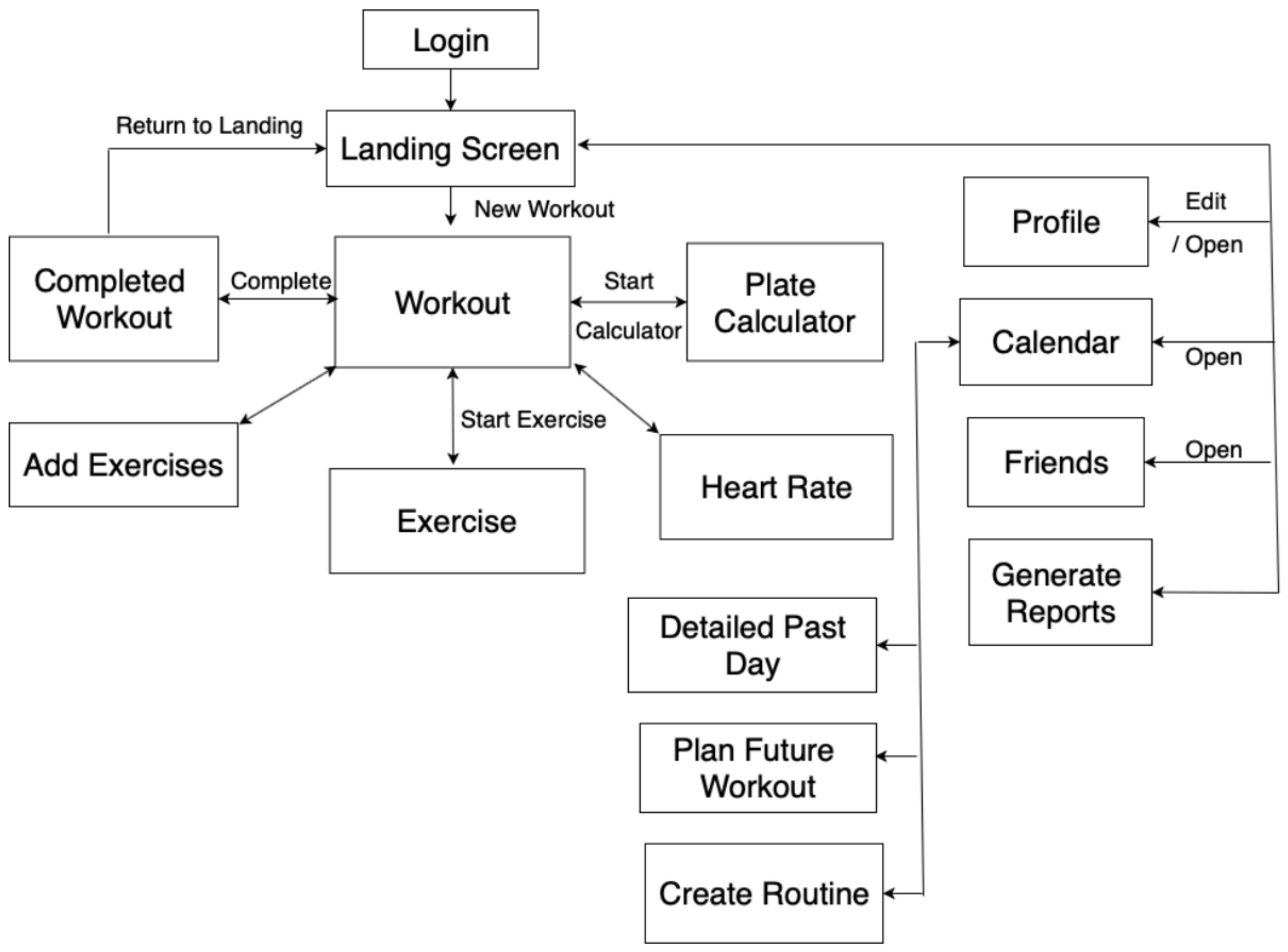
Beginning UX Development

When starting UX development for the first time, it might be tempting to plan all of the main screens of the application and how they will all map together, resulting in something like this:



In previous semesters of this class, teams would build maps like this, create wireframes and mockups for these screens, and then implement the screens when the sprints began. Teams overall were successful in executing their plan, and their applications usually ended up pretty close to what was planned out.

So what's the problem with this? It seems to be a proven way to build applications.

These plans were designed by groups with largely no application experience, and yet the apps tended to end up following the plan. While this seems like a positive outcome, it is actually a negative one.

By the time sprints began, a ton of work was already put into the app, creating each screen, making them look kind of nice, and then wiring them all together.

At this point, the flow of the application and the look of the screens was largely set in stone because making any major changes would be a massive amount of work.

This means that any actual user feedback received could have very little effect on the applications' user experience design!

Applications ended up close to the original plan not because the designs were good, but because they were too hard to change.

This is not just true for Software Engineering as a class, but as a field as well. When design happens as an up front step, we lose much of the ability to adapt our user experience as we learn throughout our sprints and demos. When we must adapt, it requires much more work than had we grown our UIs with the implementation of the application.

In the present version of the class, we are going to practice developing our interfaces concurrently with the application functionality, in order to better adapt to our ever changing understanding of our product.

So if we want to grow our UIs according to user feedback, where do we start?

What screen should we make first?

Students in the past have often spent their first sprint creating a login page. When it comes time to demo their app, they showcase the ability to log in.

What does this tell us about how well they are meeting the needs of their users?

While a login screen might be necessary for a finished app, it at best represents a step that the user must do before getting any value out of the app. In other words, the user gets no immediate value out of a login screen, and yet this is what teams tend to want to build first because it is “what happens first” according to the their plan.

In an Agile approach, we want to focus on delivering the highest amount of user value in the smallest possible release. A login screen has no place in the demo for sprint 1, it is at best a side task in adding user value.

Instead, what we need to build first is a screen that represents the core of the value we are offering to the customer.

In a game, this might be an early rendition of the most basic gameplay elements.

In an organizer, this might be the earliest version of a screen that allows the user to enter and view the elements meant to be organized.

In a map based app, this is a map with some basic intractability related to our app goals.

In order to choose the first screen to develop, we need to look at the user outcomes we are trying to achieve, and approach solving them in the simplest possible way.

Building this screen first, and getting feedback on it early, is the best chance we have for ensuring that we are on the right track to meeting our users needs.

Making that first screen

We have decided on a rough idea of what outcome our screen needs to try and solve, but how do we begin turning that into a screen in our application.

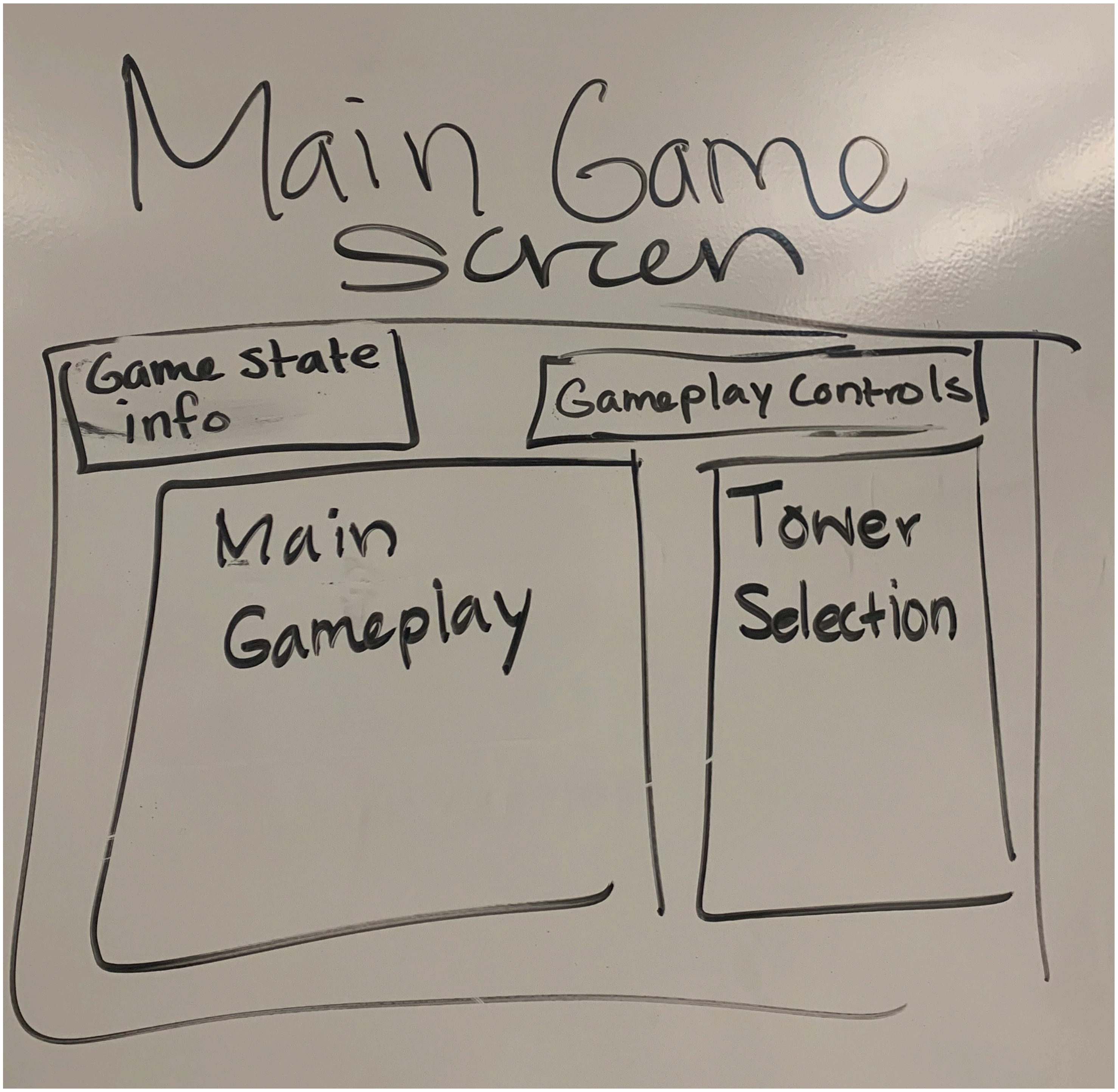
Like every other aspect in Agile Software Engineering, it starts with a conversation. As a team we need to discuss:

- What is accomplished on the page?
- What is the intended outcome?
- What information needs to be provided to accomplish this outcome?
- What actions need to be performed to accomplish this outcome?
- How should this info and these actions be organized?

As these questions are discussed by your team, members should be quickly sketching out ideas, on paper or a whiteboard. By the time the team has discussed all of this information, you should be ready to create a wireframe.

Wireframes

At its core, a wireframe is just a quick representation of the content areas and layout elements of an app screen.



It is meant to serve as a record of the conversation you had to answer the questions just discussed.

The wireframe is intended to serve as a discussion and team communication tool in order for everyone to understand the very basics of what a page does, as well as what information and choices are presented to the user.

In *Lean UX* a great example of using wireframes is described:

“A few years ago, Jeff was designing a dashboard for a web app targeted at the 'The Ladder’ recruiter and employer audience. There was a lot of information to fit onto one screen and he was struggling to make it all work. Instead of buying too much time at his desk pushing pixels (referring to making a detailed high fidelity prototype), he grabbed a whiteboard and asked Greg, the lead developer, to join him. Jeff sketched his original idea about how to lay out all of the content and functionality for this dashboard. The two of them discussed the idea, and eventually Jeff handed Greg the marker. He sketched his idea on the same whiteboard. They went back and forth, ultimately converging on a layout and flow that they felt was both usable and feasible, given that they needed to deliver a solution within the current two week sprint. At the end of that two-hour session, they returned to their desks and began working. Jeff refined the sketch into a more formal wireframe and workflow while Greg began to write the infrastructure code necessary to get the data they needed to the presentation layer (the stuff that the user sees)."

This is exactly the intended purpose of the wireframe. They discussed what the page was for, and quickly discussed the main layout elements just enough for both to come to an agreement about the work that needed to be done next.

Wireframes are not the time to go into a ton of detail about interaction mechanisms or the look and feel of the app, they should be fast to make and able to be thrown away without anyone feeling a ton of time was wasted.

Keep in mind that throughout this process, necessary backend tasks might be discovered. These should be added to the task breakdown of the User Story card.

In summary, wireframes should: * Be created as a part of a team discussion about accomplishing a user outcome. * Contain just enough information to understand the main layout elements and tasks needed to make this screen work.

Alright, so we have got our first wireframe, but how do we turn it into a screen?

Mockups

In industry, prototyping a user interface is often referred to as having a certain amount of fidelity. Fidelity refers to how close to the realized screen the prototype looks like.

A wireframe has a low fidelity, in that it only will vaguely resemble the implemented screen.

In this class we use a term called “Mockups” to refer to an intermediate stage of medium to high fidelity prototyping.

In this stage, we are trying to capture what this screen might actually look like in the application, so that we can get feedback on its appearance and usability.

In industry, these prototypes could still be made using image editing software like photoshop or specialized prototyping tools.

In this class, we will be making our mockups in the actual tool the screens will be made in. For Android apps, this means the mockups will be coded in XML.

The reason that we do this is to make sure that your design imagination does not far outstrip your ability to implement. It is often the case that the students in SE will come up with designs that are too complicated for them to actually make. If a lot of time was spent making these designs in an intermediary tool, that time would be wasted.

Therefore, we feel it is in your best interest for your prototyping to be done in the actual tool you will be using for your application.

The development of the mockups in this class will take place in two stages:

- Medium fidelity and non interactive
- High fidelity and interactive

Medium Fidelity and Non Interactive

Remember, at this point all your team has agreed to is a rough sketch outlining some simple boxes. Each member may have a completely different idea of what this screen is going to end up looking like.

The point of this stage is to quickly establish a clearer vision for the look of this screen and gain approval by teammates and potential customers.

There are multiple methods for the creation of this step that can work, so we won't prescribe one to you. Instead we will just list a few methods we have seen:

- Creating the mockup as a team. Members are sketching their ideas and providing feedback as one member codes.
- Creating individual mockups. Multiple people have competing ideas and each go and create their mockup, and then the team discusses which elements they like best.
- A single member has a strong vision for what the screen should look like, the team agrees with this vision, the member implements it, and then gets feedback from their team.

Each of these methods can certainly produce nice mockups, although the last two methods tend to involve more mockup iterations.

At the medium fidelity level, we are still trying to quickly produce a approximation of what the screen might look like. We want to make sure we have the main layout and primary interaction elements. Stylistic details and “frills” might be left out. See Figure 1 for an example mockup from a previous students.

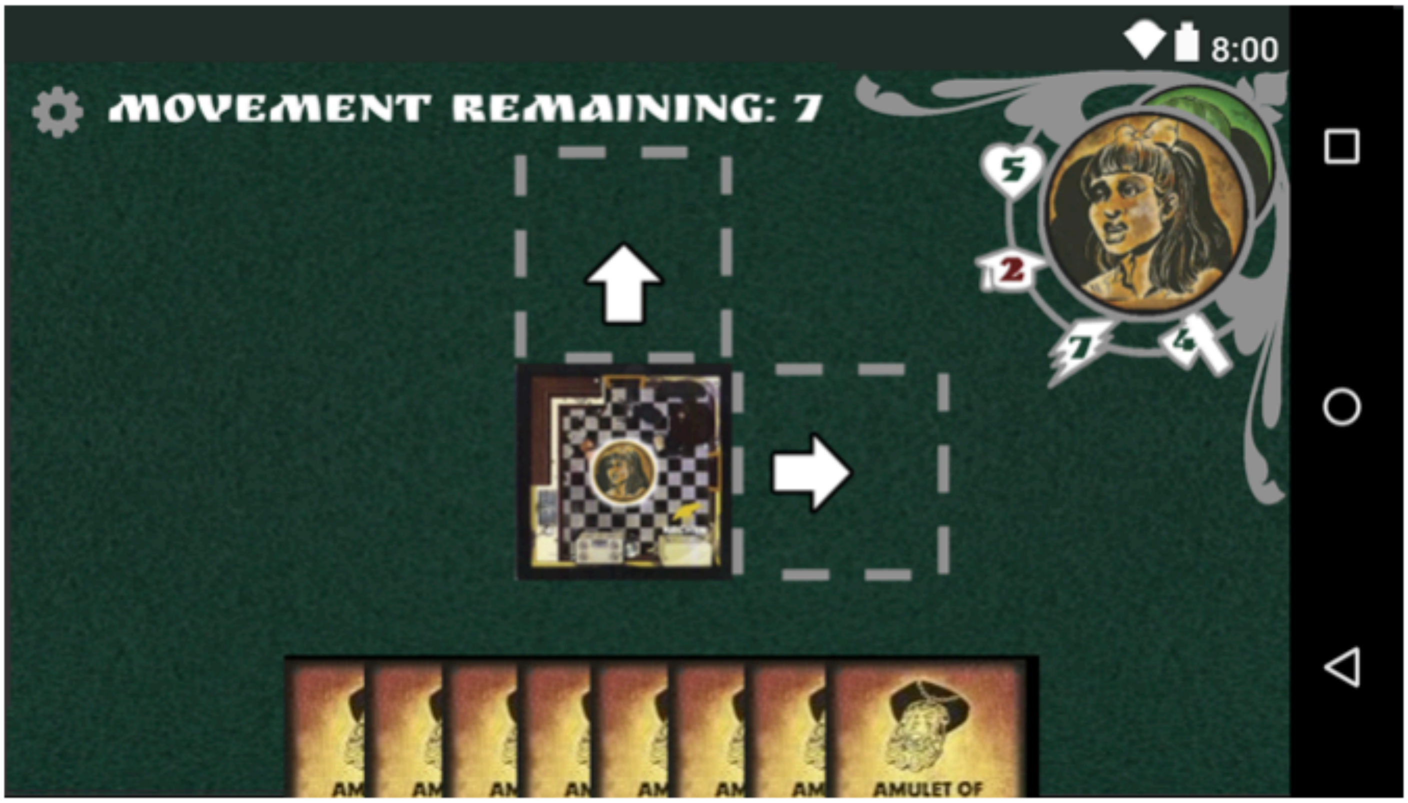


Figure 1: An example mockup from a student capstone assignment.

In this class, mockups at this stage are non interactive, meaning that the UI elements aren’t coded to do anything. At this point we are just organizing the look and layout of the screen.

Once we have finished making the medium fidelity mockup, it is time to bring it back to the team for discussion. Any mockup for a screen should be approved by the team before a significant amount of work is put into making it function.

The following is an example acceptance criteria for a good medium fidelity mockup:

- It is clear how this page helps the user accomplish their intended outcome.
- The main UI elements involved in the new user story are displayed.
- The page looks nice and if applicable, matches the look and feel of the rest of the app.
- There is an appropriate level of detail (not too much or too little)

When something is rejected, be sure to provide clear reasoning on why it was rejected and how it could be improved. This is not just a valuable learning experience for the mockup creator, but for the entire team.

When you team has approved the mockup, we are ready to proceed.

High fidelity Interactive Prototype

When a medium fidelity mockup is approved by a team, you are really approving more time and effort being spent into making that mockup look nice and beginning to make it functional.

The end result of this stage is essentially the screen implementation you are going to use during your sprint 1.

This is the time to flesh out the interaction elements, as well as to have those interaction elements do something.

At this point, the interaction elements won't actually do all that they are supposed to do. when the screen is fully implemented, What they should do is at least indicate to the user what is supposed to happen.

This can take the form of going to a new screen (although probably not in sprint 0, we want to keep our application small), or just having a pop up message that indicates what will happen when this element has been fully implemented.

In future sprints, as your application grows, this will be the point in which you fully integrate this screen into the flow of the other screens of your application.

When this mockup is finished, it should be once again brought to the group for approval.

The following is an example acceptance criteria for a good high fidelity prototype:

- The screen would also pass mockup approval (looks nice, etc.)
- The purpose of the interactive elements is made clear to the user.
- The prototype has been tested by others.
- Any layout changes are reflected in the wireframes.

As you have developed the mockup and the prototype, you will probably have learned a lot of new information about the needs of that screen, possibly causing the screen to diverge from what is in the wireframe.

The wireframe should be updated as new information is brought to the teams attention, but the changes should be made as a team, and not by individuals.

It is important to keep the wireframe up to date with the application because it will often serve as a starting point for future modifications or a complete redo of a screen. The wireframe therefore needs to hold our current understanding of the necessary screen elements and information.

The following is an example acceptance criteria for a change to a wireframe:

- The modification clearly belongs on this page.
- The page retains a clear purpose.

- The main UI elements involved in the new user story are displayed.
- Nothing is removed that is necessary.
- Appropriate detail level (Not too much, not too little).

In the end of this process, you will have a version of the screen that is ready to be implemented throughout the sprint, as well as a basis for planning future changes to that screen.

Establishing A Look And Feel

At this point we have established a process for the creation and integration of a main screen for your application. You might be wondering “But how do I make it look nice?”

Unfortunately, there is no secret process that will allow any developer to suddenly start creating beautiful user experiences for any type of application. I am certainly not at that level, nor have I ever met anyone like that.

Creating a great user experience can be highly individual to each application we make, and can often require a great deal of vision, creativity, and sometimes years of experience in that application area.

However, there are some good practices to follow that do tend to apply to most applications we create. There will of course be some exceptions for applications that seek to push the boundaries of user experiences.

1. Have a consistent look throughout your application.
2. Have your interfaces be consistent with industry standards.
3. Keep information simple and readable at a glance.

Having a consistent look:

Have you ever noticed almost every screen of a popular application looks kind of the same as the other screens of that application. They might use the same colors in similar functions and locations, or the same kind of buttons throughout. Perhaps there is a top application bar that follows the user throughout their whole application experience. All of these things are constantly reinforcing that these screens are part of the same application.

When a user is assessing the quality of an app, it turns out that consistency is one of their highest metrics. Users are unhappy when the screens are inconsistent throughout or with each other.

How do we make sure our application is consistent?

1. Have a color scheme
2. Use the same kinds of buttons for the same kinds of purposes

3. Repeat the main navigation systems as much as possible.

More information:

Making and using a color scheme

More about consistency

Have your interfaces be consistent with industry standards

Have you ever noticed that every music app has the exact same interface for controlling music playback?

The reason for this is that customers are expecting that interface any time they want to play music on a phone, regardless of what service they are using.

Using these industry standards have several benefits.

- They make your app look more professional (it looks like other professionally made apps).
- Users will likely already know how to use the parts of your app that use these standard interfaces.

When we deviate from these industry norms and "reinvent the wheel", defining our own unique icons and buttons, we run the risk of confusing users who expect the normal interface.

More information:

A collection of UI design patterns and guides

Keep information simple and readable at a glance

Keep in mind that this section is referring to developer provided information, and not user generated content such as social media posts or messages.

Students in this class tend to drastically overestimate the amount of information that a user is willing to read on a particular screen.

When you have more than just a few words on an application screen, unless users are heavily invested in the application, a large majority of those users are just going to skip over the text completely or at best skim it.

The best way to follow this rule is to not try and provide large amounts of information at a time to your users. If possible, show them rather than tell them.

If for some reason you have to convey a lot of information to your users, try to employ some of the following techniques for breaking it up:

- Use small paragraphs
- Have meaningful sub-headers
- Use bulleted lists
- Remove any unnecessary wording

More information:

How Users Read on the Web

In Summary:

Throughout this reading we have introduced you to the activities involved with developing an initial user experience for your application. Specifically we covered:

- Starting our application's design in an Agile way.
- Choosing the first screen to create.
- Creating wireframes, mockups, and high fidelity prototypes.
- Creating a nice look and feel for our applications.

Of course, entire industries are built around UI and UX development, so this is not even close to everything there is to know. It is however, enough to get started learning.