KIT305/KIT607

Assignment 2

Implementation and Evaluation of a Mobile Application
Raffle Drawing Application

Due Date

The assignment has three assessable components:

- The application **code** (worth 50%; pair-based work), due on **Monday 25th May 2020 23:55 (week 13).**
- A **report** (worth 40%; pair-based work), due on **Monday 25th May 2020 23:55 (week 13).**
- A **demonstration** (worth 10%; pair-based work), due in your **scheduled tutorial timeslot (week 13).**

Background

In assignment 1, your task was to design and prototype a mobile application that allowed community organisations organise and run raffles. Your task for this assignment 2 is to now implement and evaluate a *native* mobile Android or iOS application that provides the functionality. A revised and simplified list of requirements and additional requirements is provided in this document.

In the week 7 tutorial and in your own time, you and your partner will together evaluate your different prototypes from assignment 1, to determine which ones meet the usability goals discussed in this unit, and which should form the basis of your assignment 2 code.

The goal of this assignment is to assess your ability to implement and evaluate mobile applications, and to demonstrate your understanding of the fundamentals of mobile device programming when applied to a practical context.

Group Work

This is a pair-based assignment. Group members will need to agree on a target mobile OS (Android or iOS), and - after some usability think-aloud testing of the available prototypes – agree on the design that is to be implemented in code. **You MUST have a partner chosen by your week 7 tutorial.** If you have difficulties finding a partner, use the class Discord or WeChat, and speak to other students looking for partners. Those who have filled in the survey indicating they don't have a group will be assigned to a group before semester break, and you are expected to make contact with your group within 48 hours of being assigned to a group.

Changes to group members **MUST** be approved by the Unit Coordinator. Late changes may not be approved.

Group members should contribute equally to ALL components of the assignment (report, code, demonstration). Given the current situation with online learning, it is especially important that all team members promptly agree on communication and file sharing methods. WeChat, Discord, Zoom, and Skype are recommended, or you may use another instant messaging way of communicating. For sharing files GitHub is recommended but can be complex to use. You may also consider Dropbox. Teams using Mac-in-cloud may choose to share their account with their partner *only*. Sharing Mac-in-cloud accounts with other students is prohibited and may be considered as academic misconduct, and will be monitored.

This assignment will be subject to peer assessment, further details will be provided on MyLO.

Code (50%)

The *native* Android or iOS application that you are to implement must consist of at least:

- A way of creating a new raffle and saving it.
    - At a minimum, a raffle should have a name, and description.
- A way of selling tickets for a raffle.
    - At a minimum, tickets should have a price, number, purchase date/time, and be associated with a customer.
- A way of drawing a winning ticket/tickets at random.
- A way of listing all raffles.
    - Including a way of deleting raffles that *haven't been started yet*.
- A way of listing all tickets for a raffle.
- A way of editing raffle settings.
    - Change name or description (note that some settings in the additional requirements below should be restricted for editing after the raffle has started—i.e. at least one ticket has been sold).
- A way of editing customer details on a ticket.
    - You do not need to implement deleting a ticket.

Data entered by the user should *persist* between runs of the application. You can use any data persistence method *taught in this unit* to accomplish this (e.g. SQLite, file writing, or SharedPreferences/UserPreferences). You should *not* "pre-fill" the database with user-entered information (such as existing raffles/tickets/customers), as the application should function properly when there are no raffles or tickets in the system. Remember that any testing data you enter while developing your application will *not* be visible to your marker, as they will be installing a fresh version of your application on their emulator. **You are not allowed to use an online database such as MySQL or Firebase for this assignment.**

An important feature of your application will be its usability. For the purpose of this assignment, particular emphasis will be placed on the "efficiency" aspect of usability. This means that it should be quick and easy to sell tickets to customers. TIP: consider how to handle *repeat customers*, and *customers purchasing multiple tickets at once.*

Aside from checking for functionality, your program will be assessed on its robustness (i.e. does the application crash), usability, and aesthetics. You will not be assessed on your coding style, commenting, or indentation, however given the scale of this application, you are advised to do this to assist in development.

Additional requirements

To gain higher marks, additional functionality should be added based upon the list below

- The organiser can associate an **image** (selected from the photo library or camera) with a raffle as a "cover".
- The organiser can use the built-in OS **sharing** functionality to share an individual ticket's information to another application in plain-text.
    - An example of the shared text might be "Lindsay Wells, Ticket 001, $5.00, Purchased 31/03/2020".
- Raffles may have a **maximum** number of tickets available, and your app should prevent extra tickets being sold.
- Organisers can choose to create a "**Margin Raffle**". The details of this have been explained in previous lectures, but you may like to use this website for an example:
    - https://www.malatwell.com.au/football-margin-tickets.html
    - The app must be able to handle margin raffles AND "normal" raffles.
    - Some constraints on margin raffles for Assignment 2:
        - In margin raffles, tickets should be given out in a random order ("normal" raffles, usually have tickets given out in order, e.g. 001, 002, 003). This is to prevent an advantage for buying tickets early (as margins tend to be lower numbers).
        - You will need a maximum number of tickets implemented for this, in order to select at random.
        - No two customers may have the same ticket.
        - If no customer has a ticket matching the margin at the end of the game, *nobody wins*.

If your assignment 1 prototype doesn't include functionality for these requirements, you can still implement this by adding this to your application.

Use of Tutorial Code and Outside Code

You are more than welcome to use the Week 5 tutorial base code or completed Week 5 tutorial work as a base of your assignment—in fact this is encouraged. You do not need to reference this in your assignment.

You are expressly **prohibited** from using any other code online as a template for this assignment. Small snippets of code (such as stackoverflow answers) may be used with code comments with at a minimum the URL of where the code came from.

Third-party libraries (for example, for things like date-pickers, data persistence, user interface elements, camera/sharing, etc.) **must not be used**.

You must use either Xcode (using Swift) or Android Studio (using Java) for your assignment. You must not use cross-platform tools such as Flutter, or languages beyond the scope of this unit (such as Objective-C or Kotlin). You are welcome to attempt this assignment in your own time using Flutter or Cordova as an exercise in comparing the development approaches.

Changes from Assignment 1 (i.e. things you don't have to do)

Note that your code assignment is *not* required to match your prototype design from assignment 1, or your updated prototypes as a result of usability testing in assignment 2. While this doesn't emulate the real world, this is being done for practical reasons, as some features from your initial prototypes may be too difficult to implement.

Functionality from assignment 1 which is not specified in the "Code" section of this document is *not* required to be complete. Examples of things that people included in their assignment 1 that are **not** required to be functional in your code assignment (this list is not exhaustive):

- A login/account screen;
- A customer-facing interface (you ONLY need to consider the organiser-side for this assignment);
- QR Code readers;
- Credit-card payment systems (imagine ALL payment is a cash transfer in the real world);

Use your judgement on how to modify your designs if you included these features in your assignment one. You may like to simplify your assignment prototypes or create new simple prototypes or sketches to use as a guide for how your final application will look.

Report (40%)

You will have been reading many articles throughout the semester that report on the design, iterative refinement, and evaluation of new mobile and ubiquitous computing applications. Your readings have also been complemented by articles on interaction design and discount usability methods. Accompanying the implementation of your application will be a report outlining how you were able to apply the process of usability testing (and specifically think-aloud evaluation) to iteratively improve the design and usability of your application.

You must write a report that is up to six (6) pages long (single-spaced, 10pt Times font). *This page count includes figures and tables, but excludes references and appendix.*

Your report should include the following sections:

- Introduction;
- Description of Application (including selected screenshots of lo-fi prototype and final application);
    - Max 2 pages
- (Usability) Methodology;
- Results;
- Discussion/Conclusions;
- ***References***; and
- Appendices.

The report will focus on your **usability testing**, not the process of how you programmed your application in Xcode/Android Studio.

The Methodology section will outline the usability testing that was employed. It will also include details on how the testing was conducted, including a description of the user participants and how they were recruited, and how the experiments were conducted. A strong assignment will *justify* the choices made, highlighting alternate approaches that could have been taken. The intent of this section is to document how you conducted your experiments such that they can be replicated by other people, and critiqued for any potential sources of bias or confounds in your data. The week 6 lecture recording is a good source of information that you need to document.

It is expected that for a HD-level assignment, that you conduct *iterative* testing with at least three rounds of testing with 3-5 participants in each test (these participants may be the same in each test or different, and this will be documented in your Methodology section).

The Results section will report the results of the usability testing. You should present a table of results, however only simple summary statistics are necessary; you are NOT expected to conduct accurate statistical analysis such as t-Tests, ANOVAs, or correlation). The purpose of this section is to show that you understand the testing process and have conducted testing, *not* to get perfect usability scores. You will not be assessed on whether or not your results were positive or not. **All results must be anonymised.**

The Conclusions/Discussion section will provide a self-assessment critique of the prototypes in terms of usability, and will include a summary of the strengths and weaknesses of the application. You should refer to your results section to back up your statements with data.

The report will also reference the set readings, lecture slides, and other literature and resources as appropriate to justify decisions and provide background information.

You should use the Appendices section to document your test-tasks, requirements, and task-requirement matrix (likely adjusted from Assignment 1). You may choose to include additional screenshots or tables of data/notes in here, however this section is for your marker's information only, it will not be assessed.

Coronavirus and You(r Usability Testing)

Conducting usability testing and finding participants is difficult even under normal circumstances. This semester the usability testing will need to facilitated by online video conferences with participants. Current law prevents gatherings of more than 2 people outside of a primary residence, which means conducting usability tests in person outside of your home is prohibited for this assignment. You will need to make use of video conferencing software, and consider:

- Participants access to video conference;
- How to share your prototypes (screenshare, fileshare, etc);
- Video recording of usability tests using video conference software is allowed, however **you must explicitly ask for the consent of your participants before beginning any recording**.

Your approach to the usability testing in the current climate will need to be documented in your Methodology.

For further information and tips, please see the week 6 lecture *recording* where Lindsay discusses this.

Demonstration (10%)

You are required to demonstrate your application during the week 13 tutorial timeslot. The demonstrations will be limited to four (4) minutes total. **All** group members must participate in the demonstrations, and the 4-minute timeslots will be strictly enforced to ensure that all presentations are completed within the tutorial session. The group demonstrations will need to:

- State the goals of the work for a broad public;
- Outline the design and testing that was conducted;
- Provide a convincing example of how a person would use the prototype (using either the emulator, or a series of screenshots); and
- Provide convincing evidence that the application meets its goals.

You may choose to create a PowerPoint presentation, but this is not necessary. You choose how to best communicate the information above. **At least one team member needs to screenshare and ALL team members must use their microphone.** As with assignment 1, we will be on a strict schedule, with limited time allowed to handle technical difficulties. **Key to this demonstration is to practice, and only include the most relevant and *interesting* information.**

Assignment Submission

The following files must be submitted via MyLO before 23:55 on Monday 25th May (Week 13):

- One zip file, containing the project files. The zip filename should start with your UTAS account name (either members' name is fine).
    - For Android, you should create this ZIP file using the File -> Export to Zip option. Submit the ZIP file which is created.
    - For iOS, you need to find your .xcodeproj file and project folder in Finder, select them both, right-click and choose "Compress". Submit the ZIP file which is created.
- A group assignment coversheet (available from the ICT website): http://www.utas.edu.au/__data/assets/pdf_file/0005/161375/GroupAssignmentCover.pdf
- The group report, in PDF format, with a filename that starts with your UTAS account name (either members' name is fine).
- A group assignment coversheet (available from the ICT website): http://www.utas.edu.au/__data/assets/pdf_file/0005/161375/GroupAssignmentCover.pdf

Only one group member needs to submit these deliverables. Feedback will be sent to that team member and that team member will be expected to send that feedback to their partner.

Peer assessment will be conducted using an online form are to be announced on MyLO. These will be due in Week 13. These forms will ask you to indicate each team members % contribution to the main parts of this assignment (code, report, demonstration). Final marks will take the combined answers of team members into account plus any additional comments from students.

Plagiarism and Cheating:

Practical assignments are used by the Discipline of ICT for students to both reinforce and demonstrate their understanding of material which has been presented in class. They have a role both for assessment and for learning. It is a requirement that work you hand in for assessment is your own.

Working with others

One effective way to grasp principles and concepts is to discuss the issues with your peers and/or friends. You are encouraged to do this. We also encourage you to discuss aspects of practical assignments with others. However, once you have clarified the principles of the problem, you must develop a solution entirely by yourself in your pair. In other words; you and your partner must develop the application *yourselves*. You can discuss problems, but not share solutions. Assistance with solutions should be provided by staff.

Cheating

- Cheating occurs if you claim work as your own when it is substantially the work of someone else.
    - This includes the use of third-party code from online resources.
- Cheating is an offence under the Ordinance of Student Discipline within the University. Furthermore, the ICT profession has ethical standards in which cheating has no place.
- Cheating involves two or more parties.
    - If you allow written work, computer listings, or electronic versions of your code to be viewed, borrowed or copied by another student you are an equal partner in the act of cheating.
    - You should be careful to ensure that your work is not left in a situation where it may be used/stolen by others.

Where there is a reasonable cause to believe that a case of cheating has occurred, this will be brought to the attention of the unit lecturer. If the lecturer considers that there is evidence of cheating, then no marks will be given to any of the students involved and the case will be referred to the Head of Discipline for consideration of further action.

KIT305/KIT607 Assignment 2, Semester 1, 2020: Implementation and Evaluation of a Mobile Application					
Criterion	**High Distinction (HD)**	**Distinction (DN)**	**Credit (CR)**	**Pass (PP)**	**Fail (NN)**
Code (50%; ILO2) Students are to produce a raffle drawing application.					
Functionality – Raffle and Tickets Lists (20%): The application should consist of at least: - A way of listing all raffles. - A way of listing all tickets for a raffle.	A custom layout/view for each list item is used, showing appropriate detail for a summary screen. The user can delete raffles that haven't been started yet. The user can see more detail about a ticket by tapping on it and sensible choices were made for this. The interface for this part is highly intuitive; provides a consistent look and feel across all screens and is aesthetically pleasing. Application *never* crashes and has been tested thoroughly to avoid the existence of bugs. The raffles and tickets can be *filtered* or *searched* in some way.	A custom layout/view for each list item is used, showing *appropriate detail* for a summary screen. The user can delete raffles *that haven't been started yet*. The user can see more detail about a ticket by tapping on and *sensible* choices were made for this. The interface for this part is intuitive and *aesthetically pleasing*. Application *never* crashes. There are only very minor bugs (if any).	A custom layout/view for each list item is used, showing *details* of each item. The user can delete raffles. The user can see more detail about a ticket by tapping on it. The interface for this part is somewhat *intuitive*. There are only some bugs (if any) in the application and/or *very* rarely crashes.	The *title* of each raffle is listed, and the *ticket number* of each ticket is listed. The user can *delete* raffles. A basic list view is provided. The user can see more *detail* about a ticket by tapping on it. There are only *some* bugs (if any) in the application and/or rarely crashes.	The application does not compile OR crashes on start-up OR crashes frequently. The user is *unable* to view all raffles or tickets. No *delete* functionality is provided.
Functionality – Add/Edit Raffles and Tickets (25%): The application should consist of at least: - A way of creating a new raffle and saving it. - A way of selling tickets for a raffle.	Users can add raffles with a title and description, plus any *sensibly-chosen* additional meta-data. Users can add tickets with a ticket number, price, and *appropriate* customer data. Users can edit raffles and tickets, with appropriate fields restricted from being changed after a raffle has started. Users can sell multiple tickets and a correct total price is shown. The application implements a maximum number of allowed tickets per raffle, and *correctly* prevents excess tickets being sold. The interface for this part is intuitive and aesthetically pleasing.	Users can add raffles with a title and description, plus any *additional meta-data*. Users can add tickets with a ticket number, price, and customer data. Users can edit raffles and tickets, with appropriate fields *restricted* from being changed *after a raffle has started*. Users can sell multiple tickets and a correct total price is shown. The application includes a field for *maximum number* of allowed tickets per raffle, with some problems or not fully implemented. The interface for this part is *intuitive* and *aesthetically pleasing*.	Users can add raffles with a title and description. Users can add tickets with a ticket number, price, and customer data. Users can *edit* raffles and tickets after they are created. Users can sell *multiple tickets* and a correct *total price* is shown. The interface for this part is somewhat *intuitive*. There are only a *few* bugs (if any) in this part of the app.	Users can *add* raffles with a *title* and *description*. Users can add tickets with a *ticket number*, *price*, and *customer data*. There are only a *few* bugs (if any) in this part of the app.	The application does not compile OR crashes on start-up OR crashes frequently. The application does not contain functionality to add raffles or tickets. Crashes prevent the user from adding raffles or tickets.
Functionality – Draw Winners (15%): The application should consist of at least: - A way of drawing a winning ticket/tickets at random.	Users can draw winners at random from the tickets sold for a raffle. Multiple prizes can be drawn per raffle. Users can check the winners of past raffles. *Sensible choices* are made for this display. The interface for this part is intuitive and *aesthetically pleasing*.	Users can draw winners at random from the tickets sold for a raffle. *Multiple* prizes can be drawn per raffle. Users can check the winners of *past* raffles. The interface for this part is *intuitive*.	Users can draw a winner at random from the tickets sold for a raffle. Users can check the winners of *past* raffles. There are only a *few* bugs (if any) in this part of the app.	Users can draw a winner at random from the tickets sold for a raffle. There are only a *few* bugs (if any) in this screen.	The application provides *no functionality* for drawing a winner, or crashes prevent this part of the assignment being assessed.

Functionality – Data Persistence (15%): Data entered by the user should persist between runs of the application. You can use any data persistence method taught in this unit to accomplish this (e.g. SQLite, file writing, or SharedPreferences/UserPreferences).	*All* data entered by the user persists between runs of the application. There are *no noticeable* bugs with the persistence. The application functions when there is no data in the database.	*All* data entered by the user persists between runs of the application. There are only *some* bugs (if any) with the persistence. The application functions when there is no data in the database.		*Most* data entered by the user persists between runs of the application. There are only *some* bugs (if any) where some data does not save. The application functions when there is *no data* in the database.	Data does *not* persist when the application is closed and re-opened.
Usability - Efficiency (5%): The application should make the process of selling tickets efficiently.	The interface provides *well-thought out* functionality for selling tickets efficiently and handling repeat customers.	Basic functionality for selling tickets efficiently and handling repeat customers is provided There are *no noticeable bugs* with this implementation.		Basic functionality for selling tickets efficiently and handling repeat customers is provided. There are only *some* bugs with this implementation.	The application does *not* have any noticeable design decisions that make selling tickets an efficient process.
Additional Functionality – Camera (7.5%): The organiser can associate an image (selected from the photo library or camera) with a raffle as a "cover".	The application allows you to choose a photo from the phone's image gallery or camera and add it to a raffle. The image is displayed on the screen after the user selected the image ***and*** throughout the rest of the app in a sensible way. The image data persists between runs of the app (either file URL or image data is fine). There are *no* bugs with the camera functionality.	The application allows you to choose a photo from the phone's image gallery or camera and add it to a raffle. The image is displayed on the screen after the user selected the image ***and*** throughout the rest of the app in a *sensible* way. The image data *persists* between runs off the app (either file URL or image data is fine).		The application allows you to choose a photo from the phone's image gallery or camera and add it to a raffle. The image is displayed on the screen after the user selected the image, but the image does not persist between runs of the app.	The application features no camera functionality or the camera functionality crashes the app when triggered.
Additional Functionality – Sharing (5%): The organiser can use the built-in OS sharing functionality to share an individual ticket's information to another application in plain-text.	The application allows you to use the OS' built-in sharing functionality to share a text version of a ticket with *all* details. **Bonus (very? difficult)**: The user can share a generated image version of the ticket.	The application allows you to use the OS' built-in sharing functionality to share a text version of a ticket with *most* details.		The application allows you to use the OS' built-in sharing functionality to share one field of a ticket.	The application features no sharing functionality or the sharing functionality crashes the app when triggered.
Additional Functionality – Margin Raffle (7.5%): Organisers can choose to create a "Margin Raffle".	Margin raffles are fully implemented meeting all requirements (see specification). Margin raffles are *appropriately integrated* into the rest of the application. The application correctly handles all cases, including the no-winner case. There are *no* bugs with the margin raffle functionality.	Margin raffles are *fully* implemented meeting *all* requirements (see specification). The application *correctly* handles all cases, including the no-winner case. There are *no* bugs or *minor bugs* with the margin raffle functionality.	A *simple* implementation of margin raffles is provided, meeting most of the requirements (see specification). There are *some* bugs with the margin raffle functionality.	A basic implementation of margin raffles is provided, with some requirements missing (see specification). OR The application only allows margin raffles and not "normal" raffles.	The application does not include functionality for margin raffles or the application crashes when attempting to create or draw a margin raffle.

Report (40%; ILOs 1,3,4) Student is to create a report up to 6 pages long.					
Description of Application (25%; ILO3) The report will describe the functionality that has been implemented; including images of the low-fi prototype and final application.	All core functionality is explained in a *concise*, and well-written and *concise* manner. Screenshots are clear and show the core functionality of the app. A *reasoned* description of how the final application and lo-fi prototype differ is provided.	All core functionality is explained in a well-written and *concise* manner. Screenshots are *clear* and show the core functionality of the app. A description of how the final application and lo-fi prototype differ is provided.	*All* core functionality is explained. Screenshots show the core functionality of the app.	Report contains an overview of the functionality and some screenshots.	Fails to provide a description section.
Methodology and Usability Testing (25%; ILO4) The report will outline the usability testing that was conducted and the procedure in which it was conducted.	The employed usability testing is well-described and well founded; and described in *detail* such that the work could be replicated. Students identify potential biases introduced by their choice of methodology. Choice of methodology is *well-justified*, identifying *alternate* approaches that could have been used. Evidence that at least three rounds of testing were conducted is provided. Methodology follows the recommendations for this unit and applies them in a way that shows a *strong* understanding of usability testing.	The employed usability testing is well-described; and described such that the work could be replicated. Students identify potential *biases* introduced by their choice of methodology. Choice of methodology is *well-justified*. Evidence that at least three rounds of testing were conducted is provided. Methodology follows the recommendations for this unit in a *sensible* way.	The employed usability testing is *well-described*. Evidence that at least two rounds of testing were conducted is provided. Methodology follows the recommendations for this unit.	At least one prototype is evaluated, and most required details of the testing procedure are described. Methodology follows the *recommendations* for this unit.	Fails to provide a methodology section.
Results (25%; ILO4) The report will report the results of the usability tests.	Results are complete (and in a concise table) and include completion times *with summary statistic*s. Results match the methodology described in the report.	Results are *complete* (and in a concise table), and *match* the methodology described in the report.		*Basic* results are provided.	Fails to provide a results section, or some results are *missing* or *inconsistent* with the methodology section.
Discussion, Conclusion, and Overall Report Style (25%; ILO1) The report will provide a self-critique of the application's performance, including strengths and weaknesses of the application. The report will link to the set readings and other literature and resources as appropriate.	A thorough and *insightful* discussion of the results is provided, thoroughly explaining how the results impacted the design. A *sensible* description of way *future versions* of the app could be further improved is provided. Report covers all of the required sections. It is *very* well structured, has the required length, and has a logical flow between sections. English conventions of spelling, grammar, and punctuation are excellent. Academic referencing is consistent, with *many* references beyond the set readings. Report is within the page limit.	A detailed discussion of the results is provided, *thoroughly explaining* how the results impacted the design. Report covers all of the required sections. It is well structured, has the required length, and has a logical flow between sections English conventions of spelling, grammar, and punctuation are *excellent*. Academic referencing is consistent, with *some* references *beyond* the set readings. Report is within the page limit.	A *detailed* discussion of the results is provided, *explaining* how the results impacted the design. Report covers all of the required sections. It is *well structured*, has the required length, and has a logical flow between sections English spelling, grammar, and punctuation are *good*. Academic referencing is *consistent*, linking to the set readings. Report is within the *page limit*.	Some discussion of the results is provided. Report covers *all* of the required sections. Academic referencing links to the set readings.	Fails to provide a report, or report is missing some required sections.
Demonstration (10%; ILO5): Within the allocated 4-minute group timeslot, students are to: - State the goals of the work for a broad public. - Outline the design and testing that was conducted. - Provide an example of how a person would use the prototype. - Provide evidence that the application meets its goals.	The demonstration convincingly covers the required objectives and holds the *attention* of the audience. It is clear and rehearsed; flows nicely over the different topics and speakers. The demonstration is coherent, *concise interesting*, and *informative*; and is complete within 4 minutes. The demonstration is not rushed.	The demonstration convincingly covers the required objectives. It is clear and rehearsed; flows nicely over the different topics and speakers. The demonstration is coherent; and complete within 4 minutes. The demonstration is *not rushed*.	The demonstration covers the required objectives. The demonstration is *clear* and *rehearsed* and complete *within 4 minutes*.	The demonstration *somewhat* covers the required objectives. OR The demonstration is *longer* than 4 minutes.	Fails to demonstrate.