Team A0 : Mancala BrainBot

Israel Escobar-Camacho
Nathaniel James
Hang Shu

Use Case

- Provide easy access for anyone to play and learn mancala
- Allow users to learn via gameplay history & engine response
- Allow users to find other players over the internet and play against them
- Allow users to train on their own by playing against a high level engines
- Allow users to observe live gameplay of player vs player, player vs computer, or computer vs computer mancala matches

Use Case Requirements - Engine

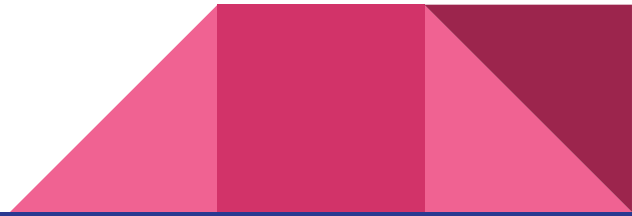
- 2-Ply Minimax strategy engine functioning as a basic opponent to users and initial training opponent to the reinforcement learning model
  - Using decision tree to maximize stone differential
- High level, self-play trained, reinforcement learning
  - This is the main engine opponent for users of the project to play against
  - Must play at or above the level of 95% of human mancala players

Use Case Requirements - Website

- Responsive website
  - Max gaming latency of 1 second
- Players can optionally make accounts
  - Ranking system among players
- Scalable
  - Hold at minimum 10 person to person games at a given time
- Provide a history record of the 5 most recent games for each player
- Intuitive gaming interface

Technical Challenge - Self-Play Reinforcement Learning

- **Minimax Strategy Engine**
  - In order to self-learn effectively, the RL model must have a relatively skillful opponent to start off against.
  - This strategy engine must also be able to know which moves it makes will allow it to move again.
- **Self-Play RL Model**
  - An efficient platform for the RL model to be able to play against itself many many times is required for effective training.



Technical Challenge - Backend

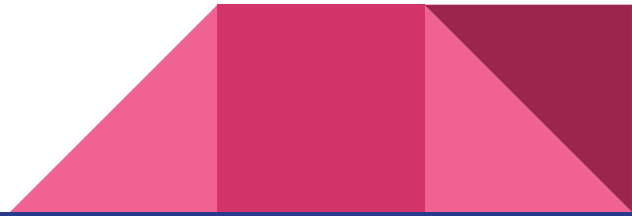
- **Supporting large scale of games at time**
  - Shared resource (games could be an array, linkedlist, hashtable, etc)
  - Resource needs to be scalable
  - Modify this resource efficiently to reach the 1 second latency goal for games
- **Storing history of games for each player**
  - Database required
  - Compression algorithm on game history to save space

Technical Challenge - Frontend

- **Game interface/site development**
  - Easy player to player game initialization
- **Viewing gameplay**
  - Multiple users concurrently retrieving data
- **Game performance**
  - Retrieve live changes in game state with 1 second latency

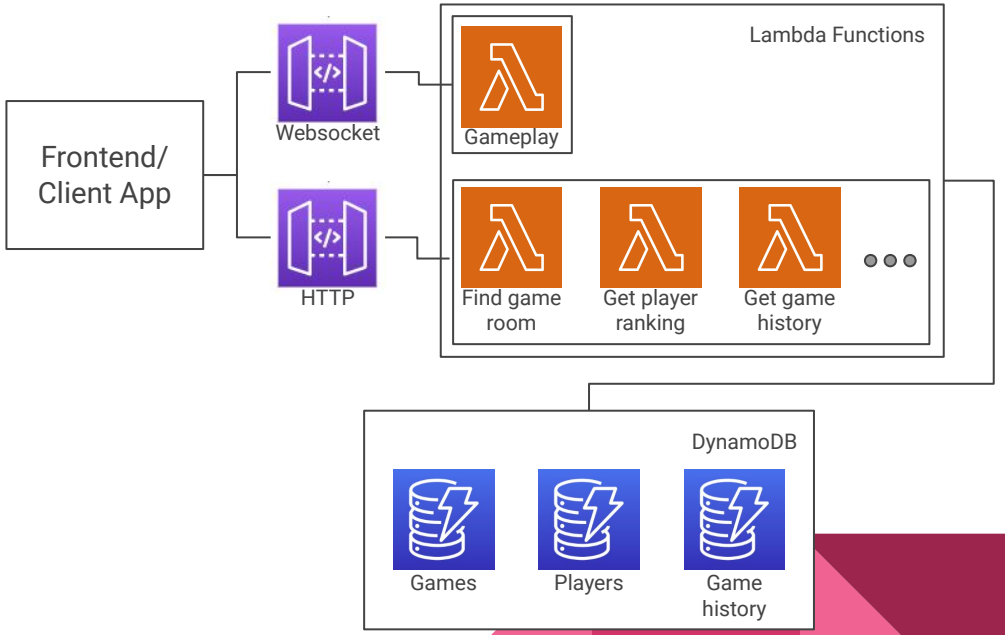
Solution - Reinforcement Learning

- Minimax engine as initial opponent for the RL
  - Pruning will be used to minimize the size of the decision tree
- Independent engine/opponent/backend structure
  - Server keeps track of whose turn it is
  - Each player (be it human or engine) takes in the board state when provided, and outputs their singular move.



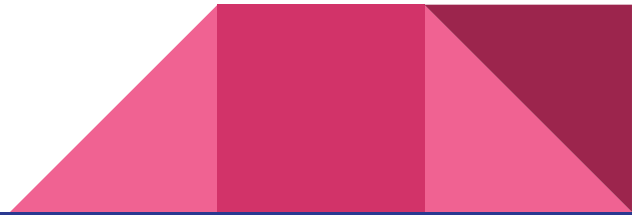
Solution - Backend

- AWS Lambda as the compute platform
  - Endpoints for some game logic
    - Finding a game room, player ranking, gameplay history
  - Websocket endpoints for gameplay (as opposed to polling)
    - Game handling (determining who is the next player, updating game state, etc)
- API Gateway
  - Need to make the endpoints accessible by the clients
- DynamoDB
  - Player accounts
  - Game history
- Route53
  - Manage the domain name for the project



Solution - Frontend

- **Game interface**
  - Apply Javascript & React along with UI components/libraries to provide a easy to interact user interface
  - Allow indicators during games to illustrate legal moves
- **Game performance**
  - Use WebSocket API for fluid interaction between backend<->frontend
- **Viewing gameplay**
  - Develop a Frontend<->Backend protocol



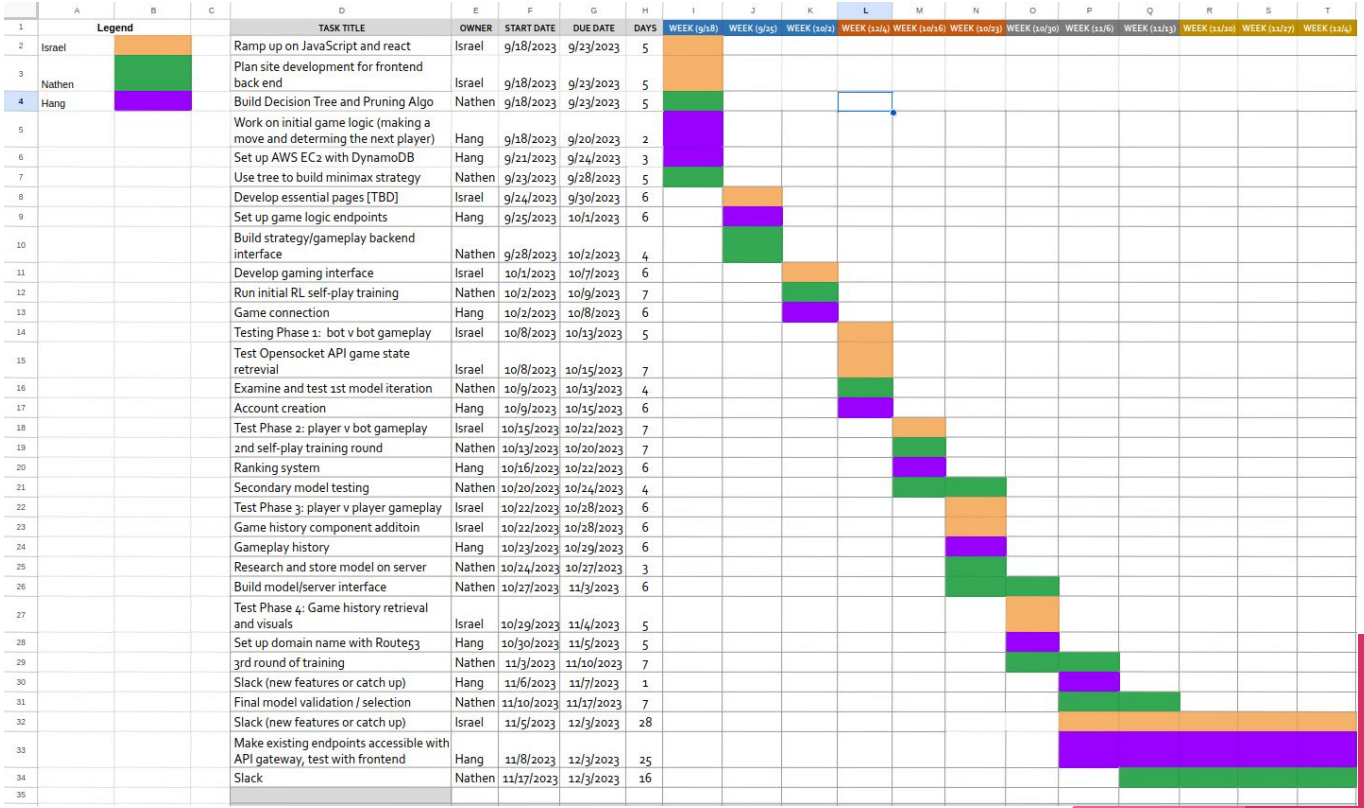
Testing, Verification, Metrics

- Test iterations of engines against each other to ensure later versions are improving (i.e. newer versions win at higher rates)
- Test final engine version against project team members to ensure it plays at a higher level than that of the average human. (beats team member >= 95% of the team)
- Unit test for each lambda function deployed for the backend
- Record response times from client to backend server communication

Division of Labor

Nathan	**Hang**	**Israel**
Engines & Reinforcement Learning	**Backend & Infrastructure**	**Frontend & Gameplay**
Build Initial 2-Ply Minimax Engine	Site Hosting Infrastructure	Websocket connection & backend protocol
Reinforcement Learning Setup	Game Logic & WebSocket Endpoints	Gaming interface development for users experience
Overseeing Reinforcement Learning & Making Adjustments	Player Information Endpoints (Ratings and Game History)	Display meta-data for games
Engine Testing	Backend Unit Testing	Frontend Unit Testing

Schedule



Legend		TASK TITLE	OWNER	START DATE	DUE DATE	DAYS
Israel		Ramp up on JavaScript and react	Israel	9/18/2023	9/23/2023	5
Nathen		Plan site development for frontend back end	Israel	9/18/2023	9/23/2023	5
Hang		Build Decision Tree and Pruning Algo	Nathen	9/18/2023	9/23/2023	5
		Work on initial game logic (making a move and determing the next player)	Hang	9/18/2023	9/20/2023	2
		Set up AWS EC2 with DynamoDB	Hang	9/21/2023	9/24/2023	3
		Use tree to build minimax strategy	Nathen	9/23/2023	9/28/2023	5
		Develop essential pages [TBD]	Israel	9/24/2023	9/30/2023	6
		Set up game logic endpoints	Hang	9/25/2023	10/1/2023	6
		Build strategy/gameplay backend interface	Nathen	9/28/2023	10/2/2023	4
		Develop gaming interface	Israel	10/1/2023	10/7/2023	6
		Run initial RL self-play training	Nathen	10/2/2023	10/9/2023	7
		Game connection	Hang	10/2/2023	10/8/2023	6
		Testing Phase 1: bot v bot gameplay	Israel	10/8/2023	10/13/2023	5
		Test Opensocket API game state retrevial	Israel	10/8/2023	10/15/2023	7
		Examine and test 1st model iteration	Nathen	10/9/2023	10/13/2023	4
		Account creation	Hang	10/9/2023	10/15/2023	6
		Test Phase 2: player v bot gameplay	Israel	10/15/2023	10/22/2023	7
		2nd self-play training round	Nathen	10/13/2023	10/20/2023	7
		Ranking system	Hang	10/16/2023	10/22/2023	6
		Secondary model testing	Nathen	10/20/2023	10/24/2023	4
		Test Phase 3: player v player gameplay	Israel	10/22/2023	10/28/2023	6
		Game history component additoin	Israel	10/22/2023	10/28/2023	6
		Gameplay history	Hang	10/23/2023	10/29/2023	6
		Research and store model on server	Nathen	10/24/2023	10/27/2023	3
		Build model/server interface	Nathen	10/27/2023	11/3/2023	6
		Test Phase 4: Game history retrieval and visuals	Israel	10/29/2023	11/4/2023	5
		Set up domain name with Route53	Hang	10/30/2023	11/5/2023	5
		3rd round of training	Nathen	11/3/2023	11/10/2023	7
		Slack (new features or catch up)	Hang	11/6/2023	11/7/2023	1
		Final model validation / selection	Nathen	11/10/2023	11/17/2023	7
		Slack (new features or catch up)	Israel	11/5/2023	12/3/2023	28
		Make existing endpoints accessible with API gateway, test with frontend	Hang	11/8/2023	12/3/2023	25
		Slack	Nathen	11/17/2023	12/3/2023	16