**Milestone 2**

**Counter-Logic**

**11/7/2016**

**Viet Lai**

**012140464**

**Martin Garcia Ripoll Munoz**

**016158296**

**Greg Paolo Violan**

**011706641**

**Daniel Sanchez**

**Digital Storage Solution**

* **Introduction:**
  + Background:
    - The current services that the university uses (Dropbox, PeopleSoft CMS) are outdated and expensive to maintain. Dropbox is extremely expensive to keep a school wide subscription. PeopleSoft has been in use since the mid 90s and looks like it. By having students develop their own solution that combines these two services into one, the university is able to cut labor costs down to almost 0.
  + Purpose:
    - Provide a backend service to store and manage data from students and faculty. Handles data for ~50,000 people.
* **Components:** 
  + Data Storage: 4
    - Can store all types of data (mostly text documents)
    - Can store student data, campus data, buildings data (location, rooms, ...), etc.
    - Add/Delete/Modify data
    - Be able to export data as csv and tabular report formats, etc.
    - Able to bulk insert, update, and export in any format
  + API: 1
    - Provide a platform for app development
    - Modeled after Google Firebase API.
    - Should have a clear and thorough reference for each command.
    - Extremely flexible, compatible with other components.
    - Allow administrator limitations on certain commands for security purposes.
  + GUI/User access: 3
    - Provide a friendly user interface on 3 platforms : Web, mobile app, and computer app.
    - Options isolated by user type (student, faculty)
  + Enrollment Management: 2, database
    - Replace PeopleSoft CMS features such as:
      * Managing enrollment
      * Student financials
      * Campus Analytics
* **Requirements:**
  + Accessibility:
    - Online access (web)
    - Mobile, tablet, any PC device
    - iOS App only, no android compatibility
    - Must be GUI intuitive
    - Important events must be shown to admins as pop-ups
    - Self-service password restore
    - There should be a way of communication between faculty and students (announcements, email messages, ims, etc.). It should have a log of the messages.
  + Data Management:
    - Must be able to import 500,000,000 records from paper
    - Must be able to query data (able to sort)
    - Student storage has a lifespan of 5 years
  + Security:
    - Must be secure against brute force
    - Must have a log system/audit trail system (never removed, 10,000 logs kept)
    - Session should expire
    - There will be only 3 tries for logging in
  + Maintenance
    - Must be able to roll-back (undo some actions)
    - Must be expandable, can add 100 million records every year (exponentially)
    - Must have key performance indicators
    - Must have system notification during maintenance
    - Should be automatically backed up regularly.

**Conditions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Precondition** | **Postcondition** | **Fail** |
| Store new data | - Storage isn’t full | - Storage has data | - Prevent upload  - Notify user storage is full |
| Update data | - Data is present  - Storage isn’t full | - Data is updated | - Notify user that data isn’t present |
| Add course to schedule | - Class isn’t full  - Student has instructor consent (if required) | - Student is enrolled in class | - Prevent enroll  - If class full, notify student.  - If no consent, then tell student to get consent |
| Remove course from schedule | - Student is enrolled in class | - Student is no longer enrolled in class | - If student is not enrolled, notify student. |
| Get homework | - Instructor has uploaded homework | - File downloaded to student computer in original format | - If no homework  available, then notify student. |
| Enter grades | - Student is enrolled in class  - It is near the end of semester | - Grade is entered | - If student isn’t enrolled, display warning. |
| Give consent | - none | - Student is allowed to enroll in class | - Confirm student and retry. |
| Classroom bulk export to csv | - none | - CSV file generated | - Confirm data and retry |
| Classroom bulk import from csv | - Space available | - Classroom data is stored | - Warn about lack of space  - Confirm data and retry |
| Upload homework | - none | - Homework is uploaded in original file format | - Confirm data and retry |
| Add class | - Class does not already exist | - New class is added | - If class already exists, warn  - Confirm information and retry |
| Remove class | - Class exists | - Class is removed | - If class doesn’t exist, warn  - Confirm and retry |
| Add user | - User does not already exist | - New class is added | - If user present, warn then confirm input |
| Remove user | - User exists | - User is removed | - If user doesn’t exist, warn. |
| Add module | - Module with same name doesn’t already exist | - Module is added | - If module with same name exists, prompt user to input new name |
| Remove module | - Module exists | - Module is removed | - If module doesn’t exist, notify user. |
| Database backup | - It is the scheduled time | - Data from web servers and database is copied to the backup server | - Notify administration and verify data for corruption. |

- The precondition that the current user has permissions to do the action is assumed.

**Project Planning**

**Budget:** ~$718,390

**Time:** ~2100 hours

**Start date:** January 1, 2017

**End date:** November 26, 2017

**Strategy:** Iterative, Spiral

**Software Estimates**

Developer Comparison:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Hourly Salary** | **Cost/40hr** | **Bug expense/40hr\*** | **Total Cost/40hr** |
| Junior | $28 | $1120 | - 10 bugs  - 5 critical bugs  $2500 | $3620 |
| Mid-level | $34 | $1360 | - 7 bugs  - 3 critical bugs  $1600 | $2960 |
| Senior | $40 | $1600 | - 5 bugs  - 1 critical bugs  $1200 | $2400 |

\* 1 bug = $100

\* 1 critical bug = $300

- Hiring anyone but senior developers is simply not cost efficient. Lower level developers may have a cheaper base rate, but the costs they incur through bugs raise their overall value dramatically.

- **2100 estimated development hours** required.

- Includes design and testing.

- Comes from the minimum of 1400 hours + 50% error margin

- Renegotiation possible during each review phase.

- Hiring **4 senior engineers** produces **40 development hours** of work per week.

- It would take **45 weeks** to complete the 2100 total estimated hours.

- A single senior engineer costs **$108,000 total** for **45 weeks** ($2400 \* 45)

Total developer pay

- $108,000 \* 4 = **$432,000.00**

**Hardware Estimates**

Hardware Design Philosophy

- Start with overkill, then scale back as project moves forward

- Use pre-built units as examples before designing custom hardware

**Data/Service Server**

Requirements

- MySQL database handles user administration (users, campus information)

- Holds data for the web servers.

- Transfer data from server to server.

- Web server requests user information from it.

- Needs high memory capacity (~2TB).

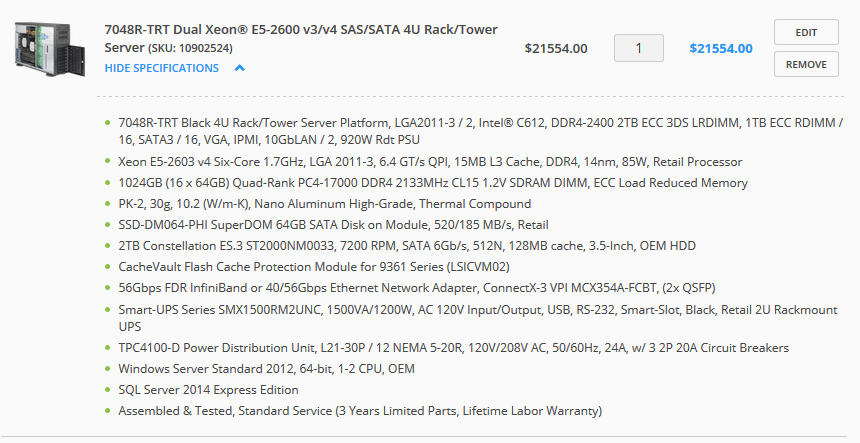
Preliminary Unit

- Pre-built server from AVADirect.

- Projected cost of current build = **$21,554.00**

**-** Need 6 data servers and 5 service servers, totaling to 11 servers

- Overall cost = **$237,094.00**



**Web Server**

Requirements

- Handle user load of ~10000

- Uses Nginx software to load balance

- High performance 64-bit processor

Preliminary Unit

- Pre-built server from AVADirect

- Projected cost of current build = **$3,081.00**

- 56Gbps network adapter to handle large load.

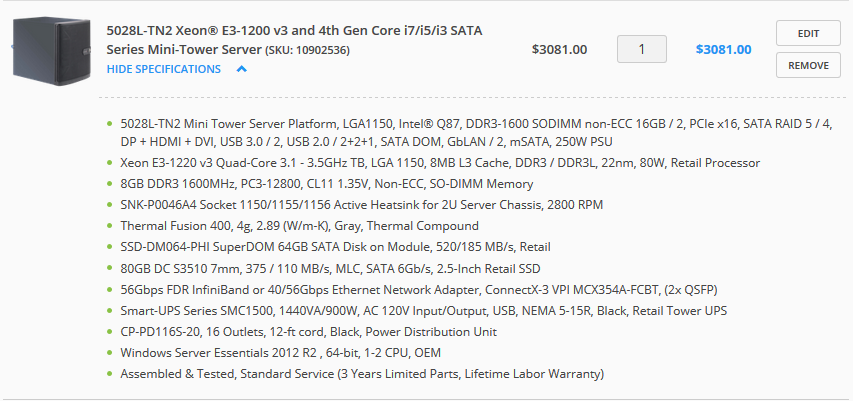
- Average upload speed in USA is ~6Mbps

- Can handle ~9300 people uploading at that speed

- Worst case scenario is 100k students uploading simultaneously, therefore at   
 least 16 units are required, 4 are back-ups.

- Storage synchronized across all 12 units, each server has copies of the same data

- Total cost = **$49,296**



Total hardware cost

- **$43,685**

**Total Costs**

Software Development - **$432,000**

Hardware Design - **$286,390**

Total - **$718,390**