CS 426: Software Engineering

Project Assignment 3

University of Nevada, Reno
Department of Computer Science & Engineering
Instructors: David Feil-Seifer, Devrin Lee
Team 13
Authors: Matthew Alighchi, Guillermo Del Valle, Sherman Lee, Kyle Respicio
External Advisors: Dr. Casey Lynch, Vivan Zavataro, Tyler Brownlow-Calkin
March 12, 2021

0 Table of Contents

1 Abstract

The project proposal will encompass a museum guidance robot with a focus on accessibility for participants through remote services or guided interactivity in place of human guides. The main purpose of this project is to address the current concerns regarding COVID-19 with the limit to the number of participants and employees in the space. ADA museum guidelines will also be addressed to accommodate visual and auditory needs. This project intends to develop a functioning general use guidance robot with general navigation and collision avoidance to assist in various museum information delivery with mobile and web user interfaces for museums to modify their exhibits in the Reno-Tahoe area with a focus on those in the University of Nevada, Reno. The purpose of this document is to outline acceptance criteria via user stories and incorporating the testing strategy and plan for the project.

2 Recent Project Changes

Established better organization and documentation for database and API routes
Added authorization guard with necessary JWT token to access routes and editor site
Reconfigured editor UI to be more user friendly for museum staff
Incorporated mobile frontend interface for museum-goers to interact with the robot
Configured flask server to be compatible with the robot

There have been no major recent changes.

3 User Stories and Acceptance Criteria

3.1 Robot Communication with Museum-Goers

1. As a museum-goer, I want to ask the robot docent questions so that I can get more information about an art piece.
    - The mobile app and the robot are connected.
    - I will click the questions button on the mobile app to ping the robot.
    - After the robot is pinged, the robot physically responds and the mobile app populates the questions on the screen.
    - I then click on any question and they are provided a transcript on their mobile device and the robot verbalizes the answer.
    - I click out of the questions menu.
2. As a museum-goer, I want to get directions from the mobile app so that I can find the exhibit I'm looking for.
    - I click the directions button on the mobile app.
    - The mobile app populates the screen with a map of the museum.
    - Then I find the location they need and click off the menu.

3.2 Robot Navigation

1. As a museum staff member, I want the robot to follow the tour that I map out in the web application so that it can perform docent services.
    - The robot can query the server for information about the tour.
    - The robot can see which pieces are part of the tour.
    - The robot can obtain the coordinates of each piece and navigate there.
2. As a museum staff member, I want to see the robot's map so that I can design tours around the museum.
    - The robot should be able to create a bitmap of the museum.
    - The robot should be able to use the API to post the image to the database.
    - The website should be able to display that image.

3.3 Authentication for Map Editor with Museum Staff

1. As a museum staff member, I should be able to log into the map editor platform and access and edit pertinent information about the museum.
    - Scenario: System user signs in with valid credentials
    - When logged out, I can go to any page and be redirected to the login page where I can enter the proper credentials (username/email and password) and click the "sign-in" button, the system signs me in.
    - A JWT token is saved onto their computer for persistence.
2. As a staff member who has recently logged in without logging out, I should be able to access the website without having to log in again.
    - Scenario: User with valid JWT token enters any page of the map editor
    -
3. As a staff member, I should be able to create new accounts for other staff members, so they can also access the same map information.
    - Scenario: Logged in user creates a new account

- When I'm on my settings page, I can click on the "create new account" button, a modal with a form will pop up asking me for the new staff member's information.
- Filling out the form with their email and privilege information and then clicking the "create user" button will make a new user with the information that the form filled out.
- The email address used will get a special link that will take the user to a new page where they can create a custom password.
    i. On the special login page, the user will see their email address and will be able to fill out a password field and "confirm password" field. Clicking the "confirm password" button will change their password.

4. As a staff member, I should be able to change my password.
    - Scenario: User forgets their password and requires an email to change it
    - When I'm on the login page and I click on the "forgot password" button, a new form will pop up asking me to for my email. If the email is valid, then the email address will get an email with a special link to take the user to a new page where they can create a custom password.
        i. On the special login page, the user will see their email address and will be able to fill out a password field and "confirm password" field. Clicking the "confirm password" button will change their password.
5. As the main staff member of the museum, I should be able to delete the old accounts of retired staff members.
    - Scenario: User wants to delete an inactive user's account
    - Given that I am an admin, on the settings page I should be able to see a list of different users that belong to the museum. If I click on the three dots button, a drop-down with the option of delete should be displayed.
    - Once I click the delete button, a popup modal should display asking if I truly want to delete the member; if so, I can input my password and click "permanently delete user" to make the final decision to delete the existing user.
6. As a logged-in user, I should be able to log out.
    - Scenario: Logged-in user logs out
    - When I'm on any page, I can click on my profile icon on the top right where a drop-down menu pops up.
    - The drop-down provides a log-out button, at which point I can click on this button and log out of my account, bringing me back to the login page.

3.4 Frontend Map Editor with Museum Staff

1. As a museum staff member, I want to edit my exhibit with different or new information like the title, subtitle, description, dates, etc. on an existing exhibit.
    - When I click the exhibit I want to modify on the sidebar, a modal should pop up with the fields that I am looking to change with the current information that already exists since my last edit.

    - I should be able to edit the fields and the information should save for next time.
2. As a museum staff member, I want to be able to create new exhibits.
    - When I click the add button on the sidebar, a new card is added to the sidebar and a new modal should pop up with empty fields.
    - I should be able to edit the fields and the information should save for the next time I want to edit it.
3. As a museum staff member, I want to be able to edit the pieces within each exhibit.
    - When I click the exhibit I want to modify on the sidebar, a modal should pop up with the piece section being at the end organized in the form of cards.
    - When clicking on the appropriate card, it should expand and have fields with the current information that already exists since my last edit.
    - I should be able to edit the fields and the information should save for next time.
4. As a museum staff member, I want to be able to add new pieces within each exhibit.
    - When I click on the floating add button on the map editor, it should pop a modal with empty fields to fill out.
    - After filling in the fields, I specify which exhibit to add to, and when I click submit, it should add to that exhibit in the sidebar under the piece section.

4 Testing Workflow

4.1 Robot Communication with Museum Goers

Scenario: Museum Goer asks the Robot Docent a question.

a. Happy Workflow
    1. The museum-goer clicks the questions tab on the mobile app.
    2. The Robot Docent physically responds and the mobile app is populated with the questions.
    3. The museum-goer clicks on one or more questions.
    4. The Robot vocalizes the answer and the transcript is displayed on the mobile device.
    5. The museum-goer closes out of the mobile app and the Robot Docent resumes its operation.
    6. This workflow is considered validated.
b. Unhappy Workflow
    1. The museum-goer accesses the mobile app outside of the Robot Docent's connection parameter: The questions tab will instead bring up a loading tab that will indicate to the museum-goer that they are not connected and will wait for the connection to be established.
    2. The museum-goer opens the questions menu but doesn't ask any questions and exits: The robot will pause and resume operation.
    3. The museum-goer does not close out of the questions tab: If the museum-goer does not give the robot a closeout response, then it will resume operation after a timeout.
    4. Multiple users ping the Robot Docent at the same time: The museum goers' requests will go in a queue and the robot will respond in the order of the queue.
    5. This workflow is considered validated.

4.2 Robot Navigation

Scenario: Robot initiates tour and begins navigation.

a. Happy Workflow
    1. Tour is triggered.
    2. Robot switches to "Start State" (non-interruptible), letting museum-goers know that the tour will start soon.
    3. Robot initializes tour information by querying the server.
    4. After a minute, the tour starts. Robot queries the first piece.
    5. Robot switches to "Motion State" (non-interruptible), activating navigation stack.
    6. Robot navigates to the coordinates of the first piece.
    7. Robot switches to "Present State" (non-interruptible). Presents information about the piece.
    8. Robot switches to "Interactive State" (interruptible). Allows users to ask questions.

9. After a minute of no interaction, the robot switches back to “Motion State” (non-interruptible) and repeats this process with the next piece, until the tour ends.
10. Once the tour ends, the robot navigates to the original start point.
11. This workflow is then considered validated.

b. Unhappy Workflow
    1. Robot is in “Motion State” (uninterruptible) and navigating to the next piece.
    2. Obstacles are blocking the path to the next piece.
    3. After some attempts, the robot gets stuck.
    4. Robot goes into “Error State” (uninterruptible) and requires external assistance.
    5. Once obstacles are removed, “Error State” (uninterruptible) is manually deactivated. The robot picks up where it left off.
    6. This workflow is considered validated.

4.3 Frontend Map Editor with Museum Staff

Scenario: Museum staff member adds a new exhibit

a. Happy Workflow
    1. The museum staff member clicks on the add button on the sidebar.
    2. The editor will update the state with a new instance of an exhibit component prompting a modal with empty fields to fill out.
    3. The museum staff member fills in the required title field and all other fields to save to the newly created exhibit.
    4. The editor will save all fields when the user clicks away from the selected field.
    5. The museum staff member will finish and exit the exhibit modal.
    6. The editor will save all information and send a post request to the server and update the database.

b. Unhappy Workflow
    1. Users will try to exit the modal after creating a new exhibit without filling out any information, especially the required title field. An alert modal will appear to notify the required title field and ensure if the user wants to remove the empty exhibit. If the user confirms to exit, the new exhibit is deleted.

4.4 Backend Server for Map Editor with Museum Staff

Scenario: Authentified users will be able to access all backend routes associated with the map editor.

a. Happy Workflow
    1. The museum staff member gets onto the login page and inputs in their correct password and email address.
    2. Upon clicking the login button or pressing the “enter” button, the user will be taken to their homepage.
    3. The user can observe that the top right corner displays their icon to indicate that they have successfully logged in.

4. The user will also be able to enter any page and edit information without receiving a 401 error indicating that they are not authenticated.
5. Unhappy Workflow
6. An unauthorized user tries to access another page other than the login page.
7. At this point, the user will get kicked out of the page and onto the login page.
8. The backend will block any routes (besides any authorization route) from being accessed. The service will respond with a 401 (Not Authorized) error.
9. The user cannot access any information without proper credentials.

Scenario: Staff Members try to save information to the database such as adding a new exhibit or piece.

a. Happy Workflow
    1. The museum staff member completes a form on the front end such as the form to create a new exhibit.
    2. Upon clicking the update/edit/add/delete button, their request will enter through the backend.
    3. If authorized, the backend will parse through the different values to check if they are valid inputs. For example, a string field will not accept an int/float datatype to be saved.
    4. Moreover, the data is cleaned so that improper data (such as a SQL injection) will be rejected.
    5. Once the inputs are validated, they will be saved in the database for persistence.
    6. The server will respond with a 200 response message indicating that all has gone well.
    7. The user will be able to note these changes by seeing that all the information is correct, meaning updates have been processed as expected.
b. Unhappy Workflow
    1. Say the user inputs invalid data such as an attempted SQL injection in one of the fields.
    2. The input gets processed through and notices that there is something fishy about the input.
    3. On one hand, the input is saved as a string literal in which the database saves the information. Here, we send a 200 response code; however, the input might not be as what the user may have expected, in which case they can re-update the form for more accurate results.
    4. On the other hand, the database cannot properly save the information and so an error response (perhaps a 403 forbidden response) is sent out.

5 Testing Strategy

For the testing strategy, we will be conducting automated unit tests at a low level and manual acceptance testing at a high level. The automated unit testing will be to ensure that there are no low-level functionality errors while the manual acceptance testing will be performed by the stakeholders of the project. Presumably, the automated unit testing will ensure that there are no errors within the capabilities of the system. The manual acceptance testing will not be checking for correctness but rather, that the capabilities of the system are what the stakeholders expect.

There are four main subsystems to be tested: the robot's server, the robot's user interface, the web application's server, and the web application's interface. Each of these subsystems will have a suite of automated unit tests that will ensure that the low-level functionality works well. These unit tests are elaborated on the testing plan diagram.

The unit tests fall into one of four categories, a category for each subsystem. The tests for the robot's server will be testing that the base ROS nodes of the robot properly work, communicate with each other and that the robot can successfully query the API that the team has built. The tests for the robot's user interface will be testing that the mobile website can display information about the robot's current state and that the robot reacts to the mobile website's actions. The tests for the web application's server will be testing that the different API routes work, and that information can be successfully posted to the database. This can be done with python's unittest package, which works well with flask. The tests for the web application's interface will be testing that the web application's GUI is functional and activates the database and robot when needed. Testing for web applications can be done with Selenium. Specific testing procedures can be found in our testing diagram below.

Since the person who is in charge of a specific subsystem created the user story, the person running the unit tests will be someone else. Sheman Lee will be running the unit tests for the robot's user interface, Kyle Respicio will be running the tests on the robot's server, Matthew Alighchi will be running the tests on the web application interface and Guillermo Del Valle will be running the tests on the web application server. The team plans on running these unit tests together when all the functionality of the product has been finished. The current timeline is expecting the core functionality to be done by early to mid-April.

When defects are found within the unit tests, they will be posted to the team's agile board and handled by the person responsible for that subsystem. Once the defects are reported fixed, the person responsible for testing that subsystem (not the person responsible for the subsystem) will run the tests again.

The project will be considered complete once all of the unit tests are passed and the stakeholders give the final approval after an extensive acceptance test session with them. As stated above, the unit tests will ensure no bugs are in the final product and the acceptance test will ensure that the product is what the stakeholders desired.

Test No.	Test type	Target File or Screen	Test Name	Purpose of Test	Test Data or Situation	Expected Result	Actual Result	Outcome and Actions Required
1.	Unit	Robot-Docent-Mobile	Robot Docent Responds to Mobile App	Tests that the R.D and mobile app are properly connected, and when questions are clicked, the R.D. responds accordingly	Database is populated with "exhibit 1 Q1, Q2, Q3 A1, A2, A3" 1. User clicks question button 2. User clicks question Q2. 3. User closes menu	1. The Robot Docent physically responds and the mobile app is populated with questions Q1, Q2, Q3 2. "A2" will appear on the mobile app and the robot will verablize "A2" 3. The robot will resume its operation.	No tests ran yet	Implementation needed
2.	Unit	Robot-Docent-Mobile	Mobile App Synchronize with Database	Test that when the questions and answers in the database are updated, the mobile app gets the updated data	Database is populated with "exhibit 1 Q1, Q2, Q3 A1, A2, A3" and then changed to "exhibit 1 Q3, Q4, Q5 A6, A7, A8"	The mobile app will refresh in real-time to the new question-answers in the proper order.	No tests ran yet	Implementation needed
3.	Unit	Robot-Docent-Mobile	Mobile App Displays Museum Map	To test if the mobile app will display the map of the museum on request.	The user clicks the map tab on the mobile app.	The screen is populated with the map of the museum.	No tests ran yet	Implementation needed
4.	Integration	Map-Editor	Map Editor Updated by Database	Test when the database is updated, the information pulled is correctly sent to the map editor fields for exhibits and pieces.	Manually update the database through SQL.	The newly modified information should update in the map editor in the text fields.	No tests ran yet	Implementation needed
5.	Integration	Map-Editor	Database Updated by Map Editor	Test when the map editor is properly saved and updated, the information is sent to the server and updates the database.	Either add a new exhibit or edit the title of a new exhibit to save to the map editor.	The database should receive the information via the server and update the 'exhibit' table based on the new exhibit or new title.	No tests ran yet	Implementation needed
6.	Unit	robotic_docent_core/server.py	Robot Server API Test	Tests all of the routes within the robot's server	Database and Flask Server should both be interacted with and return the proper information	POST and GET methods all return "success" messages and update the database as needed.	No tests ran yet	Implementation needed
7.	Unit	robotic_docent_core/tour_functionality.py	Robot Tour Finish Test	Tests that the robot can successfully traverse through all pieces in a given tour and return to starting position	Database should be interacted with to get the tour length and get the start position	Robot successfully marks each piece as visited and returns to start position	No tests ran yet	Implementation needed

8.	Unit	robotic_docent_core /tour_functionality.py	Robot State Test	Tests that all states are mutually exclusive and that the tour_functionality file can activate them properly.	The ROS nodes representing the states will all be interacted with	Robot successfully cycles through each state, ensures no other state can be activated and verifies functionality. Sates are Error, Start, Motion, Present and Interactive.	No tests ran yet	Implementation needed
9.	Unit	robotic_docent_core /motion.py	Robot Navigation Test	Tests that the robot can properly navigate to the appropriate piece.	The ROS node with navigation will be interacted with	Robot successfully navigates to the given coordinates	No tests ran yet	Implementation needed
10.	Unit	robotic_docent_core /present.py	Robot Presenting Test	Tests that the robot can properly broadcast the appropriate piece description.	The ROS node with communication will be interacted with.	Robot successfully narrates the description and exits out of the Present state.	No tests ran yet	Implementation needed
11.	Unit	robotic_docent_core /tour_functionality.py	Robot Tour Finish Test	Tests that the robot can successfully traverse through all pieces in a given tour and return to starting position	Database should be interacted with to get the tour length and get the start position	Robot successfully marks each piece as visited and returns to start position	No tests ran yet	Implementation needed
12.	Unit	robotic_docent_core /tour_functionality.py	Robot State Test	Tests that all states are mutually exclusive and that the tour_functionality file can activate them properly.	The ROS nodes representing the states will all be interacted with	Robot successfully cycles through each state, ensures no other state can be activated and verifies functionality. Sates are Error, Start, Motion, Present and Interactive.	No tests ran yet	Implementation needed
13.	Unit	robotic_docent_core /motion.py	Robot Navigation Test	Tests that the robot can properly navigate to the appropriate piece.	The ROS node with navigation will be interacted with	Robot successfully navigates to the given coordinates	No tests ran yet	Implementation needed
14.	Unit	robotic_docent_core /present.py	Robot Presenting Test	Tests that the robot can properly broadcast the appropriate piece description.	The ROS node with communication will be interacted with.	Robot successfully narrates the description and exits out of the Present state.	No tests ran yet	Implementation needed
15.	Unit	backend/routes /user.py	Authentication Function Test	Tests that authentication validates the correct email and password combination.	Test 1: Correct Email and Password Test 2: Incorrect Email and Correct Password Test 3: Correct Email and Incorrect Password Test 4: Incorrect Email and Password	Test 1: Response with a success message along with a valid JWT Token Test 2,3,4: Response with error message saying that email and password combination don't match	No tests ran yet	Implementation needed
16.	System Integration	backend/routes	Auth Guard Test	Routes have an auth guard protecting them from being accessed if not authentified	Test 1: Authorized Test 2: Not Authorized	Test 1: Route can process normally i.e. a get function will response correct JSON in return Test 2: Return a 401 Unauthorized	No tests ran yet	Implementation needed

Fig. 1: Test Plan

6 Time Worked on Project Concept

Matthew worked a total of 2 hours on sections 3, 4, and 5 for the museum-goer subsections.
Guillermo worked a total of 2 hours on sections 3, 4, and 5 for the robot server subsections
Sherman worked a total of 2 hours on sections 3, 4, and 5 for the front-end map editor subsections, abstract, recent project changes, and formatting.
Kyle worked a total of 2 hours on sections 3, 4, and 5 for the back-end map editor subsections.