Design Review of Speech Synthesis Web App

Presented by SynthLang

Intro

Jalen Jensen

Paul Mayoral

Tyler Bryant

Logan Hunt

Problem Statement - The Big Picture

Why?

- The Global Importance
- The Data Gap
- The Impact

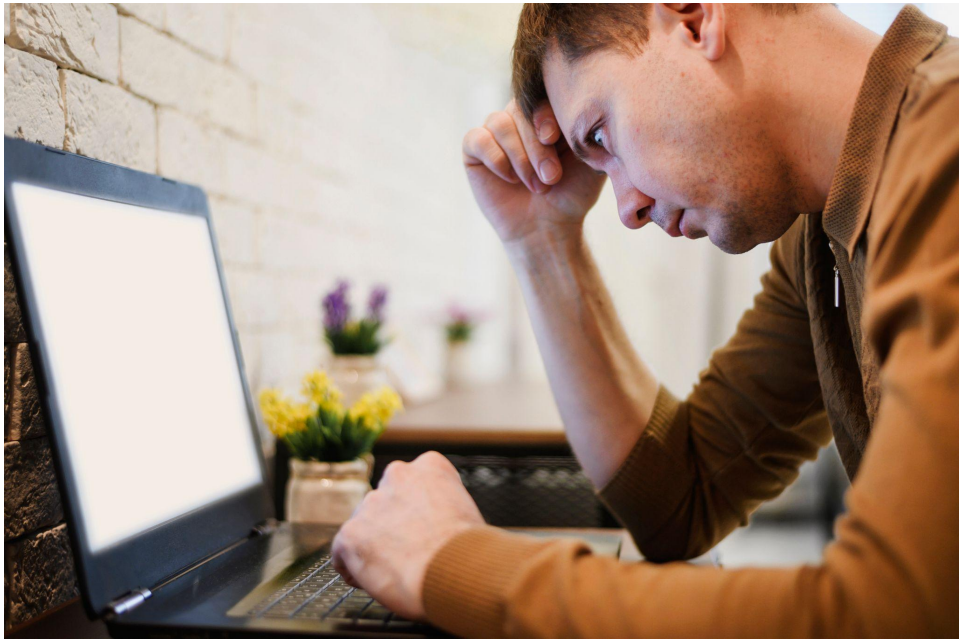
Problem Statement - The Sponsor

- Dr. Benjamin Tucker - leading researcher in indigenous languages.
- Creating TTS models for low-resource languages.
- Building a framework supporting dozens of under-represented dialects.

Problem Statement - The Pain Points

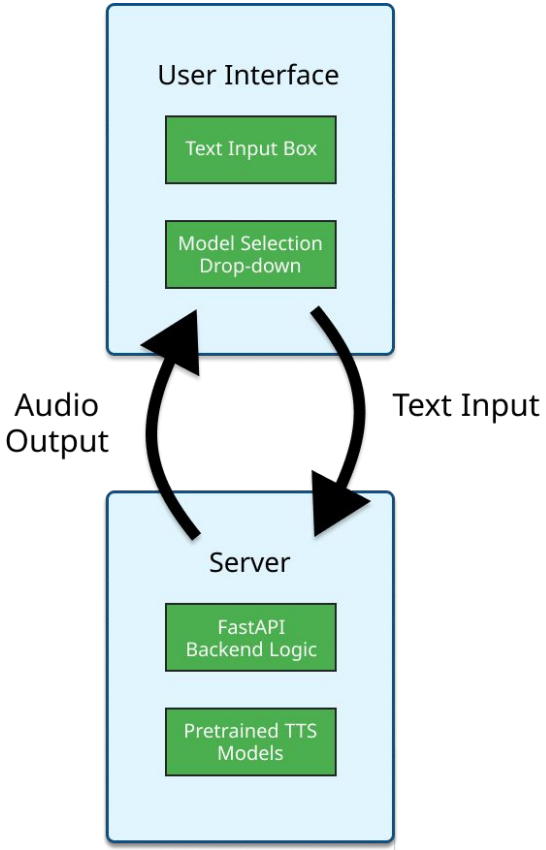
Specific Pain Points:

- Legacy Dependency
- The "Black Box" Barrier
- Computational Bottleneck
- Fragile Infrastructure



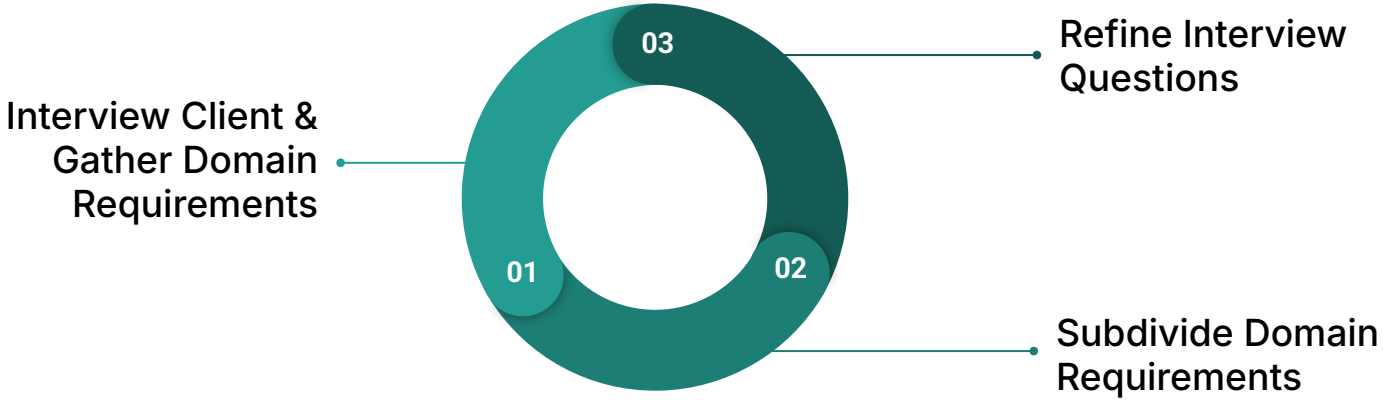
Solution Overview

- An efficient web interface
  - Version controlled to allow long-term support
- Customizable Text-to-speech Training Pipeline
  - Runs offline, separate from the web interface
  - Abstraction layer allows for swapping of models with little to no coding



Requirements Acquisition

- Iterative, directed interviews
- Synthesis with tech challenges



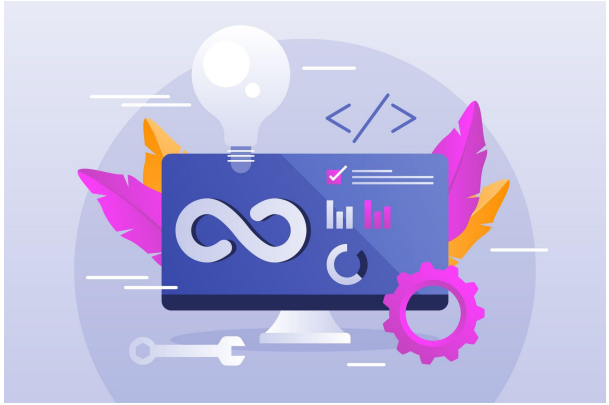
High-Level Requirements

Training Pipeline

- Easy to use
- Configurable
- Reproducable

Web Application

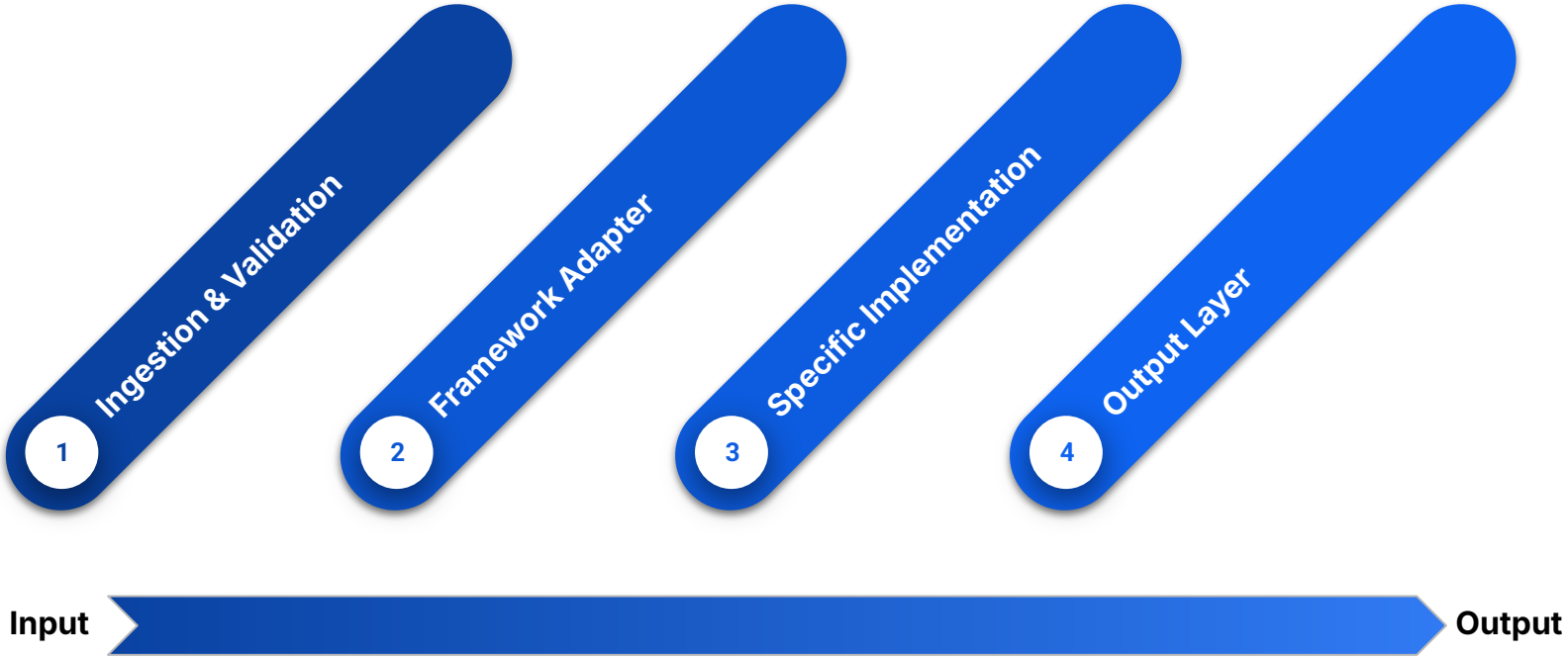
- Upload Models
- Run Inference / Generate Audio



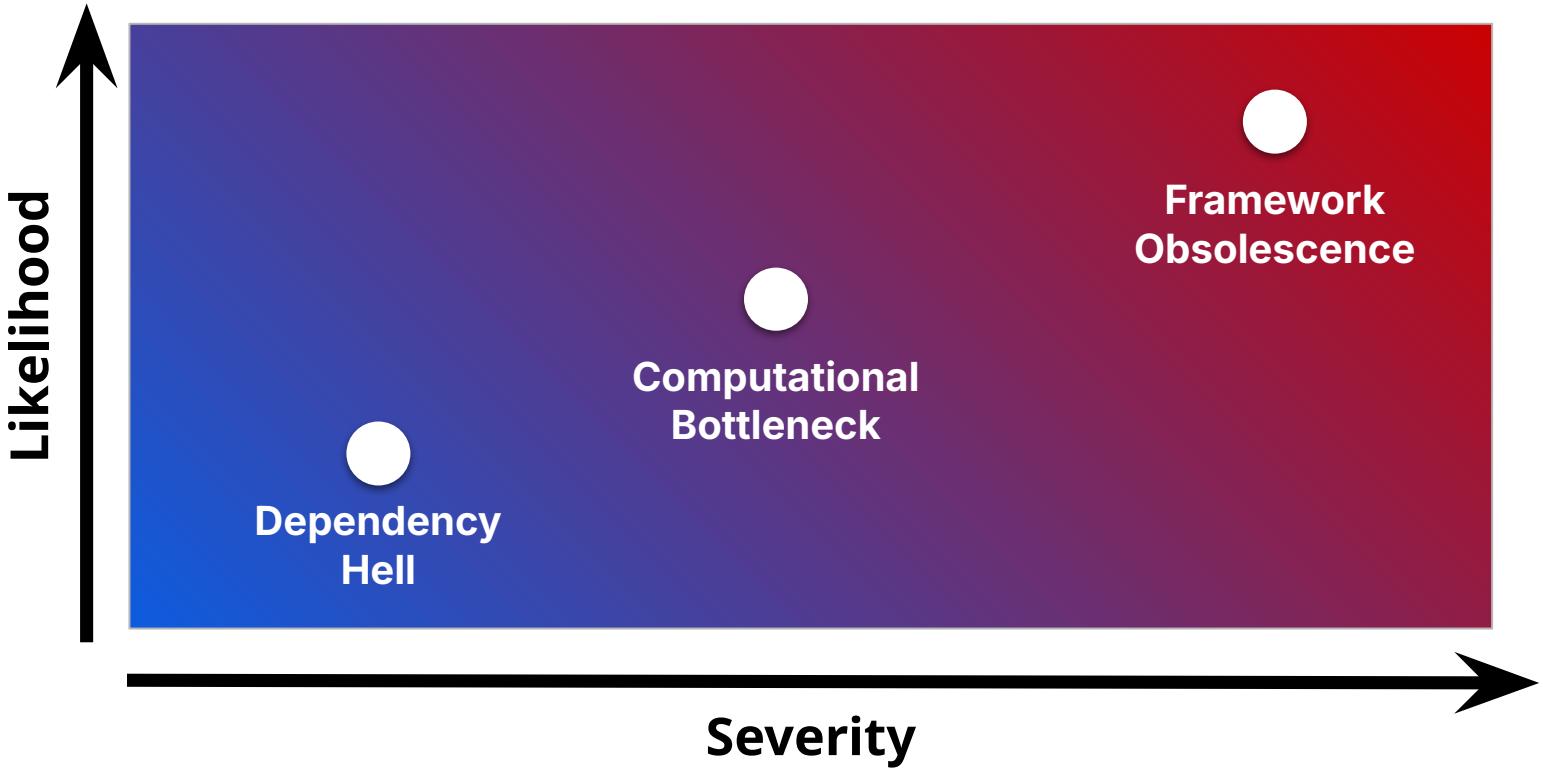
Training Pipeline Requirements

Functional	Performance	Environmental
Ingestion & Validation of Datasets	Configuration in **< 30 Minutes**	Uses **Python**
Framework Abstraction	In **< 10 Lines**	Training on **GPU**

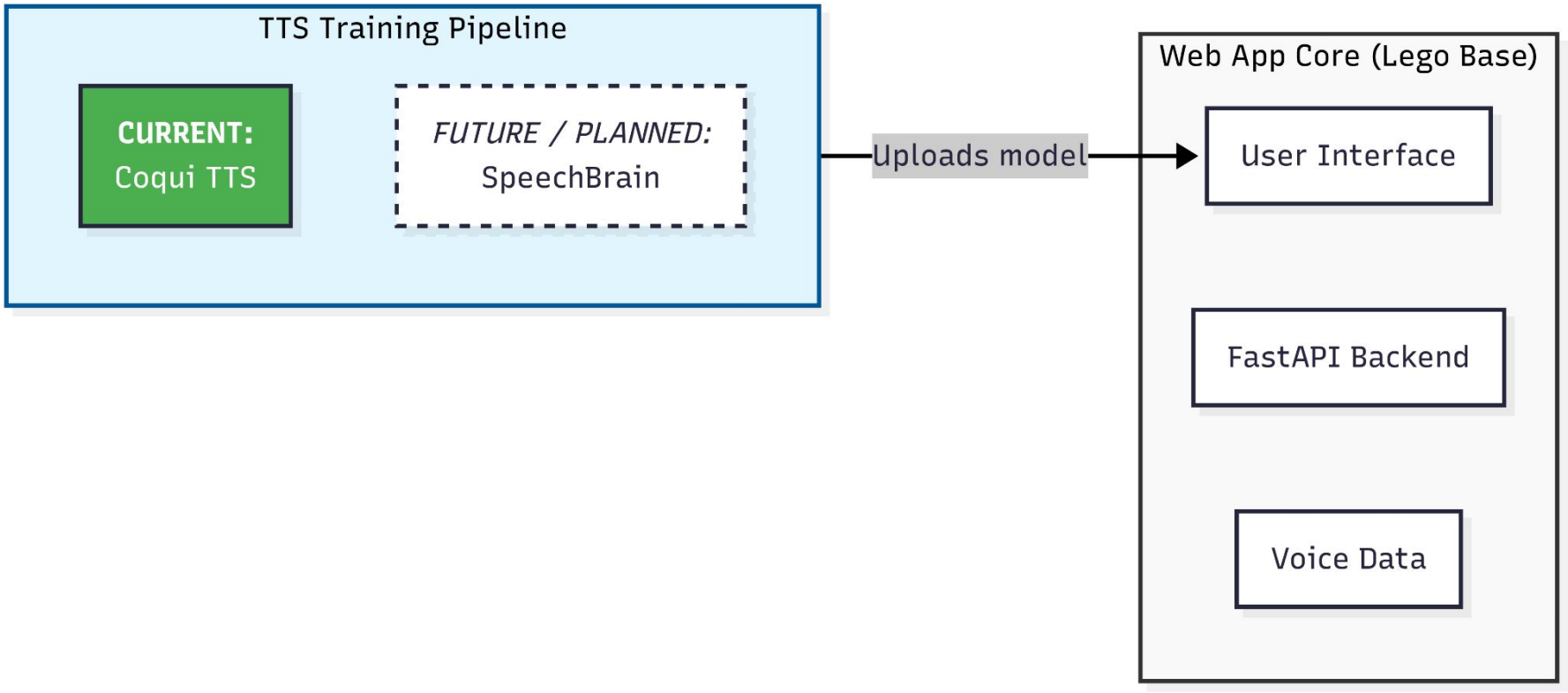
Training Pipeline Data-Flow



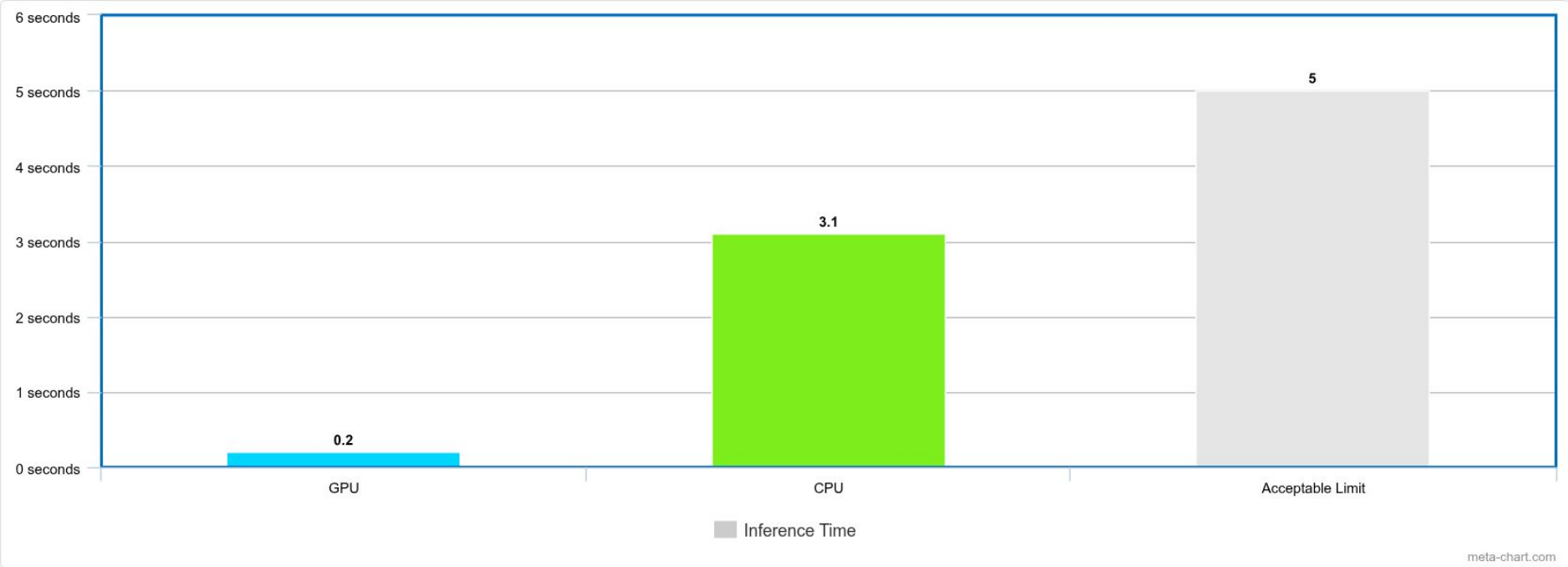
Risks


Likelihood
Framework Obsolescence
Computational Bottleneck
Dependency Hell
Severity

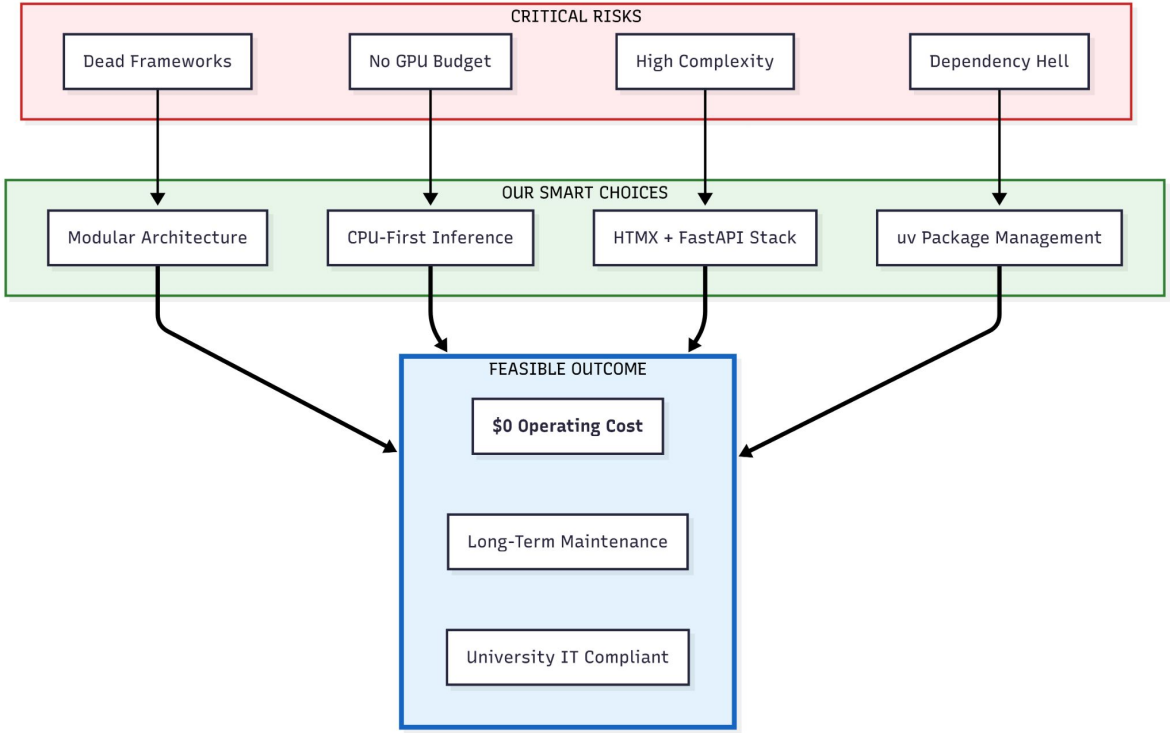
Feasibility (Risk Mitigation)



Feasibility (Risk Mitigation) (cont...)



Feasibility (Risk Mitigation) (cont...)



Schedule - Approach

- Milestone-based development
- Start with prototyping for early validation
- Follow with iterative development and testing
- Focus on reducing risks and staying aligned with goals

Schedule - Key Milestones

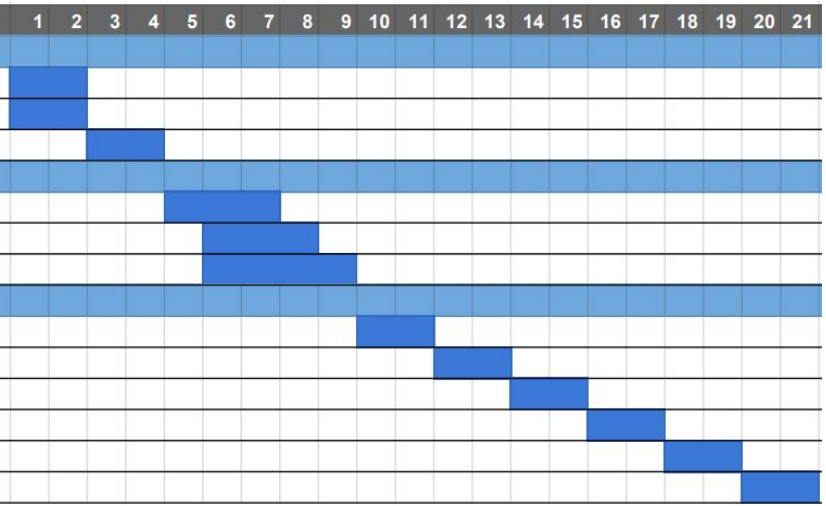
- **Requirements & Tech Selection**
  Define system needs and choose tools (Coqui TTS, PyTorch, FastAPI, HTMX)
- **Prototype Development**
  Test TTS performance on CPU and GPU
- **Backend Development**
  Build API for processing text → audio
- **Frontend Development**
  Create simple UI for input and playback
- **TTS Training Pipeline**
  Data processing + model training + transfer learning

Schedule - Key Milestones (cont.)

- **Performance Optimization**
  Improve speed and efficiency (PyTorch tuning)
- **Integration & Testing**
  Unit, integration, and usability testing
- **Deployment**
  Host system and manage environment (uv)
- **Final Validation**
  Testing, feedback, and refinements

Schedule - Timeline Structure

TASK	START	DAYS	END
Phase 1			
Requirements Analysis	2/2/2026	14	2/15/2026
Tech Selection	2/2/2026	14	2/15/2026
Research & Setup Environment	2/16/2026	14	3/1/2026
Phase 2			
Backend Development	3/2/2026	21	3/22/2026
Frontend Development	3/9/2026	21	3/29/2026
TTS Training Pipeline Implementation	3/9/2026	28	4/5/2026
Phase 3			
Prototype Development	4/6/2026	14	4/19/2026
Performance Optimization	4/20/2026	14	5/3/2026
System Integration	5/4/2026	14	5/17/2026
Testing Phase	5/18/2026	14	5/31/2026
Deployment & Environement Setup	6/1/2026	14	6/14/2026
Final Validation & Refinement	6/15/2026	14	6/28/2026



Conclusion

- SynthLang: Creating a TTS platform for underserviced languages
  - Easy to use web interface for generating speech from text
  - Customizable training pipeline, runnable on Dr. Tucker’s machine
- Avoid
  - Framework Obsolescence
  - Computational Bottlenecks
- Goals
  - Long term operation
  - Quick speech generation
  - High quality, trainable models