BYU ITC446 Project Definition Document

The Kashti 2.0 team (Students):

Cody Uhi
Daniel De La O
Derryck Dunn
Spencer Sundrud
Stephen Kendall

The Sponsor:

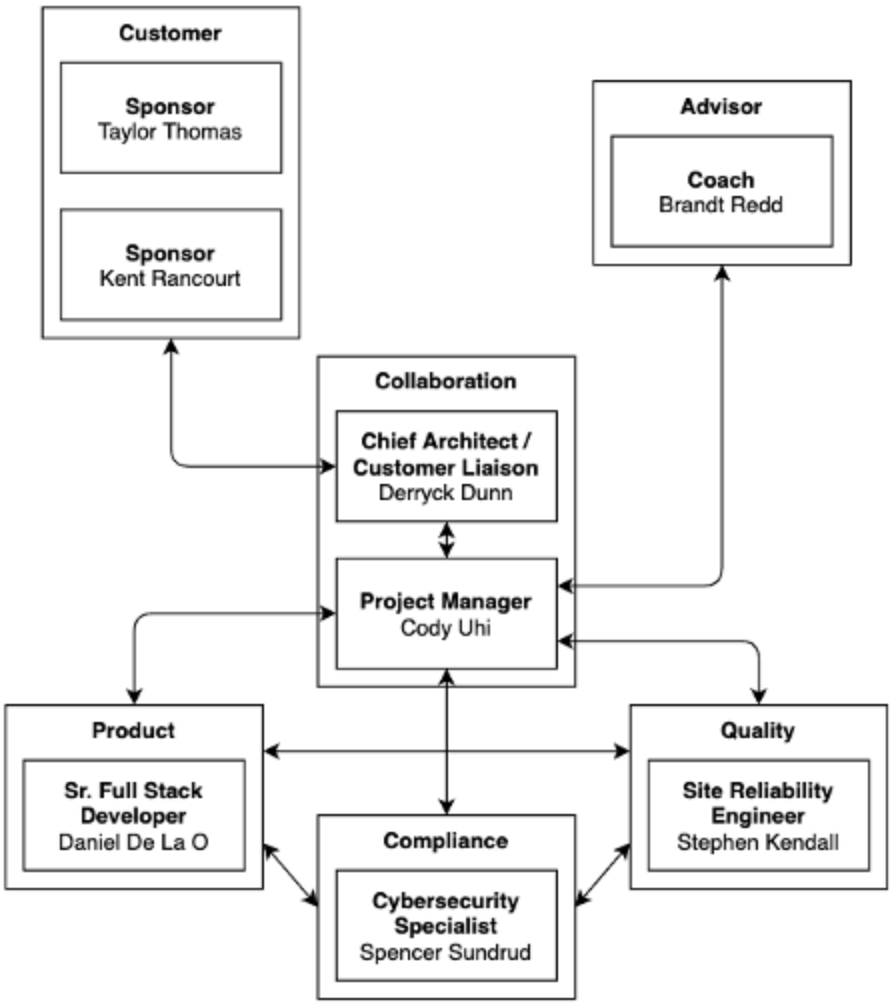
Kent Rancourt
Taylor Thomas
Microsoft

The Coach:

Brandt Redd

The Customer:

The Brigade/Kashti Open Source Community



Kashti 2.0 Team Organization Chart

Project Objective Statement:

Create a secure, interactive web-based User Interface (UI) that offers full functionality for the Brigade event-driven scripting platform by May 2021.

Introduction:

Microsoft and the open-source community (**the sponsors/stakeholders**) have built an event-driven scripting platform called Brigade which allows users to connect arbitrary events with arbitrary actions by running scriptable, automated tasks in the cloud as part of a Kubernetes cluster. The first version of the backend application (Brigade) is being iterated upon to improve functionality and add features. Because the Brigade backend is being improved, the

User Interface (Kashti) must also be updated and improved to expand its feature set and match those of the Command Line Interface (**the problem**).

Our team will be primary contributors to the creation of the User Interface. The sponsor requires that our team build an improved version of Kashti that supports full functionality for Brigade 2.0 by the end of April 2021 so it can be presented at Kubecon in Summer 2021 (**the vision**).

Research:

Stakeholder Objectives:

Since Brigade and Kashti are currently already in production, the use-cases for the application have been well-defined by its users. User feedback and developer input has been consolidated and refined in The Brigade 2.0 Proposal, which outlines the new features and functionality that are to be included in Brigade 2.0.

Primary Research:

User comments in GitHub forums are the primary research source identified in the proposal. Usability issues are a common complaint by users in interacting with the CLI. Brigade 2.0 hopes to address those usability issues. Users also observe that the Kashti UI's read-only implementation restricts its usefulness in achieving the diverse use-cases of the application.

Secondary Research:

As a secondary research source, the Kashti 2.0 team will take inspiration from the current implementation of Kashti and build upon it. The following similar event-driven scripting platform UIs could all serve as potential secondary research sources for our design of Kashti 2.0's UI:

- Kubernetes web-based dashboard
- Docker Swarm's Swarmpit
- Hashicorp's Nomad
- DC/OS

Value Proposition:

Creating Kashti 2.0 will offer value to Brigade users because it will increase accessibility to Brigade's functionality for a wider customer base. Implementing a browser-based UI alternative to Brigade's CLI increases user involvement, allows for visual analysis of running jobs and resources, and makes it easier to communicate abstract actions to non-technical personnel.

In other words, as outlined in Open Professional Group's "The Importance of User Interface", Kashti 2.0 will improve the following characteristics of the application:

- Clarity
- Conciseness
- Familiarity/Intuitiveness
- Responsiveness
- Consistency
- Attractiveness
- Efficiency
- Fool-proofness

See "The Importance of User Interface" for more information (Open Professional Group, 2020).

Constraints:

In order to succeed in this endeavor, the Kashti 2.0 team will need the support of the Brigade 2.0 developers as the web UI's functionality is built to compliment the CLI's functionality through the use of an API. Therefore, as outlined in Brigade 2.0's development plan, the Kashti 2.0 team requires the completion of Brigade 2.0's API by the end of December 2020.

Throughout the development process, the Kashti 2.0 team will find success by meeting regularly with the sponsors to re-sync expectations and provide coaching to improve the quality of the product. Furthermore, the Kashti 2.0 team will need access to development environments where Brigade is running which shall be funded by the sponsor.

Requirements:

Stakeholder Requirements:

The requirements listed below by the groups they pertain to are determined by the information provided in the sections above.

- The Kashti 2.0 Team
    - The Kashti 2.0 team needs to build a knowledge base in the relevant technologies for building the product. Some of this knowledge that is based in Brigade 2.0's implementation will need to be obtained through interactions with the sponsor and coach.

    - The Kashti 2.0 team needs sufficient access to development resources to build the product. This means that development environments, software licenses (if necessary), learning opportunities, and necessary business relationships must be provided by the sponsors for the successful production of Kashti 2.0.
    - The Kashti 2.0 team needs to deliver a fully functional Angular UI (Kashti 2.0) by the end of the second semester
    - The Kashti 2.0 team needs to deliver a fully functional TypeScript SDK by the end of the second semester
- The Sponsor
    - The project sponsors need clear communication channels with the Kashti 2.0 team. This means that they need an available point of contact within the student team and access to any communication channels that the students use (such as Slack workspaces and Zoom meetings).
    - The project sponsors, who will develop the Brigade 2.0 API, will need feedback from the Kashti 2.0 team to help them include features in the API that will improve its integration with the UI.
- The Coach
    - The coach, like the sponsor, needs access to clear communication channels with the Kashti 2.0 team and the sponsor.
- The Customer
    - The customer needs access to the development progress to offer feedback. This means that the GitHub repo and any planning resources related to the product should be made public so it can be accessed by the open-source community.

Use-Cases:

Use-cases for this product include:

- Scripting together multiple tasks and executing them inside of containers
- Building cron jobs using Kubernetes Cronjob to trigger various events at appointed intervals
- Creating fully-automated testing and continuous integration, due to native Git repositories support, meaning pull and push requests can be triggers for various jobs
- Outlining workflows for processing Big Data, with fan-out/fan-in or map/reduce being the simplest and most common of the ETL workloads available due to parallel and serial execution
- Batch processing, like running a data analysis or automated in-container backups
- Building own event gateways for various tasks like sorting and responding to incoming emails, monitoring the issue queue

Use-cases above were described in Itsvit's article, "Brigade: the new Kubernetes management tool from Microsoft" (Itsvit, 2017).

Deliverables:

Major Components and Functions:

Major components and functions to be delivered in the project include:

- A High Fidelity prototype completed by the end of the first semester
- A presentation to shareholders outlining project progress performed at the end of the first semester
- A publicly accessible, web-based UI built in Angular with OpenID Connect that supports Brigade 2.0 integration

Task Breakdown:

There are several tasks and milestones that need to occur over the project's life cycle. These tasks are outlined below:

1. Build a knowledge base regarding technologies that will be used (as per Armour)
    a. Front-end/Design specialists focus on Angular
    b. Kubernetes/SDK Production specialists focus on Kubernetes and Producing an SDK
    c. Security specialists focus on OpenID Connect
    d. Chief Architects focus on Brigade 2.0 implementation/business requirements
2. Develop a Low Fidelity prototype that satisfies all business requirements
3. Iterate upon the Low Fidelity prototype to create a High Fidelity prototype. The High Fidelity prototype will include assets which can be used in the production of the real application
4. Map all functionality of Brigade 2.0's completed API to a Kashti 2.0 feature via a developed TypeScript SDK
5. Prepare a presentation that showcases the Kashti 2.0 High Fidelity prototype and all previous work processes that have occurred up to this point
6. Present the Kashti 2.0 High Fidelity prototype and work accomplishments at the end of the first semester
7. Assign short-term and long-term tasks for individuals to focus on areas of the app to deepen their understanding of business requirements

8. Build a Minimum Viable Product that can be deployed as the Kashti 2.0 UI to support Brigade 2.0's functionality
9. Gather feedback on features and vulnerabilities of the application and iterate upon the existing product to implement improvements
10. Create/maintain documentation regarding the installation/usage of Kashti 2.0 and make it available as part of the GitHub repository
11. At the end of the project timeline (second semester), package the latest Kashti 2.0 implementation for wide-scale deployment
12. Prepare a final presentation/walkthrough of the finished Kashti 2.0 product and all work that occurred through the project life cycle
13. Present the finished Kashti 2.0 product and all lessons learned at the Capstone-ending presentation
14. Deploy the Kashti 2.0 product and turn over ownership to the sponsors and shareholders

"Is/Is Not" Exercise:

To define the scope, this project IS the following things:

- Kashti 2.0 is a web-based UI that allows for interactive use of the Brigade 2.0 API
- Kashti 2.0 is the UI element for Brigade 2.0, which is an event-driven scripting framework
- Kashti 2.0 allows for read/write actions to take place for certain operations that are included in the Brigade 2.0 CLI feature set
- Kashti 2.0 is an open-source project sponsored by Microsoft and will be maintained by the open-source community after the end of the Capstone class timeline

To realistically limit the scope, the project IS NOT the following things:

- Kashti 2.0 is not a web-based re-creation of Brigade 2.0
- Kashti 2.0 and Brigade 2.0's use-cases are not limited to CI/CD pipelines

Delivery Mechanism:

Once complete, the product and deliverables will be delivered in two stages. First, after the first semester is completed, the major deliverables will be outlined in a presentation to the major stakeholders of the project. This includes a high fidelity prototype of the Kashti 2.0 UI, and a clear plan of how the UI should interact with the Brigade 2.0 CLI to offer functionality.

After the second semester is completed, the finished Kashti 2.0 UI will be given to Microsoft and the open-source community in a GitHub repository. It should be noted that throughout the project life cycle, development resources will be made publicly available in the GitHub repository even before the project is complete.

Documentation:

Because this is a largely front-end product, the presentation at the end of the semester will include a walkthrough of the high fidelity prototype. We will also outline our work processes leading up to the presentation at the end of the first semester. We track this process and any tasks by using the GitHub Projects page below:

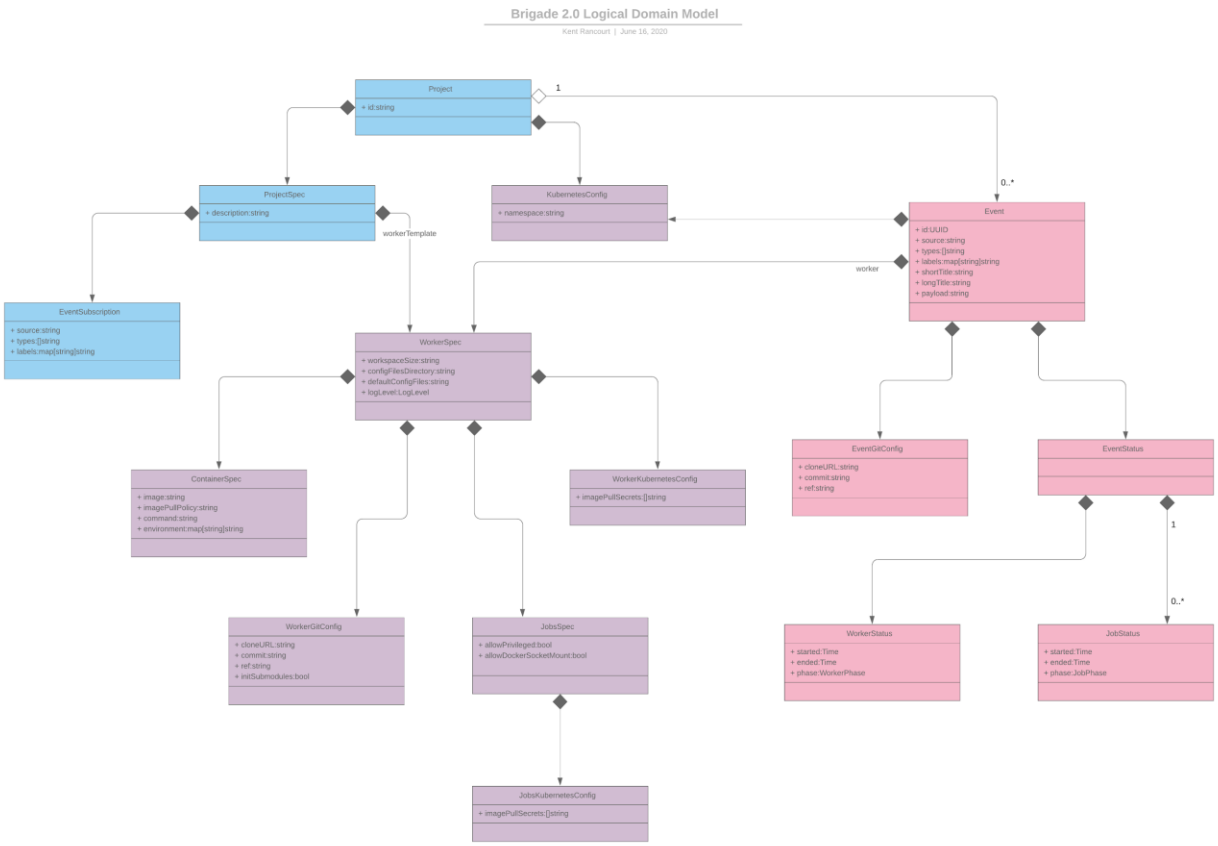
- https://github.com/orgs/kashti-byu-capstone/projects/2

Our high fidelity prototype will be built using Angular. This Angular prototype will be presented at the end of the first semester.

After the second semester, the product will be complete and deployed as a UI that can be used by Brigade 2.0 customers. The front-end product will be built with Angular, implement OpenID Connect for authentication, and will be fully integrated with the Brigade 2.0 API via a TypeScript SDK. All documentation for the Kashti 2.0 UI will be included in the GitHub repository in the ReadMe.md file.

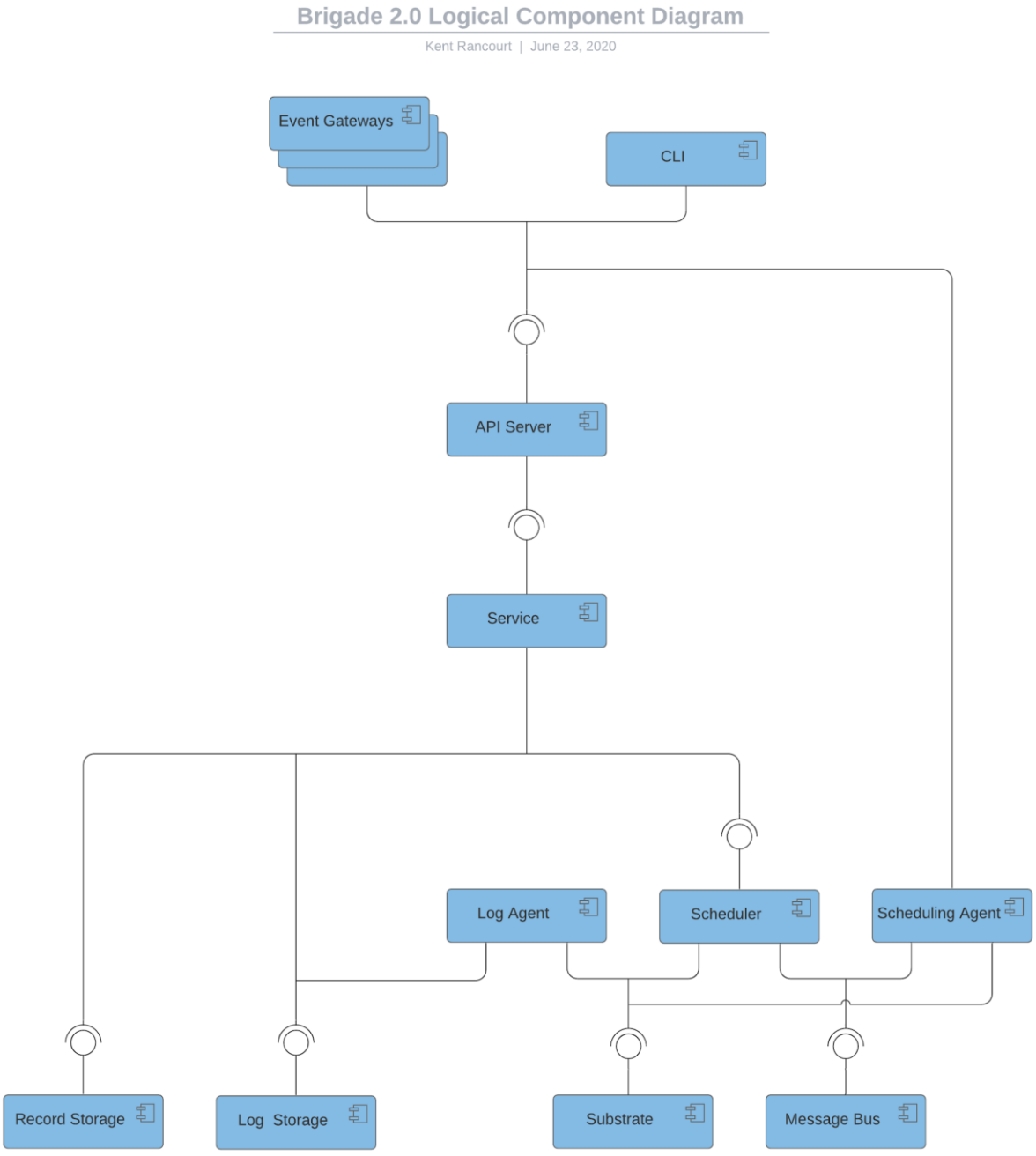
Logical Design:

An architectural diagram of Brigade is given in the Brigade 2.0 Proposal (provided by Microsoft). The organization of the way that Projects, Events, Workers, and Jobs interact with one another can all be described in the Logical Domain Model created by Microsoft. In summary, the highest level data object in the organization of the project is the Project object. Projects can have any number of Events that are associated with the Project. Events can have Jobs and Workers associated with them and are the lowest level of data objects in the structure of how these objects interact with one another. The Logical Domain Model is provided below:



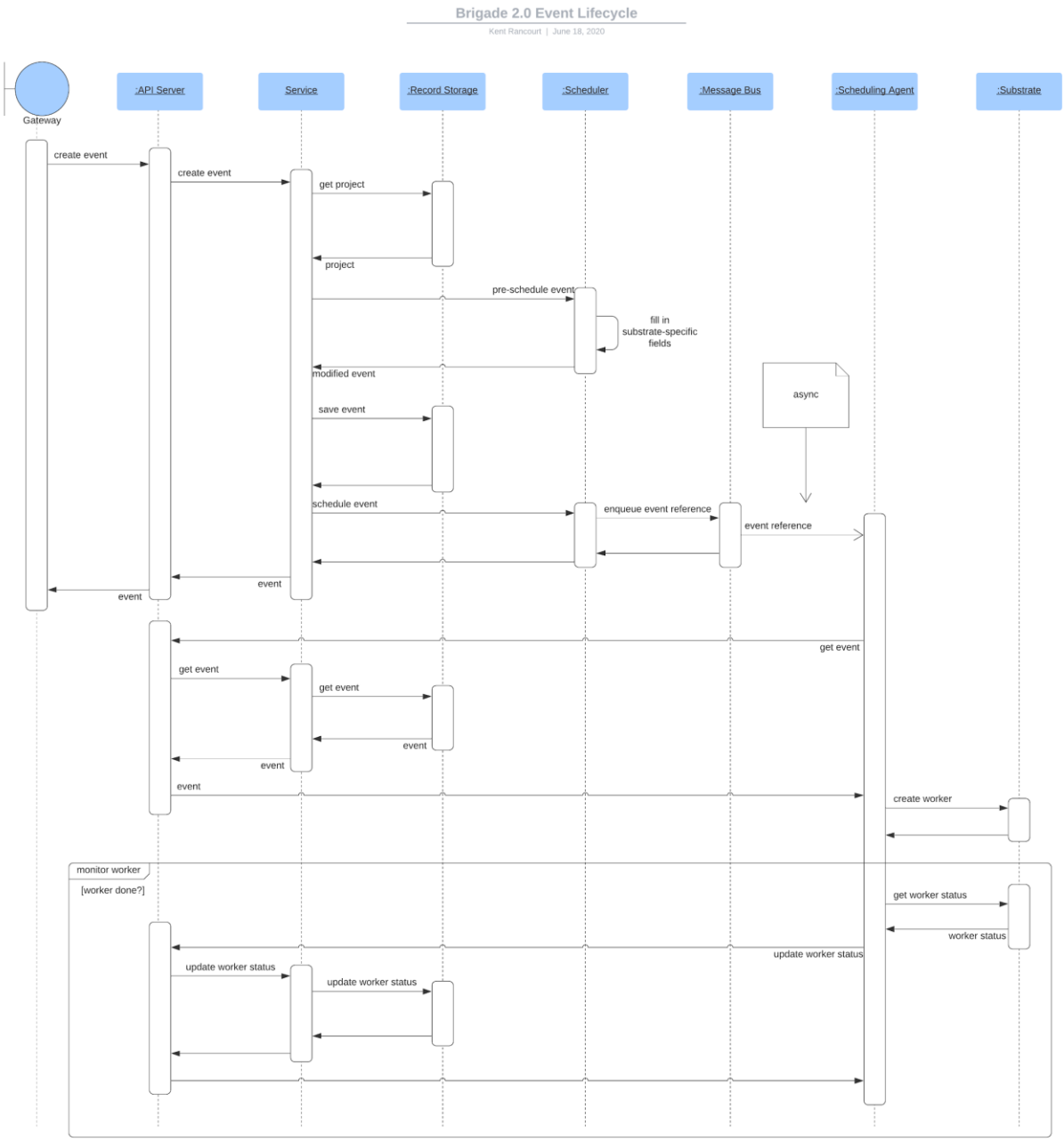
Brigade 2.0 Logical Domain Model

Using the Logical Domain Model provided above, a Logical Component Diagram details the way that Brigade connects with users and the services that it contains in order to perform its functionality. In short, Event Gateways or user-given CLI input is given to trigger functionality by calling the API server. The API server triggers the Brigade service to respond to the event/input and scripts are run, logs are produced, and responses are provided. The Logical Component Diagram is provided below:



Logical Component Diagram

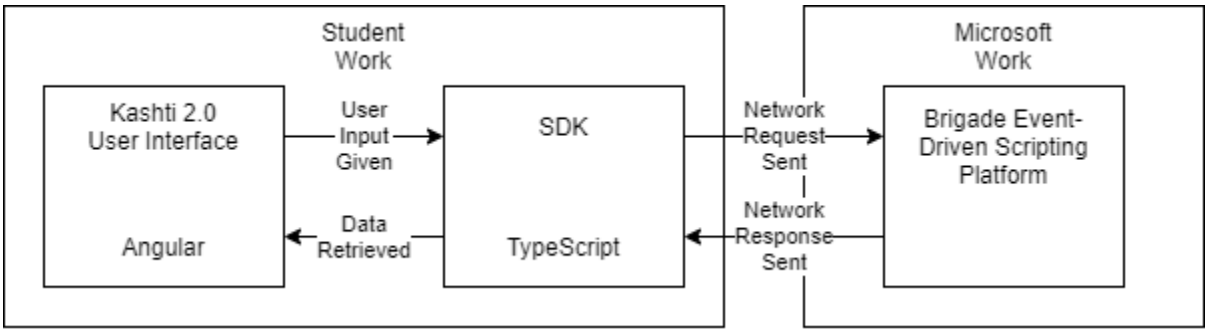
The last architectural diagram to describe the work of Brigade explains the Event Lifecycle. This diagram explores the workflow from the Logical Component Diagram in more detail by providing which services interact with one another and in what order of the workflow. The Event Lifecycle is provided below:



Event Lifecycle

Brigade is only part of the project's full architecture, and that part of the project is already taken care of by Microsoft. The students' architectural contribution lies in the addition of a TypeScript SDK and a User Interface that makes calls to Brigade's API Gateway. In relation to the Logical Component Diagram above, Kashti 2.0 would act as one of the Event Gateways by accessing the API and making calls to create, read, update, or delete items in the Brigade platform. That relationship is described in the Kashti 2.0 Architecture Diagram. In short, users interact with the Kashti 2.0 UI by requesting data or performing operations. The Angular UI leverages the TypeScript SDK to perform API calls to the Brigade platform, then a response is

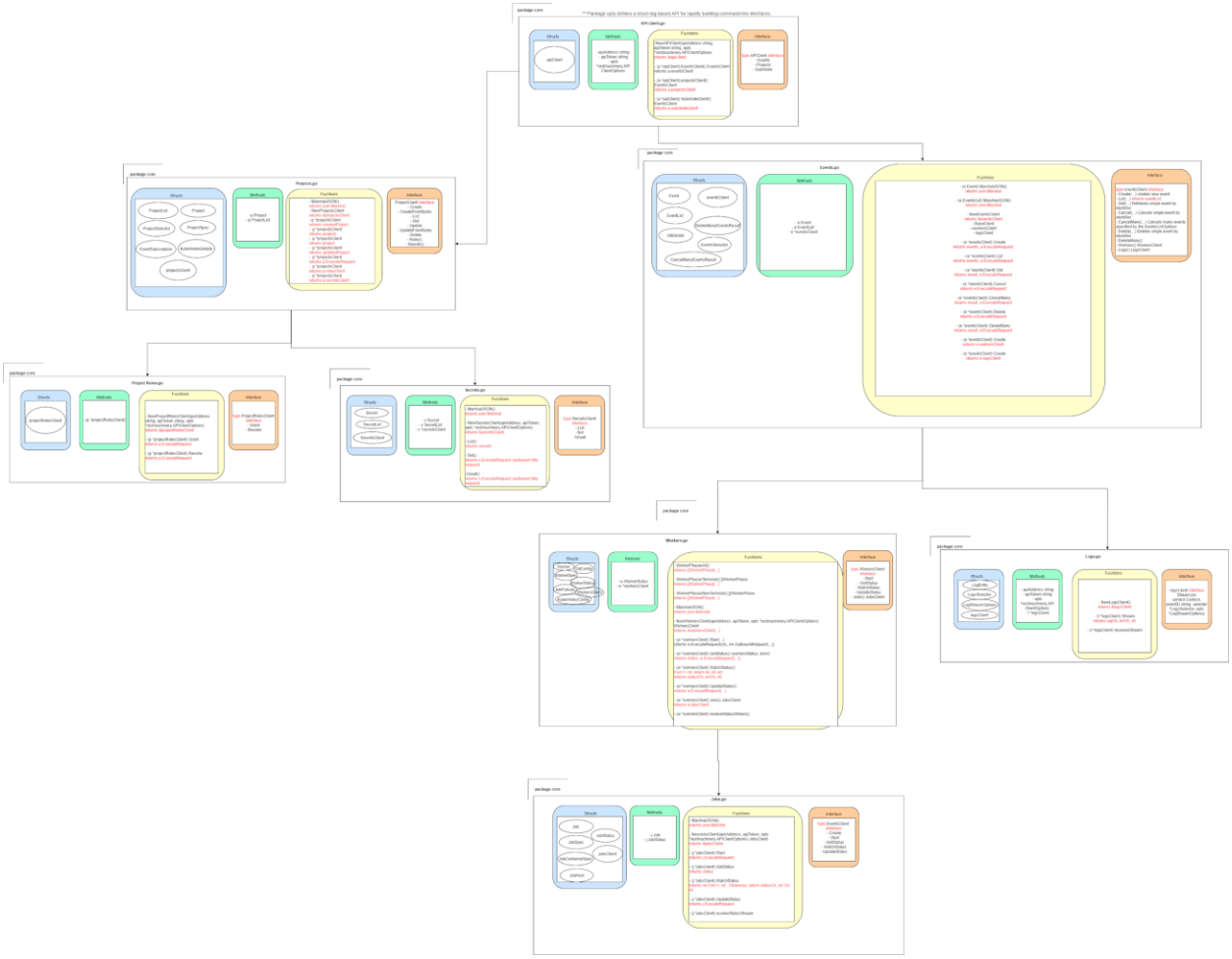
returned to the SDK and subsequently the UI for a seamless user experience. The full architecture is explained in this YouTube video. The Kashti 2.0 Architectural Diagram is given below:



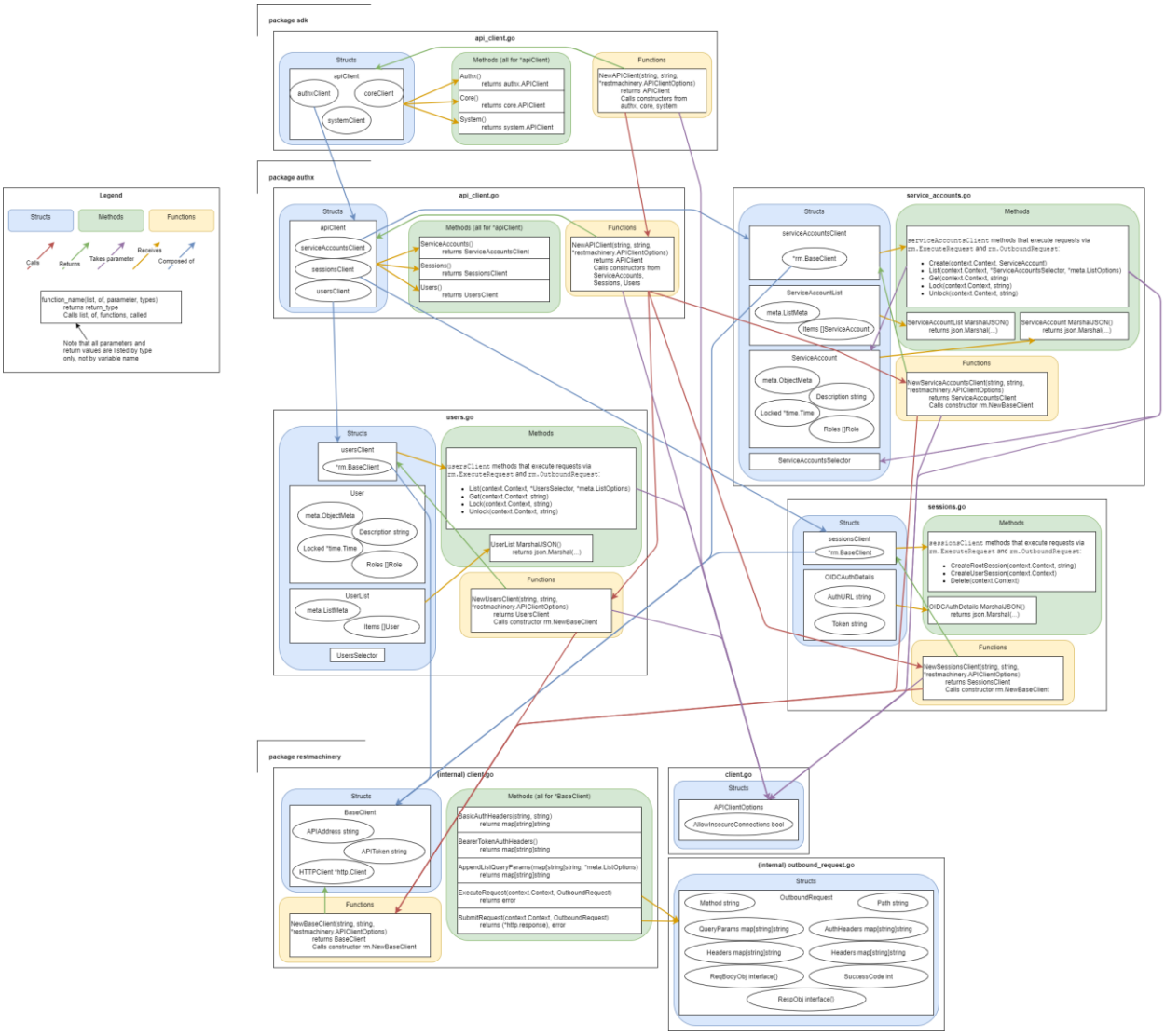
Kashti 2.0 Architecture Diagram

Another aspect of the project architecture that must be understood is the way the SDK is organized. The SDK has a few different API clients available that access different API groups on Brigade. The Core client handles API calls that relate to Brigade's core functionality (Projects, Events, etc.). The Authx client handles API calls that relate to all authentication functionality (login/logout). Just like explained in the Kashti 2.0 Architecture Diagram, these API clients simply serve as intermediaries between the UI and the Brigade platform. Commentary on our work regarding the SDK is available in this YouTube video. The architecture of these API clients are given in the diagrams below:

Core Folder



Core API Client Architecture Diagram

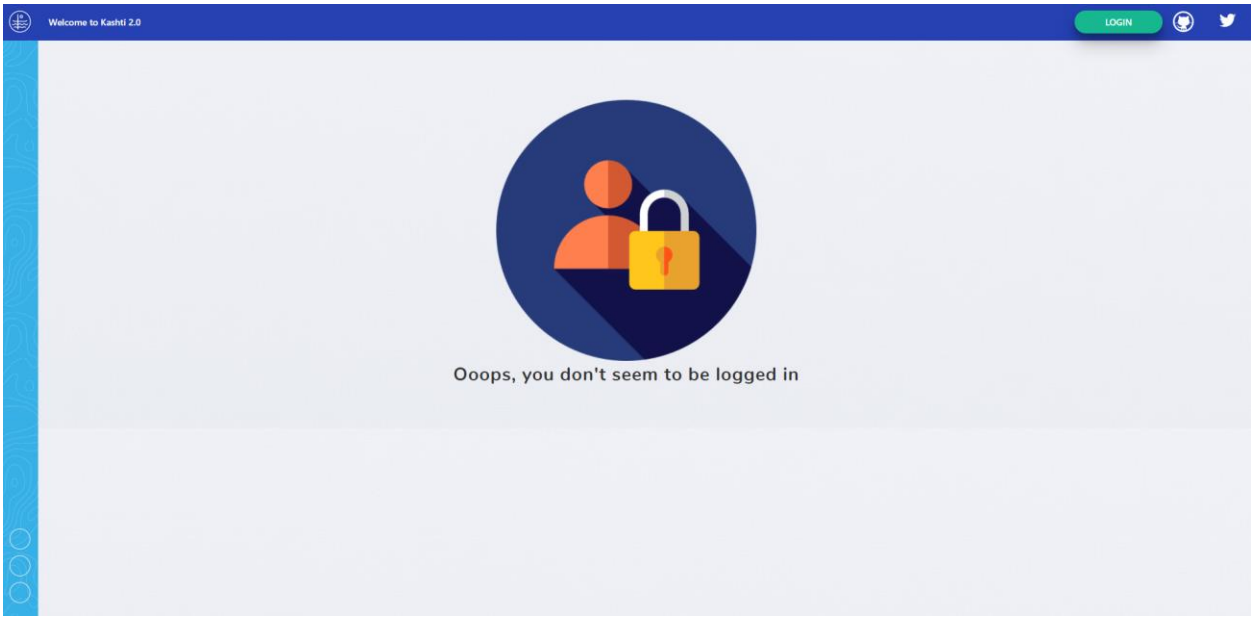


AuthX API Client Architecture Diagram

The diagrams given in this section effectively detail the logical architecture of the Brigade backend, the SDK, and the full Kashti 2.0 system.

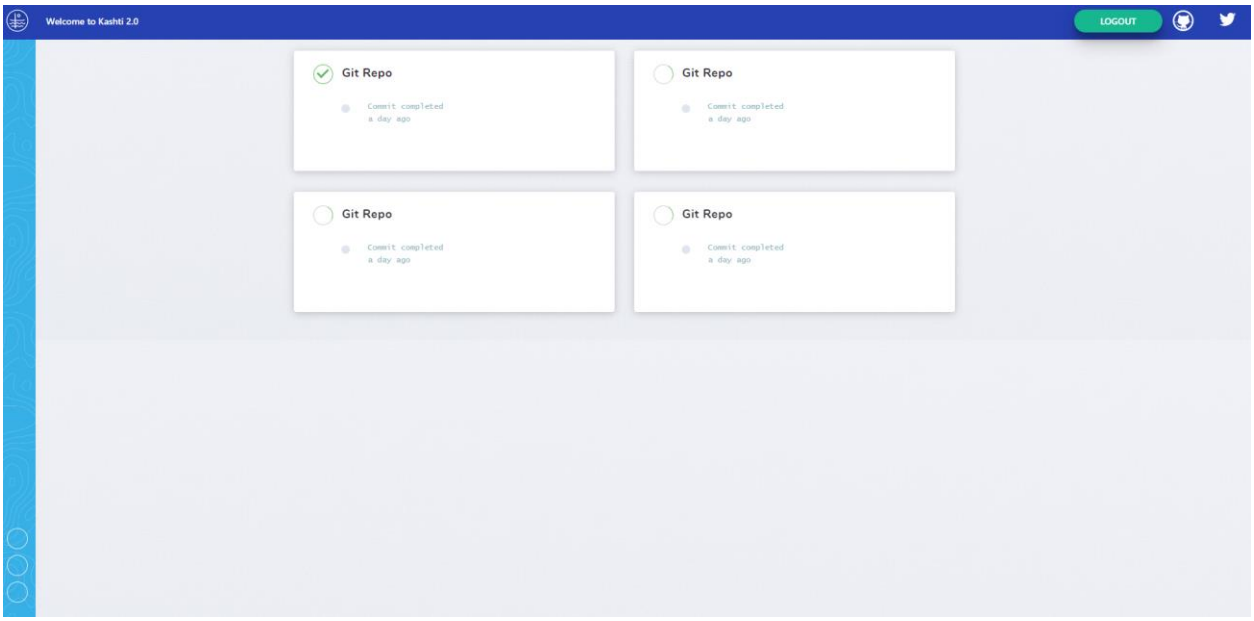
Prototype Implementation:

The Angular UI Prototype is available at the Kashti 2.0 Team's Github Repo. The pages from this prototype are given below. When the project is first accessed, the user is prompted to login.



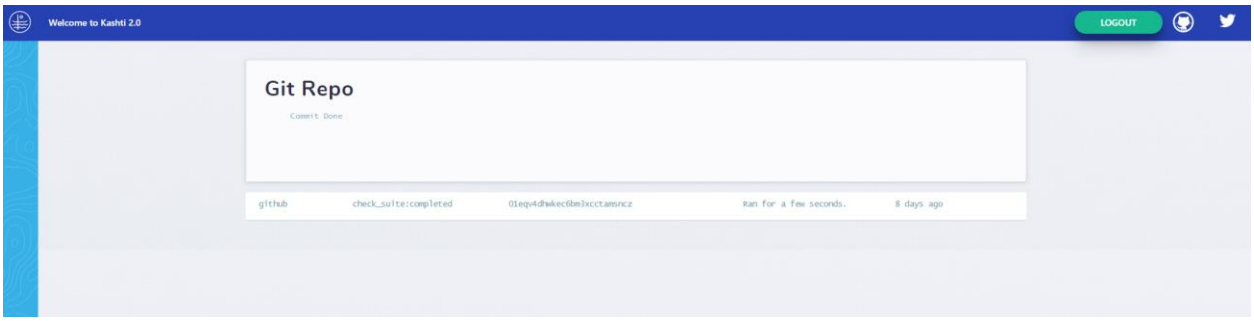
Login Page

After logging in, the user is brought to the Projects page where he/she may access the different configurations in place for their event-script pairs.



Projects Page on Prototype

The user can view the list of all Events that have occurred on a project by clicking on any of the Project tiles on the Projects page.



Project Page on Prototype

The user can view the details of any single Event by clicking on the Event from the list of Events that are displayed on the Project page.

The team has many potential ideas to implement in the second semester which include ideas from the wireframes provided in the appendix (item 1).

The SDK also has a prototype available for testing. The SDK is downloadable from NPM as the “kashti-sdk” package. This package is viewable on NPM. A demo of how the package can be used in a sample app is available by watching this YouTube video.

Cybersecurity Risks:

Kashti 2.0’s authentication is handled by retrieving an auth token from Brigade and including that token in the request header of requests. Any request without a valid auth token which was generated and stored within the Brigade instance will be rejected. Most of the cybersecurity risks associated with the project, therefore, have to do with maintaining the privacy and integrity of this auth token.

The auth token is retrieved at the time of login and stored in local storage in the browser that is rendering the Angular Kashti app. Communications over the network take place using HTTPS protocol, meaning that the encrypted auth token/JSON data can be expected to be secure while in transit. The auth token is not used for anything on the client side other than to be included in the payload for when requests are sent to Brigade, meaning that the auth token has significantly low risk of being compromised while in use. The area of largest concern is the state of the auth token and its security while it is at rest in local storage. Being a web app, Cross-Site Scripting (XSS) is a major concern that should be prevented. See this YouTube video for commentary on our security posture regarding Kashti 2.0 and XSS.

A successful XSS attack could compromise the auth token and lead to session hijacking. Therefore, every step of our process integrates a level of Quality Assurance led by the cybersecurity expert on the team. This practice will minimize the risk of XSS vulnerabilities unknowingly making their way into how the Angular Kashti 2.0 app is created.

The SDK, because it is simply a bridge between Kashti and Brigade, does not maintain any data on its own. Therefore the security risk associated with the SDK is small. We will

prevent the SDK from being misused by allowing very little customization (you can only give the url for your Brigade instance) and hard-coding values to hit Brigade's API endpoints specifically.

References:

D2IQ (2020). *Using the UI for DC/OS management.* Web page. https://docs.d2iq.com/mesosphere/dcos/2.1/gui/

De La O, Dunn, Kendall, Sundrud, Uhi (2020). *Kashti 2.0 Project Repo*. Web page. https://github.com/orgs/kashti-byu-capstone/projects/2

Dunn, Derryck (2020). *Kashti 2.0 Architecture.* Video. https://youtu.be/WV08FnODxlY

Fedak, Vladimir (October 2017). *Brigade: the new Kubernetes management tool from Microsoft.* Itsvit 2017. Web page. https://itsvit.com/blog/brigade-new-kubernetes-management-tool-microsoft/

Google (2020). *Angular Home Page.* Web page. https://angular.io/

HashiCorp (2020). *Explore the Nomad 1.0 Beta.* Web page. https://learn.hashicorp.com/nomad

Kendall, Stephen (2020). *TypeScript Brigade SDK.* Video. https://youtu.be/mKcwQ2yh6Bs

The Kubernetes Authors (September, 2020). *Web UI (Dashboard).* Web page. https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

OpenID (2020). *Welcome to OpenID Connect.* Web page. https://openid.net/connect/

Open Professional Group (2020). *The Importance of User Interface.* Web page. https://www.openprofessionalgroup.com/news/importance-user-interface/

Rancourt, Kent (June, 2020). *The Brigade 2.0 Proposal.* Web page. https://github.com/brigadecore/brigade/tree/master/2.0-PROPOSAL

Sundrud, Spencer (2020). *Kashti 2.0 Security.* Video. https://youtu.be/JDY0UHvBPUQ

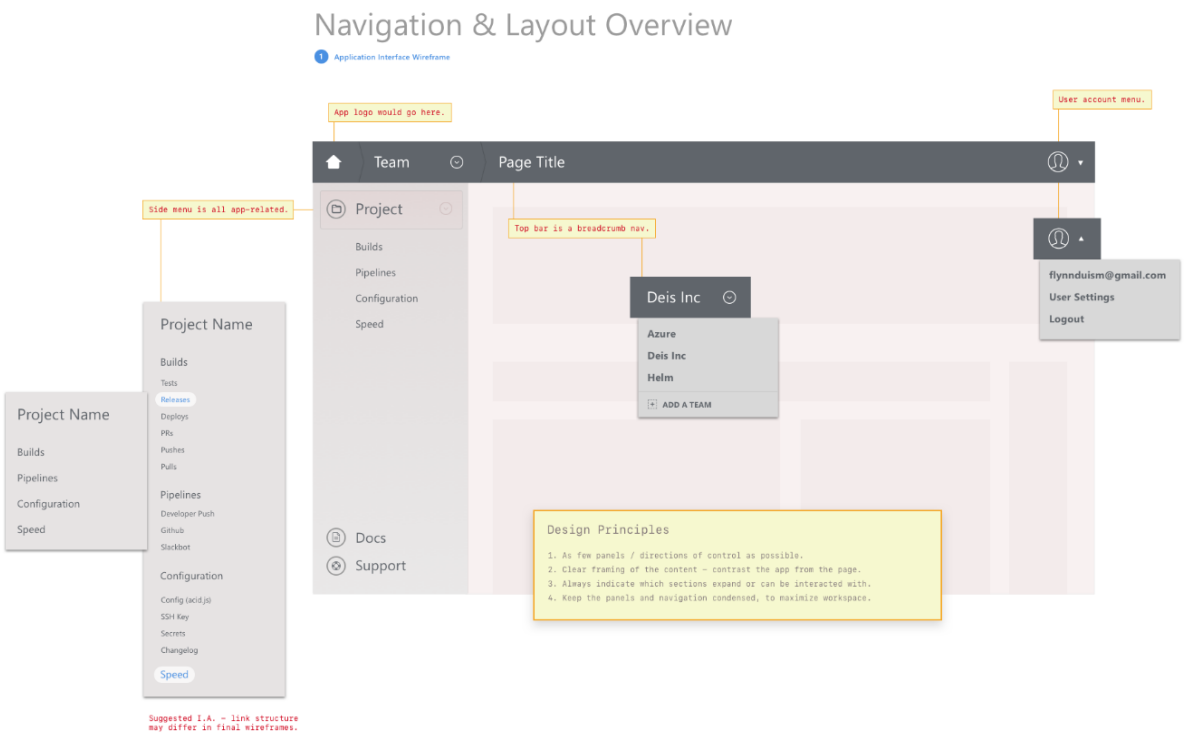
Swarmpit (2020). *Swarmpit web user interface for your Docker Swarm cluster.* Web page. https://dockerswarm.rocks/swarmpit/

Uhi, Cody (2020). *brigade-sdk NPM Package.* Web page. https://www.npmjs.com/package/kashti-sdk

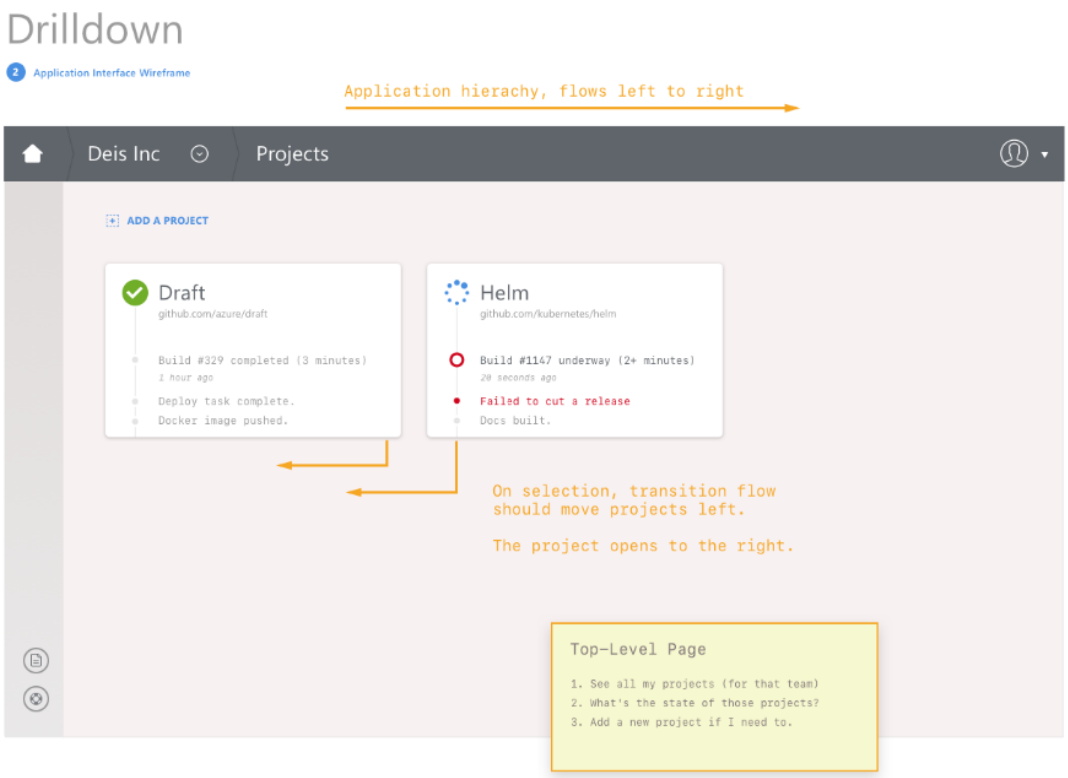
Uhi, Cody (2020). *Demo of Brigade TypeScript SDK.* Video. https://www.youtube.com/watch?v=PEfIbbLN9mM

Webster, Bruce F. (2020). *Presenting on 'The Five Orders of Ignorance (Armour)'* Web page. http://bfwa.com/itc446/2020f/itc446-200916-armour.pdf

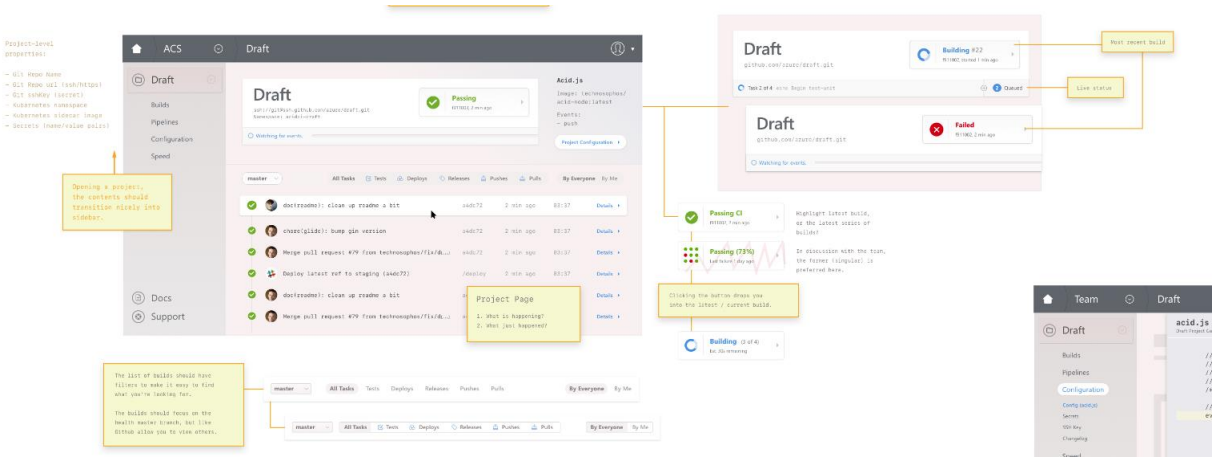
Appendix:



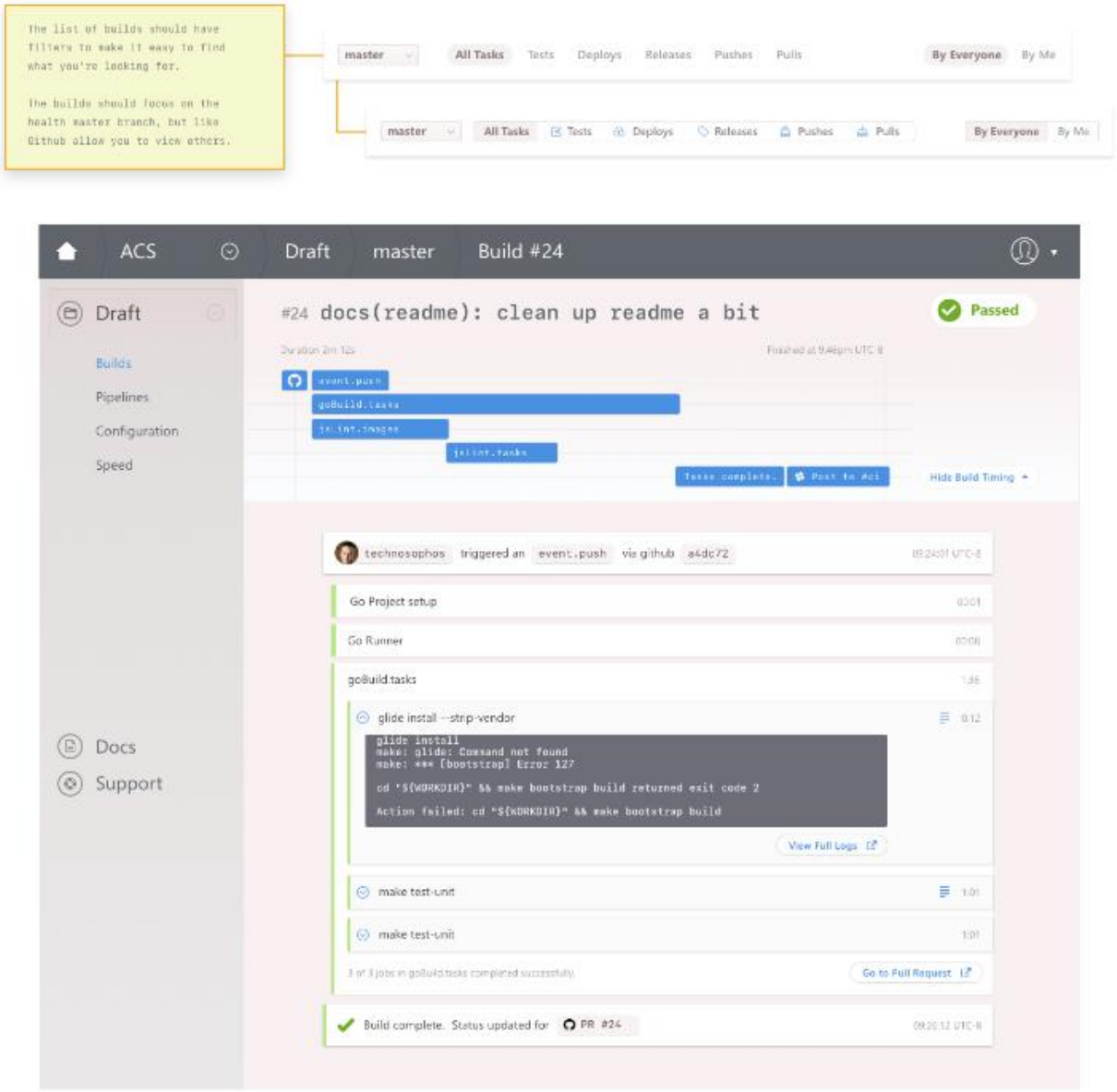
1a. Navigation & Layout Overview Wireframe



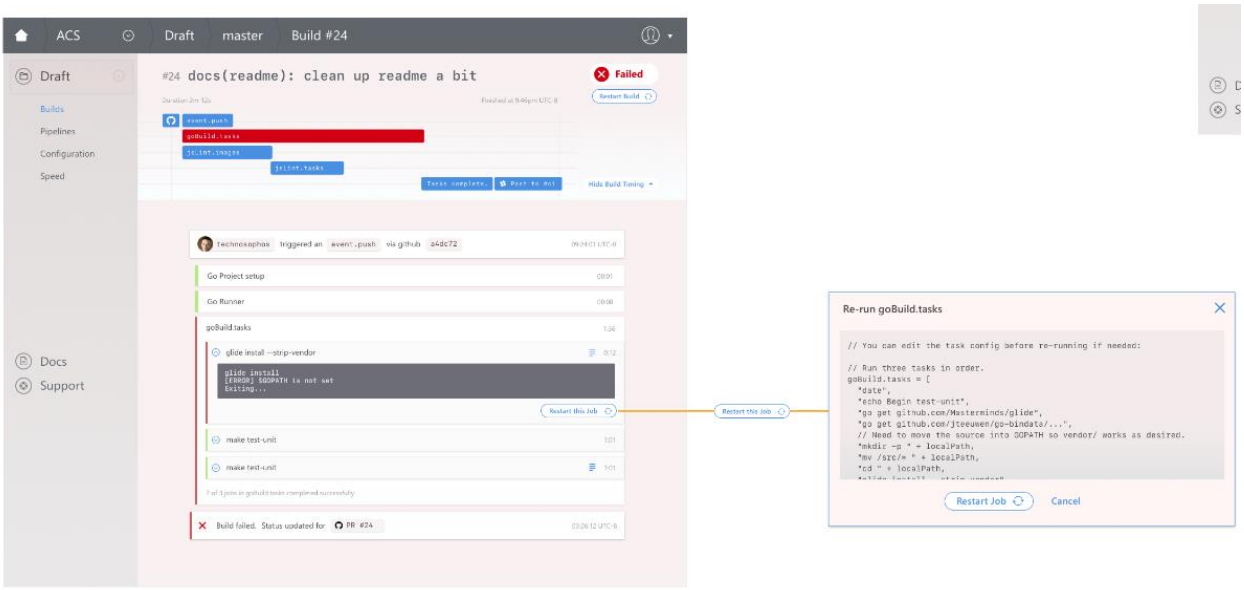
1b. Projects Page Wireframe



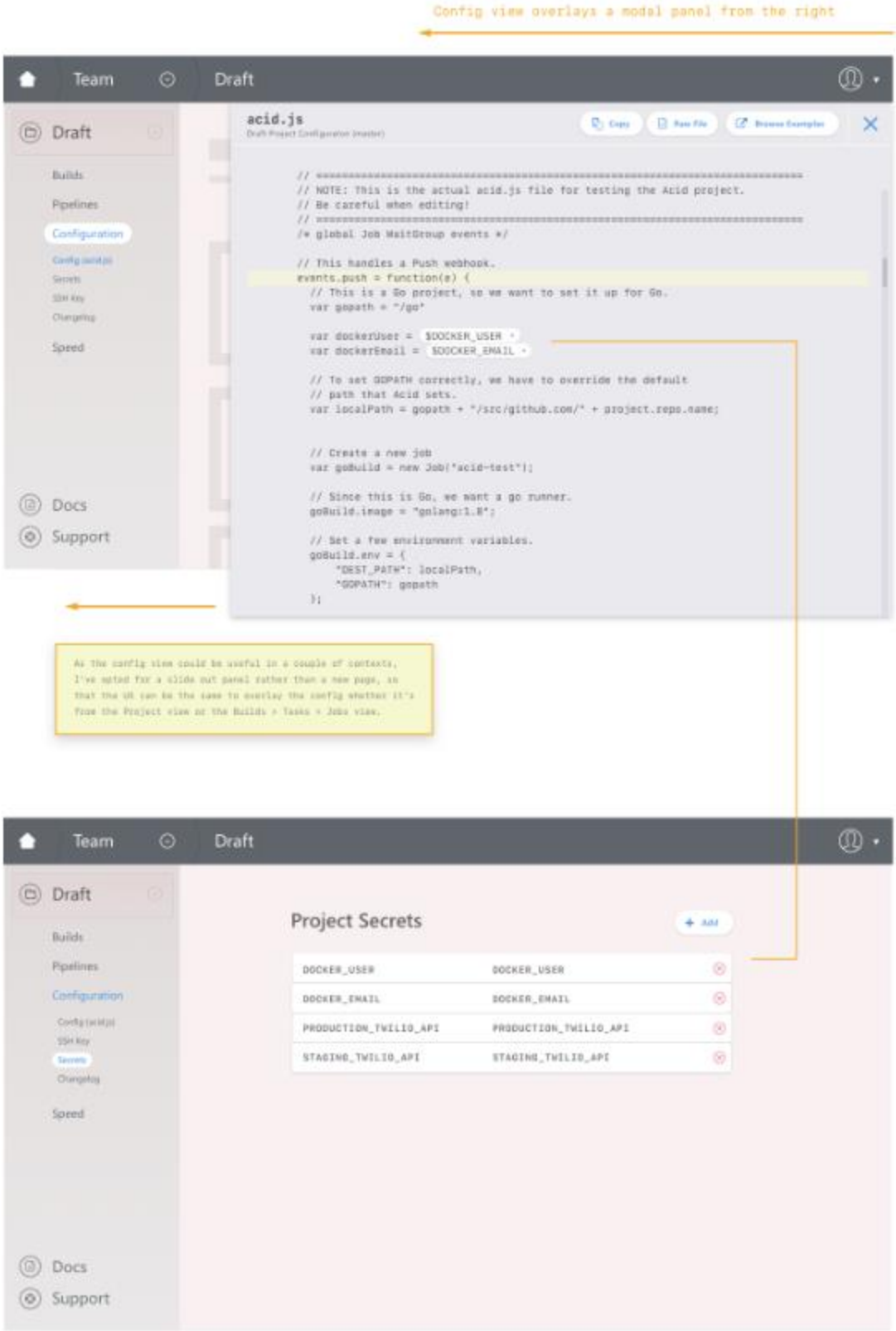
1c. Project Page Wireframe



1d. Event Page Wireframe



1e. Worker Page Wireframe



1f. Job & Secrets Page Wireframe