



MOBILE APP DESIGN
AND DEVELOPMENT
498-02

Story Mapping

Story Map as grading rubric

2/3rd project graded based on Story Map (UIUX the rest)

Two checkpoints:

1. Skeletal Product (Thu, 11/12, 2 pm)
2. MVP (Wed, 12/09, 2 pm)

} feature list determined by hand;
LLM has no concept of effort
and time required

Together they form 53% of course grade, **cumulative**

- features not completed by SP check point can be completed by MVP
with 25% penalty; MVP must include all SP features

Re-negotiation: Skeletal Product or MVP feature may be
replaced with **equally weighted** Stretch Goal feature(s)

How many features?

If you have a small number of features, a large portion of your grades will be riding on each

- in the **extreme**: only 1 feature, not completed: **team gets 0**
- no **partial credit** because we don't know how much you have done

Instead, break each feature down into **unit-testable** sub-features and/or functional blocks

- give yourselves credit for partially implemented features
- list implementation details/steps for partial credit, **may ask LLM to assist**

Core features

Focus on **core features** that deliver your **value proposition**

No need to list supportive features such as:

- account setup and login
- navigational: next, back, submit buttons
- reporting bugs and giving feedback
- integrating 3rd-party payment, subscription
- **LLM already so instructed**

Features with the **same** or **duplicated** implementations will be **merged** and **counted as one**

No UIUX Features

For example: for [Find Books](#), core features are:

- [search](#): look up search term in database
- [browse](#): display related titles
- [wish list](#): enter titles into user's database

[Exclude](#) View or UI/UX features for now, such as:

- create search box
- create search button
- create wish list button

No UI/UX Features

For example: for **Find Books**, core features are:

- **search**: look up search term in database
- **browse**: display related titles
- **wish list**: enter titles into user's database

DO include functional blocks, such as, for **search**:

- design database schema
- implement search query
- search external database(s)

} we'll review your EGM
Scoping results
together and decide
on finer scoping

Assume use of 3rd-party SDKs/tools/libraries, e.g., SQLite

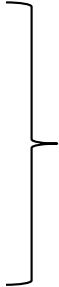
Storymap: highlights

1. As **contract**: grading rubric for 2/3 project grade
2. Skeletal product and MVP: **cumulative** with **25% penalty**
3. Break down features for **partial credit**
4. **Core** vs. support features, no UI features yet
5. **Do include** implementation **functional blocks**
6. Stretch goals: for feature replacement, requires **re-negotiation**
(no removal without **replacement**)

Use Trello board

To keep track of course project: required lists

(<https://trello.com/b/MyPhHMGd/441-template>):

- 1. Project backlog
 - 2. Skeletal Product Completed
 - 3. MVP Completed
- 
- used in grading

Ok to track Presentation and other tasks on Trello,
but use [separate lists/columns](#), not mixed up with project tasks

On Trello

Story map as cover image for [Project Backlog](#)

ALL skeletal product features, in cards

Each feature in a separate card with:



1. [Acceptance Criterion](#)
2. "Skeletal Product" label
3. sprint number and timebox as label
4. assigned member(s)
5. link to [specific lines of source code](#) (when available, at code drops)

Acceptance Criteria

Written in modified Gherkin syntax:

- StoryMap feature title used as GIVEN



- Trello card title **MUST match EXACTLY** the title in your StoryMap, else lost mapping

- WHEN an **event**, such as a **user action** or function call, **is triggered**

- **can only be ONE, no AND!**

- THEN the **feature responds** with **certain action** with some **expected outcome**

- AND some additional outcome (optional)

LLM will get these right?

Sprints

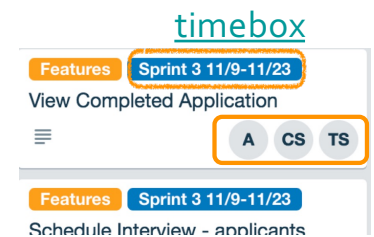
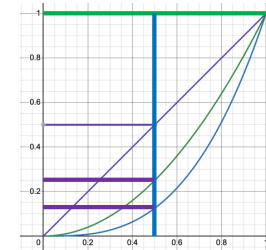
How much
review is
required?

Skeletal Product due in about 6 weeks

- you have N features to be completed by then
- how many and which features you want done the first week?

That's your **first sprint**

- who will work on which feature(s) in the 1st sprint, etc.?
 - used to compute **non-constant individual grade function!**
- label each feature card with a **sprint timebox**:
[Sprint 1: 10/3-10/10] and **assigned members**
- at the end of each sprint:
 - move completed features to **SP-completed** or **MVP-completed**
 - for features not completed, add comments on what was completed, what is the blocker, and move card to a new sprint



Master Competency

During in-person [code reviews](#):

- [pinpoint](#) where in your code certain functionalities are implemented
- explain [how](#) the functions are implemented
- explain [what](#) a piece of your code does and [why](#) your program needs it
- [modify](#) your code to achieve a specific outcome
- provide [engineering justification](#) for your design and implementation choices

Keep it flat and simple: see [AGENTS.md](#)

- no AI slop that you don't understand, no dead code
- no over-engineered solution
- no [legacy patterns](#)

Using EGM to Construct Story Map

Project Scoping Alternatives DUE Thu, 10/01, 2 pm

Story Map DUE Thu, 10/08, 6 pm

First Draft

Context Refresh:

- Elephant generates a hard summary of final constraints (Exhibit E-H)

First Draft Technical Proposal:

- Elephant proposes implementation using prose and block diagrams (Exhibit E-I)
- Strictly no code, use plain English; team argues with AI until the design is solid

Story Map



Acceptance Criteria:

- Elephant writes GIVEN-WHEN-THEN rules (Exhibit E-J)

Goldfish 6 (QA Lead):

- Critiques criteria until consensus is reached

Story Map (on Trello): each feature (E-I)

- assigned a UX stage and acceptance criteria (E-J) and
- put on a Trello card, with a Sprint, contributor(s), and, eventually, linked to code

Documentation

On GitHub Wiki: initial documentation

- to be written using GitHub Markdown
- manuals available in spec
- free to GitHub Student accounts (must sign up)

Documentation

The GitHub Wiki :

1. Getting started: 3rd-party tools, libraries, SDKs (for now)
 2. LLM exhibits: your Scoping documents
 3. Project architecture and features (Source of Truth)
 - a) The Elevator Pitch: updated as necessary
 - b) The Technical Plan (with block diagrams)
 - c) Alternatives (requires 2 rejected alternatives per major decision)
 4. View UI/UX (placeholder)
 5. Team roster: individual contributions (placeholder)
- } initial version,
final version due
end of term
- >> Clear context and read in #a, b, c <<

Project Architecture and Features

A coherent, big-picture summary of your capstone project

- must explain how core features are implemented, include [links to relevant code](#) (when available)
- for non-core features, can simply link to code

Keep a running list of all 3rd-party SDKs/tools/libraries you use (no need to list dependencies of those tools)

Engine architecture

1. Rigorous Features of the engine architecture
 2. Control and data flows between them
 3. List all 3rd party libraries/tools you intend to use, if known
- } block diagram

The Critic Review

Project Architecture and Features Documentation

Goldfish 7 (Technical Reviewer):

- Hunts for missed assumptions, edge cases, and logic errors in the Wiki
- Loops until critiques are reduced to minor nitpicks

Goldfish 8 (Software Engineer):

- Verifies the Wiki has 100% of the information needed for a successful first-pass implementation
- Team answers any remaining questions directly in the document

