Requirements

Version 2.0 | December 10, 2013 | CS 476

The Aviators

Charles Chavez

Dillon Postage

Mark Malone

Hasini Wickramasooriya

Table of Contents

Introduction

In this document, we are going to cover what our basic problem is, give a simple explanation for fixing it, and sponsor expectations. After the introduction, we will go over how our system will work. This will include basic concepts our system will apply, like user management, processing data, and the computation process.

After that, we are going to dive in to the Functional, Non-Functional, and Environmental requirements. The functional requirements goes over our basic strategy in terms of theories for handling users. The Non-Functional requirements will talk about the limitations of the server in terms of process-ability, reliability, and verifiability. The Environmental requirements will talk about our software and hardware use.

Finally, we will cover the potential risks of our project, how we can avoid them, and our project plan. This will be a section outlining status. After this there is reference material for any unknown terms in the Glossary, and an Appendix listing the sections.

Problem and Solution Statement

The Sponsor and Challenges

Our sponsor is True Course Simulations, and they face a number of challenges in regards to their pilot training course.

Our Sponsor

The program is a 3D Flight Simulator program from the company Lockheed Martin called Prepar3D. It is at http://prepar3d.com. Their goal is to train pilots using a simulation before putting them in an aircraft. This should alleviate training time and cost for potential new pilots. We have two points of contact: Raynald Bedard is the sponsor contact, and Dr. Arjomand Kalayeh who is the CEO.

Challenges of the organization

True Course Simulations is facing the problem of presenting the statistics of users advancing through the lessons in the course. Without a central system for the users of the training program, tracking progress becomes difficult. They also no way to provide access information to institutions.

Addressing the challenges

By using our software system, users will be able to login and view their progress in the modules needed for safe aircraft piloting. The progress points can be send at any time, so we will have a server running constantly to process the requests. Once they are processed, the users will be able to view the progress on a web site.

The System

Diagram

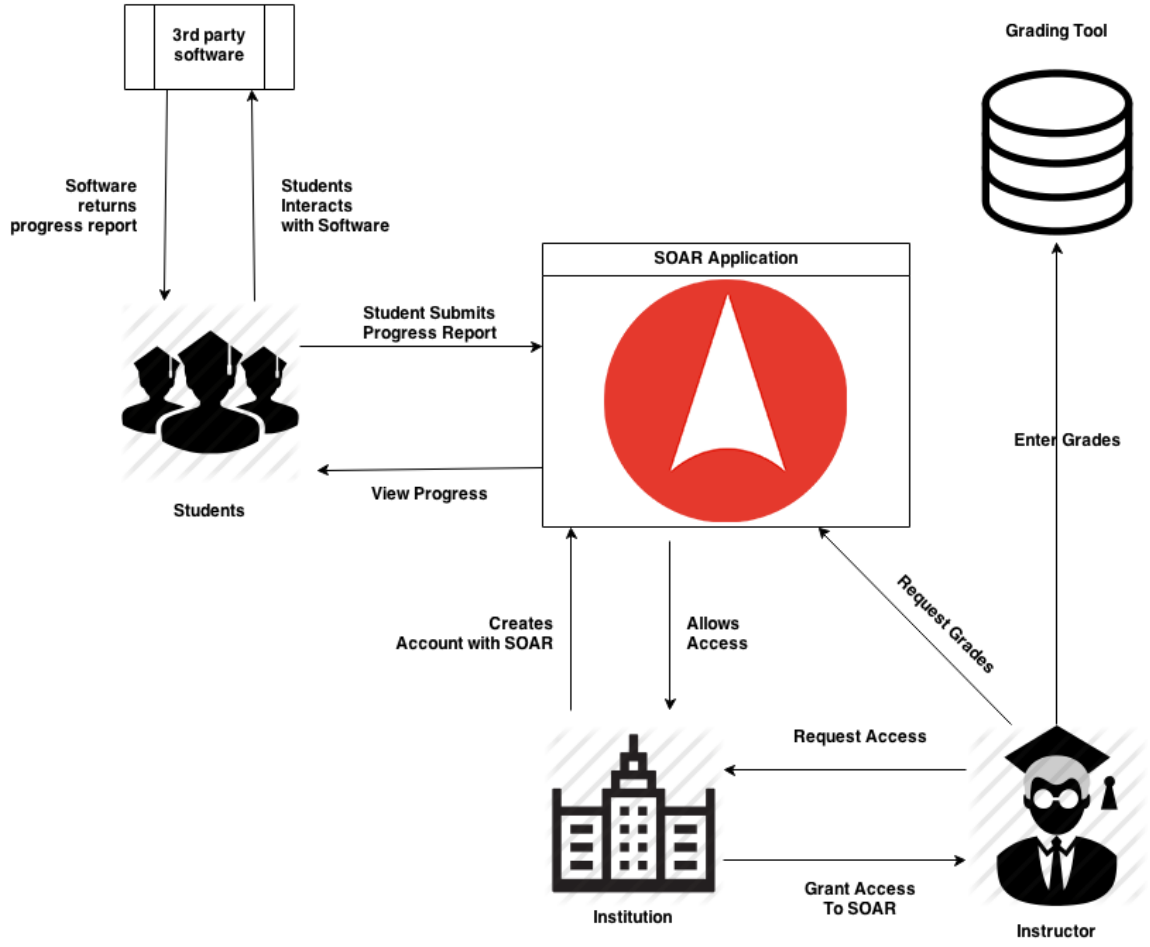

Figure 1—A graphical overview of how our web application interacts with clients and 3rd party software as institutions and instructors.

User Management System

The User Management System is secure way to view course progress. There are three types of accounts or privilege levels. There are System Administrators, Group Administrators, and Students. The System Administrators are allowed to create Group Administrators (this is like a School allowing Teachers to teach students). The Group Administrators can enroll students into their created “groups”. (This is like a teacher allowing students to join the class). Finally, the Students can view their courses that they are enrolled in from the Group administrator, and their course progress which is saved

into a database from the Tin Can API calls from the Prepar3D simulation software. These groups will also be able to change their “look and feel” by a custom CSS.

System Data

Our system will save data into a MySQL database. We will define a table structure with relationships according to the type of events completed, and save them. Data our system uses comes from the Tin Can API calls from the Prepar3D software. Course progress modules display information based on the database information

Computational Processes

Main computation processes are going to be when the web application makes calls to the database to retrieve information, save information, and sending information. There will be concern whenever Students are checking their grades concurrently. The grades are parsed on the fly, so that will matter on concurrent connections to the database server from the Prepar3D software. Finally, generating reports will be optimized so the queries will not bog down the database server.

Sponsor Adaptation

The way the system is developed, the sponsor should be able to use the same software for multiple learning sources. As an example, if a software or lesson is created that wants to be saved and reported to the learning record store, they just need to implement the Tin Can API into their software, and save configuration information that points to the web server.

Impact and Other Solutions

This software can help people track progress in courses specific to aviation. The institution can easily set up their own learning environment for individuals, and add on to their working business model.

Other solutions

We have been given the direction of the Tin Can API and we must be able to interpret it using modern web technologies. This will be JavaScript Object Notation to SQL translating.

Functional Requirements

1. User Management System

This mechanism will manage the roles of individuals on our web application.

1.1. User Levels

Individuals who access the web app will have different levels which allow them to access different functionality.

1.1.1. System Administrator

An individual who is in charge of assigning institution administrators to our web application.

1.1.1.1. Should be able to log into the web app.

1.1.1.2. Should be able to manage institution administrator accounts.

1.1.1.3. Should be able to manage other system administrator accounts.

1.1.2. Institution Administrator

An individual that can create new student accounts and set up the course environment.

1.1.2.1. Should be able to log into the web app.

1.1.2.2. Should be able to manage student access to the web app.

1.1.2.3. Should be able to manage the look of the course environment for the institution.

1.1.2.4. Should be able to manage instructor access to the web app.

1.1.3. Instructors

Individuals who are allowed view progress and grades for students enrolled in a course.

1.1.3.1. Should be able to log into web app.

1.1.3.2. Should be able to access a list of courses that they are instructing.

1.1.3.3. Should be able to manage student access to the web app.

1.1.3.4. Should be able to access progress of students enrolled in the course.

1.1.3.5. Should be able to export information of students.

1.1.4. Student

Individuals who can access available courses and upload their progress to specific course lessons. They can also view statistics of their attempts in their courses.

1.1.4.1. Should be able to log into the web app.

1.1.4.2. Should be able to upload progress reports to selected course.

1.1.4.3. Should be able to review progress in course that they are enrolled in.

2. User Authentication

The mechanism that will secure the web app to only legitimate users.

2.1. Registration

System that students will use to authenticate their account.

2.1.1. Web Registration Portal

The means through which students register.

2.1.1.1. Should be able to list institutions supported by the web app.

2.1.1.2. Should be able to select courses that the institution provides.

2.1.1.3. Should be able to limit registration to only individuals who attend the university.

2.1.2. Registration Verification

Only individuals who meet the guidelines can register for the web app.

2.1.2.1. Should only allow students with a university email to register.

2.1.2.1.1. Should send activation code to individual's university email that will allow them to complete registration.

2.1.3. User Profile

The information for a user will be stored for our web app.

2.1.3.1. Should take the individual's personal information: first name, last name, university email, etc…

2.1.3.1.1. Should store the information received through registration into a user database.

2.2. Login

System responsible for granting session access to the web app.

2.2.1. Web Login Portal

The means through which a user will be authenticated into the proper user level.

2.2.1.1. Should be able to log in with their university email and password.

2.2.2. Login Redirection

Based on the information provided in the web login portal, the user will be redirected to correct dashboard.

2.2.2.1. Should look up user profile based on the email provided.

2.2.2.2. Should use password to match with the user profile.

2.2.2.3. Should redirect users to appropriate dashboard based on user level.

2.3. Access Control

System that determines whether or not a user has access to the requested resource.

2.3.1. Should allow a resource to define a level of security.

2.3.2. Should actively check a user's level when taking an action.

2.3.3. Should present an error when a user does not have the proper access rights.

3. Dashboard

The mechanism that ties together the various services provided by the web app.

3.1. User Level Differentiation

The dashboard should be able to differentiate between different user levels based on the user authentication.

3.1.1. System Administrator Panel

This dashboard should allow system administrators to manage the web app.

3.1.1.1. Should be able to add institution administrators to the web app.

3.1.1.2. Should be able to access all features available to all user levels.

3.1.2. Institution Administrator Panel

This dashboard allows institution administrator to manage an institution in the web app.

3.1.2.1. Should be able to add instructors to the institution.

3.1.2.2. Should be able to create courses for the institution.

3.1.2.3. Should be able to assign instructors to the institution.

3.1.2.4. Should be able to change the look of the institution portal.

3.1.3. Instructor Panel

This dashboard should allow instructors to manage courses and students enrolled.

3.1.3.1. Should allow instructors to manage students to course.

3.1.3.2. Should allow instructors to view and export student progress.

3.1.4. Student Panel

This dashboard should allow students to submit and review progress.

3.1.4.1. Should be able to upload progress files to the web app.

3.1.4.2. Information displayed should be derived from progress data.

3.1.4.3. Should be able to view measurements based of their progress.

3.2. Progress Representation

The dashboard should be able to display progress on each lesson.

3.2.1. Overall completion of the individual lesson.

The lesson should display the progress made towards the full completion of a lesson.

3.2.2. Overall completion of the course.

The dashboard should display the progress made towards the full completion of the course.

3.3. Lesson Selection

Student should be able to select a specific lesson within the course.

4. Data Management

The mechanism that manages the data associated with the web app.

4.1. Institution Data

This is how we store the institution data.

4.1.1. Institution should tie information of courses and teachers together.

4.1.2. Should maintain data associated with the look of the institution.

4.2. Course Data

This is how we store course data.

4.2.1. Should tie lessons and students together.

4.2.2. Should contain lessons available in a course.

4.3. Student Data

This is how we store student data.

4.3.1. 3rd party software will broadcast information on the progress of students through the Tin Can API.

4.3.2. The data stored should indexed according to the student, the time the data was acquired, and lesson taken.

Environmental Requirements

We have two major environmental requirements. The first is the use of the Tin-Can API, and the second is that the system must be a web application.

The Tin-Can API is a standard for receiving information that is gathered from user activity modules. It restricts our application by limiting us to the API provided, which will affect the way we read student progress data.

The requirement for the system to be web-based is a limitation on technologies that we can use to develop the application. With this restriction, we are limited to the types of languages and frameworks that we will use during development.

Non-Functional Requirements

The non-functional requirements are going to include a couple categories that we will use to benchmark the project. There is the measurement of time taken to process the web pages, reliability of our web application to store data and retrieve data, recoverability of the web applications state in case of failure, and integrity of information provided by the web application.

The time taken to serve web pages is how the software will deliver results to users. This includes having low delays during heavy traffic on the web application, as well as retrieving data efficiently during high demand.

Reliability is measured by whether the web application can be accessed to write or read. Reliability will make sure the data that is saved is valid, and test cases can be made to check data validity.

Recoverability is the ability to return to a functioning state if something were to go wrong. This is achieved by creating backups regularly. The backups will be stored to separate environments that can be cloud-based, or on secured off-line hard drives both on site and off site.

The integrity of the information provided by our web application should be enforced by following strict guidelines. Any information that doesn't follow the guidelines will be rejected by the web application.

In the end, we want to be able to test every aspect of the web application. If we can set this in place now, we can avoid potential problems in the future.

Potential Risks

Every piece of software can have many risks associated with it, while ours does not have any risks that would be considered safety critical, it does have some minor risks associated with it. If the web app receives or interprets data incorrectly, there are two prominent effects. One effect of this is that the student may not be able to receive the data they want, in this case they may not get to certain milestones they were trying to, such as 60% completion. The other side effect of this can be if the user receives the data they want but the data is incorrect, in this case a student may think that they are further along or even done with a course they have not finished. These two effects can also effect the entire system from the admins point of view.

If the admin relies on our web app to track progress from enrolled students, the admin could interpret the data in both the same ways as the student does, in that they can think a student is not as finished, or finished with a specific lesson. This could lead to an overall system failure in that it is no longer achieving its goal of reporting progress to users. This system failure can lead to students passing who should not and students failing who should not.

Outside of the effects that cause complete system failure, there are the minor inconveniences that can occur from time to time. Server failures can cause an interruption in service, while this can be seen as a system failure because the system is no longer accessible, we will assume disruption in service based on server errors will be fixed relatively quickly which would then only cause a brief interruption in service, thus an inconvenience at most.

Below we have a risk analysis which analyzes certain risks that may affect our system.

Risk	Risk Severity	Primary Cause	Mitigation Strategy
Loss of data	High	Server or database instability	Regular data backups to remote server
Server down		High network traffic or system vulnerability	Application-state backups and remote/local access to server
Malformed Data	Medium	Progress File does not meet file syntax	Only accept proper data and notify user of failure
Improper Reporting		Misinterpretation of data received	Strict enforcement of file syntax and strong unit testing
Web Application Errors	Low	Many causes, human to application errors	System Administrators equipped to handle variety of problems

Project Plan

Our project plans consists of 3 large milestones which provide functionality for the main components of the system. Each one of these milestones contains smaller milestones underneath symbolized by letters. A project plan overview is shown in the Gant Chart below which had to split up due to size, which is supported by a slightly more detailed description below it.

1. General website functionality
    a. Login
        i. Database queries
    b. Different user levels
        i. Different views and/or functionality based on the type of account logged in
    c. Group admin view changing functionality
        i. A different view for students under different groups

        ii. Edited by the group admin
    d. Functional dashboard
        i. Provide some understanding of the Tin Can API
            1. Attempt to receive fake data through the Tin Can API so that we know exactly how to handle it
        ii. Accurate and efficient navigation
2. Reporting functionality
    a. Data processing
        i. First ensure that we know how to receive Tin Can-like data
    b. Begin the process of attempting to receive data from the third-party software
        i. Correct interpretation of this data
        ii. Efficient interpretation of this data
        iii. Reporting of this data to separate users
    c. Award badges and Achievements based on progress.
        i. Interpret the data and look for certain cases where users deserve specified achievements or badges
3. Unit testing
    a. Website functionality testing
        i. Ensure no bugs exist on website
    b. Data processing testing
        i. Make sure that functionality works as intended
    c. Traffic Stress testing
        i. Ensure the system will not crash under high traffic situations/