ASTL: A Simple TODO List

Name: Hongyan Gu

UCLA ID: 205025476

Contents

Introduction

ATDL: A Simple TODO List is a simple TODO List management application which runs on web browsers. The front end uses Bootstrap and the backend uses Django. With the aid of application, the user can add event, manage undone event, see missed event and mark event as urgent. All data are stored in a Database and could be retrieved after the browser is closed. User can also migrate the database to other applications without losing the data.

Requirements

Required libraries/env:

- `Python3`
- `Django 2.2`
- `Bootstrap 4 (by CDN)`

Run project (choose an available port)

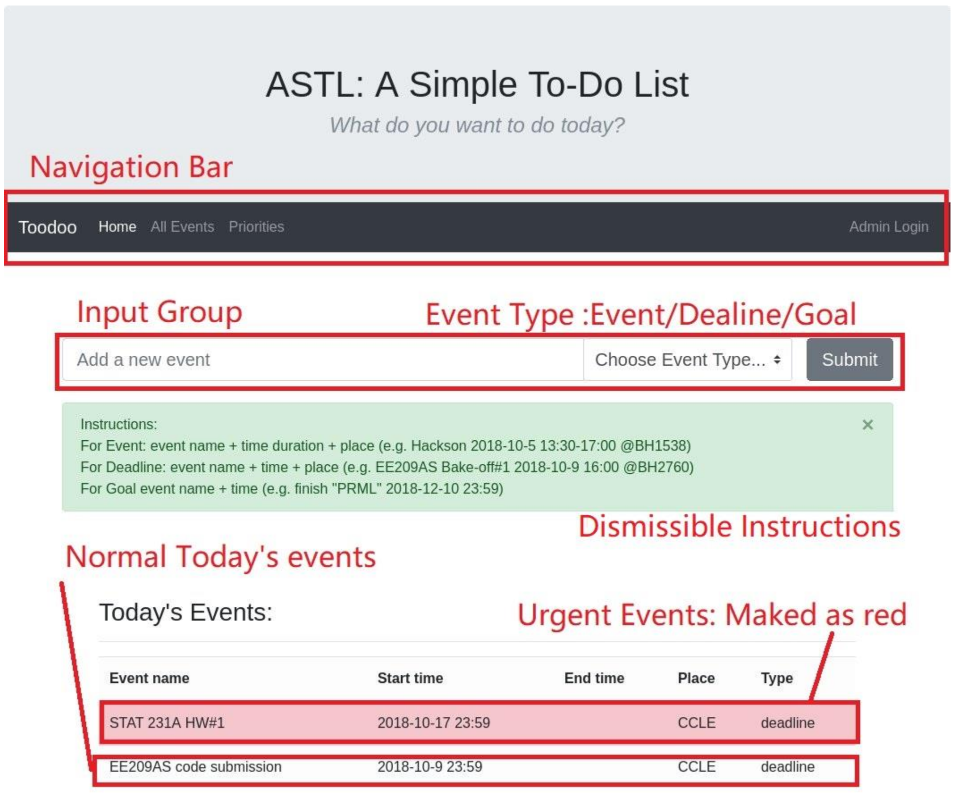
- `cd ./foo/atsl`
- `python manage.py runserver 0.0.0.0:24445`
- `Go to browser, and visit http://127.0.0.1:24445`

Front end and Usages

The Front end uses Bootstrap 4. The scripts are cited by CDN so the app should be linked to a valid internet. There are three parts/applications matched with applications in Django (see Part 3).

Home Page

Home page (index.html) contains Navigation Bar, Input Group, Instructions, Messages, Today's events. The layout of home page is as follows.



Instructions:
For Event: event name + time duration + place (e.g. Hackson 2018-10-5 13:30-17:00 @BH1538)
For Deadline: event name + time + place (e.g. EE209AS Bake-off#1 2018-10-9 16:00 @BH2760)
For Goal event name + time (e.g. finish "PRML" 2018-12-10 23:59)

Today's Events:

Event name	Start time	End time	Place	Type
STAT 231A HW#1	2018-10-17 23:59		CCLE	deadline
EE209AS code submission	2018-10-9 23:59		CCLE	deadline

Fig. 1 Layout of **Home page**

Navigation bar

Navigation bar directs to the other two activities (namely "**All Events**", "**Priorities**"). The "**Admin login**" tuple can redirect to Django backend administration page.

Input Group

Input Group can receive formatted inputs and pass the value to backend event identification algorithm "add_2_db" in sd.views. The user can choose to input three types of events, namely Event, Deadline, Goal. An Event should be consisted of a name, start time, end time, place. A Deadline should be consisted of a name, start time, place. A Goal should be consisted of a name, start time. A summary of input requirement is shown in Table 1.

Table 1 Required information for input group

Event Type	Name	Start time	End time	Place	example
Event	Y	Y	Y	Y	Hackson 2018-10-5 13:30-17:00 @BH1538

Deadline	Y	Y	N	Y	EE209AS Bake-off#1 2018-10-9 16:00 @BH2760
Goal	Y	Y	N	N	finish "PRML" 2018-12-10 23:59

A mandatory event type should be added by an option list on the right of input group. See Fig. 2 for details.

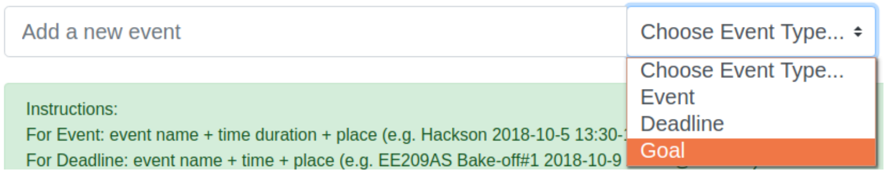

Add a new event
Choose Event Type...
Choose Event Type...
Event
Deadline
Goal
Instructions:
For Event: event name + time duration + place (e.g. Hackson 2018-10-5 13:30-1
For Deadline: event name + time + place (e.g. EE209AS Bake-off#1 2018-10-9

Fig. 2 Add event type to input group

Instructions

Instructions is a dismissible message box which can remind user of the input grammar defined in **Input Group.**

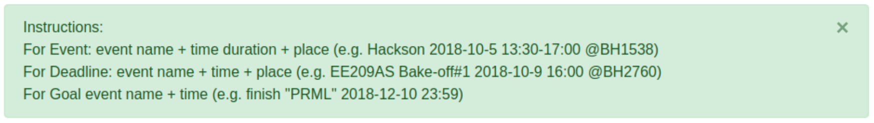
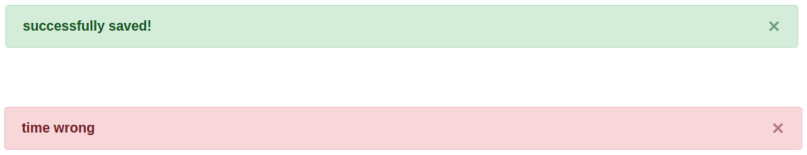

Instructions:
For Event: event name + time duration + place (e.g. Hackson 2018-10-5 13:30-17:00 @BH1538)
For Deadline: event name + time + place (e.g. EE209AS Bake-off#1 2018-10-9 16:00 @BH2760)
For Goal event name + time (e.g. finish "PRML" 2018-12-10 23:59)

Fig. 3 Instructions

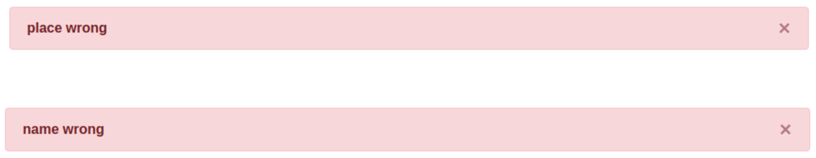
Messages

Messages is a message box indicating the add status of the submitted data from **Input Group**.

There are 4 types of messages: (1) success: the event is successfully submitted; (2) danger: time wrong: the format of time is incorrect; (3) danger: place wrong: the input does not indicate a place; (4) danger: name wrong: the input does not indicate an event name (name with only spaces(/s) is not allowed).


successfully saved!

time wrong

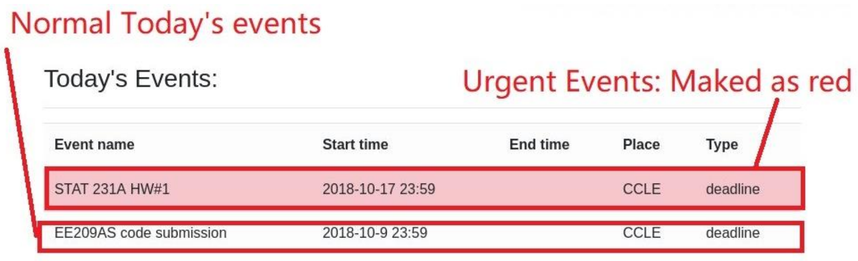

place wrong ×

name wrong ×

Fig. 4 Four types of messages (success, time wrong, place wrong, name wrong)

Today's Events

Today's Events is a table which can display urgent events (marked as red) and the current day's undo events from saved database. The layout can be shown as follows. The urgent events are always placed on the top and the following current day's undo events are placed by upcoming order.



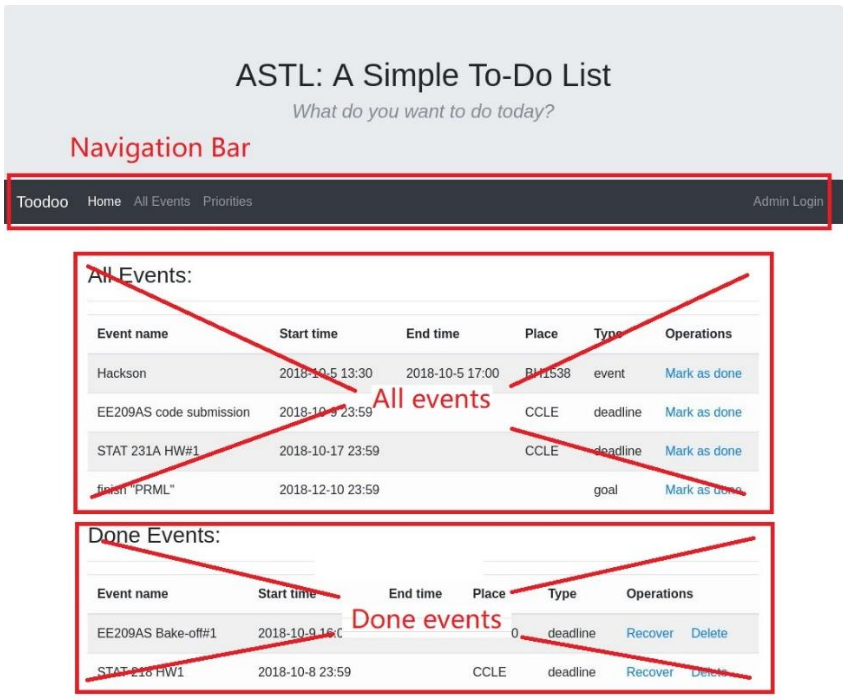
Today's Events:

Event name	Start time	End time	Place	Type
STAT 231A HW#1	2018-10-17 23:59		CCLE	deadline
EE209AS code submission	2018-10-9 23:59		CCLE	deadline

Fig. 5 Layout of Today's Events.

All Events

All Events page (allEvents.html) contains Navigation Bar, All Events and Done events.



All Events:

Event name	Start time	End time	Place	Type	Operations
Hackson	2018-10-5 13:30	2018-10-5 17:00	BH1538	event	Mark as done
EE209AS code submission	2018-10-9 23:59		CCLE	deadline	Mark as done
STAT 231A HW#1	2018-10-17 23:59		CCLE	deadline	Mark as done
finish "PRML"	2018-12-10 23:59			goal	Mark as done

Done Events:

Event name	Start time	End time	Place	Type	Operations
EE209AS Bake-off#1	2018-10-9 16:0		0	deadline	Recover Delete
STAT 218 HW1	2018-10-8 23:59		CCLE	deadline	Recover Delete

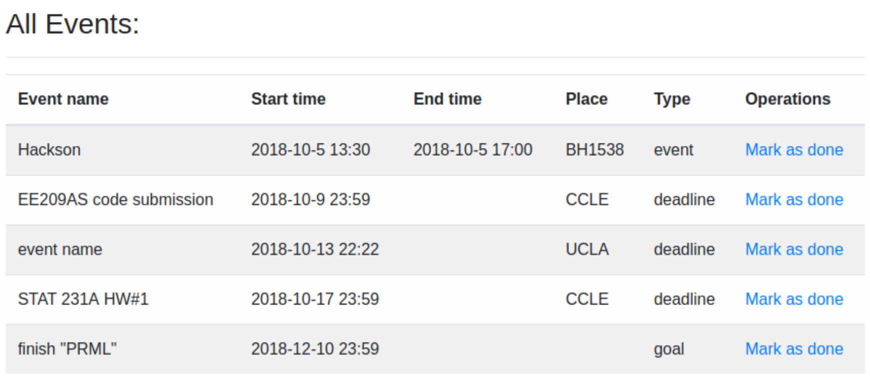
Fig. 6 Layout of **All Events**

Navigation Bar

Same as in **Home Page**.

All Events

All Events contains all undo events, all event items are placed with upcoming order. The table indicates the event's name, start time, end time (optional), place, type, operations.



All Events:

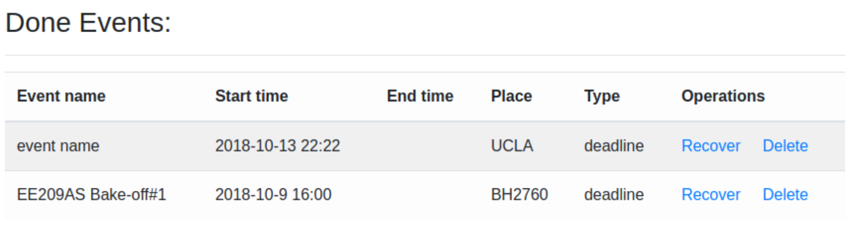
Event name	Start time	End time	Place	Type	Operations
Hackson	2018-10-5 13:30	2018-10-5 17:00	BH1538	event	Mark as done
EE209AS code submission	2018-10-9 23:59		CCLE	deadline	Mark as done
event name	2018-10-13 22:22		UCLA	deadline	Mark as done
STAT 231A HW#1	2018-10-17 23:59		CCLE	deadline	Mark as done
finish "PRML"	2018-12-10 23:59			goal	Mark as done

Fig. 7 All Events Table

There is a “Mark as Done” operation in each row of the events. Once clicked, the event in the corresponding row would “mark as done” and be removed from **All Events**. The “mark as done” events can be found in **Done events**.

Done events

Done Events contains all done events, all event items are placed with reverse time order. The table indicates the done event’s name, start time, end time (optional), place, type, operations.



Done Events:

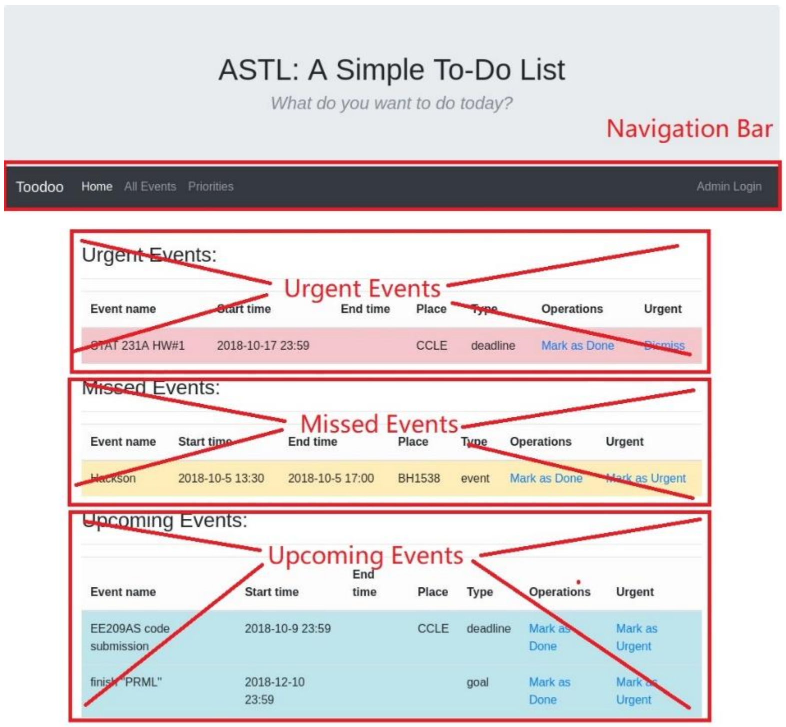
Event name	Start time	End time	Place	Type	Operations
event name	2018-10-13 22:22		UCLA	deadline	Recover Delete
EE209AS Bake-off#1	2018-10-9 16:00		BH2760	deadline	Recover Delete

Fig. 8 Done Events Table

There are two operations, “Recover” and “Delete” in each row of the events. Once “Recover” is pressed, the event is marked as “undone”, removed from Done Events and reappear to **All Events** Table. Once “Delete” is pressed, the event is removed from backend database, and would not appear again.

Priorities

Priorities page (weeklySchedule.html) contains Navigation Bar, Urgent Events, Missed Events, Upcoming Events.



ASTL: A Simple To-Do List

What do you want to do today?

Toodoo Home All Events Priorities Admin Login

Urgent Events:

Event name	Start time	End time	Place	Type	Operations	Urgent
STAT 231A HW#1	2018-10-17 23:59		CCLE	deadline	Mark as Done	Dismiss

Missed Events:

Event name	Start time	End time	Place	Type	Operations	Urgent
Hackson	2018-10-5 13:30	2018-10-5 17:00	BH1538	event	Mark as Done	Mark as Urgent

Upcoming Events:

Event name	Start time	End time	Place	Type	Operations	Urgent
EE209AS code submission	2018-10-9 23:59		CCLE	deadline	Mark as Done	Mark as Urgent
finish "PRML"	2018-12-10 23:59			goal	Mark as Done	Mark as Urgent

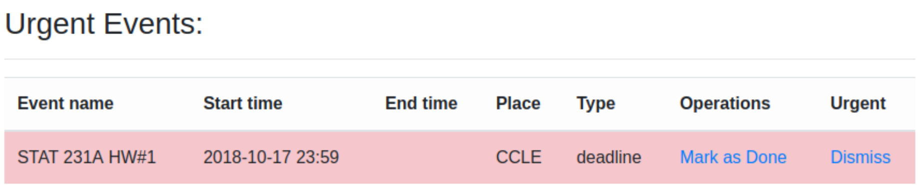
Fig. 9 Layout of **Priorities**

Navigation Bar:

Same as in **Home Page**.

Urgent Events

Urgent Events displays a list of events marked as "urgent". The table indicates the urgent event's name, start time, end time (optional), place, type, operations, urgent. The "urgent" event is also shown in **Home Page**.



Urgent Events:

Event name	Start time	End time	Place	Type	Operations	Urgent
STAT 231A HW#1	2018-10-17 23:59		CCLE	deadline	Mark as Done	Dismiss

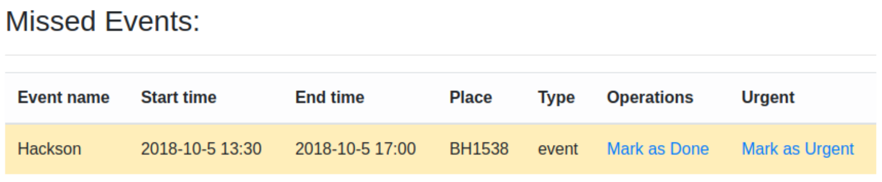
Fig. 10 Urgent Events Table

In Operations field, the user can mark the event as done, but the event would still remain as urgent.

In Urgent field, the user can dismiss the urgent event, and the event would be removed from urgent events (and may to missed events or upcoming events, according to the status)

Missed Events

Missed Events displays a list of events which is undone passed events/deadline/goals. The table indicates the urgent event's name, start time, end time (optional), place, type, operations, urgent.



Missed Events:

Event name	Start time	End time	Place	Type	Operations	Urgent
Hackson	2018-10-5 13:30	2018-10-5 17:00	BH1538	event	Mark as Done	Mark as Urgent

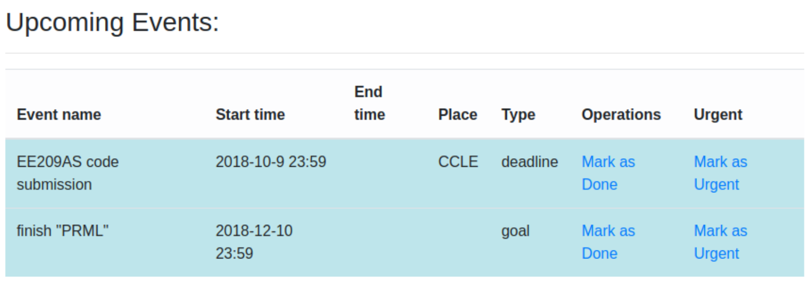
Fig. 11 Missed Events Table

In Operations field, the user can mark the event as done and the event would be moved to **Done List** in **All Events**. (and would reappear if the user recovers the event)

In Urgent field, the user can mark the event as urgent, and the event would go to **Urgent Events Table** and **Today's events** in **Home Page.** (and would reappear if the user dismisses the event)

Upcoming Events

Upcoming events displays a table of upcoming undone events in time-increase order. The table indicates the urgent event's name, start time, end time (optional), place, type, operations, urgent.



Upcoming Events:

Event name	Start time	End time	Place	Type	Operations	Urgent
EE209AS code submission	2018-10-9 23:59		CCLE	deadline	Mark as Done	Mark as Urgent
finish "PRML"	2018-12-10 23:59			goal	Mark as Done	Mark as Urgent

Fig. 12 Upcoming Events Table

In Operations field, the user can mark the event as done and the event would be moved to **Done List** in **All Events**. (and would reappear if the user recovers the event)

In Urgent field, the user can mark the event as urgent, and the event would go to **Urgent Events Table** and **Today's events** in **Home Page.** (and would reappear if the user dismisses the event)

Backend and Data Structure

The project uses Django 2.2 w/ sqlite Database as backend.

File Structure & Applications

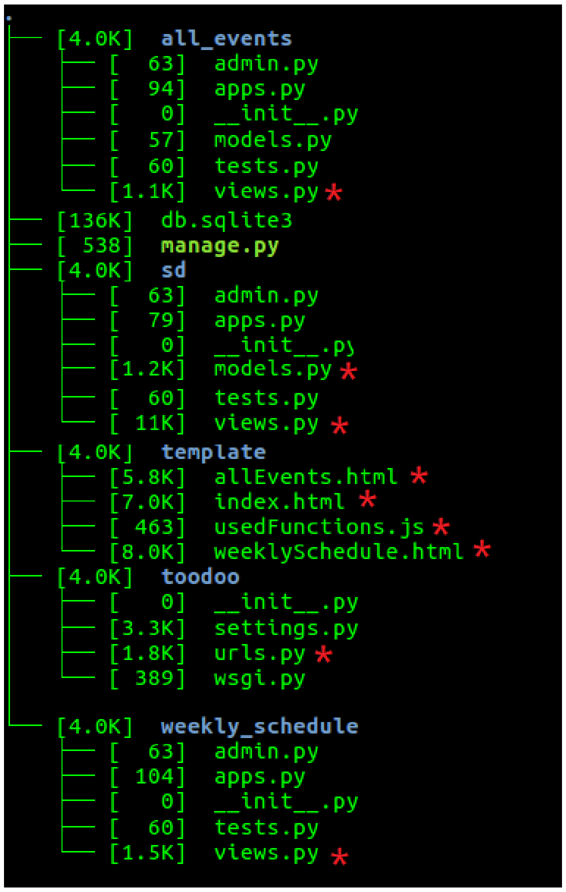
There are three applications under Django Framework. The basic data structure, uq_event() in defined in sd.models().

sd: corresponding to **Home Page**

all_events: corresponding to **All Events**

weekly_schedule: corresponding to **Priorities (changed function as development)**

The file structure is generated automatically under Django Framework. Below is the file path and files marked as (*) are modified/written additionally.



```
.
├── [4.0K]  all_events
│   ├── [  63]  admin.py
│   ├── [  94]  apps.py
│   ├── [   0]  __init__.py
│   ├── [  57]  models.py
│   ├── [  60]  tests.py
│   └── [1.1K]  views.py *
├── [136K]  db.sqlite3
├── [ 538]  manage.py
├── [4.0K]  sd
│   ├── [  63]  admin.py
│   ├── [  79]  apps.py
│   ├── [   0]  __init__.py
│   ├── [1.2K]  models.py *
│   ├── [  60]  tests.py
│   └── [ 11K]  views.py *
├── [4.0K]  template
│   ├── [5.8K]  allEvents.html *
│   ├── [7.0K]  index.html *
│   ├── [ 463]  usedFunctions.js *
│   └── [8.0K]  weeklySchedule.html *
├── [4.0K]  toodoo
│   ├── [   0]  __init__.py
│   ├── [3.3K]  settings.py
│   ├── [1.8K]  urls.py *
│   └── [ 389]  wsgi.py
│
└── [4.0K]  weekly_schedule
    ├── [  63]  admin.py
    ├── [ 104]  apps.py
    ├── [   0]  __init__.py
    ├── [  60]  tests.py
    └── [1.5K]  views.py *
```

Fig. 13 File structures

Event Object

The main data structure object in the project is uq_event() object. The object is defined in `sd.models.py`. The object has the following attributes.

Table 2 Attributes of Event Object

Attribute	Description	Type	Default
evt_name	Event name	char	Required Field
evt_place	Event place	char	
evt_exact_time	Exact event time	char	Required Field
evt_year	Year	char	Required Field
evt_month	Month	char	Required Field
evt_date	date	char	Required Field
evt_end_time	Event end time	char	
evt_done	Whether event is done	boolean	False
evt_type	Event type: (event, deadline, goal)	char with choice	
Uuid	Unique event ID	char	NA
unix_tmp	Unix timestamp of event exact time	float	1000000000.0
urgent	Whether is urgent	boolean	False

Event Object Automata

Below is an automata of uq_event(). Once the input pass integrity check, an uq_event() is established.

- If the user marks the event as done (in **All Event**), the object would go to *Done* State.
- If the user mark the event as urgent (in **Priorities**), the object would go to *Urgent* State.
- If the user recovers the event in *Done* State (in **All Event**) and the current time is *smaller* than the event's unix timestamp, the object would go to *Initial* State.
- If the user deletes the event in *Done* State (in **All Event**) the object would go to *Delete* State and the object would be removed directly.
- If the user recovers the event in *Done* State (in **All Event**) and the current time is *larger* than the event's unix timestamp, the object would go to Missed State.
- If an object in *Missed* State in marked as Urgent (in **Priorities**), the object would go to *Urgent* State.
- If an object in *Missed* State in marked as Done (in **Priorities**), the object would go to *Done* State.
- If an object in *Urgent* State is dismissed (in **Priorities**) and the unix timestamp is *smaller* than current time, the object would go to *Missed* State.

- If an object in *Urgent* State is dismissed (in **Priorities**) and the unix timestamp is *larger* than current time, the object would go to *Missed* State.

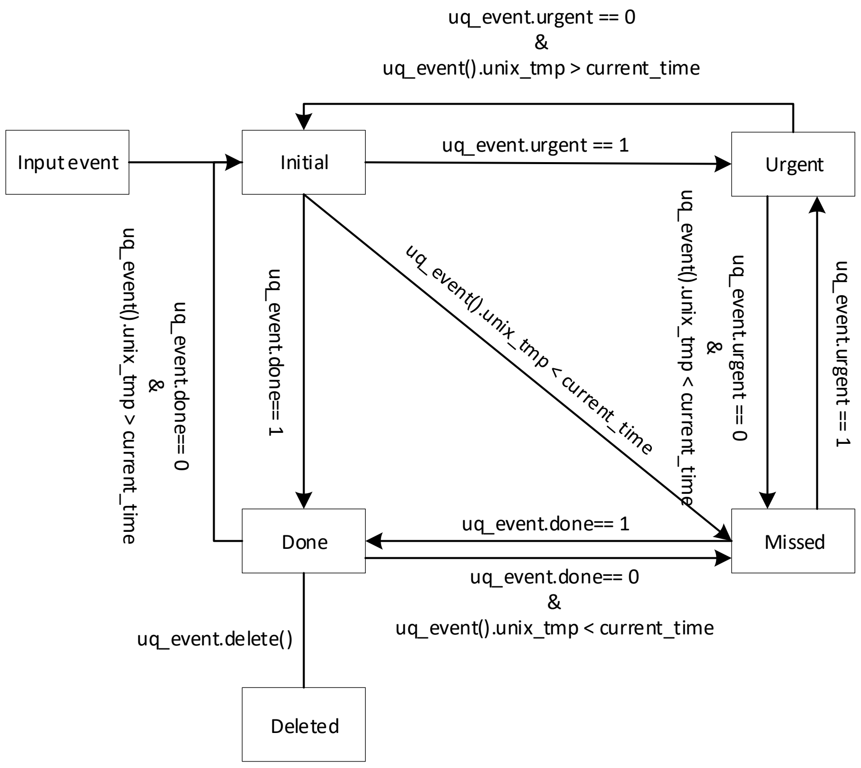


Fig. 14 Automata of uq_event()