Assassin – 2018/19 Class Ref Proj.

- History – Assassin started as a game on paper around 2011 or 2012ish. Japhet's current website has been in operation for at least 3 years. Jon has created a js Assassin prototype.
- Player version - App with notifications by PorcFest 2019. Also web-based for computers. Free to download.
- Admin version – Can be an app only, website not required, but might be handy just in case. We should be able to charge for this app down the road. Admins can make good money running games.
- https://en.wikipedia.org/wiki/Assassin_(game)
- We should probably make generic (Not tied to squirt guns – More like Tag)
- Whova app – This was a fairly robust group in Pfest 2018 marketing. We market it as a reboot to Assassin, recruit some of the early players. Support the software team. More features, more fun, more rewards.

Assassin – Plan

- Documentation – Discovery, Requ, HLD, LLD, Code, Test Plans, Mgt. Plan – Use these as templates for Class.
- Decent cut of requirements done, HLD started.
- Review current prototype – basic functionality works.
- Data Model – Firestore – Collections and Documents
- Javascript website first. Risk some re-coding when porting to app.
- Research App frameworks (Several js frameworks available)
- Apps – Android and Mac
- Scope Approach – Agile, small batches of features. Expand from single game to multiple. Security.

HLD Considerations

- Website first, Research App framework in parallel. DB should be 100% reusable. JS code, we'll see?
- Limitations – Game has loopholes without notifications, timed re-starts are not possible in Japhet version, low bandwidth at Pfest especially with possible monopoly on wifi at Rogers.
- Offload work from admin to players
  - Pre-register, upload pic, respond to pings
- Point-or-bounty-based, don't mention money. Admin's responsibility to collect $ and activate players.
- Design – Closed loop, waiting queue. Explicit data structures vs. stored in Player.
- Event driven – Join, kill, re-buy, quit…
- Admin payout options – tips within the app, auto % tiered,

JS Prototype

1. Stored in Git Hub. This is a combined admin and players prototype game.
2. Login id tied to either player or admin. Logout and back in to switch between players and admin. See the global data at the top of the .js file for existing log ins (players array, admins array)
3. Live game data structures always shown at bottom so its easy to test out the functionality.
4. Some error checking, but not much.
5. Only uses a players data structure. The next target id s stored right in player object. There is no explicit chain or waiting queue construct in this prototype. These constructs can be derived from the player status.
6. Somewhat “React-like”. Action in game creates events, events are saved/logged, renderGame() is then called which redraws the screen based on the last event. Real database will have live, push data change notifications after events occur.

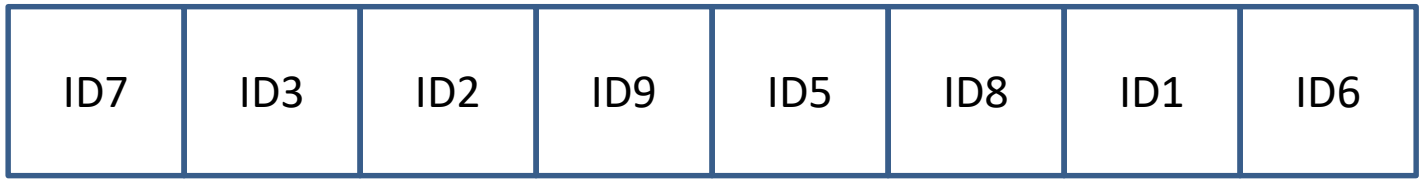
Firestore Status

1. Jon has had some success with small prototypes:
    1. Remote 2 action player game (Runny Game)
    2. Remote tic-tac-toe (Alternating Turns)
2. Firestore over Firebase
3. Pages – Videos (“Get to know Cloud Firestore #1” on You Tube - series)
    1. https://firebase.google.com/docs/firestore/
    2. https://www.youtube.com/watch?time_continue=3&v=v_hR4K4auoQ

Chain Scenarios

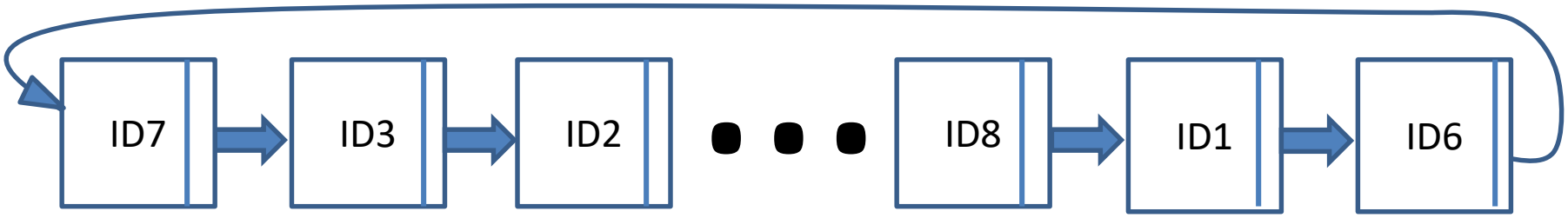
1. Start game, all players awaiting assignment
2. Simple kill, nobody waiting
3. Simple kill, players waiting
4. Break, nobody waiting
5. Break, players waiting
6. Scheduled Admin Force-Break
7. Bomb

Chain Options – Simple Array



- The target of each ID is the next element in the array.
- The target of the last element is the first element.
- Pros
  - Chain data structure outside of Player data, safer?
  - Easy update for single kill with no wait (remove the killed player)
- Cons
  - Changes to array will get pushed to every user. User will rarely need the change.
  - Updates for new player queue may cause many push updates (Insert new players in the middle)?

Chain Options – Linked List



- Each node in the list has the ID of a Player and a “Next” field that points to the Target node.
- The target of the last element is the first element.
- Pros
  - Easy update for single kill with no wait (remove the killed player)
  - Easy update for new player queue entering in the middle.
- Cons
  - More complex, more fields, more work?

Chain Options – Only in Player Data

- Each Player has a status. Status drives what other data the Player will currently have stored.
- If the player is live ina game, the Player's Target id is stored right in that Player's data record
- Pros
  - No additional data structures to manage
- Cons
  - Single "copy of chain". May be hard to rebuild if error condition.
  - May be harder/longer for basic operations because you may have to traverse the list more.

Assassin Competitors

- https://www.theassassingame.com/welcome
- http://www.tagmobilegame.com/
- http://www.lyteshot.com/

Rules Considerations

- Start and end day
- Start and end time of day
- Witnesses
  - Decline?
  - Max?
- Stun time