Veteran e-Post Hub

Prototype Project Report

Whitman County Veteran Department

A logo of a county

Description automatically generated

A black and white logo

Description automatically generated

**Hero’s Hub**

Melvin Manning

Anthony Graves

[09.19.24]

**TABLE OF CONTENTS**

1. **Introduction.........................................................................................................3**
2. **Requirements Specification...............................................................................4**

**II.1. Use Cases.....................................................................................................4**

**II.2. Functional Requirements..........................................................................14**

**II.3. Non-Functional Requirements..................................................................18**

**III. System Evolution..............................................................................................19**

**IV. Glossary............................................................................................................20**

**V. References.........................................................................................................21**

# I. Introduction

Veterans’ Services Officers have been assisting the veteran community since World War I, helping them navigate the complex tasks of filing claims and communicating with the Department of Veterans affairs (VA). Traditionally, these services relied on mail correspondence and in-person meetings. However, in the information age, the responsibilities of VSOs have evolved; they are now tasked with not only with claims but also keeping veterans informed about beneficial events and opportunities. This shift in responsibilities has made apparent the need for a more efficient and user-friendly means of communication with veterans.  
  
To address this, we are developing Veteran e-Post Hub, a website designed to bridge the gap between the needs of veterans and the current methods of sharing events and opportunities. Veteran e-Post Hub will enhance the tools employed by our local VSO, including her monthly e-Post, and provide a user-friendly and intuitive platform for our VSO and other interested parties to submit, share, and provide feedback on events. Veteran e-Post Hub will ensure that our VSO has the tools she needs to make vital information accessible to all veterans and their dependents.

# II. Requirements Specification

## II.1. Use Cases

**Use Case Diagram:** [**Use Case Diagram**](https://i.imgur.com/jlX2m6L.jpeg)A diagram of a diagram

Description automatically generated

**User Login**

|  |  |
| --- | --- |
| **Pre-condition** | * User is not currently logged in |
| **Post-condition** | * User is logged in |
| **Basic Path** | * Locate “Login” button * The user selects the “Login” button * User is redirected to the login page * User enters their correct credentials and presses the “Login” button * User is redirected to the homepage and has been successfully logged in |
| **Alternative Path** | * In step 4, if the user enters the wrong credentials, the page reloads, and a message "Invalid login credentials" is displayed. |
| **Related Requirements** | * The system must provide a mechanism for users to create client and admin accounts. * The system should allow users to reset their password if they forget it. * The system must ensure secure handling of user credentials during login. |

**Create member account**

|  |  |
| --- | --- |
| **Pre-condition** | * The user’s account does not exist |
| **Post-condition** | * The user’s account exists |
| **Basic Path** | 1. Locate the “create member account” button 2. The user selects the “create member account” button 3. User is redirected to the create member page 4. User enters name, email, password, confirm password, and phone number(optional) 5. User submits information for account 6. User is redirected to the homepage, logged in to the new account, and is given a message for successful account creation |
| **Alternative Path** | * In step 4, if the user’s email is not valid, the page is reloaded and a message saying “invalid email format” is shown * In step 4, if the user’s password and confirm password does not match, the page is reloaded and a message saying “passwords do not match” is shown * In step 4, if the user does not enter required information, the page is reloaded and a message saying what is missing is shown. |
| **Related Requirements** | * The system must validate email format and provide appropriate feedback. * The system must ensure password confirmation and provide feedback if they do not match. * The system must check that all required fields are filled in before allowing submission. * The system should automatically log the user in after successful account creation. |

**Create admin account**

|  |  |
| --- | --- |
| **Pre-condition** | * The user’s account is not created |
| **Post-condition** | * The user’s account exist |
| **Basic Path** | 1. Locate the “create admin account” button 2. The user selects the “create admin account” button 3. User is redirected to the create admin page 4. User enters name, email, password, confirm password, and phone number(optional), work hours(optional), work address(optional), work phone number(optional), and current creator’s code 5. User submits information for account creation 6. User is redirected to the homepage, logged in to the new account, and is given a message for successful account creation |
| **Alternative Path** | * In step 4, if the user’s email is not valid, the page is reloaded and a message saying “invalid email format” is shown * In step 4, if the user’s password and confirm password does not match, the page is reloaded and a message saying “passwords do not match” is shown * In step 4, if the user does not enter required information, the page is reloaded and a message saying what is missing is shown. * In step 4, if the user does not enter the correct creator’s code, the page is reloaded and a message saying “incorrect creator’s code” is shown |
| **Related Requirements** | * The system must validate email format and provide appropriate feedback. * The system must check for password confirmation and provide feedback if they do not match. * The system must validate that all required fields are filled in before submission. * The system must verify the creator’s code before allowing account creation. * The system should ensure that the new admin account is logged in automatically upon creation. |

**User logout**

|  |  |
| --- | --- |
| **Pre-condition** | * User is logged in |
| **Post-condition** | * User is not logged in |
| **Basic Path** | 1. Locate the “Logout” button 2. The user selects the “Logout” button 3. The homepage is reloaded, and the user is no longer logged in |
| **Alternative Path** | * N/A |
| **Related Requirements** | * The system must ensure that the user’s session is properly terminated upon logout. * The system should provide feedback confirming that the user has been logged out (e.g., a message or redirect to the homepage). * The system must prevent access to logged-in features after logout. |

**Create Event**

|  |  |
| --- | --- |
| **Pre-condition** | * Event Post does not exist * User is logged in |
| **Post-condition** | * A new event post is created and exists in the system |
| **Basic Path** | 1. Locate the “Create Post” button 2. The user selects the “Create Post” button 3. User is redirected to the create post page 4. User enters event name, date, description(optional), website link(optional) contact information, and flyer of pdf or image format. 5. User submits event 6. The system confirms that the event has been created and displays a success message |
| **Alternative Path** | * In step 1, if the user is not logged in then the create post button will not be visible * In step 5, if the user is an admin then the post will be approved automatically |
| **Related Requirements** | * The system must authenticate the user before displaying the “Create Post” button. * The system must validate the entered information (e.g., date format, required fields). * The system should ensure that uploaded files meet specified formats and size limits. * The system must notify the user of successful creation and provide an option to view the new event post. |

**Comment on post**

|  |  |
| --- | --- |
| **Pre-condition** | * The user is logged in |
| **Post-condition** | * A comment will be added to an event post |
| **Basic Path** | 1. Locate the event name 2. The user selects the event name 3. The user will be directed to the post’s page 4. Locate the comment box 5. User enters comment and submits it 6. Comment will be added to the event for all users to see |
| **Alternative Path** | * In step 4, if the user is not logged in then the comment box is not visible |
| **Related Requirements** | * The system must authenticate the user before allowing comments. * The system must ensure comments are displayed in chronological order. * The system may have moderation features to manage inappropriate comments. |

**Interested button**

|  |  |
| --- | --- |
| **Pre-condition** | * The user is viewing a post |
| **Post-condition** | * The “interested” counter will increase by one |
| **Basic Path** | 1. Locate the event to comment on 2. The user enters the event’s page 3. Locate the “interested” button 4. User clicks the “interested” button 5. The interested counter is increased by one |
| **Alternative Path** | * N/A |
| **Related Requirements** | * The system must ensure the user can only express interest once per event. * The system must visually update the interested counter immediately after the button is clicked. |

**Print Event document**

|  |  |
| --- | --- |
| **Pre-condition** | * User has access to the event data (no login required) |
| **Post-condition** | * A printable page of events is generated and displayed. |
| **Basic Path** | 1. Locate the “Print All Events” button on the interface 2. Click the “Print All Events” button. 3. The system redirects to a printable page with all events |
| **Alternative Path** | * In step 1, if the user wants to print all filtered events, they can filter them first and then proceed to step 2 |
| **Related Requirements** | * The system must allow users to filter events before printing. * The printable page must be formatted appropriately for printing (e.g., clear layout, no navigation elements). * The system should provide options for users to print or save the document as a PDF. |

**Filter events**

|  |  |
| --- | --- |
| **Pre-condition** | * User has access to the event data (no login required) |
| **Post-condition** | * Events displayed are filtered based on user selection |
| **Basic Path** | 1. User locates the filter menu. 2. User selects the filters that apply to their search criteria. 3. User presses the "Search" button. 4. Only events matching the selected criteria are displayed. |
| **Alternative Path** | * N/A |
| **Related Requirements** | * The system must handle cases where no events match the selected criteria, displaying an appropriate message. * The system should allow users to reset filters to view all events. |

**Admin Approve/Reject Events**

|  |  |
| --- | --- |
| **Pre-condition** | * The user is logged in as an admin. * There are events awaiting approval or rejection |
| **Post-condition** | * Events are either approved or rejected based on the admin’s selection, and relevant notifications are sent to event organizers. |
| **Basic Path** | 1. The user navigates to the page displaying all events awaiting approval or rejection. 2. The user reviews the listed events. 3. The user selects the "Approve" or "Reject" button for each event as needed.    * If the event is rejected, the admin is prompted to enter a reason for the rejection. 4. The admin clicks the "Save" button to confirm the decisions. 5. The system updates the status of the events accordingly. 6. For approved events, the system automatically sends a confirmation email to the event organizer. 7. For rejected events, the system sends an email to the event organizer with the rejection reason. |
| **Alternative Path** | * In step 3, it is possible to no check Approve or Reject so that a decision can be made later. |
| **Related Requirements** | * The system must authenticate the user as an admin before allowing access to the approval page. * The system must send automatic email notifications for approved events. * The system must send rejection emails that include the reason for rejection for any rejected events. |

**Edit Profile**

|  |  |
| --- | --- |
| **Pre-condition** | * Profile is not up to date * User is logged in |
| **Post-condition** | * create. |
| **Basic Path** | 1. User navigates to the "Edit Profile" page. 2. User updates their name, email, password, or phone number. 3. User clicks the "Save" button to apply the changes. 4. The page reloads, and a confirmation message appears indicating that the changes have been saved. |
| **Alternative Path** | * In step 2, if the user is an admin, additional fields such as work hours, work address, and work phone number are available for updating |
| **Related Requirements** | * The system must authenticate the user before allowing profile edits. * The system must validate the entered information (e.g., email format, password strength). * The system must display a confirmation message upon successful updates. * The system must ensure that changes are saved correctly in the database. |

**Modify Events**

|  |  |
| --- | --- |
| **Pre-condition** | * Event information is not up to date * User is logged in as an admin or the user is modifying their own event post |
| **Post-condition** | * Event is updated with new information |
| **Basic Path** | 1. User navigates to the event page. 2. User clicks the “Edit Event” button, which redirects to the edit event page. 3. User enters the new name, date, description, website link, contact information, or uploads a new flyer. 4. User submits the updates. 5. The page reloads, displaying a message confirming that the changes have been saved. |
| **Alternative Path** | * If the user cancels the update at any point, no changes are made, and the user is returned to the event page without any modifications. |
| **Related Requirements** | * The system must authenticate the user as an admin. * The system must validate the entered information before saving. * The system must confirm that the changes have been successfully saved and displayed to the user. |

**Delete Event**

|  |  |
| --- | --- |
| **Pre-condition** | * The event exists in the system * The user is logged in as an admin or the user is deleting their own event |
| **Post-condition** | * The event is successfully deleted |
| **Basic Path** | 1. The user navigates to the event page 2. User clicks the “Delete Event” button 3. User confirms deletion 4. The event is removed from the system and no longer exists |
| **Alternative Path** | * In step 3, if the user does not confirm deletion, it will be cancelled |
| **Related Requirements** | * The system must authenticate the user as an admin. * The system must provide a confirmation prompt before deletion. * The system must ensure that the event is removed from all relevant listings and databases. |

**Share Event**

|  |  |
| --- | --- |
| **Pre-condition** | * User has access to the event data (no login required) |
| **Post-condition** | * A link to the event’s internal or external page is successfully copied |
| **Basic Path** | 1. The user navigates to the event page. 2. The user clicks the "Share" button. 3. The user copies the internal link for sharing. |
| **Alternative Path** | * In step 3, the user has the option to copy the external link |
| **Related Requirements** | * The system must provide both internal and external links for each event. * The interface must include a "Copy" button for easy sharing. |

**Modify Creator Code**

|  |  |
| --- | --- |
| **Pre-condition** | * The creator code is outdated or needs to be changed. * The user is logged in as an admin |
| **Post-condition** | * Creator code is successfully updated |
| **Basic Path** | 1. The user navigates to the "Creator Code" page. 2. The user changes the code to a desired one that is at least 6 characters long and case-sensitive. 3. The user submits the changes. 4. The page reloads, and a message is displayed confirming that the code has been updated. |
| **Alternative Path** | * N/A |
| **Related Requirements** | * The system must ensure that only admin users can access and modify the creator code. * The new creator code must meet a minimum of 6 characters and be case-sensitive. * The system should validate the new code before submission. |

**Delete Comment**

|  |  |
| --- | --- |
| **Pre-condition** | * A comment exists on the event page. * The user is logged in as an admin or the user is deleting their own comment |
| **Post-condition** | * Comment has been successfully removed from the system |
| **Basic Path** | 1. The user navigates to the event page. 2. The user finds the comment they wish to delete. 3. The user clicks the "Delete Comment" button. 4. The system prompts the user to confirm the deletion. 5. The user confirms the deletion. 6. The comment is now removed from the system. |
| **Alternative Path** | * N/A |
| **Related Requirements** | * The system must allow only admin users to delete comments. * The system should prompt for confirmation before deleting a comment. * The deletion should be irreversible, and the comment must be removed from all views. |

## II.2. Functional Requirements

**USER INTERFACE:**

**Registration**

|  |  |
| --- | --- |
| **Description:** | The system must allow visitors and interested parties to register as a member or admin. |
| **Source:** | Internal team decision. |
| **Priority:** | Priority level 0: Essential and functional requirement. |

**Login**

|  |  |
| --- | --- |
| **Description:** | The system must require users to login to access a user interface or dedicated admin interface. This will be a requirement to access basic features like posting, commenting or submitting events. |
| **Source:** | Client requirement |
| **Priority:** | Priority level 0: Essential and required functionality. |

**Profile Modification**

|  |  |
| --- | --- |
| **Description:** | The system must allow users to modify and edit their profiles, including but not limited to, personal details, password, email address and event history. |
| **Source:** | Internal team decision. |
| **Priority:** | Priority level 0: Essential and required functionality. |

**User Event Submission**

|  |  |
| --- | --- |
| **Description:** | The system must allow users who have logged in to submit events. |
| **Source:** | Client requirement. |
| **Priority:** | Priority level 0: Essential and required functionality. |

**User Commenting**

|  |  |
| --- | --- |
| **Description:** | The system allow members to on events and leave feedback. |
| **Source:** | Internal team decision. |
| **Priority:** | Priority level 0: Essential and required functionality. |

**Comment Modification:**

|  |  |
| --- | --- |
| **Description:** | The system must allow members and admins to edit or delete comments made on postings of events. This functionality will be available to those who have made the comments as well as the page admin. |
| **Source:** | Client requirement. |
| **Priority:** | Priority level 0: Essential and required functionality. |

**Event Approval / Rejection**

|  |  |
| --- | --- |
| **Description:** | The system must allow admins to approve or reject events submitted by members. |
| **Source:** | Internal team decision. |
| **Priority:** | Priority level 0: Essential and required functionality. |

**Event Interest:**

|  |  |
| --- | --- |
| **Description:** | The system must allow users to express interest in an event. |
| **Source:** | Internal team decision. |
| **Priority:** | Priority level 1: Desirable functionality. |

**Event Sharing**

|  |  |
| --- | --- |
| **Description:** | The system must allow users to share events to Facebook or Twitter. |
| **Source:** | Client requirement. |
| **Priority:** | Priority level 0: Essential and required functionality. |

**Event Modification**

|  |  |
| --- | --- |
| **Description:** | The system must allow admins to modify events through a dedicated admin interface. |
| **Source:** | Client requirement. |
| **Priority:** | Priority level 0: Essential and required functionality. |

**Filter Events**

|  |  |
| --- | --- |
| **Description:** | The system must allow users to filter the list of events to suit their own interests. |
| **Source:** | Internal team decision. |
| **Priority:** | Priority level 0: Essential and required functionality. |

**Delete Events**

|  |  |
| --- | --- |
| **Description:** | The system must allow members or admins to delete the events they have created or are responsible for monitoring. |
| **Source:** | Internal team decision, |
| **Priority:** | Priority level 0: Essential and required functionality. |

**INTEGRATION:**  
  
**Data Management**

|  |  |
| --- | --- |
| **Description:** | The system must utilize a database to store, retrieve and manage event data, user information and admin actions. |
| **Source:** | Internal team decision. |
| **Priority:** | Priority level 0: Essential and required functionality. |

**SECURITY:**

**User Authentication:**

|  |  |
| --- | --- |
| **Description:** | The system must securely authenticate users and administrators using a username and password. All passwords must be encrypted and stored securely to prevent unauthorized access. |
| **Source:** | Internal team decision. |
| **Priority:** | Priority level 0: Essential and required functionality |

## II.3. Non-Functional Requirements

**Usability:**

**Accessibility:**

* The system shall follow accessibility standards to ensure that all users, including those with disabilities, can easily navigate and use the website.

**User Interface:**

* The system shall provide a user-friendly, visually appealing and intuitive user interface. To accomplish this, Veteran e-Post Hub will be done in a layout familiar to most people, it will be well organized and have a consistent layout across pages, with descriptive images and icons to enhance navigation of the website.

**Feedback:**

* The system shall provide a way for users to leave feedback on the website, including reporting a problem to the admins. In addition, the system shall include direct links to contact the administrators and the Whitman County Veteran’s Services office for further assistance.

**Reliability:**

**Security:**

* The system shall utilize a password encryption algorithm to ensure that in the event the database is compromised, all user information will remain secure. Additionally, all sensitive data shall be encrypted during transmission using HTTPS.
* The system shall implement secure session management practices, including session authentication and cookies, to protect the integrity of users. Session timeouts will be automated to terminate a session within 15 minutes of user inactivity, ensuring that inactive sessions are closed securely.

**Hosting:**

* The system shall utilize a reliable webhosting service that guarantees uptime of at least 99% with options for scaling to handle increased traffic. The hosting provider must also offer backup services to ensure the website remains up in the event of network failure.

**Performance:**

**Response Time:**

* The system shall respond to user-interaction, such as page loading or button clicks within 2 seconds or less on all I/O devices, including PCs, smartphones, and tablets.

**Storage Requirements:**

* The system shall have adequate storage to handle event submissions, user information and associated media files, with expansion capabilities for future growth.

**Process Requirements:**

**Technology Stack:**

* The system will utilize JavaScript, HTML, and CSS for both front-end and back-end development. The back end will be powered by Node.js, Flask, and SQL, while React will be employed for the front end to ensure optimal performance. We will implement unit testing using Pytest and manage version control through GitHub.

**Development Methodology:**

* The system shall employ the agile development methodology, incorporating sprints and iterative development combined with feedback from the client and other stakeholders. Agile is fast replacing waterfall as the industry standard and preferred method for accomplishing a task within a reasonable timeframe.
* The system shall be built using a version control system. GitHub will be used for easy collaboration between team members and to enhance the existing agile development process.

**Testing:**

* The system shall be subject to test driven development (TDD) and unit testing using Pytest to ensure functionality.

# III. System Evolution

A point to consider when building Veteran e-Post Hub will be the changing nature of technology. The system will be developed using JavaScript, HTML, CSS, SQL, Node.js and React. As these languages and frameworks change, updates will be required to implement new features, enhance existing ones, and optimize the website. Regular user feedback will be used in the process of improving the system to better meet the needs of veterans, administrators, and other interested parties. This strategy will allow the platform to remain useful over time.  
  
Not every problem can be anticipated, and the development team understands that due to the nature of the project, we will almost inevitably encounter roadblocks. To mitigate these, test-driven development (TDD) and unit testing will be used throughout the development process. By testing first, we can identify potential issues early and address them quickly. This will allow the system to remain current as user needs change.

# IV. Glossary

**Veterans’ Services Officers (VSO)** - A government official who assists veterans with filing claims and other interactions with the Department of Veteran’s Affairs (VA).  
  
**Department of Veteran’s Affairs (VA)** - A federal agency responsible for providing vital services to America’s veterans, including healthcare and benefits.  
  
**Veteran e-Post Hub** – A website designed to assist in the communication between the Veterans of Whitman County and the Veteran Services Officer (VSO) responsible for these services, meant to provide tools for sharing events and opportunities.  
  
**e-Post** – A monthly digital newsletter that is sent to veterans containing information about events and opportunities.  
  
**Use-Case** – A description of how users will interact with the system.  
  
**Pre-Condition** - The state of the system before an action takes place.

**Post-Condition** - The state of the system after an action has taken place.  
  
**Basic Path** – The most common steps users follow to accomplish a task in a use-case.  
  
**Creator’s Code** – A code used to validate the identity of the user creating an admin account.

**GitHub** – A web-based platform used for version control, collaboration and project management in software development.

**Agile** – A project management and development methodology that focuses on iterative progress, with frequent feedback from stakeholders.

**Sprint** – A set period during which specific work must be completed.

**Test-Driven Development (TDD)** - An application development process where tests are written before code is implemented to be sure of functionality.

**Unit Testing** – Testing individual components of applications to be sure they work as expected.

**Pytest** – A testing framework in python used for writing and running unit tests.

**Encryption** – The Process of converting information into a code to prevent unauthorized access.

**Session Timeout** – The automatic termination of a session after a period of inactivity.

**HTTPS** – A protocol for secure communication between the client and the server.

**JavaScript, HTML, CSS, SQL, Node.js, React** – Various programming languages and frameworks used for the development of front-end and back-end applications.  
  
**Database** – A structured set of data held on a computer.

**Accessibility Standards** – Guidelines that ensure a website is usable by people, including those with disabilities

**Feedback** – Information from users that helps improve the system’s functionality and user experience.

# V. References:

[no references at this time]