

**Nazarbayev University**  
School of Sciences and Humanities  
Department of Computer Science

CSCI 409: Senior Project II  
Final Project Report — Spring 2026

## **Cozmo as a Social Robot for APRIL**

### **Team members**

Zhannur Salkenova  
Sandugash Burambekova  
Rakhman Abdirashov

### **Advisers**

Anara Sandygulova  
Marat Isteleyev

Astana, Kazakhstan  
2026

## 1. Executive Summary

Children diagnosed with Autism Spectrum Disorder (ASD) encounter substantial problems during conversations, interactions and emotional engagement. Traditional therapies are effective but limited in the access of professional therapists, and there is a difficulty in supporting children's attention for a period of time.

The solution was found in social robots that offer an engaging and effective environment for children taking the therapy. Our project is intended to develop 2 things: create a microservice where a Cozmo robot performs as an assistant during therapies and develop a service for therapists to use for the automation of lesson preparation with Cozmo and their customization.

The first outcome of our team's work, a Cozmo Agent Microservice, which is a backend service based on the FastAPI integrated with Anki Cozmo robot to use in the APRIL platform. It consists of 22 scripts, such as interactive games, music, dances, storytellings, movements and animations. During the execution of actions, metrics are collected and computed for further analysis.

The second service is the Script Automation Platform, a full-stack web application ([april-script.qys.kz](http://april-script.qys.kz)). The main goal of this service is to generate lesson plans, quizzes, stories, games, and songs using the AI-assisted chat. The service allows text-to-speech in 3 languages (English, Russian, and Kazakh) and adapts to the robot's SDK while generating the action script.

Our project is adapted to act as an additional tool for the therapists working with children with ASD and address their needs, so improve and digitalize the whole process using the current available technologies.

## 2. Introduction

### 2.1. Problem and Motivation

According to the World Health Organization, 1 in 100 children in the world has ASD. In Kazakhstan, there are over 14500 children diagnosed with ASD. However, the accessibility to the professional treatment is limited, which causes significant challenges for the ASD treatment.

Traditional therapies are developing right now, and robot-assisted approaches are promising additional tools for the ASD treatment process. Social robots increase the engagement, attention and interaction during the therapies. The APRIL platform, developed at Nazarbayev University's Human-Robot Interaction Laboratory, provides a unified framework for deploying multiple social robots (including NAO and Furhat) in therapy sessions with children. However, these robots are expensive (NAO costs approximately \$9,000, Furhat approximately \$25,000), limiting their accessibility in resource-constrained settings.

Cozmo is an alternative for non-affordable and not easily accessible robots for the educational centers in Kazakhstan. Cozmo is advanced and capable of expressive animations, movements,

lighting, audio release, camera use and many more, which is already a useful skill set to improve the therapeutic sessions. Due to Cozmo's potential usefulness, we implemented a new microservice to the APRIL platform supporting Cozmo's functionality.

Another challenge, which is encountered right now by the specialists, is manual lesson planning as the methodology and approaches should be adapted to each child. Furthermore, the ready activities to work with robots are also not adaptive and limited in their variety. Such challenges slow down the work and decrease the quality of the therapy content.

## **2.2. Proposed Solution**

To solve both problems, this project offers 2 systems. Firstly, the Cozmo Microservice integrated into the APRIL platform, which allows the use of the robot during the therapies. The architecture of the microservice is correlated with the existing NAO and Furhat microservices, and the whole implementation was adapted to the APRIL platform with a unified HTTP API for the execution of scripts, status monitoring and data logging. Microservice offers a library with actions, including games, exercises, storytelling and quizzes suitable for the sessions with children.

Secondly, the Script Automation Platform was developed, which offers a web service to create robot interaction scripts using an AI assistant. The platform structurizes lesson plans, generates a corresponding Python file for the robot in use, and compresses everything to download.

One of the main things is that it does not require programming experience and knowledge, while having an advantage of personalized therapies' activities and content in the fastest way without requiring many resources.

## **2.3. Report Organization**

The rest of the report is presented as follows. Section 3 discusses the constraints on designing and implementing the project. In section 4, we will review the relevant prior research literature in the area of child-robot interactions and script authoring tools. Section 5 outlines our project approach including an overview of the overall system architecture, a description of each of the individual components of the system and a description of the roles within our team. Section 6 provides an account of how the project was implemented during two semesters, highlighting design choices made by the group and challenges faced. Section 7 explains our evaluation methods and results. Section 8 summarizes our contribution to knowledge and provides a direction for future study. Section 9 lists all reference citations.

# **3. Identification of Constraints**

## **3.1. Economic Constraints**

Operating under the constraints of an undergraduate Senior Project, the major hardware components - Anki Cozmo Robots - were made available by the Human-Robot Interaction Laboratory at the University. As such, no other hardware components needed to be purchased. From a software perspective, all frameworks/libraries (e.g., FastAPI, SvelteKit, PyCozmo,

Cozmo SDK, MongoDB) employed by the Script Automation Platform are completely open-source and free to utilize. However, there will be a cost per request associated with using the language model capabilities of the Open AI API for the Script Automation Platform. The EZPECS service utilized to provide the text-to-speech capability is an external micro-service run by the research group. Additionally, the deployment infrastructure for the Script Automation Platform (i.e., hosting at april-script.qys.kz) was also supplied through the existing infrastructure of the research lab. In terms of the selection of Cozmo as a therapy robot being an economical advantage; at around \$200 dollars on the secondary market Cozmo is significantly cheaper than NAO (\$9,000) or Furhat (\$25,000), allowing for more accessibility into robot-based therapy.

### **3.2. Environmental and Sustainability Constraints**

As a battery powered robot with relatively low operational time per charge, the operation of Cozmo limits the length of each therapy session. Thus, this factor has led to planning for necessary charging breaks/periods between each therapy session. A key feature of the micro-service is that it is designed to be very lightweight and therefore able to operate off of a standard laptop, eliminating the necessity for a specific piece of server hardware in clinical environments. The Script Automation Platform is developed and deployed as a containerized web application which allows for highly efficient usage of resources through shared cloud infrastructure.

### **3.3. Social and Ethical Constraints**

This project utilizes technology intended for use with vulnerable populations; specifically, children diagnosed with Autism Spectrum Disorder (ASD). Due to working with these types of populations, there are several stringent ethical considerations for this project:

- **Data Protection:** All information collected per session (session logs/sensor events/engagement metrics) should be anonymized and securely stored. Robot should not collect any personal identifiable information about children. Any data collected is only used for therapeutic purposes and stored on secure platforms.
- **Informed Consent:** Therapists and/or parents/guardians should be informed of what kind of data the robot is collecting, how the data will be used, etc. Therapists should be informed that the system is simply a tool to be used under direct supervision and that Cozmo is not autonomous or truly aware of emotions.
- **Safety for Children:** The behavior patterns employed by Cozmo have been designed to be both predictable and child friendly. Each animation/movement/audio employed has been thoroughly vetted to ensure they will not elicit feelings of fear/distress/discomfort. At any time during a session, the therapist may stop all movement/animations performed by Cozmo utilizing the emergency stop endpoint.
- **Sensitivity/Culture:** The Script Automation Platform currently supports three languages (English, Russian, and Kazakh), representing the multi-language environment.

### 3.4. Health and Safety Constraints

Although Cozmo is a small/lightweight robot (< 0.5 kg) posing virtually no physical harm/risk, certain health & safety factors were considered when developing this project:

- **Emergency Stop Endpoint:** There exists an API endpoint (emergency stop) that immediately stops all motion/animations by Cozmo.
- **Behavior Design:** Behaviors employed by Cozmo are designed to move at speeds/amplitudes that minimize startling effects upon children. During each session, the therapist is always present and maintains complete control over all aspects of interaction with Cozmo.
- **Risk Reduction:** Due to its design (operation on a tabletop), the risk of Cozmo causing harm through accidental fall/collision with a child is greatly reduced.

### 3.5. Technical and Manufacturability Constraints

- **Dependency on Cozmo SDK:** The original intention to employ the PyCozmo library (operating without mobile app dependency) had to be abandoned due to issues including audio distortion/packet loss/incomplete functionality. Instead, the team chose to implement the official Cozmo SDK. However, this requires a connection to either a tablet/phone as an intermediary device, though providing robust/comprehensive control of Cozmo.
- **Connectivity Issues:** Cozmo communicates over its own WiFi access point. Therefore, in order to communicate with Cozmo, the controlling laptop must be connected to Cozmo's WiFi network. Since laptops/tablets cannot connect to two different networks concurrently, connecting to Cozmo's WiFi network precludes them from having concurrent connections to both their local Internet connection and APRIL Backend (over WiFi).
- **Hardware Discontinuance:** Although still functional, Anki, manufacturer of Cozmo ceased business operations in 2019. Although the robot remains functional, no new versions exist. The continued development/maintenance of software supporting Cozmo exists solely as open-source community maintained libraries (PyCozmo).
- **Visual Feedback Limitations:** Limited display capabilities exist for Cozmo's 128×32 pixel OLED display; thus limiting visual feedback provided to children during interactions.

### 3.6. Political and Regulatory Constraints

The project operates within applicable political/regulatory boundaries defined by laws/data protection regulation/and applicable regulations governing use of technology with minor children in the Republic of Kazakhstan. Data collection/handling methods have been established to adhere to all applicable privacy regulations. Research is conducted according to institutional guidelines for research involving human subjects at Nazarbayev University.

## 4. Background and Related work

### 4.1. Social Robots in Child-Robot Interaction

Social robots have been tested with children as therapeutic devices or therapy supplements to enable skills training [9]. Robots are hypothesized to be good interaction partners with children with ASD as they can be consistent, patient, fun and socially simpler than human-human interactions. Robots do not change facial expressions unpredictably, can endlessly repeat interactions, and can react to particular behaviors on command through programming. Common robots used with children on the autism spectrum include the humanoid robot NAO [14]. NAO is popular in therapy with children on the autism spectrum as it has a large library of movement behaviors and easily controllable expressions. Furhat is another robot which projects a face onto a translucent mask. This allows Furhat to provide rich facial expressions during social interactions. At Nazarbayev University we control both NAO and Furhat through the APRIL platform. However, their cost precludes many from being used, especially in third world countries where therapy may be even more limited.

### 4.2. Cozmo Robot in CRI

The use-case of children interacting with Cozmo is quite broad. How the robot is utilized changes on a case-by-case basis and over periods of time. Researchers De Jong et al. [1] conducted an extensive study that collected data over many months from children interacting with a domestic social robot. They found that the acceptance dropped off significantly after the first week due to children getting other toys and losing interest/motivation. This points to the importance of content diversity over longer periods of time which this project's Script Automation Platform addresses.

Singh et al. [2] conducted research regarding children using Cozmo robot as tutors in classroom settings. They found that younger children tended to treat Cozmo as more of a toy or pet than a tutor during the first activities, requiring adult intervention to confirm if an answer was correct. As time went on, however, some children began to treat Cozmo as more of a "peer" to play and learn with. This research helped prove that Cozmo is capable of acting as an interaction partner with children given enough time.

Research has also been done on how Cozmo's morphology can affect how it is treated by children. Barco et al. [3] presented children with Cozmo's robot and had them associate it with either the humanoid NAO robot or the animal-like robot Pleo. Most children tended to associate Cozmo with NAO due to its unrealistic but exaggerated features that fell into a middle ground of neither human-like nor animal-like. This could potentially have implications for therapy due to the robot's social presence.

Empathy is another factor that has seen inconclusive results. Pashevich [4] completed a best-evidence synthesis which found that there is no significant increase in empathy with short-term human-social robot interaction with adults. However, there may be positive results

with long-term child-human Social Robot interaction. This further drove the need for diverse actions that can be utilized over long periods of time through a script creation tool.

### **4.3. Skills Development with Cozmo**

The engaging elements of Cozmo's interactions have already shown promise for skill development with young children. While studies directly related to programming skill development with Cozmo are limited, other educational implementations have had positive impacts on young children gaining skills in emotional intelligence [6], computational thinking [7], and STEM [8].

Smith et al. [9] investigated impacts on seventh graders' computational thinking skills, competency beliefs, and engagement as they learned coding with facilitation through Cozmo versus a traditional computer-based program (Scratch). Results showed significant improvement in both classes' computational thinking skills and competency beliefs. Although both interventions were successful, the authors found Cozmo to be better at getting students engaged than Scratch even though students encountered technological difficulties such as application crashes and device lagging.

Rafique et al. [6] implemented Cozmo as an engaging approach to learn Emotional Intelligence (EI) with young children ages 4-8 through coding activities. The study with 500 children participating showed a significant increase in EI score following a week-long curriculum involving lessons on empathy and social responsibility. Children were able to identify emotions that Cozmo displayed through his visual expressions, and were also able to improve programming logic skills including sequencing, conditionals, and loops.

Balasuriya et al. [11] implemented five weekly workshops with Cozmo and six intellectually disabled participants. Results of this study showed participants interacting with Cozmo engaged in collaboration, experienced significant engagement, and positive affect. Participants showed positive affect while playing games with Cozmo that required reacting to emotions the robot displayed differently. Children were also more engaged when they worked with peers and the robot was included.

Druga et al. [7] studied children from ages 7-12 across four different countries interacting with multiple different types of AI agents including Cozmo. Children reported liking to play with Cozmo freely in explorer mode and by giving him commands through the tablet. Children also reported especially liking when they used the robots text-to-speech feature to act like Cozmo and create characters to act out stories. This study found Cozmo effectively increased curiosity surrounding AI concepts.

Jung et al. [13] found that pairing unplugged activities with robotics-based programming had positive effects on elementary students' computational thinking skills and helped decrease cognitive load when beginning to use physical computing devices. More recently, Dong et al. [8] piloted a 9-week afterschool program that utilized social robots to supplement STEM education

and found children had positive gains in how they conceptualized robots' mental capacity and engagement in learning.

#### **4.4. Script Authoring Tools for Therapy Robots**

The development of scripts that define interactions for social robots generally necessitates a developer's level of proficiency in programming which serves as a barrier to entry for therapy professionals who may be knowledgeable about the field, yet do not have experience or training in programming.

A number of visual programming tools including Google Blockly and Scratch have been employed to create robot scripting capabilities; these enable developers to sequence actions via drag-and-drop user interfaces rather than through lines of code. Choregraphe, a visual programming tool for developing NAO-based robotics applications, presents users with a graphical interface to develop robot behaviors but still require a high degree of technical sophistication to utilize effectively.

In recent years, LLMs (large language models) have begun to be successfully utilized on coding generation tasks as they demonstrate their capacity to translate natural language specifications into executable computer programs. As a result, there are now several possibilities to explore using LLMs to generate therapy scripting languages for social robots; specifically, clinicians will provide descriptions of the desired behaviors of the robot in natural language form and receive executable code back.

The Script Automation Platform was also developed within this project as it utilizes this model to combine conversational AI with robot-specific code generation based upon a structured Generate-Critique-Refine pipeline.

#### **4.5. The APRIL Platform**

APRIL is a research platform that has been created at Nazarbayev University to deploy social robots in both therapeutic and educational settings with children having Autism Spectrum Disorder (ASD).

The APRIL platform was designed as a robot agnostic microservices architecture. In other words, each robot type was implemented as its own unique micro-service. All of the supported robots were required to expose a common HTTP API. This design enables the control panel associated with APRIL, which is used by therapists to send commands and receive information from all of the supported robots utilizing the same API.

Before the development of this project, APRIL supported two types of robots: NAO and Furhat. Both robot agents support the same API contract: action execution (queue management, start/stop), robot control/status (connect/disconnect, battery/volume), observing (collect sensor data and engagement metrics), and backend integration (logging session, notify callbacks). The Cozmo microservice developed during this project also supports this same architecture allowing for seamless integration without requiring any modification to APRIL's universal components.

## 5. Project approach

### 5.1. System Overview

The project delivers two software systems:

1. Cozmo Agent Microservice: A FastAPI-based backend service running on port 5052 that controls the Cozmo robot, manages action queuing and execution, collects observations and engagement metrics, and communicates with the APRIL backend. It is containerized via Docker for deployment.
2. Script Automation Platform: A full-stack web application consisting of a SvelteKit frontend and a FastAPI backend with MongoDB storage. It enables therapy specialists to create lesson plans, quizzes, songs, and games for Cozmo, NAO, and Furhat robots through a conversational AI interface. The platform is deployed at [april-script.qys.kz](https://april-script.qys.kz).

Both systems are developed as independent, deployable services that integrate into the broader APRIL ecosystem.

### 5.2. Cozmo Agent Microservice

The Cozmo Agent Microservice is built with APRIL's strict distinction between cross-cutting modules and robot-specific modules in mind. Cross-cutting components, those provided by APRIL and used by every robot agent include: Router (handles incoming API requests), QueueManager (handles ordering of actions), Service (responsible for loading/executing scripts), Action API (facilitates triggering robot actions), and the Observation API + metric framework. Robot specific components developed by our team include the Operations layer which houses cozmo-specific system actions (connection check, volume, battery, etc), the Observer which listens to and collects face detection and engagement events from the cozmo SDK, and our Action Library which contains therapy scripts. Each action launches cozmo's connection, runs the intended script with access to a raw robot object and a cooperative stop, and is designed to automatically end the connection once finished using `cozmo.run_program()`.

#### 5.2.1. Architecture

Cozmo doesn't maintain persistent connection so it connects/disconnects for each action with `cozmo.run_program()`. This `cozmo.run_program()` function blocks the calling thread, connects to the robot over WiFi, runs the async program you provide with a Robot object that's connected to the robot, then automatically disconnects when your program returns.

Each action script exposes a single entry point `run(robot: cozmo.robot.Robot, stop_event: threading.Event)`. The robot parameter is the raw Cozmo SDK object providing direct access to all motor, audio, animation, and sensor functions. The `stop_event` is a `threading.Event` that scripts check before each robot command to support cooperative stopping.

Operations that interact with the system - checking connection, setting volume, getting battery level, etc. - each spawn their own `cozmo.run_program()` so they follow the same connect-run-disconnect pattern.

The application is structured as follows:

```

app/
  main.py          - FastAPI entry point, lifespan management
  config.py        - Configuration (robot name, port, retries, intervals)
  interface/
    robot.py       - RobotInterface (motor, head, audio, animation control)
    operations.py  - Robot operations (connection, battery, volume, stop)
    observer.py    - ObserverInterface: event-driven + polling observation
    metric_computation.py - Metric calculation (face detection, attention, proximity)
  core/
    queue_manager.py - QueueManager (asyncio queue processing)
    service.py      - Service (action execution with retry logic)
    observer_base.py - ObserverThread base class
    types.py        - Internal type definitions
  routers/v1/
    action.py       - Action queuing and execution endpoints
    robot.py        - Robot status and control endpoints
    dependencies.py - FastAPI dependencies (connection checks)
  schemas/
    action.py       - Action model, ActionStatus enum
    api.py          - Request/response schemas
    observation/    - Metrics, RawData, Mood, PersonPerception models
  actions/          - Robot action scripts (organized by category)
    DrawFigures/    - Geometric movement scripts
    QuizFruits/     - Fruit quiz game with audio
    Zaryadka/       - Physical exercise routine with audio

```

### 5.2.2. Operations Layer

The operations layer implements system-level robot operations that map to APRIL's expected interface. Each operation uses its own `cozmo.run_program()` call - connecting to the robot, performing the operation, then disconnecting automatically:

- `connection_operation(ip)`: Verifies the robot is reachable by making it set his head to maximum angle

- `stop_operation(ip)`: Halts all robot actions by calling `abort_all_actions()` and `stop_all_motors()`
- `battery_operation(ip)`: Retrieves battery voltage and converts it to a percentage ( $\text{voltage} / 4.2 \times 100$ )
- `volume_operation(ip, volume)`: Sets speaker volume using `set_robot_volume()`

Since all operations use `cozmo.run_program()` internally, which is blocking, they are executed via `run_in_executor()` in the `Service._operation_wrapper()` to keep the FastAPI event loop free.

### 5.2.3. QueueManager and ScriptRunner

`QueueManager` handles queuing and the lifecycle of executing actions. It holds an `asyncio.Queue` containing actions to be executed and knows which action is currently running.

- `enqueue_action()`: Constructs an `Action` with `QUEUED` status and enqueues it, and starts the processing loop if not already running.
- `stop_all_actions()`: Sets stop cooperative flag, empties queue, sets status of all actions in queue to `STOPPED`, calls `stop_operation()`
- `get_action()` and `get_queue()`: Return action state and queue contents.

`QueueManager`'s `run_loop` dequeues actions, sets status to `RUNNING`, calls the `Service` to execute script, sets callbacks when done firing off HTTP callback to APRIL backend notifying script completed (`{BACKEND_API_URL}/{ROBOT_NAME}/action-completed`), including ID, session ID, status, timestamps, metrics, and raw observed blobs.

The `Service` handles threading of action execution. Cozmo uses a background thread because the cozmo SDK is not safe to run in multiple processes. The robot connection lives in one memory space and cannot be shared across processes using multiprocessing.

The execution follows these steps:

1. `Service.run_action_script()`: Clears stop flag and calls `asyncio.to_thread(_action_runner_wrapper)`, which runs blocking action code in a thread pool while keeping FastAPI event loop free.
2. `_action_runner_wrapper()`: Constructs plain dictionary `shared_observations`, calls `_run_action_module()`, then fires the async completion callback via `run_coroutine_threadsafe()`.
3. `_run_action_module()`: Imports action module dynamically from `actions/{script_path}.py` file using `importlib`, then calls `cozmo.run_program(cozmo_program)`. `cozmo.run_program()` creates `Robot` instance, connects to the robot, and runs async `cozmo_program` function
4. `cozmo_program(robot)`: Begins `ObserverInterface` thread, calls `module.run(robot, stop_event)`. When `module.run` returns, stop `Observer`.

5. After `cozmo.run_program()` returns to `_run_action_module()`: `finalize_shared_observations()` is called. This parses raw observations dictionary into `ObservationsData` object and computes all engagement and attention metrics.

The Service handles stopping actions by setting `_stop_event` - a threading. Event that scripts poll cooperatively before executing each robot command. The robot itself is stopped by calling `stop_operation()`, which aborts `_all_actions()` and `stop_all_motors()`. System operations are also run by the Service with retry logic (`MAX_CONNECT_RETRIES=3` by default) and 10 second timeout for each attempt.

#### 5.2.4. Observer and Metrics Computation

The `ObserverInterface` runs as a thread alongside each action and collects child engagement data using two complementary mechanisms.

Event Handlers (registered with the Cozmo SDK - called automatically from the SDK's event loop):

- `EvtFaceAppeared` / `EvtFaceDisappeared`: Tracks when a face enters or leaves the camera's field of view
- `EvtFaceObserved`: Updates face emotion, 3D position (x, y, z in mm), and engagement zone every time the SDK detects a face update. Facial expression estimation is explicitly enabled (`enable_facial_expression_estimation=True`). Emotion strings from the SDK ("happy", "surprised", "angry", "sad", "neutral") are mapped to standardized labels (Happiness, Surprise, Anger, Sadness, Neutral). Engagement zone is computed from the face's Z depth: zone 1 (< 250mm), zone 2 (250–500mm), zone 3 (> 500mm)
- `EvtObjectTapped`: Records the exact timestamp and cube ID whenever a light cube is tapped
- `EvtRobotStateUpdated`: Detects pickup events by comparing consecutive `robot.is_picked_up` states

Polling Thread (every 0.5 seconds): Reads the current face state maintained by event handlers and appends a snapshot to `shared_observations["raw_data"]["event_agnostic"]`. Each snapshot records whether a human is present, face position and zone, and current emotion.

The observation structure is:

`raw_data`:

`event_agnostic`: list of timestamped snapshots (one per 0.5s poll)

`event_related`:

`cube_tapped`: list of {timestamp, cube\_id}

`robot_picked_up`: list of {timestamp}

Thread safety is handled by `_observations_lock` (a `threading.Lock` in the base class) - all writes from both the polling thread and SDK event handlers are protected by this lock.

The metric computation module computes 15 engagement metrics from the collected observations:

- `attention`: Fraction of snapshots where a face was detected
- `engagement`: Weighted score combining attention (40%), proximity zone (30%), cube interaction (15%), and robot handling (15%)
- `peak_engagement`: Maximum engagement score across all snapshots
- `engagement_consistency`: Standard deviation of engagement scores
- `cube_taps_total`: Total number of cube taps recorded
- `picked_up_count`: Number of times the robot was picked up
- `human_presence_duration_seconds`: Total seconds a face was visible
- `kpi_sustained_attention_span_seconds`: Longest continuous streak of face detection
- `kpi_interaction_density_per_min`: Cube taps per minute
- `kpi_proximity_profile`: Fraction of time spent in each engagement zone (1, 2, 3)
- `avg_smile`: Average smile score across face-present snapshots
- `avg_valence`: Average valence score
- `smile_rate`: Fraction of face-present snapshots where emotion was Happiness
- `dominant_emotion`: Most frequently detected emotion
- `emotion_counts`: Count of each emotion (Happiness, Surprise, Anger, Sadness, Neutral)

### 5.2.5. API Endpoints

The microservice exposes the following REST API endpoints:

Action Endpoints:

- POST `/action/start`: Enqueue a new action (accepts `script_path`, `session_id`, `backend_action_id`, `params`)
- GET `/action/stop`: Emergency stop - terminates current action and clears queue
- GET `/action/current`: Returns currently running action
- GET `/action/{action_id}`: Retrieves action by ID from queue or storage

Robot Endpoints:

- POST `/robot/connect`: Initializes `CozmoManager` and connects to robot
- POST `/robot/disconnect`: Clears stored robot connection
- GET `/robot/status/queue`: Returns current action and queue state
- GET `/robot/status/battery`: Returns battery level

- GET /robot/status: Full system status (current action, queue, battery)
- POST /robot/set-volume: Sets speaker volume

Health:

- GET /health: Returns 204 No Content (used by Docker healthcheck)

### 5.2.6. Action Library

Contents of the microservice's actions library are divided into categories by theme:

Interactive Games:

- QuizFruits/quiz\_sdk.py: Fruit themed quiz game with five questions. Questions are served by pre-recorded MP3 audio files played back using pygame. The child taps Cozmo's light cubes to answer (Cube 1, 2 or 3 for the three available answer options). Correct/incorrect tracking and animated encouragement is provided after each question.
- joystick.py: Tap joystick control movement in realtime. Tap Cube 1 to drive forward, Cube 2 to lift arm up/down, and Cube 3 to turn 90 degrees left/right. Script makes use of cooperative stopping (polling stop\_event) as well as scheduling cube-tap handlers as asyncio tasks.

Physical Activity:

- Zaryadka/exercises.py: Physical workout helper program. Performs a short sequence of physical exercises along with Cozmo to help work out in the morning ("zaryadka" means morning exercise in Russian/Kazakh). Cozmo narrates exercises in audio instructions played back using pygame. Script demonstrates an interruptible sleep helper function which polls the stop event every 100 ms, allowing mid-animation interruption.

Educational Storytelling:

- elochka.py: An interactive rendition of the Russian fairy tale about a little fir tree, using Cozmo's built-in SongNote synthesizer and spoken narration.

Geometric Movement (DrawFigures category):

- DrawFigures/circle.py: Drives in a circle pattern, then plays a celebration animation
- DrawFigures/square.py: Drives in a square pattern
- DrawFigures/triangle.py: Drives in a triangle pattern

Each Action script provides a common interface: an async run(robot\_interface) method which accepts the RobotInterface as a parameter. Scripts should poll robot\_interface.stop\_event (or robot\_interface.\_check\_stop()) to allow for cooperative interruption when stop is requested.

### 5.3. Script Automation Platform

#### 5.3.1. Architecture

The Script Automation Platform is a full-stack web application. It has a web frontend that serves as the therapist-facing interface, providing a layout with a chat interface and a lesson document editor. Also, it has a backend that orchestrates the flow of script generation, stores sessions, generates executable Python scripts, and handles text-to-speech synthesis.

#### 5.3.2. Multi-step Lesson Plan Generation

The pipeline began as a single LLM call. A therapist would provide input, and the model would return a lesson plan. However, it was not reliable enough. The model would select animations unrelated to the activity, or produce translations in Russian and Kazakh that were wrong (it may directly translate “I have a question” to “Я имею вопрос”). It did not improve with more extensive, direct prompting to not to do that. Therefore, the most right solution was to add an additional reviewer call to check the generated lesson plan, and if something was wrong, it would fix the needed points. A single LLM call has transformed to a first call that generates a lesson plan, a second call that reviews the plan and identifies problems, and a third call corrects all critical issues, producing output that passes Pydantic validation.

#### 5.3.3. Lesson Plan Structure

Each lesson plan mainly consists of steps of four types:

- Text: the robot delivers a line in English, Russian, or Kazakh while performing an animation.
- Quiz: the robot poses a question answered by tapping one of Cozmo's light cubes.
- Instruction: a direction for the therapist; the robot takes no action.
- Pause: a timed silence in the interaction.

This rigid structure is followed by story and quiz categories. This structure is not followed by the song and game category. In the song category, LLM analyses the song uploaded by a specialist for beat tracking, and then generates a choreography plan that maps robot movements to sections of the music. In the game category, a therapist can describe a game concept in natural language, and the system produces a complete interactive game script using Cozmo's cubes, display, and movement capabilities.

#### 5.3.4. Frontend

The frontend of the Script Automation Platform is a SvelteKit application with two main parts: a chat interface and a document editor. The chat interface is where therapists interact with the AI assistant. It supports text input, suggested response buttons, multi-select options, and file upload for music. Responses are shown with Markdown formatting. The document editor shows the generated lesson plan as step cards. They are directly editable by users. Therapists, if needed, can add or edit them directly, reorder or delete steps. The panel also lets therapists download a ZIP archive containing the lesson plan in PDF format, the executable Python script for the Cozmo

robot, and all text-to-speech audio files. Application state is saved to the database after each change, so therapists can leave the website without losing their progress. The interface is available in widely used languages: Kazakh, Russian, and English.

### **5.3.5. Session Management**

Sessions are identified by UUIDs and stored in MongoDB. Each session contains enough information so that therapy specialists can resume the work from the last step. Sessions are automatically deleted after 30 days of inactivity.

## **5.4. Third-Party Libraries**

The following third-party libraries were used:

Cozmo Agent Microservice:

- Cozmo SDK (by Anki): Official SDK for robot control, requiring a mobile device as intermediary
- FastAPI: Python web framework for building the REST API
- pygame: Audio playback library for MP3 audio in action scripts
- pydub: Audio processing utilities
- Docker: Containerization for deployment

Script Automation Platform:

- OpenAI API: Large language model for conversational AI, lesson plan generation, and code generation
- SvelteKit: Frontend web framework
- Tailwind CSS: CSS utility framework
- MongoDB with Motor: Document database with async Python driver
- librosa: Audio analysis library for BPM detection and beat tracking in song category
- WeasyPrint: HTML-to-PDF rendering engine which included in ZIP file
- EZPECS TTS: Internally developed TTS with child's voice for text-to-speech audio in English, Russian, and Kazakh

## **5.5. Team Roles and Responsibilities**

Team Members and Role Division:

Zhannur Salkenova: Took the initiative of developing the Cozmo Agent Microservice. Created the backbone of the backend application, docker configuration, api endpoints, and integration with APRIL.

Sandugash Burambekova: Created action script library, completed collection of observation and Engagement metrics, implemented TTS and Audio-playback functionality, aided in APRIL backend integration.

Rakhman Abdirashov: Designed and constructed Script Automation Platform.

Channels of Communication: We used GitHub and Discord to communicate every day to discuss what we would work on. We had weekly meetings with our advisor, Anara Sandygulova.

## **6. Project Execution**

### **6.1. Fall 2025 Semester**

Fall 2025 served as our groundwork semester, wherein we dove into understanding what Cozmo was capable of and developed our microservice prototype.

Weeks 1-4: The team installed development tools/libraries (PyCozmo, FastAPI, MongoDB, Docker) and dug into APRIL's architecture by inspecting the existing NAO and Furhat microservices. The team became acquainted with the robot agnostic design pattern and took note of what pieces needed to be created for Cozmo.

Weeks 5-8: Initial development of Cozmo microservice. Started off by implementing everything with the PyCozmo library as it seemed to allow SDK-free control and didn't need to pair with a mobile app. Implemented life-cycle connection management for the CozmoRobotInterface, along with basic motor functions (drive, head, lift), animation playing, and audio playing. Created operations layer with connection, stop all behaviors, and set volume functions. Planning for Script automation began.

Weeks 9-12: Completed integration of ScriptRunner with dynamic module loading/retry logic. Developed initial slate of action scripts (forward, backward, turn left, turn right, play animation, set light, simple express). Began development of FastAPI endpoint integration. Found and solved initial integration blockers with APRIL backend through mocking out APIs. Created prototype for Scripting style of automation

Weeks 13-14: Tested and finalized initial integration with APRIL backend. Jotted down issues we came across (animations not loading, no equivalent of behavior manager) and set the vision for Spring 2026. By the end of Fall 2025, we had a microservice that could connect to Cozmo, perform some actions, and send/receive messages with the APRIL backend.

### **6.2. Spring 2026 Semester**

During the Spring 2026 semester, the team was most productive as they finished developing Cozmo microservice, developed the entire Script Automation Platform and expanded the action library several times.

Cozmo Agent Microservice (Zhannur Salkenova and Sandugash Burambekova)

Completed the Observer layer for live observations streaming from robot to APRIL backend, implemented using Cozmo SDK event handlers for face, cube taps, and robot state observation as well as a 0.5 second poll-thread. Facial expression estimation was also enabled to measure child emotion during the session. Metrics Computation module developed which computes five different metrics: face detection fraction, attention-to-robot (using cosine similarity metric on gaze estimation), proximity, sitting fraction, and standing fraction. Created action library populated with thematically separated scripts: fruit quiz game, Zaryadka exercise routine, cube joystick action, Elochka fairy tale script, and geometric movement behaviors. Re-architected Cozmo microservice to utilize shared APRIL subprocess-based execution framework. All action scripts are executed in isolated child processes while observations are streamed to the parent process through multiprocessing proxies.

Script Automation Platform (Rakhman Abdirashov):

The entire Script Automation Platform was created during this semester from the ground up. Progression of development:

- backend created with chat endpoint plus automatic lesson plan generation
- added Cozmo robot plugin with robot animation catalog
- integrated quiz generation, and cube-enabled interactions
- added TTS integration through EZPECS service with 3 language support
- added song choreography with music analysis (librosa-based BPM detection)
- added game generation as experimental feature
- added session persistence with MongoDB
- created frontend with Document editor component
- added PDF generation, and bundling ZIP to download all files
- created Docker containerization + deployed to april-script.qys.kz

### **6.3. The PyCozmo to Cozmo SDK Pivot**

The most impactful technical decision we made throughout the project was moving away from the PyCozmo library in favor of using the official Cozmo SDK. It impacted nearly every aspect of our design, so it deserves some discussion.

PyCozmo was listed as the primary library we would use to interface with Cozmo in our project proposal. PyCozmo is an open-source library maintained by community developers which connects directly to Cozmo over UDP, allowing you to control it without needing the official Cozmo smartphone app. The advantage of this was that we wouldn't need to deal with the

complexities of installing an extra piece of software for our therapist to run; they would simply bring a laptop and the robot.

As we began working with PyCozmo, we started running into problems:

1. Audio wouldn't play correctly: Aggressively choppy UDP packets caused severe audio compression. Reliance on text-to-speech or music is necessary for some therapy scripts, so this was a showstopper
2. Everything the robot did was stuttering: Movements would constantly lag/jerk. Smooth predictable movement is critical when working with children with ASD
3. Missing features: PyCozmo lacked several features we knew we'd want to use, such as control over Cozmo's light cubes. Light cubes are used in some of our core games, such as answering quizzes and physically pushing on the joystick. We also lacked several animation triggers we knew we wanted to use.
4. Poor documentation: What documentation existed was hit or miss. Some methods didn't match their definition in the documentation comments. We found ourselves spending an inordinate amount of time debugging through trial-and-error.

We settled on using the official Cozmo SDK. Communicating with Cozmo via the SDK requires having the Cozmo mobile app installed on a smartphone and connected to the robot. The downside to this library was that we now require therapists to have a mobile device as well as a laptop. But the pros far outweighed that single con: smooth, consistent audio playback, buttery smooth motor control, access to all of Cozmo's light cubes, full access to Cozmo's animations, and an overall stable and well documented API.

One microservice architecture decision this fueled was how we interacted with Cozmo. The official Cozmo SDK is not multiprocessing safe. The connection to your Cozmo instance lives in a single memory space, and cannot be passed between processes safely. Furhat's proposed architecture used a subprocess for each action that the robot performed. We instead opted to handle actions in background threads using `asyncio.to_thread()`, with automatic handling of connecting and disconnecting from the robot in the `cozmo.run_program()` coroutine. This meant we did not need a connection manager singleton, or a robot-specific wrapper class for our interface. It resulted in a much cleaner architecture that still fits neatly into the universal `QueueManager` and `Service` layers provided by APRIL.

## 6.5. Design Decisions and Challenges

**Subprocess-Based Execution:** Redesigned microservice to use Python multiprocessing (spawn context) for executing actions, like the rest of the APRIL agent. Action scripts are executed in separate child processes, and observe their environment via proxies provided by `multiprocessing.Manager`. Actions are able to stop cleanly via cooperative multitasking using the `ActionStoppedError` exception: each action script polls `robot_interface._check_stop()` every step, and `ActionStoppedError` bubbles cleanly up the async call stack. Actions can be force-stopped

with `ActionSubprocessManager.terminate_subprocess()`, which issues `SIGTERM` and eventually `SIGKILL` if the subprocess fails to exit after 5 seconds.

**Offline-First Data Logging:** Microservice was designed to log all data locally (to `ended_actions/` as JSON files) and sync to the APRIL backend when possible. This design decision was made because Cozmo's Wi-Fi access point takes up the host laptop's entire Wi-Fi connection.

**Plugin Architecture for Script Automation:** To avoid having to build entirely separate systems for each robot, Script Automation Platform was built with plugin architecture in mind from the start. Plugins were developed to support Cozmo, NAO, and Furhat robots on shared infrastructure.

**Generate-Critique-Refine Pipeline:** Instead of trying to generate perfect lesson plans with a single LLM call, we implemented the three-phase generate-critique-refine pipeline. The critique step catches errors like structural issues and negative/depressing content. The refine step fixes these errors automatically. Lesson plans produced after the pipeline consistently pass Pydantic validation.

**Multilingual Support:** We designed Michael to support English, Russian, and Kazakh. This decision was made because of the bilingual (Russian/Kazakh) therapy context in Kazakhstan. Implementation included integration with trilingual TTS, allowing for fully multilingual content in steps included in lesson plans, and internationalized labels in the frontend.

## 6.6. Teamwork and Collaboration

The team adopted an Agile workflow for its collaboration process. It held weekly group meetings; utilized daily messaging through Discord to communicate amongst themselves. For the management of code versions, the team used GitHub.

Development responsibilities were split so that members could develop their respective parts of the project in parallel. Development of the Cozmo Agent Microservice and Script Automation Platform were conducted independently. This allowed the team to work in parallel without impeding one another.

Zhannur Salkenova served as the primary contact for coordination efforts with the APRIL platform team. She also made sure the architecture was compatible. Sandugash Burambekova was responsible for the content and interaction portions of the project, including developing action scripts to meet therapeutic needs. Rakhman Abdirashov designed and implemented the full-stack automation of the Script Automation Platform, which included designing and creating the database to deploy it to production.

Collaborative testing of alternatives to resolve the conflict regarding the PyCozmo library decision occurred. Both libraries were evaluated based upon empirical evidence. Demonstrations were scheduled with the advisor on a regular basis to ensure that all parties involved remained aligned with project objectives, providing an opportunity for immediate and ongoing feedback.

## 7. Evaluation

### 7.1 Evaluation Approach

In order to verify whether the Cozmo Agent Microservice and Script Automation Platform solved the issues outlined in the introduction we set out to answer: (1) does Cozmo present children with ASD interesting and age-appropriate interaction; and (2) can non-technical therapy specialists author content for the robot using the Script Automation Platform?

### 7.2 Evaluation Method

5 Participants over 3-4 sessions (~20 minutes each) were interviewed either in person or remotely. An expert in child and psychological science was consulted on clinical soundness of generated content. Two teachers were consulted on Script Automation Platform usability as proxy non-technical users. Team members performed session observation with children using the Cozmo robot and took notes on behavioral reactions that would inform future feature decisions. Notes were taken during semi-structured conversation.

### 7.3 Results

Children's engagement during cube-based quiz interactions was immediately visible, they leaned in. Interest in interaction was apparent during quiz activities using the cubes. Children leaned in, tapped more quickly, and were animated by Cozmo's reactive animations. Assessment: cube UI was working and there was face-validity for increasing the animations library to support over 20 reactions/animations (joy, cheer, sadness, surprise). Song choreography movements - summoning Cozmo to dance along to music streamed from PC - maintained attention and caused children to mirror movements. Storytelling and geometric movement behaviors saw similar reactions. The robot's stature and lack of threatening features was universally cited as a positive for intended users. Two clinical soundness concerns were raised by the child / psychological science consultant: (1) stories were too long for auditory learners (many children with ASD learn best through hearing) and needed to be segmented into smaller repeated steps; and (2) while trilingual TTS was supported, authored content regressed to English language output when prompting in Russian, and asset labels/animations were only present in English.

Teacher experts found UI lucid and single page. AI generated lesson plans were easy to follow. Pain points included: font size was too small; no pre-made game/activity templates; unable to attach image assets to steps; unable to drag to reorder steps; unknown how to restart session or why chat would become seemingly disabled on transitions.

### 7.4 Analysis

As demonstrated by our evaluation, the Cozmo Agent Microservice and Script Automation Platform functioned as intended and addressed the challenges described in the introduction. Cozmo was able to hold children's attention in a fun way and non-programmers were able to use

the Script Automation Platform. Content language consistency, story length, and missing authoring affordances are all identified gaps that will be addressed before the upcoming release.

## 8. Conclusion and Future work

### 8.1. Conclusion

Below are the two contributions the project made to the APRIL platform.

- We contributed the Anki Cozmo robot as an additional working robot agent to the APRIL ecosystem. Cozmo was added through APRIL's universal robot API. This meant we did not have to make any changes to APRIL to support Cozmo. Cozmo records five sources of engagement signals during therapy sessions: face detection, attention, proximity, sitting, standing. Logs all that data appropriately to the backend. Cozmo is significantly cheaper than NAO or Furhat. Thus we can provide better budget options for clinics with less funding. We also proved Cozmo can function as a therapy robot despite being discontinued by Anki. Changing our API from PyCozmo to official fixed all reliability issues.
- We developed a Script Automation Platform which allows therapy experts to write robot scripts without the need of a developer. Describe your activity in natural language and the system will create a lesson plan for you, test that lesson plan, and help you make changes to that lesson plan. Compatible with English, Russian, and Kazakh speech. Easily integrates with Cozmo microservice. Prior to this, if you wanted a new activity for your therapy session you needed a programmer to create it from the ground up. The portal can be found at [april-script.qys.kz](http://april-script.qys.kz).

### 8.2. Future Work

This project successfully achieved its goals however there are additional features that could be added to create a better experience when using Cozmo robot for therapy:

- Better Engagement Metrics: Right now our engagement metrics utilize the sensors Cozmo already has. We could potentially attach an external webcam that could analyze facial expression or detect child movement. This would allow us to see how engaged a child really is during the course of the interaction.
- Deployment + Data Collection: In the future we plan to be sending out our Cozmo robot + iPad to therapy centers that specialize in autism therapy. Therapists will administer activities with children and collect real data from the interactions. This data can then be used to see how effective current activities are, what behaviors children commonly did during the activities, and allow us to tweak social scripts and engagement pipeline when needed.
- Expand Scriptability: As of now we have 4 categories that Cozmo can fulfill. Story, game, quiz, and song. Each of these categories have a certain pre-determined pseudo code that goes along with it. For instance if the step is in the story category Cozmo will say some TTS and play an animation. While this works for most of what specialists need,

it doesn't allow too much room if you wanted to maybe plug a game in the middle of a story or tell a story based on user input.

- Script Automation Platform is its own website not integrated with the main APRIL website. It's functional but feels like its own standalone website. It would be ideal to integrate it into the APRIL website as its own dedicated page.
- Extend Nao & Furhat to ability to generate scripts: We could look to allow script generation for these robots as well.

## 9. References

- [1] C. De Jong, J. Peter, R. Kuhne, and A. Barco, "Children's acceptance of a domestic social robot: how it evolves over time," *ACM Transactions on Human-Robot Interaction*, vol. 13, no. 2, pp. 1-20, 2024.
- [2] D. K. Singh, S. Sharma, J. Shukla, and G. Eden, "Toy, tutor, peer, or pet? Preliminary findings from child-robot interactions in a community school," in *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 325-327.
- [3] A. Barco, C. de Jong, J. Peter, R. Kuhne, and C. L. van Straten, "Robot morphology and children's perception of social robots: An exploratory study," in *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 125-127.
- [4] E. Pashevich, "Can communication with social robots influence how children develop empathy? Best-evidence synthesis," *AI & SOCIETY*, vol. 37, no. 2, pp. 579-589, 2022.
- [5] D. DiPaola, "How does my robot know who I am?: Understanding the impact of education on child-robot relationships," Ph.D. dissertation, Massachusetts Institute of Technology, 2021.
- [6] M. Rafique, M. A. Hassan, A. Jaleel, H. Khalid, and G. Bano, "A computation model for learning programming and emotional intelligence," *IEEE Access*, vol. 8, pp. 149616-149629, 2020.
- [7] S. Druga, S. T. Vu, E. Likhith, and T. Qiu, "Inclusive AI literacy for kids around the world," in *Proceedings of FabLearn 2019*, 2019, pp. 104-111.
- [8] J. Dong, S. Yu, K. Choi, and M. Jeon, "A child-robot theater afterschool program can promote children's conceptualization of social robots' mental capacities and engagement in learning," *Frontiers in Robotics and AI*, vol. 12, p. 1529421, 2025.
- [9] S. Smith, E. Novak, J. Schenker, and C.-L. Kuo, "Effects of computer-based (Scratch) and robotic (Cozmo) coding instruction on seventh grade students' computational thinking, competency beliefs, and engagement," in *International Conference on Intelligent Human Computer Interaction*, Springer, 2021, pp. 325-336.

- [10] J. Dong, S. Yu, K. Choi, and M. Jeon, "A child-robot theater afterschool program can promote children's conceptualization of social robots' mental capacities and engagement in learning," *Frontiers in Robotics and AI*, vol. 12, p. 1529421, 2025.
- [11] S. S. Balasuriya, L. Sitbon, M. Brereton, and S. Koplick, "How can social robots spark collaboration and engagement among people with intellectual disability?," in *Proceedings of the 31st Australian Conference on Human-Computer-Interaction (OzCHI '19)*, ACM, 2020, pp. 209-220.
- [12] S. Druga, S. T. Vu, E. Likhith, and T. Qiu, "Inclusive AI literacy for kids around the world," in *Proceedings of FabLearn 2019*, 2019, pp. 104-111.
- [13] Y. Jung, M. Tscholl, and Y. Xie, "Enhancing computational thinking in upper elementary: A two-phase study examining the effect of unplugged activities," *International Journal of Child-Computer Interaction*, p. 100765, 2025.
- [14] A. Sandygulova, G. I. Dragone, and G. M. P. O'Hare, "Privet! Exploring effects of a robot speaking a foreign language on child-robot interaction," in *Proceedings of the 14th International Conference on Interaction Design and Children*, ACM, 2015.
- [15] OpenAI, "GPT-4 Technical Report," arXiv preprint arXiv:2303.08774, 2023.
- [16] Anki Inc., "Cozmo SDK Documentation," 2018. [Online]. Available: <https://developer.anki.com/>
- [17] PyCozmo Contributors, "PyCozmo: A pure-Python communication library for Cozmo," GitHub repository, 2020.
- [18] S. Tiranno, "FastAPI: A modern, fast web framework for building APIs with Python," 2019.
- [19] R. Harris, "SvelteKit Documentation," Svelte Society, 2024.
- [20] MongoDB Inc., "MongoDB Manual," 2024.