ELECOMP CAPSTONE
VIRTUAL SUMMIT
FRIDAY, MAY 8, 2020

THE UNIVERSITY OF RHODE ISLAND

THINK BIG WE DO™

RUReady

Cooperative Backlog Refinement Software for Agile Development

Team Members: Patrick Hurney (CPE), HopeRose Puroll (CPE & CSC)

eMoney Advisor

Technical Directors: Daniel Jaquez, Gary Jutras, Darius Strasel, Elliot Young | Consulting Technical Director: Brenden Smerbeck ('17)

Project Motivation

In the Agile process, developers use a set of acceptance criteria called a "Definition of Ready" (DoR) to ensure a task is in a state that is both workable and poses little risk of delayed completion. A team agrees on this set of standards that all tasks must meet before it is included in a sprint. Currently, there is no tool that will walk a team through comparing their work items to this DoR during the backlog refinement process. Most teams follow manual checklists and the use of verbal consent from participants. During the backlog refinement process, there is a lot of discussion about each item that is not captured in any form. There are many times when team members want to recall key points from these discussions, but they must rely on their imperfect memory for details. The RU Ready application will automate much of the process of walking through a team's DoR.

Key Accomplishments

Jira Controller: The Jira controller verifies that a user either has been authenticated through Jira and has allowed RUReady to make asynchronous API calls on their behalf. If a user is not authenticated, the controller redirects the web page to Jira for the user to input their credential and saves their token information in session for future use

Frontend API Link: Front end API calls for a list of teams via a Redux and React framework as shown in **Fig. 3**. This connects to the Teams API, discussed below and shown in **Fig. 4**.

Teams API: The Teams list controller verifies that the user has Jira tokens in session and either redirects them to sign in if that is not or forwards the request to the Team's service. This service first forwards the appropriate CRUD request to the repository which packages the contents in Http to be forwarded to Jira. Upon return from Jira and the repository, the data is then processed in the service to comply with data models before being converted to JSON and sent back to the Teams controller and then ultimately back to the front end to be rendered **Fig. 4**. Because there is no way to directly get a list of teams for a user, the team's service works through several Jira API calls to completely compile a list of teams. A complete list of teams is used to determine a complete list of roles that are in turn used to determine a complete list of users. This is then compared to session information and information from previous calls are combined to create a functional list.

Team Rendering: Front end data mappers from Json to Team list were implemented to parse the files received from the backend Teams API, referenced previously. This linked the JSON file to individual teams which were contained within a page that renders a list of Teams **Fig. 1**, to account for any members on multiple teams. This was styled with both inline and individual CSS files.

DoR API: The DoR controller has endpoints for Get, Post, and Delete REST actions. These requests are then forwarded to the DoR service. The service forwards Get and Delete requests immediately and does validation checking on Post body before forwarding the request to the repository. The repository creates, deletes, reads and writes JSON files to the file system to store a team's DoR information. Upon return from the repository, the service then verifies that a DoR is present and either returns a default or formats the DoR to be sent back to the front end to be rendered **Fig. 4**.

Criterion/DoR Rendering: Front end API calls for a list of criteria functioned similarly to those done for the Teams list, but the link to the backend varied since in these instances the front end was actually able to update the data and send API requests directly. These changes required their own frontend data mapper from JSON to the DoR criterion, and like the Teams page, there needed to be a page that renders a list of criteria **Fig. 2**. The page contained card components that render a single criterion, with each type of criterion being rendered in a slightly different way to represent the different information contained within. This was styled with external CSS.

Front End Actions: The frontend links to the backend, referenced previously, works using Redux Data modeling and transfer, using action creators, processors, gateways, and data storage. This is a bit beyond what is shown in **Fig 3** and **Fig. 4** but is an important part of how the program actually functions.

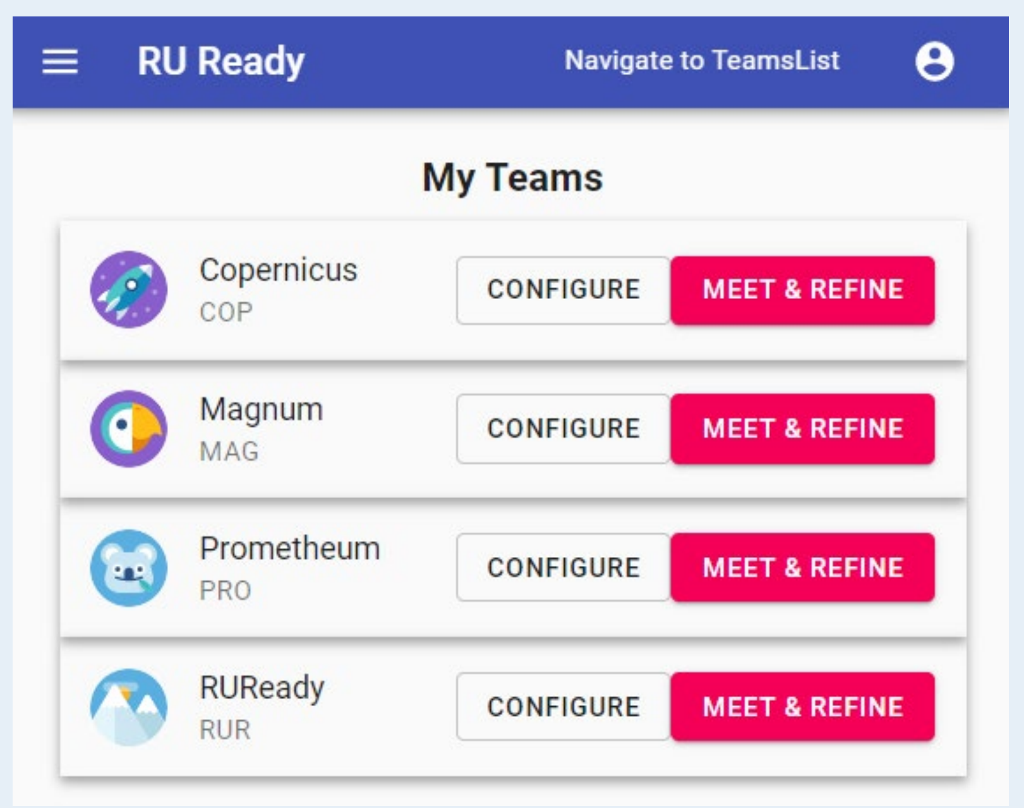
Anticipated Best Outcome

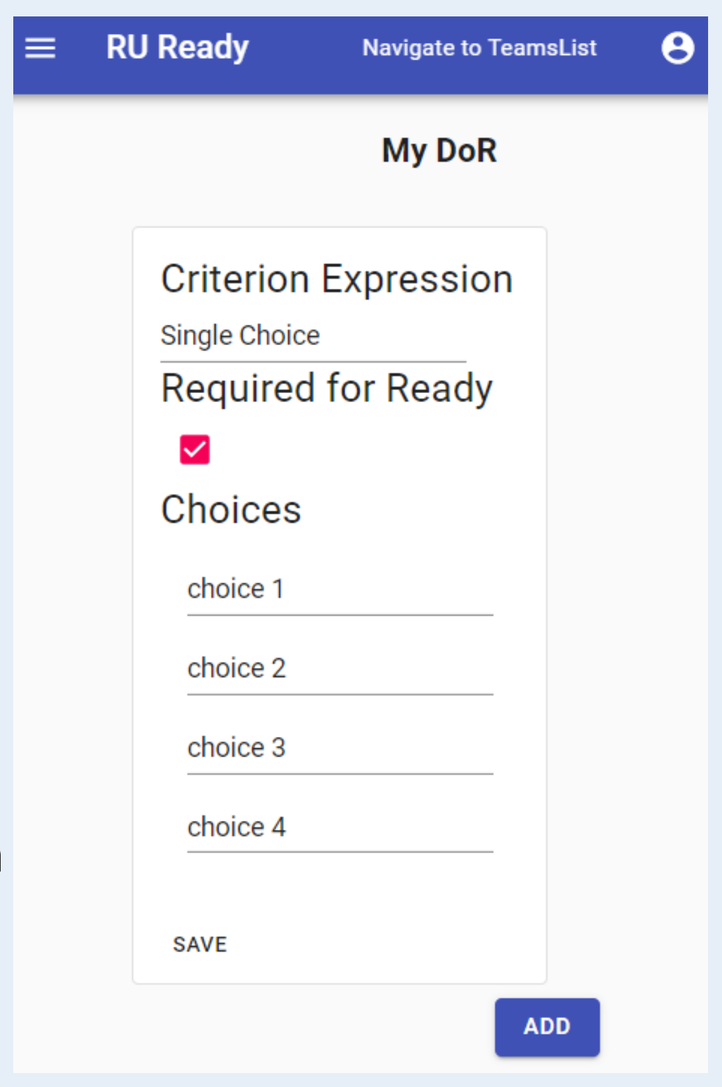
The anticipated best outcome for this project has evolved since the Symposium. We have chosen to focus on the development of the definition of ready, allowing the creation, editing, and deletion of the DoR criterion. We aimed to integrate with Jira to retrieve a list of teams. To create a container page for the DoR configuration. To implement DoR creation and deletion via linking through Redux store actions and features. And finally to allow the editing and reorganizing of the DoR. All these things link to the backend through a complex network of React, Redux, and C# API calls.

Project Outcome

Updated Best Anticipated Outcome Achieved. Initial goals were out of scope, but once updated the work was completed.

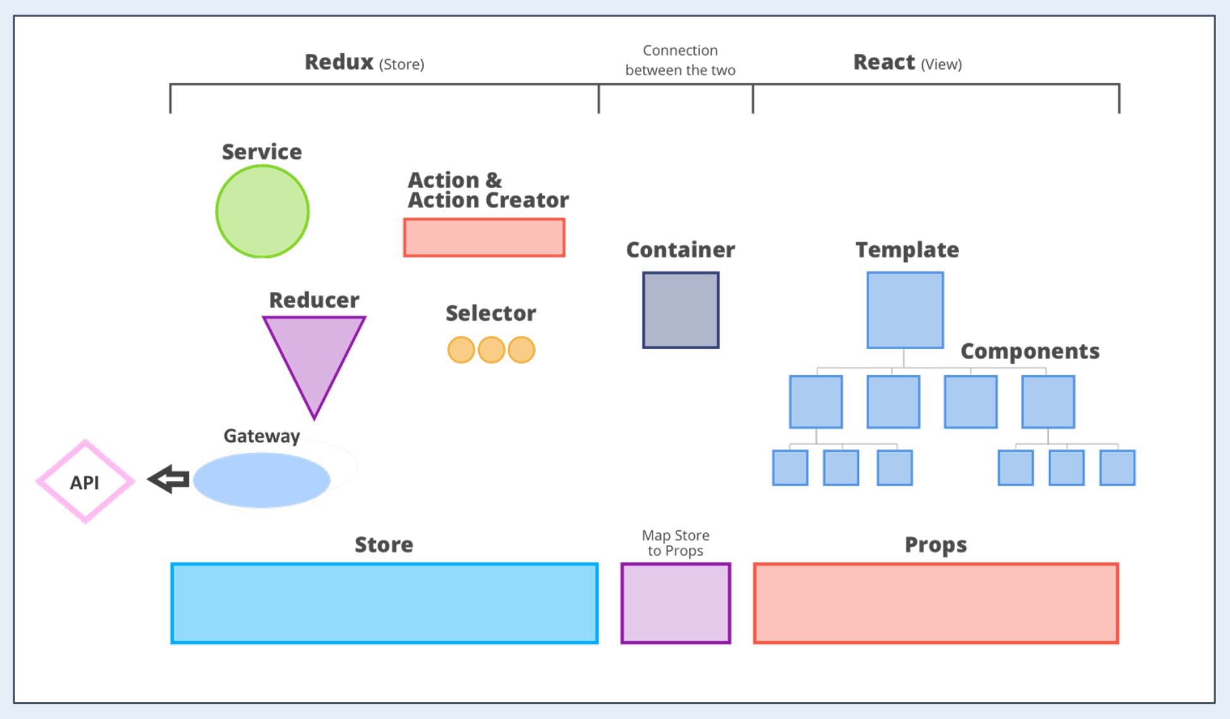
Figures



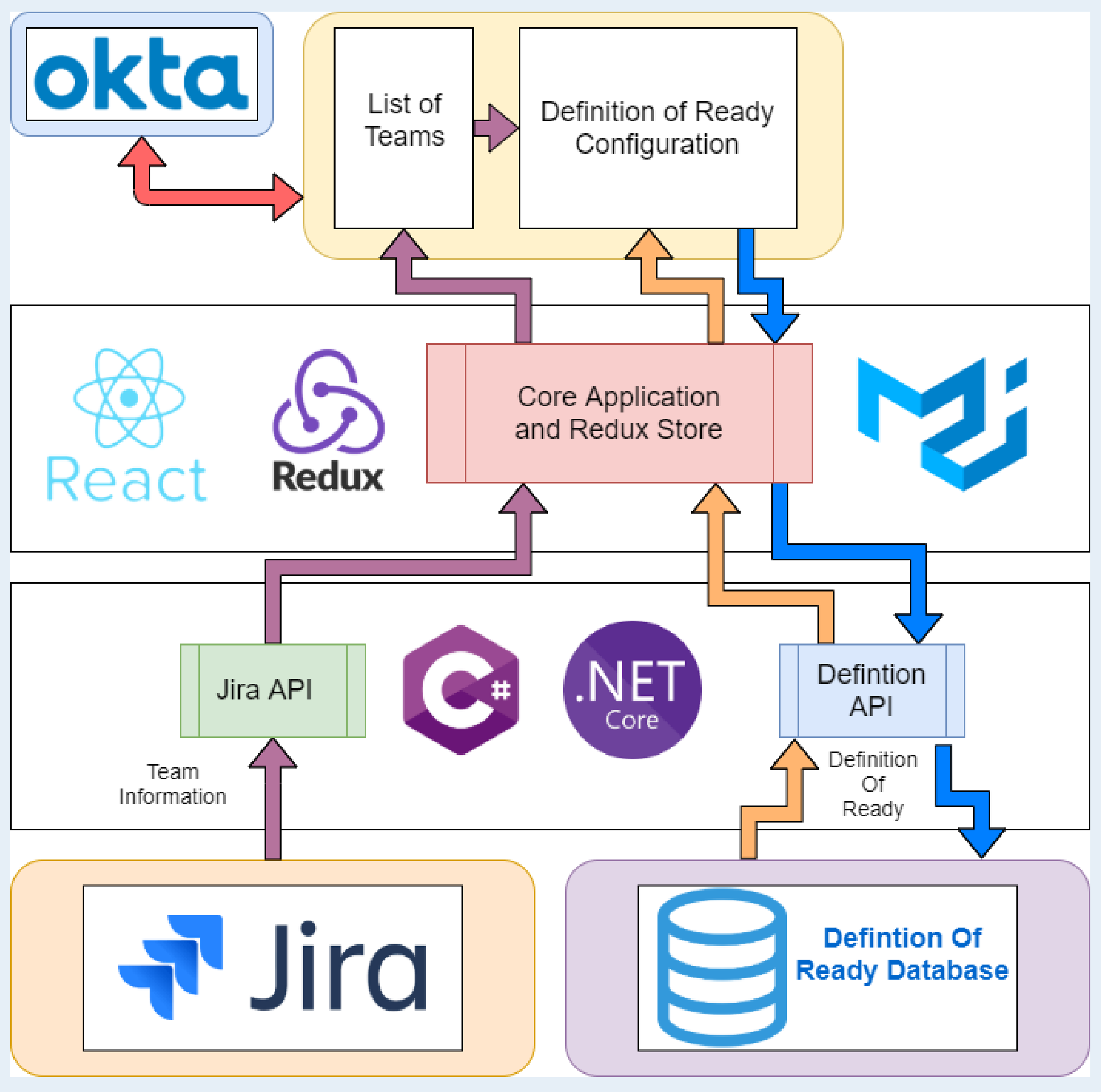


Top: Screenshot of Teams list page populated with Team data from Jira.

Right: Screenshot of a single Definition of Ready criterion being edited.



React and Redux architecture, with link to external C# API



Data and software flow diagram