Sprint (DevOps Tool for students)

Requirements and Specification Document  
2023-02-27, v1.0

Project Abstract

Sprints is a planning and tracking DevOps software platform that is built for students and professionals willing to leverage agile methodologies for their development. It provides teams with access to a comprehensive suite of Agile resources to assist in the development of their software projects such as: Sprint planning, Kanban and Scrum boards, Backlogs, Story Poker, CI with GitLabs, feedback loops and more: All designed with Agile methodologies in mind to ensure quick delivery of working software.

Designed with simplicity and ease of use in mind, Sprints offers an intuitive and user-friendly interface that streamlines collaboration and enables developers to be responsive to changing requirements, reduce project risk, and increase customer satisfaction. With a powerful suite of features and tools, Sprints is the ultimate platform for anyone seeking to build better software faster and more efficiently with Agile methodologies.

Document Revision History

Rev. 1.0 <2023-03-07>: initial version

Rev. 2.0 <YYYY-MM-DD>:

Customer

Our primary consumers are students who need to build software but do not know Agile or lack experience with software development. This includes all future teams in project focusses classes, teams in hackathons, and small businesses. Sprints can also be used by professional teams looking to learn Agile for their projects.

It is a well-known fact that Agile methodology is far superior to several others for developing working software with ability to adapt and overcome hurdles associated with development. However, those looking to develop with Agile are limited to online knowledgebases like YouTube and books. There are very few tools which integrate seamlessly into a developer platform to learn and *practice* agile on the fly. Sprints fills this gap in a multi-billion-dollar industry. It guides teams through Agile in an intuitive way to ensure every team is keeping up with the best Agile practices.

Upon discussing with other students in the class (CS506: Software Engineering), we found out that students find the textbook clunky, annoying and difficult to absorb without practicing Agile methods first-hand. Hence, we have external project interest. We also asked some working professionals (recent graduates) who surprisingly didn’t have a lot of knowledge with agile although their team was ‘agile’.

User Requirements

This section lists the behavior that the users see. This information is presented in a feature format with sub-features listed below it.

* User Stories
* story poker (tool for figuring out how long stories will take)
* creating diagrams stemming from stories (rather than just a list)
* creating user story flashcards
* Ability to add/modify stories and attach supporting documents like pictures.
* Task Board
* Create a Kanban board with columns for to-do/not started, in-progress, done, help, and backlogged task cards.
* Each task has a priority rating (colors or other indicator)
* Tasks can be assigned to single/multiple team members with story poker prompt for duration and deadline.
* Multiple views (group task board and individual task board views)
* Chat/email features when new task is created/archived/overdue (reminders).
* In-app chat with team members to set up meeting times and delegate tasks.
* Evaluation Tool
* Scrum master notes worked on, what will work on, roadblocks per person, sends out a ‘confirmation’ card for each member to check if the SM correctly understood.
* If the info on the card is correct, the team member ‘confirms’ the card, if not, they edit it with correct info.
* The meeting card are posted and collected somewhere, where people can drag it into tasks or just refer back on
* Educational Resources
* have an icon that symbolizes explaining the agile methodology behind a feature
* have a SHORT blurb about that feature popup and a link to a lengthier resource (or even link to a ‘resources’ page)
* Manage stale tasks and sprint backlogs
* if a task is still incomplete by the end of the spring, it turns into a stale task
* There is a section composed of features that can be implemented but are not immediately necessary
* There are reminders and prompts for stale tasks to be completed before they turn stale
* Prompts to pass off tasks to other members or backlog tasks before they turn stale
* Being able to export work
* Having multiple different users
* Users can log in with their own username, pass
* each person will be able to view certain collective shared items, along with their personal individualized view
* ex. will be able to sort/view their individual tasks

Use Cases

* Tasks – **must have (All team)**
  + I want seamless task creation and assignment. My team and I should be able to intuitively navigate the task board. They should be encouraged to ask for clarification if they need specific information (comments/chat).
  + Team members can create their own tasks in the team board. Those assigned to the task must drag the task into their own ‘assigned’ column to show they have accepted the task. The team lead can edit priorities of tasks.
* User stories - **useful (actors: developer, product owner)**
  + My goal is to create and prioritize user stories for a software product and evaluate their feasibility.
    - The product owner and developer will be pre-logged into the tool, and have a vision for the product they are building. They want to create, rank, and evaluate user stories.
    - Flow: There is a ‘user stories’ tab. It displays all stories ranked by priority. Developer can create a new story and product owner can decide it’s priority. The developer fills out a form regarding complexities, feasibility, and risks. PO or the tool can calculate a feasibility score and display it.
* Meetings tool – **useful** (team & non-team members, meeting organizer and participants)
  + My goal is to schedule and conduct meetings for the project
    - The meetings tab displays upcoming meetings in a calendar format. Organizers can create a new meeting, add participants. If people who are not on the tool must be invited, their e-mails can be added to send them an e-mail notification.
    - All participants receive in-tool notification and email to accept or rsvp the meeting. The meeting can be conducted on zoom or any other conference app. Next to the scheduled meeting there is a chat participants can use for specific questions about the meeting, or agenda/notes. Organizer can upload files to the chat as well.
* Educational resources – **must have** (students, teams, PO, SM)
  + My goal is to easily navigate the tool and learn agile methodologies in a seamless way that doesn’t require hiring an Agile Coach.
    - The tool includes include in-house educational resources, tutorials, and guidance for team members who are new to Agile methodologies.
    - The user can activate “learn” mode which simplifies the tool and includes pop-up paragraphs explaining a small amount of theory justifying the feature.
    - There is also a ‘playground’ mode where made-up stories and tasks can be tested with different methodologies to determine the estimated time completion of the project via Agile compared to other methodologies.
* Story Poker – **useful** (developers, product owner, scrum master)
  + My goal is to provide a simple tool so that Agile teams can conduct story pokers on their user stories in a simple manner
    - The tool includes a page that allows one person to add a story and every other team member to add their time estimates.
    - There will be a timer integrated in.
    - After everyone is done, the page shows everyone’s estimates in a table as well as maximum, minimum and average.

User Interface Requirements

Describes any customer user interface requirements including graphical user interface requirements as well as data exchange format requirements. This also should include necessary reporting and other forms of human readable input and output. This should focus on how the feature or product and user interact to create the desired workflow. Describing your intended interface as “easy” or “intuitive” will get you nowhere unless it is accompanied by details.

* Graphical User Interface Requirements:
  + The interface should have a modern and clean design with consistent layout and color scheme. Several features are built in a way that is consistent with other tools to simplify learning curve for users.
  + There should be a login page with google SSO.
  + The platform should have a responsive design that adapts to different screen sizes and devices and accessibility features.
  + The navigation should be easy to use, and users should be able to quickly find and access the features they need.
* Data Exchange Format Requirements:
  + The platform should allow users to import and export data in commonly used formats such as CSV and Excel.
  + The platform should have APIs to allow integration with third-party tools and services.
  + The tool also supports pdf uploads, figma links, git links, and other customizable web links.
* Reporting and Output Requirements:
  + The platform should provide necessary reports and human-readable output such as Sprint progress, burndown charts, and team velocity. Using a simple color scheme and lots of white space, the tool limits options available to users enabling abstraction and standardizing view modes.
  + The platform should provide real-time feedback to users on their progress towards completing their projects.
  + The tool supports e-mailing and slack or discord channel exports that can directly export something to these softwares or import from them.
* Workflow Requirements:
  + The platform should allow users to create and manage Sprints, Kanban boards, and backlogs with ease.
  + The platform should provide a collaborative workspace where team members can communicate, share documents, and provide feedback.
  + The platform should provide clear guidance and instruction on how to use each feature and how to apply Agile methodologies to their projects.
  + The platform should have features such as story point estimation and task assignment to enable users to plan and manage their work effectively.

Security Requirements

Discuss what security requirements are necessary and why. Are there privacy or confidentiality issues? Is your system vulnerable to denial-of-service attacks?

* Authentication and Authorization:
  + The Sprints software platform will require user authentication and authorization for access to the system. Authentication will be managed through a secure login process, with users being required to enter a unique username and password. Authorization will be managed through role-based access controls, which will restrict access to certain features and data based on a user's assigned role.
* Data Privacy and Confidentiality:
  + The Sprints platform will adhere to all relevant data privacy and confidentiality regulations, including but not limited to GDPR, CCPA, and HIPAA. The system will ensure that all user data is securely stored and transmitted, and that access to sensitive data is restricted to authorized personnel only.
* Encryption:
  + All user data stored in the Sprints system will be encrypted both at rest and in transit using industry-standard encryption algorithms.
* Denial-of-Service Prevention:
  + The Sprints platform will be designed to prevent and mitigate the effects of denial-of-service attacks. This will include measures such as rate limiting and IP blocking to prevent malicious users from overwhelming the system.
* Logging and Monitoring:
  + The Sprints system will maintain detailed logs of all user activity, system events, and security-related events. These logs will be monitored on a regular basis to detect and respond to any potential security threats or breaches.

System Requirements

List here all of the external entities, other than users, on which your system will depend. For example, if your system inter-operates with send mail, or if you will depend on Apache for the web server, or if you must target both Unix and Windows, list those requirements here. List also memory requirements, performance/speed requirements, data capacity requirements, if applicable.

1. CSL VM for database storage
2. GitLabs for CI/CD functionality.
3. The web-tool will run on cloud servers and hence only require a strong internet connection from users
4. The tool might require access to APIs such as Figma, Slack, Discord, Amazon AWS, Heroku, Docker, and potentially more.
5. Since the cloud platform will be pre-determined, it need only support a single OS: Linux.
6. The cloud server will require vast amounts of data storage. Up to potentially 4-5 gigabytes per user depending on attachments, products and more. In all, the system must be able to store 1TB of data and scale from there.
7. Sprints must be able to support 500 users at any given time and have a maximum response time of 3 seconds, with a failure or try again message post 3 seconds.

Specification

A detailed specification of the system. Every possible execution should be in the specification, though not every aspect need be covered in extraordinary depth. UML, or other diagrams, such as finite automata, or other appropriate specification formalisms,

This tab outlines the detailed specification of the Sprints system. It includes a description of the system's database architecture, as well as any UML or other diagrams used to represent the system's design.

Database Architecture:

The Sprints system uses a relational database to store all of its data. The database is composed of several tables, including a table for users, projects, sprints, stories, and tasks. Each table is designed to capture the necessary data related to its respective entity.

Users Table:

The users table contains fields for user ID, first name, last name, email address, username, and password. The user ID is the primary key for this table.

Projects Table:

The projects table contains fields for project ID, project name, project description, start date, and end date. The project ID is the primary key for this table.

Sprints Table:

The sprints table contains fields for sprint ID, sprint name, project ID, start date, and end date. The sprint ID is the primary key for this table, and the project ID is a foreign key that references the project table.

Stories Table:

The stories table contains fields for story ID, story name, sprint ID, description, story point estimate, and status. The story ID is the primary key for this table, and the sprint ID is a foreign key that references the sprints table.

Tasks Table:

The tasks table contains fields for task ID, task name, story ID, description, estimated time, and status. The task ID is the primary key for this table, and the story ID is a foreign key that references the stories table.

Story Poker Design Outline:

