**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    - The current services that the university uses (Dropbox, PeopleSoft CMS) are outdated and expensive to maintain.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.
* **Roles:** 
  + Student
    - Enroll and drop courses
    - Store and download their own data
    - Download homework
  + Faculty
    - Can view class list
    - Allow students to enroll in their class
    - Enter grades
    - Upload homework
  + Department chiefs
    - Adding courses
  + Administrator (system administrator)
    - Maintenance
      * Solve software and hardware issues
    - Manage users (students, faculty, department heads)

**Conditions**

**Student**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 class | - 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 class | - 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. |

**Faculty**

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Precondition** | **Postcondition** | **Fail** |
| 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 |

**Department Head**

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Precondition** | **Postcondition** | **Fail** |
| 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 |

**Admin**

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Precondition** | **Postcondition** | **Fail** |
| Add user (student, faculty, head) | - 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. |

**Responsibility Matrix**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Public** | **Student** | **Faculty** | **Dept. Chief** | **Admin** |
| **Log in** | X | X | X | X | X |
| **See buildings info** | X | X | X | X | X |
| **Read/Write messages** |  | X | X | X | X |
| **Post announcements/News** |  |  | X | X | X |
| **Manage buildings info** |  |  |  |  | X |
| **Modify profile** |  | X | X | X | X |
| **See student data** |  | X | X | X | X |
| **Manage grades** |  |  | X | X | X |
| **Manage classes** |  |  |  | X | X |
| **Monitoring System Access** |  |  |  |  | X |
| **See log info** |  |  |  |  | X |
| **Export data** |  |  |  |  | X |
| **Backups** |  |  |  |  | X |
| **Import data** |  |  |  |  | X |
| **Roll-back** |  |  |  |  | X |
| **Block users** |  |  |  |  | X |

**Project Planning**

**Budget:** ~$624,000

**Time:** ~2670 hours

**Start date:** January 10, 2017

**End date:** January 28, 2019

**Specifications:** 3

**Strategy:** Iterative, Spiral

**Estimations**

* Time estimation:

Best Average Worst

( 1600\*1 + 2400\*2 + 3200\*3 ) / 6

= 2666.66 hours

* Budget estimation:

→ Total Budget = data cost per unit \* 500,000,000 records + service fee

+ other hardware necessities

* Data cost per unit = 50 Kb ( average document ) \* 500,000,000 records

→ 25 billion Kb = 25 Tb + allowance of 100 Tb = 125 Tb

→ 1 Gb = 15¢, 1000 Gb \* 15¢ = 1 Tb = $150/month

= 125 Tb \* $150/month

= $18,750/month

* Service fee = specifications \* estimated time \* hourly wage

= ( 3 \* (2700 \* 50) )

= $405,000

* Other hardware necessities = ~$200,000

= Total Budget = $18,750 + $405,000 + $200,000

= $623,750