S2021 Project Description

AU Lost And Found

*The problem:* The AU police department currently keeps an excel spreadsheet of items found on campus. This means students must show up to the police department in the hope that their item was recovered rather than search online. It also means that AU Police needs to rely on the accuracy of the spreadsheet for omissions and deletions and cannot get good reports about items. Anyone with access to the spreadsheet can add or delete items and they won’t know who did it.

*The solution:* Create a web application that allows police staff login into the site and update the lost and found data. The AU police need to keep track of the item found, a location found, a detailed description and an overall general description (see “***Who found it and Claimed it”*** below). In addition, they need to log who found the item and their contact data. (see ***Item Categories”*** below). Students can login to see general categories found items (see “***Item Categories”*** below). When items are claimed, AU police need to record data about the person claiming the item (see “***Who found it and Claimed it”*** below). In addition, the AU Police should record the approximate value of the item. Items with a higher value need extra identification before claiming. (see “***Item Details”*** below).

The app has the following basic requirements:

1. ***Campus Data*** - Have a way to properly label each building on campus. It should know how many floors each building has. For example, it should know that Stephens has 2 floors and a basement. When you find something in Stephens, you must indicate where you found it (either the first or second or basement) and room number (if applicable).
2. ***Item Categories*** - Have a way to recognize a valid way to list general categories of items and count of how many. For example, a non-privileged user might be able to see: Keys: 2 sets, gloves: 3 sets, hats: 1, etc. For each set of items, students need to be able to see when they were found and the general area (for example ‘Dunham’ or near Dunham).
3. ***Who found it and who claimed it –*** when an item is found, the application should record their name, AU ID, phone number and/or driver license number and date/time they reported it. The same information needs to be recorded when an item is claimed about the person who claimed the item.
4. ***Item Details*** - Record specific details about an item found that includes, where found (see “***Item Location”***), what category the item is (E.g., See ***“Item Categories”***), when it was found, approximate item value, and a detailed item description. This detailed description, is required and it will provide a way someone to identify the item.
5. ***Item location*** - Have a way to properly log finding an item outside. For example, you might record finding an item ‘near’ Dunham. Provide a comment box for additional details. For example, a user might record “in the parking lot closest to the building”. If the item is found on campus you must record the building, floor and room.
6. ***Items Report Per Month -*** Provide a way for a privileged user to list out all recovered and non-recovered items found in a given month. For example, AU police should be able to list out all items found in January that were recovered by a user, when they were found, when they were claimed and who claimed them.
7. ***Admin user –*** police members should have admin access to the data that allows them to operate the application. Each police officer should have their own ID. Non-admin users (e.g., students or facility) can only pursue the general categories of items found and not yet claimed.
8. ***Non admin users –*** Students should have access to a menu of item categories. For example, clothes, keys, flash drives, etc. When they select a category, a report of the item, its ID (like flash drive 1) and the date recovered should be shown.

AU Lost and Found Tests

You need to build this app and provide a coverage report of the tests. These requirements include

1. REST API – You need to provide a full suite of Postman tests that completely test your REST API. Your final report needs to convinced show that you have fully covered the API in tests. So, this means you provide:
   1. Clear documentation of the REST API and a corresponding set of tests that show the application is tested
2. Java Script front-end – Your font-end need to be written in such a way that you can test it. This means clear use of functions and classes each with tests that fully cover the functionality and edge conditions. You will need to show the details of each method/object and the full coverage tests for them.
3. Node.js back-end – The back-end produces the API by using several paths that impact the data. You will need to produce a set of tests that show the variable routes into your system and full coverage.

Final Project Rubric

|  |  |  |
| --- | --- | --- |
| What | Details | Points |
| Functionality | Clearly shows correct implementation of the 8 major requirements. Demonstrates each feature clearly implemented | 30 |
| Verification Activities | A clear description of activities completed to define coding standards, review each other’s code and verify you are building the product correctly. | 10 |
| API Tests | Clear, Well documented API. A clean set of tests | 20 |
| Java Script Tests | Clear statement of at least 4 major methods/objects. Demonstrate full coverage of tests. | 20 |
| Node.js Test | Clear statement of at least 8 route variants. Demonstrate full coverage of tests. | 20 |
| Total |  | 100 |

Project Prototypes

After Project Approval will complete a requirements statement that includes:

1. ***Introduction*** – Purpose, User Profile and workflows. Here you should describe the overall project and indicate who is using the system and how they will interact with the system.
2. ***Requirements*** – Functional and non-functional requirements. This is a restatement of the requirements with analysis of items you are committing to and clarification items as needed.
3. ***User Stories from the requirements –*** The requirements (above) asks you to think through the provided requirements and rephrase them in your own works. During this step you will create a set of user stores (at least 5 based on the requirements provided).



1. ***GUI Prototype*** – Drawing a UI pictures and descriptions. You will need to draw at least 3 images that describe how the user will interact with the application.

Prototype Rubric

|  |  |  |
| --- | --- | --- |
| What | Details | Points |
| introduction | Clearly describes the purpose, the users and how they will interact with the system. | 10 |
| Requirements | Clear restatement of requirements in the group’s own words. | 20 |
| User Stories | Clear user stories in the formatted provided in course. In particular, the acceptance criteria are testable, atomic, and implementation free. | 35 |
| Prototypes | Prototype pictures are clear, well thought out and provide clear value in clarifying what the UI will look like. There should be little omissions or non-clear areas. | 45 |
| Total |  | 100 |

Release Planning

After Project Approval will complete a requirements statement that includes:

1. ***Introduction*** – Purpose of release planning.
2. ***Release Schedule -*** The schedule of sprints that lead up to the first overall release. Indicate the overall user story per sprint.
3. ***The Definition of Done –*** Indicate the quality DoD for these items:
4. Unit tests passed – What is your coverage goal?
5. Code reviewed - How will you ensure code is reviewed?
6. Code standards passed – what coding standards will you use?
7. Functional/non-functional tests passed – what functional tests will you require?
8. Product Owner accepts the User Story

|  |  |  |
| --- | --- | --- |
| What | Details | Points |
| introduction | Describe the product and overall goals | 10 |
| Release Schedule | Clear set of task with a realistic schedule of completion. | 45 |
| DOD | Clear user stories in the formatted provided in course. In particular, the acceptance criteria are testable, atomic, and implementation free. | 45 |
| Total |  | 100 |

Status Reports – End of Each Sprint

At the end of each sprint, each team member will complete a requirements statement that includes:

1. Project Title: (provide a title for the project).
2. Team Members Names:
3. Overall Status: Pick either: on-schedule, behind, ahead. Describe why you selected that (complete 3-4 complete sentences).
4. Hours: (replace ## with your hours for each item)

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Total Hours worked since last update | Total hours works on project so far | Estimated total hours at completion |
| Team member 1 |  |  |  |
| Team member 2 |  |  |  |

1. Accomplishments

|  |  |  |
| --- | --- | --- |
| Name | Sprint Goal / Team member | Accomplishments towards goal |
| Team Member 1 |  |  |
| Team Member 2 |  |  |

1. Challenges and what needs improving

|  |  |
| --- | --- |
| What needs improving | Description (3-4- complete sentences) |
| 1 |  |
| 2 |  |
| 3 |  |

7. Plans / Goals for next sprint (Include upcoming tasks, milestones, goals,

and deliverables)

|  |  |
| --- | --- |
| What needs improving | Description (3-4- complete sentences) |
| 1 |  |
| 2 |  |
| 3 |  |

8. Questions for the instructor. What question(s) do you have for me? What do you need help on? What do you need clarification on?