

# Logic Learning Assistance Tool

Ali Altamimi  
Christopher Brantley  
Nadia Doudou

Harinder Badesha  
Joshua Crofts

*A Visual Improvement to the Pedagogy of  
Introductory Logic*

Senior Capstone Project - Final Presentation



# Why Did We Choose This?

- Formal logic is a tough subject for many students
  - Difficult and unfamiliar notation
  - Proof-based
- Few tools give students what they need
  - Online websites exist, but are few and far between
    - <https://www.umsu.de/trees/>
      - If the website *does* exist, it can be a bit daunting for an intro student
  - Textbooks may give examples, but practice makes perfect
  - Symbols and terminology vary from subject to subject and source to source
  - Want: a way to “combine” all notations and teach the underlying concepts
- Visual aids complement learning experience
  - Interaction is key when learning new subjects!



# What Is The Project?

- A new and improved tool for students to learn propositional and first-order predicate logic
  - Material learned in CSC-350 (Chapters 6-7) and PHI-310 (Introduction to Formal Logic)
  - *Logic Learning Assistance Tool (LLAT - pronounced /L/-LÆT)*
- Design and construct truth tables, truth trees, examine well-formed formulas, and more
  - Complements the traditional lecture(s) with definitions and examples of all symbols
- Aimed at college students, but may also be used by high-schoolers, teachers, and professors
  - Exportable as LaTeX and PDF (*Other formats are planned in future versions*)



# Frontend

- Constructed with JavaFX and follows MVC design
  - Application utilizes BorderPane (top, right, bottom, left, center)
  - Sections were constructed piece by piece and then joined
- Application view takes in wff's and outputs results in appropriate formats, and provides helpful information.
  - Truth tree(Abego library), parse tree(Abego library), truth table
  - Tooltips, rules, and examples
- Connecting views and models
  - Communication between views done with an event bus
    - Each view gets a listener which implements procedures
  - Views and models connected through the controller
    - Controller contains models, view accesses controller



# Language Translation API

- Google Translation API

- Based on Java 8 and our project is using Java 15
  - *As a result, this API was not working!*

- Alternative: Google Script

- A Google Script was created which is used as a simple API call
- Works well for what we need (simpler than a “full API”)
- Cons: performance
  - *Tried to multithread, but it didn't work well...*



# Language Translation API

The screenshot displays the Logic Learning Assistance Tool interface. On the left, there are sections for **Propositional Logic** (with symbols  $\wedge$ ,  $\vee$ ,  $\sim$ ,  $\oplus$ ,  $\leftrightarrow$ ,  $\rightarrow$ ,  $\vdash$ ,  $\models$ ) and **Predicate Logic** (with symbols  $\exists$ ,  $\forall$ ). Below these is an **Errors/Warnings** section. The main area shows a **Settings** dialog box with tabs for **General Algorithms**, **Propositional Logic Algorithms**, and **Predicate Logic Algorithms**. The **Appearance** tab is active, showing options for **Language** and **Advanced**. A **Confirmation** dialog box is open, asking "Are you sure?" and stating "Saving the current changes will require the application to restart. Are you sure you want to continue?" with **Cancel** and **OK** buttons. The bottom of the interface has a **Solve** button and an **Apply** button. On the right, there is a sidebar with sections for **Symbol Name** (And), **Formal Name** (Conjunction), **Alternative Symbols** ( $\&$ ,  $\cdot$ ,  $\wedge$ ,  $\wedge$ ), **Explanation** (The statement  $(A \wedge B)$  is true if A and B are both true. Otherwise, it is false.), and **Examples** (1.  $(A \wedge B)$ , 2.  $((A \supset B) \& (B \supset A))$ , 3.  $((A \vee B) \& (A \vee C))$ ).



# Language Translation API

The screenshot displays the Logic Learning Assistance Tool interface. On the left, there is a sidebar with sections for 'Logique propositionnelle' (propositional logic symbols), 'Prédisez la logique' (predict logic buttons), and 'Erreurs / avertissements' (errors/warnings area). The main workspace shows a 'Settings' dialog box titled 'Apparence' (Appearance) with options for 'Langue' (Language) and 'Avancée' (Advanced). A 'Confirmation' dialog box is also present, asking 'Êtes-vous sûr?' (Are you sure?) and providing a warning about saving changes. The right sidebar contains a table of logic symbols and their properties.

Nom du symbole	Nom officiel	Symboles alternatifs	Explication	Exemples
Et	Conjonction	[&, ·, ^, ∧]	L'énoncé $(A \wedge B)$ est vrai si A et B sont tous les deux vrais. Sinon, c'est faux.	1. $(A \wedge B)$ 2. $((A \supset B) \text{ et } (B \supset A))$ 3. $((A \vee B) \text{ et } (A \vee C))$

At the bottom of the interface, there are buttons for 'Résoudre' (Solve), 'Annuler' (Cancel), 'Sauvegarder' (Save), and 'Appliquer' (Apply).

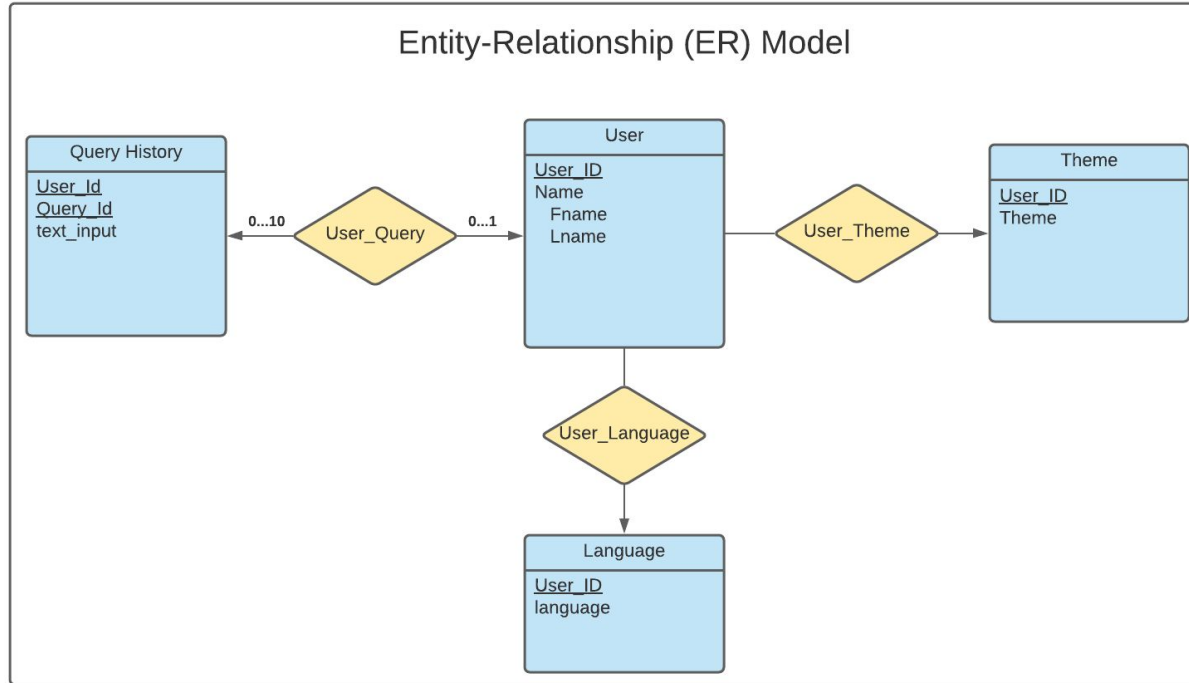


- Database was designed with Amazon AWS database and MySQL
- Database is used for user login credentials for registering and login an account, user input history, and user preference of theme and language
- With this, user can have their preferred theme and language preselected for them automatically upon logging in and their history input



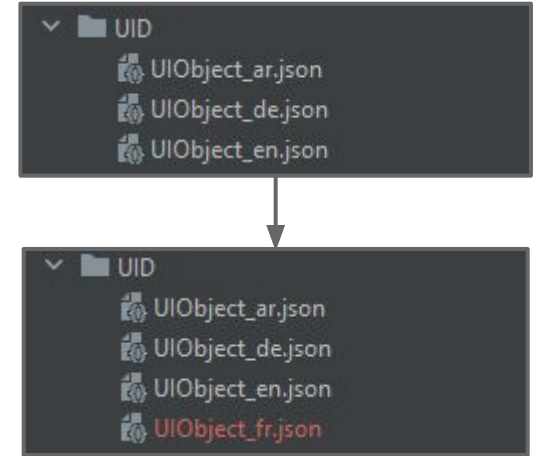


# E-R Diagram

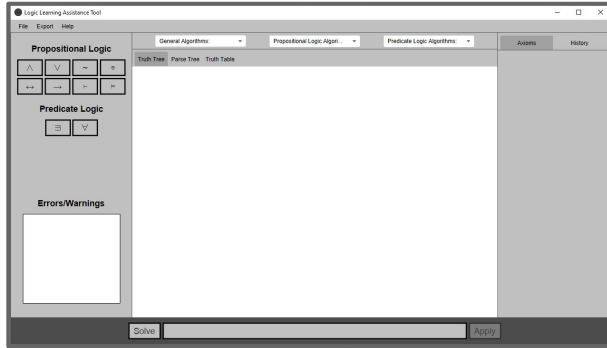


# Local Storage Subsystem

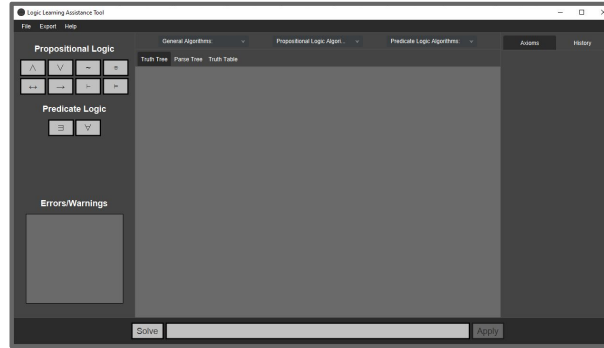
- User Interface Descriptions (UID)
  - Holds all the labels and descriptions of the application.
  - Generate JSON file that holds the missing language
- Settings
  - Holds the changes that the makes in the settings
    - Preferred theme and language
- Credentials
  - Holds user account information when signed in



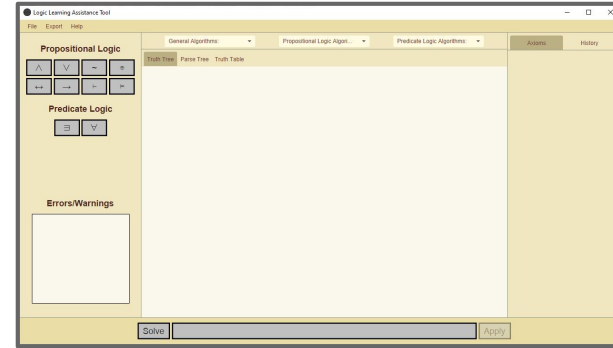
# Different Theme Selection



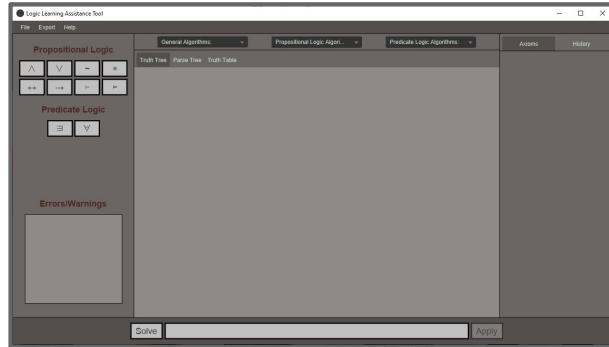
Default



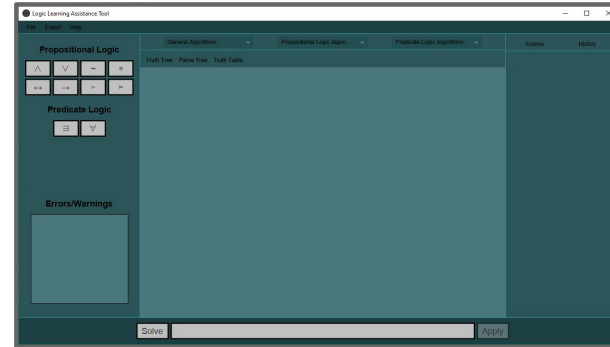
Dark



Tan



Sepia



Teal



# Future Work

- Some of the algorithms we have are *very* complex
  - NP-Complete: SAT
  - co-NP-Complete: Theorem proving
- As a result, we want to find approximations or faster algorithms
  - Truth tables have an atom limit
  - First-order predicate logic is already *semi-decidable*...
- Practice *does* make perfect, but...
  - Our random PL and FOPL formula generators are arbitrary and sometimes undecidable
    - No good research available, hopefully we can improve its efficacy
- Better error messages and solution steps

