CS 410 software engineering

Product Backlog

3/9/2021

By: Team Prestige Worldwide

Members:

Sujith Nakkala

Christopher Schmitt

Marian Slepecky

Joe Smith

Jordan Thurston

Here are our refined user stories.

Each Epic is assigned an integer, and each user story that falls under an Epic is assigned another integer following a period (e.g. 5.4 would be user story 4 inside of Epic 5).

Each story has assigned Story points and priority associated with it at the beginning of the story. Each story also has pre and post conditions.

(SP = story points, H = high priority, M = medium priority, L = low priority)

0. (Epic) As a user, I want to visit a website where I can either view or submit a claim, so I can decide what I want to do on the website.

    0.1. (2 SP, M) As a developer, I want a front end for claimants with a button that leads to a page where a claimant can submit a claim and a button that leads to a page where a claimant can view a claim so that a claimant can choose whether or not they are going to submit a claim or view a previous claim.
        Preconditions: None.
        Postconditions: There is a web page that a claimant visits first which leads to either another web page where the claimant can file a claim, or another web page where the claimant can view a claim.

    0.2. (2 SP, M) As a developer, I want a front end for an insurer so that the insurer has a web page separate from the claimant's webpage where they can view claims.
        Preconditions: None.
        Postconditions: There is a web page that an insurer visits to view claims.

1. (Epic) As a claimant, I want to submit a claim easily, so that I can be compensated for an auto accident.

    1.1. (8 SP, H) As a developer, I want a front end where the claimant can fill out the necessary information for a claim so that the claimant can submit a claim.
        Preconditions: User story 0.1.
        Postconditions: A working web page where a claimant can fill out their information about their claim.

    1.2. (8 SP, H) As a developer, I want to create a back end API that can write to a database so that a claim can be stored.
        Preconditions: None.

Postconditions: A back end API endpoint with a database that can be called to store a claim.

1.3. (3 SP, H) As a developer, I want the front end to call the back end API endpoint for storing claims so that when a claimant submits a claim into the front end it can be sent to the back end API and stored in a database.

Preconditions: User stories 1.1. and 1.2.

Postconditions: The front end can send claims to the back end to be stored in a database.

1.4. (5 SP, L) As a developer, I want the back end API to make calls to an autocomplete API so that when the user types their address it can be automatically completed.

Preconditions: User story 1.2.

Postconditions: The back end API can be asked by the front end to autocomplete text using the autocomplete API.

1.5. (5 SP, L) As a developer, I want the front end to call the back end autocomplete API endpoint whenever the claimant types a part of their address so that the front end can display suggested completions of the address that the claimant is typing.

Preconditions: User stories 1.1. and 1.4.

Postconditions: The back end can be asked to return automatically complete addresses, and the claimant can use the suggestions to auto complete their address.

1.6. (3 SP, M) As a claimant, I want to see a confirmation page that shows me that my claim was successfully submitted so that I can ensure that my claim will be processed.

Preconditions: User story 1.3.

Postconditions: There will be a web page with text confirming successful submission that displays after the back end has stored a claimant's submitted claim.

2. (Epic) As a claimant, I want to view my claim so that I can see what I submitted and know its status.

2.1. (5 SP, M) As a developer, I want there to be a back end API endpoint for claimants that can read from the database so that the claimant can view their claim.

Preconditions: User stories 1.1. and 1.2.

Postconditions: A claimant's claim can be retrieved from the back end.

2.2. (5 SP, M) As a developer, I want a front end where the claimant can provide information to prove their identity so that they can access only their claim.

Preconditions: User story 0.1.

Postconditions: There will be a web page where the claimant can fill out their information to request to view a claim.

2.3. (3 SP, M) As a developer, I want a front end which can display a claim so that the claimant can view their claim.

Preconditions: User stories 2.2. and 2.1.

Postconditions: A web page showing the claim of the claimant, and text indicating its status.

3. (Epic) As an insurer, I want to view a claim submitted by a claimant so that I can process it.

3.1. (8 SP, H) As a developer, I want a front end UI so that the insurer can search for and view claims.

Preconditions: User story 0.2.

Postconditions: There is a web page where an insurer can input information to search for a claim .

3.2. (5 SP, H) As a developer, I want a back end API endpoint for the insurer that can search for and retrieve claims so that the insurer can retrieve them using the front end.

Preconditions: None.

Postconditions: Claimants' claims can be retrieved based on the search criteria.

4.(Epic) As a security expert, I want information to be accessed only by the people who are authorized to access it so that this information is not misused.

4.1. (8 SP, L) As a developer, I want a back end API server for authenticating the back end API calls.

Preconditions: None.

Postconditions: Any backend API calls will be authenticated.

The cumulative story points equal 70. For the first sprint we will be committing to user stories 1.1, 1.2, and 1.3. Which totals to 19 story points. 19/70 is 27%. Since we will have a total of 4 sprints, and since 27% for the first sprint is above 25%, we will be on schedule.

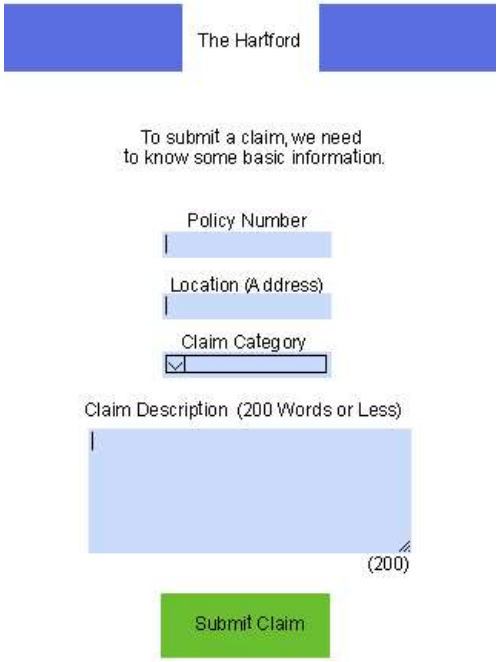
Functionality: After Sprint 1, a claimant can use a web page to select whether they would like to submit a claim or view a claim. They will only be able to submit a claim by filling out a form, the claim is then stored in a database using the back end API endpoint for claimants. The API

endpoint for an insurer to access claims based on search criteria will be implemented as well, but the page that an insurer uses to search for and view these claims will not have been implemented yet.

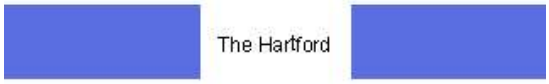
key features of the user interface; **sketches** of designs

Claimant Landing Page:

Claimant Claim Submission Page:



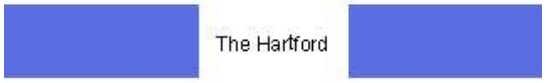
Claimant Claim Submission Success Page:



Success!

Your claim has been successfully recorded.

Claimant Retrieve Claim Authentication Page:



To retrieve your claim, we need to know your policy number and your name.

Policy Number

First Name

Last Name

View Claim

Claimant View Retrieved Claims Page

We found 2 claims under your name.

Name (last, first, middle):
Smith, Joseph, Johnson

Policy Number:
13243546

Location:
123 Alphabet St.

Date:
March 9th, 2021

Category:
Collision

Claim Description:
It wasn't me, chief! He started it!

Status:
Sent To Adjuster

Name (last, first, middle):
Smith, Joseph, Johnson

Policy Number:
13243546

Location:
1230 Main St

Insurer View Claim Page

Find A Claim

Policy Number
13243546

Name (last, first, middle):
Smith, Joseph, Johnson

Policy Number:
13243546

Location:
123 Alphabet St.

Date:
March 9th, 2021

Category:
Collision

Claim Description:
It wasn't me, chief! He started it!

Status:
Sent To Adjuster