RENDERER

●●●

REFACTORING

Renderer document

PROBLEM

1) Scalability
   - High coupling
   - Not Fault tolerance
   - Hard to deploy closer to the business
2) Maintainability
   - The renderer is stuck with Ruby 2.2
   - There is a problem with the codebase structure changes
   - The API is not self documented
   - No test coverage for basic APIs
   - The codebase contains pieces that require deep refactoring

SOLUTION

1. Remove the Page logic from the App server and move to the Renderer server
2. Remote control of participants by host done via renderer path(muting/unmuting, interactivity)
3. Appearance(online state) of the host and participants is controlled by the heartbeat logic(moved to Mercury)
4. Migrate the Renderer server to new programming language (Python, Node, any non blocking I/O server)

ADVANTAGES

- Independent from App, only init set up and can be scaled independently with no dependencies.
- Fault tolerance. A demo between host and participant continues even the App server is down.
- Easier to scale from unpredictable load, more flexibility.
- Easy deployment to regions closer to businesses without any latency constraints of the main backend location.
- Better resource usage, since it’s all centralized to the single renderer server, without any dependencies.
- Introduce fixes and changes faster due to hierarchical structure and up to dated versions.
- Access to a larger base of packages.
- Get rid of outdated version of Ruby 2.2 and its packages.
- Better performance of the server.
- Auto generated documentation.
- Test coverage.
- The faster learning curve of the server.
- Access to larger base of engineers

QUESTIONS

1. How urgent it is?
2. How many sprints?
3. Solve bugs?
4. Simplify debugging?