Technology Director HackPSU

A Comprehensive Guide

Contributors:
Sushrut SHRINGARPUTALE
sush.shring@gmail.com
Matthew HEILMAN
mdh5389@gmail.com
John DURRELL
john.m.durrell@gmail.com

Contents

INTRODUCTION

Congratulations on your acceptance as Technology Director! We're sure you were selected from a competitive pool of applicants as the best choice for the role. The Technology Director is a very important role within HackPSU. A large portion of the success of the event hinges on the success of the tech team achieving their goals, so it is important to prepare you for anything the event may throw at you.

This document provides a high-level overview of the responsibilities of HackPSU's Technology Director as well as an overview of the technologies currently used. It's not the director's responsibility to be an expert all across the stack, or even to work on every project, but you should at least have some familiarity and background knowledge with everything, and this document will help you with that.

Source: *xkcd.com/349* [1]

1. Checklist

This page lists tasks to be taken care of during the semester. This list is not exhaustive, and not every item will be applicable every semester, but it should give you a good sense of how to structure the semester. The list is formatted as *[item - suggested time to complete]*

Pre-event

- Synchronize Sponsors on website and mobile app - Continuous process
- Budget Proposal - 14 WBE[1]
- Finalize Landing Page Design - 12 WBE
- Open Registration - 10 WBE
- Report registration data to the organization - 3 WBE
- Send email to all registrants with QR code - 1 WBE
- Finalize and update schedule on the database - 1 WBE
- Finalize and update workshops on the website - 1 WBE
- Finalize and update prizes on the Website/Devpost - 1 WBE

Week-of event

- Talk to PSU IT and boost building internet capacity - 2 WBE
- Make event schedule for tech team members

Post-event

- Downgrade server and database - 1 WAE [2]
- Change landing page - 1 WAE
- Team Debrief - 1 WAE
- Change active hackathon on database - 2 WAE

2. Essentials

This section contains an overview of several important items of responsibility for the technology director as well as points of access for further help.

2.1 Task Descriptions

2.1.1 Pre-event

- **Budget Proposal -** Estimate the necessary budget of the technology team for the semester. This budget needs to be submitted to the finance team at the beginning of the semester. It does not need to be extremely accurate, but it should reflect a tough estimate of the expenses of the semester to help the organization keep track of expenditure by each team. Typical expenditures for tech team are:
    - Cloud hosting - roughly $30 per month during the semester and roughly $80 for the month of the hackathon.
    - Domain name registration - roughly $30 per year.
    - Apple developer license - $100 per year

- **Finalize Website Design -** Coordinate with the design team to construct a general idea of what the website is going to look like for this hackathon and identify tasks for tech team to complete towards this goal during the semester.

- **Open Registration** - Activate registration functionality in the website. Test that it works properly for the current hackathon and open it up to the public.

- **Synchronize Sponsors on website and mobile app** - Whenever sponsors are confirmed by the sponsorship team, add their logos to the required platforms.

- **Report registration data to the organization** - Give an attendee headcount and dietary restriction distribution to the logistics team, and give the headcount and t-shirt size distribution to the design team.

- **Send email to all registrants with QR code** - Use the admin tool to send an email tool to all registered participants containing their QR code to present at check-in. This serves as a reminder that the event is happening and ensures participants have their code on hand to speed up check-in.

- **Confirm schedule in the database** - Ensure that the schedule information, including check-in, food, workshops, and entertainment, is correct on the website and in the mobile app. Check with the education, entertainment, and logistics teams for this.

- **Confirm prizes on the website** - Link to the Devpost for the hackathon's challenges and prizes.

- **Update times for live countdown timer -** Alter the production environmental variables in Vercel to have the correct time for the current hackathon. These values represent milliseconds since the Unix timestamp.

2.1.2 Week of event

- **Talk to PSU IT and boost building internet capacity** - Talk to Christy Long or IT directly to have this handled. PSU IT configures any special wireless networks we may need and also boosts the capacity of the PSU network in the building to be able to handle nearly 1000 participants.
- **Make event schedule for tech team members** - Create a master schedule that defines where each tech member needs to be during the event. The director will likely float around to deal with any issues that come up during the event, but everybody else will have defined stations. Ensure that each check-in table and each workshop have at least one tech member to scan participants, and assign tech members to help the logistics team as needed for meals and other setup. Ensure a dedicated 8-9 hour time slot to sleep for every member (*including you*). Refer to past schedules for a better picture of how to organize this.

2.1.3 Post-event

- **Give uploaded resumes to Sponsorship** - Download the resumes from the registration data and give them to Sponsorship for the resume book to give to sponsors.
- **Change landing page** - The landing page should be changed to a thank you page after the hackathon in preparation for the next event.
- **Team Debrief** - Have a discussion with the team to gauge what went well during the event and what should be worked on for next semester.
- **Change active hackathon in database** - Use the admin tool to create a new active hackathon for the next semester.

2.2 Accounts and Services

This section lists all the associated accounts and services that the Tech team comes in contact with. The technology director should own the two-factor authentication for all of these services.

2.2.1 Google

The HackPSU Google account is under the email hackpsudev@gmail.com. This account should be on the forwarding list for all emails that come to the HackPSU domain to maintain a record of them for future team members. It has multiple filters and folders set up for this purpose to automatically categorize emails based on which teams they apply to. This email should only be given to Google Cloud (because it requires payment), and to Namecheap (because this lets us set up the email forwarding to all other addresses). The technology director usually handles this account on their own, but giving co-executives access to it is probably also fine.

2.2.2 Google Cloud Platform (GCP) and Firebase

Google Cloud Platform is where most of our services are hosted, including our database, API, resume storage, and email sender. Firebase handles our user authentication, and although it is technically a different service, it's also owned by Google and integrates

easily with our GCP project. Although we used to use the standard HackPSU Google account to manage this service, certain events have convinced us to separate it out to a different Google account: <email goes here> (the reasoning for this is explained in Security).

Ideally, the base account should only be used for actions that specifically require admin privileges and its use in GCP should be restricted to the director. To allow team members to work on the project, delegate out roles to their accounts individually through IAM (Identity and Access Management). This allows us to easily revoke access once team members graduate and avoids giving more access than necessary. Additionally, ensure team members have two-factor authentication enabled on the accounts they use to access GCP or Firebase.

Some applications also require a service account to handle GCP actions (for example, the API requires a service account to access the database), and their permission can also be handled through IAM. They can be disabled, deleted, or re-enabled at will through the Service Accounts tab. When creating a service account, be sure to give it only the minimum permission it needs for the tasks it needs to execute. ***DO NOT*** give it blanket editor permissions, as this can (and previously did) give an attacker access to all of our services if the credential is ever compromised. Never put service account credentials in a git repository, and prefer to handle application access using server-side secrets instead, when possible.

The payment for GCP services is typically done through linking the technology director's credit card, and it will typically cost about $30 per month, with a cost increase to about $80 for the month of the hackathon. See the Cost Reimbursement section for more information.

2.2.3 Namecheap

Management of the hackpsu.org domain is performed at namecheap.com. This includes hosting records for the main landing page, sub-domain pages, service verifiers (mainly email service), and email forwarders. Both the hackpsu.org and hackpsu.com domains must be renewed every year, which costs about $15 each.

Hosting records tell other services that we own the domain. This allows us to host our website and api on the hackpsu.org domain as well as letting Firebase send emails from the hackpsu.org domain.

Domain redirects allow us to redirect anyone who visits a page on our domain to any other URL. Among other things, we redirect devpost.hackpsu.org to our Devpost page for the current hackathon, and discord.hackpsu.org to our Discord server for the current hackathon.

Email forwarding allows any email sent to hackpsu email domains to be sent to team members. Team members will be able to view these emails and respond to them from their personal (or school) email account. It is recommended to use these forwarded domain addresses to sign up for any additional services because we can always add or remove new forwarding addresses to manage access when members join or leave the organization. Listed below are the commonly used addresses and who they should be forwarded to:

- communication@hackpsu.org: communications director
- design@hackpsu.org: design director
- education@hackpsu.org: all education members
- exec@hackpsu.org: co-executives (idk if this one is really used?)
- marketing@hackpsu.org: marketing director
- sponsorship@hackpsu.org: all sponsorship members
- team@hackpsu.org: all team executives.
- technology@hackpsu.org: technology director
- <catch-all>: forward any other email sent to the hackpsu domain to hackpsudev@gmail.com (be aware that this probably includes spam, though).
- Additionally, forward all of the above to hackpsudev@gmail.com. This account should retain a record of all forwarded emails so that future team members have access to them. It has filters set up to help organize them (although they only apply back to late fall 2022, since that's when they were created).

2.2.4 Github

All of HackPSU's source code is stored on Github. All the repositories are owned by the Hack-PSU organization, which you should be an admin for. The organization page is at github.com/Hack-PSU. Most of our active repositories have some kind of continuous deployment set up to automatically deploy new changes to production upon pushing to the master branch. Some notes on our repo structure:

- The master/main branch of each repo should be protected, and a direct push to it should be prevented unless absolutely necessary. Any code pushed to master should be production ready, and, depending on the configuration, may be directly put into production.
- The dev branch should be used to collect developments before they are ready to be pushed to production. The changes from each member's branch should be merged into dev before pushing them all at once from dev to master.
- Individual branches should be created for each task/issue/project that a team member works on. Development should be handled on these branches, and pushed to Github for backup. Once finished, a pull request should be made to merge these into the dev branch, and reviewers should be added as needed. Once the changes are determined to be good, they should be merged into dev.
- Team members should put information in commit messages to indicate what was changed in each commit, and why it was changed. It isn't necessary to write entire paragraphs, but ensure that more is being written than one-word messages.
- For now, all repositories are private. We had issues in the past with credentials getting leaked by being accidentally included in source control, and it seems to be safest to keep the repos private because of this. However, it *is* very valuable for tech members to have public projects that they worked on attached to their Github accounts, so perhaps this could be revisited in the future.

2.2.5 Vercel

Vercel is the service we use to deploy our frontend website and our admin web application, which we get for free because we're a student organization. All commits to the frontend and admin-web-app repositories automatically generate a preview deployment on Vercel, which we can use to see how things look before actually committing it to the dev branch. Pushing to master on those repos will also automatically deploy to our

production. Changing environment variables for the production instance (such as the time of the hackathon) must be done through Vercel, and will be updated on the next build.

2.2.6 Apple Developer / Google Play Developer

HackPSU has developer accounts through Apple and Google to list our mobile application on the Apple Appstore and Google Play store. The releases of each application should be submitted to each platform at least a week before the hackathon to give them enough time to review and release it, which can be done using Codemagic and each platform's respective portal. For Apple, you will need to ensure that deleting an account authenticated with AppleId properly hits their endpoint to revoke the access token (this should already be done in the app and API). For Google, you will need to fill out data privacy forms that reflect what data we collect and who we share it with. For both, you will need to provide a test account, and ensure that there are some events in the database so that the reviewers have something to look at.

Right now, the name on our Apple developer account is "Rahul Ramkumar", but the email is hackpsudev@gmail.com. As far as we know, it is not possible to change this name. Apple periodically locks accounts for no reason because they're stupid, and you might have to call their support to reactivate it if you don't have access.

Apple's developer license costs about $100 per year, and Google's is free. See Reimbursement for more details.

2.2.7 Codemagic

Codemagic is the CI/CD service we use for our mobile application. Once its build is finished, it will automatically push the built package to both our Android and Apple releases, where you can finish out the release process as described above.

Before initiating the Codemagic build, be sure to update the version numbers in the pubspec.yaml. The final number in the version (after the '+') needs to be incremented for each build performed to match the build number in Codemagic (it will fail otherwise).

2.2.8 Expo

Expo (https://expo.dev/accounts/hackpsudev) is the platform we use to deploy and distribute the admin mobile app. The HackPSU gmail account is the owner of the "hackpsu" organization on Expo, which should contain the admin mobile app project. The technology team leader should have "admin" access to this organization, technology team members working on this project should have "developer" access to it, and all other organizers should have "member" access. Organizers must accept the invitation through their Expo account to get access to the admin mobile app.

2.2.9 Devpost

Devpost (*devpost.com*) is the submission platform for projects created at our hackathon. Each hackathon recognized by MLH is required to have a new Devpost page that outlines the logistics, challenges, and prizes for the event. This is a standard service for hackathons all over the world, and it is good for us to adhere to that standard. Once the prizes have been declared and announced at our closing ceremonies, the winners should also be announced on Devpost on the same day. In the past, both the technology director and

co-executives have handled this responsibility together, so be sure to check in with them for the current hackathon.

See *help.devpost.com* for more information on how to work with Devpost. You can also refer to our previous hackathons to get an idea of how to handle this.

2.3 Team Structure

The technology group is divided into three main teams: Frontend, Backend, and Mobile App. The frontend encompasses our main website and admin web application, the backend encompasses our API and database, and the mobile apps encompasses the user-facing and admin mobile apps.

In the past, we have had a project manager that helps keep each project on track for the hackathon. Depending on the experience level of the members in the team, there may be an additional mentor (project leader) that helps newer members get familiar with the codebase, but this may end up being the same person as the project manager.

When selecting new members, it is important to take note of what skills the team needs to be successful, as well as judging the candidate by how much they are willing to learn. Learning is a crucial part of the HackPSU Tech experience, so this should be the cornerstone for any team member that is being added. Depending on what the demographics of the team are, we recommend looking for either fresher or more experienced students. This ensures that there is a good distribution of people that are willing to learn new things and take over in the coming semesters and team members that are excited to lead projects and have the technical skill-sets to make that happen.

2.4 Important Resources

2.4.1 Christy Long (Advisor)

Our current advisor is Christy Long. She is Director with Penn State IT, and a huge supporter of HackPSU. She has incredible contacts all across the board and in the past has helped us with mission crucial tasks such as implementing special WiFi at the event for hardware, coordination with Penn State IT for any other requirements, reaching out to companies (Microsoft, Apple, Google) for sponsorship, and any other administrative hurdles that we may have needed to cross while building up HackPSU. She is an amazing person and an incredible resource.

2.4.2 Owned Hardware

Supposedly, the Tech team owns the following:

- 1 soldering iron
- 6 RFID tracking system boxes (and one development version)

These items are listed here for posterity. However, these are from a deprecated RFID project, and as of current, it is not known whether they work or what we would even use them for.

2.4.3 Recommended Practices

- **Attend one or more hackathons**
  A very important experience for all team members is to attend one or more external hackathons. It provides team members with an important perspective about how other hackathons are run and helps generate ideas for new things we may want to implement for HackPSU.

PROJECTS

3. Website

The website, hosted at *hackpsu.org*, is the first destination for our participants. We direct them here for hackathon information, registering for the event, and retrieving their QR code for checking in to the event. It is built in React and maintained by the Frontend team. Each semester, the site needs to be updated for the hackathon's new theme, and you will need to work with the design team to develop assets for the site. This project is currently located in the frontend-template repository.

3.1 Landing Page

The landing page is the "homepage" of the website, and it provides an overview of the entire event for anyone interested in participating. It commonly includes:

- Event date and location (including countdown)
- Link to registration
- FAQ and event rules
- Schedule (including workshops, food, and entertainment)
- Submission link
- List of event sponsors
- Link to sponsorship packet (for potential sponsors)
- Social media links

3.2 Registration Page

The registration page collects users' information to register for the event, and optionally, their resume to submit to sponsors. This information is sent to the API and stored in the database, where you will need to retrieve it so that you can give it to MLH before (and after) the hackathon. Several fields are required by MLH each semester (which are laid out in the partnership packet that co-execs will handle), and several more are useful for demographics that we can aggregate and give to sponsors.

3.3 Profile Page

The profile page should allow participants to see and edit their information. The most important piece of information is their QR code, which participants will need to check in on the day of the hackathon. It should also let them sign up for any classes that are offering extra credit for hackathon participation.

3.4 Deployment

The website is set up to automatically deploy through Vercel upon pushing to the main. It also automatically generates preview deployments for the dev branch and pull requests. The environment variables for deployments are maintained as “production” and “preview” environment variables in Vercel. This includes the hackathon start and end times, which will need to be updated each semester.

4. Backend

The backend for HackPSU drives all the data and interfaces across our several applications. The API is written in TypeScript, and it interfaces with an SQL database to store all the data. Both are hosted on Google Cloud Platform. Additionally, it interfaces with Firebase for user authentication to ensure that users can only access their own data, and to manage permissions for various operations. The backend team is (predictably) responsible for maintaining all the pieces included in this project.

4.1 Database

The database is backed by MySQL 8.0 and is hosted through Cloud SQL. It contains all of the event schedule, administration, judging, and participant information (with the exception of resumes). It contains a production environment with actual data, a staging environment used for manual testing, and a test environment reserved for integration testing with the API.

You can connect to the database using MySQL Workbench and Google’s Cloud SQL Proxy, or by using the Cloud Shell in GCP. Each tech member using the database should have their own account for access, which should be revoked upon graduation or leaving the team. Additionally, there is an account for integration testing and for root access to the database. Accounts can be managed through the Google Cloud Console.

Cloud service accounts that access the database will need to be granted permissions through IAM (Identity and Access Management) in Cloud Console. These accounts will have access to sensitive user data, including names, email addresses, and phone numbers, so special care should be taken to ensure that the credentials for such service accounts are not exposed anywhere.

Inside the database, users are uniquely identified by their Firebase uid, which lets us track their registration information and extra credit information. Registrations are stored with a field that identifies their hackathon, which is also used to identify hackathon workshops and project scores. Projects and scores are linked to a list of organizers that do the scoring, which should be maintained each semester to include only current members of HackPSU.

4.2 API

The API interfaces with the database to provide data access to all of our other applications as well as handling resume and image file uploads to GCP storage buckets, sending notifications to mobile app users, and distributing judging assignments. It is written in TypeScript using the NestJS framework, deployed as a Docker container on

Cloud Run, and provides a REST API based on HTTP requests. Its source code is currently available in the apiv3 repository.

Each operation in the API has a permission level that determines what level of user is allowed to perform each operation. For example, scoring projects should be limited to HackPSU team members, and opening judging should be restricted to the technology executive. The permission levels are as follows:

- **Level 0: Participant** - Participants have the least permissions. They are allowed to submit any forms found on the registration app, read the website, and read schedule information. They also have the permission to change their information, but currently there is no support for that on the frontend. Additionally, they are only allowed to see their own registration information and nobody else's.
- **Level 1: Volunteer** - The next level of permission is a Volunteer. Volunteers are people that are helping organize the event day-of, but are not part of the team during the rest of the semester. They have access to the admin app.
- **Level 2: Team Member** - HackPSU organizers have access to most features that will not completely affect the integrity of the data for the event.
- **Level 3: Director** - Directors have access to view registration information of participants and to make any changes to the data barring some database tables on whom the success of the event depends.
- **Level 4: Tech Executive** - Literally everything. Members of the tech team have full access to do anything they want to any data, anywhere. This permission level is the "sudo" of the HackPSU system, so it should not be handled lightly.

Additionally, there is support for a level of permission that requires user authentication to match, and a level of permission that does not require matching authentication. For example, a *participant* can only view their own registration information, but the tech executive can view *anyone's* registration information.

The API should be deployed through a Github actions workflow in the Github repository. The credentials for this are stored as Github organization secrets, so they should not appear in the build logs.

4.3 Firebase

Firebase handles user authentication, permissions, and hosts the images for sponsor logos on the website. It is part of Google Cloud Platform, and can be accessed through the firebase console with the same project id as the GCP project. We don't particularly want to handle passwords and cryptography on our own because it's extremely complicated, but Firebase takes care of this for us. Permissions are managed through the Firebase Custom Claims API.

In addition, we use Firestore to store user messaging ids for mobile app push notifications and feature flags that indicate whether certain features are active (specifically, judging).

4.4 Misc

Google Cloud Platform has several different miscellaneous features that are covered by

the backend, but don't really fit into their own category.

Google Cloud storage buckets are used to store resumes and sponsorship logos. Typically, we store a "dark theme" and "light theme" logo for sponsors to display them properly on different platforms. After each hackathon, you will need to export the resumes from the storage bucket to be given to the higher-tier sponsors.

IAM, or Identity and Access Management, manages the Google Cloud permissions for various users. For team members, it is probably okay to give editor access to people that need it, but be sure that they have some form of two-factor authentication enabled for their account. Also, be sure to remove their access once they either leave the team or graduate.

5. Mobile App

The HackPSU mobile app is available on Android and iOS and lets participants do most of the functionality they could do on the website, with the addition of being able to subscribe to workshop and entertainment notifications. It is written in Dart and uses the Flutter framework to keep cross-platform development as easy as possible. Its source code is currently available in the mobile-app repository.

5.1 Deployment

Deploy the mobile app to Google Play and the Apple app store at least a week and a half before the hackathon. This will give them enough time to review and approve our new version to be made available for download. Building and deploying the application package is done through Codemagic, which integrates with our Google and Apple Developer accounts to automatically push the build to the release staging area. Follow the steps below to deploy the application:

1. Update the version numbers in the pubspec.yaml file (in the Github repo). The final number in the version (after the '+') needs to be incremented for each build performed to match the build number in Codemagic (it will fail otherwise).
2. Go to Codemagic to manually initiate the build, which will push to our Google and Apple developer app automatically upon finishing.
3. Create two test accounts in the production instance so that the reviewers can see our application's functionality. Additionally, create various sponsors and workshops/entertainment events in the production database so that these show up properly for the reviewers (unnecessary if the actual versions are already finalized and entered).
4. If any UI elements have changed since the previous release, you will need to take screenshots of the app on several platforms, which is most easily done using emulators.
5. For Android: Go to Google Play Console. You will likely need to fill out a form that details exactly what data we collect and how we share it. Refer to the registration information in the database for what we collect, and the registration information required by MLH (which may change between semesters) for what will need to be disclosed.

6. For iOS: Go to Apple Developer. If you've changed any UI elements, they'll be *really* anal about getting screenshots for every possible screen size combination, so be ready for that.

Secrets in the secrets.dart file are injected into the build by Codemagic via the "MY_SECRETS" environment variable. If this file has changed between builds, be sure to update this environment before building by copy-pasting the base64-encoded contents of the file into the environment variable.

6. Admin Mobile App

The Admin Mobile App is available on Android and iOS and allows organizers to scan QR participants' QR codes for tracking event and workshop attendance as well as submitting scores for judging. It is written in Typescript using React Native and is distributed using the Expo platform. Its source code is currently available in the admin-mobile-app repository Permissions for organizer accounts are handled as custom claims in Firebase and judging assignments are determined using tables of judges and projects in the database.

6.1 QR Code Scanning

The first major feature of the admin mobile app is QR code scanning. This allows us to track attendance for the hackathon as a whole as well as specific workshops, if necessary for extra credit purposes.

Each participant has a unique QR code that is generated from their Firebase user id. They present this QR code to an organizer (typically a member of the technology team) at Check-In, where the organizer scans the QR code using the admin mobile app to record them as having attended the event. Check-in tables are generally set up at each entrance area (front and back of the Business Building). A participant's QR code can be viewed by signing in on our website or through the mobile app.

We are *required* as an MLH partner hackathon to have a method of tracking which registrants actually attended the hackathon, and this allows us to fulfill that requirement. For distribution to MLH, participants' registration data can be obtained as a .csv file from the database.

6.2 Judging

The judging screen can be enabled or disabled based on flags managed by the API. The app queries the API to check the status of the flag, which is stored in Firestore. If disabled, the app shows a placeholder. Otherwise, it will display the judging assignments for the logged in organizer.

The admin mobile app works in tandem with the admin web app to facilitate the judging process for project submissions at HackPSU. The web app provides a list of organizer judges and project submissions to the API, which generates assignments for each judge and records them in the database. Upon logging into the admin mobile app, these assignments are queried from the API and are viewable in the app for each user.

Organizers are expected to fill out the form in the app for each project they have been assigned. Upon scoring a project, the app sends the scores to the API, which records them in the database to be later fetched and viewed in the admin web app.

6.3 Deployment

Deployment and distribution of the admin mobile app is managed through the Expo platform. It can be accessed through the Expo Go mobile app using an Expo account that has access to the “hackpsu” organization. Ensure that the “production” version of the app is the latest published version to minimize confusion from organizers about which version to select.

The admin mobile app is currently deployed from the command line using the Expo CLI while logged into an account that has developer permissions in the organization, although it would be prudent to set up a Github actions (or similar) workflow to standardize this. Since the Expo SDK changes constantly, it is important to ensure that the production version of the app targets (and works properly on) a version of the SDK that is currently supported by the Expo Go mobile app. Check Expo’s documentation for more details.

6.4 Pre-hackathon Setup

It is important to ensure that each organizer has access to the admin app and understands how to use it. Before the hackathon, first confirm with other HackPSU executives that the student roster is updated correctly, then ensure that all organizers on the roster have been added to our Expo organization and have the correct permissions within our system (this can be checked using the admin web app or by querying the API directly through Postman). Also conduct a demo at the last GBM before the hackathon to show everyone how to use it and to solve any set-up issues.

Keep in mind that the users of this app (especially judging) use it at most twice a year and will be largely unfamiliar with it, so the development process should prioritize keeping the interface as simple as possible and minimizing the responsibilities of the user.

7. Admin Web App

The admin web app allows several teams in HackPSU to more easily manage various information about the hackathon, including start and end times, workshop, entertainment, and food schedules, sponsors, and judging. It is written in Typescript using the React framework, it is hosted using Vercel, and its source code is currently available in the admin-web-app repository.

This application is hosted at admin.hackpsu.org, and the domain is managed through Namecheap and Vercel. The admin web app is deployed automatically upon pushing changes to the production branch.

7.1 Judging

Right now, the most important job of the admin web app is to help manage the judging

process. Determine which organizers are participating in judging, then once the judging submission deadline has passed, use the list of judges and projects to generate the assignments each judge is responsible for judging. Then, once organizers have finished with their assigned teams, the web app provides a scoring breakdown for each project that will allow the deliberation to proceed more smoothly. It is important that the scoring averages should *NOT* be used as the sole metric by which to decide the winners, and are understood instead as a “starting point” for the deliberation that can help inform the judges which projects are worth considering for which categories.

OTHER USEFUL INFORMATION (TODO: find a better name for this section)

8. Security

HackPSU holds personal information from our participants and organizers, including names, email addresses, phone numbers, resumes, and project scorings, so it is important to limit access to our systems to prevent unauthorized access. In addition, we do not want attackers to steal our cloud resources and stick us with a large bill or to gain access to our social media or email accounts and start spamming our participants and sponsors. For these reasons, HackPSU, and especially the technology team and its director, need to be vigilant about security.

8.1 API Permissions

As discussed earlier in the backend section, our API utilizes permission levels attached to accounts through Firebase custom claims for accessing different API routes. A token is provided to a user upon each authentication with Firebase, which can then be passed along to the API. Each token's audience is decoded for a specific Firebase project, so it is (nearly) impossible for an outside user to obtain a token for a user without actually being able to authenticate as that user. User authentication is handled entirely through Firebase because password cryptography is much more complicated than we are able to handle on our own.

These permissions are used to validate authorization for organizers logging into the admin mobile app and admin web app. They are also used to validate access to specific user data as a tech admin or a matching user.

8.2 Service Accounts and Google Cloud IAM

We lease cloud resources from Google Cloud, which can be very expensive if used incorrectly, so it is extremely important to protect access to our Google Cloud project. Permissions for various users and service accounts can be managed through the IAM (Identity and Access Management) screen in Google Cloud Console. Be sure that any account with Google Cloud access is protected by two-factor authentication to help protect against stolen account credentials. Members of other teams (including co-execs) should *not* have any access to our Google Cloud project.

The HackPSU gmail account, and (optionally) the technology director should have "owner" and "billing" permissions, and any other technology team member that needs access can

be given permissions as needed.

For technology team developers, it might not be immediately clear which permissions are required for their current projects, so it might be fine to give them blanket permissions for Firebase Admin, or even Editor.

Service accounts are Google Cloud accounts that can be generated to perform tasks automatically. When assigning permissions to service accounts, the minimum list of required access roles should be determined as a process of development, and they should be granted *only* those permissions. Multiple keys can be created for accessing a service account, and a different one should be created for each service or developer that uses it. These keys should *never* be stored inside a git repository (even in encrypted form), and should instead be managed as secrets by CI/CD providers. Developers that require local service accounts (e.g. all of the backend team) should place their key in an entirely separate folder from their Github repository and provide the absolute path to it in their local environment. These steps help prevent leaking credentials, limit the blast radius of a compromised credential, and help diagnose the cause of a compromised credential disclosure to determine if other credentials might have been compromised in the same way.

8.3 Passbolt

Passbolt is a free and open source password manager which we use to manage the passwords for various accounts throughout the organization, including both technology-specific accounts and social media accounts that are managed by other teams. Currently, we host our own Passbolt server on a Compute Engine VM in Google Cloud. The technology director is responsible for ensuring that access to passwords is restricted to members who specifically require it, for ensuring that passwords are not duplicated across accounts, and for ensuring that passwords are of adequate strength.

8.4 Organizational Turnover

HackPSU is a student organization, and so we have regular turnover of members all across the organization. It is the responsibility of the technology director each semester to ensure that all members of the organization are granted proper access upon joining and have their access revoked upon leaving (with the exception of certain members staying in contact in an advisory capacity). This includes, but might not be limited to:

- Google Cloud and Firebase IAM permissions
- Developer service account keys
- Developer database logins
- Developer Github access
- API permissions
- Expo organization permissions
- Email forwarding
- Passbolt access

In addition, it is a good practice to change the passwords for all accounts each semester. We note that while Microsoft Teams access has typically been handled by co-execs, it may

end up being the responsibility of the technology director as well.

8.5 The Development Secrets Repository

Currently, we maintain the team-only-files Github repository to store files that are necessary for local development, but should probably not be shared publicly. This includes Firebase configs and the mobile app secrets.dart file, among other things. These files can be retrieved by team members when setting up their local development environment.

8.6 Security Breaches

In the event of a compromised credential, the technology director (and, depending on the credential, other applicable team executives) is responsible for diagnosing the breach, changing passwords and revoking credentials as necessary to lock out attackers, and recovering the affected account, if necessary.

8.6.1 Cloud Breaches and Crypto VMs

The most serious type of breach is a breach on our cloud resources. In the past, there have been two incidents where the credential file for a service account with too many privileges was leaked and picked up by an attacker who then used it to create hundreds of Compute Engine VMs for mining cryptocurrency. This racked up thousands of dollars in charges overnight.

In this scenario, first disable a compromised service account or de-privilege a compromised developer account to prevent the attacker from taking further actions. Use the audit log to determine which account has been compromised, which can be accessed by querying Logs Explorer or (at the time of writing) by clicking the notification bell at the top right of Cloud Console, then clicking “see all activities”. We also have a saved Logs Explorer query to view the audit log. Next, attempt to undo the damage that was caused by the attacker, particularly deleting any Compute Engine VMs that may have been created to mine cryptocurrency. Then, get in touch with Google Cloud billing support to get the bill adjusted. They typically offer users a one-time adjustment for this type of thing, and will probably just tell you to be more careful with credential files. Keep in mind that the charges will not fully propagate for about 36 hours, so there will be some time before the bill finalizes itself (you should still get in touch with support immediately, though). Finally, assess how the credential leak happened to avoid it in the future and re-issue credentials as necessary for our systems.

Both past incursions of this type were caused by a credential file being published on a public Github repository. In the first, the credential file was supposedly ignored by the .gitignore, but showed up in the compiled Javascript code anyways. A bot picked it up and started using it within an hour. In the second, the credential file was encrypted, but it seems that the encryption key was guessed, and that it was leaked anyways. For these reasons, we now recommend that .gitignore files include quite liberal wildcard matching for common credential file names, and that credential files should be stored in an entirely separate directory from the local copy of the git repository to avoid being picked up and pushed to a public repository.

If a credential file is exposed at any point on a public repository, revoke and reissue the

credential, even if it does not seem like an attacker has discovered it yet. Git history on Github is functionally *permanent*, even if overwritten by force pushing. In addition, Github keeps permanent references to branch states in PRs, so any PR (even closed or merged ones) can be checked out to find the necessary files, *even* if deleted by force pushing to other branches, and *even* if removed by tools like git-filter-repo, BFG, or git obliterate. Credential files are very easy to replace, so just be safe and revoke it.

8.6.2 Other Breaches

There are several lesser breaches that might happen, such as a social media account being compromised and getting used for spam. In these cases, simply change the password on the account, inform the executive team, and delete the unauthorized posts. If an attacker changed the password *before* you get to it, remember that you likely have access to the email used to create the account via the forwarding service in Namecheap. You can probably forward any "forgot password" emails to yourself (or the HackPSU gmail account) to recover the account.

9. Budget and Reimbursement

The technology director is responsible for ensuring that all required resources, domains, licenses, etc. are paid for. Typically, these will be paid from your credit card or bank account and then reimbursed by the club. Upon making a purchase, make sure to save the receipt with your name on it and forward it to the finance director to be reimbursed by the club through the ASA office. If the receipt does *not* have your name on it, you might be able to photoshop it in there as long as nobody asks and you're being honest about your spending (i.e. if you paid for it, you get it reimbursed).

We pay for the hackpsu.org and hackpsu.com domains through Namecheap, which each cost around $15 per year. These receipts are usually available upon domain renewal and will be emailed to you as well.

We pay for an Apple developer license so that we can distribute our mobile app on the iOS App Store, which costs $99 per year. This purchase confirmation won't look exactly like a receipt, but it has been accepted for reimbursement by the ASA office in the past (just make sure it has your name on it). Android development is free, and distribution on the Google Play store costs nothing.

We pay for Google Cloud resources, including Cloud SQL, Cloud Run, Cloud Storage, Secret Manager, Artifact Registry, and Compute Engine. This typically costs around $30 per month, with the bulk of the cost coming from Cloud SQL. However, during the month leading up to the hackathon, our site gets a lot of traffic which hits our API more often, so costs for this period have shot up to about $60-90 for that month in the past. In the past, we've also spent about $40 for a higher tier of email service through Sendgrid for this month to ensure that we can send all of the reminder emails to registrants right before the hackathon.

We have a student sponsorship from Vercel, so hosting the frontend costs nothing.

In total, it can be expected for the technology team to spend about $500 in club funds per year.

TECHNOLOGY TEAM HISTORY

10. History

HackPSU was founded through InnoBlue in 2012 and became an independent student organization in 2015. As such, we have had many different iterations of the technology team and its stack over the years. While this section is not strictly pertinent to the stack that you will be taking over as a new director, it will give you some insight into ideas that were tried in the past, which might help inform future decisions that you make as a director. In addition, in the event that any old providers or accounts start sending you emails, this might help you understand them.

10.1 Old Projects

10.1.1 Prehistoric Websites

These HackPSU websites pre-date the official Hack-PSU github organization. Their authorship and dates used are not known exactly to the current author of this document as of December 2023. It is also possible that other previous projects from this time period exist, but are entirely unknown.

The first known HackPSU website was developed in 2014 and (presumably) used for the Fall 2014 and Spring 2015 hackathons. Its code can be found in the hackpsusite repository belonging to Christina Platt[1] and Albert Guo[2], which itself is an upstream fork of an older repository from the same semester. It consists of a single webpage built largely with HTML/CSS with some help from jQuery and Bootstrap.

The second known HackPSU website was constructed separately from the first version (as opposed to forking), and was under development from Fall 2015 to Spring 2016. It was (presumably) used for hackathons during those semesters, and its code can be found in the hackpsusite2 repository belonging to (and developed by) Christina Platt. It has one primary page and one signup page. Like the previous version, it primarily uses HTML/CSS with Bootstrap and jQuery, but additionally includes custom Javascript. It was later forked for the Fall 2016 hackathon site, which lives in our official Github organization.

---

[1] Christina Platt is credited by the hackpsusiteF2016 repository as the creator of the “first and original” HackPSU website (although other developers exist). She appears to be the sole developer on the second known version of the site, and appears to have worked on HackPSU from Fall 2014 through Spring 2016.
[2] Although uncredited by official documentation, Albert Guo is listed on all commits to both repositories for this version of the site.

10.1.2 Old Websites

The HackPSU Fall 2016 website, available in the archived hackpsusiteF2016 repository, was forked from the hackpsusite2 repository. Changes were only made to the ReadME and the CNAME file for DNS. It is presumed that this version of the website was used for the Fall 2016 hackathon.

It is currently unknown (as of December 2023 by the current author of this document) which websites were used for the Spring 2017 and Fall 2017 hackathons.

The first website to undergo long-term use and continued development can be found in the frontend repository. It was the hackathon's primary website from Spring 2018 to Spring 2023 (although it still handled some functionality in Fall 2023). This repository featured a "static" landing page (under the "website" directory) built with plain HTML/CSS as well as a larger web application (under the "user-registration-app" directory) backed by the Angular framework, which handled registrations, profile management, extra credit registration, and the day-of-hackathon livesite, which showed a more detailed schedule, including workshops and entertainment events fetched from the backend. This website was originally hosted through Firebase with images stored in an AWS s3 bucket, but was moved to Vercel around 2019. The static landing page was discontinued after the Fall 2022 hackathon because the team found it annoying to develop two different versions of essentially the same site, and instead preferred the Angular application (whose homepage was already including most of the same information as the static page anyways). However, due to accumulated style bloat, the deprecation of the angular-materialize package for future versions of Node.js (which Vercel forces us to migrate to eventually), and the team's favorability of the React framework, this version of the website was eventually retired.

10.1.3 APIs

The v1 and v2 versions of the backend monolith for HackPSU can be found in the api repository. The first version was written in Javascript, stores information in an SQL database, and was hosted on Google Cloud's App Engine. It provided support for hacker registration, RSVP, workshop events, push notifications to the livesite, scanning records for RFID, and table assignments, as well as a documentation site auto-generated from annotations in code on each route handler. It appears to have been in beta in Spring 2018 and fully launched in Fall 2018.

Version 2.0.0 was released in Spring of 2019 and featured a migration to Typescript to enable better intellisense while coding and better runtime type safety (something Javascript is notorious for mishandling). Support for scans via RFID was removed in 2020 and was eventually replaced in Spring 2022 with check-in via Firebase uid (obtained by an admin by scanning the participant's auto-generated QR code). The original implementation of table assignments and judging do not seem to have been used after Fall 2018, and they were replaced by a new judging system in Fall 2022. Eventually, support was also added for retrieving an ordering of sponsor logos for the website and mobile app.

Both versions handled deployment automatically using Travis CI/CD. This used an encrypted GCP service account credential file in the repository which could be decrypted with a passcode. These encrypted files have since been removed from the public repository, however, as leaking the passcode to the files resulted in a security breach by

crypto-miners.

Ultimately, the v2 API was discontinued due to frustration with coercing data mappers to fit certain interfaces that didn't really match their use case or the underlying data format. It was used in transition during the Spring 2023 hackathon and fully retired for the Fall 2023 hackathon in favor of API v3, which uses an object-relational model within the API itself to better match the database.

There also briefly existed an API for sending notifications from the admin mobile app to participant mobile apps (hosted using Cloud Functions) as well as a websocket API for updating events and sponsors more quickly (hosted using Cloud Run). These can be found in the notifications-api and api-ws repositories. These repositories were made defunct and rolled into API v3 for the Spring 2023 hackathon.

10.1.4 RFID Scanners

The RFID scanner system was in use from Spring 2018 to Fall 2019. Participants would be given an RFID band at check-in, which they would use to scan into workshops and other events around the hackathon. Its code can be found in the hackPSUS2018-rfid, scanner, and scanner-pole repositories. It used customized Raspberry Pi and Arduino microcontrollers and was written in C++ and python. To deal with spotty wifi at the event, the hardware communicated with a redis cache hosted on a local server (redis-server repository) owned by the technology team and physically located at the hackathon. Ordering RFID bands for all participants was expensive, and it was a huge pain to set up the local server on the day of the event.

The RFID system was unofficially abandoned in 2020 for virtual hackathons during the COVID-19 pandemic, and it was officially retired in 2021 in favor of using the admin mobile app to scan QR codes. Additionally, the redis cache was no longer necessary due to wifi upgrades, so the admin mobile app communicates directly with the API instead of with a local server.

10.1.5 Admin Tools

Over the years, we have had several different instances of administrative tools that help us run the hackathon on the day-of. The most notable of these is the old admin app (found in the admin repository), which was written in Typescript, backed by Angular, and hosted through Firebase. This application helped manage workshop events and locations, registrations, and checking out air mattresses or loaner hardware. However, it fell out of use around 2020 due to its UI being very sketchy, and it was replaced by the React-based admin web app in 2023.

We also used to have systems to manage the day-of Slack channel (which was eventually replaced by a human-managed Discord server). This included a push notifications bot (hackybot) and a fork of mangobot, which helped manage requests for mentorship. These were abandoned in favor of the communication team's management of the Discord server and the ability for participants to set their *own* event reminders in the mobile app.

Judging has gone through some changes as well. The team forked a repository and briefly dabbled in using HackMIT's Gavel as a judging system, but it is unknown whether this was

ever actually used at a HackPSU event. For several years, judging was handled by manually exporting a set of projects from Devpost, manually distributing them across the HackPSU organizers, then copy-pasting scores from a local spreadsheet into one giant Google spreadsheet, which was a mess every single time. This system was replaced by having judges be automatically assigned by the API, scored via the admin mobile app, and viewed in the admin web app.

10.1.6 Passbolt

Passbolt is an open-source password manager that we once used to manage a bunch of secrets for the technology team and the HackPSU organization in general. It ran on a Compute Engine VM instance and connected to the same Cloud SQL instance as the main database (although information was stored in a different schema). In addition, the GPG keys were managed by our team. The passbolt_api repository demonstrates this previous deployment. This fell out of use in 2020 before being revived under a slightly different implementation in Fall 2023.

10.2 Previous Technology Team Directors

There have been several directors of the HackPSU technology team over the years, some of which have chosen to provide contact information for future students. In the event that something old resurfaces that you do not understand, you might be able to reach out to the previous directors[3] listed below:

- Matthew Heilman
  - F2017 - F2018
  - mdh5389@gmail.com
- Sushrut Shringarputarle
  - ??? - F2018
  - sush.shring@gmail.com
- Julia McCarthy
  - S2019 - S2020
- Rahul Ramkumar
  - F2020 - S2022
- John Durrell
  - F2022 - F2023
  - john.m.durrell@gmail.com
- Kanishk Sachdev
  - S2024 - present

[3] This is the list of previous technology directors known to the current author of this document as of December 2023. It is possible that there are previous or subsequent directors that are unknown. Although additional previous *developers* are known, they predate even the original version of this document, and It is unknown whether they held the title of “director”. As such, they have been omitted from this list.