CS 3704: Intermediate Software Design

Group 2

TA Queue

Design II: Mock UI & Algorithm Design

Sam Lightfoot, Pasha Ranakusuma, Omar Elgeoushy, Thejus Unnivelan

Check out our GitHub Project here:

https://github.com/Intermediate-Software-Design/ISD-Group-Project

April 15, 2022

Contents

Chapter 1

Mock UI

Please note that the primary function of TA Queue is to serve as a mobile app for general-purpose uses. More specific uses, such as sharing a screen and handling files may be better suited over a desktop. For these reasons, the majority of the UI examples given are as they would appear on a Android mobile device, save for the screen sharing, as that appears as it may on a desktop/laptop device.

1.1 Course UI

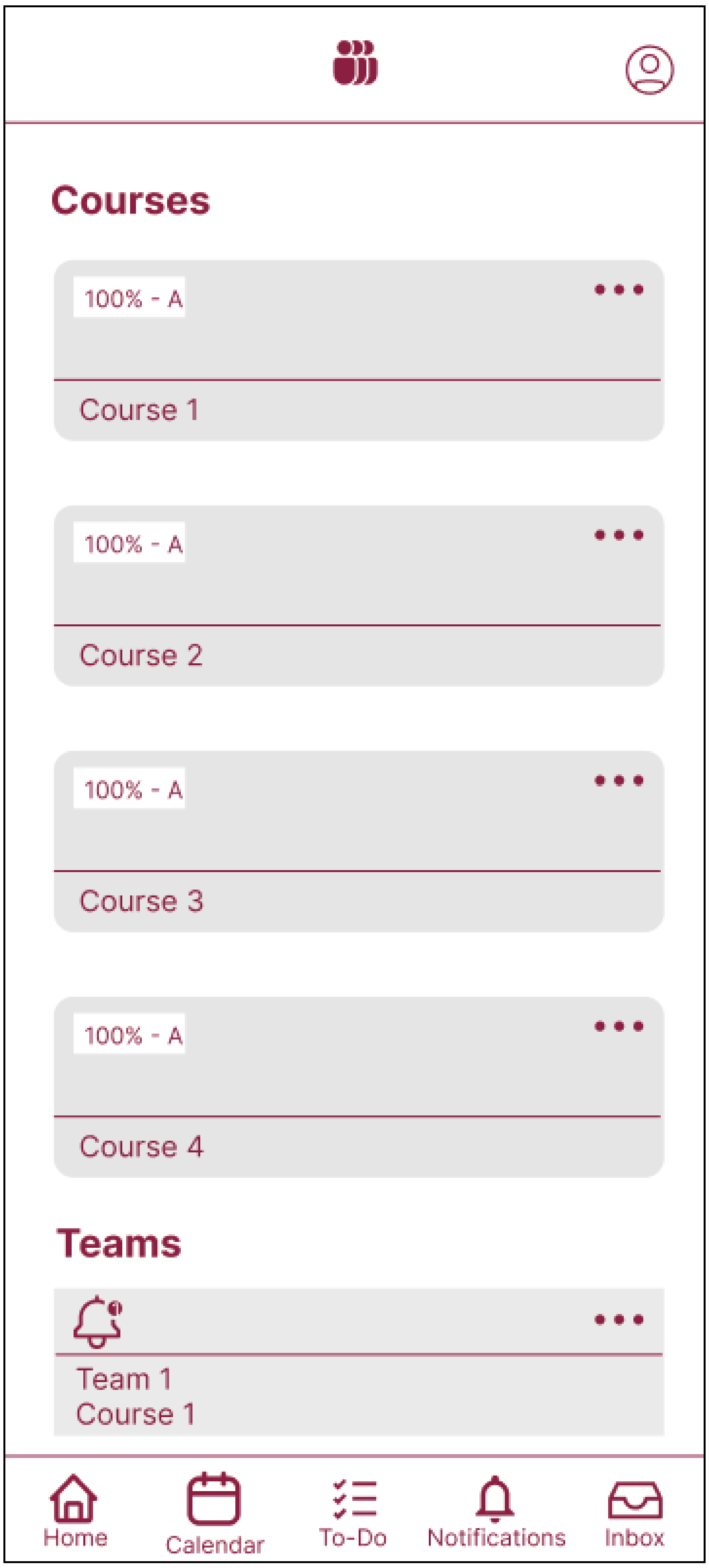


Figure 1.1: Homepage

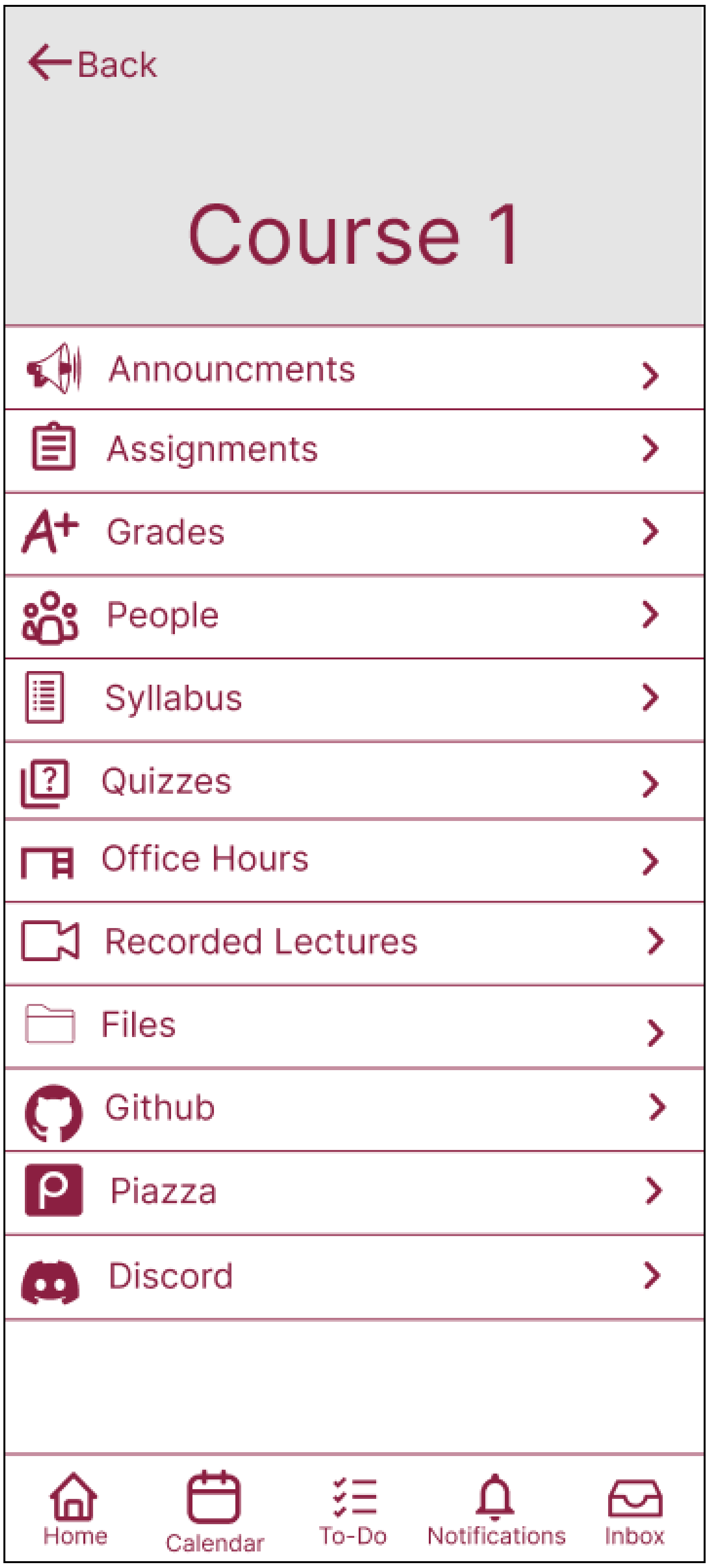
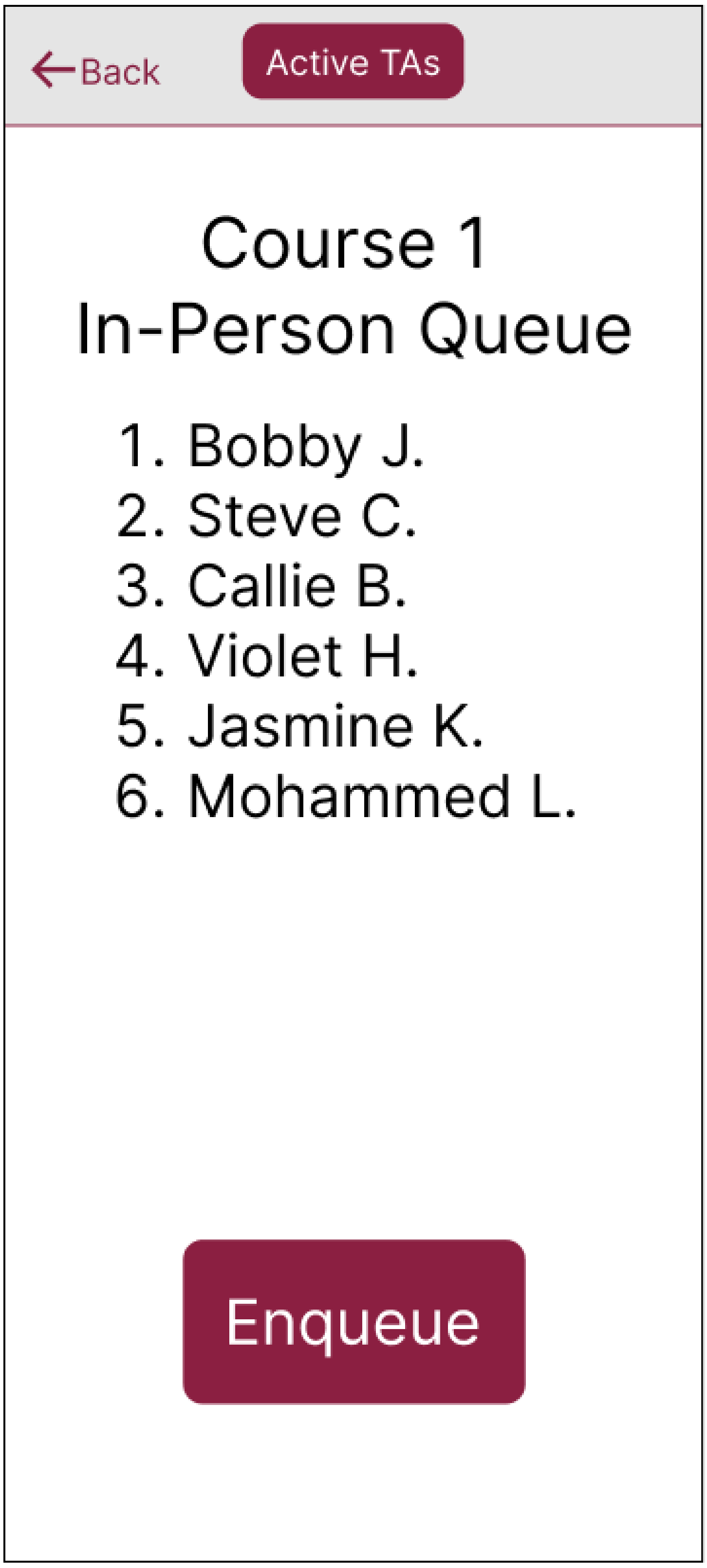


Figure 1.2: Course Overview

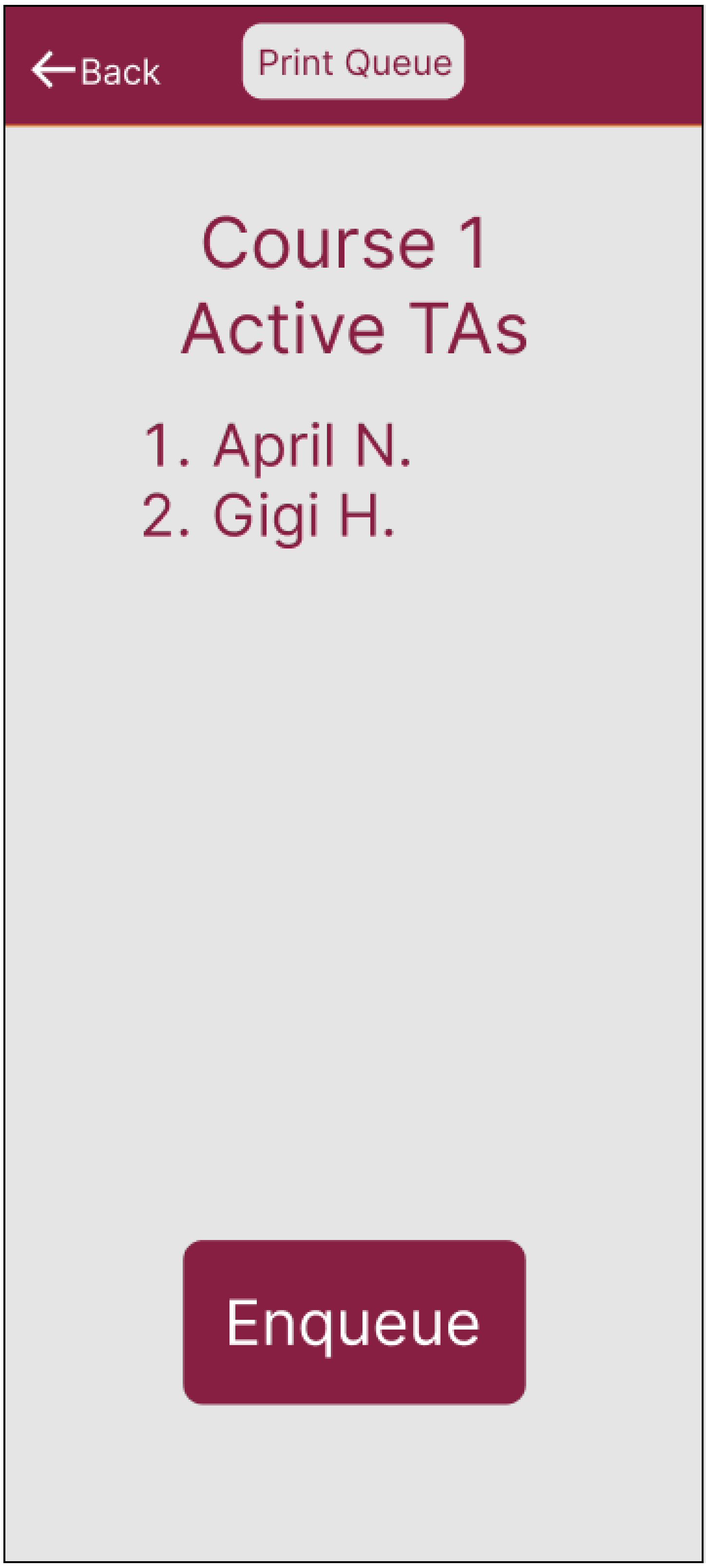
1.2 Priority Queue UI



Course 1
In-Person Queue

1. Bobby J.
2. Steve C.
3. Callie B.
4. Violet H.
5. Jasmine K.
6. Mohammed L.

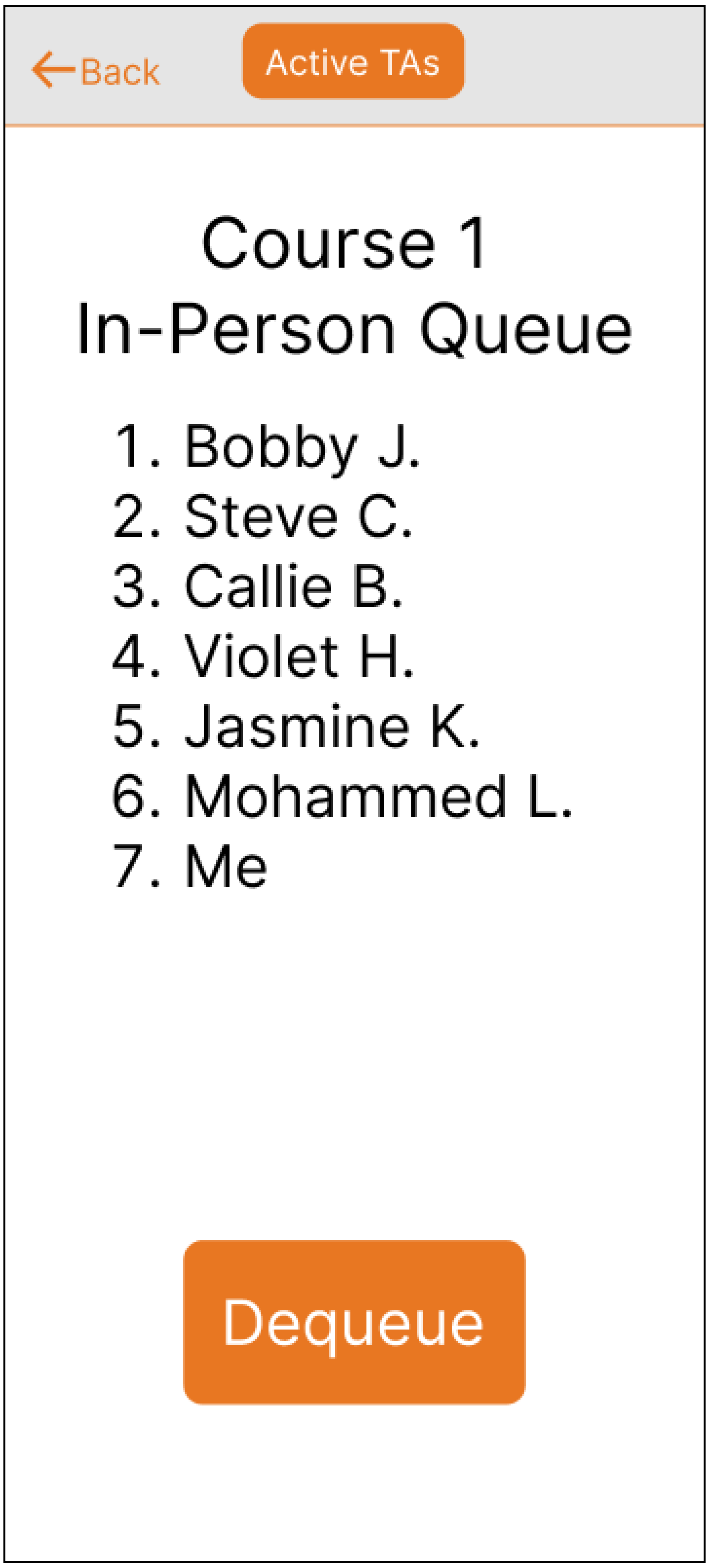
Figure 1.3: Student View (Dequeued) - Print Queue



Course 1
Active TAs

1. April N.
2. Gigi H.

Figure 1.4: Student View (Dequeued) - Active TAs



Course 1
In-Person Queue

1. Bobby J.
2. Steve C.
3. Callie B.
4. Violet H.
5. Jasmine K.
6. Mohammed L.
7. Me

Figure 1.5: Student View (Enqueued) - Print Queue

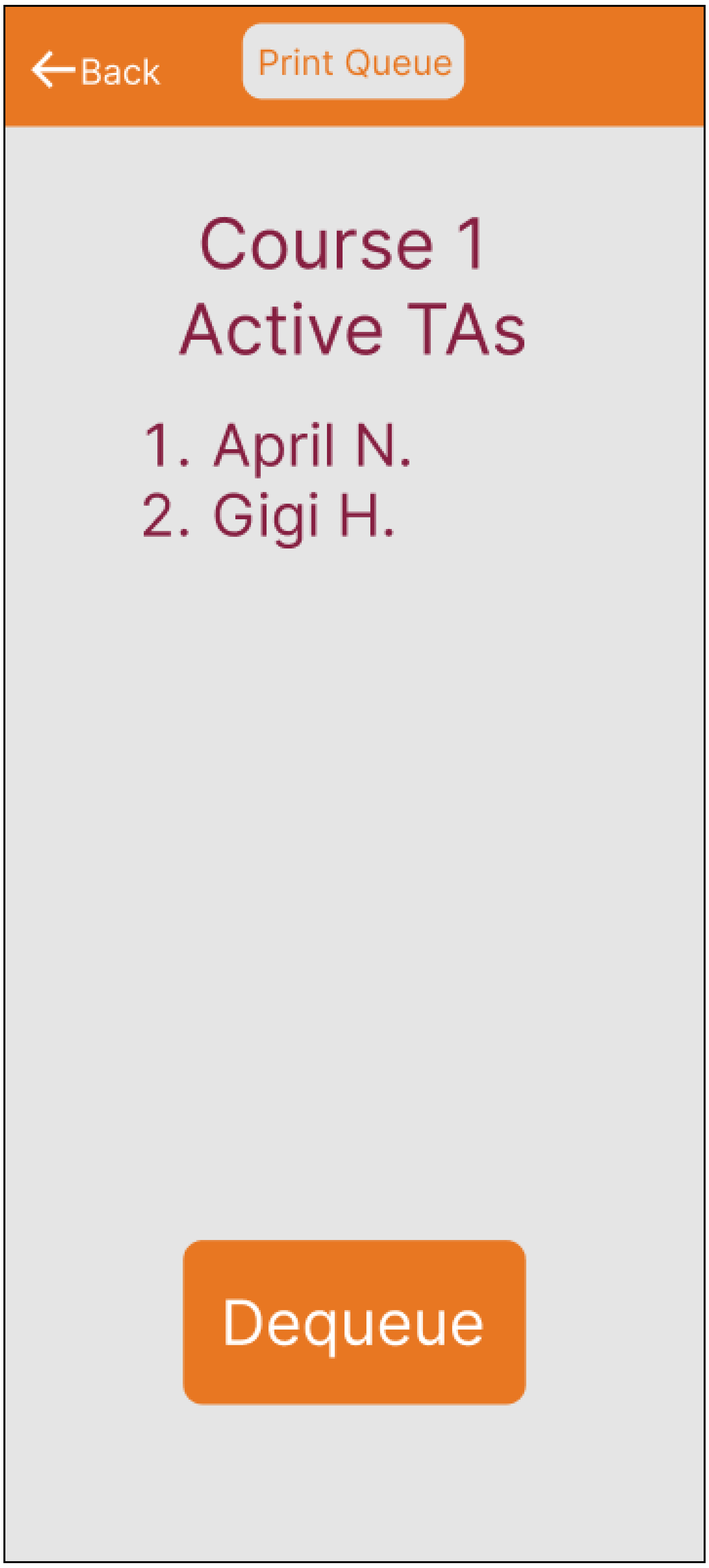
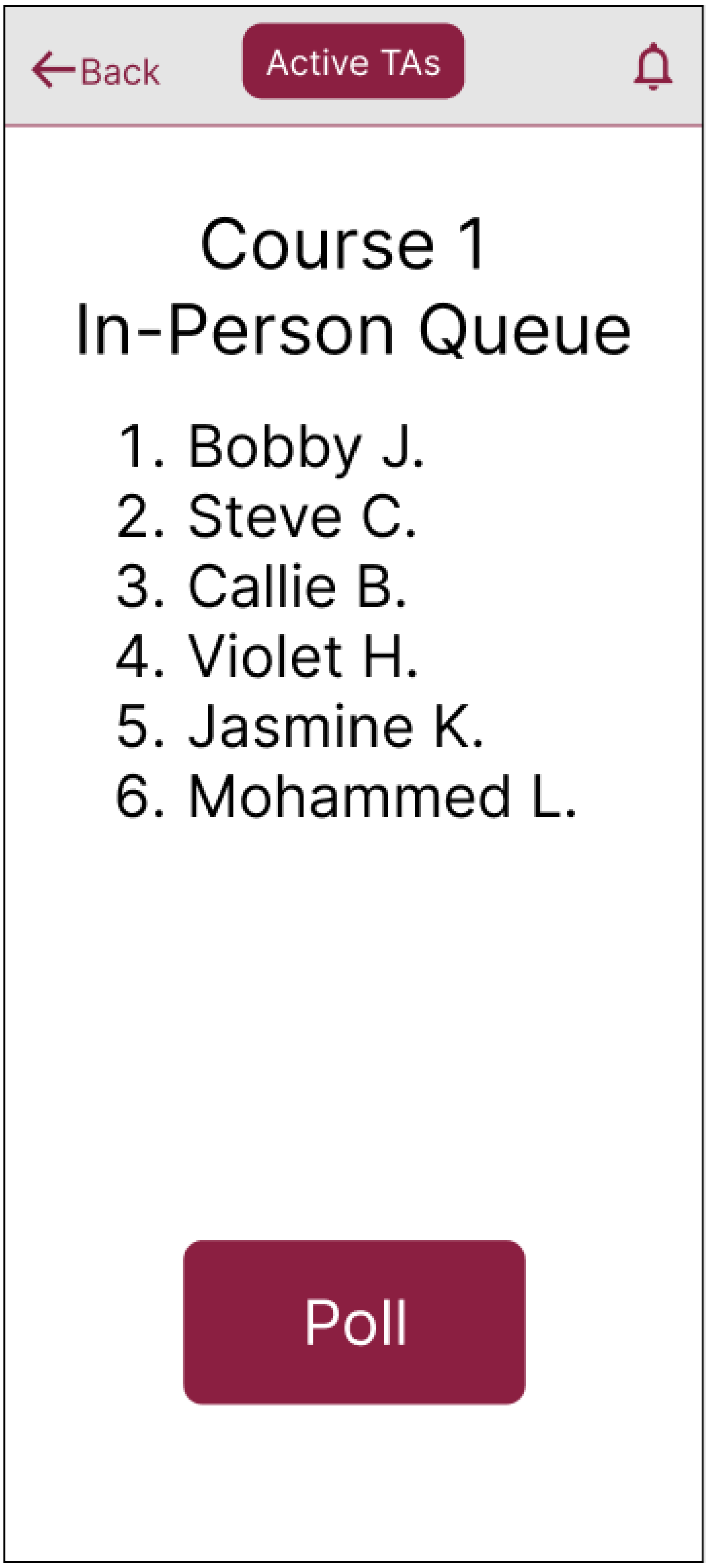


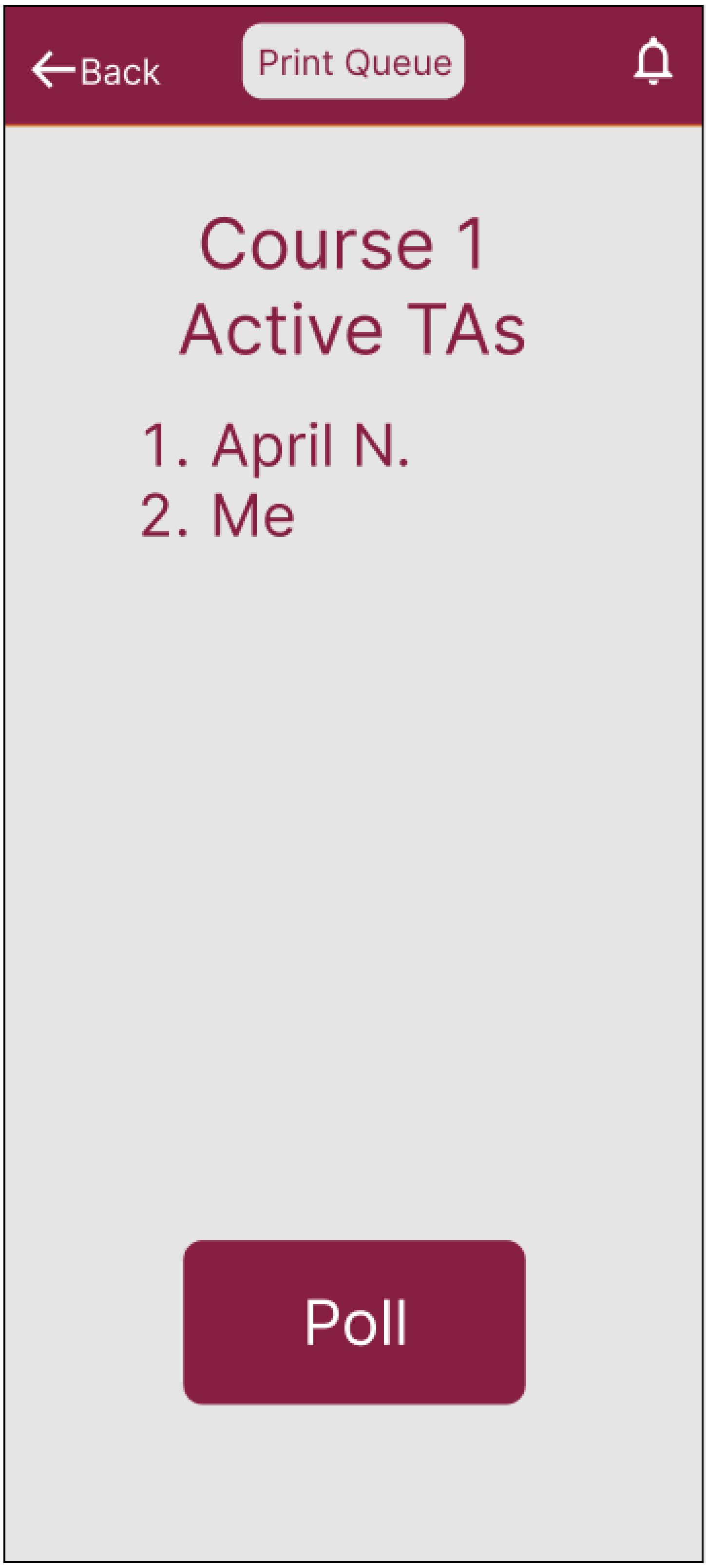
Figure 1.6: Student View (Enqueued) - Active TAs



Course 1
In-Person Queue

1. Bobby J.
2. Steve C.
3. Callie B.
4. Violet H.
5. Jasmine K.
6. Mohammed L.

Figure 1.7: TA View - Print Queue

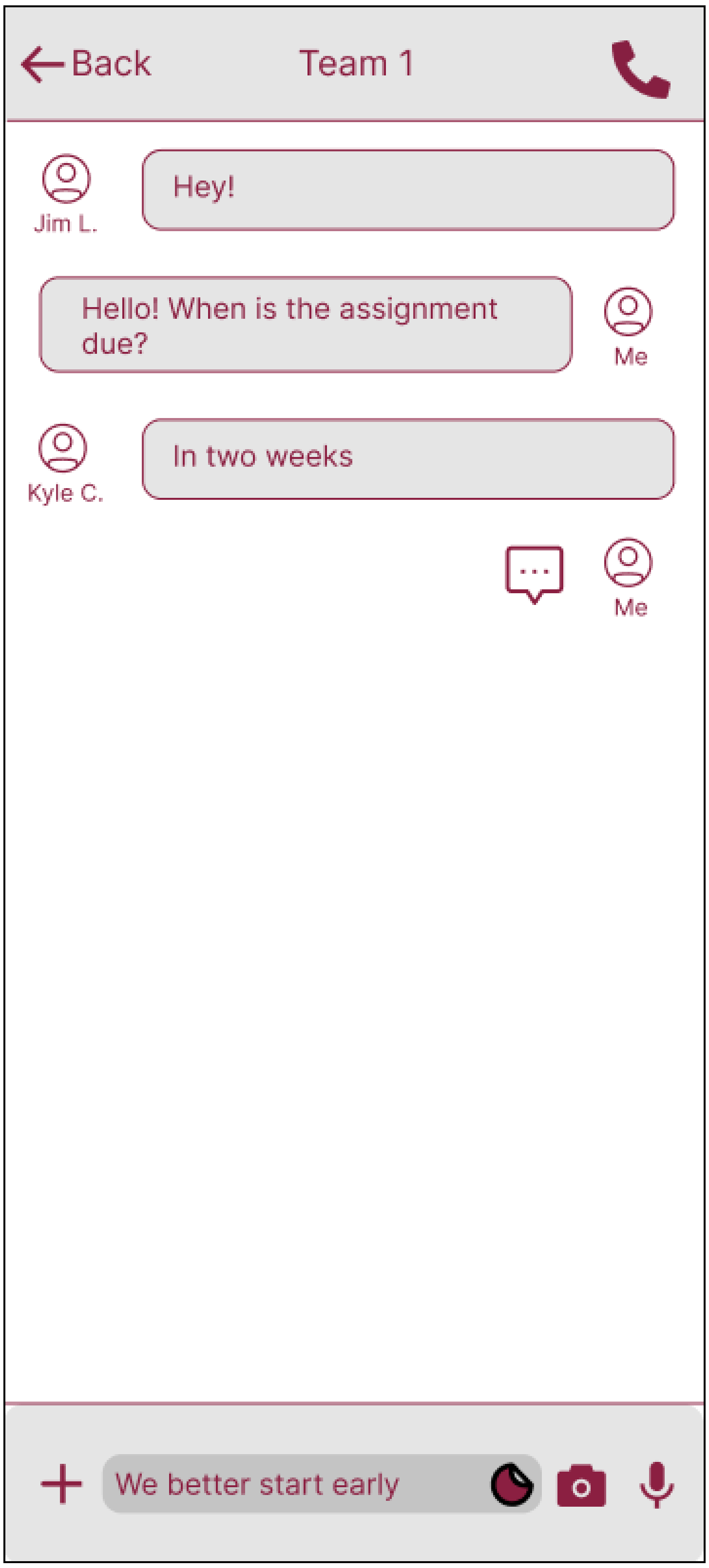


Course 1
Active TAs

1. April N.
2. Me

Figure 1.8: TA View (Enqueued) - Active TAs

1.3 Communication UI



Hey!

Hello! When is the assignment due?

In two weeks

We better start early

Figure 1.9: Group chat among team members

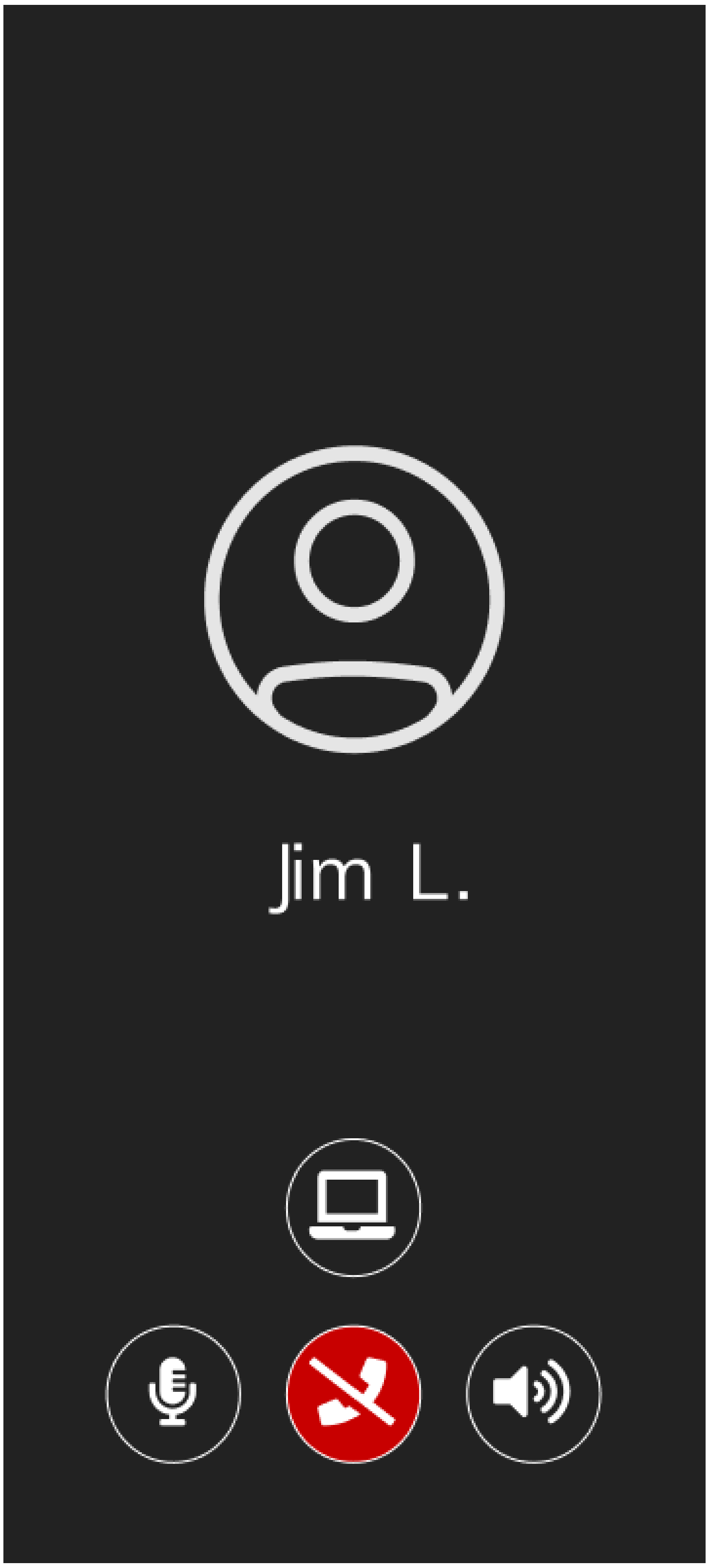


Figure 1.10: Neutral Call Screen

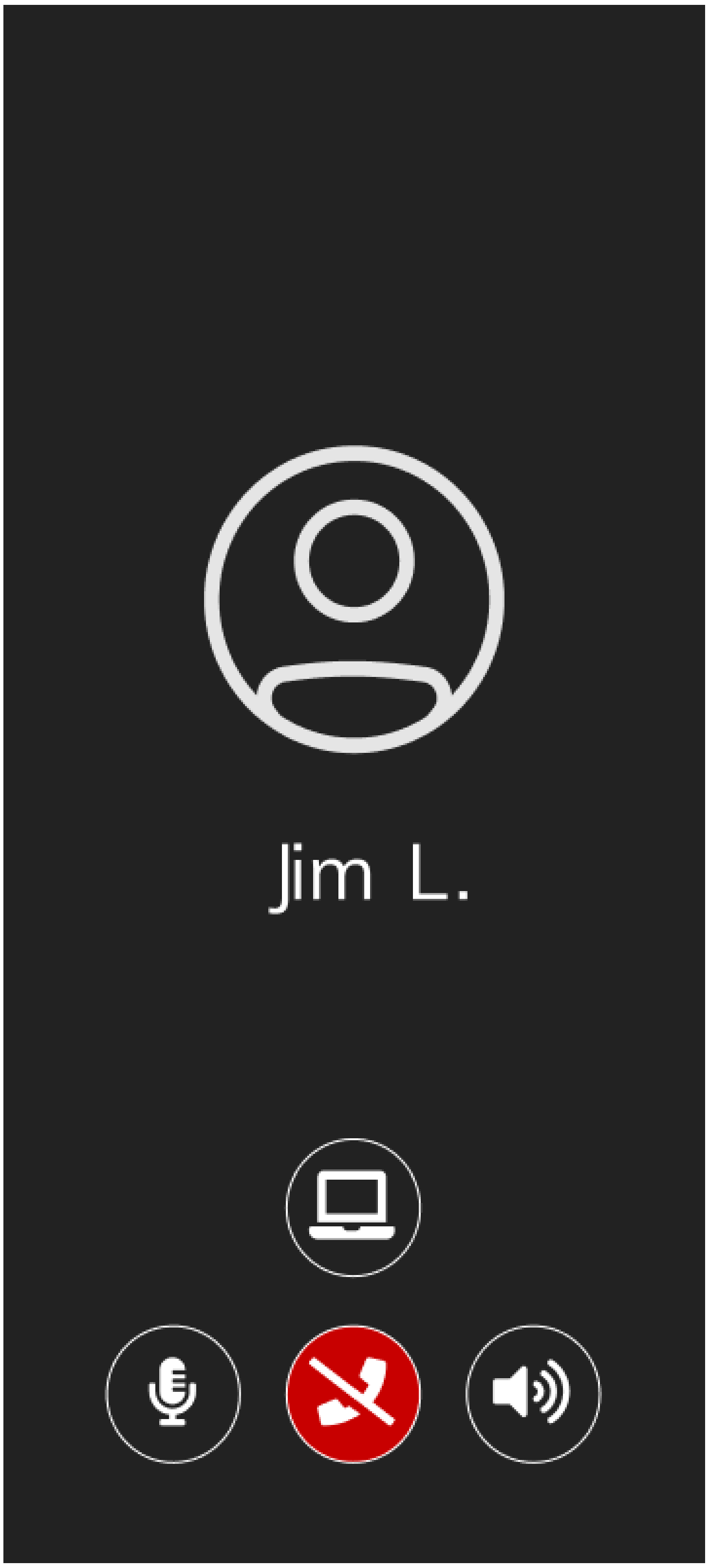
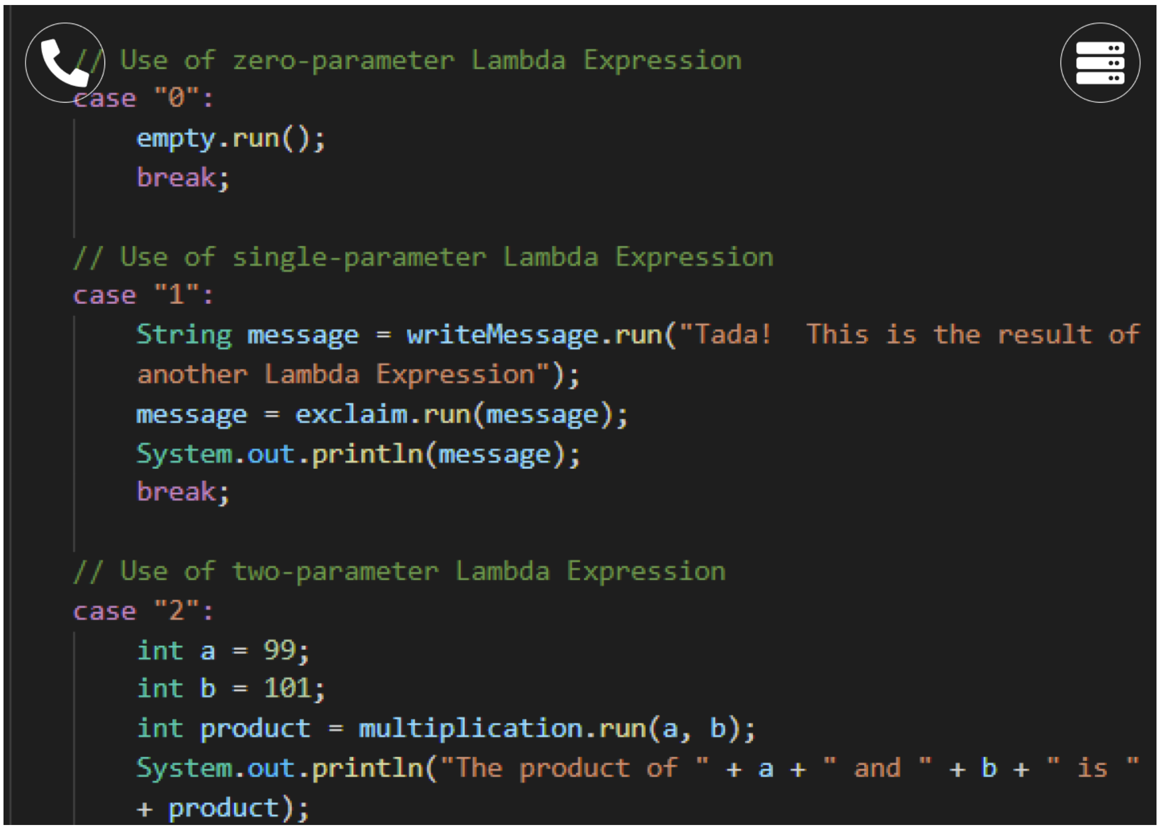


Figure 1.11: Neutral Call Screen (Muted) (Deafened)

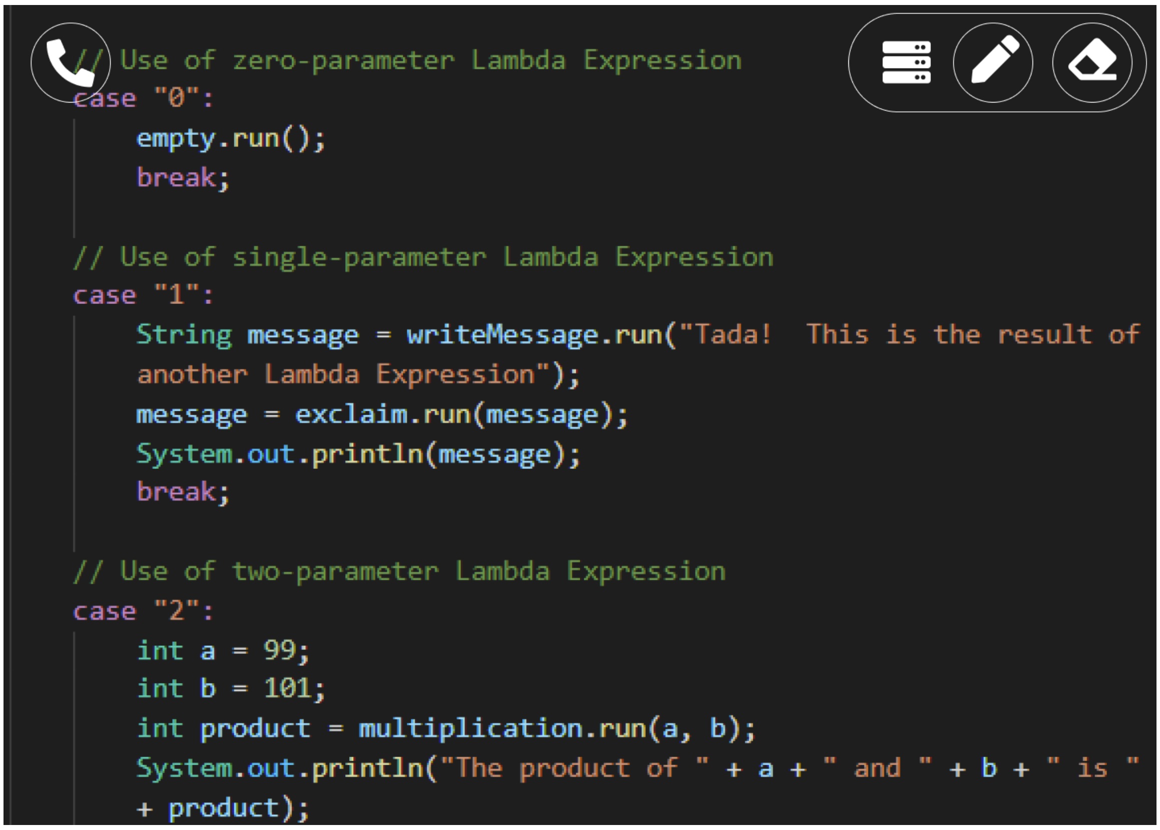


```
// Use of zero-parameter Lambda Expression
case "0":
    empty.run();
    break;

// Use of single-parameter Lambda Expression
case "1":
    String message = writeMessage.run("Tada!  This is the result of
    another Lambda Expression");
    message = exclaim.run(message);
    System.out.println(message);
    break;

// Use of two-parameter Lambda Expression
case "2":
    int a = 99;
    int b = 101;
    int product = multiplication.run(a, b);
    System.out.println("The product of " + a + " and " + b + " is "
    + product);
```

Figure 1.12: Shared Screen Over Call

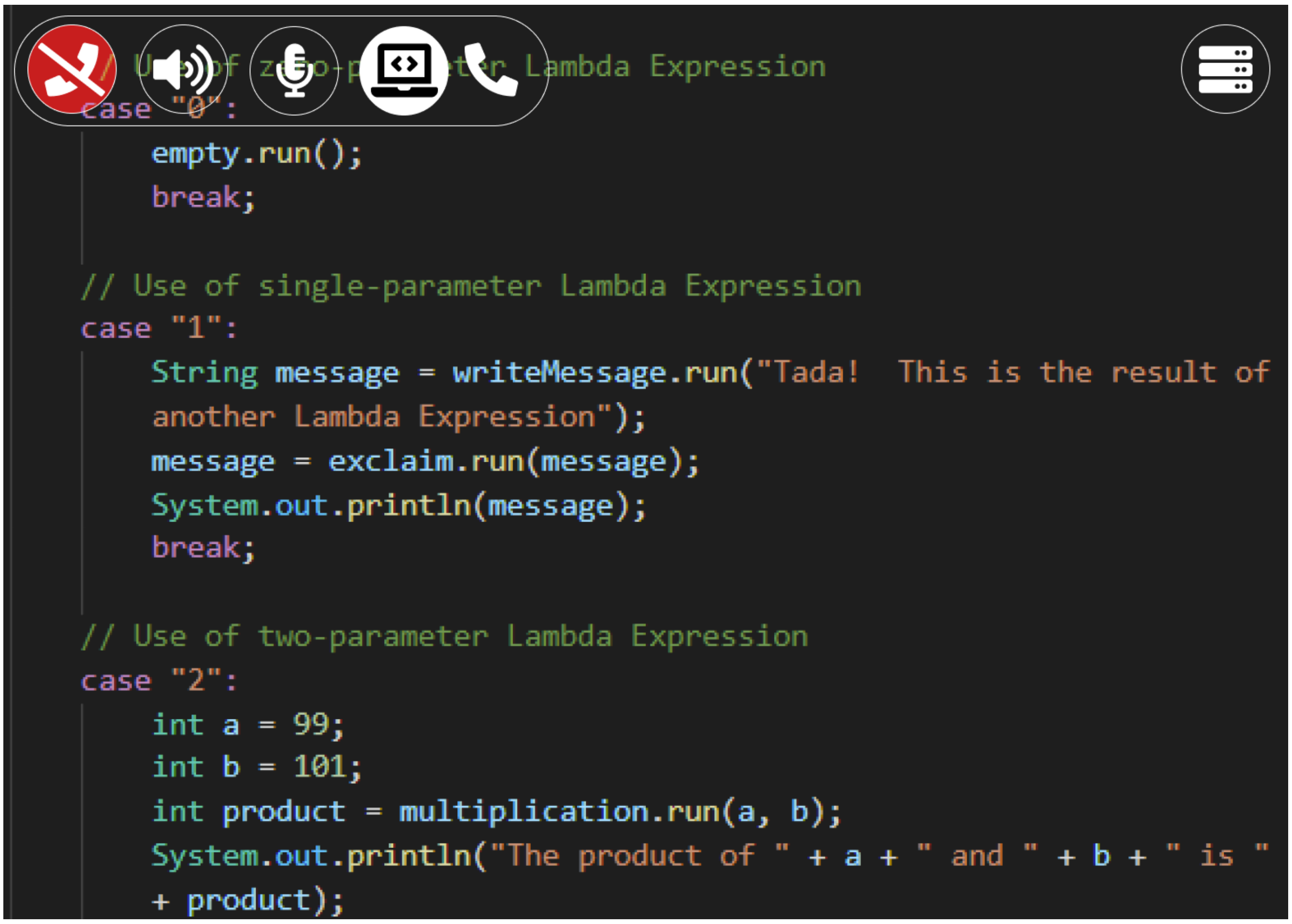


```
// Use of zero-parameter Lambda Expression
case "0":
    empty.run();
    break;

// Use of single-parameter Lambda Expression
case "1":
    String message = writeMessage.run("Tada!  This is the result of
    another Lambda Expression");
    message = exclaim.run(message);
    System.out.println(message);
    break;

// Use of two-parameter Lambda Expression
case "2":
    int a = 99;
    int b = 101;
    int product = multiplication.run(a, b);
    System.out.println("The product of " + a + " and " + b + " is "
    + product);
```

Figure 1.13: Shared Screen Over Call (Edit Screen Menu Open)



```
// Use of zero-parameter Lambda Expression
case "0":
    empty.run();
    break;

// Use of single-parameter Lambda Expression
case "1":
    String message = writeMessage.run("Tada!  This is the result of
    another Lambda Expression");
    message = exclaim.run(message);
    System.out.println(message);
    break;

// Use of two-parameter Lambda Expression
case "2":
    int a = 99;
    int b = 101;
    int product = multiplication.run(a, b);
    System.out.println("The product of " + a + " and " + b + " is "
    + product);
```

Figure 1.14: Shared Screen Over Call (Edit Screen Menu Open)

Chapter 2

Algorithm Design

2.1 addStudent(Student) : boolean

```
If Student is valid AND is enrolled AND is verified AND paid tuition AND pre-requisite has been met
AND Student.major matches required
    If class.size != FULL
        classRoster.add(Student)
    Else return class full error (False)
Else return cannot add student error (False)
Return success
```

2.2 addCourse(Course) : boolean

(precond: addStudent was successful)

```
If Student.classSchedule is NOT full and Course.Semester is current semester
    Student.classScheduleadd(Course)
HomePage add Course icon
Add Student to applicable TA Queue
```

2.3 gradeAssignment(Student, Course) : boolean

```
Int Score = 0
If student is valid AND enrolled AND Student is in Course
    checkAssignment()
    Compare Submission to Answer Key
    For each answer in submission matching Answerkey
        Increment Score with Answer Weight
Assignment Grade = Score
Add all assignment grade and divide by number of assignments and multiply by weight
Post Grade Average
```

2.4 EnQueue(Student) : boolean

```
If Student is in Course AND Project is Active AND TA Queue is open/accepting
    If Student is not hogging TAQueue
        Iterate through Queue,
        Place Student in Queue index so that Queue is sorted by least recently
        Seen by TA on that Day
        Display Student Name, and Place in the Queue
```

2.5 HogAlert(String) : boolean

```
Iterate through all Students present in Queue
    Add all Student's number of visits, then divide by number of Students to get average number of
    visits.
    Add all Student's visit duration, then divide by number of Students to get average visit duration.
    If a Student's particular number of visits or duration of visit is 2.5 times the average, notify the TA
    by printing to queue room
"¡Student¿ is Hogging TA, please use caution".
```

2.6 call(UserID, UserIDList) : Call

```
Get callingUser from User Database using userID
CreateCallGroup called CallGroup
Add callingUser to CallGroup
For calledUser in UserIDList
    If callingUser in calledUser.friends
        Add calledUser to CallGroup
Notify CallGroup
```

2.7 screenShare(UserID, CallGroup) : boolean

```
If UserID in CallGroup
    Retrieve screen information from userUD
    Add screen information to CallGroup
    Notify CallGroup
```

2.8 forumPost(UserID, Body, Files, CourseID) : boolean

```
Search Course Database for Course with CourseID
If UserID in Course.Students OR UserID in Course.TA OR UserID in Course.Professor
    Add to Posts Database with Body, UserID, Files, and Empty [] for comments
    Add PostID to Course.Posts list
```

2.9 clockIn(UserID, CourseID) : boolean

```
Search Course Database for CourseID
If UserID in CourseID.TA
    Search Student Database for Student by UserID
    Create new Log in Log Database with current system Time
    Add Log to Student.Logs
```

2.10 clockOut(UserID, CourseID) : boolean

```
Search Course Database for CourseID

If UserID in CourseID.TA

    Search Student Database for Student by UserID
    Get most recent Log from Student.Logs
    Add endTime with current system Time to Log
```