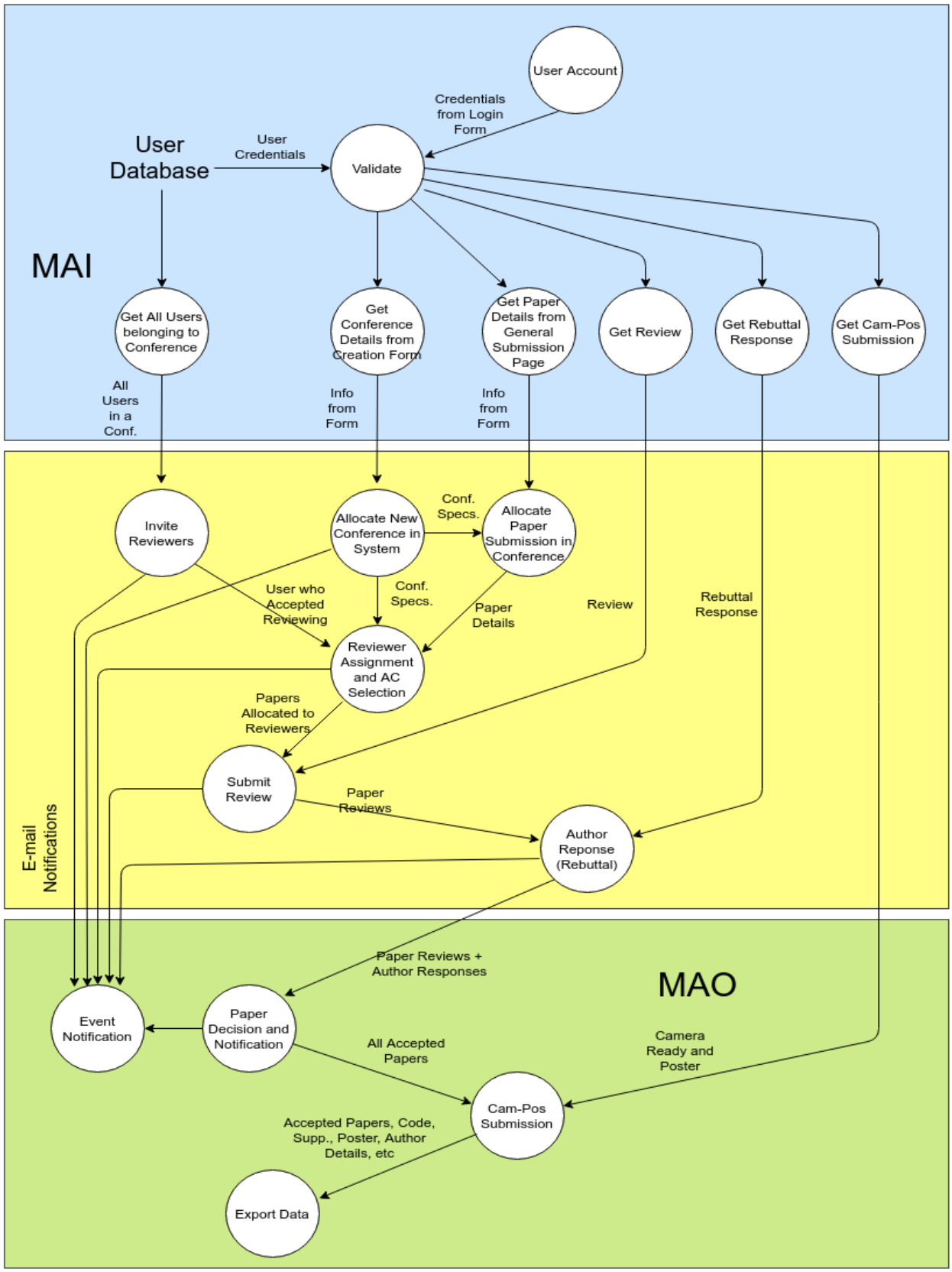
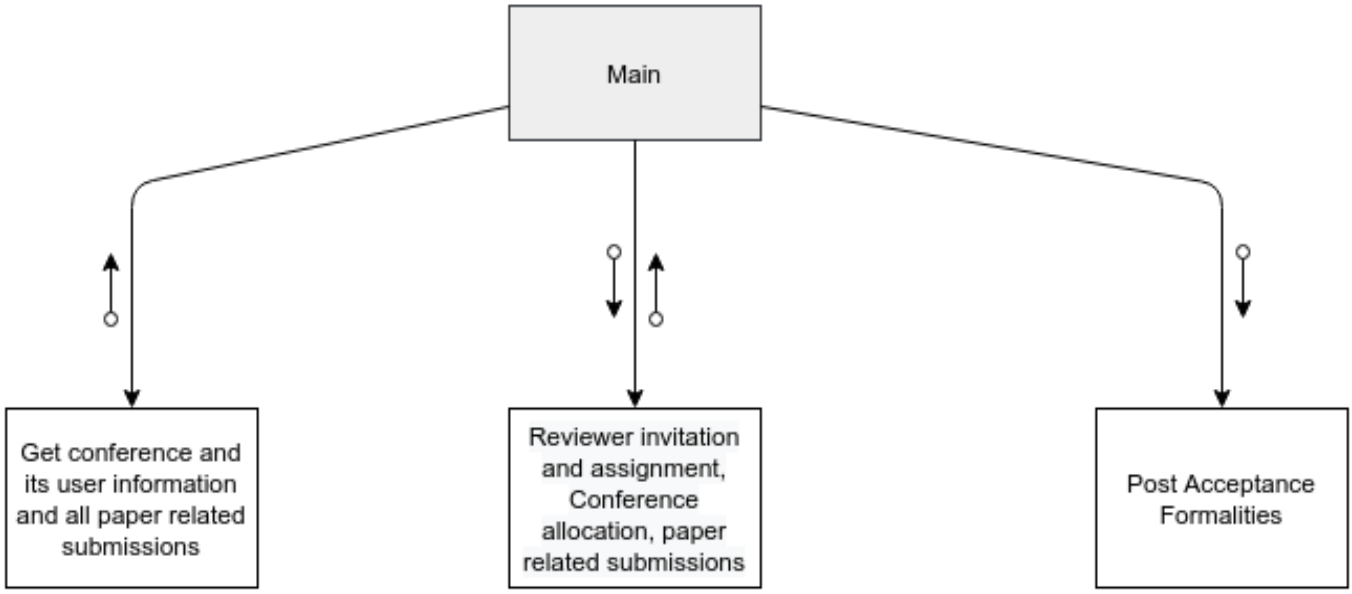
Software Design for CMS

Ajinkya Bokade, CS17BTECH11001
Yash Khasbage, CS17BTECH11044
Jatin Chauhan, CS17BTECH11019
Happy Mahto, CS17BTECH11018
Himanshu Bishnoi, CS16BTECH11018

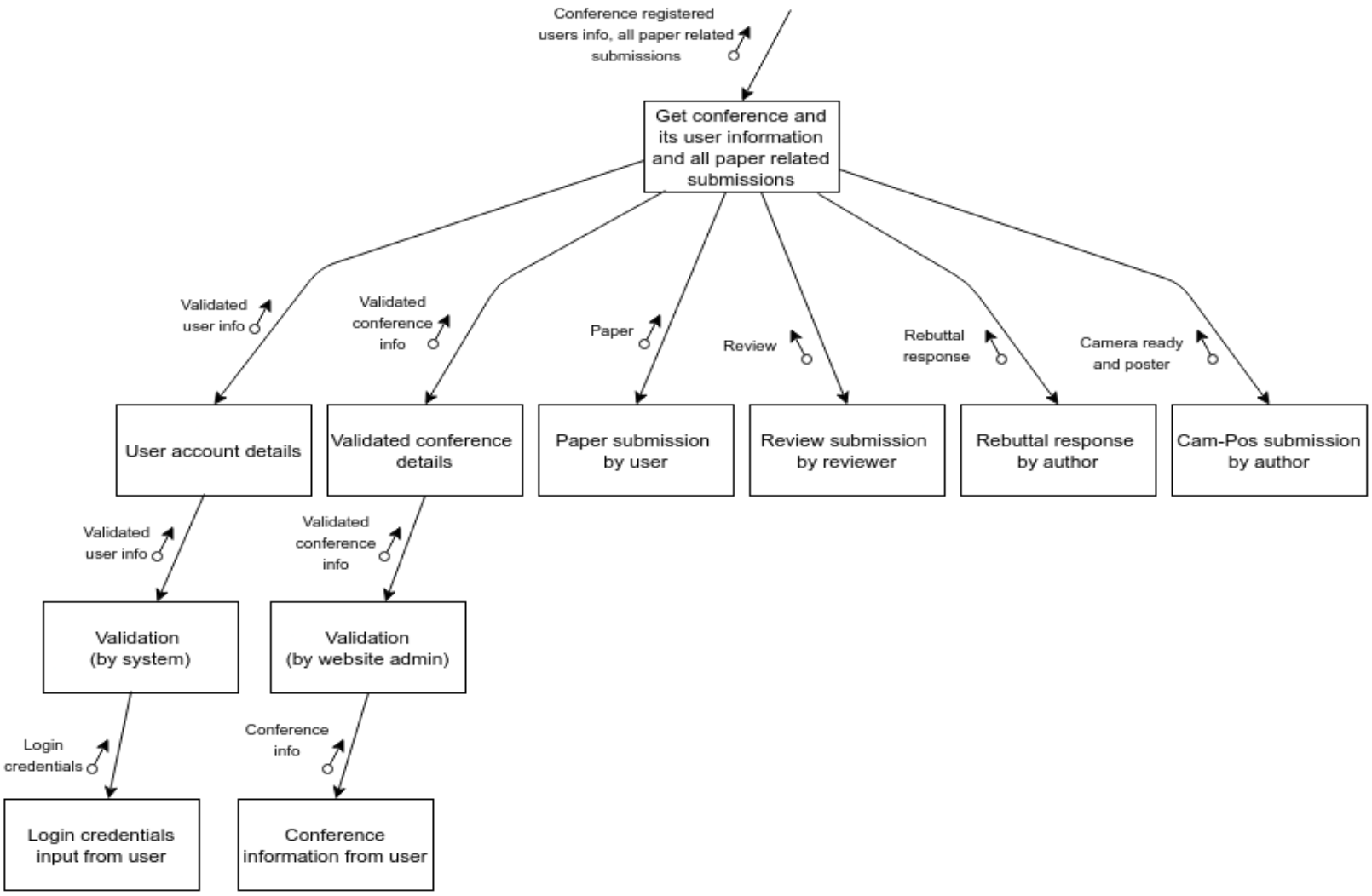
Data Flow Diagram



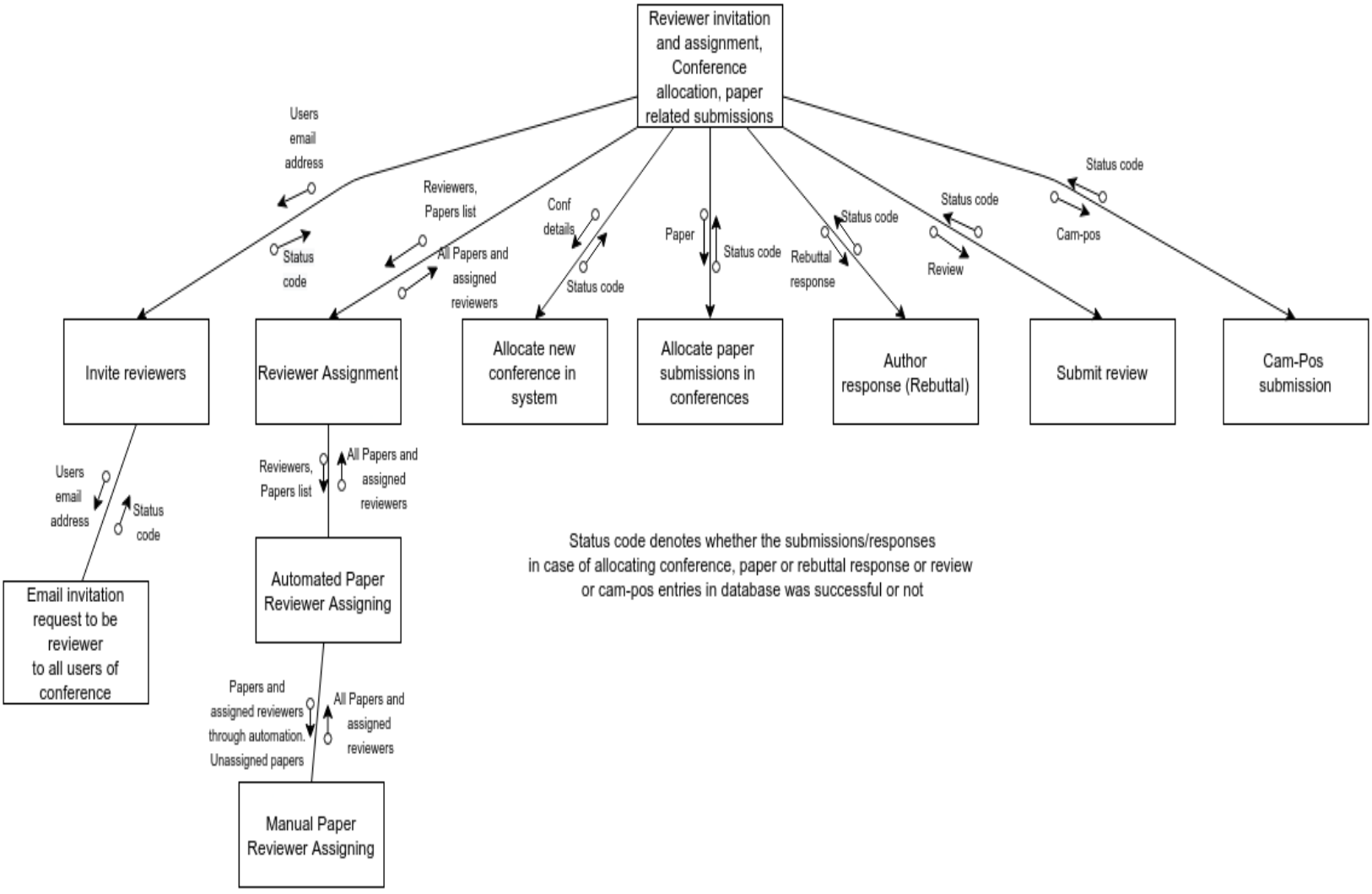
First level factoring



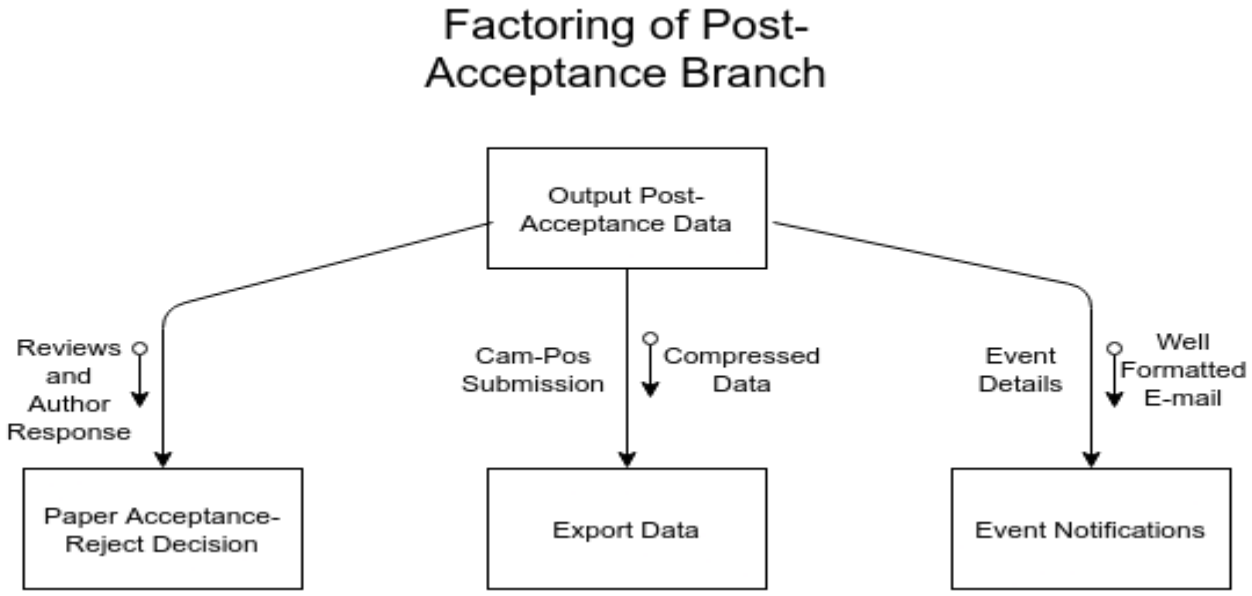
Factoring of input branch



Factoring the central transform



Status code denotes whether the submissions/responses in case of allocating conference, paper or rebuttal response or review or cam-pos entries in database was successful or not



Design Specification

The structure charts above describe the modules, the main parameters that are being passed across them, and the factoring across different levels. We now describe the interface of each of the functions involved, followed by the table which summarizes their LOCs. We then provide the top-3 modules along with counts in terms of fan in and fan out and lastly provide the most complex and error-prone module names.

Interface of Functions

The interface of each of the functions involved is defined in this subsection. For each function, we provide a brief description of what it does (in the comments form, which can be directly utilized during the documentation), followed by the function. The inputs and the outputs of the functions are assumed to be python objects, primarily dictionaries, which are converted “from and to” database objects via the access layer.

Note that python dicts can hold arbitrary data types, making it easier to store different types of variables required, into a unified dictionary object.

Data Definitions

Here we define the structures of python dictionaries and lists used in the interfaces sub-sub section below:

1) *user_info* - the python dictionary object containing the information pertaining to the user such as: username, password (hashed version for security), and additional optional details regarding the profile.
2) *conference_info* - the python dictionary object containing the information pertaining to the conference (requested to be created) such as: name, details, dates, usernames of CAs, etc.
3) *paper* - the python dictionary object containing the information pertaining to the paper submission made by a user such as: titles, abstract, paper pdf, supplementary, etc.
4) *review* - the python dictionary object containing the information pertaining to the review of a paper made by a user (who is a reviewer of the corresponding paper) such as: summary, strengths, weaknesses, score etc.
5) *rebuttal_response* - the python dictionary object containing the information pertaining to the rebuttal response of a review of a paper, made by a user. It will contain the pdf of response (more specifically, a pointer to the pdf file saved in the system, rather than the file itself).
6) *cam_pos* - the python dictionary object containing the information pertaining to camera-ready and poster submission of an accepted paper such as: the pdf files of the finished main paper, supplementary file, and the poster. Again these will be pointers to the pdf files saved in the system, rather than the files directly.
7) *reviewer_paper_mapping*- the python dictionary object containing the information pertaining to the assignment of each paper to its corresponding set of reviewers, making it a nested dictionary. The outer dictionary containing all the papers and each of the inner dictionaries correspond to each paper. The ids of the papers and the reviewers will be stored.
8) *email_notif_status_code* - the python dictionary object containing the status code of the email notification requests. Similar to JSON status code responses.

9) *ac_response* - python dictionary containing the response of the area chair pertaining to the paper in consideration such as: decision, meta_reviews.
10) *user_email_address* - python list containing the email addresses of all users in the management system, required to send invitations to.
11) *reviewer_list* - the python list containing all users who have accepted to become the reviewer for the given conference.
12) *paper_list* - the Python list containing the ids of all papers submitted to the given conference.

Interfaces

Each of the functions is separated by the series of #'s as below.

```
############################################################
"""
This is the main module that serves as the junction between the input
and the output modules. It takes the information parsed via the HTML
forms through the subordinates of the input module as its parameters,
then call the subordinate module to perform the core operations of
the management system, and finally returns all the necessary python
objects to be used by the output modules.
"""
main(user_info,conference_info,paper,review,rebuttal_response,
cam_pos):
      {
          SUBORDINATES: get_conf_user_paper_info(),
          perform_assignment_conf_allocation(),
          output_post_acceptance()
      }
      return user_info, conference_info, paper,
      review,rebuttal_response, cam_pos, reviewer_paper_mapping,
      email_notif_status_code
############################################################

############################################################
"""
This is the top-level module on the input domain, which calls its
```

```
subordinate modules that take the various types of parsed user inputs
as well as validate them. The obtained MAIs are returned to the main
module for further computations.
"""
get_conf_user_paper_info(user_info,conference_info, paper, review,
rebuttal_response, cam_pos):
      {
           SUBORDINATES:user_account_details(),
           validated_conference_details(),
           get_paper_submission_by_user(),
           get_review_submission_by_reviewer(),get_rebuttal_response()
           get_cam_pos_submission()
       }
       return user_info, conference_info, paper, review,
       rebuttal_response, cam_pos
#################################################################

#################################################################
"""
This module takes the validated user_info python object (validated by
its subordinate) and returns it as the output, which is passed to
get_conf_user_paper_info function.
"""
user_account_details(user_info):
      {
            SUBORDINATES: user_acc_validation()
      }
      return user_info
#################################################################

#################################################################
"""
This module takes the parsed user_info object via its subordinate and
performs the different types of authentication and validation
required to confirm the user. The user_info object is then returned
to its parent function.
"""
user_acc_validation(user_info):
      {
```

```
            SUBORDINATES: user_login_details()
      }
      return user_info
#################################################################

#################################################################
"""
This module takes the parsed user information from the login screen
via the HTML forms. This information retrieved is converted into the
user_info type dict and then passed to user_acc_validation module for
validation.
"""
user_login_details():
      {
            SUBORDINATES:None
      }
      return user_info
#################################################################

#################################################################
"""
This module takes the validated conference_info python object
(validated by its subordinate) and returns it as the output, which is
passed to get_conf_user_paper_info function.
"""
validated_conference_details(conference_info):
      {
            SUBORDINATES: admin_conf_validation()
      }
      return conference_info
#################################################################

#################################################################
"""
This module takes the parsed conference_info object via its
subordinate, which first performs different types of automated
authentication and validation to remove any incorrect detail
regarding the new conference requested, and then another step of
validation is performed by the website admin manually. The
```

```
conference_info object is then returned to its parent function.
"""
admin_conf_validation(conference_info):
      {
            SUBORDINATES: conf_info_input()
      }
      return conference_info
#################################################################


#################################################################
"""
This module takes the parsed conference information from the user
screen via the HTML forms. This information retrieved is converted
into the conference_info type dict and then passed to
admin_conf_validation module for validation.
"""
get_conf_info_input():
      {
            SUBORDINATES:None
      }
      return conference_info
#################################################################

#################################################################
"""
This module takes the parsed information of the different fields of
the paper submission from the user screen via the HTML forms. This
information retrieved is converted into the paper type dict and then
passed to get_conf_user_paper_info module.
"""
get_paper_submission_by_user():
      {
            SUBORDINATES:None
      }
      return paper
#################################################################

#################################################################
```

```
"""
This module takes the parsed information of the different fields of
the review submission from the user (who is a reviewer for the
corresponding paper) screen via the HTML forms. This information
retrieved is converted into the review type dict and then passed to
get_conf_user_paper_info module.
"""
get_review_submission_by_reviewer():
      {
            SUBORDINATES:None
      }
      return review
#################################################################

#################################################################
"""
This module takes the parsed information of the different fields of
the rebuttal submission from the user (who is an author for the
corresponding paper) screen via the HTML forms. This information
retrieved is converted into the rebuttal_response type dict and then
passed to get_conf_user_paper_info module.
"""
get_rebuttal_response():
      {
            SUBORDINATES:None
      }
      return rebuttal_response
#################################################################

#################################################################
"""
This module takes the parsed information of the different fields of
the camera-ready and poster submission from the user (who is an
author for the corresponding paper and the paper is accepted) screen
via the HTML forms. This information retrieved is converted into the
cam_pos type dict and then passed to get_conf_user_paper_info module.
"""
get_cam_pos_submission():
      {
```

```
            SUBORDINATES:None
        }
        return cam_pos
##############################################################
```

Central transforms

```
##############################################################
"""
This is a top-level module for allocation of the conferences,
inviting reviewers and paper-related submissions. The submodules in
this directory perform the desired operations to obtain this
information and store them in the database.
"""
perform_assignment_conf_allocation(user_email_address, reviewer_list,
paper_list, conference_info, paper, review, rebuttal_response,
cam_posreviewer_paper_mapping, email_notif_status_code):
     {
          Subordinates:invite_reviewers(),reviewer_assignment(),
           allocate_new_conf(),paper_submission_in_conf(),
           author_response(), submit_review(), cam_pos_submission()
     }
     return email_notif_status_code, reviewer_paper_mapping
#####################################################################

#####################################################################
"""
This module takes the information about the users in the management
system and calls the subordinate function to perform an invitation to
be the reviewers.
"""
invite_reviewers(user_email_address):
```

```
     {
          Subordinates: email_invitation_for_review()
     }
     return email_notif_status_code
#########################################################################

#########################################################################
"""
This module sends the invitation emails to the users of the website
to be a reviewer for the given conference.
"""
email_invitation_for_review():
     {
          Subordinates: None
     }
     return email_notif_status_code
#########################################################################

#########################################################################
"""
This module calls the subordinate function to perform the first step
of reviewer assignment(automated assignment via algorithm) of the
papers of a given conference. It returns the final reviewer paper
"""
reviewer_assignment(reviewer_list, paper_list):
     {
           subordinates:automated_paper_reviewer_assignment()
     }
     return reviewer_paper_mapping
#########################################################################

#########################################################################
"""
This module performs the automated assignment of reviewers to the
papers via the algorithm, which is then passed to its subordinate for
manual confirmation
"""
automated_paper_reviewer_assignment(reviewer_list, paper_list):
     {
```

```
          subordinates:manual_reviewer_assignment()
     }
     return reviewer_paper_mapping

#####################################################################
"""
This module allows the conference admins (CAs) to perform the manual
assignment of the reviewers to papers, by changing the
reviewer_paper_mapping dictionary inplace and returning to its
parent.
"""
manual_reviewer_assignment(reviewer_paper_mapping):
     {
          subordinates: None
    }
    return reviewer_paper_mapping
#####################################################################

#####################################################################
"""
This module allocates the new conference in the database, after all
its validation has been finished
"""
allocate_new_conf(conference_info):
     {
          Subordinates: none
     }
     return conference_status_code
#####################################################################

#####################################################################
"""
This module allocates the new paper submitted to a given conference
in the database
"""
paper_submission_in_conf(paper):
     {
          Subordinates: none
     }
```

```
     return None
#################################################################

#################################################################
"""
This module stores the response of the author(rebuttal) in the
database.
"""
author_response(rebuttal_response):
     {
          Subordinates: none
     }
     return None
#################################################################

#################################################################
"""
This module takes care of the reviews submitted by reviewers for
papers and stores them in the database.
"""
submit_reviews(review)
     {
           Subordinates: none
     }
     return None
#################################################################

#################################################################
"""
This module takes care of the camera-ready poster submission of the
paper in a conference and stores them in the database.
"""
cam_pos_submission(cam_pos)
     {
           Subordinates: none
     }
     return None
#################################################################
```

OUTPUT

```
#####################################################################
"""
This module is a top-level module and calls its subordinates for
paper acceptance or rejection decisions, sharing the event details
with users of the conference, and lastly, allows data export by CAs.
"""
output_post_acceptance(reviews, cam_pos, ac_response)
     {
          Subordinates:paper_accept_reject_decision(),export_data(),
           event_notification()
     }
     return None
#####################################################################

#####################################################################
"""
The module displays the acceptance/rejection details of the papers,
via the HTML pages.
"""
paper_accept_reject_decision(ac_response)
     {
          Subordinates: None
     }
     return None
#####################################################################

#####################################################################
"""
This module allows the CAs to export the data of the accepted papers
in compressed format.
"""
export_data(cam_pos)
     {
          Subordinates: none
     }
     return compressed_data
```

```
#########################################################################

#########################################################################
"""
This module sends the event notification to the corresponding users.
"""
event_notification(event_details)
     {
          Subordinates: none
     }
     return formatted_email
#########################################################################
```

Summary table

Module	LOC
main	50
get_conf_user_paper_info	100
user_account_details	20
user_acc_validation	50
user_login_details	30
validated_conference_details	20
admin_conf_validation	50
get_conf_info_input	20
get_paper_submission_by_user	50
get_review_submission_by_reviewer	30
get_rebuttal_response	30
get_cam_pos_submission	30
perform_assigment_conf_allocation	30
invite_reviewers	30
email_invitation_for_review	20
reviewer_assignment	20
automated_paper_reviewer_assignment	150
manual_reviewer_assignment	50
allocate_new_conf	30
paper_submission_in_conf	30
author_response	20
submit_reviews	30

cam_pos_submission	30
output_post_acceptance	30
paper_accept_reject_decision	50
export_data	50
event_notification	150
Total size	1200

Total # of modules - 27
Expected LOC of software - 1200
(Note that this contains the expected LOC for core python code. Expected LOC for the frontend and UI is difficult to provide since it can easily change by factors of 2-3 because of the abstract nature of web dev aspects.
We still give a rough estimate for completeness. We expect 7-8 major static web pages in the project. Hence, a 1000 LOC HTML code is expected. Accordingly, 500 LOC CSS code is expected.
)

Top 3 modules in terms of fan-in and fan-out

Module - ***get_conf_user_paper_info***
Fan-in - 1
Fan-out - 6

Module - ***reviewer_conference_paper_info***
Fan-in - 1
Fan-out - 7

Module - ***output_post_acceptance***
Fan-in - 1
Fan-out - 3

Complex and error-prone modules

***Most error-prone*:**

1) get_conf_user_paper_info:
   get_conf_user_paper_info is the top-level module of input side factoring. Since it is calling various subordinate modules related to different inputs from the user, its fan out is 6, thereby increasing the dependence of the module on its subordinates. Also, this increases the information flowing into the module, thereby increasing the inflow. Thus the complexity of this module is high as compared to other modules. This is inherent to the problem statement and to handle all the various inputs, it's required.

2) reviewer_conference_paper_info:
   reviewer_conference_paper_info is the top-level module of central transform factoring. It is calling various subordinate modules for different tasks like reviewer assignment, various paper-related submissions allocation in the database. Thus, its fan-out is 7, thereby increasing the dependence of the module on its subordinates. Thus the complexity of this module is high.

Thus due to the high complexity of the above two modules, they are most error-prone.

Most complex:

1) output_post_acceptance_data:
   output_post_acceptance_data is the top-level module of output side factoring. Its fan_out is 3 as it is calling different subordinate modules depending on the different outputs which are compressed data of the conference, accept/reject the decision of all the papers, and all the events details for notifying the users. Thus outflow of this module is more as compared to others. Thus this module is complex relative to other modules.