

Summary Phase 1 Report

Table of Contents

1. Data Types
2. Business Constraints
3. Markup Annotations
4. Task Decomposition with Abstract Code:
 - a. Login
 - b. Main Menu
 - c. Add Available Resource Form
 - d. Add Emergency Incidence
 - e. Search Resources
 - f. Generate Resource Report

Data Types

A. Users:

Attribute	Data Type	Constraint
<u>Username</u>	String / Char	PK, NOT NULL
Password	String / VarChar	NOT NULL
DisplayName	String / Char	NOT NULL

→ Cert_Member

Attribute	Data Type	Constraint
PhoneNumber	String / VarChar	NOT NULL

→ Resource_Provider

Attribute	Data Type	Constraint
Number	Char	NOT NULL
Street	String / VarChar	NOT NULL
ApartmentNumber	Char	NULL

City	String / VarChar	NOT NULL
State	String / Char	NOT NULL

→ Admin

Attribute	Data Type	Constraint
Email	String / VarChar	NOT NULL

B. Resources:

Attribute	Data Type	Constraint
<u>Resource_id</u>	String / Char	PK, NOT NULL
Resource_name	String / VarChar	NOT NULL
Description	String / VarChar	NOT NULL
Capabilities	String / VarChar	NULL
Distance	Decimal (4, 1)	NULL
Cost	Decimal (5, 2)	NOT NULL
Unit_id	String / Char	FK (Unit_id), NOT NULL
Primary_function	Integer	FK (Function_id), NOT NULL
Secondary_function	String / Char	COMPOSITE FK (Resource_id, Function_id), NOT NULL
Owner	String / VarChar	FK (Username), NOT NULL

→ Cost_unit

Attribute	Data Type	Constraint
<u>Unit_id</u>	Integer	PK, NOT NULL
Unit	String / VarChar	NOT NULL

→ Function

Attribute	Data Type	Constraint
<u>Function_id</u>	Integer	PK, NOT NULL
Description	String / VarChar	NOT NULL

→ Secondary_function

Attribute	Data Type	Constraint
<u>Function_id</u>	Integer	FK, NOT NULL
<u>Resource_id</u>	String / VarChar	FK, NOT NULL

C. Incidents:

Attribute	Data Type	Constraint
<u>Incident_id</u>	Int	PK, NOT NULL
Date	Date	NOT NULL
Description	String / VarChar	NOT NULL
Category	VarChar	FK (Category_id), NOT NULL

→ Category

Attribute	Data Type	Constraint
<u>Category_id</u>	Int	PK, NOT NULL
Type	String / VarChar	NOT NULL

Business Logic Constraints (These are our assumptions based off of the given business requirements)

- A resource is available if it is not currently being used to respond to an incident.
- New resources entered into the system are available by default.
- In no circumstances should the system allow a resource that is currently in use be deployed to respond to another incident.
- In no circumstances should the system allow a resource be deployed to respond to an incident if the resource cannot be delivered to the site of the incident in a reasonable time to assist.
- In no circumstances should the system allow a resource be deployed to respond to an incident if the resource cannot be realistically accessed due to scenarios; such as, the resource being damaged because of the incident, the owner not being present to give access to the resource, financial incapability, etc.
- A resource cannot be deployed to an incident that has already passed.
- A resource provider can only submit resources that they are realistically owners of, and have available to deploy at any time.
- A submitted resource must be in compliance with the law.
- The system does not account for SQL injections.
- User can only login from one machine at a time.

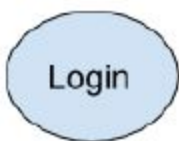
Markup Annotations

- **Bold Underline: Form / View.**
- **Bold Italics: Buttons**
- **Bold: Task.**
- Italics: Form Input fields / Column names in tabulated form.
- \$XYZ: Database field/column named 'XYZ'.

Task Decomposition with Abstract Code:

Login

Task Decomposition:



Abstract Code:

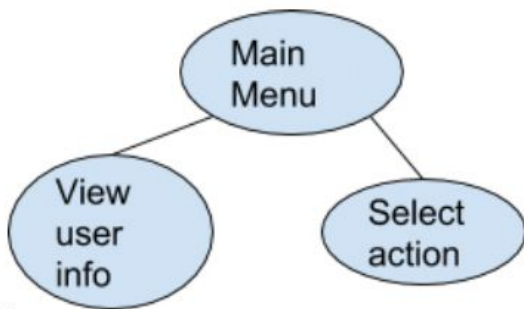
- User enters *Username* ('\$Username') and *Password* ('\$Password') input fields.
- If data validation is successful for both *Username* and *Password* input fields, then:
- When ***Login*** button is clicked:

```
SELECT COUNT(1) = 1 AS valid_login
FROM `user`
WHERE username = '$Username' AND password = '$Password';
```

- If *Username* exists but *Password* != '\$Password', or *Username* is not found:
 - *Username* and *Password* input fields are invalid. Go back to **Login** form and display error message.
- Else, log user in and go to **Main Menu** view.

Main Menu

Task decomposition:



Abstract Code:

- User successfully logged in from the **Login** form
- Run the **view user info** task by querying the user to display the user's '\$Name' and user's specific information:

```
SELECT u.disp_name, a.email, c.phone_number, r.street_number, r.street,
IFNULL( r.apartment_number, ' '), r.city, r.state, r.zip
FROM `user` u
LEFT JOIN `admin` a ON (u.username = a.username)
LEFT JOIN `cert_member` c ON (u.username = c.username)
LEFT JOIN `resource_provider` r ON (u.username = r.username)
WHERE u.username = '$Username';
```

- If user type = "admin", show '\$specific_info' (email).
- If user type = "cert member", show '\$specific_info' (phone number).
- If user type = "resource provider", show '\$specific_info' (address).

- Run the **select action** task to display “**Add Resource**”, “**Add Emergency Incident**”, “**Search Resources**”, “**Resource Status**”, and “**Resource Report**” links.

Upon clicking:

- **Add Resource** button - Jump to Add Resource form.
 - **Add Emergency Incident** button - Jump to Add Emergency Incident form.
 - **Search Resources** button - Jump to Search Resources form.
 - **Resource Status** button - Jump to Resource Status view.
 - **Resource Report** button - Jump to Resource Report view.
 - **Exit** or “**X**” button - Invalidate login session and go back to Login form.
- When **Exit** button is clicked on Main Menu view:
 - Exit the current form, invalidate login session, and take user back to the Login form.

New Resource Information

Task Decomposition:



Abstract Code:

- User selects **add resource** task from **select action** task.
- Display Add Resource form, and run the **add resource** task.
- Display *Owner* ('\$Owner') with the logged in user display name.
- User enters *Resource Name* ('\$Resource_name'), *Primary Function* ('\$Primary_function'), *Secondary Function* ('\$Secondary_function'), *Description* ('\$Description'), *Capabilities* ('\$Capability'), *Distance* ('\$Distance'), *Cost* ('\$Cost'), and *Unit* ('\$Unit') input fields, drop down menu, and list view:
 - *Primary Function* drop down menu:

```
SELECT CONCAT (function_id, " ", description) AS "Primary Function"
FROM `function`;
```

- Query (**Function**) table -> ('\$Function_id') and ('\$Description'), return and display *Primary Function* ('\$Function_id') ('\$Description').
 - If *Primary Function* drop down menu selected value is "value", then:
 - *Secondary Function* list view does not include "value"
- If data validation is not successful for all fields, then:
 - Display red error message instructing user to input correct format next to input field or drop down menu.
- Else if data validation is successful for all fields, then:
- When **Save** button is clicked:
 - If *Resource Name*, *Primary Function*, *Cost*, or *Unit* is empty or not filled out:
 - Go back to **Add Resource** form, and display red error message next to required empty fields (error message: " * ").
 - Else, uniquely generate ('\$Resource_id')


```
INSERT INTO `resource` (`username`, `primary_function_id`,
`resource_name`, `description`, `capabilities`, `distance`, `cost`, `unit_id`)
VALUES
('$Username', '$primary_function_id', 'resource1', NULL, NULL, NULL,
'500.00', '$Unit_id'),
('$Username', $primary_function_id, 'pasadena city college', NULL, NULL,
NULL, '900.00', '$Unit_id');
```
 - Insert user-entered *values*.
- User adds new resource.
 - Run **display added resource** task: display user-entered fields as read-only.
- When "+" button is clicked:
 - Clear user entered values from input fields and drop down menus.
- When **Cancel** button is clicked:
 - Exit the current form, and take user back to the **Main Menu** form.

New Incident Information

Task Decomposition:



Abstract Code:

- User selects **add emergency incident** task from **select action** task.
- Display **Add Emergency Incident** form, and run the **add incident** task.
- User enters *Category* ('\$Category'), *Date* ('\$date'), *Description* ('\$Description') input fields and drop down menu.

- *Category* drop down menu:

```
SELECT category_id, description
FROM `incident`;
```

- Query (**Category**) table -> ('Category_id') and ('\$type'), return and display *Category* ('\$type').

- If *Date* format is not "mm/dd/yyyy", then:
 - Display red error message instructing user to input correct format: **mm/dd/yyyy**
- Else if data validation is successful for all fields, then:
- When **Save** button is clicked:
 - If *Category*, *Date*, and *Description* is empty or not filled out:
 - Go back to **Add Emergency Incident** form, and display error message next to required empty fields (error message: " * ").
 - Else, uniquely generate ('\$Incident_id')

```
INSERT INTO `incident` (`username`, `category_id`, `date`, `description`)
VALUES
('$Username', '$Category_id', '2019-05-02 00:00:00', 'Active shooter'),
('$Username', '$Category_id', '2019-05-08 00:00:00', 'Earthquake drill');
```

- Insert user-entered *values*.

- User adds new incident.
 - Run **display added incident** task: display user-entered fields as read-only.
- When "+" button is clicked:
 - Clear user entered values from input fields and drop down menus.
- When **Cancel** button is clicked:

- Exit the current form, and take user back to the **Main Menu** form.

Search Resources

Task Decomposition:



Abstract Code:

- User selects **search resources** task from **select action** task.
- Display **Search Resources** form, and run the **search results** task.
- User enters *Category* ('\$Category'), *Date* ('\$Date'), *Incident* ('\$Incident'), and *Distance* ('\$Distance') input fields and drop down menu.
 - *Primary Function* drop down menu:

```
SELECT CONCAT (function_id, " ", description) AS "Primary Function"
FROM `function`;
```

- Query (**Function**) table -> ('\$Function_id') and ('\$Description').

- If data validation is not successful for all fields, then:
 - Display red error message instructing user to input correct format next to input field or drop down menu.
- Else if data validation is successful for all fields, then:
- When **Search** button is clicked, query the (**Resource**) table to run a comparison:

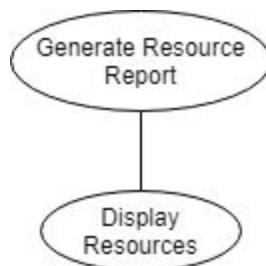
```
SELECT u.username, r.resource_id, r.resource_name, r.cost, c.unit, r.distance
FROM `user` u
LEFT JOIN `resource` r ON (u.username = r.username)
LEFT JOIN `cost_unit` c ON (r.unit_id = c.unit_id)
WHERE r.resource_name LIKE '%" $Keyword "%
      OR r.description LIKE '%" $Keyword "%
      OR r.capabilities LIKE '%" $Keyword "%;
```

- If *Keyword* == ('\$Resource_name') or ('\$Description') or ('\$Capability'), then:

- Return ('\$Resource_id'), ('\$Resource_name'), ('\$Owner'), ('\$Cost'), ('\$Unit'), ('\$Distance'). Sort by resource's distance and display.
 - Else if "value" selected in *Primary Function*, then:
 - Return ('\$Resource_id'), ('\$Resource_name'), ('\$Owner'), ('\$Cost'), ('\$Unit'), ('\$Distance') where ('\$Primary_function') == "value"
 - Else if "value" input in *Distance*, then:
 - Return ('\$Resource_id'), ('\$Resource_name'), ('\$Owner'), ('\$Cost'), ('\$Unit'), ('\$Distance') where ('\$Distance') to 0.1 precision <= "value".
- ```
SELECT u.username, r.resource_id, r.resource_name, r.cost, c.unit, r.distance
FROM `user` u
LEFT JOIN `resource` r ON (u.username = r.username)
LEFT JOIN `cost_unit` c ON (r.unit_id = c.unit_id);
```
- Else if no data input from user, then:
      - Return all resources.
  - Search returns results matching the user's search criteria.
    - Run **display search resources** task: display *Resource ID* ('\$Resource\_id'), *Resource Name* ('\$Resource\_name'), *Owner* ('\$Owner'), *Cost/Unit* ('\$Cost') ('\$Unit'), *Distance* ('\$Distance') in table list fashion.
  - When "+" button is clicked:
    - Clear user entered values from input fields, drop down menus, and search results.
  - When **Cancel** button is clicked:
    - Exit the current form, and take user back to the **Main Menu** form.
  - User searches for resources using Keyword (optional).

## Generate Resource Report

Task Decomposition:



Abstract Code:

- User selects **generate resource report** task from **select action** task.
- Displays the user's **Resource Report** form according to each resource's primary function sorted by their Primary Function #.

```
SELECT f.function_id AS "Function ID", f.description AS "Function",
COUNT(r.primary_function_id) AS "Total Resources"
FROM `function` f
LEFT JOIN `resource` r ON (f.function_id = r.primary_function_id)
WHERE username = "aSmith"
GROUP BY f.function_id, f.description WITH ROLLUP;
--ORDER BY f.function_id
```

- Displays title "Resource Report" on top of the form.
- The table format of the report consists of:
  - *Primary Function # (PF#)*: pulled from *PF* in the database. *PF#* starts from 1 and goes up in ascending order.
  - *Primary Function (PF)*: a brief description of the *PF* in the database. Description of the *PF* is originally entered on **Add Resource** form which serves as a mandatory field.
  - *Total Resources*: row-wide aggregation by grouping all the resources belonging to each *PF* in the system for the user.
  - *Total*: column-wide aggregation by grouping all the resources across all *PFs* in the system.