ITUPass Documentation

Release 0.1

Emin Mastizada

Dec 22, 2017

Contents:

Team itucsdb1739

Member Emin Mastizada

StudentID 150120914

Keep calm and pass lectures

ITUPass helps to easily track lectures and have connection with classmates. The only idea is to make university life a bit easier.

Checkout Developer Guide to learn about implementations and User Guide to have an idea how to use system.

CHAPTER 1

User Guide

At homepage, go to Login at the top of the page, There you will need to login using credentials.

You can use one of our temporarily demo accounts, or Register to create a new account.

After registration or login, it will redirect you to the dashboard. You can access dashboard any time using navigation header.

To access dashboard from the navigation menu, open your personal navigation menu by clicking to your name in header panel, select Dashboard.

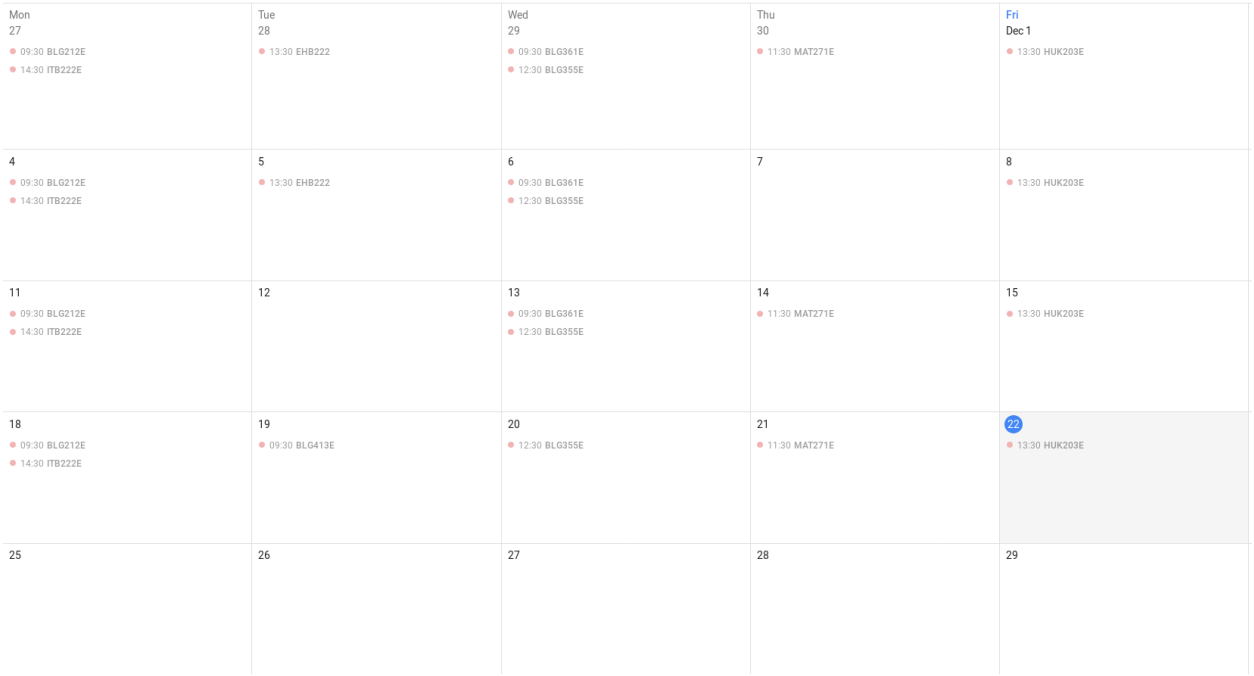
Dashboard has the list of subscribed events (like academic calendar, Computer Department announcements and etc.) and your lectures.

Click to add lecture to register for a new lecture. In registration page, easily filter departments and lectures for those departments.

After registering for lectures and choosing subscribed events, you will notice iCal file URL at the top of the dashboard page. That is a Calendar file using iCal format for all your schedule.

You can easily use that file to add your schedule to Thunderbird, Google Calendar and any other Calendar application you want. For online calendars, use “Import from URL” to get live updated for your schedule.

- Expected result:


Mon
27
09:30 BLG212E
14:30 ITB222E
Tue
28
13:30 EHB222
Wed
29
09:30 BLG361E
12:30 BLG355E
Thu
30
11:30 MAT271E
Fri
Dec 1
13:30 HUK203E
4
09:30 BLG212E
14:30 ITB222E
5
13:30 EHB222
6
09:30 BLG361E
12:30 BLG355E
7
8
13:30 HUK203E
11
09:30 BLG212E
14:30 ITB222E
12
13
09:30 BLG361E
12:30 BLG355E
14
11:30 MAT271E
15
13:30 HUK203E
18
09:30 BLG212E
14:30 ITB222E
19
09:30 BLG413E
20
12:30 BLG355E
21
11:30 MAT271E
22
13:30 HUK203E
25
26
27
28
29

CHAPTER 2

Developer Guide

2.1 INSTALL

Project uses Flask and PostgreSQL database to work. Add your Flask settings to *itupass/.env* file to operate, or to your server environment variables.

Prepare Environment:

```
* Create virtual environment in venv folder:
```

$ virtualenv venv -p python3

- **Install project:** *$ pip install –editable .*
- **Set flask app name:** *$ export FLASK_APP=itupass $ export FLASK_DEBUG=1*
- **Initialize database:** *$ flask initdb*
- **Run application:** *$ flask run*

Environment Variables:

```
* `DATABASE_URI` - Database uri address (postgres://user:password@server:port/db)
* `VCAP_SERVICES` - Bluemix settings
* `SECRET_KEY` - Secret key for cookies
* `SENTRY_DSN` - Sentry (debug tool) DSN
* `TravisCI` - Check for Travis Continues Integration environment (pull request test␣
↪service)
```

2.2 Database Design

- Main tables: *users*, *lectures*, *events*, *departments*

- *lecture_departments* used to store department information for lectures, students of which departments can take those lectures. It connects to the *lectures* table using *lecture* key and to the *departments* table using *department* key. It was used to create ManyToMany relationthip between *lectures* and *departments* tables.
- *lecture_schedule* used to store schedule information for lectures. As same lectures can have multiple schedules (like Math lectures) they are stored seperately and connected to the *lectures* table using *lecture* reference key.
- *user_lectures* is for storing lecture registrations for users (Check User Guide for more information).
- *event_categories* stores category names in multiple languages for Events.

2.3 Model Design

- Models are located under *itupass/models* folder, they all has similar structure:

```
class <Name>(object):
  @classmethod
  def get(cls, pk):
    # Get item from database using select command with identifier, can be␣
↪multiple identifiers for different entities
    # Return object with same time, initialized with database result
    # If row is not found, return None, do not raise exception

  @classmethod
  def count(cls, **kwargs):
    # Use "SELECT COUNT" to get number of rows
    # Use `kwargs` to make filter, same as in `filter` function
    # Return number of rows as integer

  @classmethod
  def filter(cls, limit=10, offset=0, order="id DESC", **kwargs):
    # Use SELECT command to fetch multiple rows from database
    # Use limit and order for select command
    # offset is used for pagination
    # `kwargs` is used for filtering parameters, use `where_builder` from␣
↪`Database` class to build `WHERE` clause and its dictionaty values
    # Return list of objects in same type
    # If no results, return empty array

  def delete(self):
    # Delete current object from database using identifier (Primary Key)
    # Do not return anything

  def save(self):
    # If object is new, use INSERT command to add to the database
    # If existing object (check using identifier), find differences from␣
↪the one existing in database using `diffkeys`
    # Use UPDATE command to update current item in database
    # Return new object using identifier and `get` function

  class Meta:
    table_name = "database_table_name_for_object"
```

- *__init__* function of the object should take parameters in the same order as in schema.sql file.
- After creating model add it to itupass/models/__init__.py file for easy import.

Create Form for each Model:

- For safety and validations, use Forms for creating models in views.
- Forms are stored in *itupass/forms* folder.

Creating View:

- To show your new model in client you will need add view for it.
- Create view file under *itupass/views* folder.
- Make Blueprint variable for your view, add all views for that Blueprint.
- Add your Blueprint variable to *itupass/views/__init__.py* file for easy import.
- Register your view in *itupass/itupass.py* file to make it accessible.

Conclusion:

- After adding Model, Form, View and Template for that view, registering Blueprint for that view in *DEFAULT_BLUEPRINTS* variable, your new item will be added to the app.
- There are 2 main view, *client*, *dashboard* and *admin*, which is for main control. Most of the models will use only these views to show themself instead of own views.