

Design Use Cases

Team

Ur Career Schedule Daily


PROJECT MANAGER:	Chen Xu
SOFTWARE ARCHITECT:	Shaoyuan Xu
BUSINESS ANALYST:	Hao-Tzu Deng
SENIOR SYSTEM ANALYST:	Cong Li
QUALITY ASSURANCE LEAD:	Liu Liu, Jingsong Chen
ALGORITHM SPECIALIST:	Jingsong Chen, Yuhang Jiang
SOFTWARE DEVELOPMENT LEAD:	Yunshu Gao
UI SPECIALIST:	Lu Yu
DATABASE SPECIALIST:	Jingsong Chen, Shih-Han Chan, Jiarong Liu

Table of Contents

Number	Title	Page
User Relatives		
DUC1	Sign Up	4
DUC2	User Creation	6
DUC3	Sign In	8
DUC4	Password Recovery	10
DUC5	Change password	11
DUC6	View User Profile	13
DUC7	Upload Resume	14
DUC8	Edit Profile	15
DUC9	Change Profile Photo	16
DUC10	View Profile Information	17
DUC11	Log out	18
DUC12	Accessing Personal Links (Cancelled)	19
DUC13	Accessing Useful Links	20
DUC14	Calendar Plan	21
Events Relatives		
DUC15	View All Events	22
DUC16	Add Event to Favorite	23
DUC17	Remove Favorite Event	24
DUC18	View Favorite Events	25
DUC19	View Events Information	26
DUC20	Filter by Today (Event)	27
DUC21	Filter by Week (Event)	28

DUC22	Filter by Month(Event)	29
DUC23	Go Events	30
DUC24	Not Going Events	32
Jobs Relatives		
DUC25	View All Jobs	34
DUC26	View Jobs Information	26
DUC27	Add Job to Favourite	36
DUC28	Remove Favorite Job	37
DUC29	View Favorite Jobs	38
DUC30	Apply Jobs	39
DUC31	Search by Keywords	40
DUC32	Filter by Constraints	41
Referral Relatives		
DUC33	Add Referral	43
DUC34	Toggle Referral Opportunity List	46

DESIGN USE CASE #1 - Sign Up

Description	The user signs up for a new account. The user is asked to enter new valid credentials and then initially complete the user profile information.
Desired Outcome	The user has a new account to start using functionalities that are otherwise unavailable. The system saves the user's info and credentials on the database.
User Goal	The user wants to create an account and be able to use functionalities that are not otherwise available.
Dependent Design Case	N/A
Priority	High
Status	Done
Requirements	SR01
Pre-condition	The user is on the homepage with an unauthenticated email that does not associated with any existed account registered already.
Post-condition	The system shall keep the new user logged in. A pop-up window shows up and tells the user to finish the sign-up process by clicking the "Next" button.
Trigger	The user does not have an account yet and wants to create an account.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the homepage of Ur Career Schedule Daily. 2. The user shall click on the "Sign in" button. 3. The frontend shall render signin.HTML and display the sign-in pop-up window to the user. 4. The user shall click " on the top right corner of the pop-up window. 5. The frontend shall render signup.HTML form and display the sign-up pop-up window. 6. The user shall enter a valid email address, password, and confirm password, then click the "REGISTER" button to send a account creation form with POST request. 7. The backend shall create a new account creation form with the request received to validate user's input.

	<ol style="list-style-type: none"> 8. The backend shall create a new user object with the info provided by calling the function <code>create_user(self, email, password)</code> in <code>models.py</code> under Registration application. 9. The backend shall store the user's information into Account model. 10. The frontend shall render <code>successful.HTML</code> and pop up a window to inform the user have signed up successfully.
Alternative Workflow	<p>A text field is left blank:</p> <ol style="list-style-type: none"> 6. The user shall leave any one(s) of the text fields blank. 7. The backend shall catches there is a blank text field. 8. The frontend shall display an error message telling the user to fill up the data field. <p>Invalid email address:</p> <ol style="list-style-type: none"> 6. The user shall enter an invalid email address. 7. The backend shall validate the user's email address. 8. The frontend shall display an error message telling the user to re-enter valid email address. <p>Confirmation password does not match password:</p> <ol style="list-style-type: none"> 6. The user shall enter a password that does not match the entered password. 7. The backend shall catch the error that the password and confirmation password do not match. 8. The frontend shall display an error message telling the user that two passwords do not match.

DESIGN USE CASE #2 - User Creation

Description	The user creates an account in the User model by filling all necessary information and submitting them.
Desired Outcome	The user will be able to create an account after the user fills out the profile form so that the user can use it to get access to our website.
User Goal	The user wants to create an account.
Dependent Design Case	DUC 1
Priority	High
Status	Done
Requirements	SR02
Pre-condition	The user has already logged into his new account and is at the sign_intergrated1.js page which allows the user to fill out the user's basic information.
Post-condition	The user will be redirected to the user's profile page that displays the information the user input before and other useful sections such as "Favorite Events", "Favorite Jobs", etc.
Trigger	The user wants to access more advanced functionalities of the website such as network, user profile, favorite lists, and etc. To do so, the user needs to login to get access of those functions.
Workflow	<ol style="list-style-type: none"> 1. The user passes the signup page of the web-application. 2. The user shall click on "Next" button. 3. The frontend shall render the user_creation.HTML. 4. The user shall fill out the NewUserForm form 5. The user shall click "Submit" button. 6. The backend shall validate the user inputs by calling users(request) in view.py under User application. 7. The backend shall update user's information and store it into the User model as a user object by calling users(request). 8. The frontend shall render profile.HTML and display the user profile page.
Alternative Workflow	<p>No major/degree selected:</p> <ol style="list-style-type: none"> 3. The user shall ignore the major or degree scrolling bar. 4. The user shall click the "Submit" button.

5. The backend shall detect that the major/degree field value are missing by calling users(request).

6. The frontend shall display an appropriate message telling the user select to an option from the drop-down bar for the missing data field.

Missing text field:

3. The user shall leave one or more of the text field blank.

4. The user shall click the "Submit" button.

5. The backend shall catch the error that there is a missing text field by calling users(request).

6. The frontend shall display an appropriate message telling the user to input some content for the missing text field.

Invalid email address:

3. The user shall enter anything but a valid email address.

4. The user shall click the "Submit" button.

5. The backend shall validate the user's email address by calling users(request).

4. The frontend shall display an error message telling the user to enter a valid email address.

DESIGN USE CASE #3 - Sign In

Description	The user signs in with an existing account. The system verifies the email address/password combination and then either logs the user in or displays an error.
Desired Outcome	The user signs in with their account.
User Goal	The user wants to be logged in to use advanced functionalities.
Dependent Design Case	DUC 2
Priority	High
Status	Done
Requirements	SR03
Pre-condition	The user has an registered account already and is currently logged out.
Post-condition	The user signs into their account.
Trigger	The user wants to access more advanced functions of the website such as network, user profile, favorite lists, and etc..
Workflow	<ol style="list-style-type: none"> 1. The user shall be on either one of the main pages of the web-application. 2. The user shall click on the "Sign in" button on the top of the page. 3. The frontend shall render sign in form called LoginForm and display the " LOGIN" pop-up window by rendering redirect_sign_in.HTML. 4. The user shall input the email, password and click the "Login" button. 5. The back-end shall call UserLogin(request) to verify the login credentials in the Account model. 6. The frontend shall pop up a window to inform the user have signed in successfully. 7. The back-end shall set user's status as online while logging in successfully. 8. The user shall click the "OK" button in the pop-up window. 9. The frontend shall close the pop-up window.
Alternative Workflow	<p>Empty password data field:</p> <ol style="list-style-type: none"> 4. The user shall leave the password text field empty. 5. The backend shall detect the user has entered an empty password by calling UserLogin(request).

6. The frontend shall display a message to the user that states "This field is required".

Email is not registered in the Account model database:

4. The user shall enter an email address that is not registered on the system and click the "Login" button.

5. The backend shall fail to authenticate the login email address when calling `UserLogin(request)` because the input email does not in User model.

6. The frontend shall display a message to the user that "Please Register an Account".

Incorrect Email format:

4. The user shall enter incorrect format email address and click the "Login" button.

5. The backend shall fail to authenticate the login when calling `UserLogin(request)` because the `EmailField` in the `model.py` under User Application requires correct email format.

6. The frontend shall display an error message to tell the user that the user has input incorrect email format.

DESIGN USE CASE #4 - Password Recovery

Description	If the user forgets their password, an email containing a link to reset their password will be sent to them
Desired Outcome	The user sets a new password for the account.
User Goal	The user forgets their password but wants to sign into their account.
Dependent Design Case	DUC 1,DUC2, DUC 3
Priority	High
Status	Done
Requirements	SR04
Pre-condition	The user already has an account, and is on the sign in pop-up window, but forgets the corresponding password thus is not yet signed in.
Post-condition	The user is emailed a link to a form to reset the password.
Trigger	The user forgets their password but wants to retrieve their account .
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the sign in pop-up window. 2. The user shall click the "forgot password?" button. 3. The frontend shall render the password_reset_form.HTML and display it to the user. 4. The user shall enter the email that they use to sign in. 5. The backend shall validate the user's email address in Account model by calling UserLogin(request). 6. The backend shall call PasswordRetrievalEmail() to send an email containing a link for the user to change their password. 7. The user shall input a new password for the account and click "Change My Password"
Alternative Workflow	<p>The email entered is invalid:</p> <ol style="list-style-type: none"> 4. The user shall enter anything but an email address. 5. The backend shall validate the user's email. 6. The frontend shall display an error message saying that the input is invalid.

DESIGN USE CASE #5 - Change Password

Description	The user can change their old password and get a new one.
Desired Outcome	The user gets a new password.
User Goal	The user wants to get a new password for their account.
Dependent Design Case	DUC 2
Priority	High
Status	Done
Requirements	SR05
Pre-condition	The user has an account that is already authenticated in our database.
Post-condition	The user's account password is changed to a new one.
Trigger	The user wants to change their password.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the profile page. 2. The user shall click on the "change password" button. 3. The system shall call ChangePassword(Request). 4. The frontend shall render change_password.HTML that requires the user to enter old password and new password and re-confirm the new password. 5. The user shall enter old password and new password and re-confirm the new password entered before. 6. The user shall click on "Save Changes" button. 7. The backend shall validate the user's old password that stored in model.py under Registration Application. 8. The backend shall update the user's password that stored in the model.py under Registration Application.
Alternative Workflow	<p>Wrong old password:</p> <ol style="list-style-type: none"> 4. The user shall input wrong password for the account that is already authenticated in the model.py under Registration Application. 5. The backend shall fail to validate the login request when calling UserLogin(request). 6. The frontend shall display an error message to tell user that he has input "Incorrect password" <p>New password and the confirm new password do not match:</p> <ol style="list-style-type: none"> 4. The user shall input two different new passwords

5. The backend shall find the two input passwords are different with django built-in PasswordResetForm and respond to the frontend with an error.

6. The frontend shall display an error message to tell user that “confirm password is not the same”.

Empty password data field:

4. The user shall leave the password text field empty.

5. The backend catch the incomplete form from the frontend, verify that the password field is empty, and send the error message to the frontend.

6. The frontend shall display an error message indicating the appropriate error.

DESIGN USE CASE #6 - View User Profile

Description	The user can view their user profile page.
Desired Outcome	The user is on the user profile page.
User Goal	The user wants to go to the profile page from any main page of the website.
Dependent Design Case	DUC 1, DUC 2, DUC 3
Priority	High
Status	Done
Requirements	SR06
Pre-condition	The user has an account already and is logged in on the website.
Post-condition	The user is redirected to the profile page
Trigger	The user wants to access their profile page and its functionalities.
Workflow	<ol style="list-style-type: none"> 1. The user shall sign in to their account 2. The user shall be on any one of the main pages of the website. 3. The user shall click the "Hello, your 'user name'" button on the top black bar of the page. 4. The backend shall call profile(request, account_id). 5. The frontend shall render and display profile.HTML.
Alternative Workflow	N/A

DESIGN USE CASE #7 - Upload Resume

Description	The user wants to upload their resume to the website
Desired Outcome	The user shall be able to upload the resume to the website
User Goal	The user wants to upload their resume.
Dependent Design Case	DUC 1, DUC 2, DUC 3
Priority	Low
Status	Cancelled
Requirements	SR07
Pre-condition	The user has an account already and is currently signed into the his own account.
Post-condition	The user's resume is stored into our database.
Trigger	The user wants to upload their resume to the website.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on their profile page. 2. The user shall click on the "Upload Resume" button located on the left side of the page. 3. The frontend shall display a pop-up window showing all the user's available files under their current local directory. 4. The user shall select the resume file on their system and click the "submit" button. 5. The system shall upload the file 6. The backend shall store the file in the database. 7. The frontend shall close the pop-up window and inform the user that the file is uploaded successfully.
Alternative Workflow	N/A

DESIGN USE CASE #8 - Edit Profile

Description	The user can change their profile information except their last and first name that they entered during the user creation process.
Desired Outcome	The user updates the profile.
User Goal	The user wants to edit the user profile.
Dependent Design Case	DUC 1, DUC 2, DUC 3, DUC 6
Priority	High
Status	Done
Requirements	SR08
Pre-condition	The user has an account already and is currently signed in.
Post-condition	The user's profile information is changed and stored to the model.py under User Application.
Trigger	The user wants to update new information that displayed on profile page
Workflow	<ol style="list-style-type: none"> 1. The frontend shall render profile.HTML 2. The user shall click on the "Edit Profile" button. 3. The backend shall call edit_profile(request). 4. The frontend shall render profile.HTML to the user for edition. 5. The system shall display an editable user profile information section. 6. The user shall input the information that they want to change into the corresponding box. 7. The user shall click on the "Save Changes" button. 8. The backend shall store the user's new information into model.py under User Application. 9. The frontend shall display the newly modified information by rendering profile.HTML.
Alternative Workflow	Cancel: <ol style="list-style-type: none"> 5. The user shall click on the "Cancel" button. 6. The system shall display the unmodified user information. 7. The system shall render profile.HTML to the user.

DESIGN USE CASE #9 - Change Profile Photo

Description	The user can change their profile photo.
Desired Outcome	The user's profile photo is changed.
User Goal	The user wants to change profile photo.
Dependent Design Case	DUC 6
Priority	Low
Status	Cancelled
Requirements	SR09
Pre-condition	The user has an account already and is currently signed in.
Post-condition	The user's profile photo is changed.
Trigger	The user wants a new profile photo.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on their profile page. 2. The user shall click on the "Change Photo" tab. 3. The frontend shall display a pop-up window showing the user all available photos under their current local directory. 4. The user shall select the profile photo in their system and click the "change" button. 5. The system shall have upload the photo. 6. The frontend shall close the pop-up window and inform the user that the photo is uploaded successfully. 7. The backend shall store the photo and switch the current profile photo to the updated one.
Alternative Workflow	None

DESIGN USE CASE #10 - View Profile Information

Description	The user can view their profile information.
Desired Outcome	The user views their profile information.
User Goal	The user wants to see their profile information.
Dependent Design Case	DUC6
Priority	High
Status	Done
Requirements	SR06
Pre-condition	The user has an account already and is currently signed in.
Post-condition	The system displays the user's profile information on the screen.
Trigger	The user wants to view their profile information.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the profile page. 2. The frontend shall render profile.html to the user containing the user's profile information. 3. The backend shall call profile(request, account_id) to get the user's information from the model.py under User Application. 4. The system shall display the user's profile information.
Alternative Workflow	None

DESIGN USE CASE #11 - Log Out

Description	The user can log out of their account.
Desired Outcome	The user is successfully logged out of their account.
User Goal	The user wants to log out of their account.
Dependent Design Case	DUC3
Priority	High
Status	Done
Requirements	SR 11.
Pre-condition	The user has an account and has already logged in.
Post-condition	The user is logged out of their account.
Trigger	The user wants to log out of their account.
Workflow	<ol style="list-style-type: none"> 1. The user shall click on the "Logout" button. 2. The back-end shall call UserLogin(request). 3. The frontend shall display a pop-up window briefly with a message indicating the successful logout. 4. The frontend shall close the pop-up window.
Alternative Workflow	<p>Logging out on the profile page:</p> <ol style="list-style-type: none"> 2. The frontend shall refresh the current window with a message indicating the successful log out briefly. 3. The frontend shall render index.html and redirect the user back to the home page.

DESIGN USE CASE # 12 - Accessing Personal Links

Description	The user can access their personal links through their profile page.
Desired Outcome	The user wants to access their personal links.
User Goal	The user is directed to their personal link.
Dependent Design Case	DUC 6
Priority	Medium
Status	Cancelled
Requirements	SR12
Pre-condition	The user has an account already and is currently signed in.
Post-condition	The system opens another window for the user's personal link.
Trigger	The user wants to browse their personal links.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on their profile page. 2. The user shall click on one of their personal links under the "Personal LINK" section. 3. The system shall open another window with the corresponding personal link.
Alternative Workflow	N/A


DESIGN USE CASE # 13 - Accessing Useful Links

Description	The user can access their personal links through their profile page.
Desired Outcome	The user is directed to another useful website that they wants to go.
User Goal	The user wants to go to other useful websites listed on the left side of the profile page.
Dependent Design Case	DUC6
Priority	Medium
Status	In progress
Requirements	SR13
Pre-condition	The user has an account already and is currently signed in.
Post-condition	The system opens another window for the user's useful link.
Trigger	The user wants to see one of the useful websites that listed on the left side of their profile page.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on their profile page. 2. The user shall click on one of their useful links under the "Useful Links" section on the left of the profile page. 3. The backend shall use the newuser_page() function to render the profile.htm to the frontend. The system shall open another window with the corresponding useful link.
Alternative Workflow	N/A


DESIGN USE CASE # 14 - Calendar Plan

Description	The user can see the events that they decides to go on the calendar when the time reaches the date of that event.
Desired Outcome	The user could use the calendar functionality to remind of themselves to go to the events.
User Goal	The user wants to be notified by the calendar when the event date comes.
Dependent Design Case	DUC 6
Priority	Medium
Status	Under development.
Requirements	SR14
Pre-condition	The user is on the profile page.
Post-condition	Calendar is displayed on the right of the profile page
Trigger	The user wants to make sure they are attending the events on time.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the profile page 2. The user shall click
Alternative Workflow	



DESIGN USE CASE #15 - View All Events

Description	The user can view a gallery of career related event summaries on the system.
Desired Outcome	The user sees a list of overviews of the events.
User Goal	The user wants to have a clear overview of the events.
Dependent Design Case	DUC 3
Priority	High
Status	Done
Requirements	SR15
Pre-condition	The user is on the homepage.
Post-condition	The full list of events is displayed to the user.
Trigger	The user wants to view all available events.
Workflow	<ol style="list-style-type: none"> 1. The user shall open their browser and navigate to UCSD's home page. 2. The frontend shall render index.js 3. The user shall click the “” on the upper right corner. 4. The user shall click on “Events” button. 5. The frontend shall render events.HTML and display a list of all available events sorted by date, from newest to oldest.
Alternative Workflow	N/A

DESIGN USE CASE #16 - Add Events to Favorite

Description	The user can add favorite events(s) from the all event page.
Desired Outcome	The user saves the events in interest and has quick access to them through the favorite event list.
User Goal	The user wants to save the events they are interested in as favorite events.
Dependent Design Case	DUC15
Priority	High
Status	Done
Requirements	SR16
Pre-condition	The user is currently signed in and is on the all events page.
Post-condition	The user's favorite events list is modified and the change is stored in the model.py under Event Application.
Trigger	The user wants to add events to the favorite events list.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the events page 2. The user shall click “  ” on the upper-right corner of the event card. 3. The backend shall call the method <code>add_to_favorite(request)</code> and store the changes to the model.py under Event Application.
Alternative Workflow	N/A

DESIGN USE CASE #17 - Remove Favorite Event

Description	The user can remove event(s) from their favorite event list.
Desired Outcome	The user removes an event or events from their favorite event list.
User Goal	The user wants to remove event(s) from their favorite event list.
Dependent Design Case	DUC 6, DUC15
Priority	High
Status	Done
Requirements	SR17
Pre-condition	The user has an account already and is currently signed in.
Post-condition	The user's favorite event list is modified and the change is stored in the model.py under Event Application.
Trigger	The user wants to remove events from their favorite event list.
Workflow	<ol style="list-style-type: none"> 1.The user shall be on the profile page. 2.The user shall click the "Favorite Events" tab. 3.The backend shall call <code>get_favorite_status(self)</code> to pull the data from Event model that store all the information about event. 4.The user can view all the favorite events added before. 5.The user shall click on the "x" button on the top right corner of the event that they want to remove from the list. 6.The backend shall remove that event from the user's favorite event list and store the changes to the Event/model.py. 7.The frontend shall display the newly modified list.
Alternative Workflow	<ol style="list-style-type: none"> 4. The user shall click on the  of the event(s) the user wants to remove at the all events page. 5. The backend shall call <code>remove_favents(self)</code> and pull the data from model.py under Event Application. 5. The frontend shall display the start icon becomes  which indicates the event has been removed from the favorite list.

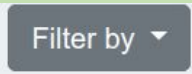
DESIGN USE CASE #18 - View Favorite Events

Description	The user can view their favorite event list.
Desired Outcome	The user can view all events that saved in favourite page.
User Goal	The user wants to view their favorite events.
Dependent Design Case	DUC 6
Priority	High
Status	Done
Requirements	SR18
Pre-condition	The user has an account already and is currently signed in.
Post-condition	The user's favorite events are displayed.
Trigger	The user wants to see or change their favorite event list.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on their profile page. 2. The user shall click on the "Favorite Events" tab. 3. The backend shall call get_favevents(self) and pull the data from the User model. 4. The user can view all the favorite events added before.
Alternative Workflow	N/A

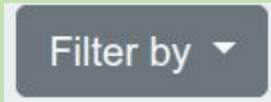
DESIGN USE CASE # 19 - View Events Information

Description	The user clicks on the title of the event in the event card and the user sees the detailed description of the selected event from Event model.
Desired Outcome	The user will be able to see a detailed description after the user clicks on the title of the event.
User Goal	The user wants to see a detailed description of certain events.
Dependent Design Case	DUC 18
Priority	High
Status	Done
Requirements	SR19
Pre-condition	The user is on the event page.
Post-condition	The user is able to see detailed descriptions of events pulled from the model.py under Event Application.
Trigger	The user wants to check more information on the event before the user decides whether to go to the event.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the event page. 2. The user shall click on the title of an event on the event card. 3. The frontend shall render a http response which pop out a window contains event detailed information.
Alternative Workflow	None

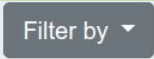
DESIGN USE CASE #20 - Filter by Today (Event)

Description	The user can filter the events in the system the events take place today.
Desired Outcome	The user gets all the events in the system that take place today.
User Goal	The user wants to view only the events that take place in a specific time period.
Dependent Design Case	DUC 15
Priority	High
Status	Done
Requirements	SR 20
Pre-condition	The user is on the events page.
Post-condition	The user is able to see a list of events that are filtered by today only and the list of events are pulled from the model.py under Event Application.
Trigger	The user wants to choose the date range of the events which are available for the user to choose so that the user does not need to go through a whole list of events.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the event page. 2. The user shall click this “  ” button on the upper-right. 3. The user shall choose a filter on the list which the filter is by today. 4. The backend shall call the function <code>get_queryset(self)</code> in <code>EventListViewToday(ListView)</code> 5. The frontend shall display the list of events automatically.
Alternative Workflow	None

DESIGN USE CASE #21 - Filter by Week (Event)

Description	The user can filter the events in the system the events take place this week
Desired Outcome	The user gets all the events in the system that take place this week.
User Goal	The user wants to see the events that are happening this week.
Dependent Design Case	DUC 15
Priority	High
Status	Done
Requirements	SR21
Pre-condition	The user is on the event page.
Post-condition	The user is able to see a list of events that are filtered by the current week only and the list of events are pulled from the model.py under Event Application.
Trigger	The user wants to choose the date range of the events which are available for the user to choose so that the user does not need to go through the whole list.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the event page. 2. The user shall click this “” button on the upper-right. 3. The user shall choose a filter on the list which the filter is by week. 4. The backend shall call the function <code>get_queryset(self)</code> in <code>EventListViewWeek(ListView)</code> 5. The frontend shall display the list of events filter by this week automatically.
Alternative Workflow	None

DESIGN USE CASE #22 - Filter by Month (Event)

Description	The user can filter the events in the system the events take place during this month.
Desired Outcome	The user gets all the events in the system that take place this month.
User Goal	The user wants to see the events that are happening this month.
Dependent Design Case	DUC15
Priority	High
Status	Done
Requirements	SR22
Pre-condition	The user is on the event page.
Post-condition	The user is able to see a list of events that are filtered by the current month only and the list of events are pulled from the model.py under Event Application.
Trigger	The user wants to choose the date range of the events which are available for the user to choose so that the user does not need to go through the whole list.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the event page. 2. The user shall click this “  ” button on the upper-right. 3. The user shall choose a filter on the list which the filter is by month. 4. The backend shall call the function <code>get_queryset(self)</code> in <code>EventListViewMonth(ListView)</code> 5. The frontend shall display the list of events filter by this month automatically.
Alternative Workflow	None

DESIGN USE CASE #23 - Go Events

Description	The user can click “go” button at the bottom of event card to remind them to go to the event.
Desired Outcome	The user marks the interesting events as to go.
User Goal	The user wants to register for the events in interest.
Dependent Design Case	DUC15, DUC18
Priority	Medium
Status	Done
Requirements	SR23
Pre-condition	The user shall have an account and has already logged into the website. The user clicks the “go” button at the bottom of the event card.
Post-condition	The “go” button changes to “not going”.
Trigger	The user wants to go to some events and wants the website to remind them.
Workflow	<ol style="list-style-type: none"> 1. The user shall have an account for the website. 2. The user shall go to the event page. 3. The frontend shall render the http of event page. 4. The user shall click on the “go” button at the bottom of the event card. 5. The backend shall call add_to_favorite function to add the event into user’s favorite events. 6. The backend shall call go_to_event to set the status to true. 7. The frontend shall display the event that user wants to go when it comes to the day that event holds.
Alternative Workflow	<ol style="list-style-type: none"> 1. Click “go” button from the user’s profile page. 2. The user shall on the profile page. 3. The user shall click the “Favorite Events” tab. 4. The frontend shall render http response to show “Favorite Events”. 5. The user shall click on the “go” button at the bottom of the event card. 6. The frontend shall provides the button of function to call the function in the backend.

7. The backend shall call `go_to_event` to set the status to true.
8. The user shall refresh the page.
9. The frontend shall display the event that user wants to go when it comes to the day that events holds.

DESIGN USE CASE #24 - Not Going Events

Description	The user can make an event as not going and take off the profile page calendar.
Desired Outcome	The user marks a going event as not going.
User Goal	The user wants to cancel going to an event that is previously planned as going.
Dependent Design Case	DUC15, DUC18, DUC23
Priority	Medium
Status	Done
Requirements	SR24
Pre-condition	The user shall have an account and has already logged into the website. The user shall already clicked "Go" at the bottom of the event card. The user clicks "Not Going" at the bottom of the event card.
Post-condition	The button at the bottom of the event card will be reset to "Go".
Trigger	The user does not have time or does not want to go to the event anymore, and the user does not want the system to remind the user regarding the event.
Workflow	<ol style="list-style-type: none"> 1. The user shall have an account for the website. 2. The user shall be on the event page of the website. 3. The frontend shall render the events.HTML of the website. 4. The user shall click on the "Not Going" button at the bottom of the event card. 5. The backend shall call <code>add_to_favorite(request)</code> to delete the events from the user's favorite events. 6. The backend shall call <code>go_to_event</code> to set the status to be false. 7. The frontend shall delete the event from "Favorite Event" section under profile.HTML.
Alternative Workflow	<ol style="list-style-type: none"> Click "Not going" button from the user's profile page. 2. The user shall be on the profile page. 3. The user shall click the "Favorite Events" tab. 4. The frontend shall render http response to show "Favorite Events". 5. The user shall click on the "Not Going" button at the bottom of the event card.


6. The frontend shall provides the button of function to call the function in the backend.

7. The backend shall call `go_to_event(self,user)` to set the status to false.

8. The user shall refresh the page.

9. The frontend shall remove the events from the reminder system.


DESIGN USE CASE #25 - View All Job

Description	The user can view a gallery of job summaries on the system.
Desired Outcome	The user sees a list of overviews of the jobs.
User Goal	The user wants to view jobs provided by the website.
Dependent Design Case	N/A
Priority	High
Status	Done
Requirements	SR25
Pre-condition	The user is on the homepage.
Post-condition	The user is directed to the search job page.
Trigger	The user visits the website and wants to start searching for jobs are available on the website.
Workflow	<ol style="list-style-type: none"> 1. The user shall open their browser and navigate to UCSD's home page. 2. The user shall click the “” on the upper right corner. 3. The user shall click on “Jobs” button. 4. The backend shall call the JobDefault as the view of the jobs list for the frontend to display all the jobs information from the model.py under Job Application. 5. The frontend shall render jobs.HTML and display a list of all available jobs.
Alternative Workflow	None


DESIGN USE CASE #26 - View Job Information

Description	The user wants to view specific information regarding the job.
Desired Outcome	The user will be able to see the detailed information from database regarding the certain jobs.
User Goal	The user wants to see detailed information regarding the job.
Dependent Design Case	DUC 25
Priority	High
Status	Done.
Requirements	SR26
Pre-condition	The user is on the job page.
Post-condition	The detailed information of the job is accessed from model.py under Job Application.
Trigger	The user wants to see more information on the jobs listed that might be their interests.
Workflow	<ol style="list-style-type: none"> 1.The user shall be on the job page. 2. The user shall click on the title of the job in some certain job cards. 3. The frontend shall render jobs.HTML and display the specific job page. 4. The backend shall call <code>get_query_set(self)</code> to display the information for the job. 5. The user shall see the detailed information of the job pulled out from the database.
Alternative Workflow	2. The user shall see the description below the job title in the job card.

DESIGN USE CASE #27 - Add Job to Favorite

Description	The user adds some certain jobs to be the favorite jobs.
Desired Outcome	The user will have a list of favorite jobs under “Favorite job” section in the user’s profile page.
User Goal	The user is able to add the user’s favorite jobs to favorite.
Dependent Design Case	DUC25
Priority	High
Status	Done.
Requirements	SR27.
Pre-condition	The user needs to sign in to the web-application and the user is on the job page.
Post-condition	The user’s favorite jobs are added to “Favorite job” and are stored in model.py under Job Application.
Trigger	The user wants to add the user’s favorite jobs to the favorite list so that next time the user can just go to “Favorite jobs” to view the jobs and be convenient.
Workflow	<ol style="list-style-type: none"> 1.The user shall sign in to their account. 2. The user shall be on the job page. 3. The user shall click “” on the upper-right corner of the job card. 4.The frontend shall render jobs.HTML with the empty star being filled. 5.The backend shall call add_to_favorite(request) to add the job to favorite and store the job in the database. 6. The user shall see the star being filled once the job is added to favorite. 7.The system shall display the certain job added to the favorite jobs in the user’s “Favorite Jobs” section and store it into the model.py under Job Application.
Alternative Workflow	None.

DESIGN USE CASE #28 - Remove Favorite Job

Description	The user can delete the favorite jobs from the favorite job list.
Desired Outcome	The user deletes some jobs from the favorite job list.
User Goal	The user wants to delete some saved jobs.
Dependent Design Case	DUC 25, DUC 29
Priority	High
Status	Done
Requirements	SR28
Pre-condition	The user has an account already and is currently signed in.
Post-condition	The user's favorite job list is modified and the change is stored in the model.py under Job Application.
Trigger	The user wants to remove events from their favorite event list.
Workflow	<ol style="list-style-type: none"> 1.The user shall be on the profile page. 2.The user shall click the "Favorite Jobs" tab. 3.The backend shall call get_favjobs(self) and pull the data from model.py under Job Application. 4.The user can view all the favorite jobs added before. 5.The user shall click on the "x" button on the top right corner of the job that they want to remove from the list. 6.The backend shall remove that job from the user's favorite job list and store the changes into the model. 7.The frontend shall display the newly modified list.
Alternative Workflow	<ol style="list-style-type: none"> 2. The user shall be on the view all jobs page 3. The user shall click on the  of the the job(s) the user wants to remove at the all jobs page.

DESIGN USE CASE #29 - View Favorite Jobs

Description	The user can view their favorite job list.
Desired Outcome	The user can view all jobs that saved in favourite page.
User Goal	The user wants to view their favorite jobs.
Dependent Design Case	DUC1, DUC2, DUC3, DUC6
Priority	High
Status	Done
Requirements	SR29
Pre-condition	The user has an account already and is currently signed in.
Post-condition	The user's favorite jobs are displayed.
Trigger	The user wants to see or change their favorite job list.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on their profile page. 2. The user shall click on the "Favorite Jobs" tab. 3. The backend shall call get_favjobs(self) and pull the data from database. 4. The user can view all the favorite jobs added before.
Alternative Workflow	N/A

DESIGN USE CASE # 30 - Apply Jobs

Description	The user can apply for the jobs on the job providers' official career website.
Desired Outcome	The user is brought to the job provider's official webpage.
User Goal	The user wants to apply for the jobs they are interested in by going to the job provider's official webpage.
Dependent Design Case	None
Priority	High
Status	Done.
Requirements	SR30
Pre-condition	The user is on the all jobs page or the favorite jobs section
Post-condition	The user is on the job application page
Trigger	The user wants to apply for the certain job
Workflow	<ol style="list-style-type: none"> 1.The user shall be on the job page that lists all the jobs. 2. The user shall click on the “Apply” button. 3. The frontend shall redirect the user to the website for the job application.
Alternative Workflow	None


DESIGN USE CASE #31 - Search by keywords(Jobs)

Description	The user input keywords in the search bar and click “search”. Database will pull valid jobs that match the keywords inputted.
Desired Outcome	The user will get a list of jobs that match the keywords the user inputted.
User Goal	The user wants to search jobs by keywords.
Dependent Design Case	DUC25
Priority	High
Status	Done.
Requirements	SR31
Pre-condition	The user is on job page.
Post-condition	The desired jobs with keyword input by user are pulled from the database.
Trigger	The user wants to access more targeted jobs from the long list of jobs and wants to access more advanced functions in the job page.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on the job page. 2. The user shall enter the keywords in the search bar. 3. The user shall choose whether the user wants to have a desired location for the jobs. 4. The user shall click on “search” beside the search bar. 5. The backend shall call the function <code>search_By_Keywords(keywords)</code> in <code>models.py</code> and shall show all jobs filter by the given keyword. 6. The frontend shall display a list of jobs that contain the keyword that user input.
Alternative Workflow	N/A

DESIGN USE CASE #32 - Filter by Constraints

Description	The user filters the job list by constraints such as degree, work duration, work authorization etc.
Desired Outcome	The user sees a list of filtered jobs that have all the information suitable for the user.
User Goal	The user wants to filter the jobs based on the constraints that they will apply.
Dependent Design Case	DUC25
Priority	High
Status	Done
Requirements	SR32
Pre-condition	The user logs into the website and is on the job page.
Post-condition	The user can view all the relevant jobs from the database based on the constraints the user selected.
Trigger	The user wants to see their suitable jobs based on the filters instead of a long list of random jobs.
Workflow	<ol style="list-style-type: none"> 1. The user shall login into the web application. 2. The user shall be on the job page. 3. The user shall select all the filters they want to apply, such as work duration, degree, and work authorization, on the sidebar. 4. The backend shall filter jobs from the model.py under Job Application by calling <code>get_queryset(self)</code> from JobSearch class. 5. The user shall see a list of filtered jobs pulled from the model.py under Job Application.
Alternative Workflow	N/A

DESIGN USE CASE # 33 - Add Referral

Description	The user can add their own referral opportunity under their profile page.
Desired Outcome	The user has a new referral listing in their "My Referral" tab.
User Goal	The user wants to add a referral opportunity to the website.
Dependent Design Case	DUC1, DUC2, DUC3, DUC6
Priority	High
Status	Done
Requirements	SR34
Pre-condition	The user has an account already and is currently signed in.
Post-condition	A new referral listing is added under the user's "My Referral" tab and the data is stored in the database.
Trigger	The user has a referral opportunity and they want to share it on the website.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on their profile page. 2. The user shall click the "My Referral" tab. 3. The frontend shall render the referral page and display the referral list of the user. 4. The user shall click on the "  " button under the "My Referral" tab. 5. The backend shall call the function <code>add_referral(request)</code> in <code>views.py</code> 6. The frontend shall render <code>add_referral.HTML</code> and display the "Offer Your Referral" pop-up window. 7. The user shall enter the title, location, company, major and degree required for the job. 8. The user shall click the "NEXT" button. 9. The backend shall validate the user inputs. 10. The user shall pick at least one from the categories list and provide a brief job description with minimum 20 characters in length. 11. The user shall click the "NEXT" button. 12. The backend shall validate the user inputs. 13. The user shall leave the "Do you require a resume?" unchecked.

	<ol style="list-style-type: none"> 14. The user shall provide some additional job information with minimum 20 characters in length. 15. The user shall click the "SUBMIT" button. 16. The backend shall validate the user inputs. 17. The frontend shall display a success message in the pop-up window telling the user that their referral opportunity has been added. 18. The user shall click the "NO THANKS" button. 19. The frontend shall close the pop-up window.
Alternative Workflow	<p>Incorrect/missing data field during step 1:</p> <ol style="list-style-type: none"> 6. The user shall enter invalid inputs in any one or more of the data field except the title field. 7. The user shall click the "NEXT" button. 8. The backend shall catch that the data fields are not complete. 9. The frontend shall display error message(s) indicating the appropriate incorrect/missing text field(s) that needs to be filled <p>Missing category selection(s) during step 2:</p> <ol style="list-style-type: none"> 9. The user shall pick nothing from the category list. 10. The user shall click the "NEXT" button. 11. The backend shall catch the missing category selection(s). 12. The frontend shall display an error message telling the user to pick at least one category. <p>Missing job description during step 2:</p> <ol style="list-style-type: none"> 9. The user shall leave the job description empty. 10. The user shall click the "NEXT" button. 11. The backend shall catch the missing job description 12. The frontend shall display an error message telling the user to provide a valid job description. <p>Require resume submission:</p> <ol style="list-style-type: none"> 11. The user shall check the "Do you require resume?" Option. <p>Missing additional information:</p> <ol style="list-style-type: none"> 13. The user shall leave the "ADDITIONAL INFORMATION" empty. 14. The user shall click the "SUBMIT" button.

15. The backend shall check if there is any missing additional information

16. The frontend shall display an error message indicating the missing data field.

Offer another referral opportunity at the end:

17. The user shall click the "OFFER ANOTHER REFERRAL" button.

18. The system shall take the user back to workflow #6.

DESIGN USE CASE # 34 - Toggle Referral Opportunity List

Description	The user can make their currently available referral opportunities no longer available or unavailable ones available again.
Desired Outcome	The user's referral opportunity status is changed.
User Goal	The user wants to change the status of their referral listing(s).
Dependent Design Case	DUC1, DUC2, DUC2, DUC6
Priority	High
Status	Done
Requirements	SR35
Pre-condition	The user has an account already and is currently signed in.
Post-condition	The status of the user's referral opportunity is updated.
Trigger	The user wants to show/hide their referral opportunity.
Workflow	<ol style="list-style-type: none"> 1. The user shall be on their profile page. 2. The user shall click on the "My Referral" tab. 3. The frontend shall render the referral page and display the referral list of the user. 4. The user shall click on the "No Longer Available" button of the referral opportunity they want to hide. 5. The backend shall call <code>toggle_activate_referral</code> and store the change to the database. 6. The frontend shall turn that referral opportunity entry grey.
Alternative Workflow	<p>Make the job opportunity available again:</p> <ol style="list-style-type: none"> 4. The user shall click on the "Available Again" button on the greyed-out referral opportunity. 5. The backend shall store the change to the model.py under Job Application. 6. The frontend shall restore the color of that particular referral entry.