Table of Contents

Data Types

User

Attribute	Data Type	Nullable
name	String	Not Null
username	String	Not Null
password	String	Not Null

Return to Table of Contents

Individual

Attribute	Data Type	Nullable
job title	String	Not Null
date of hire	String	Not Null

Municipality

Attribute	Data Type	Nullable
municipality category	String	Not Null

Government Agency

Attribute	Data Type	Nullable
agency name and local office	String	Not Null

Company

Attribute	Data Type	Nullable
location of headquarters	String	Not Null
number of employees	Integer	Not Null

Resource

Attribute	Data Type	Nullable
name	String	Not Null
primary ESF	String	Not Null
additional ESFs	List<String>	NULL
model	String	NULL
capabilities	List<String>	NULL
home location latitude	Decimal Degrees	Not Null
home location longitude	Decimal Degrees	Not Null
cost	Float	Not Null
cost per	String	Not Null
maximum distance	Float	NULL
start date	Datetime	NULL
return by	Datetime	NULL
Status	String	Not Null

Return to Table of Contents

Incident

Attribute	Data Type	Nullable
ID	Integer	Not Null
type	String	Not Null
description	String	Not Null
date	Datetime	Not Null
location latitude	Decimal Degrees	Not Null
location longitude	Decimal Degrees	Not Null

Requests

Attribute	Data Type	Nullable
start date	Datetime	NULL
status	String	Not Null
return date	Datetime	Not Null

Business Logic Constraints

User

- When adding a resource, the current user is automatically set as the owner of the resource.
- When adding an emergency incident, the current user is automatically set as the owner of the incident.
- The system must record the municipality category (city, county, state, country) of municipalities.

Resource

- The system should be preloaded with the ESFs defined by FEMA for selection.
- The resource primary ESF should not also appear as an additional ESF.
- Resource requests for a specific incident must be initiated by the incident owner.
- Resources are deployed for a specific incident only when the resource owner accepts a resource request.
- A given resource cannot be used to respond to multiple incidents at the same time—i.e., a resource must return to the available status before it can be in use again.

Return to Table of Contents

- If a resource request is rejected, then the pending request should be removed from the current user and requesting user's resource status screens. It is possible that a rejected resource request might appear again if the requesting user performs another search and requests the resource again.
- Resources must be returned to the available status before they can be deployed again.
- Once a resource has been returned back to available status, the system should prevent the same resource from being requested again for the same incident. However, the returned resource may be requested to respond to other incidents.

Incidents

- There are four types of incident declarations that can be entered into ERMS.
    - Major Disaster Declaration (abbreviated MD)
    - Emergency Declaration (abbreviated ED)
    - Fire Management Assistance (abbreviated FM)
    - Fire Suppression Authorization (abbreviated FS)

Task Decomposition and Abstract Code

Login

Task Decomposition

Lock Types: Read-only on User table
Number of Locks: Single
Enabling Conditions: None
Frequency: Around 1000 logins per day, high frequency
Consistency (ACID): not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User enters *username* ('$username'), *password* ('$password') input fields
- If data validation is successful for both *username* and *password* input fields, then:
    - When ***Log In*** button is clicked:
        - If *username* is found but *password* is incorrect:

Return to Table of Contents

        - Go back to **Log In** form, with error message.
    - Else:
        - Store login information as session variable '$username'.
        - Go to **Main Menu** form.
- If both *username* and *password* input fields are invalid, then go back to **Log In** form with error message

Main Menu

Task Decomposition

Lock Types: User and the applicable Municipality/Individual/Agency/Company Tables are Read-only.
Number of Locks: Single
Enabling Conditions: Trigger by successful login.
Frequency: User Detail and Menu Options have the same frequency.
Consistency (ACID): not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed

Abstract Code

- Query for information about the user and their profile where $Username is the ID of the current user using the system from the HTTP Session/Cookie.
- Show **Main Menu** form
    - Find the current User using the User.username and Display user's name
    - If User is Municipality, Find Municipality.category and Display the category
    - If User is Agency, Find Agency.Name and Display agency's name
    - If User is Company, Find Company.location of headquarters and Company.number of employees. Display both.
- Upon:
    - Click ***Add Resource*** button - Jump to the ***Add Resource*** task.
    - Click ***Add Emergency Incident*** button - Jump to the ***Add Emergency Incident*** task.
    - Click ***Search Resources*** button - Jump to the ***Search Resources*** task.
    - Click ***Resource Status*** button - Jump to the ***Resource Status*** task.
    - Click ***Resource Report*** button - Jump to the ***Resource Report*** task.
    - Click ***Exit*** button - Logs user out of system and displays the **Login** form.

Return to Table of Contents

Add Resource

Task Decomposition

Lock Types: Read-only on User table and Update on Resource Table
Number of Locks: Couple different schema constructs needed
Enabling Conditions: Enabled when user clicks on ***Add Resource*** button from **Main Menu**
Frequency: Medium
Consistency (ACID): not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed

Abstract Code

- User clicked on ***Add Resource*** button from **Main Menu**
- Query for information about the user and their profile where $Username is the ID of the current user using the system from the HTTP Session/Cookie.
- Show **Add New Resource** form
    - Find the current User using the User.username and Display user's name as Owner
    - Create unique numeric Resource ID ($ResourceID) **and** Display Resource ID
- User enters *Resource Name* ($name), selects *Primary ESF* ($primary esf), selects *Additional ESFs* ($additional esfs), enters *Home Location* ($Lat and $Long), enters *Cost* ($Cost), selects *Cost/per* ($Costper), optionally enters Model ($Model), enters Capabilities ($Capabilities), and enters *Maximum Distance* ($maximum distance).
- Upon click **Cancel button** - Jump to the **Main Menu** form
- If data validation is successful for all fields, then:
    - Upon click **Save** button:
        - Store resource information as row in Resources and set Resource.status as "Available"
- Else if required fields not selected or any input fields invalid, display **Add New Resource** form with error message

Return to Table of Contents

Add Emergency Incident

Task Decomposition

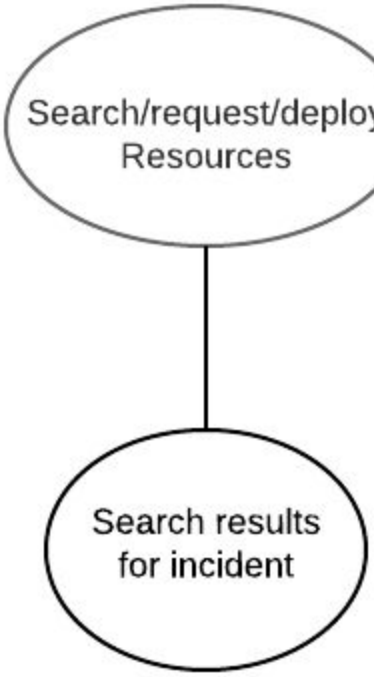
Lock Types: Read-only on User table and Update on Incident Table
Number of Locks: Couple different schema constructs needed
Enabling Conditions: Enabled when user clicks on ***Add Emergency Incident*** button from **Main Menu** form
Frequency: Medium
Consistency (ACID): not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed

Abstract Code

- User clicked on ***Add Emergency Incident*** button from **Main Menu**
- Query for information about the user and their profile where $Username is the ID of the current user using the system from the HTTP Session/Cookie.
- Show **New Incident** form
    - User selects *Declaration* ($Declaration), enters *Date* ($Date), enters *Description* ($Description), enters *Location* ($Lat and $Long).
- Upon click **Cancel button** - Jump to the **Main Menu** form
- If data validation is successful for all fields, then:
    - Upon click **Save** button:
        - Store incident information as row in Incident
        - Create unique *IncidentID* ($IncidentID)
            - Find *Declaration* ($Declaration) input by user and concatenate with autogenerated numeric unique ID
            - Store as *IncidentID* ($IncidentID)
- Else if required fields not selected or any input fields invalid, display **New Incident** form with error message

Return to Table of Contents

Search Resources



Task Decomposition

Lock Types: Read-only/Update on Resource, Incident, and Requests table
Number of Locks: Couple different schema constructs needed
Enabling Conditions: Enabled when user clicks on ***Search Resources*** button from **Main Menu** form
Frequency: Medium
Consistency (ACID): Critical since available/in-use resources must be up-to-date
Subtasks: Search Results for Incidents task

Abstract Code

- User clicked on ***Search Resources*** button from **Main Menu** form
- Query for information about the user and their profile where $Username is the ID of the current user using the system from the HTTP Session/Cookie.
- Show **Search** form
    - Displays text input for search keywords
    - Displays dropdown of ESF functions
        - Find all unique ESF functions from Look-up on Resource.primaryESF and Resource.additionalESF and display

Return to Table of Contents

    - Displays text input for proximity to incident field
    - Displays dropdown of User-owned Incidents
        - Find the current User using the User.username and find all unique incidents that the User owns by look-up on Incidents table on Incident.ID
        - Look up Incident.ID and Incident.Description, concatenate, and display in dropdown
    - Show ***Cancel*** and ***Search*** buttons
- User optionally enters keywords, optionally selects ESF function, optionally selects proximity to emergency incident by entering a distance and selecting an incident from incident list dropdown
- Upon
    - Click ***Cancel*** button - Jump to the **Main Menu** form
    - Click ***Search*** **button** - Jump to the ***Search Results for Incident*** subtask

Search Results for Incident

Task Decomposition

Lock Types: Read-only on User and Resource tables and Update on Requests Table
Number of Locks: Couple different schema constructs needed
Enabling Conditions: Requires a successful query from the ***Search for Resources*** Task
Frequency: Typically same frequency as ***Search for Resources*** Task
Consistency (ACID): critical, since resource status should be up-to-date
Subtasks: Mother task is **Search Resources** task

Abstract Code

- User clicked on ***Search*** button from **Search Resources** and query was successful
    - If User has input any search criteria, all search criteria must be matched using “and” conditions:
        - If User inputs keywords, Find matching substrings in Resource.Model, Resource.Capabilities, and Resource.name
        - If User selects ESF, Find matching substrings in Resource.primaryESF and Resource.additionalESF
        - If User selects Incident:

Return to Table of Contents

                - If User selected distance, then find resources that are within the selected distance of incident by finding Resource.Lat and Resource.Long and calculating distance of each resource from Incident.Location
                - If distance is not input, then distance is not calculated
        - If all fields are empty, read-only on resources table and return all resources
            - Find Resource.ID, Resource.name, User.username of Resource owner, Resource.cost, Resource.status, and Requests.returndate
- Display **Search Results for Incident** form displaying incident name, Table of results, and **close** button
    - Incident Name
        - If User had selected Incident in **Search Resources** form, Display Incident.Description and Incident.ID
        - Else leave blank
    - Table of results - Present the query results from the ***Search for Resources*** Task in a tabular format. Sort results by distance, availability, and then name.
        - Always display ID (Resource.ID), Name (Resource.Name), Owner (User.username who owns Resource), Cost (Resource.cost), Status (Resource.status), Next Available (Requests.returndate) columns
        - If Resource.status is Available
            - Display text “NOW” in Next Available Column
        - If Resource.status is not Available:
            - Display Requests.returndate
        - If User had input incident and distance in **Search Resources** form, also Display Distance columns
            - Display Distance of each resource calculated from query results
        - If User had input incident in **Search Resources** form, also Display Action columns
            - In Action column, Display **Request**, **Deploy**, or no button
                - If Resource.status is available:
                    - If the resource owner $user_id and incident owner $user_id match:
                        - Display a **Deploy** button under the Action column.
                    - Else if no request is found in Request table for specific resource by current User:
                        - Display a **Request** button under the Action column.
                    - Else:
                        - Display an empty cell under the Action column.
    - Upon click **Close button** - Jump to the ***Search Resources*** task
    - Upon click **Request** button for a table record:

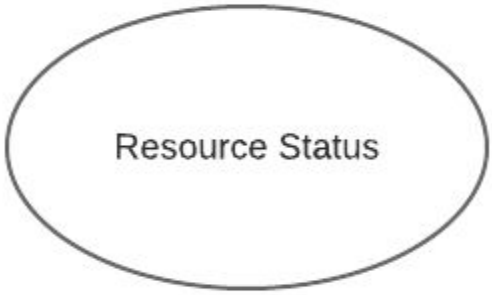
Return to Table of Contents

- Prompt user for an expected return date
    - If User enters return date and clicks “enter”:
        - Store return date information as a new record in Requests.returndate
        - Store return date information as a new record in Requests
        - Leave start date blank
        - Store Requests.status as “pending”
        - Return to **Search Results for Incident** form and Remove **Request** button for the table record
    - Else if User closes prompt, return to **Search Results for Incident** form

- Upon click **Deploy** button for a table record:
    - Prompt user for an expected return date
        - If User enters return date and clicks “enter”:
            - Store return date information as a new record in Requests.returndate
            - Store return date information as a new record in Requests
            - Store start date as Requests.startdate as the current date
            - Store Requests.status as “deployed”
            - Return to **Search Results for Incident** form and Remove **Deploy** button for the table record
        - Else if User closes prompt, return to **Search Results for Incident** form

Return to Table of Contents

Resource Status



Task Decomposition

Lock Types: read-only and updates lookups of Resources, Requests and Incident
Number of Locks: Several different schema constructs are needed
Enabling Conditions: Enabled when user clicks on ***Resource Status*** button from **Main Menu** form
Frequency: Updates will vary by change in requests statuses
Consistency (ACID): Critical so that resource statuses, and request statuses are up-to-date

Abstract Code

- User clicked on ***Resource Status*** button from **Main Menu**
- Display Resources in Use table, Resources Requested by Me table, and Resource Requests Received by Me table
- Show **Resources in Use** Table
    - Find all resources that User had requested and is using (where Resource.status is deployed):
        - Display ID (Resource.ID), name (Resource.name), incident (Incident.name), start date (Requests.startdate), return date and (Requests.returndate)
        - Under Action column - Display **Return** button
    - Upon click **Return** button
        - Update Requests.status to “Returned”
        - Update Resource.status to “Available”
        - Remove resource from displaying on **Resource in Use** table
- Show **Resources Requested by me** Table
    - Find all resources in Resources table where Requests.status is “Pending” and Requests.username matches User.username

Return to Table of Contents

        - Display ID (Resource.ID), name (Resource.name), related incident (Incident.name), owner (User.username) and return date (Requests.returndate)
        - Under Action column - Display **Cancel** button
    - Upon click **Cancel** button
        - Delete record from Requests table
        - Remove display of record from **Resources Requested by me** Table

- Show **Resource Requests Received by Me** Table
    - Find all resources that User owns in Resource table where Requests.status is “pending”
    - Display ID (Resource.ID), name (Resource.name), related incident (Incident.name), owner (User.username), return date (Requests.returndate) and Action columns.
        - If Resource.status is “Available”
            - Display “**Deploy**” and “**Reject**” buttons under Action Columns
        - Else if Resource.status is “In Use”
            - Display “**Reject**” buttons under Action Columns
        - Upon click **Deploy** button:
            - Update Resource.status to “in use”
            - Update Requests.status to “deployed”
            - Remove display of record from **Resources Received by Me** Table
        - Upon click **Reject** button:
            - Delete record from Requests table
            - Remove display of record from **Resources Received by Me** Table

Return to Table of Contents

Resource Report

Task Decomposition

Lock Types: Read-only look-up on Resources
Number of Locks: Enabled when user clicks on ***Resource Status*** button from **Main Menu** form
Enabling Conditions: All are enabled by a user's login
Frequency: ESF# have the same frequency
Consistency (ACID): order is not critical

Abstract Code

- User clicked on ***Resource Report*** button from **Main Menu**
- Query for information about the user and their resources where $UserID is the ID of the current user using the system from the HTTP Session/Cookie
- Display **Resource Status** table with columns ESF#, Primary ESF, Total Resources, Resources in Use
    - Display ESF # and Primary ESF columns - Find all unique ESFs in Resource.PrimaryESF; Display ESF# and ESF name
    - Display Total Resources column
        - Sum number of resources If User owns Resource, Display sum by ESF# in table row
    - Display Resources in Use column
        - Sum resources if User owns Resource and if Resource.status is "In Use", Display sum by ESF# in table row
    - Display last row of Table as Total Row
        - Display Sum of "Total Resources" column and Sum of "Resources in Use" column

Return to Table of Contents