**Grapes: NLP Web Craft**

**Design Document**



#### Session: 2021-25

**Submitted by:**

Naqeeb Ahmed 2021-CS-616

Munib Ur Rehman 2021-CS-605

Ahsan Bashir 2021-CS-628

**Supervised by: Dr Zeeshan Ramzan**

Department of Computer Science, New Campus **University of Engineering and TechnologyLahore, Pakistan**

# Table of Contents

[List of Figures 3](#_bookmark0)

1. [Software Requirement Specification 4](#_bookmark1)
   1. [Functional Requirement 4](#_bookmark2)
   2. [Non-functional Requirement 5](#_bookmark3)
2. [Design specification 7](#_bookmark4)
   1. [System Behavioral Design 7](#_bookmark5)
      1. [Use case Diagram 7](#_bookmark6)
      2. [Activity Diagram 17](#_bookmark8)
      3. [State Diagram 29](#_bookmark21)
      4. [Sequence Diagram 40](#_bookmark33)
      5. [Collaboration Diagram 74](#_bookmark36)
   2. [System Structure Design 110](#_bookmark39)
      1. [Class Diagram 110](#_bookmark40)
      2. [Component Diagram 111](#_bookmark42)
      3. [Deployment Diagram 112](#_bookmark44)
   3. [User Interface Design 113](#_bookmark46)
      1. [Wireframes 113](#_bookmark47)
   4. [Database Design 113](#_bookmark48)
      1. [ER Diagram 113](#_bookmark49)

# List of Figures

[Figure 1: Use Case Diagram 7](#_bookmark7)

[Figure 2: Sign up 17](#_bookmark9)

[Figure 3: Sign In 18](#_bookmark10)

[Figure 4: Manage Profile 19](#_bookmark11)

[Figure 5: Edit User 20](#_bookmark12)

[Figure 6: Manage Template 21](#_bookmark13)

[Figure 7: View Prompt History 22](#_bookmark14)

[Figure 8: Sign Up 23](#_bookmark15)

[Figure 9: Sign In 24](#_bookmark16)

[Figure 10: Create Website 25](#_bookmark17)

[Figure 11: GrapeJs Editing 26](#_bookmark18)

[Figure 12: View Features 27](#_bookmark19)

[Figure 13: Contact Us 28](#_bookmark20)

[Figure 14: Sign Up 29](#_bookmark22)

[Figure 15: Sign In 30](#_bookmark23)

[Figure 16: Manage Profile 31](#_bookmark24)

[Figure 17: manage Template 32](#_bookmark25)

[Figure 18: View Prompt History 33](#_bookmark26)

[Figure 19: Sign Up 34](#_bookmark27)

[Figure 20: Sign In 35](#_bookmark28)

[Figure 21: Create Website 36](#_bookmark29)

[Figure 22: GrapeJs Editing 37](#_bookmark30)

[Figure 23: View Features 38](#_bookmark31)

[Figure 24: Contact Us 39](#_bookmark32)

[Figure 25: Admin End 40](#_bookmark34)

[Figure 26: User End 44](#_bookmark35)

[Figure 27: Admin End Collaboration 74](#_bookmark37)

[Figure 28: User End Collaboration 78](#_bookmark38)

[Figure 29: Class Diagram 110](#_bookmark41)

[Figure 30: Component Diagram 111](#_bookmark43)

[Figure 31: Deployment Diagram 112](#_bookmark45)

[Figure 32: ER Diagram 113](#_bookmark50)

Chapter 1

# Software Requirement Specification

### Functional Requirement

In developing a robust NLP Web Craft system, a comprehensive set of functional requirements has been defined. The functional requirements are grouped into four categories: Business requirements, administrative requirements, user requirements, and service requirements. Each category is assigned a unique identifier, such as B for business, A for administrative, U for user, and S for service requirements. The requirements are then numbered within each category, starting with 01.

The following list describes the main functions the system will perform:

* FR-B-01: The system shall provide web development business services aimed at individuals and small businesses.
* FR-B-02: The system shall target businesses looking for quick and efficient website solutions.
* FR-B-03: The system shall cater to content creators seeking to establish their online presence through customizable websites.
* FR-B-04: The system shall provide value-added services to enhance user experience, such as SEO optimization.
* FR-B-05: The system shall create business opportunities through affiliate marketing and partnerships with template providers.
* FR-B-06: The system shall offer subscription-based services for users seeking premium templates and advanced features.
* FR-A-01: The system shall enable and disable admin access for managing user accounts and roles.
* FR-A-02: The system shall allow admins to manage user roles and permissions, ensuring proper access control for different services.
* FR-A-03: The system shall share login IDs and new passwords with admins as needed for account recovery.
* FR-A-04: The system shall enable admins to view user activity logs to monitor interactions with the chatbot and website creation services.
* FR-U-01: The system shall provide a chatbot that interacts with users, asking questions about the type of website they want (e-commerce, blog, portfolio) to facilitate web development services.
* FR-U-02: The chatbot shall understand the user’s responses and create a website plan based on

their input, guiding them in content management and design services.

* FR-U-03: The system shall offer ready-to-use templates for different types of websites, including e-commerce, blogs, and portfolios, enhancing the template design services.
* FR-U-04: Users shall have the ability to customize design elements such as headers, footers, and color schemes for their websites, utilizing user experience (UX) services.
* FR-U-05: After editing, users shall be able to download their websites as a ZIP file containing all the HTML, CSS, and JS files, providing interoperability and export functionality.
* FR-U-06: The downloaded website files shall be fully editable by users for future modifications, ensuring ongoing user support and technical assistance.
* FR-S-01: The system shall provide a chatbot service to gather user requirements and preferences for website creation.
* FR-S-02: The system shall offer template design services, allowing users to select from various pre-built website templates.
* FR-S-03: The system shall facilitate customization services, enabling users to modify design elements of their websites.
* FR-S-04: The system shall support content management services, allowing users to organize and manage their website content effectively.
* FR-S-05: The system shall enable downloading services for users to export their website files as editable ZIP files.
* FR-S-06: The system shall provide ongoing technical support services for users to assist with any issues related to their websites

### Non-functional Requirement

In developing the NLP Web Craft system, we aim to create a robust and user-friendly platform that empowers users to effortlessly build and customize their websites using advanced natural language

processing technologies. The project is designed to simplify the web development process for individuals and businesses, making it accessible even to those without technical expertise. By integrating a chatbot interface and ready-to-use templates, users can quickly generate website plans and customize design elements, ultimately enhancing their online presence.

To ensure the system meets the needs of diverse users, we've defined a comprehensive set of non- functional requirements. These requirements focus on various aspects of the system's performance,

usability, reliability, and scalability. Each requirement has been categorized and assigned a unique identifier for easy reference.

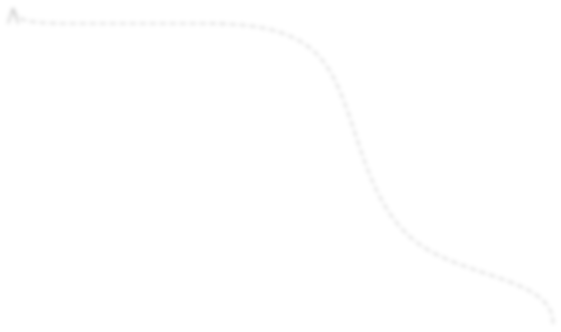
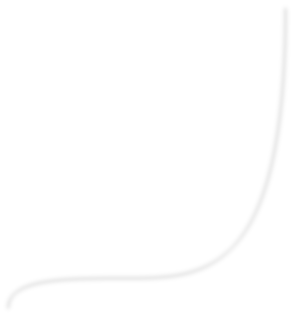
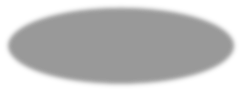
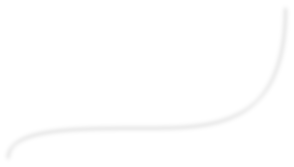
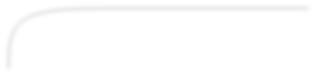
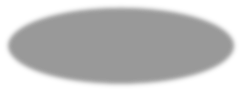
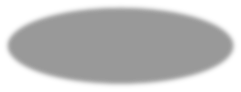
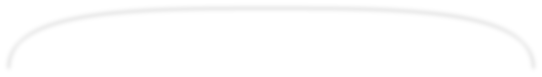
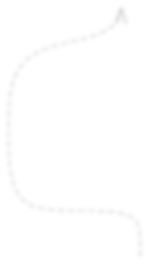
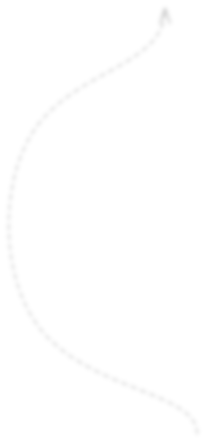
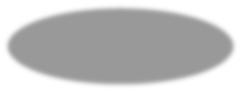
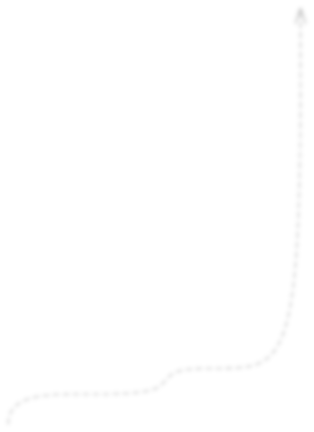
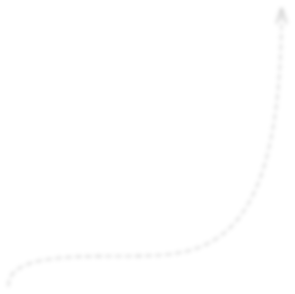
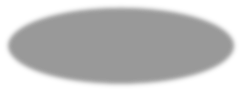
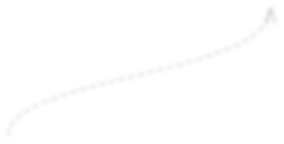
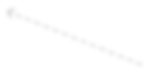
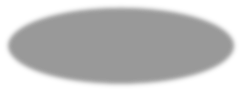
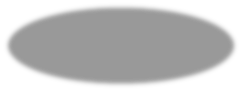
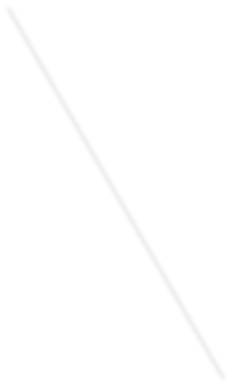
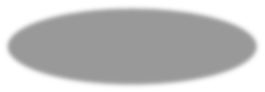
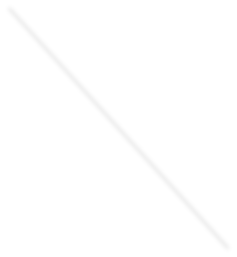
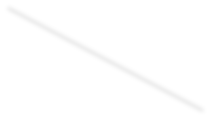
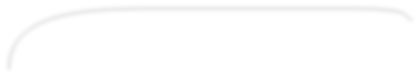
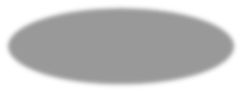
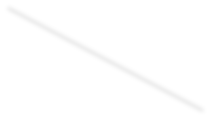
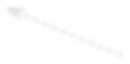
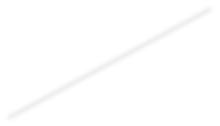
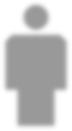
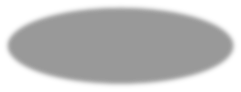
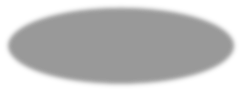
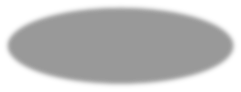
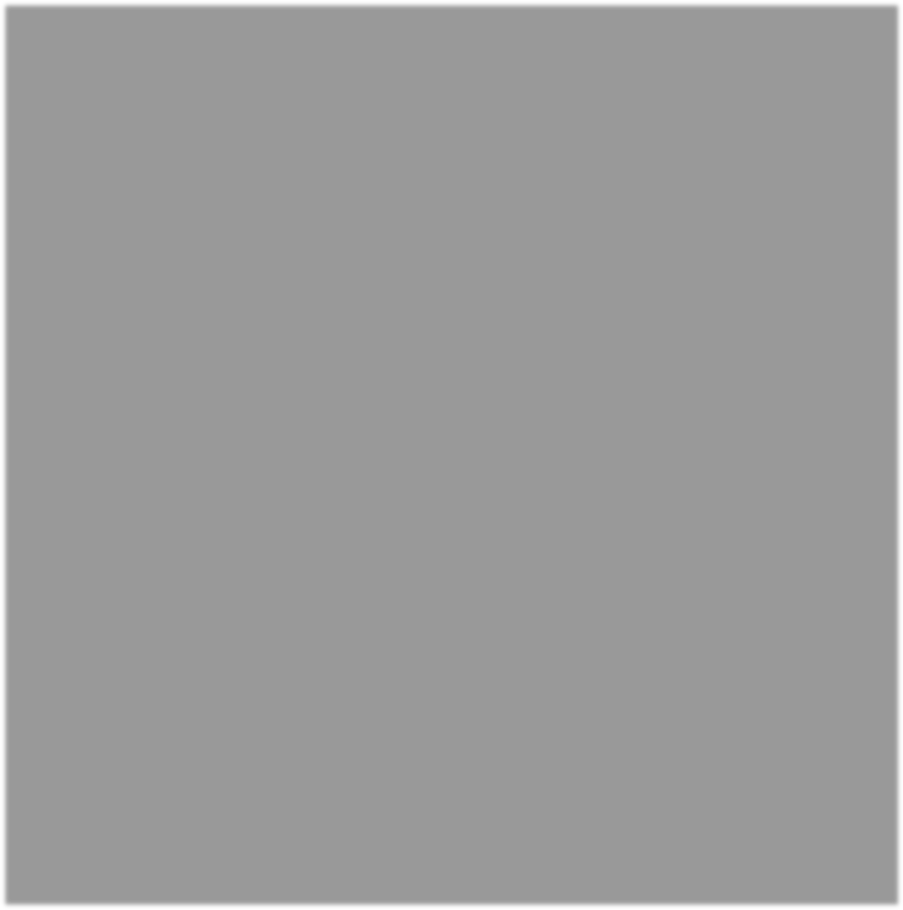
1. NFR-U-01: The interface must be simple and intuitive for non-technical users.
2. NFR-U-02: Users must be able to preview their websites in real-time while editing.
3. NFR-U-03: The system must be mobile-friendly for easy use across all devices.
4. NFR-U-04: Users must have the ability to undo/redo changes during website editing.
5. NFR-R-01: The system must handle temporary internet disruptions gracefully.
6. NFR-R-02: The system must gradually degrade without failure under heavy load.
7. NFR-S-01: The system must accommodate increases in data storage without performance degradation.
8. NFR-S-02: The system must support new features without requiring significant rewrites.
9. NFR-S-03: Specific components of the system must scale independently (e.g., the chatbot or API).
10. NFR-S-04: New templates and design elements must be easy to add over time.
11. NFR-P-01: The system must respond to user requests within 2 seconds.
12. NFR-P-02: API data searches must complete in 1 second or less.
13. NFR-P-03: Templates and previews must be updated in real-time during editing.
14. NFR-P-04: The system must function smoothly on low-bandwidth connections.

Chapter 2

# Design specification

## System Behavioral Design

#### Use case Diagram



Addition Info

Enter Credential

<<extend>>

<<include>>

<<include>>

Sign Up

depends

Create website via

Prompt

Sign In

depends

depends

depends

User

Add Template

Admin

depends

Choose Template

depends

depends

depends

Delete Template

depends

Edit Website(GrapeJS)

depends

<<extend>>

Prompt History

Preview Website

Download Website

*Figure 1: Use Case Diagram*

**Use Case Name:** User Sign-Up for NLP Web Craft

**Primary Actor:** User

**Trigger:** When the **"Create Account"** button is pressed to register a new user.

**Goal:** Successfully create a user account to use the chatbot and website-building tools.

**Precondition:** The user is not already registered in the system.

**Postcondition:** User successfully creates an account and gains access to the system features.

##### Basic Flow:

1. User opens the NLP Web Craft platform.
2. User navigates to the "Sign-Up" section.
3. System prompts for user details (name, email, password).
4. User enters the required information.
5. System validates the information (e.g., email, password strength).
6. If valid, the system creates a new user account.

##### Alternate Flow:

N/A

##### Quality Requirements:

1. The sign-up process should be secure (password encryption).
2. The system should provide a better user experience.

**Exception:** If there is a technical issue (e.g., server/database failure), the system should display an error message and suggest retrying or contacting support.

**Use Case Name:** Admin Sign-Up

**Primary Actor:** Admin

**Trigger:** When the **"Sign Up"** button is pressed to create a new admin account.

**Goal:** Successfully create a new admin account for accessing and managing the system.

**Precondition:** The administrator is not already registered in the system.

**Postcondition:** The administrator successfully creates an admin account and gains access to the administrative features of the application.

##### Basic Flow:

1. The administrator opens the car showroom application.
2. The administrator navigates to the "Admin Sign-Up" section.
3. The system prompts the administrator to provide necessary information (username, email, password, and contact details).
4. The administrator enters the required information.
5. The system validates the entered information (e.g., checks for valid email format, password strength).
6. If the information is valid, the system creates a new admin account.

##### Alternate Flow:

N/A

##### Quality Requirements:

1. The admin sign-up process should be secure, using encryption for passwords.
2. The system should handle the creation of admin accounts efficiently and securely.

**Exception:** If there are technical issues during the admin sign-up process (e.g., server error, database connection failure), the system should display a friendly error message and provide guidance on what the administrator should do next.

**Use Case Name:** User Sign-In for NLP Web Craft

**Primary Actor:** User (Website Builder, Admin)

**Trigger:** When the **"Log In"** button is pressed to access the User dashboard.

**Goal:** Successfully create a user account to use the chatbot and website-building tools.

**Precondition:** The user is not already registered in the system.

**Postcondition:** User successfully creates an account and gains access to the system features.

##### Basic Flow:

1. User opens the NLP Web Craft platform.
2. User navigates to the "Sign-Up" section.
3. System prompts for user details (name, email, password).
4. User enters the required information.
5. System validates the information (e.g., email, password strength).
6. If valid, the system creates a new user account.

##### Alternate Flow:

N/A

##### Quality Requirements:

1. The sign-up process should be secure (password encryption).
2. The system should provide a seamless user experience.
3. Adequate logging should be implemented to track account creation for audit purposes.

**Exception:** If there is a technical issue (e.g., server/database failure), the system should display an error message and suggest retrying or contacting support.

##### Admin Sign-In

**Use Case Name:** Admin Sign-In

**Primary Actor:** Admin

**Trigger:** When the **"Log In"** button is pressed to access the admin dashboard.

**Goal:** Successfully log in to the admin account to manage the application.

**Precondition:** The administrator has a registered admin account.

**Postcondition:** The administrator is logged into the system and can access administrative features.

##### Basic Flow:

1. The administrator opens the application.
2. The administrator navigates to the "Admin Sign-In" section.
3. The system prompts the administrator to enter their username and password.
4. The administrator enters the required login credentials.
5. The system validates the credentials against the database.
6. If the credentials are valid, the system grants access and redirects the administrator to the admin dashboard.
7. The administrator can now manage the application.

##### Alternate Flow: N/A

**Quality Requirements:**

1. The sign-in process should be secure and protect against unauthorized access.
2. The system should provide feedback for failed login attempts.

**Exception:** If there are technical issues during the sign-in process (e.g., server error, database connection failure), the system should display a friendly error message and provide guidance on what the administrator should do next.

##### Create Website Via Prompt

**Use Case Name:** Create Website Via Prompt

**Primary Actor:** User

**Trigger:** When the **"Create Website"** button is pressed after selecting a type and writing a prompt.

**Goal:** Successfully create a website based on the user’s prompt and selected type.

**Precondition:** The user is logged in to the system.

**Postcondition:** The website is created based on the user’s specifications and is accessible for further

editing.

##### Basic Flow:

1. The user opens the NLP Web Craft platform.
2. The user navigates to the "Create Website" section.
3. The system presents three options for website types: E-commerce, Blog, and Portfolio.
4. The user selects one of the options.
5. The system prompts the user to enter a description or prompt for the website.
6. The user writes and submits the prompt.

##### Alternate Flow:

N/A

##### Quality Requirements:

1. The website creation process should be efficient and responsive.
2. The system should handle user prompts securely and protect against malicious input.

**Exception:** If there are technical issues during the website creation process (e.g., server error, prompt processing failure), the system should display a friendly error message and provide guidance on what the user should do next.

##### Choose Template

**Use Case Name:** Choose Template

**Primary Actor:** User (Website Builder)

**Trigger:** When the **"Select Template"** button is pressed to choose a template for the website.

**Goal:** Successfully choose a template that fits the user’s website design preferences.

**Precondition:** The user has created a website via prompt.

**Postcondition:** The selected template is applied to the website for further customization.

##### Basic Flow:

1. The user navigates to the "Choose Template" section.
2. The system displays a selection of available templates for E-commerce, Blog, and Portfolio websites.
3. The user selects a template that best suits their best.
4. The system applies the selected template to the website.
5. The user can now proceed to customize the website further with the chosen template.

##### Alternate Flow:

N/A

##### Quality Requirements:

1. The template selection process should be user-friendly and visually appealing.
2. The system should load templates efficiently to enhance user experience.
3. Adequate logging should be implemented to track template selections for auditing purposes.

**Exception:** If there are technical issues during the template selection process (e.g., server error, database failure), the system should display a friendly error message and provide guidance on what the user should do next.

##### Edit Website (GrapeJS)

**Use Case Name:** Edit Website Using GrapeJS

**Primary Actor:** User

**Trigger:** When the **"Edit"** button is pressed to open the GrapeJs editor for customization.

**Goal:** Successfully modify and customize the website using GrapeJS editor.

**Precondition:** The user has selected a template for their website.

**Postcondition:** The changes made to the website are saved, and the user can preview the updated website.

##### Basic Flow:

1. The user navigates to the "Edit Website" section using GrapeJS.
2. The system loads the selected template into the GrapeJS editor.
3. The user interacts with the GrapeJS interface to modify elements (text, images, layout).
4. The user can add, remove, or rearrange components within the website.
5. The user previews the changes in real-time using the GrapeJS preview feature.
6. The user saves the changes once satisfied with the edits.
7. The system confirms that the changes have been successfully saved.

##### Alternate Flow:

N/A

##### Quality Requirements:

1. The GrapeJS editor should be intuitive and responsive, providing a seamless editing experience.
2. The system should ensure all changes are saved securely and efficiently.

**Exception:** If there are technical issues during the editing process (e.g., server error, failure to save changes), the system should display a friendly error message and provide guidance on what the user should do next.

##### Download Website

**Use Case Name:** Download Website

**Primary Actor:** User (Website Builder)

**Trigger:** When the **"Download"** button is pressed to download the completed website files.

**Goal:** Successfully download the website files in a zip format.

**Precondition:** The user has edited and finalized their website using the GrapeJS editor.

**Postcondition:** The website files are downloaded to the user's device.

##### Basic Flow:

1. The user navigates to the "Download Website" section.
2. The system presents an option to download the completed website.
3. The user clicks the "Download" button.
4. The system compiles the website files (HTML, CSS, JavaScript, assets) into a zip file.
5. The system prompts the user to save the zip file to their device.
6. The system confirms that the download is complete.

##### Alternate Flow:

N/A

##### Quality Requirements:

1. The download process should be efficient and secure.
2. The system should ensure that all necessary files are included in the zip package.

**Exception:** If there are technical issues during the download process (e.g., server error, failure to compile files), the system should display a friendly error message and provide guidance on what the user should do next.

##### Add Template

**Use Case Name:** Add Template

**Primary Actor:** Admin

**Trigger:** When the **"Add Template"** button is pressed to upload a new template.

**Goal:** Successfully add a new template to the system for user selection.

**Precondition:** The admin is logged into the system with appropriate privileges. **Postcondition:** The new template is available for users to select when creating their websites. **Basic Flow:**

1. The admin opens the NLP Web Craft platform.
2. The admin navigates to the "Add Template" section.
3. The system prompts the admin to upload the template files (HTML, CSS, JavaScript, assets).
4. The admin uploads the necessary files and enters relevant details (template name, description, category).
5. The system validates the uploaded files and details.

##### Alternate Flow:

N/A

##### Quality Requirements:

1. The template addition process should be secure and validate file types to prevent malicious uploads.
2. The system should handle template uploads efficiently and provide feedback on upload status.
3. Adequate logging should be implemented to track template additions for audit purposes.

**Exception:** If there are technical issues during the template addition process (e.g., server error, database failure), the system should display a friendly error message and provide guidance on what the admin should do next.

##### Delete Template

**Use Case Name:** Delete Template

**Primary Actor:** Admin

**Trigger:** When the **"Delete"** button is pressed to remove an existing template from the system.

**Goal:** Successfully delete a template from the system.

**Precondition:** The admin is logged into the system with appropriate privileges.

**Postcondition:** The selected template is removed from the system and is no longer available for user selection.

##### Basic Flow:

1. The admin navigates to the "Manage Templates" section.
2. The system displays a list of existing templates.
3. The admin selects the template they wish to delete.
4. The system prompts the admin to confirm the deletion.
5. The system confirms that the template has been successfully deleted.

##### Alternate Flow:

N/A

##### Quality Requirements:

1. The deletion process should be secure, preventing unauthorized access to template management features.

**Exception:** If there are technical issues during the deletion process (e.g., server error, database failure), the system should display a friendly error message and provide guidance on what the admin should do next.

##### Prompt History for Admin

**Use Case Name:** View Prompt History

**Primary Actor:** Admin (Car Showroom Administrator)

**Trigger:** When the **"View Prompt History"** button is pressed to access the list of previous prompts.

**Goal:** Successfully access and review the prompt history to monitor user interactions.

**Precondition:** The admin is logged into the system with appropriate privileges. **Postcondition:** The admin can view the details of previous prompts submitted by users. **Basic Flow:**

1. The admin opens the NLP Web Craft platform.
2. The admin navigates to the "Prompt History" section.
3. The system retrieves and displays a list of previously submitted prompts, including details such as user ID, prompt content, timestamp, and status.

##### Alternate Flow:

N/A

##### Quality Requirements:

1. The prompt history retrieval process should be efficient and fast to enhance the admin’s ability

to monitor user interactions.

1. The system should ensure that all user data is displayed securely and without exposing sensitive information.

**Exception:** If there are technical issues during the retrieval of prompt history (e.g., server error, database failure), the system should display a friendly error message and provide guidance on what the admin should do next.

## Activity Diagram

* + - 1. *Admin End*

Sign Up:

Start

[If user Already exist/

Invalid Credentials]

Enter credentials

Check Credentials

[Proceed]

Save user detail

Sign up Successfull

End

*Figure 2: Sign up*

Sign In:

Start

[Incorrect]

Enter username

Check user name

[Correct]

[Incorrect]

Enter password

verify password

[Correct]

Login Successful

End

*Figure 3: Sign In*

##### Manage Profile

Start

Manage profile

list of users

View all users

Want to delete

[Yes]

want to edit

[Yes]

wants changing

Delete User

Edit User

[If no]

Save Changes

Save Changes

User list

End

*Figure 4: Manage Profile*

##### Edit user:

Start

Edit Profile

Retrieve Profile

[Not Validate]

update Profile

Check Credentials

[If Validate]

Profile Updated

End

*Figure 5: Edit User*

##### Manage Templates:

Start

Available Templates

View Templates

Want to Edit

[Yes]

Want to delete

[Yes]

wants to manage

[No]

End

Edit Template with

GrapeJs

Delete Template

Save Changes

Save Changes

Template List

*Figure 6: Manage Template*

View Prompt history:

Start

View prompt

History

Retrieve Prompts

From Database

[If yes]

Want details

Display Info

[If No]

list of Prompts

Viewed List of

Prompts

End

*Figure 7: View Prompt History*

* + - 1. *User End*

Sign Up:

Start

[If user Already exist/

Invalid Credentials]

Enter credentials

Check Credentials

[Proceed]

Save user detail

Sign up Successfull

#### End

*Figure 8: Sign Up*

Sign In:

Start

[Incorrect]

Enter username

Check user name

[Correct]

[Incorrect]

Enter password

verify password

[Correct]

Login Successful

End

*Figure 9: Sign In*

Create website:

Start

Enter Prompt

analyze of promt

API Analyze promp

Structure and

template are

created

initialized website

code

#### End

*Figure 10: Create Website*

##### GrapeJs Editing:

Start

Initialized Code

GrapeJs Editor

Wants to change code

[If yes]

[If No]

Code edited with

GrapeJs

[Updated code]

Final Code

Download Code

*Figure 11: GrapeJs Editing*

##### View Features:

Start

View Features

Display Features

Wants the detail

[If Yes]

[If No]

Detailed Features

Viewed Features

End

*Figure 12: View Features*

Contact Us:

Start

[User select Contact us]

[Incorrect]

Enter Credentials

Check credentials

[Correct]

Enter Message

Save to database

Successfully

Submitted

#### End

*Figure 13: Contact Us*

#### State Diagram

* + - 1. *Admin End*

Sign Up:

Start

**Credentials**

[User Already Exist]

do/ User Enter

Credentials

Check Credentials

End

[Correct]

**Register User**

do/ User added to Data

Base

**Sign Up Successful**

do/ Display The Homepage

End

*Figure 14: Sign Up*

Sign In:

Start

**Input Details**

do/ Enter Username

[Username Not

Found]

Check Credentials

[Correct]

**Input password**

entry/ Username entered

do/ Enter Password

[Incorrect

Password]

Check Password

End

[Correct]

**Login**

Do/ Display Homepage

exit/ login successfully

#### End

*Figure 15: Sign In*

##### Manage Profile:

start

**Viewing Profile**

entry / fetch current profile

do/ display information

[edit]

[delete]

**Editing Profile**

entry/ make fileds editable do / admin edit profile information

Choose what to perform

**Delete profile**

do / admin deletes profile exit / admin deleted

End

**Save changes**

entry / confirm changes have

been saved

Do/ display confirmation message exit/profile updated

**Save Changes**

do/system saves the changes

exit / saved successfully

End End

*Figure 16: Manage Profile*

##### Manage Template:



Start

**Viewing Templates**

entry/vload available templates

do/ admin browses templates

Select Action

**Edit Template**

**Add Template**

**Delete Template**

entry/ select template

to edit

do/ admin modifies template

exit/ save template changes

entry/ enable input for

template details

do/ enter the new template data

entry/ select template to

delete

do/ admin deletes template

exit/ template deleted successfully

**Save template**

do/ system saves the template in database

exit/ successfully saved

End

End

End

*Figure 17: manage Template*

##### View prompt history:

Start

**Prompt history**

entry/ fetch data of user

prompts

do/ display a list of user

prompts

Detail?

End

[Yes]

**Prompt Details**

entry/ select the prompt for detail do/ display detail information

of prompt

exit/ return to prompt history

End

*Figure 18: View Prompt History*

* + - 1. *User End*

Sign Up:

Start

**Credentials**

[User Already Exist]

do/ User Enter

Credentials

Check Credentials

End

[Correct]

**Register User**

do/ User added to Data

Base

**Sign Up Successful**

do/ Display The Homepage

End

*Figure 19: Sign Up*

Sign In:

Start

**Input Details**

do/ Enter Username

[Username Not

Found]

Check Credentials

[Correct]

**Input password**

entry/ Username entered

do/ Enter Password

[Incorrect

Password]

Check Password

End

[Correct]

**Login**

Do/ Display Homepage

exit/ login successfully

#### End

*Figure 20: Sign In*

**Create Website:**

Start

**No input**

Do/ user wants to enter

Prompt

**prompt**

entry /Display Prompt screen

do/User Enter a prompt

**Analyze prompt**

exit/ Starts API call

do/API analyze the prompt data

**Selecting Template**

entry/ Display Template options do/ User select the template from template list

**Website Generated**

Do/ generates website structure

exit/ save generated website

End

*Figure 21: Create Website*

**GrapeJs Editing:**

Start

[If No]

User Wants to edit

[Yes]

**GrapeJs Editor**

entry/Initialize GrapeJs

Editor

do/load website into editor

**Editing Website**

do/ User edit the website

exit / save changes

**Download Code**

do/ User download the code exit/ code downloaded into Zip file

#### End

*Figure 22: GrapeJs Editing*

##### View Features:

Start

**Loading Features**

exit/ initialize feature do/ fetch feature data

**View Feature**

entry/ Show available features do/ Allow user to browse through Features

Want Detail

End

**View Detail**

entry /Show Detailed Information

do / Provide user all Information

End

*Figure 23: View Features*

##### Contact Us:

[Valid]

Start

**Contact form**

entry / initialize contact form

do/ load fields

**Fill Form**

entry / display contact form

do/ User enter credentials

[Invalid Email]

Check Email

End

**Enter Message**

do / Enter mesage or review

**Submit**

do/ user submit form exit / form submitted successfully

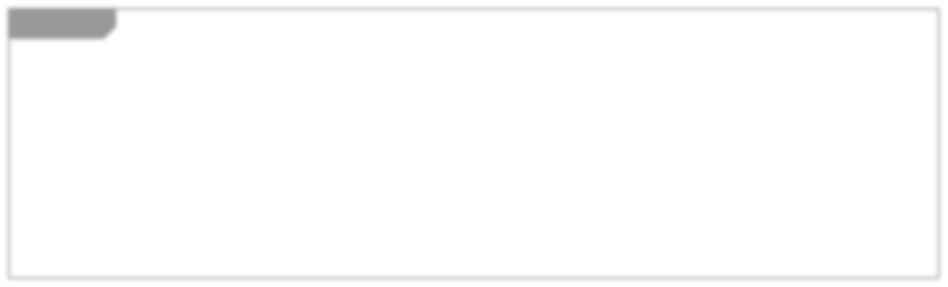
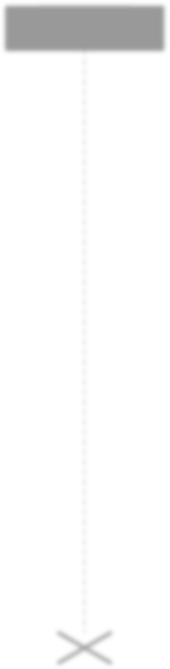
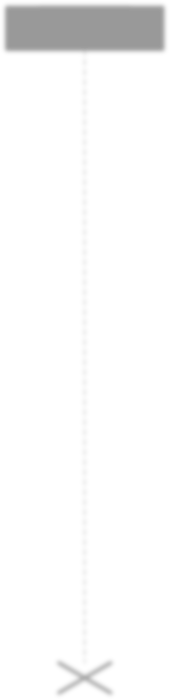
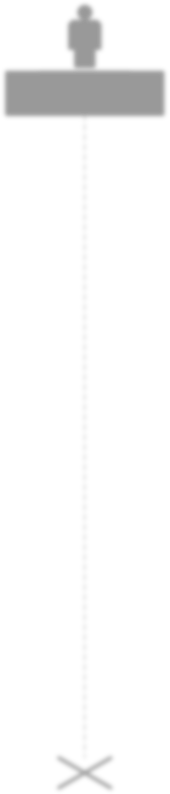
#### End

*Figure 24: Contact Us*

#### Sequence Diagram

* + - 1. *Admin End*

##### Sign Up:



**a1:admin**

**s1:Sign Up**

**b1:Database**

Start

Enter Name

store Name

Enter UserName

Verify all credential

store UserName

Enter Password

store Password

Success

**alt**

Suceess

Sign Up Successfully

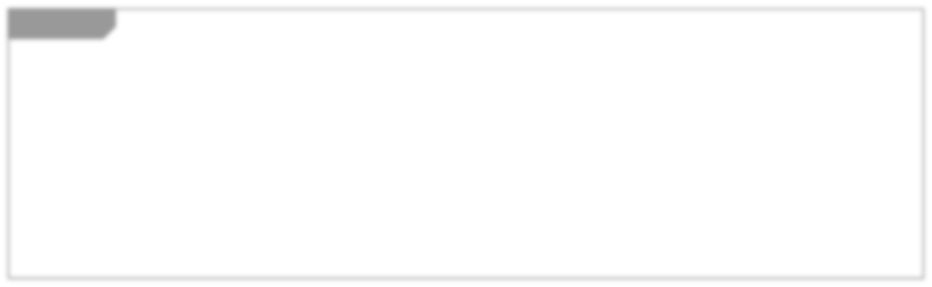
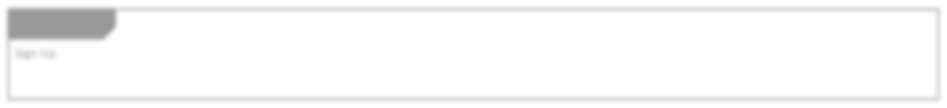
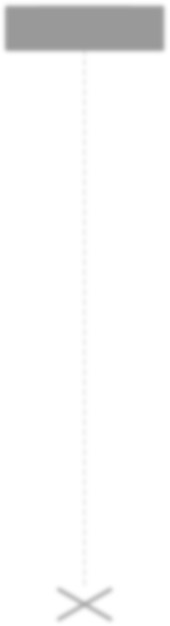
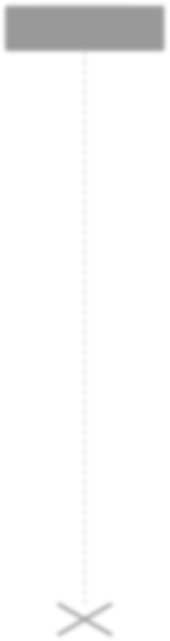
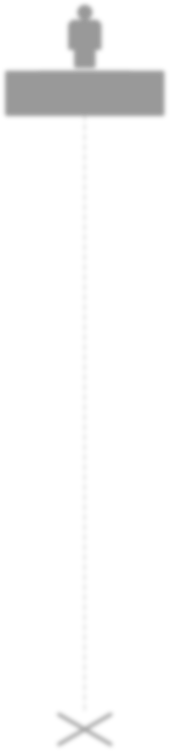
Failure

Failure

Failure to success

*Figure 25: Admin End*

**Sign In**



Start

Enter UserName

Checked UserName

Verify all credential

Enter Password

Checked Password

Success

Success

Failure

Failure

Failure

**alt**

Success

**Ref**

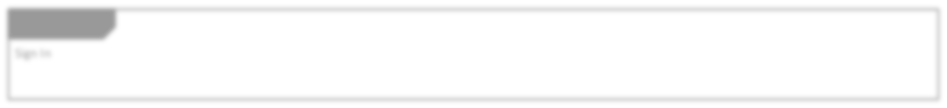
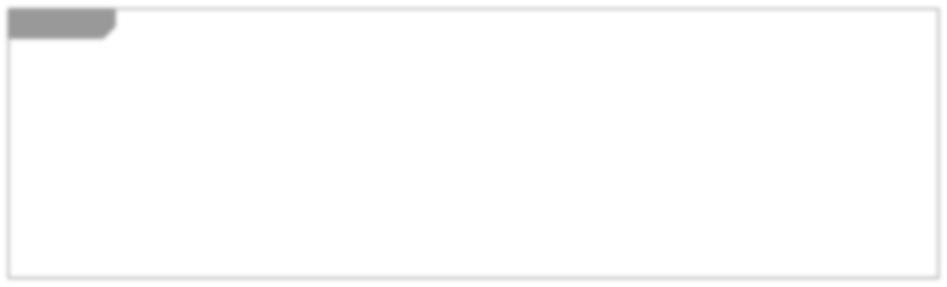
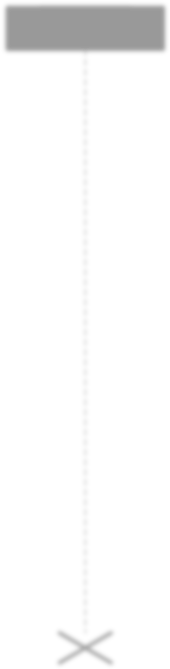
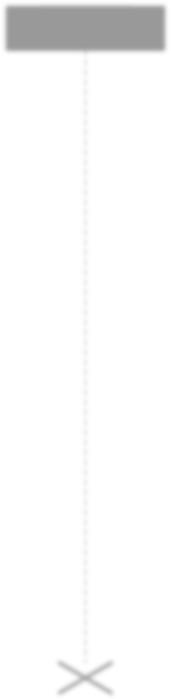
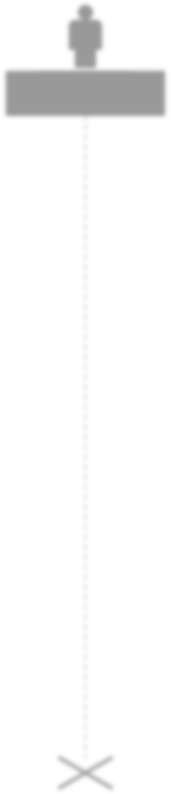
Sign Up

**b1:Database**

**s1:Sign In**

**a1:admin**

**Edit Profile:**



Start

Edit Name

store Edit Name

Edit UserName

store Edit UserName

Edit Password

Verify all credential

store Edit Password

Success

Edited Successfully

Failure

Failure to Edit

Failure

**alt**

Suceess

**Ref**

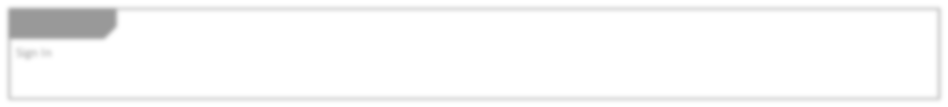
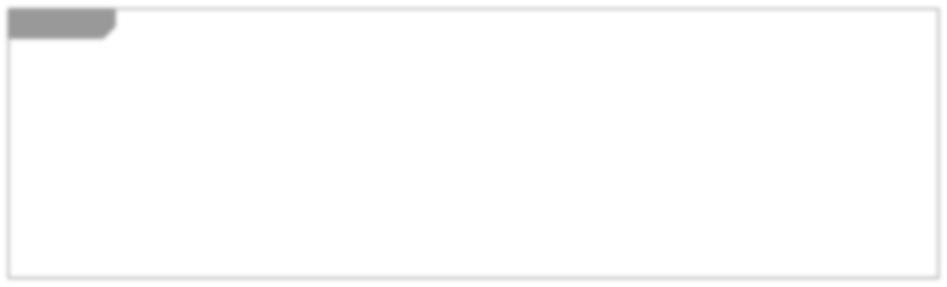
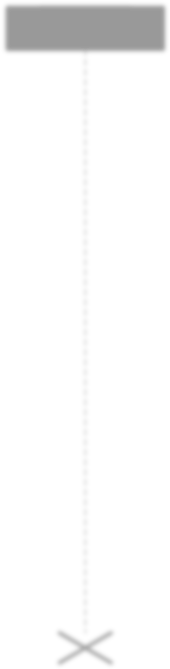
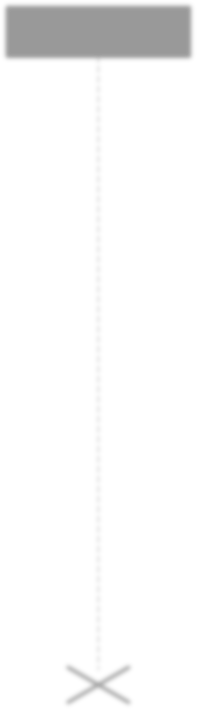
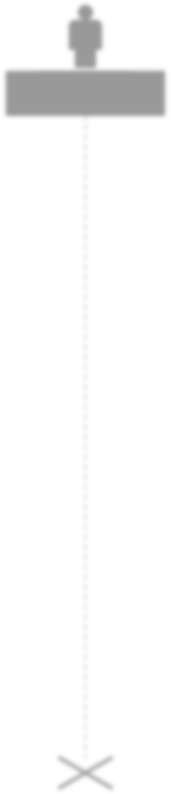
Sign In

**b1:Database**

**a1:admin**

**s1:Edit Profile**

**Manage Template:**



**a1:admin**

**s1:Manage**

**Template**

**b1:Database**

**Ref**

Sign In

Start

Add Template

store Template

Delete Template

Verify all credential

Request for delete

**alt**

Success

Suceess

Added or Deleted Successfully

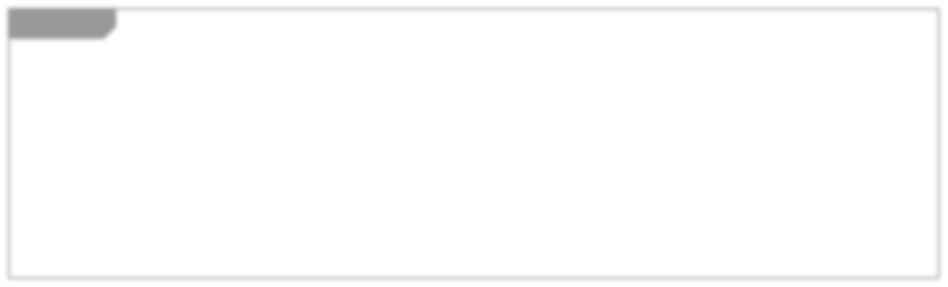
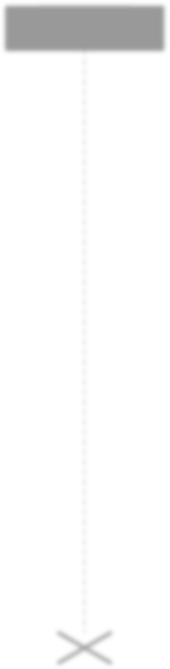
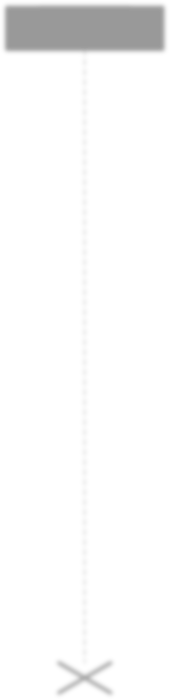
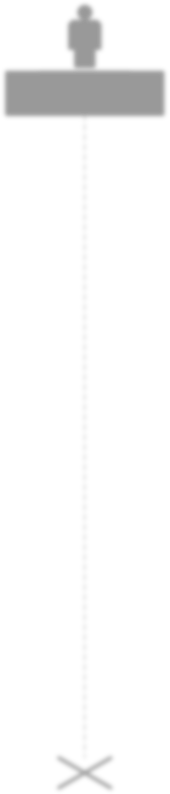
Failure

Failure

Failure to Added or Deleted

* + - 1. *User End*

##### Sign Up:



**a1:actor**

**s1:Sign Up**

**b1:Database**

Start

Enter Name

store Name

Enter UserName

Verify all credential

store UserName

Enter Password

store Password

Success

**alt**

Suceess

Sign Up Successfully

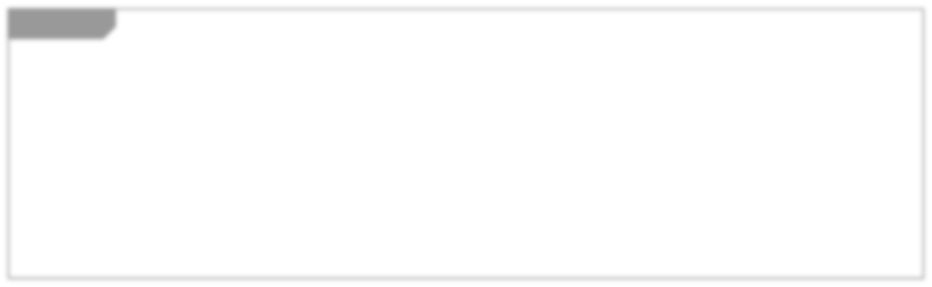
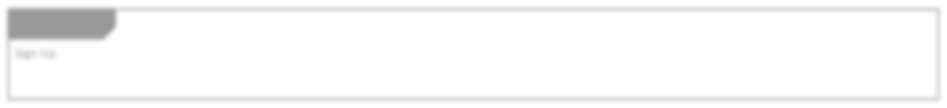
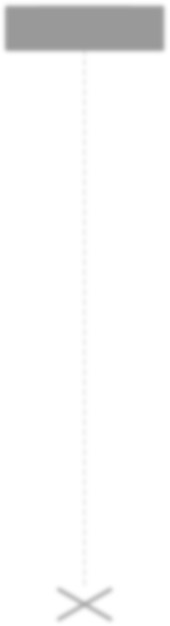
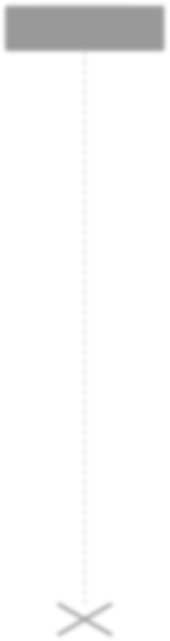
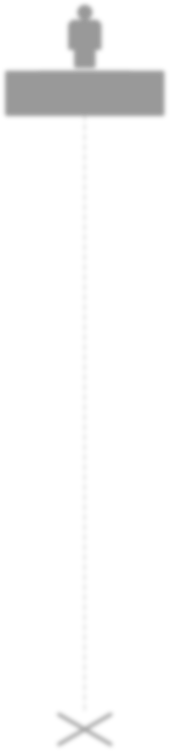
Failure

Failure

Failure to success

*Figure 26: User End*

**Sign In**



Start

Enter UserName

Checked UserName

Verify all credential

Enter Password

Checked Password

Success

Success

Failure

Failure

Failure

**alt**

Success

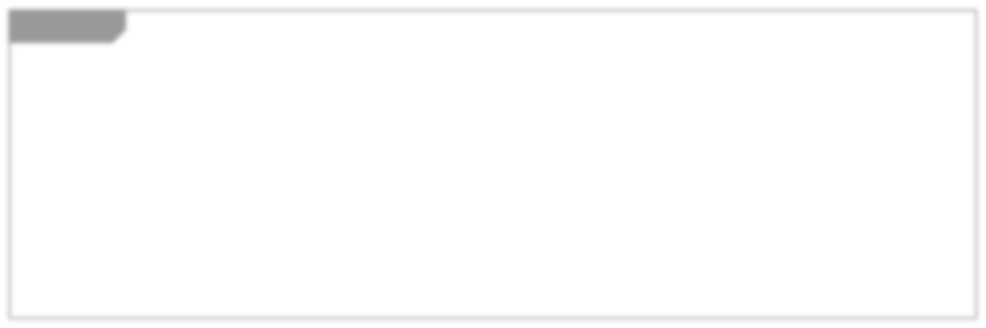
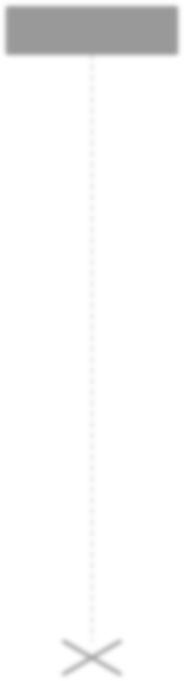
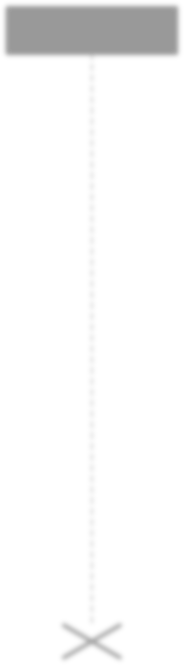
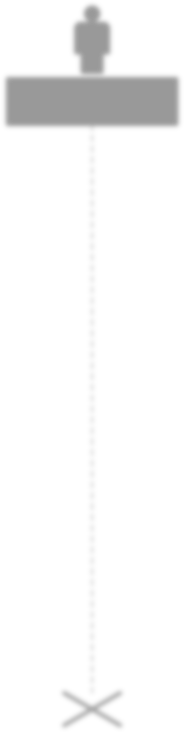
**Ref**

Sign Up

**b1:Database**

**s1:Sign In**

**a1:Actor**



Start

Enter Prompt

Send Prompt

**alt**

Failure

Prompt could not be understandable

Please enter specific Prompt

Success

Prompt understood succesfully

Success: Redirecting to Main Page

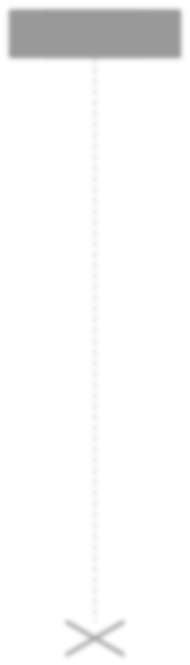
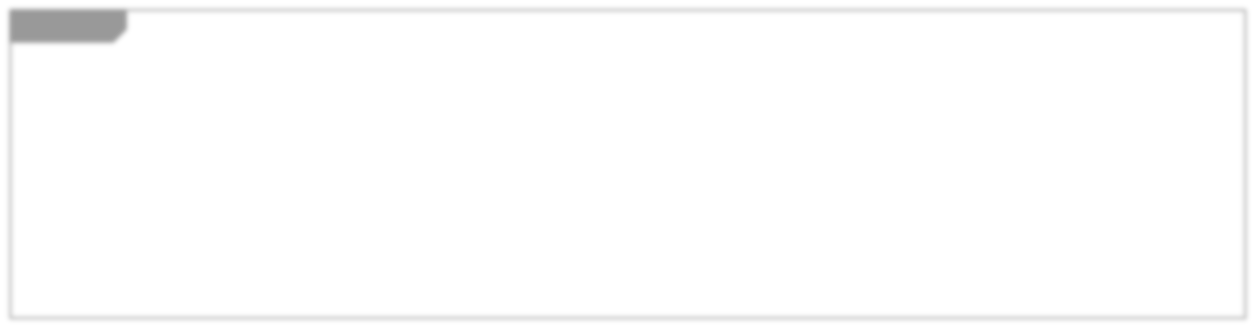
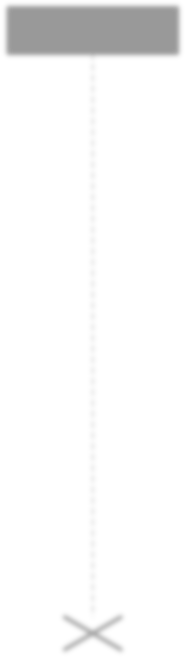
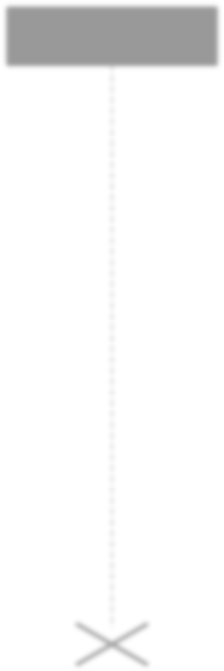
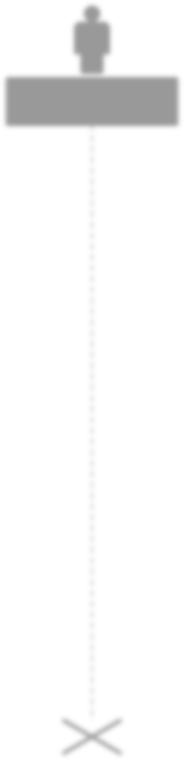
**Ref**

Prompt Page

**o2:API**

**o1:Prompt**

**a1:Actor**



**a1:Actor**

**o1:Main E Commerce**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Main Page Items

Request send

**alt**

Items not Found

Failure

Show Error Message

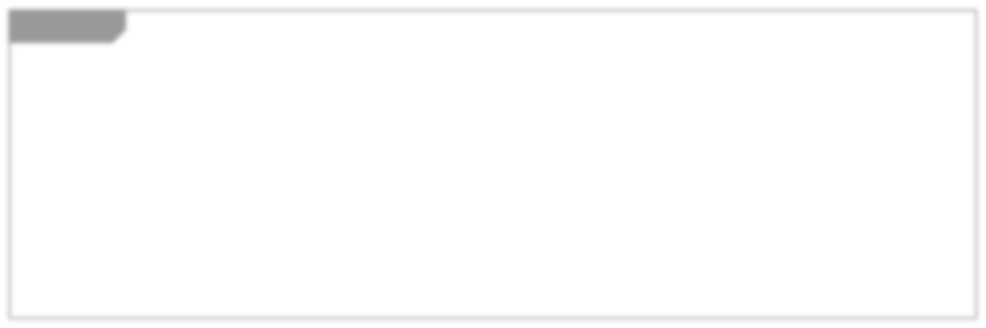
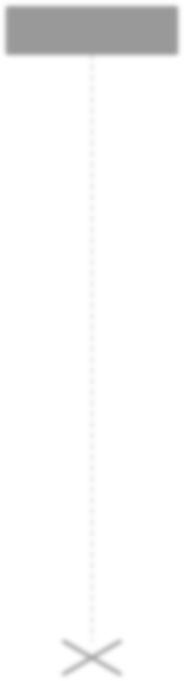
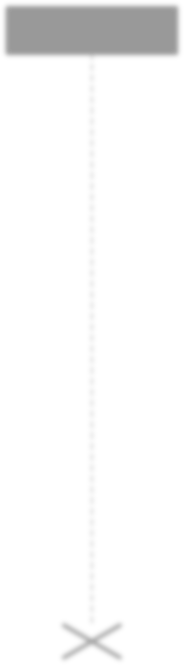
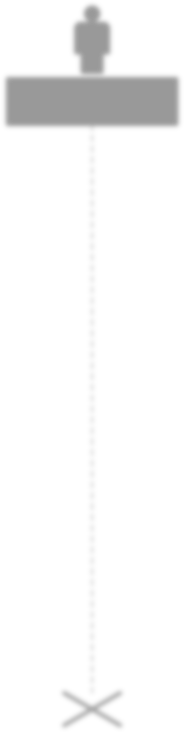
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

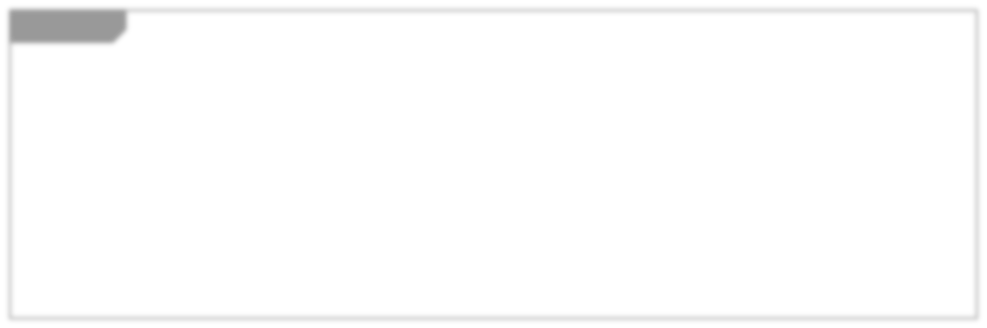
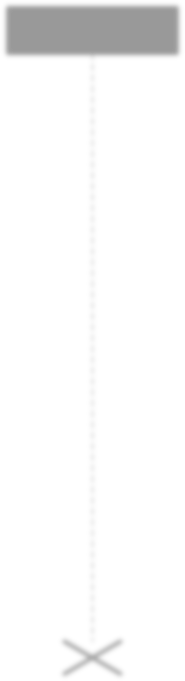
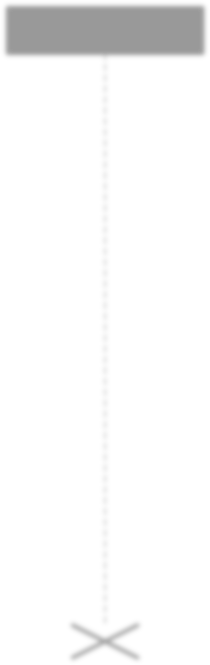
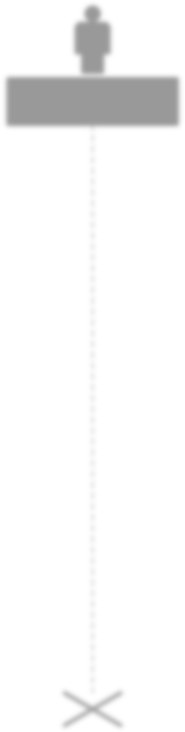
**Ref**

Prompt Page

**o2:Coding Page**

**o1:Main**

**a1:Actor**



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

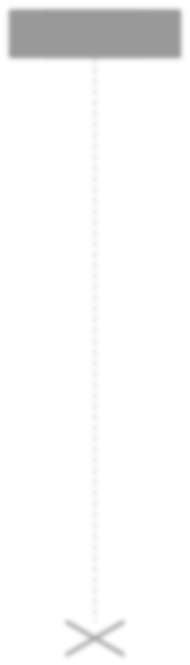
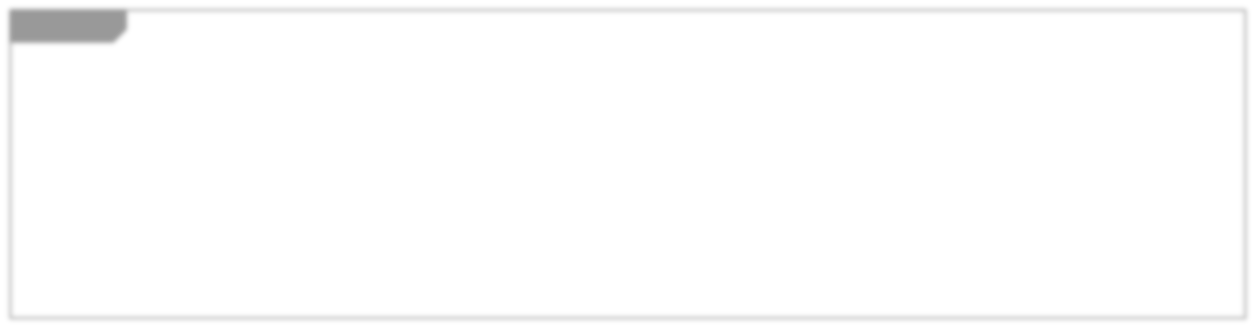
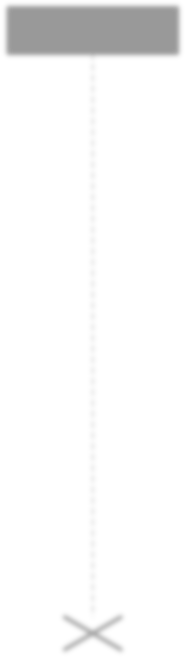
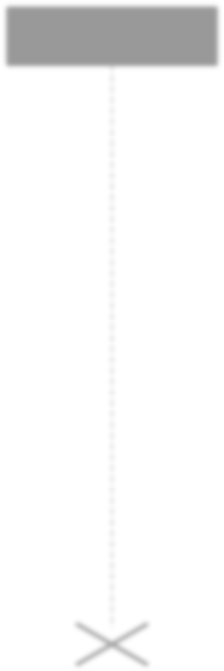
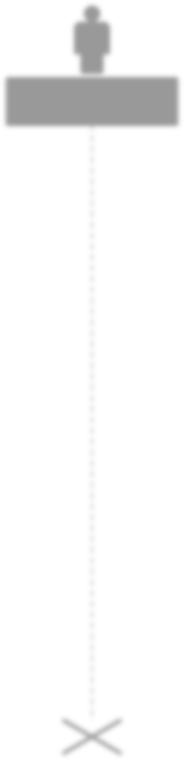
**Ref**

Main Page

**o2:Coding Page**

**o1:Category**

**a1:Actor**



**a1:Actor**

**o1:Category**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Category Page Items

Request send

**alt**

Items not Found

Failure

Show Error Message

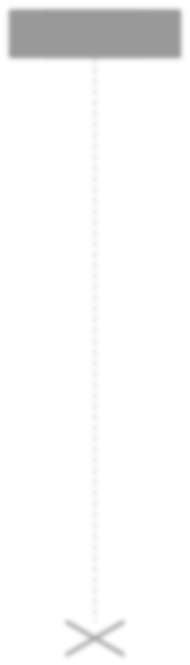
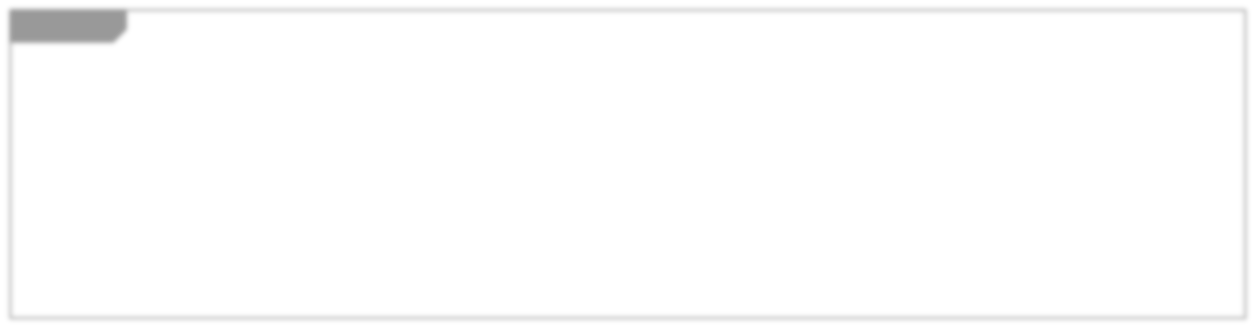
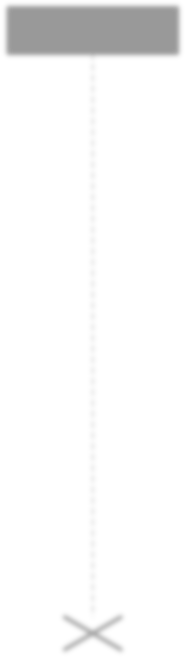
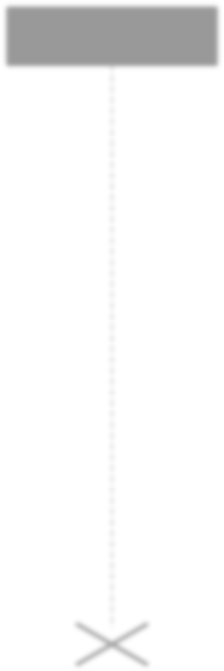
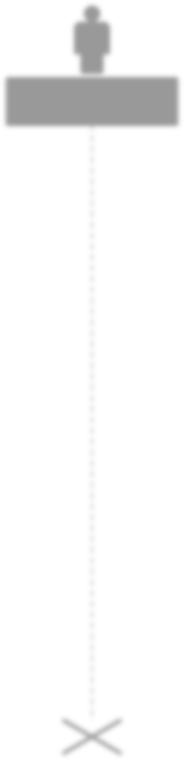
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



**a1:Actor**

**o1:Contact**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Contact Page Items

Request send

**alt**

Items not Found

Failure

Show Error Message

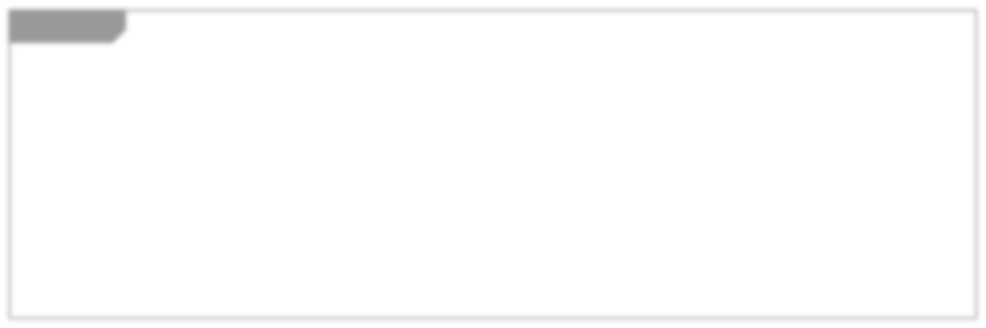
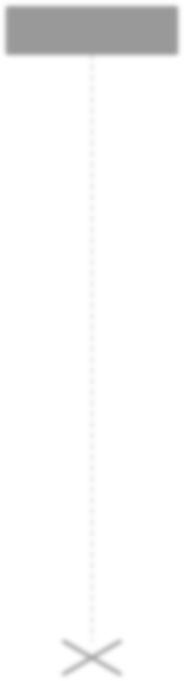
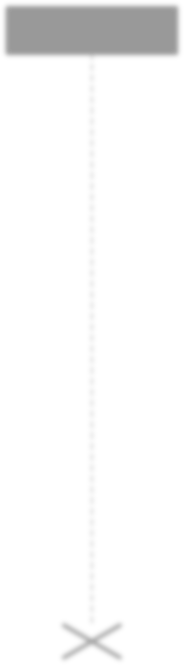
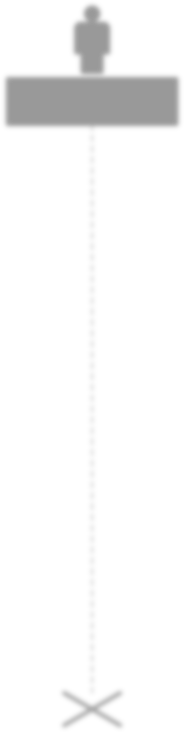
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

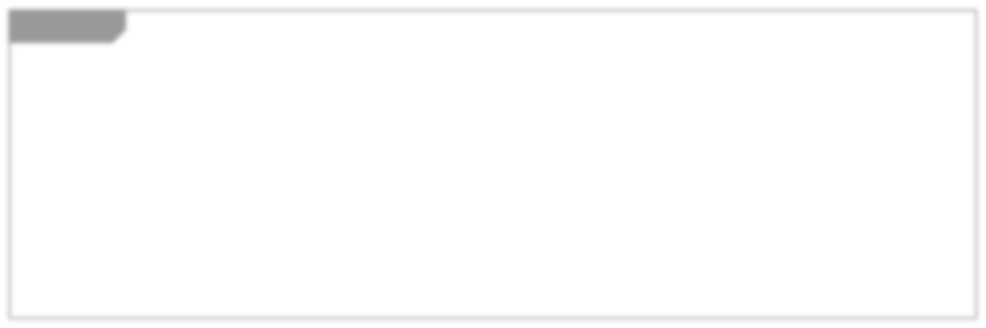
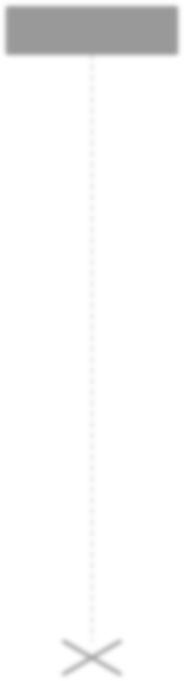
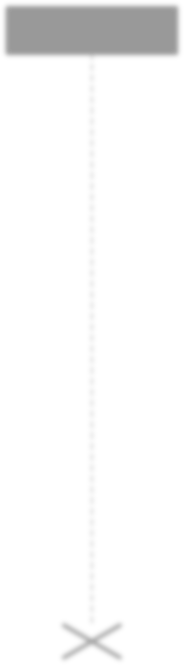
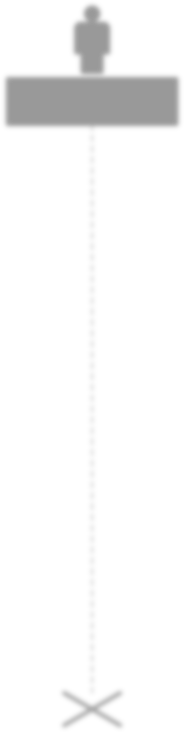
**Ref**

Prompt Page

**o2:Coding Page**

**o1:Contact**

**a1:Actor**



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

**Ref**

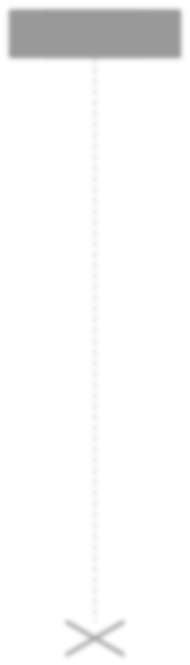
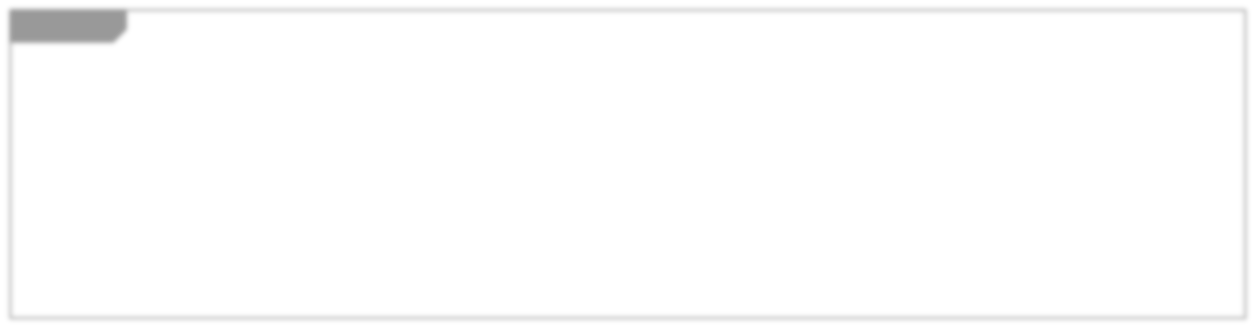
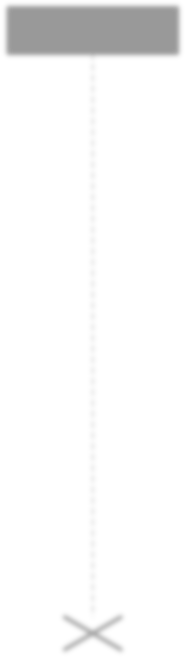
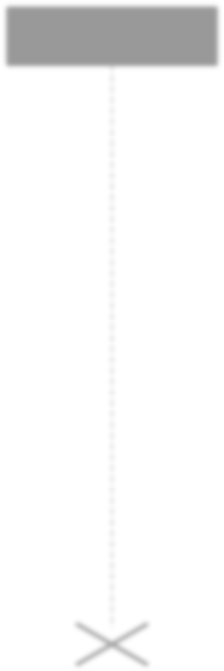
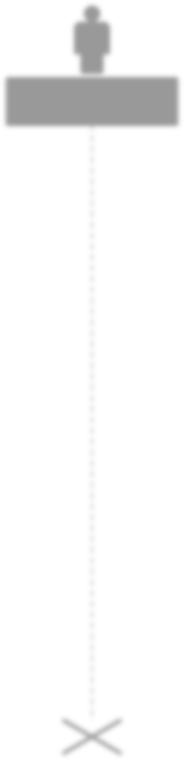
Prompt Page

**o2:Coding Page**

**o1:Customer**

**Login**

**a1:Actor**



**a1:Actor**

**o1:Customer Login**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Customer Login Page Items

Request send

**alt**

Items not Found

Failure

Show Error Message

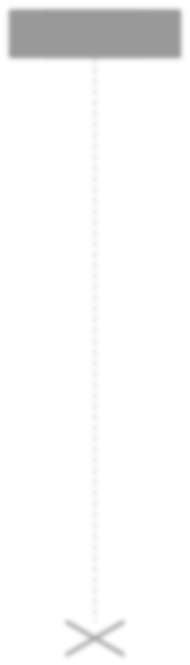
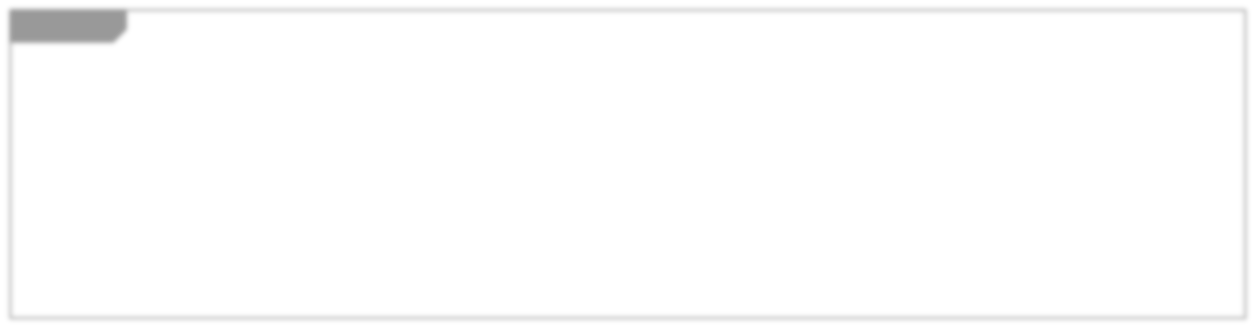
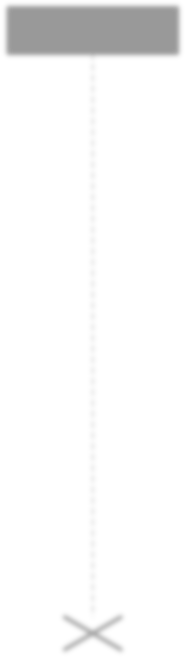
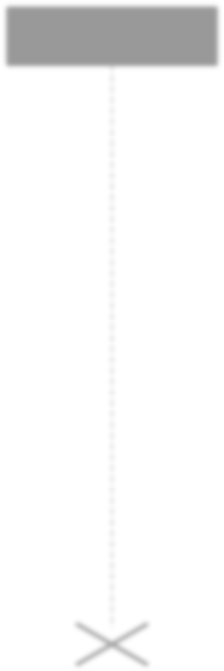
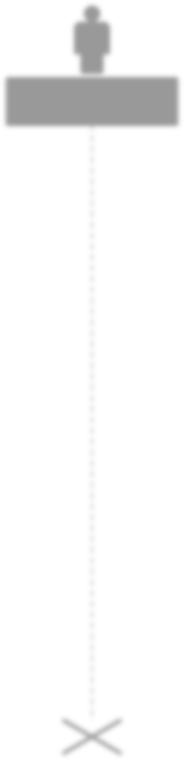
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



**a1:Actor**

**o1:Customer Details**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Customer details Page Items

Request send

**alt**

Items not Found

Failure

Show Error Message

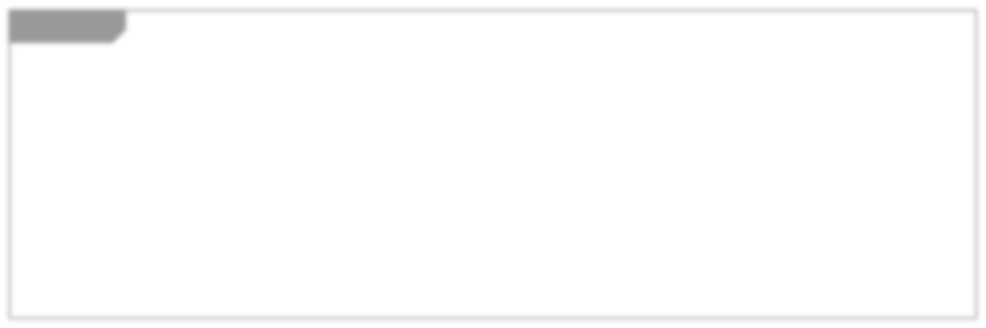
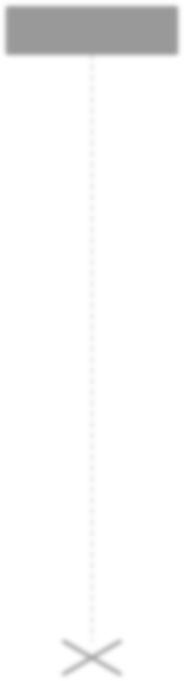
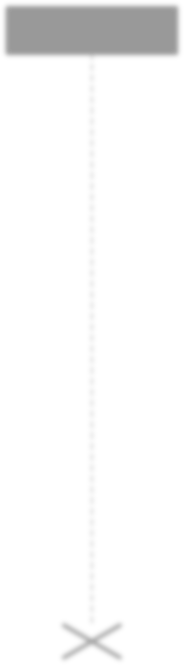
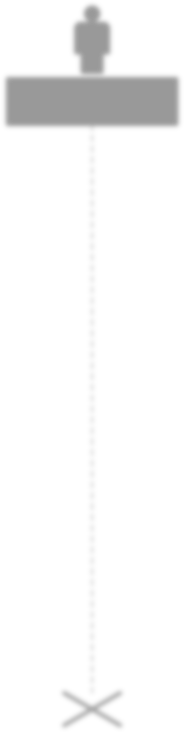
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

**Ref**

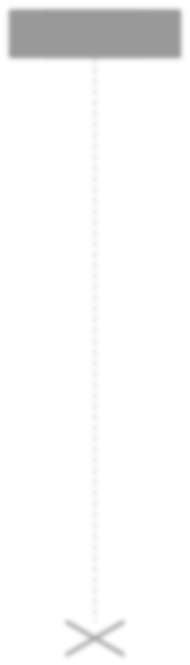
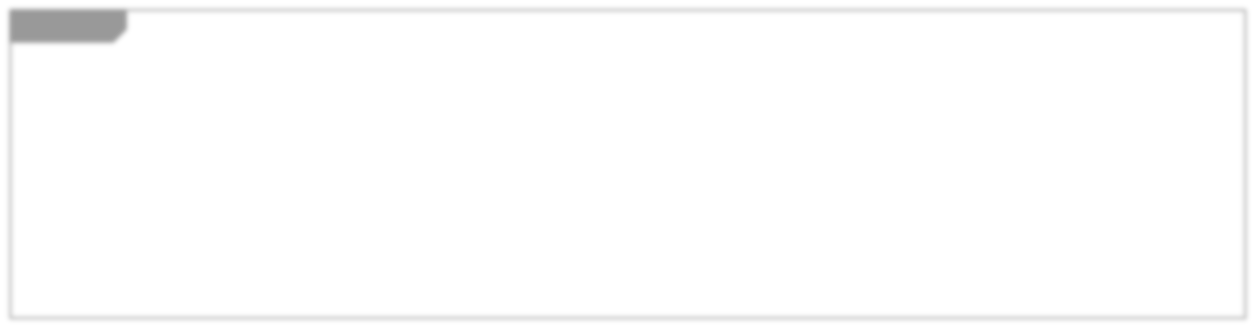
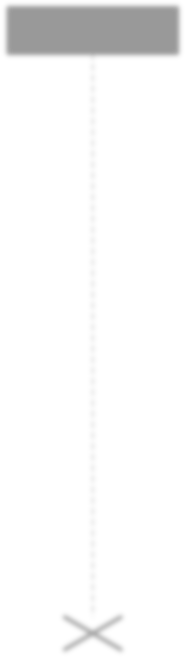
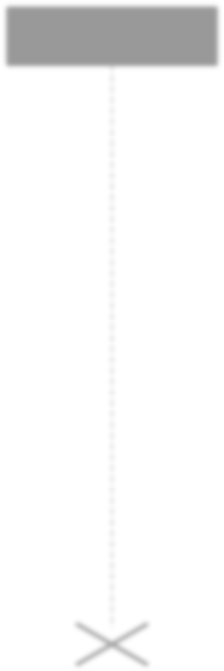
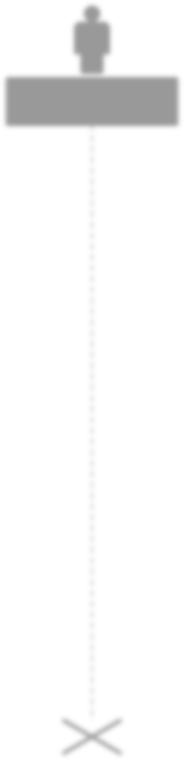
Prompt Page

**o2:Coding Page**

**o1:Customer**

**details**

**a1:Actor**



**a1:Actor**

**o1:Cart**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Cart Page Items

Request send

**alt**

Items not Found

Failure

Show Error Message

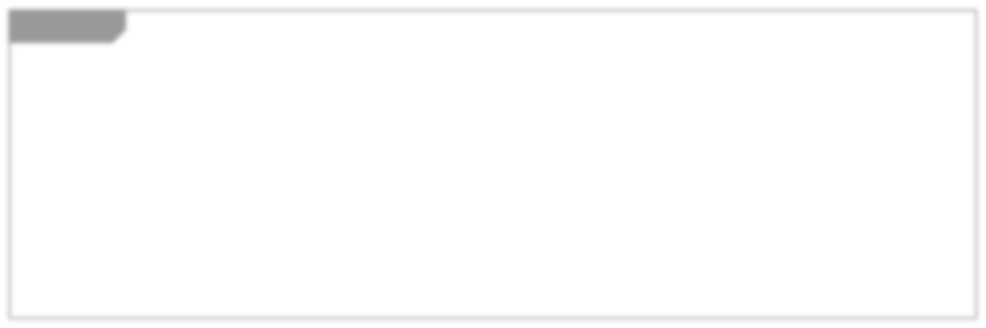
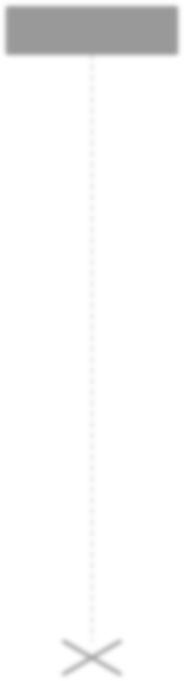
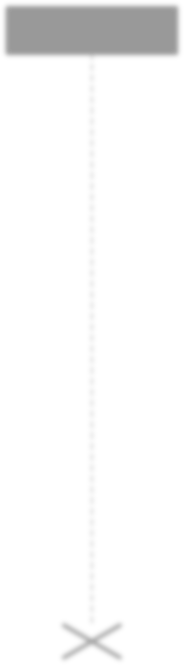
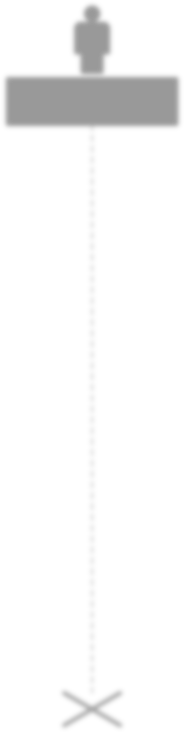
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

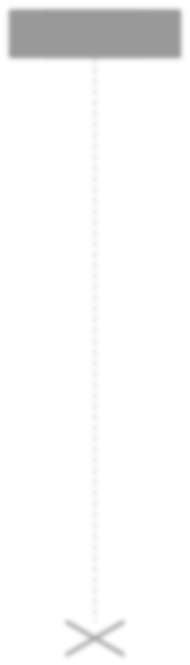
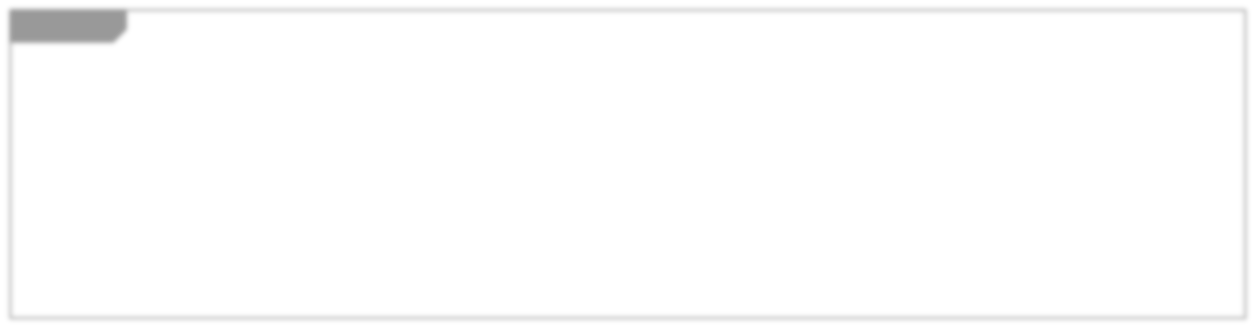
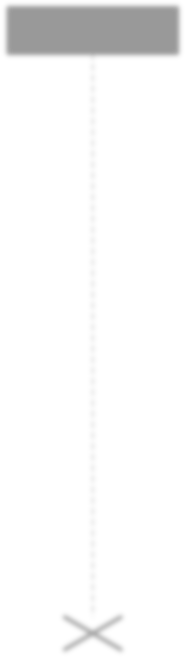
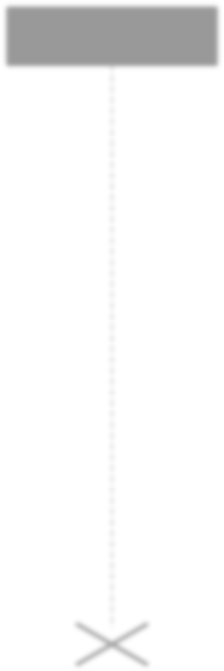
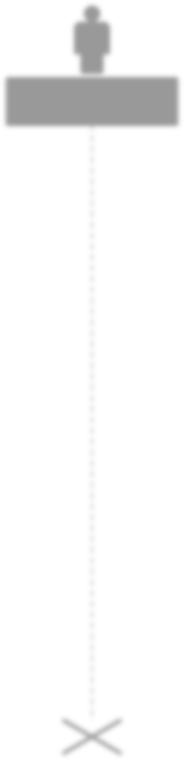
**Ref**

Prompt Page

**o2:Coding Page**

**o1:Cart**

**a1:Actor**



**a1:Actor**

**o1:Main Blogs**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Main Page Items

Request send

**alt**

Items not Found

Failure

Show Error Message

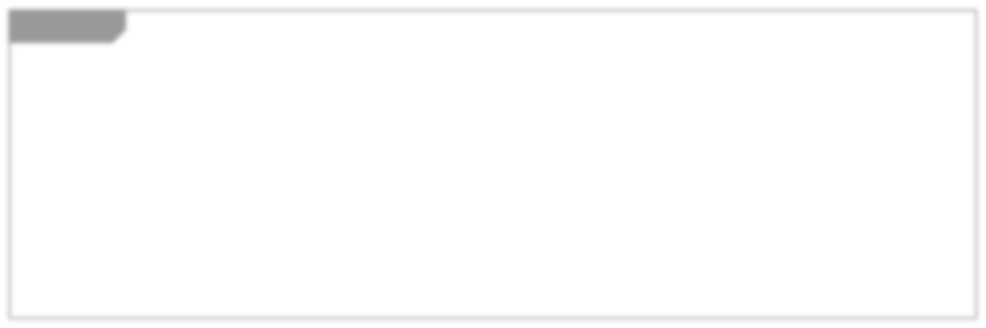
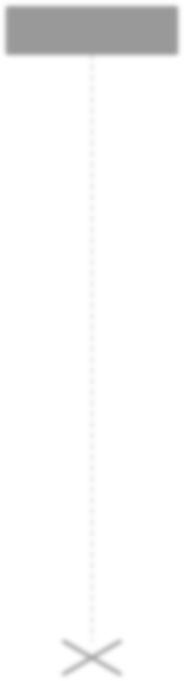
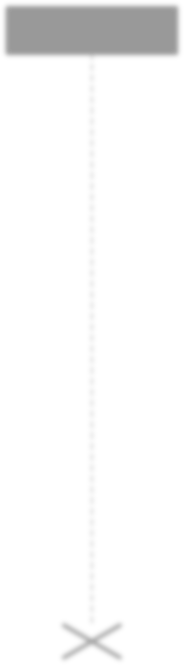
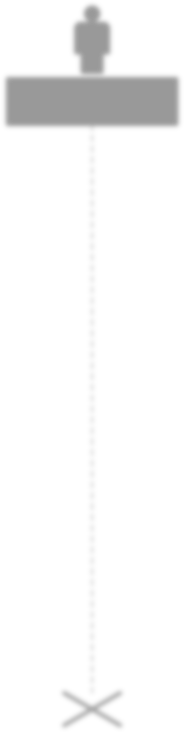
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

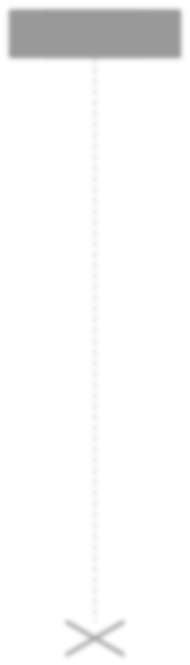
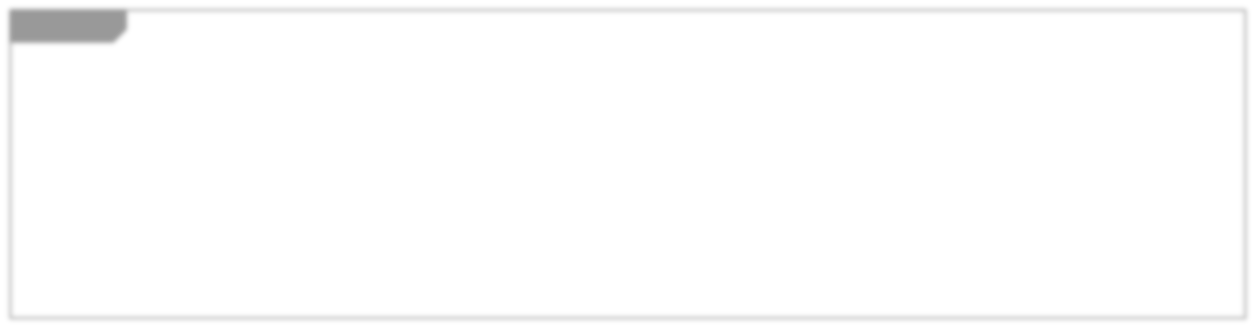
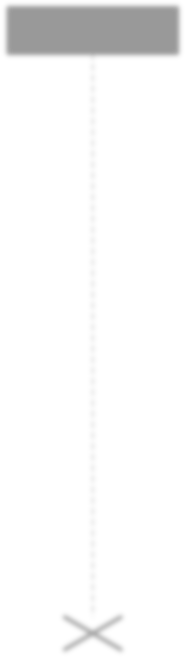
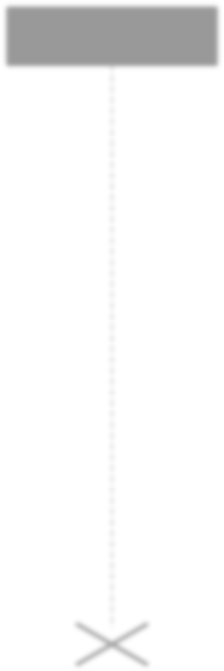
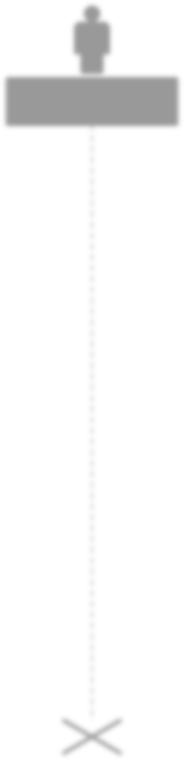
**Ref**

Prompt Page

**o2:Coding Page**

**o1:Main Blogs**

**a1:Actor**



**a1:Actor**

**o1:Post Images**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Post Images Items

Request send

**alt**

Items not Found

Failure

Show Error Message

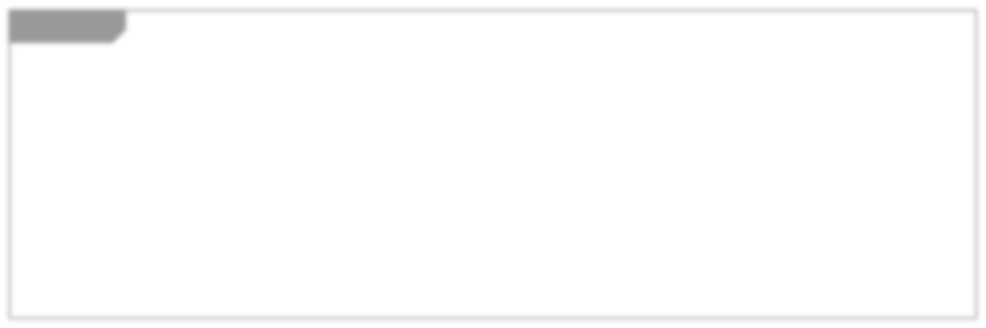
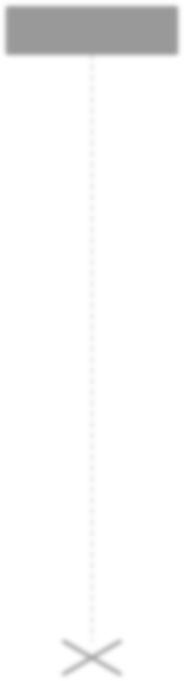
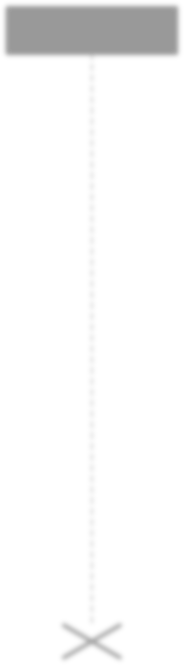
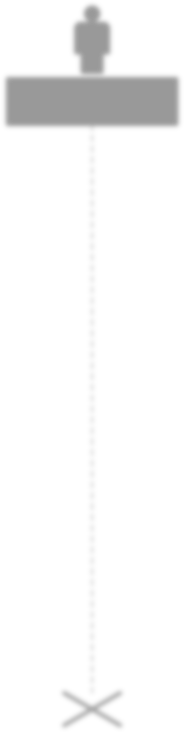
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

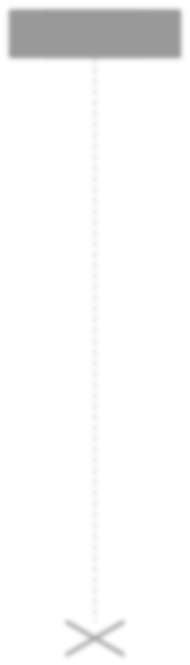
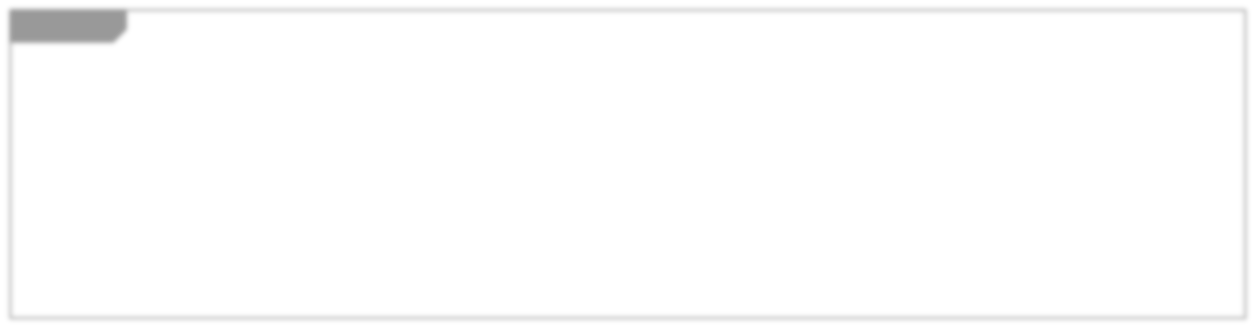
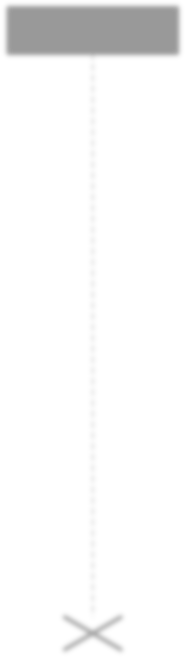
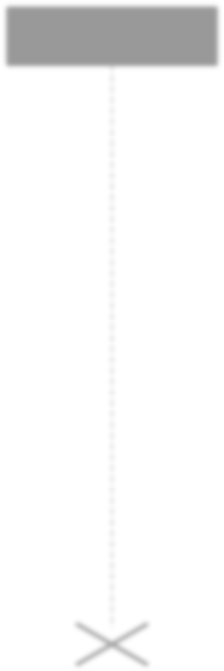
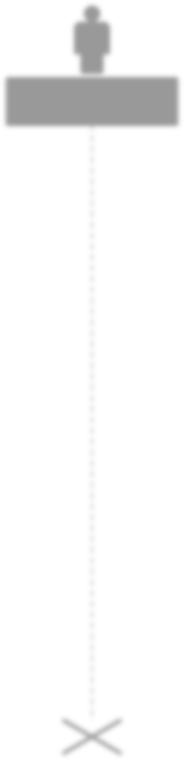
**Ref**

Prompt Page

**o2:Coding Page**

**o1:Post Images**

**a1:Actor**



**a1:Actor**

**o1:Post Vedios**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Post Vedios Items

Request send

**alt**

Items not Found

Failure

Show Error Message

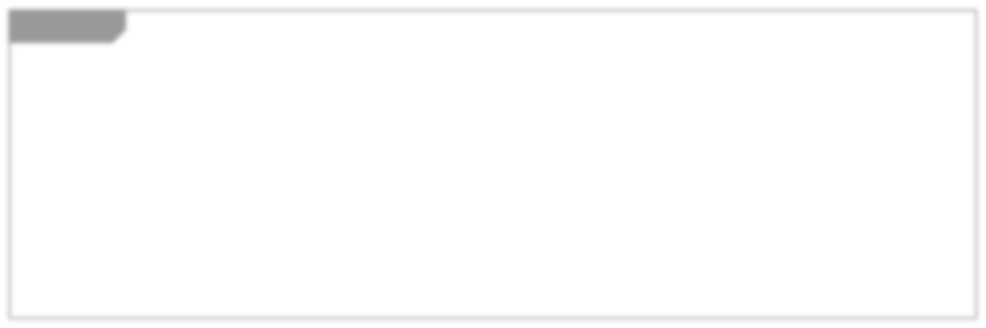
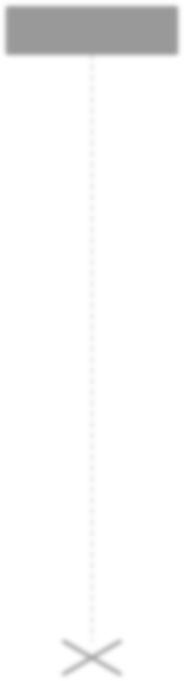
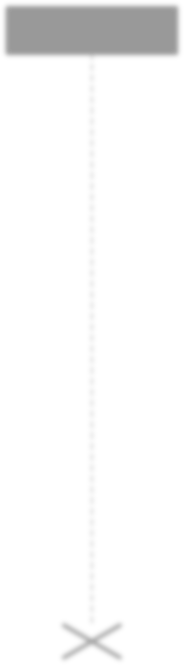
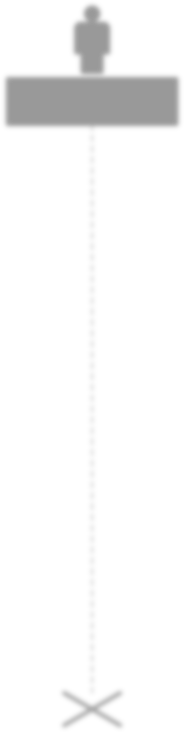
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

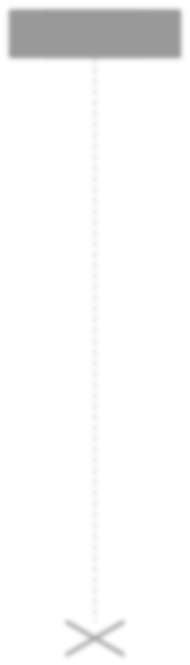
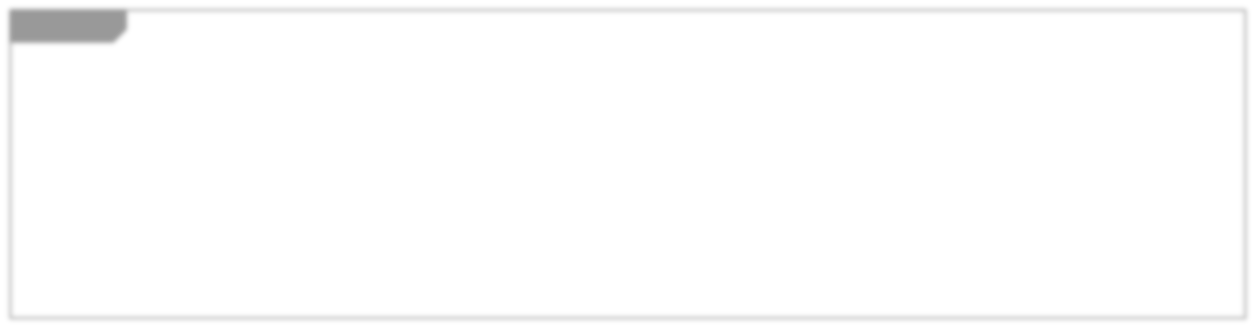
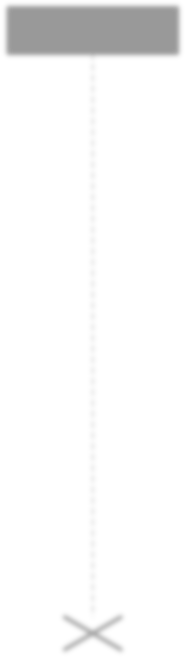
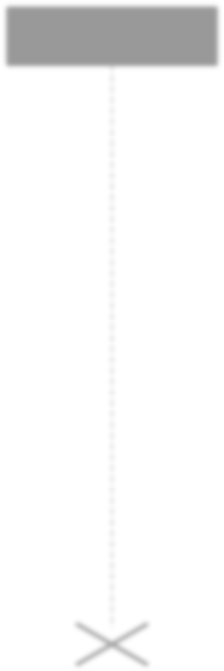
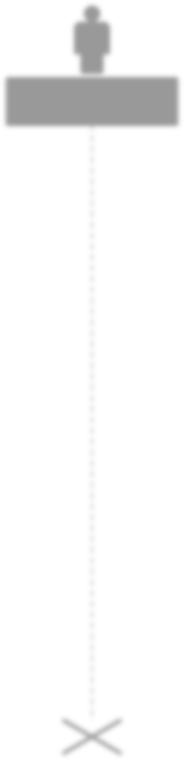
**Ref**

Prompt Page

**o2:Coding Page**

**o1:Post Vedios**

**a1:Actor**



**a1:Actor**

**o1:Second Style**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Second Style Items

Request send

**alt**

Items not Found

Failure

Show Error Message

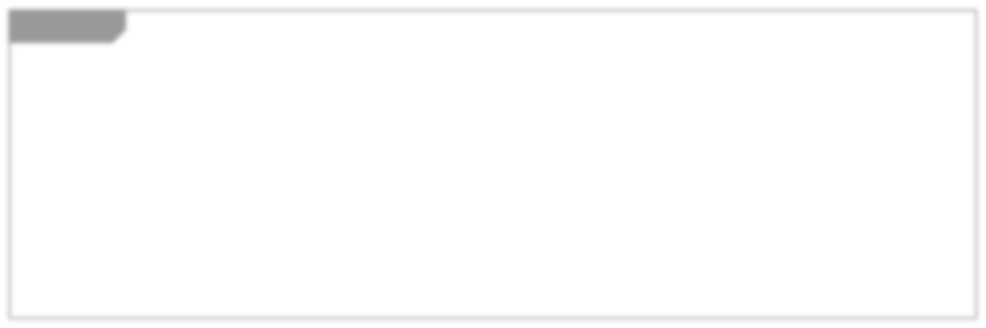
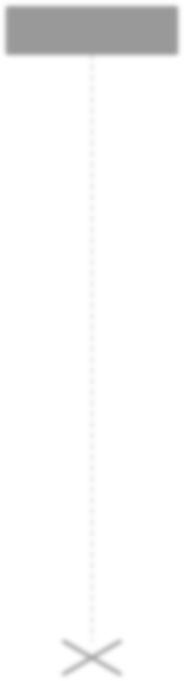
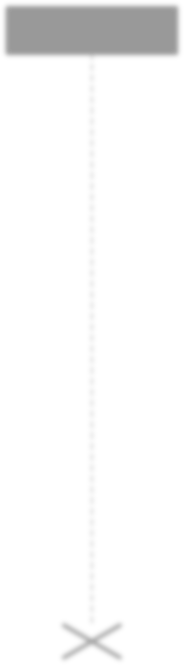
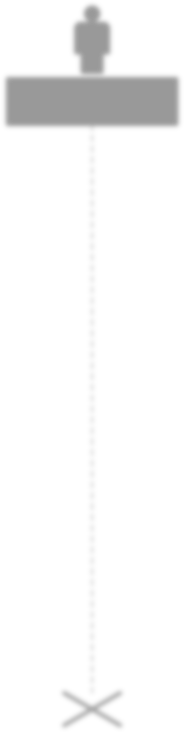
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

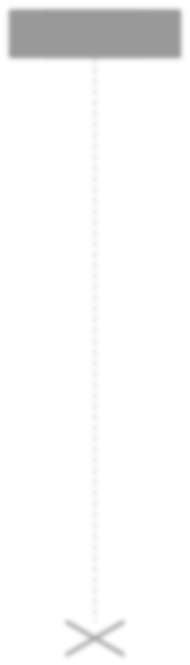
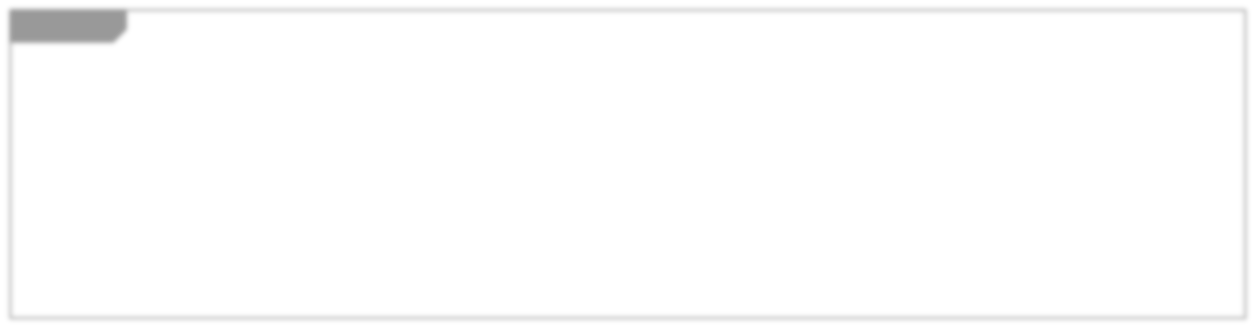
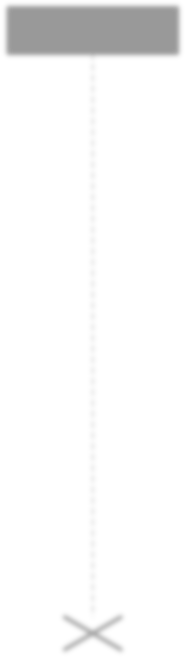
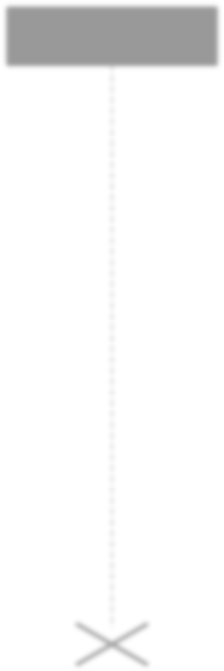
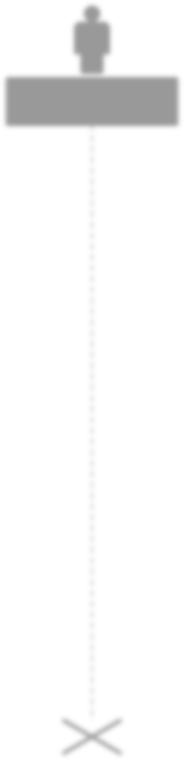
**Ref**

Prompt Page

**o2:Coding Page**

**o1:Second style**

**a1:Actor**



**a1:Actor**

**o1:Main Portfolio**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Main Portfolio Items

Request send

**alt**

Items not Found

Failure

Show Error Message

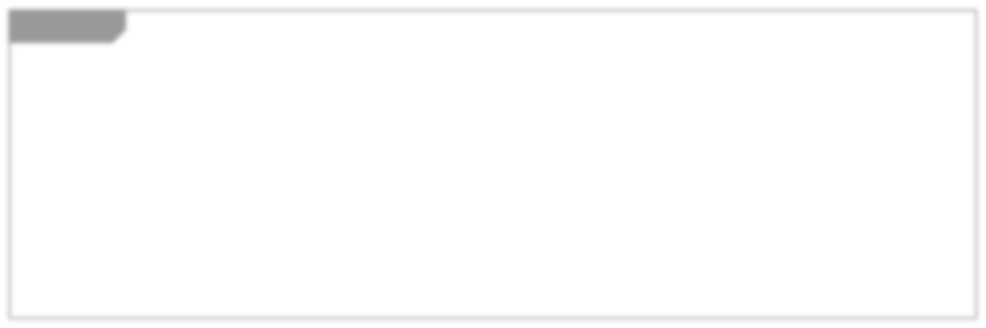
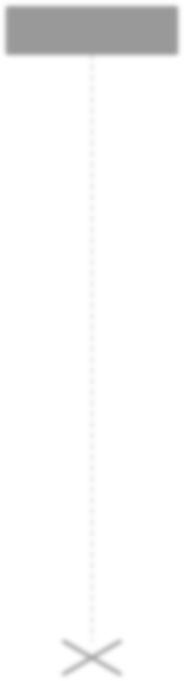
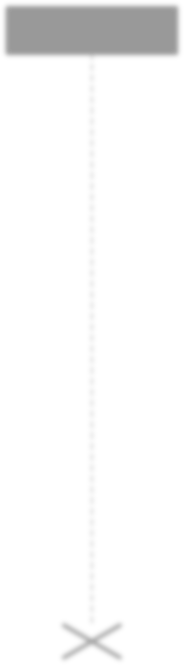
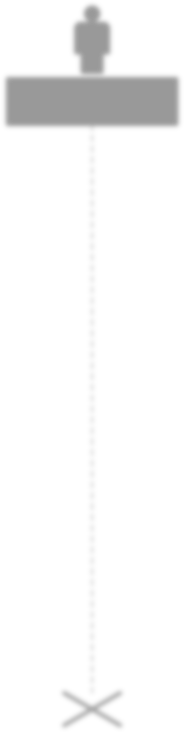
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



Start

Change Items Info

Request send

**alt**

Failure

Items could not be change

Show Error Message

Success

Items Changed

Show Changed Items

Generate Code

Save Html Code

**Ref**

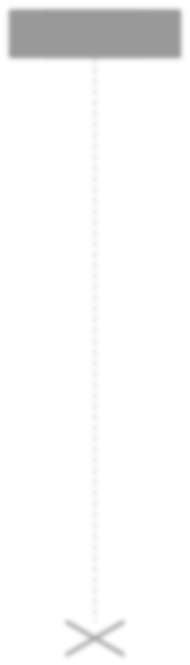
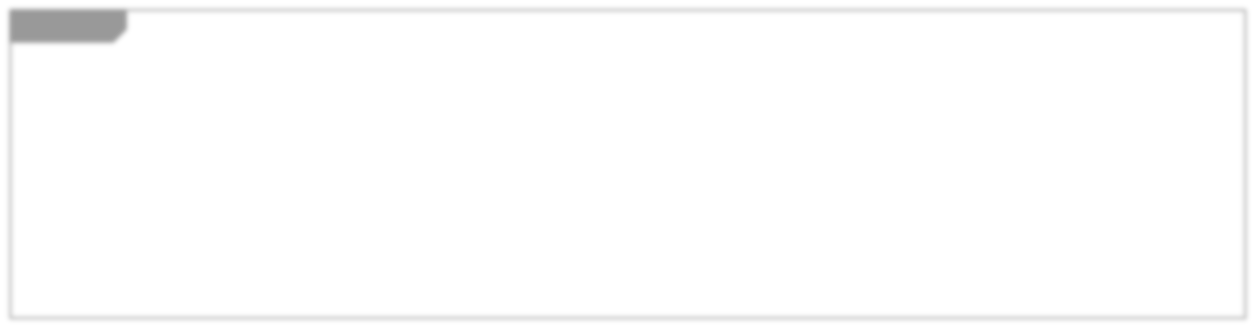
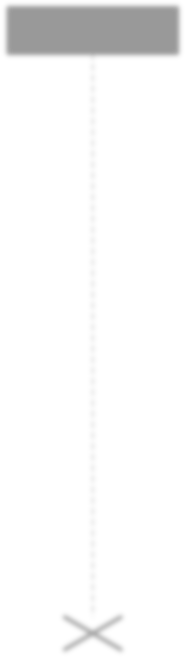
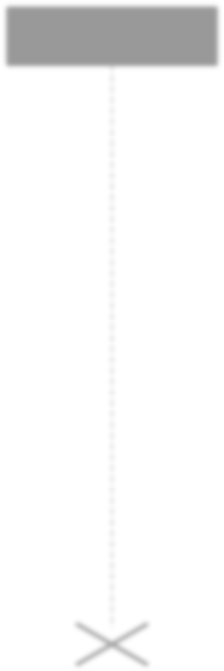
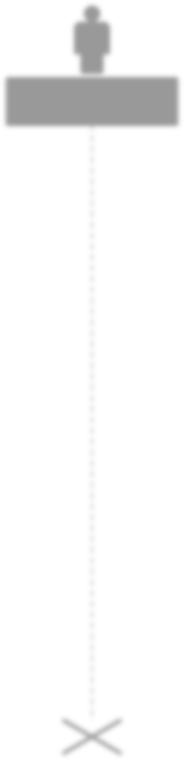
Prompt Page

**o2:Coding Page**

**o1:Main**

**portfolio**

**a1:Actor**



**a1:Actor**

**o1:Projects**

**o2:API**

**o1:Coding Page**

**Ref**

Prompt

Start

Requesting for Projects Items

Request send

**alt**

Items not Found

Failure

Show Error Message

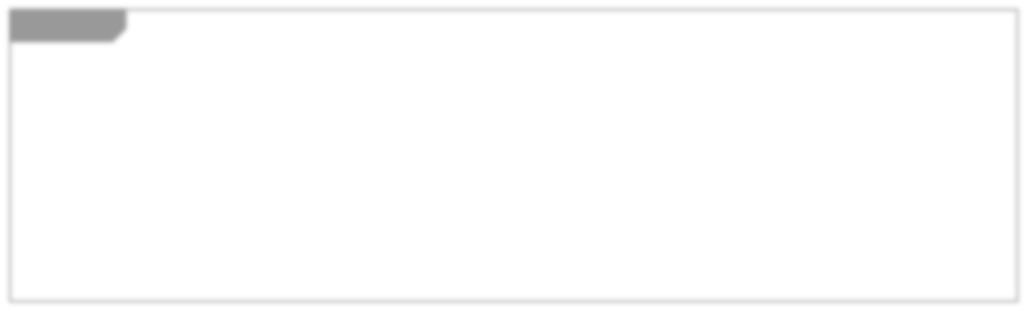
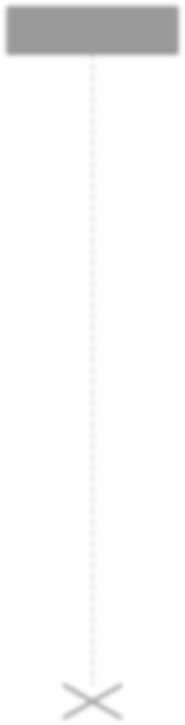
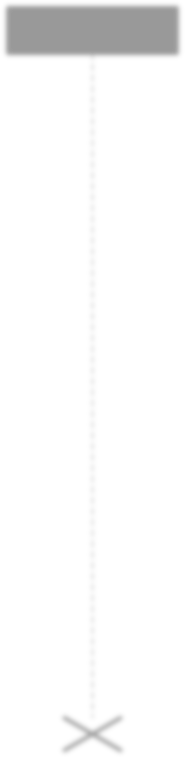
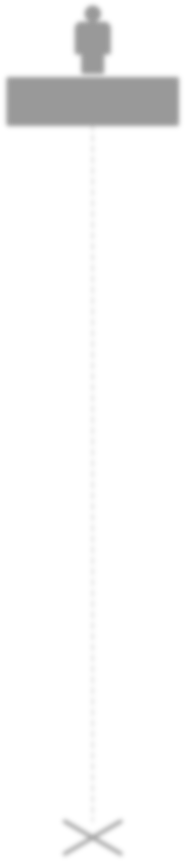
Success

Items Found

Show Items on screen

Generate Code

Save HTML code



**a1:Actor**

Start Gate

Enter Name

store Name

Enter UserName

store UserName

Verify all credential

Enter Password

store Password

**alt**

Success

Suceess

Sign Up Successfully

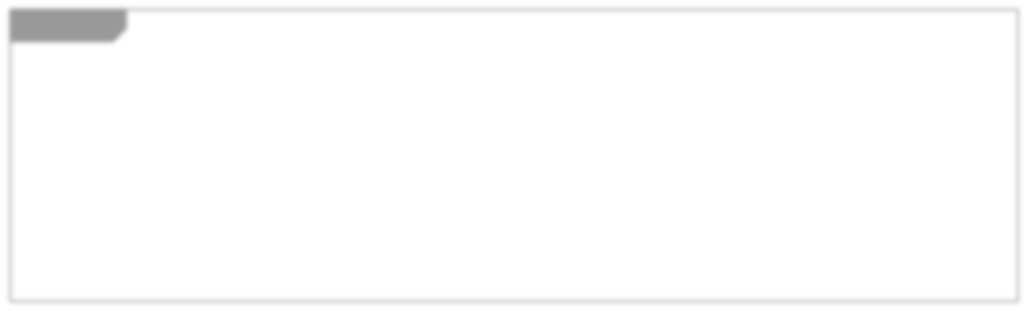
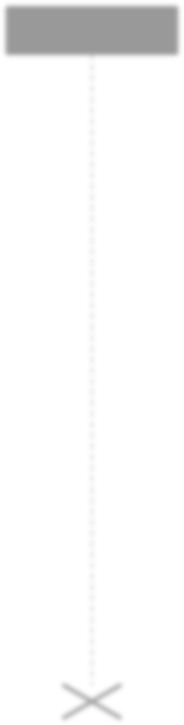
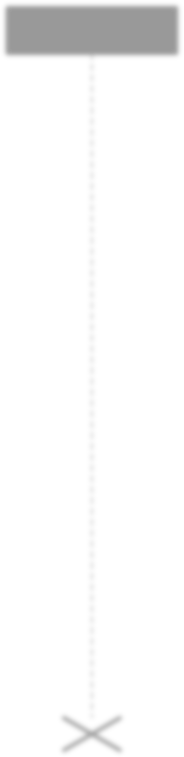
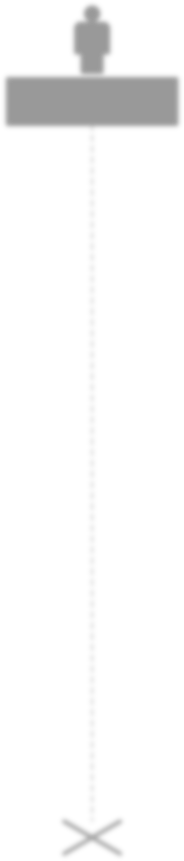
Failure

Failure

Failure to success

**b1:Database**

**s1:Sign Up**



**a1:Actor**

Start

Gate

Enter Username

Checked username

Verify all credential

Enter Password

Checked Password

**alt**

Success

Suceess

Success

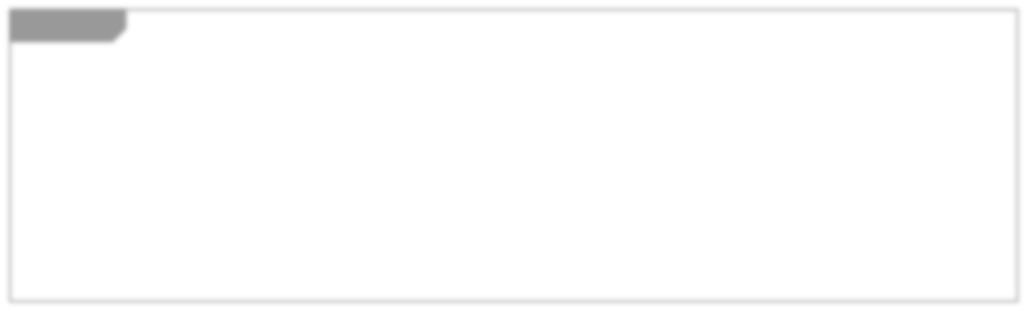
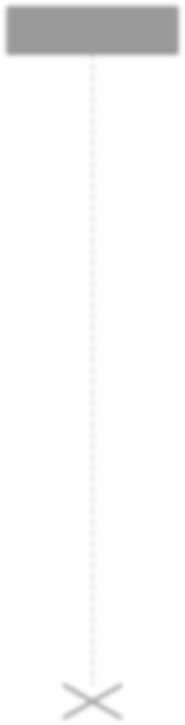
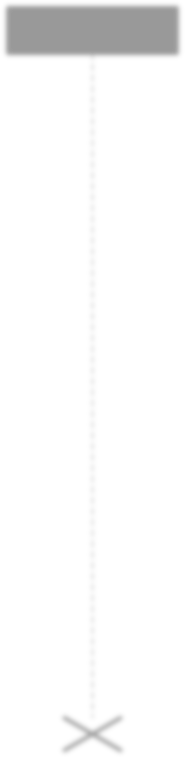
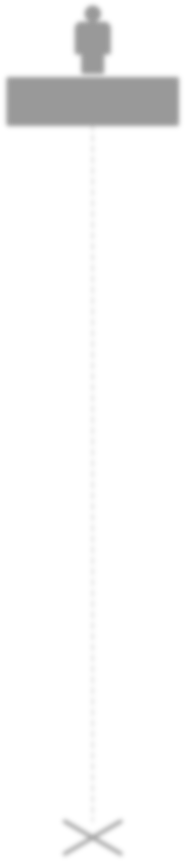
Failure

Failure

Failure

**b1:Database**

**s1:Sign In**



**a1:Actor**

Start Gate

Enter Username

Checked username

Store all credential

Enter Contact info

Checked Contact info

**alt**

Success

Suceess

Success

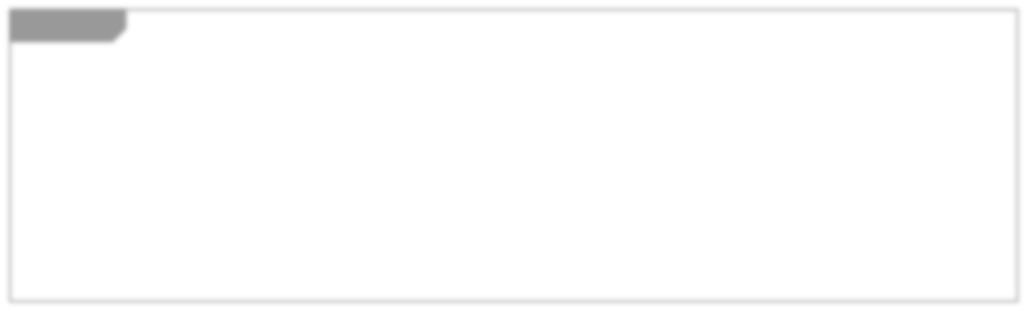
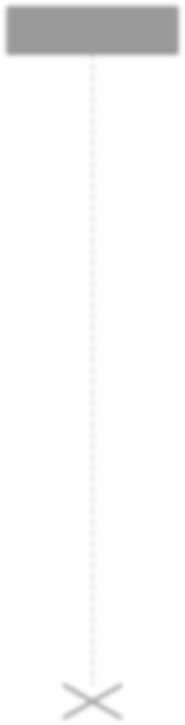
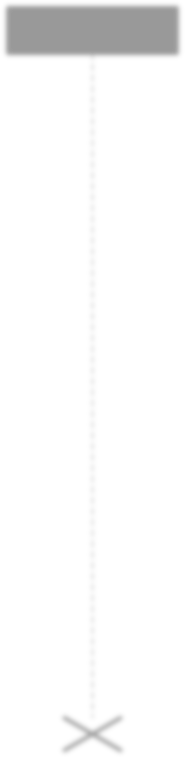
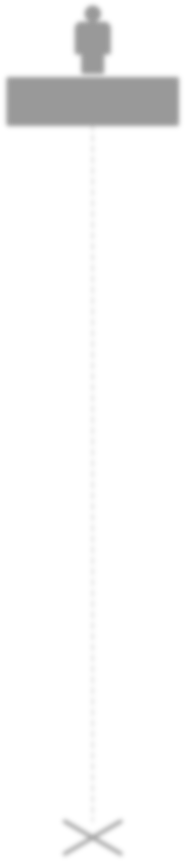
Failure

Failure

Failure

**b1:Database**

**s1:Contact Us**



**a1:Actor**

**b1:API**

Start Gate

Select Website Type

Send to those API

Analyzed and sending response

Enter Prompt

Analyze Prompt

**alt**

Success

Suceess

Directing to Website

Failure

Failure

Please Enter more specific Prompt

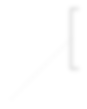
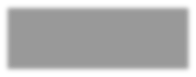
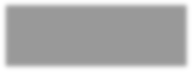
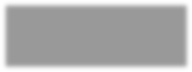
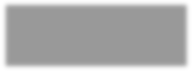
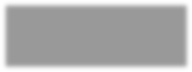
**s1:Prompt**

#### Collaboration Diagram

* + - 1. *Admin End*

Start

* + - * 1. **Initiate**



End

**7. End**

Admin

6.0 Result

* 1. Enter Name
  2. Enter UserName
  3. Enter Password

Sign Up

5.0 Result

* 1. Store Name
  2. Store UserName
  3. Store Password

4.0 Verify All the

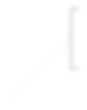
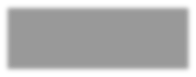
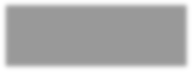
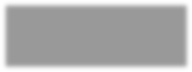
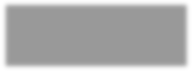
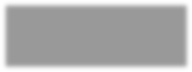
Database

credential

*Figure 27: Admin End Collaboration*

Start

**1. Initiate**



End

**7. End**

Admin

6.0 Result

2.0 Enter UserName

2.1 Enter Password

Sign In

5.0 Result

3.0 Check UserName

3.2 Check Password

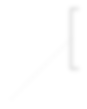
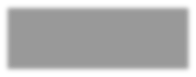
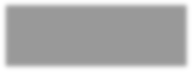
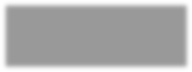
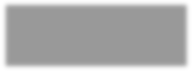
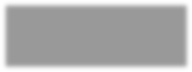
4.0 Verify All the

Database

credential

Start

**1. Initiate**



End

**7. End**

Admin

6.0 Result

* 1. Edit Name
  2. Edit UserName
  3. Edit Password

Sign Up

5.0 Result

* 1. Store Edit Name
  2. Store Edit UserName
  3. Store Edit Password

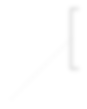
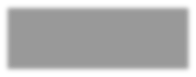
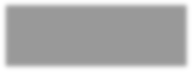
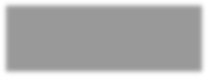
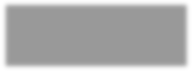
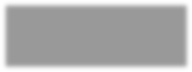
4.0 Verify All the

Database

credential

Start

**1. Initiate**



End

**7. End**

Admin

6.0 Result

2.0 Add Template

2.1 Delete Template

Manage Template

5.0 Result

3.0 Store Template

3.1 Request For Delete

4.0 Verify All the

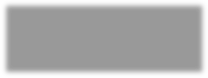
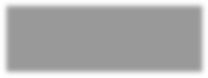
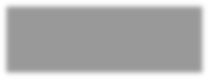
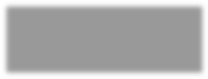
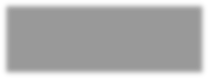
Database

credential

* + - 1. *User End*

Start

* + - * 1. **Initiate**



Actor

Select Web TYPE

* 1. E-Commerce
  2. Blog
  3. Portfolio

Web Selection

Activate API

* 1. E-Commerce
  2. Blog
  3. Portfolio

4.0 Verifying the API

DataSet

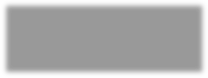
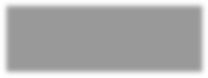
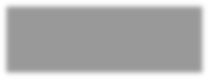
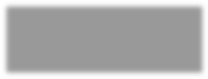
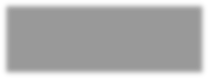
**5. End**

End

*Figure 28: User End Collaboration*

Start

1. **Initiate**



Actor

* 1. Select Prompt to generate Websites

Prompt

* 1. Send Prompt Key to API
  2. Verifying the Prompt Key

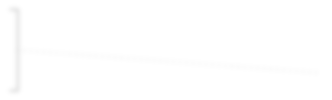
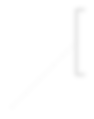
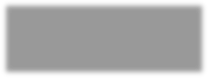
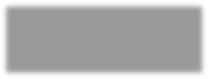
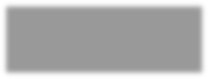
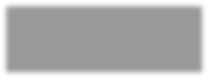
DataSet

1. **End**

End

Start

* 1. **Initiate**



End

Actor

**7. End**

* 1. Requesting for websites

1. Save Code As HTML Format

Coding Page

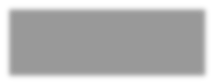
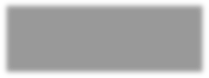
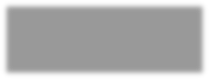
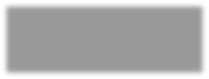
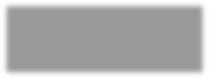
Main

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

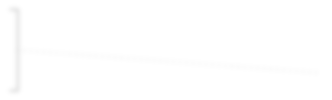
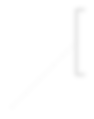
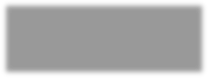
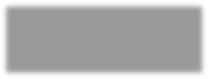
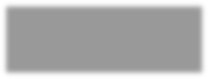
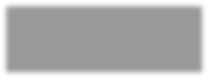
Main

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

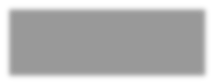
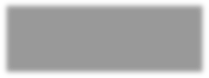
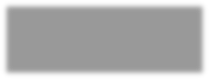
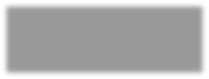
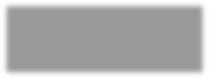
Category

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

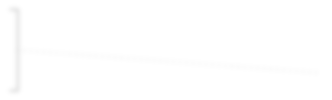
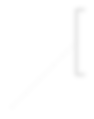
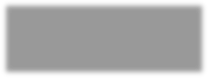
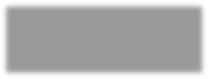
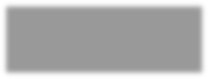
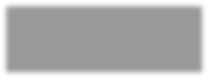
Category

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

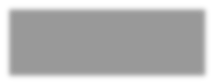
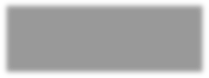
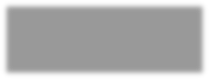
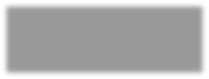
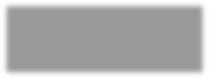
Contact

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

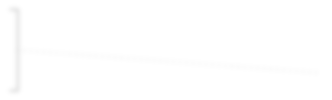
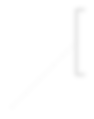
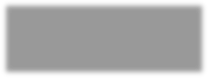
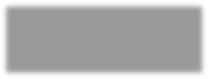
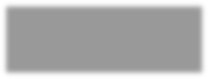
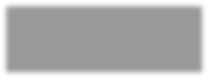
Contact

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

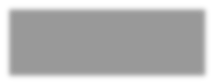
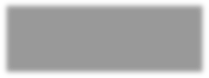
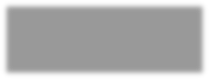
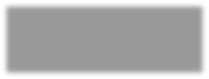
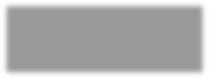
Customer Login

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

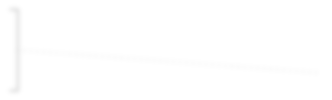
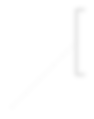
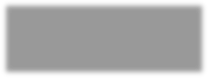
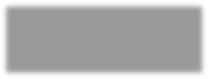
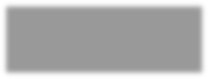
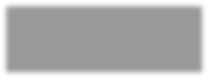
Customer Login

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

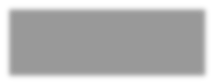
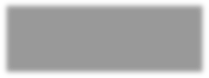
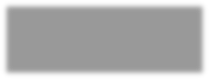
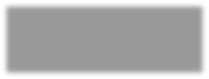
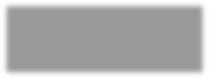
Customer Details

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

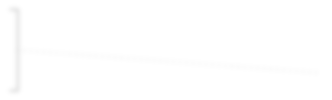
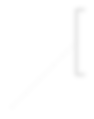
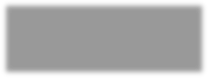
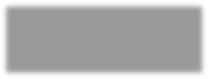
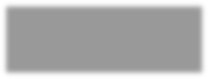
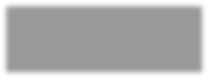
Customer Details

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

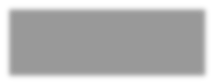
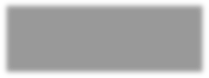
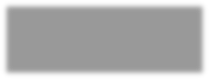
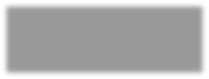
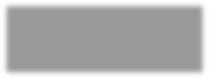
Cart

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

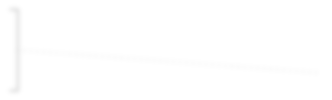
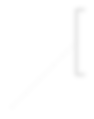
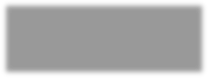
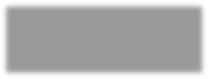
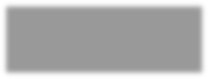
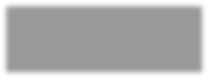
Cart

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

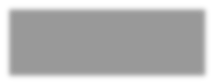
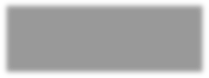
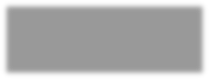
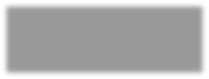
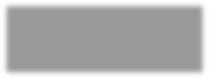
Main Blog

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

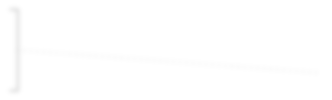
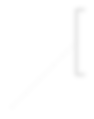
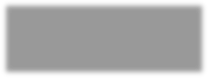
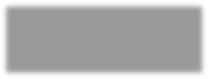
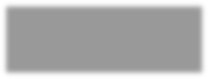
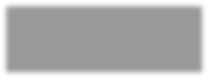
Main Blog

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

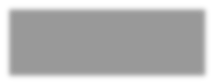
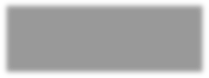
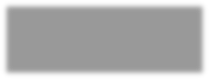
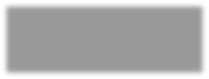
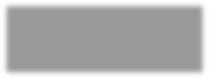
Main Blog

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

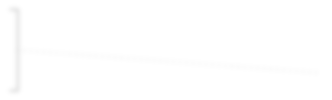
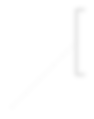
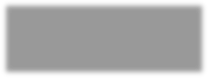
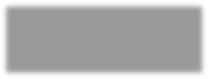
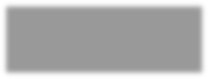
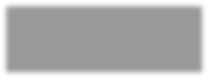
Post Image

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

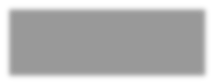
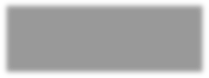
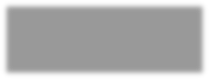
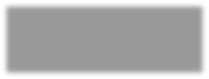
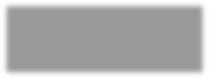
Post Image

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

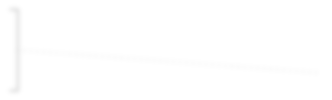
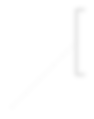
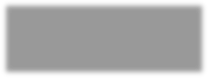
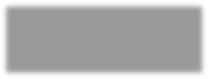
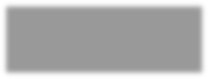
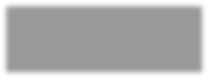
Post Video

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

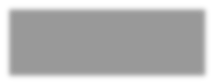
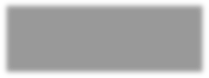
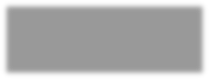
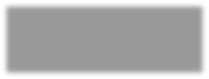
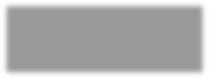
Post Video

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

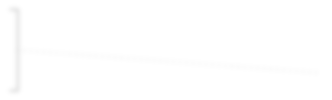
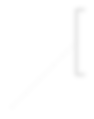
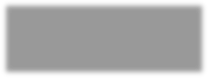
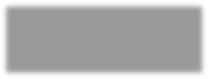
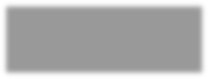
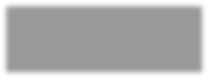
Second Style

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

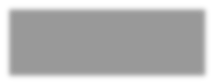
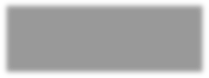
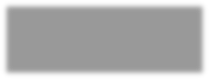
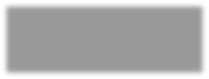
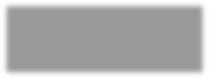
Second Style

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

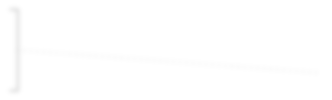
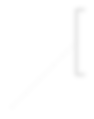
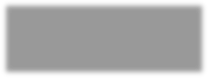
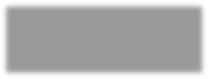
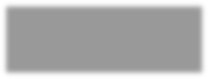
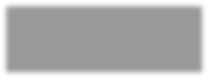
Main Blog

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

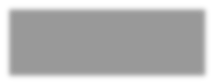
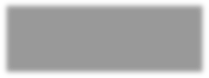
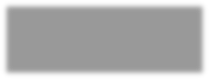
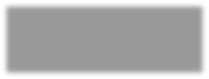
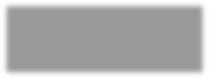
**Main Portfolio**

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

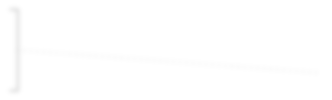
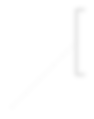
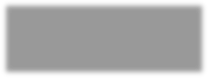
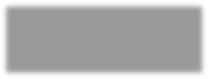
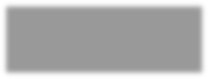
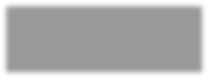
**Main Portfolio**

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

Actor

1. **End**
   1. Requesting for websites
2. Save Code As HTML Format

Coding Page

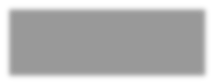
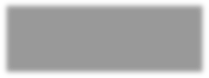
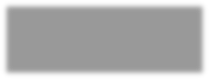
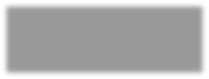
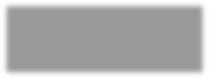
**Projects**

1. Result
2. Request Send
3. Verify All the credential

DataSet

Start

* 1. **Initiate**



End

**7. End**

Actor

1. Show Information
2. Requesting for changing the information of Items

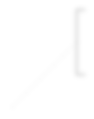
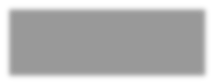
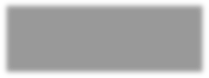
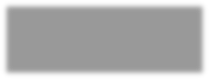
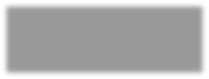
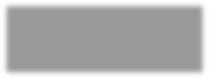
**Projects**

1. Result
2. Request Send
3. Save the HTML Code

Coding Page

Start

* 1. **Initiate**



End

**7. End**

Actor

1. 0 Result
   1. Enter Name
   2. Enter UserName
   3. Enter Password

Sign Up

5.0 Result

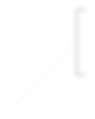
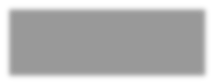
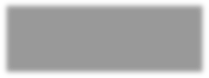
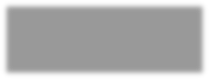
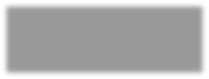
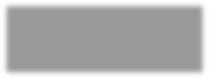
* 1. Store Name
  2. Store UserName
  3. Store Password

4.0 Verify All the credential

Database

Start

**1. Initiate**



End

**7. End**

Actor

6.0 Result

2.0 Enter UserName

2.1 Enter Password

Sign In

5.0 Result

3.0 Check UserName

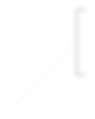
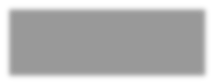
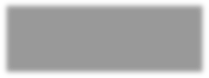
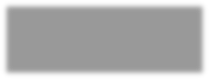
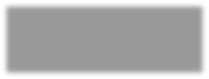
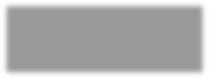
3.2 Check Password

4.0 Verify All the credential

Database

Start

**1. Initiate**



End

**7. End**

Actor

6.0 Send API

2.0 Select Website Type

2.1 Enter Prompt

Sign In

5.0 Result

3.0 Choose API

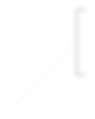
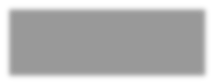
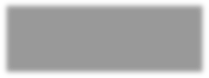
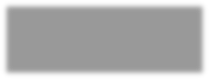
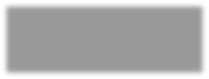
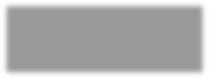
3.2 Analyze Prompt

4.0 Verify All the Analyzed data

API

Start

**1. Initiate**



End

**7. End**

Actor

6.0 Result

2.0 Enter UserName

2.1 Enter Contact Info

Contact Us

5.0 Result

3.0 Check UserName

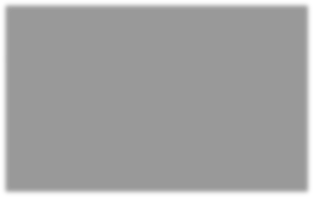
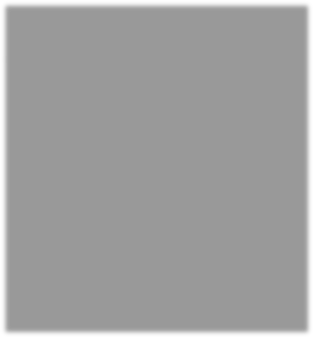
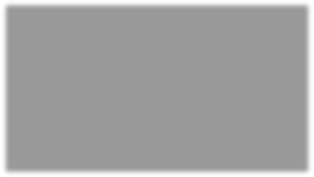
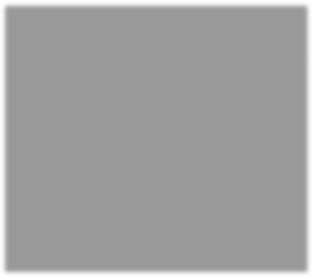
3.2 Check Contact Info

4.0 Store All the credential

Database

# System Structure Design

#### Class Diagram



Admin

1

N

1

-adminID:Int

-adminUsername:String

-email:String

-password:String

-IsLoggedIn:Boolean

+ signUp(): void

+ signIn(): void

+ manageUsers(): void

+ editProfile(): void

+ manageTemplate: void

+ viewPromptsHistory(): void

+ viewContactRequests(): void

+ sendContactResponse(): void

1

Manages

1

Sends

Response

Writes

N

Contact

- contactID: int

N

N

* name: String
* email: String
* message: String
* dateSent: Date
* response: String
* promptID: int
* promptText: String
* promptType: String
* createdBy: User
* dateCreated: Date

WebsitePrompt

-userID:Int

-username:String

-email:String

-password:String

-IsLoggedIn:Boolean

+ signUp(): void

+ signIn(): void

+ editProfile(): void

+ viewPrompts(): void

+ sendContactRequest(): void

User

*Figure 29: Class Diagram*



User

RI7

GrapeJs Editing

Contact

View Features

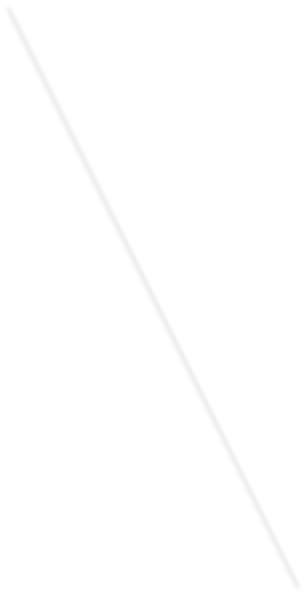
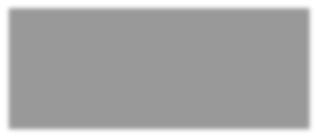
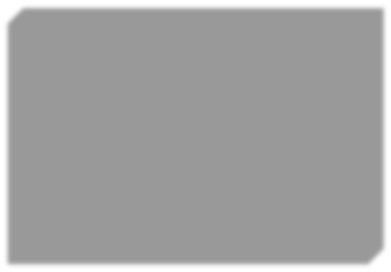
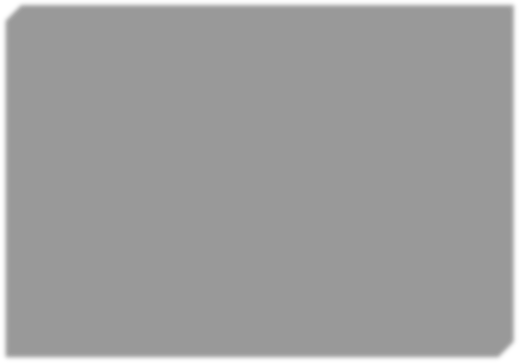
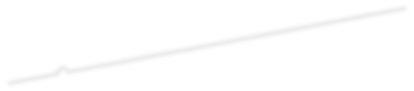
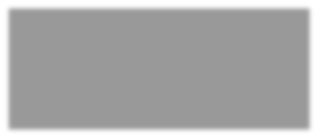
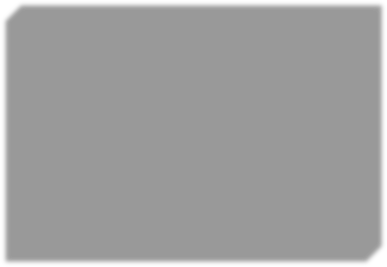
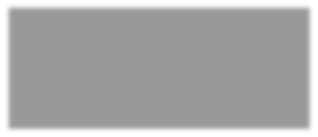
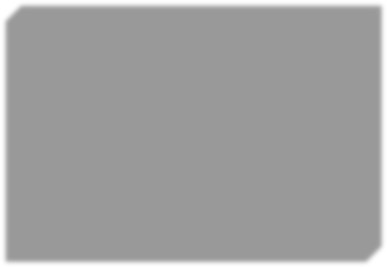
Create Website

Home page

## Component Diagram

*Figure 30: Component Diagram*

## Deployment Diagram



<<device>>

Computer

<<device/environment>>

User Client

<<device>>

Browser

<<Artifact>>

HTML5

<<device>>

Web Python Server

<<Artifact>>

Website

<<Artifact>>

API With Json

*Figure 31: Deployment Diagram*

N

1

N

1

N

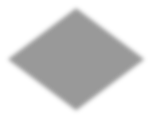
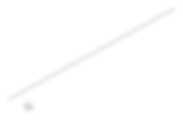
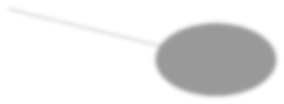
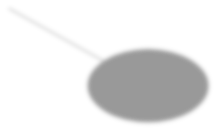
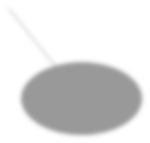
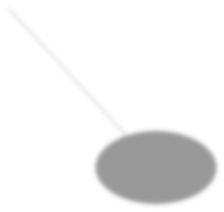
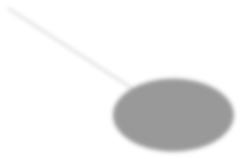
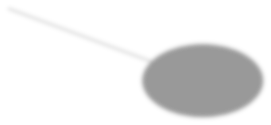
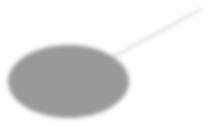
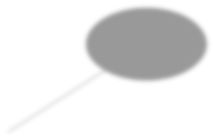
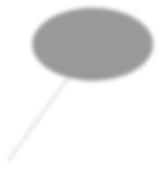
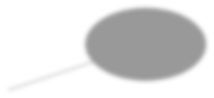
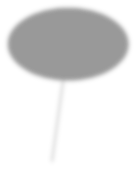
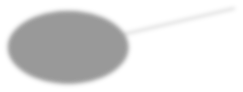
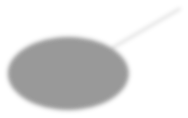
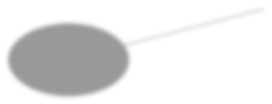
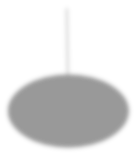
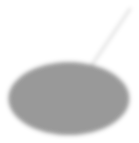
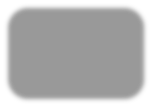
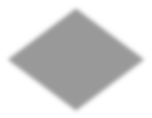
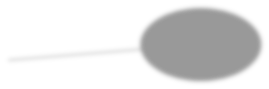
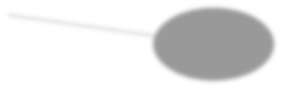
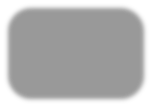
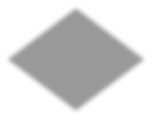
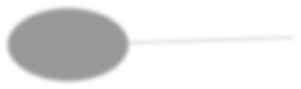
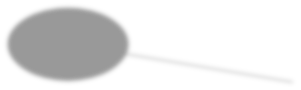
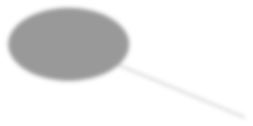
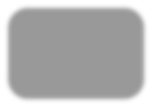
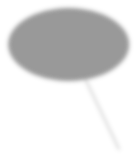
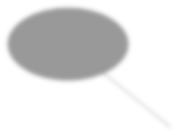
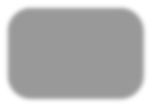
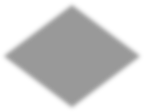
* 1. User Interface Design

#### Wireframes

* 1. Database Design

## ER Diagram

*Figure 32: ER Diagram*



Email

UserName

Password

Admin ID

IsLoggedIn

Manage

Admin

1

User ID

UserName Email

User

Create

Password

IsLoggedIn

Response

ResponseText

Send

PromptText

WebsitePrompt

PromptID

Contact

Created By

Date

Date Created

Response

Message

Email

Prompt Type

User ID

Name

Contact ID