Applican

Final Project Design

Team #2

BY:
HARRISON LUO
EMILIA PAZ
TRENTON POTTER
JUSTIN RODERMAN
ALEX SHADLEY

Project Synopsis

Applican matches vetted students with software engineering internships and full time positions through quick adaptive quizzes and an interview.

Project Description

Applican connects vetted students with technical internships. Internship searches are easier, and companies recruit faster, better, and cheaper.

Candidates take a technical screen followed by a one-on-one interview. Applican presents companies with student profiles, each with assessment results, company culture preferences, projects, past experience. This allows companies to have a more accurate and holistic view of a candidate.

Whether a 200 head recruiting department or a 10 head start-up, Applican gets your company more candidates alongside the information to quickly make decisions.

Project Milestones

Month	**Milestones**			
	Product	**Acquiring Students**	**Acquiring Companies**	**Administrative**
Nov	1. Landing page hosted 2. Ability to gather company interests page		1. Sales pitch outline 2. Schedule meetings with HR departments	1. Emails through domain name 2. Cap table/ vesting schedule / etc.
Dec	1. Gather company requirements 2. Plan and Implement Company Reqs 3. Student Evaluation rolled-out	1. Plan marketing campaign	1. Info/Sales meetings with companies 2. Follow-up meetings scheduled	1. Bank account 2. Tracking expenses 3. Freelance software engineer (?)
Jan	1. Company dashboard completed	1. Tabling 2. Execute marketing campaign	1. Contracts created and signed	1. Legal structure in place
Feb	1. Iterative improvements on product	1. Student interviews 2. Execute marketing campaign	1. Training/ support for company users	1. Plan funding goals 2. Incubator applications/ pitch competitions/ etc.

Mar	1. Bug fixes and maintenance 2. Plan next iteration of product	1. Schedule interviews with companies	1. Schedule interviews with candidates	1. Charge companies
Apr	1. Plan next dev cycle	1. Schedule Comp. Interviews 2. Gather feedback	1. Schedule Cand. interviews 2. Gather feedback	1. Charge companies

Project Budget

Resource	**Cost Per Unit**	**Units Monthly**	**Monthly Expense**	**Total Expense**
EC2 Compute	$0.0128 / hr	744	$9.52	$85.68
RDS	$0.0030/GBhr	~1200	$3.60	$32.40
Networking	~$4.30	1	$4.30	$38.70
Domain Hosting	$12.00	1	One Time	$12.00
Marketing	$50	10	500	$500
TOTAL				**$668.78**

Work Plan

- Business planning: Trenton Potter
- Company acquisition: Harry
- Student acquisition: Emilia
- Back end: Harry, Trenton
- Full stack: Alex
- Front end: Emilia, Justin

Preliminary Project Design

Software Stack

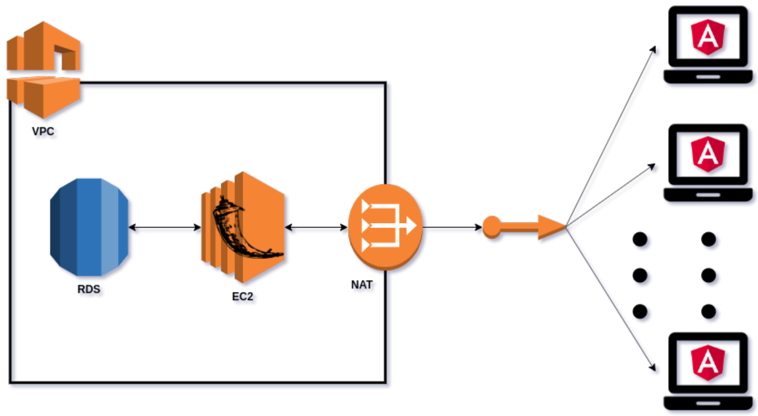
Applican's product is a web application, meaning it fundamentally consists of a web frontend, web API (backend), and a database. These three pieces are fairly interchangeable, and this shows in the course of our architecture decisions. Our API technology has been straightforward with little change over the course of the project. We chose Flask, a Python web framework that is designed to allow rapid prototyping. Flask achieves this by allowing more fluidity and customization to developers, as opposed to other frameworks such as Django and Ruby on Rails. In our experience, this has been a beneficial experience, since it allows us to select which modules we need and not worry about learning those we don't. For example, we use a Python library for JSON validation, to handle user logins, and to abstract database access.

The last type of library mentioned is referred to as an Object Relation Mapper (ORM). This class of library abstracts the process of interfacing with a database, making it possible to swap out the underlying database without changing any code. This has allowed us to develop with SQLite3 in development environments, and to use PostgreSQL in production. This is convenient because SQLite3 is a no-configuration database, meaning developers do not need to configure a database server in order to write new features. PostgreSQL however, is more performant, making it desirable in production settings.

Front end technologies have been something of a contentious point in our development group. Since we need to build a highly interactive web page, which will provide users with a timed quiz, animations, and a dynamic dashboard, a web framework is certainly advisable. Initially, we chose Angular as our framework, since one of our developers had some prior experience working in it. However as time went on, we realized two things: first, that we were spending much more time working on Applican than on anything in the past; and second, we hated working in Angular. These two facts led us to devalue the importance of prior experience, and consider changing framework. One of the major pitfalls of Angular is that for any simple task there exist at least several, possibly many, ways of accomplishing it. This results in a higher, more confusing learning curve, as developers need to learn multiple ways to read code.

Two frontrunners emerged: Elm and React. React is a very popular framework for frontend technology that uses vanilla javascript, albeit much more functionally than is typical, to render frontends. Elm by contrast is a functional language designed for web development. Given their extended learning curve, functional languages would normally have been a non-starter, but our team had 3 members who were somewhat acquainted with the language, and all of us had some experience with functional programming. However, React is a much more proven framework, with much higher industry adoption. This theoretically increases our ability to quickly onboard new developers, and makes finding documentation easier. Ultimately, we chose React for all new development, which entails the company dashboard. The student-facing application, which was written in Angular, would continue in Angular. Ideally a rewrite will take place at some point for this component, but realistically this is not in the near future.

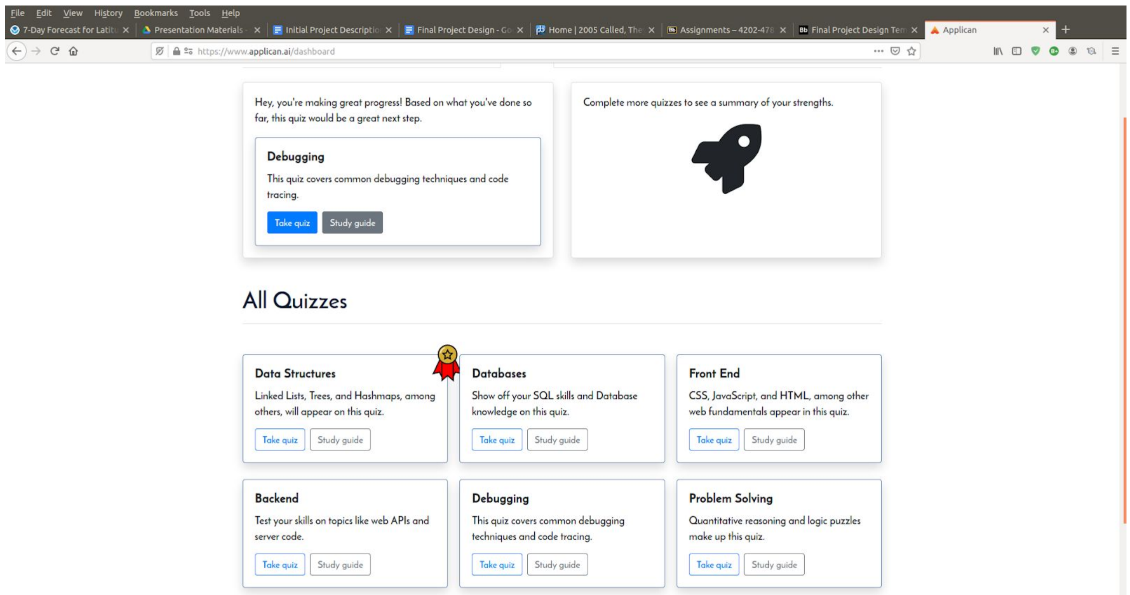
We use AWS as our hosting provider. One of our team members worked at AWS last year, so we had the advantage of being somewhat familiar with the system already. To improve portability, all of our application is containerized. We have 3 containers -- one container contains Nginx, which we use to route traffic. The Nginx container is networked to our Flask container, which serves the API, and the Express container, which serves our Angular application. Normal deployments of Angular applications do not require a web server, but we do because we are using Server Side Rendering (SSR). SSR renders parts of the page on the server before sending it to the client. This has two advantages: first, pages load faster, since fewer calls back to the server are required; and second, page crawlers (such as those used by search engines to index pages) have a much easier time crawling these, since many do not execute javascript on the page. ECS (EC2 Container Service) is the container orchestration service we use, which pulls containers from ECR (Elastic Container Repository). We have a simple deploy script which builds containers on the developer's machine and uploads them to ECR, then transitions the new containers into production.



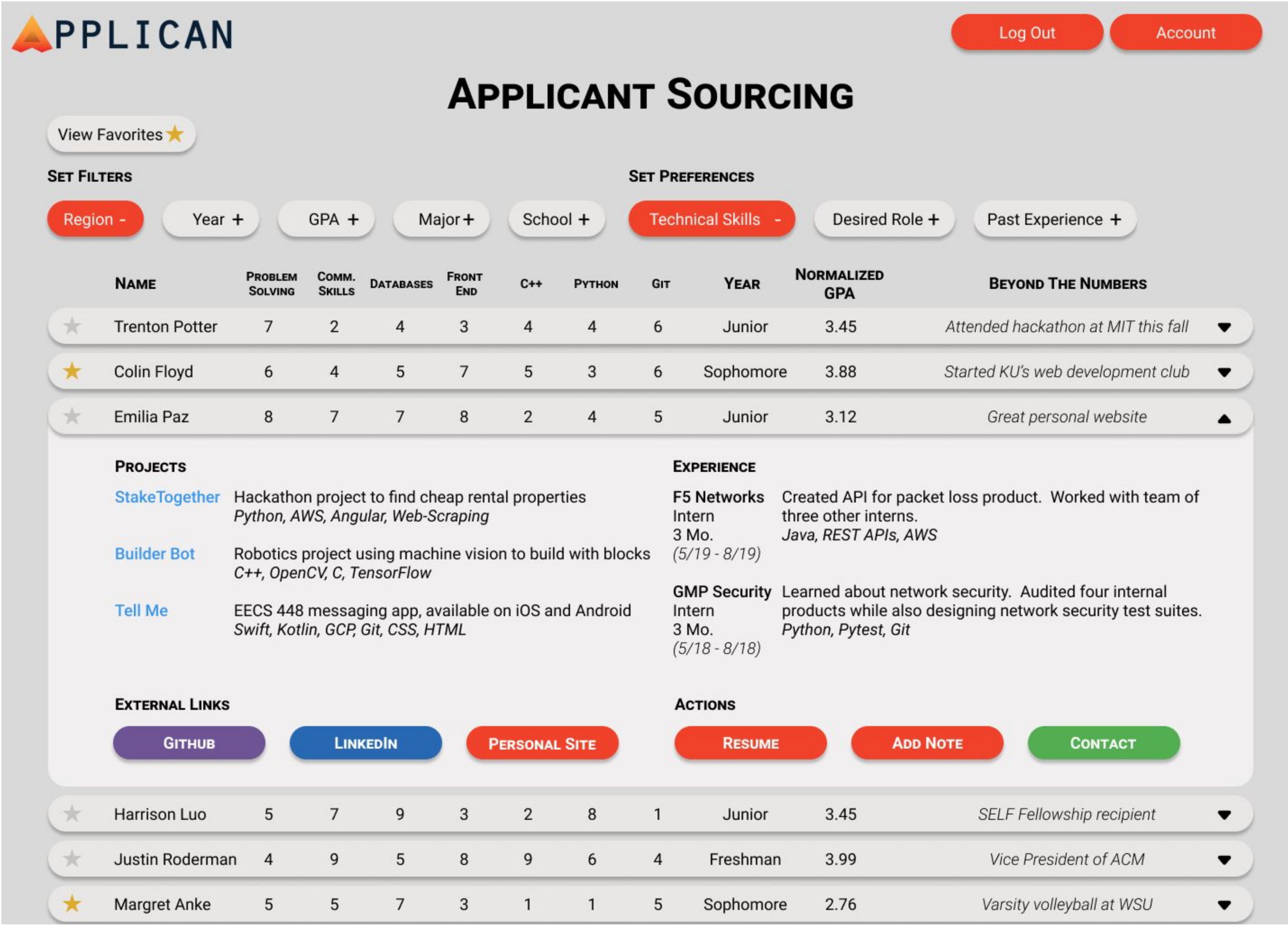
Product

Our initial concept was to have the user complete a quiz of about 45 minutes. The quiz would be multiple-choice and cover a variety of CS topics. These results would be stored in a database and could be scored later, producing category-specific scores that would allow us to gauge the candidate's skills in an automated fashion. After this process, an Applican team member would reach out to set up an interview with the candidate. Finally, the candidate would be entered into our matching system, where an algorithm would find compatible companies for the applicant. These companies would reach out to candidates they found promising, and the hiring process would continue from there.

We launched this product in the fall, and it became immediately apparent that this system would not be effective. A 45 minute quiz was incredibly off-putting to students, many of whom were skeptical of our platform to begin with. However, we needed to ask a good number of questions to get enough data to assess a candidate fairly. To solve this problem, we pivoted to a system of bite-size quizzes, built for specific categories of CS knowledge, such as Databases and Front End. The system is tied together with a dashboard that informs the applicant of their progress through the quizzes, as well as providing an interface for filling out information and uploading a resume.



The company dashboard is where companies will be able to see the skill profiles of potential candidates, and decide which to schedule interviews with. This dashboard will offer a variety of filtering and sorting options, for aspects such as Major, GPA, Year, and technical proficiencies as measured by our quizzes. Company users will also have the ability to favorite and comment on candidates. This data will be shared between company users, allowing them to collaborate more effectively.



Technical Constraints

Programming Language

While we have not explicitly set language restraints, by choosing to work with Flask and Angular technologies, we are siloed into using Python and Angular for these portions of the project. Further, we have explicitly decided to use Python 3.7 for our implementations due to current support and widespread acceptance.

Target Platform

We have decided to make our application and process web-first. This means we are designing with web in mind. This design constraint allows us to focus on the most relevant version of the product but could also hurt us in the long term should we need to transition into desktop application or mobile application development.

Hosting Service

Goto Intern is using AWS as our sole hosting provider. This allows us to have a cohesive deployment ecosystem alongside services meant to be used with one another. A potential downside to this decision is our reliance on a single provider not allowing us to shop around for the best deals. To remedy this in the future, our architecture could be refactored into another service or differently ported into a multi-deployment framework such as terraform.

Libraries and Frameworks

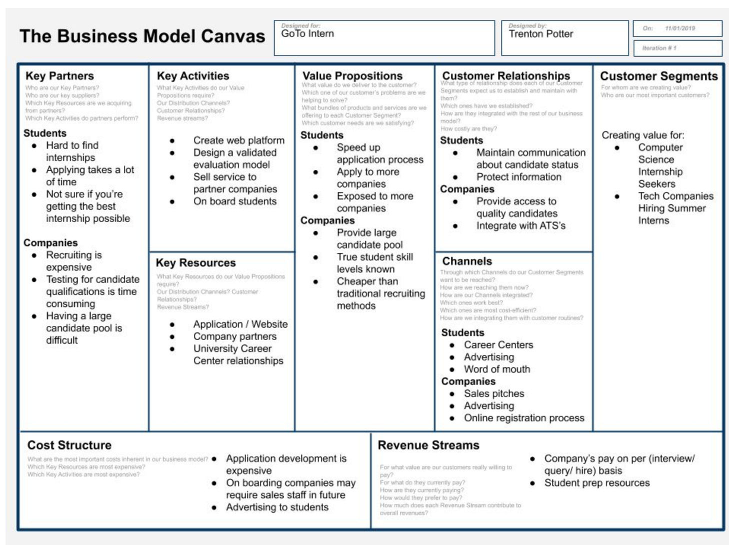
As mentioned briefly in the languages constraint section, we have decided to use Flask and Angular for our front and back ends respectively. This forces our reliance on the continued maintenance of both projects. Transitioning from these frameworks to another would require a significant refactoring process.

Business Constraints

Schedule

We have both self-imposed and academic constraints on our planned schedule. In order to provide a valuable tool before recruiting season is over, we plan to have our product fully launched before the spring career fair at KU. In addition, we are faced with constraints from EECS 581. These include presentations, reports, charts, and updates. More explicit listing of constraints on both sides can be found with our gantt chart above.

Budget



While not a seriously concerning constraint, we are limited to the support from the EECS department for our project. Since our only costs currently are associated with hosting the web server, we are not currently accruing large expenses. In the future as usage increases, we'll have to be wary of increases to our budget.

In addition, it is important to note that our budget could increase should we gain acceptance to the KU Catalyst program and receive funding for our project. This would alleviate any budget constraints we may face with hosting while also introducing additional costs relative to legal, marketing, and accounting expenses.

Personnel

During the course of this capstone, we are restricted to maintaining the team we started with. This disallows both additions and removals from our current team. While this is not currently an issue, it is important we address problems regarding personnel early and seriously to avoid problems.

Gantt Charts

TASK NAME	TA RT	N D	EAM MEMB ER	T CO MP LE TE	SEP	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY
Project definition													
Brainstorm project ideas	9/16	9/18	All	100%									
Career Fair field research	9/18	9/19	All	100%									
Design													
Design UI/UX	9/24	9/29	J/E	100%									
Front End basic functionality	9/30	10/18	J/E	100%									
Database table design	10/30	10/03	A/H	100%									
Back end basic functionality	10/30	10/15	A/H/T	100%									
EC2 and domain routing	10/30	11/1	T	100%									
MVP													
Front end: Home screen functionality	10/7	11/1	J/E	70%									
Front end: Quiz screen functionality	10/9	11/1	J/E	80%									
Back end: Endpoints	10/5	10/25	A/H/T	100%									
Front end design	10/28	11/15	J/E	10%									
Questions dataset	09/25	12/12	All	50%									

Test run	11/22	11/22	All	50%
Full product				
Front end implementation	01/21	01/31	J/E	20%
Back end implementation	01/7	01/7	J	20%
Talk with companies	01/21	02/8	All	0%
Test run #2	02/8	02/8	All	0%
Work with companies	02/8	5/1	All	0%

Ethical Issues

One potential ethical issue comes from our screening quiz. If our prescreen assessment is not well made and well tested, we may be unfairly judging some candidates' skills over others'. Some of our candidates may have come from a background where lower-level skills are not learned, but their programming skills may surpass those of another candidate who can answer exam questions but can't write a Hello, World script.

To mitigate this risk, we will be taking a two step process to the creation of our assessment. The first phase is the creation phase, which involves writing the questions. In the effort to make questions as fair to users as possible, we will be taking our heaviest inspiration from existing interview questions from large tech companies such as Microsoft, Google, and Facebook, as well as taking more inspiration from school and personal knowledge. This phase also includes the research of interview practices. In order to make sure our prescreen assessment is as close to a screening interview from a tech company as possible, we will be participating in heavy research to find out how screening interviews are accomplished in order to most effectively craft our quiz. After the creation of a large bank of potential questions, we will move onto phase two, which involves verification. The first step of verification involves testing our questions with a large group of students. We are looking for an even distribution of correct/incorrect answers, depending on the student's skill level. If a question is too hard or too easy, it will be tweaked until it is fair. Concurrently, we will be checking with recruiters and the KU Career Center to see if our questions accurately portray the screening process. College recruiters are our main focus here, as their opinion of our assessment will determine if their company trusts our system or not. With this two step process in place, we can mitigate the risk of an unbalanced assessment judging students unfairly.

A variety of ethical issues come from the sources of potential revenue this company can receive. We have three potential sources of revenue: charging companies for access to candidates, charging candidates for access to companies, and charging candidates with access to additional prep material.

We will start by discussing the primary potential source of revenue: our company partnerships. The way this will work is that a company will pay us a sum of money in order to gain access to our list of candidates and their scores. Differing amounts of money can provide companies with different lists of candidates. For example, a small company paying a small sum of money may not receive a large list of candidates, and the skill levels of the candidates received may not be as strong as the company has to offer. However, a large company with a large checkbook may receive a massive list of candidates with the highest tiered candidates first. This causes an issue with companies, as smaller companies are punished for not having a high budget by receiving lower-quality candidates. This can be mitigated by giving a lower paying company access to a randomized subset of candidates with an even distribution of skill levels across the candidates. This however raises an ethical problem for the candidates, as now the candidates are potentially not being connected with their correct fit just because of luck.

Another potential revenue source comes from the other side of the project: the candidates. We have two potential avenues for gaining revenue from candidates, and they both have ethical issues associated with them. The first revenue stream comes from charging candidates for access to higher tiered companies. The way this will work is that we will only put the lower paying candidates on lists given to less prestigious companies, while higher paying candidates will be put on lists handed out to much more prestigious companies (for example, Microsoft and Google). This causes a problem for smaller companies as they are receiving much fewer and potentially lower quality candidates due to their status as a smaller company. This also causes a problem for larger companies, as they may not be able to have access to stronger candidates that can't afford to pay the money to get on their radar. This same problem applies to the candidates. The second revenue stream associated with gaining revenue from candidates involves giving higher paying candidates access to prep materials, such as practice questions, one-on-one tutoring, and resume access. This gives an unfair advantage to people who can afford the price of additional practice. However, this solution is more ethical than the alternative. Ultimately, all of our pricing models are not entirely ethical. However, these problems can be balanced with our need to meet the bottom line by balancing revenue coming from companies by using a pay-for-access model and revenue coming from candidates by giving them access to prep materials.

Another ethical issue comes from the release of data concerning candidates (including skill level in various subjects, graduation date, and email). For a company to utilize our system to the fullest extent, they must have access to a candidate's skill levels in various subjects. This will allow them to determine who would work best for their company. This may raise a problem for some candidates, but hopefully this will be mitigated by having a fair pre screen assessment that accurately describes a candidate's skill levels. Another metric companies can have access to is graduation date. The graduation date will give companies an idea of the age level / expected skill level of a candidate. This could raise issues of companies favoring closer graduation dates as it theoretically indicates a higher level of knowledge, which can leave some of our younger candidates unfairly disadvantaged. The final thing companies need to have access to is a candidate's email in order to get in contact with them. This may pose an issue for some candidates who do not wish to give out that kind of personal information. This can be mitigated by the system acting as a buffer between the candidate and the company. If the company is interested in a candidate, it can reach out to GOTOintern in order to communicate with the candidate. Then, GOTOintern can receive consent from the candidate to give the company its contact information. The downside to this solution is it makes our job a lot more difficult. Alternatively, all the issues of data privacy can be mitigated with a statement of consent before the assessment, allowing GOTOintern to distribute a candidates' skill levels, graduation date, and email.

Intellectual Property Issues

There are a few companies that have a similar business model to Applican. The most notable one is Triplebyte, who has a very similar way of evaluating its applicants. The notable difference here is that Triplebyte is focused on procuring full-time offers for seasoned professionals in the industry, while Applican's target demographic is students looking for summer internship opportunities. Triplebyte's model is very similar to ours' - software developers will take a short quiz that goes over a wide variety of different topics to get a good idea of your strengths and weaknesses. Then, after the quiz, the developer will have a one-on-one interview with a recruiter at Triplebyte. So far, this is very similar to our platform. The difference comes with our matching algorithm. Triplebyte gives users who pass the one-on-one a recruiter that will find them opportunities for full-time positions. This is expensive and time consuming, so for Applican we use a machine learning algorithm that connects students to companies, and instead offload the job of reaching out to students to the companies themselves, which will cut down on cost and time. Also, Triplebyte's market is the top 5% of all software developers, while Applican targets all students, and is able to offer most of its users internship experiences thanks to the breadth of quiz categories we have and the ability for companies to focus on those specific skills.

Another intellectual property issue that may arise comes from the development of questions for the variety of quizzes. Due to our low budget, all of the questions for the quiz are generated by our own team. However, to make sure that they are accurate and indicative of what companies use in their job searching, we strive to create questions that are similar to questions either that we have seen before or questions that are published from interviews with different companies. Due to this, we run into potential issues of questions being identical between ours and company onboarding questions. While this shouldn't land us into any trouble, it is something to be aware of, especially if a company is hyper aware of people stealing their interview questions.

Change Log

- New name! GOTOIntern would restrict us to only CS internships. We wanted to leave ourselves open to other disciplines in the future, which inspired our name change to Applican.
- Our software design section now includes a discussion of React and Elm, a journey we embarked upon since the last design document.
- The deployment infrastructure of our application has become considerably more sophisticated. We now containerize our application, and additionally are now using server side rendering as well.
- Our intellectual property segment now contains a more in-depth discussion of other companies working in our space, as well as other considerations.
-