Due: 4/9

Release Plan Template – CSE 115a – Software Engineering

The team needs to capture the result of their release planning meeting in a document with the structure outlined below. This document (as well as other project documents should be made accessible to the team's TA (e.g., via shared Google docs or drive). There is no further submission process. All project documents are created for the benefit of your project and team, not the entertainment of the teaching staff.
You may in addition use web-based agile tools if you so choose.

- **Heading**: Document name ("Release Plan"), product name, team name, release name, release date, revision number & revision date.
- **High level goals**: A description of the top-level goals for the release. Examples include, for a game: "Be able to play one complete level (but with limitations xx, yy, & zz),""Have all controller capabilities implemented," "Be able to create levels using a level design tool;" or for the Osric system: "Be able to handle service requests for new and existing customers with access to requests by managers and technicians." These high- level goals may map to a single user story, but more typically will map to multiple user stories.
- User stories defining the scope of the release: A listing of all the user stories that are needed to implement the high-level goals. Each user story must have a level of effort estimate in story points. Each user story must be sized to fit within a single sprint. Each user story must be assigned to one of the sprints within the development period (usually 4 two-week sprints in a quarter-length course; 3 one-week sprints in a five-week summer course).
    - Either list the user stories in priority order within each sprint or indicate the priority of each user story explicitly.
    - Recall that a user story should take the form, "As a {user role}, I want {goal} [so that {reason}]". User stories should meet the "INVEST" criteria (independent, negotiable, valuable, estimatable, sized appropriately, and testable).
    - It is a good idea to identify each user story by a unique label that allows the user story to be referenced across different tools and documents.

- **The complete list of user stories will take the form of**:
    - Sprint 1
        - {priority} User story 1.1 [story points]
        - {priority} User story 1.2 [story points]
        - …
        - {priority} Unser story 1.N1 [story points]
    - and so on for subsequent sprints.
    - Note: "User story x.y" is a "meta name"; use some more descriptive label instead.

- **Sanity check your release plan.**

    - Is the plan within the team's capacity? Given what you know about your team's capabilities at this point, is the the total amount of work doable (add up the story points for all user stories and compare with the team's capacity).
    - Is the work distribution across sprints reasonable? Did you allow for time spent on infrastructure tasks and spikes? Holidays? Midterms?

- **Product backlog**: A listing of all high-level goals and user stories that were discussed in the release planning meeting, but which did not make it into the release at this point. User story priorities may change in the course of the project and therefore the PO may decide to downgrade some user stories currently in the release plan and promote some user stories currently in the backlog. The release plan and product backlog should be revisited and updated after each sprint.
  The product backlog remaining at the end of the last sprint can serve as the starting point for a subsequent release.

Initial Presentation: The release plan will be the basis for your team's initial presentation.

Product name: Synthura Security

Team name: Synthura Security

Release name: Synthura

Release date (Sprint 1 ending): 23rd

Revision number & date: 0 : 4/6/2024

High level goals:

A home, corporate, and government security system product where you can create environments for clusters of cameras and track detailed analytics of live video feeds and have databases of labeled video footage for easy travers-ability however needed.

Specifically relating to analytics: We want to be able to track all current objects within our video video feed, all activities, and scene understanding.

Example headers with details:

Current objects:

Human 1, human 2, hat, phone, knife

Activities:

Motion

Scene understanding:

Potentially dangerous circumstance

User case example:

Corporation has several stores within the State of California, so they set up a security environment with Synthura. Within their environment they have feeds labeled for every store they have within California and then within those feeds they define clusters of cameras which keep track of the above analytics. They do this for universal and scalable highly detailed active security and at all times they know exactly what's going on with any store.

List of current user stories ((priority number) user story [story points]):

(1) As a user, I want to be able to log in to a website using my Google account, connect my camera, and receive detailed object labeling and tracking, because I need precise and automated identification of objects within my video feed for enhanced monitoring and security. [4]

(2) As a user, I want an information feed where I can view current items and detect motion, because I need to be immediately aware of any changes or movements within the camera's view for security and monitoring purposes. [2]

(3) As a user, I want the ability to connect multiple cameras to a centralized security cluster with features to add or delete cameras, and connect or disconnect them as needed, because I need a flexible and scalable security setup that can adapt to varying surveillance requirements. [5]

(4) As a user, I want to be able to create and manage a security environment with options to create and delete environments and clusters, because I need a structured way to organize and access multiple feeds for efficient monitoring across different locations. [5]

(5) As a user, I want more detailed analytics for scene understanding from my cameras, specifically tracking potentially dangerous scenarios, because I need advanced insight into the footage for preemptive security measures and situational awareness. [4]

(6) As a user, I want email alerts for potentially dangerous scenarios detailing the specific environment, cluster, camera, time, and triggering event (e.g., a knife), because I need immediate and actionable information to respond to security threats effectively. [3]

(7) As a user, I want additional detailed security analytics and thought-out research, because I need deeper insights and data-driven decisions to enhance security measures and strategic planning. [3]

(8) As a user, I want camera feed video storage features where I can record, pause, stop, and extract footage, with each environment organized as a directory containing files associated with camera clusters, because I need comprehensive and organized access to recorded video for review and evidence collection. [6]

(9) As a user, I want the ability to track and search footage details via a feed based on tags or specific information, specifically to locate footage involving potentially dangerous scenarios and navigate to relevant points easily, because I need efficient ways to review and analyze specific events captured by my security system. [6]

Features:

(1) I want a website where I can login via google, connect my camera, and have detailed object labeling and tracking. [4]

(2) I want an info feed where I can see present items and detect motion. [2]

(3) I want to be able to connect multiple cameras for my easily accessible security cluster, using the feature of add / delete camera, and then connect / disconnect camera. [5]

(4) I want to be able to create a security environment (Buttons to create and delete environments and then buttons to add and delete clusters) where I can access multiple feeds. [5]

(5) I want more detailed analytics relating to scene understanding of the scenes within my cameras. Basically I want a more detailed info feed. Specifically relating to tracking “potentially dangerous scenarios”. [4]

(6) I want email alerts for potentially dangerous scenarios detailing which environment, cluster, and which camera it was in. Registering what time it happened and what set it off, example: a knife. [3]

(7) I want additional detailed security analytics: Other thought out analytics / research. [~3]

(8) I want camera feed video storage features where you can click record, pause, stop, and extract. Most are self explanatory but upon extract: Each environment is a directory and each directory has files associated with camera clusters, inside the camera cluster folders we have all of the footage. (WARNING: High degree of uncertainty regarding success and implementation.) [6]

(9) I want info tracked on details relating to the footage of each video. A form of feed where users can traverse videos based upon tags or info. Example being I want to find the footage where we had possible dangerous circumstances and I want to be able to find the point in the footage easily. (WARNING: High degree of uncertainty regarding success and implementation.) [6]

/ 38

Product backlog / Stretch features:

- Review after release planning meeting. (Where are unable to achieve features will go.)

Previous possible user stories:

- As someone who is monitoring a security feed, I want real-time video feed capture and processing to help me monitor my feed and be aware of threats in real time.
- I want to view all of my video feeds on a single page
- I want basic object detection features of my feed to identify potential threats
- I want important events to be time stamped by the software so that I can go back and look at it
- I want the software to understand the scene, to notify me when the scene is dangerous
- I want the feed videos to be saved so that I can refer to them in the future
- Database integration
- I want to easily view my feed and simply be made aware of anything the software detects
- I want to be alerted when there are any interesting events so that I can respond in a timely manner

Sanity check:

Given the capabilities of my team which span deep learning, computer vision, front-end, back-end, and databases. Sprint plans 1, 2, and 3 are very doable with enough training, preparation, and dedication. However currently sprint 4 seems to be the most ambitious and it’s likely we’ll fail or pivot so the tasks will probably end up going to the product backlog.

Initial presentation: The release plan will be the basis for your team's initial presentation.

Sprint Plan Overview:

Sprint 1: Setup and Basic Functionality

- **Duration:** 2 weeks
- **Main Objectives: (1 + 2)**
    - Establish the basic web framework for connecting cameras and displaying feeds.
    - Begin development on basic object and motion detection.

Sprint 2: Enhancing Functionality

- **Duration:** 2 weeks
- **Main Objectives: (3 + 4)**
    - Implement multi-camera connectivity and management.
    - Expand on the basic analytics to include more detailed scene understanding.

Sprint 3: Advanced Features and Analytics

- **Duration:** 2 weeks
- **Main Objectives: (5 + 6 + 7)**
    - Develop complex scene analytics and dangerous scenario detection.
    - Set up notification and alert systems for identified scenarios.

Sprint 4: Storage and Accessibility

- **Duration:** 2 weeks
- **Main Objectives: (8 + 9)**
    - Implement video storage solutions with detailed tagging and searching capabilities.
    - Review and refine all features, focusing on database storage and user accessibility.

Divided up work per sprint:

Sprint 1 - Scrum Leader: Githika

- **Levi**
    - Focus on setting up machine learning models for object and motion detection.
    - Implementing computer vision capabilities.
    - Helping facilitate connection of models to the web page.

- **Owen**
    - Strongly support web development tasks for getting the foundational web application functioning.
    - Specifically work with Githika on adding and deleting environments and clusters functionality.
- **Githika**
    - Lead the frontend development for the web interface.
    - Integrate google login and logout feature.
- **Nam**
    - Support frontend development, specifically making sure the recordings page is working.
    - Look into setting up the database for later on.
- **Jerry**
    - Work on initial setup for object detection alongside Levi.
    - Learn and assist in backend processes connecting to Owen.
- **Zachary**
    - Assisting in the ML/CV algorithm connection to the web application.

Additional Notes:

- The team's main focus is the web page and camera connections. When tasks are finished, focus on gravitating towards that focus via collaborating with others.

Sprint 2 Scrum Leader: Owen

- **Levi**
    - Enhance the ML/CV algorithms for better object and scene understanding.
- **Owen**
    - lead facilitating multi-camera connectivity features / working with model connections.
    - Implement camera management functions in the web interface.
- **Githika**
    - Improve the UI/UX for managing multiple camera feeds and analytics.
- **Nam**
    - Optimize database schema for handling more complex data from multiple cameras.
    - Support UI/UX and web design.
- **Jerry**
    - Assisting in enhancing the ML/CV algorithms for better object and scene understanding.
- **Zachary**
    - Assisting in enhancing the ML/CV algorithms for scene understanding.

Additional Notes:

- The team's main focus is the web page and camera connections. When tasks are finished, focus on gravitating towards that focus via collaborating with others.

Sprint 3 Scrum Leader: Jerry

- **Levi**
  - Assisting Jerry in algorithms for detecting potentially dangerous scenarios and other analytics.
- **Owen**
  - Set up the email alert system for notifications.
- **Githika**
  - Refine the user interface for analytics and alerts.
- **Nam**
  - Ensure the backend supports the new analytics and notification systems.
- **Jerry**
  - Leading analytics systems and developing algorithms for detecting potentially dangerous scenarios and various other analytics.
- **Zachary**
  - Leading analytics systems and developing algorithms for enhanced videos statuses and accuracy.

Additional Notes:

- The team's main focus is on implementing multi-camera connectivity and management with the web page, more detailed analytics feed, and getting email notifications working from the expanded analytics (scene understanding = possible danger alters).

Sprint 4 Scrum Leader: Nam

- **Levi**
  - Finalizing outstanding ML/CV features. Jumping to any holes we have within the development process.
- **Owen**
  - Final touches on functionality and bug fixes.
- **Githika**
  - Working on final user testing, ease of UI/UX.
- **Nam**
  - Lead the Implementation and refine the video storage and extraction features.
- **Jerry**
  - Finalizing outstanding ML/CV features. Jumping to any holes we have within the development process.
- **Zachary**

    - Finalizing outstanding ML/CV features. Jumping to any holes we have within the development process.

Additional Notes:

- The team's main focus is on database implementation. Functionality touch ups and computer vision models shouldn't take long, then people will gravitate towards helping Nam Tran develop the database system.

Scrum Leaders

- **Sprint 1:** Githika
- **Sprint 2:** Owen
- **Sprint 3:** Jerry + Zachary
- **Sprint 4:** Nam

Technology Overview:

1. Hardware

Cameras: We will evaluate cameras based on resolution, frame rate, and other performance metrics. Initially, we plan to use laptop cameras for testing and simulation purposes.

2. Software

Programming Language: Python is selected for its extensive support for AI and machine learning libraries.

3. Libraries for Computer Vision and Deep Learning

OpenCV: Utilized for basic image processing and computer vision tasks.

PyTorch: Applied in deep learning models and algorithms.

Scikit-learn: Employed for simpler machine learning tasks and data processing.

4. Web Development Stack

Frontend: Utilizes HTML, CSS, and JavaScript, with frameworks such as React or Angular to develop the user interface.

Backend: Flask or Django, which are Python-based frameworks, will handle server-side logic, API endpoints, and integration with AI models.

Database (Future Implementation): Plans to use PostgreSQL or MongoDB to store video metadata, event timestamps, and analytics results.

5. Object Detection and Motion Detection

Pre-trained Models: We will use state-of-the-art object detection models such as YOLOv8, Detectron2, Detr, etc.

Motion Detection: Algorithms will be implemented to detect changes in video frames to identify motion, using tools like OpenCV.

6. Scene Understanding

This aspect involves expanding functionality and connectivity with various infrastructural elements, which is more complex.

7. Alert System

A system will be developed to send alerts based on specific criteria, possibly integrating with web technologies and messaging services like Twilio or email platforms for notifications.

Additional Details:

Scalability and Accessibility: The system will be designed for security and scalability, enabling connections to multiple cameras per environment. This includes a video database with environment-specific files and camera data.

Experimentation and Iteration: Ongoing experimentation with different models and algorithms will be crucial for refining the analytics capabilities of the system.