

The scrum process is an agile development methodology applied in software development based on iterative and incremental processes. Scrum is an adaptable, fast, flexible and effective agile framework designed to deliver value to the customer throughout the project’s development. Scrum’s primary objective is to satisfy the customer’s need through maintaining transparency in communication, collective responsibility and continuous progress. The development begins with a vague idea of what requires to be built, developing a list of characteristics ordered by priority that the product owner wants to obtain.

Agile Scrum Development Methodology

At Mindbrowser, once the project plan is outlined, we utilize scrum practices for development and management of the product. Using agile project management methodology we deliver new components every 2-4 weeks to our clients. Within a Sprint, a planned amount of work is completed by our team and made ready for review.

Every Sprint comprises daily Scrum meetings, Sprint Retro, Demo days, UAT, and Code Review. This means that every sprint solidifies your product and does not leave anything for chance.

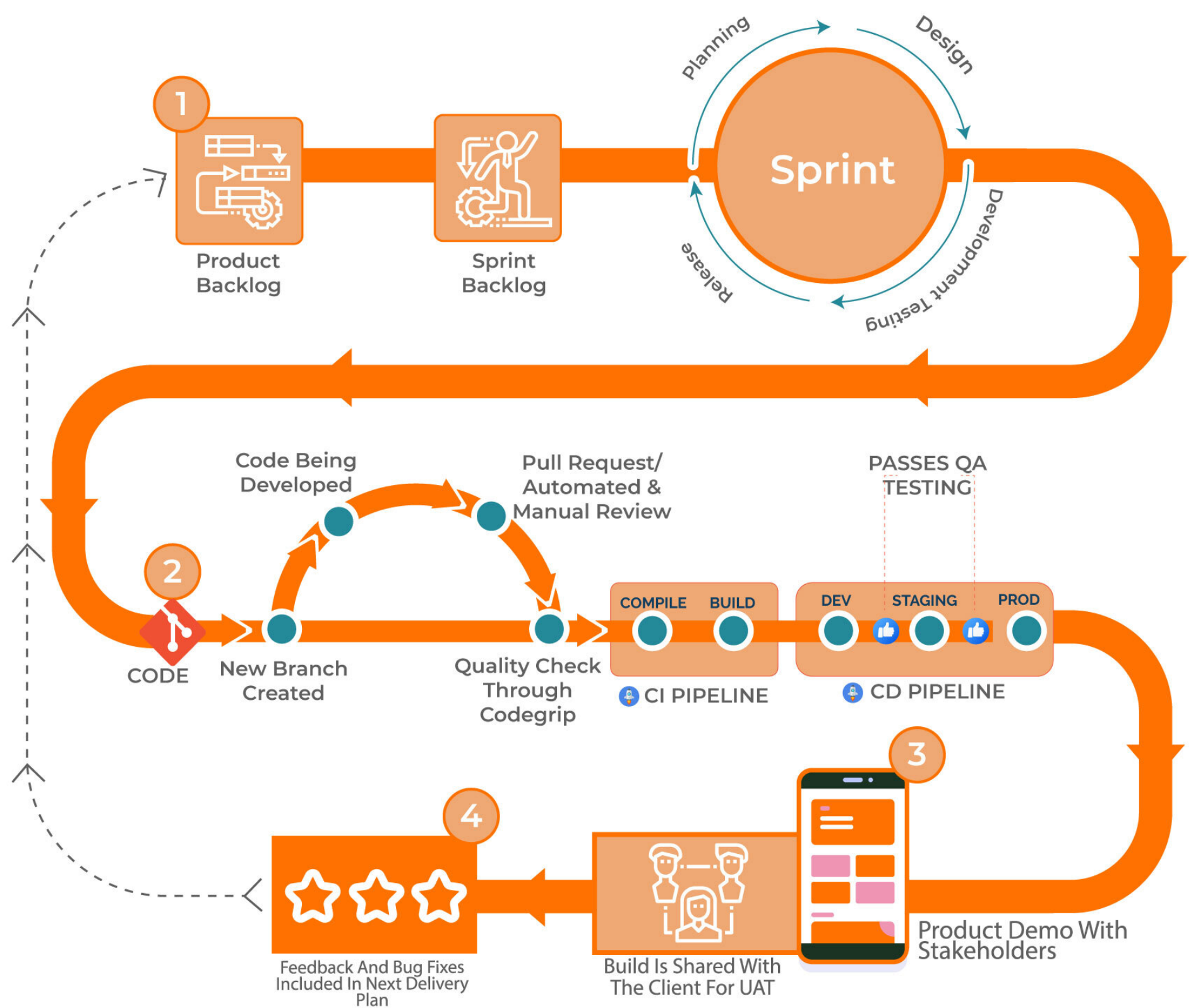


Fig- Agile project management methodology

Scrum framework is best suited for projects that have planned features or have an ongoing demand for new development. The project follows time box cycles which have a set of features fixed along with a date of delivery.

At Mindbrowser, while we follow the process as suggested by Scrum Framework, we have also improvised some of the events to make them better. Here are the events we perform in Mindbrowser for a project development using Scrum Framework.

1. Complete Sprint Plan

This is more for a client to give him/her an initial idea about the timelines (with exact dates) for planning all development to launch activities.

As Scrum uses a sprint structure so we estimate all the sprints and their delivery dates in advance only. This helps the client to get the exact dates of delivery so he can plan his strategy accordingly for e.g. pre-launch, marketing & etc.

The output of this exercise is a complete sprint plan, with a list of features in each sprint to be covered.

The process involves the whole development team. The team decides the priority of the features. The feature priority is decided by the complexity, time taken to complete it, availability of the resources, dependency on other features, and demand from the end user. Based on this discussion we prepare sprints and with the rough time estimates, we decide the start and end date of every sprint.

2. Sprint Planning Meeting

This event is the actual start of any sprint. Here all the stakeholders, the development team and project stakeholders sit together and decide which sprint/features to cover first. Most of the things are already planned during the Complete Sprint plan but here we meet to put things on track and plan against any new challenges.

In this meeting, the team would perform the following actions:

List out features to cover.

Go through the wireframes and have discussions as detailed as possible. We try to catch every scenario/use case (negative positive booths).

Break down the features into user stories.

Every team member adds their own sub-tasks for each User Story. For e.g., a login feature needs UI and backend APIs as well, so the front-end dev will add sub-tasks for creating UI, applying validations and integration of APIs, etc. Similarly, a back-end dev would add sub-tasks to create the required APIs.

All sub-tasks are added with their efforts in hours. We try to subtask for at most 4 hours only; if anything longer than that, then we break it down further.

QA adds their efforts for writing test cases.

Once all the user stories and their sub-tasks have been added, we finalize the dates of all the developers based on their sub-tasks estimations.

Once the developers' dates are finalized, QA can also add their final staging build testing estimates.

Collectively we finalize the date of the whole sprint delivery. The final dates are for the internal demo and client demo of the sprint.


One more important thing we do is decide days in the week when the QA expects builds from the development team. For e.g., if we choose Monday (evening) and Wednesday (evening) as build days, then developers must share builds on these days and on decided times only, not on any other day. This strategy gives enough time for QAs to do quality testing.


All these User Stories are added to JIRA under the sprint. All team members are assigned to sub-tasks they estimate.

Recently Updated


✓ UR-647 14 sub-tasks As a patient I should be able to edit Personal Details again

Done

>  **UR-658** 6 sub-tasks As a patient I should be able to edit Medical Records details again

>  **UR-664** 2 sub-tasks As a developer I should be able to delete attachments directly from AWS

✓  **UR-667** 5 sub-tasks As a patient I should get list of all favorite physicians so that I can select my favorite physician easily

>  **UR-672** 4 sub-tasks As a user I should be able to view/access Side menu from dashboard

✓ **UR-677** 5 sub-tasks As a patient I should be able to filter physicians by the languages filter

S

> **UR-681** 2 sub-tasks Need to change the work logic for Available slots section

✓ **UR-684** 4 sub-tasks As a physician I want to view my profile so I can see details again I filled.

UR-686

D

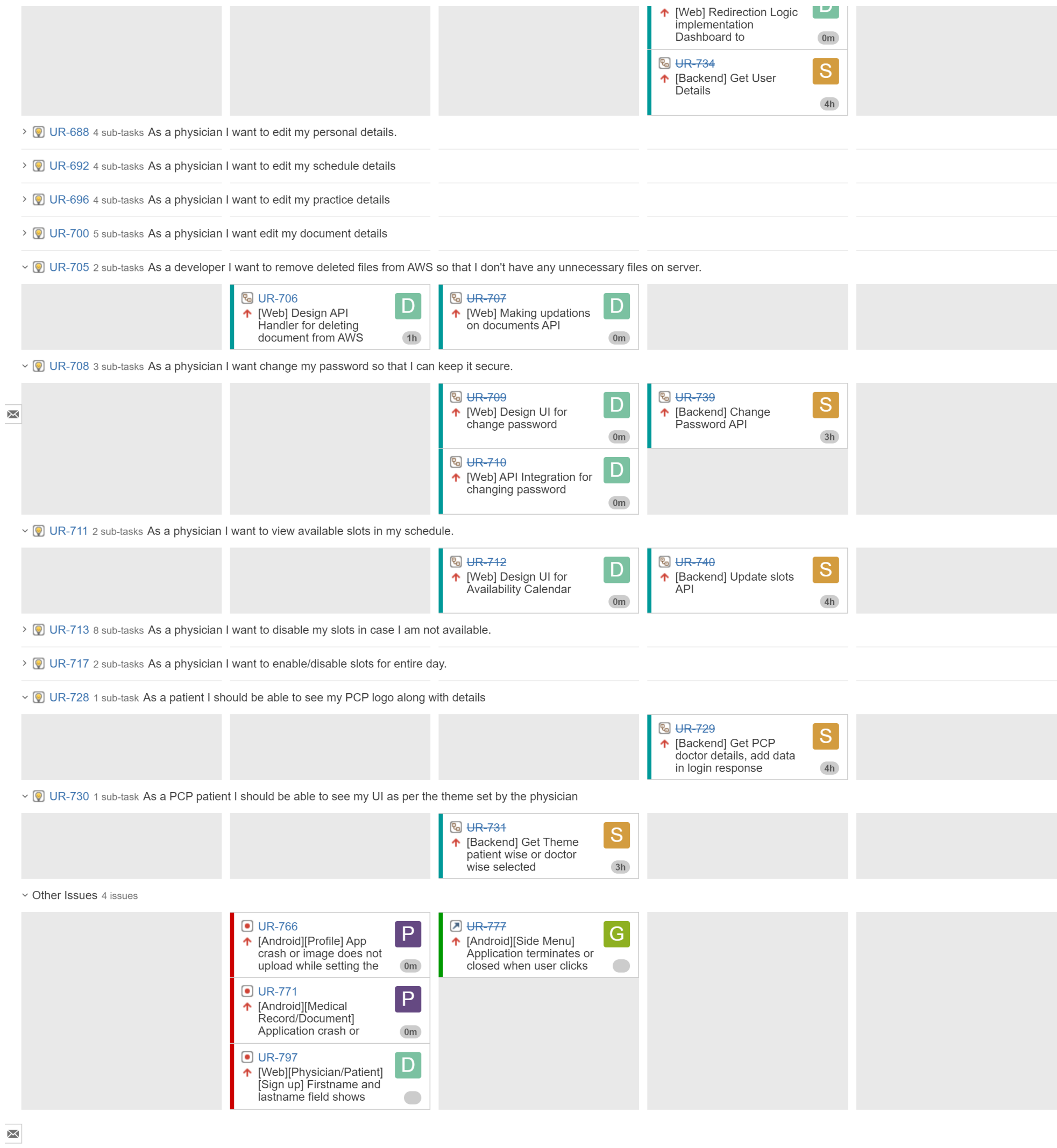


Fig: Screenshots of project from Jira- The project management tool showing the different user stories and their status

Once we have planned all the tasks, we start the actual development, or we can say we start a sprint

3. Daily Standup Meeting

This is one of the most important events in Scrum. The whole team joins together and each member updates on the tasks they completed on the last day, what he/she is going to cover today, and if they have any blockers.

This is a very standard way of doing Daily Standup.

To track the work, we ask developers to choose any 8 hours tasks from their estimates (sub-tasks they created) and start progress on JIRA as well. This aids the team to monitor the variation between estimates and actual work done. It is helpful for developers as well to improve. We also track the progress of the sprint from all the team members. Whether they are on track or not, we can take appropriate actions proactively.

4. Code Review

A code reviewer is assigned to every developer. Pull requests are raised every alternate day to the reviewer. The reviewer checks for the code and makes comments if it can be improved. The criteria for this check is not just to check coding standards or use the correct architecture but also covers the overall logic used if the code can be further optimized. Once all the checks are done then only the PR will be merged. Code Reviews are also automated as part of the [DevOps](#) Pipeline.

5. CI/CD Process

Continuous integration (CI) and continuous delivery/continuous deployment (CD) are the processes that are used to build, test, package and deploy your application. [CI/CD process](#) represents fully automated processes without any manual intervention. This automated process saves loads of time for developers. The developers don't need to worry about generating builds manually.

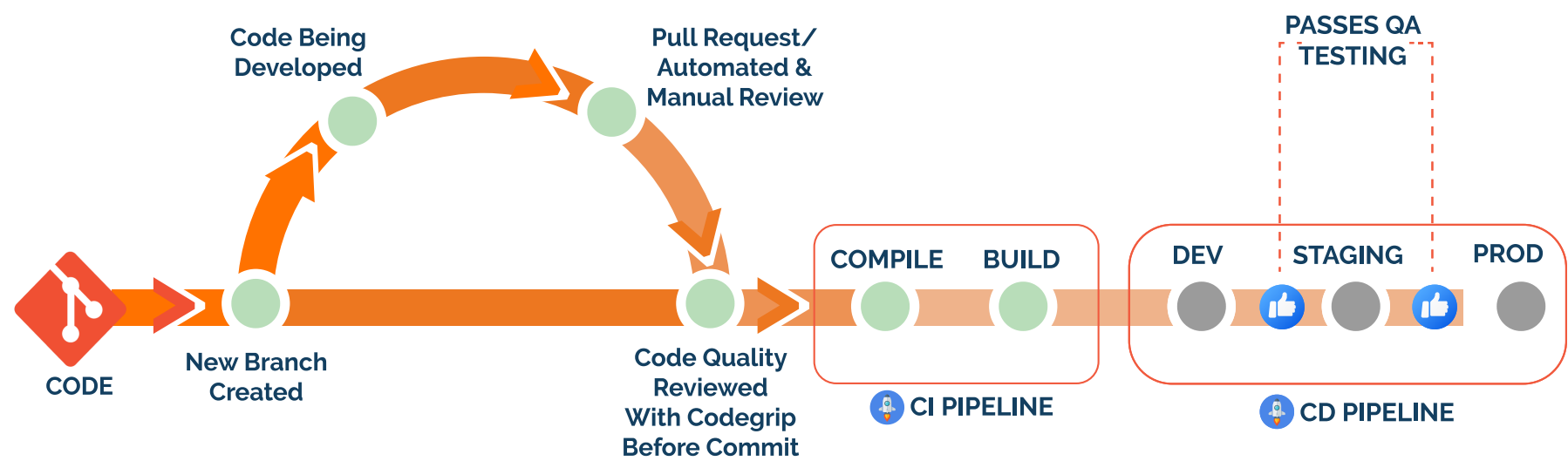


Fig: CI/CD process with automated code review

Improve The Quality And Reliability Of Software With Mindbrowser's DevOps Consulting Services

Contact Us Today

Branching Strategy

This process requires developers to push their code on the git repo and everything is done like magic. We use Bitbucket as our Git code management tool. For the Git repository, we follow a pretty standard branching strategy.

We make sure we have 3 main branches in the Bitbucket repo.

- Master
- Staging
- Develop

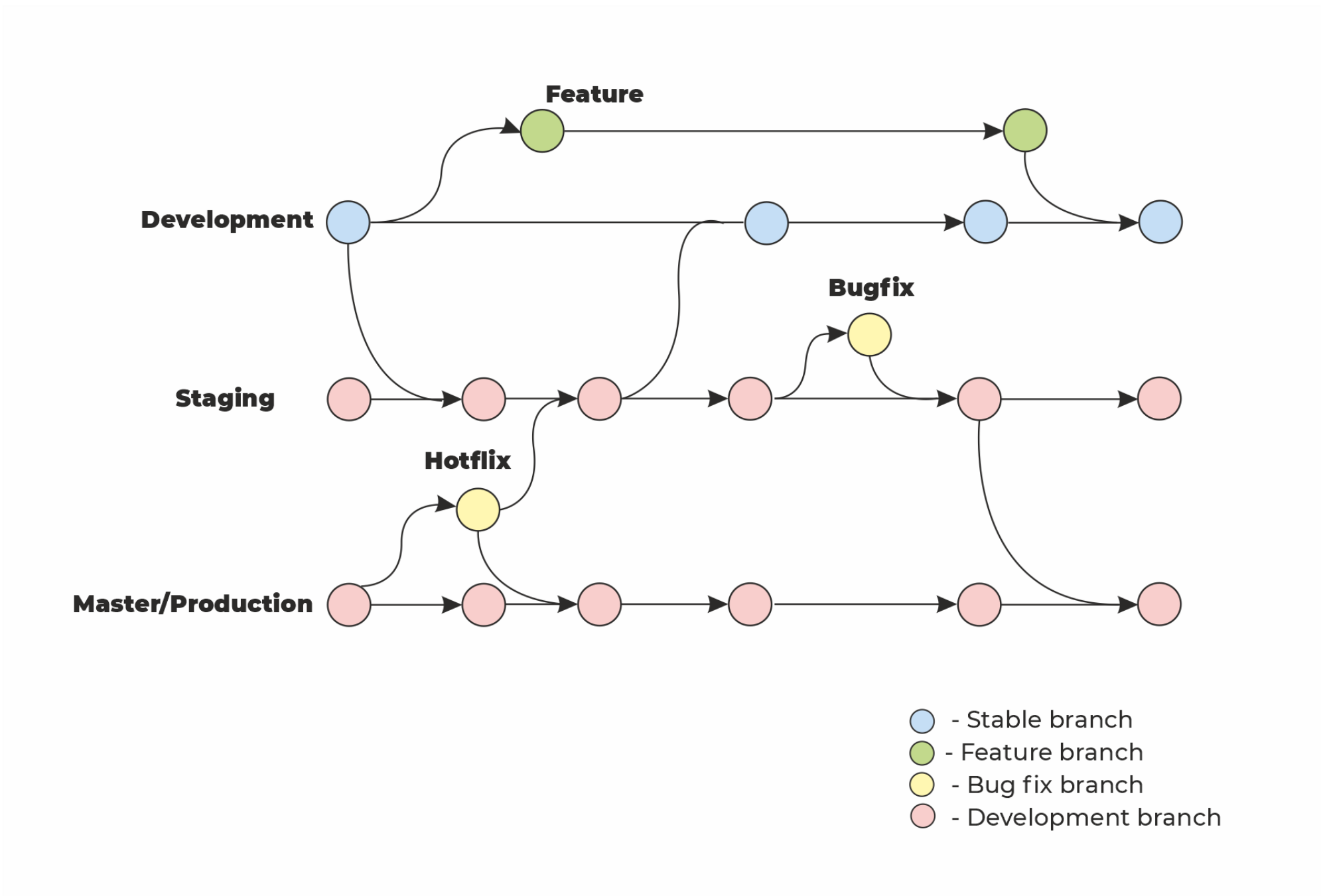


Fig: Branching Practice for code

Developers can create their own sub-branches from the development branch. To merge their work with the main development branch, every developer needs to raise pull requests to their Code Reviewer. Only after approval from the reviewer can the code be merged.

This is an excellent way to check the code quality and verify that good work is done.

The branches have their own significance.

Every commitment on the development branch should be associated with the alpha release. The build generated is used for internal testing.

All commitments to the staging branch are associated with the beta release. The staging build will be shared with clients once tested and demoed internally by the development team.

All commitments to the master branch should be associated with the stable release. This is usually used for production environments.

A developer cannot directly commit to these main branches- Pull requests must be raised to the reviewer and after approval, only their work gets merged.

Reviewers can decline pull requests if any improvement is required in the code. Reviewers can put their comments in the code sections where changes must be made. In this case, developers need to make changes as suggested and raise PR again.

The CI/CD Process checks for every commit/merge made to these 3 main branches and deploys new updates and generates builds. The generated builds are shared with QAs and team members. Builds generated in staging and production environments are shared with clients as well.

Android build (APK file) is shared via Firebase App Distribution.

iOS build is shared via TestFlight.

Web builds are deployed on dedicated servers for different environments.

Every time a commit/merge is done, the whole team is notified using a dedicated Slack channel. The Slack channel also updates on the build generation and deployment steps and notifies you if it fails or succeeds.

6. Internal Demo

Once all the features are developed and shared, tested and verified by QA, in the staging environment we conduct an internal demo. In this event, all the work delivered is demoed by QA to the whole team, and then everyone gives their feedback on the work. Team members can also check for each other's work on their end and provide feedback.

Feedback includes bugs and suggestions for improvement. Based on our feedback changes, we decide whether to make changes only or keep it as part of the new sprint.

Finally, the build is ready to be shared with clients.

7. Client Demo

Often founders complain that they do not really know where their project progress lies until one day they see the project state that has gaps and is too late to fix.

Keep a dedicated time for the demos and product walkthrough at every milestone or major feature update of the project. If your project is an ongoing engagement, then you can have a [demo every 2-3 weeks](#). The demo should be happen over live video sharing followed with sharing of build that you can play around in your own comfortable setting as well

During the demo, understand thoroughly and provide feedback to the team. To avoid too much discussion, we suggest clients keep notes and discuss at the end of the demo. Additionally at start of every sprint, Detailed sprint planning meeting should happen with end to end screen walkthrough addressing team’s understanding about it to avoid further gaps and delays

8. Client Feedback

Feedback can help you understand what customers think about the work and we get a chance to comprehend their expectations. In addition, feedback allows us to maintain transparency between conversations and development.

We keep a dedicated time for the demos and product walkthrough at every milestone or major feature update of the project. Once the work update is shared with the client, we share all the links (Web URLs), APK and TestFlight updates for iOS, and a feedback sheet where we ask the client to provide their feedback and any issues he/she is facing.

9. Sprint Retrospective Meeting

This is a very important event in the scrum framework, but sometimes the team tends to skip it and jumps to the next sprint development. But at Mindbowser we believe in regular assessment of the work, not just for development.

Once the sprint is done and delivered, then we retrospect for the work done. It gives an opportunity to understand the rights and wrongs.

As a part of the process the whole team sits together with a questionnaire filled. We ask the following standard questions in this meeting.

- What went well during the sprint?
- What would we like to change?
- How can we implement the change?

We have also added one more question to understand the learning, mostly in terms of technical improvements.

What did you learn in this sprint?

This helps us to improve, we take every member’s input and try to implement all the feasible changes/suggestions in the new sprint.

	A	B	C	D	E	F	G	
1		24 July 2019	Jack	Anne	Josh	Peter	Matt	
2								
3	1	What went well during the sprint?	Went well:-We did quality of work in code,deliver build on time compare to last time.Went Wrong:-Not designed screens in micro level like somewhere miss to set a proper color, size, padding and margin. So affect us in rework.Not deliver iOS build early like android for internal testing	Tasks were planned properly. Functionality needed to develop was discussed much better than the previous sprint plan.	Improved communication of loopholes in design and development. Also, proper time estimation of tasks.	React project components divided properly and new learning in reactjs while working in this sprint and learned a figma Design.	1. Division of tasks among the developers was quite better. 2. The app development was complete before the time estimation and thereafter much time was given to testing and bug resolution.	
4	2	What would we like to change?	Design Screen in Micro level. deliver iOS and android build early. so developer get time to resolve the issues.	Need to work on time estimation of tasks and also what can be achieved easily and what could take much time should be discussed.	Planning of sprint tasks individually needs some modification.	Need to modify show response in react js,Need to work on time estimation .	1. As a developer, the rework in designing can be prevented by taking care of minor details as well. 2. Responsibility for a particular task should be taken care of as a team.	
5	3	How can we implement the change?	Plan to divide sprint in three Phase with date.After each phase will deliver the both build for test.	Team communication can help to improve sprint delivery.	Effective communication of missing requirements and pending features. Making a checklist for tasks. The planing of tasks based on previous learning.	To improve a we need to divide a each tack into small one so we can estimate a sprint plan.	1. For preventing rework, while designing minor details should be taken care of and regression unit testing can also be done for same. 2. Team coordination can be improved a bit.	
6								

Fig: Sample Sprint plan

10. New Sprint

For new sprint again, we have a Sprint Planning Meeting. Apart from the new features to work on, we consider changes/improvements suggested by the team members in the Internal Demo and the client in the [Client Demo](#).

If there are any queries and doubts about any features, we don't hesitate to connect with clients and clear everything before starting a new sprint.

The inclusion of changes/improvement points depends on the efforts & priority.

After Building 350+ Successful Products-This Is Our Winning Process

[Read More](#)

How Our End-to-end Approach Helps Your Business

Following a systematic approach to our craft has allowed us to avoid pitfalls that many others may succumb to. The whole process ensures that the applications we develop do not end up having spaghetti code. This makes future revisions a breeze.

Our rigorous testing and best practices ensure that the product is launched without crashes and failures. We make it a priority to have a well-defined system architecture and have a clear vision of what we're building. We utilize industry-standard coding conventions and best practices to ensure readability and understandability.

Our meticulous planning, process and control leads to predictable, surprise-free software delivery. This helps us avoid the issues that come with scaling up and also helps us manage our technical debt to be as low as possible.