AI Model Evaluation Plan - STEMBlock.ai Benchmarking System

1. Overview

1.1 Purpose

Develop a proprietary AI model evaluation benchmark system for assessing language models' performance on STEM education and VEX robotics tasks. This system will serve as intellectual property (IP) for STEMBlock.ai and provide competitive differentiation.

1.2 Inspiration

Drawing from existing benchmarking frameworks: - **Cybench** (https://cybench.github.io/): Cybersecurity-focused LLM benchmarking - **CyberGym** (https://www.cybergym.io/): Interactive security training environments

1.3 Goals

1. Create comprehensive STEM/VEX robotics evaluation dataset
2. Benchmark multiple AI models (Gemini, GPT-4, Claude, etc.)
3. Establish baseline metrics for model selection
4. Continuously evaluate model performance
5. Publish findings to establish thought leadership

2. Benchmark Design

2.1 STEMBlock Benchmark Suite

2.1.1 Domain Coverage Programming Evaluation (30%): - Code correctness assessment - Algorithm efficiency analysis - VEX-specific code patterns (motor control, sensor usage) - Common coding errors detection - Optimization suggestions quality

Engineering Design Analysis (25%): - Robot design evaluation from images - Component identification accuracy - Mechanical design assessment - VEX competition rule compliance - Improvement suggestion relevance

Engineering Notebook Assessment (20%): - Documentation completeness - Design process clarity - Iteration and testing evidence - Reflection quality - Rubric-based scoring accuracy

Problem-Solving and Reasoning (15%): - STEM concept understanding - Step-by-step problem solving - Logical reasoning quality - Explanation clarity

Educational Feedback Quality (10%): - Constructive tone - Age-appropriate language - Specific and actionable suggestions - Encouragement and motivation - Error explanation clarity

2.2 Evaluation Metrics

2.2.1 Quantitative Metrics **Accuracy Metrics**: - **Correctness**: % of correct evaluations vs. human expert baseline - **Precision**: TP / (TP + FP) - avoiding false positives - **Recall**: TP / (TP + FN) - catching all issues - **F1 Score**: Harmonic mean of precision and recall - **Mean Absolute Error (MAE)**: For numerical scores

Performance Metrics: - **Response Time**: Time to generate evaluation (ms) - **Token Efficiency**: Tokens used per evaluation - **Cost per Evaluation**: Dollar cost per API call - **Throughput**: Evaluations per minute

Consistency Metrics: - **Inter-evaluation Consistency**: Same input → same output - **Intra-evaluation Consistency**: Similar inputs → similar outputs - **Score Variance**: Standard deviation of scores for equivalent work

2.2.2 Qualitative Metrics **Feedback Quality**: - **Specificity**: Vague (1) to Highly specific (5) - **Actionability**: Not actionable (1) to Clear actions (5) - **Tone**: Discouraging (1) to Encouraging (5) - **Relevance**: Off-topic (1) to Highly relevant (5) - **Depth**: Surface-level (1) to Deep analysis (5)

Educational Value: - **Learning Opportunity**: Creates teaching moments - **Concept Explanation**: Explains underlying concepts - **Growth Mindset**: Promotes learning and improvement - **Age Appropriateness**: Suitable for K-12 students

2.3 Dataset Construction

2.3.1 Dataset Categories **1. Programming Submissions Dataset** (500 samples): - Python code (VEX V5, VEX IQ) - C++ code (PROS, VEXcode) - VEXcode Blocks exports - Range: Beginner to Advanced - Categories: Drivetrain, Autonomous, Sensors, Competition code

2. Robot Design Images Dataset (300 samples): - Front, side, top views of robots - VEX IQ and V5 robots - Various competition games - Annotated with: - Component list - Design strengths - Design weaknesses - Competition readiness score (1-10)

3. Engineering Notebooks Dataset (200 samples): - PDF/DOCX formats - Range: Poor to Excellent (based on VEX rubric) - Annotated with: - Rubric scores (0-25 per category) - Missing sections - Strengths and weaknesses

4. Short Answer Questions Dataset (400 samples): - STEM concepts questions - VEX-specific questions - Problem-solving questions - Student responses (various quality levels) - Expert-graded with scores and feedback

5. Edge Cases and Adversarial Examples (100 samples): - Deliberately ambiguous submissions - Partial submissions - Multilingual code comments - Unusual approaches - Malformed inputs

2.3.2 Data Collection Strategy **Phase 1: Seed Dataset (Months 1-2)**: - Create 100 synthetic examples per category - Use team members as "students" - Generate varied quality samples - Expert annotation by coaches

Phase 2: Pilot Data (Months 3-4): - Collect from pilot program (100 students) - Obtain informed consent - Anonymize data - Expert annotation

Phase 3: Production Data (Months 5+): - Ongoing collection from live system - Continuous annotation (sample-based) - Expand edge cases - Version control dataset

2.3.3 Annotation Process **Human Expert Annotators**: - Experienced VEX coaches (3-5 people) - Provide ground truth evaluations - Use same rubrics as AI - Consensus on difficult cases

Annotation Format:

```
{
  "submission_id": "sub-001",
  "type": "code",
  "content": "...",
  "ground_truth": {
    "overall_score": 85,
    "correctness": 90,
    "efficiency": 80,
    "style": 85,
    "vex_best_practices": 80,
    "feedback": "Well-structured code with good sensor usage...",
    "strengths": ["Clear variable names", "Proper motor control"],
    "improvements": ["Add error handling", "Optimize autonomous routine"],
    "code_issues": [
      {
        "line": 42,
        "severity": "warning",
        "description": "Consider timeout for sensor wait",
        "suggestion": "while (sensor.value() < 100 && timeout < 1000)"
      }
    ]
  },
  "metadata": {
    "difficulty": "intermediate",
    "competition_type": "VEX_V5",
    "annotator": "coach-001",
    "annotation_date": "2025-10-18",
    "annotation_time_minutes": 12
  }
}
```

3. Evaluation Framework

3.1 Model Selection

Models to Evaluate: 1. **Google Gemini 1.5 Pro** (Primary) 2. **Google Gemini 1.5 Flash** (Cost-effective) 3. **Google Gemini 1.5 Pro Vision** (Image analysis) 4. **OpenAI GPT-4o** (Comparison) 5. **OpenAI GPT-4o-mini** (Cost-effective comparison) 6. **Anthropic Claude 3.5 Sonnet** (Comparison) 7. **Anthropic Claude 3 Haiku** (Cost-effective comparison)

3.2 Evaluation Pipeline

```
1. Load Test Dataset
    ↓
2. For each model:
    ↓
3. Generate predictions (with prompt template)
    ↓
4. Parse structured output
    ↓
5. Compare to ground truth
    ↓
6. Calculate metrics
    ↓
7. Generate report
```

Implementation (Python evaluation script):

```python
import json
from typing import List, Dict
from dataclasses import dataclass
import vertexai
from vertexai.generative_models import GenerativeModel

@dataclass
class EvaluationResult:
    model_name: str
    accuracy: float
    precision: float
    recall: float
    f1_score: float
    avg_response_time_ms: float
    cost_per_eval: float
    feedback_quality_score: float

class ModelEvaluator:
    def __init__(self, dataset: List[Dict], prompt_template: str):
        self.dataset = dataset
```

```
        self.prompt_template = prompt_template

    def evaluate_model(self, model_name: str) -> EvaluationResult:
        """Evaluate a specific model on the dataset."""
        predictions = []
        response_times = []
        costs = []

        for sample in self.dataset:
            # Generate prediction
            start_time = time.time()
            prediction = self.generate_prediction(model_name, sample)
            response_time = (time.time() - start_time) * 1000

            predictions.append(prediction)
            response_times.append(response_time)
            costs.append(self.calculate_cost(model_name, prediction))

        # Calculate metrics
        accuracy = self.calculate_accuracy(predictions, self.dataset)
        precision = self.calculate_precision(predictions, self.dataset)
        recall = self.calculate_recall(predictions, self.dataset)
        f1 = 2 * (precision * recall) / (precision + recall)
        feedback_quality = self.evaluate_feedback_quality(predictions)

        return EvaluationResult(
            model_name=model_name,
            accuracy=accuracy,
            precision=precision,
            recall=recall,
            f1_score=f1,
            avg_response_time_ms=np.mean(response_times),
            cost_per_eval=np.mean(costs),
            feedback_quality_score=feedback_quality
        )

    def generate_prediction(self, model_name: str, sample: Dict) -> Dict:
        """Generate prediction using specified model."""
        # Format prompt
        prompt = self.prompt_template.format(**sample)

        # Call model API
        if "gemini" in model_name:
            model = GenerativeModel(model_name)
            response = model.generate_content(prompt)
        elif "gpt" in model_name:
```

```
            # OpenAI API call
            pass
        elif "claude" in model_name:
            # Anthropic API call
            pass

        # Parse response
        return self.parse_response(response.text)

    def calculate_accuracy(self, predictions: List[Dict], ground_truth: List[Dict]) -> float
        """Calculate accuracy by comparing scores."""
        score_diffs = []
        for pred, truth in zip(predictions, ground_truth):
            score_diff = abs(pred['overall_score'] - truth['ground_truth']['overall_score'])
            score_diffs.append(score_diff)

        # Accuracy: % within 10 points of ground truth
        accuracy = sum(1 for diff in score_diffs if diff <= 10) / len(score_diffs)
        return accuracy

    def evaluate_feedback_quality(self, predictions: List[Dict]) -> float:
        """Evaluate feedback quality using another LLM as judge."""
        # Meta-evaluation: Use GPT-4 to score feedback quality
        quality_scores = []

        for pred in predictions:
            judge_prompt = f"""
            Evaluate the quality of this educational feedback on a scale of 1-5:

            Feedback: {pred['feedback']}

            Criteria:
            1. Specificity (vague to highly specific)
            2. Actionability (not actionable to clear actions)
            3. Tone (discouraging to encouraging)
            4. Relevance (off-topic to highly relevant)
            5. Depth (surface-level to deep analysis)

            Return JSON: {{"score": X, "reasoning": "..."}}
            """

            # Get judgment from meta-evaluator
            score = self.meta_evaluate(judge_prompt)
            quality_scores.append(score)

        return np.mean(quality_scores)
```

3.3 Prompt Engineering for Evaluation

Template Structure:

```
System Prompt:
- Role definition (VEX robotics expert, STEM educator)
- Evaluation criteria
- Output format specification
- Tone guidelines (constructive, encouraging)

User Prompt:
- Submission content
- Context (student level, competition type)
- Specific evaluation request

Examples (Few-shot):
- 2-3 examples of input → output
- Demonstrate desired format and quality
```

Example Prompt Template (Code Evaluation):

```
You are an expert VEX robotics coach evaluating student code.

EVALUATION CRITERIA:
1. Correctness (0-100): Does the code work as intended?
2. Efficiency (0-100): Is the code optimized?
3. Style (0-100): Is the code readable and well-structured?
4. VEX Best Practices (0-100): Does it follow VEX-specific guidelines?

SUBMISSION:
Language: {language}
Competition: {competition_type}
Student Level: {level}

CODE:
```{language}
{code_content}
```

INSTRUCTIONS: 1. Analyze the code thoroughly 2. Provide scores for each criterion 3. Give specific, constructive feedback 4. Identify strengths and areas for improvement 5. Suggest code improvements with examples

OUTPUT FORMAT (JSON): {{ “overall_score”: <0-100>, “scores”: {{ “correctness”: <0-100>, “efficiency”: <0-100>, “style”: <0-100>, “vex_best_practices”: <0-100> }}, “feedback”: “”, “strengths”: [“”, “”], “improvements”: [“”, “”], “code_issues”: [{{ “line”: , “severity”: “info|warning|error”, “description”: “”, “suggestion”: “” }}] }} “‘
```

3.4 Evaluation Schedule

Initial Evaluation (Month 1): - Baseline evaluation of all models - Select primary and backup models - Identify prompt improvements

Ongoing Evaluation (Monthly): - Re-evaluate on new data samples (100 samples) - Track model performance over time - Detect model drift - Evaluate new model versions

Comprehensive Re-evaluation (Quarterly): - Full dataset evaluation - Compare new models - Update model selection - Publish benchmark report

4. Benchmark Report Structure

4.1 Public Benchmark Report

STEMBlock AI Benchmark Report v1.0

Executive Summary: - Best performing model for each task - Key findings and recommendations - Model selection rationale

Methodology: - Dataset description - Evaluation metrics - Prompt templates used - Annotation process

Results:

Table: Overall Model Performance

Model	Accuracy	F1 Score	Avg Response Time	Cost per Eval	Feedback Quality
Gemini 1.5 Pro	87%	0.85	2,500ms	$0.05	4.2/5
GPT-4o	89%	0.87	3,200ms	$0.08	4.5/5
Claude 3.5 Sonnet	88%	0.86	2,800ms	$0.06	4.3/5
Gemini 1.5 Flash	82%	0.80	1,200ms	$0.02	3.8/5

Task-Specific Performance: - Programming evaluation results - Engineering design analysis results - Notebook assessment results - Problem-solving results

Cost-Performance Analysis: - Cost vs. accuracy tradeoff - Recommendations by use case

Limitations and Future Work: - Current limitations - Planned improvements - Upcoming model evaluations

4.2 Internal Evaluation Dashboard

Real-time Metrics: - Daily model performance - Cost tracking - Evaluation volume - Error rates

Alerts: - Performance degradation (accuracy drops >5%) - Cost spikes - High error rates

5. Implementation Plan

5.1 Timeline (Integrated with Main Project)

Month 1 (Weeks 1-4): - [ ] Define evaluation framework - [ ] Create initial prompt templates - [ ] Build evaluation script infrastructure - [ ] Create 100 synthetic samples per category - [ ] Recruit expert annotators (coaches)

Month 2 (Weeks 5-8): - [ ] Complete seed dataset annotation (500 samples) - [ ] Run baseline evaluation on all models - [ ] Analyze results and select primary model - [ ] Refine prompts based on results - [ ] Begin integration with evaluation service

Month 3 (Weeks 9-12) - MVP: - [ ] Collect pilot program data - [ ] Expand dataset to 1,000 samples - [ ] Run monthly evaluation - [ ] Deploy selected model to production - [ ] Create internal dashboard

Months 4-6 - Full Release: - [ ] Continuous data collection - [ ] Quarterly comprehensive evaluation - [ ] Publish first benchmark report - [ ] Expand to new model versions - [ ] Refine evaluation metrics

5.2 Team Responsibilities

Developer 1 (20% time): - Build evaluation pipeline - Integrate model APIs - Create reporting dashboard - Automate evaluation runs

Developer 2 (10% time): - Support data collection - Build annotation tools - Create data export scripts

Intern (50% time): - Create synthetic samples - Assist with annotation - Run evaluation experiments - Compile benchmark reports - Research new models

UX Designer (5% time): - Design annotation interface - Design benchmark report visualizations

5.3 Tools and Infrastructure

Evaluation Framework: - Python scripts (evaluation logic) - Jupyter notebooks (analysis) - MLflow or Weights & Biases (experiment tracking)

Model APIs: - Google Vertex AI SDK - OpenAI Python SDK - Anthropic Python SDK

Data Management: - PostgreSQL (dataset storage) - Cloud Storage (raw submissions) - Git LFS (dataset version control)

Annotation Tool: - Simple web UI (Next.js) - Expert annotator access - Annotation queue management - Inter-annotator agreement tracking

6. Intellectual Property Strategy

6.1 Proprietary Assets

Dataset: - STEMBlock VEX Robotics Dataset - Largest annotated VEX evaluation dataset - Continuous expansion with production data

Benchmark Framework: - Evaluation methodology - Metrics and rubrics - Prompt engineering templates

Research Findings: - Model performance comparisons - Best practices for STEM education AI - Prompt optimization techniques

6.2 Publication Strategy

Public Benchmark (Open Source): - Evaluation methodology - Sample dataset (100 anonymized examples) - Benchmark results and leaderboard - Attract attention and establish authority

Proprietary Components (Closed Source): - Full dataset (1,000+ samples) - Detailed prompt templates - Annotation guidelines - Production evaluation pipeline

Academic Publications (Optional): - Submit to education technology conferences - Publish research papers - Build academic partnerships

6.3 Competitive Advantage

Differentiation: - Only VEX robotics-specific AI benchmark - Continuous evaluation with production data - Proven model selection methodology - Thought leadership in STEM education AI

Business Value: - Informed model selection (cost savings) - Higher evaluation quality (better user experience) - Marketing and PR opportunities - Partnership opportunities with AI vendors - Potential licensing revenue (future)

7. Success Metrics

7.1 Technical Metrics

- ☐ Dataset: 1,000+ annotated samples by Month 6
- ☐ Inter-annotator agreement: Cohen's kappa > 0.7

☐ Evaluation accuracy: >85% within 10 points of human
☐ Benchmark report published: Month 6
☐ Model selection validated: Production metrics match benchmark

7.2 Business Metrics

☐ Cost reduction: 20% savings through optimal model selection
☐ User satisfaction: NPS > 50 for AI feedback quality
☐ PR mentions: 5+ articles about benchmark
☐ Academic citations: 3+ papers cite STEMBlock benchmark
☐ Partnership inquiries: 2+ AI vendors interested in collaboration

8. Risks and Mitigation

8.1 Risks

Risk	Likelihood	Impact	Mitigation
Insufficient dataset size	Medium	High	Start with synthetic data, collect from pilot
Annotation quality issues	Medium	High	Clear guidelines, inter-annotator agreement checks
Model performance insufficient	Low	High	Evaluate multiple models, have backup options
High evaluation costs	Medium	Medium	Use cost-effective models where appropriate
Time constraints	High	Medium	Prioritize core evaluation, defer advanced features

8.2 Contingency Plans

If primary model underperforms: - Switch to best-performing alternative - Hybrid approach (use multiple models) - Adjust evaluation criteria

If dataset too small: - Prioritize high-quality subset - Use data augmentation techniques - Defer comprehensive benchmark

If timeline slips: - MVP with basic evaluation only - Publish preliminary benchmark report - Expand in later phases

---

Document Version: 1.0 **Last Updated**: 2025-10-18 **Owner**: Development Team (Intern lead, Dev 1 support) **Timeline**: Months 1-6 (integrated with main project)