CS307 – Design Document

Purdue Course Assistant

Team Members:

Chen Fang
Zhengshang Liu
Ransen Niu
Jiaping Qi
Ziqi Wang
Hao Xu

Contents

1. **Purpose**

Overview: Presently, students have myPurdue to look up for courses and advisors to help scheduling. However, the information provided by myPurdue and advisors is too limited for students scheduling based on their own will. Our program intends to offer students another way to be familiar with courses and professors before registration. Users will have an overview of class based on reviews from other students and sample questions.

Our program will achieve the following functionalities:

i. Rating system: Except to browsing detailed information about courses and academic tracks, students are able to rate professors and courses, view ratings and add comments.
Users are able to:

- rate each course
- write comment for each course
- see the rating and comment for each course
- check the official information of courses
- view courses in categories of department

ii. Pre-test system: Students are able to get familiar with course topics through taking a pre-test which consists of up-to-date questions drawn from previous semesters.
Users are able to:

- take a pre-test for each course and get graded
- receive email about updating pre-tests

iii. Recommendation System: Students are able to receive recommending schedules from the app, through submitting their taken courses, academic tracks and personal interests. Besides, they are able to add tags to courses.
Users are able to:

- view and filter out courses based on tags
- tag each course with specific characteristics
- get recommendation courses based on course history and personal interests

iv. User Account: Students are able to create their own accounts, customize their account information and keep their activity history.
Users are able to:

- create an account

- change their password
- find their password or username if they forget them
- view other’s profile
- add taken courses to their profile

2. Design Outline

Overview and Components

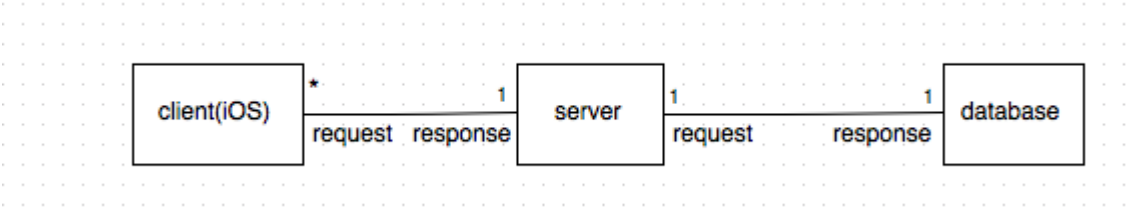
Our program will implement client-server architecture with the following components

- Server: The server is the underlying system that manages the requests and the information. The server is connected to both client and database, and handles request from clients.
- Client (Application user): The client would be written in iOS. It would allow users interact with server. The client also includes organized user interface for users' convenience.
- Database: The database would store all the data used in the program. It is only connected to the server and responses to the server. The data includes the course and related professor information, the pretest data for the course and other collected information from users.

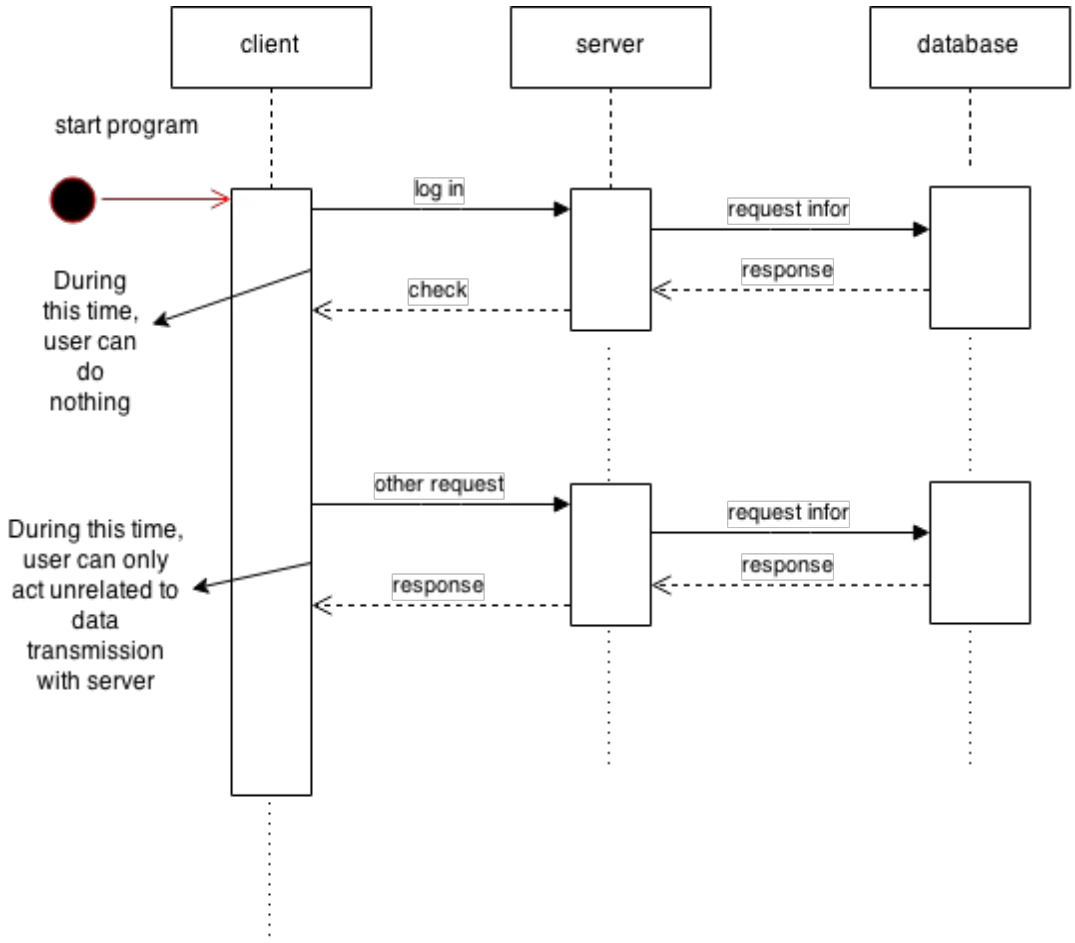
Architecture

The application will based on the common client-server architecture. For the client, it is based on the multi layer hierarchy.

- Client-Server Model(High level): At one moment, multiple clients can be connected to the server when users need more information or action. When received the request from client, server needs to retrieve the related information from database and then returns the analyzed information back to the client.



- UML Sequence Diagram(broader view): When the program starts, the first step is to login and the server would compare the login information with the corresponding information it retrieved from database, and then it would decide if it can allow user do any further action in the program. After the user has done the first step, it can send multiple requests to server for responses, like viewing user's profile and displaying all the courses.



3. Design Issues

1. Fat-server or thin-server
   Option 1: Using fat-server
   Option 2: Using thin-server
   Decision: We would like to choose a fat-server because saving most data at client side requires too much memories. Moreover, making a recommandation need the information of a large range of course data. So that, we choose to put most data and calculation part at server side.

2. How to help user to get their password back?
   Option 1: Use pre-defined Q&A
   Option 2: Send a confirmation link to user's e-mail
   Decision: Option 2 was chosen for two reasons: 1). Adding Q&A will increase the size of database also the complexity of searching in database. 2). User might also forget the answer for the question they pre-defined, however, the e-mail will always be a good way.

3. TCP connection or UDP connection
   Option 1: TCP connection
   Option 2: UDP connection
   Decision: We decide to use TCP connection to connect server and clients because package lost is unacceptable. Also, small delay will not affect the performance much and we can use pre-store technic to prevent those delays.

4. How to organize the comments?
   Option 1: list the 10 comments which users added recently
   Option2: Randomly list the 10 comments from the users
   Decision: Option 1 was chosen because old comments may not include up-to-date information and consequently are less valuable than the most recent ones.

5. How to grade the pretest that user finished.
   Option 1: Download a copy of correct answer when download the pre-test questions. Add a grader class that is able to compare the correct answer with user's answer and generate the result locally.
   Option 2: Add a grader class that is in the server and user, which is able to compare the correct key in the database with user's response, and then send the correct answer back.
   Decision: Option 1 was chosen since though the whole workload is the same, however, it can significantly decrease the waiting time after user submitting their answers.

6. How to make recommendation about courses for user?
   Option 1: Add tags to courses and give recommendations based on user's courses history and his preferences.
   Option 2: Give recommendation based on user's year(freshman, sophomore, junior and senior) and his major.
   Decision: We would choose option 1 because using tags to classify courses will give user a more subjective view before they learning more details of that course. Moreover, since we consider the courses history of user, we can easily satisfy the Prerequisites of courses and avoid redundancy.

7. How to add a course in the profile (list, text field)
   Option1: list all classes in the profile and let user choose courses what he/she already taken.
   Option2: give a text field and let user type the courses what he/she already taken.
   Decision: we would choose the opinion 2. Because option 1 have to let user select each course by himself, it would waste too much time. Like user who is junior or senior, he may has already taken more than 25 courses. But for the option 2,user just need to type courses he has done in the text field, it would save user much time that the opinion 1.

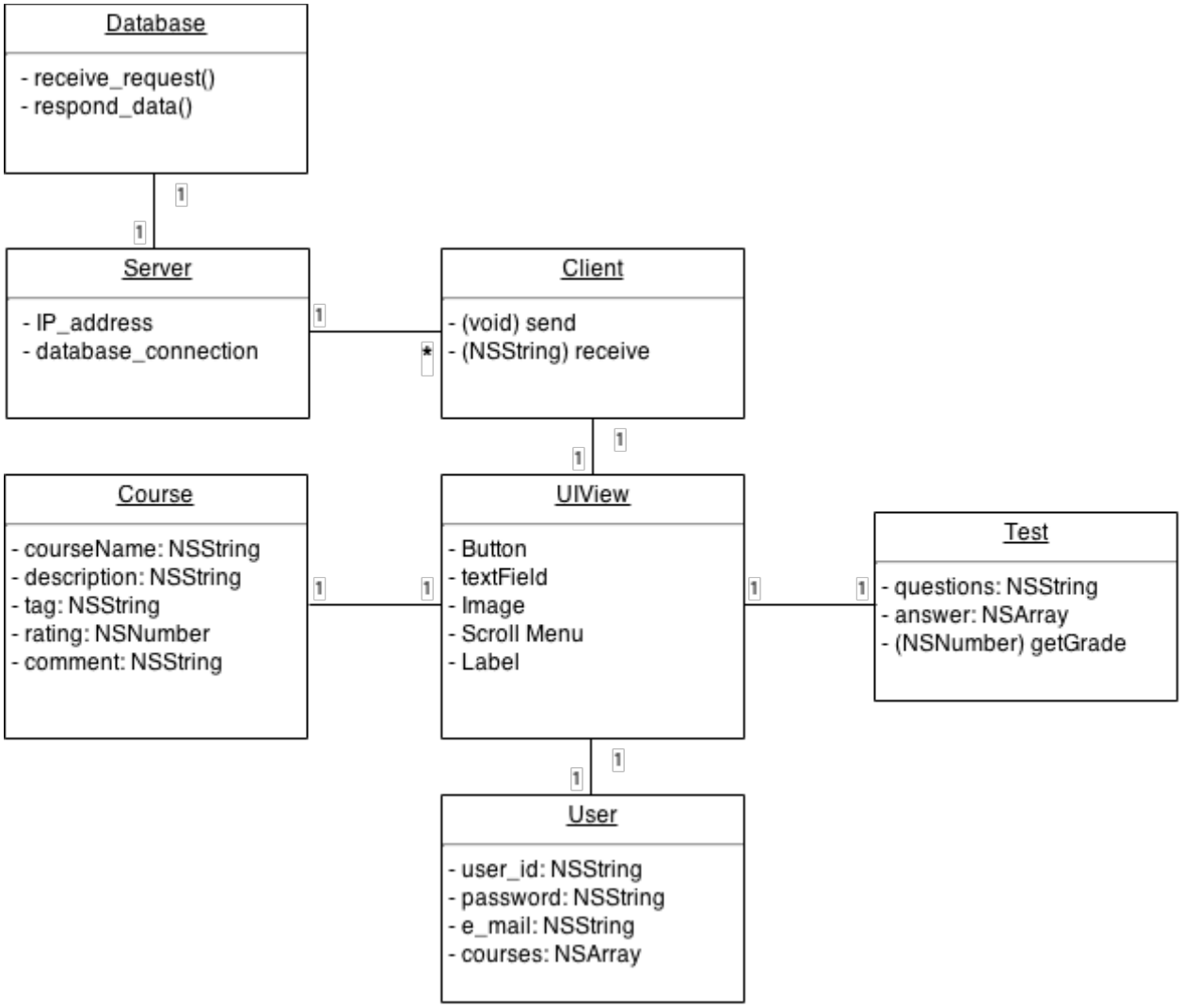
8. How to generate the course selection?
   Option1: first show the department first, then list all of courses under this department.
   Option2: list all of courses directly and allow user choose the course by himself.
   Decision: We would choose the opinion1. Purdue university may has thousands of courses. If we just list all of them in one list , user would to be hard to find the courses. But if we first list the department first and using the first alphabet searching on iPhone, people would easy to find the department. And user would clearly find the courses under the department.

9. How to generate the grade display?
   Option 1: only show the grade.
   Option 2: list all of question whether user answered correct or not with explanations.
   Decision: We chose option 1 for two reasons. Firstly, it is clear and easy to display. Secondly, the purpose for the pre-test system is letting users to have a basic idea on how well they are ready for the course, it is not necessary to display all the answers again.

10. How to display pretest questions?
    Option 1: One question per page
    Option 2: Show all the questions in one page
    Decision: We chose Option 2 because it will make user easier to reach any question they want. Also, it is easy to implement in consider of the time and people limitation.

4. Design Detail

1. Class Design



2. Class Detail

- MainView:
  A) MainView class is the start point of the app.
  B) Access popover login page which allows user interact through text fields, send content to Client class.
  C) Access to CourseDicView, UserProfileView and CourseFilterView.

- CourseFilterView:
  A) CourseFilterView class receive data from Client class and generate user interface accordingly.

- CourseInfoView:
  A) CourseInfo class receive data from Client class and generate user interface accordingly.
  B) Displays course title, overview rating and the most popular comments.
  C) Access feedback view which allows users interact through a words limited text field, and send the content to Client class.
  D) Accessable from CourseDicView

- UserProfileView:
  A) UserProfileView class receives data from Client class and generate user interface accordingly.
  B) Displays username, nickname, course history and marked courses.

- TestView:
  A) TestView class receives pre-test data from Client class and generate user interface accordingly, receives feedback from PreTest class to display test feedbacks.
  B) Allows user interact through checkboxes and buttons to submit answers or cancel a test.

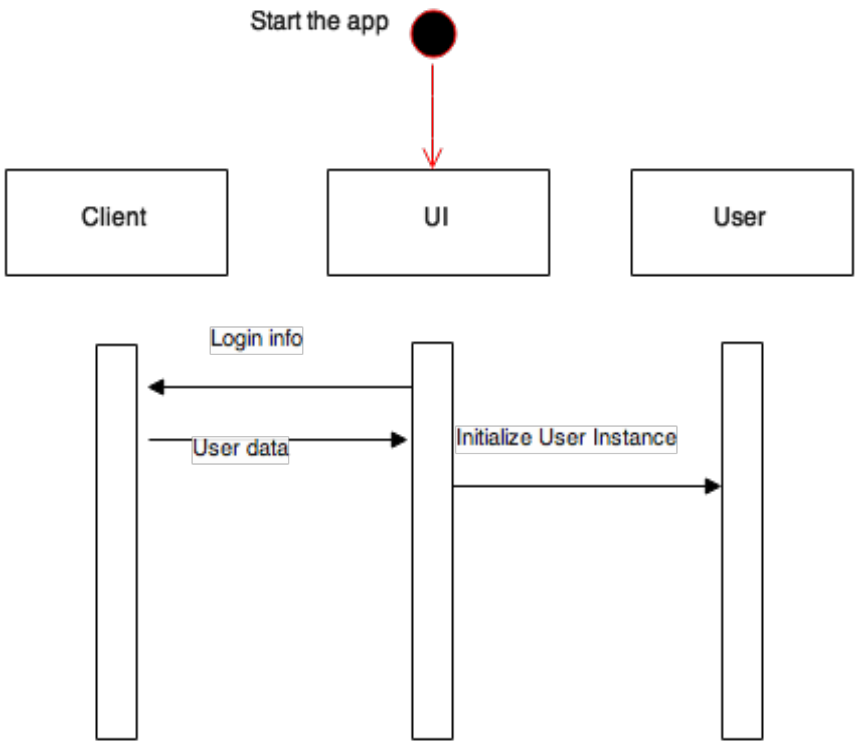
- PreTest
  A) PreTest class receives data from Client and TestView class, comparing user answer receiving from TestView class, with preset solution receiving from Client class, and send feedback to TestView class.

- User:
    A) User class contains current user's information including user_name, user_pw, user_email, taken_courses and set(), get() functions for every element.

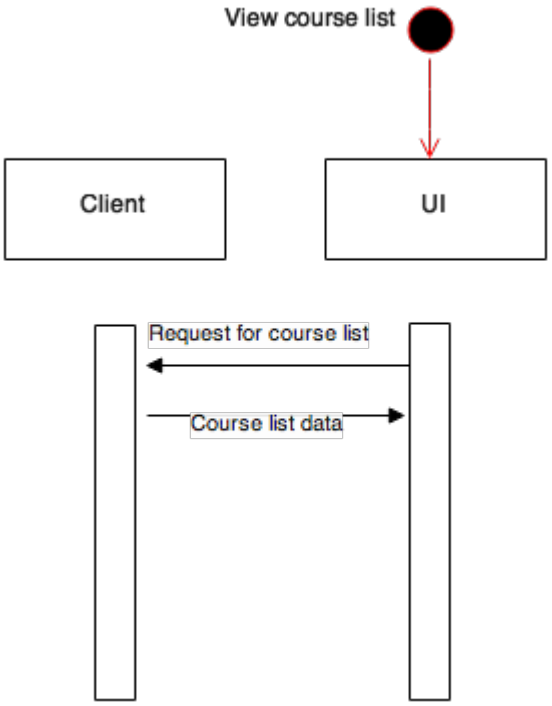
- Course:
    A) Course class contains course information such as course_name, course_detail, course_tags, course_rate, course_comment, course_test, course_test_answer and set(), get() function of all elements.

- Client:
    A) Client class contains all information for TCP connection. It also contains functions to communicate with server including sendmsg(), getmsg(), setconnection(), etc.

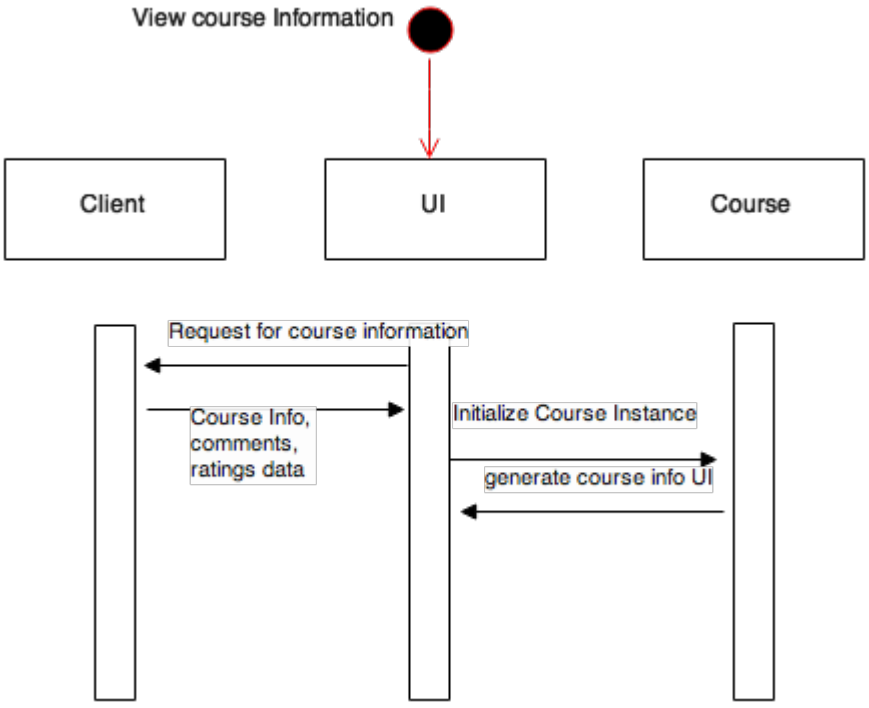
3. Sequence Diagram:



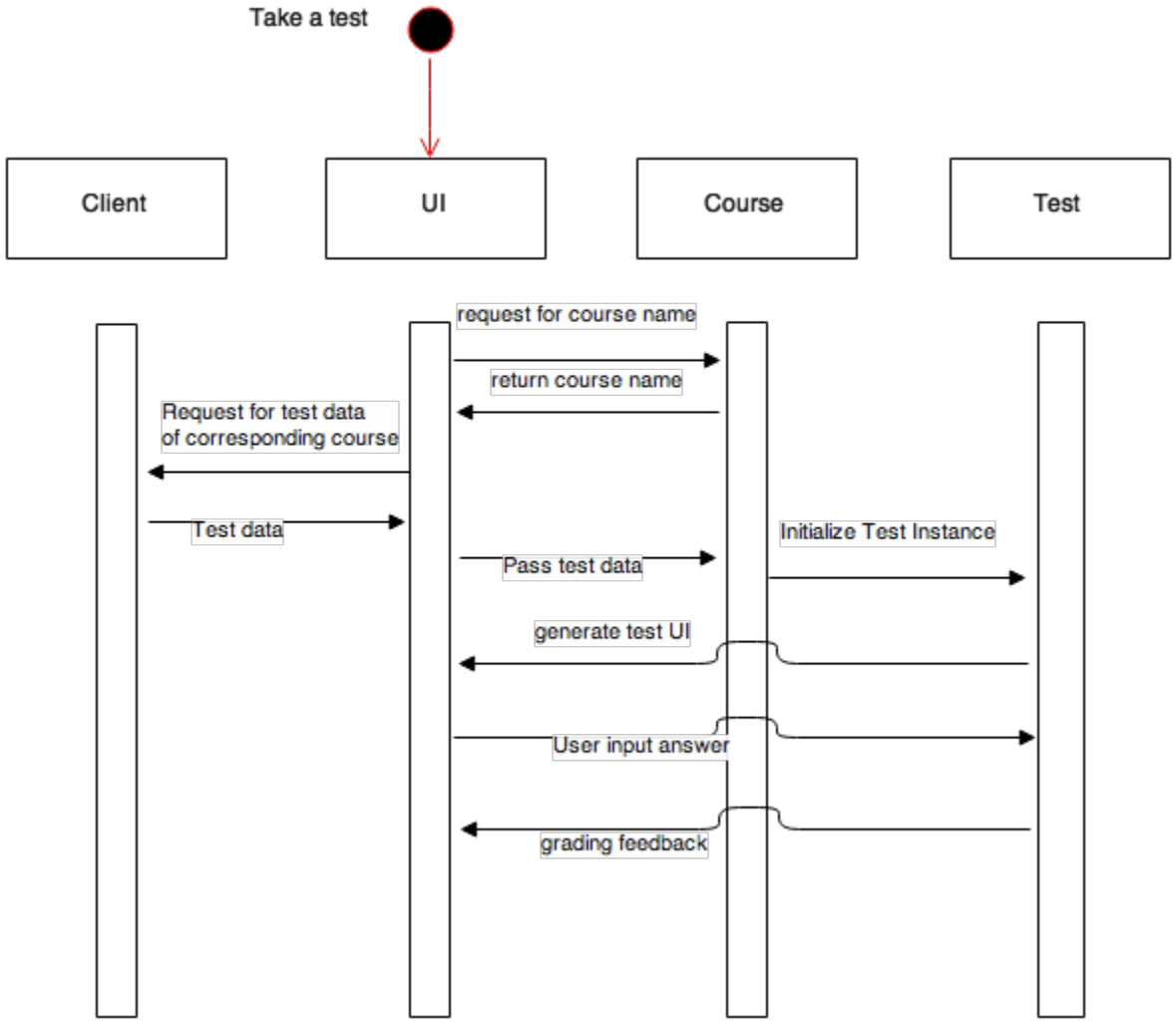
Start the app



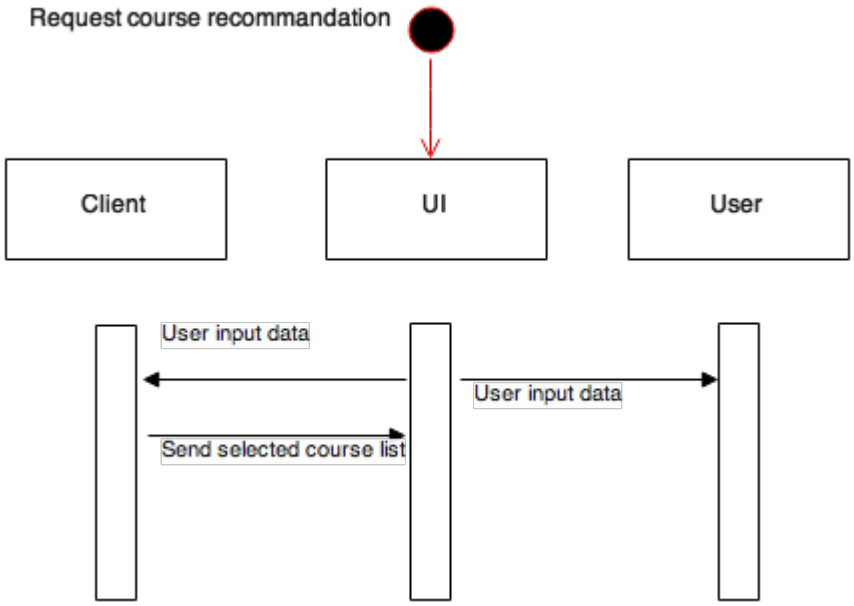
View Course List



View Course Information


Take a test
Client
UI
Course
Test
request for course name
return course name
Request for test data
of corresponding course
Test data
Initialize Test Instance
Pass test data
generate test UI
User input answer
grading feedback

Take A Test

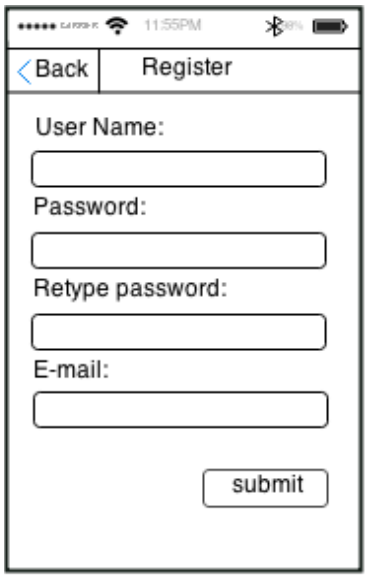


Get Course Recommendation

4. UI mockups :

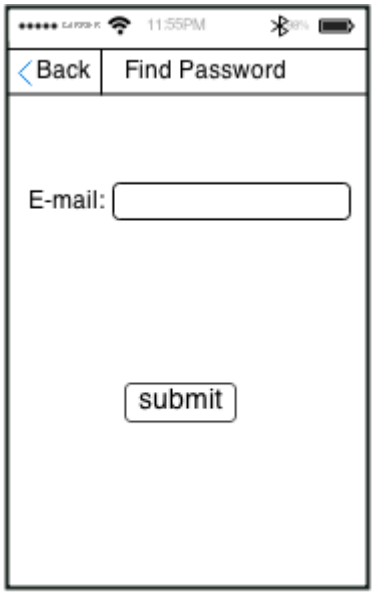
(1) Welcome Page

(2) Login Page

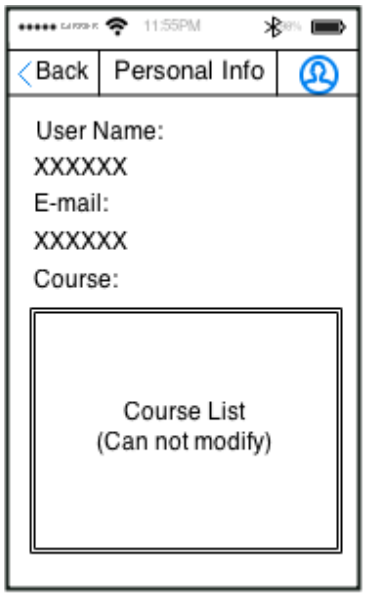


(3) Register Page

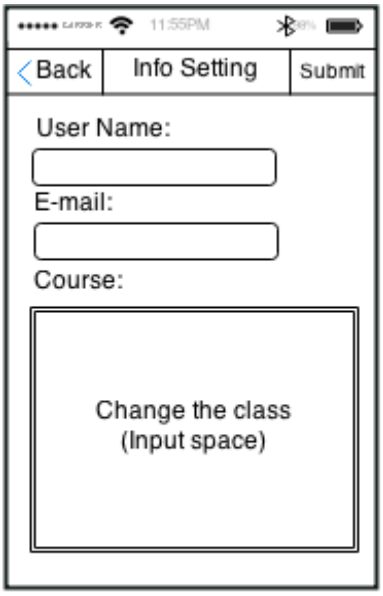
(4) Home Page
(Successfully Login)



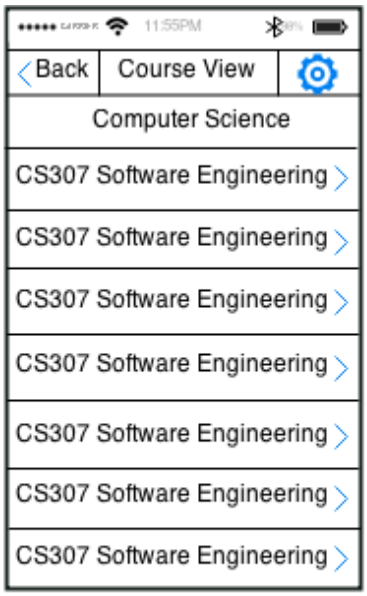
(5) Find Password



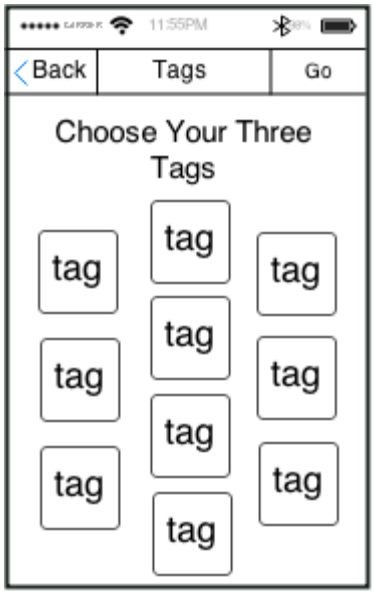
(6) User profile



(7) Modify Profile

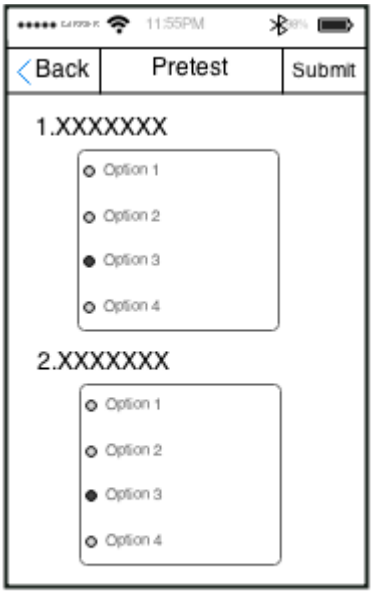


(8) Course View



(9) Recommendation Tags

(11) Course Info



(12)Pretest

(13)Pretest grade


11:55PM
Back
Feedback
Submit
CS307
Rating:
4.9 / 5.0
Tags for this course:
TAG
TAG
TAG
TAG
TAG
TAG
TAG
TAG
TAG
TAG
Comment:

(14) Feedback