Team 3 Design Document for “Classical”

John Du, Mitch Holm, Luke Kong, Shawn Nirappil, Timothy Vincent, Jintao Zeng

Table of Contents

1. Purpose
2. Design Outline
    a. Architecture
    b. Browser/Application
    c. Backend
3. Design Issues
    a. Logging in
    b. Chatting
    c. Cheating
    d. Credibility
4. Design Details
    a. Course
    b. Chatroom
    c. Message
    d. Forum
    e. Thread
    f. Report System
    g. Student
    h. Schedule
    i. Event
5. UI Mockups
    a. Web
    b. Android
    c. iOS

Purpose

There is no intuitive and ubiquitous app for students to manage their time in and out of class collaboratively with their peers. A lack of community is a massive problem that plagues collegiate academic classes. To solve this, we are building a cross platform service to connect Purdue students to each other based on what classes they are in. This service will have scheduling, group messaging, and a forum to provide a wide variety of options for students to connect with each other.

The following items indicate functionality that will be be present in the final product. These include both user facing features, and server tools that the user will never see.

- On every platform (Android, iOS, Web), a user will be able to:
    - Securely Login
        - Handled by Purdue Career Account
    - View their schedule
    - Modify their schedule in the following ways
        - Lookup classes
            - This can be used for students to audit classes before enrolling
        - View events (Study sessions, club callouts, SI sessions)
            - Add events
            - Edit events
            - Remove events
    - Interact with peers through median of the forum
        - Ask questions
            - Vote on best answer(s) to a question in the forum
        - Answer questions
            - Edit your own postings
            - Delete your own posting
        - View questions
            - Report posts that are offensive or rule breaking
                - 5 reports will email the professor and delete the post
            - Search for posts (time permitting)
    - Chat with classmates in real time
        - Clients will connect to an IRC server
            - Every class has a unique private section of the server called a channel

            - Clients will parse the messages received from the server and display them in a user friendly manor.
- A teaching assistant or professor will be able to:
    - Receive emails of reported posts
        - A threshold of 5 reports must be issued before a message is sent
    - Ban a student from participating in a class
    - Notify the student that they were reported and the reason for the report
- The server will be able to:
    - Handle the forum interaction between the database and client
        - Post information from the client to the database
        - Get information from the database and post

Design Outline

Our project will be divided into four components and adhere to the Client-Server model. Three of these components are separate and equivalent clients on Android, iOS, and popular Web browsers. The final component is a single web server which provides a means of communication from client to client. The three Model View Controller based clients will be able to view a calendar, live chat, and a forum for questions. The forum and calendar will be call upon the server periodically, or upon swing down in the fourm or calendar view. While the chat will maintain an active connection to the server in order to send and receive messages in realtime.

- Overview of the Architecture
    - Our application will complete these tasks by using a client-server architecture
        - Fat server, slim client paradigm
            - Removes issues on slow client hardware
            - Removes issues with platform dependence
        - Client will
            - Be a means to display data to the user in a lightweight way
            - Be cross platform for browser and mobile applications written in
                - Web: HTML/CSS
                - Android: Java and XML
                - iOS: Swift
        - Multiplatform allows for a wider user base.
        - Backend will
            - Have a web server that
                - Handles all data interaction between database and client
                - Is written in Java
            - Have databases
                - That are MongoDB noSQL based
                - That are divided into Users and Forums databases

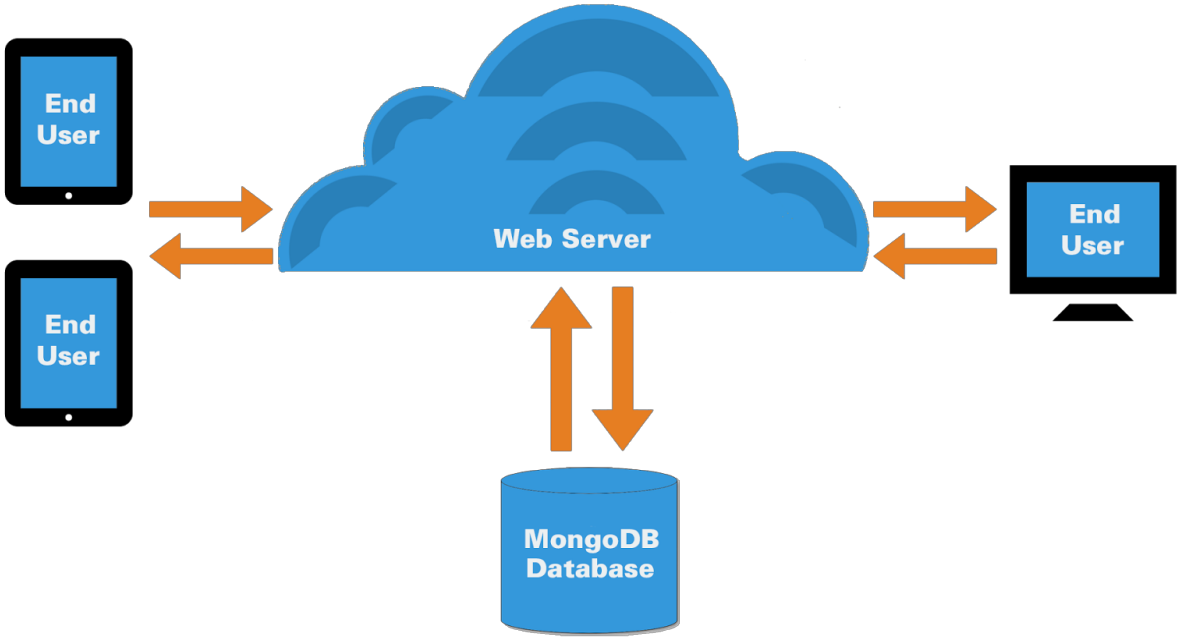
- Browser/Application
    - All user interaction will be done through one of the three median platforms (Android, iOS, Browser)

    - First time login through my Purdue account will create a database entry for the user, holding their career account username and course schedule.
        - Subsequent logins will reference this career account username to fetch the user's schedule from the database
    - Upon launching the application, all data will be refreshed to reflect the most recent changes found in the database.
    - For the mobile platforms a request to refresh forum data from the server can be issued as needed by pulling down while in the forum view.
    - Real time chat will use asynchronous polling in the background.
    - Mobile users will be able to request an updated schedule (to reflect classmate changes) by pulling down in the schedule view
    - Will function as a thin client, and run most code on the server. While server performance is a concern, the level of social integration makes most local processing impossible.

- Backend
    - Server will handle all data post and get requests from clients
        - The server will protect the database from malformed data
        - The server will present data in a succinct way to the client
    - Databases will be divided into
        - A database that holds Purdue career username and class information gathered from each user's Purdue Career Account.
        - A database that holds posts on the forum which comprise of :
            - Purdue Career account name of Poster
            - Full name of Poster
            - The title of the post ("" if it's a reply)
            - Forum post text
            - Forum post id
            - Score of the post
            - If it is a forum post or reply
    - Forum replies will be sorted and given to the user based on score of the post.
        - Order determined client side then displayed in descending order
    - The server will store new content of the user on request of the client.
    - The database will not be bound by size requirements, as all input will be text based and archived/deleted each semester.
    - Scalability will available for allocating new universities to new servers, as each server and database pair will handle a single university students

        - This allows for a server and database to be local to each university for optimal content distribution
    - The report system will check for posts with 5 reports or more if found it will
        - Remove the post
        - If a majority of the reports are for cheating
            - Email the professor with the content of the post

The final application will have interactions of the following layout:



Design Issues

Issue 1: How will the user login in, in a way that ties them to their status as a Student?

- *Option*: Use the myPurdue secure login system. When the system is rolled out to other schools, their secure login system will be hooked into the app.
- *Option*: Ask the user for their school as part of account creation.
- *Option*: Parse the email address for a school name proceeding .edu address
- *Decision*: Use the myPurdue secure login system. This allows us to access all relevant course information and grab accurate names, emails and schedules.

Issue 2: How will chatting be handled?

- *Option*: Small group style chats in which you can add participants from single classes. This fosters friendships as you can select which classmates you wish to chat with
- *Option*: One-on-One chatting for classmates to contact one another privately.
- *Option*: Entire class chat, available to every student in the class. “Open Discussion.” Prevents cliquey behavior and allows fair resources for every student in the class.
- *Option*: Anonymous message board, this a very easy implementation which just requires users to connect to our chat server.
- *Decision*: Entire class chat, available to every student in the class. “Open Discussion.” Prevents cliquey behavior and allows fair resources for every student in the class. Having non anonymous posters also reduces the chances of attempted cheating.

Issue 3: How is cheating prevented and handled in Classical?

- *Option*: All chats and forum posts are reportable. These reports are sent to Professors with the infringing students names and PUIDs.
- *Option*: Allow Teaching Assistants and Teachers to access an administrator access level of the app, in which they can monitor classes and mute offenders in chat.
- *Option*: Give EULA stating that cheating is not allowed on Classical.
- *Option*: Moderation by Developers. Hire moderators for additional support.

- *Decision*: All chats and forum posts are reportable. These reports are sent to Professors with the infringing students names and PUIDs. There will be thresholds before the post is hidden and the teacher is notified automatically.

Issue 4: How is student credibility factored into Classical?

- *Option*: Special status (“Top Answerer”) for those who answer questions frequently and accurately.
- *Option*: All comments are treated equally.
- *Option*: Allow students to rate answers as “good answer”.
- *Decision*: Allow students to rate answers as “good answer”, thus the replies to the question will be displayed in descending order from most “good answer” votes to least. We chose not to have special status for high answerers so that the classroom can be more egalitarian.

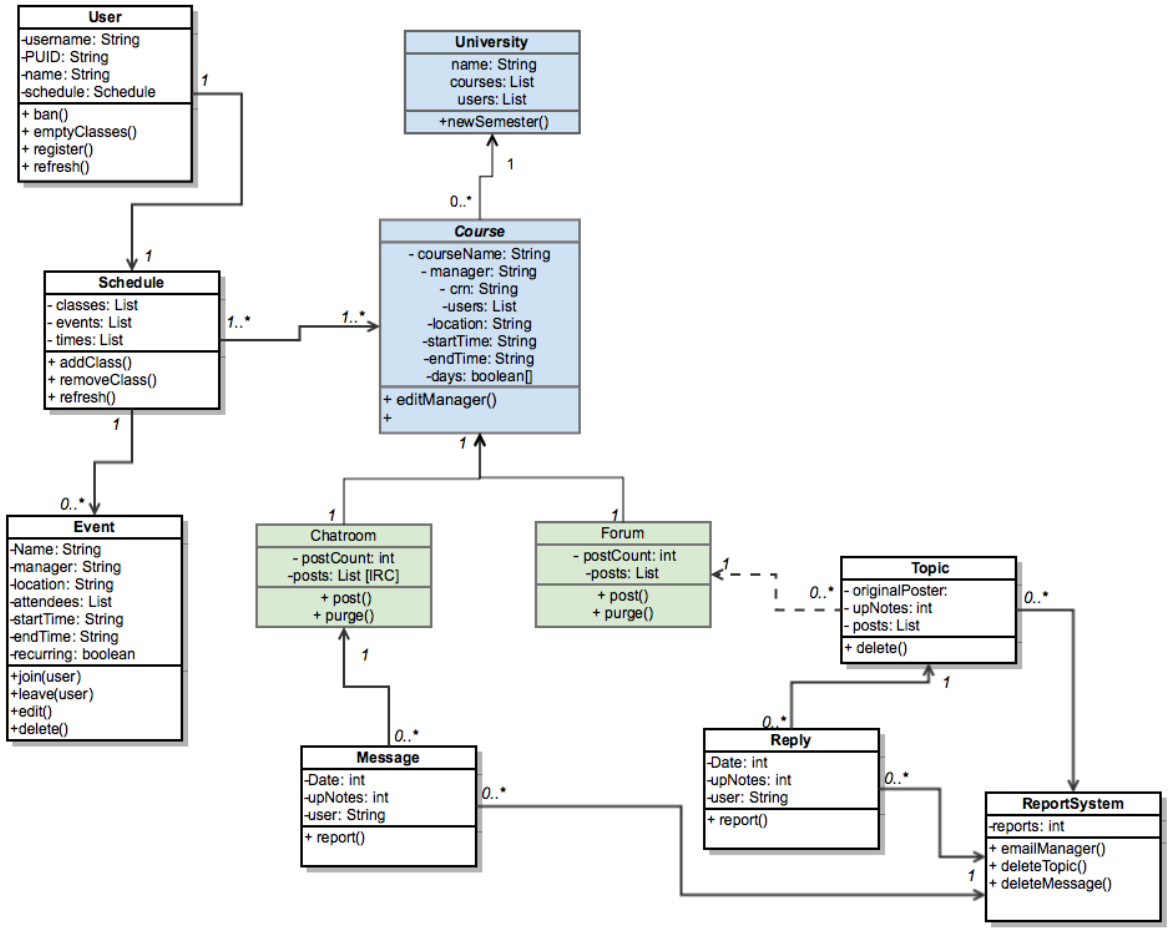
Issue 5: What database architecture would best be suited for our application’s needs?

- *Option*: MongoDB
- *Option*: mySQL
- *Decision*: MongoDB is intuitive and integrates well with our Java webserver, much of our team is familiar with it and it is simple to integrate into our systems.

Issue 6: How much access does the Professor have to Classical?

- *Option*: Administrator level account
- *Option*: Regular user account
- *Option*: No access
- *Decision*: Professor’s have regular user accounts, with access to the courses they teach. This allows the professor to monitor and contribute to their classes if they choose to without having strong control over discussion.

Design Details



Course

A course will be the primary communities that students will join and interact with. They will be identified by CRN. Member variables will include course name, list of students, and times for the schedule. This will be the container for the majority of user interaction. The manager will be a string that specifies who manages the course, be it a TA or professor. The editManager function will be used to change the manager and will only be used by the current manager, or by the server itself.

Chatroom

The chatroom will be one of the primary forms of communication between students within a class. These will be modified IRC channels designed primarily

for single line messages and casual conversation. Each message will have the ability to be reported, but not voted on.

Message

Messages will make up chatrooms. They will have a user identifier, ID, content body, and a report function.
The user identifier would identify the author of the message.
The ID would primarily be used for identifying the message when reported.
Content body will begin with the time and date followed by the actual content.
The report function will allow users to report messages for cheating, and spam.

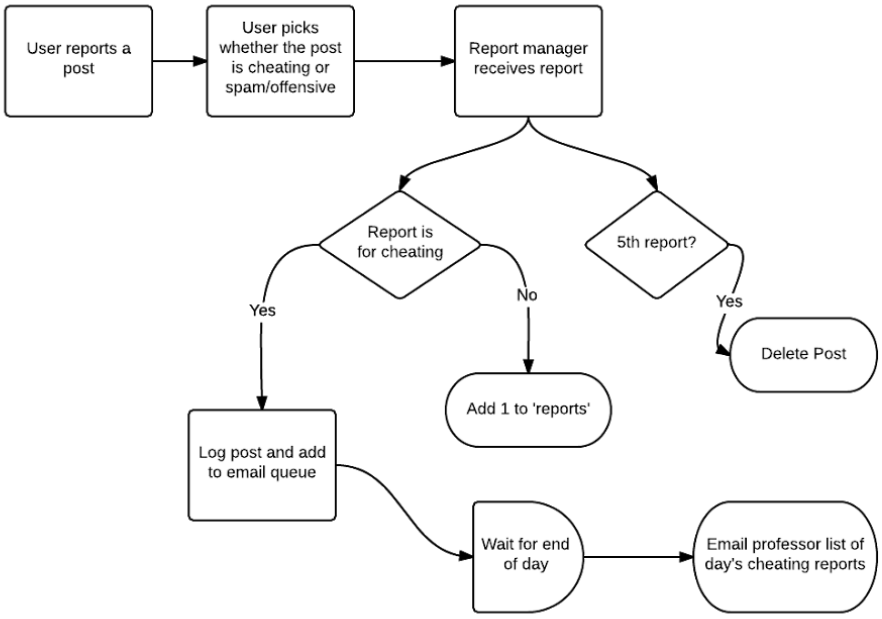
Forum

Forum will also be one of the primary forms of communication between students within a class. The forum will contain an arbitrary number of forum threads in a list.

Forum Post

Every forum post will consist of a user identifier, topic title, topic or reply identifier, ID, score, content body, and a report function.
The user identifier would identify the author of the post.
The post title will be used as a forum thread title, while subsequent replies will have a null title.
The reply identifier is used to determine whether the post is a forum topic or a reply to a topic.
The ID will be used to link between posts such that the ID will be preceded by “@” similar to Piazza topic linking.
Content body will simply be the body of the post that will be displayed to users.
Score will be used by the client to determine the top reply to a post. Users will be able to increment or decrement a post once.
Every forum post will be stored in a single “forum post” database.
The report function will allow users to report forum posts for cheating, and spam.

ReportSystem



Report system will handle reports from messages and forum posts. It will store instances of reports by post ID and type of report. There will be three options for reporting content: cheating, and spam.
Cheating reports for each course will be compiled over a day and forwarded to the course instructor. After a certain amount of reports, the offending post or message will be deleted.
Spam reports will deleting the post from the database or the message from the chatroom after a certain amount of reports.
Currently the plan is to delete posts and messages after five reports.

Student

Students will each have a name, Purdue email, and enrolled courses. Everyone who uses the service will be considered a student, including professors.
A student name will be used as a display name for messages and forum posts.
The Purdue email will be used as a login.
Enrolled courses will make up their schedule and communities that students will be a part of.

Schedule

Every student/user will have a schedule that lists all of their classes and events. There will be functions for adding and removing courses and events.
The schedule will have a list of the classes that regularly occur for the user, and events that may occur regularly. Users will NOT be able to manually add courses, but will be able to create events for their individual classes. ‘Peeking’ will not be a part of the scheduling system.

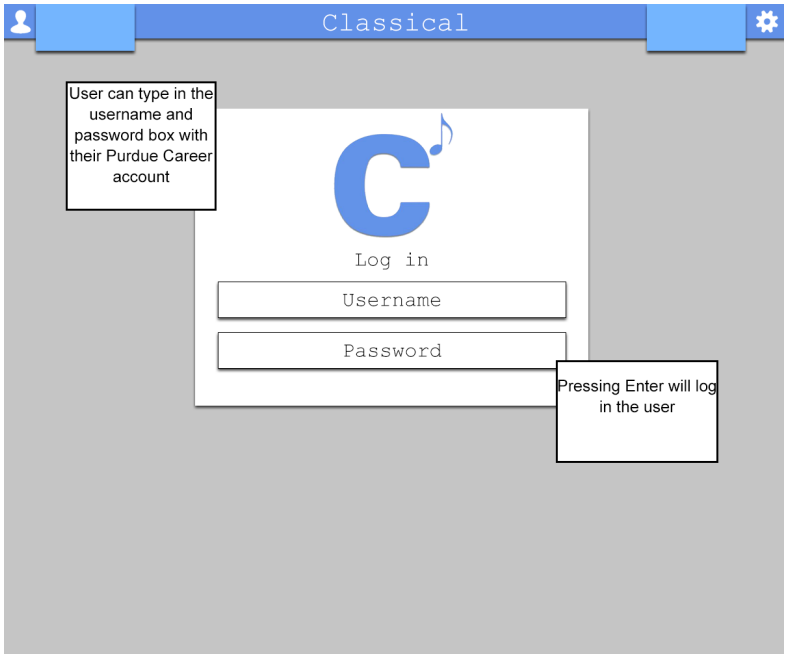
Event

Events can be created by students and given a name, time, location, and whether it will be recurring. This can be edited by the event creator. Students will have the option to join the event, adding their name to the list, or to leave the event and remove it. It will be visible on their calendar for that course.

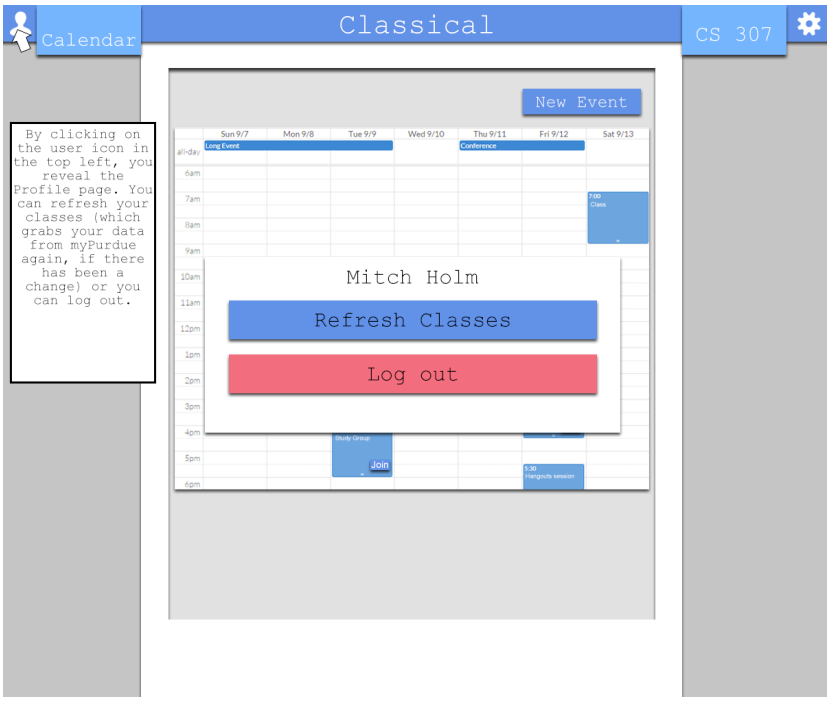
Mockups

Web Mockup

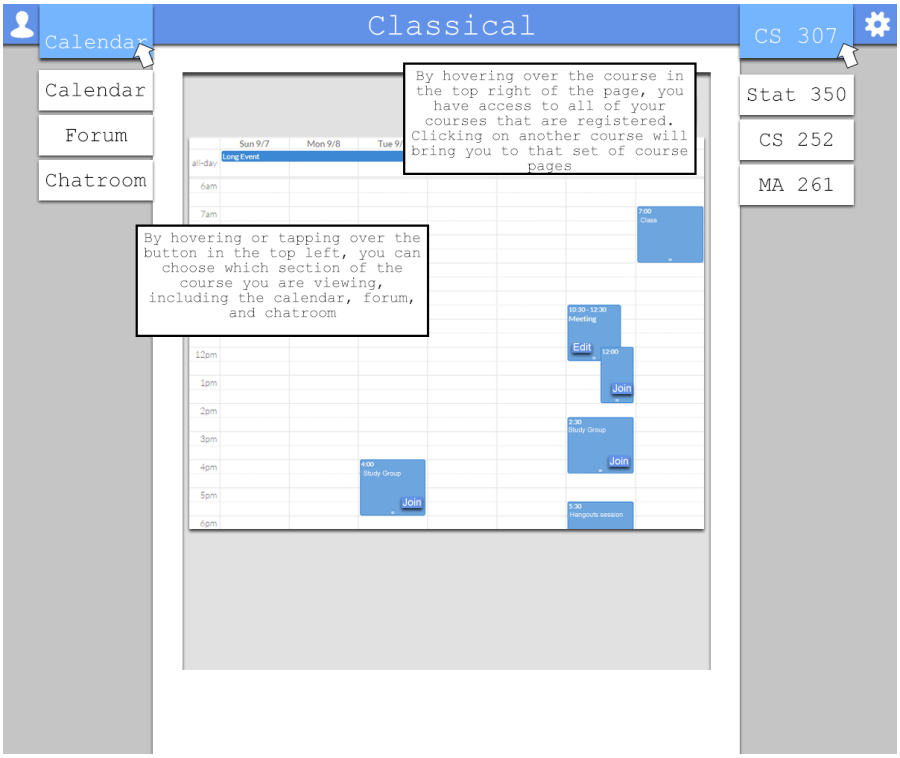
Logging in



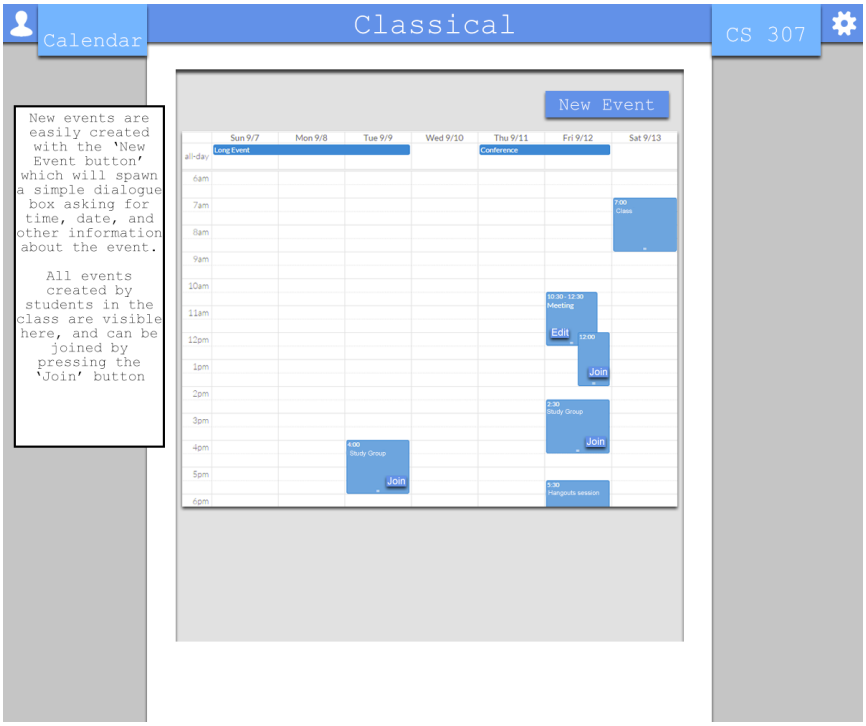
Viewing profile after logging in (in front of calendar)



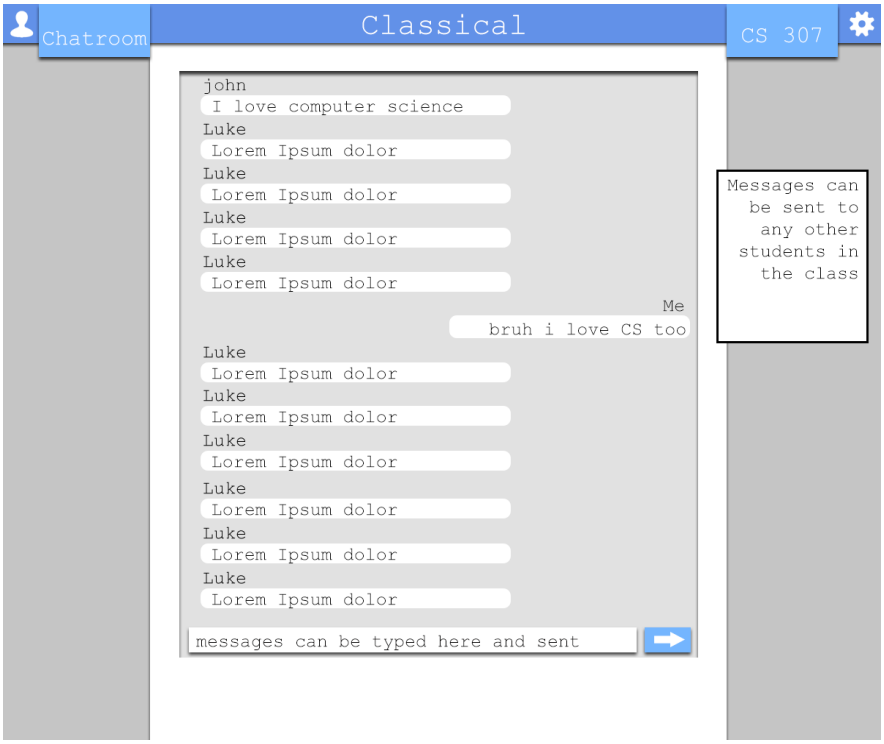
Navigating the menus (in front of calendar)



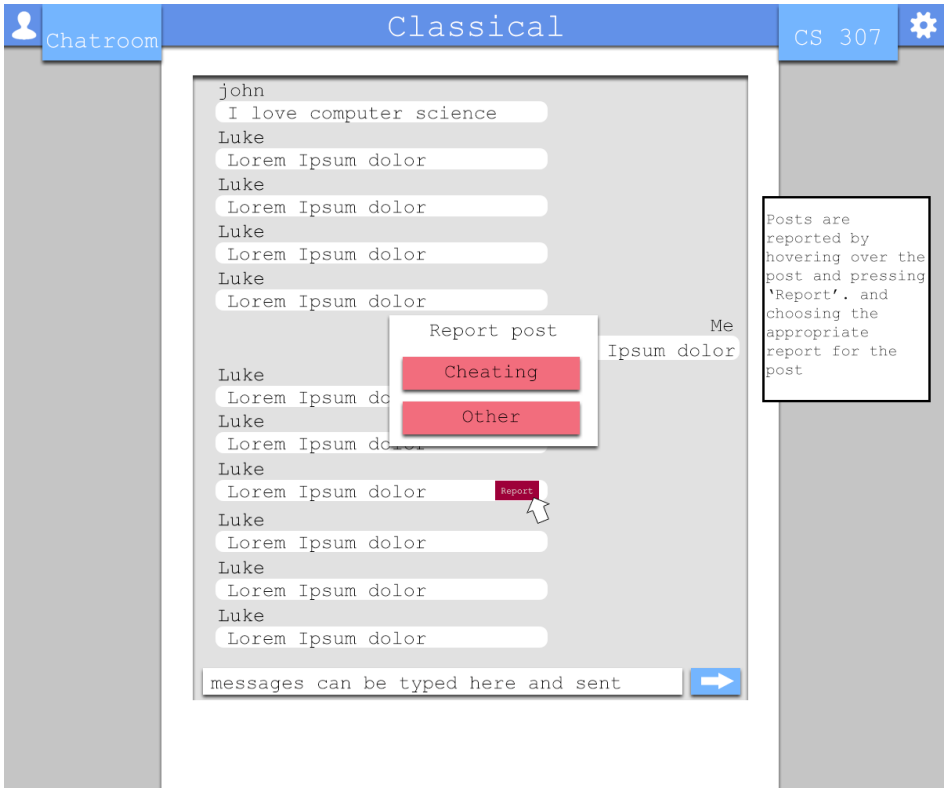
Calendar View



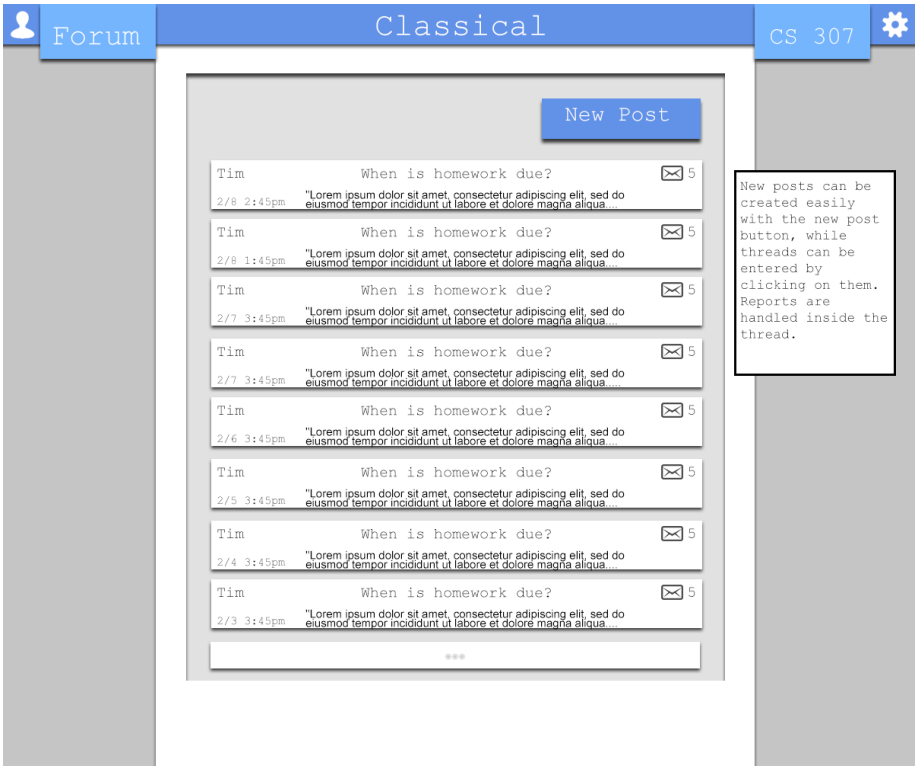
Chatroom view



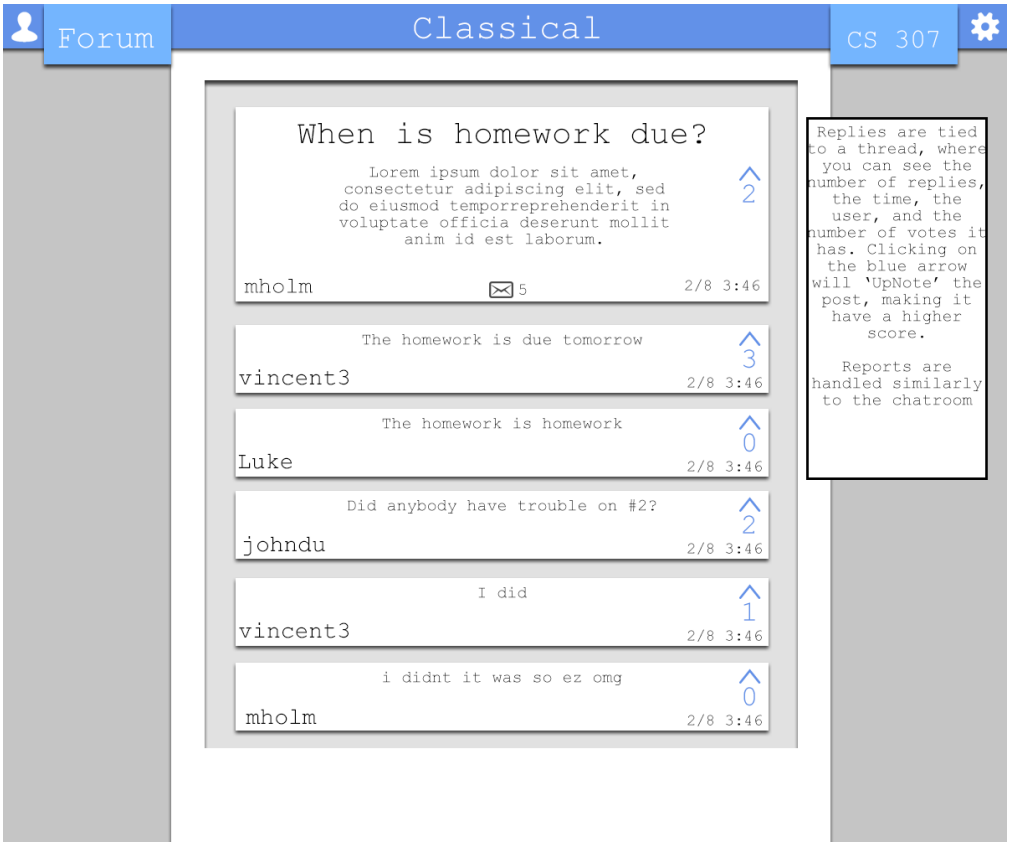
Reporting view



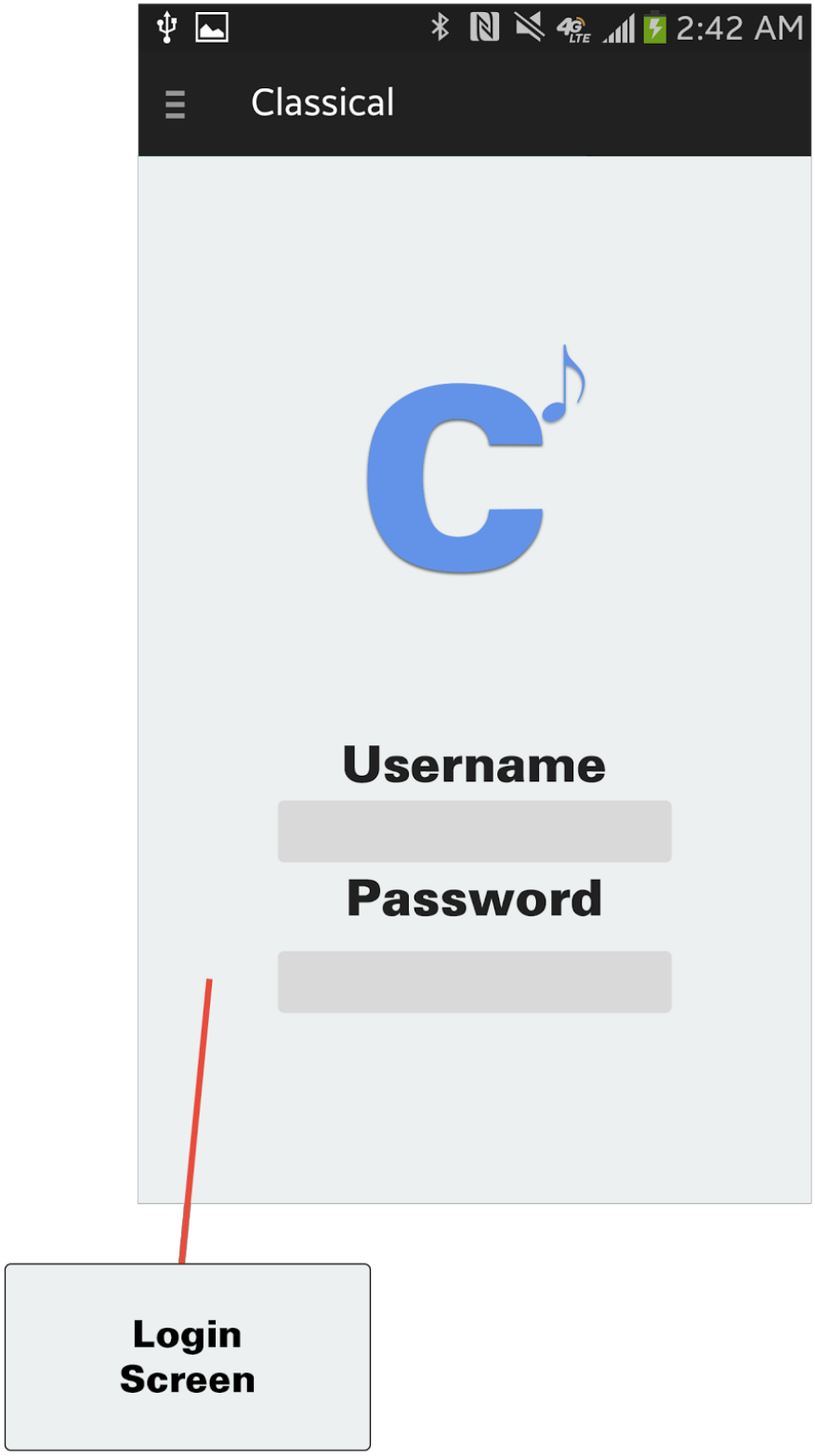
Forum view

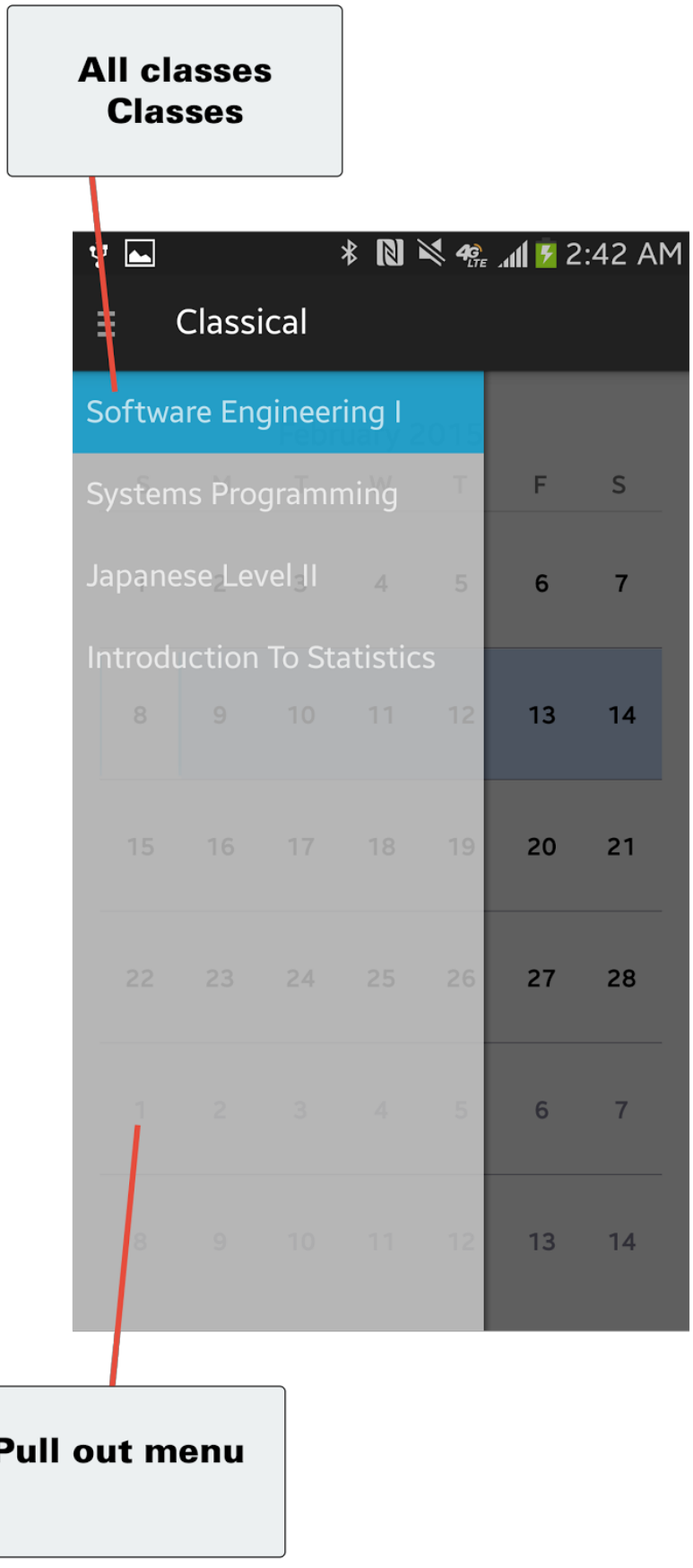


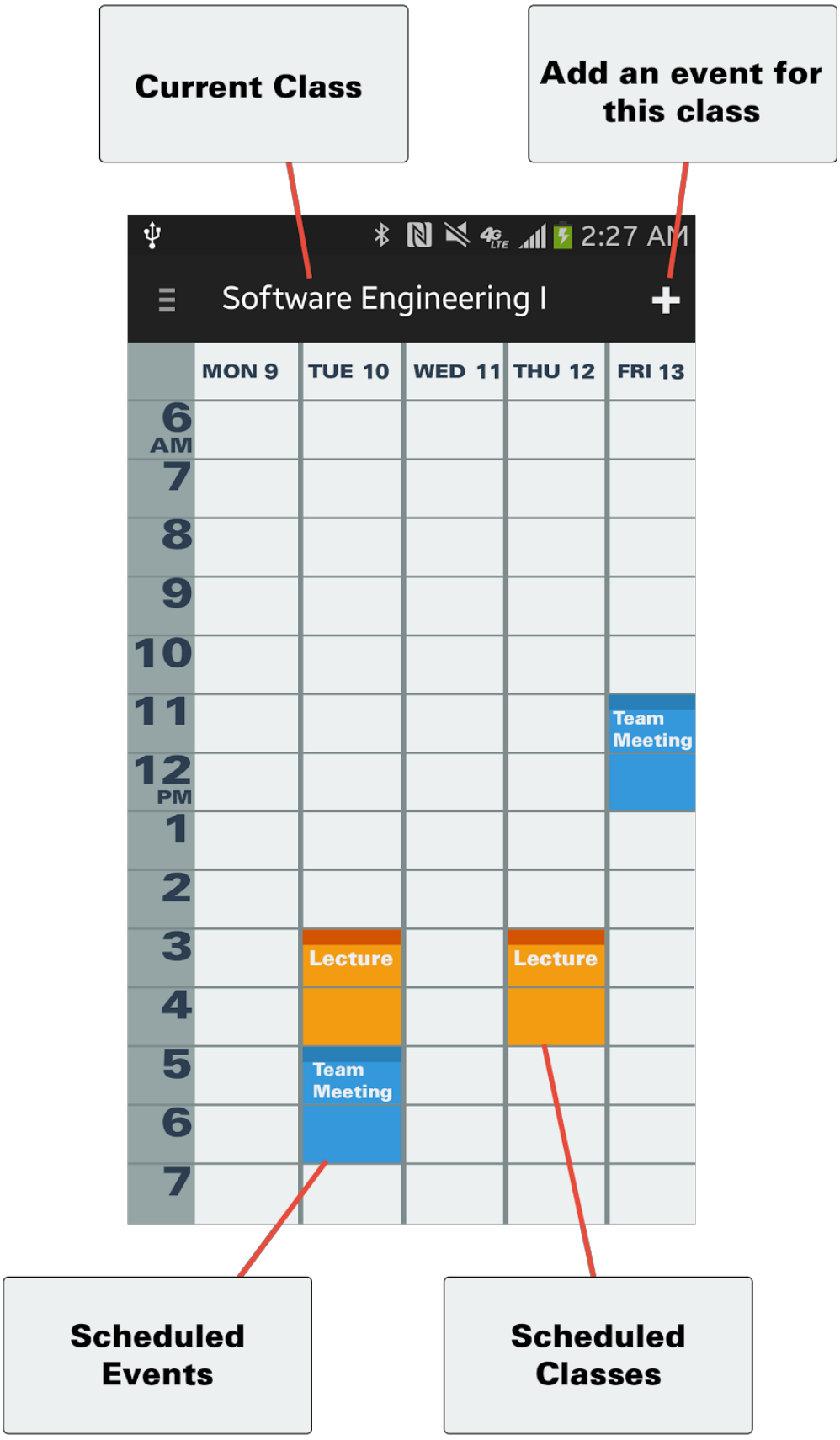
Thread view



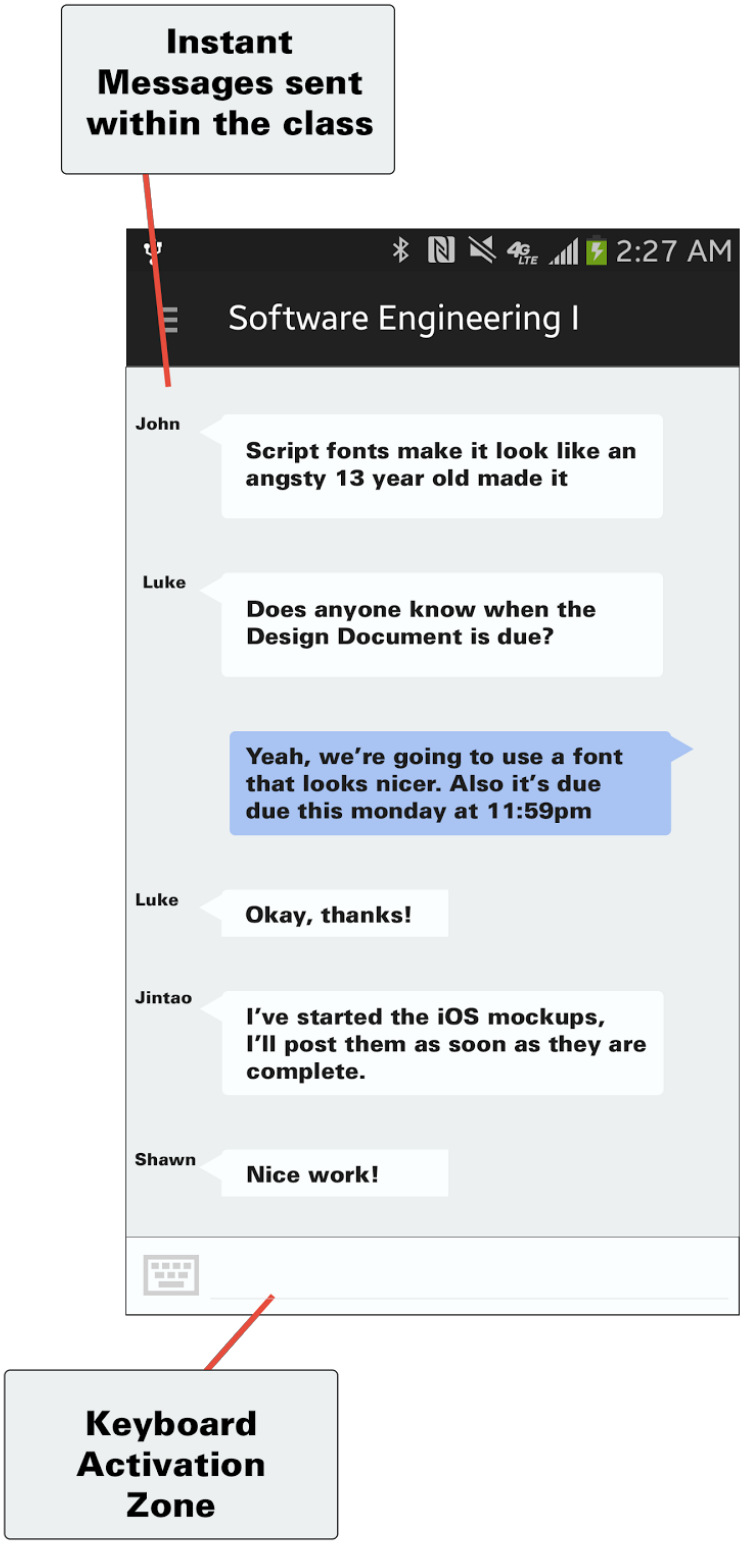
Android Concepts :



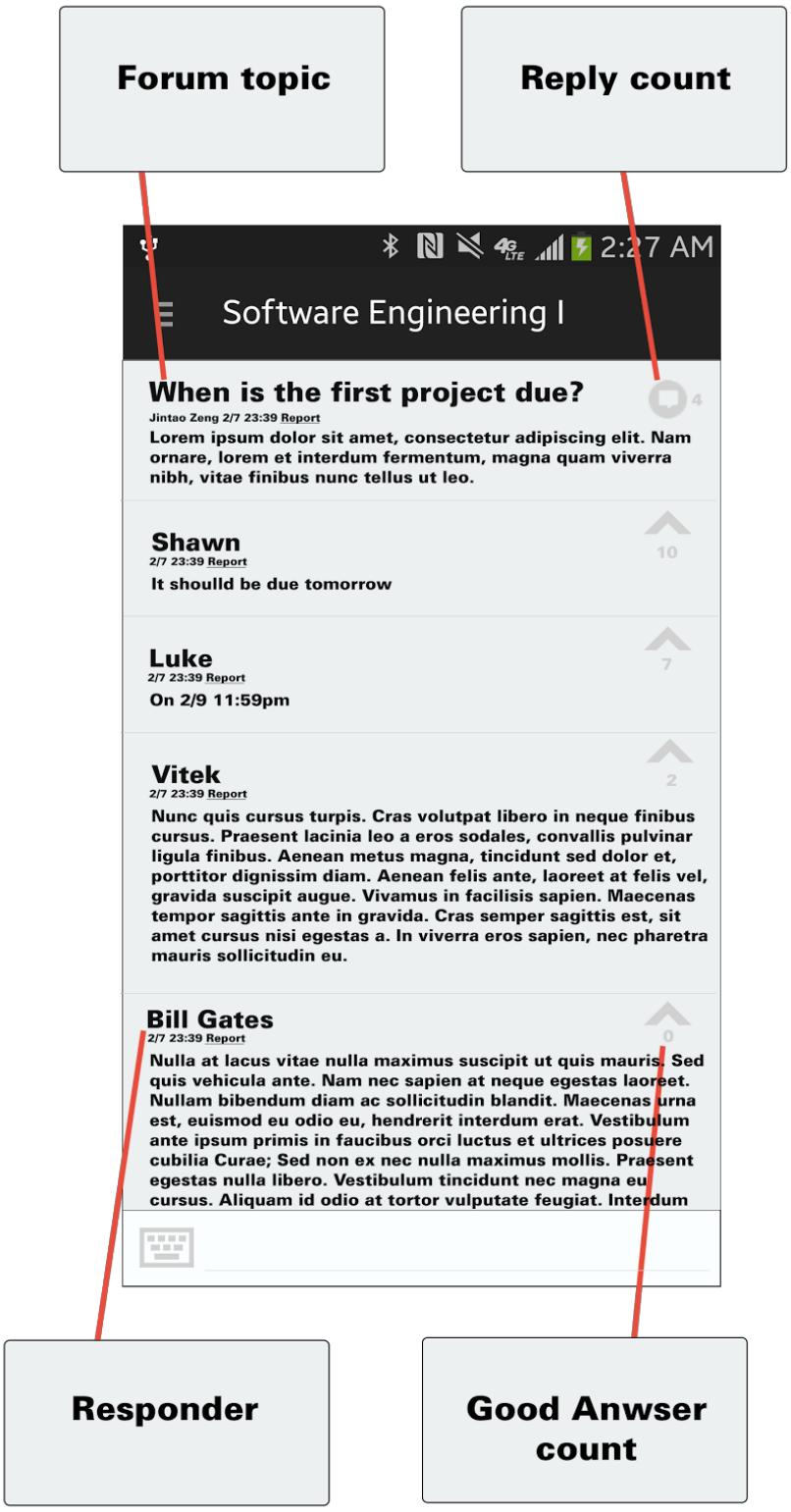

All classes
Classes
2:42 AM
Classical
Software Engineering I
Systems Programming
Japanese Level II
Introduction To Statistics
F
S
6
7
13
14
20
21
27
28
6
7
13
14
Pull out menu


Current Class
Add an event for this class
2:27 AM
Software Engineering I
MON 9
TUE 10
WED 11
THU 12
FRI 13
6 AM
7
8
9
10
11
12 PM
1
2
3
4
5
6
7
Team Meeting
Lecture
Lecture
Team Meeting
Scheduled Events
Scheduled Classes

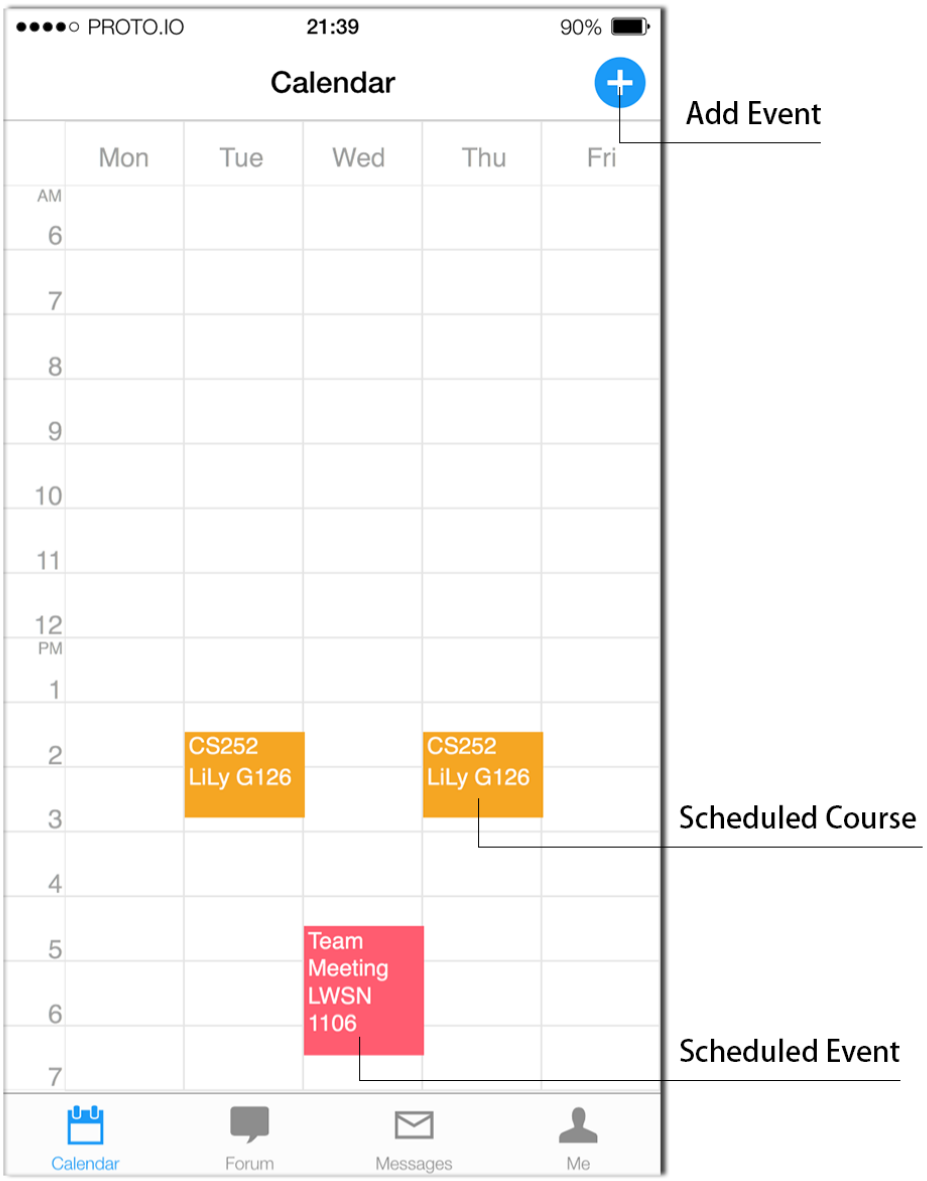
By Swiping left from the calendar view we are brought to the Instant Messenger view

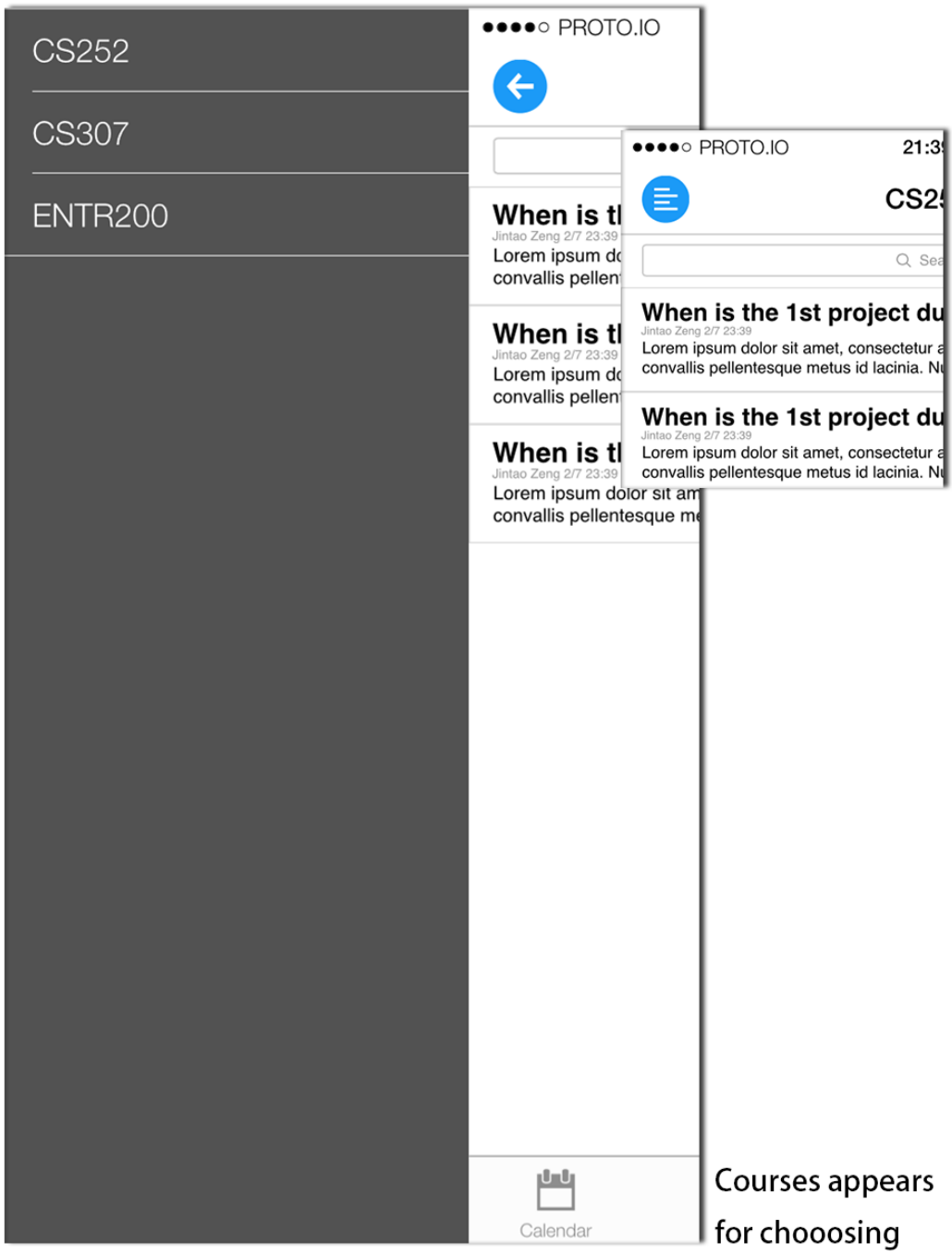


By Swiping left from the Instant Messenger view we are brought to the Forum view

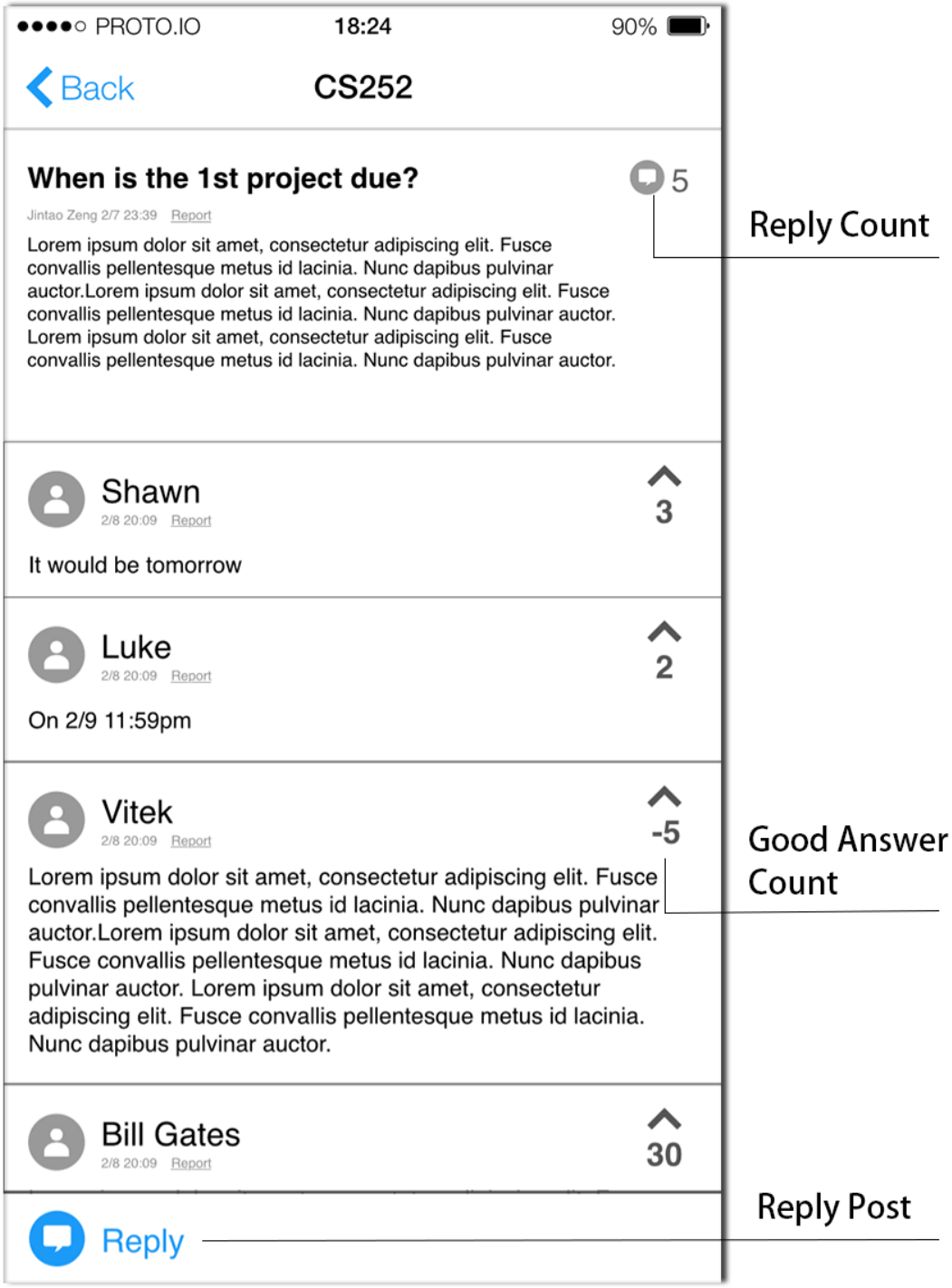


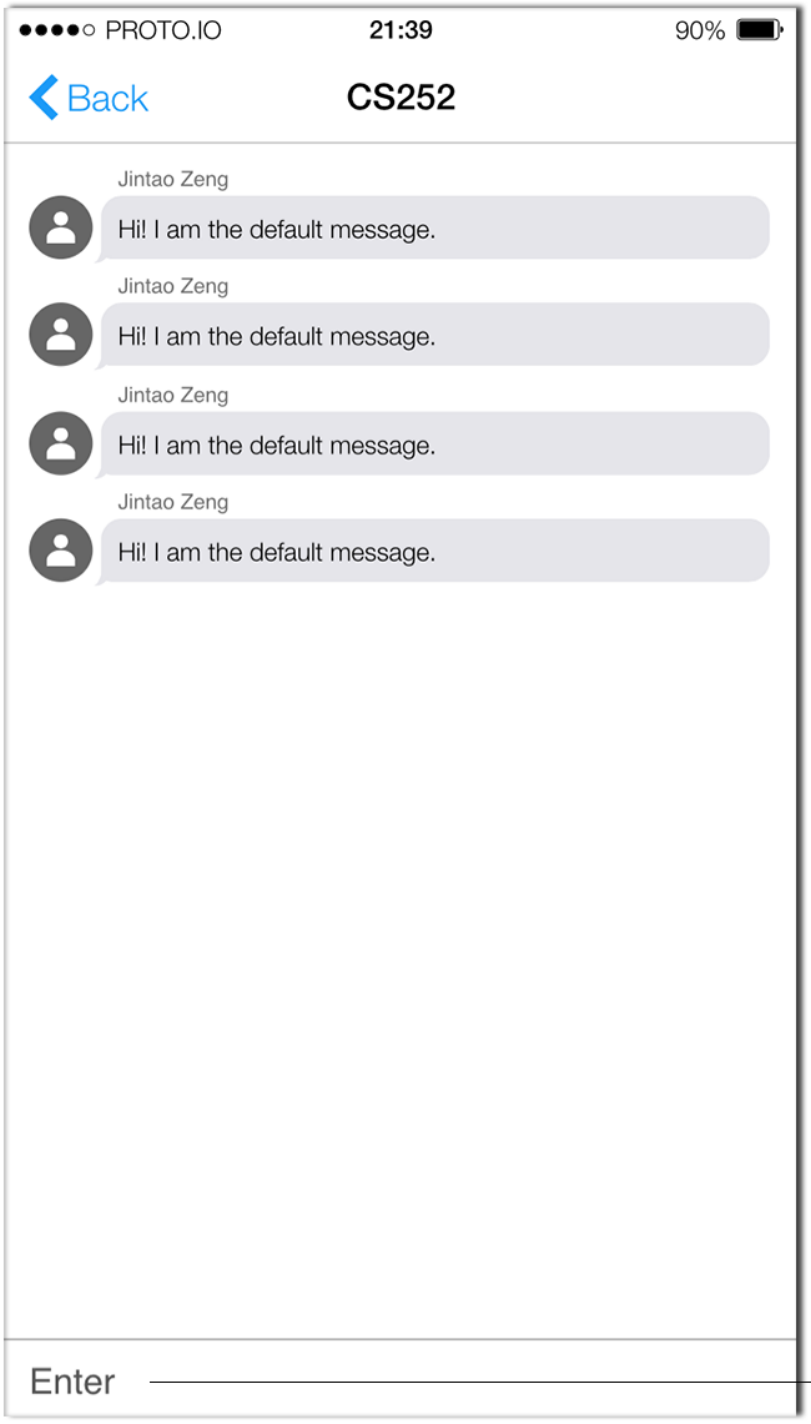
iOS mockups


PROTO.IO
21:39
90%
Calendar
Add Event
Mon
Tue
Wed
Thu
Fri
AM
6
7
8
9
10
11
12
PM
1
2
3
4
5
6
7
CS252
LiLy G126
CS252
LiLy G126
Scheduled Course
Team
Meeting
LWSN
1106
Scheduled Event
Calendar
Forum
Messages
Me



Courses appears for chooosing


●●●●○ PROTO.IO
18:24
90%
Back
CS252
When is the 1st project due?
5
Jintao Zeng 2/7 23:39 Report
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce convallis pellentesque metus id lacinia. Nunc dapibus pulvinar auctor.Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce convallis pellentesque metus id lacinia. Nunc dapibus pulvinar auctor. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce convallis pellentesque metus id lacinia. Nunc dapibus pulvinar auctor.
Reply Count
Shawn
2/8 20:09 Report
3
It would be tomorrow
Luke
2/8 20:09 Report
2
On 2/9 11:59pm
Vitek
2/8 20:09 Report
-5
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce convallis pellentesque metus id lacinia. Nunc dapibus pulvinar auctor.Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce convallis pellentesque metus id lacinia. Nunc dapibus pulvinar auctor. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce convallis pellentesque metus id lacinia. Nunc dapibus pulvinar auctor.
Good Answer Count
Bill Gates
2/8 20:09 Report
30
Reply
Reply Post



Instant Message Sent within the class

Keyboard Activation Zone