Auri's Journey

An Educational Game for Teaching Programming Logic and Computational Thinking Using Adaptive Machine Learning

1 Project Overview

Auri's Journey is an educational mobile application designed to introduce programming logic and computational thinking to beginners, primarily teenagers, through interactive, story-driven problem solving.

The application does not aim to train users to become professional programmers. Instead, it focuses on developing foundational thinking skills such as logical reasoning, abstraction, sequencing, and decision-making, which are increasingly important across many technology-driven fields.

A core component of the application is a machine learning–based learning assistant, which observes user interaction patterns and provides adaptive support in the form of hints, learning tips, and post-task knowledge checks.

2 Educational Motivation

Modern digital platforms often encourage passive interaction (scrolling, tapping, short attention cycles). At the same time, technology has expanded into nearly all professional domains, including healthcare, finance, education, engineering, and business. This creates a need for early development of computational thinking, even among students who do not intend to pursue computer science as a career.

Auri's Journey addresses this gap by:

- Encouraging active thinking instead of passive consumption
- Teaching logic-first programming concepts rather than syntax memorization
- Embedding learning inside a narrative-driven interactive experience

- Using machine learning to support learners individually rather than enforcing static difficulty

3 Narrative Context and Learning Scenario

The game is built around a single, cohesive narrative scenario.

The player follows Auri, a newly created service robot being introduced by a technology company. During a public demonstration, Auri is expected to perform simple tasks that require reasoning and instruction. However, Auri cannot act independently and must rely on the player's guidance.

The story unfolds through illustrated scenes accompanied by short dialogue. At specific moments, Auri pauses and asks for help. The player then completes short programming-related tasks to guide Auri's behavior.

The narrative context provides meaning to each task, transforming abstract programming concepts into concrete actions.

4 Learning Interaction Design

Each learning interaction follows the same structure:

Story Frame

A visual scene is shown (e.g., Auri on stage, in a room, or observing a problem).

Problem Prompt

Auri explains what needs to be done using simple, human-centered language.

User Action Area

The player completes the task using drag-and-drop logic blocks as the primary interaction method. No free-text programming or keyboard input is required.

Execution and Feedback

Auri visually responds to the player's solution. Feedback is delivered through:

- Speech bubbles
- Visual highlights
- Simple textual explanations

Reflection Support

Optional hints or learning tips may appear. After completion, a short knowledge check may be generated.

5 Programming Concepts Taught

Within this single narrative scenario, the following programming fundamentals are introduced progressively.

5.1 Output and Expression

- Displaying text messages
- Understanding that instructions cause visible effects

5.2 Variables and Data Storage

- Assigning values to named containers
- Using variables to store simple data (e.g., names or numbers)

5.3 Sequence and Order

- Executing instructions in the correct order
- Understanding cause-and-effect relationships

5.4 Conditional Logic

- Decision-making using simple conditions
- Conceptual understanding of `if/else` logic

5.5 Comparison and Evaluation

- Comparing values (greater than, less than, equal to)
- Understanding Boolean outcomes (`true` / `false`)

5.6 Iterative Thinking

- Conceptual understanding of repetition
- Recognizing patterns that can be automated

Syntax details are intentionally abstracted away to maintain focus on thinking patterns rather than language mastery.

6 Core Application Features

6.1 Story-Driven Learning

Learning tasks are embedded in a continuous narrative to increase engagement and contextual understanding.

6.2 Block-Based Logic Interface

Users interact with programming logic through visual blocks instead of text-based code, reducing cognitive load.

6.3 Immediate Visual Feedback

Every user action produces a visible change in the game world, reinforcing learning through observation.

6.4 Adaptive Hint System

Hints are provided based on user behavior patterns rather than fixed triggers.

6.5 Post-Task Knowledge Checks

Short reflective questions are generated after task completion to reinforce understanding.

6.6 Beginner-Centered Design

All interactions assume no prior programming knowledge.

7 Machine Learning Component

7.1 Purpose of Machine Learning

Machine learning functions as a learning assistant rather than an evaluator. Its role is to adapt learning support to individual users.

7.2 Input Data for ML Models

The system collects interaction-level data, including:

- Number of attempts per task
- Types of mistakes made
- Order of block placement
- Time spent per task
- Hint usage frequency
- Task completion success or failure

No personal or sensitive user data is collected.

7.3 ML Output

Based on observed behavior, the ML system produces:

- Difficulty adjustment signals
- Context-aware hints
- Short learning tips
- Post-task knowledge check questions

7.4 Training Data Sources

Due to the lack of an initial external dataset, training data is obtained through:

- Synthetic data generation
- Simulated user behavior patterns
- Early user interaction logs
- Data collected during testing phases
- Incremental learning and continuous refinement

Models are designed to perform effectively with small datasets and improve over time.

7.5 Chosen ML Characteristics

The application employs:

- Lightweight models suitable for mobile devices
- Interpretable outputs for educational transparency
- Classification and pattern recognition techniques rather than deep prediction

Example model types include decision trees, rule-based classifiers, and simple probabilistic models.

8 Input and Output Summary

8.1 Application Inputs

- User interaction events
- Block placement actions
- Timing and sequence data

8.2 Application Outputs

- Visual feedback in the game
- Adaptive hints and tips
- Knowledge check questions
- Adjusted difficulty progression

9 Academic Contribution

This project contributes to educational technology research by:

- Addressing passive digital behavior through active learning design
- Introducing programming logic without formal coding syntax
- Demonstrating machine learning as a supportive educational tool
- Providing an adaptive, narrative-driven learning experience for beginners

10 Technology Stack and Implementation

The application is developed using **Swift** and **SwiftUI**, leveraging Apple's modern mobile development framework to ensure performance, accessibility, and a native iOS user experience.

For the machine learning component, **TensorFlow** or an equivalent lightweight ML framework is used for model training and experimentation. Trained models are integrated into the application for on-device or near-device inference, enabling adaptive learning support without requiring constant network connectivity.

This technology stack was selected to:

- Ensure efficient performance on mobile devices
- Support lightweight and interpretable ML models
- Enable seamless integration between learning logic and user interface
- Allow future scalability and refinement of adaptive learning features