# **Learning and Capability Report**

# User Stories

## Define the Product

Our first step was confirming what we had to develop. We discussed this as a team and with the Client Peter. We confirmed that we are building a repository that contains evidence for software engineering techniques.

## Identify Product Users

The product brief states the types of users using the repository: Searcher, Submitter, Analyst, Moderator, Administrator. A Searcher can be a Submitter also.

## Non-Functional Requirements

A Searcher should be able to search the SEER database.

A Searcher should be able to view search results.

A Submitter should be able to submit an article to be included in the SEER.

A Moderator should be able to approve/decline submissions.

An Analyst should be able to enter a submission in the SEER.

An Administrator should be able to amend database records and user details

## Refine Requirements

Our non-functional requirements would become epic user stories that would be too big for a team member to complete within a 2 week sprint. Therefore we had to break them down into sets of smaller user stories.

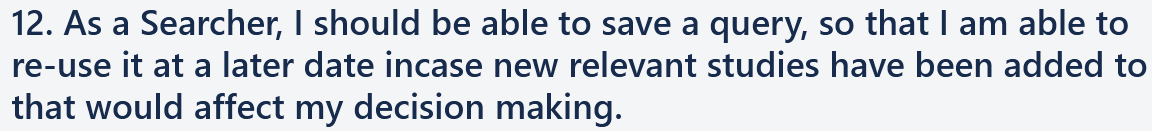
User stories should identify the user, state what they want to be able to do and why they would want to do it. We us the following format to ensure consistency – As a <user> I want to <be able to do this> so that <I get this outcome>.

## Analysing User Stories

We compared our user stories to the INVEST acronym to ensure they contained characteristics e.g. size that would enable them to be implemented in sprints. This also gave us feedback regarding their quality.

* Independent – be as independent or loosely-coupled from other user stories as possible.
* Negotiable – the details of the user story should be negotiated between all stakeholders.
* Valuable – must provide value to user or customer.
* Estimable – should be able to gauge the amount of work involved.
* Small – small enough to implement in a sprint.
* Testable – if possible, should be testable producing a pass/fail result.

A screenshot of a cell phone

Description automatically generatedFigure – User story with INVEST analysis

## Conversations

During sprint planning we decided which user stories to put in the sprint backlog. This enabled us to discuss if the user story was still necessary, whether it needed changing and how we would approach implementing it. By giving user stories the flexibility to change they evolved as we gained insight into how we could best develop the product.

## Acceptance Tests

User stories contained acceptance tests to make sure they were developed correctly and worked as the customer expected them to. Tests should be independently testable and have a pass/fail result. Format: Given <set of circumstances> when <an event happens> then <the expected result>.

A screenshot of a social media post

Description automatically generated

Figure – Acceptance tests

A screenshot of a cell phone

Description automatically generated

# GitHub

## Gitflow Branches

It was important to understand the purpose of each branch before installing Gitflow.

### Master Branch

Contains the latest release version of source code in production

### Develop Branch

Derived from the master branch. Used for merging feature branches.

### Feature Branch

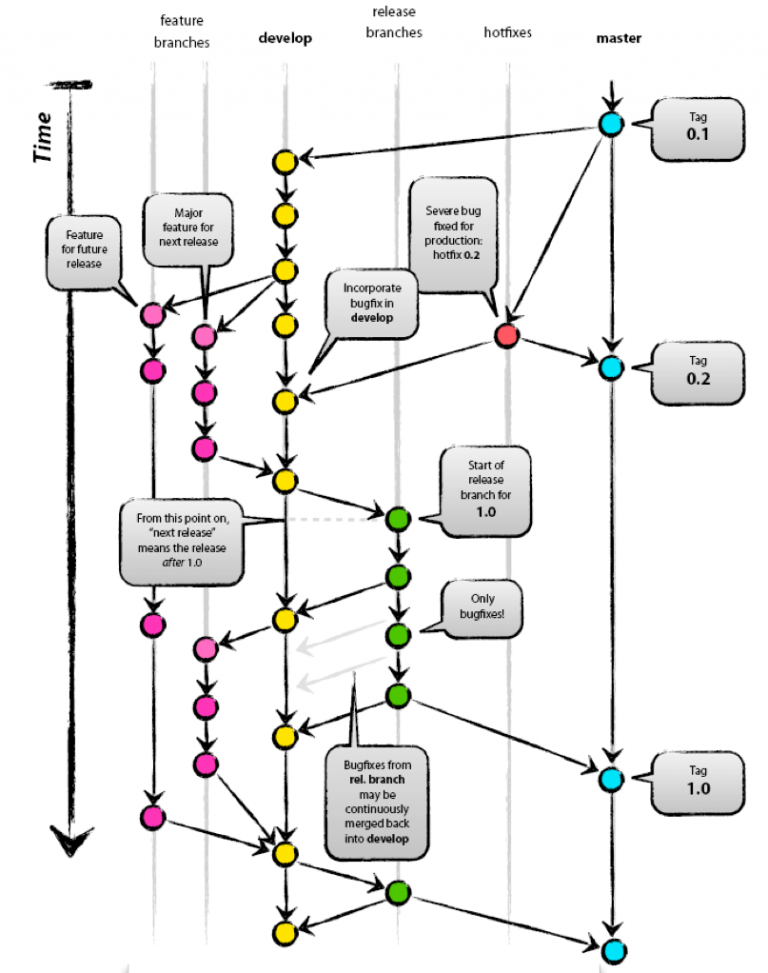
Derived from develop branch. Used to create features.

### Release Branch

Derived from develop branch and adds source code to master

### Hotfix Branch

Derived from master branch and is used to fix bugs identified in production.



### GitFlow Windows Setup

1. Add files (getopt.exe, libintl3.dll, libiconv2.dll) from the util-linux package and dependencies packages and put them in your Git installation inside a bin directory i.e. C:\Program Files\Git\bin
2. Clone this repository: <https://github.com/nvie/gitflow>
3. A close up of a black background

   Description automatically generatedUsing Git as administrator navigate the cloned repository to ‘repo/giflow/contrib/’ then type: msysgit-install.cmd “C:\Program Files\Git”

### Setting up Gitflow in Repository

1. A screenshot of a cell phone

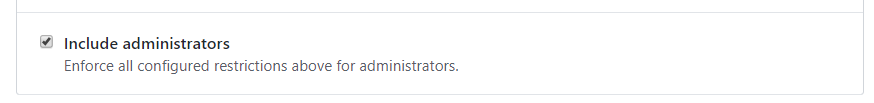
   Description automatically generatedInitialize Gitflow inside repository:
2. A black sign with white text

   Description automatically generatedCreate feature branches:
3. A black sign with white text

   Description automatically generatedPush develop, master, feature/james, feature/justin, feature/zane branches to GitHub:

### A screenshot of a social media post Description automatically generatedGitHub repository and branches

### Branch Protection

* A screenshot of a social media post

  Description automatically generatedMaster branch protection rules

## Give Repository Admin Access to Team Members

1. A screenshot of a cell phone

   Description automatically generatedCreated an organization on GitHub called SE-Team19:
2. A screenshot of a cell phone

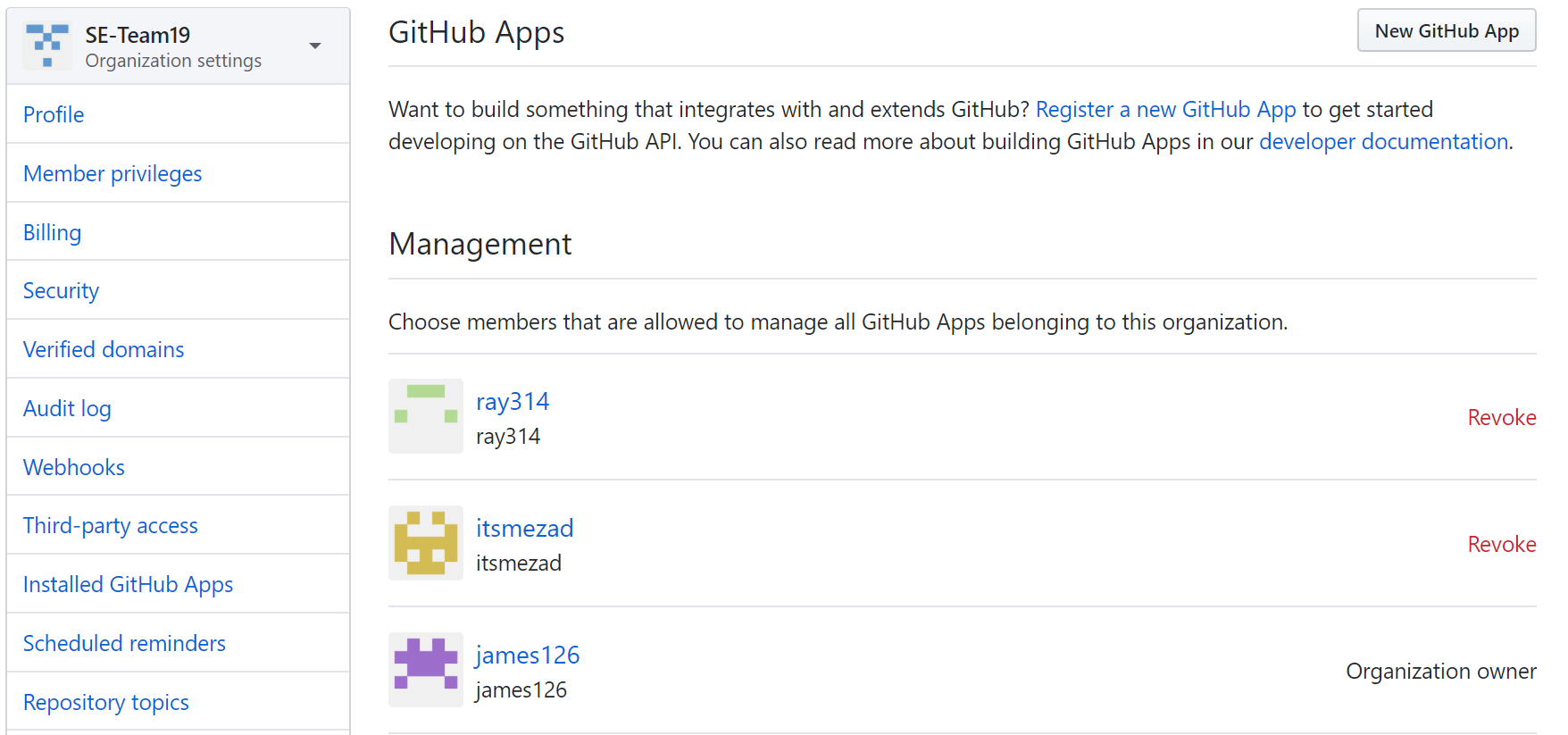
   Description automatically generatedCreated a team also called SE-Team19 within our organization:
3. A close up of a logo

   Description automatically generatedTransferred ownership of our repository from my GitHub account to the team account :
4. A screenshot of a social media post

   Description automatically generatedGave all members of the team Admin permission:
5. Added Justin and Zane to the team:

A screenshot of a social media post

Description automatically generatedJustin has already accepted the invitation.

1. Granted Team members access to manage GitHub Apps e.g. TravisCI and Heroku
2. A screenshot of a social media post

   Description automatically generatedWe can now each add third-party applications to our GitHub repository
3. A screenshot of a computer

   Description automatically generatedCreate a SSH key-pair to authenticate commits from my computer

A close up of a clock

Description automatically generatedCreates the following:

id\_rsa is my private key

id\_rsa.pub is my public key

A black sign with white text

Description automatically generated9. Check git remote is added to repo so I can push and pull commits

A screenshot of a cell phone

Description automatically generated10. After working on the search form I want to push my feature branch to the develop branch.

A close up of a black background

Description automatically generated11. Finishing feature/james deleted my branch! So I created a new branch feature/james that diverges from develop which I merged into previously.

A close up of a screen

Description automatically generated12. Pushing my newly created branch feature/james to GitHub which creates creates a remote branch.

# Mock-up

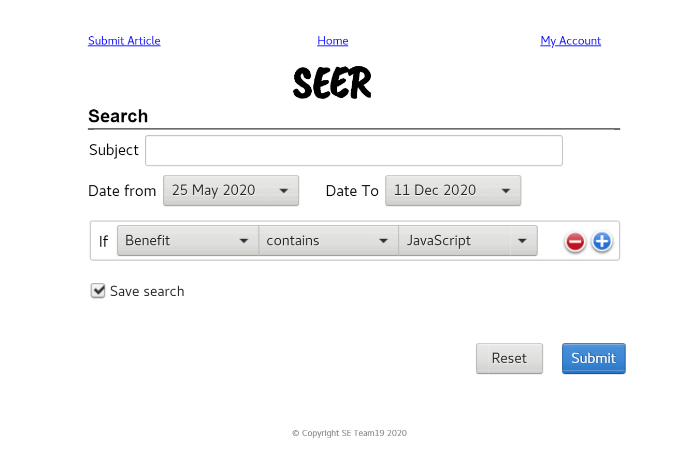
I created a mock-up of the search page using a program called Pencil Project which is an open-source GUI prototyping tool that’s free. The functional requirements of the page needed to be confirmed before I began.

## Functional Requirements

* Users will be able to search for articles
* Users should be able to specify from search criteria: software engineering methods/practices, research outcomes, dates, author and title
* The searching should have pull down values where appropriate
* The user should be able to add any number of constraints
* A query can be saved, and the user can use that query at a later date

## Creating the Mock-up

I created a form which allowed the user to input the search subject, select a range of date to search between and menu bar with drop down values. The menu bar provides additional search criteria and additional menus bars and be added. There is also a option to save the search so that the user can re-use it.

In creating this web page, I found it helpful to have already identified the requirements and planned how I was going to implement it. I saved me time as I knew what I was trying to create before I started coding. This method is more efficient that coding and making up how I will implement requirements at the same time, which of results in a lot of deleted code, wasted effort and frustration. In the future I will definitely create a prototype or wireframe for a web page that contains the key functional requirements for a project.

A screenshot of a cell phone

Description automatically generatedFigure - mock-up

Figure – web page created from mock-up

# Heroku

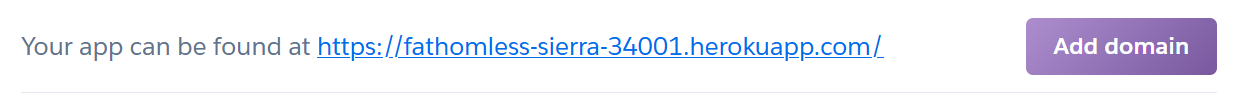
The following screenshots show how I added my GitHub branch feature/james to Heroku and deployed it.

A screenshot of a cell phone

Description automatically generated

* Create repository
* A black sign with white text

  Description automatically generatedShow Git remotes for pushing/pulling branch
* Added a remote to my local repository
* A black and white photo

  Description automatically generatedPushed to heroku from GitHub branch feature/james
* In my Heroku account settings, the feature/james branch can be viewed from the link below

Dwarkani, B. (2017, July 20). Gitflow: The Easy Release Management Workflow. Retrieved May 13, 2020, from https://blog.axosoft.com/gitflow/

Flow cheatsheet. (n.d.). Retrieved May 13, 2020, from https://danielkummer.github.io/git-flow-cheatsheet/?fbclid=IwAR34RjbFiPNIndHrdnNaqi5Y4ITHimuyMuoqxsZQgxGMZKuGtIrRjIm6Fjw

James126. (n.d.). James126/CISE\_S1\_2020\_team19. Retrieved May 13, 2020, from https://github.com/james126/CISE\_S1\_2020\_team19

## **Task Distribution**

*Contents of Learning and Capability Report:*

|  |  |  |
| --- | --- | --- |
|  | **Description** | **Assigned** |
| **1.** | **Iteratively Understanding what to build (Requirements Management)** | **James** |
| **a.** | Explain how you **User stories** are good quality and were used to capture user requirements - including who it’s for and why they want it (the value/purpose to for the user). INCLUDE LINK TO WORK BOARD or SCREENSHOT showing these | **James** |
| **b.** | Evidence shows user stories and how some changed over time and why. | **James** |
| **c.** | Explain how the user stories are functionally testable with examples of success criteria (Acceptance tests). INCLUDE LINK TO WORK BOARD or SCREENSHOT showing these | **James** |
| **d.** | Discuss how A mock-up is used at least once during product development to verify understanding of the requirements. | **James** |
| **e.** | Short descriptions of the main user types and what characteristics are important and how this might affect your application design or requirements. | **James** |
| **f.** | A list of non-functional (quality) requirements is shown with a description of how they were incorporated into the product and how they were tested. | **James** |
| **g.** | A User story map of the product | **James** |
| **2.** | **Forecasting commitment deliverables and checking on progress (Planning and Monitoring)** | **Justin** |
| **a.** | Evidence of use of techniques used to forecast what will be delivered by the end of each sprint (e.g. planning poker). Discuss the pros and cons of each of the meetings? Was it useful to timebox the meeting? | **Justin** |
| **b.** | Evidence it was done at the start of the sprint | **Justin** |
| **c.** | Description of the process (e.g. planning poker) with photos/audio/video | **Justin** |
| **d.** | List the results of the process (forecasts) for each sprint | **Justin** |
| **e.** | Compare the actual work compared with the forecasted results. Comments on differences | **Justin** |
| **f.** | Photos of all meetings and the purpose and outcome of the meetings. | **Justin** |
| **g.** | How did you keep track of time in meetings? | **Justin** |
| **h.** | Explain how you monitored progress with evidence of monitoring (e.g. photos of story board, burndown charts) SEE LINK TO STORY BOARD – check workflow columns | **Justin** |
| **i.** | Show the **roadmap** at the beginning of the development and explain how it changed during the 3 sprints and why. | **Justin** |
| **3.** | **Thinking about what could go wrong and trying to avoid it happening or limit the impact if it does (Risk management)** |  |
| **a.** | Describe a list of the Top 3 risks that could jeopardise successful product development (in the eyes of your team and PO) and how you will reduce the chance they could happen and/or the impact they would have on re-work or quality of the product. | **Zane** |
| **b.** | Discuss how these were updated during development. Show examples. | **Zane** |
| **c.** | Were any examples of impediments or blockers identified during the “daily” planning “standup meetings”? Give examples and how they were resolved. | **Zane** |
| **d.** | How was the risk of misunderstanding the PO lessened? Give examples. | **Zane** |
| **4.** | **Working Collaboratively** |  |
| **a.** | Explain how useful (or not) the daily communications and planning meeting was to help the team coordinate and collaborate, with evidence. | **Zane** |
| **b.** | Explain how GitHub was useful for team collaboration? Give examples. | **Zane** |
| **c.** | Explain how you got participation of all team members at meetings? Give examples. | **Zane** |
| **d.** | How did it work with no team roles? In what ways were you a self organising team? Give examples. | **Zane** |
| **e.** | How did you get agreement on decisions? Give examples. | **Zane** |
| **f.** | How did you handle conflict and disagreements in the team? Give examples. | **Zane** |
| **g.** | Explain the Team agreement and if it was useful to collaboration and if it changed and why. | **Zane** |
| **5.** | **Product Architecture** | **Justin** |
| **a.** | Show evidence of separation of data and business rules and user views. (e.g. MVC structure) | **Justin** |
| **b.** | Explain the high-level product architecture and how the Technology stack was used implemented it | **Justin** |
| **6.** | **Code craft** | **James** |
| **a.** | Show examples of code that show **intentionality** | **James** |
| **b.** | Show examples of code design that show small (a few lines) methods/functions/components | **James** |
| **c.** | Show examples of code that show code design using small unit tests | **James** |
| **d.** | Explain what code standards were used and show examples of code that show coding standards | **James** |
| **7.** | **Capability with Tools** |  |
| **a.** | Show evidence of the use of all tools used including screenshots and set up instructions and configuration files. This should include | **Zane** |
| **b.** | Git/GitHub with branching, use by all team members, good commit messages, pull-requests | **James** |
| **c.** | TravisCI – yaml file, and screen shots of it running and passing/failing tests report | **James** |
| **d.** | Examples of Unit tests in your code using a suitable test framework (e.g. Mocha/Chai, Jest) and any screen shots of configuring these tools | **Justin** |
| **e.** | Heroku – screenshot of any configuration or settings | **Zane** |
| **f.** | Use of Trello or Asana (or similar) for user stories and acceptance criteria – useful column headings, frequent use, use by all team members | **James** |
| **8.** | **The product quality** |  |
| **a.** | Describe the use of unit testing and how it was useful. EXTRA MARKS for showing evidence of Test first development (or Test Driven Development) | **Justin** |
| **b.** | Describe your continuous integration process with evidence including a description of the role of TravisCI | **Zane** |
| **c.** | Describe your review process for code quality - including pull requests in GitHub CHECK GITHUB for evidence | **James** |
| **d.** | Describe how you tracked bugs or other issues | **Zane** |
| **e.** | Describe how you tracked whether unit tests and acceptance criteria and any other tests passed or failed and when they were re-tested | **Justin** |

## **1. Iteratively Understanding what to build**

1b

ghg

## **2. Forecasting commitment deliverables and checking on progress**