Trillium Health Grant Management Design Document

Version: Final
Prepared by: Team Ulysses
10/22/2014

Table of Contents

Introduction

Trillium Grant Management System will standardize the process for Trillium Health. This is limited to notification of grant status, grant information, task creation, task reminders, and document upload, revision and approval. Grant status shall be communicated to the user by use of the dashboard by showing the status of the current tasks on a grant. Grant information shall be stored in the system and will be able to be view by viewing each individual grant. The system will allow for tasks to be created for grants. The system will notify users when a task is coming closer to its due date. The system will allow for upload of documents that are related to the grants. The system will keep track of previous revisions of documents so documents can be turned back to previous revisions. The system will allow for a process which requires grant owners to approve of documents before they are finalized.

System Overview

The system overview contains a general description of the functionality, context and design of the project. The overview should only briefly describe these aspects and the comprehensive explanations will be done in the sections to follow. The overview should serve as an introduction to these sections.

The purpose of this grant management system is to unify grant workflows for collaborators and to enable collaboration on grants.

The scope of our project can be broadly divided into the following concerns:

- Tasks and Workflows
- Documents
- Role Management

Tasks and workflows encapsulate recurring grant work such as preparing for meetings, aggregating reports, and so on. Documents are the deliverable artifacts for each grant lifecycle. Role Management is a necessity because users may need to deal with sensitive information such as budgets and CVs which should not be available to *all* users.

The approach we have taken to deliver a grant workflow and collaboration solution is to create a web application that can be deployed in Trillium's intranet. Since we (the development team) are most familiar with Java EE and Rails and Ruby on Rails may not work as well in a Windows environment, we opted for Enterprise Java.

The grant management system will be deployed on a Windows 2008. However, the individual developer's environment may be Windows, OS X, or Linux, and the continuous integration environment is Ubuntu 12.04. Two of the varying technologies in these environment are the database and identity stores. The table of the technologies used for every environment is given below:

Purpose	**Environment**	**Storage**	**Identity**
Development	Windows, OS X, Linux	HyperSQL	UnboundID
Staging (QA)	Ubuntu 12.04	PostgreSQL	OpenLDAP
Production	Windows Server 2008	PostgreSQL	ActiveDirectory

System Architecture

The high level architecture of the system serves a blueprint for this project and provides a way for developing this application. Major quality attributes such as performance, usability, availability, and security were address to make sure that the design yields an acceptable system.

<<High-level-design diagram goes here>>

- Client
  - Browser
  - {See Client Design}
- Application Layer
  - Services
  - DAO
  - Model
  - LDAP
- Data Layer
  - JPA/Hibernate
  - {See Database Design}

The architecture consists of 3 main components, the client layer, the domain/application layer, and the data layer.

The Client Layer - The user will have an access to the application through a browser. The acceptable browsers are Chrome, Firefox, and Internet Explorer. This will be an Angular JS application that will send and receive information to and from REST interfaces in the application layer.

The Application Layer - consists of three major modules (Services, DAO, and Models). This layer is uses the Spring Framework to integrate everything together.

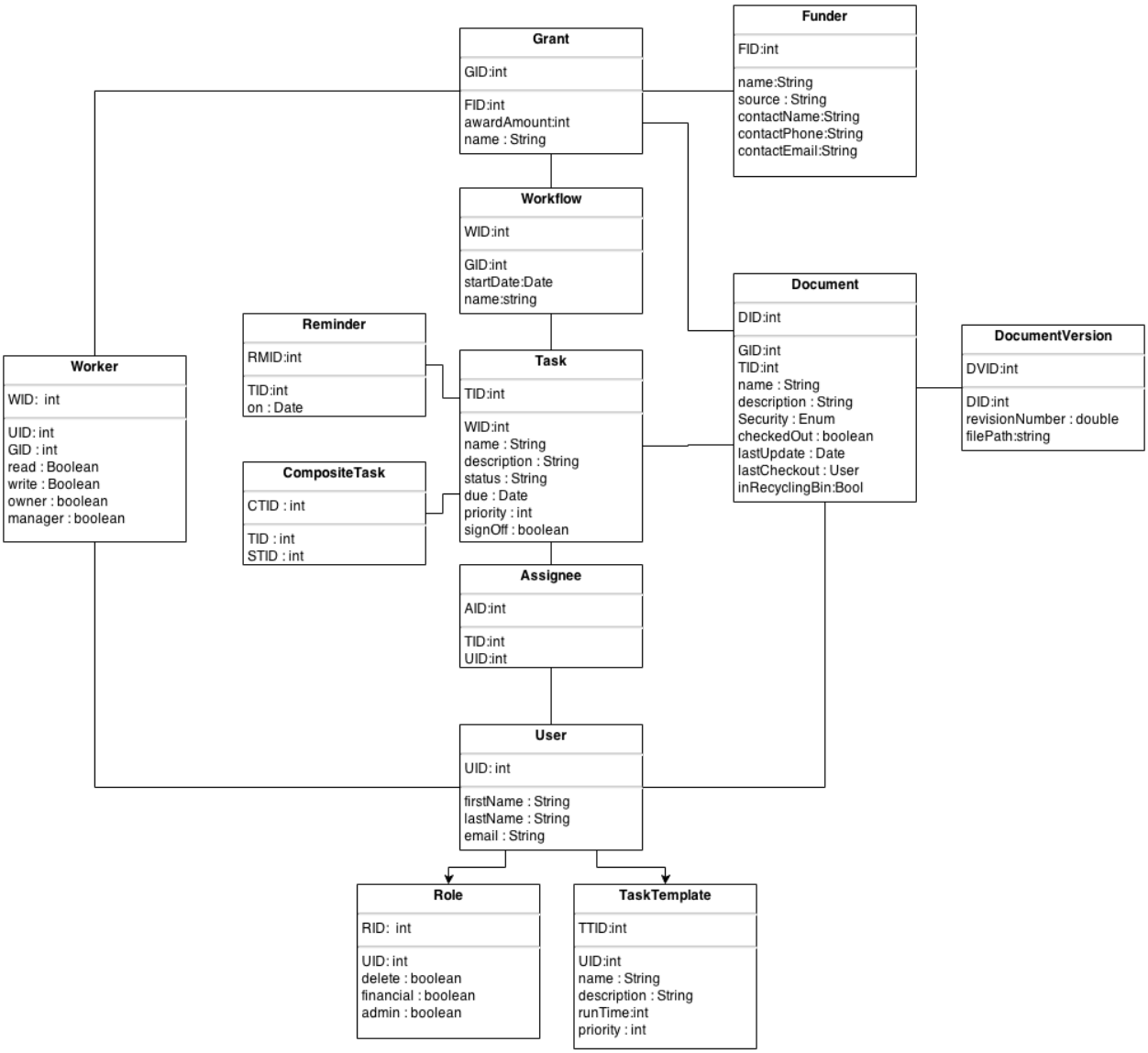
> **Services** - are REST interfaces that the client will send information to. After processing the request, it will send information back to the user in a JSON format. This module has a Jersey REST package dependency that will be used.
>
> **Data-Access-Object (DAO)** - these are the data objects that the services will use to interact with the database. The DAO will perform the basic CRUD operation plus additional operations that are needed to query from the database. DAO module will be able to use LDAP and interact with that database to get the user account information and authenticate them using Active Directory.
>
> **Models** - are the data components with fields that will be used for anything within the application layer. They will have the JPA and Hibernate annotations that will allow the application to persist data objects to the database.

The Database Layer - this is where all the grant information will be stored, along with the documents and any other relevant information pertaining to this application.

**Tactics for each of the quality attributes mentioned above are still be considered, therefore, the application architecture will evolve with the system.

Database Design

Our database is relational and uses 3rd normal form.



Component Design

In this section, the functioning and responsibilities of each component of the design are going to be described in detail.

Composite Pattern

In order to manager the nested nature of the tasks we will use the composite pattern, which allows each Task object to have references to other Task objects below/within it. The Major and Minor Task objects will also keep track of other Task related information such as due date, and description (see Class Diagram).

- **Task** = a Task object is an interface used to describe both MajorTask objects and MinorTask objects.
- **Major Task** = A MajorTask object is one that include a reference to 1 to many subtasks.

- **MinorTask** = a MinorTask object or “Leaf” is a task with no tasks below it.

Memento Pattern

The memento pattern is used when “snapshots” of an object’s current state want to be saved for possible reference. Here, we use the pattern to control document versioning.

- **Document** = this object holds a reference to the actual file we are currently working with, as well as other pertinent information like revision number and tags.
- **Version** = This is the snapshot of the document, which keeps track of the state of the Document object when it asked to create this version of Memento of itself.
- **CareTaker** = the CareTaker keeps track of all the different Version objects as they are created over time. It maintains them all so that any one of them can be referenced or reverted back to in the future if necessary

Strategy Pattern

We use the Strategy Pattern in relation to roles and permissions of users. Each Role represents a set of permissions allowable in the system. Each User will hold an instance of a Role object using one or more of the following permissions “strategies”. This will allow for easy adjustment of permissions.

- GrantWorker
- GrantOwner
- Technical Admin
- ReadOnly/Auditor
- Grant Manager
- Financial Analyst

Factory Method

We used the Factory Method pattern for the creation of Workflows to get rid of some overhead when they are created. Certain workflows can be designed with tasks already in place to help with the grant process. When creating a workflow, the WorkFlowFactory will create all of the necessary tasks “behind the scenes”.

- **WorkFlow** = an interface of the different kinds of workflows
  - Research
  - CFA
  - Reapply
  - Audit
- **WorkFlowFactory** = creates the specific workflows

Other

- **Reminders** = Reminder objects manage the frequency and type of reminders for Task objects. Task objects each hold one Reminder object.
- **RecyclingBin** = handles a list of documents that have been flagged for deletion. These documents can then either be permanently deleted or restored by an Administrative user.
- **Grant** = keeps track of general information surrounding a grant such as funder, grant amount, etc.
- **Funder** = keeps track of general information surrounding a Funder such as name, phone number, etc.
- [LDAP]
- [Proxy]

Human Interface Design

- *This section will contain the complete information about how the user interface of the software will work and how it will look like.*
- *The functionality of the software from the user's perspective should be described*
- *It should be properly explain how the user will access all the features being offered by the software*
- *Explain how the feedback information will be displayed for the user.*
- *The information in this section should be accompanied with proper images showing how exactly the designer visions the interface to be like. The images can be hand-drawn or can be draw with the help of some software.*

Use Cases

<< link to use case diagrams >>
<< link to use case descriptions >>

Concerns

Our sponsors have a few usability concerns that directly relate to quality attributes in our interface. Those concerns are:

- Users are not technically inclined, thus potentially not familiar with some browser tools such as the back button.
- Users are not technically inclined, thus may have certain "ticks" such as double-clicking everything
- Users may get "lost" in the website
- Users may not know or forget what an icon means or what a button does

Addressing these concerns leads to less support phone calls for IT.

Conventions

To address some of the usability concerns, we decided to use a few existing design tactics to reduce the likelihood of those concerns becoming a reality.

Since users can easily get "lost" on the website such as when exploring or accidentally clicking on a stray link, every page should be navigable back to the homepage and every page should show a "trace" or breadcrumb trail of what part of the website the user is in, such as "Home → MyGrant → Audit → MyTask". This tactic also addresses the concern for not knowing about the back button since there is always a way to navigate back to where you previously were

Since users are also not technically inclined, another mode of failure would be if they clicked on something and there is no feedback or the feedback isn't apparent, even if the underlying system does actually do something. To address this,

- Creation and update of items (tasks, workflows, grants, documents, user permissions) are done through a modal widget. If all creation and update goes through a modal dialog, then users will form a mental model where a modal indicates that some action is happening.
- When the creation is completed, a dismissible dialog appears confirming the create or update of the item. With this convention in place, users will have feedback for any action that they do. When some action is successfully completed, the dismissible dialog will appear and detail the action that has been performed.
- One-step updates (such as deletion and in most/all cases, completion), can skip the modal step.
- "Dangerous" operations such as deletion, change of permissions/owner, << more >> should have a confirmation dialog before committing to the action, since there is a possibility the action is accidental.

- There are multiple ways to do one action. For example, tasks may be updated on the task detail page, or in the dashboard detail. This mitigates the risk that users won't be able to find the one page or part of a page that performs some action and is not available anywhere else.
- In our casual testing, we can try to see if anything undesirable happens when links, buttons, or any other actionable item is double-clicked.
- All buttons, icons, and informational components will have alt/hover text that will remind the user of what something does. Since users may not be aware of this feature, this should be covered in training as a "helpful trick".

One of the main user workflows is to update some task or document related to a grant and workflow (CFA, Audit, …). A convention that was proposed to us and we will use is to have a "dashboard" that gives high-level detail about each grant, but can "drill down" into the grants' workflows, and again into a workflow's tasks.

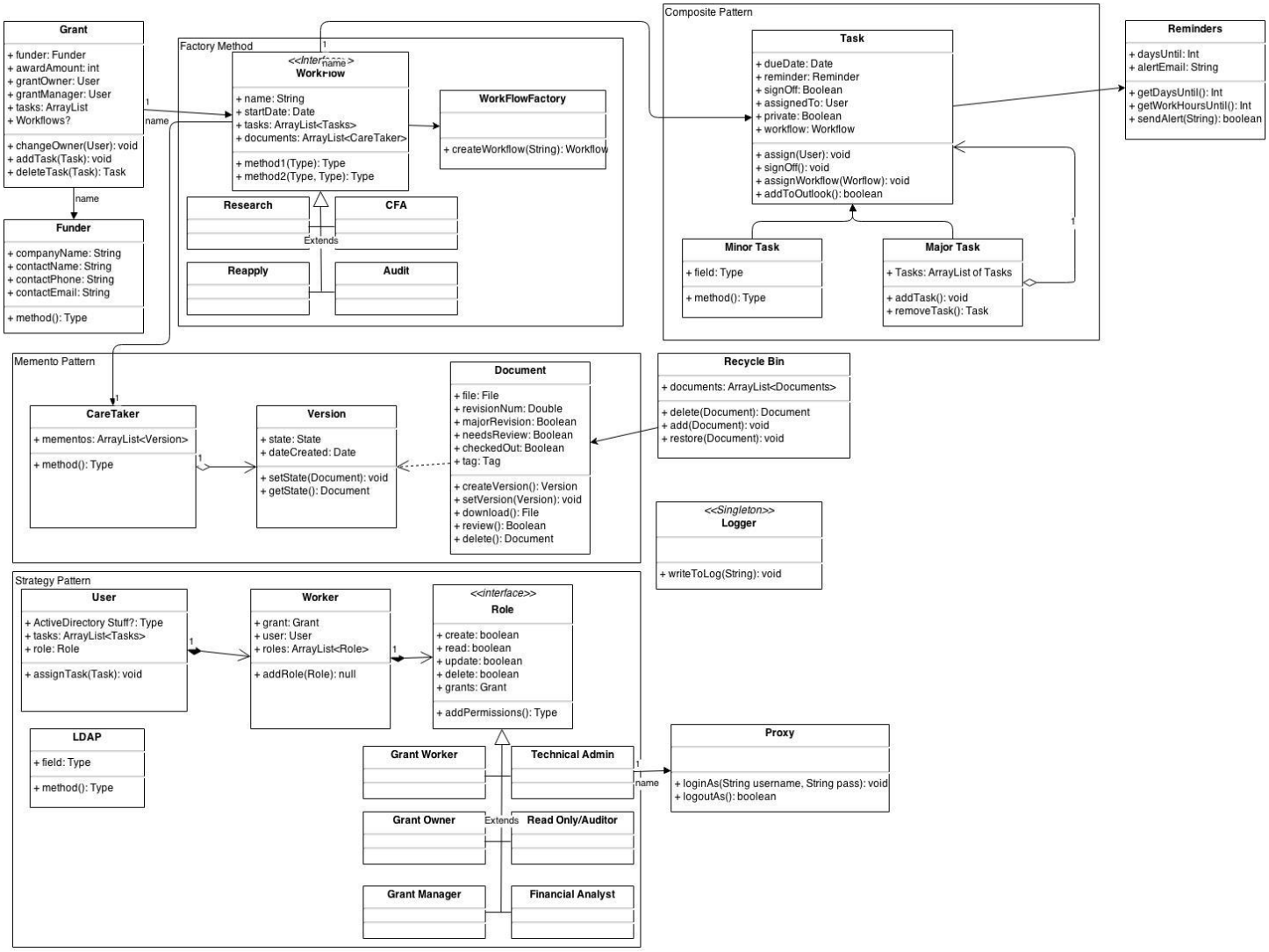
This "drill-down" dashboard will take the form of a master-detail page, where one the master panel shows the "drill-down" and the detail panel shows more detail about the currently selected drill-down item. Frequent actions such as marking a task complete and creating a new task may be performed directly from the master-detail page. In addition, there will also be a "management" page which will provide access more infrequent actions in addition to the frequent actions. This addresses the "multiple ways of performing an action" concern.

Appendices

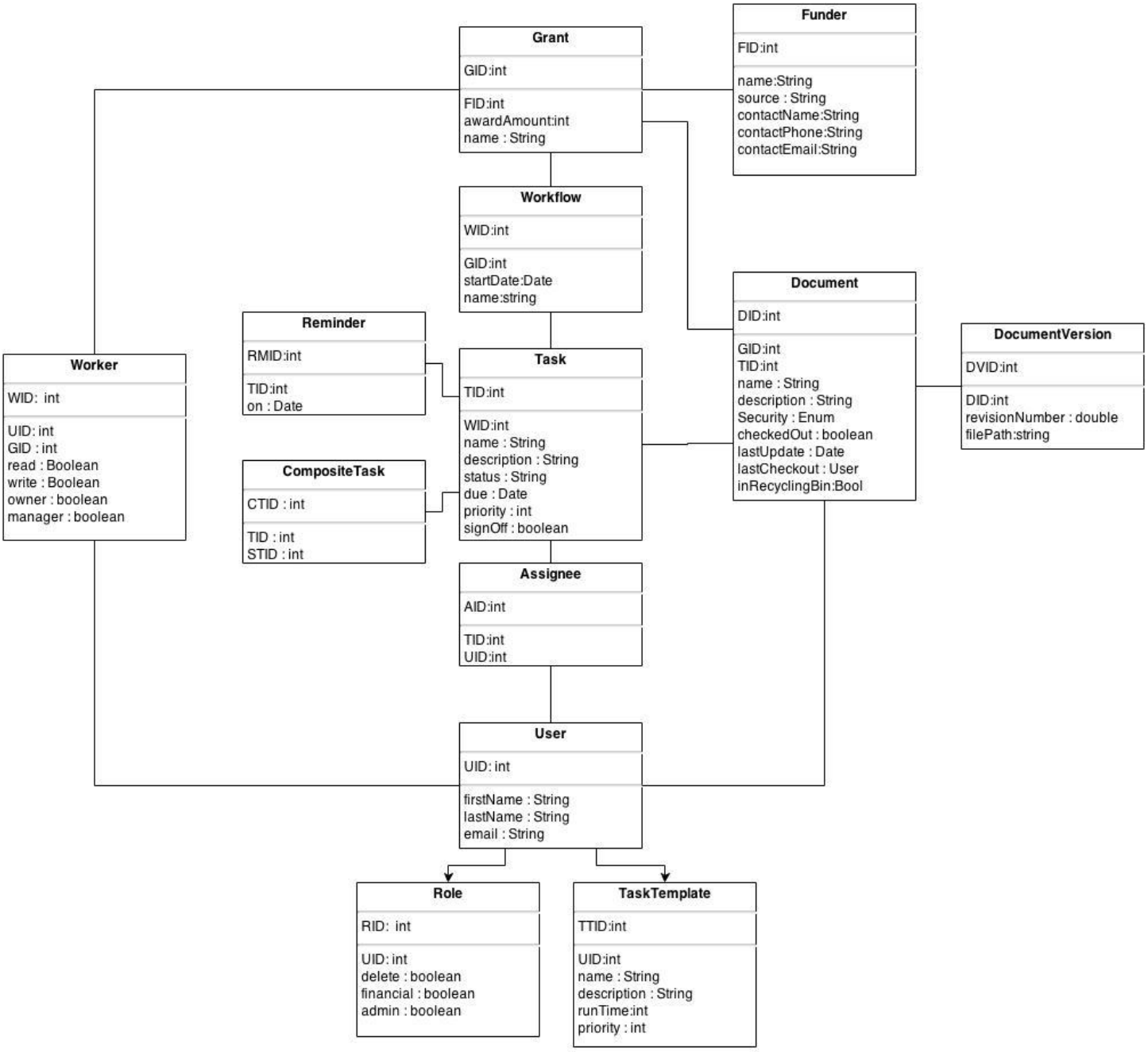
This section is optional and can be included if the need be. Kinks or references to supporting documents can be provided in this section which will help in the better understanding of the concept of software development.

- Long Term Planning Document - http://www.se.rit.edu/~ulysses/artifacts/Team_Ulysses_SPP.pdf
- Requirements Doc - http://www.se.rit.edu/~ulysses/artifacts/Requirements_Document_Final.pdf

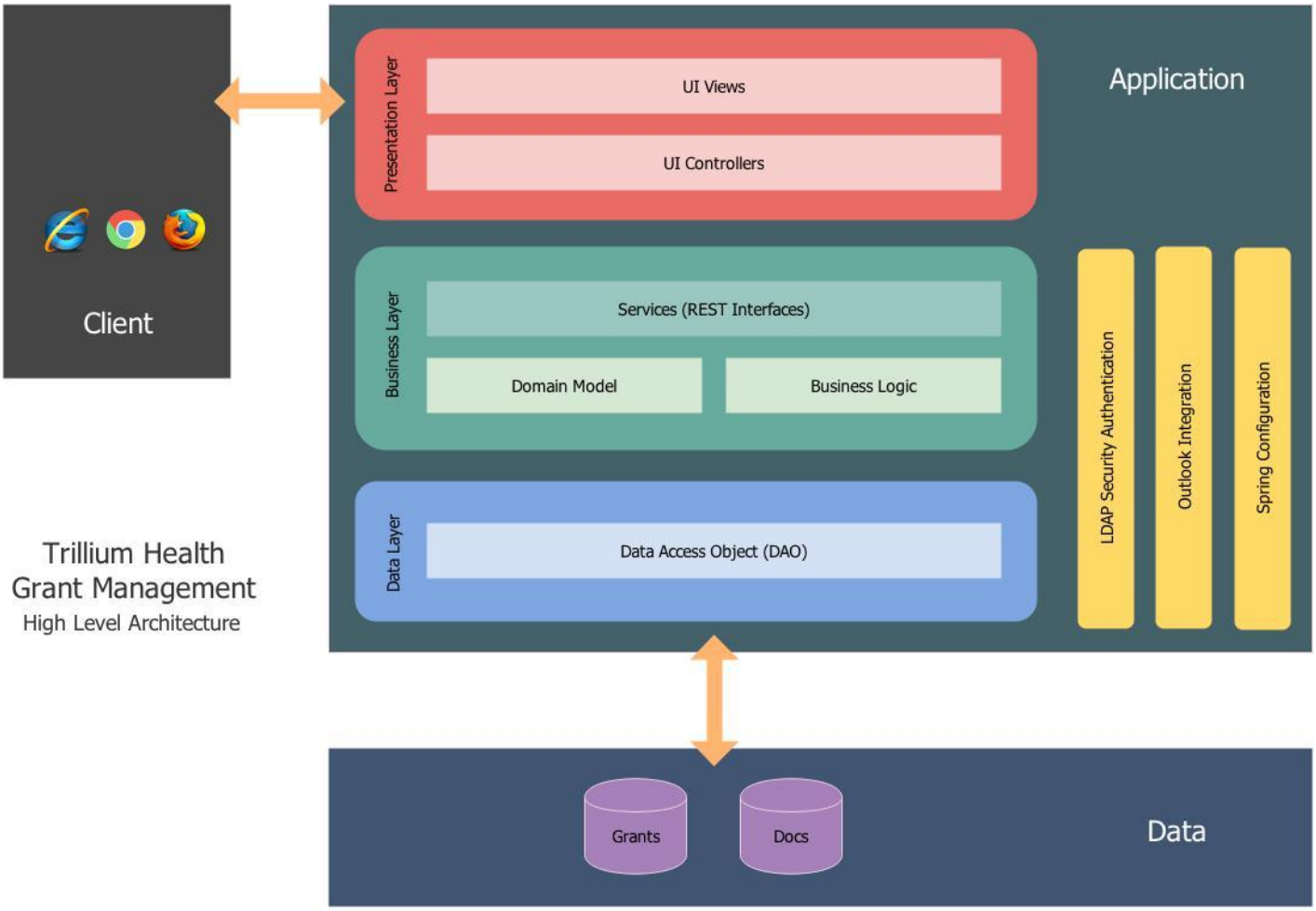
- Class Diagram



- ER Diagram (Database)

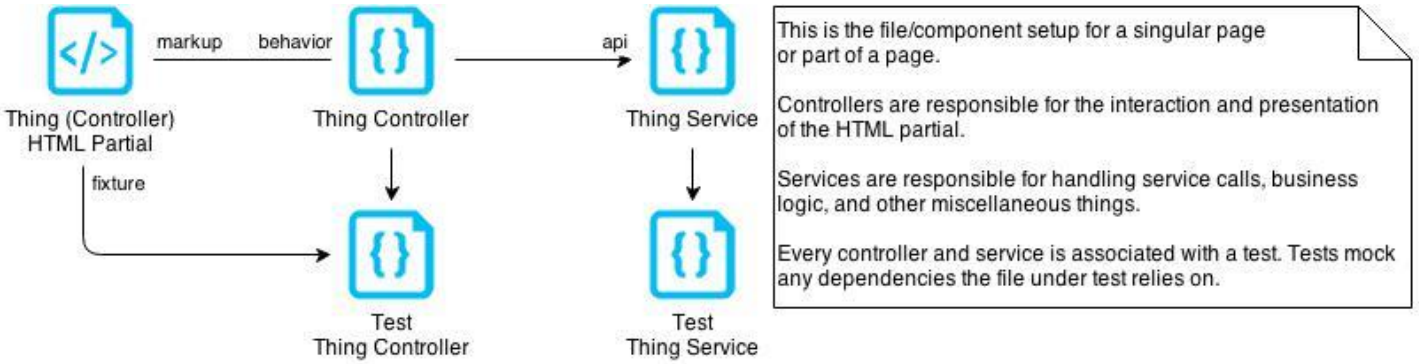


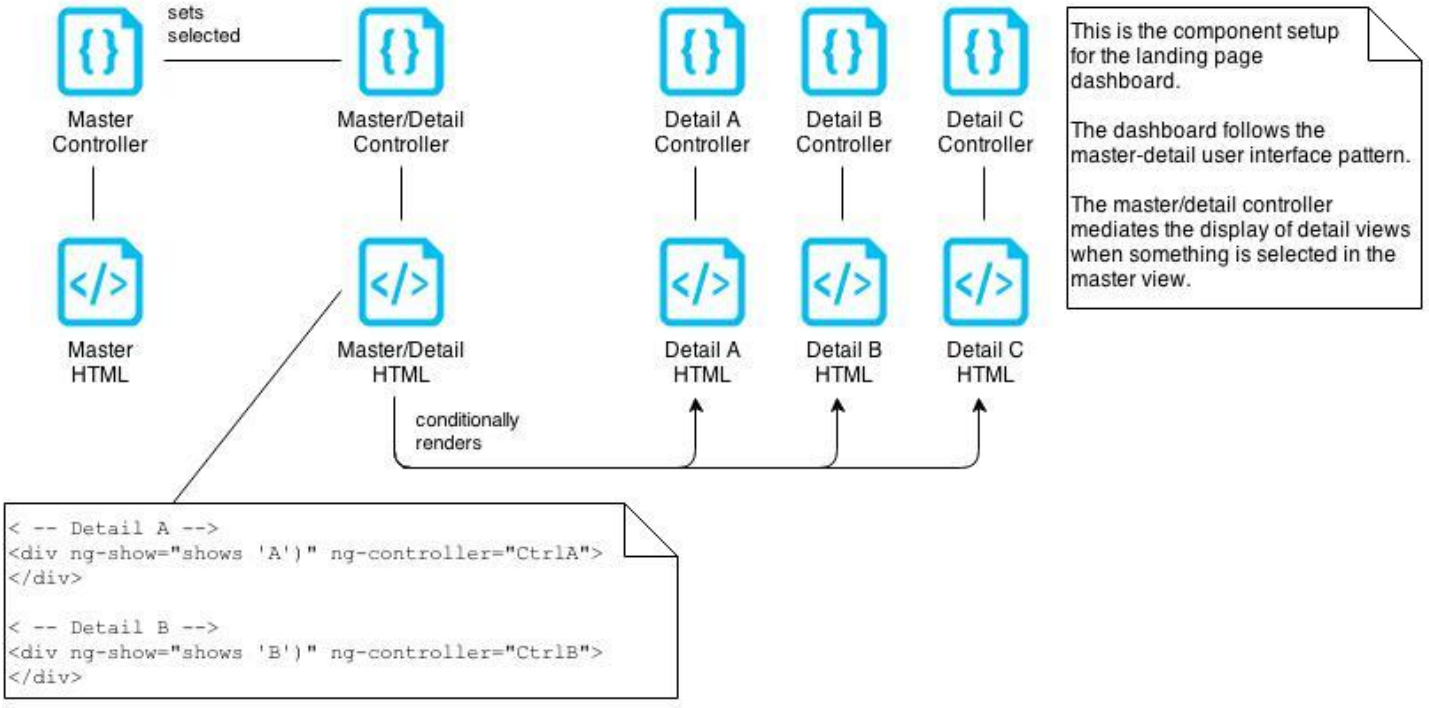
- System Overview



Trillium Health
Grant Management
High Level Architecture

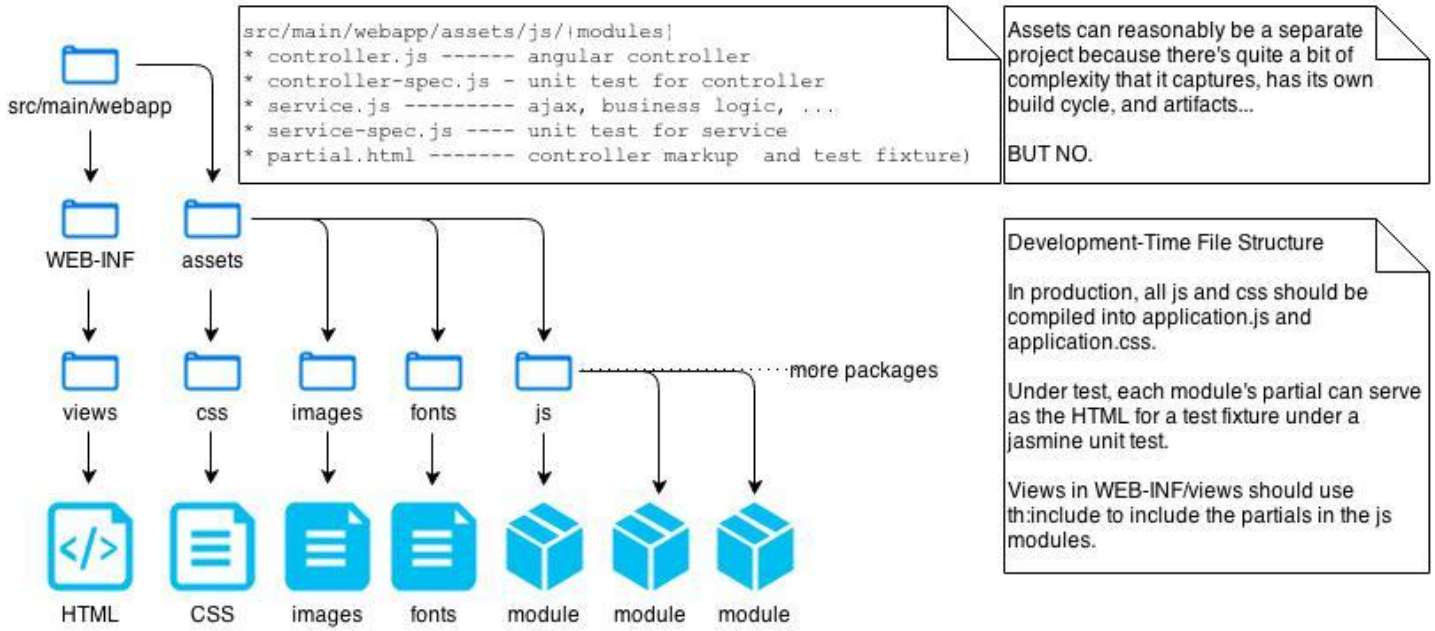
- Client Architecture



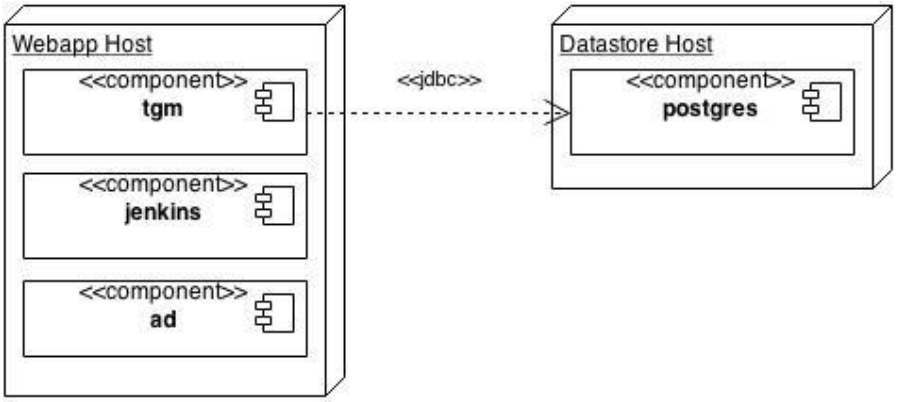



```
< -- Detail A -->
<div ng-show="shows 'A')" ng-controller="CtrlA">
</div>

< -- Detail B -->
<div ng-show="shows 'B')" ng-controller="CtrlB">
</div>
```



- Deployment



RIT Software Engineering, ulysses.se.rit.edu

Trillium Intranet