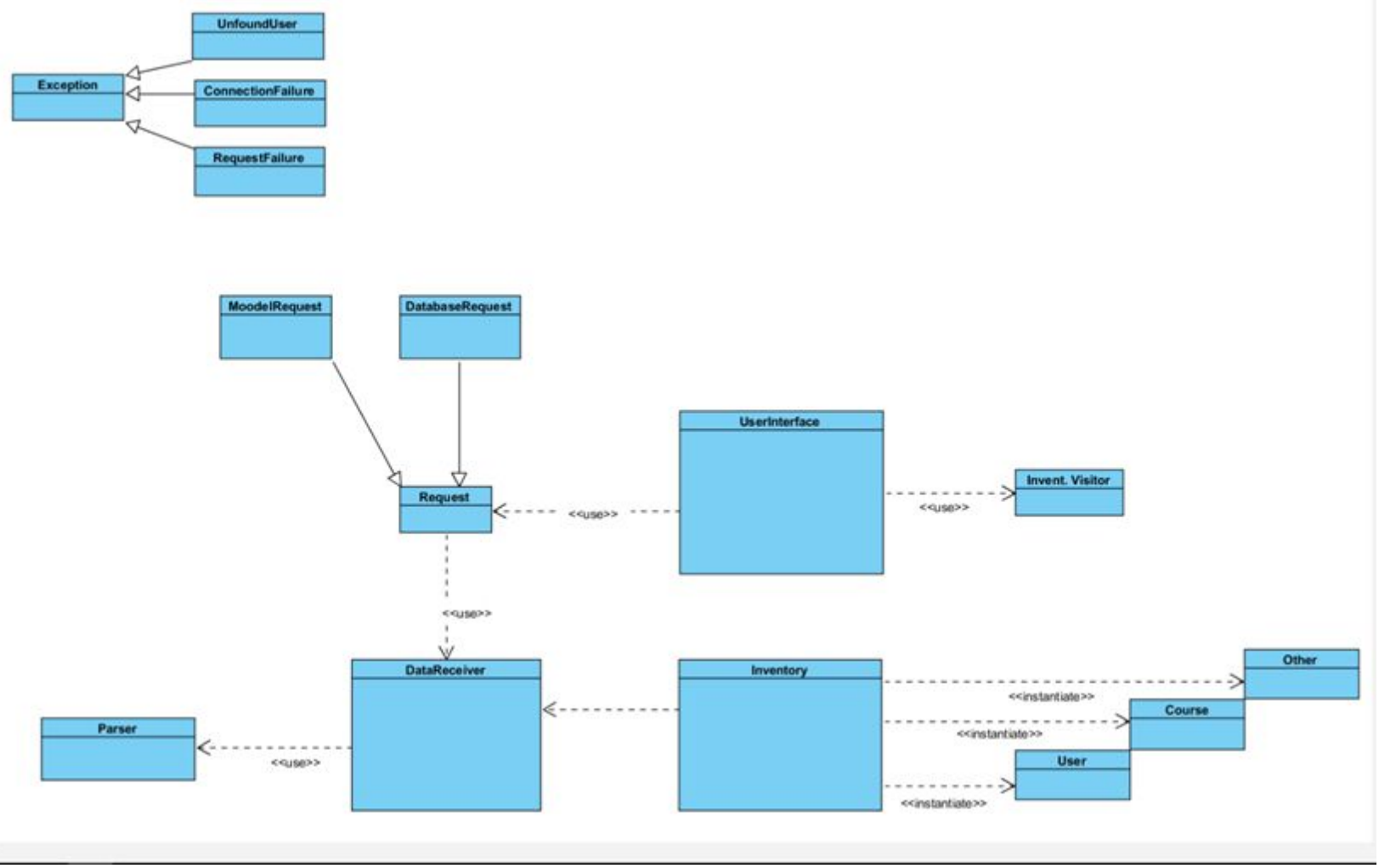
GauchoSpace mobile

Haonan Jiang, Haorui Jiang, Haowen Zhang, Tianyi Ma

Design stages:

1. Idea
    a. The first stage of our project is coming up a idea. Our goal of this project is to build a app that can help UCSB students to login to their gauchospace accounts and get those information that they need quickly and efficiently. With the current situation, students in UCSB have to login to their gauchospace with browser on mobile phone and to simply check a deadline of assignment or grade, students need to click multiple button in order to get those information. Our project provides the convenience to allow students obtain those information more efficiently. User only need to login their account to know the assignments, deadlines and grades of all the courses he or she has enrolled for current quarter. Besides that, user can add those courses that using its own website (eg.CS184) to the dashboard in this app. In that way, user can access all the courses information easily.
2. Domain analysis:
    a. User Analysis:
        i. The target group of users using our application is the entire UCSB student body. Providing a mobile version gauchospace will bring tremendous convenience to everyone who is using gauchospace to check their homework deadlines, course slides and grades.
        ii. Therefore our application is designed under the easy-to-use principle; straightforward user interface and functions that meet the very basic but core needs of the user.
    b. Task Analysis:



i. To build our project, we first divide our works that need to be done to several parts, including user interface, request for data, local data structure, parser, and etc. Above is a illustration of our works.

ii. The entire workflow can be captured by use cases in the following order

a.

Use Case 1: Log In
Actor: User
Basic Flow: The app will direct the user to the gauchospace webpage and the user should enter his/her identifications there. Afterwards, the app navigates the user to the dashboard where he/she can look up courses information in details.
Alternative Flow: User will be asked to enter again if

account name or password is incorrect

b.

Use Case 2: Html Parse
Actor: Html Parser
Basic Flow: The html pages from gauchospace webpage will be parsed in background. All necessary information will be collected at this point and no further request or parsing is required. The result of this process is a Inventory Tree.
Alternative Flow: N/A

c.

Use Case 3: Visiting Inventory Tree
Actor: Inventory Tree Visitor
Basic Flow: Visitor traverses through the Inventory Tree and wrap data into Information Class that will be used to form views
Alternative Flow: N/A

d.

Use Case 4: Add a Course
Actor: User; UI
Basic Flow: User adds a course to the dashboard. The UI will ask the user to enter the course name and the website it links to
Alternative Flow: N/A

e.

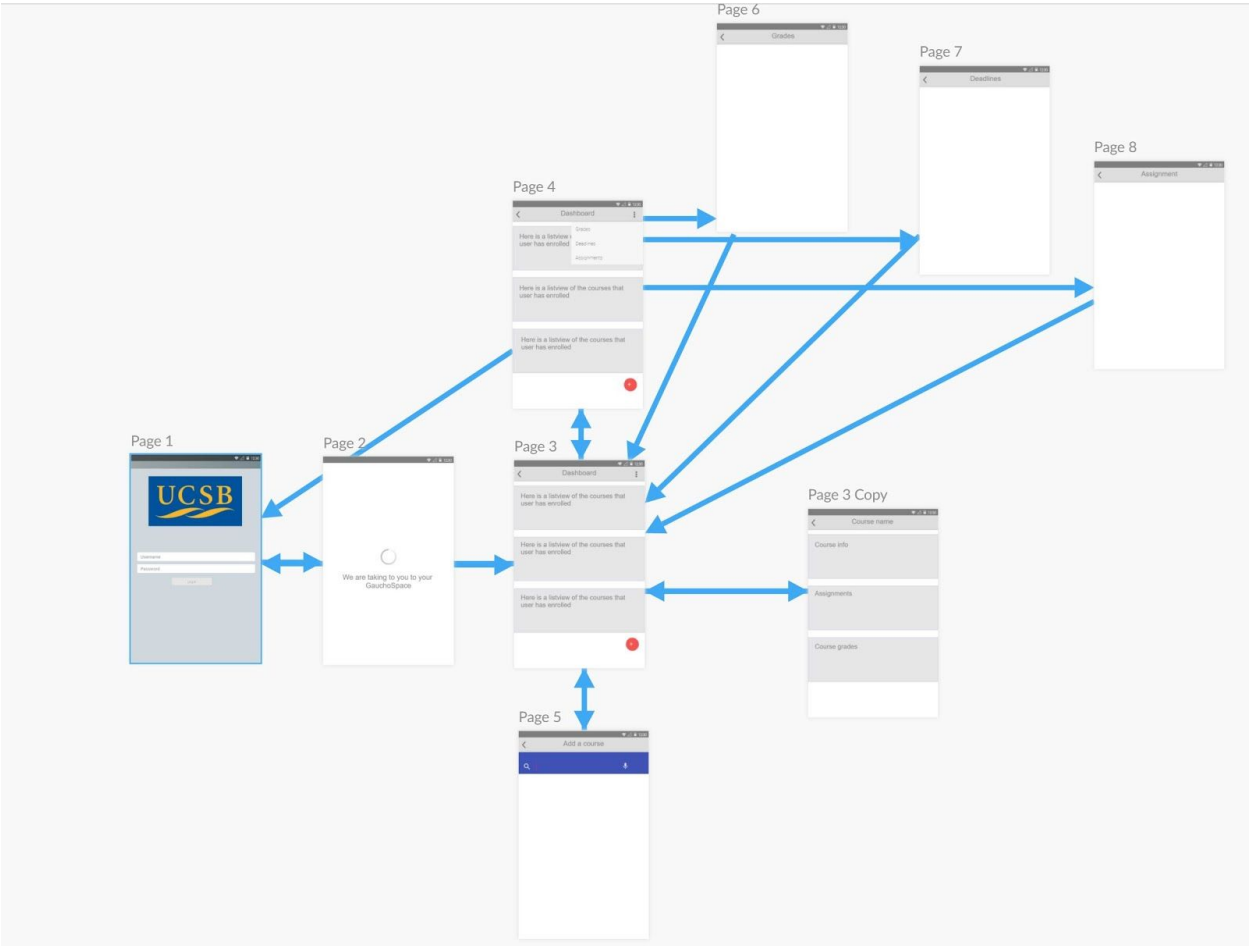
Use Case 5: Look at Course Information
Actor: User, UI
Basic Flow: After the user click the course name on the dashboard, the UI will navigate the user to the courseinfo view.
Alternative Flow: If the course clicked is added by

the user, instead of going to the courseinfo view, the UI will navigate the user to the website that the course is linked to.

f.

Use Case 6: Go back button / Logout
Actor: User, UI
Basic Flow: As the user clicked the go-back button, the UI navigates the user back to dashboard. If the user is currently at dashboard, such action will log the user out.
ALternative Flow: N/A

iii. Overall, the functioning process of our designed app includes several steps. The first step is to let user to login their account on gauchospace. Then the app will send request and fetch the html file from the moodle. Next, we build a parser to parse all the information that we need and store to our local data structure. Then a visitor can visit this data structure and put information to the appropriate location for user to view. A simple illustration of our UI is shown below.



Design decisions:

It mainly uses Activity as UI support, and divides pages based on functional modules. It controls page interaction and interaction behavior through Activity's jump and data transmission. The UI of the page and the control are implemented with the XML layout file, each of which corresponds to the layout file of a page. The display of the list uses a ListView control, and each of the lists uses the corresponding layout file for UI control.

The decision of using a tree as data structure is obvious, since the assignment and its details (eg. deadlines, grades and etc.) belong to a specific course node, and the course node belongs to a specific user node. A visitor that we used is convenient to iterate the whole data tree and get the data that we need.

Implementation difficulties:

Getting the raw data from GauchoSpace is never easy without a well-structured API documentation. So we decided to access the GauchoSpace html pages, which contain

all the data we need, and parse them. Although this is not a very smart way to get data and it consumes a lot of time waiting the webpage to be transferred to local, that is the only we can have access to the data on the GauchoSpace. Since we use html parser to get data, the coherence of the html dom structure is very important to us. For example, some homework has “due date” field and some doesn’t. So that’s caused many problems to us but finally we handled them all.

Besides the difficulties of getting data from html, another major concerns is the time of parse data and creating data structure (eg. nodes in our data structure tree). The time of parsing and creating data nodes depends on connection of the Internet heavily. We make a loading activity to wait 25 seconds for the parsing process and creating data node process. We believe that it can fix this issue. But this implementation also require a longer debugging time for us.

External resources:

We used Jsoup(https://jsoup.org/) library to help us parse html doms and make http requests. We include *compile 'org.jsoup:jsoup:1.11.2'* in the gradle file. A lot of dom manipulation are performed with the help of method under “org.jsoup” package, and we really appreciate the Jsoup author’s great work.

The UI build process is full of the use of Android native controls and resources, and no external resources are used.