**Evaluation** means grading the usability of the system, as in how easy it is for user to interact with. Evaluation can help with identifying what problems there are with the usability of a system (if there are any). Without evaluation it’s difficult to know if the system is usable or not or how to make it even more usable.

Systems can be evaluated quantifiably or quantitively.

Evaluations can be done through surveys or interviews etc.

The doctor wanted us to focus on qualitative approaches.

**Qualitative evaluation** is basically the experience of the user or a description of it. It’s hard to quantify an experience. Developers try to put themselves into the user’s place when evaluating their design to get an assessment of the usability of the system and the experience to evaluate themselves or with users.

# Cognitive Walkthrough Evaluation

To effectively evaluate the system, there are requirements:

## Requirements

There must be a **description or prototype of the interface**.

There must be a **description for the tasks that are going to be performed** by the design.

There must be a **list of actions to complete task**; steps that must be taken to complete at task. The evaluation should take into account if the actions are clear or easy to perform.

The **users’ background must be known**. It should be known who the users that will be using the software are.

## Looking for during evaluation

Will user know how to perform the actions?

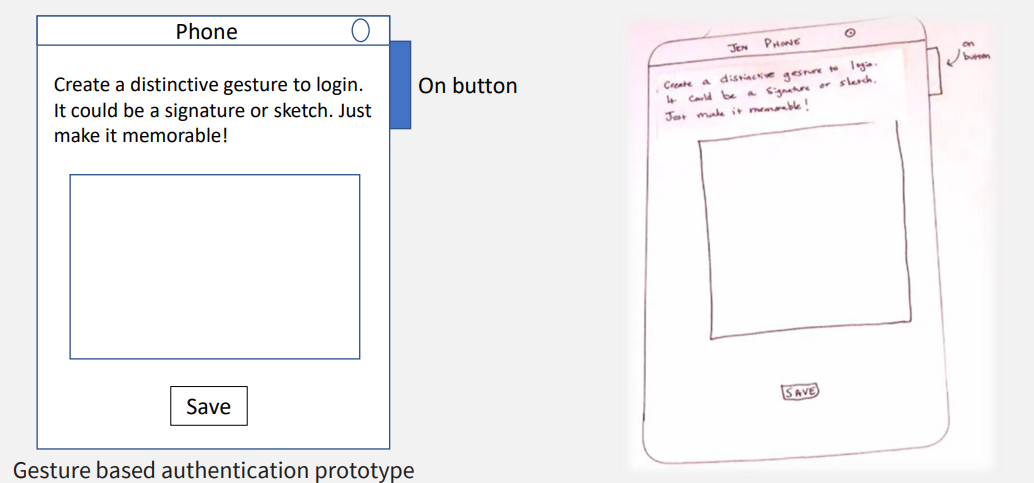
Will user see and recognize the control? (The power button)

Will users know that the control does what they want?

Will users understand the feedback?

The example used by the doctor was a phone authentication app.

The doctor evaluated this prototype:

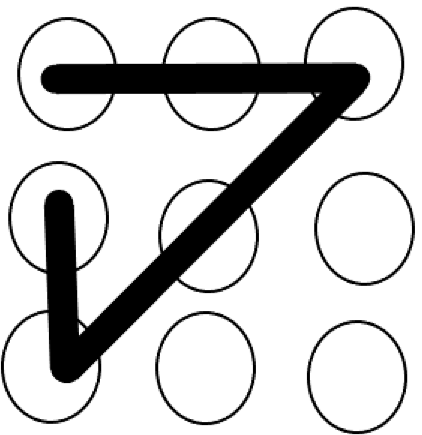


As it can be seen here, the prototype is crudely drawn, which is no problem for the purpose of evaluation.

The prototype is supposed to be a phone screen.

As agreed in class, the term “gesture” was a terrible term used to describe what the app actually does. The term was changed to “pattern”. Wherever in the images it says gesture disregard it and assume it says pattern.

The authentication uses the nine-dot pattern that most android use, it looks something like this:



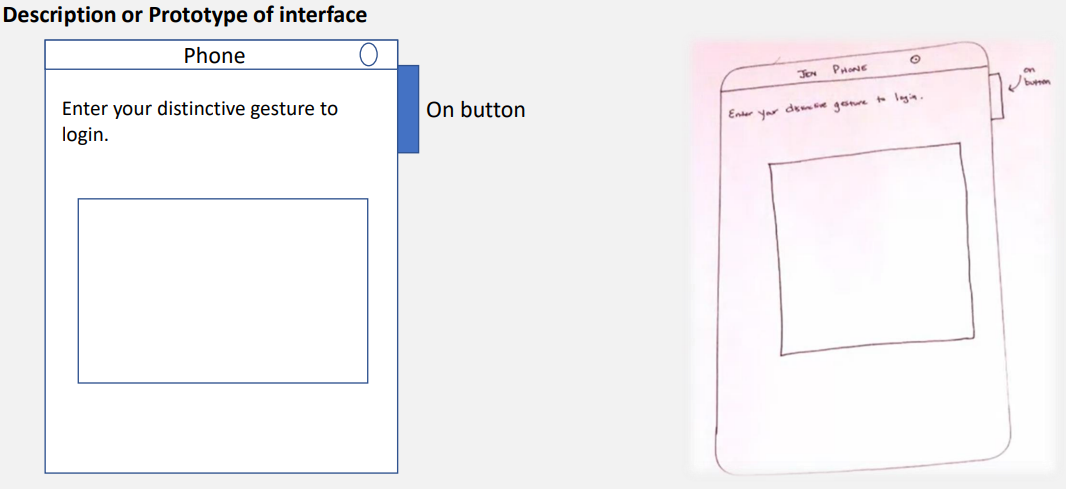
The on button the side is the control of the application.

The save button is used to save the pattern the user has chosen.

The box is used to draw the pattern (I think there’s supposed to be some dots in it).

The previous screen us used for the user to create their own pattern to be used for logging in.

### Task Description



This screen is the login screen for the application. The user can use their previously created pattern to login in to the mobile phone.

Prototypes can be drawings or 3D.

The **task description** for this screen is simply login, or authenticate to be logged in.

*It’s a simple task so it doesn’t need any more for its description.*

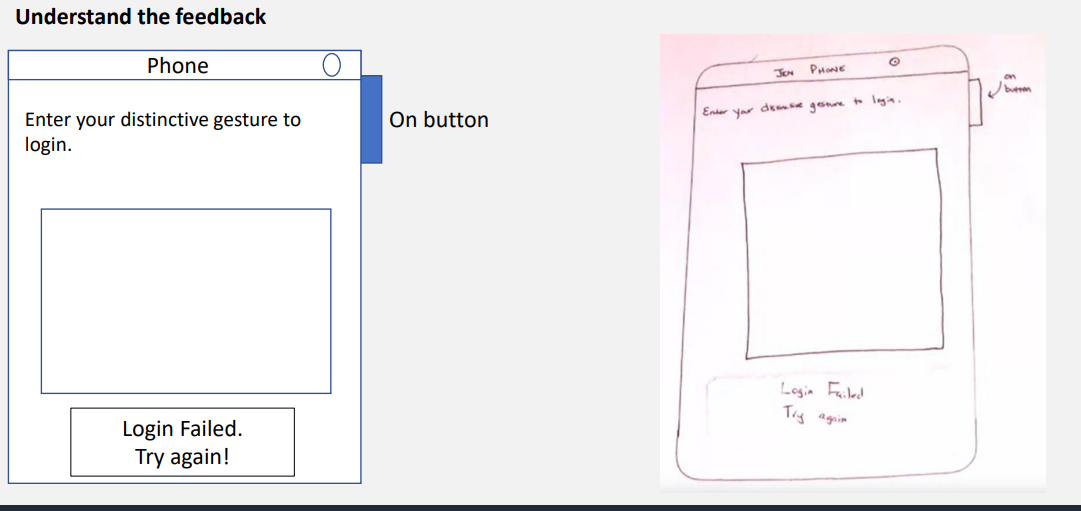
The actions taken on the login screen:

* Drawing the Pattern
* Clicking control (on) button
* Login screen appears
* Authentication successful, user logs in
* Feedback message

The slides are missing the feedback message, which is the message that appears when the gesture is entered incorrectly. (the error message is the feedback in this case). A user must be able to understand and fix any problems that might appear in an error message and continue towards their task. Feedback is when information is being relayed from the program to the user.

For any pattern login screen, using a log in button is very unusable for mobile phones. It’s best to use buttons in mobile apps except if absolutely necessary.

### Understanding Feedback



This screen contains the feedback when the user enters the wrong pattern.

Understanding the feedback is important. If the login fails, a message should be displayed to the user stating to try to enter their pattern again. In other words, if the user enters the wrong pattern, they should be told that and be given information on what to do next (in this case “Login Failed” is the telling the user they used the wrong pattern, the “try again” is telling the user what to do next)

# Heuristic Analysis

Heuristic analysis is also a qualitive evaluation.

Heuristic analysis bases the evaluation on a group of guidelines placed by experts of the domain where the program will be placed. These guidelines are called the rules of thumb.

If the guidelines are met, then it can be said the design is usable. If the guidelines are not met, it does not necessarily mean the design is not usable, because the guidelines could have been created based on a very complex system with many sub-systems and features while the program that is being tested is not nearly as complex.

The more guidelines met, the more usable the design is.

Sometimes, the system can be so simple that does not need to meet some of the guidelines can those guidelines can be skipped. The guidelines that should be focused on in this case are the important guidelines. If those guidelines are missed then it should affect the evaluation of the usability of the design.

## Nielson Heuristics

(can be found in more details here <https://www.nngroup.com/articles/ten-usability-heuristics/>

and here <https://aelaschool.com/en/interactiondesign/10-usability-heuristics-ui-design/>

There are major differences from what the doctor explained versus the actual person who came up with the heuristics)

Nielson Heuristics is one of the approaches that uses heuristics analysis. It’s basically some guidelines. They denote that the design should:

* **Have a simple design** and **natural dialogue** (for example pattern is a better term to use than dialogue)
* **Speaking the user’s language:** means not to use technical words (like SSL or Certificate) or internal jargon that a user may not understand. Real world conventions should be used. (In other words, users shouldn’t have to ponder or look up what the meaning of a word is)
* **Minimizing the user’s memory load:** (like with psychological acceptability) it’s best not to overburden the user with too much information. The user shouldn’t need to memorize the screen, the user should only need to know remember only the pattern and how to enter it. (Unlike with PowerPoint where every little detail is needed to be memorized in order to use it efficiently)
* **Being Consistent:** All the designs of the software should be similar to the user. The software should also be consistent to the environment.
* **Giving Good Feedback:** The user should be told concisely and clearly what went wrong in plain language and suggest a solution.
* **Providing Clearly Marked Exits:** The user should not need to press the control button exit the program, there should be a clear exit button like in excel (or like a cancel button?)
* **Having Shortcuts for Expert Users:** Shortcuts should not be implemented for small projects like authentication they’re more used for bigger projects. Shortcuts can help speed up the process for people who have interacted with the program before. The shortcuts should be hidden for new users.
* **Preventing Errors:** It’s best to try to design the screen in a way where mistakes are less often to occur. An example of preventing an error can be avoid placing a save button next to the exit button or the power button. The design should be able to place the positions somewhere logical.
* **Giving Good Error Messages:** Feedback given to the user should contain broader information, as in the error and what to do next. But error messages give only what went wrong.
* **Providing Help in Documentation:** Sometimes, it’s more appropriate to not place additional explanations in the program itself and instead be placed in an external source. But the external source should contain the necessary information a user may need to solve a problem they are facing when using the software.

If the authentication prototype does not follow all of Neilson’s rules, it does not mean it is not usable.

Shortcuts and providing help in documentation are hard to implement in a small program like this, since it’s very trivial.

For practice, it’s best to go through the prototype and try to apply the guidelines just discusses to see which guidelines are applied and which aren’t. Can check the slides for this too.

# Personas

Personas are basically just a person with a goal and a certain background. Based on what their goal and their background is, it can be deduced how they may act when presented with the design. A persona should represent a group of users, where whatever applies to the persona applies to the users. In other words, if the persona finds the program usable, then the group of users should also find the program usable too.

Here is the frank persona:



It’s important to take a look at the background to know the persona’s cognitive skills.

Based on the information given in this persona, it can be said that frank has experience with technology and would have no trouble with a laptop or mobile application since he owns an iPhone and laptop.

Frank will probably be interested in encryption, especially for encrypted communication, since he a political activist.

A savvy computer user is a smart computer user, can deduce a solution and solve their own computer problems.

Since Frank is a mobile phone user, especially a savvy computer user, Frank would probably have no trouble using the authentication app, it might even be considered easy for him to use.

Personas are often difficult to build, they may need an entire team to build it.

# Conclusion

Qualitative evaluation shows how usable a system is without needing measurements or timings (like numbers).

There are different levels of Qualitative complexity evaluations, heuristic evaluation can be easy and straightforward to use. While developing and using personas can take a long time to do (hours or days) and may need an entire team to invest their time and effort to build the personas then do an analysis based on the personas.

While some of the qualitative evaluation approaches are quick and inexpensive, they might not take into account important details that a user may provide. That is why there are some types of qualitative evaluation that involve users.