

Portfolios for Programmers

Anuhya Ghorakavi, Isabella Tromba, Patricia Saylor, Ryan Cheu

Overview

Purpose and Goals (Patricia)

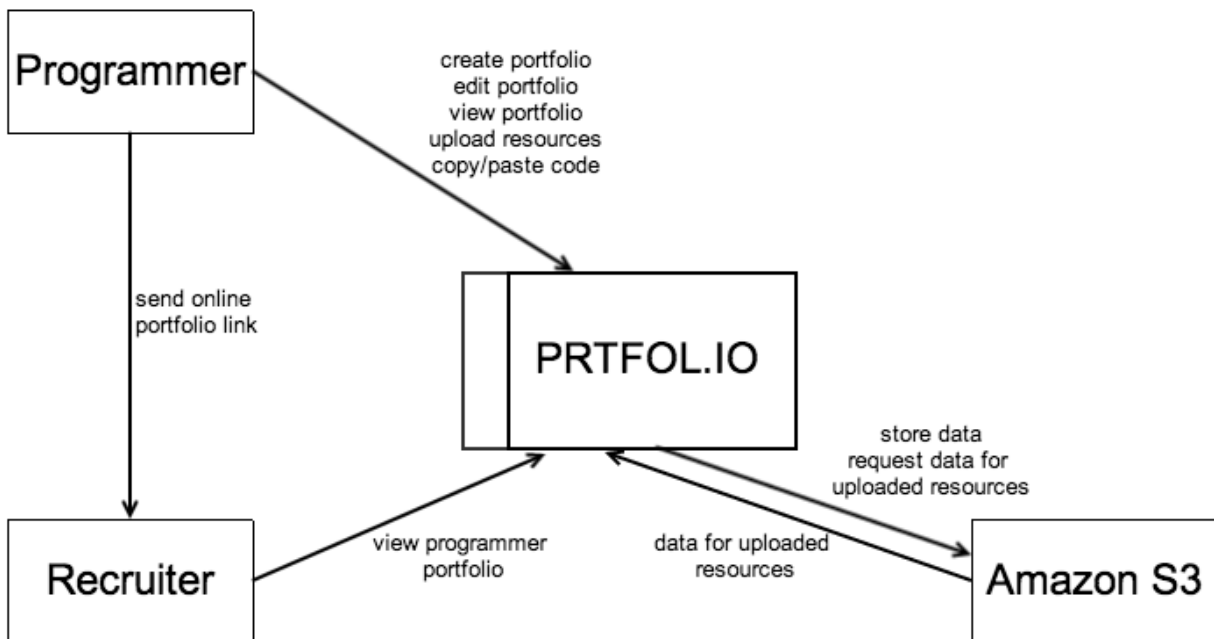
Portfl.io is a web application that simplifies the process of making and hosting an online programming portfolio.

Typically when a programmer applies for a job, she must provide a resume and, optionally, a code sample and her personal website to give the recruiter a sense of her skills and her personality. However, resumes are not very effective at accurately conveying one's skills, especially with the trend of resume padding in which simple tasks may get spun as feats of technical prowess. This is where code samples come in—they provide real substance demonstrating one's actual skill level. The personal website, on the other hand, usually conveys one's personality and highlights the projects of which they are most proud. But designing, implementing, and hosting a personal webpage is a lot of nontrivial work, especially for many programmers who aren't web designers.

Prtfol.io deals only in tangible products from your projects, like code, pictures, and videos, not unsubstantiated claims on a resume—no padding, no fluff, just you and the projects you care about. Here you can create your own online programming portfolio, including code snippets and pictures or videos of your projects in action, from screenshots to inputs/outputs, to demonstrations. Prtfol.io helps you style and lay out your portfolio—no web experience required! When you're done, link your recruiters and interviewers to your portfolio to give them a comprehensive introduction to your skills and interests.

GitHub is a great site for displaying your full coding project online, but there's little room to explain the details of the project or your role in it, and potential employers don't want to sift through your whole life story in code. With our mix of code and media, you can pick which projects are most important to you and then highlight the interesting parts with snippets and visuals, giving something for both technical and non-technical viewers. Other solutions, such as LinkedIn or coderbits.com, suffer from resume padding and fluff—in the first, your grandma can endorse you for a programming language she's never heard of; in the latter, your code is hidden away on GitHub in favor of aggregations and “points” that have an unknown correlation with reality (please see the Appendix for illustrative visuals).

Context Diagram (Anuhya)



Concepts

Key Concepts (Patricia)

Portfolio A user's *portfolio* is a collection of projects she selects to highlight her programming skills. The portfolio should be a small set of projects that collectively demonstrate the depth and breadth of her knowledge. All portfolios are public and can be shared with just the URL.

Project A user's *project* is a programming project written in one or more programming languages that the user has worked on directly and should demonstrate some aspect of the user's programming skills. The display for each project is customizable by the user with a variety of templates and project resources. All of a user's projects are part of her portfolio.

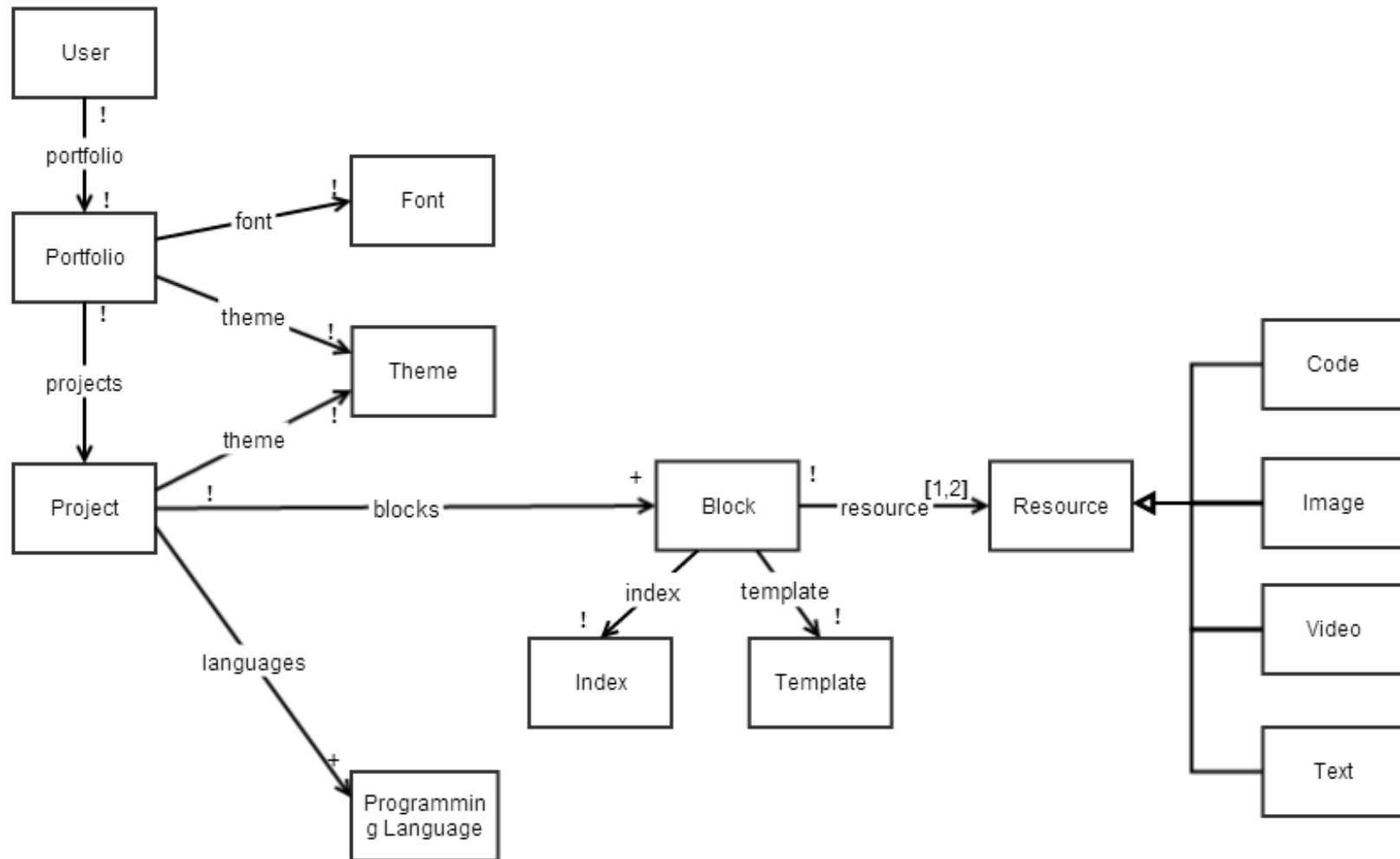
Resource A *resource* of a project is a description, image, video, or code snippet that provides a tangible sense of the scope and success of the project. Examples of relevant images include, but are not limited to, the following: screenshots of the final product (if it were a UI for example); inputs and outputs of an image-processing project; diagrams of the system architecture. Examples of relevant videos include the following: inputs and outputs of a video-processing project; a demonstration of how to use a Leap-motion-like interface built by the user.

Block A *block* is a displayable unit of 1 or 2 resources. The resources will be displayed according to the block's template.

Template A *template* specifies the layout of a block's resources when the block is displayed.

Theme A *theme* is a palette of colors used when displaying a portfolio or project.

Data Model (Patricia)



Behavior

Feature Descriptions (Isabella)

Showcase Your Skills Create our own online portfolio, complete with your most interesting projects. No web experience? Not a problem! We'll make your portfolio attractive and professional so you can send it to your recruiters and interviewers to highlight your skills through your previous projects.

Portfolio Customization We provide the framework, now you provide the details. Customize your portfolio with a selection of color schemes, fonts, and templates. Describe your projects with technical depth and visual clarity by adding code snippets, videos and images.

Multimedia Upload Upload your high quality videos and images for each project directly through our site. We'll handle storing and displaying them for you!

Language Filter Filter through a user's projects by a programming language of interest so you can hone in on examples of the skills you care about.

Security Concerns

Security Requirements (Patricia)

Prtfol.io has one two key security requirements:

1. Only the user herself may view or modify her account information, such as her email and password.
2. Only the owner of portfolio or project may modify it.

Both of these security requirements will be addressed with access control mechanisms that ensure the current user is authorized to access and/or modify the given information. We all have experience successfully implementing this sort of access control, so we are confident in our ability to address this requirement.

Threat Model (Patricia)

As an app geared towards displaying user's skills mainly for potential recruiters and interviewers, we recognize that attackers may be motivated to maliciously modify another user's portfolio or the projects, possibly to sabotage their page if they are competing for the same or similar positions. This attack may be attempted through our website or from the database directly, though we assume that Heroku mitigates the risk of attackers accessing our database directly.

Another attacker may try to steal user's private account information, including emails and passwords, either for the sake of having them or for the purpose of trying to hack into their accounts on other sites. For this reason, we store encrypted passwords.

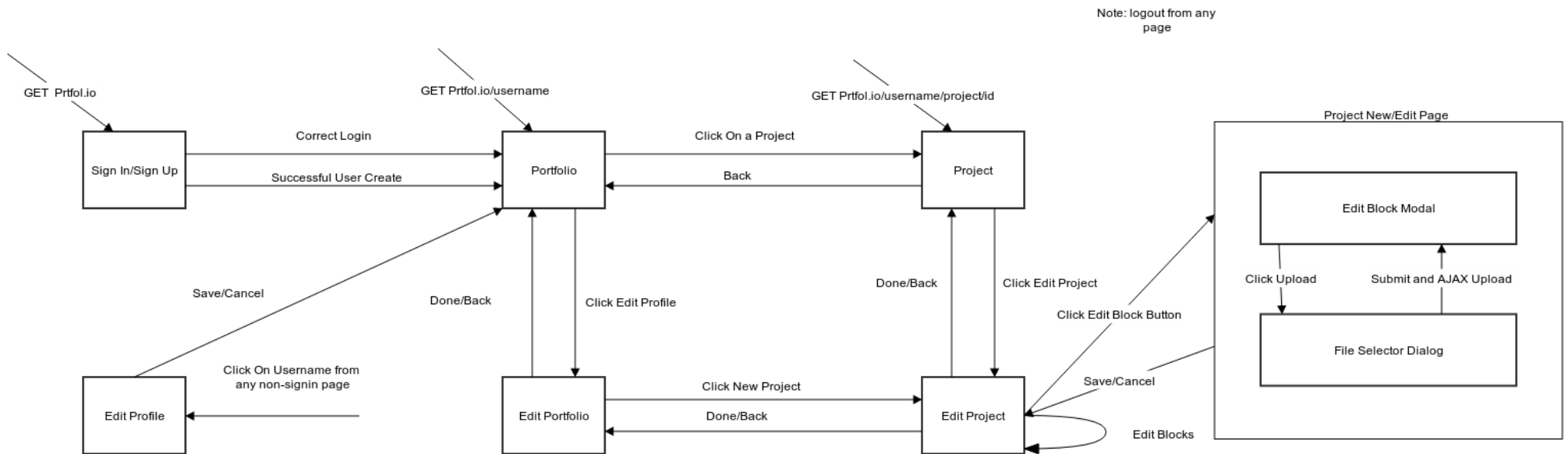
Besides this, most information on our site is publicly accessible and we don't expect it to draw interest from either the government or criminal organizations, unless they are looking for good hackers.

Standard Attacks and Mitigations (Isabella)

1. **Secure data transfer.** Data may be read over insecure channels, but all private user information (password/email/etc) must be protected.
Mitigation: Data will always travel on secure channel: https. All password data will be encrypted and authenticated according to IPsec (Internet Protocol Security). Rails takes care of this if we set the SSL checking so that all requests happen over https.
2. **Man-in-the-Middle Attacks**
Mitigation: Use SSL so that attackers cannot read the raw data or send us raw data.
3. **CSRF** (Cross-Site Request Forgery)
Mitigation: We need to make sure that sessions expire and we need to include security tokens in non-GET requests that check on the server side. If the security token does not match what was expected, the session will be reset. Rails protects against CSRF by default by including `protect_from_forgery` with: `:exception` in each new controller.
4. **Session Hijacking**
Mitigation: Session hijacking would allow an attacker to use a web application in someone else's name. In an insecure network this could be accomplished by sniffing the cookies being passed. In order to deal with this issue, we decided to always force an SSL connection. Also, we display prominently a log-out button so that users will be encouraged to log out of their sessions in the case that they leave a public terminal.
5. **Session Fixation**
Mitigation: Session fixation is when an attacker sets someone else's session id to an id that they set themselves. In order to deal with this issue, on successful login the session needs to be reset. This means that the fixed session id that the attacker created is no longer valid and they cannot use it to co-use the application. Rails takes care of this resetting and so does devise (which we may use for authentication).
6. **SQL injection attacks**
Mitigation: The main problems here include attackers trying to bypass authentication and trying to read or create/delete objects in the database. We do not need to worry about SQL injection attacks because we are using ActiveRecord to access/create objects in the database and ActiveRecord handles these potential injection attacks.
7. **XSS**
Mitigation: We will use `ActionController::Parameters` `require` and `permit` to restrict what parameters we accept in the controller.

User Interface (Ryan)

Interaction Diagram



UI Wireframes

Our UI Wireframes can be viewed online at <https://wireframe.cc/pro/p/4dca9f018> which allows you to click on links in the UI wireframe and have it take you to the linked page. We recommend viewing it in the online format, but if you prefer, we have also included the wireframes as a separate pdf.

Design Challenges (Patricia)

Programming Languages

Options: One feature we wanted to add is searching/filtering by programming language. This could be done in a number of ways, for example you could search through *users* by what programming languages they have used in projects, or you could search *projects* by what programming language they include. At a lower level in the data model, these correspond to having Programming Language connected to either User or Project, respectively. The first option for this feature would lend itself to searching through users trying to find someone with particular skills, but this does not align with our goal of helping a user create a portfolio that showcases herself and her skills. The latter aligns with this goal, since you can limit the scope of the feature to filter a particular user's projects (instead of all projects) by language, but it does not facilitate discovery of other people as much.

Solution: We decided to go with the second option, making Programming Language connected to a Project, so that we can filter a particular user's projects by what language they are written in. Having a feature that is aligned with our goals is more important than having a feature that is useful for something else outside the scope of our goals.

User Browsing

Options: Should a viewer, either user or non-user, be able to browse all users in some fashion, or should a viewer only be able to see one user's portfolio given a link to it? The first way, recruiters might get distracted when visiting an applicant's portfolio and start browsing other portfolios on the site, which steals attention from the person whom we are trying to showcase. However, this browsing feature could be a natural extension of our goals, promoting discovery of your portfolio by users and non-users (such as recruiters) alike which adds value for the users. The second option does not promote this serendipitous discovery, since you must have a link to someone's profile in order to see it, but it also ensures that when you send someone to view your portfolio, they will not be distracted by other users.

Solution: On a user's portfolio page, we won't put any links in the navbar to other portfolios or an index of users, but on the homepage we will allow searching for a user by name. This gives us the best of both worlds, since when a viewer ought to be focused on someone's portfolio, they will not see any links that may distract them, but a viewer visiting the home page can look up someone by name. This could be useful for recruiters who were referred people (typically either by name or resume) and want to learn more about them by browsing their online portfolio.

Scope of Customization

Options: Since we are trying to ease the process of creating a personal programming portfolio, we need to find a good balance between automaticity and customization. Some styling/layout should be handled automatically, otherwise the user might as well have just made their own website. Other things should be customizable so that the user can create a project page that appeals to their tastes and makes their project stand out. For example, not everyone wants their pictures in a carousel, so we should not force that on the user, but forcing everything to have padding or margins of a certain size will save them a lot of time.

This is such a broad design challenge—the most significant one of our project—since it impacts everything from the UI to the concepts to the data model, since anything the user can customize needs to be remembered somewhere. What formatting should the user have control over? Font, size, color, placement on the page of each resource? Should we provide a bunch of defaults for non-web developers, with a way for advanced web programmers to customize their portfolio almost arbitrarily? If we allow the user to control something like placement of resources, at what granularity do we allow that—can they put anything anywhere, or is it confined to predefined positions? Should the portfolio page and each project page be customizable, and to what extent?

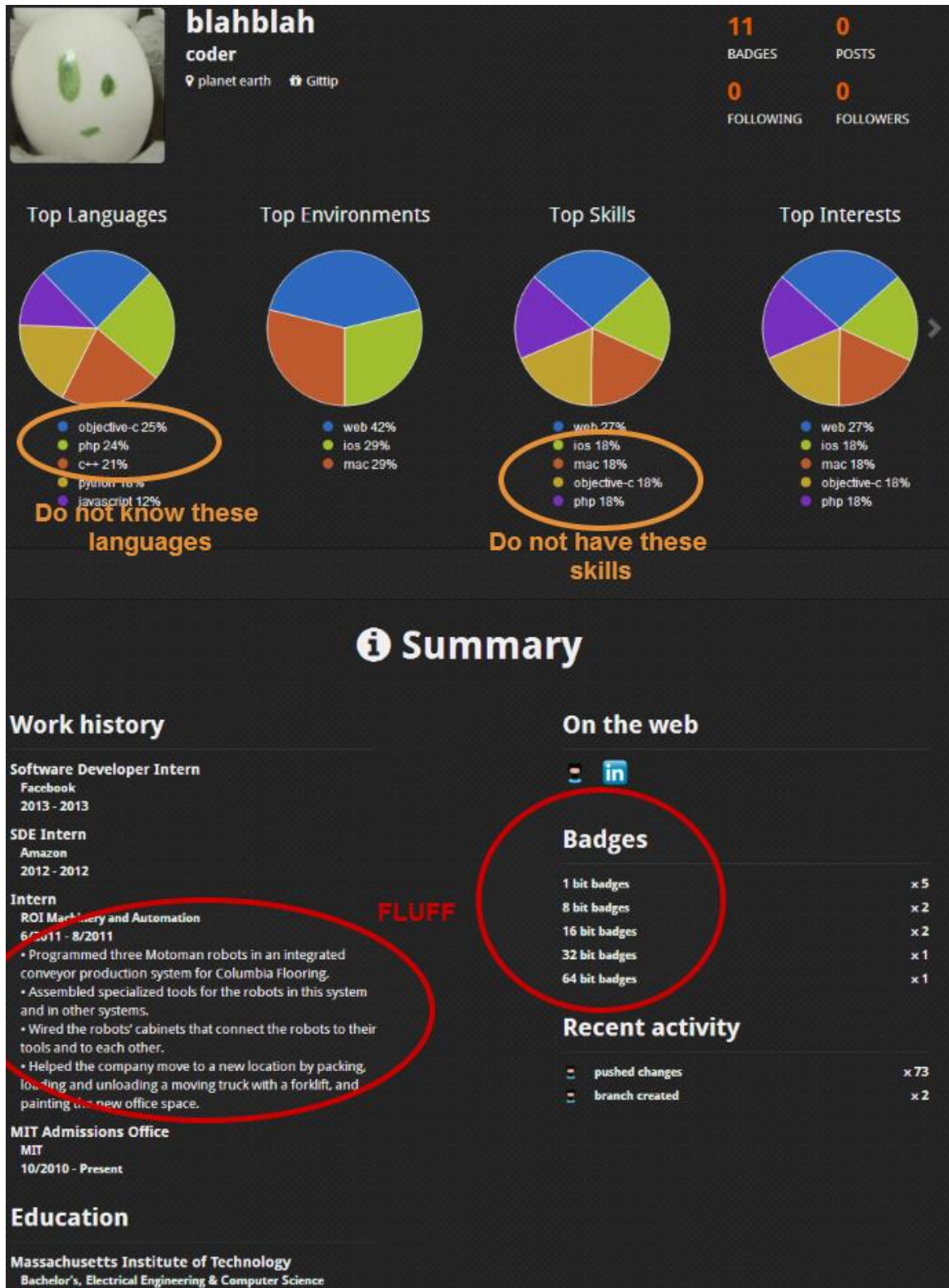
In this challenge there are many considerations at play: we don't want every portfolio or project to look the same; the customization we decide to allow should be simple for the user to do, and not too complex to implement; the customization should enable the user to make each project stand out in a format that is appropriate for the given project (e.g. a carousel is not right for everyone); the parts that are built-in need to be attractive and professional, and should significantly simplify the user's workload.

Solution: We devised a customization scheme that uses modular components and a small set of options to manage its complexity, making it both easier for users to use and for us to implement. Customization is mainly done on project pages, though users can select a color theme for their portfolio page, and can select one font to be used throughout the entire portfolio. The main modular component in this scheme is a block, which has either 1 or 2 resources (defined by the user) and a template (selected from a set of predefined templates) that specifies the layout of those resources within the block. The project page has a color theme and a number of blocks to display in order on the page.

Having these modular blocks allows for the user to customize the layout of their project page by selecting the template for each block and the relative ordering of the blocks; it also simplifies their task of creating the page since they won't be overwhelmed with options. Then we can make sure that our set of built-in color schemes, fonts, and templates are attractive and professional, which would be hard to do if they could be arbitrarily set by the user.

Appendix

A. Coderbits Summary




B. Coderbits Source Code

</> Source Code

Own

Code repositories the developer has created, owns, or administers. In some instances forked or cloned repositories will be included as well.

4 repositories




6.570ExampleApp

This is the source code for the demo Android App that will be built over the first 2 weeks of 6.570.

Cloned **10 months ago** on **GitHub**, **6.570ExampleApp** was last updated **10 months ago**

0 followers	java language	git vcs
cloned repository	94208 bytes	



CollabDesk

6.813 Group Project

Created **7 months ago** on **GitHub**, **CollabDesk** was last updated **6 months ago**

0 followers	javascript language	git vcs
original repository	2297856 bytes	

C. Coderbits Traits

Traits

The core traits the developer exhibits through actions taken on linked accounts.

3 traits

Teamwork

The measure of action taken in the cooperative effort to achieve common goals.

123
points

Influence

The measure of action taken in affecting people, things, or the course of events.

25
points

Productivity

The measure of action taken in achieving specified results.

1
point

Where did these arbitrary points come from?

How did I only earn 1 point for productivity?