

Virtual Classroom (Vlass)

High Level Design

COP 4331C, Fall, 2015

### Modification History

Version	Date	Who	Comment
V 0.0	10/15/2015	J. Casserino	Initial Draft
V 0.1	10/27/2015	J. Bender	Rough Draft - Added Design Architecture image
V 0.2	10/28/2015	C. Armstrong J. Bender J. Casserino M. Friedman	Rough Draft - Added Design Issues - Added Interfaces
V 1.0	10/29/2015	J. Casserino	Final Draft - Proof read and error checking

Team Name: Group 26

Team Members:

Joseph Bender      [jbender94@knights.ucf.edu](mailto:jbender94@knights.ucf.edu)

Joshua Casserino      [Joshua.casserino@knights.ucf.edu](mailto:Joshua.casserino@knights.ucf.edu)

Chad Armstrong      [chad.armstro@knights.ucf.edu](mailto:chad.armstro@knights.ucf.edu)

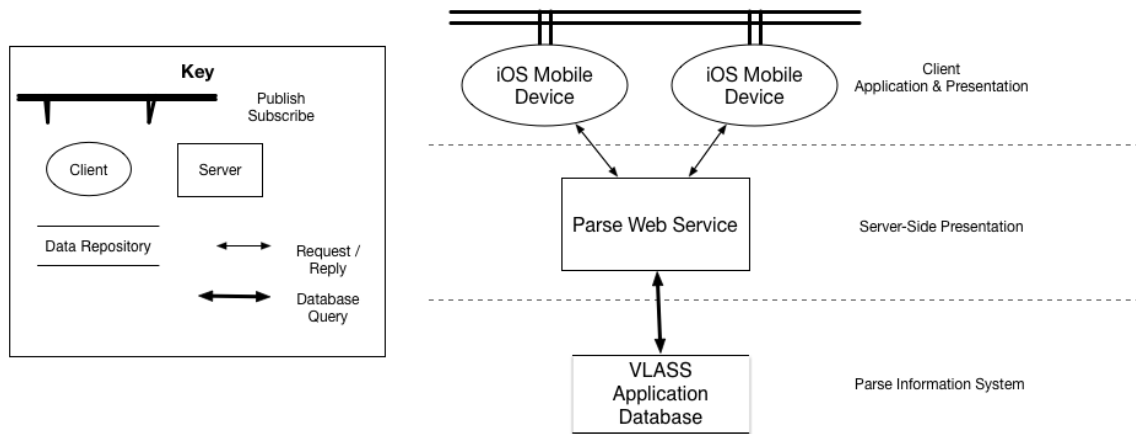
Miles Friedman      [milesfriedmanfl@gmail.com](mailto:milesfriedmanfl@gmail.com)

Contents of this Document

High-Level Architecture.....	3
Design Issues.....	4

## High-level Architecture

The design architecture that will be used is a mix of the publish-subscribe and client-server architecture.



The client-server architecture is a great design for this app because each User will have access to specific data which is stored in the database (which is handled by a third party service). The addition of the publish-subscribe architecture provides the notification to all Users assigned to that course.

### Interfaces

#### Client (iOS Mobile Device)

- The Client is the end user that is using the iOS app. The App will send requests to the Server based off of the actions of the end user.

#### Server (Parse Web Service)

- The Server will handle all Client requests. Based off the access level of the Client's profile, the Server will either grant or deny the request. If granted, the Server will access the

database to retrieve the data. If denied, the Server will send a notification back to the Client that the request was denied.

Database (Parse Information Service)

- The database is where all the information for the system will be stored. This interface is only accessible from the Server.

## **Design Issues**

Design issues that affect our project include reliability, reusability, maintainability, testability, performance, portability, and security. No prototypes will be needed to evaluate this project, as the current App will be tested in the process of being built.

### **Reliability**

The software design significantly aided in this project's reliability. By checking and testing every milestone and feature along the way the project became stable very early in the coding process. This was the biggest deciding factor in the decision to use the V-Model design.

### **Reusability**

Since the Vlass app is by theory infinitely scalable, it was required that the program reused as much code as possible. This is incredibly hard as the app is also very versatile. The project was actually scaled back, to have less features than originally planned. This was to ensure that working functions and features were reusing as much code as possible.

### **Maintainability**

While the Vlass classroom app was developing, the need for maintainability of the code has become increasingly apparent. Due to the majority of data being entered and maintained by casual computer users, the odds for incorrect or unsupported data input is high. To combat this

issue the project team decided that an Admin user would take a Superuser role that can lessen or even prevent minor and common errors in the system.

**Testability**

The V – Model software design requires testing and checking of the project to complete each milestone. Because of that demand, the code was designed to be tested as individual functions and as a whole. Another factor that has increased the testability of the project was the reusability of the code. If the code is significantly reused, you can test a small number of functions in a greater number of ways.

**Performance**

Performance for this application will be divided into client side performance and server side performance. Client side will depend on user's connection and device. Server side will be based on server and database performance, which will be optimized.

**Portability**

Due to application's specific platform (iOS), this application will not be portable to any other system at this time.

**Security**

Since Vlass will store data entered by users, it is required to limit access to the database to only the User's assigned data. During the profile creation the User will be assigned an ID number that limits the data that the User can access. All roles, but the Admin, will have a limit to which data is accessible.

**Design Trade-Offs**

The Vlass app will use a combination of Client-Server and Publish-Subscribed architectural styles. This combination actually matches the scope of the app perfectly with no design trade-offs.

**What Technical Issues Are Expected**

Learning how to use Parse and Swift for coding the iOS app.

**How Will The Technical Issues Be Solved**

The project sections were split up among the group members. Each member would then start reading and learning about their project section's requirement (i.e. swift/coding, APA/documentation, parse/database/service). That member would then be seen as the lead for that aspect of the project.

**The Rationale for Selecting this Architecture**

This architecture was chosen because it had very few drawbacks in comparison to the other architecture options. The client-server architecture covers all needs of this system, and the publish-subscribe architecture includes the function of notifying the Users assigned to the course. This feature is seen as a key component to the app's successfulness.

**Risks**

The potential risks for the Users of the system are;

Security: Since User data is stored on a database it is possible for that data to be obtained by other persons.

Reliability: Like all systems, this program isn't perfect and because of that it is possible for the system to crash. If this system crashes it may become unavailable to the User,

which can affect their standing in the course. An error in the system could also cause User data to be lost, which in turn, would affect their standing in the course.

Virtual Classroom (Vlass)

Detailed Design

COP 4331C, Fall, 2015

**Modification History**

Version	Date	Who	Comment
V0.0	10/15/2015	J. Casserino	Initial Draft
V0.1	10/27/2015	C. Armstrong J. Bender J. Casserino M. Friedman	Rough Draft - Added Design Issues - Added Class Diagram - Added Sequence Diagram - Added Trace Requirements
V1.0	10/29/2015	J. Casserino	Final Draft - Proof Read and Error Checking

Team Name: Group 26

Team Members:

Joseph Bender      [jbender94@knights.ucf.edu](mailto:jbender94@knights.ucf.edu)

Joshua Casserino      [Joshua.casserino@knights.ucf.edu](mailto:Joshua.casserino@knights.ucf.edu)

Chad Armstrong      [chad.armstro@knights.ucf.edu](mailto:chad.armstro@knights.ucf.edu)

Miles Friedman      [milesfriedmanfl@gmail.com](mailto:milesfriedmanfl@gmail.com)



## Contents of this Document

Design Issues.....	3
Detailed Design Information.....	5
Trace of Requirements to Design.....	7

## **Design Issues**

### **Reliability**

In order to provide a dependable application, a consistent back-end architecture is the most critical component for reliability. Many third party data web services were analyzed until a reliable candidate, Parse, was selected to integrate with the application.

### **Reusability**

The swift code in the iOS app is written in the form of MVC and class structures. This structuring of the code allows for operations and attributes to be shared across view controller classes. This feature of the design allows the developers to simplify the design and shorten development time, by not having to write as much code.

### **Maintainability**

Implementing the Admin perspective into the application is a critical feature to ensure maintainability. The Admin account is hard coded into the database with specified strict credentials. This account has all of the functionalities and abilities of non-administrative Users. It also contains extra functionalities to provide additional services to the standard Users, and revert system errors that may occur.

### **Testability**

The app is to be heavily unit tested throughout development to ensure all features are functional before the developers move forward in the design. The results are verified with the expected outcome. The app is also tested as a whole on a physical device. It is written to a physical device to ensure that no discrepancies occur between the simulator and the physical device.

**Performance**

The performance of the Vlass application is a critical feature of the design. In order to achieve a maximum performance, all query operations to the Parse data service must be made as efficiently as possible. Querying of redundant data or non-essential data is avoided at any occurrence. Also, the internal algorithms of the application to verify data and receive data are all to be within the runtime of  $O(n^2)$ .

**Portability**

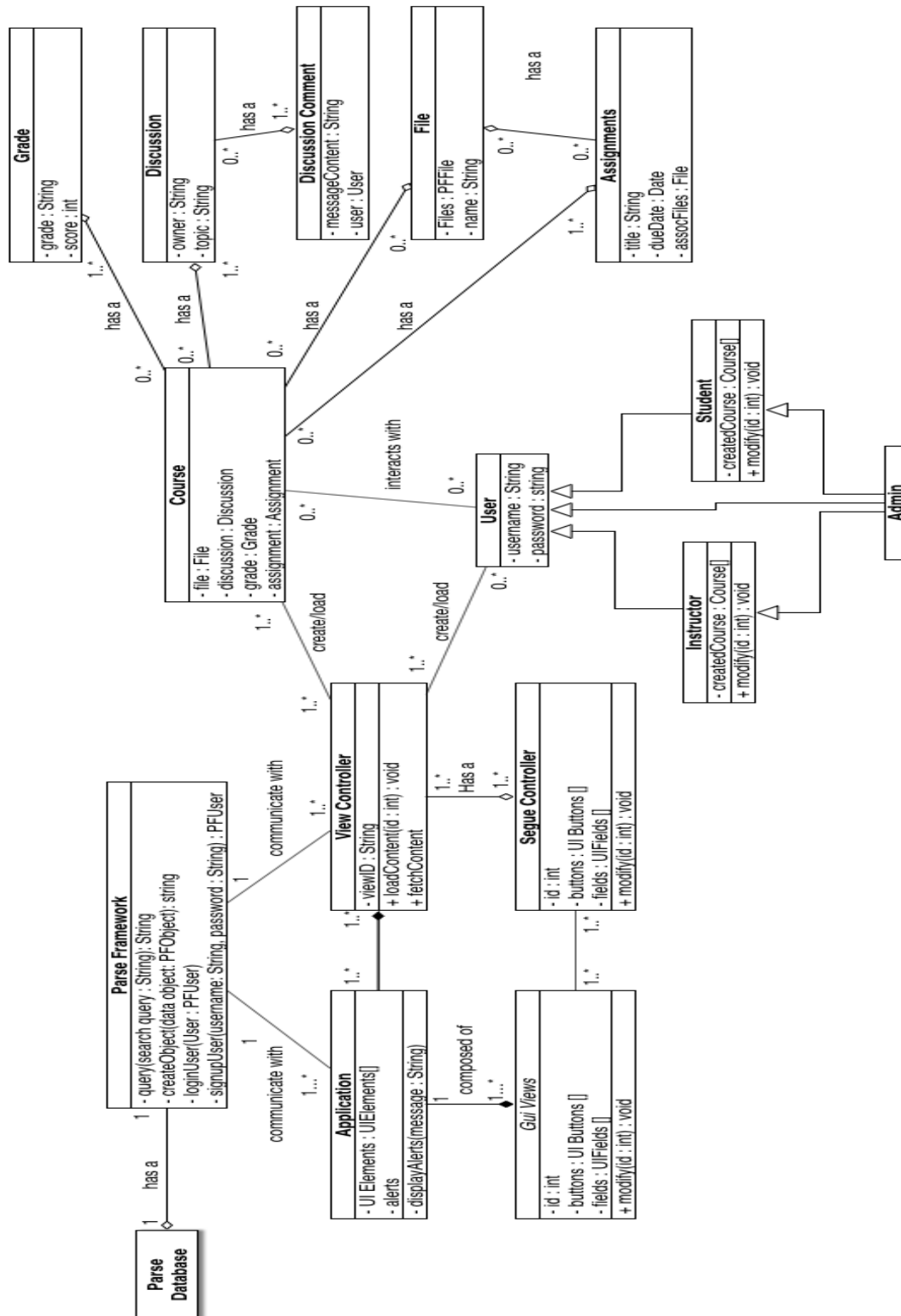
If the application were taken to market on the Apple iTunes<sup>®</sup> Store, it would provide portability because the UI design has been setup to work with a variety of iPhone and iPad screen sizes. This requires that all UI elements in each perspective are set with the correct constraints to accommodate any resolution.

**Security**

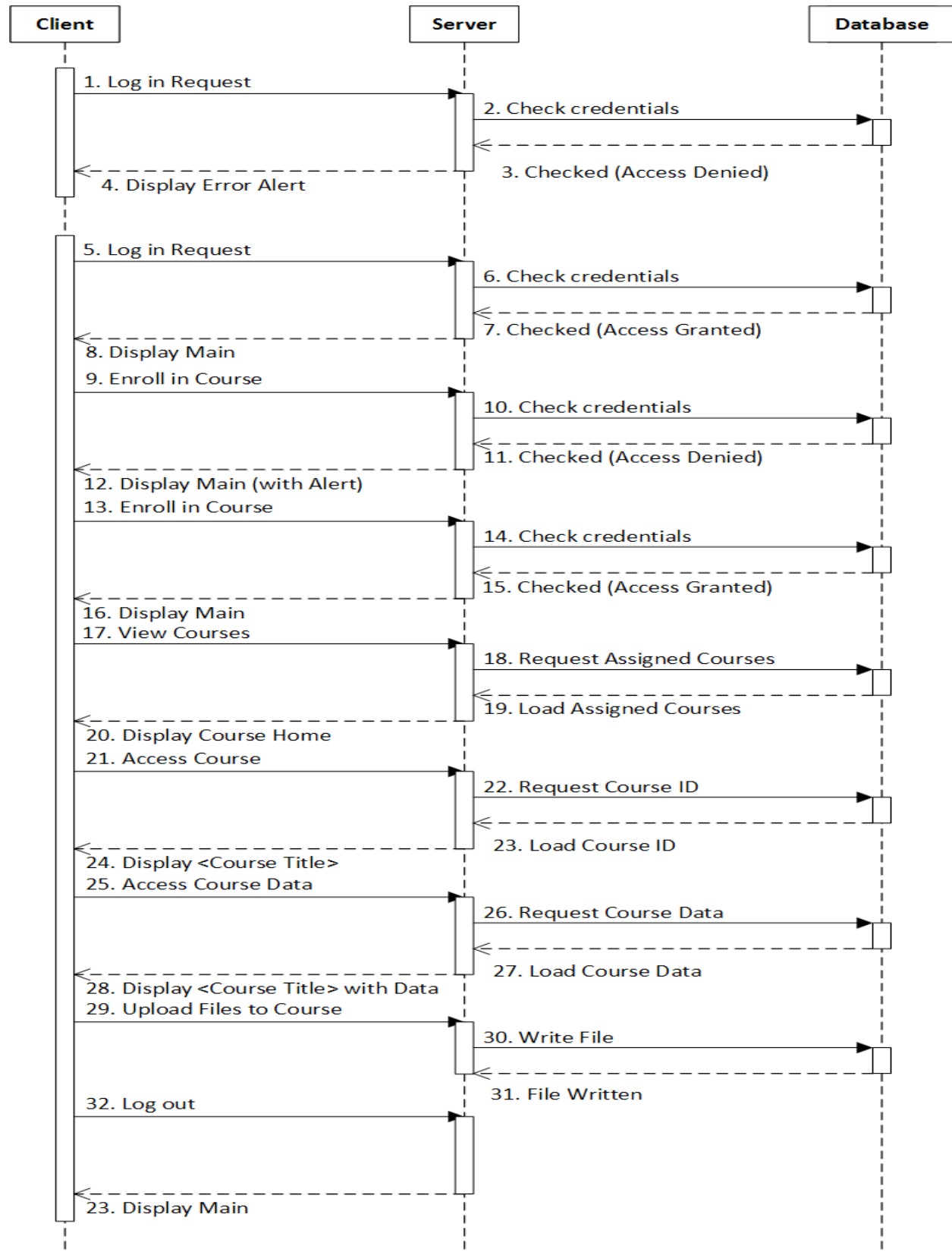
The security of the Vlass application is evident in the user credential enforcement. The password criteria were determined from field standards for validation. The credentials are thoroughly enforced in the swift code of the application. The transfer of data to the Parse data service is also encrypted. This is another feature that led to the decision to select Parse.

## Detailed Design Information

## Class Diagram



## Sequence Diagram



**Trace of Requirements to Design**

<b>Design Component</b>	<b>Requirement Number</b>
Client (iOS Mobile Device)	01, 02, 03, 04, 05, 06, 07, 08, 09, 10
Server (Parse Web Service)	01, 02, 03, 04, 05, 06, 07, 08, 09, 10
Database (Parse Data Service) *The Database content is only updated	01, 02*, 03*, 04, 05*, 06, 07*, 08, 09,10*