Computer Science Summer School Projects: Guidelines & Map

This guide is to help you use what you learned on June 9 and apply it toward building a project for our next gathering on June 26. Below is a roadmap and suggested reading for each project. Choose a project from your assigned group (AI, Cyber Security, or Game Development), follow the steps, and be creative!

🧠 AI & Machine Learning

**✅ Project Option 1: Emotion-Based Response Bot (Beginner)**

**Builds on:** Sentiment Analysis mini-activity  
**Goal:** Create a simple chatbot that detects the emotion of user input and gives a response that matches the mood.

**Suggested Team Roles:**

* NLP / sentiment detection
* Chatbot interface design

**Steps:**

1. Use the Google Colab Sentiment Analysis example as your base.
2. Create a chatbot interface using input() or widgets in Colab.
3. Add conditions: if sentiment is negative, respond positively. If positive, respond normally.

**Timeline:**

* June 10–12: Setup & training
* June 13–17: Integrate chatbot logic
* June 18–21: Polish & test

**Resources:**

* [TextBlob Docs](https://textblob.readthedocs.io/en/dev/)
* [HuggingFace Sentiment Models](https://huggingface.co/models?pipeline_tag=text-classification)
* [Real Python Chatbot Tutorial](https://realpython.com/python-chat-bot/)
* [Simple Chatbot GitHub](https://github.com/python-engineer/chatbot-python)

**✅ Project Option 2: AI Typing Speed Analyser (Intermediate)**

**Builds on:** Typing Speed Game + basic ML logic  
**Goal:** Build an app that not only measures typing speed but uses simple AI to detect improvements over time or give feedback based on performance.

**Suggested Team Roles:**

* Typing game mechanic (JavaScript or p5.js)
* Stats tracker (Python/Notebook)
* Feedback logic (basic AI pattern recognition or rules)

**Steps:**

1. Build on the Typing Speed Game HTML provided.
2. Add data capture (words per minute, errors, time).
3. Use a basic logic rule or ML model to analyse:
   * Has the player improved over time?
   * Where do most errors occur?
4. Provide a visual dashboard or feedback summary.

**Timeline:**

* June 10–13: Typing game + data logging
* June 14–18: Analyser logic
* June 19–24: Visualisation + testing

**Resources:**

* [Typing Speed GitHub Projects](https://github.com/topics/typing-speed)
* [p5.js for UI](https://p5js.org/reference/)
* Intro to Data Analysis with Python
* Beginner ML Feedback App (Medium)

**✅ Project Option 3: AI Image Classifier Web App (Advanced)**

**Builds on:** Image Classifier & AI concepts

**Goal:** Create an image classifier (e.g., plants vs. animals vs. objects) using pre-trained models and let users upload images.

**Suggested Team Roles:**

* Frontend for image upload
* Backend for model classification
* Results UI + confidence graph

**Steps:**

1. Use MobileNet or Teachable Machine model.
2. Build image upload form in HTML.
3. Display prediction with confidence score.

**Timeline:**

* June 10–14: Build frontend and model integration
* June 15–21: Connect prediction + UI
* June 22–25: Testing, polish, docs

**Resources:**

* [Teachable Machine](https://teachablemachine.withgoogle.com/)
* [ml5.js ImageClassifier](https://learn.ml5js.org/#/reference/image-classifier)
* [Google Colab ML Examples](https://colab.research.google.com/github/tensorflow/examples)
* [Hands-On ML with Scikit-Learn & TensorFlow (book)](https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/)

🔒 Cyber Security

**✅ Project Option 1: Hack Me If You Can (Beginner)**

**Builds on:** SQL Injection and XSS mini-activities

**Goal:** Create a mock login form with known vulnerabilities. Let other groups try to break it, then secure it.

**Suggested Team Roles:**

* Form developer
* Security tester
* Patch writer

**Steps:**

1. Use the provided SQL Injection simulation code.
2. Add password field, comment box.
3. Demonstrate vulnerabilities → fix with sanitation.

**Timeline:**

* June 10–13: Build vulnerable form
* June 14–17: Demo & attack
* June 18–21: Secure + explain

**Resources:**

* [OWASP Top 10](https://owasp.org/www-project-top-ten/)
* [XSS Prevention Cheat Sheet](https://owasp.org/www-community/xss-prevention)
* [SQLi Lab Practice](https://github.com/Audi-1/sqli-labs)
* [Cybrary SQLi Course](https://www.cybrary.it/course/sql-injection/)

**✅ Project Option 2: Encryption Toolbox (Intermediate)**

**Builds on:** Caesar Cipher and Python Encryption

**Goal:** Build a tool that encrypts and decrypts messages using Caesar, ROT13, and Fernet.

**Suggested Team Roles:**

* Python encryption coder
* UI builder (Jupyter/HTML)

**Steps:**

1. Start with Caesar Cipher code.
2. Add Fernet encryption from activity.
3. Create toggles between methods.

**Timeline:**

* June 10–12: Core encrypt/decrypt logic
* June 13–18: UI + method switcher
* June 19–24: Test and polish

**Resources:**

* [Cryptography in Python (Fernet)](https://cryptography.io/en/latest/fernet/)
* [ROT13 Algorithm](https://en.wikipedia.org/wiki/ROT13)
* [GitHub: PyCryptodome Examples](https://github.com/Legrandin/pycryptodome)
* [YouTube: Caesar Cipher Tutorial](https://www.youtube.com/watch?v=6uGm8tKO1uU)

**✅ Project Option 3: Secure Chat App Simulation (Advanced)**

**Builds on:** Multiple activities incl. encryption, XSS

**Goal:** Create a simple encrypted chat between two users in-browser or notebook.

**Suggested Team Roles:**

* Message encryption logic
* Frontend chat interface
* Session simulation

**Steps:**

1. Build input-output message UI.
2. Encrypt messages on send.
3. Decrypt on receive.

**Timeline:**

* June 10–14: UI + logic
* June 15–20: Encryption + message history
* June 21–25: Testing + write-up

**Resources:**

* [Socket.IO (Advanced)](https://socket.io/)
* [Intro to Python WebSockets](https://websockets.readthedocs.io/)
* [Build a Chat App Tutorial](https://www.freecodecamp.org/news/build-a-realtime-chat-app-with-node-express-socket-io-and-heroku/)
* [Cryptography Chat Example](https://github.com/agusmakmun/pychat)

🎮 Game Development

**✅ Project Option 1: Dino Runner with Level Unlocker (Beginner)**

**Builds on:** Dino Runner + Level Unlocker mini-activities

**Goal:** Make the dino runner game harder as you score more points.

**Suggested Team Roles:**

* Game logic
* UI/UX + level unlocking

**Steps:**

1. Use the full Dino Runner game provided.
2. Add level variable based on score.
3. Increase speed / obstacles.

**Timeline:**

* June 10–12: Basic game setup
* June 13–18: Levels + polish
* June 19–25: Testing

**Resources:**

* [W3Schools - Local Storage](https://www.w3schools.com/js/js_storage.asp)
* [p5.js Docs](https://p5js.org/reference/)
* [JavaScript Game Dev YouTube Playlist](https://www.youtube.com/playlist?list=PL0Zuz27SZ-6Mx9fd9elt80G1bPcySmWit)
* [HTML5 Game Dev Forum](https://www.html5gamedevs.com/)

**✅ Project Option 2: Maze Explorer vs. Solver (Intermediate)**

**Builds on:** Maze Game mini-activity

**Goal:** Create two game modes: player mode and auto-solve mode.

**Suggested Team Roles:**

* Maze generation + rendering
* Player logic
* AI auto-solver logic

**Steps:**

1. Interactive maze (20x20 or 40x40)
2. BFS/DFS auto-solver
3. Compare paths

**Timeline:**

* June 10–14: Build full game
* June 15–20: Add AI + display
* June 21–25: Polish + test

**Resources:**

* [Recursive Backtracking Algorithm](https://weblog.jamisbuck.org/2011/2/27/maze-generation-recursive-backtracking)
* [Pathfinding Visualiser](https://qiao.github.io/PathFinding.js/visual/)
* [A\* Maze Solver GitHub](https://github.com/AmeyGawali/A-Maze-Solver)
* [YouTube: Maze Game Tutorial](https://www.youtube.com/watch?v=HyK_Q5rrcr4)

**✅ Project Option 3: Multiplayer Scoreboard Battle (Advanced)**

**Builds on:** Score Counter + Catch the Target + Sound Game

**Goal:** Create a fast-paced clicking or keyboard game with score battles.

**Suggested Team Roles:**

* Game logic
* Scoreboard management
* Timer logic

**Steps:**

1. Combine scoreboard + timer + player buttons
2. Add random targets
3. End after time and show winner

**Timeline:**

* June 10–13: Basic game
* June 14–18: Multiplayer logic + scoreboard
* June 19–25: Test + enhancements

**Resources:**

* [Multiplayer Game Tutorial](https://developer.mozilla.org/en-US/docs/Games/Multiplayer_games)
* [Basic Game Design Tips](https://www.gamedeveloper.com/)
* [Phaser Multiplayer Guide](https://phaser.io/news/2020/04/building-multiplayer-games)
* [HTML5 Multiplayer Game GitHub](https://github.com/nkholski/html5multiplayergame)