KIDS PROJECT IDEAS

AI Project Ideas Class 12 — 50 Practical, Student-Friendly Projects

November 11, 2025

Artificial Intelligence (AI) is no longer a topic only for university courses or industry labs. For Class 12 students, doing AI projects builds practical problem-solving skills, strengthens

understanding of algorithms, and makes your school portfolio stand out for college applications.

This article lists **50 AI project ideas class 12** students can build with basic programming knowledge, simple mathematics, and a willingness to learn.

Each project idea below is written for students: clear description, the core AI techniques involved, suggested tools and libraries, sample datasets or data sources, short step-by-step implementation guidance, expected features, difficulty level (Beginner / Intermediate / Advanced), and what you'll learn. Use these ideas as-is or modify them to match your interests (health, environment, games, education, etc.).

Before you start any project, follow these short practical tips:

- Pick a topic you find interesting — motivation helps finish projects.
- Start with clear objectives and simple success criteria (what does "working" look like?).
- Use available datasets (Kaggle, UCI, government/open-data portals) or collect small datasets yourself.
- Document everything: problem statement, data sources, preprocessing steps, model selection, evaluation, and demo screenshots.
- Keep the UI simple: a basic web app (Flask/Streamlit) or notebook demo is enough for a school submission.

MUST READ: 97+ Best English Project Class 12 Topics For Students in 2025-2026

Table of Contents

How to use this list

- If you are new to AI, choose projects marked **Beginner** or **Beginner–Intermediate**.
- If you're comfortable with Python, try projects that use libraries like scikit-learn, TensorFlow/Keras, PyTorch, OpenCV, and NLTK/spaCy.
- For implementation, Jupyter notebooks and Google Colab are friendly: they let you run code without installing heavy software.

- Each idea is designed to be completed on a typical student laptop and with free datasets or easily collectable data.

50 AI Project Ideas Class 12 (detailed)

1. Handwritten Digit Recognition (MNIST) — Beginner

Description: Build a model to recognize digits (0–9) from images.
Techniques: Image classification, neural networks (simple CNN).
Tools: Python, Keras/TensorFlow, Google Colab.
Data: MNIST dataset (built-in in Keras).
Steps: load data → preprocess → build CNN → train → evaluate → demo predictions.
Features: Upload an image or draw digits to test.
Learning: CNN basics, data preprocessing, model accuracy measurement.

2. Spam Email Classifier — Beginner

Description: Classify emails as spam or not spam using text features.
Techniques: Text classification, feature extraction (TF-IDF), Naive Bayes or logistic regression.
Tools: Python, scikit-learn, NLTK.
Data: Public spam datasets (e.g., SMS Spam Collection).
Steps: clean text → vectorize (TF-IDF) → train classifier → evaluate → show confusion matrix.
Features: Simple web form to test messages.
Learning: NLP preprocessing, vectorization, model evaluation.

3. Movie Recommendation System (Basic) — Beginner–Intermediate

Description: Recommend movies to a user based on ratings.
Techniques: Collaborative filtering, similarity metrics.
Tools: Python, pandas, scikit-learn.
Data: MovieLens small dataset.
Steps: load ratings → compute user/item similarity → provide top-N recommendations → test with user profiles.
Features: Input a user's liked movies to get suggestions.
Learning: Recommender basics, similarity calculations.

4. Face Detection with OpenCV — Beginner

Description: Detect faces in images or webcam feed.
Techniques: Classical computer vision (Haar cascades) or modern DNN detectors.

Tools: Python, OpenCV.
Data: Use webcam or sample images.
Steps: load cascade → detect faces → draw bounding boxes → display results.
Features: Live webcam demo.
Learning: Image processing, working with camera input.

5. Sentiment Analysis of Movie Reviews — Beginner–Intermediate

Description: Predict positive/negative sentiment from reviews.
Techniques: Text classification, word embeddings (optional).
Tools: Python, scikit-learn / Keras, NLTK.
Data: IMDB movie review dataset (small).
Steps: clean data → vectorize → build model → evaluate → visualize metrics.
Features: Input text box to test a review's sentiment.
Learning: Sentiment modeling, cross-validation.

6. Hand Gesture Recognition for Simple Commands — Intermediate

Description: Recognize hand gestures and map to commands (e.g., play/pause).
Techniques: Image classification, CNNs, OpenCV for preprocessing.
Tools: Python, OpenCV, TensorFlow/Keras.
Data: Self-collected images or public gesture datasets.
Steps: collect images → label → train CNN → test with webcam.
Features: Real-time gesture-based control demo.
Learning: Data collection, augmentation, real-time inference.

7. Traffic Sign Recognition — Intermediate

Description: Classify traffic sign types from images (helpful for driver-assist projects).
Techniques: CNN, transfer learning.
Tools: Python, Keras, OpenCV.
Data: German Traffic Sign Recognition Benchmark (GTSRB).
Steps: data preprocessing → use transfer learning (MobileNet) → train → evaluate.
Features: Upload an image to classify sign type.
Learning: Transfer learning, multiclass classification.

8. Plant Disease Detection (Leaf Images) — Intermediate

Description: Detect if a plant leaf has a disease and suggest action.
Techniques: Image classification, data augmentation.
Tools: Python, TensorFlow/Keras.
Data: PlantVillage dataset (public).

Steps: download dataset → preprocess → train model → build simple UI.
Features: Upload leaf photo to get diagnosis and care tips.
Learning: Practical CV and dataset balancing.

9. Chatbot for School Information — Beginner–Intermediate

Description: Create a rules-based or ML chatbot to answer common questions about your school (timings, courses).
Techniques: Retrieval-based chatbot, intent classification.
Tools: Python, NLTK, Flask or Streamlit.
Data: Create small Q&A dataset from frequently asked questions.
Steps: design intents → train simple classifier → map to replies → web interface.
Features: Chat UI for Q&A.
Learning: Dialogue design, intent recognition.

10. Digit Recognizer Using Webcam Draw — Beginner

Description: Draw digits on screen (canvas) and predict them using a trained model.
Techniques: Image preprocessing, CNN inference.
Tools: JavaScript or Python (Flask/Canvas) + Keras.
Data: MNIST for training.
Steps: train model → deploy model → create drawing canvas → send image for prediction.
Features: Interactive drawing and prediction.
Learning: Model deployment basics and frontend-backend interaction.

11. News Article Classifier (Topic) — Intermediate

Description: Classify news articles into categories like sports, politics, tech.
Techniques: Text classification, TF-IDF or embeddings, SVM or neural nets.
Tools: Python, scikit-learn, NLTK/spaCy.
Data: AG News or custom scraped dataset.
Steps: collect articles → preprocess → vectorize → train → test.
Features: Input a news paragraph for category prediction.
Learning: Multi-class text classification.

12. Real-time Object Detection (YOLO/SSD) — Advanced

Description: Detect multiple objects in live camera feed using a pre-trained model like YOLO.
Techniques: Object detection, transfer learning.
Tools: Python, OpenCV, pre-trained YOLOv3/v4 or SSD.
Data: Pre-trained weights (COCO dataset).
Steps: load model → run on webcam frames → draw bounding boxes and labels.

Features: Live detection with confidence scores.
Learning: Advanced CV, model inference optimization.

13. Music Genre Classifier — Intermediate

Description: Classify songs into genres based on audio features.
Techniques: Audio feature extraction (MFCCs), classification with SVM or neural nets.
Tools: Python, librosa, scikit-learn.
Data: GTZAN dataset or small curated set.
Steps: extract MFCCs → train classifier → evaluate → demo with audio file.
Features: Upload audio to predict genre.
Learning: Audio signal processing and feature engineering.

14. Face Recognition Attendance System — Intermediate

Description: Mark attendance by recognizing student faces via webcam.
Techniques: Face detection + face recognition (embeddings).
Tools: Python, OpenCV, face_recognition library (dlib).
Data: Photos of students (small dataset you collect).
Steps: collect labeled photos → compute encodings → compare webcam face encodings → mark attendance.
Features: Attendance log in CSV.
Learning: Ethical data collection, face embeddings.

15. Optical Character Recognition (OCR) for Documents — Intermediate

Description: Extract printed text from scanned images or photos.
Techniques: OCR using Tesseract or custom model for specialized fonts.
Tools: Python, pytesseract, OpenCV.
Data: Scanned documents or printed notes.
Steps: preprocess images → run OCR → clean text → export to text file.
Features: Upload image to get editable text.
Learning: Image preprocessing and text extraction.

16. AI Tutor: Quiz Question Generator — Beginner–Intermediate

Description: Use simple NLP to generate MCQs from a short text passage.
Techniques: NLP, rule-based or transformer-based summarization (optional).
Tools: Python, NLTK, Hugging Face (optional).
Data: Text passages from textbooks.
Steps: identify key sentences → create question stems → generate distractors → present quiz.

Features: Auto-generate quizzes for study practice.
Learning: NLP concept extraction and evaluation.

17. Emotion Detection from Voice — Intermediate

Description: Recognize basic emotions (happy, sad, angry) from short audio clips.
Techniques: Audio feature extraction, classification.
Tools: Python, librosa, scikit-learn or Keras.
Data: RAVDESS or Emo-DB datasets.
Steps: extract features → train model → test with voice recordings.
Features: Live test or file upload.
Learning: Audio ML and feature selection.

18. Fake News Detector — Intermediate–Advanced

Description: Predict whether a news article is real or fake using textual cues.
Techniques: NLP, TF-IDF, transformers (optional).
Tools: Python, scikit-learn, Hugging Face.
Data: Fake news datasets from research repositories.
Steps: collect labeled articles → preprocess → train → evaluate for precision/recall.
Features: Article URL or text input to flag potential fake news.
Learning: Model fairness, dataset bias awareness.

19. Emotion Recognition from Facial Expressions — Intermediate

Description: Classify emotions from facial images or webcam feed (happy, sad, neutral).
Techniques: Image classification, facial landmark preprocessing.
Tools: OpenCV, Keras/PyTorch.
Data: FER2013 or CK+ dataset.
Steps: detect face → preprocess → predict emotion → show label.
Features: Live emotion meter or heatmap.
Learning: Facial analysis and dataset limitations.

20. Smart To-Do List with Priority Prediction — Beginner

Description: Predict which tasks are likely important based on short descriptions and suggest priorities.
Techniques: Text classification, simple ML models.
Tools: Python, scikit-learn, Streamlit.
Data: Create small dataset of tasks labeled by priority.
Steps: collect labeled tasks → train classifier → integrate into to-do UI.
Features: Suggest due dates/priority.
Learning: User-focused ML and product thinking.

21. Handwritten Name Recognition (Custom) — Beginner

Description: Build a model to read a student's handwritten name (few specific patterns).
Techniques: OCR + classification (small vocabulary).
Tools: Python, OpenCV, Keras.
Data: Collect 20–50 handwritten samples per name.
Steps: collect data → train simple CNN → evaluate.
Features: Recognize names from scanned forms.
Learning: Small-data handling and augmentation.

22. Voice-Controlled Home Simulation — Intermediate

Description: Control a simulated home (lights, fan) using voice commands.
Techniques: Speech recognition, basic command mapping, state machine.
Tools: Python, SpeechRecognition library, Flask.
Data: Command phrases (create your own dataset).
Steps: capture voice → convert to text → map to actions → update UI.
Features: Web dashboard to show device states.
Learning: Integrating speech recognition with apps.

23. Road Lane Detection for Smart Bikes — Intermediate

Description: Detect lane lines in front of a camera to warn riders when drifting.
Techniques: Image processing (edge detection, Hough transform), simple ML.
Tools: OpenCV, Python.
Data: Road videos or images.
Steps: preprocess frames → detect edges → find lanes → overlay lines.
Features: Visual warnings on video feed.
Learning: Real-world CV with noisy environments.

24. Text Summarizer for Study Notes — Intermediate

Description: Automatically produce short summaries of long chapters or articles.
Techniques: Extractive summarization (TextRank) or abstractive with small transformers.
Tools: Python, NLTK, gensim, Hugging Face (optional).
Data: Textbook chapters or long articles.
Steps: preprocess text → apply summarizer → show key points.
Features: Adjustable summary length.
Learning: Important NLP algorithms and trade-offs.

25. Handwritten Equation Solver — Advanced

Description: Recognize a handwritten math equation from an image and solve it.
Techniques: OCR for symbols, expression parsing, symbolic math (SymPy).

Tools: OpenCV, TensorFlow, SymPy.
Data: Collect symbol images or use EMNIST for characters.
Steps: detect symbols → classify → reconstruct equation → solve.
Features: Photo-to-solution demo.
Learning: Combining CV, parsing, and symbolic math.

26. Traffic Flow Analysis from Video — Intermediate–Advanced

Description: Count vehicles, measure speed range, and estimate congestion from road video.
Techniques: Object detection + tracking, counting logic.
Tools: OpenCV, YOLO or SSD, tracking algorithms.
Data: Road camera videos (public traffic cams).
Steps: detect vehicles → track across frames → count and compute flow metrics.
Features: Dashboard reporting hourly counts.
Learning: Tracking algorithms and practical metrics.

27. Language Translator (Small) — Intermediate

Description: Translate short phrases between two languages (e.g., English ↔ Hindi) using sequence-to-sequence models or an API.
Techniques: Seq2Seq, tokenization, attention (optional).
Tools: Python, Hugging Face transformers or translate API.
Data: Small parallel corpus (many public datasets exist).
Steps: preprocess pairs → train model or call API → test translations.
Features: Input box to get translated text.
Learning: Basics of machine translation and tokenization.

28. Predicting Student Grades (Regression) — Beginner–Intermediate

Description: Predict final exam marks based on attendance, assignments, and midterm scores.
Techniques: Regression (linear, tree-based).
Tools: Python, scikit-learn, pandas.
Data: Create a synthetic or anonymized dataset.
Steps: clean data → choose features → train regression → evaluate (RMSE).
Features: Visualize predictions vs actual.
Learning: Regression, feature importance, ethical use of predictions.

29. Book Recommendation via Content-Based Filtering — Beginner

Description: Recommend books based on text features (synopsis, genre).
Techniques: TF-IDF on book descriptions, cosine similarity.
Tools: Python, scikit-learn.
Data: Goodreads dataset or small collection of books.
Steps: vectorize synopses → compute similarity → recommend top matches.
Features: Enter a book to get similar books.
Learning: Content-based recommenders and text similarity.

30. Pneumonia Detection from Chest X-rays (Educational) — Intermediate–Advanced

Description: Detect signs of pneumonia from X-ray images (for learning only—do not use clinically).
Techniques: CNN, medical image preprocessing, transfer learning.
Tools: Python, Keras/TensorFlow.
Data: Public chest X-ray datasets (research use).
Steps: preprocess images → train/finetune model → evaluate with sensitivity/ specificity.
Features: Upload X-ray and show likelihood (for study only).
Learning: Responsible use of AI in healthcare and evaluation metrics.

31. Optical Music Recognition (Basic) — Advanced

Description: Convert images of simple sheet music into playable MIDI.
Techniques: Symbol detection, sequence reconstruction.
Tools: OpenCV, Python, music21 (MIDI generation).
Data: Images of simple sheet music.
Steps: detect staff lines → detect notes → map to pitch/duration → export MIDI.
Features: Play extracted music.
Learning: Complex vision-to-sequence pipeline.

32. Food Calorie Estimator from Image — Intermediate

Description: Estimate the calorie range of a plated food image.
Techniques: Image classification + portion estimation (approx).
Tools: Python, Keras, pre-trained image models.
Data: Food image datasets (Food-101) and approximate calorie references.
Steps: detect food type → estimate portion/weight heuristically → compute calories.
Features: Upload food photo to get calorie estimate.
Learning: Practical limitations and uncertainty reporting.

33. Keyword Extractor for Notes — Beginner

Description: Extract important keywords from lecture notes automatically.
Techniques: TF-IDF, RAKE, TextRank.

Tools: Python, gensim, NLTK.
Data: Class notes or textbook paragraphs.
Steps: preprocess → run keyword extraction → present top keywords.
Features: Adjustable number of keywords.
Learning: NLP summarization primitives.

34. Image Colorizer (Black & White to Color) — Advanced

Description: Colorize grayscale images using deep learning.
Techniques: Image-to-image translation (CNNs, U-Net).
Tools: Python, Keras/PyTorch.
Data: Color image datasets (train on color images converted to grayscale).
Steps: prepare pairs → train model → demo colorized outputs.
Features: Upload B/W image to get colorized version.
Learning: GANs/U-Nets and perceptual loss concepts.

35. Weather Prediction (Short-Term) Using Time Series — Intermediate

Description: Predict next-day temperature using historical data.
Techniques: Time-series forecasting (ARIMA or LSTM).
Tools: Python, pandas, statsmodels, Keras.
Data: Local weather data from public APIs (download CSV).
Steps: preprocess time series → train model → evaluate with MAE/RMSE.
Features: Graph of predicted vs actual.
Learning: Time-series modeling basics.

36. License Plate Recognition — Intermediate–Advanced

Description: Detect and read number plates from vehicle images.
Techniques: Object detection + OCR.
Tools: OpenCV, pytesseract, deep learning detector.
Data: Images of cars (collect or use public datasets).
Steps: detect plate → crop → enhance → run OCR → format text.
Features: Output license string and location on image.
Learning: OCR in constrained environments.

37. Personal Expense Categorizer — Beginner

Description: Automatically categorize expense descriptions into categories (food, transport).
Techniques: Text classification, simple ML.
Tools: Python, scikit-learn.
Data: Create a labeled expense dataset.

Steps: vectorize descriptions → train classifier → integrate into simple app.
Features: Upload CSV to categorize expenses.
Learning: Practical feature engineering and automation.

38. Language Detection Tool — Beginner

Description: Detect which language a short text is written in.
Techniques: Character n-gram models, Naive Bayes.
Tools: Python, scikit-learn.
Data: Use multilingual sample texts for training.
Steps: train model on n-gram features → test with short phrases.
Features: Show probable language and confidence.
Learning: Simple text modeling and multilingual handling.

39. AI-based Flashcard Generator — Beginner

Description: Convert study notes into flashcards automatically.
Techniques: Key-sentence extraction and question generation (basic).
Tools: Python, NLTK, simple templates.
Data: Study notes or chapter text.
Steps: extract key sentences → transform into question-answer pairs → export flashcards.
Features: Downloadable CSV of flashcards.
Learning: Automated content creation for revision.

40. Signature Verification (Basic) — Intermediate

Description: Verify if two signatures belong to the same person using image features.
Techniques: Feature extraction, similarity metrics, Siamese network (optional).
Tools: OpenCV, Keras (Siamese).
Data: Collect signatures (consent required).
Steps: preprocess → extract features or train Siamese network → compute match score.
Features: Input two images to get match probability.
Learning: Biometrics and security considerations.

41. Crop Recommendation System for Farmers — Beginner–Intermediate

Description: Recommend the best crop to grow based on soil, season, and rainfall data.
Techniques: Rule-based system or simple ML classification.
Tools: Python, pandas, Flask.
Data: Public agriculture guidelines and local climate data.
Steps: collect rules/data → build mapping or classifier → deploy simple UI.
Features: Input soil type and season for recommendations.
Learning: Building useful domain-specific AI tools.

42. Image Caption Generator — Advanced

Description: Generate natural language captions for images using encoder-decoder models.
Techniques: CNN encoder + RNN decoder (or attention models).
Tools: TensorFlow/Keras, pre-trained CNN.
Data: Flickr8k or MSCOCO subsets.
Steps: extract image features → train decoder on captions → generate captions for new images.
Features: Upload image to get caption.
Learning: Multimodal AI and sequence modeling.

43. Real-time Sign Language Translator (Static Signs) — Advanced

Description: Translate a set of static sign language gestures into text in real-time.
Techniques: Image classification + webcam input, possibly pose estimation.
Tools: OpenCV, MediaPipe (for landmarks), Keras.
Data: Collect images of signs or use existing datasets.
Steps: detect hand landmarks → classify sign → display text output.
Features: Live translation for a limited vocabulary.
Learning: Human pose estimation and accessibility tech.

44. Voice Cloning (Demo with Caution) — Advanced (Ethical Considerations)

Description: Create a demo that mimics a voice using a tiny dataset (educational only).
Techniques: Speech synthesis models (e.g., Tacotron/Glow-TTS) — only study—don't deploy.
Tools: Python, TTS libraries.
Data: Short recorded speech (with consent).
Steps: study pre-trained models → experiment with small fine-tuning → generate short clips.
Features: Understand TTS pipeline; do not misuse.
Learning: Ethics in AI and speech synthesis.

45. Road Damage Detection for Local Municipality — Intermediate

Description: Detect potholes and cracks from road images to help maintenance planning.
Techniques: Image classification/detection, segmentation.
Tools: Python, OpenCV, Keras.
Data: Collect images or use public road-defect datasets.
Steps: label images → train classifier or segmentation model → produce heatmap of damage probability.

Features: Simple report with affected coordinates.
Learning: Practical CV for civic applications.

46. Fake Image Detector (Manipulation Detection) — Advanced

Description: Detect if an image has been manipulated (basic tampering detection).
Techniques: Forensics features, CNNs trained on tampered vs original images.
Tools: Python, Keras.
Data: Create tampered images or use public datasets.
Steps: generate tampered samples → train classifier → evaluate robustness.
Features: Input image returns likely genuine/manipulated score.
Learning: Image forensics and limits of detectors.

47. Automated Timetable Generator — Beginner–Intermediate

Description: Generate a school timetable automatically respecting constraints (teacher availability, room).
Techniques: Constraint solving, simple heuristic algorithms.
Tools: Python, OR-tools (Google).
Data: Teacher lists, subjects, room availability.
Steps: model constraints → implement solver → output timetable.
Features: Export timetable PDF or CSV.
Learning: Optimization and practical scheduling.

48. Personality Predictor from Short Text (MBTI-lite) — Intermediate

Description: Predict a simplified personality trait from short social text (educational).
Techniques: Text classification, supervised learning.
Tools: Python, scikit-learn, transformers (optional).
Data: Create small labeled dataset or use public resources carefully.
Steps: prepare dataset → train classifier → show traits with confidence and disclaimers.
Features: Provide ethical disclaimer; not for serious use.
Learning: Responsible AI and limitations of personality prediction.

49. Color Palette Generator from Images — Beginner

Description: Extract dominant colors from an image and present a palette for designers.
Techniques: K-means clustering on pixel colors.
Tools: Python, OpenCV, scikit-learn.
Data: Any images (photos).

Steps: load image → run k-means on pixels → display top K colors and hex codes.
Features: Downloadable palette or CSS variables.
Learning: Clustering and applied color extraction.

50. Road Sign Voice Alert System for Drivers — Intermediate

Description: Recognize road signs and speak the sign aloud (audio alert) to assist drivers.
Techniques: Object detection + text-to-speech (TTS).
Tools: OpenCV, pre-trained detector, pyttsx3 or TTS API.
Data: Traffic sign dataset for detection and mapping to voice messages.
Steps: detect sign → map to message → speak with TTS → overlay on video.
Features: Live demo with audio alerts.
Learning: Multimodal outputs and accessibility.

MUST READ: 50 English Project Ideas for Exhibition 2026

Implementation Checklist — Practical steps every student should follow

1. **Choose scope carefully:** Select a limited scope that is doable within your time and resources.
2. **Data first:** Find or create datasets before coding the full solution. Even small datasets are fine for demos.
3. **Start simple:** Build a baseline model (e.g., logistic regression) first, then improve with more advanced methods.
4. **Document experiments:** Keep a notebook with parameter changes, results, and conclusions.
5. **Make a demo:** Use Streamlit, Flask, or simple Jupyter Notebook with sample inputs to show working features.
6. **Prepare a report/presentation:** Include problem statement, approach, dataset description, model details, evaluation, limitations, and future work.

Conclusion

This collection of **50 AI project ideas class 12** is designed to give you a wide range of choices across domains: computer vision, natural language processing, audio, optimization, and practical applications for school and society. Each project is crafted to be approachable for a student with basic Python skills while still offering room to explore advanced concepts if you want to push further.

Pick one that excites you, read about the required libraries, gather a small dataset, and start building. Remember that demonstrating a clear problem statement, showing results, documenting your approach, and reflecting on limitations is more important than making a perfect system. Good luck — finish one project well, and you'll learn far more than starting ten and leaving them incomplete.

Project Ideas

‹ 110+ Best SIDP Project Ideas For Class 9 Students

Search

Search

Recent Posts

110+ Best SIDP Project Ideas For Class 9 Students

119+ New Sustainable Agriculture Project Class 12

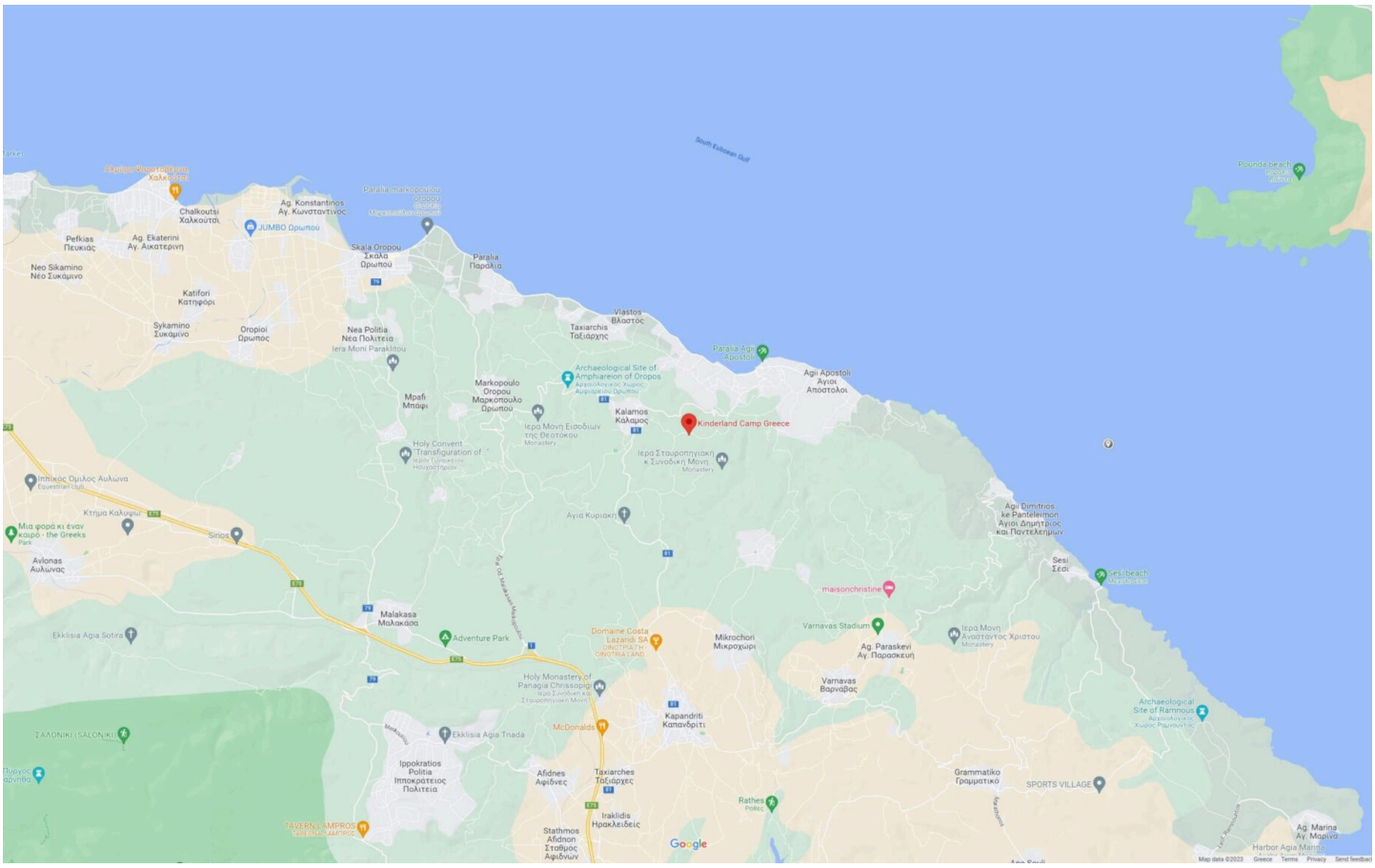
50 English Project Ideas for Exhibition 2026

97+ Best English Project Class 12 Topics For Students in 2025-2026

49+ Chemistry Project Ideas for Class 12 For 2026

Recent Comments

No comments to show.



Do not miss this experience!

ASK US ANY QUESTIONS

GET IN TOUCH

KIDS PROJECT IDEAS

About us

Whether you're a parent looking for weekend fun, a teacher searching for class ideas, or a young creator full of imagination — KidsProjectIdeas.com is the perfect place to begin your creative journey!

Address

Ten assi, Cali 190 14, USA

Contact

Office hours: 09:00am - 6:00pm

chloekidsprojectideas@gmail.com

Privacy Policy

© 2025 Kids Project Ideas