Gambit Timeline:

1. Board Representation
   1. 6, 2 bitboards
   2. Parse FENs – consider UCI throughout design process
   3. Print board
2. Move Generation
   1. Generate pseudo legal moves, then add legality checks
      1. Pseudo legal move gen using magic bitboards
      2. Knights: Precompute lookup table for possible moves a knight could make for all 64 squares.
      3. Kings: Precompute lookup table for possible moves a king could make for all 64 squares.
      4. Pawns: Shift them all set wise and use masks?
      5. Bishops: Magics
      6. Rooks: Magics
      7. Queens: Magics (Rook glued to a bishop)
      8. Pawn promotions
      9. En Passant
      10. Castling
      11. Add legality checks
      12. “Make move” (function for searching, only changes its own boards whilst searching. To actually play a move is where UCI comes in)
      13. Implement make move as a random mover (which would be considered a V1)
3. Make UCI Compliant
4. Perft – For various positions, generates the expected number of move nodes in a reasonable amount of time. (UCI software generally has a perft command)
5. Search Function
   1. Negamax
   2. Alpha Beta
   3. Consider legality of the move just after you make it. Check for legality, if its legal, then proceed with the search, else go and take it back. Keep a flag to have at least one legal move played, if not then either its a check mate (if the king is in a check) or stalemate.
6. SPRT (Determine whether all expected move gen is accurate, and SPRT helps give a baseline of performance compared to future iterations of the project to see if additions help improve it), SPRT may have to be done after completion of search and eval as it needs to compare version 1 to version 2, although could compare it to other random movers. SPRT random mover vs various searches and evals

you SPRT to ascertain if the change makes the engine better or not, even if the change should 'just work' it can help catch bugs in your implementation

1. Eval Function
   1. Count material first, can make use of piece square tables and core chess principles, like castle before its too late, develop etc
   2. Incorporate idea of Gambit (look at contempt, similar vibes but not quite. May also want to make positions more complex when losing or against a worse opponent)
2. Testing
   1. Test with Lichess
   2. Test with chess society people
   3. Conduct “Turing Test”

Term 1:

Week 0 (work over summer):

1. Be able to represent a board state given any FEN
2. Be able to print a board to the terminal

Board representation, move gen for precomputed pieces and start on magics

Week 1 (Begins 30th September):

Work more on magics

Week 2:

Finish magics

Week 3:

Pawn promotions, en passant and castling

Week 4:

Legality checks

Week 5:

Make UCI compliant

Week 6:

Work with a tool like cutechess to test perft and SPRTs later. Debug move gen till it works well

Week 7:

Make move and random mover

Week 8:

Start negamax and basic eval

Week 9:

Finish negamax and basic eval

Week 10 – 11:

Presentation and interim report

Term 2:

Week 0 (Work over break):

SPRT negamax vs random

Week 1: (Begins 13th January)

Implement alpha beta search

Week 2:

SPRT alpha beta vs negamax

Week 3:

Work on gambit stuff

Week 4:

Work on gambit stuff

Week 5:

Work on gambit stuff

Week 6:

Work on gambit stuff

Week 7:

Work on gambit stuff

Week 8:

Finalise Gambit functionality

Week 9:

Test on people and gather findings and host on lichess

Week 10 – 11:

Final cleanup and prep for final submission

Extensions:

Be able to interact with a UI like cutechess-gui

Produce a wiki page

Suggested Extensions:

* Most chess players have an Elo assigned by sites like Chess.com. However, should they not have that to hand, develop a means of evaluating their Elo
* Provide an opening book. Perhaps with well-known gambit lines and/or regular openings.
* Custom UI for interacting with Gambit
* Two modes, standard which uses the Gambit approach, and the other uses the standard alpha beta approach.