Lecture Notes
October 8, 2020

Rest of addendum from September 29 (fifth page) - licensing software engineers

Final individual project assignment “I3: Saving Program State” due tonight.
https://courseworks2.columbia.edu/courses/104335/assignments/482611

First assessment postponed from starting this Friday (October 9-12) to start instead the next Friday (October 16-19).

The first and second assessments from the last time I taught this course are available in Courseworks Files
https://courseworks2.columbia.edu/courses/104335/files/folder/from%202018/individual%20assessments
The first assessment references teams because the team project ran the entire semester last time, no individual project. The first assessment this semester will not reference teams or team projects, the second assessment will reference teams.

The team formation assignment is still due before class (9:59am) next Thursday October 15, but the deadline for the team preliminary proposal assignment has been extended.

“Assignment T0: Team Formation” due October 15
https://courseworks2.columbia.edu/courses/104335/assignments/486855

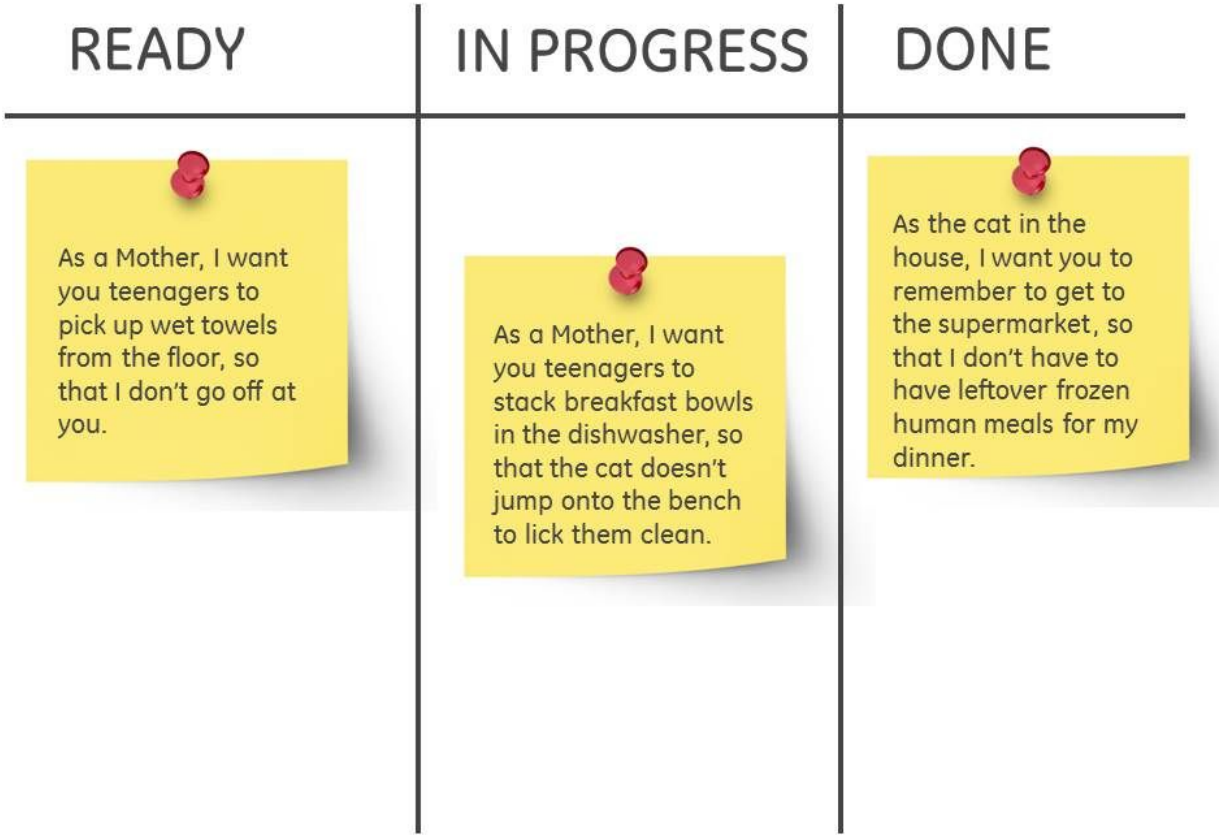
Try to form a team of exactly 4 members. Contact me asap (via piazza, not email) if you’d like to form a team of 5 members. Two or three members is not acceptable.

(Except for CVN students: There are 7 CVN students, so it’s ok in your case to form one team of 4 and one of 3.)

Submit team members (full name and uni), team name, programming language(s) and platform(s), team github repository. If you want to use multiple languages and/or platforms, e.g., front end vs. back end, please explain. Everyone in the team must submit the same thing.

Working in a team: *software requirements*

User stories may not provide enough detail for time estimation, for breaking down into tasks, for informing design, or for acceptance testing



What details are missing from these user stories that would be needed to make them testable?

Use cases are an alternative to user stories

The term “use case” is often used more generally to refer to business processes and products that do not necessarily involve software. Here we mean use cases that define software requirements

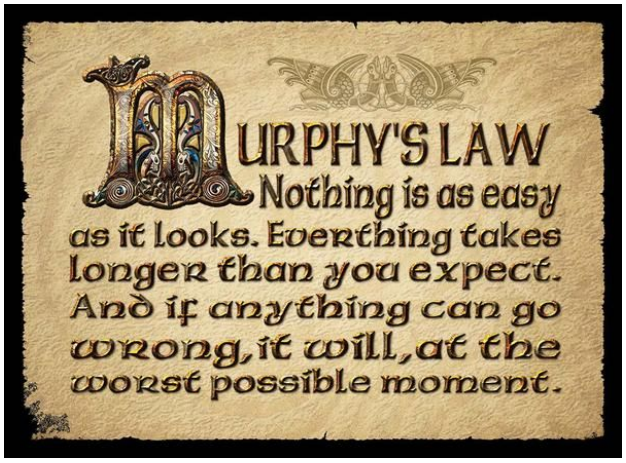
Use cases can be written from scratch, instead of user stories, or use cases can *expand* user stories - either during initial project planning or, more agilely, during iteration planning at the beginning of each iteration

Use cases describe the step by step actions or events of how the user role interacts with the software to exercise the feature and achieve the goal or customer value - longer and more elaborate than user stories

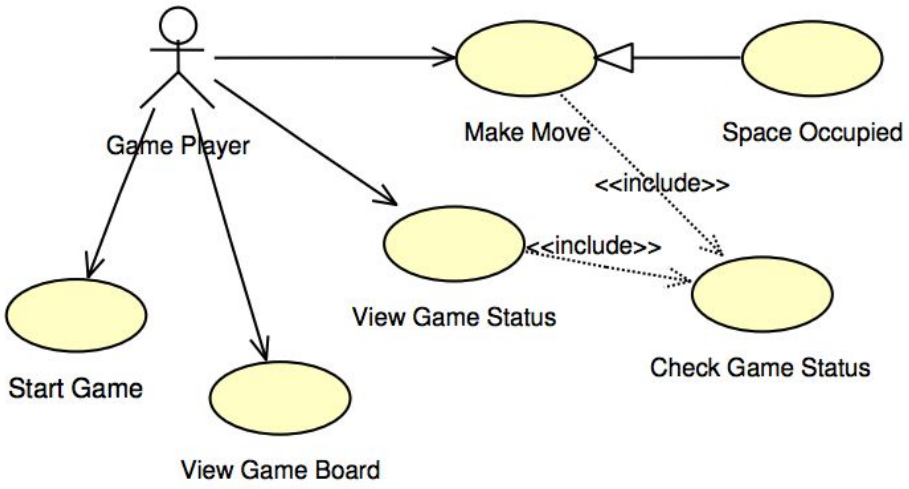
Use cases can also describe contingencies that should lead to different steps, or different ordering of steps

Testing can then check that actions indeed happen step by step as described under the appropriate conditions

Use cases also try to capture anything and everything that *can go wrong*, which might not be fully addressed by conditions of satisfaction



Use cases are often associated with use case diagrams, but not in the course - if you are asked for a use case and you submit a use case diagram, *you are doing it wrong*



Title: Edit an article

Actor: Member *(Registered User)*

Description: Part of a Wiki system. The member edits any part (the entire article or just a section) of an article he/she is reading. Preview and changes comparison are allowed during the editing.

Preconditions: The article with editing enabled is presented to the member.

Triggers: The member invokes an edit request (for the full article or just one section) on the article.

Basic flow:

1. The system provides a new editor area/box filled with all the article's relevant content with an informative edit summary for the member to edit. If the member just wants to edit a section of the article, only the original content of the section is shown, with the section title automatically filled out in the edit summary.
2. The member modifies the article's content till satisfied.
3. The member fills out the edit summary, tells the system if he/she wants to watch this article, and submits the edit.

4. The system saves the article, logs the edit event and finishes any necessary post processing.
5. The system presents the updated view of the article to the member.

Success Guarantees:

- The article is saved and an updated view is shown.
- An edit record for the article is created by the system, so watchers of the article can be informed of the update later.

Extensions:

2-3.

a. Show preview:

1. The member selects *Show preview* which submits the modified content.
2. The system reruns step 1 with addition of the rendered updated content for preview, and informs the member that his/her edits have not been saved yet, then continues.

b. Show changes:

1. The member selects *Show changes* which submits the modified content.
2. The system reruns step 1 with addition of showing the results of comparing the differences between the current edits by the member and the most recent saved version of the article, then continues.

c. Cancel the edit:

1. The member selects *Cancel*.
2. The system discards any change the member has made, then goes to step 5.

4a. Timeout: ...

…

Use cases capture *what* the software needs to do, not *how* the software should do it, but at finer granularity than user stories - thinks through user's needs and how they will use the system

Use case template:

Title (or Name) - capture a goal or purpose. The list of use cases names should summarize everything the system will offer

Description - one or two paragraphs

Actor(s) - user role who engages in use case, able to make decisions

Basic flow (happy path, sunny day path, success scenario) - numbered list of steps to accomplish goal

Alternate flows or extensions - numbered list of steps addressing less common user/system interactions, special cases; alternate flows split off from basic flow at step N start with Na, Nb, etc., can also converge into basic flow at some step

Inclusions (if any) - modular use cases that are “included” in this and other use cases, e.g., register, login, contact customer service

Exception flows - error cases, things that can prevent the user from achieving their goal

Preconditions (entry conditions) - anything that must be or is expected to be true before use case begins, specify whether can assume true or need to check if true (give error if not)

Triggers (more entry conditions) - if something must happen, such as user selects a menu item or the application state changes due to some previous activity, to initiate use case

Success guarantees (postconditions) - anything that is necessarily true when the basic flow successfully ends, because the flow makes its true

Minimal guarantees (more postconditions) - anything that is necessarily true when any flow ends

Events - any new triggers (for other use cases) initiated by the completion of the use case

Sometimes use cases are written where the basic and alternative steps are intermingled

Name of Use Case: Take Exam Online
Actor(s): ExamTaker
Flow of events:

1. ExamTaker connects to the Exam server.
2. Exam server checks whether ExamTaker is already authenticated and runs authentication process if necessary.
3. ExamTaker selects an exam from a list of options.
4. ExamTaker repeatedly selects a question and either types in a solution, attaches a file with a solution, edits a solution or attaches a replacement file.
5. ExamTaker either submits completed exam or saves current state.
6. When a completed exam is submitted, Exam server checks that all questions have been attempted and either sends acknowledgement to ExamTaker, or saves current state and notifies ExamTaker of incomplete submission.
7. ExamTaker logs out.

Entry conditions:

1. ExamTaker must have authentication credentials.

2. Computing requirements: supported browser.

Use cases can have multiple actors

Use Case Name: Place Order

Actors:

- User (shopper)
- Fulfillment System (processes orders for delivery to customers)
- Billing System (bills customers for orders that have been placed)

Triggers: The user indicates that she wants to purchase items that she has selected.

Preconditions: User has selected the items to be purchased.

Postconditions:

- The order will be placed in the system.
- The user will have a tracking ID for the order.
- The user will know the estimated delivery date for the order.

Normal Flow:

1. The user will indicate that she wants to order the items that have already been selected.

2. The system will present the billing and shipping information that the user previously stored.
3. The user will confirm that the existing billing and shipping information should be used for this order.
4. The system will present the amount that the order will cost, including applicable taxes and shipping charges.
5. The user will confirm that the order information is accurate.
6. The system will provide the user with a tracking ID for the order.
7. The system will submit the order to the *fulfillment system* for evaluation.
8. The *fulfillment system* will provide the system with an estimated delivery date.
9. The system will present the estimated delivery date to the user.
10. The user will indicate that the order should be placed.
11. The system will request that the *billing system* should charge the user for the order.
12. The *billing system* will confirm that the charge has been placed for the order.
13. The system will submit the order to the *fulfillment system* for processing.
14. The *fulfillment system* will confirm that the order is being processed.

15. The system will indicate to the user that the user has been charged for the order.
16. The system will indicate to the user that the order has been placed.
17. The user will exit the system.

Alternate Flows:

3A1: The user enters billing and shipping information for the order. The user desires to use shipping and billing information that differs from the information stored in her account. This alternate flow also applies if the user does not maintain billing and / or shipping information in their account, or if the user does not have an account.

1. The user will indicate that this order should use alternate billing or shipping information.
2. The user will enter billing and shipping information for this order.
3. The system will validate the billing and shipping information.
4. The use case continues

5A1: The user will discover an error in the billing or shipping information associated with their account, and will change it.

1. The user will indicate that the billing and shipping information is incorrect.
2. The user will edit the billing and shipping information associated with their account.
3. The system will validate the billing and shipping information.
4. The use case returns to step 2 and continues.

5A2: The user will discover an error in the billing or shipping information that is uniquely being used for this order, and will change it.

1. The user will indicate that the billing and shipping information is incorrect.
2. The user will edit the billing and shipping information for this order.
3. The use case returns to step 3A1 step 3.

10A1: The user will determine that the order is not acceptable (perhaps due to dissatisfaction with the estimated delivery date) and will cancel the order.

1. The user will request that the order be cancelled.
2. The system will confirm that the order has been cancelled.
3. The use case ends.

Use cases and user stories intentionally do not provide any direct means for describing the desired UI

One approach to UI requirements is to associate with wireframes. This is usually easier to do for use cases than user stories, since the steps likely to correspond to screens, commands, prompts, etc. already laid out

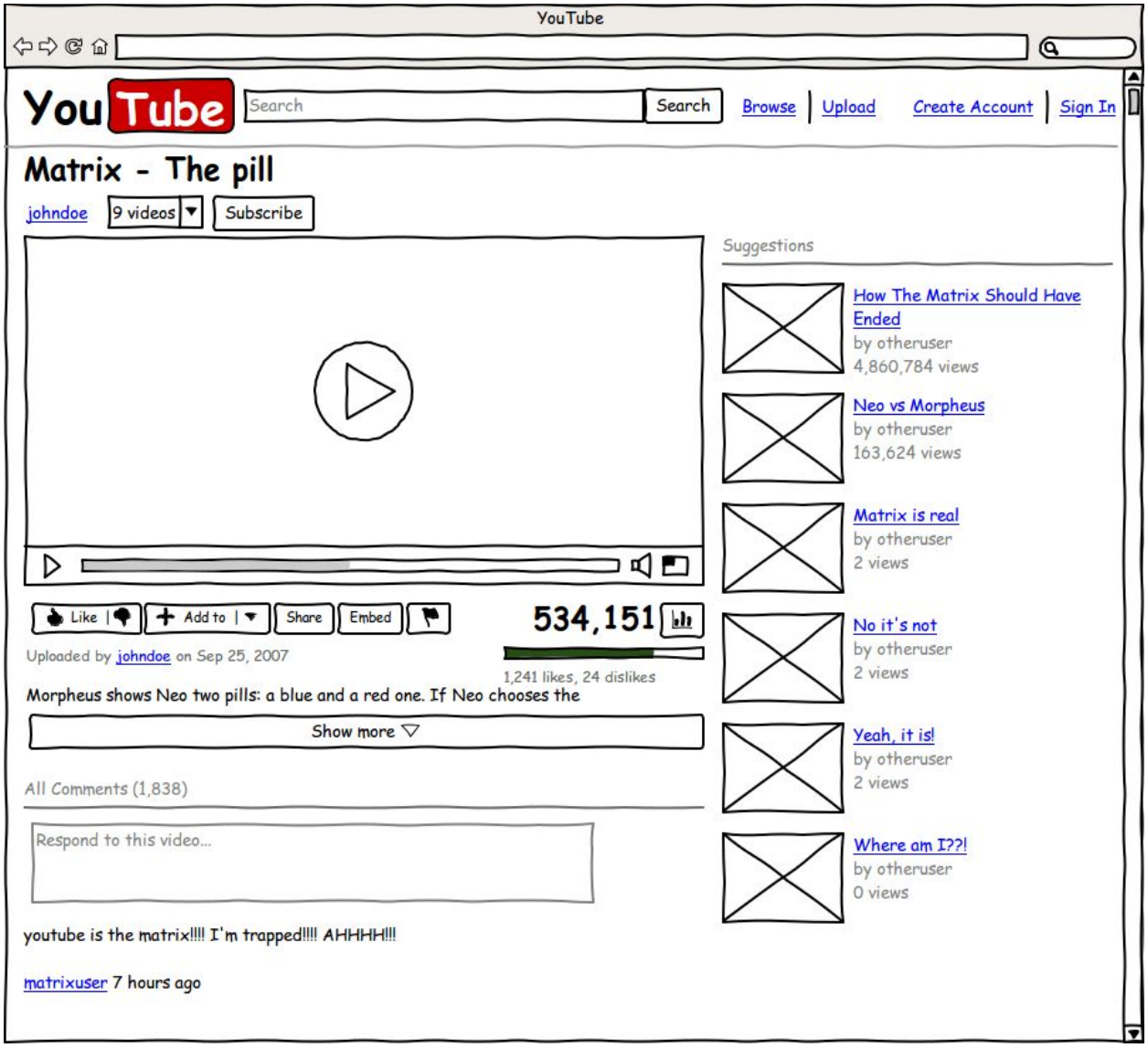
A wireframe is a sketch of a GUI screen, web page, command line console, etc. The concept arose in engineering design for skeletal three-dimensional models of physical objects or structures

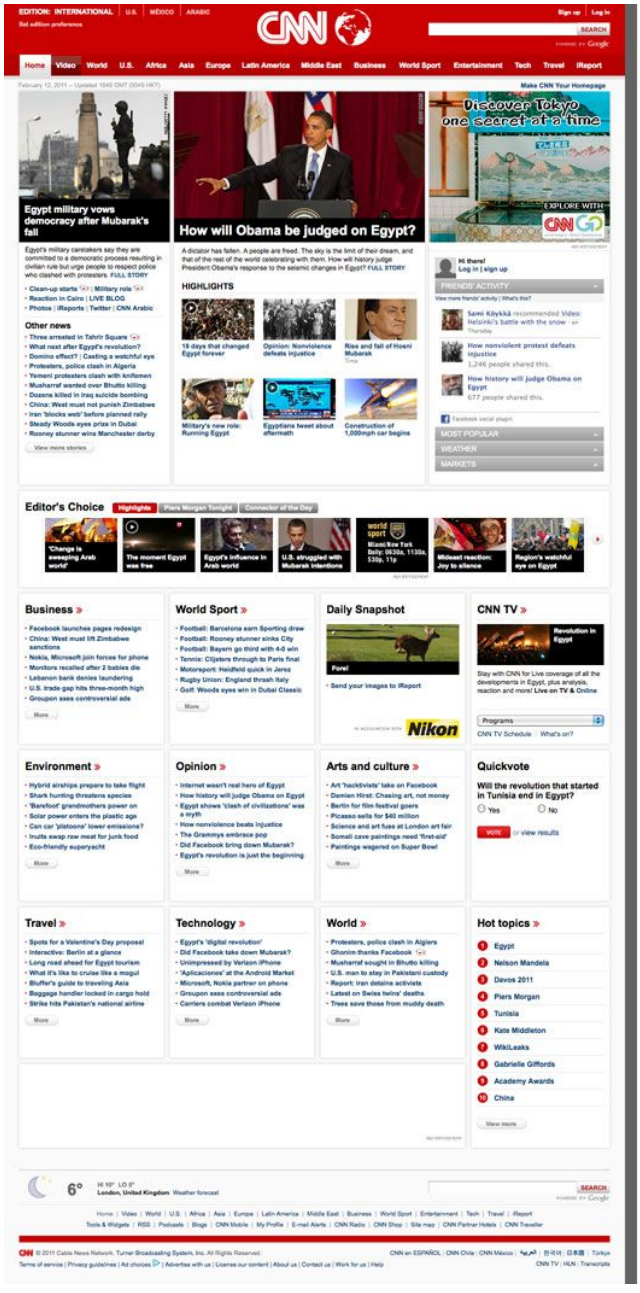
A software engineering wireframe lays out content and functionality, establishes basic structure and what interface elements will be shown

In the case of single-page applications - no you do not put everything in the same wireframe (unless that page literally never changes), different wireframes depict different states of that single-page

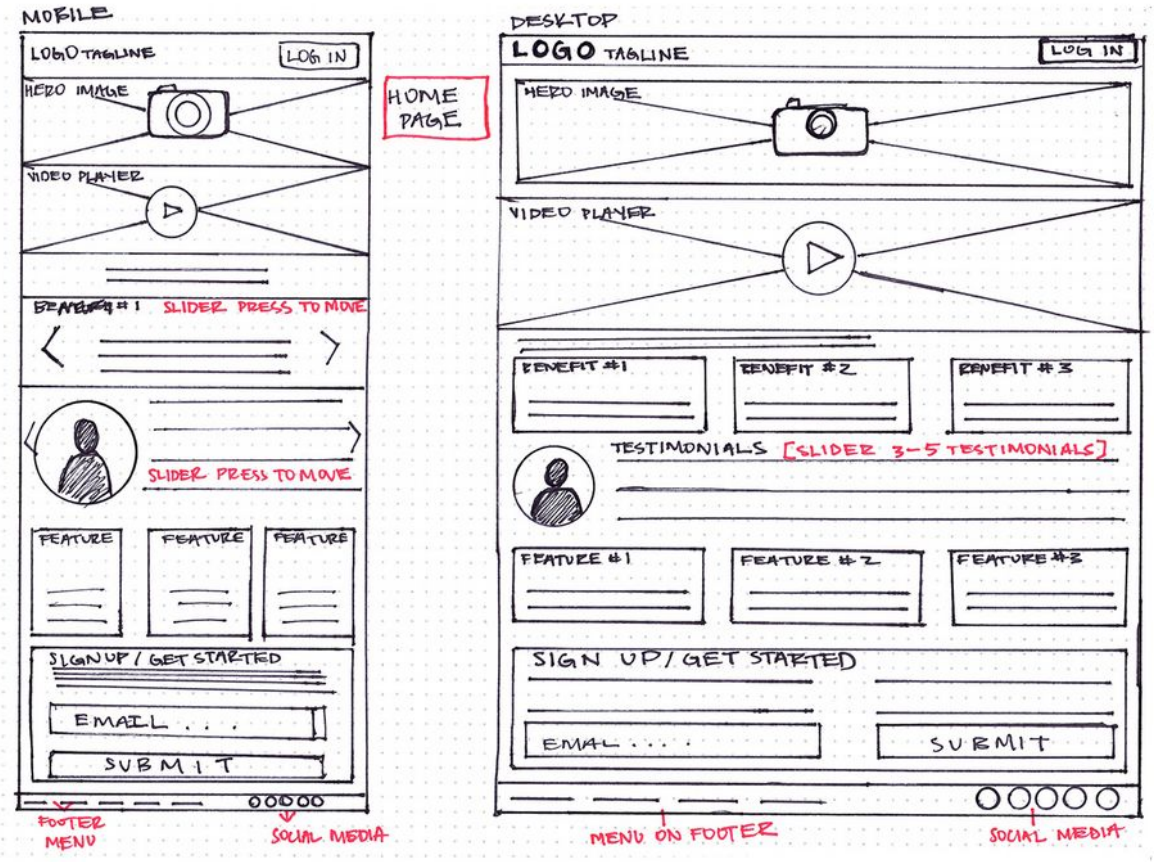
A *storyboard* is a sequence (or graph) of wireframes. The concept arose with video, as in movies or TV, for mapping out the shots to be filmed. Often used for the “fake it” prototypes of design sprints

Sometimes called *mockup*, but sometimes mockup means a lightly-functional rapid prototype that does something but not much, rather than just pictures


YouTube
YouTube
Search
Search
Browse
Upload
Create Account
Sign In
Matrix - The pill
johndoe
9 videos
Subscribe
Suggestions
How The Matrix Should Have Ended
by otheruser
4,860,784 views
Neo vs Morpheus
by otheruser
163,624 views
Matrix is real
by otheruser
2 views
No it's not
by otheruser
2 views
Yeah, it is!
by otheruser
2 views
Where am I??!
by otheruser
0 views
Like
Add to
Share
Embed
534,151
Uploaded by johndoe on Sep 25, 2007
1,241 likes, 24 dislikes
Morpheus shows Neo two pills: a blue and a red one. If Neo chooses the
Show more
All Comments (1,838)
Respond to this video...
youtube is the matrix!!!! I'm trapped!!!! AHHHH!!!
matrixuser 7 hours ago


EDITION: INTERNATIONAL
CNN
Home Video World U.S. Africa Asia Europe Latin America Middle East Business World Sport Entertainment Tech Travel iReport
Egypt military vows democracy after Mubarak's fall
How will Obama be judged on Egypt?
Discover Tokyo one secret at a time
Other news
HIGHLIGHTS
Editor's Choice
Business »
World Sport »
Daily Snapshot
CNN TV »
Environment »
Opinion »
Arts and culture »
Quickvote
Will the revolution that started in Tunisia end in Egypt?
Travel »
Technology »
World »
Hot topics »
1 Egypt
2 Nelson Mandela
3 Davos 2011
4 Piers Morgan
5 Tunisia
6 Kate Middleton
7 WikiLeaks
8 Gabrielle Giffords
9 Academy Awards
10 China


edition.cnn.com
Volkside
Wireframe created with Wirify by Volkside on 13 Feb 2011
Check out Naview for IA work and Wirify export preview
edition.cnn.com via wirify



There are various free wireframing tools online, e.g., wireframe.cc, and other drawing tools that can be used for drawing wireframes, e.g., diagrams.net (aka draw.io)

“Assignment T1: Preliminary Project Proposal” due October 27
https://courseworks2.columbia.edu/courses/104335/assignments/486922

Each team proposes their own project, within constraints:

- Must impose authenticated login.
- Must be demoable online (e.g., zoom or discord).
- Must store some application data persistently (database or key-value store).
- Must use some publicly available API beyond those that “come with” the platform. It does not need to be a REST API.

Describe Minimal Viable Product (MVP) both in prose - a few paragraphs - and via 3-5 “user stories”.

< label >: As a < type of user >, I want < some goal > so that < some reason >.

My conditions of satisfaction are < list of common cases and special cases that must work >.

Describe acceptance testing plan that covers these cases.

List the tech you plan to use. If different members of the team plan to use different tools, please explain.

Two sample team projects from a previous offering of this course are linked below. All submitted assignments as well as the code are included in each repository. These projects had three iterations because the team project ran the entire duration of the course, there was no individual project.

Code Phoenix - https://github.com/s4chin/coms_4156

Space Panthers - https://github.com/wixyFun/openSpace