

PRECISE KNOWLEDGE TRACING MODEL FOR FUTURE EXERCISE PREDICTION ACCURACY

A PROJECT REPORT

Submitted by

PRAVEENKUMAR T (422420104027)

REGUNATHAN V (422420104029)

SRITHARAN T (422420104036)

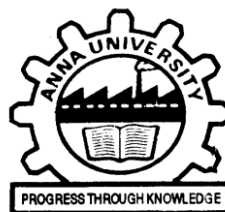
in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



UNIVERSITY COLLEGE OF ENGINEERING TINDIVANAM

ANNA UNIVERSITY: CHENNAI 600025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project report “**Precise Knowledge Tracing model for Future Exercise Prediction Accuracy**” is the Bonafide work of “**PRAVEENKUMAR T (422420104027), REGUNATHAN V (422420104029), SRITHARAN T (422420104036)**” who carried out the project work under my supervision.

SIGNATURE

Dr. L. JEGATHA DEBORAH, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Assistant professor,
Department of CSE,
University College of Engineering,
Tindivanam,
Melpakkam – 604001

SIGNATURE

Dr. R. KARTHIKA, M.E., Ph. D.,

SUPERVISOR

Assistant professor,
Department of CSE,
University College of Engineering,
Tindivanam,
Melpakkam – 604001

Submitted for the University Examination held on _____

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

We would like to thank our revered Dean, **Dr.P.Thamizhazhagan, M.E., Ph.D.**, for providing infrastructure facilities and whole hearted encouragement for completing our project successfully.

For mostly, we pay our grateful acknowledgement and extend our sincere gratitude to **Dr.L.Jegatha Deborah, M.E., Ph.D.**, Assistant professor and Head, Department of Computer Science and Engineering, University College of Engineering Tindivanam, for extending the facilities of the department towards our project and for unstinting support.

We express our thanks to our guide, **Dr.R.Karthika, M.E., Ph.D.**, Assistant professor, Department of Computer Science and Engineering, University College of Engineering Tindivanam, for guiding us for every phase of the project. We appreciate her thoroughness, tolerance and ability to share her knowledge with us. We thank her for being easily approachable and quite thoughtful. We owe her harnessing our potential and bringing out the best in us. Without her immense support through every step of the way, we could never have it to this extent.

We extend our thanks to the project coordinator **Ms.P.Sathya, M.Tech.**, for her continual support during the entire project schedule.

We thank all our teaching and non-teaching faculty members of the department and also our fellow friends for helping us in providing valuable suggestions and timely ideas for the successful completion of the project .We sincerely thank all of them.

ABSTRACT

Knowledge tracing is a technique used in the field of educational data mining and intelligent tutoring systems to model and predict students' mastery of concepts or skills over time. This involves two steps: Updating learner's knowledge state and Predicting learner's correctness of next exercises. The proposed work introduces Precise knowledge tracing (PKT) that contains two types of difference from exercise and from prior knowledge. The first type of difference is the exercise level difference which refers to difference among the exercises in term of their content difficulty and relationship to other exercises. The second type of difference is the learner level difference which refers to the difference among learners in term of their prior knowledge and learning abilities. PKT uses deep learning method of recurrent neural network (RNN) to model knowledge state (KS) and predicts the performance. Using deep learning method the recurrent neural network (RNN) is handle sequential data by maintaining a hidden state that captures information about previous inputs in the sequence. Experimental analysis shows that the proposed precise knowledge tracing model outperforms existing models in their prediction accuracy.

KEYWORDS: Knowledge tracing, intelligent tutoring system, e-learning, recurrent neural network.

திட்டப்பணிச்சுருக்கம்

அறிவுத் தடமறிதல் என்பது கல்வித் தரவுச் செயலாக்கம் மற்றும் அறிவார்ந்த பயிற்சி முறைமைகள் ஆகியவற்றில் பயன்படுத்தப்படும் ஒரு நுட்பமாகும், இது காலப்போக்கில் மாணவர்களின் கருத்தாக்கங்கள் அல்லது திறன்களின் தேர்ச்சியை மாதிரியாகவும் கணிக்கவும் பயன்படுகிறது. இது இரண்டு படிக்களை உள்ளடக்கியது: கற்பவரின் அறிவு நிலையை புதுப்பித்தல் மற்றும் அடுத்த பயிற்சிகளை கற்பவரின் சரியான தன்மையை கணித்தல். முன்மொழியப்பட்ட வேலை துல்லியமான அறிவுத் தடமறிதலை (PKT) அறிமுகப்படுத்துகிறது, இது உடற்பயிற்சி மற்றும் முன் அறிவிலிருந்து இரண்டு வகையான வித்தியாசங்களைக் கொண்டுள்ளது. முதல் வகை வேறுபாடு உடற்பயிற்சி நிலை வேறுபாடு ஆகும், இது பயிற்சிகளின் உள்ளடக்க சிரமம் மற்றும் பிற பயிற்சிகளுடனான உறவின் அடிப்படையில் உள்ள வேறுபாட்டைக் குறிக்கிறது. இரண்டாவது வகை வேறுபாடு கற்பவர் நிலை வேறுபாடு ஆகும், இது அவர்களின் முன் அறிவு மற்றும் கற்றல் திறன்களின் அடிப்படையில் கற்பவர்களிடையே உள்ள வேறுபாட்டைக் குறிக்கிறது. PKT, அறிவு நிலையை (KS) மாதிரியாக மாற்ற, தொடர்ச்சியான நரம்பியல் வலையமைப்பின் (RNN) ஆழ்ந்த கற்றல் முறையைப் பயன்படுத்துகிறது மற்றும் செயல்திறனைக் கணிக்கின்றது. ஆழ்ந்த கற்றல் முறையைப் பயன்படுத்தி, தொடர்ச்சியான நரம்பியல் வலையமைப்பு (RNN) வரிசைமுறையில் முந்தைய உள்ளீடுகளைப் பற்றிய தகவலைப் பிடிக்கும் ஒரு மறைக்கப்பட்ட நிலையைப் பராமரிப்பதன் மூலம் தொடர்ச்சியான தரவைக் கையாளுகிறது. முன்மொழியப்பட்ட துல்லியமான அறிவுத் தடமறிதல் மாதிரியானது, ஏற்கனவே உள்ள மாதிரிகளை அவற்றின் கணிப்புத் துல்லியத்தில் விஞ்சுகிறது என்பதை பரிசோதனை பகுப்பாய்வு காட்டுகிறது..

முக்கிய வார்த்தைகள்: அறிவுத் தடமறிதல், அறிவார்ந்த பயிற்சி அமைப்பு, மின் கற்றல், மீண்டும் மீண்டும் வரும் நரம்பியல் நெட்வொர்க்.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT-ENGLISH	iv
	ABSTRACT-TAMIL	v
	TABLE OF CONTENTS	vi
	LIST OF FIGURES	x
	LIST OF TABLES	xi
	LIST OF ABBREVIATIONS	xii
1	INTRODUCTION	1
	1.1 Intelligent Tutoring Systems	1
	1.2 Deep Learning Approach	2
	1.3 Knowledge Tracing	2
	1.4 Precise Knowledge Tracing	3
2	LITERATURE SURVEY	4
3	PROBLEM STATEMENT	9
	3.1 Objective	9
	3.2 Scope	9
	3.3 System Description	10
4	REQUIREMENTS SPECIFICATION	11
	4.1 Functional Requirements	11
	4.1.1 Hardware Requirements	11
	4.1.2 Software Requirements	11
	4.2 Non-Functional Requirements	12
5	SYSTEM DESIGN	13
	5.1 System Architecture	13

5.2 Module Description	14
5.2.1 Registration	14
5.2.1.1 Learner Sign In / Sign Up	14
5.2.1.2 Course Enrollment	14
5.2.2 Assessment	15
5.2.2.1 Learning Knowledge Concept	15
5.2.2.2 Knowledge Concept Assessment	15
5.2.2.3 Calculate Prior Knowledge	15
5.2.3 Precise Knowledge Tracing	16
5.2.3.1 Module Assessment	16
5.2.3.1.1 Exercise Embedding	16
5.2.3.1.2 Integration Function	19
5.2.3.2 Relevant Prior Knowledge	19
5.2.3.1.1 Scaled Dot Product Attention	22
5.2.4.1.2 Softmax Function	22
5.2.5 Learner State Evaluation	22
5.2.5.1 Recurrent Neural Network	23
5.2.5.2 Hyperbolic Tangent Activation Function	23
5.2.5.3 Update Current Knowledge State	24
5.2.6 Prediction	25
5.2.6.1 Rectified Linear Unit Activation Function	26
5.3 UML Diagrams	26
5.3.1 Use Case Diagram	26
5.3.2 Class Diagram	28
5.3.3 Sequence Diagram	29
5.3.4 Activity Diagram	30

	5.3.5 State Chart Diagram	31
	5.3.6 Collaboration Diagram	32
	5.3.7 Deployment Diagram	33
	5.3.8 Package Diagram	34
	5.4 Dataflow Diagram	35
	5.4.1 DFD LEVEL-0	35
	5.4.2 DFD LEVEL-1	36
	5.4.3 DFD LEVEL-2	37
6	IMPLEMENTATION	38
	6.1 Dataset	38
	6.2 Technologies Used	39
	6.2.1 Python	39
	6.3 Previous Study	40
	6.3.1 Performance for all KT methods	40
	6.4 Snapshots	41
7	TESTING	44
	7.1 Testing objectives	44
	7.2 Types of tests	45
	7.2.1 Unit testing	45
	7.2.2 Integration testing	45
	7.2.3 Functional testing	46
	7.2.3.1 Performance testing	46
	7.2.3.2 Stress Testing	47
	7.2.3.3 Structure testing	47
	7.2.3.4 System testing	47
	7.3 Testing techniques	48
	7.3.1 Testing	48
	7.3.2 White box testing	49

	7.3.3 Black box testing	49
	7.3.3.1 Boundary Score analysis	49
	7.3.3.2 Equivalence partitioning	50
	7.3.3.3 Comparison testing	50
	7.4 Test strategy and approach	50
	7.4.1 Integration testing	51
	7.4.2 User acceptance testing	51
8	RESULTS AND DISCUSSIONS	52
	8.1 Performance Analysis	52
	8.2 Validation Loss	54
9	CONCLUSION AND FUTURE WORK	55
	9.1 Conclusion	55
	9.2 Future Work	55
	REFERENCES	56
	APPENDIX	59

LIST OF FIGURES

FIG.NO	FIGURE NAME	PAGE NO.
5.1	System Architecture	13
5.2.1	Calculating RPK	20
5.2.2	PKT framework	23
5.3.1	Use case Diagram	27
5.3.2	Class Diagram	28
5.3.3	Sequence Diagram	29
5.3.4	Activity Diagram	30
5.3.5	State Chart Diagram	31
5.3.6	Collaboration Diagram	32
5.3.7	Deployment Diagram	33
5.3.8	Package Diagram	34
5.4.1	DFD-level 0	35
5.4.2	DFD-level 1	36
5.4.3	DFD-level 2	37
6.1	Initialize Assisst17 dataset	41
6.2	Starting epoch 1 to 6	41
6.3	Evaluate epoch 7 to 11	42
6.4	Evaluate epoch 12 to 17	42
6.5	Final result of Assisst17 dataset	43
8.1	Performance comparison for all KT methods	52
8.1.1	Performance analysis for Assisst17	53
8.1.2	Performance analysis for Assisst09	53
8.2	Validation loss graph	54

LIST OF TABLES

Table No.	TABLE NAME	PAGE NO.
I	Dataset	38
II	Performance analysis for all KT methods	40

LIST OF ABBREVIATION

S.NO	ABBREVIATION	DEFINITIONS
1	BKT	Bayesian Knowledge Tracing
2	PKT	Precise Knowledge Tracing
3	KT	Knowledge Tracing
4	RNN	Recurrent Neural Network
5	DKT	Deep Knowledge Tracing
6	DHKT	Deep Hierarchical Knowledge Tracing
7	DKVMN	Dynamic Key Score Memory Network
8	LSTM	Long-Short Term Memory
9	NLP	Natural Language Processing
10	SAKT	Self-Attentive Knowledge Tracing
11	MANN	Memory-Augmented Neural Network
12	HMM	Hidden Markov Model
13	PFA	Performance Factor Analysis
14	AFM	Additive Factor Model
15	AUC	Area Under Curve
16	ACC	Accuracy
17	RMSE	Root Mean Square Error

CHAPTER 1

INTRODUCTION

The rapid development of Intelligent Tutoring Systems (ITS) and online education is due to the integration of computer technology in the educational sector. It emphasizes the importance of adaptive learning in education platforms and the role of knowledge tracing in predicting learners' performance based on sequential learning interactions. The integration of computer technology into education has led to the rapid development of intelligent tutoring systems and open online courses. Its aim is to update a learner's current knowledge state and predict learners' performance by using sequential learning interaction. It can be very useful to tailor instruction and educational platform and recommend the best exercise.

1.1 Intelligent Tutoring System

Intelligent Tutoring Systems (ITS) that provide personalized and adaptive learning experiences to students. ITS use knowledge tracing models to track and model the learner's knowledge state over time. By analyzing the learner's interactions with the system, ITS can infer the learner's strengths, weaknesses, and learning progress to provide targeted support and feedback. It offers immediate and personalized feedback to learners based on their responses to exercises and assessments. This feedback can help learners understand their mistakes, reinforce correct concepts, and guide them towards mastery. ITS are designed to cover a wide range of subjects and domains, language learning and programming. The system's domain expertise allows it to provide accurate and relevant instruction in various educational areas. ITS can continuously adapt and improve based on user feedback, performance data, and advancements in educational technology.

1.2 Deep Learning Approach

Deep learning models can automatically learn relevant features from the data, reducing the need for manual feature engineering and potentially uncovering hidden patterns that may not be apparent with traditional approaches. Variations in integration functions used in deep learning models can lead to performance gaps. Selecting the appropriate integration function is crucial for optimizing the model's predictive capabilities. This model requires large amounts of data for training, and there is a risk of overfitting when the model learns noise in the data rather than meaningful patterns. Balancing model complexity with data availability is crucial.

1.3 Knowledge Tracing

Knowledge tracing has gained prominence in educational research and practice due to its potential to offer personalized learning experiences. By analyzing students' responses to exercises, knowledge tracing models can provide insights into individual learning trajectories, identify areas of strength and weakness, and offer tailored interventions to support learning. This personalized approach is particularly valuable in today's diverse classrooms, where students come from varied backgrounds and possess different levels of prior knowledge.

While traditional knowledge tracing models have proven useful, they have certain limitations that hinder their effectiveness. One major limitation lies in their narrow focus on exercise-specific differences, often overlooking the influence of students' prior knowledge. By solely analyzing students' responses to exercises, these models fail to account for the diverse ways in which students approach learning tasks based on their existing knowledge and skills. Bayesian Knowledge Tracing provides a clear and interpretable framework for estimating a learner's mastery level of each knowledge concept. It uses a Hidden Markov Model (HMM) to compute the probabilistic estimation of mastering the

corresponding knowledge concept. BKT assumes that a learner's mastery level of each knowledge concept is a binary latent variable, making it a foundational model in knowledge tracing research. Traditional knowledge tracing models, including BKT, often overlook the actual differences among exercises. These models primarily focus on knowledge concepts and may not fully consider the unique characteristics and variations present in different exercises. Struggle to understanding long-term dependencies is crucial for capturing the evolution of a learner's knowledge state over time.

1.4 Precise Knowledge Tracing

The proposed precise knowledge tracing model does the following

- 1) Actual difference among the exercise at respective knowledge concept. Because the same knowledge concept has different question based on their difficulties.
- 2) The actual different among the prior knowledge, so we are proposed a RPK methods to predict how the prior knowledge gaining for the current exercise or knowledge concept.
- 3) By combining both actual difference among exercise and RPK to produce a raw exercise representation.
- 4) Next to combine the raw exercise representation and historical learning interaction to fed into LSTM to update the learners correct knowledge state.

CHAPTER 2

LITERATURE SURVEY

2.1 C. Piech, “Deep knowledge tracing,” in Proc. Adv. Neural Inf. Process. Syst. 28: Annu. Conf. Neural Inf. Process. Syst. Montreal, QC, Canada, (2015).[15]

Deep Knowledge Tracing (DKT) is a novel approach that leverages deep learning techniques to model and predict student knowledge mastery in intelligent tutoring systems. This paper introduces DKT as a method to address the challenges of accurately tracking and predicting student learning progress over time. By utilizing Long Short-Term Memory (LSTM) networks, DKT can capture the sequential nature of student interactions with educational content and dynamically update the estimation of student knowledge. The model's ability to handle temporal dependencies enables it to provide real-time feedback and personalized learning paths. Through experiments on large-scale datasets, DKT demonstrates superior performance compared to traditional knowledge tracing methods, showcasing its potential to enhance adaptive learning systems. This paper contributes to the advancement of student modelling techniques in educational technology and lays the foundation for future research in deep learning-based approaches to knowledge tracing.

Merits: DKT is capable of capturing long-term dependencies in student learning sequences.

Demerits: In scenarios where data collection is challenging or limited, DKT may not perform optimally.

2.2 A. T. Corbett and J. R. Anderson, “Knowledge tracing: Modeling the acquisition of procedural knowledge,” User Model. User-Adapted Interaction, vol. 4, no. 4.[4]

This paper presents the concept of Knowledge Tracing as a method for modeling the acquisition of procedural knowledge in educational settings. Knowledge Tracing aims to track the learning progress of students as they acquire new skills and knowledge over time. By incorporating Bayesian Knowledge Tracing (BKT) models, which utilize probabilistic inference techniques, this approach can infer the probability of student mastery for specific skills based on their observed performance. The paper discusses the theoretical underpinnings of Knowledge Tracing and its application in educational contexts, highlighting its utility in providing insights into individual student learning trajectories. Through empirical evaluations and case studies, Knowledge Tracing demonstrates its effectiveness in predicting student performance and informing instructional interventions. This paper contributes to the field of user modeling and adaptive learning systems by offering a principled framework for assessing and supporting the acquisition of procedural knowledge in educational environments.

Merits: BKT enables individualized learning by tracking each student's knowledge state and predicting their future performance on specific concepts. This personalized approach can help educators tailor interventions and support to meet the unique learning needs of each student.

Demerits: BKT assumes that student mastery of knowledge concepts is binary (either mastered or not mastered). This simplistic representation may not fully capture the nuances of student learning progress, such as partial mastery or varying levels of understanding.

2.3 C. Yeung and D. Yeung, “Addressing two problems in deep knowledge tracing via prediction-consistent regularization,” in Proc. 5th Annu. ACM Conf. Learn. Scale, London, U.K., (2018).[19]

This paper introduces a novel approach to addressing two critical challenges in Deep Knowledge Tracing (DKT) models through prediction consistent regularization techniques. Deep Knowledge Tracing plays a pivotal role in modeling and predicting student learning progress in intelligent tutoring systems. However, DKT models often face issues related to overfitting and the interpretability of their predictions. To tackle these challenges, this study proposes a regularization framework that promotes prediction consistency across diverse training instances. By incorporating prediction-consistent regularization into the DKT architecture, the model can mitigate overfitting while enhancing the interpretability and generalization of its predictions. Through empirical evaluations on educational datasets, the effectiveness of the proposed approach is demonstrated, showcasing improvements in predictive performance and model robustness. This work contributes to the advancement of Deep Knowledge Tracing techniques by offering practical solutions to common challenges encountered in educational data analysis. Moreover, it lays the groundwork for future research in regularization techniques tailored to enhance the reliability and interpretability of predictive models in educational settings.

Merits: DKT+ aims to improve the model's ability to reconstruct the observed input data accurately. DKT+ can enhance the model's capacity to capture and represent student learning interactions effectively.

Demerits: Demerits: DKT+ introduces additional regularization terms to address specific issues in the original DKT model, which can increase the complexity of the model.

2.4 J. Zhang, X. Shi, I. King, and D. Yeung, “Dynamic key-Score memory networks for knowledge tracing,” in Proc. 26th Int. Conf. World Wide Web, Perth, Australia, (2017).[21]

This paper presents Dynamic Key-Score Memory Networks (DKVMNs) as an innovative approach for knowledge tracing in educational settings. Knowledge tracing predict a student learning progress over time. DKVMNs extend the traditional Key-Score Memory Network (KV-Mem Nets) by introducing dynamic mechanisms that adaptively update memory representations based on student interactions. By incorporating attention mechanisms and recurrent neural networks, DKVMNs can capture the temporal dynamics of student learning behavior and facilitate more accurate predictions of knowledge mastery. Experimental results on educational datasets demonstrate the effectiveness of DKVMNs in outperforming baseline models, showcasing their potential for enhancing adaptive learning systems. This paper contributes to the advancement of knowledge tracing techniques by introducing dynamic memory architectures tailored to capture the evolving nature of student learning trajectories. Furthermore, it paves the way for future research in leveraging dynamic neural network models for educational data analysis and personalized learning interventions.

Merits: DKVMN incorporates a memory component that allows the model to store and retrieve information dynamically. This memory-augmented architecture enables the model to retain relevant knowledge representations over time, facilitating more effective learning state updates and performance predictions.

Demerits: The use of memory in DKVMN introduces additional memory overhead, especially as the size of the memory bank grows with the number of key-Score pairs stored.

2.5 C. Yeung, “Deep-IRT: Make deep learning based knowledge tracing explainable using item response theory,” in Proc. 12th Int. Conf. Educ. Data Mining, (2019).[20]

This paper introduces Deep-IRT, a novel approach that aims to enhance the explainability of deep learning-based knowledge tracing models by integrating principles from Item Response Theory (IRT). Knowledge tracing techniques are essential for understanding and predicting student learning progress in educational environments. However, deep learning-based models often lack interpretability, hindering their practical utility in educational settings. Deep-IRT addresses this limitation by incorporating latent skill factors inspired by IRT into the deep learning architecture. By modeling the relationship between student responses and latent skills, Deep-IRT provides interpretable insights into the factors influencing student performance. Experimental evaluations demonstrate the effectiveness of Deep-IRT in improving model interpretability while maintaining competitive predictive performance. This work contributes to bridging the gap between deep learning methodologies and traditional psychometric theories, offering a promising avenue for developing explainable knowledge tracing models. Additionally, it opens new possibilities for leveraging insights from educational psychology to inform the design of advanced learning analytics techniques.

Merits: The use of IRT allows for a more transparent understanding of how learners' abilities and knowledge concepts' difficulty levels influence performance predictions, providing insights into the underlying mechanisms of the model.

Demerits: Deep-IRT's performance may be sensitive to variations in the distribution of data or the characteristics of learners and knowledge concepts

CHAPTER 3

PROBLEM STATEMENT

Knowledge tracing models consider only the conceptual understanding of knowledge and do not account for differences among individual exercises. It does not consider the relevant prior knowledge so the learner's relevance knowledge is not accountable. The correctness of prediction performance is not accurate.

3.1 OBJECTIVES

- To propose Precise knowledge tracing (PKT) to capture the actual difference among the exercises and actual difference in relevant prior knowledge.
- To capture relevant prior knowledge from historic learning interactions and then obtain individual exercises representation by combining relevant prior knowledge and raw exercises representation.
- To improve the prediction performance accuracy.

3.2 SCOPE

- Online learning platform
- Intelligent tutoring systems
- Adaptive learning technologies

3.3 SYSTEM DESCRIPTION

The system focuses on improving knowledge tracing models used in intelligent tutoring systems to assess learners' learning states and predict future performance. Traditional KT models have limitations in ignoring exercise-level differences and neglecting the impact of individual prior knowledge. The system introduces the PKT model, which captures actual differences among exercises and individual prior knowledge. The PKT model leverages an integration function to explore effective exercise representation by combining knowledge concept embedding and exercise embedding. The "overfitting" problem caused by direct use of exercises in traditional KT models. The PKT model incorporates an effective attention mechanism to capture the relevance between assessment exercises and historical interactions, enabling the system to better predict learners' performance based on their prior knowledge and learning patterns. The system acknowledges the challenges of obtaining relevant datasets for modeling exercise differences and prior knowledge. It highlights the importance of considering semantic similarity among exercises and the need to address data sparsity issues in education datasets.

CHAPTER 4

REQUIREMENT SPECIFICATION

SYSTEM REQUIREMENT

The requirement analysis is the technical requirement of the software product. It is the first step in the requirement analysis process. The requirements also provide useful constraints from a user, an operational and administrative perspective. It is a parameter which describes its user interface, hardware, and software requirements. The requirement analysis has two factors.

- Functional requirements.
- Non-functional requirements.

4.1 FUNCTIONAL REQUIREMENTS

A functional requirement is a declaration of the intended function of a system and its components.

4.1.1 Hardware requirements

Processor	- Intel (R) Core (TM) i5-12450H
Graphic Processor	- NVIDIA GEFORCE RTX 2050
Speed	- 4 GHz
Ram	- 16 GB and above
Hard Disk	- 1 TB

4.1.2 Software requirements

Operating System	- Windows or Linux
Programming language	- Python
IDE	- Visual Studio Code or Pycharm Community.

4.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements or NFRs are a set of specifications that describe the system's operation capabilities and constraints and attempt to improve its functionality. These are basically the requirements that outline how well it will operate including things like speed, security, reliability, data integrity, etc.

Scalability

- The capability of a system, network or process to handle a growing amount of work.

Reliability

- To quantify the probability that a system performs its intended function correctly throughout its mission time, or its complement Score.

Performance Characteristics

- Speed, throughput and execution times depends on the size of the input dataset.

Security

- Provide security of the code from unknown uninvited guest.

Flexibility

- Supports easy adaptation of the system to meet future requirements and changes through configuration, rather than new development.

5.1 SYSTEM ARCHITECTURE

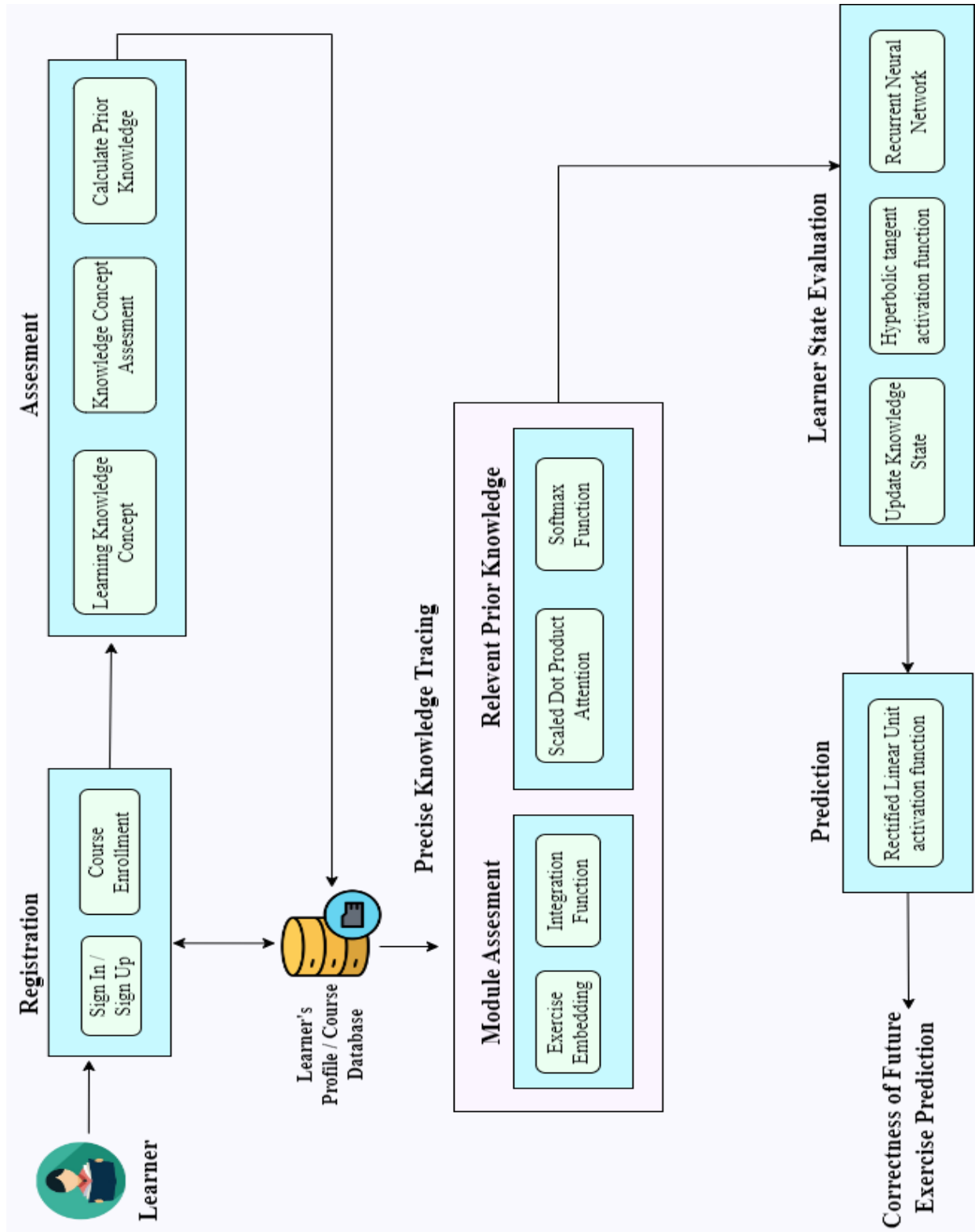


Fig 5.1 System Architecture

5.2 MODULE DESCRIPTION

- ❖ Registration
- ❖ Assessment
- ❖ Precise Knowledge Tracing
- ❖ Learner state evaluation
- ❖ Prediction

5.2.1 Registration

The Registration module focuses on managing learner registration details and tracking enrollment status. It handles administrative tasks related to course registration, such as verifying user information, updating enrollment records, and managing course capacities. This module ensures smooth and efficient registration processes for learners.

5.2.1.1 Learner Sign Up / Sign In

This module is responsible for managing the user authentication and registration process within the learning platform. It allows new users to sign up for accounts by providing necessary information and existing users to sign in securely. This module ensures that learners can access the platform and their personalized learning content.

5.2.1.2 Course Enrollment

The Course Enrollment module facilitates the process of learners enrolling in specific courses or programs offered on the platform. It includes features such as browsing available courses, viewing course details, selecting desired courses, and registering for them. This module ensures that learners can easily discover and enroll in courses that align with their learning goals.

5.2.2 Assessment

This module contains the learner's learning material, exercise problem, knowledge concept relation and conduct exam to evaluate a prior knowledge. Each module contains at least five knowledge concept and had some exercises problem based on knowledge concept. The correctness of exercise is

5.2.2.1 Learning Knowledge Concept

Once learners enroll in the course, they gain access to materials such as video streaming and notes to learn about different knowledge concepts through various modules, each containing different types of concepts and exercises. Offering constructive feedback and fostering reflection are crucial elements of the learning journey.

5.2.2.2 Knowledge Concept Assessment

After learning the knowledge concept, the learner must attend the assessment exam. The exercise may differ from the weightages of the Question(strong, medium, hard). To differentiate the actual difference among exercises covering the same knowledge concept, as well as preserve the correlation of knowledge concept, the combination of common information and particular information is applied to represent exercise representation.

5.2.2.3 Calculate Prior Knowledge

Observe students' interactions, discussions, and responses during class discussions or informal activities related to the topic. How much knowledge gain after learn the concept and attend the assessment. If learner's correctly response the exercise, then this module should assume they learn the concept and their response are stored in database.

5.2.3 Precise Knowledge Tracing

PKT is advanced module focuses on tracking and analyzing learner progress at a granular level. It enables the system to monitor individual knowledge acquisition, identify learning patterns, and detect knowledge gaps. Precise knowledge tracing allows for personalized learning interventions and adaptive content delivery based on learners' specific needs and learning trajectories.

5.2.3.1 Module Assessment

The module assessment is used to trace the individual learner's current response. In this assessment, the exercises problem contains from different module from different knowledge concept. What knowledge are the learners currently retaining?

5.2.3.1.1 Exercise Embedding

Each exercise is represented as a dense vector in a continuous vector space. The embedding vector captures the semantic meaning and relationships of the exercise based on its features and context. Exercises with similar characteristics or concepts are expected to have similar embedding vectors in the continuous space. This allows the model to generalize better and make predictions based on the relationships learned from the embeddings. To differentiate the actual difference among the exercise covering the same knowledge concept, we should acquire a Question (Q_t), Key (K_t) and Score (V_t) as

$$Q_t = WQ \cdot \phi(k_t, e_t) \quad (1)$$

$$K_t = WK \cdot \phi(k_t, e_t) \quad (2)$$

$$V_t = WV \cdot \phi(k_t, e_t) \quad (3)$$

where $\phi(k_t, e_t)$ indicates the integration function, it incorporates the knowledge concept embedding and exercise embedding to obtain exercise representation. At each exercising step t , the input to the network is a combined encoding with both exercise embedding x_t and the corresponding score r_t . Since the exercise with right score (i.e., 1) and wrong score (i.e., 0) have different influences on student states during the exercising process, we need to find a appropriate way to distinguish these different effects for a specific student.

Methodology-wise, we first extend the score value r_t to a feature vector $=0$ with the same 2^d dimensions of exercise embedding x_t and then learn the combined input vector $V_t \in \mathbb{R}^{2d}$ as:

$$V_t = \begin{cases} [V_t \oplus r_t], & \text{if } r_t = 1 \\ [r_t \oplus V_t], & \text{if } r_t = 0 \end{cases} \quad (4)$$

V_t denotes integrated vectorial-representation for learner has responded exercise e_t at time step t .

Effectiveness of actual differences among exercise

In our experiment, the actual differences among exercise will be loaded. Because, each exercise has different difficulty in same knowledge concepts. And it's useful to predict the learner capabilities.

For example: In same knowledge concept (loop), the exercises are

<pre>1. int i = 1; do { printf("%d\n",i); i++; } While (i < 6);</pre>	<pre>2. for (int i = 0; i < 10; i++) { if (i == 4) { continue; } printf ("%d\n", i); }</pre>
---	---

Exercise embedding algorithm

Input: Raw interaction sequence $X = (x_1, x_2, \dots, x_n)$ and vector representations Q , K , and V for each exercise response

1. Initialize an empty sequence S

2. Check the length of the interaction sequence:

a. If the length is smaller than N :

Pad the sequence with triples (exercise, knowledge concept, response) to the right side until the length equals N

b. If the length is greater than N :

Split the interaction sequence into subsequences of length N

3. Extend the vector V by incorporating a full zero vector $r_t \in \{0\}^d$ with the same dimensions as V

4. Create an integrated vector representation $V_t \in \mathbb{R}^{2d}$ based on the correctness of the learner's response:

a. If $r_t = 1$ (correct response):

$$V_t = [V_t \oplus r_t]$$

b. If $r_t = 0$ (wrong response):

$$V_t = [r_t \oplus V_t]$$

where \oplus denotes vector concatenation and N is the maximum length that can be processed by our model

5. Output: Integrated vector representation V_t for the learner's response on exercise e_t at time step t

5.2.3.1.2 Integration Function:

The Integration Function module focuses on integrating different components of the learning system, such as exercise embedding, difficulty and assessment result. The exercise representation not only needs to retain the common information of knowledge concepts and also added some information with exercise level.

$$\diamond \text{ concatenation: } \varphi(k_t, e_t) = [k_t \oplus e_t] \quad (5)$$

$$\diamond \text{ multiplication: } \varphi(k_t, e_t) = k_t * E e_t \quad (6)$$

$$\diamond \text{ concatenation and multiplication: } \varphi(k_t, e_t) = [k_t \oplus (k_t * E e_t)] \quad (7)$$

5.2.3.2 Relevant Prior Knowledge

The Relevant Prior Knowledge (RPK) is a mechanism introduced to explore and utilize the learners' prior knowledge of the assessment exercise from their historical performance data. It aims to extract relevant information from past interactions to predict the learner's performance on the current assessment exercise accurately. The RPK is designed to address the challenge of scattered historical learning interactions by focusing on the specific knowledge and skills required for the assessment exercise. By aggregating and analyzing the relevant prior knowledge, the model can better understand the learner's capabilities and tailor the predictions accordingly.

Effectiveness of relevant prior knowledge

In our experiment, the RPK of the correct exercise also loaded. Because, how the learner has gain prior knowledge for perspective exercise. So its useful to predict the next exercise performance.

Example: Let's consider a simplified example with historical exercises' performances, responses, and relevance weights:

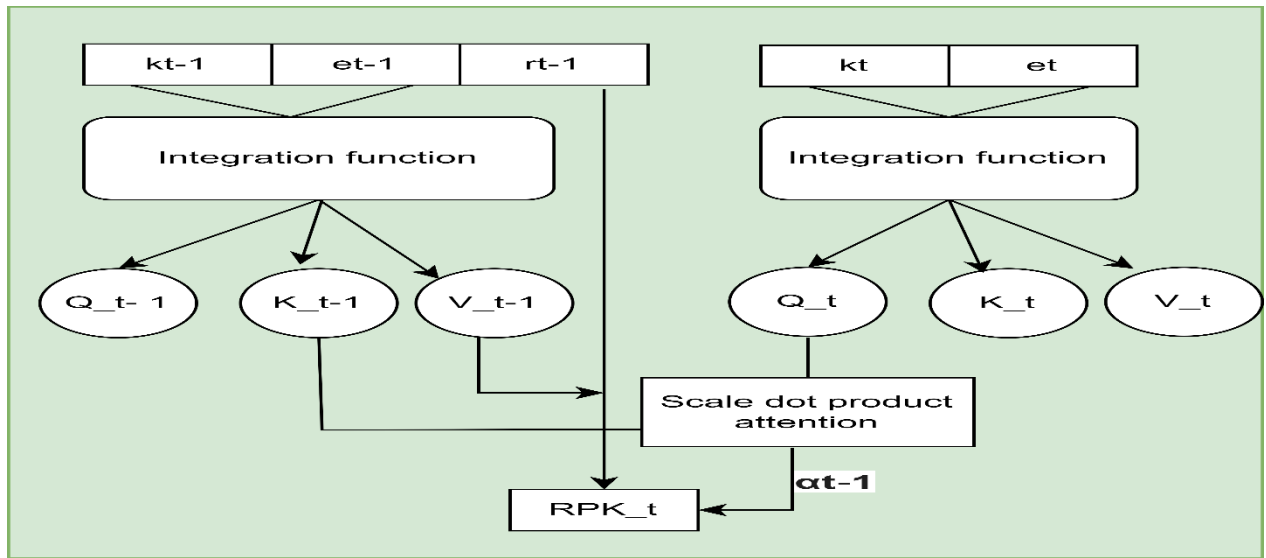


Fig 5.2.1 Calculating RPK

Historical exercises' performances: $\{V1 = 0.8, V2 = 0.6, V3 = 0.7\}$

Historical responses: $\{r1 = 1, r2 = 0, r3 = 1\}$ - Relevance weights: $\{\alpha1 = 0.5, \alpha2 = 0.3, \alpha3 = 0.4\}$

Current assessment exercise: $et = 0.9$

Using the algorithm:

1. Initialize $RPK_t = 0$

2. For each historical exercise i:

a. Calculate relevance score: -

For $i = 1$: $\text{relevance } 1 = 0.5 * \text{similarity}(0.8, 0.9) = 0.5 * 0.8 = 0.4$

For $i = 2$: $\text{relevance } 2 = 0.3 * \text{similarity}(0.6, 0.9) = 0.3 * 0.6 = 0.18$

For $i = 3$: $\text{relevance } 3 = 0.4 * \text{similarity}(0.7, 0.9) = 0.4 * 0.7 = 0.28$

b. Update $RPK_t = 0 + (0.4 * 1) + (0.18 * 0) + (0.28 * 1) = 0.4 + 0 + 0.28 = 0.68$

3. Return $RPK_t = 0.68$ as the Relevant Prior Knowledge for the assessment exercise e_t

RPK weightage algorithm:

Inputs:

Historical exercises' performances: $\{V_1, \dots, V_{t-1}\}$ (performance on historical exercises)

Historical responses: $\{r_1, \dots, r_{t-1}\}$ (responses on historical exercises)

Relevance weights: $\{\alpha_1, \dots, \alpha_{t-1}\}$ (weights indicating the relevance of historical interactions)

Assessment exercise: e_t (current assessment exercise)

1. Initialize $RPK_t = 0$ (Initial Relevant Prior Knowledge for the assessment exercise e_t)

2. For each historical exercise i from 1 to $t-1$:

a. Calculate the relevance score for historical exercise i :

$$relevance_i = \alpha_i * similarity(V_i, e_t)$$

where $similarity(V_i, e_t)$ is a function that computes the similarity between the historical exercise performance V_i and the current assessment exercise e_t .

b. Update the Relevant Prior Knowledge (RPK) for the assessment exercise e_t :

$$RPK_t = RPK_t + relevance_i * r_i$$

(Weighted sum of historical responses based on relevance scores)

3. Return RPK_t as the Relevant Prior Knowledge for the assessment exercise e_t .

5.2.3.2.1 Scaled Dot Product Attention

The scaled dot production technique is used to concatenate the historical key value (K_t) and current exercise Question (Q_t) to obtain how much knowledge gather on current exercise Question based on knowledge concept.

$$z_i = \begin{cases} Q_t k^i / \sqrt{d} & \text{if } i \in (1, t-1) \\ \text{masking}(e_i) & \text{if } i \in (1, N) \end{cases} \quad (8)$$

Where Q is current exercise Question and K is a weightage of historic exercise.

5.2.3.2.2 Softmax Function

To ensure that exercises responded to after time step t are not considered in the calculation of relevance weights, a masking operation is applied. This masking operation helps focus only on the relevant exercises that occurred before the assessment exercise in the sequence.

$$\alpha_i^j = \text{softmax}(Z_i) = e^{Z_i} / \sum_{j=1}^N e(Z_i)$$

$$Rpk = \sum_{j=1}^N \alpha_j V_i \quad (9)$$

Where α_t^p represents the attention weights that quantify the relevance or importance of previous exercises in relation to the current assessment exercise. The relevance weights determining how much influence past interactions have on predicting a learner's performance on a new exercise.

5.2.4 Learner State Evaluation

This model treats the learner's interaction records as a sequential task, where each exercise-response pair contributes to the learner's evolving

knowledge state. Individual exercise-response representations are used as inputs to a Long Short-Term Memory (LSTM) network to capture the sequential nature of the learning process.

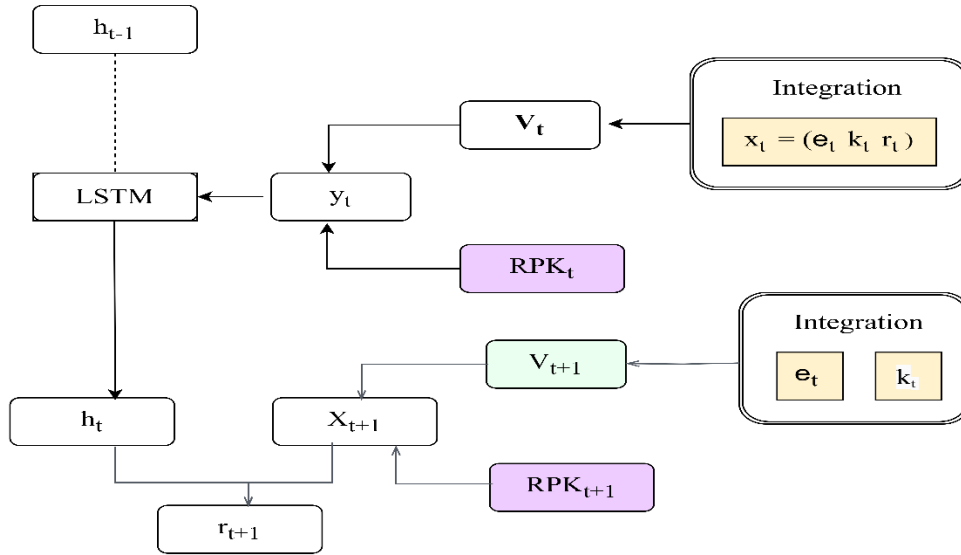


Fig 5.2.2 PKT framework

5.2.4.1 Recurrent Neural Network

RNNs are a type of neural network designed to handle sequential data by maintaining a hidden state that captures information about previous elements in the sequence. The LSTM network processes the sequential exercise-response data to update the learner's knowledge state over time. It consists of gates (input, forget, output) that regulate the flow of information, allowing the model to retain relevant information and forget less important details. The LSTM network's hidden states capture the learner's evolving understanding and mastery of educational concepts.

5.2.4.2 Hyperbolic tangent activation function

1. *Input: Sequential interaction records of the learner (exercise-response pairs)*
2. *Initialize the LSTM gates and cell state:*

- Set the initial hidden state h_0 and cell state c_0 to zero vectors

3. For each time step t :

a. Calculate the input gate i_t :

$$i_t = \sigma(W_i * y_t + Z_i * h_{t-1} + b_i)$$

b. Calculate the forget gate f_t :

$$f_t = \sigma(W_f * y_t + Z_f * h_{t-1} + b_f)$$

c. Calculate the output gate o_t :

$$o_t = \sigma(W_o * y_t + Z_o * h_{t-1} + b_o)$$

d. Calculate the candidate cell state update:

$$\tilde{C}_t = \tanh(W_c * y_t + Z_c * h_{t-1} + b_c)$$

e. Update the cell state c_t :

$$c_t = f_t * c_{t-1} + i_t * \tilde{C}_t$$

f. Update the hidden state h_t :

$$h_t = o_t * \tanh(c_t) \quad (10)$$

4. Output: Updated learner knowledge states h_1, h_2, \dots, h_t at each time step t

where:

y_t represents the individual exercise-response representation at time step t

h_{t-1} represents the previous hidden state at time step $t-1$

$W_i, W_f, W_o, W_c, Z_i, Z_f, Z_o, Z_c, b_i, b_f, b_o, b_c$ are the weights and biases for the LSTM gates and cell state

σ denotes the Sigmoid activation function

\tanh denotes the hyperbolic tangent activation function

5.2.4.3 Update Knowledge Current State

The Prior Knowledge and the individual exercise response is provided more information to predict the learner's current knowledge state. By updating

the knowledge state based on sequential interactions, the model can adapt to the learner's progress and adjust predictions accordingly.

5.2.5 Prediction:

The prediction process involves utilizing a model to estimate the likelihood that a learner will correctly respond to the next exercise, which covers a specific knowledge concept. Intelligent tutoring systems to provide personalized learning experiences and recommendations based on the learner's current state of knowledge. By effectively predicting learners' responses to upcoming exercises, knowledge tracing models can assist educators and educational platforms in tailoring instruction to meet individual learning needs and optimize the learning process for each student.

Scenario: Let's consider a student, John, who has interacted with math exercises related to datatypes, operators, and if-else Knowledge concepts. We will calculate the prediction, whether John will answer a new problem correctly using the PKT model. Assumptions: John's past interactions and responses to exercises have been recorded. This model has been trained on John's historical data to capture his knowledge state dynamically. Data Representation: John's past interactions are represented as a sequence of exercise-response pairs: Exercise 1 (Datatypes): Correct, Exercise 2 (Operators): Incorrect, Exercise 3 (If Else): Correct.

Let's assume the LSTM updates John's knowledge state vector to [0.8, 0.6, 0.9] after processing his interactions. The exercise is represented as a feature vector [0.3, 0.4, 0.5] corresponding to the division concept. This model combines John's updated knowledge state vector [0.8, 0.6, 0.9] with the feature vector of the new division exercise [0.3, 0.4, 0.5]. Let's assume the model computes the prediction using a simple weighted sum:

$$\text{- Prediction} = (0.8 * 0.3) + (0.6 * 0.4) + (0.9 * 0.5)$$

- Prediction = 0.24 + 0.24 + 0.45

- Prediction = 0.93

The prediction value of 0.93 indicates the likelihood that John will answer the new loop answer correctly based on his past interactions and updated knowledge state.

5.2.5.1 Rectified Linear Unit activation function:

- 1. Input: Current knowledge state h_t and assessment exercise representation x_{t+1}*
- 2. Concatenate the current knowledge state h_t and the individual exercise representation x_{t+1}*
- 3. Apply a fully connected network to obtain a summary vector s_{t+1} by computing with the concatenated vector.*
- 4. Feed the summary vector s_{t+1} into a Sigmoid activation function to compute the odds of correctly responding to exercise x_{t+1}*
 - a. Calculate the summary vector $st+1$:*

$$s_{t+1} = \text{ReLU} (W_3 * [h_t \oplus x_{t+1}] + b_3)$$

- 5. Compute the prediction for the learner's response:*

$$r_{t+1} = \sigma(W_4 * st_{+1} + b_4) \quad (11)$$

- 6. Output: Probability of the learner correctly responding to exercise r_{t+1}*

5.3 UML DIAGRAMS

5.3.1 Use case Diagram

Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases) and any

dependencies between those use cases. The main purpose of a use case diagram is to show that system functions are performed from actor.

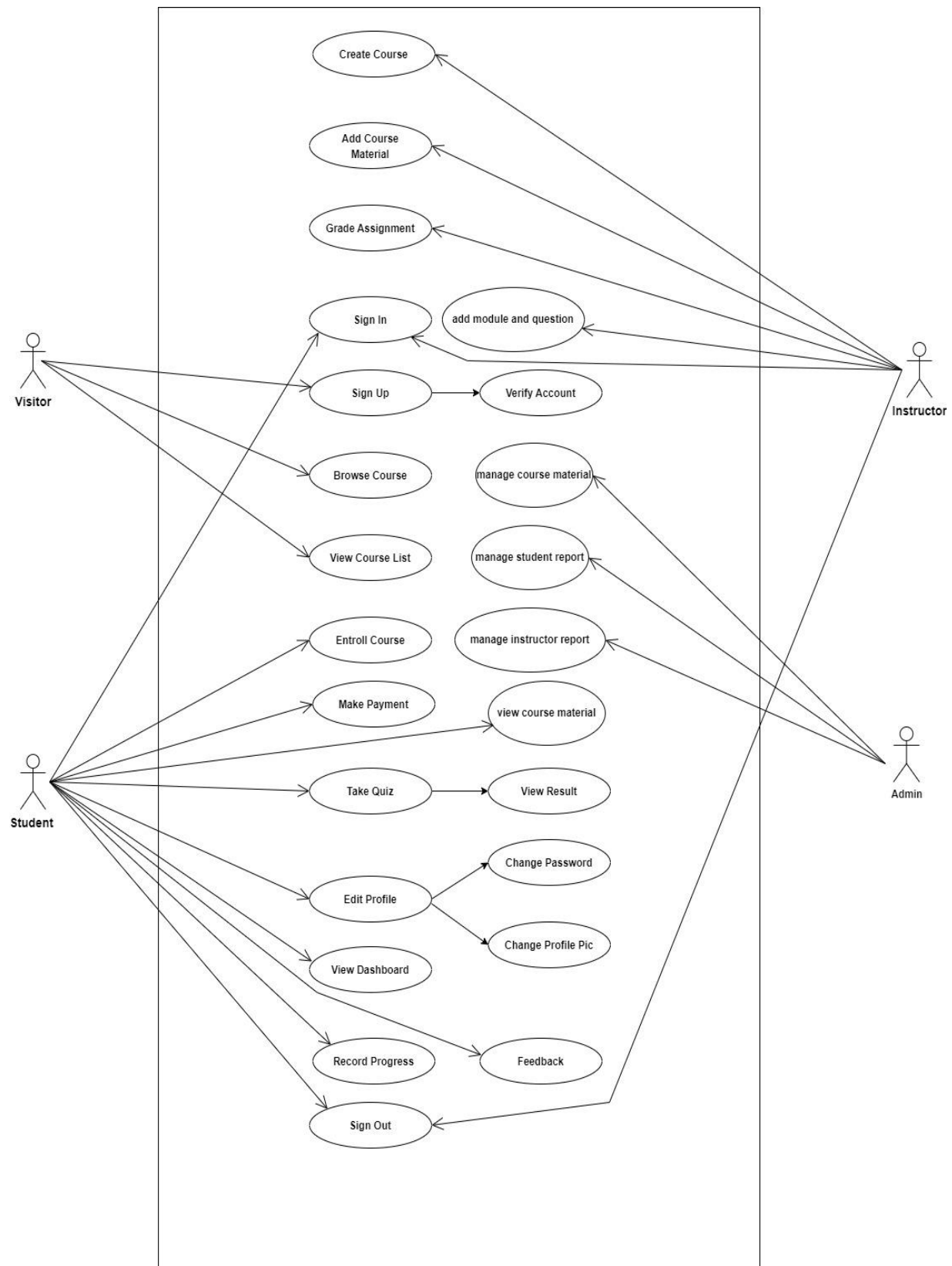


Fig 5.3.1 Use Case Diagram

5.3.2 Class Diagram

A class diagram is a static structure diagram that describes the structure of a system by showing the system's classes, their attributes and relationship between the classes. The class diagram gives detailed look about the attribute and operations of each class.

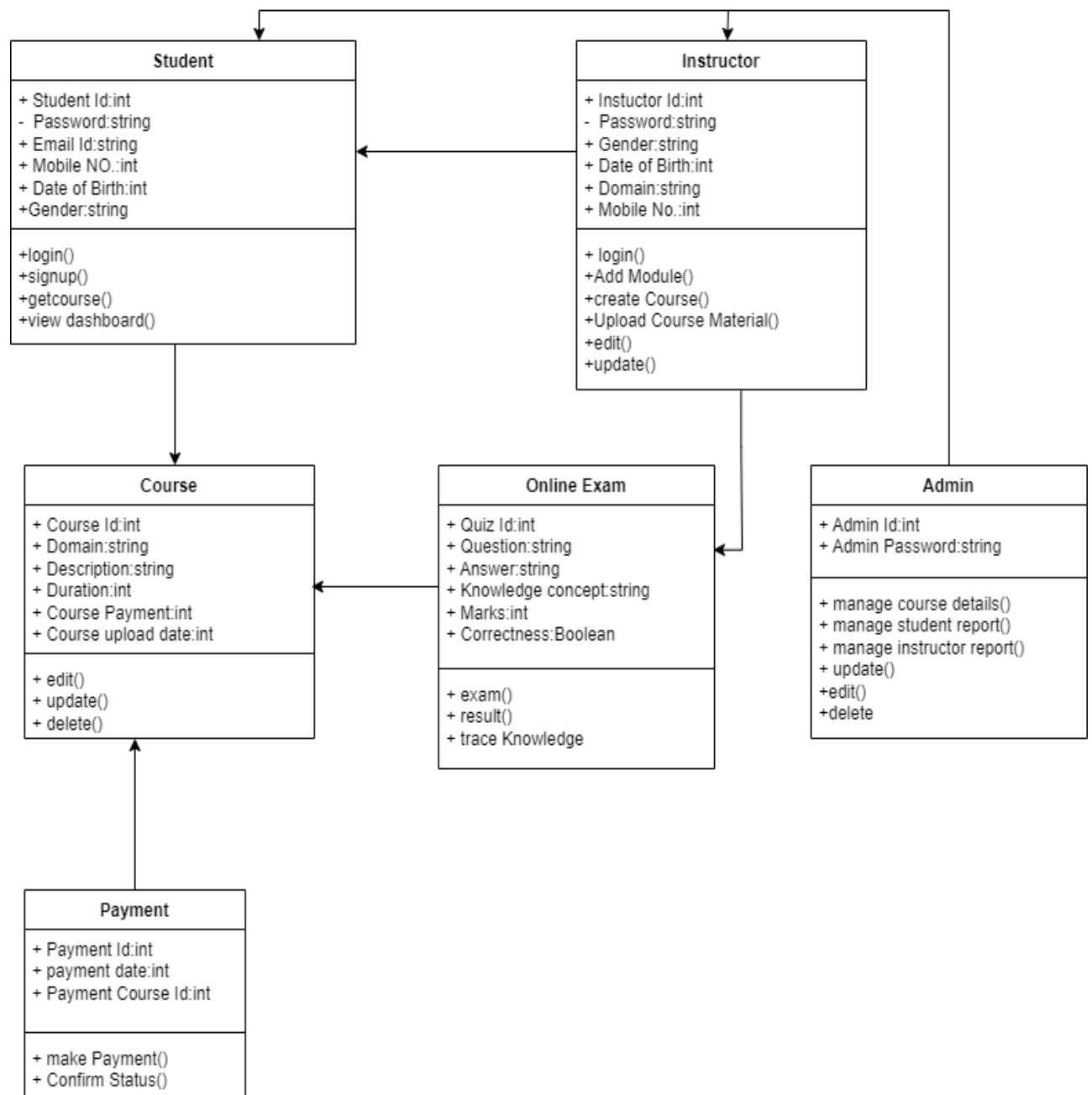


Fig 5.3.2 Class diagram

5.3.3 Sequence Diagram

Sequence diagrams are type of interaction diagrams. These interactions are modeled as exchanged of messages. Sequence diagram gives detailed interaction of the object.

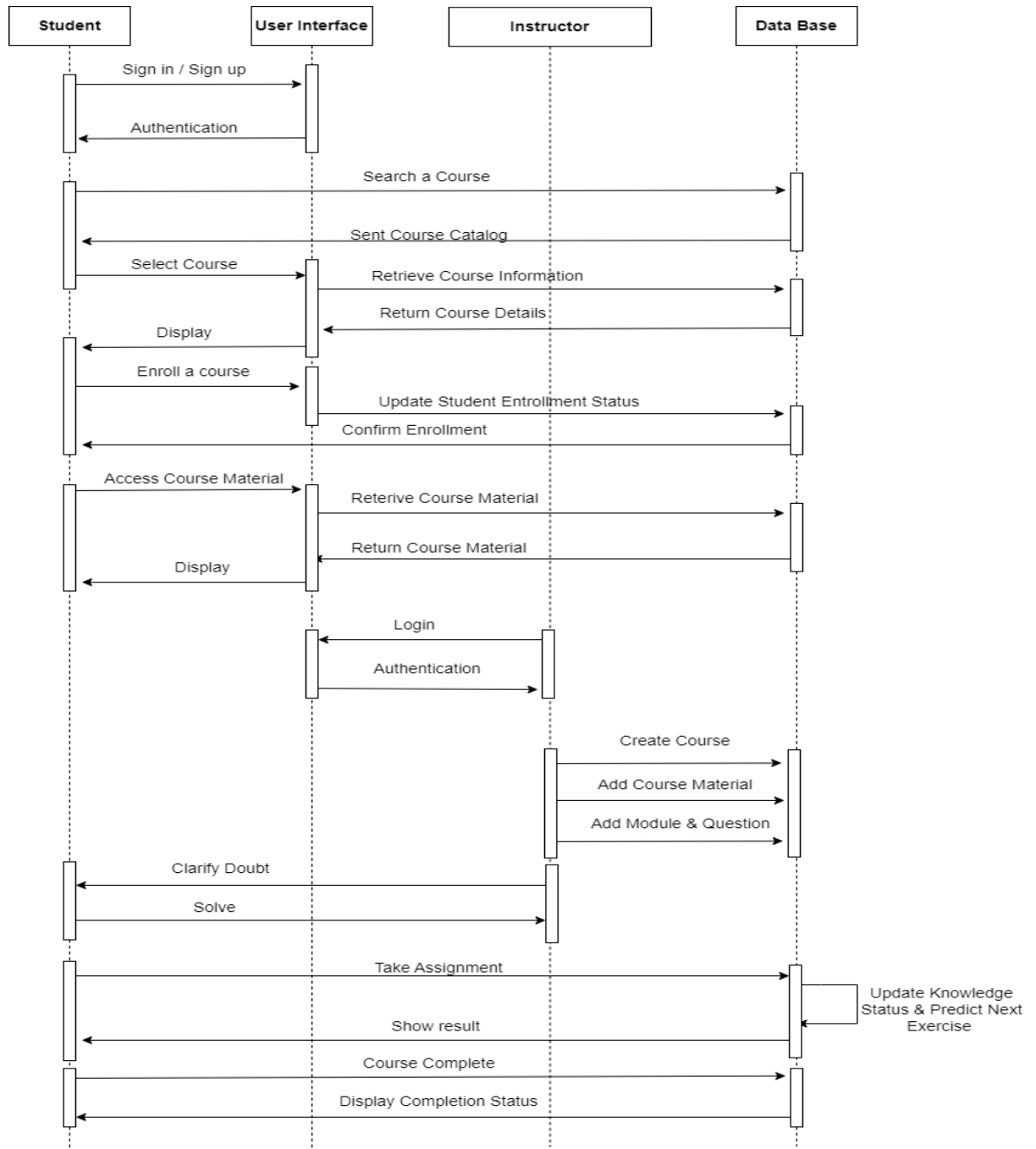


Fig 5.3.3 Sequence diagram

5.3.4 Activity Diagram

An activity diagram provides a view of the behavior of a system by describing the sequence of actions in a process.

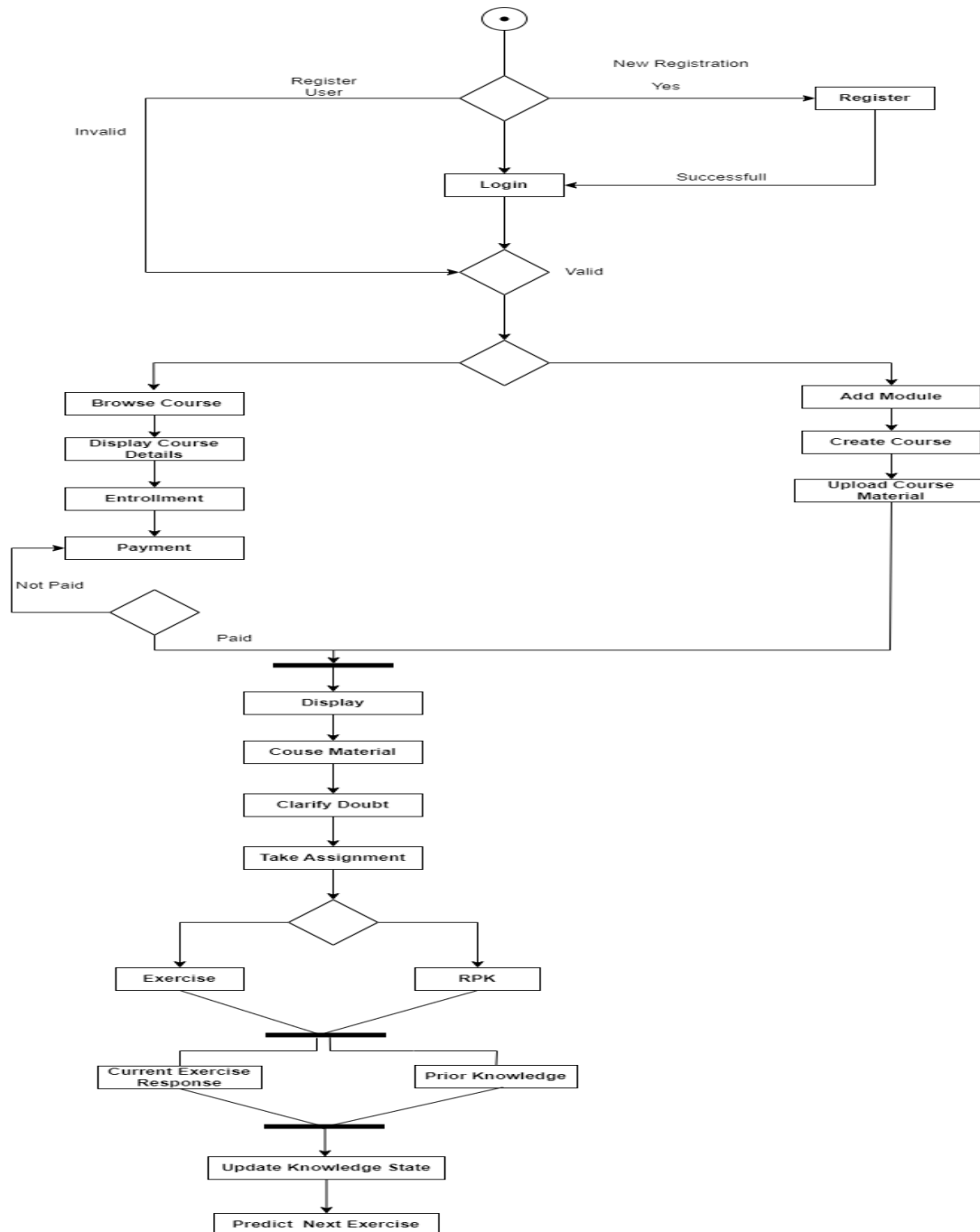


Fig 5.3.4 Activity diagram

5.3.5 State Chart Diagram

State machine diagrams are very useful to describe the behavior of objects that act differently according to the state they are in at that moment.

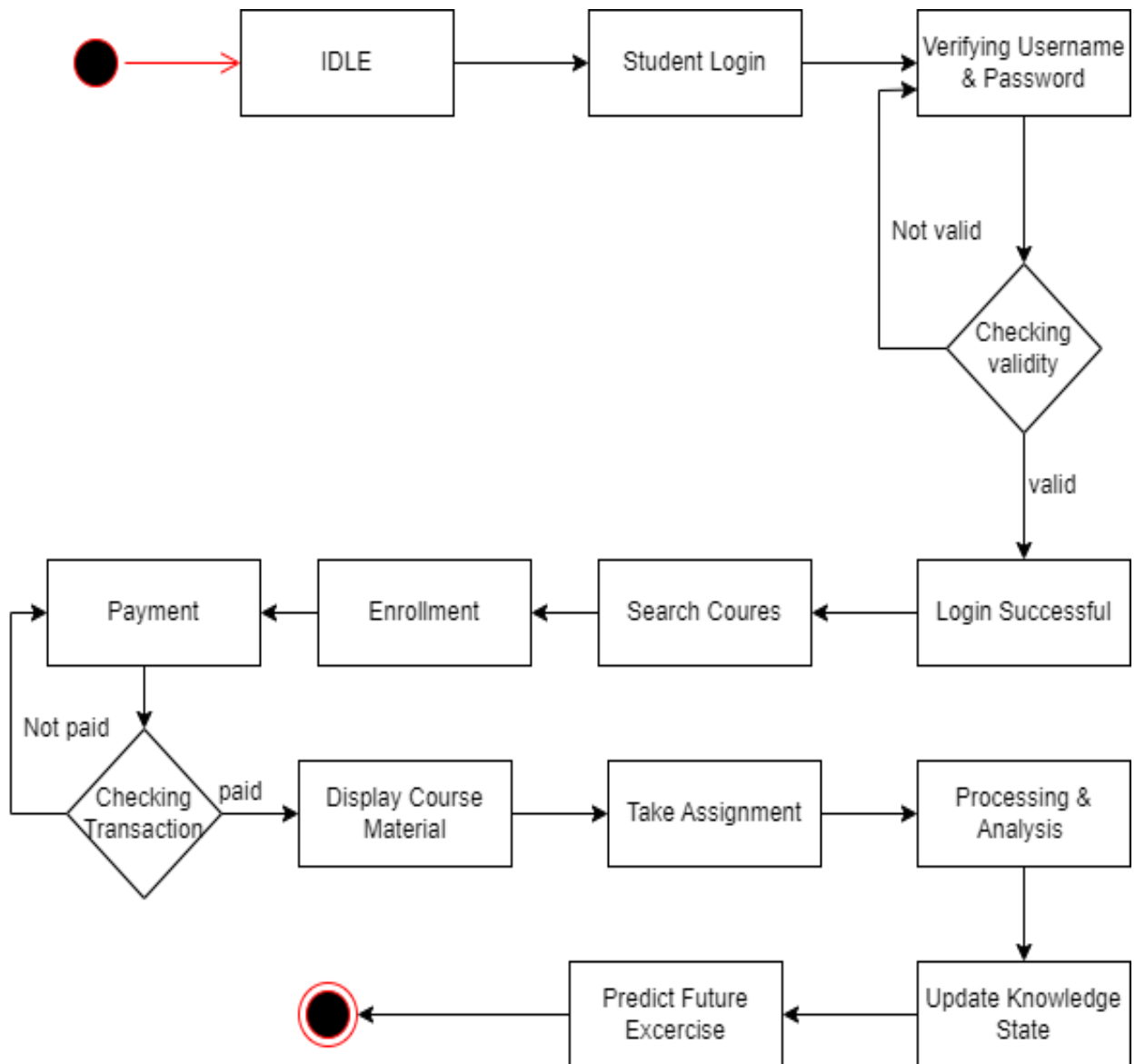


Fig 5.3.5 State chart diagram

5.3.6 Collaboration Diagram

A collaboration diagram, also known as a communication diagram, to visualize the interactions between objects or components within a system or process. It illustrates how objects or components collaborate to achieve a specific functionality or accomplish a task, showing the messages exchanged between them and the sequence of those messages.

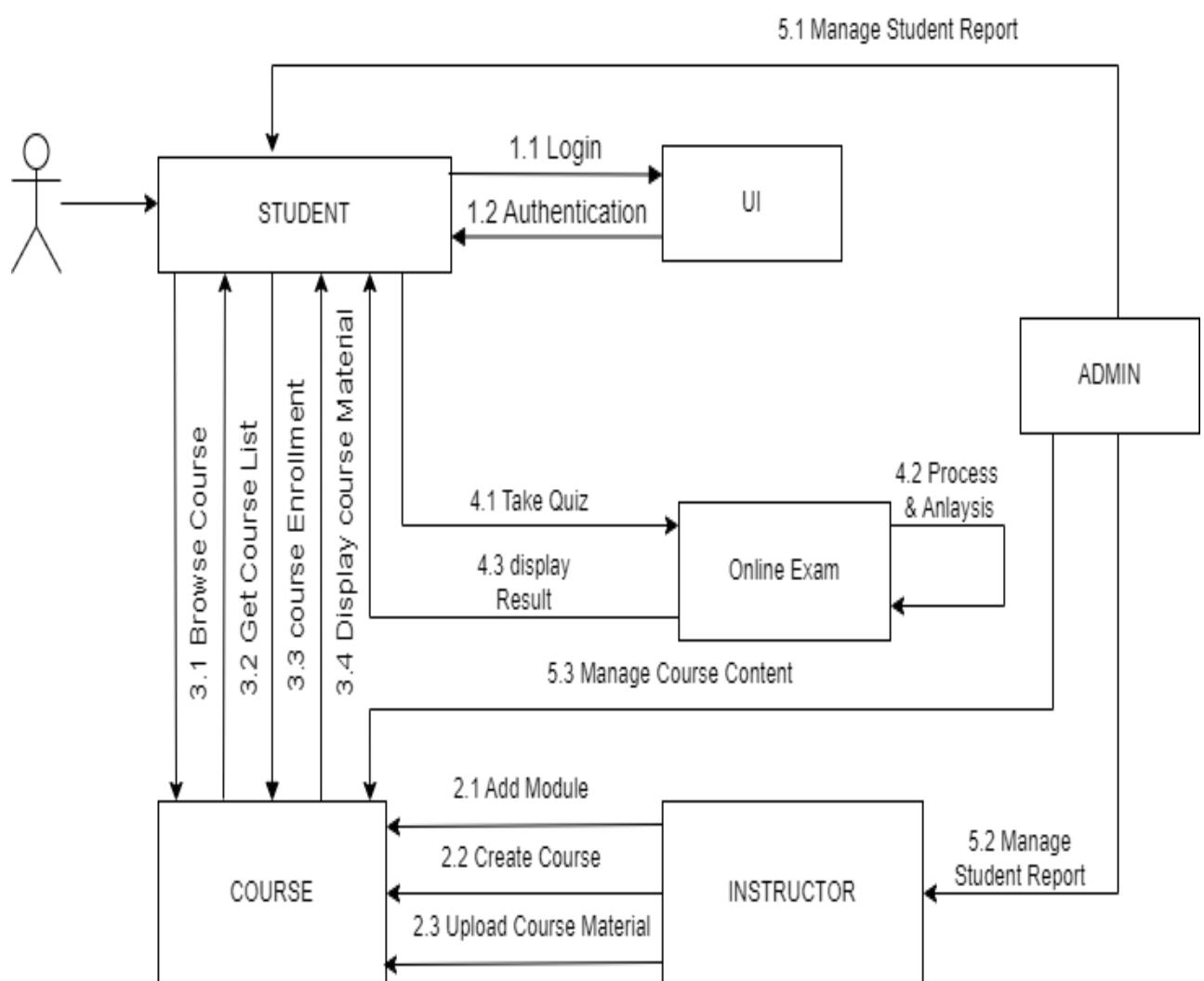


Fig 5.3.6 Collaboration diagram

5.3.7 Deployment Diagram

A deployment diagram the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system.

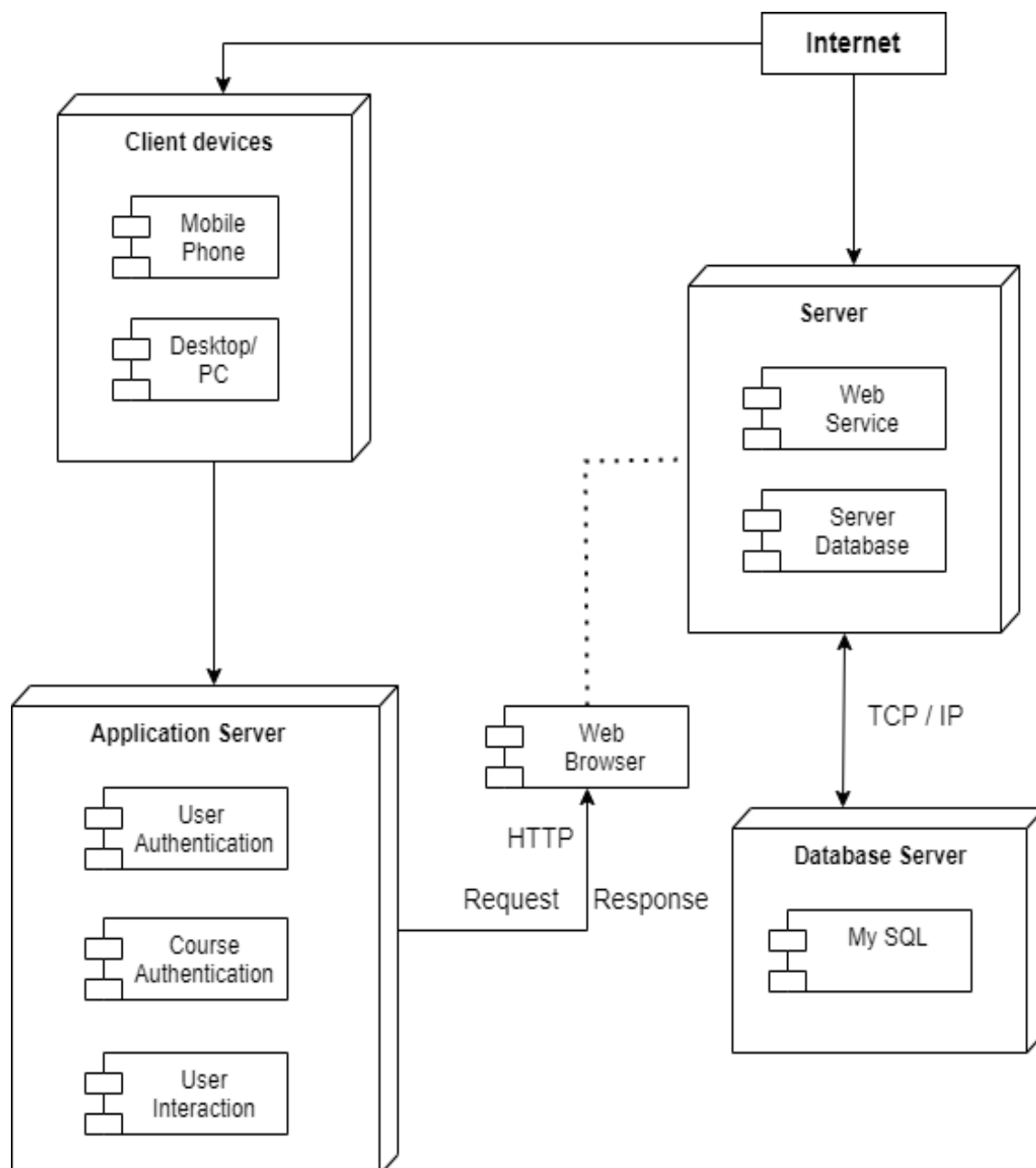
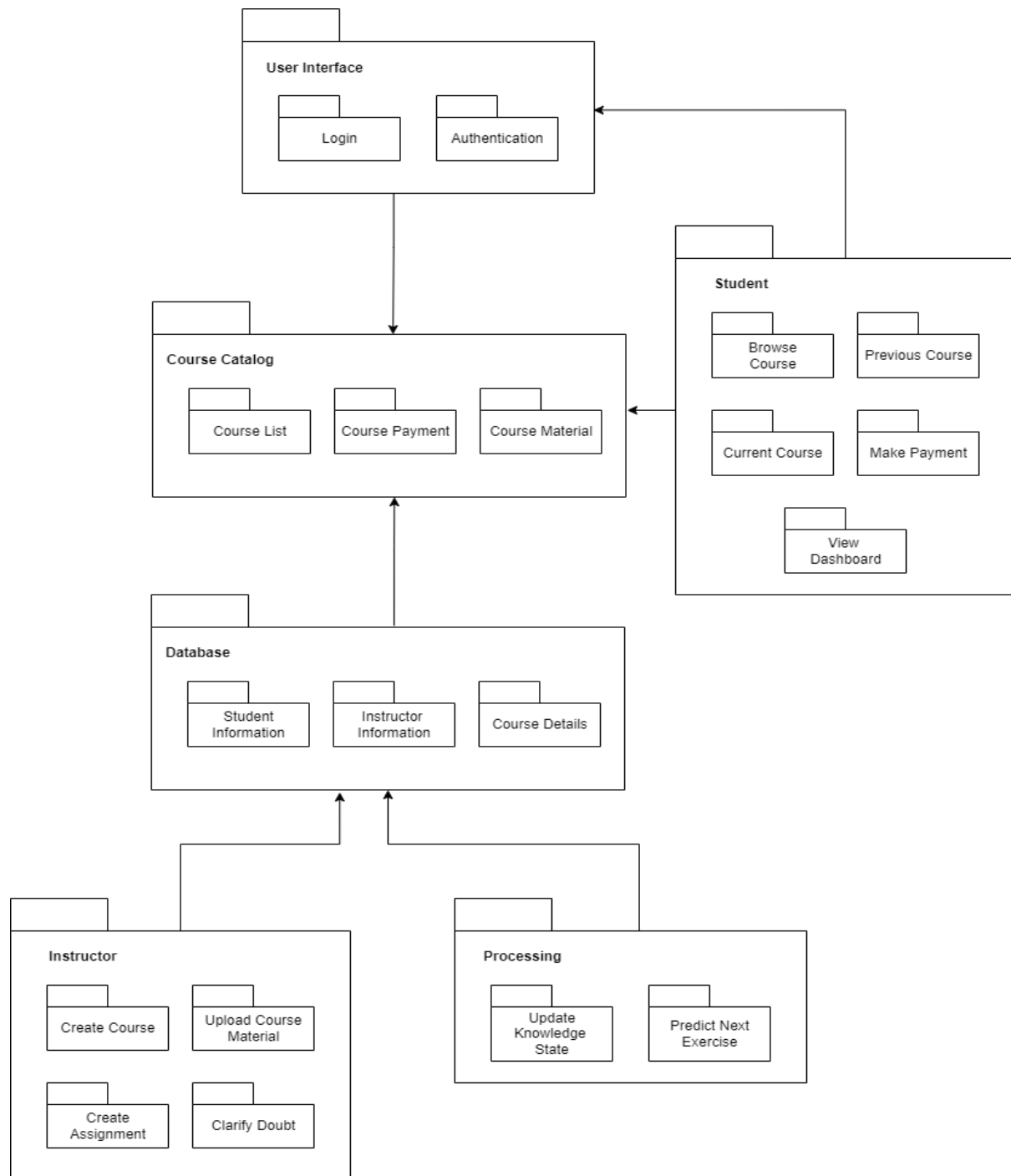


Fig 5.3.7 Deployment diagram

5.3.8 Package Diagram

Package diagram shows the arrangement and organization of model elements in middle to large scale project. Package diagram can show both structure and dependencies between sub-systems or modules.



5.3.8 Package diagram

5.4 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the 'flow' of a data through an information system. DFD is an important technique for modeling a systems high level detail by showing how input data is transformed to output results through a sequence functional transformation. DFD consists of four major components: entities, processes, data stores and data flow.

5.4.1 DFD-Level 0

Users, including students, instructors, and administrators, can securely login to the system to access relevant features and information based on their roles. Instructors can easily upload course materials such as lecture notes, presentations, readings, and other resources to the system for students to access and study. The system allows administrators to add and manage instructor details, including profiles, contact information, and course assignments. Similarly, student details such as registration information, progress tracking, and performance data can be managed efficiently. The system enables the creation and administration of online exams for students. Students can access video lectures, tutorials, or other multimedia content through the system to enhance their learning experience. The system may offer tools to manage various learning activities, assignments, and quizzes to engage students effectively.

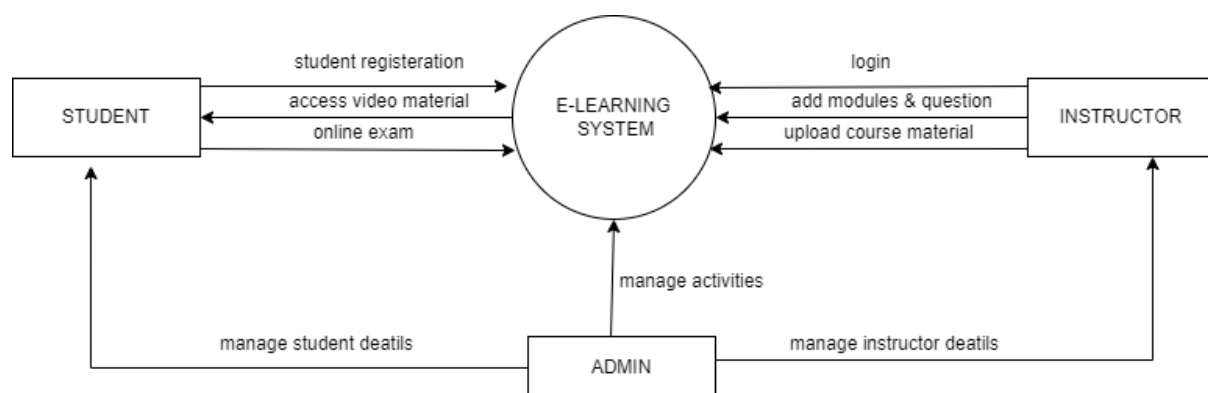


Fig 5.4.1 DFD level 0

5.4.2 DFD-Level 1

Student refers to individuals who are enrolled in courses or programs within the e-learning platform. Instructor represents educators or trainers who create and deliver educational content to students. Registration the process through which students and instructors sign up and create accounts on the e-learning platform. Activity to the interactive tasks and assignments that students engage with as part of their learning process. Administrators who manage and oversee the operations of the e-learning platform. The structured set of courses, modules, and learning materials designed to achieve specific learning objectives. Assessments and tests that students can take digitally through the e-learning platform.

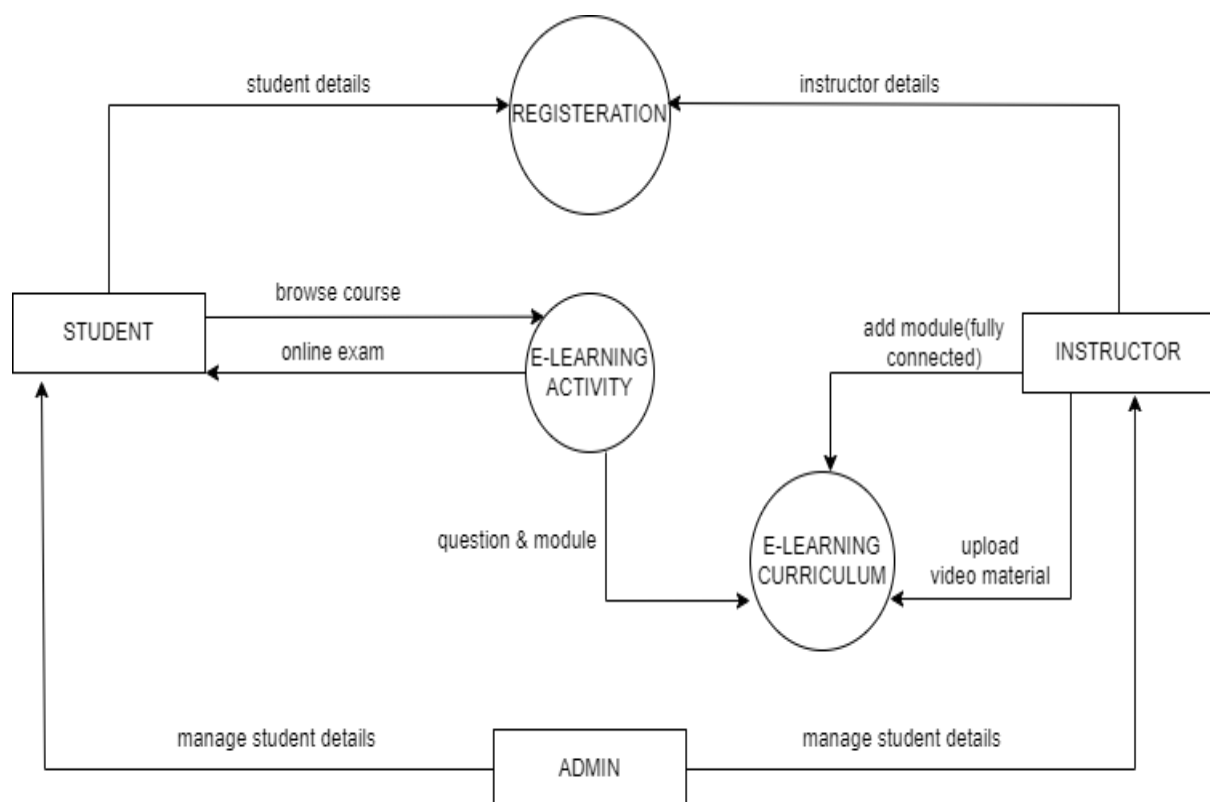


Fig 5.4.2 DFD level 1

5.4.3 DFD-Level 2

The e-learning system that provide three main user roles: admin, instructor and student. The system includes functionalities such as registration, e-learning activity, e-learning module, online examination, video tutorials, and question management. For instructors, the system allows them to view dashboard, view video tutorials, enroll students in courses, manage instructor details, manage student details, and access tutorial and question materials. Students can enroll in courses, view video tutorials, access course materials, take exams, view exam results, and update their knowledge. The system is supported by databases for registration, exams, and modules. Instructors can add modules and upload course materials, while students can take exams and view their results. Overall, the e-learning system provides a platform for instructors to create and manage courses, and for students to access educational materials, take exams, and enhance their knowledge.

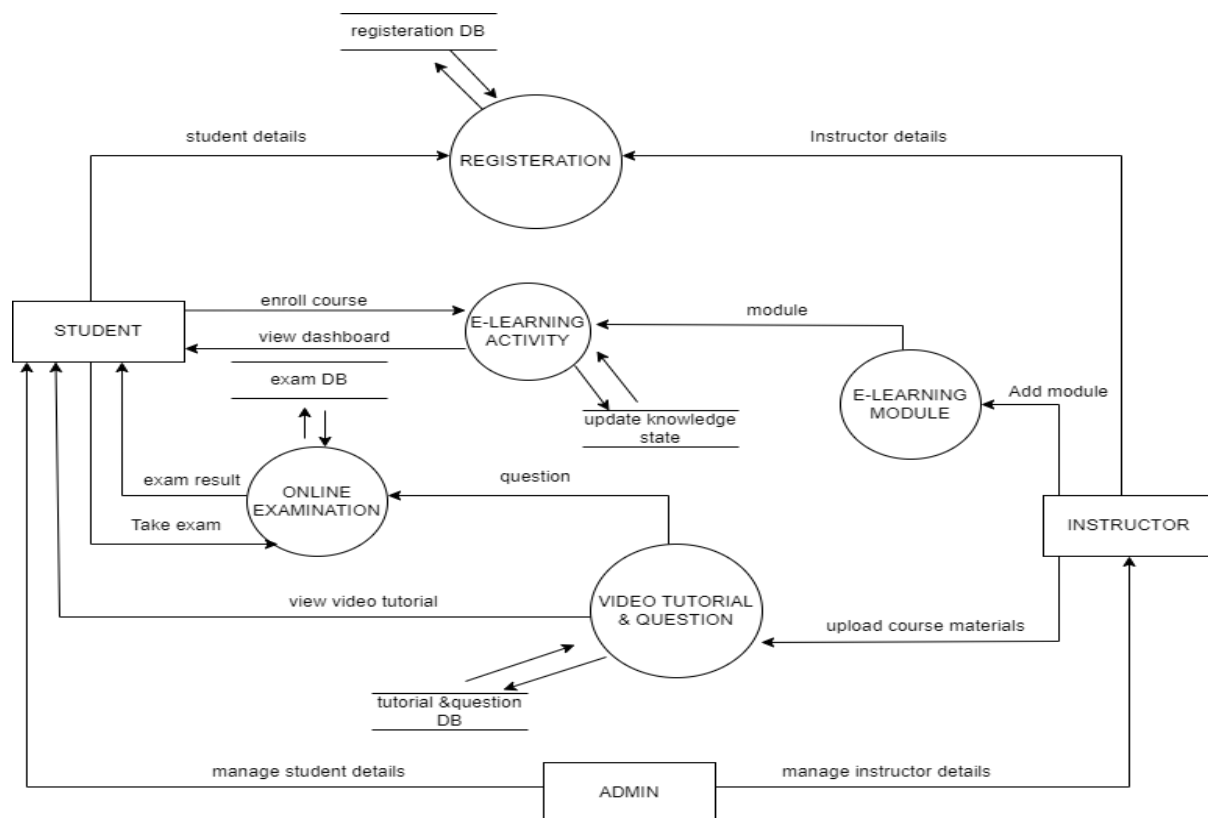


Fig 5.4.3 DFD level 2

CHAPTER 6

IMPLEMENTATION

6.1 Dataset

To evaluate our model, we employed the following three datasets: Synthetic-5, ASSISTments2009 and ASSISTments2017. All of these datasets are widely used in KT research and detailed statistics are mentioned below.

- 1) Synthetic-5: This dataset was obtained from Piechet al. [zz] with 2000 virtual learner’s learning interactions. Each learner has finished the same learning sequence of 50 exercises, and each exercise covers one of five knowledge concepts. It is worth noting that this dataset does not provide the corresponding relationship between the exercises and the knowledge concept. Hence, the integration function cannot be used to conduct the experiments on the Synthetic-5 dataset.
- 2) ASSISTments2009 (ASSIST09): This dataset was collected from ASSISTments online education platform during the school year 2009 – 2010. In the past decade, the ASSISTments2009 dataset has been the standard bench mark dataset for KT research. We used the skill-builder dataset to conduct our experiments.
- 3) ASSISTments2017(ASSIST17): This dataset was published in the 2017 ASSISTments data mining competition, and also was collected from learner’s records on the ASSISTments online education platform.

Dataset Statics

Dataset	Interactions	Exercise	Concepts	Learners
<i>Synthetic-5</i>	<i>1,00,000</i>	<i>50</i>	<i>5</i>	<i>2,000</i>
<i>ASSIST09</i>	<i>3,25,367</i>	<i>16891</i>	<i>110</i>	<i>4,151</i>
<i>ASSIST17</i>	<i>9,42,816</i>	<i>3162</i>	<i>102</i>	<i>1709</i>

Table I: Dataset

6.2 Technologies used

The project can be done successfully by using various technologies for different function or operation.

6.2.1 Python

Python is used for data analysis or machine learning tasks related to evaluating trust levels of nodes in the network, but this is not explicitly stated in the document. Python are usually used to quickly and accurately obtain the fitting results. Python's strength lies in its extensive library ecosystem.

Libraries like NumPy, tensorflow, torch, tqdm and scikit-learn are widely used for machine learning tasks. Python has a wide range of frameworks that provide pre-built libraries, tools, and templates to simplify and expedite the development process for specific applications. It is designed to be adaptable to different project sizes and requirements. Python is easy to read the CSV dataset and it takes less time to processing compare to other programming language. It emphasizes flexibility and allows developers to make choices regarding the project's structure and components.

Python modules and their version are listed below:

- ❖ Python 12
- ❖ Sklearn 0.21
- ❖ Tqdm 4.54
- ❖ Torch 1.7
- ❖ Numpy 1.19

6.3 Previous Study

In previous paper, they removed the integration function and relevant prior knowledge

- a. FGKT-NI: In this model they do not import the integration function. The knowledge concept embedding is directly applied on experiment in respective dataset (Synthetic-5).
- b. FGKT-NR: In this model they do not consider relevant prior knowledge but they use a integration function.
- c. FGKT: In this model they neglected the actual differences among exercise.

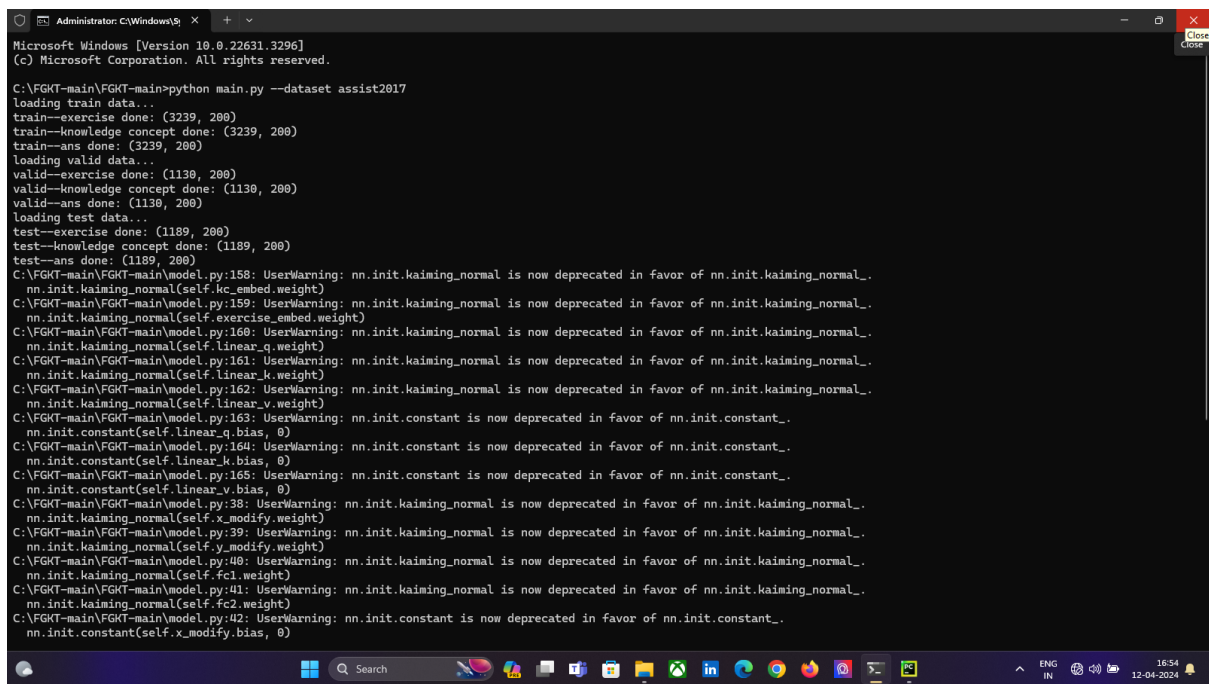
6.3.2 Performance for all KT methods

To evaluate the performance of the baselines and our model, we use the area under curve (AUC), accuracy (ACC) and root mean-square error (RMSE) as the metrics. For each dataset, 60% of learners are divided into the training set, 20% of learners are divided into the validation set and 20% of learners are divided into the test set. In validation test we used Adam optimizer function. Once the test-loss is increased than previous epoch, the early stopping counter is activated and test accuracy is increased. The maximum length of input $N=200$ and for selection to use a grid search to find the best parameter combination.

Data set	Metric	DKT	DKVMN	SAKT	FGKT	PKT
Synthetic-5	AUC	0.8118	0.8056	0.7976	0.8183	0.8300
	ACC	0.7494	0.7334	0.7067	0.7454	0.7562
	RMSE	0.4085	0.4089	0.4076	0.4072	0.4060
ASSIST09	AUC	0.7987	0.8021	0.7656	0.8134	0.8291
	ACC	0.7286	0.7324	0.7168	0.7567	0.7697
	RMSE	0.4041	0.4030	0.4389	0.3950	0.3961
ASSIST17	AUC	0.6932	0.7085	0.6425	0.7877	0.8062
	ACC	0.6940	0.7042	0.6692	0.7240	0.7454
	RMSE	0.4492	0.4538	0.4641	0.4156	0.4110

Table II : Performance analysis for all KT methods

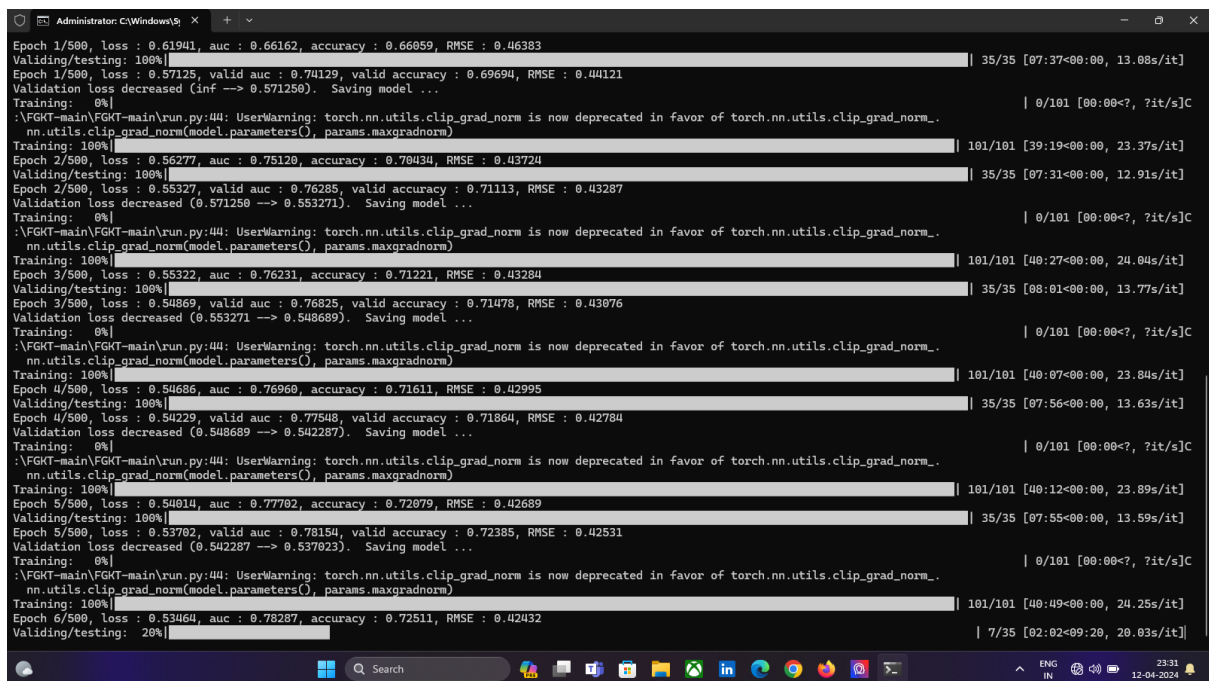
6.3.3 Snapshots



```
Administrator: C:\Windows\S...
Microsoft Windows [Version 10.0.22631.3296]
(c) Microsoft Corporation. All rights reserved.

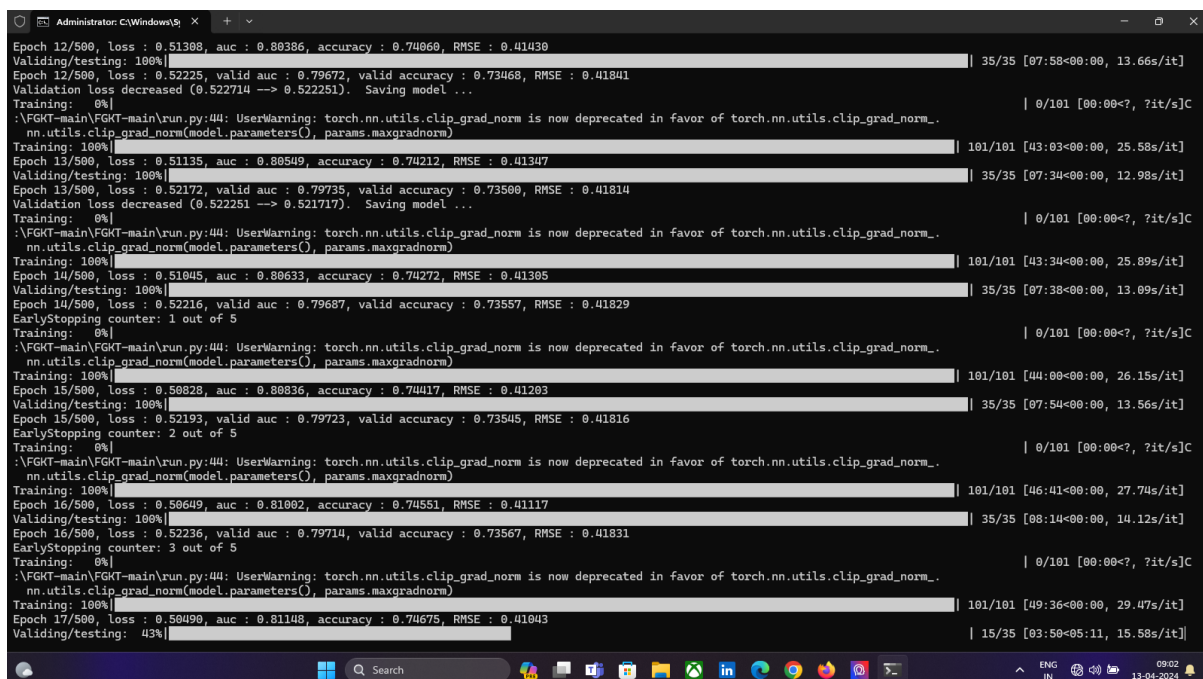
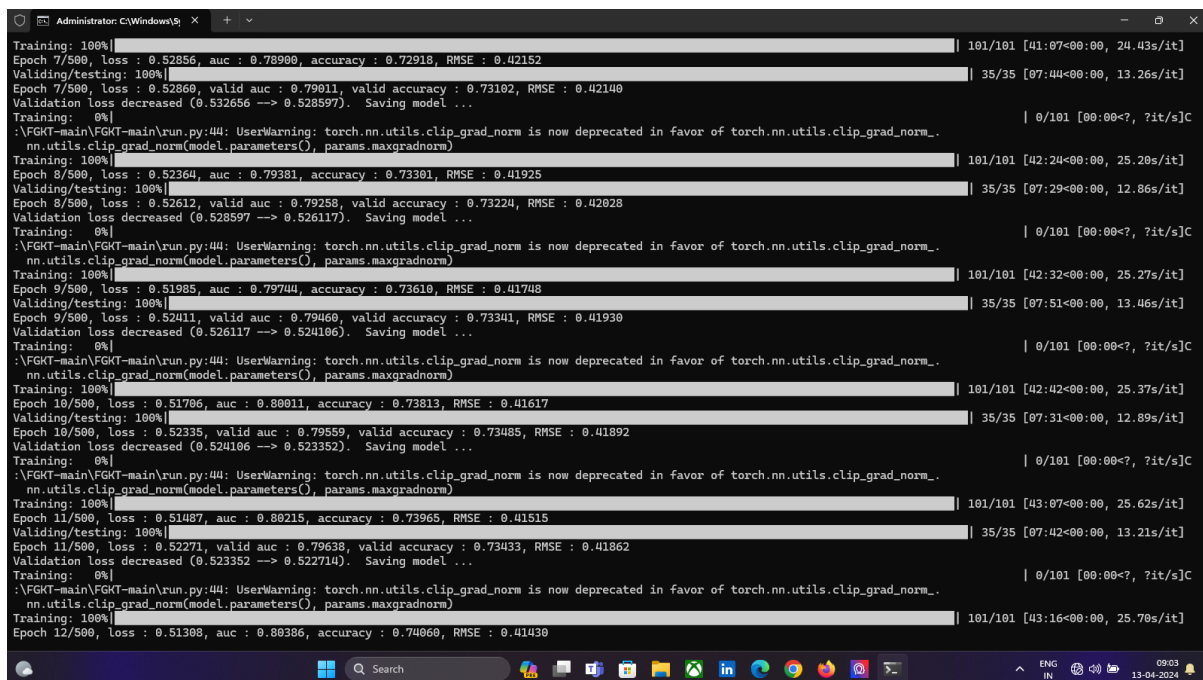
C:\FGKT-main\FGKT-main>python main.py --dataset assist17
loading train data...
train--exercise done: (3239, 200)
train--knowledge concept done: (3239, 200)
train--ans done: (3239, 200)
loading valid data...
valid--exercise done: (1130, 200)
valid--knowledge concept done: (1130, 200)
valid--ans done: (1130, 200)
loading test data...
test--exercise done: (1189, 200)
test--knowledge concept done: (1189, 200)
test--ans done: (1189, 200)
C:\FGKT-main\FGKT-main\model.py:158: UserWarning: nn.init.kaiming_normal is now deprecated in favor of nn.init.kaiming_normal_.
  nn.init.kaiming_normal(self.kc_embed.weight)
C:\FGKT-main\FGKT-main\model.py:159: UserWarning: nn.init.kaiming_normal is now deprecated in favor of nn.init.kaiming_normal_.
  nn.init.kaiming_normal(self.exercise_embed.weight)
C:\FGKT-main\FGKT-main\model.py:160: UserWarning: nn.init.kaiming_normal is now deprecated in favor of nn.init.kaiming_normal_.
  nn.init.kaiming_normal(self.linear_q.weight)
C:\FGKT-main\FGKT-main\model.py:161: UserWarning: nn.init.kaiming_normal is now deprecated in favor of nn.init.kaiming_normal_.
  nn.init.kaiming_normal(self.linear_k.weight)
C:\FGKT-main\FGKT-main\model.py:162: UserWarning: nn.init.kaiming_normal is now deprecated in favor of nn.init.kaiming_normal_.
  nn.init.kaiming_normal(self.linear_v.weight)
C:\FGKT-main\FGKT-main\model.py:163: UserWarning: nn.init.constant is now deprecated in favor of nn.init.constant_.
  nn.init.constant(self.linear_q.bias, 0)
C:\FGKT-main\FGKT-main\model.py:164: UserWarning: nn.init.constant is now deprecated in favor of nn.init.constant_.
  nn.init.constant(self.linear_k.bias, 0)
C:\FGKT-main\FGKT-main\model.py:165: UserWarning: nn.init.constant is now deprecated in favor of nn.init.constant_.
  nn.init.constant(self.linear_v.bias, 0)
C:\FGKT-main\FGKT-main\model.py:38: UserWarning: nn.init.kaiming_normal is now deprecated in favor of nn.init.kaiming_normal_.
  nn.init.kaiming_normal(self.x_modify.weight)
C:\FGKT-main\FGKT-main\model.py:39: UserWarning: nn.init.kaiming_normal is now deprecated in favor of nn.init.kaiming_normal_.
  nn.init.kaiming_normal(self.y_modify.weight)
C:\FGKT-main\FGKT-main\model.py:40: UserWarning: nn.init.kaiming_normal is now deprecated in favor of nn.init.kaiming_normal_.
  nn.init.kaiming_normal(self.fc1.weight)
C:\FGKT-main\FGKT-main\model.py:41: UserWarning: nn.init.kaiming_normal is now deprecated in favor of nn.init.kaiming_normal_.
  nn.init.kaiming_normal(self.fc2.weight)
C:\FGKT-main\FGKT-main\model.py:42: UserWarning: nn.init.constant is now deprecated in favor of nn.init.constant_.
  nn.init.constant(self.x_modify.bias, 0)
```

Fig 6.1 Initialize to implement a Assis17 dataset



```
Administrator: C:\Windows\S...
Epoch 1/500, loss : 0.61941, auc : 0.66162, accuracy : 0.66859, RMSE : 0.46383
Validating/testing: 100% | 35/35 [07:37<00:00, 13.08s/it]
Epoch 1/500, loss : 0.57125, valid auc : 0.74129, valid accuracy : 0.69694, RMSE : 0.44121
Validation loss decreased (inf -> 0.571250). Saving model ...
Training: 0% | 0/101 [00:00<?, ?it/s]C
:\FGKT-main\FGKT-main\run.py:44: UserWarning: torch.nn.utils.clip_grad_norm is now deprecated in favor of torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(), params.maxgradnorm)
Training: 100% | 101/101 [39:19<00:00, 23.37s/it]
Epoch 2/500, loss : 0.56277, auc : 0.75120, accuracy : 0.70434, RMSE : 0.43724
Validating/testing: 100% | 35/35 [07:31<00:00, 12.91s/it]
Epoch 2/500, loss : 0.55327, valid auc : 0.76285, valid accuracy : 0.71113, RMSE : 0.43287
Validation loss decreased (0.571250 -> 0.553271). Saving model ...
Training: 0% | 0/101 [00:00<?, ?it/s]C
:\FGKT-main\FGKT-main\run.py:44: UserWarning: torch.nn.utils.clip_grad_norm is now deprecated in favor of torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(), params.maxgradnorm)
Training: 100% | 101/101 [40:27<00:00, 24.04s/it]
Epoch 3/500, loss : 0.55322, auc : 0.76231, accuracy : 0.71221, RMSE : 0.43284
Validating/testing: 100% | 35/35 [08:01<00:00, 13.77s/it]
Epoch 3/500, loss : 0.54869, valid auc : 0.76825, valid accuracy : 0.71478, RMSE : 0.43076
Validation loss decreased (0.553271 -> 0.548689). Saving model ...
Training: 0% | 0/101 [00:00<?, ?it/s]C
:\FGKT-main\FGKT-main\run.py:44: UserWarning: torch.nn.utils.clip_grad_norm is now deprecated in favor of torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(), params.maxgradnorm)
Training: 100% | 101/101 [40:07<00:00, 23.84s/it]
Epoch 4/500, loss : 0.54686, auc : 0.76960, accuracy : 0.71611, RMSE : 0.42995
Validating/testing: 100% | 35/35 [07:56<00:00, 13.63s/it]
Epoch 4/500, loss : 0.54229, valid auc : 0.77548, valid accuracy : 0.71864, RMSE : 0.42784
Validation loss decreased (0.548689 -> 0.542287). Saving model ...
Training: 0% | 0/101 [00:00<?, ?it/s]C
:\FGKT-main\FGKT-main\run.py:44: UserWarning: torch.nn.utils.clip_grad_norm is now deprecated in favor of torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(), params.maxgradnorm)
Training: 100% | 101/101 [40:12<00:00, 23.89s/it]
Epoch 5/500, loss : 0.54014, auc : 0.77702, accuracy : 0.72079, RMSE : 0.42689
Validating/testing: 100% | 35/35 [07:55<00:00, 13.59s/it]
Epoch 5/500, loss : 0.53702, valid auc : 0.78154, valid accuracy : 0.72385, RMSE : 0.42531
Validation loss decreased (0.542287 -> 0.537023). Saving model ...
Training: 0% | 0/101 [00:00<?, ?it/s]C
:\FGKT-main\FGKT-main\run.py:44: UserWarning: torch.nn.utils.clip_grad_norm is now deprecated in favor of torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(), params.maxgradnorm)
Training: 100% | 101/101 [40:49<00:00, 24.25s/it]
Epoch 6/500, loss : 0.53464, auc : 0.78287, accuracy : 0.72511, RMSE : 0.42432
Validating/testing: 20% | 7/35 [02:02<09:20, 20.03s/it]
```

Fig 6.2 Starting epoch from 1 to 6



```
Administrator: C:\Windows\S...
Training: 100% | 101/101 [43:34<00:00, 25.89s/it]
Epoch 14/500, loss : 0.51045, auc : 0.80633, accuracy : 0.74272, RMSE : 0.41305
Validating/testing: 100% | 35/35 [07:38<00:00, 13.09s/it]
Epoch 14/500, loss : 0.52216, valid auc : 0.79687, valid accuracy : 0.73557, RMSE : 0.41829
EarlyStopping counter: 1 out of 5
Training: 0% | 0/101 [00:00<?, ?it/s]C
.\FGKT-main\FGKT-main\run.py:44: UserWarning: torch.nn.utils.clip_grad_norm is now deprecated in favor of torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(), params.maxgradnorm)
Training: 100% | 101/101 [44:09<00:00, 26.15s/it]
Epoch 15/500, loss : 0.50828, auc : 0.80836, accuracy : 0.74417, RMSE : 0.41203
Validating/testing: 100% | 35/35 [07:54<00:00, 13.56s/it]
Epoch 15/500, loss : 0.52193, valid auc : 0.79723, valid accuracy : 0.73545, RMSE : 0.41816
EarlyStopping counter: 2 out of 5
Training: 0% | 0/101 [00:00<?, ?it/s]C
.\FGKT-main\FGKT-main\run.py:44: UserWarning: torch.nn.utils.clip_grad_norm is now deprecated in favor of torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(), params.maxgradnorm)
Training: 100% | 101/101 [46:41<00:00, 27.74s/it]
Epoch 16/500, loss : 0.50649, auc : 0.81002, accuracy : 0.74551, RMSE : 0.41117
Validating/testing: 100% | 35/35 [08:14<00:00, 14.12s/it]
Epoch 16/500, loss : 0.52236, valid auc : 0.79714, valid accuracy : 0.73567, RMSE : 0.41831
EarlyStopping counter: 3 out of 5
Training: 0% | 0/101 [00:00<?, ?it/s]C
.\FGKT-main\FGKT-main\run.py:44: UserWarning: torch.nn.utils.clip_grad_norm is now deprecated in favor of torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(), params.maxgradnorm)
Training: 100% | 101/101 [49:36<00:00, 29.47s/it]
Epoch 17/500, loss : 0.50490, auc : 0.81148, accuracy : 0.74675, RMSE : 0.41043
Validating/testing: 100% | 35/35 [09:07<00:00, 15.65s/it]
Epoch 17/500, loss : 0.52239, valid auc : 0.79679, valid accuracy : 0.73538, RMSE : 0.41840
EarlyStopping counter: 4 out of 5
Training: 0% | 0/101 [00:00<?, ?it/s]C
.\FGKT-main\FGKT-main\run.py:44: UserWarning: torch.nn.utils.clip_grad_norm is now deprecated in favor of torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(), params.maxgradnorm)
Training: 100% | 101/101 [51:04<00:00, 30.34s/it]
Epoch 18/500, loss : 0.50324, auc : 0.81296, accuracy : 0.74814, RMSE : 0.40963
Validating/testing: 100% | 35/35 [09:09<00:00, 15.44s/it]
Epoch 18/500, loss : 0.52255, valid auc : 0.79702, valid accuracy : 0.73532, RMSE : 0.41845
EarlyStopping counter: 5 out of 5
Early stopping
Validating/testing: 100% | 37/37 [08:30<00:00, 13.79s/it]
test_auc: 0.8062 test_accuracy: 0.7454 , RMSE : 0.411030 test_loss: 0.5066
C:\FGKT-main\FGKT-main>python main.py --dataset assist2017
```

Fig 6. 5 Final result for ASSISTments2017 dataset

CHAPTER 7

TESTING

7.1 TESTING OBJECTIVES

Testing is performed to identify errors. A good test case is one that has a high probability of finding an undiscovered error. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases: assemblies and/or a finished product. It is the process of exercising software.

- Static analysis is used to investigate the structural properties of the source.
- Dynamic testing is used to investigate the behavior of the source code.

Static testing is a software testing method that examines a program -- along with any associated documents -- but does not require the program to be executed. Dynamic testing, the other main category of software testing, requires testers to interact with the program while it runs.

7.2 TYPES OF TESTING

7.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the Internal program logic is functioning properly, and that program input produces valid output. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the 38 completions of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined input and expected results.

7.2.2 Integration Testing

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the setoff modules which make up the project. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have

passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs.

The new module and its interring communications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

7.2.3 Functional Testing

Functional test cases involved exercising the code with nominal input Scores for which the expected results are known, as well as boundary Scores, such as logically related inputs, files of identical elements, and empty files. Three type of test in functional test

- Performance Test
- Stress Test
- Structure Test
- System Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items.

- Valid Input: identified classes of valid input must be accepted.
- Invalid Input: identified classes of invalid input must be rejected.
- Functions: identified functions must be exercised.
- Output: identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

7.2.3.1 Performance Testing

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit. The Performance test ensures that the output is produced within the

time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results, the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

7.2.3.2 Stress Testing

Stress test is the test designed to intentionally break the unit. A great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

7.2.3.3 Structure Testing

Structure tests are concerned with exercising the internal logic of a program and traversing particular execution path. The way in which White-Box test strategy was employed to ensure that the test cases could guarantee that all independent paths within a module have been exercise at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.

Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs.

7.2.3.4 System Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability if finding an as-yet-undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. System testing is the stage of implementation which is and steps for run program, string, system and is important in adopting a successful new system.

This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise, the program or the project is not said to be complete. Software testing is the critical element of software quality coding. Testing is the process of executing the program with the intent of finding the error. A successful test is one that undiscovered error. A successful test is one that uncovers a yet undiscovered error. Any engineering product can be tested in one of the two ways.

7.3 TESTING TECHNIQUES

7.3.1 Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Testing is the process of executing the program with the intent of finding the errors. A good test case design is one that has a high probability of finding the errors. A successful test is one that uncovers

an undiscovered error. Any engineering product can be tested in one of the following two ways:

7.3.2 White Box Testing

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

- Flow graph notation
- Cyclometric complexity

7.3.3 Black Box Testing

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software. It focuses on ensuring that the system behaves correctly from a user's perspective and meets the specified requirements.

- Graph based testing methods
- Equivalence partitioning
- Boundary Score analysis
- Comparison testing

7.3.3.1 Boundary Score Analysis

Boundary Score analysis is a software testing technique in which tests are designed to include representatives of boundary Scores in a range. The idea

comes from the boundary. Given that we have a set of test vectors to test the system, a topology can be defined on that set.

7.3.3.2 Equivalence partitioning

Equivalence partitioning or equivalence class partitioning (ECP) is a software testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived. In principle, test cases are designed to cover each partition at least once.

7.3.3.3 Comparison Testing

Comparison testing involves comparing the contents of databases files, folders, etc. A comparison testing can be carried out in two ways. Direct comparison testing: It is a testing of particular parts of each site against each other. Usually, multiple sites are compared side by side. Objective comparison testing: Execute the same test for each site separately. Usually, one site at a time is tested.

7.4 Test Strategy and Approach

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For the reason a template for software testing a set step into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer base system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.

7.4.1 Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together” interfacing. There may be the chances of data lost across on another’s sub functions, when combined may produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; goals data structure can present problems.

7.4.2 User Acceptance Testing

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developed.

CHAPTER 8

RESULT AND DISCUSSION

8.1 Performance analysis

To evaluate the performance of the experiments on comparing the AUC, ACC and RMSE for all knowledge tracing model. The synthetic-5 dataset does not exercise relationship but our experiment loaded the assisment 09 and assisment 17 it had relationship on exercise. In our experiment we loaded both actual differences among exercise and relevant prior knowledge for the current exercise.

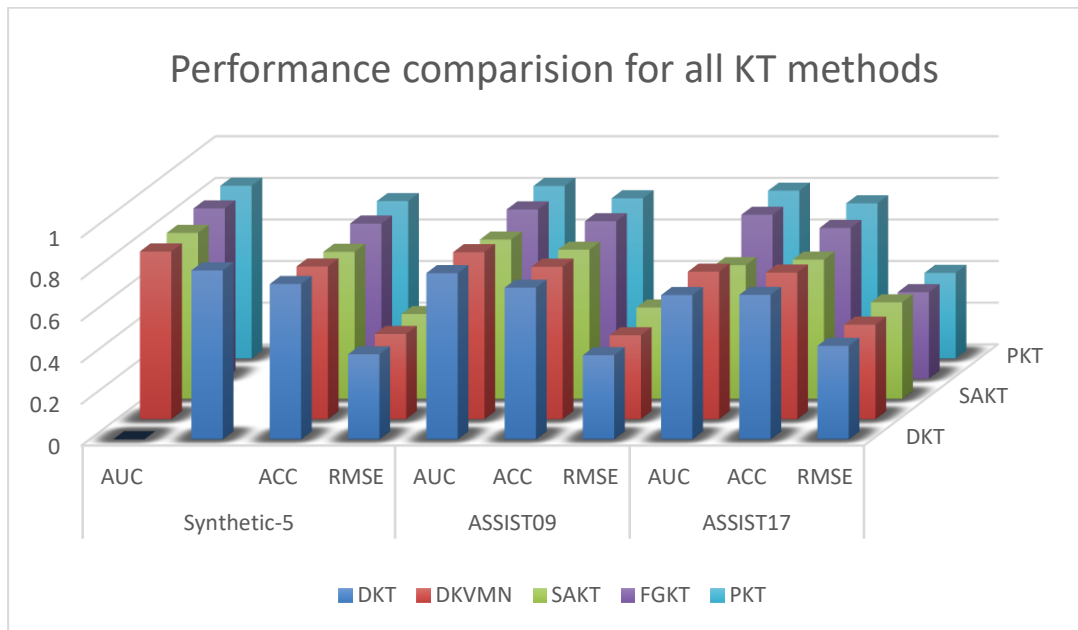


Fig. 8. 1 Performance comparison for all KT methods

In this approach ,we dicuss the dataset of ASSISSTment 17 on this experiment. In ASSISSTment 17 the Area Under Cover(AUC), Accuracy(ACC) is increased compared to previous methods .The Root Mean Square Error Score is decreased compared to previous method.

In ASSISSTment17 dataset both the integration function like concatenation and multiplication are used. The dataset of ASSISSTment17, its run the module by epoch (max 500) and the max length of exercise N=200. In our experiment 18 epoch will be implemented. In epoch 14 the test loss is increased. So the early stopping counter (upto 5) is activated. The final result of ASSISSTment 17 display the parameter are test_auc:0.8062, test_acc:0.7454, test_loss:0.5066 and RMSE:0.411030. Overall the parameter of test_auc and test_acc are increased compared to FGKT-CM.

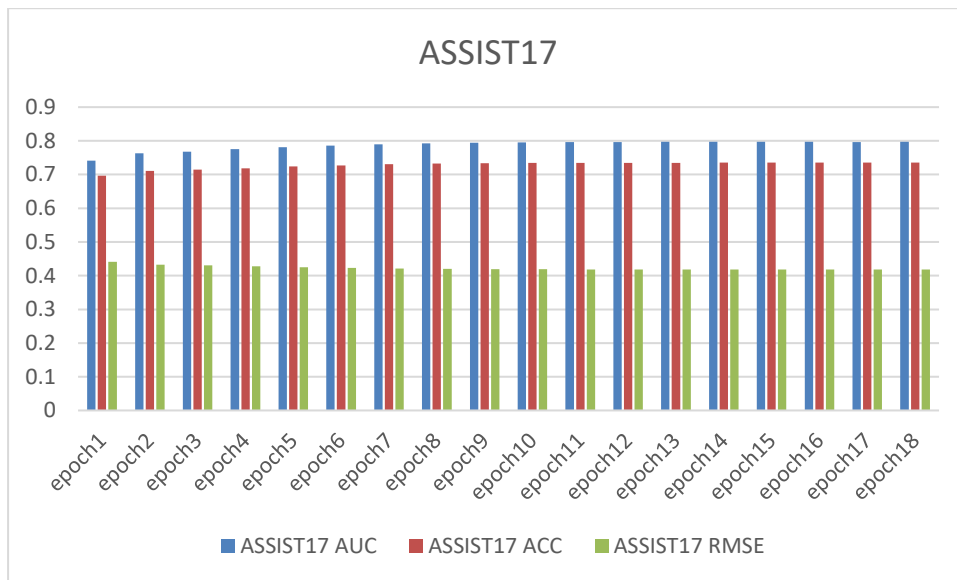


Fig. 8.1.1 Performance analysis for Assist17

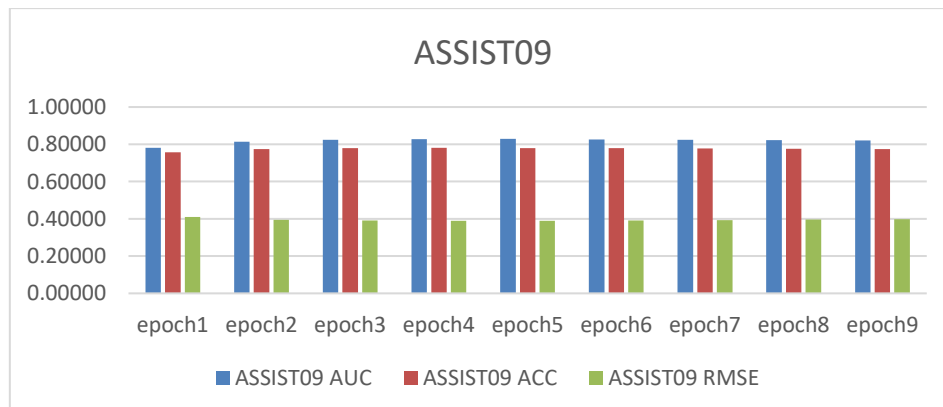


Fig. 8.1.2 Performance analysis for Assist09

8.2 Validation loss

Once the epoch will be executed, the validation loss also be generated. If the validation loss lesser than previous validation loss, the next epoch will be executed. Otherwise, the Adam Optimizer function (early stopping counter) will be activated and to display the final output parameter.

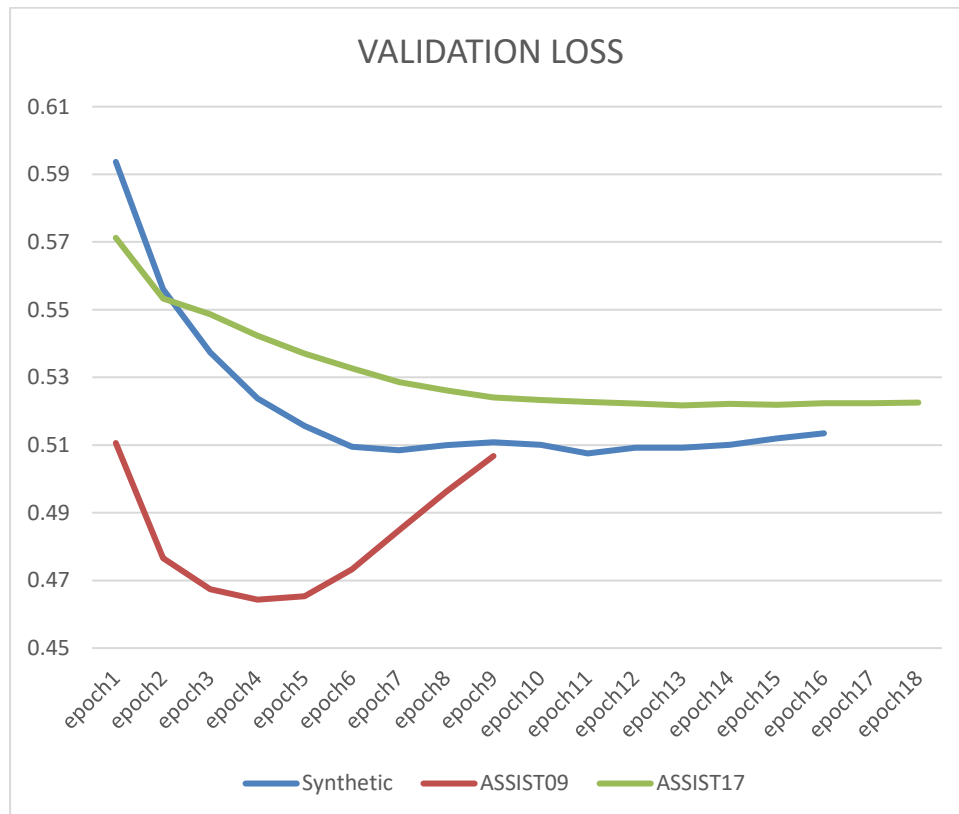


Fig. 8.2 Validation graph for three datasets

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

Overall, our focus lies on Knowledge Tracing (KT) centered around the exercise embedding, which accounts for the actual differences among exercises and leverages relevant prior knowledge through deep learning-based models. This work utilizes Long Short-Term Memory (LSTM) networks to dynamically update the learner's knowledge state and predict future exercises accordingly. Additionally, this approach conducts ongoing assessment exercises to identify and address learner knowledge gaps, thereby tracing the current knowledge state and mitigating the effects of forgetting.

9.2 FUTURE WORK

In future works, we aim to enhance exercise embedding by incorporating actual differences in exercise difficulty. This involves designing a fully connected, responsive exercise embedding system to accurately capture variations in difficulty. Additionally, we plan to introduce a mechanism to calculate learners' response time for future exercises, providing insights into their knowledge ability and enabling tailored learning experiences based on their proficiency levels.

REFERENCES

1. Abdelrahman. G and Wang .Q, “Knowledge tracing with sequential key value memory networks,” in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Paris, France, 2019, pp. 175–184.
2. Almatrafi. O and Johri .A, “Systematic review of discussion forums in massive open online courses (MOOCs),” *IEEE Trans. Learn. Technol.*, vol. 12, no. 3, pp. 413–428, Jul.–Sep. 2019.
3. Cen .H, Koedinger .K.R, and Junker .B, “Learning factors analysis - a general method for cognitive model evaluation and improvement,” in *Proc. 8th Int. Conf. Intell. Tutoring Syst.*, Jhongli, Taiwan, 2006, pp. 164–175.
4. Corbett A.T, Anderson.J.R, “Knowledge tracing :Modeling the acquisition of procedural knowledge,” *User Model. User-Adapted Interaction*, vol. 4, no. 4, pp. 253–278, 1994.
5. Crowston. K et al., “Knowledge tracing to model learning in online citizen scienceprojects,”*IEEETrans.Learn.Technol.*,vol.13,no.1,pp.123–134, Jan.–Mar. 2020.
6. De Baker R. S. J, Corbett A. T, and Aleven V., “More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing,” in *Proc. 9th Int. Conf. Intell. Tutoring Syst.*,2008, pp. 406–415.
7. Ding .X and Larson. E.C, “Why deep knowledge tracing has less depth than anticipated,” in *Proc. 12th Int. Conf. Educ. Data Mining*, Montréal, Canada, 2019, pp. 282–287.

8. Ghahramani. Z, “An introduction to hidden Markov models and Bayesian networks,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 15, no. 1, pp. 9–42, 2001.
9. Kaser. T, Klingler. A.G, Schwing ,and Gross M.H “Dynamic Bayesian networks for student modeling,” *IEEE Trans. Learn. Technol.*, vol. 10, no. 4, pp. 450–462, Oct.–Dec. 2017.
10. Miller. A.H, Fisch. A, Dodge. J, Karimi. A, Bordes. A, and Weston. J, “Key-value memory networks for directly reading documents,” in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Austin, TX, USA, 2016, pp. 1400–1409.
11. Pandey. S and Karypis. G, “A self attentive model for knowledge tracing,” in *Proc. 12th Int. Conf. Educ. Data Mining*, Montréal, Canada, 2019, pp. 3
12. Papamitsiou. Z.K, Pappas. I.O, Sharma. K and M. N. Giannakos, “Utilizingmultimodal datathroughfsQCAtoexplainengagementinadaptive learning,” *IEEE Trans. Learn. Technol.*, vol. 13, no. 4, pp. 689–703, Oct.–Dec. 2020.
13. Pardos. Z.A and Heffernan. N.T, “KT-IDEM: Introducing item difficulty to the knowledge tracing model,” in *Proc. 19th Int. Conf. User Model., Adaption Personalization*, Girona, Spain, 2011, pp. 243–254.
14. Pavlik. P.I, Cen. H, and Koedinger. K.R, “Performance factors analysis—a new alternative to knowledge tracing,” in *Proc. 14th Int. Conf. Artif. Intell. Educ.*, Brighton, U.K., 2009, pp. 531–538.
15. Piech. C, “Deep knowledge tracing,” in *Proc. Adv. Neural Inf. Process. Syst. 28: Annu. Conf. Neural Inf. Process. Syst.*2015, Montreal, QC, Canada, 2015, pp. 505–513.

16. Vidergor. H.E and Ben-Amram. P, “Khan academy effectiveness: The case of math secondary students’ perceptions,” *Comput. Educ.*, vol. 157, 2020,
17. Wang. T, Ma. F, and Gao. J, “Deep hierarchical knowledge tracing,” in *Proc. 12th Int. Conf. Educ. Data Mining*, Montréal, Canada, 2019, pp. 671–674.
18. Wu. J, Li. M, Tang. Y, and Liang. Q, “Exercise recommendation based on knowledge concept prediction,” *Knowl. Based Syst.*, vol. 210, 2020,
19. Yeung. Q and Yeung. D, “Addressing two problems in deep knowledge tracing via prediction-consistent regularization,” in *Proc. 5th Annu. ACM Conf. Learn. Scale*, London, U.K., 2018, pp. 5:1–5:10.
20. Yeung. C, “Deep-IRT: Make deep learning based knowledge tracing explainable using item response theory,” in *Proc. 12th Int. Conf. Educ. Data Mining*, 2019, pp. 683–686.
21. Zhang. J, Shi. X, King. I, and Yeung. D, “Dynamic key-value memory networks for knowledge tracing,” in *Proc. 26th Int. Conf. World Wide Web*, Perth, Australia, 2017, pp. 765–774.

.

APPENDIX

```
import torch

import argparse

from model import Model

from run import train, test

import torch.optim as optim

from earlystop import EarlyStopping

from dataloader import getDataLoader


def main():

    parser = argparse.ArgumentParser()

    parser.add_argument('--gpu', type=int, default=-1, help='the gpu will be used,
e.g "0,1,2,3"')

    parser.add_argument('--patience', type=int, default=5)

    parser.add_argument('--max_iter', type=int, default=500, help='number of
iterations')

    parser.add_argument('--lr', type=float, default= 0.001, help='learning rate')

    parser.add_argument('--dropout', type=float, default=0.3)

    parser.add_argument('--maxgradnorm', type=float, default=50.0,
help='maximum gradient norm')

    parser.add_argument('--hidden_layer', type=int, default=256)
```

```

parser.add_argument('--num_layer', type=int, default=2)

parser.add_argument('--num_heads', type=int, default=1, help='Number of
head attentions.')

parser.add_argument('--mode', type=int, default=3, help='mode of Integration
Function. '

                        '1:ca'

                        '2:mul'

                        '3:ca mul'

                        '4:rasch')

parser.add_argument('--exercise_embed_dim', type=int, default=128,
help='question embedding dimensions')

parser.add_argument('--batch_size', type=int, default=32, help='the batch
size')

parser.add_argument('--fold', type=str, default='1', help='number of fold')

parser.add_argument('--dataset', type=str, default='assist2017', help='dataset
name')


params = parser.parse_args()

dataset = params.dataset


if dataset == 'assist2009':

    params.hasConcept = 1

```

```
params.max_step = 200

params.n_knowledge_concept = 110

params.n_exercise = 16891

params.data_dir = './data/assist2009'

params.data_name = 'assist2009'


if dataset == 'assist2017':

    params.hasConcept = 1

    params.max_step = 200

    params.n_knowledge_concept = 102

    params.n_exercise = 3162

    params.data_dir = './data/assist2017'

    params.data_name = 'assist2017'


if dataset == 'statics':

    params.hasConcept = 1

    params.max_step = 200

    params.n_knowledge_concept = 98

    params.n_exercise = 1223

    params.data_dir = './data/STATICS'
```



```
params.data_name = 'STATICS'
```

```
if dataset == 'synthetic':
```

```
    params.hasConcept = 0
```

```
    params.max_step = 50
```

```
    params.n_knowledge_concept = 0
```

```
    params.n_exercise = 50
```

```
    params.data_dir = './data/synthetic'
```

```
    params.data_name = 'synthetic'
```

```
if dataset == 'assist2015':
```

```
    params.hasConcept = 0
```

```
    params.max_step = 200
```

```
    params.n_knowledge_concept = 0
```

```
    params.n_exercise = 101
```

```
    params.data_dir = './data/assist2015'
```

```
    params.data_name = 'assist2015'
```

```
params.input = params.exercise_embed_dim * 2
```

```
params.output = params.n_exercise
```

```
params.embed_d = params.exercise_embed_dim
```

```
train_data_path = params.data_dir + "/" + params.data_name + "_train"+  
params.fold + ".csv"
```

```
valid_data_path = params.data_dir + "/" + params.data_name + "_valid"+  
params.fold + ".csv"
```

```
test_data_path = params.data_dir + "/" + params.data_name + "_test"+  
params.fold + ".csv"
```

```
train_kc_data, train_respond_data, train_exercise_data, \
```

```
valid_kc_data, valid_respose_data, valid_exercise_data, \
```

```
test_kc_data, test_respose_data, test_exercise_data \
```

```
= getDataLoader(train_data_path, valid_data_path, test_data_path, params)
```

```
train_exercise_respond_data = train_respond_data * params.n_exercise +  
train_exercise_data
```

```
valid_exercise_respose_data = valid_respose_data * params.n_exercise +  
valid_exercise_data
```

```
test_exercise_respose_data = test_respose_data * params.n_exercise +  
test_exercise_data
```

```
model = Model(params.input, params.hidden_layer, params.num_layer,
params.output, params.embed_d, params.num_heads, params.dropout,
params.mode, params)
```

```
optimizer = optim.Adam(params=model.parameters(), lr=params.lr,
betas=(0.9, 0.9), weight_decay=1e-5)
```

```
if params.gpu >= 0:
```

```
    print('device: ' + str(params.gpu))
```

```
    torch.cuda.set_device(params.gpu)
```

```
    model.cuda()
```

```
early_stopping = EarlyStopping(params.patience, verbose=True)
```

```
for idx in range(params.max_iter):
```

```
    train_loss, train_accuracy, train_auc, train_RMSE = train(model, params,
optimizer, train_kc_data, train_exercise_data, train_respond_data,
train_exercise_respond_data, params.hasConcept)
```

```
    print('Epoch %d/%d, loss : %3.5f, auc : %3.5f, accuracy : %3.5f, RMSE :
%3.5f' % (idx + 1, params.max_iter, train_loss, train_auc, train_accuracy,
train_RMSE))
```

```
    with torch.no_grad():
```

```
valid_loss, valid_accuracy, valid_auc, valid_RMSE = test(model,
params, valid_kc_data, valid_exercise_data, valid_response_data,
valid_exercise_response_data, params.hasConcept)
```

```
print('Epoch %d/%d, loss : %3.5f, valid auc : %3.5f, valid accuracy :
%3.5f, RMSE : %3.5f' % (idx + 1, params.max_iter, valid_loss, valid_auc,
valid_accuracy, valid_RMSE))
```

```
early_stopping(valid_loss, model)
```

```
if early_stopping.early_stop:
```

```
    print("Early stopping")
```

```
    break
```

```
model.load_state_dict(torch.load('checkpoint.pt'))
```

```
with torch.no_grad():
```

```
    test_loss, test_accuracy, test_auc, test_RMSE = test(model, params,
test_kc_data, test_exercise_data, test_response_data, test_exercise_response_data,
params.hasConcept)
```

```
    print("test_auc: %.4f\t test_accuracy: %.4f\t, RMSE : %4f\t test_loss:
%.4f" % (test_auc, test_accuracy, test_RMSE, test_loss))
```

```
if __name__ == "__main__":
```

```
    main()
```

