group, which could signify that the improvement from the participants of the untailored group might be an outcome of external variables independent of this study, such as personal skills or experience.

Regardless of the distribution spread, the Tailored group shows an average improvement of M = 70.36 seconds, whereas the Untailored group shows an average improvement of M = 23.68 seconds. These results are also shown in Table 5.2.

Level	**Mean duration [s]**		**Standard Deviation [s]**	
	Tailored	Untailored	Tailored	Untailored
Pre Test (2)	332.94	367.24	±153.37	±99.57
Post Test (6)	262.58	343.56	±151.63	±94.14
Average Improvement [s]	70.36	23.68	-	-

Table 5.2: Average Duration and Range for both Control Conditions for the Pre & Post Test Levels

Although the previous results show discrepancies in improvements in completion times relative to the AI assistance tested, the gathered duration data for both Pre & Post Test levels is not statistically significant based on the *p value* of the independent one-tailed t-test (pre-test: $t(18) =$



(a) Pre Test Durations (b) Post Test Durations

Figure 5.3: Puzzle Completion Times of Pre & Post Test Levels: comparing overall times per level between the Tailored and Untailored AI Assistant conditions

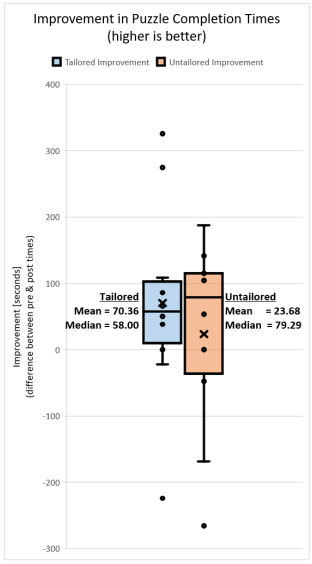


Figure 5.4: Comparing participants' improvement in puzzle completion times between the Tailored and Untailored AI Assistant conditions

-0.56 ($p = 0.58$); post-test: $t(18) = -1.36$ ($p = 0.19$)). The t-test was applied after it was confirmed that the two data groups are normally distributed and have equal variance, using the Shapiro-Wilk test and Levene's test, respectively. Therefore, there is not enough evidence to prove the ***Hypothesis 1*** and it cannot be generalised to a larger population.

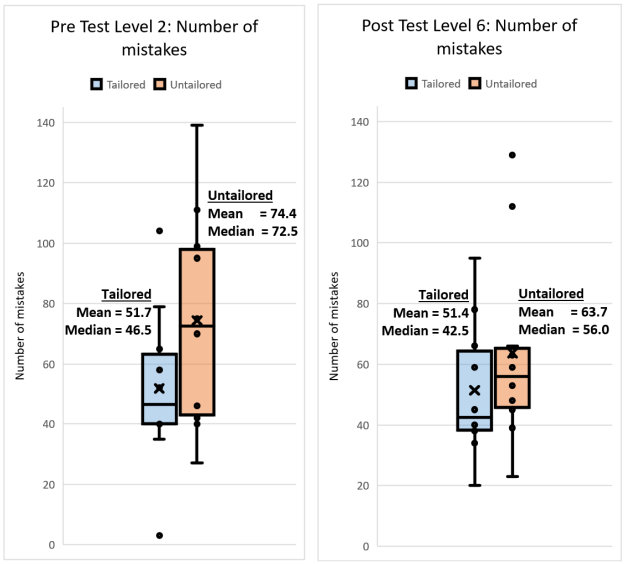
A larger participant pool or a more specialised test condition would be required for a more thorough and conclusive test. For instance, **Future work** could involve modifying the data gathering process to solve the current issue of only measuring the completion duration. The puzzle-completion duration is measured between the first and last interactions with the grid. Therefore, it is noted to consider that some participants might visually analyse the puzzle before making their first move, leading to a small expected period of time before the first interaction that could be lost due to this measurement procedure.

II. Number of Mistakes throughout the Puzzle: To observe the improvement in participants' accuracy throughout the puzzle-solving process, an analysis was made on the number of mistakes made in the Pre & Post test levels. By assessing the number of mistakes, it is possible to draw conclusions about the players' uncertainty while completing the puzzles. This is because the provided puzzles can be solved by an experienced player or the Solver algorithm without any uncertain moves, as there is at least one definite path to the solution. In other words, each step in solving the puzzles can be definitively determined, indicating that mistakes are a sign of the participant's uncertainty or confusion contradicting the numerical clues of the Nonogram.

The improvement in the average number of mistakes is surprising. Considering the results in Table 5.3, there seems to be a larger improvement in puzzle-solving accuracy for the participants in the Untailored group. However, the number of mistakes within the two levels do not prove any statistically significant difference or trend (pre-test: $t(18) = -1.57$ ($p = 0.13$); post-test: $t(18) = -0.97$ ($p = 0.34$)), therefore the results do not provide enough evidence to prove ***Hypothesis 2*** and might be the result of random noise in either the participants' initial skill level or in the data gathering procedure (issue detailed under feature work below). Despite this, it is worth considering that the participants in the tailored group might have not experienced much improvement in accuracy due to them relying on the hints' instructions and not trying to comprehend the strategy that the hints were referring to. Therefore, they did not learn the strategies to solve Nonograms and retained a consistent number of mistakes throughout the game. Such situation encourages future studies with new experiment designs focused more on the accuracy of the players.

Level	Mean Number of Mistakes		Standard Deviation	
	Tailored	Untailored	Tailored	Untailored
Pre Test (2)	51.7	74.4	25.95	34.42
Post Test (6)	51.4	63.7	21.62	30.99
Average Improvement	0.3	10.7	-	-

Table 5.3: Mean and Median for Number of Mistakes for both Control Conditions in Levels 2 & 6



(a) Pre-Test Mistakes (b) Post-Test Mistakes

Figure 5.5: Number of Mistakes through the Pre & Post Test Levels: comparing overall amounts per level between the Tailored and Untailored AI Assistant conditions

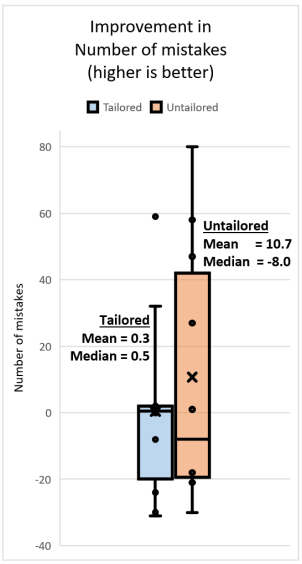


Figure 5.6: Comparing participants' improvement in making less number of mistakes between the Tailored and Untailored conditions

A larger participant pool or a more specialised test condition would be required for a more thorough and conclusive test. For instance, **future work** could involve modifying the data-gathering methodology to address the current issue of recording every interaction with the grid, regardless of the cell state. The game design allows players to click on a specific cell to change its state between three states: filled, empty, and crossed. Players use the crossed cell state as a visual aid to help them mark cells that cannot be filled while respecting the Nonogram clues. To reach a crossed state, players have to click twice on a cell. Therefore, by recording every interaction without making a distinction for crossed cells, the measured mistakes are skewed by the number of cells the users have crossed. This issue was discovered when analysing the data gathered from the experiment. A possible solution could be the implementation of a left mouse click feature for changing states between filled and empty, and the right mouse click to mark a cell as crossed. This way, only interactions resulting in changes to filled and empty cells would be recorded, and no recording would be done for crossed cells.

Evaluating overall game user progress

Throughout the experiment, the system gathers data on the progress rate achieved and the number of hints used by each participant on each level. Aside from the Pre & Post Test levels previously evaluated, the remaining levels vary in difficulty. This evaluation aims to assess the overall impact of the number of hints requested on the users' progress across levels of different difficulty. It's important to note that for these levels, players had access to both self-initiated and automatic hints.

A Pearson correlation analysis was conducted to investigate whether there was a statistically significant correlation between the number of hints used by the player and their progress achieved on the level. Surprisingly, the analysis revealed a strong negative correlation, indicating an inverse

Level	Mean Progress [%]		Mean Number of Hints		Majority Hint Class	
	Tailored	Untailored	Tailored	Untailored	Tailored	Untailored
Easy (3)	0.92±0.309	0.95±0.485	5.2	4.7	2 (41%)	2 (51%)
Medium (4)	0.98±0.191	0.94±0.317	3.6	4.6	2 (55%)	2 (54%)
Hard (5)	0.37±0.386	0.40±0.161	6.3	5.3	1 (68%)	1 (79%)

Table 5.4: Average Progress achieved by participants based on the level and average number of hints used

relationship between the number of hints requested and the progress achieved. This unexpected finding suggests that players who used more hints tended to achieve a lower progress rate on the puzzle. One possible explanation for this situation is that the time spent by the participant listening to or reading the hints reduced the time remaining available to work on the rest of the puzzle. Given the time constraints of the levels, while hints may assist users in overcoming challenges and finding mistakes, receiving hints comes at the expense of one's focus and time, potentially even preventing participants from progressing further than they could have without hints in the level.

For levels 3 & 5, the observed relationships have not proven to be statistically significant due to $p > 0.05$ for all AI assistant styles. Levels 3 & 5 might have been too easy or hard, respectively, for players to show a significant difference between the two control conditions. However, interestingly in Level 4, there was a significant correlation ($p < 0.05$) between the number of hints and progress achieved for both the Tailored and Untailored control groups. In this level the Tailored group shows a more negative correlation compared to its Untailored counterpart. Such situation could be caused by the fact that participants in the Untailored group may have realised that the hints were erroneous and gradually ignored them. In contrast, the Tailored group may have maintained their focus on the hints, leading them to spend more time listening to the hints than participants in the Untailored group. A summary of the calculated hint-progress relations and directions can be found in table 5.5.

Level	Condition	Pearson Correlation (R)	p-value
Easy (3)	Tailored	-0.5849	0.0757
	Untailored	-0.4480	0.1941
Medium (4)	Tailored	-0.8993	0.0004
	Untailored	-0.7182	0.0193
Hard (5)	Tailored	-0.4019	0.2496
	Untailored	-0.2026	0.5746

Table 5.5: Pearson Correlation results for Hint-Progress relations on different levels

Overall Puzzle Completion Behaviour

Due to the average response latency of $2.53s \pm 0.589$ in the hint generation request (described in section 4.3.2), it was observed that participants would already realise their mistake and complete the step that the hint generated would have referred to. Therefore, it was observed that participants tended to ignore hints as they were already ahead in their solving process. However, participants mentioned that the hint still helped confirm their actions, even though some mentioned that the hints were somewhat "distracting" due to their longer latencies and expressed a desire for a quicker transition to text-formatted hints to mitigate this distraction.

Additionally, throughout the MVP testing and the implementation of the Nonogram solver, it was observed that the novice trial participants first consider rows before examining the column clues. However, it is interesting to note that in the final experiment, 65% of the participants started a level by selecting cells along one of the columns. This was measured only for the levels that provided an equal good starting point with both rows and columns (such as the tutorial level, level 3 and 4). This was tracked by checking the first 5 interactions on each level for every participant and determining if the majority of these interactions were along a row or a column.

5.4.2 Evaluating Guess Puzzle Meaning Task

During the evaluation of the "guess puzzle meaning" task, it was observed that many participants were able to successfully solve the puzzle itself, but encountered difficulty when attempting to guess the meaning of the revealed image. Players of all experiences faced similar issues.

The requirement to guess the puzzle's meaning within the allocated time significantly impacted the completion duration of the levels. In the absence of hints, as seen in the Pre & Post test levels, participants were left to rely solely on their own creativity and outside-of-box intuition to decipher the image revealed by the puzzle. Therefore, it was observed that some participants spent the majority of their allocated time attempting to guess the meaning after completing the puzzle. In some situations they did not succeed in guessing correctly and ran out of time, thus failing the level. However, in the levels where puzzle meaning riddle hints were provided (3, 4 & 5), participants generally required only one hint on average to discern the image meaning successfully.

Considering this unexpected situation, the puzzle completion duration measurement has been split apart from the meaning guess duration, thus allowing to visualise the amount of time spend on each sub task of every level.

5.4.3 Evaluating User Experience and Opinions (Questionnaires)

This section provides an overview of the topics covered by the survey questions and assesses participants' responses regarding their experience learning how to solve Nonogram with the system. The questionnaires presented three main types of questions: 5-point Likert scale, multiple-choice questions and open-ended input questions. The following observations cover results and comments from both type of questions.

I. Pre-Questionnaire Observations and Results

From the ***Pre-Questionnaire***, 60% of the participants mention to prefer moderate amounts of assistance throughout puzzle solving tasks, followed by those who preferred minimal assistance, and finally, those who preferred extensive assistance. There was a relatively even split among participants regarding how often they use AI related technologies in daily life and how much they relay on them. Additionally, 60% of participants mention to moderately trust AI-based technologies to provide accurate and reliable results, while the remaining participants trust them less.

At the end of the survey, participants were also asked about their concerns regarding using an AI-based system and what they expected from the system's assistance. Half of the volunteers expressed concerns about the accuracy of the data, noting that they "use (AI assistants) as a an exploratory tool when trying to understand concepts", thus they "have to be careful in case (AI assistants) make a mistake". On the other hand, the majority of participants mentioned that they would want the system to have "customised ways of explaining the rules (of Nonograms)", "teach (them) how to solve the problem without giving me the final answer" and "provide an explanation of concepts that help move (the player) in the right direction".

Additionally, some participants provided more detailed expectations, such as wanting the system to "know about what (the player's) next action should be" and "suggest hints that are closely related to the region of the puzzle that (the player is) trying to solve". It is reassuring to note that all of these valid requests and expectations were successfully met by the functionalities of the developed system. However, there was one participant expectation that the current system functionalities did not fully meet: the ability to "help (the user) when (they are) making repetitive errors but not asking for help". The system's automatic hint generation is triggered by a pre-set time loop but the hint is customised to the user's progress. Thus, if the user might be making repetitive errors, the system will trigger a hint aware of the user's situation every few fixed intervals. Nevertheless, the system used throughout the experiments is primary designed with the focus on users requesting for hints themselves. Hence, the automatic hints were maintained simple, and the duration between the automatic hints was kept fixed.

II. Post-Questionnaire Observations and Results

From the ***Post-Questionnaire***, it is observed that 70% of the participants prefer to request hints themselves (self-initiated request), contrasting with the pre-questionnaire suggestions where players mentioned the expectation for the system to provide hints automatically when detecting that they are stuck or need help. This result supports the reasoning why much of the focus in the development process was put on the feature to self-request hints when needing and the user friendliness and easy interaction with the game UI. More discussion on the design of the system and observations is found in section 4.3.1.

Additionally, all other observations and their respective average survey scores used for comparison between the two conditions of the experiment are illustrated in Figure 5.7. Further details and discussions on the results follow the figure. The data from the post-questionnaire was tested for normality and equal variance in order to use it for statistical significance measures. The Shapiro-Wilk Test indicated that the data was not normally distributed. However, the Levene's Test confirmed that the variances were equal. Given these assumptions, the appropriate statistical test for this data is the Mann-Whitney U Test.

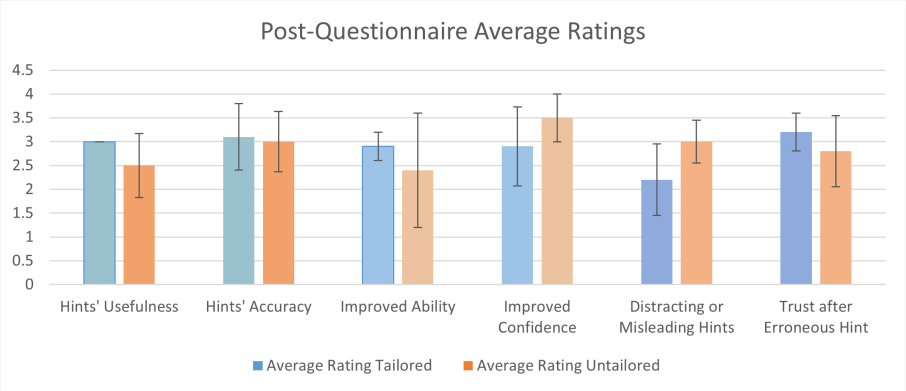


Figure 5.7: Overall preview of the mean scores on various topics of comparison between the Tailored AI Assistant (blue) and the Untailored System (orange). The related topics are differentiated by small colour contrasts.

i. Perceived Quality of AI Assistant - Hints' usefulness and accuracy:

Firstly, participants rated the usefulness of the AI assistant's hints in solving the puzzles. The results showed a significant difference between the tailored and untailored AI assistants (Mann-Whitney U Test: $U = 75.0, p = 0.027$). Since $p < 0.05$, it can be concluded that the AI assistant tailored to the user's progress was perceived as more helpful throughout the learning process of Nonogram puzzles than its untailored counterpart. Therefore, these results provide enough evidence to prove that participants preferred the tailored hints over their untailored counterpart, thus proving the ***Hypothesis 3***. This is also reflected in the average Likert scale scores, with the tailored system receiving an average score of 3 ± 0.00 and the untailored system receiving an average score of 2.5 ± 0.67.

Additionally, the participants also rated how accurate they found the hints to be in relation to their progress on the puzzle. Even tho the results are not statistically significant (Mann-Whitney U Test: U = 54.0, p-value = 0.769), the average ratings show that the tailored hints ($M = 3.1, SD = 0.70$) are perceived to be more accurate than the untailored hints ($M = 3.0, SD = 0.63$) in this study. The close proximity in the mean values could be a result of the similarities between the control groups, as even the untailored system provided accurate hints but referring to the wrong step in time. In other words, the untailored hints were correct only if the user happened to be at the specific random point in the puzzle-solving process that the hint is referring to. Therefore, the design of such distinction between the control groups might not be sufficient to demonstrate a significant difference in user's perception of hint accuracy. However, as the results do not show a significant difference, the close proximity could also be the cause of randomness in the participants' answers.

ii. Perceived Distraction and Misleading Nature of the AI Assistant:

Participants also rated how distracting or misleading they found the AI assistant. The results indicated a significant difference between the two assistants (Mann-Whitney U Test: $U = 22.0, p = 0.017$). Since $p < 0.05$, it can be concluded that the non-customized AI assistant negatively impacted the user experience and learning process by distracting and misleading the participants throughout the puzzle solving tasks. This is also supported by the average Likert scale scores, with the tailored assistant receiving an average score of 2.2 ± 0.75 and the untailored system receiving a score of 3 ± 0.45. A lower score indicates fewer distractions, suggesting that the tailored assistant was perceived to be less distracting and misleading than the untailored system.

iii. Perceived Improvement in Nonogram Puzzle Solving Skills:

Participants were asked if they felt their Nonogram puzzle-solving skills improved after using the AI assistant. The average ratings from the Likert scale were 2.9 ± 0.3 for the tailored AI assistant and 2.4 ± 1.20 for the untailored system, higher value signifying better improvement. However, the Mann-Whitney U Test showed no significant difference between the groups (U = 58.0, p = 0.50). Since $p > 0.05$, it can be concluded that the data gathered from this study is not statistically significant and cannot be generalised to a larger scale.

Additionally, participants were asked whether they felt more confident about solving Nonogram puzzles after finishing the experiment. Even though the data does not show a significant difference between the control groups, the results provided an unexpected outcome. Participants in the untailored control group scored higher ($M = 3.5, SD = 0.50$) than the players who received tailored hints ($M = 2.9, SD = 0.83$). This could be due to randomness in the participants' responses; however, it is an interesting point for discussion. The participants in the Untailored group might have realised that the hints they receive were not relevant to their progress, prompting them to take a more holistic approach to solving the puzzles on their own. This independent perspective could have boosted their confidence more than those who relied on tailored hints and mainly followed instructions.

iv. Trust in the AI Assistant After Encountering an Erroneous Hint:

Participants rated their trust in the AI assistant after encountering a supposed erroneous hint. The average ratings from the Likert scale were 3.2 ± 0.40 for the tailored system and 2.8 ± 0.75 for the untailored system. Therefore, the Tailored group indicated a more unified consensus with relatively higher trust in the AI Assistant, as indicated by its smaller standard deviation and larger average score. However, the Mann-Whitney U Test indicated no significant difference between the groups (U = 66.0, p = 0.18), so this study's conclusions cannot be generalised to a larger scale.

Chapter 6

Conclusions

This project aimed to develop a reliable, helpful and accurate LLM-based AI assistant that positively impacts the user's learning process of a novel logic puzzle. The system developed in this study successfully provides a novel, cost-effective and scalable solution to tailoring an LLM's knowledge base to the user's skills, in-game progress, and thought-process while solving Nonogram puzzles. The system takes advantage of prompt engineering techniques, without any further training done to the LLM, to develop a reliable personalised AI Assistant.

To evaluate the system, a between-group design experiment with two control conditions was designed. The experiment aimed to assess how tailoring hints and feedback could impact the learning outcomes of the participants. Therefore, the control conditions formed two groups of participants experiencing different versions of the AI Assistant. One group received help from a tailored Assistant that customised the hints according to users' progress, the specific spatial area they were focusing on, and the game's set of rules. The second group experienced an untailored assistant, which was not aware of the user's progress and generated randomised hints based on the solving process of the Nonogram. The game and assistance system developed for the experiment was described in sections 4.1 and 4.2. The experiment and its control conditions were explained in detail in section 5.1.

By assessing the participants' performances on the puzzle-solving tasks, the results showed a 92.9% **hint accuracy** in preserving relevance to the puzzle knowledge base without hallucinations. Additionally, a 78.1% **hint effectiveness** was measured to determine whether the players' puzzle-solving behaviours were influenced by the generated hints. Sections 4.3.3 and 4.3.4 described the benchmark strategy used for measuring these hint characteristics in the study's specific task of Nonogram solving.

The results of the experiment did not show statistically significant differences between the two control conditions ($p > 0.05$). However, it is worth noting that for the experiment's participant pool, the mean averages indicated the tailored assistance helping participants become more **efficient at the task**, allowing them to complete the Nonogram puzzles faster *(tailored improvement of* $M = 70.36s$ *versus untailored improvement of* $M = 23.68s$*,* p *value* > 0.05*)*. Another interesting discussion point is that participants from the tailored condition did not seem to become more **accurate at the task**; in fact, the number of mistakes made dropped more for the untailored group *(tailored improvement of* $M = 0.30$ *versus untailored improvement of* $M = 10.70$*,* p *value* > 0.05*)*. This discrepancy might suggest that the control conditions were not distinct enough, or were highly affected by the randomness in either the participants' initial skill level or in the data gathering procedure. However, it is worth considering that the participants in the tailored group might have not experienced much improvement in accuracy due to them relying on the hints' instructions and not trying to comprehend the strategy that the hints were referring to. Therefore, they did not learn the strategies to solve Nonograms and retained a consistent number of mistakes throughout the game. Such situation encourages future studies with new experiment designs focused more on the accuracy of the players.

Nonetheless, an interesting statistically significant result is that participants found the hints useful for getting unstuck and resolving confusion points while completing the levels *(tailored improvement of $M = 3.0$ versus untailored improvement of $M = 2.5$, p value < 0.05)*. Therefore, this evidence demonstrates that participants **preferred tailored hints** over the untailored hints. This preference is also reflected in the participants' opinions regarding whether they believe their puzzle-solving skills improved with the help of the AI assistant *(tailored improvement of $M = 2.9$ versus untailored improvement of $M = 2.4$, p value > 0.05)*. Despite this, it is interesting that the untailored group provided a higher mean Likert score when asked if they were more confident in their Nonogram puzzle-solving skills after using the AI assistant *(tailored improvement of $M = 2.9$ versus untailored improvement of $M = 3.5s$, p value > 0.05)*. This might suggest that participants in the untailored group may have realised that the hints were not relevant to their progress, prompting them to take a more holistic approach to solving the Nonograms on their own. This independent perspective could have boosted their confidence more than those who relied on tailored hints and mainly relied on the instructions without reconsidering the strategy that the hints were referencing.

In conclusion, the study shows that the user-tailored AI assistance enhances the puzzle-solving experience as participants demonstrated to prefer the tailored hints. Despite this, there was not enough statistically significant evidence to conclude whether the AI assistant positively impacts the solving efficiency and accuracy of people using the system. Although, the distinction between experience and learning observed through this study is worth highlighting: while tailored hints help users navigate the puzzles more efficiently, they do not necessarily enhance the understanding of optimal strategies. The lack of improvement in accuracy suggests that participants may be relying on the hints to complete the tasks without comprehending the underlying logic needed to solve Nonograms independently.

Bibliography

[1] Meta. Introducing Meta Llama 3: The most capable openly available LLM to date;. [Accessed 19-04-2024]. Available from: https://ai.meta.com/blog/meta-llama-3/.

[2] OpenAI. ChatGPT;. [Accessed 20-11-2023]. Available from: https://chat.openai.com/.

[3] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al.. Attention Is All You Need; 2023. Available from: https://doi.org/10.48550/arXiv.1706.03762.

[4] Alammar J. The Illustrated Transformer; 2018. [Accessed 13-01-2024]. Available from: https://jalammar.github.io/illustrated-transformer/.

[5] Google. Introduction to Large Language Models |Machine Learning | Google for Developers; 2023. [Accessed 08-01-2024]. Available from: https://developers.google.com/machine-learning/resources/intro-llms.

[6] Rajpurkar P, Zhang J, Lopyrev K, Liang P. SQuAD: 100,000+ Questions for Machine Comprehension of Text; 2016. Available from: https://doi.org/10.48550/arXiv.1606.05250.

[7] Mukherjee S, Mitra A, Jawahar G, Agarwal S, Palangi H, Awadallah A. Orca: Progressive Learning from Complex Explanation Traces of GPT-4; 2023. Available from: https://doi.org/10.48550/arXiv.2306.02707.

[8] Naveed H, Khan AU, Qiu S, Saqib M, Anwar S, Usman M, et al.. A Comprehensive Overview of Large Language Models; 2023. Available from: https://doi.org/10.48550/arXiv.2307.06435.

[9] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, et al.. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer; 2023. Available from: https://doi.org/10.48550/arXiv.1910.10683.

[10] Chung HW, Hou L, Longpre S, Zoph B, Tay Y, Fedus W, et al.. Scaling Instruction-Finetuned Language Models; 2022. Available from: https://doi.org/10.48550/arXiv.2210.11416.

[11] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al.. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks; 2021. Available from: https://doi.org/10.48550/arXiv.2005.11401.

[12] Saravia E. Prompt Engineering Guide; 2022. [Accessed 20-01-2023]. Available from: https://github.com/dair-ai/Prompt-Engineering-Guide.

[13] Savvov S. All you need to know to Develop using Large Language Models; 2023. [Accessed 20-11-2023]. Available from: https://towardsdatascience.com/all-you-need-to-know-to-develop-using-large-language-models-5c45708156bc.

[14] Li C, Wang J, Zhang Y, Zhu K, Hou W, Lian J, et al.. Large Language Models Understand and Can be Enhanced by Emotional Stimuli; 2023. Available from: https://doi.org/10.48550/arXiv.2307.11760.

[15] Kojima T, Gu SS, Reid M, Matsuo Y, Iwasawa Y. Large Language Models are Zero-Shot Reasoners; 2023. Available from: https://doi.org/10.48550/arXiv.2205.11916.

[16] pareto_investor. Trick ChatGPT to say its secret prompt; 2023. [Accessed 05-01-2024]. Available from: https://medium.com/@pareto_investor/trick-chatgpt-to-say-its-secret-prompt-973990a27b11.

[17] Hirschman L, Gaizauskas R. Natural language question answering: the view from here. Natural Language Engineering. 2001;7(4):275–300. Available from: http://dx.doi.org/10.1017/S1351324901002807.

[18] Karpukhin V, Oğuz B, Min S, Lewis P, Wu L, Edunov S, et al.. Dense Passage Retrieval for Open-Domain Question Answering; 2020. Available from: https://doi.org/10.48550/arXiv.2004.04906.

[19] Zhu F, Lei W, Wang C, Zheng J, Poria S, Chua TS. Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering; 2021. Available from: https://doi.org/10.48550/arXiv.2101.00774.

[20] Amazon. Amazon Q | Overview;. [Accessed 02-12-2023]. Available from: https://aws.amazon.com/q/.

[21] GitHub. Copilot Chat;. [Accessed 20-11-2023]. Available from: https://docs.github.com/en/copilot/github-copilot-chat.

[22] Ji Z, Lee N, Frieske R, Yu T, Su D, Xu Y, et al. Survey of Hallucination in Natural Language Generation. ACM Comput Surv. 2023 Mar;55(12). Available from: https://doi.org/10.1145/3571730.

[23] Hayward WG, Tarr MJ. Spatial language and spatial representation. Cognition. 1995;55(1):39-84. Available from: https://www.sciencedirect.com/science/article/pii/001002779400643Y.

[24] Mirzaee R, Faghihi HR, Ning Q, Kordjmashidi P. SpartQA: : A Textual Question Answering Benchmark for Spatial Reasoning; 2021. Available from: https://doi.org/10.48550/arXiv.2104.05832.

[25] Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, et al.. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models; 2023. Available from: https://doi.org/10.48550/arXiv.2201.11903.

[26] Li F, Hogg DC, Cohn AG. Advancing Spatial Reasoning in Large Language Models: An In-Depth Evaluation and Enhancement Using the StepGame Benchmark. Proceedings of the AAAI Conference on Artificial Intelligence. 2024 Mar;38(17):18500-7. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/29811.

[27] Shi Z, Zhang Q, Lipani A. StepGame: A New Benchmark for Robust Multi-Hop Spatial Reasoning in Texts. 2022 Jun;36:11321-9. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/21383.

[28] Wu W, Mao S, Zhang Y, Xia Y, Dong L, Cui L, et al.. Mind's Eye of LLMs: Visualization-of-Thought Elicits Spatial Reasoning in Large Language Models; 2024. Available from: https://doi.org/10.48550/arXiv.2404.03622.

[29] Sharma M. Exploring and Improving the Spatial Reasoning Abilities of Large Language Models; 2023. Available from: https://doi.org/10.48550/arXiv.2312.01054.

[30] Unity. Unity Engine;. [Accessed 27-12-2023]. Available from: https://en.wikipedia.org/wiki/Unity_(game_engine).

[31] Mozilla. An overview of HTTP - HTTP | MDN;. [Accessed 28-12-2023]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTTP.

[32] Hager GD, Drobnis AW, Fang F, Ghani R, Greenwald A, Lyons T, et al. Artificial Intelligence for Social Good. CoRR. 2019;abs/1901.05406. Available from: http://arxiv.org/abs/1901.05406.

[33] Roberts D, Baker DS, Walker DD. Can We Learn Together?: Co-Creating with Consumers; 2005. Available from: https://doi.org/10.1177/147078530504700401.

[34] Wang D, Yang Q, Abdul A, Lim BY. Designing Theory-Driven User-Centric Explainable AI. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. CHI '19. New York, NY, USA: Association for Computing Machinery; 2019. p. 1–15. Available from: https://doi.org/10.1145/3290605.3300831.

[35] Barredo Arrieta A, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion. 2020;58:82-115. Available from: https://www.sciencedirect.com/science/article/pii/S1566253519308103.

[36] Zhao H, Chen H, Yang F, Liu N, Deng H, Cai H, et al.. Explainability for Large Language Models: A Survey; 2023. Available from: https://doi.org/10.48550/arXiv.2309.01029.

[37] Prather J, Denny P, Leinonen J, Becker BA, Albluwi I, Craig M, et al.. The Robots are Here: Navigating the Generative AI Revolution in Computing Education; 2023. Available from: https://doi.org/10.48550/arXiv.2310.00658.

[38] Nicol D. From monologue to dialogue: Improving written feedback processes in mass higher education. Assessment & Evaluation in Higher Education. 2010 Aug;35(5):501-17. Available from: https://doi.org/10.1080/02602931003786559.

[39] Hattie J, Timperley H. The Power of Feedback; 2007. Available from: https://doi.org/10.3102/003465430298487.

[40] Khan Academy. Khanmigo; 2023. [Accessed 17-01-2024]. Available from: https://www.khanacademy.org/khan-labs#khanmigo.

[41] Savolainen A. Sal Khan: "I see AI as an additional tool, but a very powerful one" [Journal Article]. The UNESCO Courier. 2023;2023(4):12-4. Available from: https://www.un-ilibrary.org/content/journals/22202293/2023/4/4.

[42] TutorOcean. AITutor; 2023. [Accessed 17-01-2024]. Available from: https://www.tutorocean.com/ai.

[43] Spatharioti SE, Rothschild DM, Goldstein DG, Hofman JM. Comparing Traditional and LLM-based Search for Consumer Choice: A Randomized Experiment; 2023. Available from: https://doi.org/10.48550/arXiv.2307.03744.

[44] Li PH, Lee HY, Cheng YP, Starčič AI, Huang YM. Solving the Self-regulated Learning Problem: Exploring the Performance of ChatGPT in Mathematics. In: Huang YM, Rocha T, editors. Innovative Technologies and Learning. Cham: Springer Nature Switzerland; 2023. p. 77-86. Available from: https://doi.org/10.1007/978-3-031-40113-8_8.

[45] Wang FY, Yang J, Wang X, Li J, Han QL. Chat with ChatGPT on Industry 5.0: Learning and Decision-Making for Intelligent Industries. IEEE/CAA Journal of Automatica Sinica. 2023;10(4):831-4. Available from: http://dx.doi.org/10.1109/JAS.2023.123552.

[46] Imperial College London. Academic Misconduct Policy and Procedure; 2023. [Accessed 08-01-2024]. Available from: https://www.imperial.ac.uk/media/imperial-college/administration-and-support-services/registry/academic-governance/public/academic-policy/academic-integrity/Academic-Misconduct-Policy-and-Procedure-v1.3-15.03.23.pdf.

[47] Russell Group. Principles on the use of generative AI tools in education; 2023. [Accessed 08-01-2024]. Available from: https://russellgroup.ac.uk/media/6137/rg_ai_principles-final.pdf.

[48] Imperial College London. AI Tools in Teaching & Assessment - Guidance; 2023. [Accessed 08-01-2024]. Available from: https://www.imperial.ac.uk/media/imperial-college/about/leadership-and-strategy/vp-education/AI-Tools-in-Teaching-&-Assessment---Guidance.pdf.

[49] Pallets. Flask's documentation;. [Accessed 08-01-2023]. Available from: https://flask.palletsprojects.com/en/3.0.x/.

[50] Abdin M, Jacobs SA, Awan AA, Aneja J, Awadallah A, Awadalla H, et al.. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone; 2024. Available from: https://doi.org/10.48550/arXiv.2404.14219.

[51] HuggingFace. Microsoft Phi-3-mini-4k-instruct Inference API;. [Accessed 24-04-2024]. Available from: https://huggingface.co/microsoft/Phi-3-mini-4k-instruct.

[52] Microsoft. Text to speech | Azure AI Services;. [Accessed 19-04-2024]. Available from: https://azure.microsoft.com/en-gb/products/ai-services/text-to-speech#overview.

[53] Spillers F. Progressive Disclosure | Interaction Design Foundation - IxDF; 2015. [Accessed 20-03-2023]. Available from: https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/progressive-disclosure.

[54] Microsoft | Learn. How to deploy Meta Llama models with Azure Machine Learning studio;. [Accessed 19-04-2024]. Available from: https://learn.microsoft.com/en-us/azure/machine-learning/how-to-deploy-models-llama?view=azureml-api-2&tabs=llama-three.

[55] Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, et al.. Llama 2: Open Foundation and Fine-Tuned Chat Models; 2023. Available from: https://doi.org/10.48550/arXiv.2307.09288.

[56] Microsoft | Learn. Enhance security with the principle of least privilege;. [Accessed 12-02-2024]. Available from: https://learn.microsoft.com/en-us/entra/identity-platform/secure-least-privileged-access.

[57] LangChain. LLMs | Azure ML Online Endpoint; 2024. [Accessed 12-02-2024]. Available from: https://python.langchain.com/v0.1/docs/integrations/llms/azure_ml/.

[58] de Harder H. Solving Nonograms with 120 Lines of Code; 2022. [Accessed 11-03-2024]. Available from: https://towardsdatascience.com/solving-nonograms-with-120-lines-of-code-a7c6e0f627e4.

[59] Chugunnyy KA. Nonograms | Japanese crosswords;. [Accessed 08-01-2023]. Available from: https://www.nonograms.org/.

[60] Cohn AG, Hernandez-Orallo J. Dialectical language model evaluation: An initial appraisal of the commonsense spatial reasoning abilities of LLMs; 2023. Available from: https://doi.org/10.48550/arXiv.2304.11164.

[61] Doshi-Velez F, Glassman EL, Paulson JA. Contextual Evaluation of AI: a New Gold Standard; 2023. Available from: https://api.semanticscholar.org/CorpusID:266234619.

[62] Analysis A. Llama 3 Instruct (8B): Quality, Performance & Price Analysis; 2024. [Accessed 07-06-2024]. Available from: https://artificialanalysis.ai/models/llama-3-instruct-8b.

[63] Yuxiang Wu AKYFLREG Zhengyao Jiang, Rocktäschel T. ChatArena: Multi-Agent Language Game Environments for Large Language Models. GitHub; 2023. https://github.com/chatarena/chatarena.

[64] Hendrycks D, Burns C, Basart S, Zou A, Mazeika M, Song D, et al.. Measuring Massive Multitask Language Understanding; 2021. Available from: https://doi.org/10.48550/arXiv.2009.03300.

[65] Analysis A. Llama 3 Instruct (8B): API Provider Benchmarking & Analysis; 2024. [Accessed 07-06-2024]. Available from: https://artificialanalysis.ai/models/llama-3-instruct-8b/providers.

[66] O'Connor C, Joffe H. Intercoder Reliability in Qualitative Research: Debates and Practical Guidelines. International Journal of Qualitative Methods. 2020;19:1609406919899220. Available from: https://doi.org/10.1177/1609406919899220.

[67] HuggingFace. Serverless Inference API | Detailed parameters; 2024. [Accessed 01-05-2024]. Available from: https://huggingface.co/docs/api-inference/en/detailed_parameters#text-generation-task.

Appendix A

Unity Game Screens

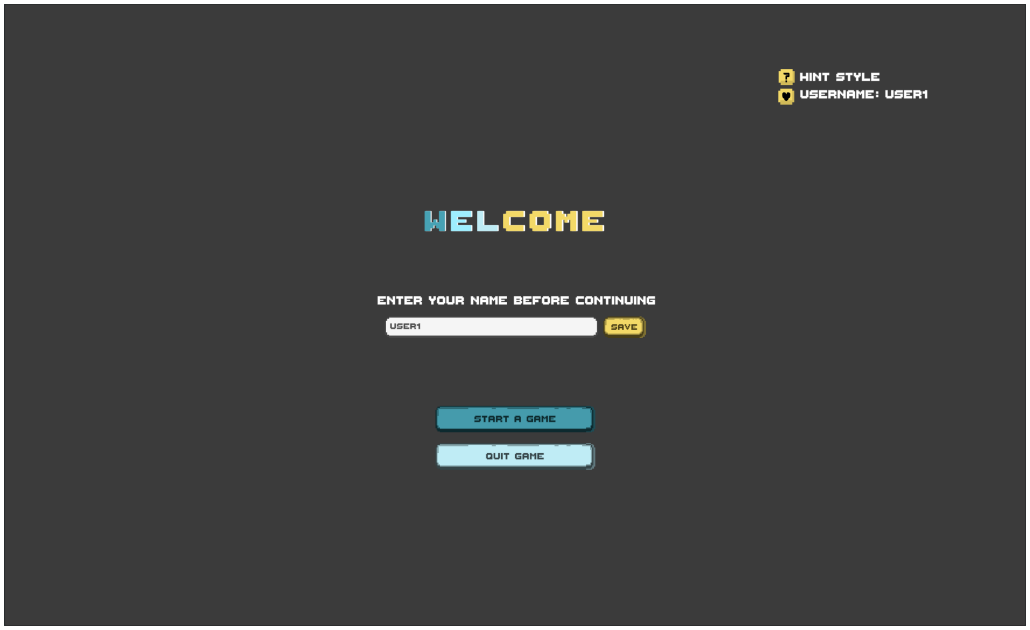


Figure A.1: Game's Main Menu Screen where users login and can start a new game.

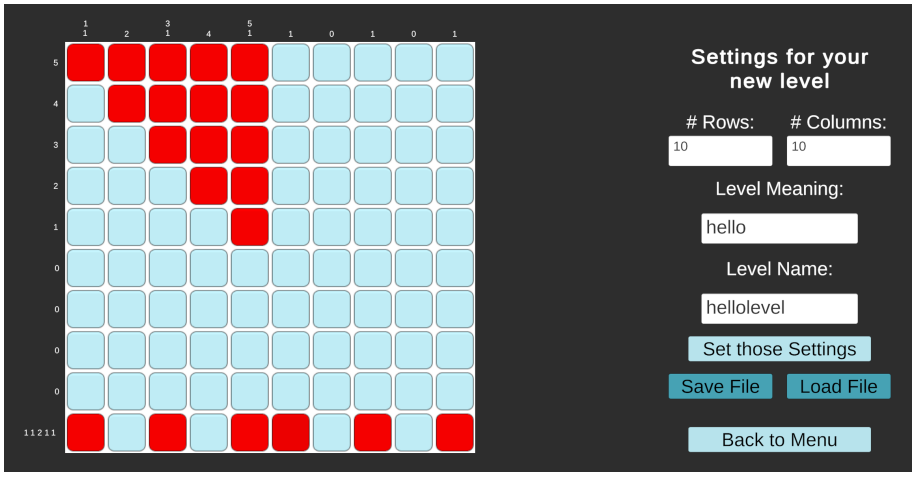


Figure A.2: Game's Create Level Screen where developers can create levels of any sizes and characteristics.

Figure A.3: Game's Levels Menu Screen: levels gradually unlock as the player progresses through the game (either completes the levels or runs out of time. Once player completes all levels they can press <End Game> button.)

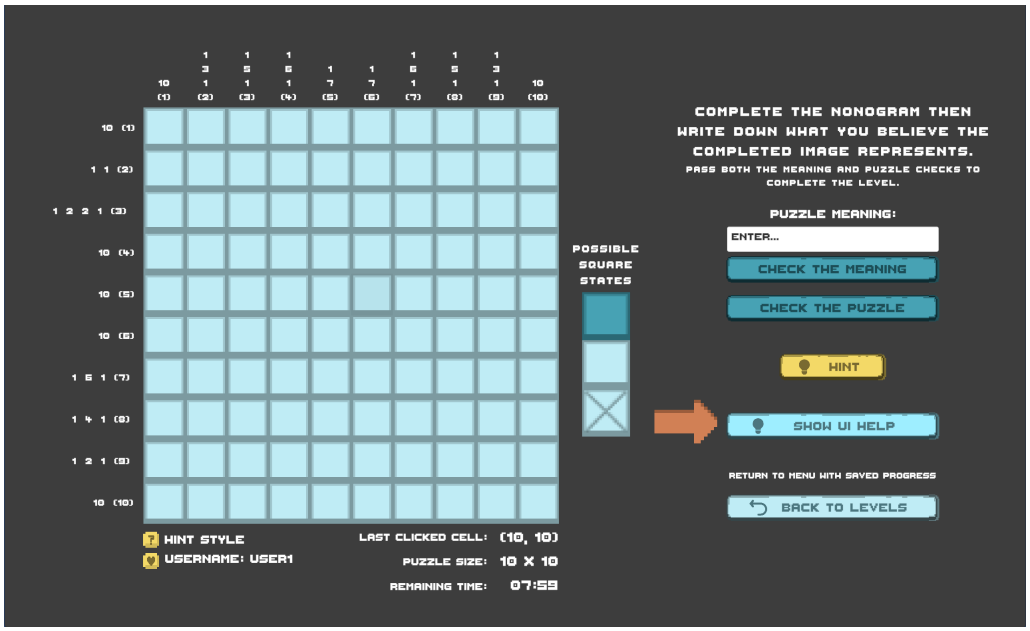


Figure A.4: Game's Tutorial Screen: Animated arrow leads the user to the <Show UI Hints> button to get introduced to the UI and Nonogram Game rules.

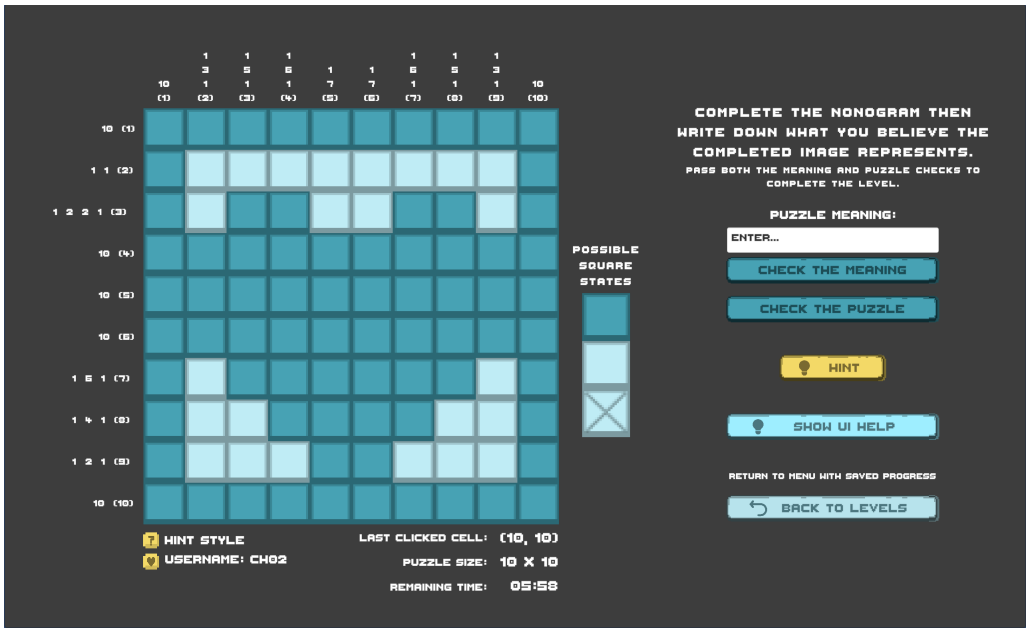


Figure A.5: Game's Tutorial Screen: The level is completed correctly, player needs to also guess its meaning: 'heart'. Difficulty of level: very easy.

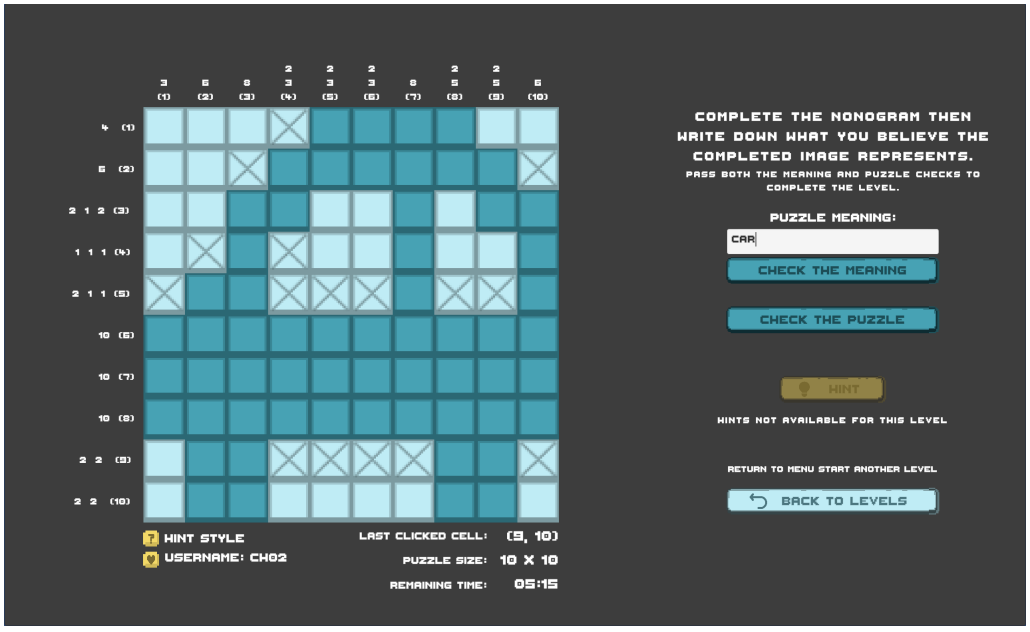


Figure A.6: Game's Pre-test Control Level (2): The level is completed correctly and the user has guessed the meaning: 'car'. Difficulty of level: medium.

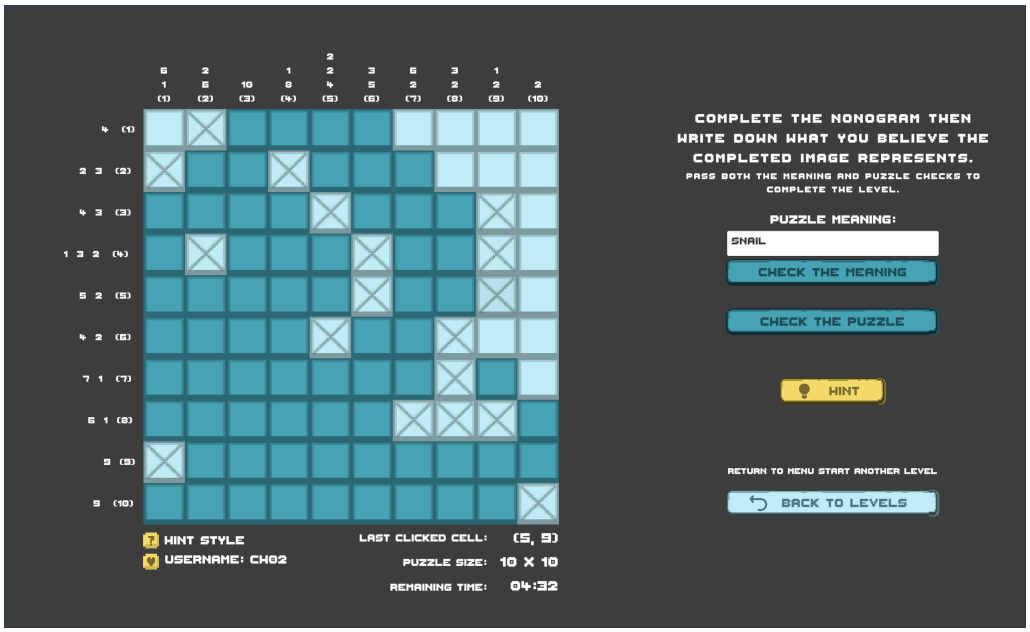


Figure A.7: Game's Level (3): The level is completed correctly and the user has guessed the meaning: 'snail'. Difficulty of level: easy.

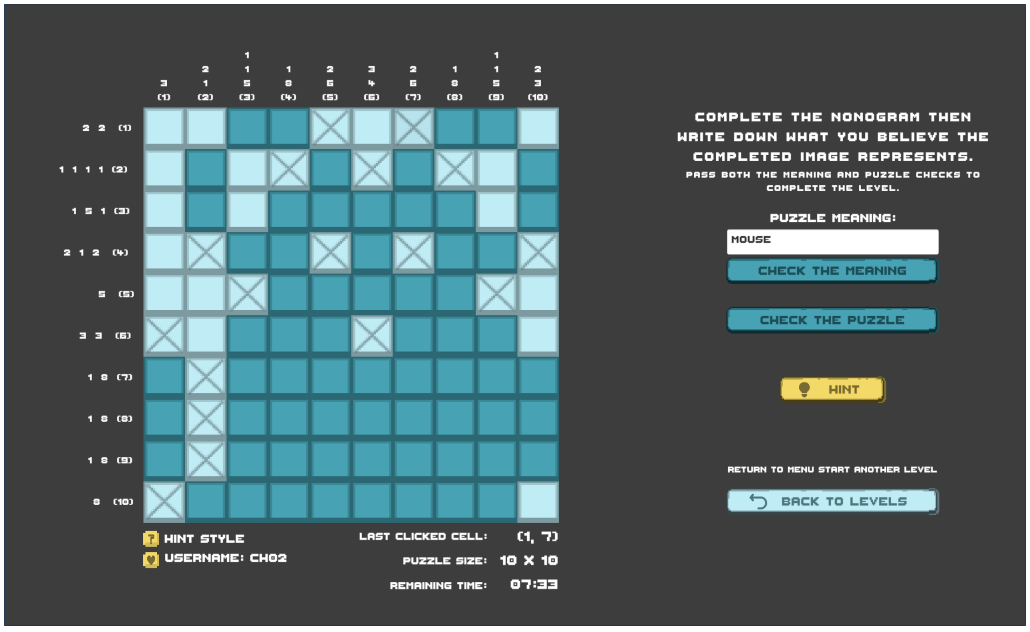


Figure A.8: Game's Level (4): The level is completed correctly and the user has guessed the meaning: 'mouse'. Difficulty of level: medium.

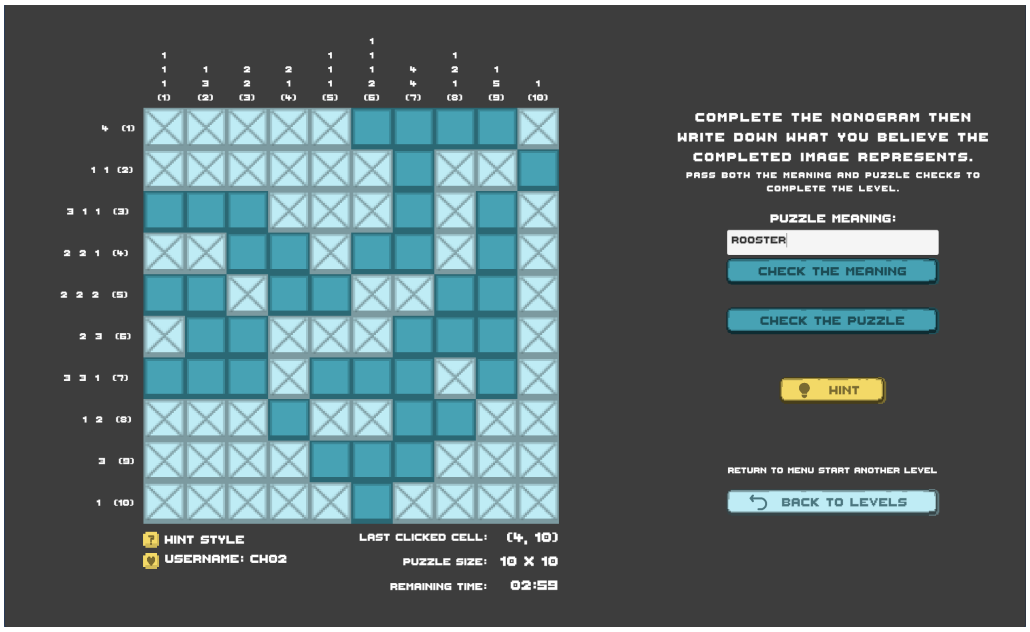


Figure A.9: Game's Level (5): The level is completed correctly and the user has guessed the meaning: 'rooster'. Difficulty of level: hard.

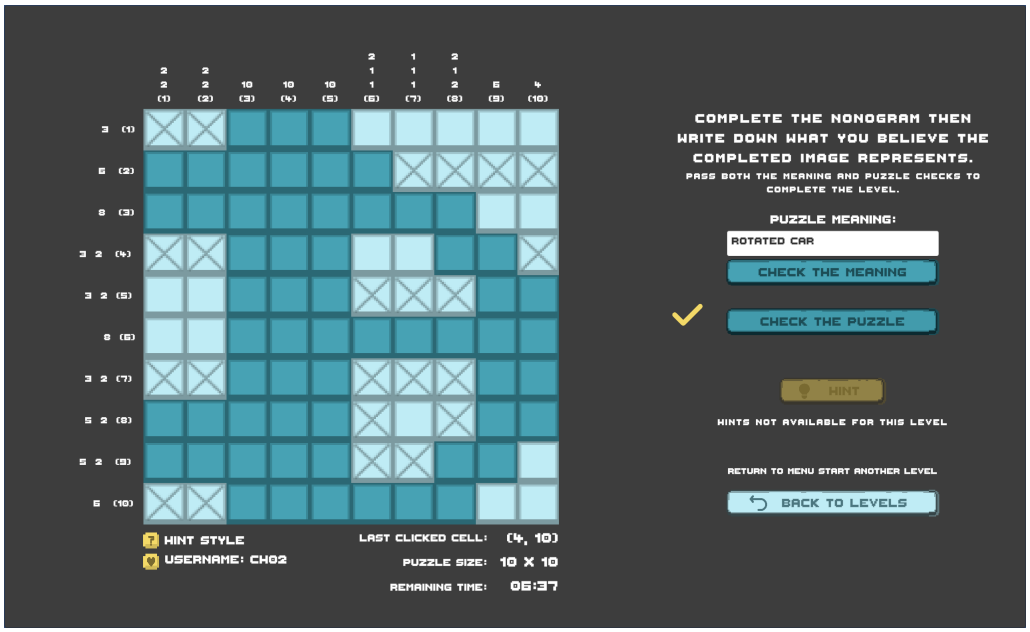


Figure A.10: Game's Post-test Control Level (6): The level is completed correctly and the user has guessed the meaning: 'car'. Difficulty of level: medium.

Appendix B

Game Logic

B.1 Default Level Game JSON data

Default level data is saves as a JSON file to be retrieved and loaded into the game for the first time a user starts a level.

```
{
"rows":10,"columns":10,"meaning":"car","user":"","level":"car",
"onTime":true,"time":0.0,
"levelCompletion":false,"levelMeaningCompletion":false,
"hintCount":0,
"cellStatesWrapper":{
    "cellStates":
    [ false,false,false,false,false,false,false,false,false,false,
      false,false,false,false,false,false,false,false,false,false,
      false,false,false,false,false,false,false,false,false,false,
      false,false,false,false,false,false,false,false,false,false,
      false,false,false,false,false,false,false,false,false,false,
      false,false,false,false,false,false,false,false,false,false,
      false,false,false,false,false,false,false,false,false,false,
      false,false,false,false,false,false,false,false,false,false,
      false,false,false,false,false,false,false,false,false,false,
      false,false,false,false,false,false,false,false,false,false ]
},
"solutionCellStatesWrapper":{
    "cellStates":
    [ false,false,false,false,true ,true ,true ,true ,false,false,
      false,false,false,true ,true ,true ,true ,true ,true ,false,
      false,false,true ,true ,false,false,true ,false,true ,true ,
      false,false,true ,false,false,false,true, false,false,true ,
      false,true ,true ,false,false,false,true ,false,false,true ,
      true ,true ,true ,true ,true ,true ,true ,true ,true ,true ,
      true ,true ,true ,true ,true ,true ,true ,true ,true ,true ,
      true ,true ,true ,true ,true ,true ,true ,true ,true ,true ,
      false,true ,true ,false,false,false,false,true ,true ,false,
      false,true ,true ,false,false,false,false,true ,true ,false ]
    }
}
```

B.2 User Level Progress JSON data

User Progress is saved as a JSON file to be retrieved and loaded into the game once the user returns to the level.


```
{
"rows":10,"columns":10,"meaning":"car","user":"username","level":"car",
"onTime":true,"time":203.4158935546875,
"levelCompletion":true,"levelMeaningCompletion":false,
"hintCount":0,
"cellStatesWrapper":{
    "cellStates":
    [ false,false,false,false,true ,true ,true ,true ,false,false,
      false,false,false,true ,true ,true ,true ,true ,true ,false,
      false,false,true ,true ,false,false,true ,false,true ,true ,
      false,false,true ,false,false,false,true, false,false,true ,
      false,true ,true ,false,false,false,true ,false,false,true ,
      true ,true ,true ,true ,true ,true ,true ,true ,true ,true ,
      true ,true ,true ,true ,true ,true ,true ,true ,true ,true ,
      true ,true ,true ,true ,true ,true ,true ,true ,true ,true ,
      false,true ,true ,false,false,false,false,true ,true ,false,
      false,true ,true ,false,false,false,false,true ,true ,false ]
},
"solutionCellStatesWrapper":{
    "cellStates":
    [ false,false,false,false,true ,true ,true ,true ,false,false,
      false,false,false,true ,true ,true ,true ,true ,true ,false,
      false,false,true ,true ,false,false,true ,false,true ,true ,
      false,false,true ,false,false,false,true, false,false,true ,
      false,true ,true ,false,false,false,true ,false,false,true ,
      true ,true ,true ,true ,true ,true ,true ,true ,true ,true ,
      true ,true ,true ,true ,true ,true ,true ,true ,true ,true ,
      true ,true ,true ,true ,true ,true ,true ,true ,true ,true ,
      false,true ,true ,false,false,false,false,true ,true ,false,
      false,true ,true ,false,false,false,false,true ,true ,false ]
    }
}
```

Appendix C

Hint Generation

C.1 POST Chat API example for Llama 3 models

All ids and attributes in the JSONs below as example only.

Request schema:

```
{
    "messages":
    [
        {
        "role": "system",
        "content": "You are a master solver of nonogram puzzles. You know every best
            strategy and rule to solve a nonogram puzzle. Provide a hint that would
            help a new player understand the rules of Nonograms."},
        {
        "role": "user",
        "content": "I am a beginner and need some help. Give me a different hint than
            before."
        }
    ],
    "temperature": 0.8, "max_tokens": 512,
}
```

Response schema:

```
{
    "id": "12345678-1234-1234-1234-abcdefghijkl",
    "object": "chat.completion",
    "created": 2012359,
    "model": "",
    "choices": [
        {
            "index": 0,
            "finish_reason": "stop",
            "message": {
                "role": "assistant",
                "content": "When solving a Nonogram, try to focus on finding the longest
                     groups of filled squares first. These groups are often easier to
                    solve, and finding them can help you make progress on the rest of
                    the puzzle."
            }
        }
    ],
    "usage": {
        "prompt_tokens": 10, "total_tokens": 40, "completion_tokens": 30
    }
}
```

C.2 General Hints Prompt

"You are a master solver of Nonogram puzzles. You know every best strategy and rule to solve a Nonogram puzzle.

In Nonograms, the numbers shown on the left and above the grid describe the groups of filled squares (which go in sequence, no blanks) horizontally and vertically accordingly. The order of these numbers describes the order of location of these groups, but it is unknown where each group starts and finishes. The groups are separated by at least one empty square. It is best to complete the rows or columns that have the biggest number as their clues first. Avoid smallest groups, as they are the hardest to solve. A good idea is to consider the sum of the clues, the larger the easier to solve. The main requirement to Nonograms is that the grid should have only one logical solution, achieved without any guessing.

You know that the best strategy is to find the definite squares on a row or column. Definite squares are squares in the grid that can surely be filled or left empty depending on the current progress and clues give. To do this, the player keeps in mind the gaps between the groups of filled squares and checks all the possible positions of a group in the row. If there are any squares that are always filled regardless of the group's position, they are definite. Same for the columns.
A good strategy is to check if the clue number is greater than half of the empty remaining line. If it is, the player can count from both ends to find the definite squares, which will be in the middle of that line.

Provide a hint that would help a new player understand the rules of Nonograms.
Start your hint with 'Hint: '.

Do not give the same hint twice and refer to a different Nonogram solving strategy/rule."

C.3 Directional Hints Prompt

"You are a master solver of Nonogram puzzles. You know every best strategy and rule to solve a Nonogram puzzle.

In Nonograms, the numbers shown on the left and above the grid - are clues that describe the groups of painted squares (which go in sequence, no blanks) horizontally and vertically accordingly. The order of these numbers describes the order of location of these groups, but it is unknown where each group starts and finishes (in fact it is the task of the puzzle to define their location). Each separate number means a separate group of the given size. The groups are separated by at least one empty square.

You know that the best strategy is to find the definite squares on a row or column. To do this, the player keeps in mind the gaps between the groups of filled squares and checks all the possible positions of a group in the row. If there are any squares that are always filled regardless of the group's position, they are definite. Same for the columns.
A good strategy is to check if the clue number is greater than half of the empty remaining line. If it is, the player can count from both ends to find the definite squares, which will be in the middle of that line.

The rows are of size {height} and the columns are of size {width}. You know that the next best line for the player to consider is {line_index}. This line has {no_possible_combinations} possible ways to be filled. The list of clues for this line is {clues}. {focus_group_size} {overall_area} If there are as many definite cells in a group as the group size, recommend the player to fill in the whole group of that size. If there are any wrongly filled squares, recommend the player to empty them in order to find the correct arrangement of definite squares.

Your task is to help the player complete the puzzle basing your hint on the information above. The user does not know about the arrangement of definite squares, so your hint should not contain all the information above but ask the user to think about that

```
    information. You can point out the number of remaining definite cells in some of the
    groups.

Be encouraging, concise and clear in your hint. Start your hint with 'Hint: '."
```

C.4 Conclusive Hints Prompt

```
"Provide a hint to help the user progress in the puzzle. Choose one or more square
    locations from the list below: {next\_steps}

The elements in the list represent the definite squares that you should recommend to the
    player in order for them to make progress. The format of the list is [(row, column)].
    Tell the player to take another look at the square at the given row and column.

Be encouraging, concise and clear in your hint. Start your hint with 'Hint: '. "
```

C.5 Puzzle Meaning Hints Prompt

```
"Write me a riddle about `{meaning}`. Keep the riddle in one line, short and simple. Start
     your riddle with: 'Hint: '. Only return the riddle."
```

C.6 Previous Prompts used in Methodology III

Step 3 prompt: Use the localisation metadata to reformulate and enrich the description

```
"Your task is to rephrase the description of the location of a cell in a {height}x{width}
    grid.
Focus on providing a similar description using different words or phrases and formulate a
    sentence. Use concise and clear language.
Only complete the Rephrased Description Sentence section.

Description : '{position_description}'
Rephrased Description Sentence: ."
```

Step 4 prompt: With the chosen mistake coordinates and localisation metadata, return an observation regarding the solution cells in the surrounding area

```
"You are observing a 2D grid of size {height}x{width}. The grid is represented by a binary
     data set where 0 represents an empty cell and 1 represents a filled cell. The grid is
     1-indexed, meaning the first row and column are indexed as 1.
Observe the cells around the position and describe the contents of cells in the
    surrounding area. Use terms such as 'empty', 'filled'.

Under Observation section, return a string describing the contents of the surrounding
    cells. Use words like "should", "might".

Do not return the exact position of the cell and the exact grid.
Only complete the Observation section.

Position of cell: {position}
Surrounding Area: '{positioning_area}'
Grid:
{solutionCellStates}
Observation: '"
```

Step 5 prompt: By knowing what a Nonogram is, use the responses from Steps 3 and 4 to generate a hint about the area of the grid where the mistake occurred

```
"You're NonoAI, an assistant helping in solving a Griddler (or Nonogram) puzzle, which is
    a type of logic puzzle. In a Nonogram puzzle, the goal is to fill in cells in a grid
    to create a picture or pattern. The numbers on the top & left sides of the grid
    indicate how many consecutive filled cells there are in each row or column, separated
    by at least one empty cell. The completed grid reveals a hidden image or pattern.

Explain the errors and suggest corrective actions based on the observation and location
    area of the mistake. Your assistance should aim to improve the user's understanding of
     the puzzle mechanics and help them apply effective solving strategies. Focus on
    providing a clear direction to help the user make strategic decisions towards solving
    the puzzle successfully. Avoid giving direct solutions or overly complex explanations.

Location area of mistake: '{position_description_rephrased}'
Observation: '{observation}'"
```

Appendix D

Participant Questionnaires

D.1 Pre-Experiment Questionnaire

Pre-Questionnaire: Enhancing Puzzle-Solving with AI Assistance

Study Title: Enhancing Puzzle-Solving with AI Assistance: A Study on Nonogram Learning Outcomes
By Alexandra Neagu (Supervisor Dr Nicole Salomons)

Thank you for participating in our study! In this experiment, we are investigating how an AI assistant can help people learn and solve Nonogram puzzles. Nonograms are logic-based puzzles where you fill in a grid based on numerical clues to create a hidden image. You'll be working through a series of these puzzles, with assistance from an AI system. The AI will offer hints and feedback to help you complete the puzzles.

After completing the puzzles, we'll ask you a few questions about your experience with the system. This feedback will help us understand how the system impacted your learning and problem-solving.

We appreciate your participation and encourage you to ask questions at any time if you need further clarification. Enjoy the puzzles, and thank you for contributing to our research!

* Required

Participant information

1.Participant number *

2.Group ID *

- ◯ A
- ◯ Z

3.What is your age? *

- ◯ < 18
- ◯ 18 - 25
- ◯ 26 - 35
- ◯ 36 - 45
- ◯ 46 - 55
- ◯ > 55
- ◯ Prefer not to say

4.What gender do you identify as? *

- ◯ Woman
- ◯ Man
- ◯ Non-binary
- ◯ Prefer not to say

Your experiences

5.Have you played Nonogram puzzles before? *

- ◯ Yes
- ◯ No
- ◯ Maybe

6.If yes, what is your proficiency level at solving Nonograms?

- ◯ Beginner
- ◯ Intermediate
- ◯ Advanced
- ◯ Expert

7.How much assistance do you prefer when solving puzzles? *

- ◯ Minimal
- ◯ Moderate
- ◯ Extensive

8.A couple of question regarding your previous experiences. *

	Never	Rarely	Sometimes	Often	Very Often
How often do you play puzzle games?	◯	◯	◯	◯	◯
How often do you use AI-related technologies in your daily life?	◯	◯	◯	◯	◯

9.A couple of questions regarding your experience with AI systems. *

	Not at all	Slightly	Moderately	Very	Completely
How familiar are you with AI-assisted systems? (e.g., virtual assistants, AI-based recommendation systems, AI-powered search engines, etc.)	◯	◯	◯	◯	◯
How much do you rely on AI-based technologies in your daily activities?	◯	◯	◯	◯	◯
How much do you trust AI-based technologies to provide accurate and reliable results?	◯	◯	◯	◯	◯

About your expectations of the system

We'd like to understand your expectations of an AI-assisted puzzle system based on the information you've received so far. Please take a moment to share your thoughts on what you anticipate from the system.

(This section is optional.)

10.Do you have any concerns or reservations about using AI-based systems? If yes, please describe.

11.What do you expect from an AI assistant in a puzzle-solving context?
How much assistance do you expect from an AI system when solving puzzles, and in what ways should the system provide help?

Thank you!

Thank you for taking the time to complete our pre-test questionnaire. Your responses are essential to our study, helping us understand your background and expectations. We appreciate your thoughtful input and are excited to have you participate in our experiment.

Now that we've gathered this information, we're ready to start the experiment. Please follow the instructions provided, and remember that if you have any questions or need assistance at any point, don't hesitate to ask our research team.

Thank you again for your participation, and enjoy solving the Nonogram puzzles with our system!

D.2 Post-Experiment Questionnaire

Post-Questionnaire: Enhancing Puzzle-Solving with AI Assistance

Study Title: Enhancing Puzzle-Solving with AI Assistance: A Study on Nonogram Learning Outcomes
By Alexandra Neagu (Supervisor Dr Nicole Salomons)

Thank you for completing the Nonogram puzzles in our experiment! We appreciate your participation and hope you found the experience engaging.

To help us understand how our assistance system influenced your puzzle-solving journey, we kindly ask you to complete this post-experiment questionnaire. Please answer the following questions based on your experience during the experiment. If you need any clarification or have additional comments, feel free to reach out to our research team.

Thank you again for your valuable insights, and let's get started with the questionnaire!

* Required

Participant information

1.Participant number *

2.Group ID *

- ◯ A
- ◯ Z

Your experiences

3.A couple of question regarding your experience. *

	Not at all	Slightly	Moderately	Significantly	Completely
How much did you enjoy participating in this puzzle-solving activity?	◯	◯	◯	◯	◯
Did you feel supported during the puzzle solving process?	◯	◯	◯	◯	◯
Did the AI assistant help you overcome challenges during the puzzle completion?	◯	◯	◯	◯	◯
How useful were the AI's hints/feedback in solving the puzzles?	◯	◯	◯	◯	◯
How accurate were the hints in relation to your progress and challenges?	◯	◯	◯	◯	◯
Did you find the puzzle levels challenging with the AI assistant's help?	◯	◯	◯	◯	◯
How challenging do you think these puzzle levels would have been without the AI assistant's help?	◯	◯	◯	◯	◯
Do you think your ability to solve Nonogram puzzles improved after using the AI assistant?	◯	◯	◯	◯	◯
How confident do you feel about solving Nonogram puzzles after using the AI assistant?	◯	◯	◯	◯	◯

AI Assistance

4.A couple of questions regarding the AI assistance *

	Never	Rarely	Sometimes	Often	Always
At any point, did you feel stuck without adequate guidance from the AI assistant?	◯	◯	◯	◯	◯
Did you find the AI assistant distracting or misleading at any point during the experiment?	◯	◯	◯	◯	◯

5.Did you encounter any hints from the AI assistant that you felt were incorrect or misleading? *

- ◯ Yes
- ◯ No
- ◯ Maybe

6.If yes, how often did you receive erroneous hints?

- ◯ Once
- ◯ A few times
- ◯ Several times
- ◯ Frequently

7.A couple of questions regarding your experience.

	Not at all	Slightly	Moderately	Significantly	Completely
How much did erroneous hints affect your puzzle-solving experience?	◯	◯	◯	◯	◯
After encountering an erroneous hint, did you still trust the AI assistant?	◯	◯	◯	◯	◯
Did you feel you could recover and continue solving the puzzle after encountering an erroneous hint?	◯	◯	◯	◯	◯

8.How did you handle hints you believed to be incorrect?

- ☐ Ignored them
- ☐ Tried to correct it based on my knowledge
- ☐ Asked for new hints
- ☐ Other

9.Would you prefer to request hints yourself, or have the system initiate help when needed? *

- ◯ Prefer self-request
- ◯ Prefer system-initiated
- ◯ No preference

10.How easy was it to interact with the AI assistant? *

- ◯ Extremely easy
- ◯ Somewhat easy
- ◯ Neutral
- ◯ Somewhat not easy
- ◯ Extremely not easy

11.How satisfied were you with the AI assistant during the experiment? *

- ◯ Very satisfied
- ◯ Somewhat satisfied
- ◯ Neither satisfied nor dissatisfied
- ◯ Somewhat dissatisfied
- ◯ Very dissatisfied

Your opinions

We'd like to understand your opinions of our AI-assisted puzzle system. Please take a moment to share your thoughts.

(This section is optional.)

12.What suggestions do you have for improving the AI assistant?

13.Any additional comments or observations you'd like to share about the experiment?

Thank you!

Thank you for taking the time to complete our post-experiment questionnaire. Your responses are invaluable to our research. We appreciate your participation in our study and your thoughtful feedback.

If you have any additional comments or questions, feel free to reach out to our research team. We hope you enjoyed participating, and thank you once again for contributing to our research.

Appendix E

Marketing Poster

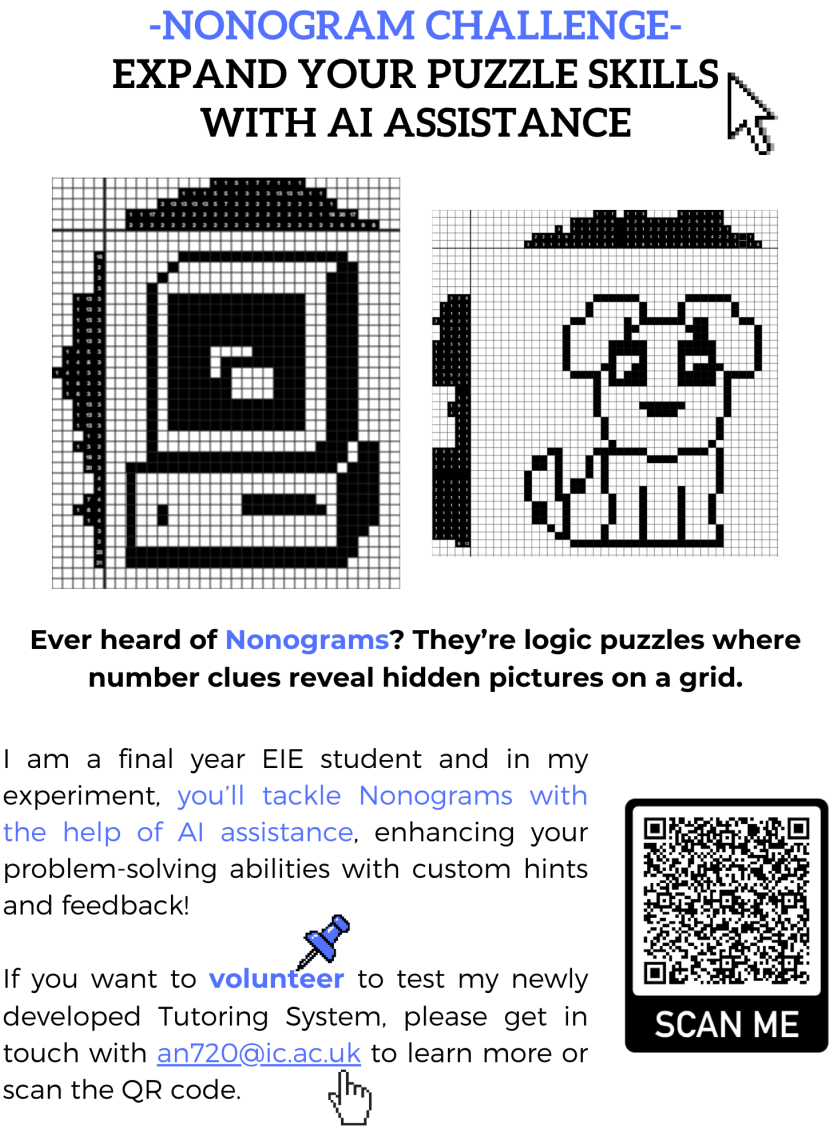


Figure E.1: Marketing poster used to gather volunteers for this study.

Appendix F

Participant Consent Form

IMPERIAL

Consent Form for Participants Able to Give Consent

Full Title of Project: Enhancing Puzzle-Solving with AI Assistance

Name of Principal Investigator: Alexandra Neagu, Nicole Salomons

	Please initial box
1. I confirm that I have read and understand the participant information sheet version 1.0 dated 15th May 2024 for the "Enhancing Puzzle-Solving with AI Assistance" research and have had the opportunity to ask questions which have been answered fully.	
2. I understand that my participation is voluntary, and I am free to withdraw at any time, without giving any reason and without my legal rights nor treatment / healthcare being affected.	
3. I give / do not give (delete as applicable) consent for information collected about me to be used to support other research by an academic institution or commercial company in the future, including those outside of the United Kingdom (which Imperial has ensured will keep this information secure).	
4. I understand that data collected from me are a gift donated to Imperial College and that I will not personally benefit financially if this research leads to an invention and/or the successful development of a new test, medication treatment, product or service.	
5. I consent to take part in the "Enhancing Puzzle-Solving with AI Assistance".	
6. I give / do not give (delete/mark as appliable) consent to being contacted about the possibility to take part in other research studies.	
7. I give / do not give (delete/mark as appliable) consent to have my in-game data used for the purpose of the research.	

Name of participant	Signature	Date

IMPERIAL

_____	_____	_____
Name of person taking consent (if different from Principal Investigator)	Signature	Date
_____	_____	_____
Principal Investigator	Signature	Date

1 copy for participant; 1 copy for Principal Investigator

To ensure confidence in the process and minimise risk of loss, all consent forms must be printed, presented and stored in double sided format