

**School of Computer Science**

**Artificial Intelligence Concepts**

**COM7032M**

**Topic: Quadratic Equation Intelligence Tutoring System**

**Student ID:**

Table of Contents

[1. Introduction 3](#_Toc185478499)

[2. Project Plan 3](#_Toc185478500)

[2.1 Milestones, Objectives and timeline 3](#_Toc185478501)

[2.2 Roles and Responsibilities 4](#_Toc185478502)

[3. Literature Review 4](#_Toc185478503)

[3.1 Existing Intelligence Tutoring Systems 6](#_Toc185478504)

[4. Development of Your ITS 9](#_Toc185478505)

[4.1 Developing the ITS 9](#_Toc185478506)

[4.2 Programming and User Interface Development 12](#_Toc185478507)

[4.3 Limitations 14](#_Toc185478508)

[5. Conclusion 17](#_Toc185478509)

[5.1 Lessons Learned 17](#_Toc185478510)

[5.2 Reflections 17](#_Toc185478511)

[References 17](#_Toc185478512)

[References 19](#_Toc185478513)

# 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. |

## Roles and Responsibilities

* Knowledge Engineer: Responsible for developing the ontology in Protégé. Implements the reasoning engine, mathematical solver, and ensures integration with the ontology for dynamic responses and 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

The Intelligent Tutoring System (ITS) is designed for the domain of mathematics, specifically focusing on quadratic equations. This domain was chosen because quadratic equations form a fundamental topic in secondary education, with significant applications in advanced mathematics, physics, and engineering. The system aims to assist students in understanding, solving, and analysing quadratic equations while providing tailored feedback and explanations based on the student's input. Protégé software is used in develop the knowledge reasoning and Java will be used to develop the front-end user-interface.

## ****4.1 Developing the ITS****

To develop the ITS capable of guiding students through the process of solving quadratic equations, understanding the discriminant, and interpreting solutions. The knowledge base was first developed using protégé. The process involves;

* **Defining the classes**

For this system, 5 classes were considered, the QuadraticEquation, Coefficient, Discriminant, Root (and its subclasses: RealRoot, ComplexRoot, RepeatedRoot) and SolutionMethod (and its subclasses: Factoring, etc.).

The QuadraticEquation class acts as the central component of the system, encapsulating the equation's properties and serving as a container for its coefficients, discriminant, and solution methods. It is responsible for computing the discriminant, evaluating the equation, and invoking the appropriate solution methods to find the roots. The Coefficient class is dedicated to storing and validating the values of the coefficients a, b, and c of the quadratic equation. The Discriminant class encapsulates the discriminant Δ=b2−4ac and its properties. It determines the type of roots the quadratic equation has whether they are real, complex, or repeated. The Root class represents the solutions of the quadratic equation and serves as the parent for three specific subclasses: RealRoot, ComplexRoot, and RepeatedRoot. The RealRoot subclass handles cases where the discriminant is positive, representing two distinct real roots, while the RepeatedRoot subclass handles cases where the discriminant is zero, representing a single real root with multiplicity. The ComplexRoot subclass is responsible for representing roots with real and imaginary components when the discriminant is negative.

The SolutionMethod class defines the framework for solving the quadratic equation and allows the system to support various solution techniques. Its subclasses, such as Factoring, CompletingTheSquare, and QuadraticFormula, implement specific methods for solving the equation. This design ensures flexibility by enabling the addition of new solving methods in the future.

* **Adding the Data Properties**

**Data properties** refer to the attributes or characteristics that describe specific aspects of the entities represented in the ontology. They are used to link entities (classes or individuals) to literal data values, such as numbers, text, or dates, providing additional descriptive information about the entities.

The table 2 defines a set of **data properties** used in the ontology to represent key attributes of the quadratic equation's components. Each property links a specific domain (the class it describes) to a range (the type of value it stores), ensuring precise and consistent data representation. The properties hasCoefficientA, hasCoefficientB, and hasCoefficientC belong to the **Coefficient** domain and have a range of xsd:double, indicating that they store the numeric values of the coefficients a, b, and c of the quadratic equation. The property discriminantValue is associated with the **Discriminant** domain and also has a range of xsd:double. It stores the numeric value of the discriminant (= b^2 - 4acΔ=b2−4ac), which determines the nature of the equation's roots (real, complex, or repeated).

Table 2 Data Properties created for the system

|  |  |  |  |
| --- | --- | --- | --- |
| Property Name | Domain | Range | Description |
| hasCoefficientA | Coefficient | xsd:double | Represents the value of coefficient a |
| hasCoefficientB | Coefficient | xsd:double | Represents the value of coefficient b |
| hasCoefficientC | Coefficient | xsd:double | Represents the value of coefficient c |
| DiscriminantValue | Discriminant | xsd:double | Represents the discriminant value |

* **Add Object Properties**

This table 3 describes **object properties** used in the ontology to represent relationships between different entities in intelligence tutoring system. Each property has a defined domain (the class it starts from), range (the class it points to). 4 object properties were created.

* hasCoefficient, which links a QuadraticEquation to its Coefficient instances, representing the equation's coefficients a, b, and c. This property establishes the foundational relationship between the equation and its numeric components.
* hasDiscriminant connects a QuadraticEquation to its corresponding Discriminant, encapsulating the computation of the discriminant (b2−4acb^2 - 4acb2−4ac) as an integral part of the equation's structure.
* determinesRootType links the Discriminant to the appropriate Root type it determines (real, complex, or repeated). This property reflects the key decision-making process in solving quadratic equations based on the discriminant value.
* usesMethod associates a QuadraticEquation with a SolutionMethod, such as factoring or the quadratic formula.

Table 3 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. |

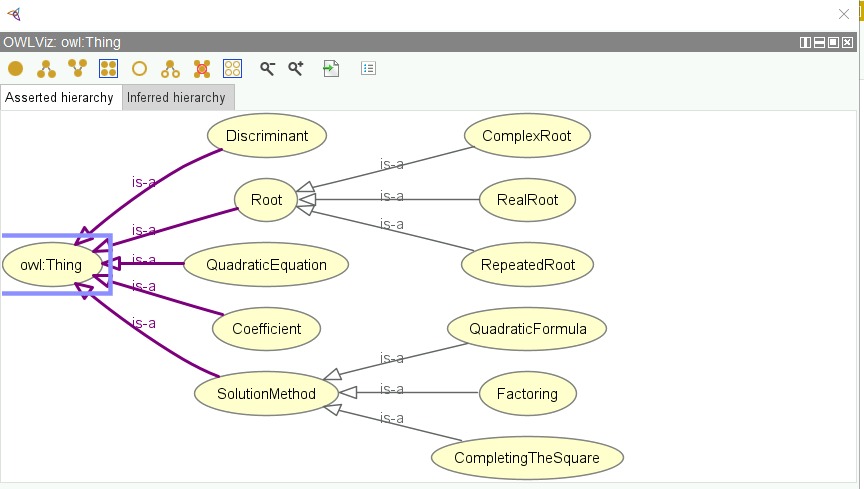


Figure 1 Ontology OWLViz

* **Test the Ontology**

To test the ontology for the quadratic equation system, different individuals were created for the classes. Then, a manual computation of the discriminant and then assigned to the data property discriminantValue for the Discriminant individual. The coefficient was also linked and values where assigned to test the system.

* **Run Reasoner**

The reasoner is launched to verify the relationship using the inferred option. The Ontology is saved to be used in Java for the system.

## ****4.2 Programming and User Interface Development****

**The user interface is designed using** Java with Swing and OWLAPI Library. Java is chosen because of its large eco system, debugging errors would be easy, and swing allows drag and drop for widgets to design an interface. The OWLAPI will allow importing the OWL file into java and assessing data from the reasoner. The major task in this interface development is importing the Quadratic Equation Knowledge OWL file, determine Root type inference using OWL API and reasoning engine, and develop a quadratic equation ITS.

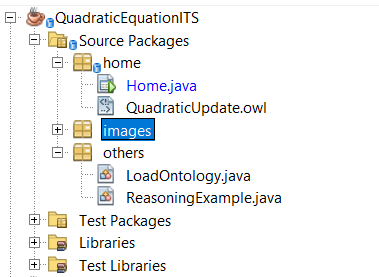
****

Figure 2 Project File Structure

Figure 1 shows the project file structure, the Home.java is the main class, that launches when the program is started. The QuadraticUpdate.owl is the file, added to the project. The LoadOntology is a public class to load the ontology file into the program (check figure 2)

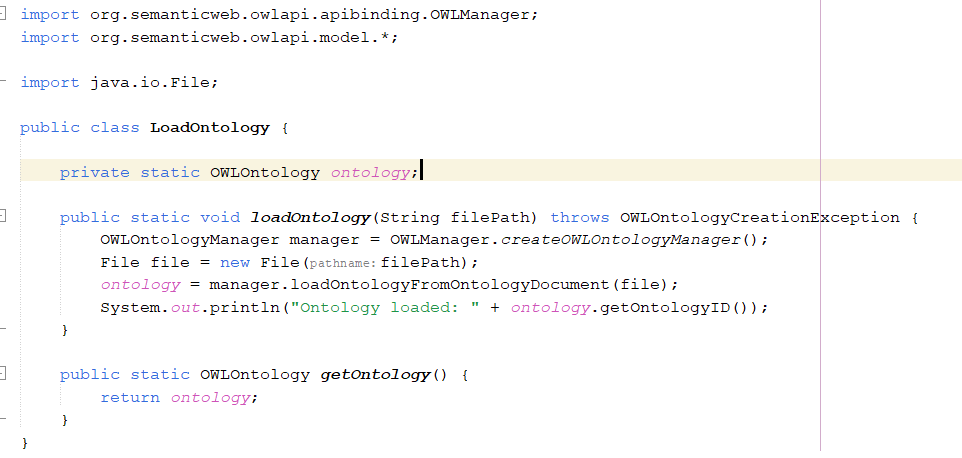
****

Figure 3 LoadOntology Class

While the Reasoning Example is a class that fetches information from the reasoning of the ontology.

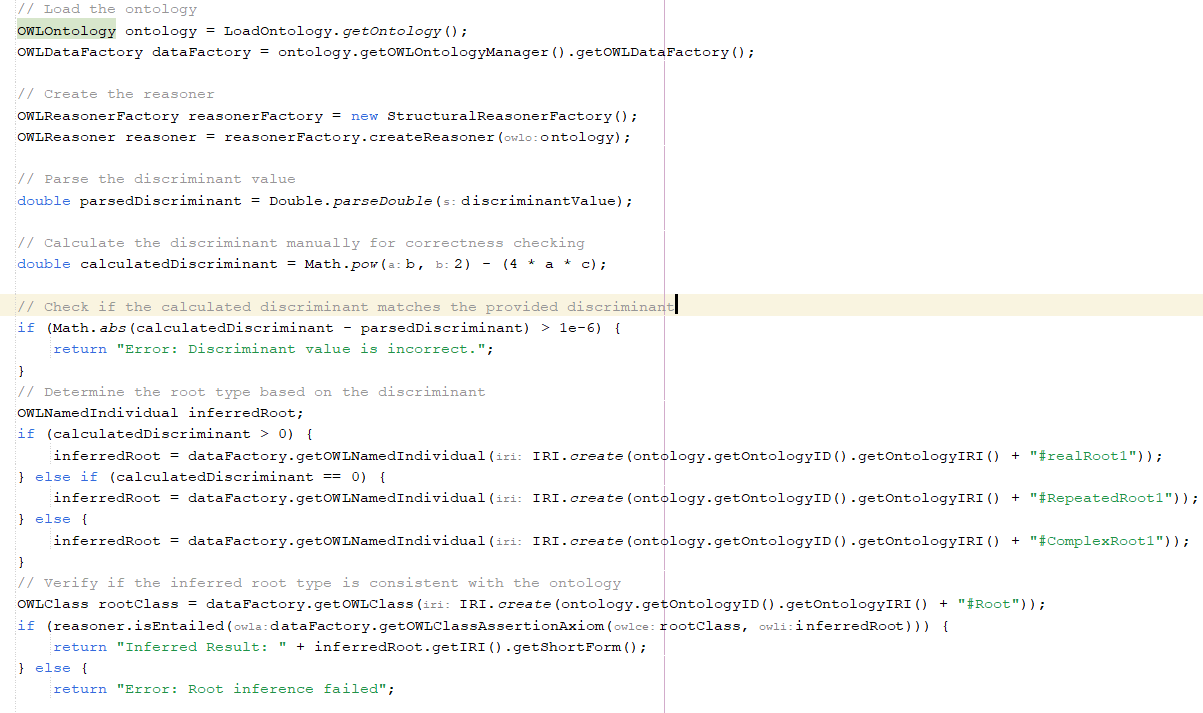


Figure 4 Reasoning Example

This method, inferAndCheckRoot, is designed to determine the type of root for a quadratic equation based on its discriminant value, using an ontology and a reasoner for validation.

It begins by loading an ontology and creating a reasoner to facilitate logical inferences. The discriminant value, provided as a string, is parsed into a double. The method then manually calculates the discriminant using the formula b2−4acb^2 - 4acb2−4ac and checks if the calculated value matches the parsed discriminant. If there is a significant mismatch, it returns an error indicating that the discriminant is incorrect.

Next, the method uses the value of the discriminant to infer the type of root. If the discriminant is positive, it assumes the roots are real and distinct; if zero, it infers repeated real roots; and if negative, it concludes the roots are complex. The inferred root type is then mapped to an individual in the ontology.

To ensure that this inferred root is consistent with the ontology, the method uses a reasoner to verify whether the inferred root belongs to the "Root" class within the ontology. If this inference holds true, the method returns the inferred root type; otherwise, it reports an inference failure.

If any errors occur during this process, such as issues with ontology loading, parsing, or reasoning, the method catches these exceptions and returns a generic error message.

## ****4.3 Limitations****

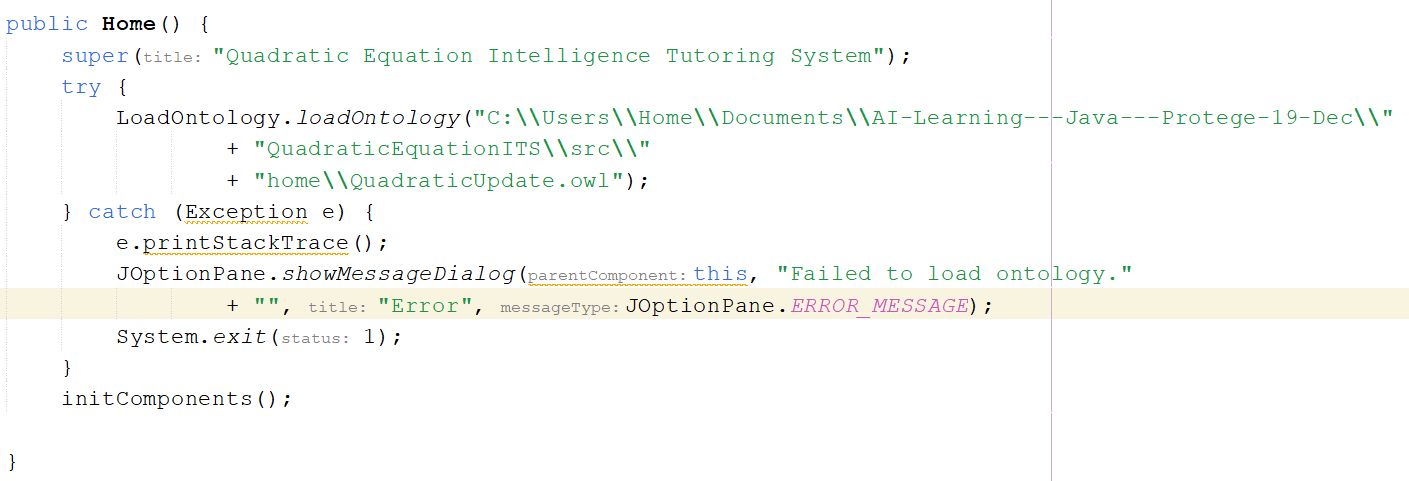
The prototype focuses solely on quadratic equations. It does not extend to other mathematical topics, limiting its applicability in a broader educational context. The reasoning process is constrained by the predefined rules. If students provide input beyond the standard quadratic equation format, the system may fail to interpret it. The feedback provided is rule-based and lacks deep personalization or the ability to understand nuanced errors in the student’s approach.

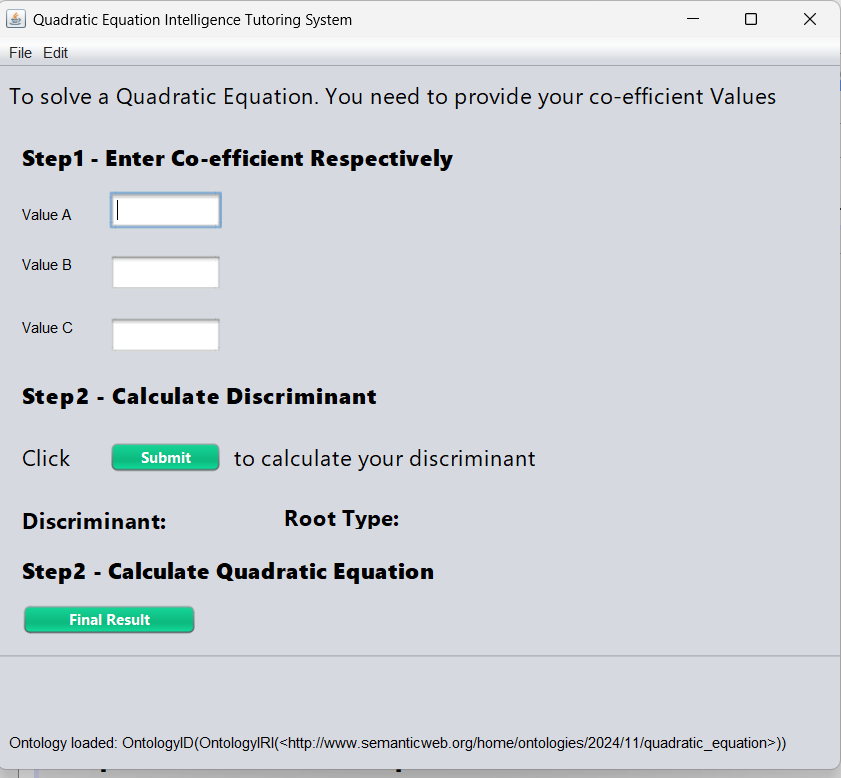
1. **Scalability**:
   1. The current system is designed for quadratic equations with three coefficients. Scaling to include systems of equations or higher-order polynomials would require significant modifications.

#### **Prototype Description**

The ITS prototype integrates an ontology developed in Protégé with a Java-based application. Below are the main components:

1. **Ontology in Protégé**:
   * **Classes**: QuadraticEquation, Coefficient, Discriminant, RootType.
   * **Properties**:
     + hasCoefficient: Links the equation to its coefficients (aaa, bbb, ccc).
     + hasDiscriminant: Links the equation to its discriminant (DDD).
     + hasRootType: Links the equation to the type of roots.
   * **Rules**:
     + D>0D > 0D>0 → RealRoot
     + D=0D = 0D=0 → RepeatedRoot
     + D<0D < 0D<0 → ComplexRoot.
2. **User Interface**:
   * Developed in Java Swing, featuring:
     + Text fields for entering coefficients.
     + Buttons for submitting inputs and calculating results.
     + Labels to display the discriminant, root type, and solutions.
     + Dynamic messages offering hints and feedback.
3. **Reasoning and Feedback**:
   * The OWL API integrates the ontology into the Java application.
   * Based on the input, the system calculates the discriminant, infers the root type, and solves the equation.
   * Feedback is generated dynamically, helping students learn the concepts effectively.

****

****

# 5. Conclusion

In conclusion, this study successfully demonstrates the development of an ontology-driven quadratic equation intelligence tutoring system, effectively integrating ontology concepts and Java Swing for implementation. The solution involves modeling key concepts such as coefficients, discriminants, roots, and solution methods, linking them through well-defined relationships and properties. This structured approach enables dynamic reasoning, step-by-step problem-solving, and interactive user feedback.

## 5.1 Lessons Learned

* The importance of designing a well-structured ontology to capture the domain knowledge comprehensively and facilitate logical reasoning.
* Effective use of data and object properties to represent relationships and attributes enhances the system’s functionality and usability.
* Implementing a tutoring interface requires careful alignment between ontology design and user interaction for clarity and efficiency.

## 5.2 Reflections

This project highlights the value of integrating semantic technologies with programming frameworks to create intelligent systems. While the solution is functional, there is potential for improvement, such as automating the computation of discriminants and expanding solution methods to handle edge cases dynamically. Additionally, incorporating graphical visualizations of solutions and extending the ontology for more complex equations could further enhance user engagement and learning outcomes. Overall, the study provides a solid foundation for building intelligent tutoring systems using ontologies and related technologies.

# 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.