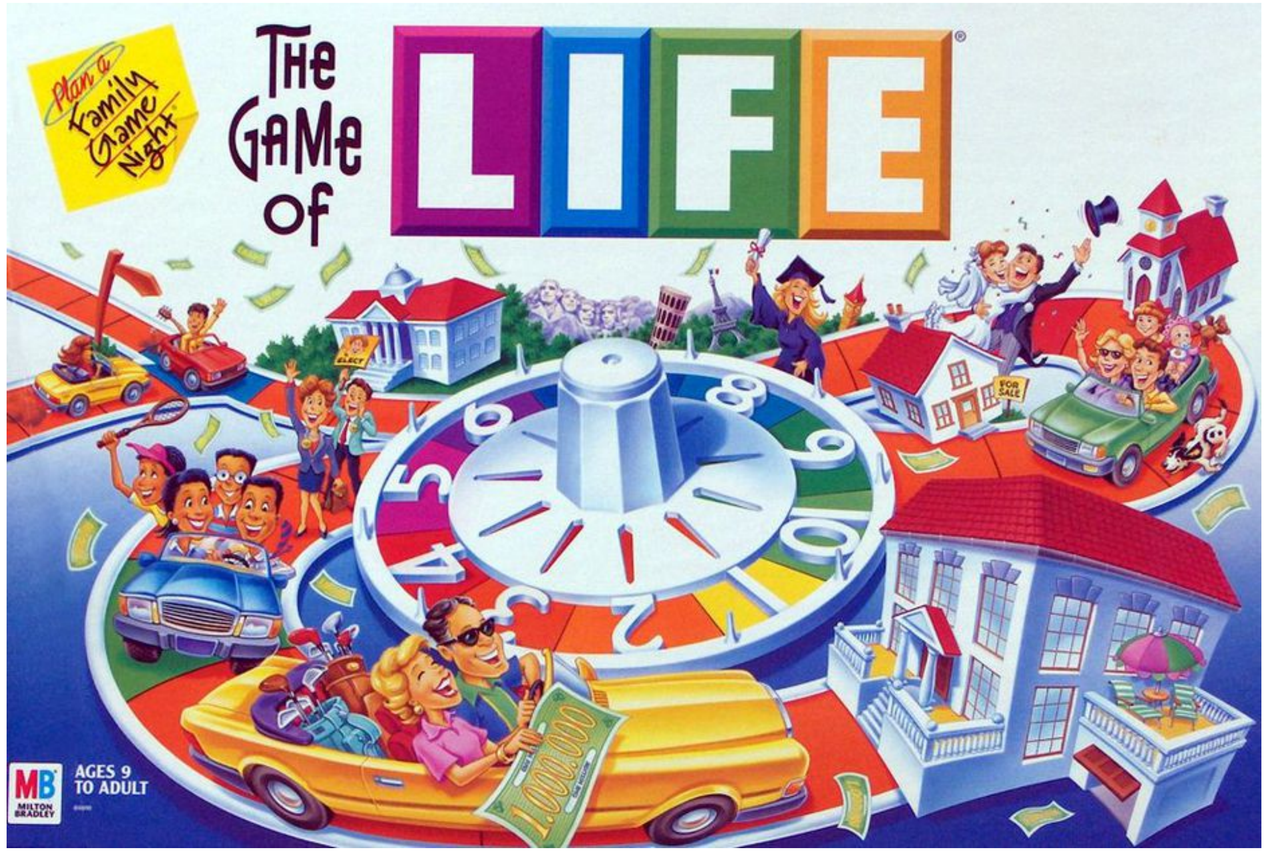
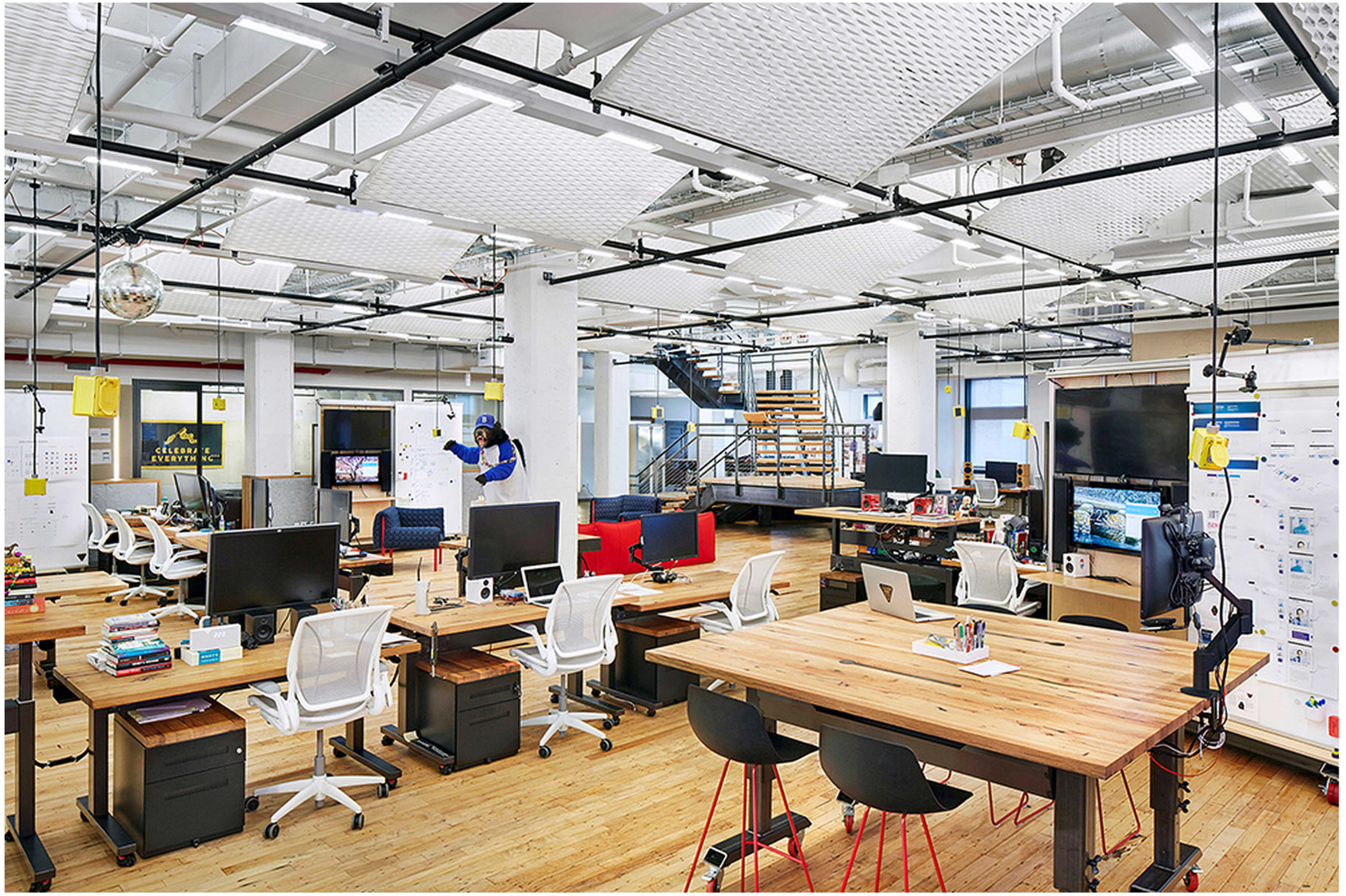
GRACE SHOPPER

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...

…Single Page Apps?

…E-Commerce Sites?

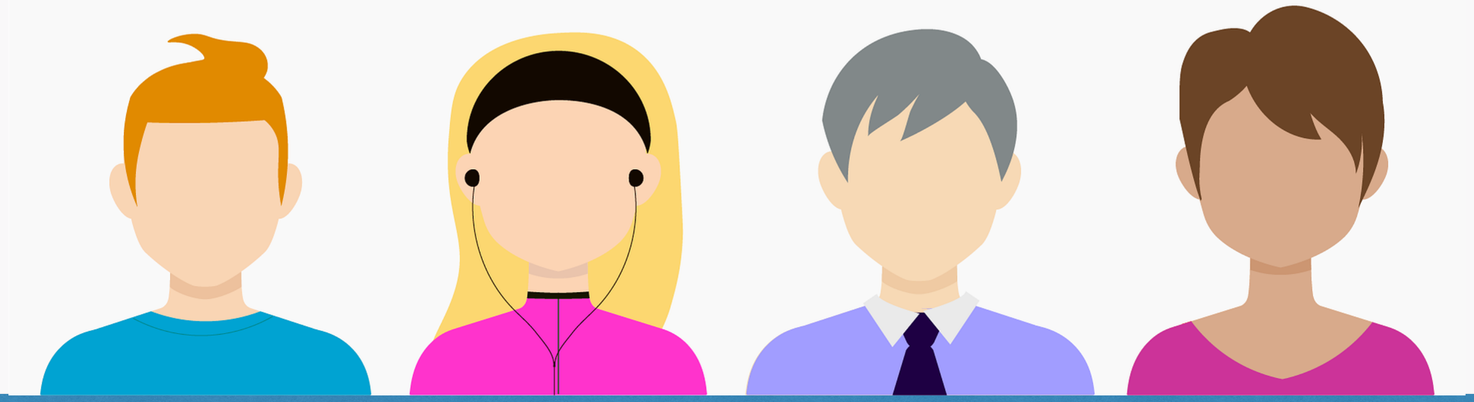
…Deploying to Heroku?

…Performant Algorithms?

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

- Logged in user & guest accounts supported
  - Persistent cart on refresh for guests, permanent cart for users
- Deployed (online!)
- See all products
- Add to cart / edit cart
- Checkout (submit order)
- Backend data validations
- Rudiments of security
- Basic styling
- All components of vertical slices have tests
  - One test per member (that they can present on 🙂)

TIER 2: E-COMMERCE ESSENTIALS

- Continuous Integration / Continuous Development (CI/CD)
- Really nice UI/UX design
  - Front-end data validations
- 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 &C.

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, two other Grace Shopper teams
- Rotating team “roles”

STAND UP

A stand up a day keeps the merge conflicts away

- Goal:
  - stay updated on each other's progress
  - organize efficiently (no double-work!)
  - remove blockers
- Short (~15 minutes or less)
- Each person in round robin style:
  - What did I do yesterday?
  - What am I doing today?
  - Do I have any blockers?
- Blocker: something outside of your control that is preventing progress on the thing you're doing today

- Alice: “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.

- Bob: “Yesterday, I finished scaffolding the Orders page with dummy data. Today, I want to make the UI interact with the live data, but my blocker is that I need Alice’s backend changes to be merged before I do that.

- Alice: “Good to know - the orders changes are ready, it’s just the product routes that need fixing. I can make a pull request right after this, if you can review it.

ROLES

Fellow === Project Manager

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

Taskmaster

- Keep track of todos / issues
  - Bite-size
  - In project management tool (GitHub Projects)
- Assign pairs / tasks each day
- Facilitate communication and minimize double work
  - Make sure everybody knows what everybody else is doing

Gitmaster

- Make sure everybody's `commit`-ing responsibly
- Make sure people are making pull requests
- Make sure people are reviewing pull requests

Testmaster

- Make sure tests are being written
- YOU DO NOT WRITE ALL THE TESTS

GIT'ING STARTED

- Create a Github organization
- Add team, “project manager,” and instructors
  - Give everyone owner privileges
- Create a repo
- Push an initial commit
- Protect the master branch (Settings > Branches)
- Create a project
  - Add at least three columns: To Do, In Progress, and Done

WRITING ISSUES

User Stories

- As a <persona>, I want to <do something> so that <reason>
- Software should be designed based on what the users of that software want
- Stories !== implementation details

Implementation Detail

- How you will actually fulfill a user story
- Split each story into specific, bite-size tasks (implementation detail)
  - Should always serve a user story
- Write a ticket for each implementation detail
  - User story !== implementation detail
  - Associate each ticket with a GitHub issue

More on this later today

Bad Issue

```
We need a route to serve up all the products
```

Better Issue

```
As a visitor, I want to see all of the products,
so that I can choose one to get more details or
add one to my cart right away
```

Better-ish Issue

An Express route on the backend (GET `/api/products`) should serve up the name, price, quantity, and imageUrl of all the products from our Postgres database. No special access control is needed, since any visitor should be able to receive this information.

Good Issue!

Story: As a visitor, I want to see all of the products, so that I can choose one to get more details or add one to my cart right away

Implementation: an Express route on the backend (GET `/api/products`) should serve up the name, price, quantity, and imageUrl of all the products from our Postgres database. No special access control is needed, since any visitor should be able to receive this information.

Feature Branches

- Never work directly on master!
- Always work from another branch (“feature” branch)
- Connect the branch to GitHub Project issues with `#issueNumber`
  - `git checkout -b my-feature-#issue`

Pull Requests

- Push your branch up to the remote
  - `` `git push origin my-feature-#issue` ``
- Go to your remote repo on Github
  - Make a pull request (it should prompt you)
- Someone else reviews your pull request
- If they request changes, make them locally and then push remotely again

Merging

- Complete requested changes
- Merge the PR on Github
- All team members can now update their local master branch
  - `git checkout master`
  - `git pull origin master`
- If you are currently working on a feature, you should update it with the latest changes from master
  - `git checkout my-other-feature`
  - `git merge master`