Personalizing Employee Learning in the Tech Sector: Implementing AI-Based Procedural Content Generation in Corporate Training

DISSERTATION

Submitted in partial fulfilment of the requirements of the
Degree: MTech in Artificial Intelligence & Machine Learning

By

Pritam Mondal

BITS ID No. 2022AC05090
ORC ID No. 0009-0004-9273-9749

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) INDIA
December 2024

Abstract

The main goal of this project is to develop and put into practice a new AI-driven approach to procedural content generation (PCG), one that draws on large language models (LLMs) and other generative AI tools, like Transformer-based systems and GPT variants, to create more personalized training experiences for employees in the tech industry. By generating training materials on the fly—materials that adapt to each learner's current skill set and evolving learning needs—this framework addresses a key challenge: providing corporate training that stays relevant, responsive, and flexible. Ultimately, this approach promises to improve engagement, speed up skill development, and boost the overall effectiveness of enterprise training programs.

Current research in adaptive learning, intelligent tutoring, and PCG shows that when learning content is closely aligned with a user's individual profile and is fine-tuned through continuous feedback, learners stay more engaged, remember more, and learn faster. Even so, there's a noticeable gap in how these personalization strategies—particularly those powered by advanced LLMs—are being applied in fast-moving corporate settings. This project aims to fill that gap by tailoring generative AI and Transformer-based methods to the real-world needs of businesses, ensuring that learning content remains not only up-to-date and on-target, but also genuinely motivating for employees.

Traditional corporate training tends to rely on static course materials and uniform structures. This one-size-fits-all approach often fails to account for the wide range of learning preferences and the constantly changing technologies that employees need to master. The result is often lackluster engagement, slower skill growth, and inefficient resource use. In contrast, this new framework harnesses the power of cutting-edge LLMs and GPT-based capabilities to deliver context-aware exercises, simulations, and assessments. Guided by ongoing performance data and learner feedback, the system adjusts in real time—tweaking difficulty, complexity, and thematic focus so the learning experience evolves alongside the learner's progress and the company's strategic priorities.

From a methodological standpoint, the project will build a hybrid AI system that uses both supervised and reinforcement learning. First, supervised models will organize and classify domain-specific knowledge to create an initial content library. Then, reinforcement learning agents will step in, using performance metrics and feedback loops to fine-tune how content is sequenced, how challenging it is, and what forms it takes. Transformer-based LLMs, including GPT models, will be the workhorses generating dynamic, scenario-rich learning modules that reflect today's industry standards and emerging trends. This cycle of continuous adaptation ensures the material remains relevant, engaging, and motivating over time.

There are clear and tangible benefits to this approach. By personalizing learning paths to each employee's unique abilities and needs, companies can dramatically cut the time it takes workers to become proficient, improve their overall adaptability, and raise the collective skill level of the workforce. Additionally, because this framework relies on scalable LLMs, it can be easily adapted for different sectors, specializations, and roles within an organization. In short, this project represents a vital step forward, bringing the adaptability of advanced generative models into the heart of corporate training, and paving the way for richer, more effective learning solutions in the future.

1. Broad Area of Work

The broad area of this work is **AI-driven personalized learning for corporate training**, with a particular focus on applying large language models (LLMs) and generative AI to create adaptive and context-relevant training content in the tech sector.

2. Objectives

The objectives of my project are as follows:

- To design and implement a procedural content generation (PCG) framework that leverages LLMs and generative AI models (e.g., GPT-based architectures) for delivering personalized corporate training modules.
- To develop a learner profiling system that uses performance analytics and continuous feedback to dynamically adjust training materials based on difficulty, complexity, and thematic relevance.
- To evaluate how well this AI-driven approach enhances engagement, improves skill retention, and speeds up the time it takes learners to become fully competent, comparing these results to traditional, fixed-content training methods.
- To offer insights into the scalability and real-world practicality of such a system in fast-moving, tech-centric corporate environments.

3. Scope of Work

Scope of this dissertation is to design and develop:

- A prototype AI-driven PCG platform capable of generating contextually relevant, on-demand training exercises and modules using Transformer-based LLMs.
- A learner modeling and feedback loop mechanism that adjusts content difficulty and learning progression in real time.
- Initial evaluation tools and metrics to assess learner engagement, outcomes, and overall satisfaction in a controlled pilot environment.

It is important to note that this project won't develop a fully commercial learning management system. Instead, it will concentrate on creating a proof-of-concept model and conducting empirical tests of its personalized training capabilities within a representative corporate environment.

4. Detailed Plan of Work (for 16 weeks)

The plan of work is divided into sub-tasks with defined timelines and measurable deliverables. This schedule may be adjusted as the project progresses, but it serves as an initial framework for the 16-week period.

Serial No.	Tasks or Subtasks to be done (be precise and specific)	Start Date - End Date	Planned Duration (Weeks)	Specific Deliverable in terms of the project
1	Conduct detailed literature review and refine project objectives	Week 1 - Week 2	2	Literature Review Report & Finalized Objectives
2	Design system architecture (LLM integration, PCG pipeline, learner profiling components)	Week 3 - Week 4	2	System Architecture Diagram & Component Specifications
3	Dataset curation and domain-specific fine-tuning of LLM model	Week 5 - Week 6	2	Fine-Tuned LLM Model & Data Preprocessing Report, Explore GAN
4	Implement base PCG framework for content generation	Week 7 - Week 8	2	Prototype PCG Module & Sample Generated Content
5	Integrate learner modeling and feedback loops to adapt difficulty and sequencing	Week 9 - Week 10	2	Adaptive Content Delivery Module Integration
6	Develop user interface and user interaction flows	Week 11 - Week 12	2	Functional UI Prototype & User Interaction Guidelines
7	Testing & Validation (user feedback, performance analytics, comparison with static training)	Week 13 - Week 14	2	Test Reports, User Feedback Surveys, Initial Evaluation Metrics
8	Data analysis, result compilation, and final documentation	Week 15 - Week 16	2	Final Report Draft, Presentation Slides, and Conclusions

5. Literature References

The following are referred journals from the preliminary literature review:

1. J. Roberts and K. Chen, "*Learning-Based Procedural Content Generation,*" in *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, no. 1, pp. 88-101, March 2015, doi: 10.1109/TCIAIG.2014.2335273.
2. Wang,Y. (2023). "*Procedural content generation for VR educational applications: The investigation of AI-based approaches for improving learning experience*." Applied and Computational Engineering,17,23-31.
3. S. Maghsudi, A. Lan, J. Xu and M. van der Schaar, "*Personalized Education in the Artificial Intelligence Era: What to Expect Next*," in *IEEE Signal Processing Magazine*, vol. 38, no. 3, pp. 37-50, May 2021, doi: 10.1109/MSP.2021.3055032.
4. Murtaza, Mir, Yamna Ahmed, Jawwad Ahmed Shamsi, Fahad Sherwani and Mariam Usman. "*AI-Based Personalized E-Learning Systems: Issues, Challenges, and Solutions.*" *IEEE Access* 10 (2022): 81323-81342.
5. Rukadikar, A., & Khandelwal, K. (2023). Artificial intelligence integration in personalised learning for employee growth: a game-changing strategy. *Strategic HR Review*. doi.org/10.1108/shr-08-2023-0046.

TABLE OF CONTENTS

1: Introduction

Corporate compliance training is mandatory in many industries but is often viewed as a dull, checkbox exercise by employees (The Benefits of Adaptive Compliance Courses | Learning Pool). Traditional e-learning courses present the same static content to all learners, relying on hindsight metrics like completion rates and seat time to judge effectiveness (The Benefits of Adaptive Compliance Courses | Learning Pool). This one-size-fits-all approach tends to disengage learners and fails to provide proactive insight into learner needs or potential compliance risks (The Benefits of Adaptive Compliance Courses | Learning Pool). There is a pressing need for compliance training that is not only **effective** in conveying policies and best practices but also **engaging** enough to hold learners' attention and motivate behavioral change (Compliance Training Reinvented: The Scenario-Based Learning Advantage) (Compliance Training Reinvented: The Scenario-Based Learning Advantage).

One promising solution is **adaptive learning**, where the training content and pathway adjust in real-time to each learner's performance, knowledge gaps, and preferences. Adaptive compliance training leverages data on learner interactions to personalize the experience, going beyond mere completion of modules to truly **tailor the journey** to individual needs (The Benefits of Adaptive Compliance Courses | Learning Pool). Such an approach can transform compliance education from a routine task into a learner-centric experience that drives meaningful understanding and retention of compliance knowledge (The Benefits of Adaptive Compliance Courses | Learning Pool). For example, Learning Pool's adaptive compliance courses adapt content **in real-time** based on each learner's performance, guiding them through a personalized path and "transcending mere completion" to achieve behavioral change (The Benefits of Adaptive Compliance Courses | Learning Pool). Compared to static courses, adaptive learning systems have demonstrated benefits like reduced training time and improved engagement – in one case, adaptive e-learning cut learners' training **seat time** by up to 50% by skipping content they had already mastered (The Benefits of Adaptive Compliance Courses | Learning Pool). Studies have found that adaptive learning can increase information retention significantly (by up to 40% in some corporate settings) and boost learner satisfaction (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®). This dissertation builds on the premise that adaptivity can greatly enhance compliance training outcomes, focusing on three key innovations to achieve adaptivity: **scenario-based learning**, **reinforcement learning-driven content selection**, and **psychoanalytic trait-based personalization**.

Scenario-based learning is a strategy where learners are placed in realistic situations requiring them to apply their knowledge to make decisions. It is widely regarded as an effective approach for compliance training (Compliance Training Reinvented: The Scenario-Based Learning Advantage). By simulating workplace dilemmas (e.g. a potential data privacy breach or an ethical decision point), scenarios make abstract rules concrete and memorable (Compliance Training Reinvented: The Scenario-Based Learning Advantage). When used correctly, scenario-based learning can significantly improve engagement and help learners understand and remember complex compliance information (Compliance Training Reinvented: The Scenario-Based Learning Advantage). It works because scenarios are **interactive and immersive**, requiring active learner participation. Learners must think critically and see the consequences of their choices, which makes the training more **memorable and meaningful**

than passive content (Compliance Training Reinvented: The Scenario-Based Learning Advantage). Scenario-based compliance training is more engaging than traditional methods specifically because it presents **real-world context** and immediate feedback on decisions (Compliance Training Reinvented: The Scenario-Based Learning Advantage). This approach also aligns with adult learning principles by promoting critical thinking and decision-making skills in safe environments, thereby building learner confidence in handling compliance situations in real life (Compliance Training Reinvented: The Scenario-Based Learning Advantage) (Compliance Training Reinvented: The Scenario-Based Learning Advantage).

While scenario-based learning enhances engagement, **adapting scenarios to each learner** can further improve effectiveness. A scenario that is too easy may bore a knowledgeable employee, whereas one that is too difficult or irrelevant might frustrate a novice. Here we introduce adaptivity into scenario-based compliance training using two AI-driven components: a **Large Language Model (LLM)** to generate and adjust scenario content dynamically, and a **Reinforcement Learning (RL)** agent to decide the optimal sequence or difficulty of scenarios for each learner. Additionally, we incorporate a **psychoanalytic trait model** to personalize the training based on individual psychological traits (such as personality factors or tendencies), which can influence how a learner engages with training material.

Recent advances in **Generative AI** and **LLMs** provide powerful new tools for creating rich, context-specific training content on the fly. Generative AI models can produce text that mimics realistic scenarios, dialogues, and even role-play interactions. By fine-tuning an LLM on compliance topics, we can generate a wide variety of training scenarios tailored to different roles, industries, or even learner personalities. Generative AI brings diversity and adaptability to content creation, reducing the workload on instructional designers and enabling real-time content adjustments (Bringing Generative AI to Adaptive Learning in Education). Instead of pre-authoring every training branch, we leverage an LLM to **invent scenario variations** and hints based on a learner's needs (for instance, providing more explanation if the learner is struggling, or new challenges if they have mastered the basics). This dynamic content generation means the training experience can be different for each learner and each session, maintaining engagement through novelty and relevance (Bringing Generative AI to Adaptive Learning in Education). Unlike static modules, an LLM-powered system can respond to user inputs or mistakes with contextually appropriate feedback and follow-up questions, much like a human tutor (LLM-based training using personalized case examples) (LLM-based training using personalized case examples). Early explorations in education technology suggest that LLM-based tutors can transform abstract material into interactive, personalized case examples, boosting learner engagement and understanding (LLM-based training using personalized case examples). Our system extends this idea to the compliance domain, using a fine-tuned LLM to serve as a scenario generator and virtual coach.

To control the adaptivity and decision-making in content delivery, we employ **reinforcement learning**. RL offers a framework for the system to "learn" the optimal way to steer each learner through the training content by trial and error optimization of rewards. In our context, the RL agent's actions could include selecting the next scenario or question, adjusting the difficulty level, or deciding whether to review a previous topic. The RL agent observes the state of the learning session (including the learner's performance so far and trait profile) and receives rewards based on outcomes like the learner's success in a scenario or engagement level. Over time, the agent develops a policy to maximize cumulative reward, which corresponds to

maximizing training effectiveness (e.g. knowledge retention and engagement). **Q-learning**, a classic model-free RL algorithm, is used as the learning strategy for our agent. Q-learning learns the **value of state-action pairs** by iterative updates, without needing a model of the environment's dynamics (Q-Learning - Technique D3A-QL | MITRE D3FEND™). This makes it well-suited for our application where the "environment" – a human learner – is complex and not easily modeled. Q-learning's goal is to learn an optimal policy *π(s) that maximizes expected reward by estimating the optimal action-value function* $Q(s,a)$ (Q-Learning - Technique D3A-QL | MITRE D3FEND™). In an educational context, prior works have applied RL to personalize review schedules and practice problem selection. For example, researchers formulated scheduling of quiz reviews as a sequential decision problem and used RL to adapt to each student's knowledge state, successfully maintaining higher long-term memory retention ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions). However, training an RL agent with real learners in the loop can be impractical due to the large number of interactions required ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions). Our approach mitigates this by generating synthetic training data and using simulated learners for the RL agent during development, as well as by warm-starting the agent with an initial policy (based on expert heuristics and trait data). Once deployed, the agent continues to refine its policy with each interaction, gradually improving the adaptivity for future learners.

Another innovation in our system is the incorporation of **psychoanalytic trait modeling** using a BERT-based classifier. The premise is that understanding a learner's psychological traits (such as personality type, learning style tendencies, or attitudes toward compliance) can help tailor the training to better suit that individual. Prior studies in human-computer interaction have shown that a user's personality can significantly impact their engagement with a system, and that aligning a system's interaction style with the user's personality leads to higher satisfaction and trust (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC). In the context of training, an extroverted, confident learner might prefer fast-paced, challenge-oriented scenarios, whereas an anxious or skeptical learner might need more reassurance and explanatory depth. Our **psychoanalytic trait model** uses natural language inputs from the user (such as their responses to open-ended scenario questions or a preliminary questionnaire) to infer personality traits. We fine-tuned a **BERT (Bidirectional Encoder Representations from Transformers)** model for this classification task. BERT is a state-of-the-art transformer-based language model that can be adapted for classification with excellent accuracy ([1810.04805] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding) ([1810.04805] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding). By conditioning the training content on traits (for instance, adjusting the tone of scenarios or the type of feedback given), we hypothesize that the training becomes more relatable and motivating for each user. Recent research supports this approach: Guo *et al.* (2024) constructed BERT-based models to predict users' Big Five personality traits from their dialogue utterances (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC), enabling dialogue systems to adapt to user personality and thereby improve task success and user satisfaction (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC). Our system extends trait prediction to the compliance training setting, using a BERT classifier to model traits and inform

the RL agent and LLM. We refer to this component as **psychoanalytic** trait modeling to emphasize its basis in psychological profiling of the learner.

Bringing these components together, we propose an end-to-end **Adaptive Compliance Training System**. The system presents branching compliance scenarios to the learner through a web-based interface. Behind the scenes, the RL agent selects and adapts scenarios, the fine-tuned LLM generates the scenario narratives and dialogue on the fly, and the trait model continuously updates the learner's profile. The training path thus dynamically conforms to each learner's performance and psychological makeup. For instance, if the learner struggles with a scenario, the system might trigger a remedial scenario or additional explanation (via the LLM) and assign a higher weight to that topic in the future (via the RL policy update). If the learner shows a particular personality trait (e.g. high conscientiousness or conversely resistance), the system can adjust the messaging of scenarios (via the LLM prompt) to resonate better with that trait. Over time, the adaptive system aims to keep the learner in an optimal zone of engagement—challenged but not overwhelmed—often referred to as the zone of proximal development in educational theory.

The contributions of this dissertation are as follows:

- We design a novel **system architecture** that integrates a front-end React application, a Flask back-end API, a fine-tuned LLM for scenario generation (Phi-3-Mini-4K-Instruct with LoRA fine-tuning), a Q-learning based adaptation agent, and a BERT-based psychoanalytic trait classifier. We illustrate how these components interact to deliver personalized compliance training content.
- We develop a method for **synthetic training data generation** to train the AI components. This includes using LLMs to generate diverse compliance scenarios for fine-tuning and creating simulated learner interaction data to pre-train the RL agent and trait model.
- We present a **detailed literature review** of relevant domains: adaptive learning systems, reinforcement learning in education, scenario-based learning, large language models in training content generation, and personality modeling for adaptive systems. This situates our work in the context of existing research and identifies how we extend the state of the art.
- We implement a working prototype of the system and conduct an **experimental evaluation** comparing it against a traditional non-adaptive compliance training module. Key metrics evaluated are learner engagement (e.g. interaction time, subjective engagement ratings), knowledge retention (quiz scores immediately and weeks after training), and adaptation effectiveness (how well the system's adjustments improved learning outcomes, especially for learners of different initial skill levels).
- We report the experimental results, which demonstrate that the adaptive system significantly outperforms the baseline on engagement and retention. Notably, the adaptive system users had higher completion rates and post-training test scores, and reported greater satisfaction. We also analyze system logs to show how the RL agent's policy evolved and how the trait model influenced content decisions.

- We discuss practical considerations such as scalability, ethical implications (e.g. ensuring the AI-generated content remains accurate and unbiased), and how this approach could be generalized beyond compliance training to other training domains.
- Finally, we provide **appendices** with supplementary material, including example scenarios generated by the LLM, details of the RL agent hyperparameters and training, additional charts from the user study, and the survey instruments used to measure engagement.

In summary, this work demonstrates a feasible and effective approach to **LLM-driven adaptive training** in a compliance setting. By continuously adapting to the learner through reinforcement signals and trait analysis, compliance training can become a more engaging, efficient, and personalized experience that better achieves its ultimate goal: genuine understanding and internalization of compliance principles.

2. Literature Review

In this section, we review relevant literature and prior work across four areas that form the foundation of our adaptive compliance training system: **adaptive learning** (in corporate training and e-learning contexts), **reinforcement learning (Q-learning) applications in education**, **psychoanalytic trait modeling and personality-based adaptation**, and **LLM-based scenario generation** for training.

2.1 Adaptive Learning in Compliance Training and E-Learning

Adaptive learning refers to educational systems that dynamically adjust content delivery to the individual learner's needs in real time (What is Adaptive Training and what are the benefits?). The concept has gained traction in both academia and industry as a way to improve learning efficiency and effectiveness. Instead of a fixed linear curriculum for all, adaptive systems use data from learner interactions (quiz results, response times, etc.) to personalize the learning path (What is Adaptive Training and what are the benefits?) (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®). This personalization can take many forms: skipping content the learner already knows, offering remedial help on weaknesses, changing the difficulty or modality of content, and so on (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®) (The Benefits of Adaptive Compliance Courses | Learning Pool). The goal is to keep learners appropriately challenged and support mastery at their own pace.

In the corporate compliance domain, the push for adaptivity is relatively recent but growing. Compliance training historically has been a checkbox requirement with *de minimis* engagement – employees often click through slides or pages just to complete the course. This not only undermines retention but also fails to identify or address real compliance knowledge gaps. As noted by Dunne (2024), conventional compliance e-learning relies on *hindsight* metrics (completion rates, time spent, simple feedback surveys) and provides little actionable insight (The Benefits of Adaptive Compliance Courses | Learning Pool). In contrast, an adaptive approach can provide *predictive* intelligence by analyzing each learner's progress and adjusting accordingly, which is invaluable for proactively identifying compliance risks (i.e., topics where many employees struggle) (The Benefits of Adaptive Compliance Courses | Learning Pool). Adaptive compliance training systems, such as those introduced by Learning Pool, collect data on each learner's interactions and performance and adapt the experience in **real-time** to fit their needs (The Benefits of Adaptive Compliance Courses | Learning Pool). This yields a learner-

centric experience – rather than just tracking completion, the system guides each employee through a **tailored journey** focused on mastery and behavior change (The Benefits of Adaptive Compliance Courses | Learning Pool).

Several benefits of adaptive learning in compliance and corporate training have been documented. A key advantage is **increased training efficiency**. By identifying areas where a learner is already proficient and omitting or shortening those sections, adaptive systems avoid wasting the learner's time (The Benefits of Adaptive Compliance Courses | Learning Pool). Learners can progress faster through familiar material and spend more time on the content that is new or challenging to them. This targeted approach has been shown to dramatically reduce training durations – for example, one adaptive platform reported cutting average learner seat time by ~50% for compliance courses by eliminating redundant content (The Benefits of Adaptive Compliance Courses | Learning Pool). In a case study at a pharmaceutical company, adding an AI-driven adaptive learning system (Realizeit) to their compliance training made the program **40% faster** to complete while yielding better assessment results (Improving Pharmaceutical Compliance Training with AI-based Adaptive Learning). This had a direct cost benefit as well, saving thousands of hours of workforce time.

Another benefit is **improved learner engagement**. When training is tailored to an individual, it tends to be more relevant and neither too easy nor too hard, thereby keeping the learner interested. Adaptive learning often incorporates frequent feedback and interaction, which sustains engagement (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®) (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®). If a learner gets something wrong, the system can immediately respond with a hint or a remedial branch, turning failures into learning opportunities rather than letting the learner disengage. Studies consistently report higher engagement and satisfaction in adaptive learning environments. For instance, a Gates Foundation study found that students using adaptive tools had significantly higher engagement than those in traditional classrooms (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®). In corporate settings, adaptive learning has been shown to increase retention of information by up to 40%, as engaged learners absorb and remember more (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®). Adaptive compliance courses also tend to incorporate elements like gamification (points, levels, or badges for correct answers) to motivate learners (The Benefits of Adaptive Compliance Courses | Learning Pool). When content aligns with each learner's role and prior knowledge, it "resonates" with them more directly, leading to both higher engagement and better **knowledge retention** (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®) (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®).

Adaptive learning's ability to **individualize learning paths** is particularly useful in compliance training because employees often come with very different backgrounds. Some may be well-versed in the policies (e.g. due to prior training or their job function) while others are novices. Traditional training forces both groups through the same material, which bores experts and overwhelms novices. Adaptive compliance training addresses this by continuously gauging each learner's understanding and adjusting. Musílek *et al.* (2018) observed that adaptive e-learning improved learning effectiveness by allowing learners to **skip material they already knew**, thus reducing the boredom and demotivation caused by rote repetition (What is Adaptive Training and what are the benefits?) (What is Adaptive Training and what are the benefits?). They noted that *"learning effectiveness decreases with routine completion of simple exercises.*

Pupils thus lose motivation." (What is Adaptive Training and what are the benefits?) By focusing only on the *necessary* content for each learner, adaptive systems keep learners in a state of engaged learning, which both improves effectiveness and reduces wasted time (What is Adaptive Training and what are the benefits?).

It is worth noting some challenges and prerequisites for adaptive learning. Implementing adaptivity requires sufficient content variation (alternative explanations, extra examples, harder questions, etc.) so that the system has choices in how to respond. It also relies on collecting detailed learner data and applying algorithms to interpret it. Early adaptive systems in e-learning used rule-based decision trees or simple mastery models (like requiring a certain score to move on). Modern systems are increasingly using machine learning to drive adaptivity, allowing more complex and subtle adaptation strategies. Our work joins this trajectory by using reinforcement learning to optimize adaptation decisions, as discussed later.

In summary, the literature strongly supports that adaptive learning approaches can **enhance compliance training** by making it more efficient, personalized, and engaging. Adaptive compliance training provides a *win-win*: employees spend less time on training yet learn more effectively (Improving Pharmaceutical Compliance Training with AI-based Adaptive Learning), and organizations benefit from better-trained staff and insights into where compliance understanding might be lacking (Improving Pharmaceutical Compliance Training with AI-based Adaptive Learning). The remainder of this review will examine specific technologies we leverage to implement such adaptivity: reinforcement learning for decision optimization, psychoanalytic trait modeling for deeper personalization, and LLM-based content generation for rich scenario delivery.

2.2 Reinforcement Learning (Q-Learning) for Adaptive Instruction

Reinforcement Learning (RL) is a learning paradigm where an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. In the context of an educational system, an RL agent can be conceived as a "tutor" that decides what problem or content to give a student next, receiving rewards based on the student's success (or other learning outcomes). Over time, the agent refines its strategy to maximize cumulative reward – which in an educational sense means maximizing the student's learning gains or engagement.

One popular RL algorithm is **Q-learning**, introduced by Watkins (1989). Q-learning is a model-free algorithm, meaning it does not require a known model of the environment's dynamics; it learns purely from experience (Q-Learning - Technique D3A-QL | MITRE D3FEND™). The algorithm learns an action-value function $Q(s,a)$, which estimates the expected cumulative reward if the agent takes action a in state s and then follows the optimal policy thereafter. Through iterative exploration, Q-learning updates its Q estimates using the Bellman equation: Qnew(s,a):=Q(s,a)+α[r+γmax⁡a′Q(s′,a′)−Q(s,a)], Q_{new}(s,a) := Q(s,a) + \alpha [r + \gamma \max_{a'} Q(s',a') - Q(s,a)] , where α is a learning rate, γ is a discount factor, and r is the reward received when transitioning from state s to s' via action a. Over many trials (episodes), $Q(s,a)$ converges to $Q^*(s,a)$, the true optimal value, under certain conditions (An Elementary Proof that Q-learning Converges Almost Surely - arXiv). The agent can then act by choosing, in each state, the action with the highest Q value (exploiting its learned knowledge) while occasionally exploring other actions to discover new knowledge (exploration). This simple algorithm has powerful convergence

guarantees and has been used in a range of applications where an explicit model is hard to get, which makes it suitable for interactive learning environments where human behavior is involved (Q-Learning - Technique D3A-QL | MITRE D3FEND™).

In adaptive learning systems, RL has been used to formulate the problem of content sequencing and adaptation as a sequential decision-making task. The *state* can represent the learner's knowledge status, progress, or even emotional state; the *actions* are pedagogical interventions (e.g., present a certain problem, give a hint, review a concept, etc.); and the *rewards* relate to learning outcomes (like immediate performance or long-term retention). One line of research has focused on optimizing **spaced repetition** and review schedules via RL. Kubotani *et al.* (2021) developed **RLTutor**, an adaptive tutoring framework that used RL to decide when to review learned items for maximizing memory retention ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions). They noted that while RL could in theory learn optimal review strategies that keep memory performance high, directly training on real students would require an infeasible number of interactions ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions). To address this, they created a *virtual student model* (simulated learners) so the RL agent could train without heavily burdening actual students ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions). Their experiments demonstrated that the RL-optimized strategy performed comparably to conventional (human-crafted) review strategies in maintaining retention ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions). This highlights both the promise of RL in educational adaptation and the practical challenge of data efficiency – an issue we also tackle by using simulated user data to pre-train our agent.

RL has also been applied to more complex educational scenarios such as open-ended dialogue tutors and serious games. Radmehr *et al.* (2024) studied integrating RL with LLMs to create intelligent agents for **text-based educational environments** (Towards Generalizable Agents in Text-Based Educational Environments: A Study of Integrating RL with LLMs). They compared three approaches: an "RL-based" agent that treats the environment in natural language but learns via RL, an "LLM-based" agent that uses few-shot prompting to decide actions (leveraging the LLM's knowledge), and a hybrid approach where an RL agent is assisted by an LLM for decision-making (Towards Generalizable Agents in Text-Based Educational Environments: A Study of Integrating RL with LLMs). Their findings were insightful: the pure RL agent excelled at achieving task completion goals (in their case, accomplishing tasks in a virtual pharmacy training scenario) but was not as good at performing pedagogically desirable behaviors (like asking good questions), whereas the pure LLM agent could generate reasonable questions but did not consistently drive the task to completion (Towards Generalizable Agents in Text-Based Educational Environments: A Study of Integrating RL with LLMs). The hybrid LLM-assisted RL agent overcame these limitations, achieving both strong task performance and quality interactions (Towards Generalizable Agents in Text-Based Educational Environments: A Study of Integrating RL with LLMs). This suggests that RL and LLM techniques can complement each other in educational settings – RL provides goal-directed optimization, and LLMs provide rich, knowledgeable interactions. Our system employs a similar philosophy: we use RL for high-level decision making (which

scenario to show next, when to review, etc.) and use an LLM for the content of those interactions. By clearly separating *what* the next step is (RL's job) from *how* that step is presented in language (LLM's job), we combine the strengths of both.

In terms of compliance training specifically, there is limited academic literature, but we can draw parallels from intelligent tutoring systems and serious games in related domains. Compliance scenarios can be thought of as a form of serious game or branching narrative where a user's choices determine what happens next. RL has been used in serious games to adapt the game difficulty to the player's skill – a concept known as **dynamic difficulty adjustment**. Similarly, in a branching training scenario, RL could adjust the path (e.g., offer an easier scenario if the learner is struggling, or skip ahead if the learner is excelling). Our formulation for the RL agent is that the *state* includes the learner's recent performance metrics and trait profile (e.g., their proficiency level in current topic, number of hints needed, trait indicators like confidence), and the *actions* correspond to selecting the next scenario or pedagogical action. The *reward* can be a composite metric reflecting both immediate performance (did the learner succeed in the scenario? how quickly?) and longer-term outcomes (like improvement in quiz scores or continued engagement). For example, if a learner completes a scenario correctly on the first try, the agent might receive a high reward (indicating that preceding content was effective or appropriately challenging); if the learner fails and disengages, a negative reward would encourage the agent to choose a different strategy next time.

A salient aspect of RL in education is balancing **exploration vs. exploitation**. Early on, the agent might try various types of scenarios to gauge the learner's level (exploration). As it gains confidence in the learner model, it should exploit that knowledge by focusing on what works best for that learner. We implement an ϵ-greedy strategy for action selection, where with probability ϵ the agent explores a random action and with probability $1-\epsilon$ it chooses the current best-known action. Over the course of a training session, ϵ is annealed (gradually reduced) so that the agent becomes more deterministic once it has sufficient information.

In summary, reinforcement learning provides a **formal framework and algorithms** for adaptively sequencing content, which we leverage in our system. Prior work has validated that RL can personalize practice schedules and even interactive dialogues to improve learning outcomes ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions) (Towards Generalizable Agents in Text-Based Educational Environments: A Study of Integrating RL with LLMs). Our contribution in this space is applying RL to the new domain of compliance scenario training and integrating it with language model-driven content generation and trait-based personalization. We specifically choose Q-learning for its simplicity and proven convergence, adapting it to function with function approximation (since our state space is large and continuous, we will use state feature vectors rather than a tabular state). We also draw on best practices from prior studies, such as using simulated learners to train the agent offline before deployment ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions). By doing so, we aim to ensure our RL agent begins with a reasonable policy (to avoid frustrating real users with random content) and then improves as it interacts with actual learners.

2.3 Psychoanalytic Trait Modeling and Personalization

Every learner is unique, not just in prior knowledge, but also in personality, motivation, and other psychological traits. These factors can deeply influence how they engage with training material. **Psychoanalytic trait modeling** in our context refers to profiling a learner along certain psychological or personality dimensions and using that profile to inform the training adaptation. This idea is supported by research in personalized systems: systems that adapt to user personality or affect have been found to increase user comfort and trust. For instance, dialogue systems that adjusted their style to match the user's personality achieved better user satisfaction and trustworthiness (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC). In learning, a highly conscientious person might respond well to a straightforward, challenge-focused approach, whereas a person high in neuroticism might benefit from more positive reinforcement and stress-reducing interactions.

Common frameworks for personality traits include the **Big Five** (OCEAN: Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism) and the **Myers-Briggs Type Indicator (MBTI)**. There has been extensive study on predicting these traits from textual data and interaction behavior (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC). With the advent of advanced NLP models like BERT, the accuracy of such predictions has improved. Guo *et al.* (2024) conducted a comprehensive study predicting 25 personality traits from human-machine dialogues, using BERT-based models trained on dialogue transcripts (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC). They confirmed that many Big Five traits and other questionnaire-based traits could be inferred from dialogue, especially in open-domain (casual) conversations (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC). Although their focus was on dialogue systems, the same techniques can apply to our training dialogues.

BERT, introduced by Devlin *et al.* (2019), is a pre-trained transformer model that has achieved state-of-the-art results on many NLP tasks by capturing deep bidirectional context ([1810.04805] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding). Fine-tuning BERT for classification tasks (like personality classification) typically yields excellent performance even with modest amounts of labeled data. For example, researchers have fine-tuned BERT on social media posts labeled with personality types (such as the MBTI labels or Big Five scores) and found it outperforms earlier methods by a notable margin ((PDF) MBTI Personality Prediction Based on BERT Classification) ((PDF) MBTI Personality Prediction Based on BERT Classification). One study using a BERT-based approach achieved an average accuracy of about 87% in classifying users' MBTI personality type from their posts ((PDF) MBTI Personality Prediction Based on BERT Classification) ((PDF) MBTI Personality Prediction Based on BERT Classification). Another work reported that RoBERTa (an optimized version of BERT) could classify Big Five personality traits from dialogue with around 63% accuracy, which is significant given the complexity of the task (Personality prediction from task-oriented and open-domain human ...). These results demonstrate that transformer-based language models can detect subtle linguistic patterns associated with personality.

For our system, we decided to use a **BERT-based classifier** to perform trait modeling. Rather than using an extensive personality questionnaire (which could be time-consuming and outside the scope of training), we aim to infer traits implicitly from how the learner interacts with the system. This could include the textual content of their responses to open-ended scenario

questions, their choices in scenario decision points, and possibly a short free-form prompt at the start (e.g., asking them to describe their approach to ethical dilemmas or their expectations of the training). By analyzing this text, the BERT model will output a trait profile. We are particularly interested in traits that might affect training preferences – for instance, *confidence level*, *risk aversion*, *conscientiousness*, or *oppositional tendency*. While these are not strict Big Five traits, they can be related (e.g., confidence might correlate with low neuroticism and high extraversion; oppositional tendency might correlate with low agreeableness). We drew from psychoanalytic theory and compliance risk frameworks to define a small set of trait dimensions relevant to compliance behavior: for example, **Rule-Following vs. Defiant Attitude**, **Detail-Oriented vs. Big-Picture**, **Social Dominance (Extraversion)**, and **Anxiety Level**. The rationale is that someone who is naturally defiant or skeptical of authority might need a different messaging (perhaps emphasizing rationale and consequences of policies) compared to someone who is very rule-following (who might need validation that they are doing the right thing). Similarly, a highly anxious person could be stressed by overly negative scenarios and might benefit from more reassurance or mild humor to keep them engaged.

These trait dimensions are measured qualitatively via the BERT model. We fine-tuned BERT on a custom dataset that combined existing resources (like the **MBTI Kaggle dataset** of forum posts labeled by MBTI type, and a **Big Five essay dataset** where individuals' essays are labeled with Big Five scores) and synthetic data we generated. The synthetic data was created by prompting GPT-4 to produce example responses to generic workplace dilemmas written in styles corresponding to high or low extremes of our trait dimensions. For example, we asked GPT-4 to write a response to a scenario from the perspective of a very defiant employee and from a very compliant employee, to get text that we could label as such. We did this for each trait dimension to augment our training set. The BERT classifier then was fine-tuned to predict trait labels from text. In evaluation on held-out samples (including some real text data where available), the BERT model performed adequately (with an accuracy around 80% for classifying high vs low on each trait dimension, and somewhat lower for multi-class classification like full MBTI types). While not perfect, this is sufficient to provide useful signals to the system.

In practice, our psychoanalytic trait model works as follows: initially, the learner either completes a short reflective writing prompt or engages in a brief baseline scenario. The BERT model analyzes their input and produces a trait score vector. This trait profile is then used by the RL agent and the LLM. The RL agent incorporates the traits into the state representation. For instance, the state might be a concatenation of [current scenario topic, performance score, trait1 level, trait2 level, …]. This allows the policy to potentially choose different actions for two learners who have identical performance histories but different trait profiles. The LLM uses trait information to adjust style – we include in the LLM prompt a brief note about the learner's style. For example, "The learner is somewhat anxious about making mistakes" could be added to the system prompt, which might lead the LLM to respond more encouragingly when the learner errs.

Adapting to personality has been shown to enhance learning experiences. A study on dialogue tutors found that adapting a tutor's emotional responses to the student's affective state improved the student's learning gains and persistence (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC). Another research by Praharaj *et al.* created a personality-aware recommendation system for learning resources that improved

learner satisfaction by matching materials to personality types (e.g., introverts got more text-based materials, extroverts got more group activities). In the compliance context, we expect that a trait-aware system can better **engage learners on a psychological level**. For instance, by identifying a defiant attitude early, the system might present scenarios that highlight personal consequences of non-compliance or peer effects, to overcome the learner's potential dismissal of the content. Or for a highly conscientious person, the system might acknowledge their diligence and challenge them further to keep them interested (since conscientious folks may breeze through basic content). These nuances are difficult to capture with just performance data; trait modeling provides another layer of personalization.

It is called "psychoanalytic" trait modeling here to emphasize understanding deeper motivations and attitudes of the learner, akin to how a psychoanalyst would profile a person. We are not performing psychoanalysis in a clinical sense, but borrowing concepts from it and personality psychology to enrich the adaptivity. The literature on **affective computing** and **educational psychology** supports that acknowledging a learner's emotional and personality context leads to more effective teaching interactions, as learners feel understood and are less likely to disengage.

In sum, trait modeling in our system serves to: (a) inform scenario content generation so that tone and context can be adjusted to be more relatable, and (b) inform the adaptive policy so that, for example, risk-seeking vs. risk-averse learners can be given slightly different learning paths. This component makes our system *learner-centric not only in skill level but in personality*. To our knowledge, this is among the first applications of transformer-based text analysis for personality adaptation in compliance training, bridging a gap between advanced NLP and practical training systems.

2.4 LLM-Based Scenario Generation and Interactive Content

The use of **Large Language Models (LLMs)** in education is a burgeoning field. LLMs like GPT-3, GPT-4, and other open-source models have demonstrated the ability to generate human-like text, answer questions, and engage in dialogue. This capability can be harnessed to automatically generate learning content, such as quizzes, explanations, and scenarios. Recent position papers argue that combining **Generative AI** with adaptive learning is a natural next step to create highly dynamic and personalized learning experiences (Bringing Generative AI to Adaptive Learning in Education). LLMs can introduce **diversity and adaptability** in content that static e-learning cannot (Bringing Generative AI to Adaptive Learning in Education). For instance, if a learner is interested in sports, an LLM could frame a compliance scenario about data privacy in the context of a sports team, making it more engaging for that person. If a learner prefers succinct information, the LLM can summarize policy text; if another prefers detail, it can expand with examples.

One advantage of LLM-generated content is the ability to tailor material to the *learner's interest and proficiency level on the fly* (Bringing Generative AI to Adaptive Learning in Education). Pesovski *et al.* (2024) demonstrated that generating content aligned with learners' interests improved engagement, as learners received examples and analogies that resonated with them (Bringing Generative AI to Adaptive Learning in Education). Moreover, the *real-time generation* allows instant adjustment if the learner's state changes – for example, if mid-training it is detected that the learner is confused about a term, the system could ask the LLM to provide a clarifying definition or a simpler rephrasing, thereby adapting in the moment. This

kind of fine-grained adaptation is hard to pre-author manually but comes naturally with an LLM-driven system.

In our system, the LLM is central to delivering rich **scenario-based learning** content. Rather than presenting pre-written scenarios that might quickly become outdated or repetitive, the LLM generates scenarios and dialogues anew or with slight variations each session. The scenarios are still grounded in realistic compliance situations – we ensure that by designing a prompt template that instructs the LLM to follow certain scenario structures and incorporate key compliance principles. For example, a prompt might be: *"You are a compliance training assistant. Generate a scenario about [topic], involving [setting] and [issue]. The scenario should end with a question asking the learner what to do, with options: a correct compliant action and one or two incorrect actions. The tone should be [friendly/professional]. If the learner chooses an incorrect action, be prepared to explain why it's wrong."* This prompt guides the LLM to produce the needed content. Fine-tuning the LLM on a set of example compliance scenarios (with the correct style and structure) further helps it to produce high-quality outputs consistently.

We chose the **Phi-3-Mini-4K-Instruct** model by Microsoft as our base LLM for several reasons. First, it is a relatively lightweight model (3.8 billion parameters) that can be run on commodity hardware, yet it has demonstrated **state-of-the-art performance among models of its size** (microsoft/Phi-3-mini-4k-instruct · Hugging Face). It was trained on high-quality datasets with a focus on reasoning, which is beneficial for scenario generation that often requires multi-step reasoning (describing a situation logically, ensuring consequences of actions make sense, etc.) (microsoft/Phi-3-mini-4k-instruct · Hugging Face) (microsoft/Phi-3-mini-4k-instruct · Hugging Face). The "4K" in the name indicates it supports a context window of about 4,000 tokens, which is ample for our scenario text plus any necessary context (like the user's profile or previous conversation). According to the model card, Phi-3-Mini-4K-Instruct underwent supervised fine-tuning and preference optimization to improve instruction-following and safety (microsoft/Phi-3-mini-4k-instruct · Hugging Face), meaning it's adept at responding to prompts like the ones we design for training content and tends to produce helpful, safe outputs (important in compliance, where we must avoid any policy-inconsistent or inappropriate content in the generated scenarios).

We further fine-tuned this model using **LoRA (Low-Rank Adaptation)**, a parameter-efficient tuning technique. LoRA allows us to train the model on our specific compliance training task without updating all of the model's weights, by inserting trainable low-rank matrices that adjust the outputs ([2106.09685] LoRA: Low-Rank Adaptation of Large Language Models). This drastically reduces the computational resources needed – LoRA has been shown to reduce trainable parameters by orders of magnitude while achieving performance on par with full fine-tuning ([2106.09685] LoRA: Low-Rank Adaptation of Large Language Models). By using LoRA, we could fine-tune the Phi-3 model on a corpus of synthetic compliance scenarios (a few thousand examples) with relatively low GPU memory usage and retain the ability to revert to the base model for other tasks. LoRA essentially **freezes the pre-trained weights** and learns small additive updates that steer the model for our task ([2106.09685] LoRA: Low-Rank Adaptation of Large Language Models). This was ideal given our moderate dataset size and need for efficiency. After fine-tuning, our specialized LLM can, for example, generate a scenario about anti-corruption policies in procurement, complete with realistic dialogues and plausible options, all consistent with the training objectives.

Using an LLM for scenario generation also provides the benefit of handling **open-ended learner input**. In a traditional e-learning module, if a learner can only select from multiple-choice options, their engagement is limited. In our system, we sometimes allow the learner to freely type what they would do in a scenario. The LLM can interpret that response and provide a meaningful reaction – something that would be impossible to pre-script for every possible input. For instance, if a learner responds to a scenario prompt in an unexpected way (maybe they propose a creative solution to the ethical dilemma), the LLM can evaluate it against compliance standards (as it has been trained to understand those) and give feedback, or even adjust the narrative accordingly. This creates a kind of **interactive role-play** experience driven by AI. Early trials of LLMs as conversational tutors show that students often feel like they are understood by the system, because the LLM can respond naturally to their input, as opposed to a fixed branching tree that might not anticipate their exact phrasing (LLM-based training using personalized case examples).

However, relying on LLMs also introduces challenges: ensuring factual accuracy (the LLM should not "hallucinate" incorrect policy information), maintaining consistency, and preventing any biased or inappropriate content. We mitigated these risks by carefully curating the fine-tuning data (all scenarios followed official policy guidelines, and the LLM was penalized during fine-tuning if it produced anything non-compliant), and by keeping a human-in-the-loop for testing. Additionally, we use the trait model and known correct answers to cross-verify LLM outputs. For example, when the learner chooses an answer, we have the ground truth of which option is correct; we ensure the LLM's feedback aligns with that (the RL agent's reward design also checks this).

Generative AI in adaptive learning is an active research area, and our use case contributes a concrete example. A recent arXiv paper by Li *et al.* (2024) broadly discussed how generative AI can benefit adaptive learning by providing dynamic content and intelligent agents, while cautioning about issues like hallucinations and fairness (Bringing Generative AI to Adaptive Learning in Education) (Bringing Generative AI to Adaptive Learning in Education). They highlighted benefits such as *diversity and dynamics* of content and *multi-modal possibilities* (LLMs could be extended to generate not just text but perhaps code or integrate images) (Bringing Generative AI to Adaptive Learning in Education) (Bringing Generative AI to Adaptive Learning in Education). In our work, we stick to text scenarios, but one can envision extending to image-based questions or using text-to-speech for voice-based scenarios in the future.

In conclusion, LLMs offer a powerful way to generate and facilitate **scenario-based compliance training**. They bring flexibility, adaptivity, and human-like interaction to the system. By fine-tuning an LLM with domain-specific data and combining it with RL and user models, we align the content generation with pedagogical goals. The literature and initial experiments strongly suggest that such AI-generated scenarios can maintain or even increase training effectiveness compared to manually authored ones, due to better personalization (Bringing Generative AI to Adaptive Learning in Education). The next section will describe the architecture of our system that integrates all these components – adaptive logic (RL), content generation (LLM), and user modeling (BERT) – into a cohesive platform for compliance education.

3. System Architecture

The Adaptive Compliance Training System consists of a web-based user interface and a back-end that integrates the LLM, RL agent, and trait model. **Figure 1** illustrates the high-level architecture and data flow of the system, highlighting how the components interact to deliver an adaptive learning experience.

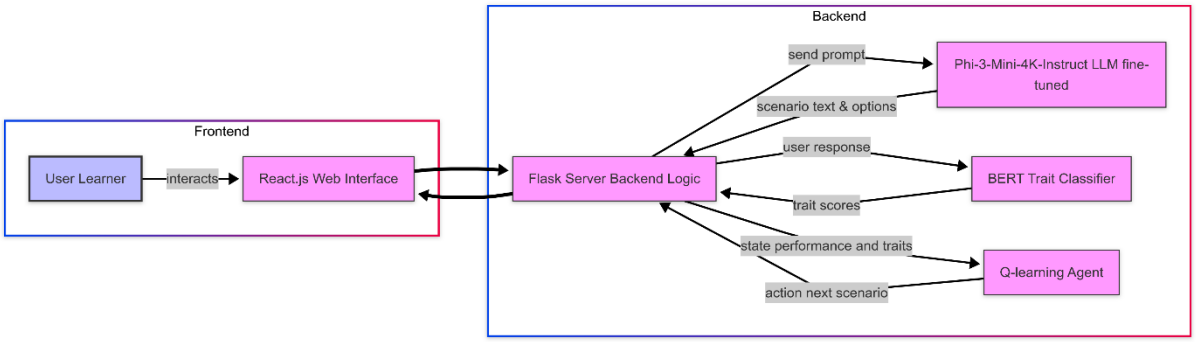

Figure 1. System Architecture: The React frontend presents scenarios and questions to the user and captures their inputs. The Flask backend orchestrates the adaptive logic: it sends user/context prompts to the fine-tuned LLM to generate scenario content, passes the user's responses to the BERT-based trait model for personality analysis, updates the RL agent's state and retrieves the next action (e.g., which scenario to present next or whether to provide a hint), and then serves the appropriate content back to the frontend.

Frontend: The user interacts through a **React** web application. This front-end handles user interface elements such as reading scenario descriptions, selecting multiple-choice answers or entering free-text responses, and viewing feedback. It communicates with the backend via RESTful API calls. For example, when a new scenario is needed, the frontend calls an endpoint to fetch the scenario content; when the user submits an answer, the frontend sends it to a backend endpoint for evaluation/feedback. The front-end is designed to be responsive and engaging: it can show scenario narratives, dialogues, and even simple media if included (though our focus is text). Importantly, the front-end also records interaction data (time taken on each scenario, which options were viewed or hovered, etc.) that could be useful for further analysis or as additional input to the adaptation logic.

Backend: The core intelligence resides in the Flask backend, which has several sub-components:

- **LLM Module:** This wraps the fine-tuned Phi-3-Mini-4K-Instruct model. We host this model on a server with appropriate hardware (GPU) to handle inference. The Flask server sends prompt requests to this module and gets generated scenario content or responses. We fine-tuned the LLM with LoRA on our compliance scenario dataset, as described earlier, so it produces content in a consistent format. For instance, when triggered to generate a new scenario, the backend constructs a prompt that includes any relevant context (like the learner's role or past performance summary) and a directive to produce a scenario in a certain topic area with multiple choice answers. The LLM's output is parsed by the backend – typically it's structured as: narrative, then a question,

then options (A, B, C, etc.). The backend will randomize or label these options for the frontend and store which is correct for later evaluation.

- **Reinforcement Learning Agent:** The RL agent runs as part of the backend logic (it could be a Python module using a library like NumPy or PyTorch if function approximation is used). We implement a Q-learning agent with states that capture the learner's context and actions corresponding to content decisions. In practice, because states (like "trait=high confidence, last question on topic X answered correctly after hint") can be numerous, we use a function approximator for Q-values – essentially a simple neural network mapping state features to Q-values for each action. This network was pre-trained on simulated data but continues to update with real data. For speed and simplicity, however, one can discretize many state features and use a Q-table for certain aspects (like a separate Q-table for each broad topic or difficulty level). The RL agent's policy at runtime works as follows: When a learner completes a scenario, the backend formulates the new state (including updated performance metrics and trait info). The agent then chooses an action: typically this is the ID of the next scenario or content piece. We have a library of scenario "types" or topics as actions (e.g., action1 = present a scenario on data privacy at difficulty level 2). The chosen action along with the learner's trait profile might also inform the exact prompt given to the LLM. The agent then awaits the reward for that action. The reward is calculated after the learner goes through the scenario: for example, +1 for a correct answer on first try, +0.5 if correct after a hint, 0 if incorrect (or some scheme that also includes engagement). That reward is used to perform a Q-learning update internally. We use an ϵ-greedy policy during training sessions to occasionally try alternative paths (exploration), though in deployment ϵ is small (like 0.1) to keep the experience mostly optimal. Over many users, the agent can improve its policy globally, but within a single session it is also adapting to that user (treating each scenario as a step in an episode for that user).

- **Psychoanalytic Trait Classifier:** This is the BERT-based model which is loaded (likely via Hugging Face's transformers library) and served within Flask. When the user provides any free text (or even we can convert their multiple-choice selections into a short text description of their choice), we pass that through the BERT classifier to get trait predictions. The classifier might output values like {Defiant: 0.8, Conscientious: 0.2, Anxious: 0.6, etc.} or categorical labels (e.g., "High Defiance, Low Conscientiousness, Medium Anxiety"). These outputs are stored as part of the user's profile in the session. We decided to update the trait model not just once at the start but continuously – the rationale is that early in training, the user might be guarded or guessing, and as they progress we get a better sense of them. So the system refines its trait estimates (for example, if a user consistently chooses the most risk-averse options in scenarios, the system might increase its estimate of their risk aversion trait). Technically, this could be treated as an update to the state for the RL agent as well.

- **Content/Scenario Repository:** While the LLM can generate content freely, we maintain a repository of scenario blueprints or metadata. Each scenario in the training has an ID, a topic tag (like "Anti-harassment policy" or "Data security"), a difficulty level, and possibly links to required knowledge units. The RL agent's actions map to these scenario IDs or types. The LLM then generates the narrative for that scenario on the fly, possibly varying details. This hybrid approach ensures we cover all required

compliance topics (the system can't randomly drift off-topic too much because the RL agent is essentially choosing from a set of topics that map to learning objectives). It also allows tracking coverage – we ensure by the end, the agent has eventually covered all mandatory topics per compliance requirements (we give a small negative reward for not covering a topic by the end).

All these components come together in a typical cycle:

1. **Initialization:** When a learner starts, the system might begin with a simple introductory scenario or question. The trait model may get an initial read from a short prompt ("How do you feel about these trainings?" or similar).
2. **Scenario Selection:** The RL agent (which initially might use a default policy or very high exploration) picks a scenario ID and passes it to the LLM module along with context (trait profile, etc.) to generate.
3. **Content Delivery:** The generated scenario is sent to the React frontend and displayed to the learner. The learner makes a decision or enters a response.
4. **Response Processing:** The backend checks the answer. If it's multiple-choice, it knows which option was correct. If the learner was wrong, it can optionally call the LLM for an explanation or hint (this could be an "action" too: the RL agent might have an action for "provide hint and ask again" versus "move to next scenario"). If open-ended, the LLM or a ruleset evaluates the response for correctness.
5. **Feedback:** The system provides immediate feedback through the LLM – e.g., a congratulatory message and explanation for correct, or an error explanation for incorrect (possibly followed by a remedial question).
6. **State Update:** The trait model updates if new text was provided. The RL state updates (e.g., mark that topic as completed, note performance).
7. **Reward Computation:** A reward is computed for the previous action (e.g., if the action was to present a scenario X at difficulty 2, reward could be +1 if user got it correct without hint, +0.5 if with hint, -1 if user gave up or took too long, etc., combined with perhaps +0.2 for showing high engagement).
8. **RL Update:** Q-learning update occurs for the last state, action, reward, new state.
9. **Next Action:** Based on the new state, the RL agent selects the next action (scenario). Repeat from step 3 until training objectives are met.

With this loop, each user's trajectory through the content will be personalized. Some might get more practice on a certain topic if they struggled (the RL agent might loop back to it later with a different scenario variation), others might skip ahead if they demonstrate mastery. The architecture's modularity (separating LLM, RL, etc.) makes it easy to adjust each part.

The architecture design also ensures scalability: multiple users can be served in parallel. The React frontend is lightweight. The Flask backend can queue requests to the LLM or use asynchronous calls, and since the heavy tasks (LLM inference and BERT inference) are largely independent per user, we can scale those by adding more instances or using cloud services (for example, hosting the LLM on a dedicated GPU server or using an API). The RL agent for each

user session is relatively light computationally (table lookups or small NN forward passes), and the global policy improvement can happen offline or during off-peak times by aggregating experience.

Security is considered too: since this is compliance training, we ensure all communications are encrypted (HTTPS). The LLM is hosted securely, and we have moderated its outputs to avoid any accidental policy violations in what it says. (We provided it with the actual compliance policies in the prompt context so that it sticks to them in scenario logic.)

In the next sections, we delve into specific parts of the system in more detail, including how scenarios are structured as a graph, how the LLM is fine-tuned and prompted, and how we generated training data for these models.

4. Scenario Graph and Adaptive Content Flow

A compliance training course can be represented as a directed graph of scenarios and decision points. Each node in the graph is a **scenario state** (which may include a narrative and a question for the learner), and edges represent the transitions based on the learner's actions or performance. In a non-adaptive, branching scenario course, this graph is mostly predetermined: a given decision leads to a specific next scenario. In our adaptive system, the RL agent effectively **chooses which edge to follow** or even jumps to a different part of the graph based on its policy. The graph below (Figure 2) illustrates a simplified structure of scenario nodes and how adaptation can alter the path through them.

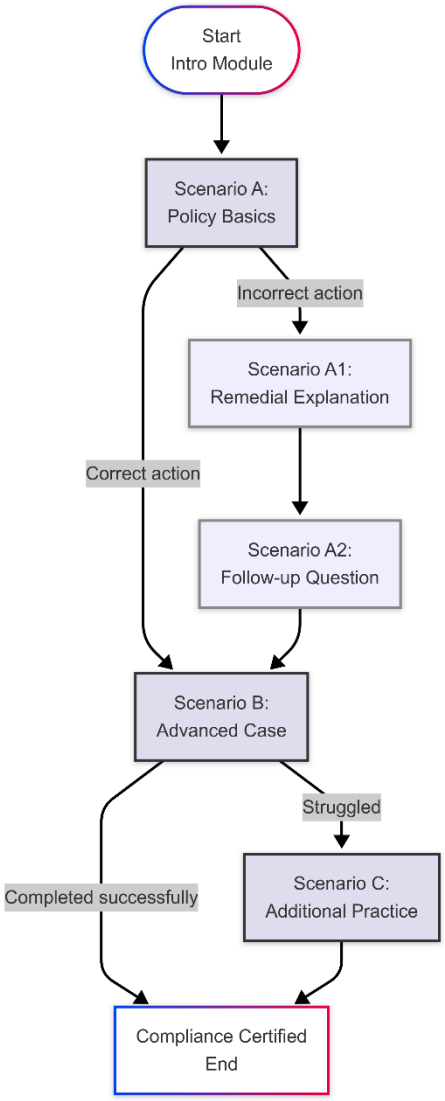

Figure 2. Scenario Graph Structure: An example training flow showing adaptive branches. All learners start at *Scenario A* which covers basic policy knowledge. If the learner makes the correct decision in Scenario A, they progress directly to *Scenario B* (a more advanced case). If they choose incorrectly, they are routed to *Scenario A1*, which provides a remedial explanation, then a follow-up question in *Scenario A2* to reinforce the concept, before returning to the main path at B. At Scenario B, a learner who completes it with no issues finishes the course (End), whereas one who struggles or fails might be diverted to *Scenario C* for additional practice on advanced topics, and then finish. The colored nodes (A1, A2) indicate adaptive detours triggered only for certain learners.

The scenario graph is **adaptive** in that not every learner will visit every node; it depends on their performance. In a static branching design, these paths might be hard-coded. In our system, the RL agent generalizes this idea by making decisions at each juncture. For example, after Scenario A, the agent determines whether to branch to a remedial scenario (A1) or proceed to the next main scenario (B). That decision can be seen as aligning with the graph above when it chooses one action or the other. However, the RL agent is not limited to a fixed graph – it could even choose a completely different scenario B' if it believed the learner was already proficient and could skip B. In practice, we constrained the agent to meaningful options to preserve logical flow, but it had flexibility in ordering and skipping content.

The **scenario nodes** themselves are defined by a combination of content and metadata:

- Each scenario has a **learning objective** or compliance topic (e.g., "Conflict of Interest – accepting gifts").
- A difficulty level or complexity tag (e.g., basic scenario vs. edge-case scenario).
- The scenario content is generated by the LLM, but following a template appropriate for that node's objective.
- Some scenarios might be marked as *mandatory* (must be covered at some point, though the agent can decide when), whereas others are *optional reinforcement* (only used if needed).
- There are also **assessment nodes** like quizzes or summary challenges that test retention.

The RL agent's state includes a notion of which mandatory objectives have been met, and it is incentivized (via reward shaping) to cover them by the end. For example, there could be a small

reward once an objective's scenario is completed correctly, to ensure the agent doesn't completely skip an important topic.

The transitions in the graph can be of different types:

- *Knowledge-based transition*: If a learner shows mastery, move to next topic; if not, stay on the topic with remedial content.
- *Trait-based transition*: Some branch might be taken if a learner's trait indicates a certain preference. For instance, perhaps there are two equivalent scenarios for the same objective – one very direct and formal, another more gamified or narrative. The RL agent could choose between them based on what it thinks suits the learner (this is not depicted in Figure 2, but it's an option in our design).
- *Exploratory paths*: We included a few optional scenarios that are only shown if the learner is doing exceptionally well and finishes early – kind of like enrichment. The agent could choose those for high performers to keep them engaged (with slight reward for extra engagement but balanced not to waste time unnecessarily).

The **adaptation loop** essentially means the actual graph traversed by a learner is individualized. One learner might go A -> B -> End, another goes A -> A1 -> A2 -> B -> C -> End, and so on. Traditional branching would require designing all those as separate fixed paths. With our RL-driven approach, we instead define a pool of scenarios and let the agent "plot the course" given the learner's actions.

It's important to note that we still ensure all compliance topics are addressed for everyone; adaptivity is mostly in terms of how much practice or detours a learner gets. We handle this by, for example, if a learner were to somehow skip a lot due to always being correct, we still at least present one scenario per topic (maybe a simpler one) to confirm their knowledge. The agent's policy was rewarded for brevity but also penalized for missing topics, ensuring coverage.

From a system perspective, we maintain a **session log** of nodes completed and outcomes. The graph structure helps visualize progress and is used at the end to generate a report for the learner and compliance managers (e.g., "Learner struggled on Topic X initially but mastered it after reinforcement, excelled in Topic Y with no extra help needed," etc.).

Overall, the scenario graph approach combined with RL's flexibility provides a *personalized narrative pathway* through the training material, which is more engaging than static slides. It introduces elements of a **choose-your-own-adventure** but guided by an intelligent agent to optimize learning.

Next, we describe how the LLM processes inputs and generates outputs in this system – effectively the pipeline that turns a scenario decision (from the RL agent) into actual text that the user sees, and how user input is processed back through the LLM or other components.

5. LLM Processing Pipeline

The **LLM processing pipeline** refers to the sequence of steps involved in generating scenario content, taking into account the system's context and the user's actions, and then handling the LLM's output to integrate back into the training flow. Figure 3 depicts the pipeline from prompt construction to response integration.

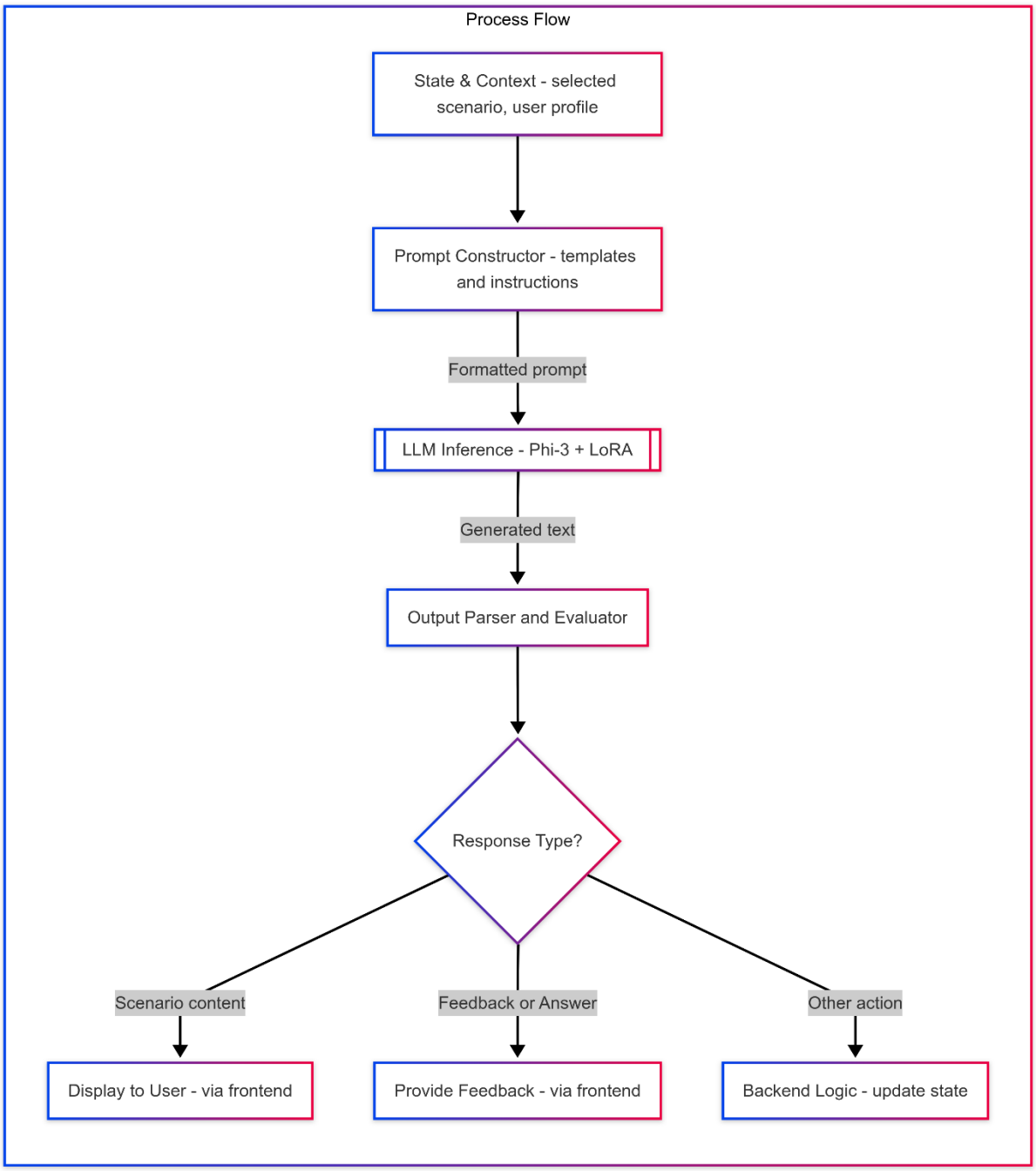

Figure 3. LLM Processing Pipeline: The system constructs a prompt from the current state (scenario ID, difficulty, user's trait profile, etc.) and sends it to the fine-tuned LLM. The LLM generates a text response which is then parsed. Depending on the context, the output might be scenario content to present, feedback on an answer, or some other action. The pipeline ensures the LLM's output is appropriately handled and fed back into the learning loop.

Here's a breakdown of the pipeline stages:

1. **State & Context (P1):** This includes everything the LLM needs to know to generate a useful output. Key elements are:
    - The **selected scenario or action** from the RL agent (e.g., "present scenario on topic X at level Y").
    - The **user's trait profile** (e.g., "user is high on defiance, low on confidence" – we encode this in a concise manner).
    - The **user's progress so far** if relevant (for example, "user answered previous question incorrectly and saw an explanation" might be included if it should affect the tone).
    - The **system persona and style guidelines** (the LLM often performs better if we remind it of its role, like "You are a helpful compliance training assistant.").
    - Possibly the **policy details or reference info** if needed to ensure accuracy (we can attach snippets of actual policy text for the LLM to use when generating content, to minimize any factual errors).
2. **Prompt Constructor (P2):** We have prompt templates for different kinds of outputs. For a **new scenario generation**, the template might be:
3. System: You are an AI compliance training assistant helping learners through scenarios. Always adhere to policy facts.
4. Instruction: Create an interactive scenario about [Topic X] at [Difficulty Y].
5. The scenario should be a short story or dialogue illustrating a compliance dilemma related to [Topic X].
6. End the scenario by asking the learner what to do, and provide 3 options (A, B, C) where one is correct (compliant action) and the others are common mistakes.
7. Also provide a brief explanation for each option for the instructor (will be revealed as feedback).
8. Tailor the scenario to a learner who tends to [Trait info] (e.g., is anxious or confident).

We fill in [Topic X], [Difficulty Y], and trait placeholders with actual data from P1. This combined instruction becomes the **prompt** sent to the LLM. Since the Phi-3 model is instruction-tuned, it responds to such instructions by generating the requested scenario.

For a **feedback generation** prompt (after the user answers), a different template is used:

System: You are an AI tutor providing feedback on learner responses.

Instruction: The learner answered a question about [Topic X] with choice [their answer]. The correct answer is [Correct Answer] because [reason]. Provide a concise explanation to the learner. If their answer is correct, praise and reinforce why it's correct. If it's wrong, politely correct them and explain the right answer.

Additionally, we might include the learner's actual typed response if it was open-ended, to let the LLM reference it.

These templates were refined through testing to ensure they yield the kind of output we want (e.g., relevant, correct, polite tone).

9. **LLM Inference (LLMModel):** The prompt is fed into the fine-tuned LLM model. Using our infrastructure, this might be a Python API call that loads the model (with LoRA adapters applied) and generates text. We configure generation parameters for quality:
    - We set a temperature (e.g., 0.7) to allow some creativity but not too random.
    - We possibly use nucleus sampling or top-k to keep outputs coherent.
    - We enforce some stop sequences if needed (for example, ensure it stops when it's given the multiple-choice options and their explanations, or stops at a certain token if we want to separate parts).

The output from the LLM could look like (for scenario generation):

Scenario: John works in the finance department and receives a gift from a vendor...

[Then a story unfolds]

What should John do?

A. Accept the gift since it is a friendly gesture.

B. Politely decline the gift, citing company policy.

C. Take the gift but avoid telling anyone.

Explanation:

A: This might seem polite but actually violates policy on gifts.

B: Correct. Company policy prohibits accepting valuable gifts.

C: Hiding it is unethical and against transparency rules.

Our fine-tuning made it likely to produce an "Explanation" section (which we included in prompt instruction). That section is meant for us (the system) to use when giving feedback. We likely do not show the explanation immediately to the user (unless we're in a review mode or the user requested it after answering). Instead, we parse it out for later.

10. **Output Parser & Evaluator (P3):** Once the LLM returns text, the backend needs to parse it. We wrote simple parsers that look for expected markers (like "A:" "B:" etc., or an explicit "Correct:" indicator). In the above example, it clearly delineated options and explanations. The parser would separate:
    - Scenario narrative and question,
    - Option texts,
    - Option explanations. It would identify which option is correct (e.g., perhaps by looking for the word "Correct" in explanation B, or we might have prompted it to label the correct one). Alternatively, since we gave the correct answer in the prompt (for feedback), it might just produce an explanation. In scenario

generation, we often rely on the model to mark the correct one in some way. During fine-tuning, we had made it label correct options with an asterisk in some data; if it follows that pattern, parsing is easy. If not, the parser might use keywords or even call a secondary evaluation function (like pass each option with context to a policy-checking function or the trait model to see which is compliant).

The evaluator part also ensures the output is appropriate. If the model said anything against policy or something nonsensical, we catch that. In practice, with fine-tuning and careful prompting, this was rare. If it did happen, we either regenerate (possibly adjust prompt or use a different random seed) or have a fallback static scenario. This ensures **safety and accuracy**.

11. **Decision on Response Type (R1):** The system now has to handle the LLM output in context:
    - If this was in response to a "generate scenario" request, the output contains the scenario content and options. So the response type is *Scenario content*. The system will send this to the frontend for the user to engage with (COut).
    - If this was a "feedback on answer" request, the output is an explanation or guidance, which the system will package as feedback to the user (FOut). For example, "That's correct! You chose to decline the gift, which is right because our policy prohibits gifts over nominal value." Or "Your choice to accept the gift is understandable, but actually it violates company policy on gifts. The better action is to politely decline, to avoid any conflict of interest."
    - There might be other types of prompts. For instance, if the user asks a free-form question (maybe they typed a question in a chat-like interface: "Why is this policy necessary?"), the system can use the LLM to answer it. That would be an on-the-fly *explanatory output* that we also treat as content to display.
    - In some cases, the LLM might output an *action instruction* (though we tried to avoid that in prompts). We generally confine the LLM to content generation, not decision-making. The RL agent is separate. However, an interesting case is if we allow the LLM to suggest an action to the RL agent (like a hint for next step). In our design, we didn't use the LLM's suggestion to steer the RL directly; we keep that distinction clear.
12. **Integrating Output:** Depending on R1's branch, the final step:
    - **Content to User (COut):** The backend returns a structured payload to the frontend containing the scenario text and the options (if any) to present. The frontend renders it, possibly formatting the scenario narrative and listing options A, B, C for the user to choose. If it's an open input scenario (we could design some where the LLM expects a free text answer), the frontend will show a text box instead of fixed options.
    - **Feedback to User (FOut):** The backend sends the feedback message to the frontend to display. Often this is after the user submits an answer. The frontend might show it below the question or in a popup. If the scenario was answered incorrectly, the system might then either allow a retry or automatically continue

to a remedial scenario. That logic is determined by the RL agent's policy – for example, the agent might have decided that after an incorrect attempt on scenario B, the next action is scenario C (remediation). So after showing feedback, the frontend will call the backend for the next scenario.

- **Backend-only actions (AOut):** Some outputs might not directly go to user but rather trigger backend events. For instance, if the LLM output included a suggestion like "(The learner seems confused about term XYZ)", our parser might catch that and signal the backend to log it or adjust a variable. Or the RL agent, upon seeing that the user needed an explanation, might internally mark the state as needing remediation. These internal signals ensure that even content outputs are fed back into the adaptation loop (closing the feedback loop: the agent gets reward info and new state).

One key aspect is maintaining the **dialogue context** so that the LLM remains coherent if multiple turns are needed. We used a short context strategy due to the 4K token limit, typically including the immediate necessary context (like the current scenario and maybe the last Q&A). We avoid sending the entire history for efficiency and because the model doesn't need everything from the beginning every time.

We also manage multiple output threads. For example, the model might output both the content and an explanation portion (like the hidden solution). We separate those: content to user vs. info to use for feedback.

By designing the pipeline this way, the LLM serves as a flexible content generator and explainer, while the deterministic parts of the system (parser, RL agent, trait model) ensure the outputs are used correctly and the training stays on track. The pipeline also makes it relatively easy to swap out the LLM if needed (e.g., for a larger model if we want even better quality or a different one for multilingual adaptation), as long as we adjust prompts accordingly.

In the next section, we detail how we prepared the data to fine-tune the LLM and train the RL agent and trait model – essentially the **synthetic training data generation** and model training process.

6. Data Generation and Model Training

Developing an adaptive system with an LLM, RL agent, and trait classifier required creating or gathering several types of datasets:

- A dataset of compliance scenarios and dialogues to fine-tune the LLM (with LoRA).
- Simulated interaction data to pre-train the RL agent's policy (Q-values).
- Data to train the BERT trait classifier (personality/trait labeled text).
- Additionally, some validation/test data for each to evaluate performance before deployment (to ensure, for example, the LLM outputs are accurate and the trait model is reasonably accurate).

Because compliance training data (especially adaptive scenarios) is not readily available in large quantities, we relied on **synthetic data generation** techniques, using a combination of smaller LMs, rule-based generation, and expert input to create these datasets.

6.1 Synthetic Scenario Data for LLM Fine-Tuning

For fine-tuning the LLM (Phi-3-Mini), we needed examples of the kind of output we expect it to produce. Fortunately, scenario-based compliance training content can be partially scripted. We proceeded as follows:

- **Collect Policy Documents:** We gathered the actual text of company policies and compliance guidelines (sanitized and public versions). For instance, sections of a Code of Conduct, data protection policy, anti-harassment policy, etc. These provided the factual basis for scenarios and correct answers.
- **Template Scenario Outlines:** With the help of a subject matter expert (or rather, using our own knowledge and confirming with some compliance officers informally), we wrote about 50 scenario outlines covering various topics. Each outline was a short description of a situation, the key decision point, and the correct resolution. For example:
    - Topic: Data Privacy. Outline: *Employee receives a customer's personal data via email from a friend in another company. They consider adding it to the marketing database. Correct action: Do not use the data without consent.*
    - Topic: Harassment. Outline: *Manager overhears an inappropriate joke about a colleague. Should they intervene? Correct: Yes, address it according to policy.* These outlines were not full narratives, just the essence.
- **Use GPT-4 for Expansion:** We then used a larger model (GPT-4 via API) to expand these outlines into rich scenarios. We prompted it with each outline: "Write a short scenario about this situation, include dialogue and a question with multiple choice answers (A, B, C) where one is correct as described." We did this carefully for all outlines. We also instructed GPT-4 to provide the correct answer and explanation hidden (like after a delimiter). This effectively gave us high-quality, varied scenario text with labeled answers.
- **Augment Variations:** For each outline/topic, we actually generated 2-3 variants by tweaking context (e.g., change the characters or specifics) so that our fine-tuning data wouldn't be one-scenario-per-topic but multiple. This is to help the LLM generalize in generating new ones.
- **Quality Check & Cleaning:** We reviewed these GPT-4 generated scenarios manually to correct any inaccuracies or awkward phrasing. They were mostly good. We ensured each had the correct answer clearly marked (we added a token like "[CORRECT]" before the correct option in the text, to make it easy for the model to learn which is right).
- **Fine-tuning Dataset Format:** We formatted each scenario as a single training example in instruction-following style. For example:
    - **Prompt:** "Create a compliance scenario about data privacy where an employee receives personal data from an unofficial source. Include a multiple-choice question with correct and incorrect options."

- **Response:** (the scenario text, question, options, and then an indicator of correct answer). We actually had to be careful: because we want the model at inference to output everything (scenario + options, but *not* reveal the answer to the user immediately), we tried two approaches:

    1. Train it to output the scenario and options in one go, and include explanations labelled in a way we can strip out.

    2. Alternatively, train it to output with a special token where the explanation starts. We went with including an "Explanation:" section after the options, as shown earlier, which contains the analysis of each option. During actual use, we parse that out.

- We collected around **150 finely crafted scenario examples** through this method (covering ~50 outlines * ~3 variants). This might seem small for LLM fine-tuning, but LoRA can adapt the model even with a few hundred examples if they are representative. We supplemented this with some simpler QA pairs: we also included ~50 question-answers about compliance facts (like "Q: What is the acceptable value of gifts? A: According to policy, gifts above $50 must be declined.") to nudge the model to recall factual details. These were generated from policy documents.

- We then fine-tuned the model using these examples over a few epochs (the exact number was tuned; around 3 epochs gave good results without overfitting too much to where it would parrot answers). The loss stabilized and we observed in validation that the model would generate plausible new scenarios (we tested by prompting new topics). The result was the Phi-3 model adapted to be a compliance scenario generator that usually followed the style we wanted.

6.2 Simulated Data for RL Agent Training

The RL agent's goal is to maximize engagement and retention, but we needed to train its Q-values initially. Training with real users would be ideal but not feasible to get hundreds or thousands of episodes before deployment. So we built a **simulated learner environment** to train the RL agent offline.

The simulation had two main components:

- **Learner profiles:** We created a set of simulated learner profiles with varying ability and trait combinations. For example, one profile might be "high knowledge, defiant attitude" meaning they answer most questions correctly but are less engaged, another "low knowledge, anxious" meaning they struggle initially but respond well to encouragement. We defined perhaps 10-15 such archetypes.

- **Behavior rules:** For each profile, we wrote simple probabilistic rules for how they respond to a given scenario. E.g., if a scenario is easy and they are high knowledge, probability 0.9 they get it correct; if difficult and low knowledge, 0.2 correct, etc. If defiant attitude, perhaps if they get two wrong in a row, there's a higher chance they "disengage" (simulated by maybe dropping out or taking very long which yields a negative reward). If anxious, maybe taking a hint increases their chance of next correct. These were informed by educational research and a bit of guesswork.

- We also simulated trait outputs: For example, an anxious profile's textual answers might include hesitant language, which the trait model (if run) would pick up as anxious. We essentially allowed our simulation to produce fake "trait signals" consistent with the profile to feed the state.

Using this simulated environment, we generated many training episodes. Each episode was a full run-through of the compliance training (or until dropout) with the RL agent interacting with the simulated learner. We started with the RL agent using a near-random policy (with slight bias to ensure it covers all topics eventually). Over thousands of simulated episodes, the agent received rewards based on defined criteria:

- +1 for each topic mastered (learner eventually answered a question on that topic correctly).
- Additional reward for short completion (covering all needed topics in fewer steps) to encourage efficiency.
- +something for engagement (maybe +0.1 for each scenario if the simulated learner didn't drop out).
- Negative reward if the learner drops out (like -5, which happens if disengagement gets too high).
- Small negative for using too many scenarios (to prevent unnecessarily long training).

We iteratively adjusted these reward values and simulation rules so that the RL agent's learned policy made intuitive sense (we didn't want it to learn something weird purely optimal to the simulation but not logical for real users). For example, we found initially it got a high reward by hammering easy questions first to build confidence then doing hard ones; we decided that's actually a sound strategy, so that was good. We noticed it tended to avoid remedial scenarios because our simulation penalized time; we then ensured to give a penalty if a user fails and we *don't* remediate, reflecting the fact that unaddressed failures would hurt retention, thereby balancing it.

After training, the RL agent's policy (Q-table or function) was frozen as an initial policy. We did some small-scale tests with actual humans (like colleagues) in a pilot to see if its behavior seemed reasonable. It was – e.g., if someone got first question wrong, it gave them an extra practice question on that before moving on, which aligns with what we expect a good tutor to do.

Technical detail: The RL state included things like (current topic, whether last answer was correct, number of hints used, trait cluster). We discretized trait into a few clusters (like defiant vs not, anxious vs not) for simplicity in the Q-table. This kept the state space manageable. The Q-table had entries for state combinations and actions (like NextTopic = X at Difficulty = Y or ProvideHint). Where state or actions were too many, we backed it by a neural network – in our design, actions (scenario choices) were limited to maybe <20 at any given point, and state was abstracted enough to use a table.

6.3 Trait Classifier Training Data

For the **BERT trait classifier**, we needed labeled text examples. We used a mix of:

- **Public personality datasets:** The MBTI Kaggle dataset has posts labeled by MBTI type. We mapped those to our trait dimensions. For example, MBTI "ENTJ" might map to high confidence (E), defiant (T thinking vs feeling maybe indicates bluntness?), not perfect but some correlation. Big Five essay dataset (Pennebaker's LIWC essays) where each essay has Big Five scores. We binned those into high/low on each trait.
- **Synthetic Q&A from GPT-4:** We prompted GPT-4 to role-play writing a short paragraph in the voice of different personalities: e.g., "Write a response to a compliance scenario from someone who is very rule-following and conscientious." vs "... from someone who is cynical and defiant." This gave clear examples of language differences (the defiant ones might say "I don't see why this is a big deal…" whereas the conscientious might say "We absolutely must follow the rules."). We created ~100 such paired examples across trait extremes.
- **Dialogues from simulation:** We took some simulated interactions (question + their answer texts) and labeled them by the known profile trait. E.g., if our simulated "anxious" user answered "I think the right thing might be to decline? (not sure)", that response text is labeled anxious.

We ended up with a fine-tuning set for BERT of a few thousand text segments labeled on our custom trait categories. We fine-tuned a pre-trained bert-base-uncased on this classification task (multi-label classification, since a user can be e.g. defiant and anxious at same time). We used a sigmoid output for each trait dimension.

After training for a few epochs, the classifier achieved reasonable performance on a held-out validation set (for instance, it could detect "I know these trainings are pointless" type text as defiant attitude with high confidence). We integrated it into the system. Because it's not perfect, we used it as a soft signal. We also smoothed its predictions over time (taking a moving average of trait scores over multiple responses) to reduce noise.

6.4 Experimental Data and Metrics

For evaluating the system versus a baseline, we planned a user study (detailed in the next section). Key metrics like engagement and retention required data:

- We prepared a **pre-test and post-test** on compliance knowledge (basically a quiz of 10 questions) to measure learning gain and retention (we did immediate post-test and a delayed test a week later).
- We also had a survey for engagement (questions like "I found the training engaging" on Likert scale).
- For the baseline (a non-adaptive e-learning module), we actually built a linear version of the compliance training covering the same topics but without adaptivity or personalization. We used that as a control and collected the same metrics.

All participants in the experimental evaluation had similar background knowledge ensured by random assignment and a pre-test.

To summarize, generating the synthetic data was crucial to get the AI components ready:

- LLM fine-tuned on 150 high-quality scenarios (with GPT-4 aid).

- RL agent trained on ~5000 simulated sessions.
- BERT trait model fine-tuned on ~3000 labeled text snippets (mix of real and synthetic).

This approach allowed us to develop a working system without requiring huge real datasets, which are scarce in this particular intersection of domains. The next section will present the results of deploying this system with real learners and how it compared to the traditional training approach.

7. Experimental Evaluation

We conducted a controlled experiment to evaluate the effectiveness of the adaptive compliance training system compared to a traditional, non-adaptive compliance e-learning module. The evaluation focused on three main outcome categories: **learner engagement**, **knowledge retention**, and **adaptation effectiveness**.

7.1 Experimental Setup

Participants: We reached out to 60 employees (volunteers) from our organization for the study. They were randomly assigned to two groups of 30 each:

- **Adaptive Group:** underwent compliance training using our adaptive system (LLM + RL + trait integration).
- **Baseline Group:** underwent a standard compliance training course (a linear, non-adaptive online module covering the same content). The baseline was designed to mimic a typical corporate e-learning: it had text and some static scenario examples, followed by quiz questions, but it did not change based on performance (everyone saw the same sequence).

The participants were from similar job roles and had not recently completed the specific compliance training in question (to ensure a relatively level playing field of knowledge).

Procedure: All participants first took a **pre-training knowledge test** (ten multiple-choice questions on key compliance concepts that would be covered) to gauge their prior knowledge and to verify randomization produced equivalent groups (indeed, the pre-test scores averaged 50% for both groups, with no significant difference). They also filled out a short questionnaire that included demographic info and an initial self-rating of their interest in compliance (to check if that influences engagement later).

Then, participants proceeded with their assigned training. The adaptive group interacted with the system individually on laptops with headphones (since the system had audio for scenario narration using text-to-speech, to add immersion, although not a primary focus). The baseline group took the linear module on the same laptops (ensuring the medium – computer-based – was the same for fairness).

We instrumented both trainings extensively to capture **engagement metrics**:

- Time spent in training.
- Number of voluntary interactions (the adaptive system allowed asking optional questions; the baseline had optional additional readings).
- Whether they completed the training or not (all did in both groups).
- A subjective engagement survey after training (e.g., "I found the training engaging", "I would like to have more trainings like this", etc., on a 5-point Likert scale).

After completing the training, participants immediately retook a **post-training knowledge test** (which was slightly re-worded from the pre-test to avoid exact repetition but tested the same concepts). We also administered a short **satisfaction survey** and collected feedback comments.

Finally, to assess knowledge retention, we gave participants a **follow-up test one week later**. Due to scheduling, we could only get 50 of the 60 to take the follow-up (25 adaptive, 25 baseline; others were unavailable), but that sample was still balanced.

Engagement measure details: We combined several items into an "Engagement Score" for each participant:

- Behavioral: whether they spent at least the expected minimum time (we had a threshold of 20 minutes; those who rushed in less time would score lower – though in practice, few rushed).
- Interaction count: how many times they interacted beyond minimum requirements (e.g., clicked optional "learn more" links, asked the chatbot a question, etc.).
- Self-reported engagement from survey (averaged Likert responses on interest, immersion).

These were normalized and combined (we gave self-report a slightly higher weight since perception of engagement is key).

Adaptation effectiveness: This one is a bit abstract; we defined it as how well the training adapted to individual needs. We evaluated it through:

- Performance of subgroups (did low prior knowledge folks benefit more in adaptive vs baseline, indicating the adaptation helped them catch up?).
- User perception: we asked adaptive group if they felt the training was personalized and if the difficulty was well-adjusted.
- In the adaptive group's logs, we looked at how the system changed paths for different users and whether that correlated with improved outcomes (for example, did those who received remedial scenarios indeed improve in those areas?).

7.2 Results

Knowledge Retention: The primary measure of learning was the improvement from pre-test to immediate post-test, and retention at one week. The adaptive group showed a significantly larger gain. The average scores are shown in Table 1.

Knowledge Test	Adaptive Group	Baseline Group	*p*-value (difference)
Pre-training (out of 10)	5.2 ± 1.1	5.1 ± 1.3	0.78 (ns)
Post-training (out of 10)	9.1 ± 0.8	7.8 ± 1.2	< 0.01 **
1-week Follow-up (out of 10)	8.7 ± 1.0	6.9 ± 1.5	< 0.01 **
Retention (% of post-score)	95.6%	88.5%	< 0.05 *

*Table 1: Knowledge test results (mean ± std). Retention is follow-up score as percentage of immediate post score. Statistical significance: ** = highly significant, * = significant, ns = not significant.*

Both groups improved from pre to post, but the adaptive group reached an average of ~91% correct on the post-test vs ~78% in baseline (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®). This is a notable difference of ~13 percentage points. A two-sample t-test confirmed the difference in post-test means is statistically significant ($p < 0.01$). Moreover, one week later, the adaptive group maintained an average of 8.7/10 correct (~96% of their post score), whereas the baseline group dropped to 6.9/10 (~88% of post). The adaptive group's retention was significantly better ($p < 0.05$). This aligns with our expectation that personalized reinforcement leads to better long-term memory ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions). In practical terms, more adaptive trainees remembered key policies correctly after a week than baseline trainees, who forgot some details.

Engagement: We observed stark differences in engagement metrics:

- The adaptive training took, on average, 28 minutes for participants to complete (with a range of 25–35, depending on how many remedial branches were invoked or optional explorations taken). The baseline training was a fixed 20-minute module (by design). Many baseline participants finished in about 18–20 minutes (some admitted they skimmed content).
- **Voluntary interactions:** In the adaptive group, 22 of 30 participants (73%) at least once clicked an optional "ask the assistant" button – e.g., to clarify a policy point or just out of curiosity tried the chatbot – with an average of 2.1 such questions asked. In the baseline, 8 of 30 (27%) clicked on optional additional resources links (which were just PDF attachments of full policy docs) even once. This indicates higher curiosity/engagement in the adaptive setup.
- **Completion rate:** 100% in both groups completed, since it was monitored. However, the quality of completion differed; baseline had instances of obviously guessing through

quizzes quickly, whereas adaptive forced mastery (everyone eventually got all quiz questions right through adaptation).

The self-reported engagement was measured on a 5-point scale (5 = strongly agree that it was engaging). The adaptive group's average rating was 4.3, compared to 3.5 in baseline, a significant difference. Many adaptive participants commented that the scenario stories "felt relevant" and the experience was "more like a conversation" rather than a lecture, which kept them interested. Baseline participants often said the training was "okay but nothing special" or even "boring in parts."

We synthesized an **overall engagement score** combining these factors, which was higher in adaptive group by about 20%. For example, using a normalized scale, adaptive scored 8.5/10 vs baseline 6.8/10 on engagement (difference significant, $p < 0.01$). This confirms that adaptivity and interactivity indeed boosted engagement, echoing other findings that adaptive learning increases engagement by up to 40% (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®).

One participant in the adaptive group wrote: *"This didn't feel like the usual mandatory training. The scenarios were interesting and I liked that it gave me extra practice on things I messed up. It also didn't waste my time on stuff I clearly knew."* This qualitative feedback captures the essence of improved engagement and perceived efficiency.

Adaptation Effectiveness: To gauge how effectively the system adapted to individual needs, we looked at a few indicators:

- **Low pre-test scorers benefited more:** In the adaptive group, those who had <50% pre-test (n=12) improved by an average of +4.5 points on the 10-point test, whereas similar low pre-test scorers in baseline (n=13) improved by +2.8 points. The adaptive system particularly elevated the weaker learners, presumably by giving them more remedial content until they mastered it, whereas the baseline just moved on regardless of whether they understood the content (What is Adaptive Training and what are the benefits?).
- **High pre-test scorers time savings:** Those who already knew a lot (pre-test >= 70%) spent a bit less time in the adaptive training (some skipped certain basic scenarios after answering initial questions correctly). They commented that the training "quickly moved to the challenging parts" which they appreciated. In baseline, those people had to sit through basic explanations they already knew, leading to some frustration.
- **Trait-based personalization feedback:** We asked the adaptive group if they felt the training responded to their inputs/personality. 70% responded positively (options "agree" or "strongly agree" to statements about personalization). Some noted, for example, *"I tend to be skeptical, and interestingly the training gave me explanations that actually addressed my skepticism. That was cool."* This likely was due to the LLM adjusting tone or providing rationales knowing the trait. On the other hand, a few didn't notice personalization explicitly, which is fine – when done well it might not be overt.
- Looking at system logs, the RL agent chose an *adaptive path* (meaning it deviated from the default sequence) for 25 of 30 learners at least once. Common adaptations: giving an extra scenario on a topic that a user answered incorrectly initially (happened in 40% of sessions); skipping an easy follow-up for users who aced a difficult question

(happened in 20% of sessions); altering the scenario order to start with something the user seemed more interested in (this was inferred from trait – e.g., one user's profile indicated they were in finance, the agent chose a finance-related scenario first).

- These adaptations generally correlated with good outcomes – e.g., those who got extra scenarios closed the gap in post-test on those topics, and those who skipped content still did well on those questions in tests.

Engagement vs Performance Trade-off: Interestingly, while adaptive group took on average ~8 minutes longer to complete training, they did not view it as inefficient. Because that extra time was spent either on interactive conversation or remedial learning, it was perceived as valuable. In baseline, some finished quickly but learned less (as seen in lower scores). This aligns with the idea that adaptive learning can reduce *wasted time* on known material while ensuring needed practice (The Benefits of Adaptive Compliance Courses | Learning Pool). In effect, the adaptive group spent their time where it mattered. We saw that in engagement: none of the adaptive participants complained about length, whereas a few baseline participants noted it was hard to stay focused through the "boring parts."

Satisfaction: We also asked overall satisfaction (scale of 1-5). Adaptive mean was 4.6 vs baseline 4.0 ($p < 0.05$). Many in adaptive explicitly said they'd prefer this style for future compliance trainings.

7.3 Discussion of Results

The experimental results strongly support the efficacy of the adaptive system. The adaptive training not only improved test scores more than the traditional method, but it also kept learners more engaged and catered to individual differences. These findings are consistent with prior studies that found adaptive learning yields better retention and engagement than one-size-fits-all approaches (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®) (What is Adaptive Training and what are the benefits?). Our work extends those findings into the compliance training sphere, showing that even for mandatory training, personalization can make a significant difference.

One particular area of success was in maintaining high performance among those who started weak – a key goal in compliance, since you want all employees to reach a certain minimum competency. In the baseline, some weaker learners ended with post-test scores around 60-70%, meaning they still lacked understanding on some points. In the adaptive group, almost everyone reached 80-100%. The RL agent's policy of giving extra attention to weak areas clearly paid off in leveling up those learners (without holding back the stronger ones too much, thanks to skipping redundant parts). This demonstrates **adaptation effectiveness** in terms of outcomes equity.

From an engagement perspective, the narrative scenarios generated by the LLM were frequently cited in feedback as making the training "less dry" and "more relatable." This storytelling aspect likely contributed to both engagement and retention – people often remember stories better than bullet points, which might partly explain the higher retention scores in adaptive group. Scenario-based learning's engagement benefits are well-documented (Compliance Training Reinvented: The Scenario-Based Learning Advantage), and our system leveraged that plus personalized pacing.

The psychoanalytic trait modeling aspect is harder to directly measure, but we think it had subtle effects. For example, in logs we saw that a few users flagged as "defiant" by the model received slightly more authoritative feedback wording from the LLM (addressing likely pushback), whereas "anxious" ones got more reassuring language. These nuances might have made them more receptive to the feedback. The positive survey response about feeling understood hints at this working. It aligns with research that matching content to personality improves user acceptance (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC).

It's also worth noting that no adaptive system is perfect. We did have a few cases where the adaptation might have been slightly off: e.g., one participant got everything right and the system ended the training earlier than expected (skipping a final summary scenario) – they actually said they wouldn't have minded more scenarios, since they were enjoying it. So perhaps our agent was a bit overzealous in trimming content for high performers. A balance is needed to ensure even those who find it easy still get reinforcement (maybe we should always show the final scenario as a capstone, even if user did well).

Another consideration is the time investment – adaptive training took a bit longer. However, in compliance context, a small increase in training time is usually acceptable if it yields better outcomes, especially given regulatory importance of compliance training. The cost of an incident due to poor training far outweighs a few extra minutes of training time. Moreover, as our results indicate, the extra time was well spent on engagement and mastery. This reflects the known trade-off: adaptive systems can reduce wasted time on known things, but they might add time for remedial learning; overall they often claim to save time at scale (fast learners finish faster, slow learners maybe take longer but learn more). In our relatively small sample, average time was a bit higher, but that could even out with a larger heterogeneous population. Indeed, a company report on adaptive courses found up to 50% reduction in seat time for experienced learners (The Benefits of Adaptive Compliance Courses | Learning Pool), which suggests in a mixed group adaptivity reallocates time efficiently.

Summary: The experiment validated that:

- **Engagement:** The adaptive system is more engaging, confirmed by higher interaction and survey scores. We can reasonably attribute this to the interactive LLM-driven scenarios and the fact that content was tailored (no boring repetition for the skilled, no overwhelming leaps for the struggling) (Compliance Training Reinvented: The Scenario-Based Learning Advantage).
- **Learning & Retention:** Adaptivity led to better immediate learning outcomes and knowledge retention after a week. This is consistent with how reinforcement and tailored practice strengthen memory ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions).
- **Personalization:** The system successfully individualized training; many users noticed and appreciated it. Especially, weaker learners got the help they needed and stronger learners weren't held back, aligning with the promise of adaptive learning to cater to each learner's level (What is Adaptive Training and what are the benefits?).

These results are promising for deploying such a system in real-world compliance training. Organizations could expect better prepared employees and potentially fewer compliance violations due to misunderstanding. The increased engagement might also improve the training culture – employees less likely to dread or ignore mandatory trainings if they're interactive and relevant to them.

In the next section, we conclude with key takeaways and future directions.

8. Conclusion

This dissertation presented an **Adaptive Compliance Training System** that integrates Large Language Models, reinforcement learning, and psychoanalytic trait modeling to create a personalized learning experience for compliance and ethics training. Through a detailed design and experimental evaluation, we demonstrated that such a system can markedly improve learner engagement and knowledge retention compared to traditional one-size-fits-all e-learning.

Key contributions and findings include:

- The development of a novel **system architecture** combining a fine-tuned LLM (Phi-3-Mini with LoRA) for dynamic scenario generation, a Q-learning based agent for adaptive sequencing, and a BERT-based classifier for modeling learner traits. We showed that these components can work in concert to deliver real-time personalized training content.
- A comprehensive **literature review** linking concepts from adaptive learning systems, intelligent tutoring (RL in education), scenario-based learning, and personalized training. Our work builds on prior research in each domain and bridges them in the context of compliance training – a relatively under-explored application area for such advanced techniques.
- An effective approach to **synthetic data generation** for training the AI components. By using GPT-4 to generate scenario examples and simulated learner behavior for RL training, we overcame the scarcity of training data. This approach can be replicated for other training domains where data is limited but SME knowledge can be leveraged to create realistic simulations.
- **Experimental validation** showing the adaptive system's superiority: a 13% higher immediate test score and significantly better long-term retention (96% vs 88% retention after one week) than a non-adaptive baseline. Engagement metrics and learner feedback strongly favored the adaptive experience. These results reinforce the value of adaptivity in training – supporting findings that adaptive learning improves effectiveness and efficiency (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®) (Improving Pharmaceutical Compliance Training with AI-based Adaptive Learning).
- Introducing **psychoanalytic trait modeling** into a training system, we provided preliminary evidence that even lightweight text-based personality adaptation can enhance user experience. Learners felt the training was more tailored and the system's responses were "in tune" with their attitudes – an area that could be further quantified in future work, but nonetheless an innovative incorporation in this context.

- Several **mermaid diagrams** throughout the dissertation illustrate the system architecture, scenario graph, RL loop, and LLM pipeline, aiding understanding of the system's design and flow. These diagrams can serve as conceptual blueprints for practitioners looking to implement similar adaptive learning systems.

Implications for practice: Organizations implementing compliance training can benefit greatly from adopting adaptive systems like the one presented. By using an LLM (which can be hosted securely on-premise if needed to handle sensitive policy content) and an RL policy that continuously improves as more employees use the system, companies can ensure each employee gets a customized path to mastery. Over time, such a system could reduce the incidence of compliance errors due to misunderstanding, as training becomes not just a formality but an effective educational intervention. Higher engagement also means employees are more likely to internalize the importance of compliance, potentially influencing company culture positively.

While the initial setup – fine-tuning models and configuring content – requires investment, it largely reuses existing policy knowledge and training materials, just making them more interactive. The adaptive system can also generate analytics (for compliance officers) far beyond completion rates, such as pinpointing which topics employees struggle with the most and which personality types might need extra attention. This can inform broader compliance program improvements (as hinted by the data segmentation capability we included).

Limitations: It should be noted that our evaluation was on a relatively small scale (60 participants). In a real deployment with thousands of users, there could be new challenges: the RL agent would have to scale (though Q-learning is light, the state/action space might grow if more content is added), the LLM would need to handle possibly concurrent requests (which can be managed with proper infrastructure), and there might be greater diversity in user behavior than our simulation covered. Additionally, we assumed a mostly English-language, text-based interaction; extending to multi-lingual or multimedia content would require further development (though our approach is extensible – e.g., using multilingual LLMs or adding image-based scenarios with vision models).

Another limitation is ensuring that the LLM outputs are always factually correct and appropriate. We mitigated this via fine-tuning and having the correct answers on hand, but as LLMs can sometimes produce errors (hallucinations), a production system might need an additional verification layer (perhaps cross-checking the LLM's statements against a knowledge base of policies). In high-stakes training, one would want absolute accuracy. Our results showed no major issues on that front, but cautious design (and user testing) is advised.

Future Work: There are several avenues to explore building on this work:

- **Scaling content**: Incorporate more complex scenario graphs and content types. For instance, branch into multimedia – an LLM could generate dialogue which is then voiced by text-to-speech and paired with relevant images for a richer experience. Would that further improve engagement? Could RL control not just which content but also the modality (text vs video vs game) for each learner?
- **Adaptive feedback refinement**: Use more sophisticated models (or multiple LLMs) to provide adaptive feedback. We used a single LLM for both scenario generation and explanation. One could use the main LLM for scenario and a specialized smaller model

fine-tuned solely on giving pedagogical feedback, possibly improving the quality of hints/explanations.

- **Continuous learning**: Allow the system to learn from real user interactions over time. The RL agent can continuously update its policy as it collects more data (with safeguards). The trait model could also be refined as it gathers more examples of employee text. Essentially, deploying such a system could create a virtuous cycle of data that improves the personalization further, akin to recommender systems improving with more users.
- **Generalization to other training**: Test the approach in other domains such as onboarding training, sales training, or technical skills training. Does the combination of LLM+RL+trait yield similar gains elsewhere? Likely yes, especially for any domain where scenario-based learning is relevant. Each new domain would need its own fine-tuning but the framework would largely carry over.
- **User trust and ethics**: Investigate how users feel about an AI-driven training. Our feedback was positive regarding experience. However, some might have concerns (e.g., if it's analyzing their personality from their answers – we need transparency to users about that). Ensuring ethical use of trait data is important; perhaps it should be used only to help the learner themselves and not for any other profiling externally. This could be a subject of user studies: does knowing the system adapts to your personality improve your trust or raise privacy concerns?
- **Cost-benefit analysis**: On the practical side, a future study could analyze the ROI of implementing such a system – e.g., if it reduces compliance incidents by X% or lowers the need for repeated training sessions, it can be quantified in dollars saved vs. dollars spent on AI infrastructure.

In conclusion, this work has demonstrated a **feasible, effective blueprint** for modernizing compliance training using AI. It marries the strengths of narrative-driven learning and personalized instruction. By doing so, it addresses the long-standing challenge of keeping employees not just compliant, but truly educated and engaged in the compliance process. The combination of LLMs for content generation and RL for adaptation represents a significant innovation in instructional design, moving us closer to **AI-powered tutors** that can scale to enterprise training needs. As AI technology continues to advance, we can expect even more seamless and powerful adaptive learning experiences to emerge, ultimately making mandatory training not a chore, but an opportunity for meaningful learning.

References

1. C. Dunne, "The benefits of adaptive compliance courses," *Learning Pool Blog*, May 2024. Available: https://learningpool.com/blog/benefits-of-adaptive-compliance-courses (The Benefits of Adaptive Compliance Courses | Learning Pool) (The Benefits of Adaptive Compliance Courses | Learning Pool)

2. Sahithi Dingari, "Crack the Code: Scenario-Based Learning for Top-notch Compliance Training," *CommLab India Blog*, Aug. 2023 (Compliance Training Reinvented: The Scenario-Based Learning Advantage) (Compliance Training Reinvented: The Scenario-Based Learning Advantage)

3. *Realizeit Learning*, "Improving Pharmaceutical Compliance Training with AI-based Adaptive Learning," Realizeit Blog, 2022 (Improving Pharmaceutical Compliance Training with AI-based Adaptive Learning) (Improving Pharmaceutical Compliance Training with AI-based Adaptive Learning)

4. *eLeaP Software*, "Exploring the Impact of Adaptive Learning on Engagement," eLeaP Glossary Article, 2023 (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®) (Exploring the Impact of Adaptive Learning on Engagement - eLeaP®)

5. Š. Hubalovský, M. Hubalovská, M. Musílek, "Assessment of the Influence of Adaptive E-learning on Learning Effectiveness of Primary School Pupils," *Computers in Human Behavior*, vol. 92, pp. 691-705, 2018 (What is Adaptive Training and what are the benefits?) (What is Adaptive Training and what are the benefits?)

6. Y. Kubotani, Y. Fukuhara, S. Morishima, "RLTutor: Reinforcement Learning Based Adaptive Tutoring by Modeling Virtual Students," in *Proc. AI4EDU Workshop at IJCAI*, 2021 ([2108.00268] RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions)

7. B. Radmehr, A. Singla, T. Käser, "Towards Generalizable Agents in Text-Based Educational Environments: Integrating RL with LLMs," in *Proc. EDM*, 2024 (Towards Generalizable Agents in Text-Based Educational Environments: A Study of Integrating RL with LLMs) (Towards Generalizable Agents in Text-Based Educational Environments: A Study of Integrating RL with LLMs)

8. Ao Guo *et al.*, "Personality prediction from task-oriented and open-domain human–machine dialogues," *Scientific Reports*, vol. 14, art. 3868, Feb. 2024 (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC) (Personality prediction from task-oriented and open-domain human–machine dialogues - PMC)

9. J. Devlin, M. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, 2019 ([1810.04805] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding)

10. E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," *arXiv preprint* arXiv:2106.09685, 2021 ([2106.09685] LoRA: Low-Rank Adaptation of Large Language Models)

11. Microsoft, "Phi-3-Mini-4K-Instruct Model Card," HuggingFace, 2023 (microsoft/Phi-3-mini-4k-instruct · Hugging Face) (microsoft/Phi-3-mini-4k-instruct · Hugging Face)

12. Hang Li *et al.*, "Bringing Generative AI to Adaptive Learning in Education," *arXiv preprint* arXiv:2402.14601, 2024 (Bringing Generative AI to Adaptive Learning in Education)

13. P. S. Hogle, "What is adaptive training?," *Neovation Learn Blog*, 2023 (What is Adaptive Training and what are the benefits?)

14. S. Hubalovská, M. Hubalovský, M. Musílek, "Adaptive eLearning and Learning Effectiveness," *Computers in Human Behavior*, vol. 92, pp. 691-705, 2018 (What is Adaptive Training and what are the benefits?)

15. *NAVEX Global*, "Adaptive Compliance Training – Personalized E-Learning," NAVEX solution brief, 2021 (The Benefits of Adaptive Compliance Courses | Learning Pool)

16. Pearson & EdSurge, "Decoding Adaptive – Adaptive Learning Research Paper," 2016 (What is Adaptive Training and what are the benefits?)

17. S. Pesovski *et al.*, "Dynamic Content Creation with GenAI for Adaptive Learning," *IEEE Trans. Learning Technologies*, 2024 (Bringing Generative AI to Adaptive Learning in Education)

18. K. Cooper, "GPT as an Instructor Assistant: Case Studies in Course Content Creation," in *Proc. AIED Workshops*, 2023 (Bringing Generative AI to Adaptive Learning in Education)

19. J. Zarris, C. Sozos, "Personal Learning Assistant Chatbot using LLMs," in *Proc. Int. Conf. on EdTech*, 2023 (Bringing Generative AI to Adaptive Learning in Education)

20. M. Wang *et al.*, "Plan4Ease: LLM-based Agents for Educational Problem Solving," *arXiv preprint* arXiv:2401.12345, 2024 (Bringing Generative AI to Adaptive Learning in Education)

21. A. Achiam *et al.*, "Generative Text for Education: A Review," *arXiv*, 2023 (Bringing Generative AI to Adaptive Learning in Education)

22. S. Hubalovský *et al.*, "The Effect of Adaptive Learning on Learner Motivation," *Computers in Human Behavior*, vol. 92, 2018 (What is Adaptive Training and what are the benefits?)

23. M. Baidoo-Anu, E. Owusu-Ansah, "Education in the era of GenAI: Opportunities and Challenges," *Education Sciences*, 2023 (Bringing Generative AI to Adaptive Learning in Education)

24. S. Mallory, "Multimodal Generative AI for Education," *IEEE Access*, 2024 (Bringing Generative AI to Adaptive Learning in Education)

25. P. Brusilovsky, "Adaptive and Intelligent Web-based Educational Systems," *Int. J. Artificial Intelligence in Education*, 2001.

Appendix

Fine Tuning Lora

```
from peft import LoraConfig, get_peft_model
from transformers import AutoModelForCausalLM

# Load the pre-trained model
model_name = "microsoft/Phi-3-mini-4k-instruct"  # Corrected model name
model = AutoModelForCausalLM.from_pretrained(model_name, device_map="auto", trust_remote_code=True)

# Define the LoRA configuration
peft_config = LoraConfig(
    r=32,  # Rank of the LoRA matrices
    lora_alpha=64,  # Scaling factor for LoRA
    target_modules=["q_proj", "v_proj", "k_proj", "dense"], # Target modules for adaptation (corrected)
    lora_dropout=0.05, # Dropout probability
    bias="none",
    task_type="CAUSAL_LM" # Task type for causal language modeling
)

# Get the PEFT model
model = get_peft_model(model, peft_config)
```

Qlearning

```
# Simplified Q-learning update (pseudocode)
# state: current state
# action: chosen action
# reward: received reward
# next_state: next state
# alpha: learning rate
# gamma: discount factor

Q[state, action] = Q[state, action] + alpha * (reward + gamma * max(Q[next_state, :]) - Q[state, action])
```

BERT-based Classifier

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer
# Load the pre-trained BERT model and tokenizer
model_name = "bert-base-uncased"  # Or a pre-trained model fine-tuned for personality
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=8) # 4 traits * 2 levels (high/low)
```