Domain: AI/ML

Beginners:

1) Alloy Property Prediction

Objective: Build a machine learning system that predicts metal alloy properties to aid engineering students and support local manufacturing with material selection.

Requirements:

- Use public datasets from Materials Project, Citrination, OQMD, AFLOWlib, or Kaggle
- Process data with pandas, numpy, and scikit-learn (handling missing values, normalization, encoding)
- Implement models using scikit-learn, XGBoost, LightGBM or CatBoost (Random Forest, Gradient Boosting, SVM, Neural Networks)
- Evaluate using MAE, RMSE, and R^2 Score with visualization tools
- Create user interface with Streamlit, Gradio, or Flask

Constraints:

- No paid services or proprietary APIs
- Use only public datasets with appropriate licenses
- Design for efficient operation on limited computational resources
- Create accessible interface for non-technical users
- Ensure reproducibility with shared code and documentation
- Maximum team size: 4 members

2) Automatic Meeting Notes Generator

Objective: Create a system that transforms meeting audio recordings into structured, accurate meeting notes by transcribing speech and summarizing key points.

Requirements:

- Transcribe conversational speech from meetings, handling overlapping dialogue and background noise
- Generate concise notes capturing decisions, action items, and key points
- Bonus: Identify and tag different speakers (P1, P2, etc.) without prior voice samples
- Provide export options (.txt, .pdf) and optional UI for uploading audio and downloading notes

Constraints:

- System should work on pre-recorded audio (up to 30 minutes)
- Speaker identification without prior registration
- Structure summaries clearly (bullet points, task lists)

Deliverables:

- Functional pipeline from audio to meeting notes
- Transcripts with optional speaker tags
- Report on architecture, models, challenges, and accuracy
- Bonus: UI and adjustable speaker diarization settings

Team Size: Maximum 4 members

3) Drone Footage Object Detection

Objective: Develop a computer vision system for detecting and classifying objects in drone footage to support environmental monitoring, urban planning, wildlife conservation, and disaster management.

Requirements:

- Use public drone imagery datasets (OpenAerialMap, DroneDeploy, UAVDT, VisDrone)
- Preprocess data with OpenCV, NumPy, pandas (frame extraction, resizing, normalization)
- Implement detection models using PyTorch or TensorFlow (YOLOv5/v8, SSD, Faster R-CNN, EfficientDet)
- Evaluate with metrics like mAP, IoU, Precision, Recall, F1 Score
- Create dashboard for uploading footage and displaying detection results

Constraints:

- Use only free, open-source frameworks (no commercial APIs)
- Respect data privacy with properly licensed datasets
- Optimize for limited computational resources
- Design accessible interface for researchers and community users
- Ensure reproducibility with proper documentation
- Maximum team size: 4 members

4) Script and Slideshow Generator

Objective: Create an application that generates structured scripts and slideshow presentations from simple user inputs to support student organizations in creating awareness campaigns and educational content.

Requirements:

- Accept inputs for topic, audience, tone, and presentation length
- Generate content using open-source NLP models (Hugging Face Transformers, T5, GPT-J, LLaMA 2)
- Create slides with python-pptx, reveal.js, or Deck.js, including titles and bullet points
- Allow content review and editing before export
- Support export as PowerPoint, PDF, or HTML

Constraints:

- No proprietary APIs or paid services
- Use only royalty-free content and proper attribution
- Optimize for local machines or free cloud services
- Design for non-technical users with minimal setup
- Provide clear documentation for reproducibility
- Maximum team size: 4 members

5) Daytime to Dusk Image Transformation Tool

Objective: Build an image transformation tool that converts daytime photographs into realistic dusk/sunset scenes to help photography students explore lighting and assist tourism by showcasing destinations at appealing hours.

Requirements:

- Allow image uploads with optional timestamp selection
- Transform images using techniques like CycleGAN or Pix2Pix with PyTorch/TensorFlow
- Use open-access datasets or properly licensed images
- Train on local machine or Google Colab free tier
- Create user-friendly interface with before/after comparison

Constraints:

- No commercial APIs or paid services
- Use only properly licensed datasets and images
- Design for operation with limited computational resources
- Create intuitive interface for non-technical users
- Ensure reproducibility with shared code and documentation
- Maximum team size: 4 members

Intermediate:

6) Gen-AI Video Editor – Intelligent Video Enhancement and Modification Platform

Objective: Build a Generative AI-powered video editing platform that enables users to upload videos and apply intelligent modifications, enhancements, or creations using state-of-the-art video generation models. The platform should support interactive and automated tools for creative and structural video editing tasks that go beyond traditional filters and effects.

Core Requirements:

- Video Upload & Interface:
  - Create a user-friendly platform (web-based or desktop) where users can:
  - Upload videos for editing.
  - Select and apply GenAI-based tools.
  - Preview and export edited videos.
- Generative AI Video Editing Tools:
  - You must integrate at least two GenAI-powered tools for video manipulation. These tools can be selected based on team preference. Some suggested tools include:
  - Object Removal in Video: Automatically remove selected objects and inpaint the background temporally across frames.
  - Scene Cut Harmonizer: Blend harsh scene transitions to maintain color and motion continuity.
  - Style Transfer: Apply a consistent artistic or cinematic style to the entire video.
  - Motion-Aware Upscaling: Enhance video resolution using AI models that preserve motion consistency.
- Use of Video Generation Models:
  - You must use at least one generative video model to power core features.

Bonus Objectives:

- Allow prompt-based video transformations (e.g., "make this scene look like winter" or "add rain effect").

Final Deliverables:

- A functioning video editing platform with upload support and working GenAI tools.
- At least two generative AI-based editing features fully implemented.
- Sample videos demonstrating before-and-after outputs.
- Technical documentation explaining architecture, models used, and pipeline design.

Team Size: 4-6 members

Recommended Skillsets: Video generation and editing (e.g., Runway, ModelScope, Stable Diffusion), frontend/backend development, computer vision, UI/UX design, temporal consistency algorithms.

7) RL Trading Strategist – Reinforcement Learning-Based Algorithmic Trading in Cryptocurrency Markets

c Design a Reinforcement Learning (RL)-based trading platform to develop, optimize, and deploy intelligent trading strategies for the BTC/USDT and ETH/USDT cryptocurrency markets. The RL agent must learn to maximize cumulative risk-adjusted returns while effectively managing market volatility and minimizing drawdowns.

Core Requirements:

1. Data Acquisition & Preprocessing:
    - Utilize historical OHLCV (Open, High, Low, Close, Volume) data for BTC/USDT and ETH/USDT from January 1, 2020 to December 31, 2023.
    - Clean and structure the data into suitable state representations for RL agents.
    - Normalize features, handle missing data, and construct appropriate observation and reward structures.
2. Reinforcement Learning Strategy Design:
    - Build two distinct RL trading agents, one each for BTC/USDT and ETH/USDT.
    - Define clear state, action, and reward functions tailored to trading.
    - Use suitable RL algorithms (e.g., DQN, PPO, A2C, SAC) for discrete or continuous action spaces.

   - Strategies should be explainable, adaptable, and capable of generalizing to unseen data.
3. Environment and Backtesting:
   - Simulate the trading environment using a custom or provided SDK with:
   - Realistic trade execution modeling
   - 0.15% transaction cost per trade
   - Slippage and liquidity constraints
   - Ensure episodic training is aligned with market session logic.
4. Risk-Aware Learning:
   - Incorporate risk management into the RL framework through:
   - Penalizing drawdowns in the reward signal
   - Learning stop-loss and position-sizing policies
   - Encouraging portfolio diversification or volatility targeting
5. Hyperparameter Tuning and Optimization:
   - Perform systematic tuning of RL agent parameters (e.g., learning rate, γ, ε).
   - Ensure the agent avoids overfitting by applying techniques like:
   - Early stopping
   - Reward regularization
   - Dropout/Noise in state representation

Bonus Objectives:

- Implement interpretable RL techniques (e.g., policy visualization, attention maps).
- Include prompt-based environment changes (e.g., "simulate high-volatility market").
- Demonstrate transfer learning across assets or market regimes.

Final Deliverables:

- RL Strategy Code: Jupyter Notebooks with documented RL pipeline and agent logic.
- Backtest Reports: Performance metrics (Sharpe, Sortino, max drawdown) and visualizations (equity curves, trade logs).
- Risk Strategy Documentation: Description of how risk constraints are handled within the RL framework.
- Technical Report: Complete development pipeline, reward design rationale, and observed learning behaviors.
- Presentation Deck: Strategy overview, learning insights, and performance summary for BTC and ETH agents.

Team Size: 4-6 members

8) Accelerate-WAN – Optimizing Inference and Training Speed of the Wan 1.3B Video Model

Objective: Design and implement a high-performance optimization pipeline to significantly accelerate the Wan 1.3B video generation model without compromising its output quality. Participants must explore and integrate cutting-edge performance-boosting techniques to improve model throughput and reduce latency in real-world deployment settings.

Core Requirements:

1. Baseline Evaluation:
   - Begin by profiling the original Wan 1.3B model on standard video input-output pipelines.
   - Measure key metrics including:
   - Inference latency (per frame and per clip)
   - GPU memory utilization
   - Throughput (clips per second)
   - Load time and batch processing efficiency
2. Optimization Techniques:
   - Apply and benchmark at least three of the following acceleration strategies:
   - FlashAttention or FlashAttention-2: Replace standard attention with memory-efficient attention mechanisms to speed up transformer blocks.
   - CFG Parallelism: Optimize Classifier-Free Guidance sampling by parallelizing unconditional and conditional branches.
   - Quantization: Apply 8-bit or 4-bit quantization (e.g., INT8, FP8, QLoRA) to reduce model size and improve execution speed.
   - Operator Fusion and TorchScript Optimization: Use tools like torch.compile() or ONNX graph optimization to merge operations and reduce runtime overhead.
   - Multi-GPU or TPU Parallelism: Implement model/data parallelism across multiple accelerators.
   - Model Pruning or Low-Rank Adaptation (LoRA): Explore structured pruning or fine-tuned low-rank adapters to reduce computational burden.
   - Memory-Efficient Attention Variants: Use Linformer, Performer, or other approximations for scalable sequence handling.
   - Offloading Techniques: Employ CPU-GPU hybrid memory techniques or IO-efficient data loaders.
3. Benchmarking and Validation:

    - Re-profile the optimized model using the same metrics as baseline.
    - Evaluate:
    - Speedup factor over baseline
    - Memory savings
    - Fidelity retention: Ensure output quality (SSIM, PSNR, FID if applicable) remains within acceptable margins compared to baseline.

Bonus Objectives:

- Build a dashboard or CLI tool to toggle and compare optimization modes.
- Show compatibility with low-resource edge devices (e.g., Jetson, RTX 3050).
- Enable real-time inference streaming (e.g., webcam-to-video generation).

Final Deliverables:

- Optimized Model Codebase: Scripts for model inference and training with configurable acceleration toggles.
- Performance Report: Before/after benchmarks, technique-wise contributions, and profiling plots.
- Quality Comparison: Demonstrations of video outputs with baseline vs optimized side-by-side.
- Technical Writeup: Detailing implementation process, tools used, trade-offs made, and recommendations.
- Demo Video (Optional): Short demonstration of runtime performance improvement.

Team Size: 4-6 members

9) PolyOCR – Building a Robust Multilingual Optical Character Recognition System

Objective: Develop a multilingual Optical Character Recognition (OCR) system capable of accurately extracting text from images across multiple languages and scripts. The system should handle printed and handwritten text, varying fonts, and diverse image conditions (noise, skew, blur).

Core Requirements:

- Support at least 5 languages including both Latin (e.g., English, Spanish) and non-Latin scripts (e.g., Hindi, Chinese, Arabic).
- Handle noisy, low-resolution, or complex backgrounds.

- Provide bounding box localization and recognized text for each detected region.
- Export results in structured format (e.g., JSON with language tags and coordinates).

Bonus Objectives:

- Implement automatic language detection for each region.
- Enable real-time OCR through a lightweight mobile/web interface.
- Add handwriting support for at least one language.

Deliverables:

- Working multilingual OCR model/codebase.
- Sample input-output visualizations.
- Evaluation on multilingual OCR benchmarks (e.g., IIIT-5K, ICDAR, MLT).
- Short report detailing model architecture, datasets used, and performance.

Team Size: 4-6 members

10) SpeakPortrait – Generating Talking Head Videos from a Single Image with Audio Synchronization

Objective: Build a generative AI system that takes a single image of a human face as input and synthesizes a realistic talking head video driven by a given audio clip or text prompt. The generated video should exhibit natural lip synchronization, facial expressions, and head movements, creating a lifelike talking avatar.

Core Requirements:

- Accept a single still image of a human face.
- Generate a video of the person speaking using:
  - Either a pre-recorded audio clip, or
  - A text input (text-to-speech synthesis integrated).
- Ensure accurate lip-sync, natural head poses, and expressive facial animation.
- Output video should be of at least 720p resolution and support real-time or near-real-time inference.

Bonus Objectives:

- Support emotional expression control (e.g., "speak happily", "speak angrily").
- Enable eye-gaze redirection or background replacement.
- Allow voice cloning from a short sample to personalize speech.

Final Deliverables:

- Fully working pipeline/codebase for image-to-video generation.
- Sample outputs demonstrating talking avatars from diverse images and audio inputs.
- Documentation covering architecture, training/inference pipeline, and model details.
- Optional: Web demo or CLI tool for user interaction.

Team Size: 4-6 members

ADVANCED:

11) IITI-Bot – Pathway-Powered Conversational Agent for IIT Indore

Objective: Design and implement an intelligent chatbot using Pathway's Agentic Retrieval-Augmented Generation (RAG) system that can accurately and autonomously answer any query related to IIT Indore. The system must leverage official institutional data and dynamically reason across documents to deliver trustworthy and context-rich responses.

Core Requirements:

1. **Dataset:**
    - Curate and utilize a comprehensive knowledge base related to IIT Indore, including but not limited to:
    - Academic programs and curriculum
    - Hostel, mess, library, and campus services
    - Clubs, events, and student bodies
    - Admissions, placements, and fee structures
    - Faculty, departments, and research centers
2. **Mandatory Use of Pathway:**
    - Build the RAG pipeline using the Pathway framework only (https://github.com/pathwaycom/pathway).
    - Utilize Pathway's DocumentStore or VectorStore for indexing and dynamic retrieval.
    - Leverage Pathway's real-time streaming and decision-making capabilities to handle user queries efficiently.
3. **Agentic RAG Design:**
    - Implement intelligent agents capable of:
    - Multi-hop reasoning and document synthesis
    - Adaptive retrieval based on query complexity
    - Fallback mechanisms in case of failure (e.g., API unavailability)
4. **Answer Accuracy and Context Awareness:**
    - Ensure answers are well-grounded in the provided IIT Indore data.
    - Avoid hallucinations and ensure transparency (e.g., return source document snippets).

Bonus Objectives:

- **Add voice or multilingual query support (e.g., English + Hindi).**
- **Provide live chat integration or deploy as a mobile/web assistant.**
- **Implement error recovery mechanisms (e.g., switching search sources or rerouting queries).**

Final Deliverables:

- **Fully functioning chatbot capable of answering a wide range of IIT Indore-specific questions.**
- **All source code in a public GitHub repository, with:**
- **README + setup instructions**
- **Demo video of chatbot handling real queries**
- **Midterm + Final reports detailing system architecture, approach, limitations, and results**

Team Size: 4-6 members

12) Generative Alchemy – AI-Powered Consistent Item Combination Game

Objective: Design and develop a generative AI-based version of the popular Little Alchemy game, where players can combine different elements to create new items. Unlike traditional implementations with a fixed combination logic, your task is to build a game in which:

1. **Combination Logic is Generative:**
    - The decision of whether two items can be combined, and what the resulting item is, should be made using generative AI (e.g., LLMs or other generative models).
2. **Symbol Generation:**
    - Each newly generated item must be visually represented by a unique symbol or icon. This symbol should also be generated using an image-generation model.
3. **Global Consistency:**
    - Once a combination (e.g., "water" + "fire") results in a new item (e.g., "steam") with a particular name and symbol, this result must remain consistent across all users and future sessions. The system must ensure that the same input pair always maps to the same output globally.
4. **Replayable and Scalable:**

- The game must support an expanding universe of items, allowing players to explore and discover hundreds of combinations. Ensure mechanisms to prevent infinite loops or meaningless combinations.

Final Deliverables:

1. **A fully playable game interface.**
2. **Documentation detailing architecture, models used, and consistency mechanisms.**

Team Size: 4-6 members

Skillsets Recommended: Prompt engineering, generative AI (LLM/image models), backend development, frontend/game development, data persistence & storage.

13) Generative Cricket Commentary – AI-Based Real-Time Sports Narration

Objective: Design and implement an end-to-end pipeline that automatically generates human-like cricket commentary for given video clips. The system must not only generate accurate and contextually rich commentary text but also convert it to speech and synchronize it with the corresponding video clip.

Key Requirements:

1. **Video Understanding:**
    - Analyze short to medium-length clips of cricket footage to understand the on-field events (e.g., sixes, wickets, dot balls, crowd reactions).
2. **Commentary Generation:**
    - Generate human-like, engaging commentary text using a generative language model. Commentary should be:
    - Relevant to the visual content.
    - Emotionally expressive and cricket-aware (e.g., using cricket terminology).
    - Stylistically similar to professional commentators.
3. **Text-to-Speech (TTS):**
    - Convert the generated text into realistic audio using TTS models. Voice should sound natural and match the pacing of typical live commentary.
4. **Audio-Video Syncing:**
    - Sync the generated audio commentary with the video clip. Since syncing becomes more difficult with longer clips, clip length and quality of synchronization will influence evaluation. Commentary should ideally

match the timing of key visual moments (like a bat swing, ball crossing boundary, or wicket fall).

Final Deliverables:

1. **A functioning pipeline that takes a cricket video clip and outputs a version with audio commentary synced.**
2. **Generated text commentary transcripts for each clip.**
3. **A short report describing the architecture, model choices, syncing strategy, and challenges.**

Bonus:

- **UI to allow users to upload clips and hear generated commentary.**
- **Syncing strategies should be adaptive to varying clip lengths and content complexity.**

Team Size: 4-6 members

Recommended Skillsets: Computer vision, NLP, generative models, TTS, video/audio processing.

14) Resource-Efficient Image Generation and Manipulation System

Objective: Design and implement an innovative image generation and manipulation system that achieves high-quality results within the strict hardware constraint of a 16GB GPU. The framework should focus on novel optimization techniques, model compression, and efficient architectures to enable advanced image synthesis and editing capabilities that would typically require more powerful hardware.

Core Requirements:

1. **Memory-Efficient Architecture:**
   - Design a lightweight generative model architecture that operates effectively within 16GB VRAM constraints.
   - Implement gradient checkpointing, model sharding, and mixed precision training/inference techniques.
   - Create efficient attention mechanisms that reduce memory requirements while preserving quality.

    - Develop novel parameter-sharing approaches across model components.
2. **Model Compression and Optimization:**
    - Implement advanced quantization techniques (8-bit, 4-bit) with minimal quality loss.
    - Design and apply knowledge distillation methods to create smaller models that retain capabilities of larger ones.
    - Create specialized pruning strategies for diffusion or GAN architectures.
    - Implement efficient caching strategies for intermediate computations.
3. **Progressive Generation Pipeline:**
    - Build a multi-stage generation system that progressively refines images to maximize quality under memory constraints.
    - Implement adaptive resolution control that focuses computational resources on relevant image regions.
    - Create a tiling mechanism that enables generation of high-resolution images through patch-by-patch synthesis.
    - Design an effective image composition strategy to seamlessly merge independently generated regions.
4. **Specialized Editing Capabilities:**
    - Implement memory-optimized versions of common editing operations (inpainting, outpainting, style transfer).
    - Create efficient region-specific editing mechanisms that avoid regenerating the entire image.
    - Design textual and visual conditioning methods that work with limited model capacity.
    - Develop techniques for maintaining editing history without excessive memory usage.

Bonus Objectives:

- **Create a benchmark suite for measuring efficiency-quality tradeoffs in constrained environments.**
- **Implement an intelligent offloading system that temporarily moves unused model components to CPU RAM.**
- **Design a user interface that provides real-time feedback on memory usage and potential optimizations.**
- **Develop a distributed computation approach that could leverage multiple 16GB GPUs in a networked environment.**

Final Deliverables:

- **Complete codebase with optimized model implementations and editing tools.**

- Technical report documenting memory optimization strategies and their effectiveness.
- Comparative analysis showing quality-vs-resource tradeoffs against larger models.
- Demonstration of at least 3 complex image generation and manipulation tasks working within memory constraints.
- Installation and usage documentation suitable for students with similar hardware limitations.

Team Size: 4-6 members

Recommended Skillsets: Model optimization, deep learning, computer vision, GPU memory management, quantization techniques, efficient attention mechanisms, and model compression strategies.

15) Training-Free 6D Pose Estimation from CAD Models

Objective: Design and implement a deep learning pipeline capable of estimating the full 6D pose (3D rotation and 3D translation) of known objects from an image with the help of their given CAD models.

Background: 6D pose estimation, which involves determining the 3D position and orientation of an object, is a critical task in robotics, autonomous systems, AR/VR, and industrial automation. Estimating the 6D pose of objects unseen during training is highly desirable yet challenging. So the aim of this project is to do a training-free 6D pose estimation.

Description: The main objective is to recreate the FREEZE paper. Go through the paper thoroughly before starting the implementation.

Part 1: Preprocessing and Feature Extraction

1. Model Selection
    - Review the FREEZE paper to shortlist suitable geometric and visual feature extraction models.
    - Evaluate and benchmark models such as DINO, CNOS, FCGF, Pointr, and others mentioned in the paper.
2. CAD Model Rendering (Multi-view Image Generation)
    - Generate synthetic views of the CAD models from multiple camera angles.

- Store RGB images and their corresponding pose annotations.
- Use the Help of CNOS for doing so.

3. Object Segmentation in Scene Images
   - Use classical or learning-based segmentation models to isolate objects in the RGB scene images.
   - Match the segmented regions with rendered CAD model views using similarity metrics.
4. Point Cloud Generation
   - Create dense or sparse point clouds from segmented RGB images (with or without depth).
   - Convert synthetic CAD views into point clouds.
5. Feature Extraction
   - Extract geometric features from the CAD and scene point clouds (e.g., using FCGF, Pointr or other geometric descriptors).
   - Extract visual features using contrastive learning-based image encoders (e.g., DINO, ResNet-based encoder).
6. Dual Point Cloud Formation
   - Construct two separate but aligned point clouds:
     - One enriched with visual features.
     - One enriched with geometric features.
   - These will be used in pose estimation.

Part 2: 6D Pose Estimation

1. Descriptor Matching and Alignment
   - Perform descriptor matching between the visual/geometric features of the scene and the CAD model point clouds.
   - Use algorithms like RANSAC to find the best rigid transformation between matched points.
   - RANSAC: It operates by sampling triplets of points from PQ (Point cloud from CAD model), and searching for their corresponding points in PT (Point cloud of the target object) by performing a nearest neighbor search within the fused feature space.
2. BONUS: Symmetry-Aware and Selection Matching (If Required)
   - Use the Iterative Closest Point (ICP) algorithm to refine Tc at point level and obtain a finer transformation Tf (T→ Translation).
   - Now for symmetry selection adjust the inaccurate initial pose by comparing the input crop image with the rendered images of some object symmetries.
   - NOTE: Our aim for symmetry-Aware is to adjust the pose for those objects for which the estimated pose would be correct without

considering the texture, i.e. when PQ and PT match, but the texture does not.

3. Pose Validation & Scoring
    - Validate the estimated pose using distance metrics (e.g., ADD, ADD-S).
    - Score pose quality by projecting CAD model back onto the image and measuring alignment.

Evaluation and Benchmarking

1. Dataset Evaluation
    - Evaluate the complete pipeline on datasets from the BOP Benchmark (e.g., LM, YCB-V, IC-BIN, HB).

Resources

1. Freeze Paper
2. IPD Dataset
3. For BOP dataset go through this→ link
4. 6DOF estimation from 2d video of known object categories but unknown instance→ link
5. Some Knowledge reference→ link
6. A beginner friendly medium article on how FoundationPose works→ link
7. Some githubs → link1, link2, link3
8. You will find a bunch of different models from the paper itself too.

Deliverables

- A working 6D pose estimation pipeline that requires no training for new objects.
- Documented codebase with modular design (CAD processing, segmentation, feature extraction, matching, pose estimation).
- Evaluation results and performance metrics across the BOP dataset.
- Optionally: A demo showing real-time inference from webcam or recorded RGB images.

Selection Criteria: For selecting a team for this problem statement, the team has to do the following:

- A proposal according to the IITISOC format
- A basic pipeline of running FASTSAM inference to segment objects in an image and Given an image create a pipeline to display its depth image (RGBD format) and its Point cloud (doesn't need to be very accurate).

Team Size: 4-6 members