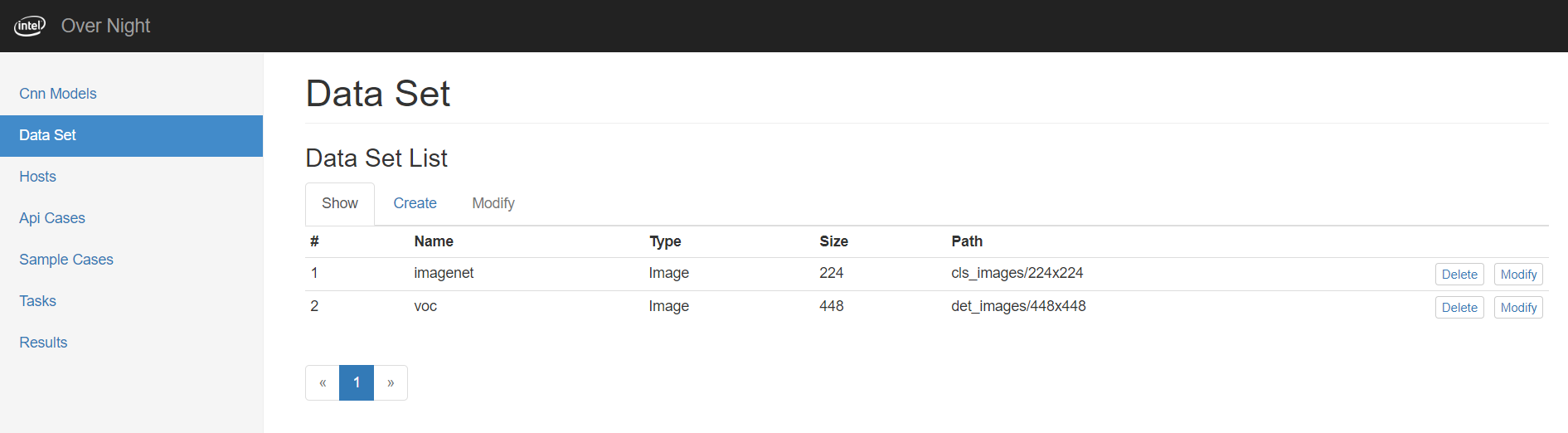
**HDDL 自动化验证框架设计开发文档**

1. Key Features (i.e. requirements):
2. 通过浏览器(Google Chrome)以操作Web界面的方式对测试用例进行管理（包括创建，执行，查询状态，统计等功能）；
3. 集成HDDL项目中的大部分自动化测试（overnight stability, 7x24 reliability, user case samples, HAL APIs, IE APIs…）；
4. Web界面要求简洁，直观，方便操作。
5. Web 界面设计

Web UI基于已有的OverNight项目：



如下几点需要新增/升级：

1. 左侧菜单增加为二级菜单，主菜单和子菜单分别为(暂定):

**Dashboard**

**Common**

|\_\_ Host Config

|\_\_ CNN Model Config

|\_\_ Data Set Config (可以和CNN model放到一起？ 待定)

**Overnight**

|\_\_ Test Items

|\_\_ Tasks

|\_\_ Results

**User Case Sample**

|\_\_ Test Items

|\_\_ Tasks

|\_\_ Results

**Reliability (7x24)**

|\_\_ Test Items

|\_\_ Tasks

|\_\_ Results

**HAL APIs (TBD)**

**IE APIs (TBD)**

各个菜单的用途：

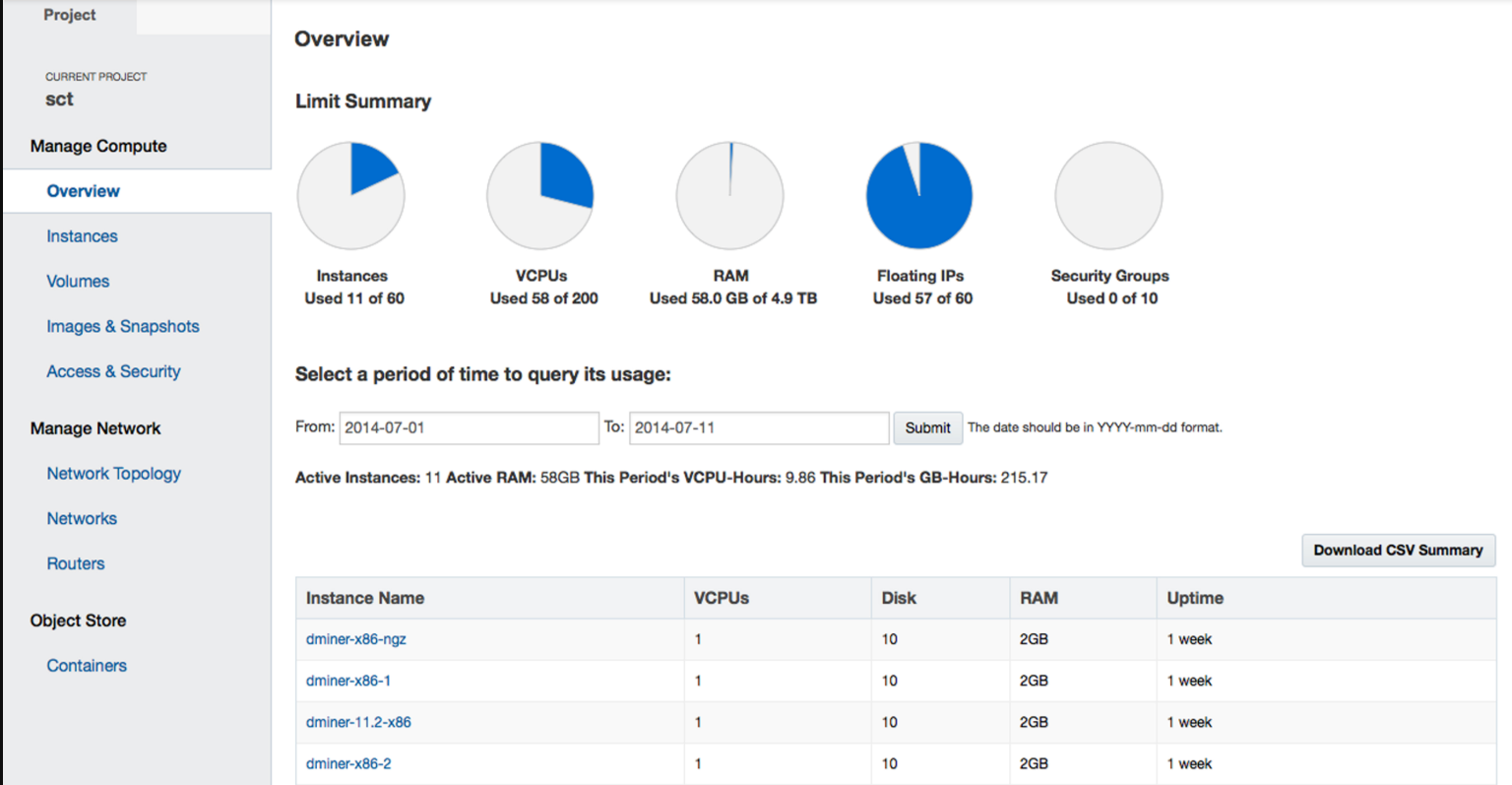
1. **Dashboard ：*核心菜单***，主要展示当前各个测试的一个总体概括，主要包括：
   1. 展示当前每个host上正在运行的测试信息（测试内容，测试状态，测试时间..）
   2. 以图表的方式展示每类测试结果在不同维度(by ww/model/host…)的历史统计信息 （refer to vega-lib）。
2. **Common:** 每类测试需要的通用数据的配置放在这里，包括host, CNN models, Data set的菜单.
   1. Host Config: 提供HDDL host的 CRUD (create/retrieve/update/delete)
   2. CNN Model Config: 提供CNN models (IR & graph) 的 CRUD
   3. Data Set Config： 提供 数据集的CRUD
3. **Over Night**: Overnight test 的所有操作放在这里:
   1. Test Items： overnight test 的配置 (IE+[sync/async/async+callback]+[hw/sw] , HAL+[sync/async/async+wait/async+callback])
   2. Tasks: 每一次用例的执行被当作一次任务, task 的添加在此执行。Task = overnight\_test + host\_ip + duration\_time(default:12h) + workweek + status + create\_time/finish\_time…)
   3. Results**:** 
      1. list任务，支持对未运行任务进行修改/删除操作。
      2. workweek, test\_name, status, 运行起止时间等字段要显示在页面。
      3. 默认显示当前workweek的tasks, 支持查询其他的workweek的tasks 。
4. **User Case Sample**：HDDL 的3个samples test, 于Overnight test 类似.
5. (TBD)
6. (TBD)
7. …..
8. 如下字段在在页面上采用下拉列表的方式展示，不用手工输入：
   1. Os type (linux/windows)
   2. test\_level (IE/HAL), test\_type(sync/async/async+callback)
   3. duration time (8h/12h/24h/7x24h)
9. Tasks 菜单中支持按照workweek**批量**创建任务
10. 数据库设计

原则上不同类型任务之间数据库表相互独立。



1. Dashboard页面设计

参考：



上半部分是各种统计图表， 下半部分是当前正在执行的各种Task。

1. 关键流程
2. 周期性Monitor 进程

在Task 执行后需要一个monitor进程来监控任务执行的情况，主要是通过定时扫描数据库的\*Task表，根据task的状态和其它字段的值对任务进行不同的处理。具体包括：

* 1. 首先对Running状态的task, 1)查看当前执行的情况是否正常 2)判断执行时间是否已经满足要求（current-time – start\_time >= plan\_execution\_time）, 并根据结果执行必要的操作（例如，结束任务，获取异常日志 …）以及更新必要的字段（例如 status， actual\_execution\_time, finish\_time ..）
  2. 其次对于New状态的task, 查看对应的host是否已经空闲，如果空闲则开始执行启动任务操作已经更新必要的字段.

1. 开发环境以及第三方依赖