File Based Test Management Service

**Problem**: Managing test collateral around plans, requirements, ideas, sessions, cases, data, and results is difficult. Existing database solutions force structure on test collateral that is limiting. UI around existing solutions is slow to use, stifling output. Most solutions exist in a walled garden system requiring per use licenses.

**Solution**: A file system based collateral management solution that allows flexible storage of test collateral in a manner consistent with just about any application for creating and viewing material, while also offering easy reporting, collection, and aggregation of testing activity.

Requirements:

1. Testing collateral can exist in any format from any application
2. Testing collateral lives in the file system. There will be a root directory, and then all other collateral will reside from that root and lower.
3. Testing collateral is described by testing manifest files which point to the actual collateral
4. Testing manifest files use a schema that supports the following constructs
   1. Projects: Describes a project that is to be tested. Projects include the following properties
      1. Name, start date
      2. Area hierarchy – a breakdown of the features within the project
   2. Test Strategy: Describes a set of testing plans for a given project
   3. Test Plan: Describes a set of test collateral which articulate the plan for testing some part of a project
   4. Test Charters: Describes a specific area of testing activity, the mission for that testing activity, and refers to the documents with more details on the charter.
   5. Test Session: Describes a time-bound event which executed on the charter, with references to any notes or reports produced during the session. Test sessions have as attributes the charter they cover, the date and time and length of the session, the person executing on the session, and revision number of the product under test if relevant.
   6. Test Session Report: Describes a document that contains a summary of activities and learnings in a testing session. Test session reports align their metadata with the test session they were created in.
   7. Test Suite: Describes a collection of test cases.
   8. Test Cases: A collection of specific testing ideas, frequently associated with a test character, that describe what to test.
   9. Test Status Report: A summary of testing for a given project which summarizes the key focus areas.
5. There will be a main test manifest file which describes the project overall. Multiple projects may exist in the same directory. Other test manifest files may also exist within the file directory. Every new manifest is considered additive to the main test manifest file for the project. Each manifest file will indicate which project it is a part of.
6. The service will provide an API for discovering and reporting on the collateral in the test management service.
   1. getProjects() – returns an array of all the projects in the system
   2. getProject() – returns a description of the project, which includes a tree of all of its features
   3. getTestStrategies() – given a project, returns a list of test strategies associated with the project
   4. getTestStrategy() – given a test strategy and a project, returns the test strategy object
   5. getTestPlans() – given a project, and optionally a test strategy, returns all test plans associated with that project or filtered to the strategy
   6. getTestCharters() – given a project, and optionally a test strategy or test plan, all the test charters associated with that project
   7. getTestSessions() – given a project, and optionally a test charter, returns all the test sessions associated with that project and filtered to that charter. Also allows filtering based on date and production revision.
   8. getTestReports() – given a project, and optionally a test session or test charter, returns all the test reports associated with that project and filtered to that session or charter. Also allows filtering based on date and production revision.
7. Web-based UI allows users to interact with the test management service
   1. Allow users to browse areas, strategies, plans, charters, sessions and reports for selected projects.
   2. Allow users to search for testing collateral based on keywords, tags, date, product version, and author. Allow filtering based on collateral type.
   3. Provide a settings page where the users can configure service behaviors
      1. Add directories where the user can browse to the root directory from which to discover manifest files.