Lost and Found Item Tracker

Deliverable for Software Engineering Project by S Vishwajith - 23BCE1145

Problem Statement

The management of lost and found items within large organizations, institutions, and public environments often suffers from inefficiencies caused by manual record-keeping, lack of transparency, and poor coordination. Traditional methods make it difficult to accurately track lost items, verify ownership, and prevent fraudulent claims, leading to confusion and decreased trust in the process. There is therefore a need for a systematic and secure digital solution that facilitates the recording, monitoring, and retrieval of lost and found items in an organized manner. The core problem addressed in this project is the development of a web-based platform that integrates user authentication, record management, and verification mechanisms to enhance accountability, accuracy, and efficiency in lost and found operations.

User Stories

1. As a **Finder**, I need to submit the lost item to the **Employee** so that they may upload it to the website.
2. As a **Claimer**, I need to check the website so that I may be able to know what items there are in the Lost and Found, check for my lost item(s) and claim them.
3. As an **Employee**, only I must have access to add new items, finders, and claimers.
4. As an **Employee**, only I must be able to edit any details of the items, finders, and claimers at any given point in time.
5. As an **Employee**, I must have my own set of credentials for security.
6. As an **Admin**, only I must be able to create new set of credentials for my employees.

System Architecture and System Design

The system for this project was designed using the MERN tech stack, which stands for MongoDB, ExpressJS, ReactJS and NodeJS. NodeJS is used for the backend, which communicates with MongoDB (the database) using a package called “mongoose” to transfer data and perform CRUD (Create, Read, Update, and Delete) operations, and ExpressJS is used to write the REST APIs (Representational State Transfer Application Programming Interface), which is used to connect the backend to the frontend. The front-end of this project is designed with ReactJS, as it is a popular choice and is very efficient at handling the operations due to its stateful infrastructure that allows to re-render certain aspects of the page as a hot-reload, rather than a complete reload of the page, increasing efficiency and reducing latency.

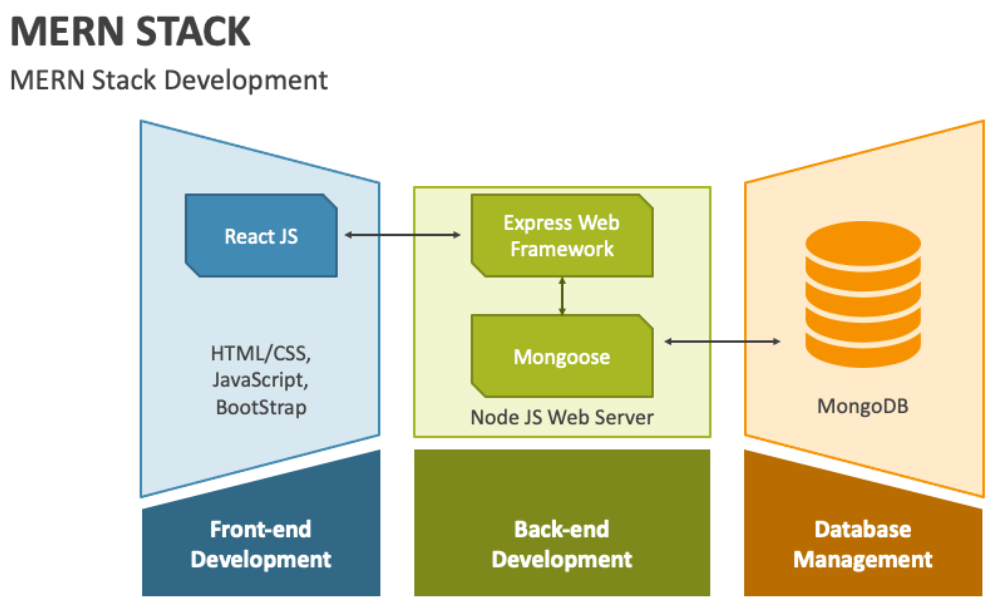


Fig 1: Architecture Diagram of an application developed using MERN tech stack.

Design of Tests

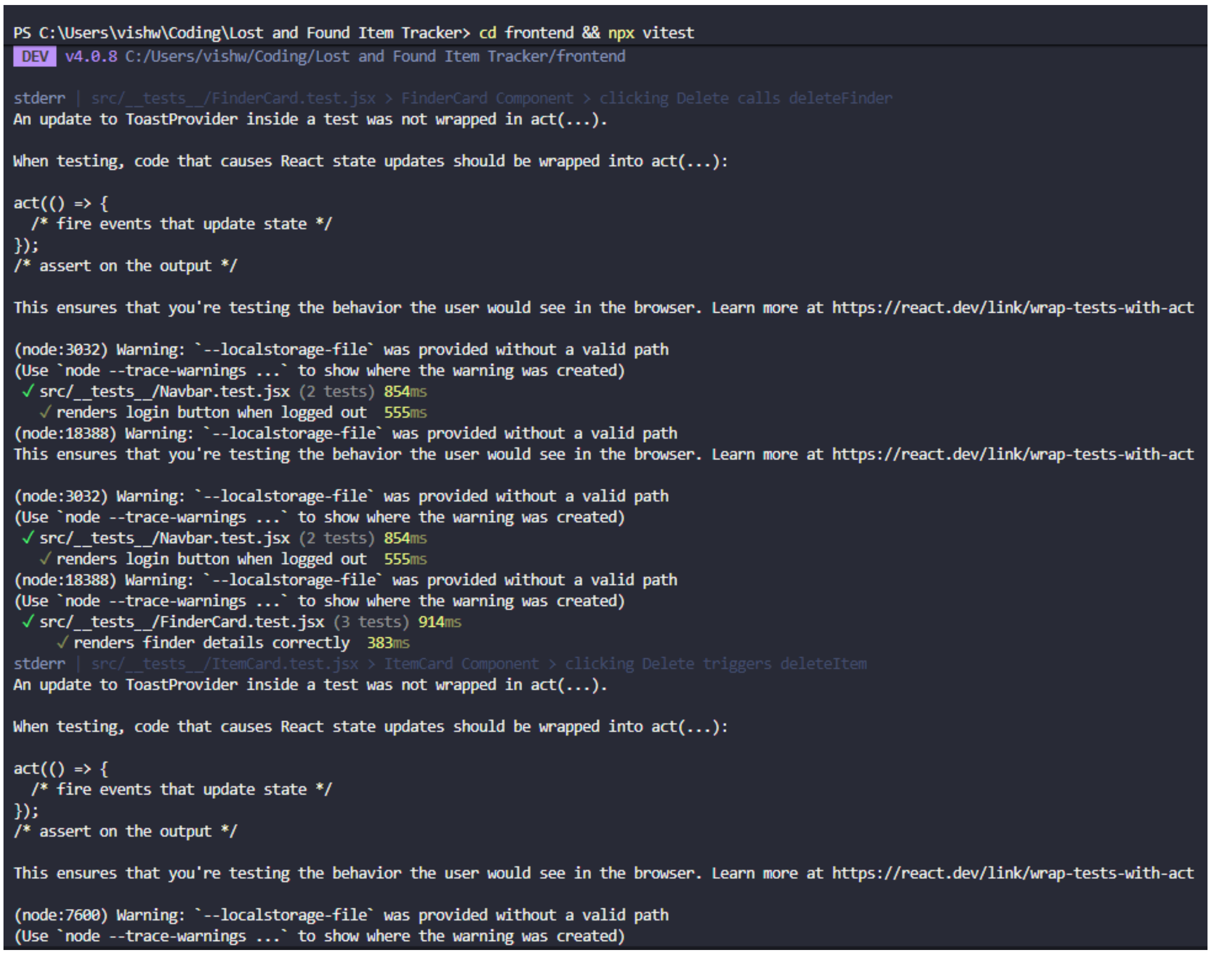
Since any application will require tests to verify if it runs properly, this project, Lost and Found Item Tracker made using MERN Stack uses the "vitest" npm package to write integration tests for the project. The tests were conducted for these interactions:

1. Login button must be rendered when the Employee has logged out.
2. Logout button must be rendered when the Employee has logged in.
3. To render buttons for item, finder and claimer in the Home Page.
4. To show "No finders found" on the finders' subpage if the list of finders is empty.
5. To call the API to fetch finders when the user clicks on the finder button.
6. To call the function responsible for maintaining the state of the variable that stores the view of the Home Page (item, finder, claimer).
7. To check if the FinderCard component renders the details of the finder perfectly in a card-like format.
8. To check if the FinderCard component shows the edit and delete buttons only when the employee has logged in.
9. To check if the delete button of the FinderCard component calls the function responsible for deleting the respective finder.

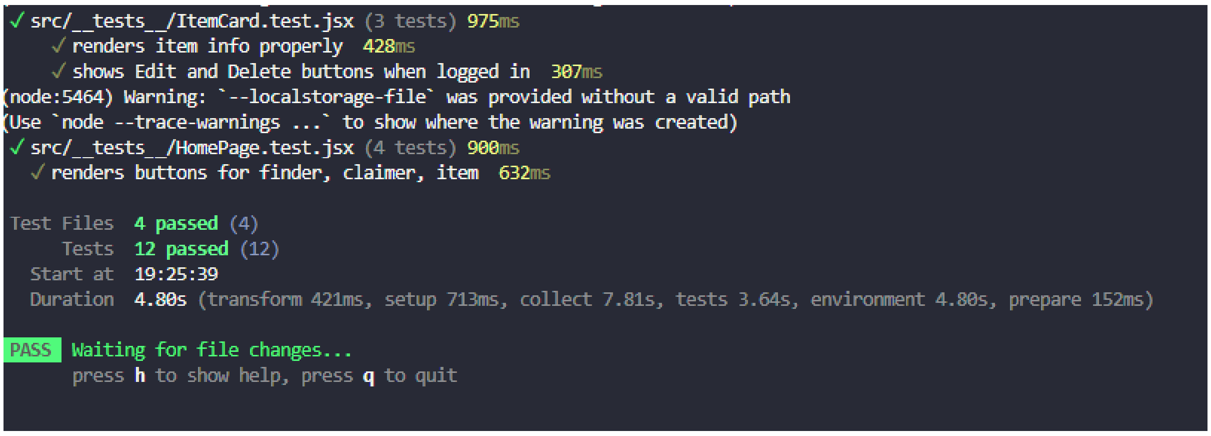
10. To check if the ItemCard component renders the details of the finder perfectly in a card-like format.
11. To check if the ItemCard component shows the edit and delete buttons only when the employee has logged in.
12. To check if the delete button of the ItemCard component calls the function responsible for deleting the respective item.

The tests for ClaimerCard were omitted since it was designed to behave the same way as FinderCard.

Some screenshots of the test are attached below:



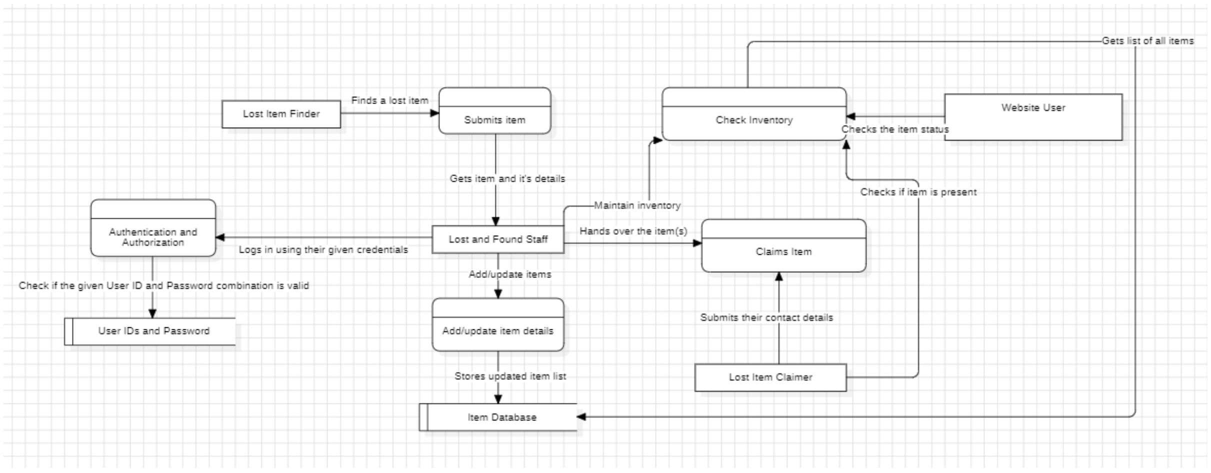
```
PS C:\Users\vishw\Coding\Lost and Found Item Tracker> cd frontend && npx vitest
 DEV  v4.0.8 C:/Users/vishw/Coding/Lost and Found Item Tracker/frontend

stderr | src/__tests__/FinderCard.test.jsx > FinderCard Component > clicking Delete calls deleteFinder
An update to ToastProvider inside a test was not wrapped in act(...).

When testing, code that causes React state updates should be wrapped into act(...):

act(() => {
  /* fire events that update state */
});
/* assert on the output */

This ensures that you're testing the behavior the user would see in the browser. Learn more at https://react.dev/link/wrap-tests-with-act

(node:3032) Warning: `--localstorage-file` was provided without a valid path
(Use `node --trace-warnings ...` to show where the warning was created)
 ✓ src/__tests__/Navbar.test.jsx (2 tests) 854ms
   ✓ renders login button when logged out  555ms
(node:18388) Warning: `--localstorage-file` was provided without a valid path
This ensures that you're testing the behavior the user would see in the browser. Learn more at https://react.dev/link/wrap-tests-with-act

(node:3032) Warning: `--localstorage-file` was provided without a valid path
(Use `node --trace-warnings ...` to show where the warning was created)
 ✓ src/__tests__/Navbar.test.jsx (2 tests) 854ms
   ✓ renders login button when logged out  555ms
(node:18388) Warning: `--localstorage-file` was provided without a valid path
(Use `node --trace-warnings ...` to show where the warning was created)
 ✓ src/__tests__/FinderCard.test.jsx (3 tests) 914ms
     ✓ renders finder details correctly  383ms
stderr | src/__tests__/ItemCard.test.jsx > ItemCard Component > clicking Delete triggers deleteItem
An update to ToastProvider inside a test was not wrapped in act(...).

When testing, code that causes React state updates should be wrapped into act(...):

act(() => {
  /* fire events that update state */
});
/* assert on the output */

This ensures that you're testing the behavior the user would see in the browser. Learn more at https://react.dev/link/wrap-tests-with-act

(node:7600) Warning: `--localstorage-file` was provided without a valid path
(Use `node --trace-warnings ...` to show where the warning was created)
```



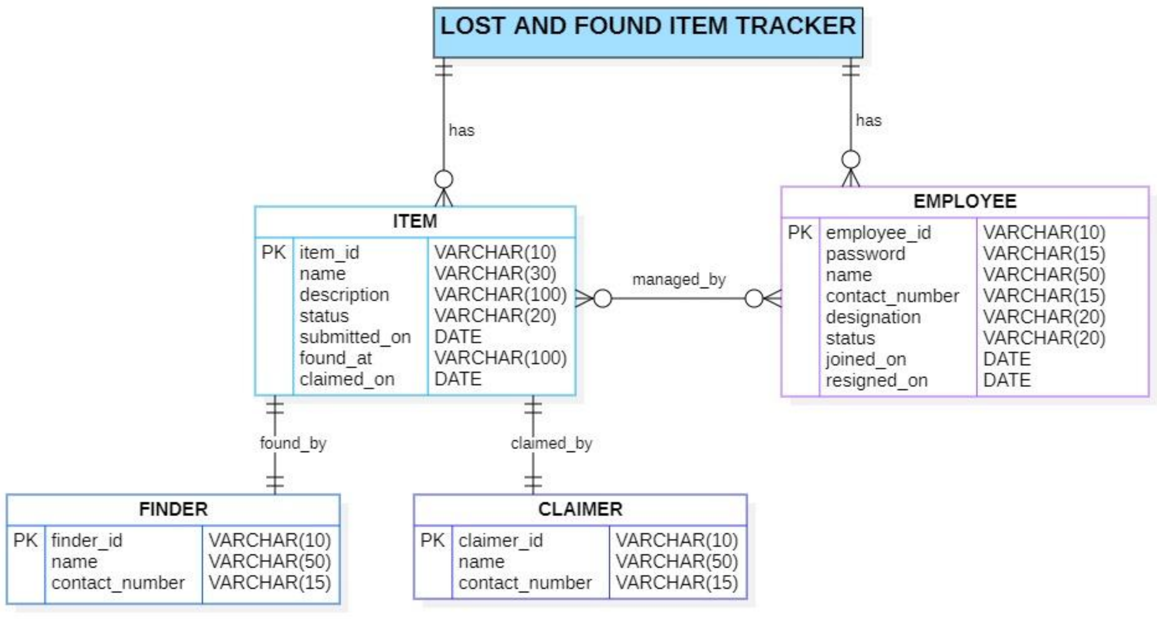
```
✓ src/__tests__/ItemCard.test.jsx (3 tests) 975ms
    ✓ renders item info properly  428ms
    ✓ shows Edit and Delete buttons when logged in  307ms
(node:5464) Warning: `--localstorage-file` was provided without a valid path
(Use `node --trace-warnings ...` to show where the warning was created)
✓ src/__tests__/HomePage.test.jsx (4 tests) 900ms
  ✓ renders buttons for finder, claimer, item  632ms

 Test Files  4 passed (4)
      Tests  12 passed (12)
   Start at  19:25:39
   Duration  4.80s (transform 421ms, setup 713ms, collect 7.81s, tests 3.64s, environment 4.80s, prepare 152ms)

 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

Appendix:

1. DFD

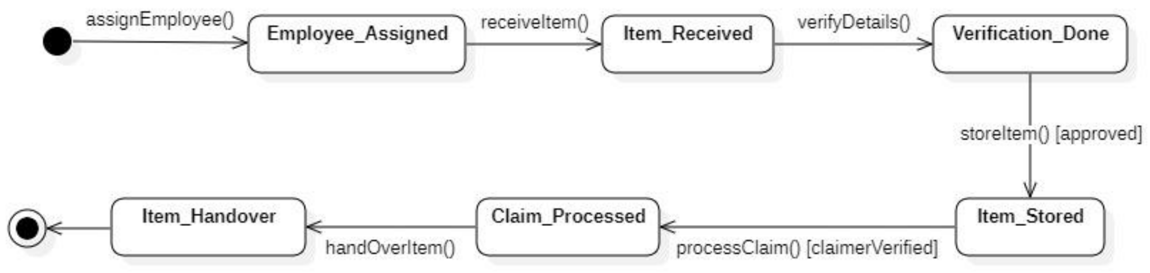


2. ERD

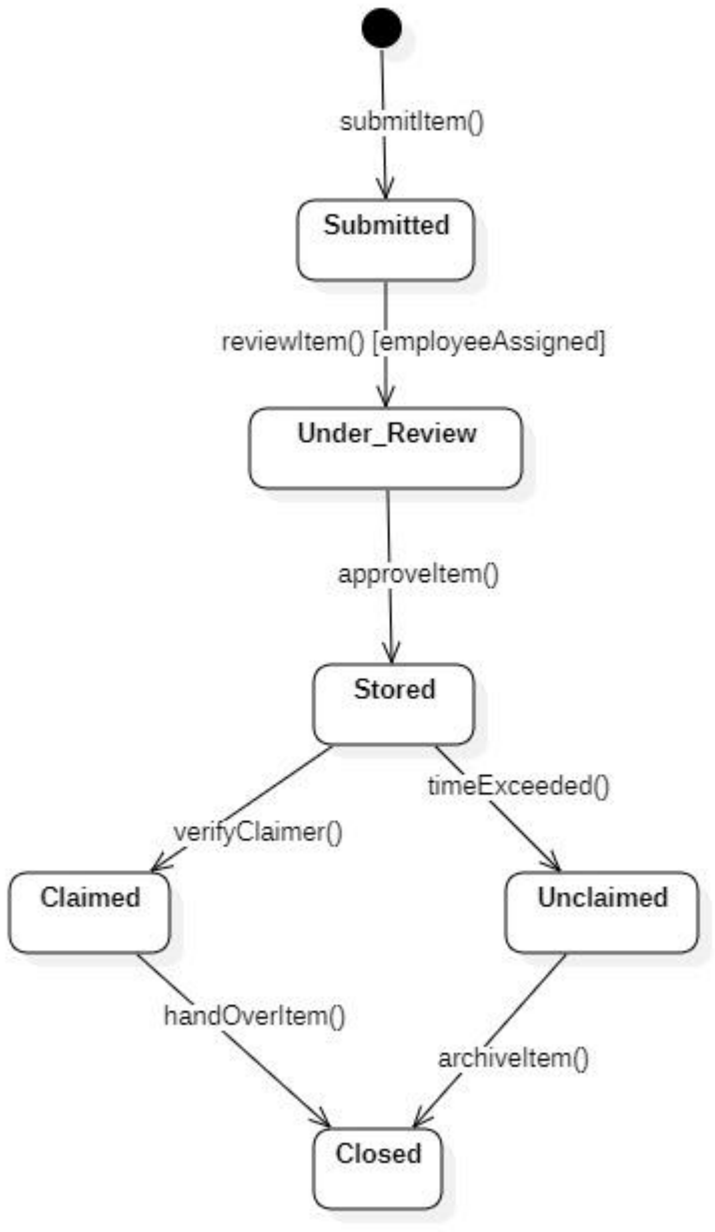


3. State Chart Diagrams

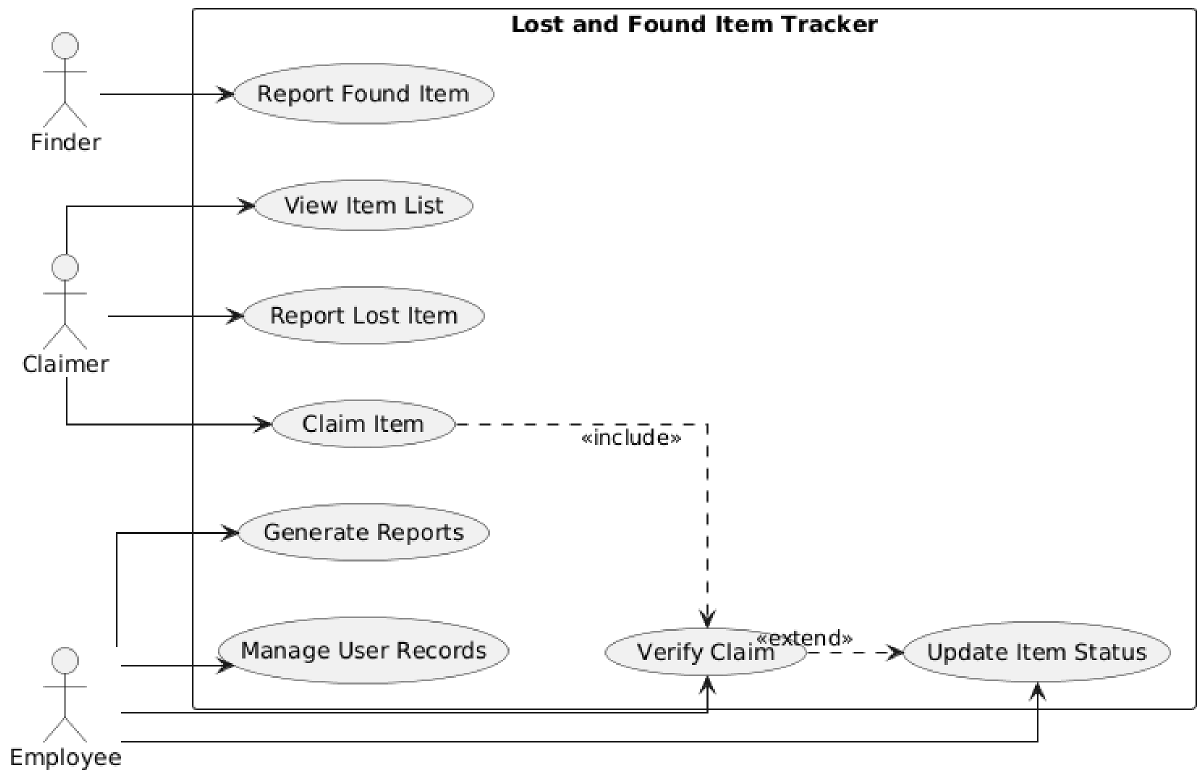
a. Employee Management of Items



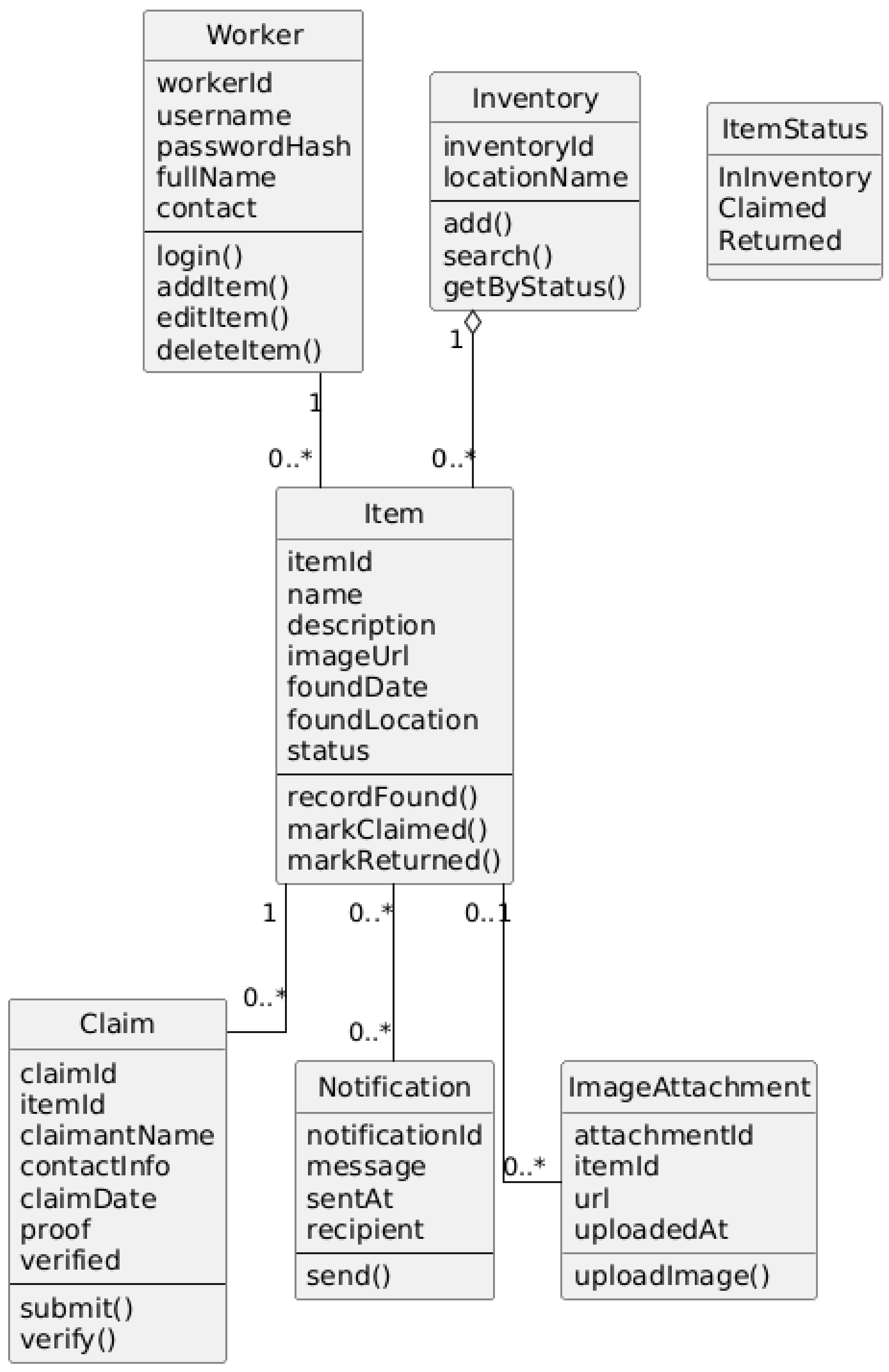
b. Item Lifecycle



4. Use Case Diagram

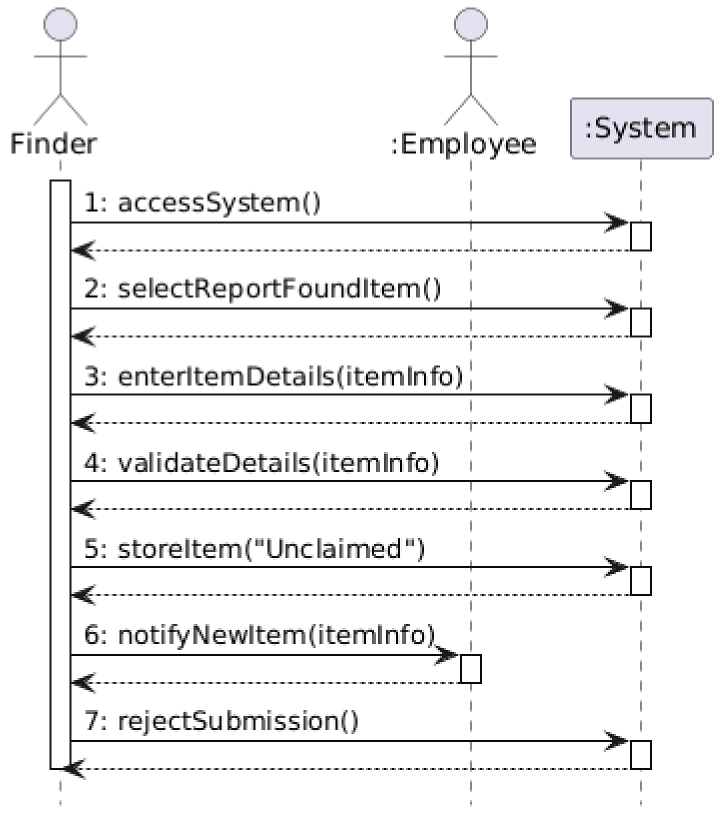


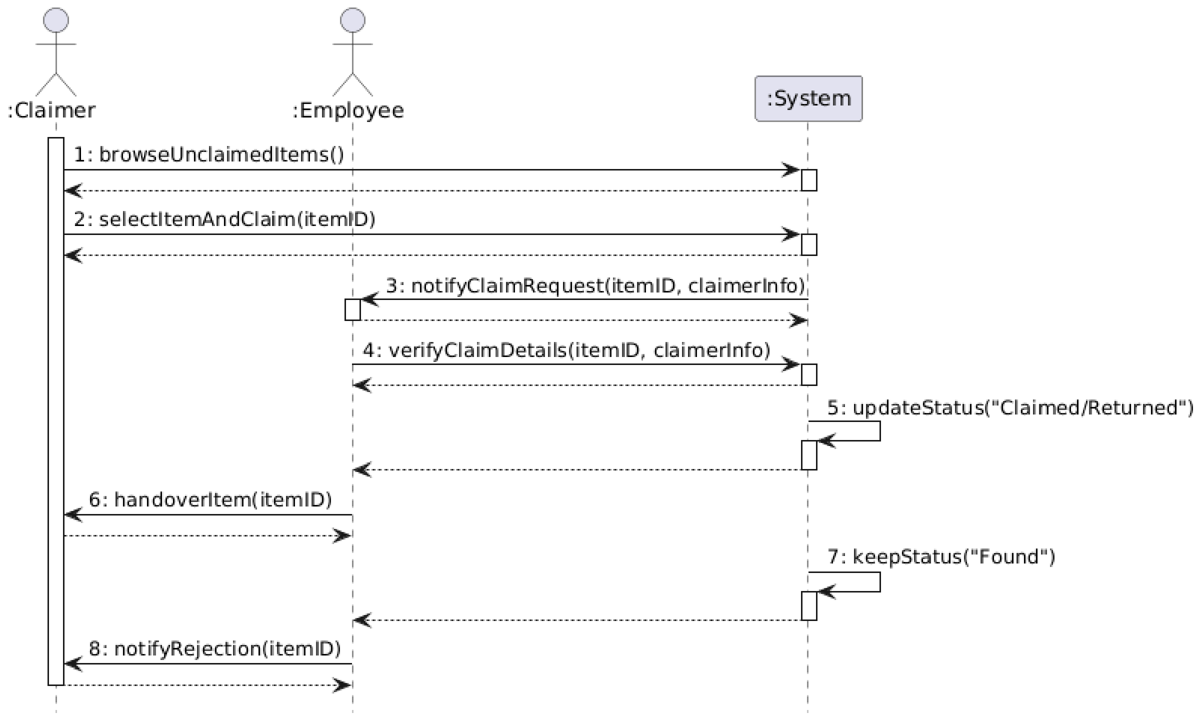
5. Class Diagram - Analysis Level



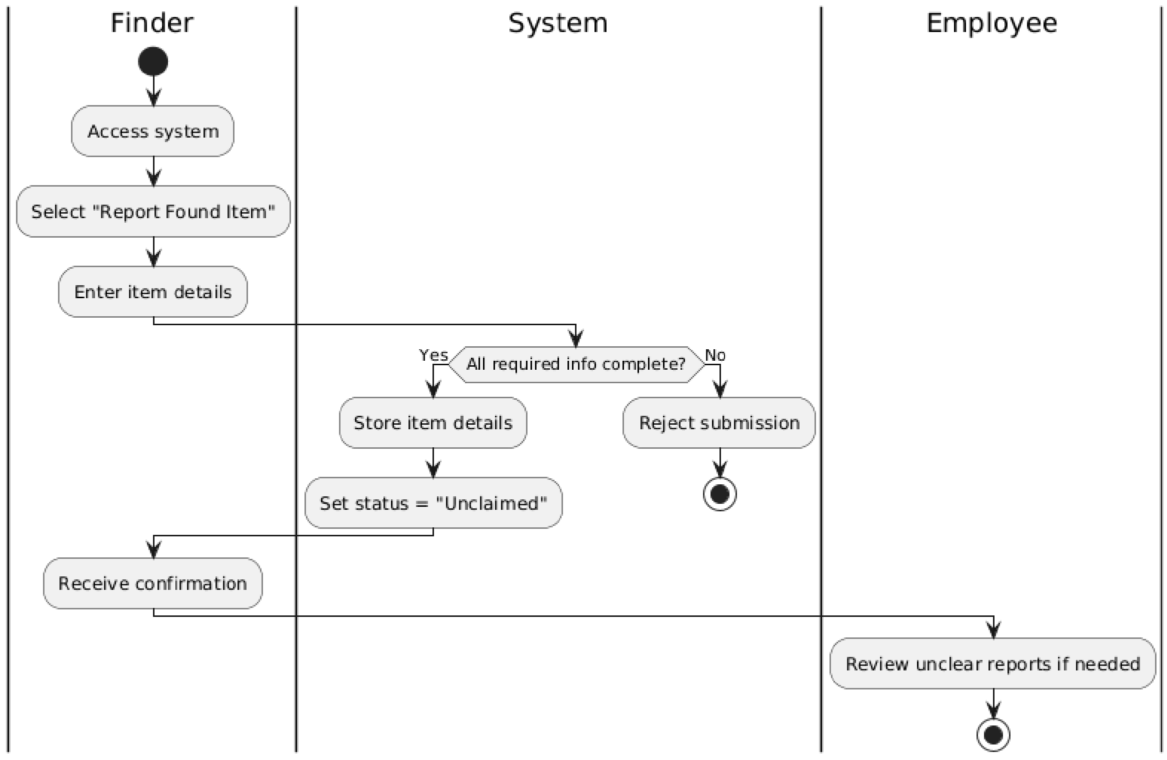
Worker
workerId
username
passwordHash
fullName
contact
login()
addItem()
editItem()
deleteItem()
Inventory
inventoryId
locationName
add()
search()
getByStatus()
ItemStatus
InInventory
Claimed
Returned
1
1
0..*
0..*
Item
itemId
name
description
imageUrl
foundDate
foundLocation
status
recordFound()
markClaimed()
markReturned()
1
0..*
0..1
0..*
0..*
0..*
Claim
claimId
itemId
claimantName
contactInfo
claimDate
proof
verified
submit()
verify()
Notification
notificationId
message
sentAt
recipient
send()
ImageAttachment
attachmentId
itemId
url
uploadedAt
uploadImage()

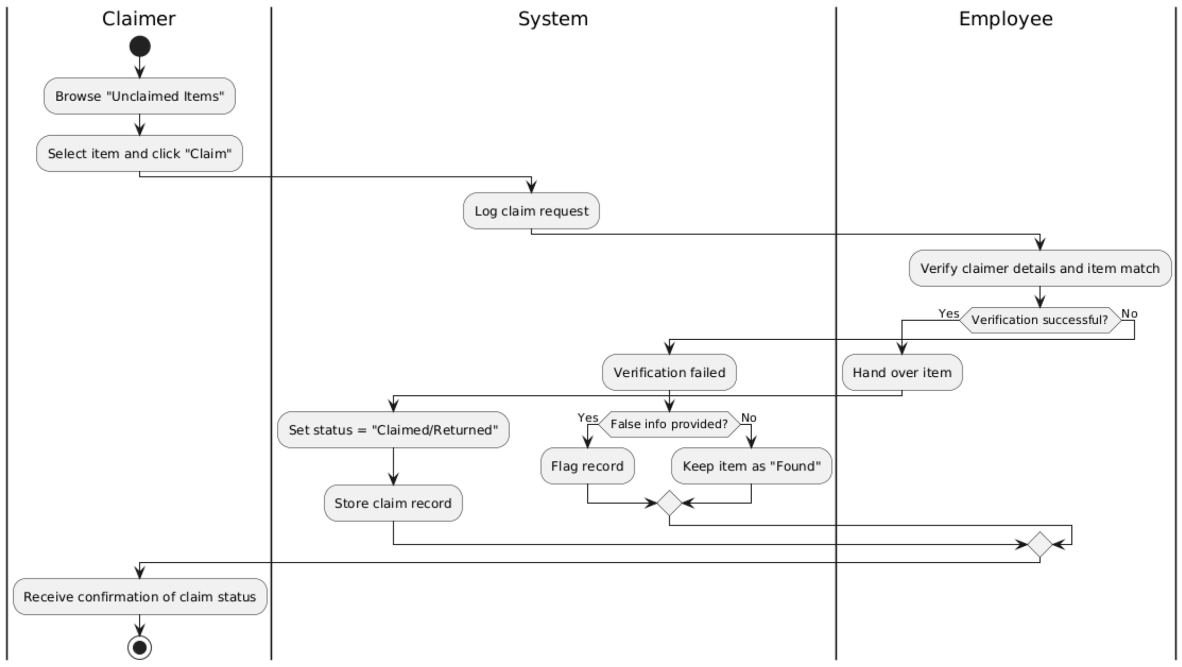
6. Sequence Diagrams


Finder
:Employee
:System
1: accessSystem()
2: selectReportFoundItem()
3: enterItemDetails(itemInfo)
4: validateDetails(itemInfo)
5: storeItem("Unclaimed")
6: notifyNewItem(itemInfo)
7: rejectSubmission()

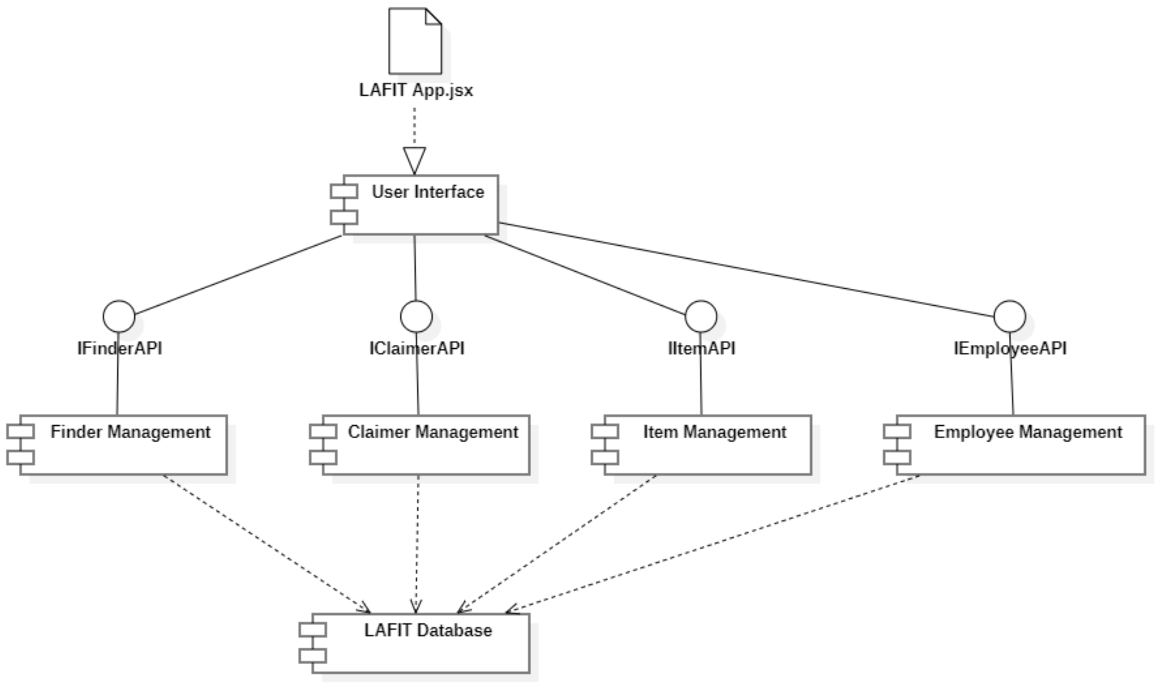

:Claimer
:Employee
:System
1: browseUnclaimedItems()
2: selectItemAndClaim(itemID)
3: notifyClaimRequest(itemID, claimerInfo)
4: verifyClaimDetails(itemID, claimerInfo)
5: updateStatus("Claimed/Returned")
6: handoverItem(itemID)
7: keepStatus("Found")
8: notifyRejection(itemID)

7. Activity Diagrams

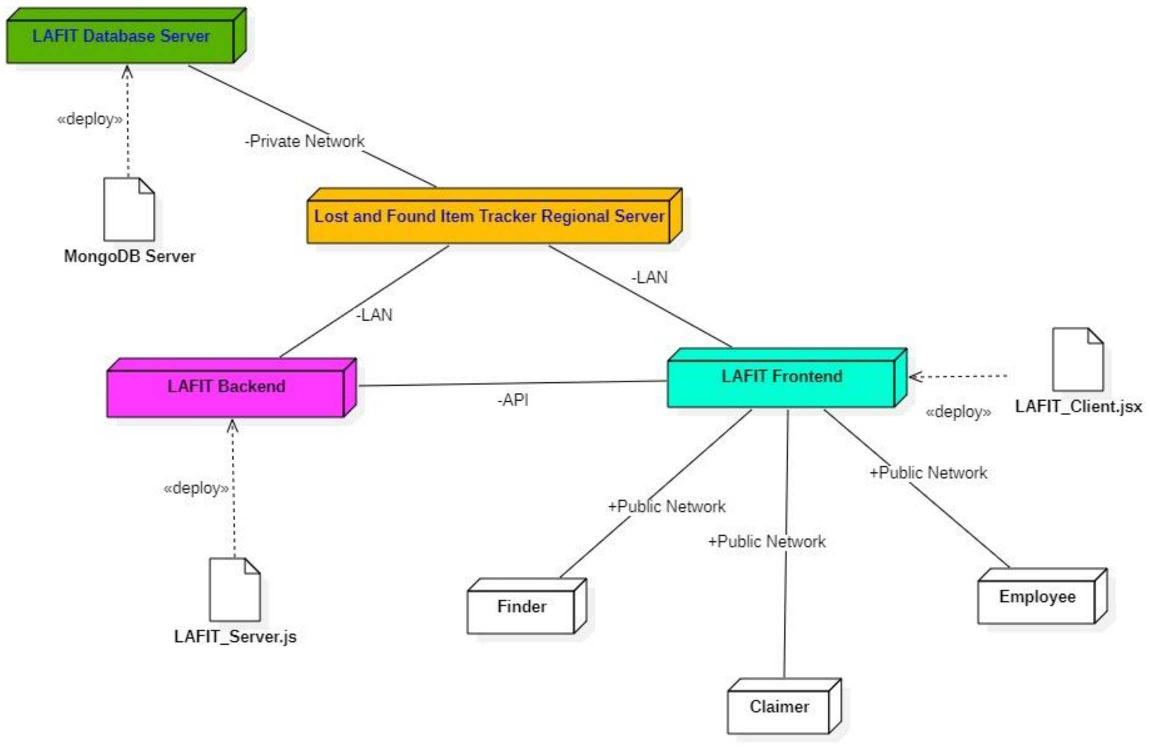




8. Component Diagram



9. Deployment Diagram



Code Listing

https://github.com/VishwajithS2005/Lost-and-Found-Item-Tracker

Done By:

S Vishwajith

23BCE1145