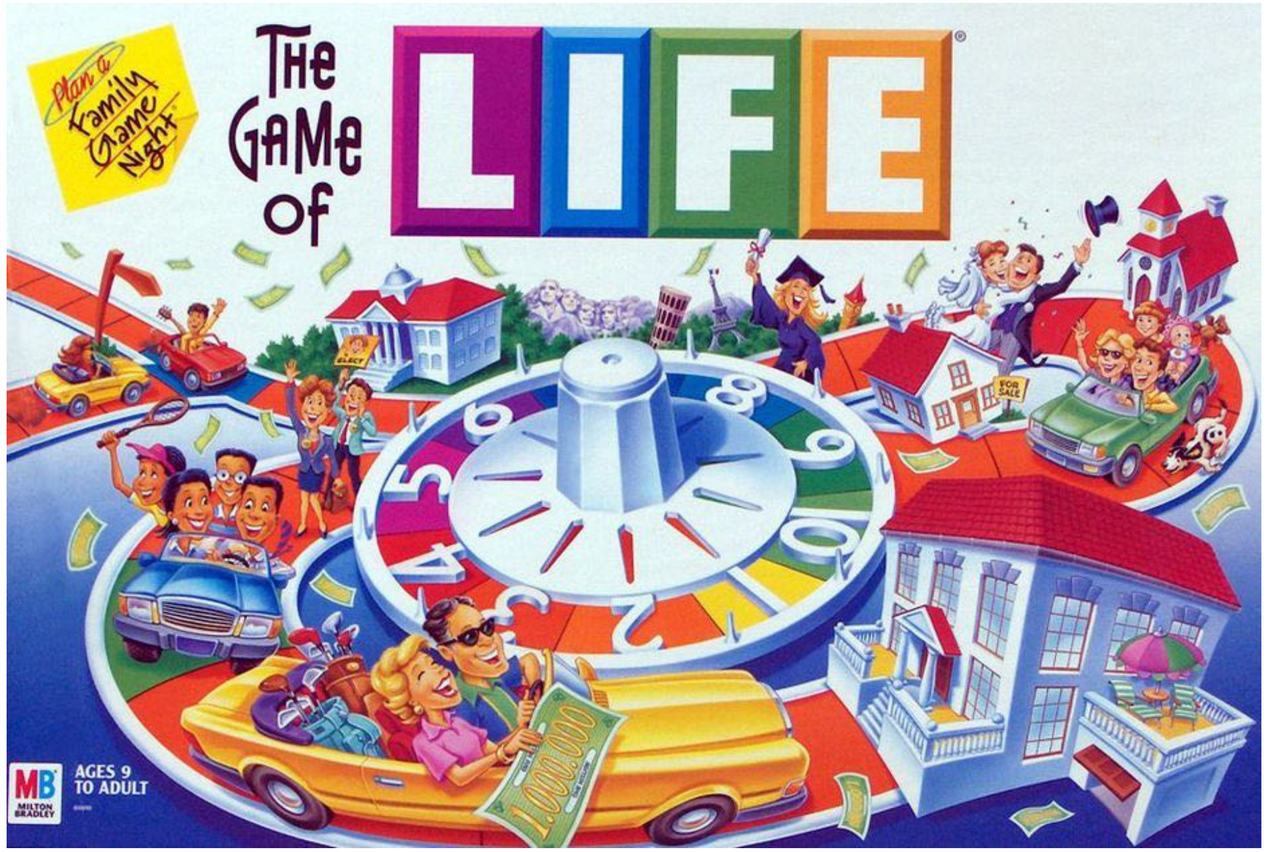
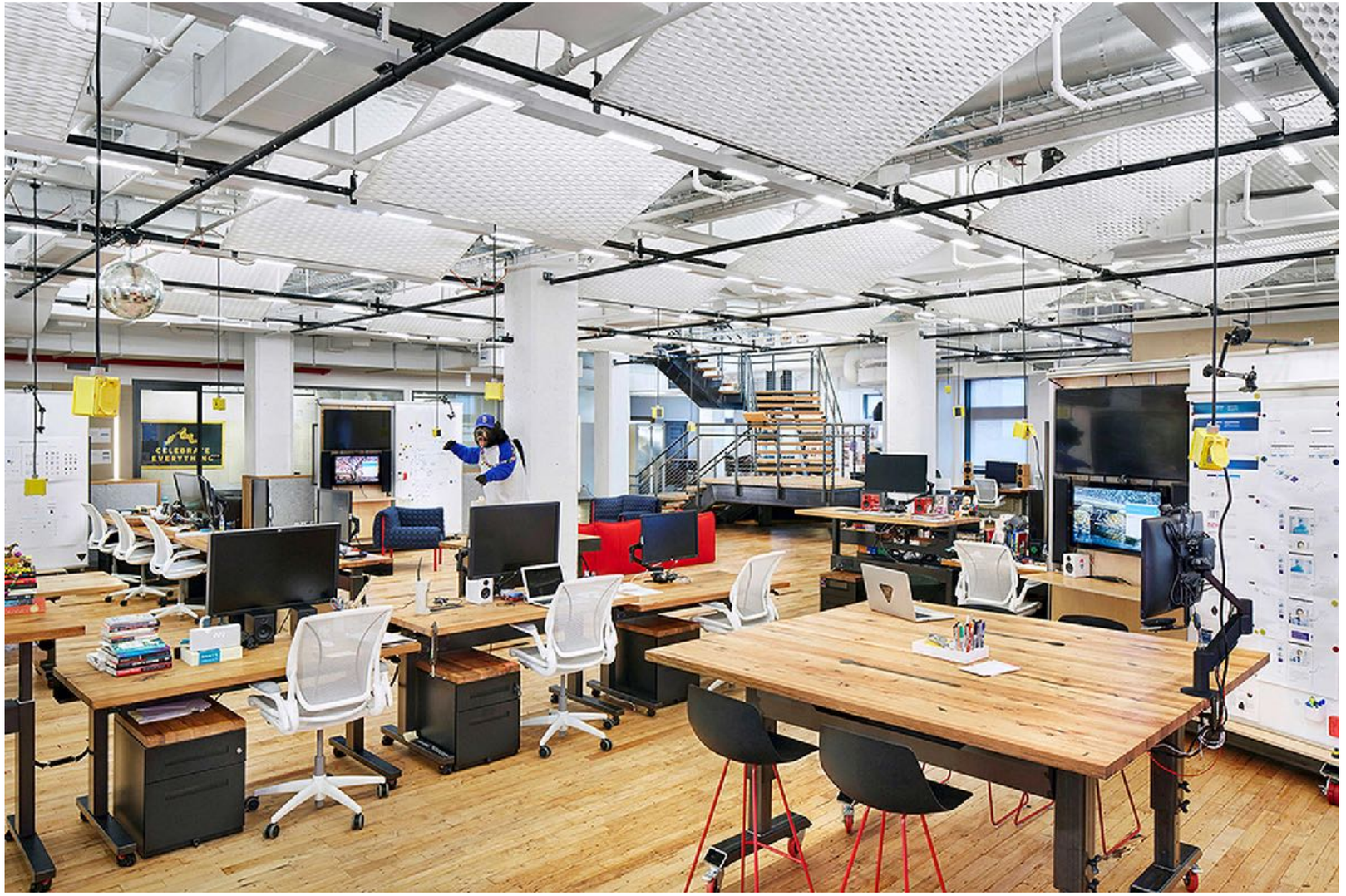
GRACE SHOPPER


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



VICTORY CONDITION

Users can visit a deployed production site, browse a list of products, add them to a cart, check out, and make a payment with a credit card using Stripe, either as guests or as logged-in users

(Reference the workshop for Tiers)

https://learn.fullstackacademy.com/workshop/5ece807ae423f6000461d41e/content/5ece826fe423f6000461d4d1/text

LOSS CONDITIONS

LOSS CONDITION #1

- **sensitive data exposed or vulnerable to simple attacks**
  - data:
    - emails
    - passwords
    - payment info
    - order history
    - API keys
  - simple attacks:
    - navigating to specific API routes in the browser
    - requests sent directly to the server w/ Postman or curl

LOSS CONDITION #2

- **lack of teamwork**
  - unreviewed PRs merged to *dev* branch
  - not all team members committed code to production site at every layer of the stack
  - skipping daily standup

LOSS CONDITION #3

- no tests

BONUS POINTS

- nice design / UI / UX
- extra features
- continuous integration
- clean, readable code
- reusable components

BONUS POINTS FOR TEAMWORK

- **pair program**
- **use and maintain Github projects board**
  - lots of small, isolated issues connected to branches and PRs
  - cleary-defined swim lanes
  - kept up to date (you might get audited!)
- **dedicated Slack channel integrated w/ Github repo**
- **clean commit history**
  - lots of small commits focused on a single problem
  - semantic commit messages

WHAT KIND OF PRODUCT?

- your choice
  - (keep it clean)

Grace Shopper Takeaways

- Education (synthesis)
- Collaboration
- Planning / Agile
- Fun!

Education

- **Synthesis of everything so far**
- **New stuff**
  - Teamwork
  - Git workflow
  - Authentication
  - Web security
  - Deploying production web app

Don’t worry about this at first

Logistics

- Teams of ~4
- Starts today
- Code reviews (4) - at start of each week (first one Wed)
- “Presentation to management” (*us!*) next Saturday (1/2 way through)

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 (10 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.

Instructional Associate === Project Manager

- **A simulation of “agile”**
- **Lead standup every day**
  - Discuss yesterday’s roles, assign today’s roles
- ***Not* your personal debugger**
- **Lean on your team (and your inner nerd) for debugging**

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 details)**
  - Should always serve a user story
- **Write an issue for each implementation detail**
  - User story !== implementation detail

Bad Issue

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

Good user story (not an 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
```

Good implementation detail (not tied to a user story)

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.

DOING WORK

Feature Branches

- **Never work directly on main OR dev!**
- **Always work from another branch (“feature” branch)**
- **Connect the branch to Github projects issues with issue number**
  - `git checkout dev`
  - `git pull`
  - `git checkout -b <branchName>-#<issueNumber>`

Commit Messages

- **Semantic**
  - `git commit -m “fix(redirects): Add redirects file so users can go directly to route”`
- **Closes https://docs.github.com/en/enterprise/2.16/user/github/managing-your-work-on-github/closing-issues-using-keywords**
  - `git commit -m “feat(login): Add ability to log in with username and password. Closes #5”`

MERGING TOGETHER

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)
  - if you have conflicts when you make a PR you will need to merge dev locally and fix the conflicts locally before pushing again
- **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
- Delete the branch

STAGE DEPLOYMENT

- **Deploy your dev branch to Heroku (stage)**
- **Practice Continuous Integration**
  - Integrate code into dev several times **EACH DAY**
  - Test locally as you develop
  - Test in the stage environment multiple times each day

PRODUCTION

- When code is merged to `main`, we auto-deploy to the “production” version of your app.
- This happens weekly, possibly more frequent.
- Simulates a regular “release schedule”

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 dev (e.g. review PRs)
- Don’t just give your opinion, ASK for other thoughts
- Challenge yourself!

ASKING FOR HELP

- Do your research FIRST
- Ask questions in the big channel if it would be helpful for all
- Ask in your team channel if it’s specific to your team
- Beware when using technologies that aren’t part of FSA’s core curriculum

Teams!

Cyber Dynasty

Xavier Loera Flores
Rubal Kaur
Jimmy Dang
Aaron Rosati

Robo Hackers

Daniel Golub
Raydan Bala
Amar Pal Singh
Edwin Mendez

Ping Intelligence

Gregory Mah
Shane Tyler
Emilio Batan
Ryan Tran

Joyful Nodes

Jason Lammers
Juan Soto
Carlos Escamilla

Tech Tycoons

Janessa Ortiz
Brandon Fillpot
Kevin Kepner

EVALUATION

- **Feature completeness / site usability**
- **Code quality and general best practices (e.g. REST, DRY)**
- **Effective use of Git (e.g. semantic commits, PR use)**
- **Effective use of project management tool**
  - i.e. moving stories associated to issues across swim-lanes
- **Existence and quality of unit tests**
- **Schema design**
- **Security**
- **UX / Design / visual appeal**

GRACE SHOPPER KOANS

- Pair Program for at least 1 hour every day, unless you are short on time. Then, Pair Program for 3 hours.
- Embrace Repetition (…but Don’t Repeat Yourself)
- The deepest mysteries of our universe can all be looked up on Stack Overflow, we just haven’t discovered the proper search query.

UP NEXT

- **Get set up!**
  - Meet with your team (video on! Talk to each other!)
  - ** Read the workshop **
  - (Optional): Change your team name
  - Edit your project board
  - Update `README.md` with a Contribution Guide
  - Organize the work you have to do with a Github project

GOALS: GIT'ING STARTED (FOR TONIGHT)

- **Establish Norms / Contract**
  - Choose a Team Name (if you don’t like ours…)
  - Go through the Team Contract together https://docs.google.com/document/d/1qBZeAX6gzYPyxdIVHNoP2HTF1ZUUf3seU0cIiBJTT1c/edit?usp=sharing
  - Go through Establishing Norms together: https://docs.google.com/document/d/1YOpRdI4d_jPBCnt2pO1rYibInkpFhBrITixKGxtBIk0/edit?usp=sharing

- **Each member**
  - Clone your repo (NOT fork)
  - Create a new branch as a test, and make a commit on that branch
  - Merge dev into that branch `git merge dev`
  - Push up the feature branch to gitHub
  - Create a PR to be reviewed by teammates
- **Edit the readme**
  - Instructions https://gist.github.com/khumphrey/3d0cc35799c3341becfec198e0b44002
- **Edit your project board**
  - Must keep at least four columns: *To Do, In Progress, Needs Review*, and *Done*