**Annex C**

**Operation Barebones Project Requirements**

Required Features:

1. Multi-lingual support for all server and client rendered pages.

(Locale information is retained in user model and will be retrieved through user model together will privilege information. After that, if we are using handlebar as render engine, we can do i18n-node)

1. Responsive layout capable of supporting both mobile and desktop browsers with identical feature sets.

(Using bootstrap-based theme to support responsivenss)

1. Integrated CMS system that supports system admin creation of page routes, multi-lingual support for new routes through the CMS, and account/system management through the CMS. CMS can be developed as either a single page application, or as a server-rendered application.

(Keystone provide the admin control panel and support route creation)

1. Independent server node application that serves as the primary business logic control point for the infrastructure, and can scale horizontally in a virtual or physical environment.

(See the infrastructure picture)

1. Independent reverse proxy nodes that can forward web traffic to n-number of server nodes.

(See the infrastructure picture)

1. Independent and arbitrated database nodes that act as redundant data stores for all application data. (MongoDB cluster, etc.)

(See the infrastructure picture)

1. Clustered session stores that can be shared and accessed from all server nodes. This is to allow a single session to failover to a different web server in the event of a proxy failure or a web server failure.

(See the infrastructure picture)

1. Planned integration point for an external API set that will be accessed by all Tesseracts via AJAX GET/POST in order to upload performance data, emergency data, and alert/warning data.

(See the infrastructure picture)

1. Utilization of an ORM library to handle all database interactions. Recommend: Waterline ORM

(Plan to use Moogoose/waterfall, but still need to see how to integrate waterfall into keystone if we are going to purge moogoose)

1. Build entire application on top of the Node.js runtime to enable frontend/backend developer movement, and to commonize the programming language set used in the HI department.

(See the infrastructure picture)

1. Structure application design to support websocket access to all data models and system functions provided a user has proper access levels.

(I don’t see any reason we cannot integrate socket.io later on besed on express communication)

1. A single sign on page that branches user logins to either employee, customer, or systems administrator pages after login.

(Point 6 already support this, see the infrastructure picture)

1. SSL encryption for all data exchanges between web clients and web servers.

(We can use HAProxy to redirect all http request? Should that be a good practise)

1. Automated SEO optimization across all pages to enable better search engine crawling of the Symbolic IO website.

(Does slug good enough for SEO opt?)

1. Automated generation of a site map that allows users to travel to any route directly from the site map.

(I think Google webmaster provide a way for generate sitemap, which is not related to CMS election)