**1. Introduction**

The project focuses on the development of an Intelligent Tutoring System (ITS) for solving quadratic equations, using Artificial Intelligence (AI) techniques to enhance personalized learning and interactive education in mathematics. Quadratic equations are useful in algebra, physics, engineering, and numerous other fields (Geeksforgeeks, 2024), yet many students struggle with understanding and solving them effectively. This ITS addresses this problem by providing an intuitive and user-friendly platform to teach the concepts and guide learners through step-by-step solutions.

The system uses AI algorithms, including knowledge representation and rule-based reasoning, to provide explanations and solutions. Ontology-based learning will be integrated to dynamically adapts its responses to the learner's input, ensuring that users receive personalized guidance based on their proficiency level. For instance, the system can offer additional explanations for concepts like discriminants or complex roots when required.

The rationale for selecting quadratic equations as the domain lies in their ubiquity in academic curricula and their challenging nature for many learners and the choice of Java for the user interface ensures a modern, interactive, and visually appealing environment, while Jena Framework will help in the ontology integration.

**2. Project Plan**

The development of the Intelligent Tutoring System (ITS) for solving quadratic equations follows a structured project plan to ensure timely delivery, clear role assignments, and alignment with the project's objectives.

**2.1 Milestones, Objectives and timeline**

Table 1 This table provides a concise overview of the project's workflow, ensuring clarity and focus at each stage.

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective** | **Milestone** | **Timeline** | **Description** |
| Understand the Problem Domain | Research quadratic equations and their challenges. | Week 1 | Study the mathematical concepts, identify key topics, and define learning goals. |
| Design the ITS Framework | Develop architecture and create ontology in Protégé. | Weeks 2–3 | Build the OWL ontology to represent concepts, relationships, and solutions. |
| Develop Core Components | Build ontology module, mathematical solver, and UI. | Weeks 4–6 | Implement ontology loading, solving algorithms, and a user-friendly interface. |
| Testing and Iteration | Conduct functional and usability tests. | Weeks 7–8 | Test all components, refine based on feedback, and ensure system reliability. |
| Documentation and Submission | Prepare final report and submit the project. | Week 9 | Write the report detailing design, implementation, evaluation, and lessons. |

**2.2 Roles and Responsibilities**

* Knowledge Engineer: Responsible for creating the ontology in Protégé, defining concepts, and ensuring compatibility with the Jena Framework.
* AI Developer: Implements the reasoning engine, mathematical solver, and ensures integration with the ontology for dynamic responses.
* Software developer: Designs and implements the Java interface.

**3. Literature Review**

In 1970, Jaime R. Carbonell unveiled the SCHOLAR tutor system, the first intelligent tutoring system (ITS) (Woolf, 2010). Intelligent Tutoring Systems (ITS) are computerized learning environments that incorporate computational models from the cognitive sciences, learning sciences, computational linguistics, artificial intelligence, mathematics, and other fields (Graesser et al, 2012). Jaime program was created to assess students' understanding of South American geography. In fact, the SCHOLAR initiated a two-way dialogue with the students by assessing their knowledge in the context of geography using the semantic network of concepts and facts (Carbonell, 1970). The educational systems have experienced an increase in the use of artificial intelligence (AI), particularly machine learning. ITSs are adaptive teaching systems that integrate educational methodologies with AI techniques. The key component of these systems is their capacity to personalize instructional activities and tactics in accordance with the learner's traits and requirements (Keleş, Ocak, Keleş, & Gülcü, 2009).

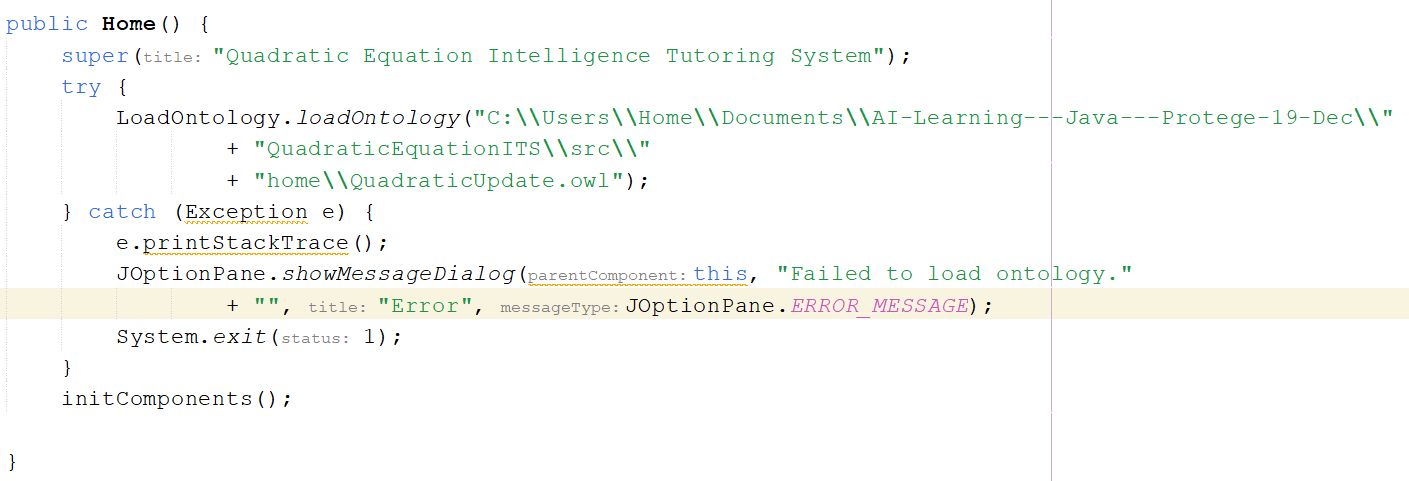
The ITSs are characterized by a classical architecture that comprises four modules, each of which is referred to by a unique designation in the field of research.

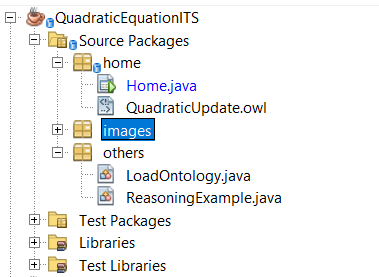
* The initial component is the expert module. This section covers the knowledge that the student desires to acquire (domain knowledge) (Ma, Adesope, Nesbit, & Liu, 2014). In this module, the student's activities in the learning process are analysed and problem-solved in a manner similar to that of human experts (Carter, 2014).
* The second module is the student diagnosis module or student model, which is constructed by factors such as the level of knowledge, activities, responses, behaviours, learning styles, student's knowledge deficiency, and other information about the learner that is gathered and updated during the learning process in the system (Brown, 2009; Ma et al., 2014).
* The module is the Instruction, tutor, or pedagogical module. It identifies the knowledge deficiency in students and concentrates on the strategies and teaching methods that can be employed to address the identified knowledge deficit in a particular field (Polson & Richardson, 2013). Adaptive feedback, navigation of the learning path, hint and recommendation generation, and the presentation of adaptive educational content comprise the core of this module (Carter, 2014).
* The final module is the user interface, which is a communication component of ITS, and it is responsible for regulating the interaction between the user and the system (Burns & Capps, 1989).

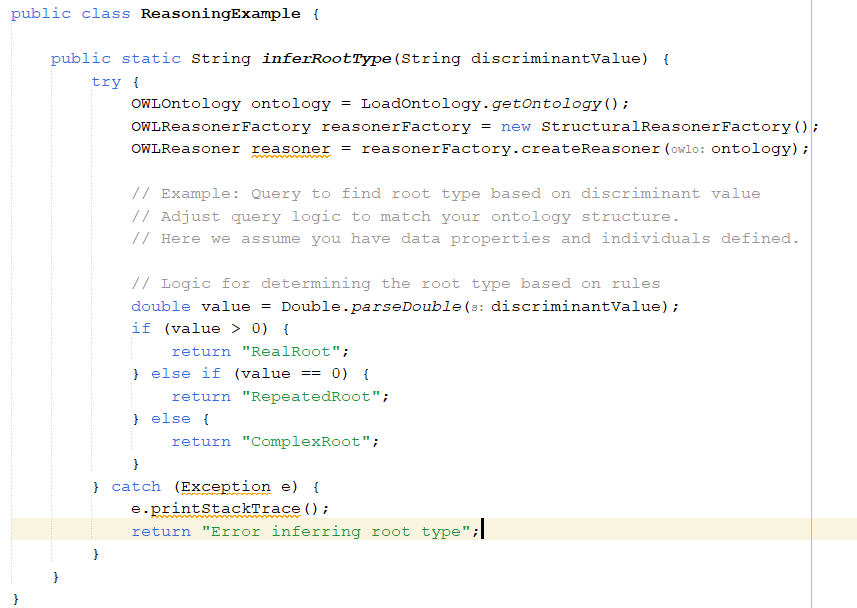
# 3.1 Existing Intelligence Tutoring Systems

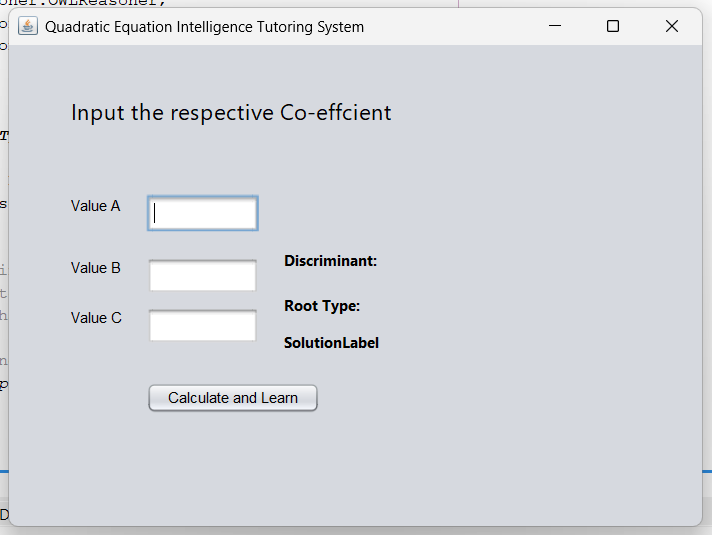
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **System Name and Author** | **Domain** | **Structure** | **Mechanisms** | **Knowledge Representation** | **User Interaction** | **Critique** |
| **AutoTutor** Graesser et al., (2011) | General Education | Modular: Tutor Module, Student Model, Discourse Management | Uses Natural Language Processing (NLP) to facilitate dialogues; employs a mixed-initiative tutoring strategy | Semantic networks and concept maps to represent subject matter | Conversational interface; encourages dialogue and student reasoning | Effective in natural language comprehension but limited in domains requiring technical simulation. |
| **ALEKS** Sujo-Montes et al., (2021) | Mathematics, Science | Web-based adaptive learning system | Uses Item Response Theory (IRT) for adaptive question generation | Knowledge spaces theory for representing student proficiency | Provides real-time feedback and progress reports | Lacks deep interaction or personalization beyond assessing correct/incorrect responses. |
| **Cognitive Tutor** Walker et al., (2006) | Mathematics | Rule-based AI architecture | Employs model tracing and knowledge tracing to guide problem-solving activities | Rule-based production systems for representing procedural knowledge | Interactive problem-solving with step-by-step guidance | Effective in teaching procedural skills but struggles with ill-defined or creative problem domains. |
| **ITSPOKE Litman (2004)** | Physics | Dialogue-based tutoring system built on AutoTutor architecture | Incorporates speech recognition for student input | Semantic and syntactic parsing for knowledge representation | Speech-based interaction to make the system more natural for users | Speech recognition errors can reduce system effectiveness, especially in noisy environments. |
| **ANDES Vanlehn et al., (2005)** | Physics Problem Solving | Graphical User Interface with a mix of free-form and structured problem-solving tools | Constraint-based modelling to evaluate student solutions | Logical representations and rules to model student reasoning | Step-based guidance with tool integration (e.g., drawing diagrams and equations) | Limited flexibility in handling non-standard student approaches or novel problem variations. |
| **SQL-Tutor Mitrovic (2005)** | Computer Science (SQL) | Standalone system with a database-backed structure | Constraint-based modeling to identify student misconceptions | Uses constraints to represent correct solutions and common errors | Provides feedback on query formulation with explanations for errors | Limited adaptability for advanced SQL topics or creative query formulation. |
| **DeepTutor** Mousavinasab et al (2021) | General Education | Cloud-based platform with deep learning integration | Uses deep learning models for adaptive feedback | Knowledge graphs and neural networks for representing subject knowledge | Adaptive interaction with personalized feedback | Complex but sometimes opaque feedback due to deep learning's "black-box" nature. |
| **GIFT (Generalized Intelligent Framework for Tutoring)** Sottilare et al (2017) | Multiple Domains | Modular, extensible architecture | Supports adaptive assessments, game-based learning, and intelligent feedback | Knowledge components linked to learning objectives | Supports a variety of learning scenarios, including virtual environments | Requires significant setup and customization for domain-specific applications. |
| **ASSISTments** Razzaq et al (2009) | Mathematics, STEM | Web-based learning platform integrated with classroom practices | Offers scaffolding and real-time hints | Represented using a mix of procedural and declarative knowledge | Interactive question-response with teacher and peer feedback integration | Effective for classroom integration but less flexible for independent, self-paced learners. |
| **Pyrenees** | Foreign Language Learning | Scenario-based immersive learning system | Employs interactive narratives and contextual feedback | Representations based on scenarios and language-specific syntax/semantics | Engages users through immersive storytelling and contextual interactions | Effective for language immersion but resource-intensive to create new scenarios. |

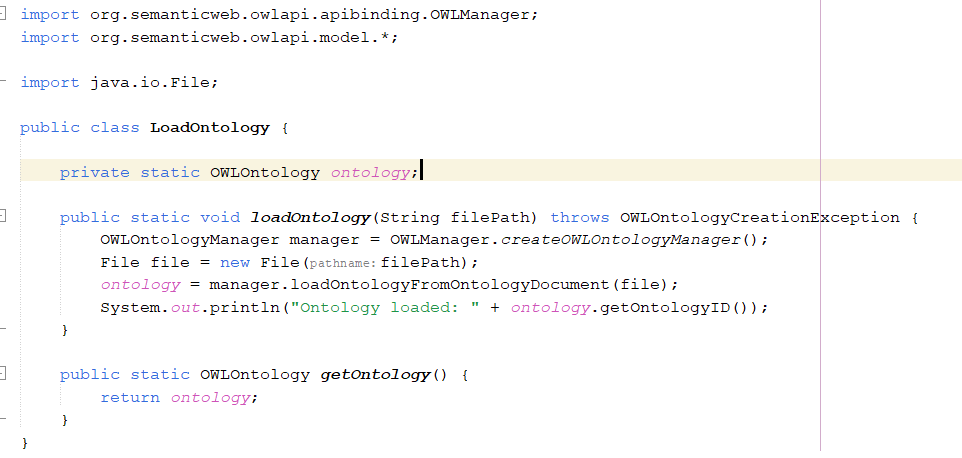
**4. Development of Your ITS**

****

****

****

****

****

**Step 1: Open Your Ontology**

1. **Launch Protégé and open your ontology.**
2. **Ensure your classes are already created:**
   * **QuadraticEquation**
   * **Coefficient**
   * **Discriminant**
   * **Root (and its subclasses: RealRoot, ComplexRoot, RepeatedRoot)**
   * **SolutionMethod (and its subclasses: Factoring, etc.)**

**Step 2: Add Data Properties (Attributes)**

**1. Create Data Properties**

1. **Go to the Data Properties tab in Protégé.**
2. **Click the Add (+) button to create a new data property.**

**2. Define Data Properties for Attributes**

**Create the following data properties:**

| **Property Name** | **Domain** | **Range** | **Description** |
| --- | --- | --- | --- |
| **aValue** | **Coefficient** | **xsd:double** | **Represents the value of coefficient a** |
| **bValue** | **Coefficient** | **xsd:double** | **Represents the value of coefficient b** |
| **cValue** | **Coefficient** | **xsd:double** | **Represents the value of coefficient c** |
| **discriminantValue** | **Discriminant** | **xsd:double** | **Represents the discriminant value** |

1. **For each data property:**
   * **Name: Enter the property name (e.g., aValue).**
   * **Domain: Assign the corresponding class (e.g., Coefficient).**
   * **Range: Set the range to xsd:double (or another numeric type, such as xsd:integer if preferred).**

**Step 3: Add Object Properties (Relationships)**

**1. Create Object Properties**

1. **Go to the Object Properties tab in Protégé.**
2. **Click the Add (+) button to create a new object property.**

**2. Define Object Properties for Relationships**

**Create the following object properties:**

| **Property Name** | **Domain** | **Range** | **Description** |
| --- | --- | --- | --- |
| **hasCoefficient** | **QuadraticEquation** | **Coefficient** | **Links a quadratic equation to its coefficients (a, b, c).** |
| **hasDiscriminant** | **QuadraticEquation** | **Discriminant** | **Links a quadratic equation to its discriminant.** |
| **determinesRootType** | **Discriminant** | **Root** | **Links a discriminant to the type of root it determines.** |
| **usesMethod** | **QuadraticEquation** | **SolutionMethod** | **Links a quadratic equation to a solution method.** |

1. **For each object property:**
   * **Name: Enter the property name (e.g., hasCoefficient).**
   * **Domain: Assign the starting class of the relationship (e.g., QuadraticEquation).**
   * **Range: Assign the target class of the relationship (e.g., Coefficient).**

**Step 4: Test Properties**

1. **Go to the Individuals tab.**
2. **Create individuals for QuadraticEquation, Coefficient, Discriminant, and others.**
   * **Assign values to aValue, bValue, and cValue for Coefficient.**
   * **Compute the discriminant manually and assign it to discriminantValue for Discriminant.**
3. **Link the individuals using object properties:**
   * **Use hasCoefficient to link a QuadraticEquation individual to a Coefficient individual.**
   * **Use hasDiscriminant to link a QuadraticEquation individual to a Discriminant individual.**
   * **Use determinesRootType to link a Discriminant individual to a Root individual (e.g., RealRoot).**
   * **Use usesMethod to link a QuadraticEquation individual to a SolutionMethod individual.**

**Step 5: Run Reasoner**

1. **Use a reasoner (e.g., HermiT) to check if relationships are correctly inferred.**
2. **Verify that individuals are linked properly and that the ontology behaves as expected.**

**Example in Protégé**

**Example Individuals and Properties**

* **Individual: quadraticEq1 (instance of QuadraticEquation)**
  + **hasCoefficient: Links to coeff1 (instance of Coefficient).**
  + **hasDiscriminant: Links to disc1 (instance of Discriminant).**
  + **usesMethod: Links to factoringMethod (instance of Factoring).**
* **Individual: coeff1 (instance of Coefficient)**
  + **aValue: 1.0**
  + **bValue: -3.0**
  + **cValue: 2.0**
* **Individual: disc1 (instance of Discriminant)**
  + **discriminantValue: 1.0**
  + **determinesRootType: Links to realRoot1 (instance of RealRoot).**

**Domain Description**

The ITS focuses on solving quadratic equations of the form ax2+bx+c=0ax^2 + bx + c = 0ax2+bx+c=0, teaching students to:

1. Identify coefficients aaa, bbb, and ccc.
2. Calculate the discriminant D=b2−4acD = b^2 - 4acD=b2−4ac.
3. Determine the nature of the roots:
   * D>0D > 0D>0: Two distinct real roots.
   * D=0D = 0D=0: One real root (repeated).
   * D<0D < 0D<0: Complex roots.
4. Use the quadratic formula to find roots: x=−b±D2ax = \frac{-b \pm \sqrt{D}}{2a}x=2a−b±D​​

**Steps in Development**

1. **Knowledge Representation**:
   * Created an ontology in Protégé to define:
     + Concepts: Quadratic equation, coefficients, discriminant, roots.
     + Relationships: "hasCoefficient," "hasRootType," etc.
   * Exported the ontology as an OWL file.
2. **System Interface**:
   * Designed a JavaFX interface with input fields for coefficients aaa, bbb, and ccc.
   * Displayed step-by-step solutions and explanations in a results pane.
3. **Logic Implementation**:
   * Used Java to:
     + Parse user inputs and validate them.
     + Query the ontology for explanations of concepts.
     + Calculate the discriminant and roots.
4. **Integration**:
   * Integrated the ontology with the application using the Jena framework.
5. **Testing**:
   * Conducted unit testing for edge cases (e.g., a=0a = 0a=0, D<0D < 0D<0).
   * Performed usability testing with mock users.

**Limitations**

* Focused only on standard quadratic equations.
* Currently lacks gamified elements to maintain engagement.
* Does not cover alternative solving methods like completing the square.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **aaa** | **bbb** | **ccc** | **Expected Output** |
| All coefficients are zero | 0 | 0 | 0 | All x |
| No solution | 0 | 0 | 5 | No x |
| Linear equation | 0 | 2 | -4 | 2.0 |
| Quadratic equation with no real roots | 1 | 2 | 5 | No x |
| Quadratic equation with one real root | 1 | -2 | 1 | 1.0 |
| Quadratic equation with two real roots | 1 | -3 | 2 | 2.0, 1.0 |
| Large coefficients | 1000 | -5000 | 2000 | 4.0, 0.5 |
| Small coefficients | 0.001 | 0.002 | 0.001 | -1.0 |

**5. Conclusion**

This ITS effectively teaches students to solve quadratic equations interactively using JavaFX and ontology-based knowledge representation. Key lessons learned include the benefits of ontology for modular knowledge structuring and the challenges of integrating it with Java. Future improvements could expand the scope to include more algebraic topics and gamification for improved learner engagement.

**6. References**

1. Wolfram Alpha Team. (2023). *Solving Quadratic Equations*. Retrieved from [Wolfram Alpha].
2. Protégé Documentation. (2023). *Ontology Development for Mathematical Domains*. Stanford University.
3. Oracle. (2023). *JavaFX: Building Modern GUI Applications*. Retrieved from [JavaFX Documentation].

**Prototype Description**

**Interface Features (JavaFX)**

1. Input fields for aaa, bbb, and ccc.
2. Buttons:
   * **Solve**: Calculates and displays roots.
   * **Explain**: Provides a step-by-step explanation of the solution.
3. Results pane:
   * Displays the discriminant and roots.
   * Shows hints if the input is invalid.

**System Logic (Java)**

1. **Input Validation**:
   * Checks if a≠0a \neq 0a=0 (not a quadratic equation otherwise).
   * Handles invalid or missing inputs.
2. **Computation**:
   * Calculates the discriminant and roots.
3. **Ontology Queries**:
   * Fetches explanations for steps like discriminant calculation or root types.

**Expected Output**

1. For a=1,b=−3,c=2a = 1, b = -3, c = 2a=1,b=−3,c=2:
   * Discriminant: D=1D = 1D=1
   * Roots: x1=2,x2=1x\_1 = 2, x\_2 = 1x1​=2,x2​=1
   * Explanation: "The discriminant is positive, indicating two distinct real roots."

# References

1. Anderson, John & Corbett, Albert & Koedinger, Kenneth & Pelletier, Ray. (1995). Cognitive Tutors: Lessons Learned. Journal of the Learning Sciences. 4. 167-207. 10.1207/s15327809jls0402\_2.
2. Brown, Q. (2009). Mobile intelligent tutoring system: Moving intelligent tutoring systems off the desktop. (Ph.D.), Drexel University, Ann Arbor. Retrieved from https://search.proquest.com/docview/304873382?accountid=41304 ProQuest Dissertations & Theses A&I; ProQuest Dissertations & Theses Global database.
3. Burns, Hugh L., & Capps, Charles G. (1989). Foundations of intelligent tutoring systems: An introduction (pp. 11–12). Texas: S LABORATORfj
4. Carbonell, J. R. (1970). AI in CAI: An artificial-intelligence approach to computer-assisted instruction. IEEE Transactions on Man Machine Systems, 11(4), 190–202.
5. Carter, E. E. (2014). An Intelligent Debugging Tutor For Novice Computer Science Students (Ph.D.), Lehigh University, Ann Arbor. Retrieved from https://search.proquest.com/docview/1540757322?accountid=41304 ProQuest Dissertations & Theses A&I; ProQuest Dissertations & Theses Global database.
6. Geeksforgeeks (2024) Real-Life application of Quadratic Equation [Online] <https://www.geeksforgeeks.org/real-life-applications-of-quadratic-equations/> [Accessed December 10, 2024]
7. Graesser, A.C., Conley, M.W. and Olney, A., 2012. Intelligent tutoring systems.
8. Graesser, Arthur & D'Mello, Santosh & Hu, Xiangen & Cai, Zhiqiang & Olney, Andrew & Morgan, Brent. (2011). AutoTutor. 169-187. 10.4018/978-1-60960-741-8.ch010.
9. Keleş, A., Ocak, R., Keleş, Ali, & Gülcü, A. (2009). ZOSMAT: Web-based intelligent tutoring system for teaching–learning process. Expert Systems with Applications, 36(2), 1229–1239
10. Litman, Diane & Silliman, Scott. (2004). ITSPOKE: An intelligent tutoring spoken dialogue system.
11. Ma, Wenting, Adesope, Olusola O, Nesbit, John C, & Liu, Qing. (2014). Intelligent tutoring systems and learning outcomes: A meta-analysis. Journal of Educational Psychology, 106(4), 901–918
12. Mitrovic, Antonija. (2002). An intelligent SQL tutor on the web. International Journal of Artificial Intelligence in Education. 13.
13. Mousavinasab, E., Zarifsanaiey, N., R. Niakan Kalhori, S., Rakhshan, M., Keikha, L. and Ghazi Saeedi, M., 2021. Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods. *Interactive Learning Environments*, *29*(1), pp.142-163.
14. Polson, M. C., & Richardson, J. J. (2013). Foundations of intelligent tutoring systems. New York: Psychology Press
15. Razzaq, Leena & Patvarczki, Jozsef & Almeida, Shane & Vartak, Manasi & Feng, Mingyu & Heffernan, Neil & Koedinger, Kenneth. (2009). The ASSISTment Builder: Supporting the Life Cycle of Tutoring System Content Creation. Learning Technologies, IEEE Transactions on. 2. 157-166. 10.1109/TLT.2009.23.
16. Sujo-Montes, Laura & Tu, Chih-Hsiung & Armfield, Shadow & Yen, Cherng-Jyh. (2021). Assessment and Learning in Knowledge Spaces (ALEKS) Adaptive System Impact on Students’ Perception and Self-Regulated Learning Skills. Education Sciences. 11. 603. 10.3390/educsci11100603.
17. Sottilare, R.A., Brawner, K.W., Goldberg, B.S. and Holden, H.K., 2017. The generalized intelligent framework for tutoring (GIFT). In *Fundamental issues in defense training and simulation* (pp. 223-233). CRC Press.
18. Vanlehn, Kurt & Lynch, Collin & Schulze, Kay & Shapiro, Joel & Shelby, Robert & Taylor, Linwood & Treacy, Donald & Weinstein, Anders & Wintersgill, Mary. (2005). The Andes Physics Tutoring System: Lessons Learned. I. J. Artificial Intelligence in Education. 15. 147-204.
19. Weber, G. and Brusilovsky, P., 2001. ELM-ART: An adaptive versatile system for Web-based instruction. *International Journal of Artificial Intelligence in Education*, *12*, pp.351-384.
20. Walker, Erin & Koedinger, Kenneth & Mclaren, Bruce & Rummel, Nikol. (2006). Cognitive Tutors as Research Platforms: Extending an Established Tutoring System for Collaborative and Metacognitive Experimentation. 207-216. 10.1007/11774303\_21.