Table of Contents

Data Types: **2**

Business Logic Constraints: **4**

Task Decomposition & Abstract Code **5**

1. Login **5**

2. Main Menu **5**

3. Add Resource **6**

4. Add Incident **7**

5. Search Resources **8**

6. Search Result **9**

7. Deploy Resource Task **10**

8. Request Resource Task **10**

9. Cancel Request **11**

10. Reject Request **11**

11. Return Resource **12**

12. View Report **12**

13. Populate Resource Status **13**

Data Types:

User		
Attribute	**Data Type**	**Nullable**
Name	String	Not Null
Password	String	Not Null
Username	String	Not Null

Municipality		
Attribute	**Data Type**	**Nullable**
City	String	Not Null
State	String	Not Null
Country	String	Not Null
County	String	Not Null

Individual_User		
Attribute	**Data Type**	**Nullable**
Job_Title	String	Not Null
Date_Hired	Date	Not Null

Government_Agency		
Attribute	**Data Type**	**Nullable**
Local_Office	String	Not Null
Agency_Name	String	Not Null

Company		
Attribute	**Data Type**	**Nullable**
Headquarters	String	Not Null
No_Of_Employees	Integer	Not Null

Declaration		
Attribute	**Data Type**	**Nullable**
Abbreviation	String	Not Null
Incident_description	String	NULL

Requests		
Attribute	**Data Type**	**Nullable**
Request_Status	String	Not Null
Return_by	Datetime	NULL
Date_requested	Datetime	Not Null

Cost_per		
Attribute	**Data Type**	**Nullable**
Unit	String	Not Null

ESF		
Attribute	**Data Type**	**Nullable**
ESF_ID	Integer	Not Null
ESF_description	String	Not Null

Resource		
Attribute	**Data Type**	**Nullable**
Resource_ID	Integer	Not Null
R_Owner	String	Not Null
Resource_Status	String	Not Null
Cost_amt	Float	Not Null
Latitude	String	Not Null
Longitude	String	Not Null
Resource_Name	String	Not Null
Model	String	Not Null
Max_Distance	Float	Not Null
Capabilities	String	NULL
Standard_cost	Float	Not Null

Incident		
Attribute	**Data Type**	**Nullable**
Incident_ID	String	Not Null
I_Owner	String	Not Null
Integeritude	Integer	Not Null
Latitude	Integer	Not Null
Description	String	Not Null
Date	Datetime	Not Null

Business Logic Constraints:

The following section lists business logic constraints that cannot be reflected in EER.

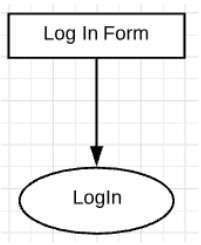
- The resource's owner is automatically set to the current user
- Latitude/Longitude attributes must be valid geographic coordinates
- Unique ID of incident should combine the abbreviation of the incident type with an automatically generated number unique to that type
- If an incident is spread-out, user should choose the central location as the location of the incident
- All incidents are private to users who create them, and incidents cannot be shared
- When using keyword as searching criteria, the function should match substring in the name, model, and capability field
- When using ESF as searching criteria, the function should match primary or additional ESF
- When using location as search criteria, the function should match resources with a home location within the given radius
- A resource can be requested directly from the search results only when the user selected the incident on the searching criteria form
- Distance and action showed only if incident was selected
- Both available and used resources can be requested
- Resources must be returned to the available status before it can be deployed again
- The system should prevent resources being deployed to the same incidents after it is returned
- The system should allow user to:
    - Request or cancel a request for resource
    - Deploy or reject a resource; deploy should be disabled for resources that are in use
- If a user requests his own resource, then it is automatically deployed
- Request ID is a combination of Incident ID and Resource ID

Task Decomposition & Abstract Code

1. Login

Task Decomposition

Lock Types: Read only on User table
Number of Locks: One DB Lock
Enabling Conditions: None
Frequency: Once per session, many sessions per day
Consistency: Not critical, order is not critical
Subtasks: Mother Task is not required
Decomposition*:* Not Required



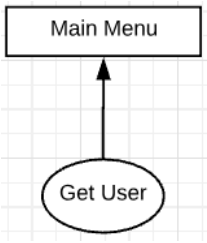
Abstract Code

- User enters *email('$Email'), password('$Password')* input fields.
- If data validation is successful for both *username* and *password* input fields then:
    - When ***Enter*** button is clicked:
        - If User record is not found
            - Go back to **Login** form with error message
        - If User record is found but User.password != '$Password'
            - Go back to **Login** form with error message
        - Else:
            - Store Login information as session variable '$UserId'
            - Go to **Main Menu** page.
- Else *email* and *password* input fields are invalid, display **Login** form with error message.

2. Main Menu

Task Decomposition

Lock Types: Read only on User table
Number of Locks: One DB Lock
Enabling Conditions: Enabled by a user's login
Frequency: Many times per session
Consistency: Not critical, order is not critical
Subtasks: Mother Task is not required
Decomposition*:* Not Required



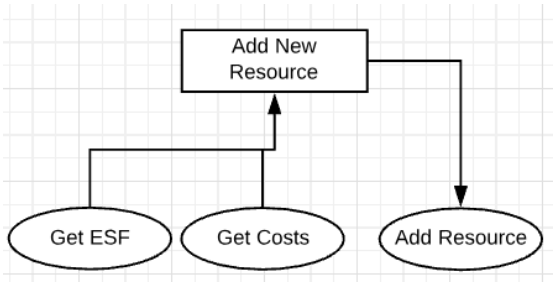
Abstract Code

- Get User.Username from session query
- *Run **Get User** task**;*** query for information about the user and their profile where:
    - If User.Type == "Municipality"; Display municipality category
    - If User.Type == "Government"; Display agency name
    - If User.Type == "Company"; Display location of HQ and Number of employees
- Display links to:
    - Add Resource Page
    - Add Incident
    - Search Resources

- Resource Status
- Resource Report

3. *Add Resource*

Task Decomposition



Lock Types: Read only on ESF table, Read on Cost_Per table, Insert to Resources table
Number of Locks: Many (2+) DB Locks needed
Enabling Conditions: Enabled by a user login and add of resource
Frequency: A few times per session
Consistency: not critical, order is not critical
Subtasks: All tasks must be done in order. Mother task is required to coordinate subtasks
Decomposition*:* Required

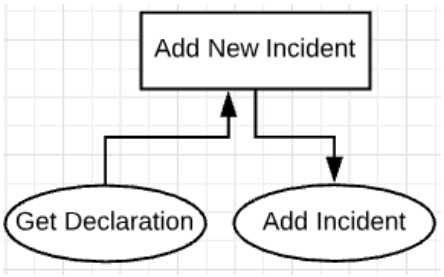
Abstract Code

- User clicked on ***Add Resource*** button from **Main Menu**
- Get User.Username from session query
- Run the **Get ESF & Costs** task; query for information about available ESFs and Cost Frequency/Cost_per.
  - From the ESF table lookup available ESFs
  - Populate ESF dropdown in **Add Resource Form**
- Run **Get Costs** task: query for information on available cost pers
  - From Cost_Per table, lookup available cost per units measure.
  - Populate *cost per* dropdown in **Add Resource Form**
- Auto-Assign Resource ID
- Auto-assign Resource Owner to the current user
- Display ***Add Resource Form***
- User enters
  - *Resource_Name(‘$Resource_Name’)*
  - *Model (‘$Model’)*
  - *Capabilities(‘[$Capabilities]*)
  - *Latitude(‘$Home.Latitude’)*
  - *Longitude (‘$Home.Longitude’)*
  - *Max_Distance(‘$Max Distance*)
  - *Cost(‘$Cost*)
- User selects
  - *Primary ESF(‘$Primary_ESF’)*
  - *Secondary_ESFs (‘[$Primary_ESF’])*
  - *Per (‘$Cost_Per*) from the respective dropdowns

- If data validation is successful for input fields then:
  - When ***Save*** button is clicked:
    - Store Resource Information as new entry to Resource table
    - Go to **Main Menu** page.
- Else *input* fields are invalid, display **Add Resource** form with error message and highlight the fields that need to be edited

4. Add Incident

Task Decomposition



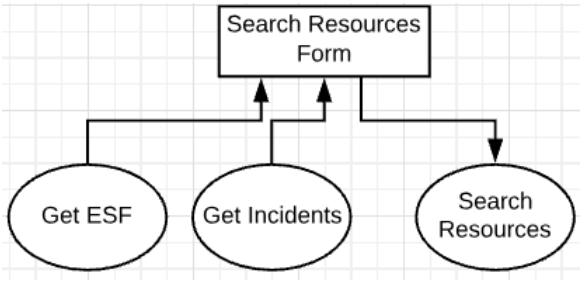
Lock Types: Read only on Declared_As table, Insert to Incident table
Number of Locks: Several schema constructs are needed
Enabling Conditions: Enabled by a user login and add new incident
Frequency: A few times per session
Consistency: not critical, order is critical
Subtasks: All tasks must be done in order. Mother task is required to coordinate subtasks
Decomposition*:* Required

Abstract Code

- User clicked on ***Add Incident*** button from **Main Menu**
- Get User.Username from session query.
- Run the **Get Incident Declaration** task; query for information about available incident types
  - From the Declaration table, lookup available incident declarations and descriptions
  - Populate *Declaration* dropdown on ***New Incident Form***
- Display ***Add Incident Form***
- User enters:
  - *Date('$Date)*
  - *Description('$Description'),*
  - *Latitude('$Home.Latitude'),*
  - *Longitude('$Home.Longitude'),*
- User selects *Declaration('$Incident_Type* from the respective dropdowns
- If data validation is successful for input fields then:
  - When ***Save*** button is clicked:
    - Auto-Assign Incident ID based on declaration
    - Auto-assign the owner of incident to logged-in user
    - Store Incident Information as new entry to Incident table
    - Go to **Main Menu** page.
- Else *input* fields are invalid, display **Add Incident** form with error message

5. Search Resources Form

Task Decomposition



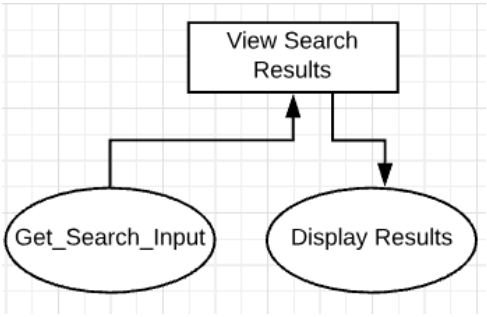
Lock Types: Read only on ESF table and read only on Incident table
Number of Locks: Two DB Locks
Enabling Conditions: Enabled by a user login and resource search
Frequency: Many times per session
Consistency: not critical
Subtasks: Mother task is not required
Decomposition*:* Not Required

Abstract Code

- User clicked on ***Search Resources*** button from **Main Menu**
- Get User.Username from session query.
- Run the **Get ESF** task; (defined above)
- Run the **Get Incidents** task
    - From Incidents table, select all available incidents regardless of the owner
- Display *Resource Search Form* with ESF and Incident dropdowns populated
- User enters
    - *keyword('$args)*
    - *ESF('$ESF)*
    - *Incident('$Incident)*
    - *Location('$MaxLoc)*
- User selects ESF and Incident from dropdown menus (optional).
- If data validation is successful for input fields then:
    - When ***Search*** button is clicked:
        - If an incident is selected, calculate the distance between the incident and all resources
        - Query database for requested parameters with multiple search criteria ANDed together.
        - Run the **Search Incident Resources** task; query for information about associated resources with a given incident.
            - From Resources select all resources that match the criteria specified in the **Search Resources** form including Keyword, ESF, Location and incident ID.
        - Go to **Search Results** page/task.
    - If no search fields are populated, return all resources in the system.
- Else *input* fields are invalid, display **Search Resources** form with error message

6. Display Results

Task Decomposition



Lock Types: Read only on Incident table, read only on Resources, read only on Requests table.
Number of Locks: Many (2+) DB Locks
Enabling Conditions: Enabled by a user login and incident search
Frequency: Many times per session
Consistency: critical. All incidents and statuses need to be shown correct
Subtasks: Mother task is required to coordinate subtasks
Decomposition*:* Required

Abstract Code

- User clicked on ***Search*** button from **Search Resources**
- From Resources select all resources that match the criteria specified in the **Search Resources** form including Keyword, ESF, Location and incident ID.
- Display *Incident Search Form Results* with Resource.Resource_ID, Resource.Name, Resource.Owner, Cost, Resource.Resource_Status for each resource.
    - For each resource, calculate the distance, if it's qualified, then display that record with distance in the result screen.
    - If Resource.Resource_Status == "Not Available" then Display Request.Return_by as the Next Available date, where Request.Resource_ID = Resource.Resource_ID.
    - If Resource.Resource_Status == "Available" then Display "NOW" as Next Available date.
- The search results will be sorted first based on distance in ascending order, then alphabetically by the resource name
    - If Resource.Owner == User.Username && Resource.Resouce_Status == ”Available” then display **Deploy** button.
    - Else:
        - Display **Request** button.

7. Deploy Resource Task

Task Decomposition

Lock Types: Update to Requests table, Update to Resources Table
Number of Locks: 2 DB locks
Enabling Conditions: Enabled by a user login and clicking **Deploy** button
Frequency: A few times per session
Consistency: Critical, DB needs to be updated correctly.
Subtasks: None
Decomposition*: Not* Required

Abstract Code

- User clicked on ***Deploy*** button from either **Search Resources** or Resource Status
- Get Incident.Incident_ID from session query.
- Run the **Update Resource Availability Task** task; update resource information
    - From Resources select resources that was selected.
    - If Resource.Owner == CurrentUser
        - Update Resource.Resource_Status to “Not Available”
    - Else reject DB update
- Run the **Update Request Task** task; update request information
    - From Request table select Request where Request.Resource_ID == Current Resource && Request.Incident_ID == Current Incident
    - If Request.Resource.Owner == CurrentUser:
        - Update Request.Request_Status to “DEPLOYED”
    - Else
        - Reject DB Update.

8. Request Resource Task

Task Decomposition

Lock Types: Insert to Resources Table
Number of Locks: 2 DB locks
Enabling Conditions: Enabled by a user login and clicking **Request** button
Frequency: Once/a few times per session
Consistency: Critical, DB needs to be updated correctly.
Subtasks: None
Decomposition*: Not* Required

Abstract Code

- User clicked on ***Request*** button from either **Search Resource Results**
- Get Incident.Incident_ID and Resouce.Resource_ID and User.Username from session query.
- Run the **Create Request Task** task; create new resource request
    - From Request table insert new request where Request.Resource_ID is a composed of Incident.Incident_ID and Resource.Resource_ID.

Request.Requested_by=CurrentUser, Date_requested=Timestamp.Now, Requests.Request_Status = “Pending”
    - If Request.Resource.Owner=CurrentUser:
        - Update Requests.Request_Status to “DEPLOYED”
        - Run the **Update Resource Availability Task**
    - Else
        - Reject DB Update.

9. Cancel Request

Task Decomposition

Lock Types: Update to Requests Table
Number of Locks: 2 DB locks
Enabling Conditions: Enabled by a user login and clicking **Cancel** button
Frequency: A few times per session
Consistency: Critical, DB needs to be updated correctly.
Subtasks: None
Decomposition*: Not* Required

Abstract Code

- User clicked on ***Cancel*** button from either **Search Resource Results**
- Get Incident.Incident_ID and Resouce.Resource_ID and User.Username from session query.
- Run the **Delete Request Task** task; remove resource request
    - From Request table delete request where Request.ID is a composed of Incident.Incident_ID and Resource.Resource_ID. and Request.Requested_by=CurrentUser.

10. Reject Request

Task Decomposition

Lock Types: Update to Requests Table
Number of Locks: 2 DB locks
Enabling Conditions: Enabled by a user login and clicking **Reject** button
Frequency: A few times per session
Consistency: Critical, DB needs to be updated correctly.
Subtasks: None
Decomposition*: Not* Required

Abstract Code

- User clicked on ***Reject*** button from either **Search Resource Results**
- Get Incident.Incident_ID and Resouce.Resource_ID and User.Username from session.
- Run the **Delete Request Task** task; remove resource request
    - From Request table delete request where Request.ID is a composed of Incident.Incident_ID and Resource.Resource_ID, and Resource.Resource_owner=CurrentUser.

11. Return Resource

Task Decomposition

Lock Types: Update to Requests table, Update to Resources Table
Number of Locks: 2 DB locks
Enabling Conditions: Enabled by a user login and clicking **Return** button
Frequency: A few times per session
Consistency: Critical, DB needs to be updated correctly.
Subtasks: None
Decomposition*: Not* Required

Abstract Code

- User clicked on ***Return*** button from either **Search Resources or Resource Status**
- Get Incident.Incident_ID and Resouce.Resouce_ID and User.Username from session query.
- Run the **Update Resource Availability Task** task; update resource information
    - From Resources select resources that that was selected.
    - If Resource.Owner == CurrentUser
        - Update Resource.Status to “Available”
    - Else reject DB update
- Run the **Update Request Task** task; update request information
    - From Request table select Request where Request.Resource AND Request.Incident== CurrentResource AND Current Incident
    - If Request.Resource.Owner=CurrentUser:
        - Update Requests.Request_Status to “RETURNED”
    - Else
        - Reject DB Update.

12. View Report

Task Decomposition

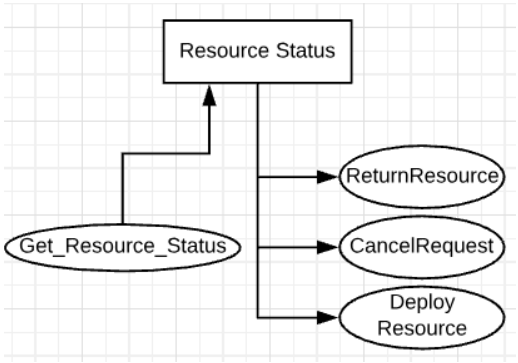
Lock Types: Read on Resources Table
Number of Locks: One DB Lock
Enabling Conditions: Enabled by a user login and clicking **Resource Report** button
Frequency: Many times per session
Consistency: Criticality not necessary
Subtasks: None
Decomposition*: Not* Required

Abstract Code

- User clicked on ***Resource Report*** button from **Main Menu**
- Run the **View Report** task; update resource information
    - From Resources select where Resource.Owner == CurrentUser
        - Count Resources based on primary ESF
        - Count not available where Resource.Status==“Not Available”
- Populate the **Resource Report** form with the requested information.
    - If the current user has no Resources, a 0 is displayed for the ESF.

13. Populate Resource Status

Task Decomposition



Lock Types: Read only on Incident table, read/write on Resources, write on Requests table
Number of Locks: Several schema constructs are needed
Enabling Conditions: Enabled by a user login and click on resource status from main menu
Frequency: Many times per session
Consistency: Critical, All incidents and statuses need to be shown correctly
Subtasks: Mother task is required to coordinate subtasks
Decomposition*:* Required

Abstract Code:

- User click on the Resource Status Screen from the **Main Menu**
- Get User.Username from the session query
- Get all resources owned by the user, all requests sent by the user and all requests received by the user through query
- Run the **GetResourceStatus** task
    - If Incident.Owner == User.Username and Request.Requested_by == User.Username and Request_status == “Confirmed”
        - then display **Return** button and the following information in the top section
            - Resource_ID
            - Resource Name
            - Incident Responding To
            - Owner Name
        - If **Return** button is clicked, run ***ReturnResource*** Task
    - Elsif Incident.Owner == User.Username and Request.Requested_by == User.Username and Request.Requested_Status == “Pending”
        - then display **Cancel** button and the following information in the middle section
            - Resource_ID
            - Resource_name
            - Incident_ID
            - Resource_Owner
        - If **Cancel** button is clicked, run ***CancelRequest*** Task

- Elsif Resource.Owner == User.Username and Request.Requested_by != User.Username and Request.Requested_Status == “Pending”
  - Then display both **Deploy** button and **Cancel** button and the following information in the bottom section
    - Resource_ID
    - Resource_Name
    - Related Incident_Name
    - Requested_By
  - If **Deploy** button is clicked, run Deploy Resource Task
  - If **Cancel** button is clicked, run Cancel Request Task