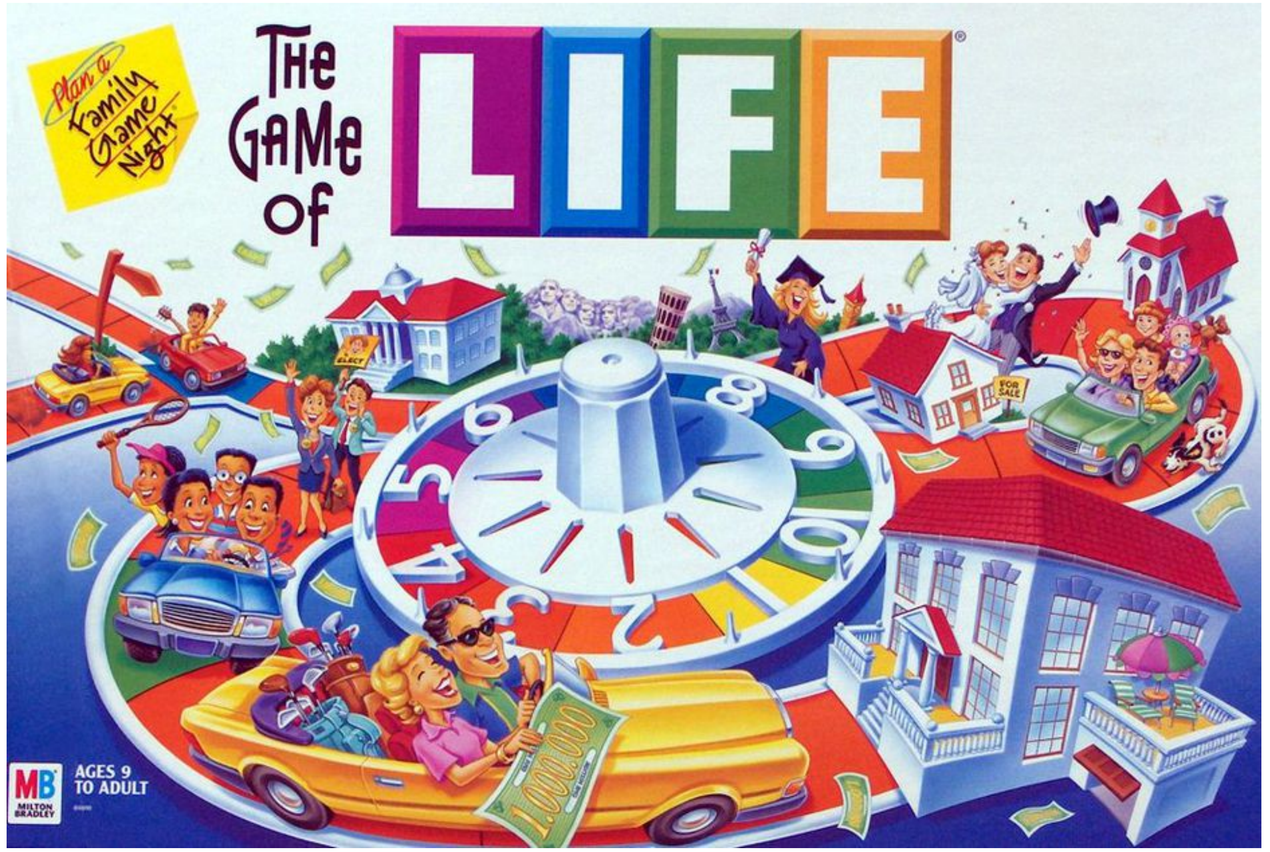
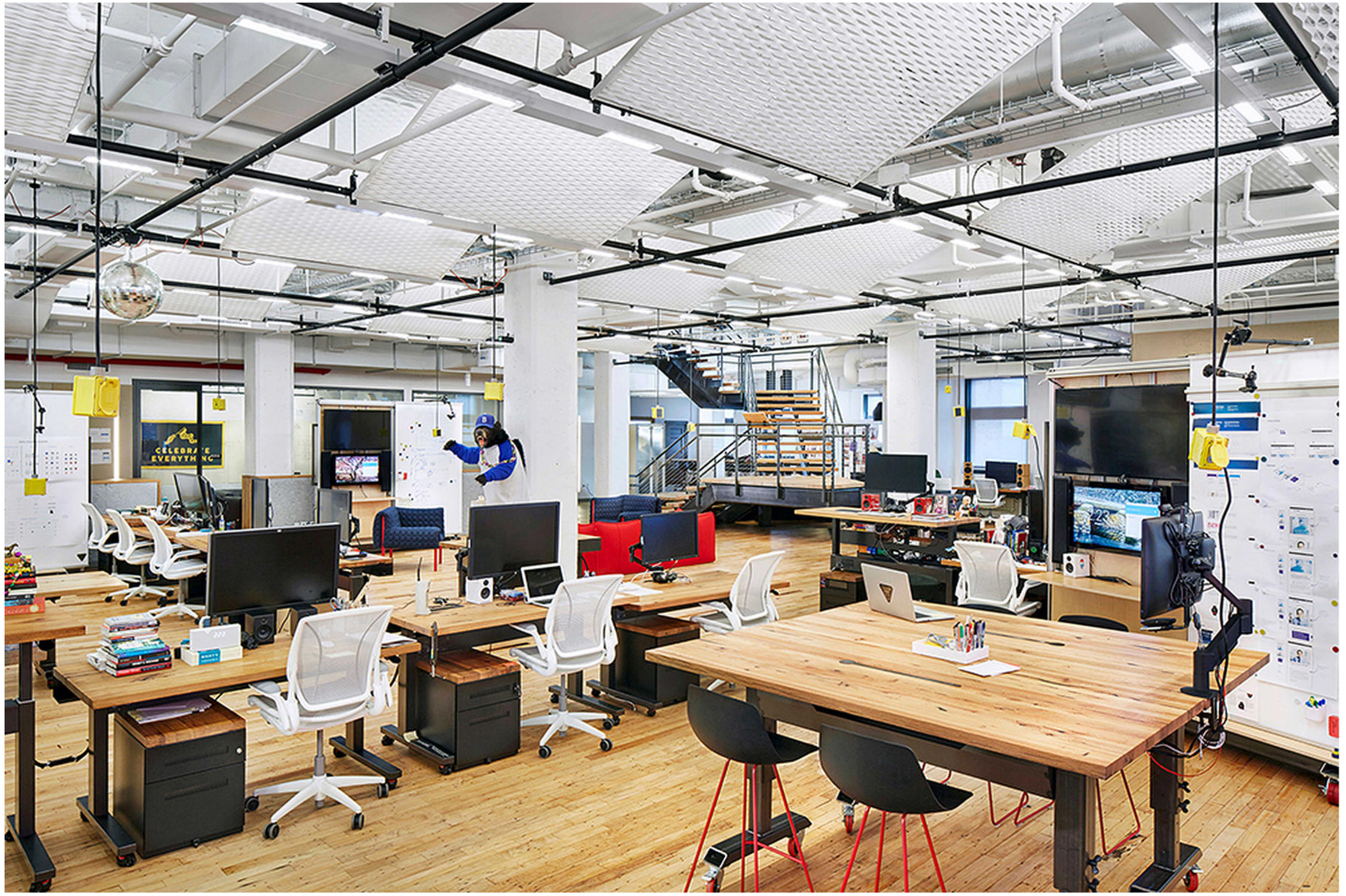
GRACE SHOPPER

Development Team Simulation Game

Part I: What?


Plan a Family Game Night
The GAME of LIFE
AGES 9 TO ADULT
MB
MILTON BRADLEY
FOR SALE



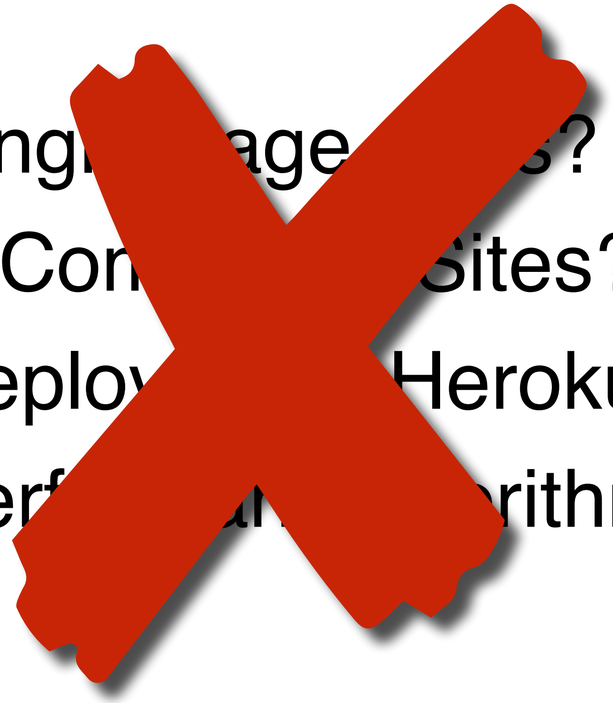
GRACE SHOPPER IS ABOUT...

…Sing[illegible]age[illegible]s?

…E-Com[illegible]Sites?

…Deplo[illegible]Heroku?

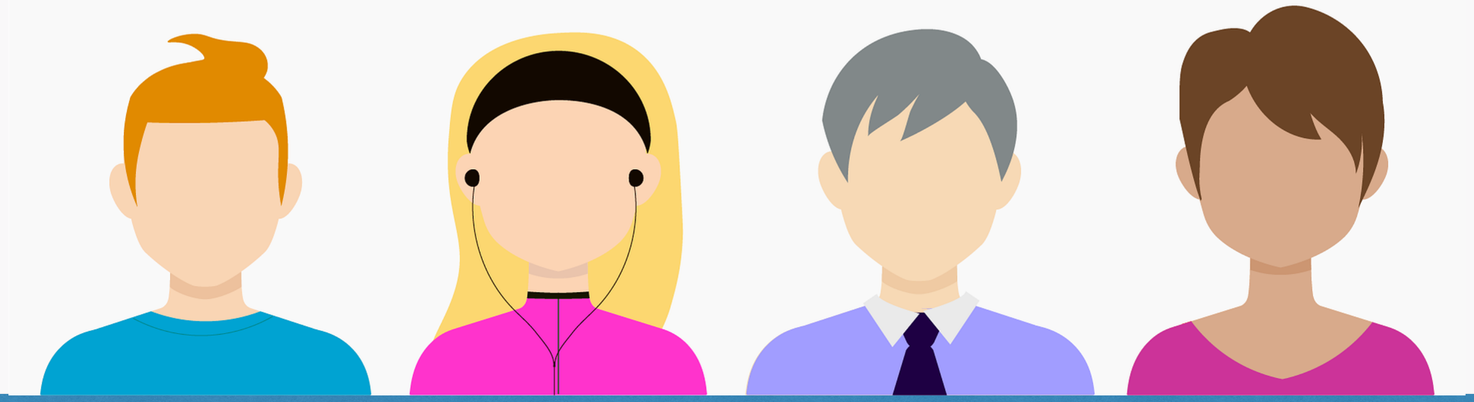
…Perf[illegible]rithms?



1. TEAM PROJECT MANAGEMENT

PRIMARY GOAL: DEVELOP AS A TEAM

- **Communication and planning**
  - Design mockups and system architecting
  - Agile methodologies and iterative design
  - Daily standup and problem solving
  - Task assignment and issue tracking
- **Collaborative development**
  - Longer-term and larger-scale
  - Pair programming
  - Code reviews
  - Merge conflict resolution



GRACE SHOPPER SUCCESS = LEARN TEAM-BASED DEVELOPMENT TECHNIQUES

2. TOOLS & TECH

SECONDARY GOAL: TOOLING & MASTERY

- **Dev Ops**
  - Deployment on Heroku
  - Travis CI (Continuous Integration / Testing)
- **Project Management**
  - GitHub Features
  - Advanced Git
  - Project Boards (GitHub Projects)

- **Full-stack Applications**
  - Node, Express, Sequelize, React, Redux practice & synthesis
  - Schema design
  - Testing patterns
  - Visual design
  - Security

GRACE SHOPPER IS AN EDUCATIONAL EXERCISE (NOT A PORTFOLIO PIECE)

3. THE PRODUCT

TERTIARY GOAL: THE PRODUCT

- Many "business requirements" and features are about to be presented to you
- These are all *motivation* for mastering team-based development, new tools, and junior phase material
- Many of these features are good to know or even important in the industry, but they are *stretch goals*

PRODUCT FEATURES ARE GOOD...
BUT NOT YOUR "GRADE"

PRIORITY REVIEW

1) TEAM-BASED DEV SKILLS

2) MASTER TOOLS & TECH

3) PRODUCT FEATURES


OK WE GET IT!
SO WHAT ARE THE
REQUIREMENTS?

GRACE SHOPPER IS AN E-COMMERCE SITE

WHAT KIND OF PRODUCT(S)?

(your choice)

…keep it clean

TIER 1: MVP SHOPPING EXPERIENCE

- Two roles: guests (not signed in) and users (signed in)
- Deployed (online!)
- See all products
- Add to cart / edit cart
- Checkout (submit order)
- Backend data validations
- Rudiments of security

TIER 2: E-COMMERCE ESSENTIALS

- Continuous Integration / Continuous Development (CI/CD)
- Really nice UI/UX design
  - Front-end data validations
- Carts persist in database (can load cart on a new browser)
- Order history (users can see theirs, including historic cost)
- User Profile (viewable, users can edit contact info / other data)
- Accept payment (Stripe, Paypal/Venmo/Braintree, Bitcoin)
- Admin page (edit products, manage users)
- OAuth integration

TIER 3: EXTRA FEATURES & FLAIR

- **Inventory tracking and management**
- **Persistent guest cart (front-end storage)**
  - Merge guest / user carts upon login
- **Accessibility (a11y)**
  - A11y checklist
  - screen reader friendly
  - keyboard navigable
  - colorblind-friendly
- **Email confirmation**

TIER 3: EXTRA FEATURES & FLAIR (CONT.)

- Error/loading states in UI
- Product Filters & Pagination
- Toast notifications for events
- Featured products
- Promo codes
- Wishlists
- Social media integrations

TIER 4: S TIER

- **Internationalization (i18n)**
- **Localization (l10n)**
- **Visualization dashboard**
- **Recommendation engine**
- **Multi-tenancy**
  - White labeling
- **Surprise us**

SURELY THIS IS IMPOSSIBLE!?

10 CALENDAR DAYS
7 CLASS DAYS
LECTURES, ETC.

PRIORITY REVIEW

1) TEAM-BASED DEV SKILLS

2) MASTER TOOLS & TECH

3) PRODUCT FEATURES

Tier 1 is Passing!

Part II: How?

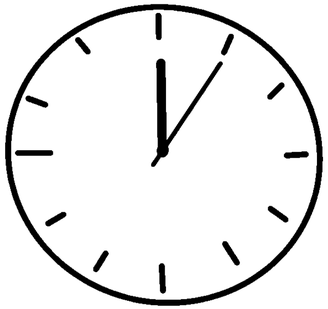
Logistics

- **Teams of ~4**
- **Starts today**
- **Code reviews (2: this Wed. & next Mon.)**
- **“Presentation to management” next Wednesday**
  - Instructors, Fellows, classmates
- **Rotating team “roles”**

HOW TO WORK TOGETHER

without killing each other

Why is it hard to work in a group?



STAND UP

A "stand up" a day, keeps the conflicts away.

- **Goals**:
  - Stay updated on each other's progress
  - Organize efficiently (no duplicated work!)
  - Remove blocks
- **Length**: 15 minutes or less
- **Round Robin style**:
  - What did I do yesterday?
  - What am I working on today?
  - Do I have any blockers?
    - A *blocker* is something outside of your control that is preventing progress on the thing you are working on today.

Stand Up Example

- **Noelle:** “Yesterday, I refactored most of the product and orders routes, but there’s still a bit left to do. Today, I’m going to finish those routes. I don’t have any blockers.”

- **Natalie**: “Yesterday, I finished scaffolding the ‘orders’ page with dummy data. Today, I want to make the UI interact with the actual database data, but I need Noelle’s updates to be merged before I can continue.”

- **Noelle**: “Good to know. The ‘order’ changes are ready, it’s just the ‘products’ routes that need a bit of tweaking. I can make a pull request right after this stand up for review so you’re not blocked.”

ROLES

Fellow == Project Manager

- **A simulation of “agile”**
- **Lead stand up every morning**
  - Discuss yesterday’s roles, assign today’s roles
  - Assign daily goal
- **Customize support for your specific team**

Taskmaster

- **Keep track of todos and issues**
  - Bite-size
  - Use a project management tool (i.e. GitHub Projects)
- **Assign pairs and tasks each day**
- **Facilitate communication and minimize duplicated work**
  - Make sure everyone knows what everyone else is doing

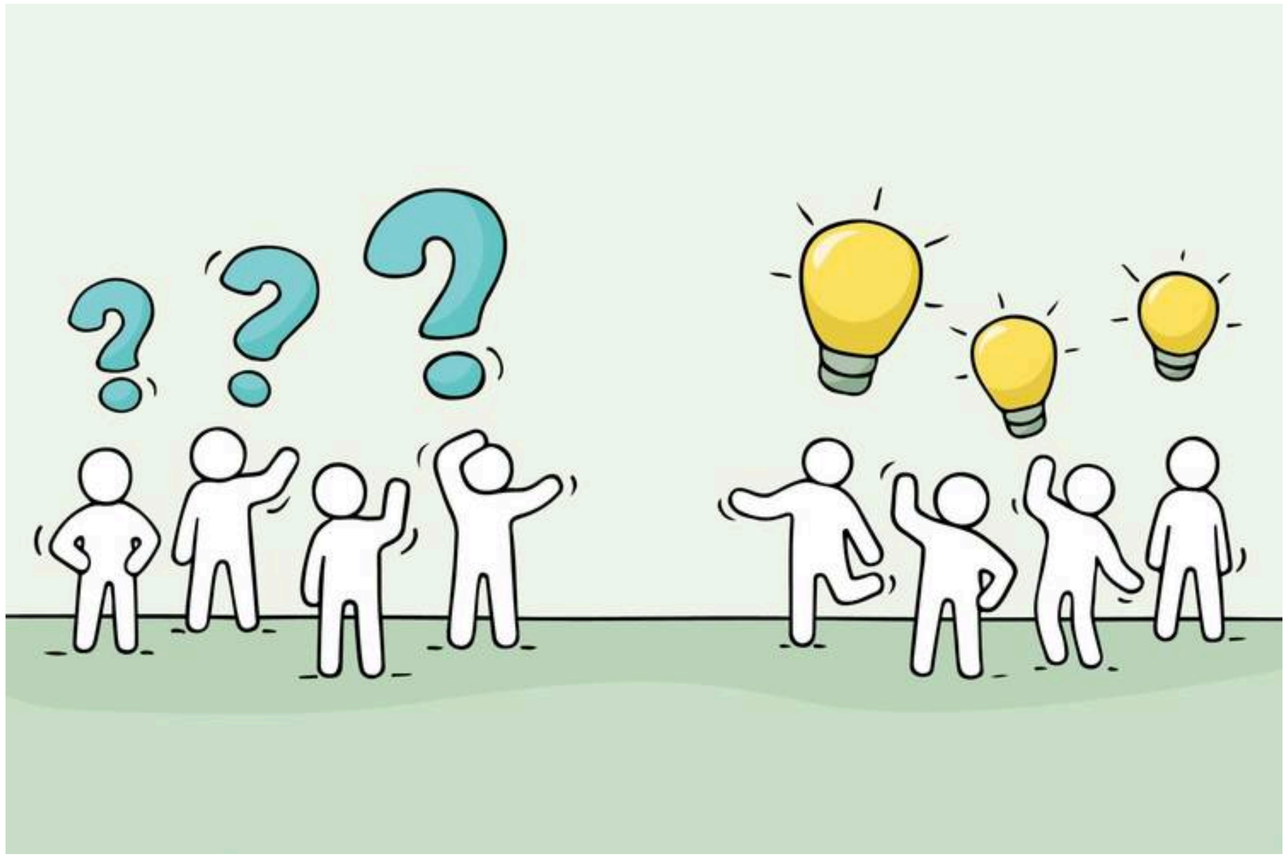
Gitmaster

- **Make sure everyone is commit-ing responsibly**
  - Note: Your commit messages are easily viewable.
- **Make sure people are making pull requests**
- **Make sure people are reviewing pull requests in a timely manner**
  - Do ***not*** blindly merge without actually reviewing and testing the features that branch implements!

Testmaster

- Make sure tests are being written
- You do NOT write all of the tests

NORMS



normalize

- Establish a set of rules that the group agrees to abide by
- These rules act as a source of truth for how the group behaves, manages work, and deals with conflict
- When in doubt: “What are our norms?”

starting norms

1. Share the mic
2. Don't interrupt - raise your hand
3. Share frustrations in a constructive matter
4. Write a GitHub issue
5. Ask for help
6. Commit *frequently*
7. Deploy *early*
8. Choose a code style and stick to it
9. Keep each other updated
10. Disagreements many, arguments few :)

set your norms

- **Each group:**
  - Take our list of starting norms and add to them based on the way your group wants to work

- **Answer the questions on the accompanying document**

TIPS

- Aim for uninterrupted focus
- Do not specialize (i.e. split on feature NOT tech)
- Rotate: pair with *each* member of your team
- Small, frequent PRs
- Protect the repo's *master* branch (e.g. review PRs)
- Don't just give your opinion, ASK for other thoughts
- Challenge yourself!

SENIOR PHASE HELP TICKETS

- **Can take longer and be more complicated**
- **Keep using the queue**
- **Do your research FIRST**
- **Beware when using technologies that aren’t part of FSA’s core curriculum**

TEAMS!

Team Apples
Helen / Noelle

Christina Kang
Roy Tessler
John Cho
-

Team Bananas
Austin / Natalie

William Xu
Henry Chong
Ming Kien Juan
Remi Mendoza

Team Carrots
Rushil / Noelle

Jasen Chan
Robert Costello
Depak Borhara
-

Team Doughnuts
Fergal / Natalie

Leo Yoshinaga
Dhroov Wadhwa
Robert Fox
Eric Yang

Team Eggs
Fergal / Noelle

Adam Mak
Tina Fun
Adel Jabbar
-

Team Fennel
Henri / Natalie

Manny Kandilas
Desi Shunturova
Erik Jenson
Peter Crosta

Team Grapes
Austin / Noelle

Kenneth Chen
Ziv Karmi
Adam Vare
Marvin Huang

Team Hericots Verts
Rushil / Natalie

Joseph Polanco
Manami Ueda
James Shen
Slava Adnavorau