TEST PLAN FOR JAMCo

ChangeLog

Version	Change Date	By	Description
1.0	2/4/23	Matt Kwiatkowski	Added Roles, Set Scope
1.1	2/15/23	Jared, Alex, Matt, Colin	Add Test Methodology Section
1.2	3/15/23	Jared Webber	Update to Include Sprint 3 Changes

1 Introduction

1.1 Scope

This document's scope covers all the functional and non functional features of JAMCo, including, but not limited to:

- Job Object Creation and Kanban Board View
- User Login/Registration
- Friend Requests & Interaction

1.2 Roles and Responsibilities

Name	Net ID	GitHub username	Role
Matthew Kwiatkowski	kwiatko2	Speuce	Devops Manager/Test Manager
Alex Kitt	kitta	drkitt	Configuration Manager
Jared Webber	webberj1	jaredwebber	Developer Manager/Code Review Manager
Colin Johnson	johns233	LokiFrostGiant	Database Engineer Manager

2 Test Methodology

2.1 Test Levels

2.1.1 Acceptance Testing

Basic automated acceptance testing runs as part of regression on every commit made to a pull request into `master`. Automated acceptance tests perform a subset of the below steps which outline complete manual acceptance testing, focusing on the most critical elements of each core feature.

Accounts

Login/Activation:

1. Load the App by connecting to localhost:8000 (or our production IP address)
2. A login popup should show
3. Click the 'signin with google' button
4. A new window should pop up, prompting for google sign in.
5. Sign in with a google account
6. After signing in, the login popup should go away
7. As a new user, the user information popup should show
8. Enter the first name 'test' in the first name field

9. Enter the last name ‘test’ in the last name field
10. Select the current day in the birthday field
11. Enter ‘test’ in the country field
12. Enter ‘test’ in the province/territory/region field
13. Enter ‘test’ in the city field
14. Leave the email field as-is
    a. The email field should be pre-populated with your email from your google account
15. Press ‘sign up’
16. The popup should now disappear

Edit User Data & Privacy Settings:
1. Load the app by connecting to localhost:8000 (or production)
2. Login with google credentials
3. Click the settings button at the top right of screen
4. User Info Modal should appear
5. User info & privacy options should not be editable
6. Click Edit button in bottom right
7. Both User Info & Privacy Checkboxes should now be editable
8. Edit any of the fields (making sure required fields are non-empty)
9. Click Save
10. Modal should close
11. Refresh the page
12. Open Settings Modal again
13. Verify updated values are present

Jobs

Create Application Card
1. *After completing account setup* as described in Login/Activation Acceptance Test
2. You should be presented with a blank Kanban Board with default columns (‘To Apply’, ‘Application Submitted’, ‘OA’, ‘Interview’)
3. Click ‘Add New Application’ button
4. A popup should appear in order to fill in application information
5. Fill ‘Position’ field with ‘test position’
6. Fill ‘Company’ field with ‘test company’
7. Click ‘Save’
8. Card should appear in leftmost Kanban column displaying the position & company

Edit Application
1. After completing Create Application Card steps, click on the newly created card
2. The same popup should appear, pre-populating Company & Position fields with inserted values
3. Modify ‘Position’ field to ‘new position’
4. Click ‘Add Deadline’ button
5. Set ‘Title’ field to ‘test’

6. Select a date from the ‘Date’ calendar picker
7. Set ‘Type’ field to ‘test’
8. Set ‘Description’ field to ‘test’
9. Set ‘Cover Letter’ field to ‘test’
10. Set ‘Notes’ field to ‘test’
11. Select ‘Application Submitted’ from ‘Status’ dropdown
12. Select ‘Save’
13. The card should now appear in the ‘Application Submitted’ column of the Kanban
14. The card should now display a light blue ‘test’ tag ● test
15. Click on the card to open the editing popup once again, and verify all fields are set as expected

Move Jobs Between Columns

1. Complete steps in ‘Create Application Card’ twice - ensure ‘Position’ and ‘Company’ fields are non-empty, and both cards have Status’ value set to ‘To Apply’
2. Begin by dragging the card at the bottom of the ‘To Apply’ column above the other card
3. The cards should switch positions without issue
4. Drag one card to ‘Application Submitted’
5. Click on the card moved to ‘Application Submitted’
6. The popup should show ‘Status’ field as ‘Application Submitted’
7. Click ‘Close’
8. Click on card in ‘To Apply’
9. The popup should show ‘Status’ field as ‘To Apply’

Edit Columns

1. After completing Move Jobs Between Columns steps
2. Click ‘Board Options’ button
3. A popup will appear listing the columns on the board numbered in the left to right order they appear
4. Edit the ‘To Apply’ column title to say ‘test’
5. Drag the ‘Application Submitted’ column to number 3 (below ‘OA’ column)
6. Click the trash button beside ‘Interview’ column
7. Click ‘Save’
8. Kanban should now display 3 columns: ‘test’, ‘OA’, ‘Application Submitted’

Friends

Request Friend:

1. User logs in to the app as normal, assuming the account has already been created.
2. User clicks the ‘friends’ button in the top right
3. The friends modal should open
4. Within the friends modal, user clicks ‘add friends’
5. The Search modal should open
6. The user types ‘speuce’ into the search box

7. The user clicks the search button (magnifying glass) in order to complete search
8. The user clicks the ‘add friend’ button on the user returned to whom they wish to send the friend request
9. The ‘add friend’ button should now display pending image

Accept Friend Request:

1. User logs in to the app as normal, assuming the account has already been created
2. The user clicks the Friends button in the top right
3. The friends modal should open
4. In the case where another user has requested to be friends with the current user, there will be a request displayed under the Friend Requests header
5. In order to accept, the User clicks the checkmark
6. The request will disappear, and the user will now be listed as a Friend, and will have 2 buttons listed, one to view their Kanban, one to remove friend
7. The user clicks on the Kanban button
8. The friends modal now disappears, and the Kanban board of the selected friend is now displayed in view-only mode

Deny Friend Request:

1. User logs in to the app as normal, assuming the account has already been created
2. The user clicks the Friends button in the top right
3. The friends modal should open
4. In the case where another user has requested to be friends with the current user, there will be a request displayed under the Friend Requests header
5. In order to accept, the User clicks the X
6. The request should disappear, and the user will not be listed as a friend

Request Cover Letter Review:

1. User logs in to the app as normal, assuming the account has already been created. The user should have at least one Job added, and at least one Friend.
2. The user clicks on a job in their Kanban board
3. The job detail modal should appear
4. The user clicks the Request Review button
5. The Review Request modal should appear
6. The user selects at least one friend to send the request, and adds a non-empty message
7. The user clicks send, to send the review request

Review Cover Letter:

1. User logs in to the app as normal, assuming the account has already been created. The user should have a pending cover letter review request.
2. The user clicks the Inbox in the top right
3. The Cover Letter Review Inbox modal should display
4. The requested review should be displayed under the Review Requests header

5. The user clicks review
6. The Review modal should appear
7. The user reads the cover letter, leaves a comment/review, and clicks send
8. The review request should be hidden

2.1.2 Integration Testing

Integration Tests for each main feature interact with the view layer (api layer) and validate requests are handled as intended, working with both valid and potential invalid request bodies. Integration Tests can be found in `backend/tests/` directory.

Accounts:

- test create account
- test update account
- test invalid account update
- test get privacies
- test invalid get privacies
- test update privacies
- test invalid update privacies
- test invalid oauth properties

Jobs:

- test create job
- test create job with error
- test update job
- test update job with invalid field error
- test update job with invalid id error
- test get minimum jobs
- test get minimum jobs nonexistant user
- test get job by id
- test get job by id with invalid user error
- test get job by id invalid job error

Friends:

- test create friend request valid
- test create friend request error
- test accept friend request valid
- test accept friend request error
- test deny friend request valid
- test deny friend request error
- test get friend requests status valid
- test get friend requests status error
- test remove friend valid
- test remove friend invalid
- test create review request

- test create review request with error
- test get review requests for user
- test get review requests for user with error
- test create review
- test create review with error
- test get reviews for user
- test get reviews for user with error

2.1.3 Unit Testing

Tests in MainView.test.js cover many components of the app, including tests for many of the main features.

Accounts

<u>Frontend</u>

AccountSetupModal.vue

- populates with default values when no props provided
- displays error when first name is empty & sign up is pressed
- displays error when last name is empty & sign up is pressed
- displays error when email is empty & sign up is pressed
- displays error when country is empty & sign up is pressed
- displays error when workField is empty & sign up is pressed
- emits updateUser when sign up clicked

GoogleSignin.vue

- calls the initialize function on mounted
- calls the onSignin method with the response and makes the post request

UserInfoModal.vue

- populates with default values when no props provided
- displays error when first name is empty & saveChanges is pressed
- displays error when last name is empty & saveChanges is pressed
- displays error when email is empty & saveChanges is pressed
- displays error when country is empty & saveChanges is pressed
- displays error when workField is empty & saveChanges is pressed
- emits updateUser when saveChanges clicked

<u>Backend</u>

test_business.py

- test user creation
- test update user birthday
- test update user birthday with invalid date
- test get all column
- test renaming column

- test reorder column with out of bounds
- test deleting a column
- test updating columns with a non existent user
- test update column with empty payload
- test get privacies
- test update privacies
- test invalid update privacies

test_query.py
- test creating an account and getting an account
- test User existence verification
- test updating Account info
- test invalid attempts to update Account info
- test create privacies
- test get privacies
- test invalid privacies get
- test update privacies
- test invalid update privacies

test_views.py
- test creating an account that sends the appropriate info to the business layer
- test updating an account
- test updating an account with invalid info
- test get user columns
- test get columns with invalid user id
- test create column
- test create column with invalid fields
- test rename a column
- test reorder columns
- test reorder columns with out of bounds request
- test delete a column
- test create column with empty payload
- test invalid OAuth token validation
- test update privacies
- test invalid update privacies
- test get user privacies
- test invalid get user privacies

Jobs

<u>Frontend</u>
JobDetailModal.vue
- populates with default values when no props provided
- displays error when position is empty & save is pressed
- displays error when company is empty & save is pressed

- displays error when deadline fields empty & save clicked
- emits close when close button clicked
- adds deadline to list when Add Deadline clicked
- deletes deadline from list when delete clicked
- deletes correct deadline from multiple when delete clicked
- saves job fields when save clicked
- updates deadline when modified
- updates correct deadline in list when modified

JobTrackingView.vue

- renders the correct number of jobs
- opens JobDetailModal when card clicked
- assigns selectedJob when card clicked
- opens JobDetailModal when New Job clicked
- closes the detail modal when the close event is emitted
- opens ColumnOptionModal when Board Options clicked
- closes the board option modal when the close event is emitted
- updates colList when param passed to updateColumns
- updates jobsByColumn when job edited

JobDetailDeadline.vue

- emits updateDeadline when datePicker updated
- emits updateDeadline when datePicker cleared
- emits updateDeadline event when title changed
- emits deleteDeadline when the remove button is clicked
- populates deadlineModel with props

ColumnCard.vue

- emits updateColumn event when name changed
- emits deleteColumn when delete clicked

ColumnOptionModal.vue

- emits updateColumns when saveClicked
- emits close when close button clicked
- deletes column if empty
- does not deleteColumn if non-empty
- sets invalidColumns to true if name empty
- updates column when passed
- limits number of columns to 8
- requires minimum of 1 column

KanbanBoard.vue

- has the correct number of columns
- updates the column of a job when it is moved
- emits showDetailModal when card clicked

Backend

test_business.py

- test Job creation, valid and invalid
- test get entire single Job
- test get single Job using invalid job id
- test get single Job using user_id not linked to Job
- test updating a Job with valid info
- test invalid attempts to update a Job
- test getting all Jobs for a user_id, ensuring all and only the desired Jobs are fetched

test_query.py

- test Job creation, valid and invalid
- test getting an entire single Job
- test getting single Job with invalid job id
- test updating a Job with valid info
- test invalid attempts to update a Job
- test getting all Jobs for a user_id, ensuring all and only the desired Jobs are fetched

test_views.py

- test creating Jobs, valid and invalid
- test fetching all Jobs for a specific User
- test updating one of a User's Jobs
- test getting a specific Job's complete information
- test getting single Job with invalid job id

Friends

Frontend

FriendModal.vue

- emits fetchUserData event after accepting friend request
- emits fetchUserData event after denying friend request
- emits fetchUserData event after removing a friend
- emits events when viewFriend called

IncomingReviewsModal.vue

- emits close when close button clicked

ReviewModal.vue

- emits close when close button clicked

ReviewRequestModal.vue

- populates with default values when no props provided
- emits close when cancel button clicked

SearchFriendsModal.vue
- searches for users when triggerSearch method is called
- does not include current user in search results
- sends friend request when sendFriendRequest method is called
- removes friend when removeFriend method is called
- triggers friend Kanban View when viewFriendKanban is called

Backend

Backend Friends tests are contained within account django app (e.g. tests listed under test_views.py, will by found alongside the unit tests listed under Account above)

test_views.py:
- test remove friend
- test invalid remove friend
- test search users by name
- test create friend request valid
- test create friend request error
- test accept friend request valid
- test accept friend request error
- test deny friend request valid
- test deny friend request error
- test get friend requests status valid
- test get friend requests status error
- test get friend data

test_business.py:
- test invalid add friend
- test invalid remove friend
- test valid remove friend
- test searching users
- test create friend request valid
- test create friend pending request exists
- test create friend request already friends
- test create friend request user not searchable
- test create friend request sub search error
- test accept friend request valid
- test accept friend request no pending request
- test accept friend request add friend error
- test deny friend request valid
- test deny friend request no pending request
- test get friend requests status
- test get friend data

test_query.py:
- test add friend

- test invalid add friend
- test remove friend
- test invalid remove friend
- test create friend request
- test accept friend request
- test deny friend request
- test get friend requests status empty
- test get friend requests status sent none
- test get friend requests status received none
- test get friend requests status populated
- test pending friend request exists true request id
- test pending friend request exists false request id
- test pending friend request exists true user pair
- test pending friend request exists false user pair
- test are friends true
- test are friends false
- test get friend data

2.1.4 Regression Testing

Regression testing will be completed by running our Unit/Integration/Acceptance tests as a part of our CI pipeline with GitHub Actions, for every commit merged into the master branch. Additionally, Acceptance tests will be run within the CD deployment pipeline via GitHub Actions.

2.2 Test Completeness

- 100% code coverage (mandatory for this project) for the front and back end
  - Code Coverage should be calculated automatically as a part of our CI
  - Code coverage excludes boilerplate files generated by project creation (such as manage.py)
- All Manual & Automated Test cases executed for each deployment
- All open bugs are fixed or will be fixed in next release

3 Resource & Environment Needs

3.1 Testing Tools

- Vitest
- Unittest (python)
- Playwright
- Coverage.py

3.2 Test Environment

The minimum hardware requirements for testing the Application are as follows:

- At least 8GB of RAM
- Docker
- Docker Compose
- Python 3
- OS Requirements:
  - 64-bit OS
  - Windows 10 (With WSL 2) or above OR
  - macOS Monterey (12.5.1) (**Preferred**) or above

4 Terms/Acronyms

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test
JAMCo	Job Application Management Console
OAuth	Open Authorization