

أسئلة متكررة بخصوص أفكار المشاريع:

- يجب أن يكون هناك طريقة واضحة لإظهار نتائج المشروع، وإدخال البيانات لـ algorithm .. ولذلك قد يتطلب المشروع Basic GUI .. في هذه الحالة يجب أن يكون هناك Basic GUI لإدخال الإختيارات وإظهار النتائج، وليس على GUI أي درجات، ولكن عدم وجوده سيتسبب في خصم درجات.
- الـ development platform هي tools والـ programming languages والـ libraries المستخدمة في المشروع.
- الـ diagrams المطلوبة هي block diagrams وـ flowcharts توضح algorithms المستخدمة، إلى جانب plots توضح النتائج، إلى جانب أي UML Diagrams تجرّنها ضرورية (مثلاً، use-case diagram توضح functions/features من وجهة نظر المستخدم).
- أفكار المشاريع التي بها puzzles كالـ Sudoko والـ N-Puzzle والـ N-Queens، المشروع هو للعثور على حل اللعبة بشكل عام (بداية puzzle يجب أن تختلف كل مرّة)، وليس لحالة واحدة خاصة فقط.
- أفكار المشاريع التي بها Two-Player Games، المشروع هو لعمل لاعب ذكي واحد فقط لهذه اللعبة، بينما الطرف الآخر من اللعبة هو إنسان وليس آلة. ويجب على اللاعب الذكي استخدام approach المحدد لنقرير كل حركة/خطوة يقوم بها من اللعبة (كرد على كل حركة يقوم بها الطرف الآخر).
- أفكار المشاريع التي بها heuristic functions على الأقل، ويجب حل المشكلة باستخدام كل heuristic function بشكل منفصل، ثم المقارنة بين تأثير كل منها على سرعة الوصول للحل وجودة الحل.
- أفكار المشاريع التي بها Two-Player Games، يجب تطبيق Minimax والـ Alpha-Beta Pruning واستخدام Heuristics بشكل منفصل (أي تطوير إمكانية تحديد كل حركة في اللعبة، مرة باستخدام Minimax فقط دون أي Heuristics "search state space" إن كان ذلك بسيط كالـ Tic-Tac-Toe، ثم مرة أخرى ولكن مع تطبيق Alpha-Beta Pruning، ثم Minimax باستخدام إحدى Heuristics، ثم Minimax باستخدام الـ Heuristic Function الثانية).
- في Tic-Tac-Toe، يجب بالإضافة لما سبق تطبيق Symmetry Reduction والـ Heuristic Reduction، كلاهما بشكل منفصل.
- أفكار المشاريع التي بها Two-Player Games أو puzzles، إذا لم يكن board size موضحاً في project description فيجب الالتزام بالـ standard (مثلاً Intelligent Go Player يجب تطبيقه على 19 board size في 19).
- أفكار المشاريع التي تعتمد على مدخلات مختلفة (Job Scheduling Problem، Faculty's Timetable Scheduling Problem، Vehicle Routing Problem، Knapsack Problem، Routing Problem)، يجب على المستخدم أن يدخل كل أو معظم المدخلات المطلوبة (وليس أن يقوم algorithm كل مرّة بحل المشكلة باستخدام نفس data).
- أفكار المشاريع التي بها Two-Player Games أو puzzles لا تحتاج إلى Dataset لأنّه لا يوجد بها training phase لأنّها تصنف ضمن AI ولكن ليس بها ML كما أوضحت سابقاً.
- بالنسبة لأفكار المشاريع التي تحتاج إلى Dataset، بامكانكم استخدام أي dataset آخر بدلاً من المقرحة في projects' description.
- جميع الأفكار التي تحتاج إلى Dataset، عند مناقشة وتجربة المشروع، يتم ذلك باستخدام dataset testing instances من المستخدمة، والطلاب غير مطالبين باستخدام data خارجية.
- جميع الأفكار التي تحتاج إلى evolution أو training، أو إلى testing، يجب عمل plots توضح evolution أو training، وجدول يوضح نتائج testing (ويفضل عن طريق N Fold Cross Validation) كما أوضحت بالمحاضرات).

- أفكار المشاريع التي بها تطبيق طرفيتين أو أكثر (Genetic Algorithm AND the both Genetic Algorithms & Differential Evolution)، يجب حل المشكلة باستخدام كل طريقة منها بشكل منفصل، ثم المقارنة بين كفاءة كل منها في سرعة الوصول للحل وجودة الحل.

- أفكار المشاريع التي بها تطبيق Decision Trees & Random Forests، يجب حل المشكلة أولاً باستخدام Decision Tree واحدة فقط، ثم باستخدام Random Forest، فكل طريقة منها تطبق بشكل منفصل، ثم يتم المقارنة بين كفاءة كل منها في سرعة الوصول للحل وجودة الحل.

- أفكار المشاريع التي بها Object Detection، المطلوب تصنيف صور بها Object واحد فقط بكل صورة، وليس تصنيف أكثر من Object من صورة واحدة.

- أفكار المشاريع التي بها تطبيق CNN، يجب حل المشكلة باستخدام ANN تقليدية، ثم إذا رغب الفريق في تطبيق CNN كحل إضافي فإمكانه عمل ذلك بعد تطبيق المطلوب.

أسئلة بخصوص يوم المناقشة:

- يجب طباعة الـ report وتسليمها كـ Hard Copy عليه أسماء كل أعضاء الفريق، ورقم كل طالب، والمستوى الدراسي والقسم/الشعبية لكل عضو.

- يجب في الـ report وضع رابط لـ shared folder عليه كل الـ code والـ report (Shortened URL) .. ويجب أن يكون الـ folder متاح حتى موعد ظهور النتائج لنتمنى من الرجوع إليه في أي وقت.

N.B.; A documentation (report) should accompany the project. Provide clear and concise (yet, comprehensive) documentation explaining the functionality, algorithms used, and instructions for users to interact with the application. The report should explain the AI algorithms implemented, any design choices, the design rationale behind heuristic functions (if utilized), the rationale behind the values of any parameters/hyper-parameters in the employed algorithms. The documentation should serve as a guide for users and developers to understand the project's inner workings.

نموذج يمكن إتباعه عند كتابة الـ report :

Introduction and Overview:

- Project idea and overview.
- Applications (*desktop, web, or mobile*) similar to the one you're developing, and what are the functionalities/features, and how they work (*if that information is available*).
- A Literature Review of Academic publications (*papers/books/articles*) relevant to the problem you're trying to solve and the approach you're trying to implement (*at least 5 resources*). You may find them by searching using Google Scholar.

Proposed Solution & Dataset:

- Main functionalities/features (*from the users' perspective*) in your proposed software/solution (*can be explained using a use-case diagram*).
- If applicable to your project, the Dataset employed (*preferably a publicly available dataset*).

Applied Algorithms:

- All the details of the AI/Machine-Learning algorithm(s)/approach(es) used to develop your project (*can be explained using block diagrams*).

Experiments & Results:

- The Experiments, testing, and the results (*including plots of the evolution or training if applicable*) and samples of the output (*and how did you test the solution*).

Analysis, Discussion, and Future Work:

- Analysis of the results, what are the insights?
- What are the advantages / disadvantages?
- Why did the algorithm behave in such a way? What might be the future modifications you'd like to try when solving this problem?