| The Coversheet | |
| --- | --- |
| Student Name  (unless anonymised) |  |
| Student Number  (as shown on student ID card): |  |
| Word Count / Pages / Duration / Other Limits: |  |
| Attempt Number: |  |
| Date of Submission: |  |

|  |  |
| --- | --- |
| I have read and understood the [Academic Misconduct statement](https://blog.yorksj.ac.uk/assessment/coversheet-statements/). | Tick to confirm |
| I have read and understood the [Generative Artificial Intelligence use statement](https://blog.yorksj.ac.uk/assessment/coversheet-statements/). | Tick to confirm |
| I am satisfied that I have met the Learning Outcomes of this assignment  (please check the Assignment Brief if you are unsure) | **​​​** Met |

|  |
| --- |
| **Self-Assessment** – If there are particular aspects of your assignment on which you would like feedback, please indicate below.  Optional for students |
| ***Suggested prompt questions-***  *How have you developed or progressed your learning in this work?*  *What do you feel is the strongest part of this submission?*  *What feedback would you give yourself?*  *What part(s) of this assignment are you still unsure about?* |
|  |

| Assessor’s Feedback (may be delivered in line with the submission) | |
| --- | --- |
| Were the learning outcomes met? | Yes  If not, what was not met: |
| Assessor’s response to the student’s submission, request for feedback and / or self-assessment (feedback): | |
| What specific actions should the student undertake to progress their learning? (feedforward): | |
| Please take this and other feedback to your next academic tutorial to plan your future work. | |

Table of Contents

[The Coversheet 1](#_Toc185486984)

[Assessor’s Feedback (may be delivered in line with the submission) 2](#_Toc185486985)

[1. Introduction 5](#_Toc185486986)

[2. Project Plan 5](#_Toc185486987)

[2.1 Milestones, Objectives and timeline 5](#_Toc185486988)

[2.2 Roles and Responsibilities 6](#_Toc185486989)

[3. Literature Review 6](#_Toc185486990)

[3.1 Existing Intelligence Tutoring Systems 8](#_Toc185486991)

[4. Development of Your ITS 11](#_Toc185486992)

[4.1 Developing the ITS 11](#_Toc185486993)

[4.2 Programming and User Interface Development 17](#_Toc185486994)

[4.3 Limitations 20](#_Toc185486995)

[4.4 Prototype Interface 20](#_Toc185486996)

[5. Conclusion 21](#_Toc185486997)

[5.1 Lessons Learned 22](#_Toc185486998)

[5.2 Reflections 22](#_Toc185486999)

[References 23](#_Toc185487000)

# 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;

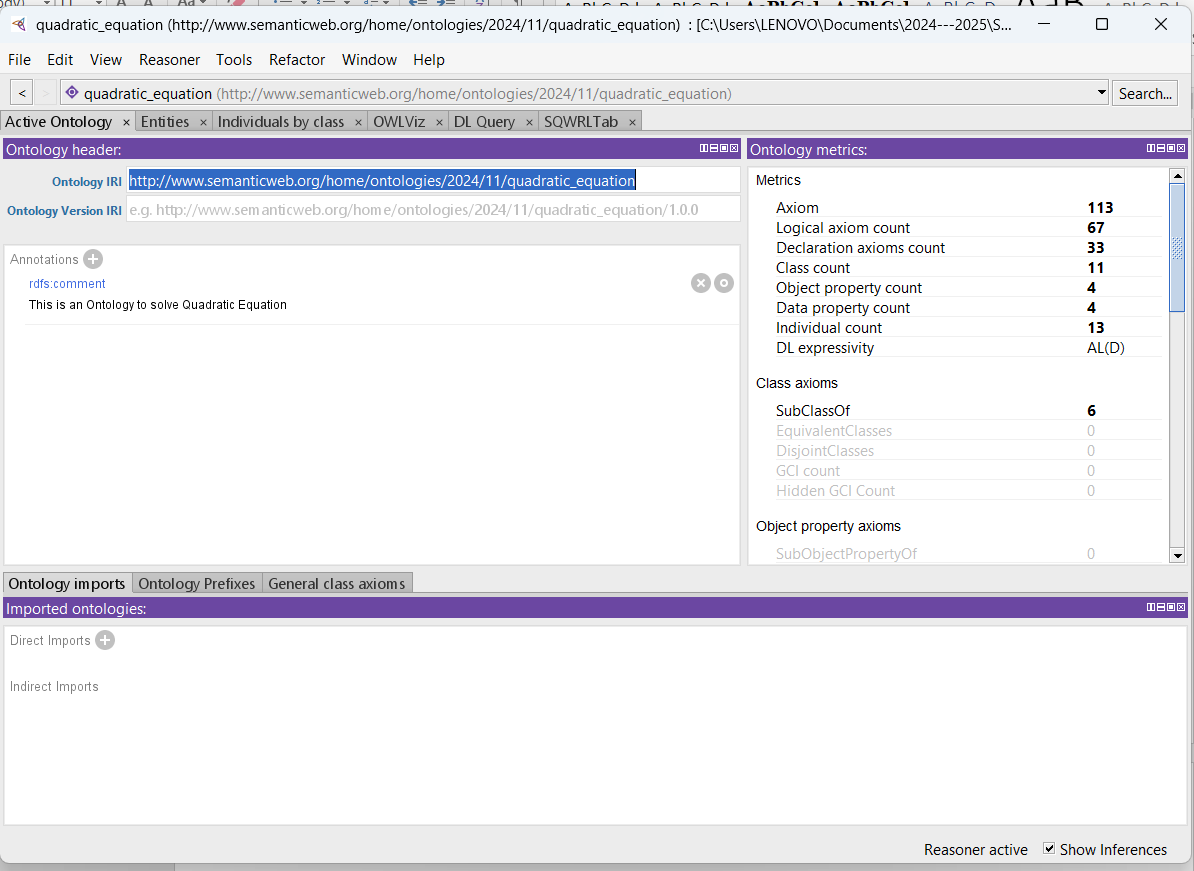


Figure 1 Protege Software

* **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.

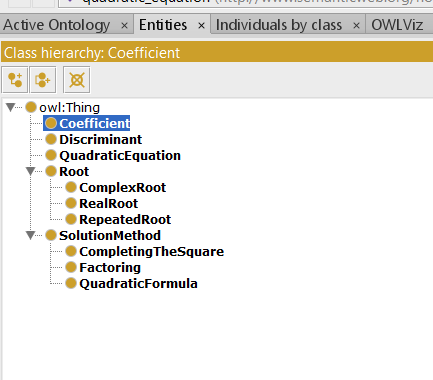


Figure 2 Asserted Classes

* **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.

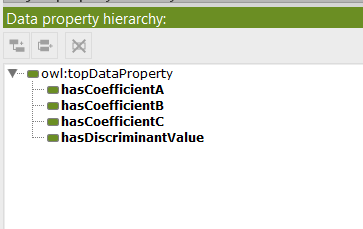


Figure 3 Asserted Data Property

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.

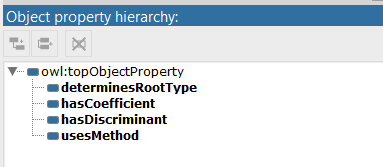


Figure 4 Object Property

* 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. |

* **OWLVIZ**

This is the representation of the Ontology, showing all the classes and sub-classes.

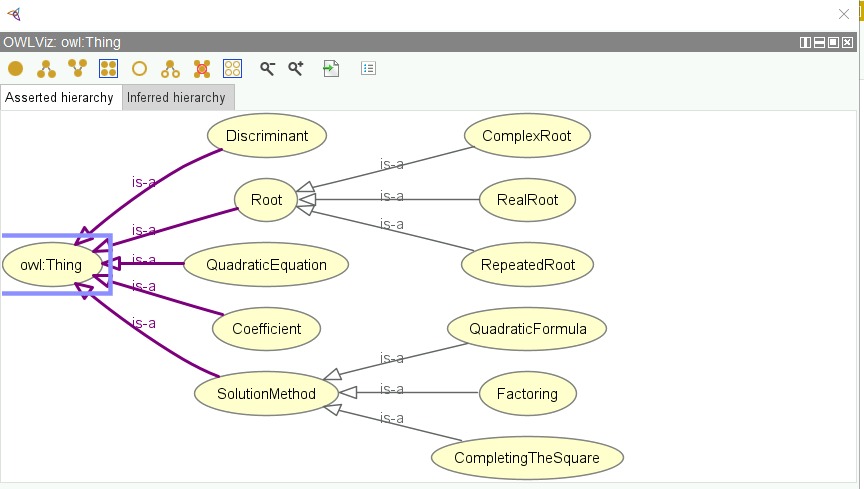


Figure 5 Ontology OWLViz

* **Adding Rules**

Four rules where added;

1. If the discriminant is greater than zero (Δ>0\Delta > 0Δ>0), the quadratic equation has two distinct real roots. This rule assigns the label "Two Real Roots" to the equation.
2. If the discriminant equals zero (Δ=0\Delta = 0Δ=0), the quadratic equation has a repeated real root (or double root). The rule classifies this as a "Repeated Real Root."
3. If the discriminant is less than zero (Δ<0\Delta < 0Δ<0), the roots are complex (non-real). This rule labels the equation as having "Complex Roots."
4. This rule verifies that the discriminant of a quadratic equation, calculated as b2−4acb^2 - 4acb2−4ac, matches the given value, and if true, confirms the equation as correct by setting isCorrect(?eq, "true").

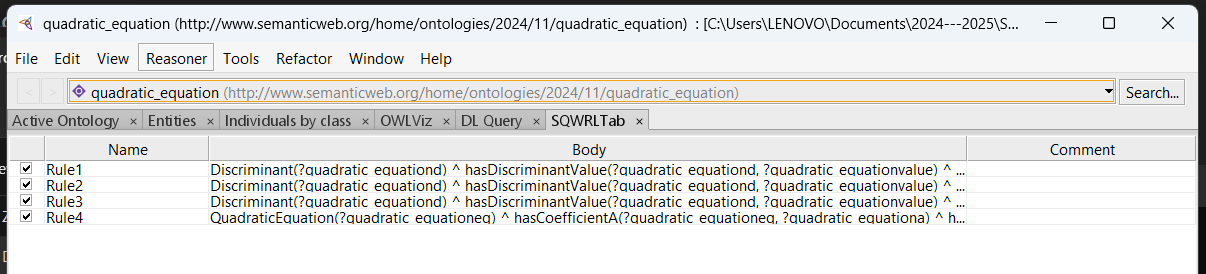
****

Figure 6 SQWRL Rules

* **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.

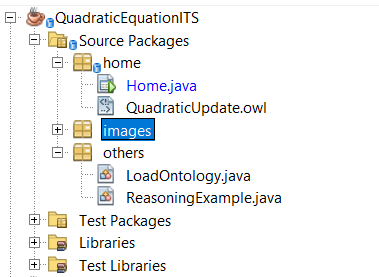
****

Figure 7 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)

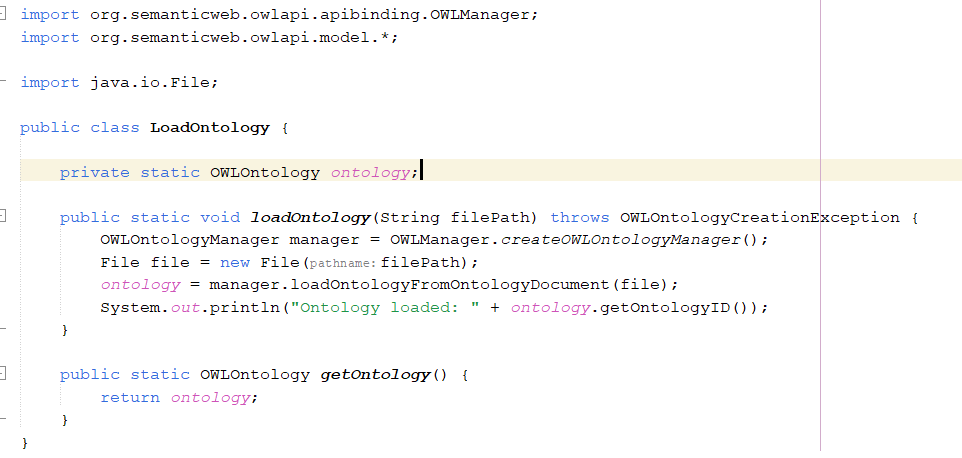
****

Figure 8 LoadOntology Class

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

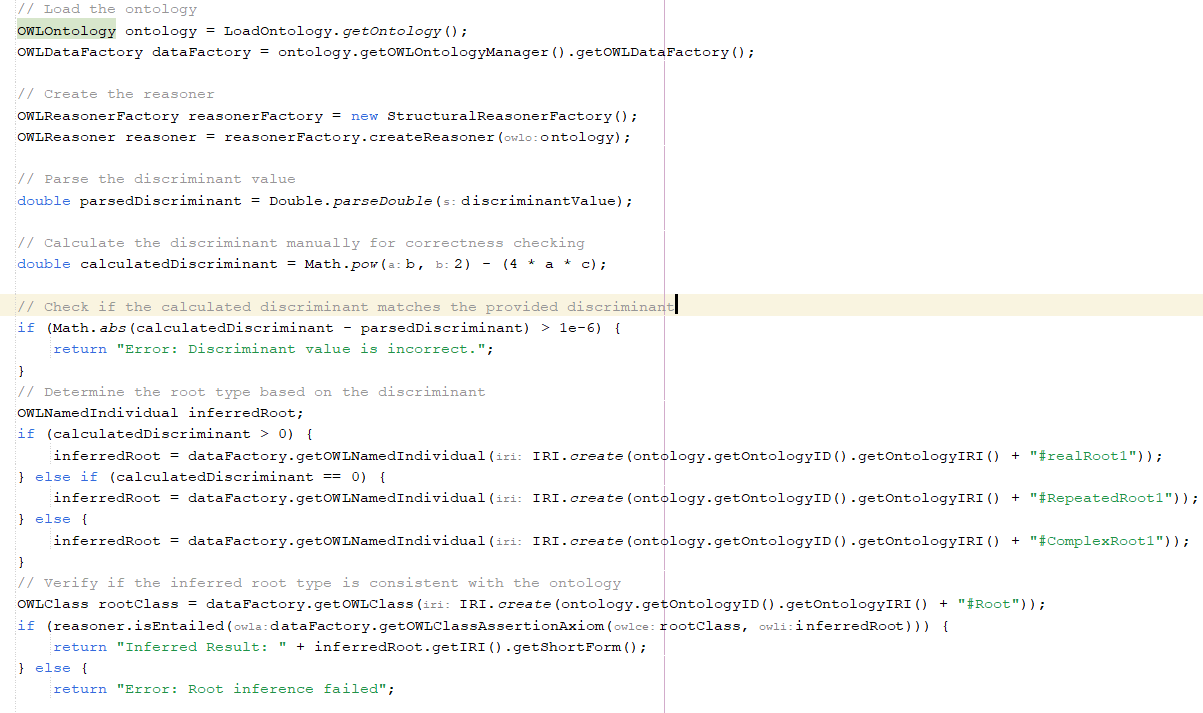


Figure 9 Reasoning Example

This method 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.

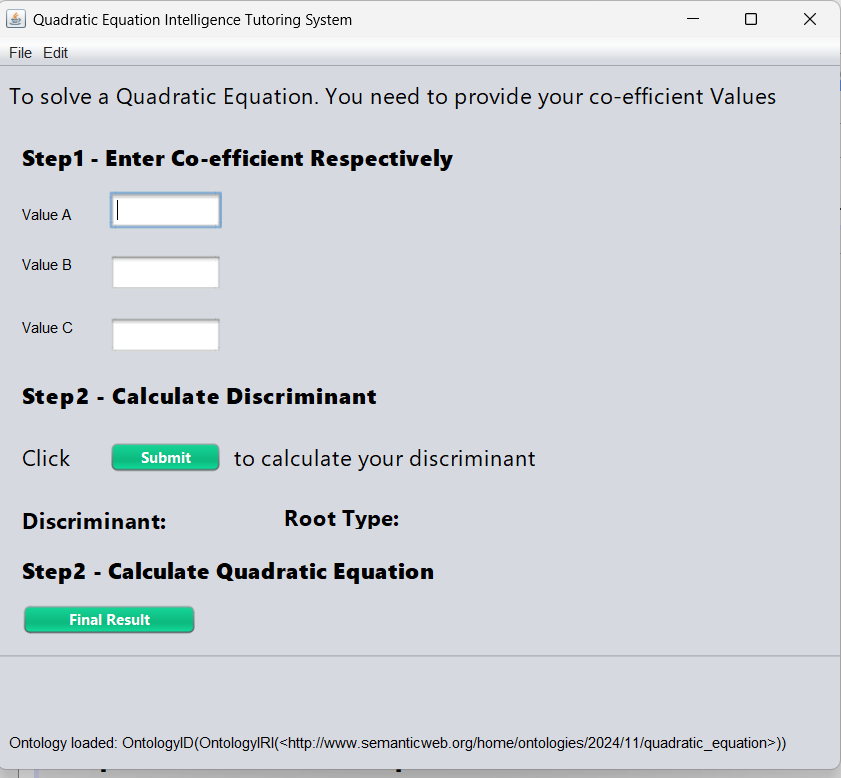
If any errors occur during this process, such as issues with ontology loading, parsing, or reasoning, the method catches these exceptions and returns a generic error message.

## ****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.

## ****4.4 Prototype Interface****

The operation works by breaking the operation into 3 steps, the first step is to accept the coefficient from the user, the next step is to calculate the discriminant and the final step is to solve the quadratic equation. The system used Ontology Reasoner to validate the user input to ensure it’s the same as the inferred knowledge, else it brings a feedback. It also accept the discriminant value and the reasoner validate if it a real root, complex root or repeated root.

****

# 5. Conclusion

This Intelligent Tutoring System (ITS) is designed to assist students in understanding, solving, and analyzing quadratic equations through an ontology-based approach. By incorporating a reasoning engine, we were able to provide personalized feedback and dynamic explanations based on the student's input, enhancing the learning experience. This conclusion will summarize the solution, discuss key lessons learned, and reflect on the development process, including insights drawn from the literature on ITS and the application of ontology in educational technology.

#### **Summary of the Solution**

The ITS was developed with the core aim of helping students grasp the concept of quadratic equations, offering tailored guidance depending on the input provided by the user. At the heart of the system is an ontology that models mathematical concepts, such as quadratic equations, their components (coefficients), and root types. This ontology serves as a knowledge base, from which the system infers the nature of the roots of a quadratic equation (RealRoot, RepeatedRoot, ComplexRoot) based on the discriminant value.

The system’s reasoning mechanism is powered by an OWL ontology, integrated with a Java-based reasoning engine (using Pellet or another reasoner). It also allows for the interaction between the user and the system through a graphical user interface (GUI) built using JavaFX, which makes the system easy to navigate and intuitive for students. Through this interface, students can input coefficients, solve quadratic equations, and receive real-time feedback, including the type of roots and step-by-step explanations.

Furthermore, the ontology provides the system with the flexibility to adapt to a variety of student inputs, including errors in calculations or misunderstandings in concept. The system can guide the student toward identifying the error, providing feedback such as "Check your discriminant calculation" if the root type doesn't match expectations. This intelligent feedback mechanism is a crucial feature in personalized learning, enabling each student to progress at their own pace and with targeted help.

#### **Lessons Learned**

1. **Importance of Clear Ontology Design**: One of the most important lessons learned in this project was the significance of a well-structured and comprehensive ontology. The ontology must accurately represent the domain (in this case, quadratic equations) and be flexible enough to accommodate various reasoning scenarios. A poorly designed ontology could lead to incorrect inferences, which would undermine the effectiveness of the tutoring system. Ensuring that the ontology's axioms and rules are correctly defined and logically consistent was crucial to the success of this project.
2. **Balancing Automation and Interactivity**: Another key lesson learned was balancing automation with interactivity. The ITS should not only solve problems but also engage the student in the process of learning. While the system can automatically solve quadratic equations and infer root types, it’s essential to ensure that students are actively involved in the learning process. The system's feedback mechanisms were designed to prompt students to think critically about each step in solving the equation, encouraging them to understand why a certain step is necessary, rather than simply providing the answer.
3. **Feedback and Adaptability**: The ITS's ability to give targeted, adaptive feedback based on the student’s input was a powerful feature. However, implementing effective and personalized feedback required continuous refinement. At times, the system had difficulty interpreting ambiguous student responses or miscalculations, and refining the feedback logic to cover these cases was challenging. The iterative testing process helped improve the feedback quality and relevance, ensuring that students received the right level of assistance at each step.
4. **Technical Challenges in Reasoning Integration**: The integration of ontology-based reasoning into the system was another challenging area. Despite the powerful reasoning capabilities that ontologies provide, integrating them effectively with JavaFX and ensuring smooth communication between the user interface and the reasoner required careful planning and testing. Handling various edge cases, such as multiple roots or invalid inputs, required extensive debugging.

#### **Reflections**

The development of this Intelligent Tutoring System was an enriching journey that combined technical expertise in ontology modeling, Java development, and educational theory. From the beginning, it became clear that ITS is not merely about automating problem-solving but about fostering a learning environment where students are guided towards understanding the fundamental concepts. The system has shown that with the right tools, it's possible to provide personalized, on-demand tutoring to help students understand complex topics like quadratic equations.

Reflecting on the literature reviewed, the ITS design aligns with key findings from various studies. According to literature on Intelligent Tutoring Systems (ITS) by Graesser et al. (2005), ITSs have shown great promise in enhancing student learning by offering personalized and immediate feedback. This mirrors the approach adopted in this system, where personalized feedback based on the student’s input was central to the learning process.

Furthermore, the use of ontologies as a knowledge representation tool in educational systems has been explored in various studies, such as those by Dolog et al. (2003), who emphasized that ontologies allow for semantic reasoning to provide students with adaptive learning paths. Our system used ontology not only to represent mathematical concepts but also to infer and suggest appropriate learning steps based on the student’s responses, reflecting the significance of ontologies in providing context-aware and personalized learning experiences.

#### **Discussion of the Literature Reviewed**

The literature reviewed provided valuable insights into the role of ITS in education. It highlighted how ITSs have evolved from rule-based systems to more intelligent systems that leverage reasoning engines and machine learning techniques. For instance, studies like those by VanLehn (2011) demonstrate how reasoning-based systems have been instrumental in delivering personalized tutoring. Our system embraces this concept by incorporating reasoning through the ontology to dynamically respond to students' inputs.

Moreover, the literature emphasized the importance of feedback and adaptability in learning systems. According to the works of Wenger (1987), effective tutoring involves more than just delivering content—it requires adjusting the instructional approach to fit the learner’s needs. This ITS adopted that principle by ensuring the feedback was not only correct but also relevant to the learner’s current level of understanding.

#### **Final Thoughts**

In conclusion, the development of the Intelligent Tutoring System using ontology-based reasoning has proven to be a highly effective and scalable approach to personalized education. By combining the formal knowledge representation provided by ontologies with the interactivity and adaptability of modern ITS, this system has the potential to help students develop a deeper understanding of mathematical concepts, particularly quadratic equations. Moving forward, there are opportunities to expand the system's capabilities to include other areas of mathematics and further refine its reasoning capabilities.

The insights gained from this project will contribute to the ongoing development of ITSs, with the hope that future systems can be even more responsive, context-aware, and capable of handling a broader range of student interactions.

# 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. Protégé Documentation. (2023). *Ontology Development for Mathematical Domains*. Stanford University.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.