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 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
- Friends

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

Accounts

<u>Login/Activation:</u>

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

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

1. User logs in to the app as normal, assuming the account has already been created.
2. User clicks the ‘friends’ icon in the top right
3. The friends modal should open
4. Within the friends modal, user clicks ‘add friend’
5. the user types ‘speuce’ into the ‘find friend’ box
6. the user clicks the ‘send’ button to send the friend request
7. Verify that the friend request was sent

2.1.2 Integration Testing

Integration testing is reserved for Sprint 3. We plan to make the following tests:

- Create an account from the view layer and verify that it exists in the persistence layer with the correct fields
- Create an account from the view layer and verify that the business layer added the default kanban columns to the persistence layer
- Modify an account from the view layer and verify that those changes are reflected in the persistence layer
- Log in on the view layer, ensure that cookies are set such that the user stays logged in
- Modify the user’s job applications (create two applications, modify one, delete the other) on the view layer and ensure that the end result of those changes (one application) appears in the persistence layer
- Attempt to create an incomplete job application (missing required fields) on the view layer, ensure that the business layer returns an appropriate error and the persistence layer does not make any incomplete changes
- Reorder a user’s kanban columns from the view layer, make sure the persistence layer reflects those changes

- Send a friend request from the view layer, ensure that the request exists in the persistence layer and is visible on the view layer for another user
- Accept a friend request from the view layer and ensure that the other user shows up as one of their friends in the persistence layer
- Reject a friend request from the view layer and ensure that the request is deleted for the recipient on the persistence layer but no activity is visible from the other user's perspective on any layer

2.1.3 Unit Testing

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

<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_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_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

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

<u>Backend</u>
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

<u>Frontend</u>
FriendModal.vue
- opens when the friends icon is clicked
- makes a request to /friends/get_pending on load
- lists all pending friend requests
- on clicking ‘accept’ on a friend request, make a request to /friends/request_update
- on clicking ‘deny’ on a friend request, make a request to /friends/request_update
- makes a request to /friends/list_all on load
- lists all current friends
- makes a request to /friends/activity with parameter page=1 on load
- on clicking ‘next page’, the page number is incremented
- on clicking ‘next page’ another request is made to /friends/activity
- on clicking ‘close’, a ‘close’ event is emitted

2.1.4 Regression Testing

Regression testing will be completed by running our Unit/Integration 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