All of api’s from backend will use the same format as following:
Online JSON format: http://jsonviewer.stack.hu/

HTTP API

General Response Structure

{
  'flag': 0/1,////0 success, 1 failed
  'data': { },
  'message': “xxxxx”
}

Definition

Flag:
  "000": "Success",
  "001": "Sign up successfully",
  "002": "Successfully Connect Websocket",
  "101": "Django Error",
  "102": "Username has been registered",
  "103": "Username should only contain digits!",
  "104": "Email has been registered",
  "105": "Illegal email format",
  "106": "Repeat password is not the same with origin password",
  "107": "Username or password wrong",
  "108": "User not active",
  "109": "Document Title could not be empty",
  "110": "Require document id",
  "111": "Cannot find document id",
  "112": "Not recognize request path",
  "113": "Malformed request",
  "114": "Nickname needed in chat request",
  "115": "Docid needed in chat request",
  "116": "User id needed in request",
  "117": "User not found",
  "118": "Indice error",
  "119": "Illegal docid",
  "120": "User not log in",
  "121": "Unsupported Request Method",
  "122": "Document has been deleted",
  "123": "Illegal Token",
  "124": "Email not found",

"125": "Change password failed",
"126": "Unsupported notification type",
"127": "No such notification",
"999": "Unknown Error",

User account control

1. **User signup:**

   Path: backend/user/signup
   Request:
   {
   'username': "",
   'nickname': "",
   'email': "",
   'password': "",
   'confirm_password': ""
   }
   Response:
   Data: {
   "uid": xxx,
   "nickname":xxx,
   }

2. User signin:

   Path: backend/user/signin
   Request:
   {
   'username': "",
   'password': ""
   }
   Response:
   Data: {
   "uid": xxx,
   "nickname":xxx,
   }

   a. Edit personal information: backend/user/edit

b. User log out: backend/user/logout
Post to this address, no parameters need

Document Management:

1. Structure Definition:
Document:
{
docid:"",
title:"",
content:"", // optional for some response
owner:"{uid}",
shared_user:[],
abstract:"",
ctime:"",
}

1. Get document content:

Path: backend/doc/get
Method: POST
Request:
{
docid: "",
}

Response:
data: {Document object}

2. Create document

Method: POST
Path: backend/doc/create
{
"title": "" // no need to pass owner
// shared users, optional parameters
"shared_user": ["username", "test"]
}

Response:
data: {Document object}

3. List documents:

Method: POST
Path: backend/doc/list

Request
{
“”
}

Response:
data:{
owned: [doc, doc...]
shared: [doc, doc..]
}

4. Search document:

Method: POST
Path: backend/doc/search

Data: {
“docs”:
[
“docid”: “”,
“title”: “”,
“owner”: “”,
“shared_user”:[],
“abstract”: “”,
“ctime”: “”,
]

}

Display document:

Data: {
[Document, Document, Document]
}

5. Delete document:

User could delete document which he created or shared to him
Method: POST
Path: /backend/doc/delete
Request data:
{
“Docid”: {docid}
}
Response data:

Flag = 0/1

6. Edit document:

User could edit document which he created
Method: POST
Path: /doc/edit
{
“docid”:””
“title”: “” // no need to pass owner
// shared users, optional parameters
“shared_user”: [“username”, “test”]
}
Response data:
Flag = 0/1

Websocket API

General Structure:

1. Websocket initial path: backend/{docid}
2. Message Structure

{
flag: “”// 0(success), -1,
message: “xxx”,
data: {
action: “chat” // “doc/add”, “doc/delete”, “doc/update”,
uid: “”,
docid:””,
}

}

3. Definition
Flag:

API Sets

1. Online-Chat
{
message: “xxx”,
data{
action: “chat”
uid: “”,
nickname: “nickname”,
docid:””,
content: “”,

```
        }
}
```

2. [edit page]Document Initialize (sent from server to client)

```
{
        message: “xxx”,
        data{
                action: “doc/init”
                uid: “”,
                docid:””,
                content: “”,
                shared_user:[],
        }
}
```

3. [edit page]Document update (synchronize between clients)

```
{
        message: “xxx”,
        data{
                action: “doc/add”// “doc/delete”, “doc/update”
                uid: “”,
                docid:””,
                start: “”,
                end: “”,
                content: “”,
        }
}
```

4. [profile page]Document access update (synchronize between clients)

```
{
        message: “xxx”,
        data{
                action: “backend/doc/delete”
                uid:””,
                docid: “”,
                involved_user:””,
        }
}
```

4. [profile page]Document delete(synchronize between clients)

{
message: “xxx”,
data{
action: “backend/doc/delete”
uid:””,
docid: “”,
}
}

5. [Notification]FrontendToBackend
Invitation:request
{
Data{
message_type:”0(invitation)”
docid:
other_user: // username
}
}
response:
{
Flag: 0/1
Data{
message_type:”1(accept)/2(decline)”
Other_user: // uid
nid:
}
}

Accept/Decline:
request:
{
Data{
message_type:”1(accept)/2(decline)”
Other_user: // uid
nid:
}
}
response:
{
Data{
message_type:”1(accept)/2(decline)”
other_user:// message source uid
nid:

```
        }
    }
Read:
    request
    {
        Data{
            message_type:”3(read)”
            nid:
        }
    }
    response
    {
        Data{
            message_type:”3(read)”
            other_user:// message source, uid
            nid:
        }
    }
```

6. Reply from server

```
{
    Flag: 0/1,
    Message: “”,
    Data: {}
}
```

6. Send notification to receiver

```
{
    flag: 0/1,
    message: “”,
    data: [
        // unread notification messages
        {
            “user_send”: {
                uid: 0,
                Nickname: “”,
            },
            time: “”
            “type”: 0/1/2/3,
            “doc”: {
                docid: 0,
                title: “”
            },
```

```
                    nid: “”
                },
            ]
        }
```