Gaurav Srivastava • Naveen Ganessin • **Alex Rosenberg** • Michael Schloss

ABC

Design Document

Team Members

Gaurav Srivastava — srivast6@purdue.edu
Naveen Ganessin — nganessi@purdue.edu
Alex Rosenberg — rosenbea@purdue.edu
Michael Schloss — mschlos@purdue.edu

Purpose

Functional Requirements

ID	Requirement
1	As an admin, I can add/delete classes
2	As an admin, I can change class storage quota
3	As an admin, I can add/delete teachers
4	As an admin, I can add/delete students
5	As an admin, I can post school-wide announcements
6	As an admin, I can get course-wide or school-wide grade stats
7	As an Instructor, I can add/delete assignments, quizzes, and tests
8	As an Instructor, I can view file submissions from students
9	As an Instructor, I can set time-limits on assignments, quizzes, and tests
10	As an Instructor, I can post class-wide announcements
11	As an Instructor, I can send these announcements as emails should they wish
12	As an Instructor, I can host content on their class
13	As an Instructor, I can specify whether their hosted content is private or public
14	As an Instructor, I can teach multiple courses
15	As an Instructor, I can submit students' grades for assignments, quizzes, and tests
16	As an Instructor, I can view grades for any of their classes
17	As an Instructor, I can generate grade reports
18	As an Instructor, I should be able to generate a random text for students to enter on their account for attendance. (if time allows)
19	As a Student, I can see school-wide announcements
20	As a Student, I can see class-wide announcements

ID	Requirement
21	As a Student, I can see their grades from every class to which they're enrolled
22	As a Student, I can submit assignments
23	As a Student, I can take quizzes or tests
24	As a Student, I can send email to their teacher from the application
25	As a Student, I can view/consume public class files
26	As a Student, I should be able to receive an email notifications about assignments and exams due (if time allows)
27	As a Student, I should be able to enter the exact teacher's random text for attendance. (if time allows)

Nonfunctional Requirements

ID	Requirement
28	The application will be responsive
29	The application will use robust encryption to keep all data secure
30	Implement a caching feature that will allow users to see some data without an internet connection
31	The application will not use too much storage
32	The application will remain light on system resource usage
33	The application will communicate with native OS features to give the best possible experience
34	The application will have an aesthetically pleasing UI
35	The application will have separate UIs for different stakeholders for the app.
36	The database will have the proper keys and data types and enforce these restrictions
37	The database will have the correct foreign key relationships and will maintain these checks automatically and strictly
38	The database will be made of InnoDB tables to allow for fast indexing and recall of data
39	The database will use SSL to communicate with the application to provide secure endpoints for data transfer
40	The database will be password protected
41	The database will return JSON formatted data to the application
42	All passwords should be stored only as hashes

Design Outline

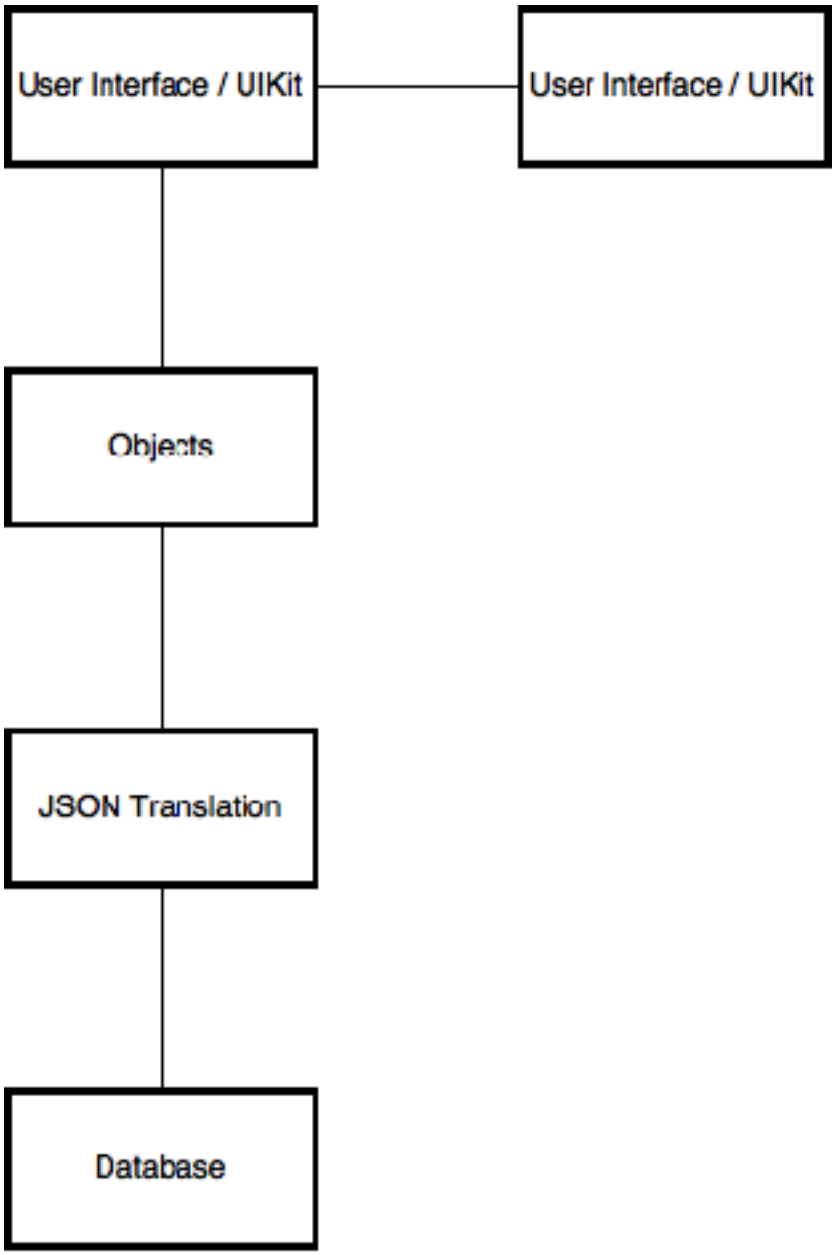
High Level Overview of System Architecture

For an earlier project, we developed a system that included a Java client and a web server client that managed a MySQL database. Due to the success and ease, this project structure was adapted into this project.

Our web server includes a couple of PHP files that assist in processing SQL queries and returning boolean data or JSON data where necessary. It also includes a MySQL database that stores all data for the application to allow for multi-user and multi-device cross platform syncing.

A diagram is included in the following section that shows the basic high-level overview of the system's components.

Detailed Overview of System Architecture



The Java client is split between 5 different layers: database, json translation, objects, ui, and uikit.

Every layer is kept sandboxed with a few exceptions. This is to ensure a minimal amount of redundancy, and allows for built-in package safety. It also allows for an easier time developing all the components.

Database

A collection of frameworks and objects that efficiently and abstractedly connect to the web server and return the necessary data.

The database layer contains a custom built SQL class that prevents SQL injection and provides for overload safety. The database layer also contains a and uploader, downloader, and dispatch queue to manage process communication over https to prevent doubling up data.

JSON Translation

The JSON Translation layer provides an abstraction between the User Interface and the database layer. The JSON layer is primarily responsible for processing queries from the UI and returning the proper objects that encapsulate the data.

To provide for easy understanding and development, each object has its own JSON Translation class to allow for custom methods and removes and huge classes from the application. JSON translation also stores and loads data from local, on-device storage.

Objects

Objects are encapsulations of database data that is translated by the JSON Translation layer to be utilized by the UI. There is a one-to-one relationship with the number of items instantiated and the number of items returned from the database. These objects conform to serialization to be stored and recalled from device storage.

User Interface

The layer the user directly interacts with. The UI is built using Swing.

The UI layer only communicates with the JSON Translation layer to retrieve data to display to the user.

UIKit

Several various classes that help with on-device storage, time management, and communication across the different layers. Also stores and manages custom fonts used throughout the app.

Broad Overview of Sequence of Events

- User provides input when clicking a button
- The UI will perform one of two options:
  - The UI will change the display to the next screen
  - The UI will prompt the JSON Translation layer for data to show the user. The JSON layer will communicate with the database layer to grab the requested data from the web server
    - Through a series of notifications, data will be passed from the database layer back to the JSON layer, and then from the JSON layer to the UI, where the UI layer will display the information to the user
- The user will continue to navigate through the UI and the cycle will repeat

Design Issues

Functional Issues

FUNCTIONAL ISSUE #1 — WHICH USER ROLE CREATES A NEW COURSE TO BE ADDED

- Option 1: A separate admin
- Option 2: The instructor in the class
  - Choice: A separate admin
  - Discussion: This reduces the responsibility of a professor and thus this feature is provided for the admin. The admin will be given a separate UI where he can view different classes and get an overview of all the classes.

FUNCTIONAL ISSUE #2 — WHICH USER ROLE ADDS A STUDENT TO AN EXISTING COURSE

- Option 1: Admin
- Option 2: The instructor of the class
- Option 3: A student
  - Choice: Admin
  - Discussion: This restricts a student from entering a class that he/she is not enrolled in. This also reduces the responsibility of the instructor. This feature is provided for the admin.

Nonfunctional Issues

DESIGN ISSUE #1 — WHAT SOFTWARE TOOL DO WE USE FOR CLIENT DEVELOPMENT

- Option 1: Java Swing
- Option 2: Javafx
- Option 3: HTML
  - Choice: Java Swing
  - Discussion: We wanted to use provide the users a application on their computer and not through the a website. Hence we used Java Swing since it is supported online and our team has experience working with it, thus reducing our learning time

DESIGN ISSUE #2 — WHAT TYPE OF DATABASE IS MOST APPROPRIATE FOR OUR DATA

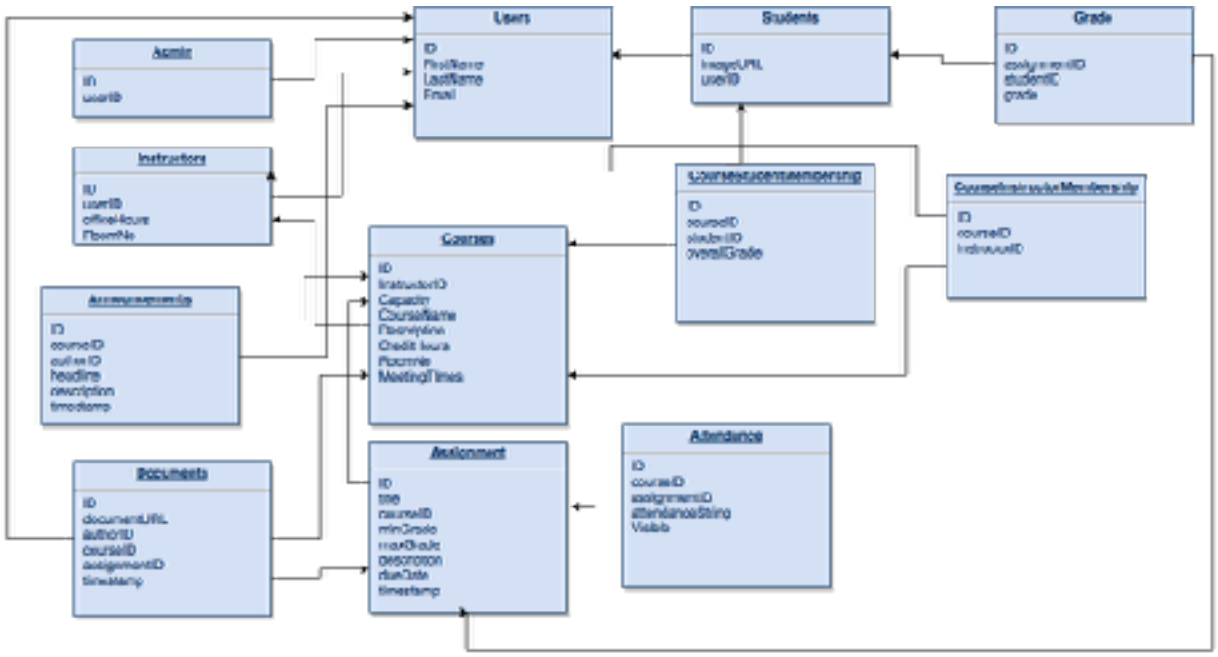
- Option 1: SQL database like MySQL or Microsoft SQL
- Option 2: NoSQL database like MongoDB
  - Choice: MySQL
  - Discussion: A relational database will be the best choice for our data. MySQL is the best SQL database to use because of its ease of implementation and cost

DESIGN ISSUE #3 — WHAT LANGUAGE SHOULD WE USE FOR OUR BACKEND SERVICES

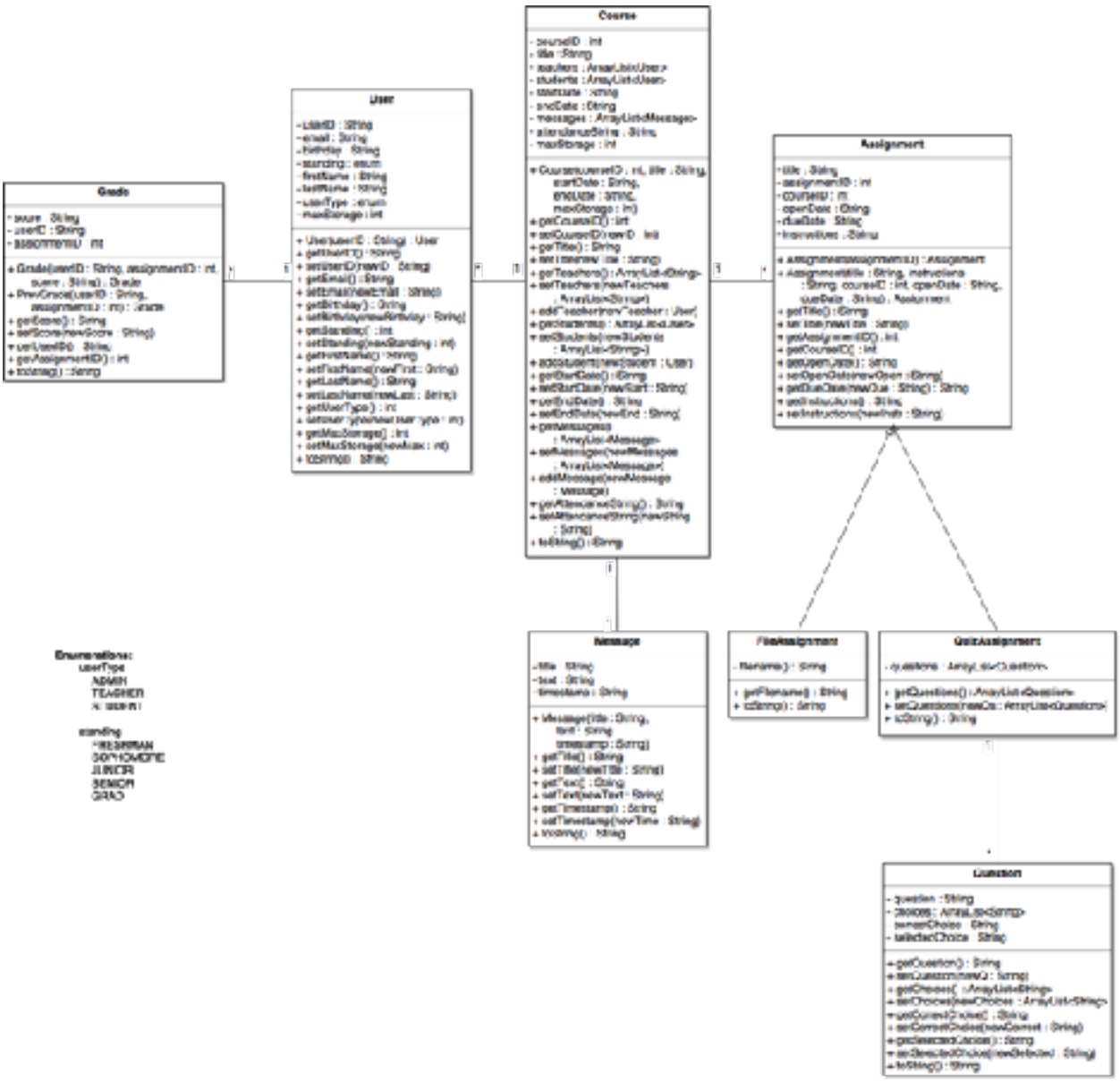
- Option 1: Python
- Option 2: PHP
- Option 3: Java
  - Choice: PHP
  - Discussion: We used PHP because it is the easiest way to interact with an MySQL database

Class Diagrams

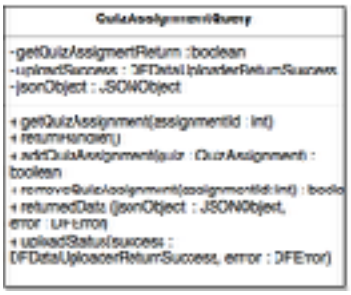
DATABASE

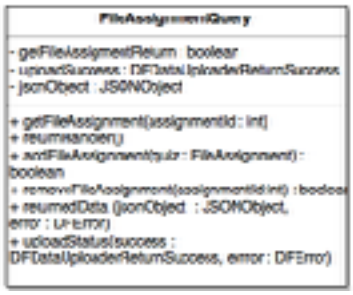


OBJECTS

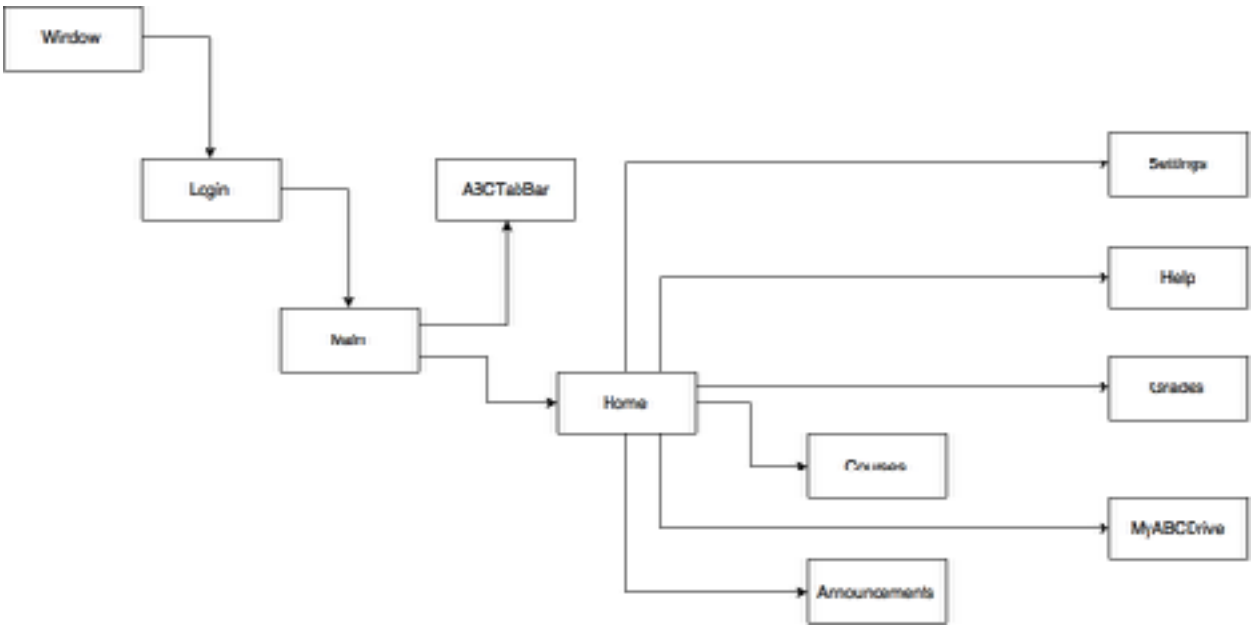

Course
User
Grade
Assignment
Message
FileAssignment
QuizAssignment
Question
Enumerations:
userType
ADMIN
TEACHER
standing
FRESHMAN
SOPHOMORE
JUNIOR
SENIOR
GRAD

JSON Translation





User Interface

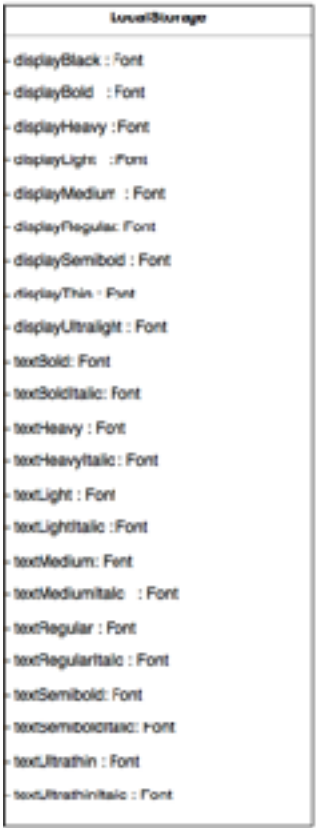


UIKit


DFNotificationCenter
+ defaultCenter: DFNotificationCenter
- register(observer, notificationName): Void
- remove(observer): Void
- post(notificationName, object): Void
DFNotificationCenterDelegate
- performActionFor(notificationName, userData): Void


LocalStorage
+ defaultManager: LocalStorage
- saveObjectToFile(object, fileName): Void
- loadObjectFromFile(fileName): Object


TimeManager
- run(): Void


- displayBlack : Font
- displayBold : Font
- displayHeavy :Font
- displayMedium : Font
- displaySemibold : Font
- displayUltralight : Font
- textBold: Font
- textBoldItalic: Font
- textHeavy : Font
- textHeavyItalic : Font
- textLight : Font
- textLightItalic :Font
- textMedium: Font
- textMediumItalic : Font
- textRegular : Font
- textRegularItalic : Font
- textSemibold: Font
- textUltrathin : Font

Description of Classes and Models

DATABASE

DFDatabase

The main interaction class for database communication. Contains encryption/decryption methods as well as entry point for SQL execution

DFDatabaseCallback

An interface for classes to adopt to respond to returned data from the database

DFError

Custom error class for non-throwable errors

DFDatabase

The main interaction class for database communication. Contains encryption/decryption methods as well as entry point for SQL execution

DFDataDownloader

Responsible for downloading files and JSON data from SQL statements

DFDataUploader

Responsible for uploading files and receiving success/failure of SQL statements

DFWebServerDispatch

Guarantees each network connection action receives exclusive access

DFSQL

The custom-built SQL class. Prevents SQL injection and overloading

JSON Translation

UserQuery

This class contains fields and methods necessary to translate fields in User Objects into a sql query to be inserted into the database. It also allows for the the retrieval of JSON user information translated into a user Object based on certain parameters. It also supports the removal and modification of a User Object.

CourseQuery

This class contains fields and methods necessary to translate fields in Course Objects into a sql query to be inserted into the database. It also allows for the the retrieval of JSON course information translated into a course Object based on certain parameters. It also supports the removal and modification of a Course Object.

GradeQuery

This is used to create grades for assignments and users within the database and also assists in the retrieval of individual grades from the database in the form of a usable Grade Object.

QuizAssignmentQuery

This allows for the addition of QuizAssignments into the database and the modification of questions

FileAssignmentQuery

This allows for the addition of FileAssignments into the database and the modification of questions

OBJECTS

User

This class contains all fields and methods directly relevant to information about students, teachers, and administrators.

Course

Class objects contain information pertaining to the specific class environment overall, such as the teachers, students, and class title.

Message

Contains strings representing the fields of an announcement, such as its title and text.

Assignment

This is an object that provides a way for students to complete assignments and teachers to view and grade them when complete.

FileAssignment

This is a specific type of Assignment object that requires a student to upload a file to the application.

QuizAssignment

This type of Assignment object allows for quizzes to be taken through the application.

Question

A Question object is used in QuizAssignment objects to allocate questions and answers to each quiz.

Grade

A Grade stores an individual score for a given student on a given assignment.

User Interface

Main
Initializer class

Window
The manager for all displayed frames on screen

Login
The login panel displayed upon first launch

Home
ABC's home page

Alert
A custom dialog that shows errors in user input

ABCTabBar
The tab bar across the top of ABC.

MouseListenerManager
Listens for mouse input across the application and forwards it correctly

UIStrings
Strings used to communicate between modules over DFNotificationCenter

UIKit

DFNotificationCenter

An observer->receiver data passer. Used to maintain sandboxing of modules from bottom to top. Observers add themselves to listen for certain strings. When another class posts a notification with that string, the object is woken and receives the data

LocalStorage

Generic storing and loading of data

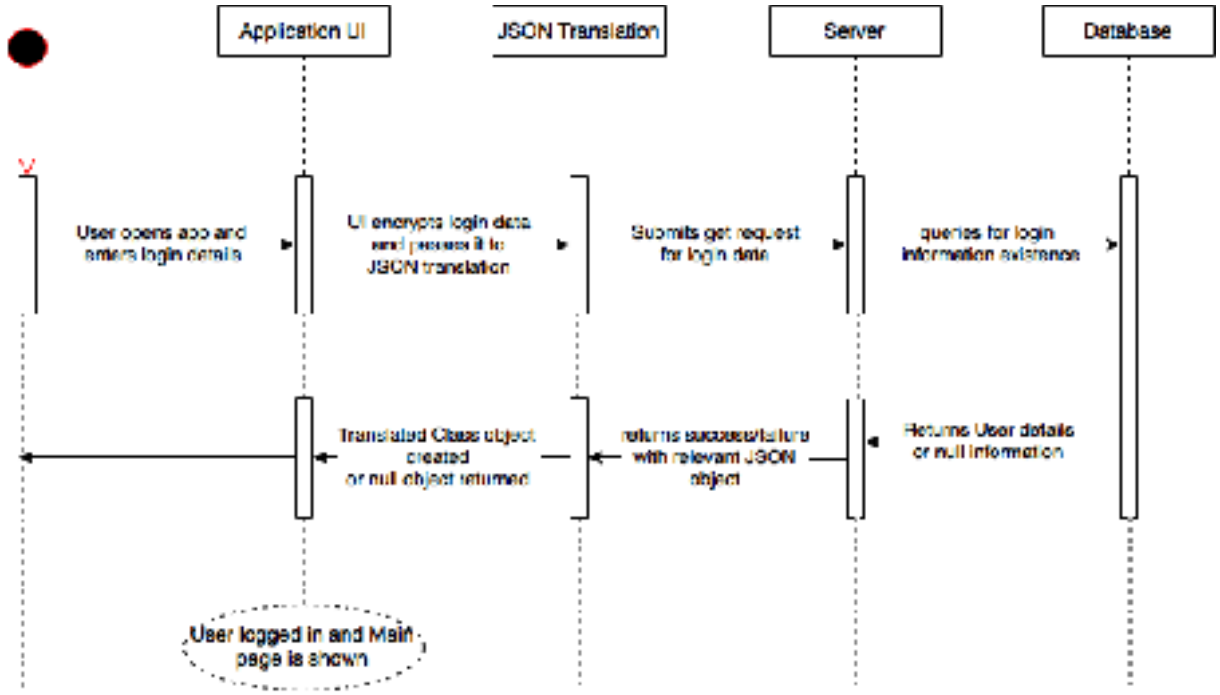
TimeManager

Fires off notifications at specific time intervals during app execution

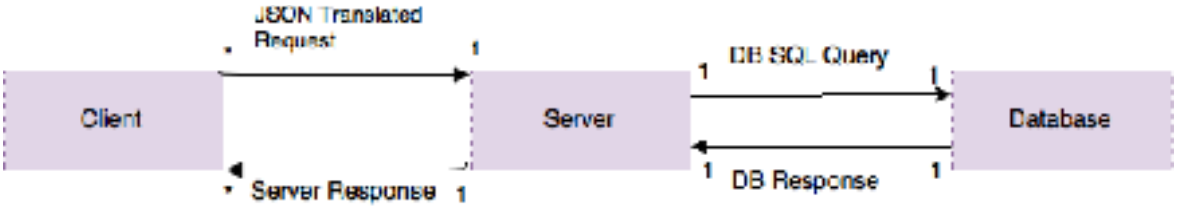
UIFont

Manages custom font throughout the application

Sequence of Events Upon Launch

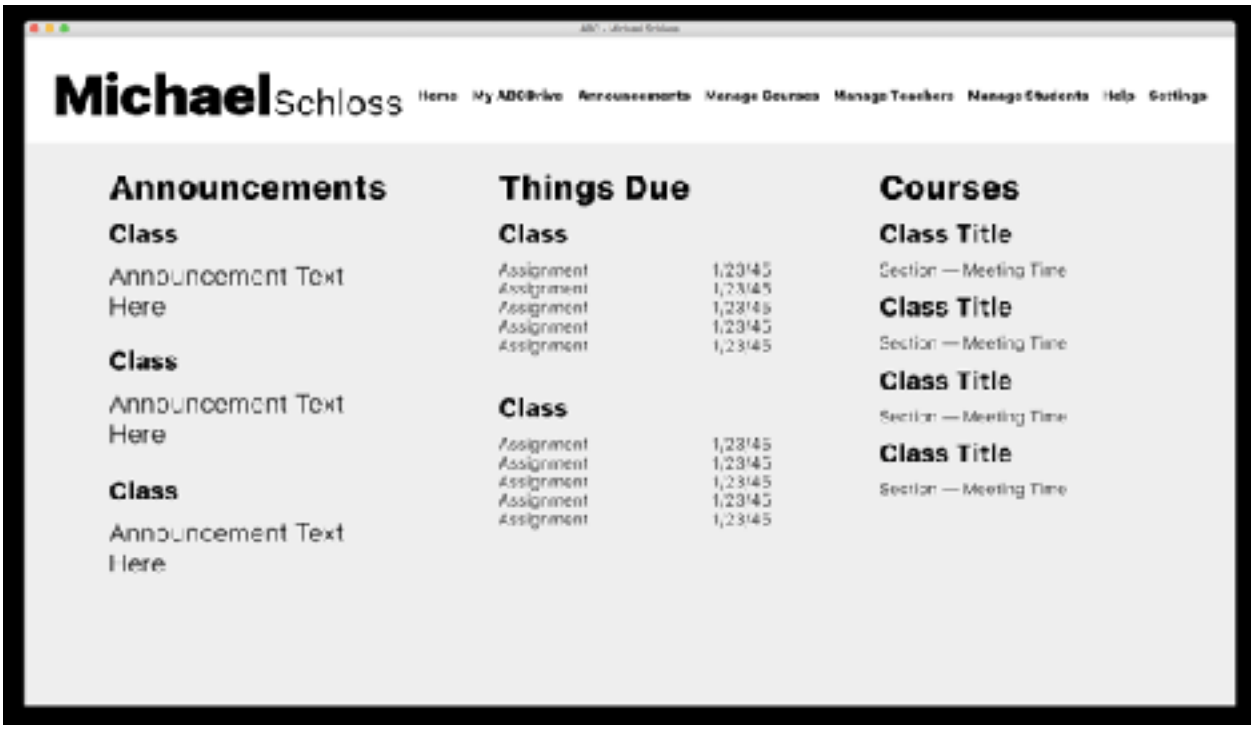


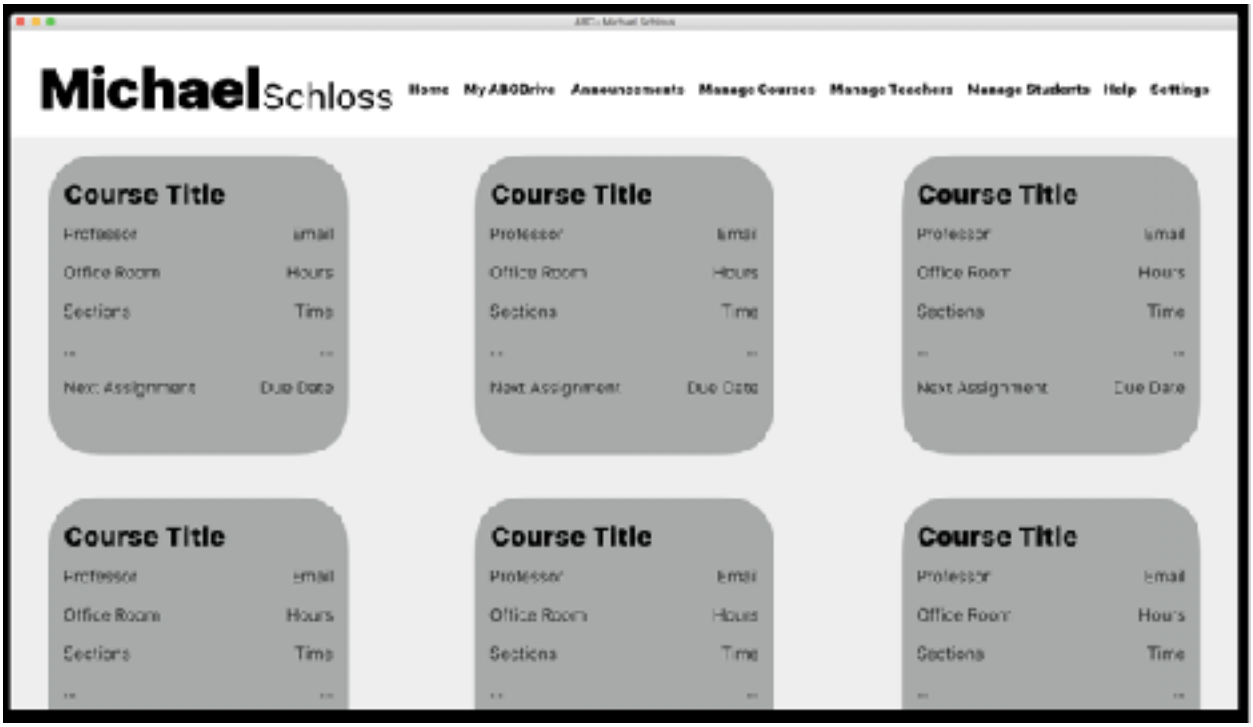
High Level Client Server Protocol



UI Mockup


ABC
Username
Password
Quit


MichaelSchloss
Announcements
Class
Announcement Text Here
Class
Announcement Text Here
Class
Announcement Text Here
Things Due
Class
Assignment 1,23/45
Assignment 1,23/45
Assignment 1,23/45
Assignment 1,23/45
Assignment 1,23/45
Class
Assignment 1,23/45
Assignment 1,23/45
Assignment 1,23/45
Assignment 1,23/45
Assignment 1,23/45
Courses
Class Title
Section — Meeting Time
Class Title
Section — Meeting Time
Class Title
Section — Meeting Time
Class Title
Section — Meeting Time


MichaelSchloss
Home
My ABQDrive
Announcements
Manage Courses
Manage Teachers
Manage Students
Help
Settings
Course Title
Professor
email
Office Room
Hours
Sections
Time
Next Assignment
Due Date
Course Title
Professor
email
Office Room
Hours
Sections
Time
Next Assignment
Due Date
Course Title
Professor
email
Office Room
Hours
Sections
Time
Next Assignment
Due Date
Course Title
Professor
email
Office Room
Hours
Sections
Time
Course Title
Professor
email
Office Room
Hours
Sections
Time
Course Title
Professor
email
Office Room
Hours
Sections
Time