Power Tools for Pivotal Tracker

Pivotal Labs
Dezmon Fernandez
Victoria Kay
Eric Dattore

June 16th, 2015

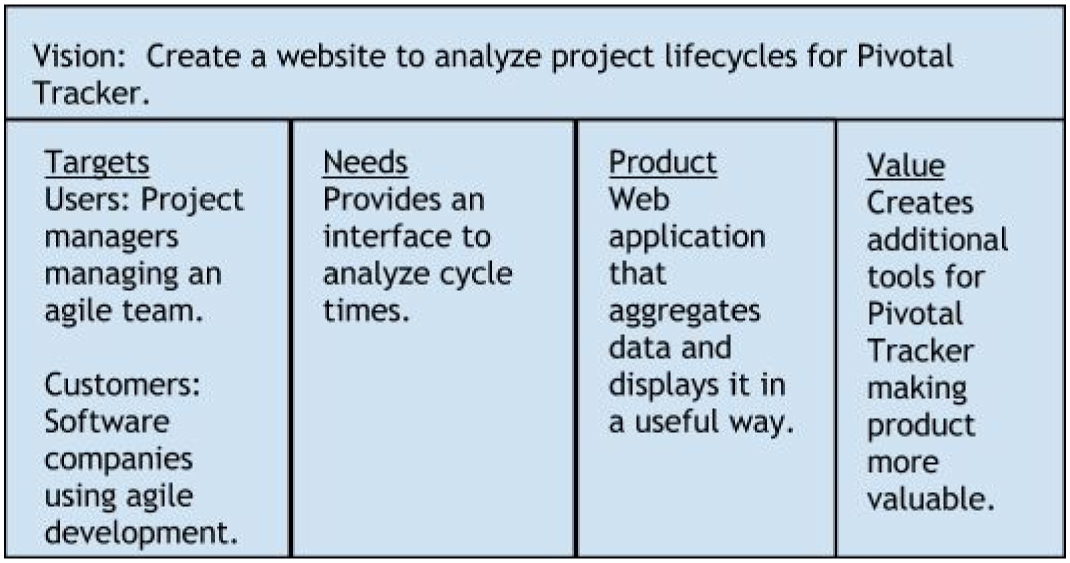
Client Description

Pivotal Labs is an agile software development company focused on the agile process. They focus on helping companies improve productivity and develop products efficiently. Doing daily standups, weekly or bi-weekly retrospectives, delivering software often, and making use of pair programming demonstrates Pivotal Labs' commitment to the agile development process.

They are the creators of Pivotal Tracker, a real-time project management tool used to help teams of software engineers keep track of their progress and collaborate effectively. Tracker displays all the stages that a project goes through in an organized way, from research, to coding, to debugging, to final release. Projects are broken down into small, manageable stories that are then prioritized. Tracker helps teams estimate their productivity by looking at progress from previous weeks to predict how much work can get done in the future. It also gives a timeline of when the project will be completed. While working with Pivotal Labs, we used Tracker to plan our project and keep everyone up to date on the progress being made.

Product Vision

At the beginning of the 2015 summer field session, our team met with Pivotal Labs to discuss what product was right for them. After our initial discussion the team understood that Pivotal wanted a product that interacted with the Pivotal Tracker; however, we were not entirely sure what direction we needed to head until after joining the Pivotal team the first week. After further discussion with our client, we realized that we needed a product that would aggregate the data retrieved from Pivotal Tracker. We needed an application that could interpret this data and display useful information for project managers. Our goal was to develop a website that would display cycle times for projects and illustrate the dynamics of a project. Figure 1 illustrates what the product vision is at a high level.



Vision: Create a website to analyze project lifecycles for Pivotal Tracker.			
<u>Targets</u> Users: Project managers managing an agile team. Customers: Software companies using agile development.	<u>Needs</u> Provides an interface to analyze cycle times.	<u>Product</u> Web application that aggregates data and displays it in a useful way.	<u>Value</u> Creates additional tools for Pivotal Tracker making product more valuable.

Figure 1

Requirements

Our client wanted us to develop the Power Tools component of Tracker. Tracker records all the state changes of tasks inside the app, however, there was no way to view and analyze these records. The Power Tools web app we were asked to build aggregated the information and allowed for the analysis and display of the raw data. Due to Pivotal Labs' lack of current tooling, many other companies had formed around harvesting the data from Pivotal Labs' API endpoint and displaying it for their customers. One such tool is Insight by SynApps (http://in-sight.io). The development of our tool helped Pivotal Labs increase the utility of their product.

Functional Requirements

There were several components to the Power Tools for Pivotal Tracker Project: a web page, a data model, and interaction with the Tracker itself.

Webpage

The website was built with two entries which accepted an API Key and a Project ID. Once these inputs were accepted, the website displayed a table with the following columns for a single project:

- Story ID
- Lead Time
- Engineering Cycle Time (ECT)
- Acceptance Cycle Time (ACT).

This was accomplished in several iterations. The first iteration was a simple web page displaying this information in a table format. Once the core functionality was completed, further iterations focused on styling and the actual display of the data.

Data Model

The data model was a database, which delivered data to the website such that the table can be displayed. The project ended up with a thin server and a fat client model. The raw data was sent to the client in JSON format and any operations on that data were performed on the client side, since client-side calculations are almost always faster than making HTTP requests.

Pivotal Tracker Interaction

The Pivotal Tracker was the source of all information pertaining to state changes and task updates. The Pivotal Tracker contained massive amounts of information for different projects. The problem with this was, as mentioned before, results took a very long time to retrieve from the API endpoint. The Pivotal Tracker API was at our disposal, so the synchronous process performed the complicated task of interaction with the Pivotal Tracker and fast data retrieval.

Figure 2 shows a high level diagram of the functions connected.

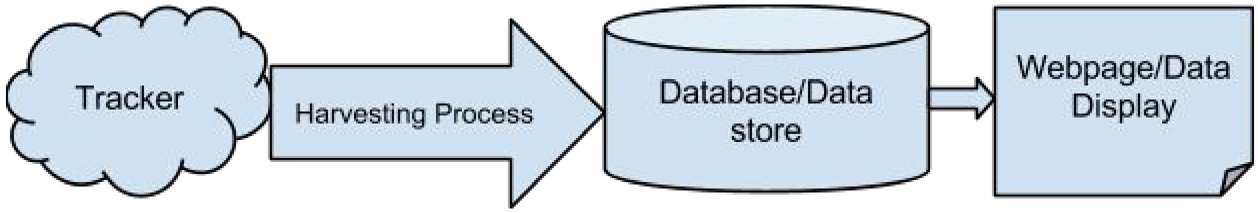


Figure 2

Non-Functional Requirements

We practiced agile-like development and worked in an open office to encourage collaboration. We also tried pair programming, however, with three people, one person was always programming alone.

Tooling

HTML, CSS, and JavaScript were used to build the website. The first version of the website was very simple and clean, and then once the basic version was completed, the focus shifted to the aesthetics of the website and adding more features. We added graphs to display the data in interesting ways as well.

Project Management

The Tracker was used to manage this project's stories and progress. This ensured that the project was getting done at a good pace and everyone was on the same page.

Data Format

Data was stored using JSON to make it easier to use. This also enabled us to pull the exact data we needed to display. The data was fixed in the beginning to get the functionality of the website working, then we pulled the data directly from Tracker. Retrieving and sorting this data took a very long time, so we focused on making this retrieval as time-efficient as possible.

Risks

When we came into this project, there were a few risks our team faced. Most of us were not familiar with coding in JavaScript and/or working with databases, so we needed to learn a bit about both of these topics before getting started. We also ran into some problems with the data. We had to find a good way of pulling the data from Pivotal Tracker quickly, so the user did not have to wait a long time to get their cycle time information. Another risk was that there may be problems extracting and interpreting the data in a meaningful way.

Final Product

The final product the client requested we build was a simple web application that harvests their API and returns data containing information from their Tracker application. At a minimum, the web application was to display all the user stories in the project, their name, type (bug, feature, etc), and point estimate (used to estimate complexity of a given user story). The table allowed for viewing further details on the user story to view the elapsed time spent at each stage in the process. In addition, there was information viewable on the average engineering cycle time (elapsed time a user story was worked on), average acceptance time (elapsed time from delivered to accepted), average lead time (elapsed time from creation of the story to accepted), rejection count of user stories, and color-coordinated current state information.

Design

System Architecture

The system architecture consisted of a website, server, database, and the Pivotal Tracker System. The design of the system was to be as independant as possible such that components could be easily replaced. Figure 3 shows a system architecture diagram describing the flow of information.

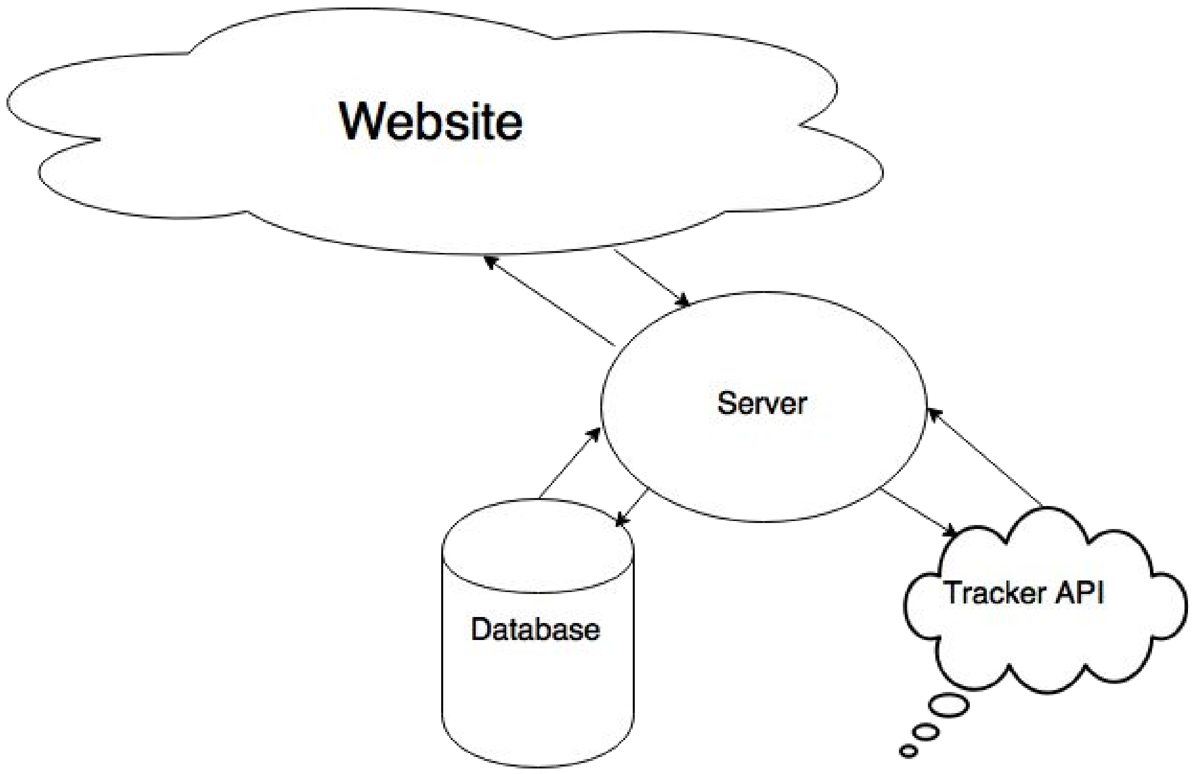


Figure 3

All of the information in our system was a two way street. Everything in front of the database is the front end of our system and handles information by means of requests.The back end dumps information into the database and communicates by restricting information.

Front End Flow of Information:

- Website asks server for information based on API and project ID
- Server pulls info from DB which can be delivered to website through JSON
- Database harvests information which the server queries

Back End Flow of Information:

- DB let's info pour in from synchronous process
- Process communicates with DB to not dump repetitive data
- Pivotal Tracker communicates with process to keep information flow within a range

Database Schema

The database was a simple design since our website displayed only story information and calculates the other required data. Our product used MySQL with two tables. Figure 4 shows an illustration of our schema with the required information.

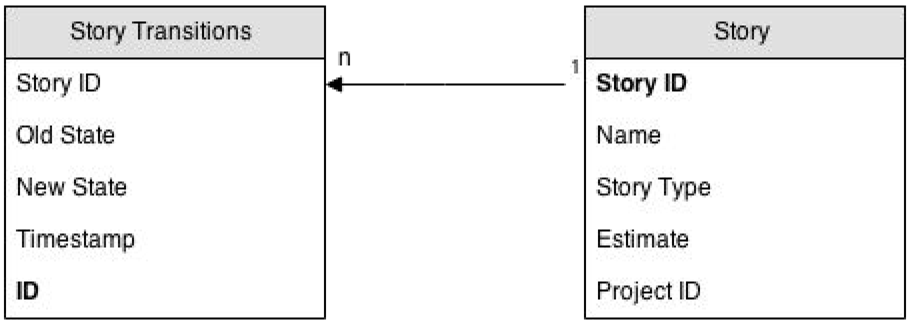


Figure 4

Each table is normalized and has primary keys.There is a foreign relationship between the Story table and the Story Transitions table. The relationship is a one to many relationship since multiple story transitions could have the same story id.

Design Decisions

Website

In deciding how to create the website, we decided to go with Javascript, since our client's company uses Javascript extensively. We also decided to use a Javascript framework called React to build our application. The framework aided the website such that it can dynamically render components without having to reload the entire page. This allowed for interaction between web page components, so one component could change another without affecting the calling component. React also has built-in functionality to calculate what needs to be changed in a way such that changes can be made with little to no lag time. Ultimately, React supported our application's need for a constantly changing environment.

Database

The storage solution for our application was MySQL. Aside from our familiarity with MySQL, there were several reasons we decided to go with MySQL over other databases:

- Easy to incorporate in our deployment scheme
- Fast and known to many

- Free

Aside from the obvious benefit of being free and familiar, MySQL was well suited for our application.

Server

The server side logic of the Power Tools for Pivotal Tracker was supported by PHP and more specifically the Laravel framework. Laravel is a fast, lightweight framework, built to interact with databases as well as deliver information quickly and easily. Once familiar with the framework, it was very easy to use. Laravel and PHP was the easiest tool to use for the server component because of our team's familiarity with the PHP language.

Build

The build system we decided to use was Gulp. Gulp is a Javascript-based task runner that's meant to automate common tasks, such as building or running tests. Gulp has an extensive library of plugins that practically do everything you could possibly need, it just takes a Gulpfile to define the different tasks you want to do. Leveraging a plugin called Browserify, we were able to easily concatenate our Javascript files and compile the React templates we used within our React views.

Deployment

The deployment decision was an easy decision to make since our client has their own internal deployment tool known as Cloud Foundry. By using Cloud Foundry to deploy our web application, we had internal resources that helped us get the job done.

Results

Our application met all of the client's core requirements ahead of time which allowed us to work on stretch goals that we vaguely discussed in our initial planning meeting. The web app was primarily tested in Google Chrome (although some alternate browser testing was done to resolve any cross-browser issues) and we used QUnit to unit test our Javascript calculations.

The major roadblocks for the project were our inexperience with the technologies we used and our deployment scheme. Most of our team had never used Javascript, HTML, or CSS before, so the learning curve was high. After the first week, most of us were on the same page and our pace quickened. We also worked on the project for the first few weeks without a deployment strategy which meant that our client couldn't oversee our progress directly. When we finally deployed the code, we had about a half-day period where we couldn't continue forward because we were waiting for our client to accept the features we had just delivered. It amounted to roughly three weeks worth of work that our client had to look over and accept in order for us to move forward. However, once we had resolved the deployment issue, things went a lot smoother and enabled us to rapidly complete the project.

Conclusion

At the completion of our project, our team had one final retrospective with our client based on our time there. We went through the goods, the puzzlers, and the bads. The retrospective gave our team a good idea of how we can improve in the future and things we should keep on doing. We learned how effective the agile process can be and the benefits of coming together as a team and understanding how everyone works. Our team also learned that we should diversify our knowledge more instead of having certain team members specialize. With every team member on board, it is much easier to help each other problem solve.

As the summer progressed our team started learning from our mistakes and following these principles more and more. By the final week, we were able to knock out nearly all of our stretch goals presented by the client. At the closure of our final meeting, we all got to check out our product and observe how it interacted with other Pivotal Tracker projects. We were all very happy with how the final product turned out. Overall, the team learned many lessons and got to experience what it was like to work in an agile environment at Pivotal Labs. We would suggest students in the future to check out the Pivotal Team, and even check the company out for opportunities. This summer really helped our team develop as young software engineers and learn more about real world experience. The Appendices below show the work we accomplished this summer and screenshots of our final product.

Appendix 1

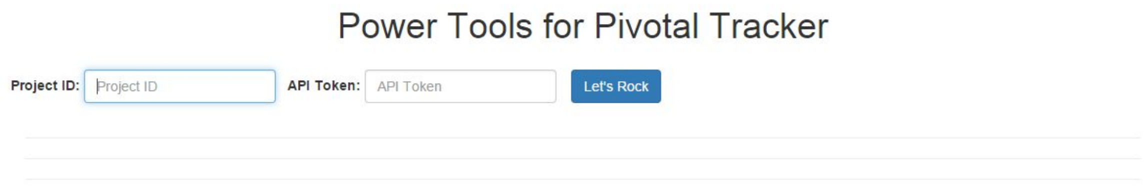
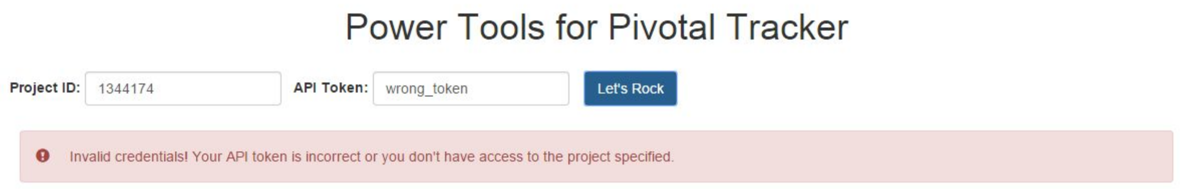


Figure 5, Power Tools web page when it is first opened.

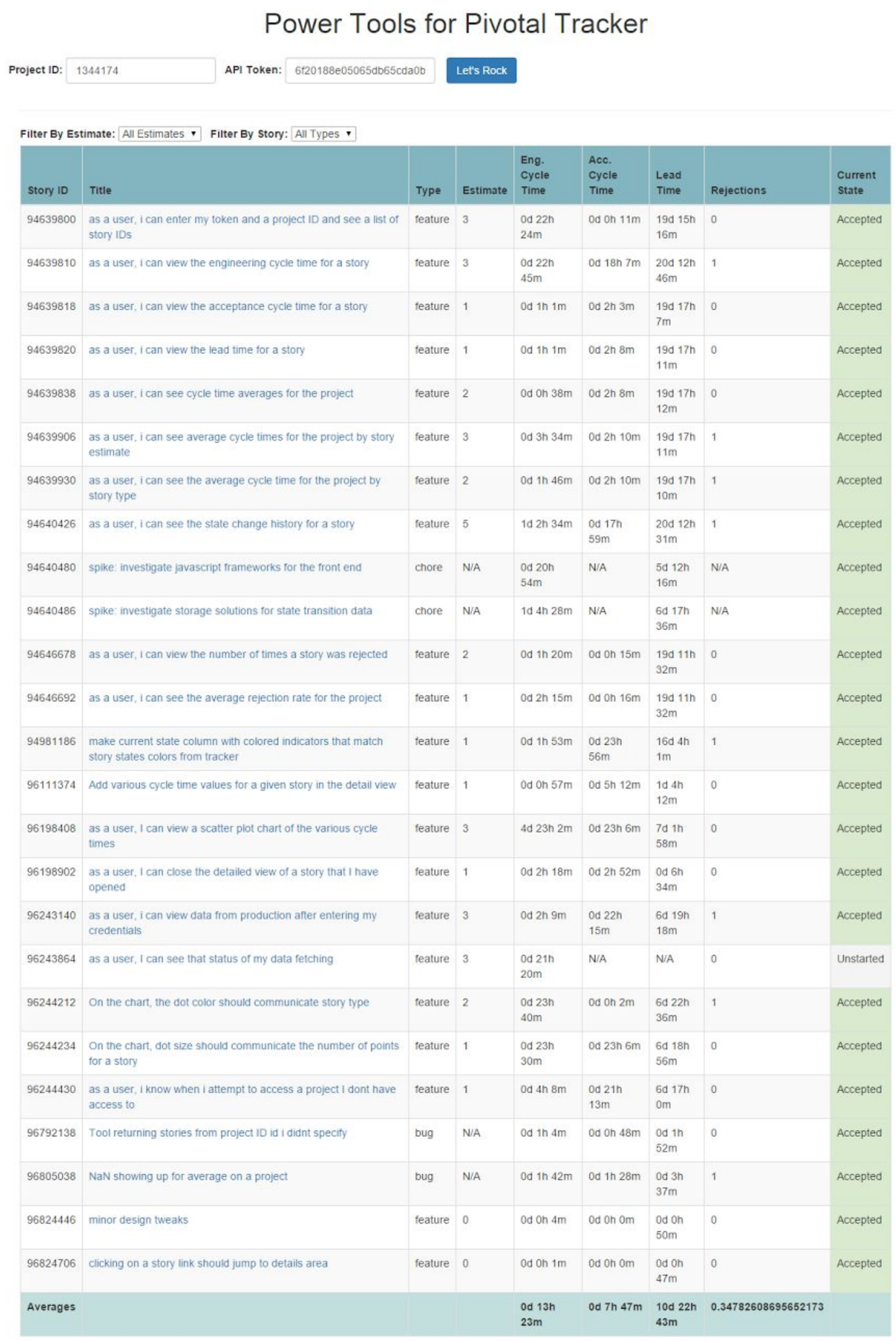


1344174

wrong_token

Invalid credentials! Your API token is incorrect or you don't have access to the project specified.

Figure 6, When the wrong information is input, no information is pulled, and an error message is displayed to the user.



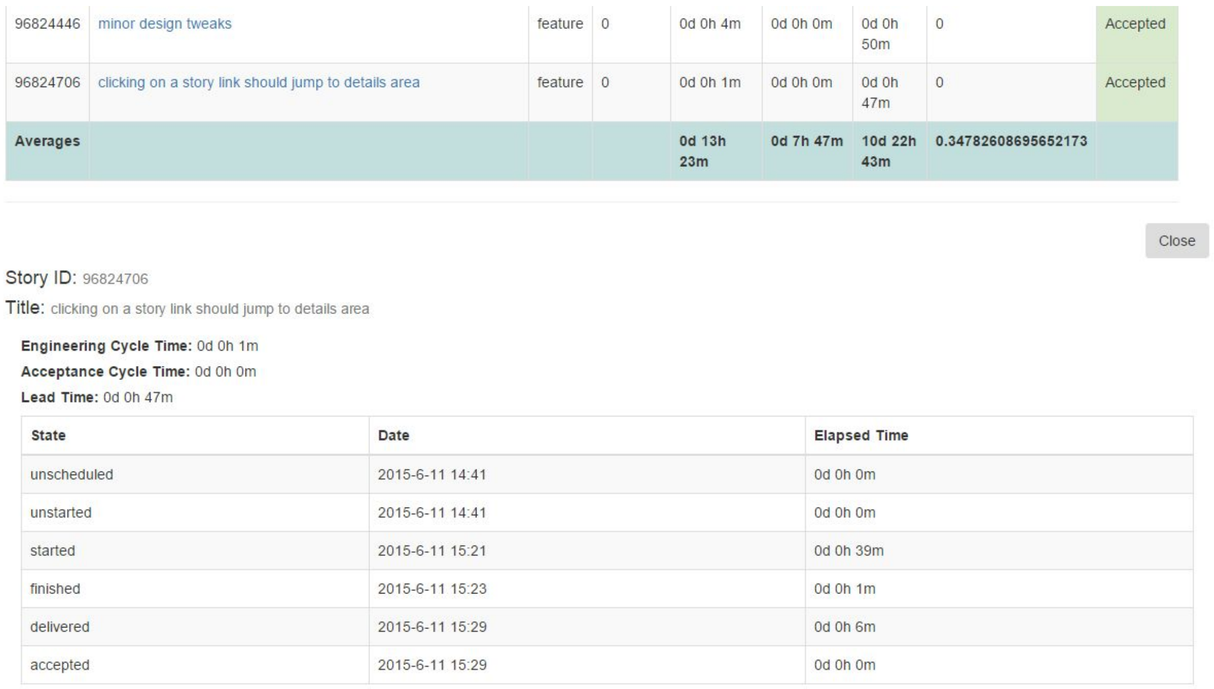
Power Tools for Pivotal Tracker

Project ID: 1344174 API Token: 6f20188e05065db65cda0b Let's Rock

Filter By Estimate: All Estimates Filter By Story: All Types

Story ID	Title	Type	Estimate	Eng. Cycle Time	Acc. Cycle Time	Lead Time	Rejections	Current State
94639800	as a user, i can enter my token and a project ID and see a list of story IDs	feature	3	0d 22h 24m	0d 0h 11m	19d 15h 16m	0	Accepted
94639810	as a user, i can view the engineering cycle time for a story	feature	3	0d 22h 45m	0d 18h 7m	20d 12h 46m	1	Accepted
94639818	as a user, i can view the acceptance cycle time for a story	feature	1	0d 1h 1m	0d 2h 3m	19d 17h 7m	0	Accepted
94639820	as a user, i can view the lead time for a story	feature	1	0d 1h 1m	0d 2h 8m	19d 17h 11m	0	Accepted
94639838	as a user, i can see cycle time averages for the project	feature	2	0d 0h 38m	0d 2h 8m	19d 17h 12m	0	Accepted
94639906	as a user, i can see average cycle times for the project by story estimate	feature	3	0d 3h 34m	0d 2h 10m	19d 17h 11m	1	Accepted
94639930	as a user, i can see the average cycle time for the project by story type	feature	2	0d 1h 46m	0d 2h 10m	19d 17h 10m	1	Accepted
94640426	as a user, i can see the state change history for a story	feature	5	1d 2h 34m	0d 17h 59m	20d 12h 31m	1	Accepted
94640480	spike: investigate javascript frameworks for the front end	chore	N/A	0d 20h 54m	N/A	5d 12h 16m	N/A	Accepted
94640486	spike: investigate storage solutions for state transition data	chore	N/A	1d 4h 28m	N/A	6d 17h 36m	N/A	Accepted
94646678	as a user, i can view the number of times a story was rejected	feature	2	0d 1h 20m	0d 0h 15m	19d 11h 32m	0	Accepted
94646692	as a user, i can see the average rejection rate for the project	feature	1	0d 2h 15m	0d 0h 16m	19d 11h 32m	0	Accepted
94981186	make current state column with colored indicators that match story states colors from tracker	feature	1	0d 1h 53m	0d 23h 56m	16d 4h 1m	1	Accepted
96111374	Add various cycle time values for a given story in the detail view	feature	1	0d 0h 57m	0d 5h 12m	1d 4h 12m	0	Accepted
96198408	as a user, I can view a scatter plot chart of the various cycle times	feature	3	4d 23h 2m	0d 23h 6m	7d 1h 58m	0	Accepted
96198902	as a user, I can close the detailed view of a story that I have opened	feature	1	0d 2h 18m	0d 2h 52m	0d 6h 34m	0	Accepted
96243140	as a user, i can view data from production after entering my credentials	feature	3	0d 2h 9m	0d 22h 15m	6d 19h 18m	1	Accepted
96243864	as a user, I can see that status of my data fetching	feature	3	0d 21h 20m	N/A	N/A	0	Unstarted
96244212	On the chart, the dot color should communicate story type	feature	2	0d 23h 40m	0d 0h 2m	6d 22h 36m	1	Accepted
96244234	On the chart, dot size should communicate the number of points for a story	feature	1	0d 23h 30m	0d 23h 6m	6d 18h 56m	0	Accepted
96244430	as a user, i know when i attempt to access a project I dont have access to	feature	1	0d 4h 8m	0d 21h 13m	6d 17h 0m	0	Accepted
96792138	Tool returning stories from project ID id i didnt specify	bug	N/A	0d 1h 4m	0d 0h 48m	0d 1h 52m	0	Accepted
96805038	NaN showing up for average on a project	bug	N/A	0d 1h 42m	0d 1h 28m	0d 3h 37m	1	Accepted
96824446	minor design tweaks	feature	0	0d 0h 4m	0d 0h 0m	0d 0h 50m	0	Accepted
96824706	clicking on a story link should jump to details area	feature	0	0d 0h 1m	0d 0h 0m	0d 0h 47m	0	Accepted
Averages				**0d 13h 23m**	**0d 7h 47m**	**10d 22h 43m**	**0.34782608695652173**	

Figure 7, With valid input values, a table is displayed with live information that is pulled from Pivotal Tracker. It shows various cycle times and basic story information.



96824446	minor design tweaks	feature	0	0d 0h 4m	0d 0h 0m	0d 0h 50m	0	Accepted
96824706	clicking on a story link should jump to details area	feature	0	0d 0h 1m	0d 0h 0m	0d 0h 47m	0	Accepted
Averages				**0d 13h 23m**	**0d 7h 47m**	**10d 22h 43m**	**0.34782608695652173**	

Close

Story ID: 96824706

Title: clicking on a story link should jump to details area

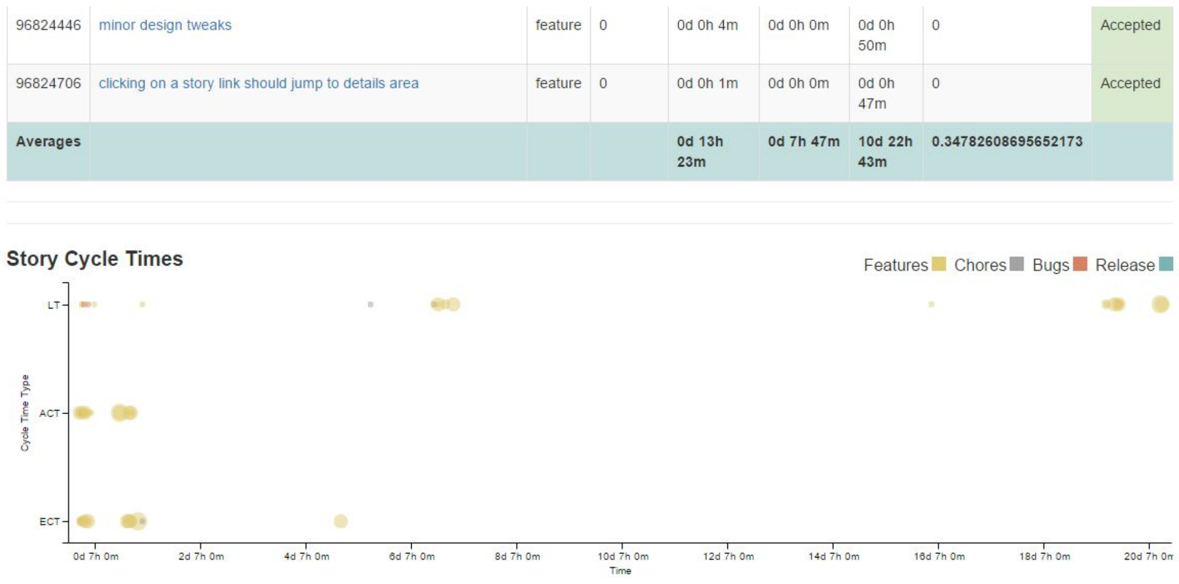
Engineering Cycle Time: 0d 0h 1m

Acceptance Cycle Time: 0d 0h 0m

Lead Time: 0d 0h 47m

State	Date	Elapsed Time
unscheduled	2015-6-11 14:41	0d 0h 0m
unstarted	2015-6-11 14:41	0d 0h 0m
started	2015-6-11 15:21	0d 0h 39m
finished	2015-6-11 15:23	0d 0h 1m
delivered	2015-6-11 15:29	0d 0h 6m
accepted	2015-6-11 15:29	0d 0h 0m

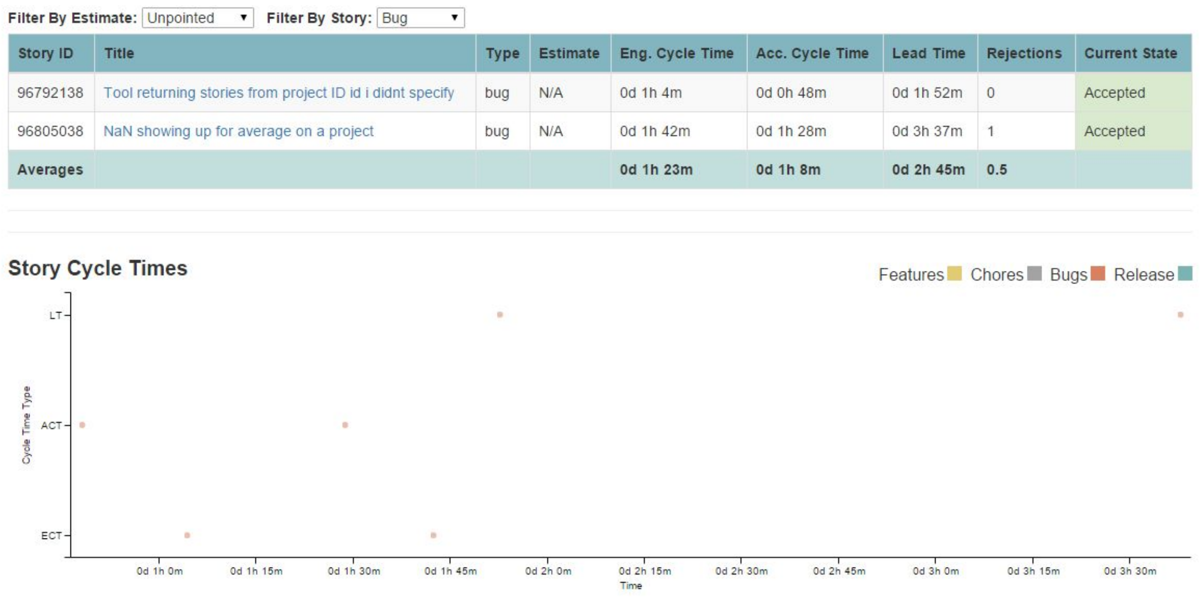
Figure 8, Detail View shows specific story information. Page automatically will jump to Detail View when a story title in the table is clicked. Detail View can also be closed with the button in the top right corner.



96824446	minor design tweaks	feature	0	0d 0h 4m	0d 0h 0m	0d 0h 50m	0	Accepted
96824706	clicking on a story link should jump to details area	feature	0	0d 0h 1m	0d 0h 0m	0d 0h 47m	0	Accepted
Averages				**0d 13h 23m**	**0d 7h 47m**	**10d 22h 43m**	**0.34782608695652173**	

Story Cycle Times

Figure 9, Scatter Plot Chart shows all of the cycle times for each story in the project. Dot size conveys the number of points associated with each story and color shows the story type.



Filter By Estimate: Unpointed ▾ **Filter By Story:** Bug ▾

Story ID	Title	Type	Estimate	Eng. Cycle Time	Acc. Cycle Time	Lead Time	Rejections	Current State
96792138	Tool returning stories from project ID id i didnt specify	bug	N/A	0d 1h 4m	0d 0h 48m	0d 1h 52m	0	Accepted
96805038	NaN showing up for average on a project	bug	N/A	0d 1h 42m	0d 1h 28m	0d 3h 37m	1	Accepted
Averages				**0d 1h 23m**	**0d 1h 8m**	**0d 2h 45m**	**0.5**	

Story Cycle Times

Figure 10, Stories in the table may be filtered by estimate and/or story type. The scatter plot chart will also update in real time to show only the stories in the table.