PatrolBot

Acceptance Criteria and Testing Strategy and Plan

University of Nevada, Reno
Department of Computer Science and Engineering

Team 08
Jesus Aguilera, Brandon Banuelos, Connor Callister, Max Orloff, Michael Stepzinski

Instructors: Devrin Lee, Dr. David Feil-Seifer, Vinh Le

Advisors: Dr. Hung La, Dr. Alireza Tavakkoli, Officer Matthew Stewart

March 11th, 2022

Abstract

The PatrolBot project is about creating a patrol robot. The PatrolBot project is important because we aim for our system to be an essential tool for campus police forces and other small security operations in properly monitoring their grounds. The project will utilize a robot connected to the internet, a web server, a website, and machine learning models. Our team must create a web server, control a robot through the internet, build the front-end of our website, connect everything within the backend, and train then deploy the machine learning models. The purpose of this document is to describe our project's acceptance criteria and testing strategy and plan.

Project Updates and Changes

Since the last update, we have made major amounts of progress in the project. We have created a basic working system of the components we chose. We reached the point where each piece of the system could work on it's own, using manual inputs and data feeds. Once each piece could work independently, we created the backend systems to allow them to work together, where they feed data into each other and call functions belonging to one another. We have created a front end for our website using JavaScript, CSS, and HTML, and a back end using Django on Python 3.7. Our website is hosted using AWS, and we purchased a domain name at patrolbotdash.com. We have acquired a robot, used ROS to control it, and used AWS-IOT to send commands to it over the internet. On the robot, we have a camera attached to our raspberry pi. The video stream produced by the camera uses AWS Kinesis. Our customized object detection and threat detection models are hosted using AWS on the back end of our server.

The only change that we have made to our project is related to our robot. Since we are unable to get the devices needed for area mapping and obstacle avoidance, we cannot truly call our robot semi-autonomous since it will need a human operator to control it, instead of our original plan where the device would patrol on it's own and alert an operator. This has not been a change in the overall operation of our project, just a change in our robot approach.

User Stories and Acceptance Criteria

Robotic Software:

User Story 1.1:
As a user, I want to see the robot's GPS coordinates so I can know where it is.

- AC1: The Raspberry Pi is powered up.
- AC2: The GPS module is connected to the Raspberry Pi.
- AC3: The Raspberry Pi is connected to the internet.
- AC4: The Raspberry Pi is communicating with AWS-IOT.
- AC5: The predicted GPS location is displayed on the website.

Robotic Hardware:

User Story 2.1:
As a user, I want to be able to view a video feed from the camera attached to the robot so I can see what the robot sees.

- AC1: The Raspberry Pi is powered up.
- AC2: The camera is connected to the Raspberry Pi.
- AC3: The Raspberry Pi is connected to the Internet.
- AC4: The Raspberry Pi is communicating with AWS Kinesis.
- AC5: The Kinesis video stream is displayed on the website.

User Story 2.2:
As a user, I want to be able to move the robot with manual inputs so I can control it myself or get it unstuck.

- AC1: The Robot is powered up.
- AC2: The Raspberry Pi is powered up.
- AC3: The Raspberry Pi is connected to the Internet.
- AC4: The Raspberry Pi is communicating with the Robot.
- AC4: The Raspberry Pi is communicating with the webpage.
- AC5: The webpage movement input is accepted by the Pi.

Web Server & UI/UX:

User Story 3.1:
As a security team member, I want to be able to see a history of malicious objects detected so I can keep track of what has been detected in the past few minutes.

- AC1: The website is online and fully functional.
- AC2: There is a button that will provide a history of objects detected.
- AC3: A CSV file containing a history of objects detected will be available to download.
- AC4: The history of objects detected will contain a timestamp of when it was detected by the model.

User Story 3.2:
As a user, I want a dashboard with a defined workflow with an intuitive streamline of all subsystems so that I will be able to complete tasks in a given process.

- AC1: The website is online and fully functional.
- AC2: The dashboard overview page is separated with each container more focused on one subsystem.
- AC3: The subsystems containers are well preserved and visually appealing.

Detection Model:

User Story 4.1:
As a security team member, I want to see when threatening objects such as bolt cutters and angle grinders are seen by the camera attached to the PatrolBot so I can ensure the area being patrolled is safe.

- AC1. The camera functions properly on the robot.
- AC2. The camera stream is displayed on the main page of the application.
- AC3. When threatening objects are within view of the camera, the object detection model detects them.
- AC4. When objects are detected by the model, their bounding boxes are displayed over the camera feed.

User Story 4.2:
As a security team member of a busy campus, I want to be able to disable the overlay of bounding boxes on the camera feed for detected persons so that I can effectively survey the campus.

- AC1. The camera functions properly on the robot.
- AC2. The camera stream is displayed on the main page of the application.
- AC3. Model Options button is displayed on the main page

- AC4. Checkboxes with possible objects to display bounding boxes for are displayed on Model Options page
- AC5. When the checkbox next to “People” is unchecked, bounding boxes for detected people will not be displayed over the camera feed.

User Story 4.3:
As the head of a security team at UNR, I want to be able to have the least laggy camera feed by disabling the object detection model entirely so that my team can respond to suspicious activity as fast as possible.

- AC1. The camera functions properly on the robot.
- AC2. The camera stream is displayed on the main page of the application.
- AC3. Model Options button is displayed on the main page of the application.
- AC4. A checkbox next to “Run Object Detection” is displayed on the Model Options page.
- AC5. When the checkbox next to “Run Object Detection” is unchecked, the model will not run on the camera feed.

Threat Prediction:

User Story 5.1:
As a security team member, I want to know when aggressive behavior is detected on campus, so that I can respond as quickly as possible.

- AC1. The camera functions properly on the robot.
- AC2. The camera stream is displayed on the main page of the application.
- AC3. The model outputs onto the log whether or not behavior is aggressive.

User Story 5.2:
As a security team member, I want to know when malicious objects are near bikes, so that the activity can be addressed as soon as possible.

- AC1. The camera functions properly on the robot.
- AC2. The camera stream is displayed on the main page of the application.
- AC3. The model outputs onto the log whether or not the intersection over union of bounding boxes for malicious objects such as bolt cutters and angle grinders on top of a bike is greater than a threshold value.

Testing Workflow

Happy Workflow Path 1:
Detection of bolt cutters

- Activate robot for patrolling.
- Open the main page of the application.
- Verify camera feed is being displayed on the main page.
- Verify all options on Model Options page are checked
- Navigate robot to point camera in direction of bolt cutters.
- Verify the correct bounding box is surrounding the bolt cutters on the camera feed.

Happy Workflow Path 2:
View history of objects detected

- Log into the dashboard.
- Verify that there is a button that will show the recent objects detected on the homepage.
- Click and download a CSV file.
- Verify that the file is downloaded correctly.
- Verify that there are timestamps of recent objects detected.

A successful text history of objects detected will consist of a download CSV file and the file containing objects detected along with timestamps.

Happy Workflow Path 3:
Detection of potential threat

- Start the robot for patrol.
- Go to the camera view and dashboard.
- Verify that the camera stream is on.
- Control the robot towards a possible threat.
- Verify that the logs reflect the detection of suspicious movement and overlapping bounding boxes of malicious items and bikes.

Happy Workflow Path 4:
Move Robot with user input

- Power up the Robot.
- Power up the Raspberry Pi.
- Login to the website.
- Access the Manual Control Page.
- Use controls.
- Verify the Robot’s movements correctly reflect the inputs.

Happy Workflow Path 5:
Get GPS Location of robot

- Begin robot patrol.
- Open the main page of the application.
- View the location map.
- View the robot's surroundings from the camera.
- Verify that the robot's estimated location lines up with the actual location.

Unhappy Testing Path Workflow 1:
Attempting to view the text history of objects detected does not prompt the user to download a CSV file.

- Click to download CSV file.
- There is no download option after an initial click.
- Alert the user about an unsuccessful download and to try again later (Trying again later will ensure that a json response will eventually happen since at each click, the web service attempts to retrieve the json object from the backend).

Unhappy Testing Path Workflow 2:
User attempts to login to the web application with the incorrect username and/or password or with no username and/or password.

- User navigates to login page
- User attempts to login without filling in username and/or password fields
- "Fill out this field" is displayed next to empty field
- User types in incorrect password and/or username and attempts to login
- "Please enter a correct username and password. Note that both fields may be case-sensitive." is displayed above.

Unhappy Workflow Path 3:
User attempts to run the threat detection neural network on the video stream, and it fails to output threats.

- User starts the robot for patrol.
- User goes to the dashboard and camera view.
- Use navigates towards potential suspicious activity.
- User doesn't see any activity detected in the logs.
- System won't crash if the model doesn't load properly, but the stream will need to be restarted.
- If the model is loaded, it is possible that the sensitivity of the confidence level in the model needs to be edited.

Unhappy Workflow Path 4:
User attempts to move the Robot, but the Robot is unresponsive.

- User starts the robot for patrol.
- User logs into the website.
- Use goes to the Manual Control Page.
- User inputs movements.
- Robot is not moving.
- If the Robot has battery power, the Raspberry Pi microcontroller needs to be restarted to communicate with the webpage (display this error to the page with guidance on fixing issue).
- If the Raspberry Pi is not connected to the Internet, it must find a network to connect to (prompt the user to ensure the Pi is connected to the Internet).
- If the Robot is out of battery, it must be recharged (let the user know that if communication with the Pi is occurring and the Robot is unresponsive, the Robot needs to be charged).

Unhappy Workflow Path 5:
User attempts to view the predicted location, but the robot location is unknown.

- User starts the robot for patrol.
- User logs into the website.
- User goes to the location map.
- User sees that the location map is blank.
- If the robot is powered on, then the robot needs to be moved around or left on long enough to gather enough location data
- If the robot is not powered on, then the user must turn the robot on.

Testing Strategy

Testing Type:
The testing will utilize acceptance testing to ensure the project is fully functional in all aspects. The testing will be performed by team members. Each team member will test aspects of the project they did not make significant contributions to. This approach is to ensure that bias does not influence testing results. The tests highlighted throughout this document will act as our acceptance test bed as the project progresses. This approach is chosen due to the difficulties in utilizing user testing as providing access to our only robot to a variety of users presents extreme challenges.

Testing Responsibilities:
There are three main sections that will need to be tested to ensure complete functionality of the PatrolBot. These include: Robot, Web Server/UI/UX, and Machine Learning Models. To ensure main developers of these sections are not the people testing them, Brandon and Jesus will conduct Robot tests, Max and Connor will conduct Web Server/UI/UX tests, and Michael will conduct Model tests. Robot functionality can be tested somewhat individually which includes camera function and movement, but movement from Web Server controls and Web Server connection to the camera will need to wait to be tested until the connection from Robot to Web Server is complete. Machine Learning Models can also be tested mostly individually which includes basic model function and model accuracy and precision, but model output from the robot camera feed input will need to wait to be tested until the connection from Robot to Web Server is complete. All Web Server/UI base functionality can be tested on its own before models have been added and connection to Robot has been completed. However, the Model Options page, controls for the robot, and camera feed must wait to be tested until all sections have been connected.

Dealing with Defects:
In the case that defects are found in a particular subsystem, the testers of the subsystem will need to speak with the implementers of the subsystem. The testers will need to provide an explanation for how to replicate the defect and a suggestion for its improvement. Then the implementer will need to address the defect within a week. The same tester will then try again and ensure that the update is up to their standards. This process will be repeated as many times as necessary for all subsystems.

Testing Plan

Test No.	Test Type	Target File or Screen	Test Name	Purpose of Test	Test Data Simulation	Expected Result
1	Acceptance Test	Dashboard/ Detection.py	User Story 5.1	Ensure that aggressive behavior can be detected	Video of somebody fighting	Logs output suspicious activity detected
2	Acceptance Test	Dashboard/ Detection.py	User Story 5.2	Ensure that malicious objects near bikes can be detected	Video of somebody get near a bike with bolt cutters and angle grinders	Logs output malicious object near bike
3	Acceptance Test	Dashboard/ Views.py/Detection.py	Unhappy Workflow 3	Ensure the system can fail gracefully	Comment out the loading of the action detection model	Actions won't be detected, but the system won't crash
4	Acceptance Test	Dashboard/ camera.py	User Story 2.1	Ensure that Camera hardware is communicating with the webpage properly	Raspberry Pi and Website user are on different networks.	Camera Feed is visible on the dashboard.
5	Acceptance Test	Manual Controls Page	User Story 2.2	Ensure that webpage control inputs are successfully sent to the Robot.	Raspberry Pi microcontroller and Website user are on different networks.	Robot movement reflects user inputs

6	Acceptance Test	Manual Controls Page	Unhappy Workflow 4	Ensure guidance is provided to the user if a hardware malfunction occurs	Disconnect Raspberry Pi microcontroller.	Prompts user to check hardware connections.
7	Acceptance Test	Dashboard Overview Page	User Story 3.1	Ensure that the backend is correctly recording a text history of objects detected.	Correct text history in CSV file.	At each object detection, there is a timestamp along with the actual object listed.
8	Acceptance Test	Dashboard Overview Page	Unhappy Workflow 1	Ensure the user is able to retrieve a history of text logs	Frequent json requests to the backend.	Correct prompt of CSV download.
9	Acceptance Test	Robot location map	User Story 1.1	Ensure the user can view the robot location on a UNR campus map	Uses current GPS location whether on campus or not	Displays true GPS location using human judgment to decide on proximity
10	Acceptance Test	Robot location map	Unhappy Workflow 5	Ensure that if the robot location is not available then the user knows what to do	Leave robot off and let website check for data	Notifies user of location unknown error

11	Acceptance Test	Dashboard/ Detection.py	User Story 4.1	Ensure that objects of interest can be detected	Pictures with objects of interest included	Displays bounding box around objects accurately
12	Acceptance Test	Dashboard/ Detection.py	User Story 4.2	Ensure overlay of bounding boxes can be turned on and off	Pictures with objects of interest included	Does not display bounding boxes on objects that are unselected
13	Acceptance Test	Dashboard/ Detection.py	User Story 4.3	Ensure model can be turned on and off	Pictures with objects of interest included	Does not use camera stream to run model
14	Acceptance Test	Login Page	Unhappy Workflow 2	Ensure proper error messages are displayed when user incorrectly logs in	Missing log in data and/or incorrect log in data	Correct error messages are displayed

Contributions of Team Members

Connor

Tasks: User Stories and Acceptance Criteria for Robot Hardware, Happy Workflow 4, Unhappy Workflow 4, Testing Strategy: Testing Types
Time: 1 hour

Brandon

Tasks: User Stories and Acceptance Criteria for Threat Model, Threat Model Happy Workflow, Threat Model Unhappy Workflow, Defects in Testing Strategy, Threat Model Test Plan
Time: 1 hour

Max

Tasks: User Stories and Acceptance Criteria for Detection Model, Detection of bolt cutters Path, Testing Responsibilities, Detection Model Test Plan, Formatting
Time: 1 hour

Jesus

Tasks: User stories and acceptance criteria for Web Server & UI/UX, Happy Workflow 2, Unhappy Workflow 1, Web Server Test Plan
Time: 1 hour

Michael

Tasks: Cover page, abstract, project updates and changes, User stories and acceptance criteria for Robot Software, Happy Workflow 5, Unhappy Workflow 5, Robot software testing plan
Time: 2 hours