PROJECT REPORT P05

UG Hostel Allocation and Management

Ankur Arora, Ankur Shukla, Anshika Mittal, Anushka Goyal

Group 05

TABLE OF CONTENTS

Clickable Table of contents and bookmarks in pdf

Software Requirements Specification

For

P05 UG Hostel Allocation and Management

Version 1.2

Prepared By Ankur Arora, Ankur Shukla, Anshika Mittal, Anushka Goyal

The LNMIIT, Jaipur

06-Nov-2015

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
	22-08-2015	Initial Preparation	V 1.0
	20-09-2015	Added business rules	V 1.1
	06-11-2015	Updated server specifications, updated business rules	V 1.2

1. Introduction

1.1 Purpose

This software package is developed from scratch exclusively for the *Hostel* in order to,

- Reduce the difficulties in record management-data redundancy, data update and data recovery part.
- Manual allocation of *rooms* in the *hostel* is time consuming and a waste of human and material resources.
- Reduce the difficulties in generating information about those *students* who had left the *hostel*.
- Lessen the difficulty of tracking the history of a *room*.
- Reduce the stress associated with searching for information on a *student* in a bundle of registers.

.

1.2 Document Conventions

The following documentation conventions are followed in preparing this SRS:

a) All key-words related to the *hostel*/College are formatted in italics.
b) The priority of a requirement is specified at the end of that requirement in curly braces and using the notation {Priority: nn}, where 'nn' is an integer in the range 00 (lowest priority) to 99 (highest priority).

1.3 Intended Audience and Reading Suggestions

This document is created for,

i) The *students* residing in the *hostel*, to provide them their *room* preferences and for the efficient allocation of *rooms*.
ii) The *wardens* and caretakers to review, allocate and update (if any) the *rooms*.
iii) The software development team for their use in analyzing the requirements.

1.4 Product Scope

i) To help the *students* in the *hostel* to securely access this '*Hostel* Management' software package in order to make their preferences of the *room*, subsequently know the status of the *rooms* allotted to them and avoiding any kind of *room* conflict among the *students*.

ii) To cater all activities of *Hostel* Allocation and Management, viz, from submitting the *room* preferences to allocation of *rooms* and removing the conflicts associated with it, subsequently knowing the *rooms* allotted to *students*.
iii) To help the *hostel wardens* and caretakers to know about the profile of a *student* and its respective *room* number.
iv) To cater the medical unit department for knowing the *room* no. of a particular *student* in case of a medical emergency.
v) To send important notices by *wardens* to *students.*

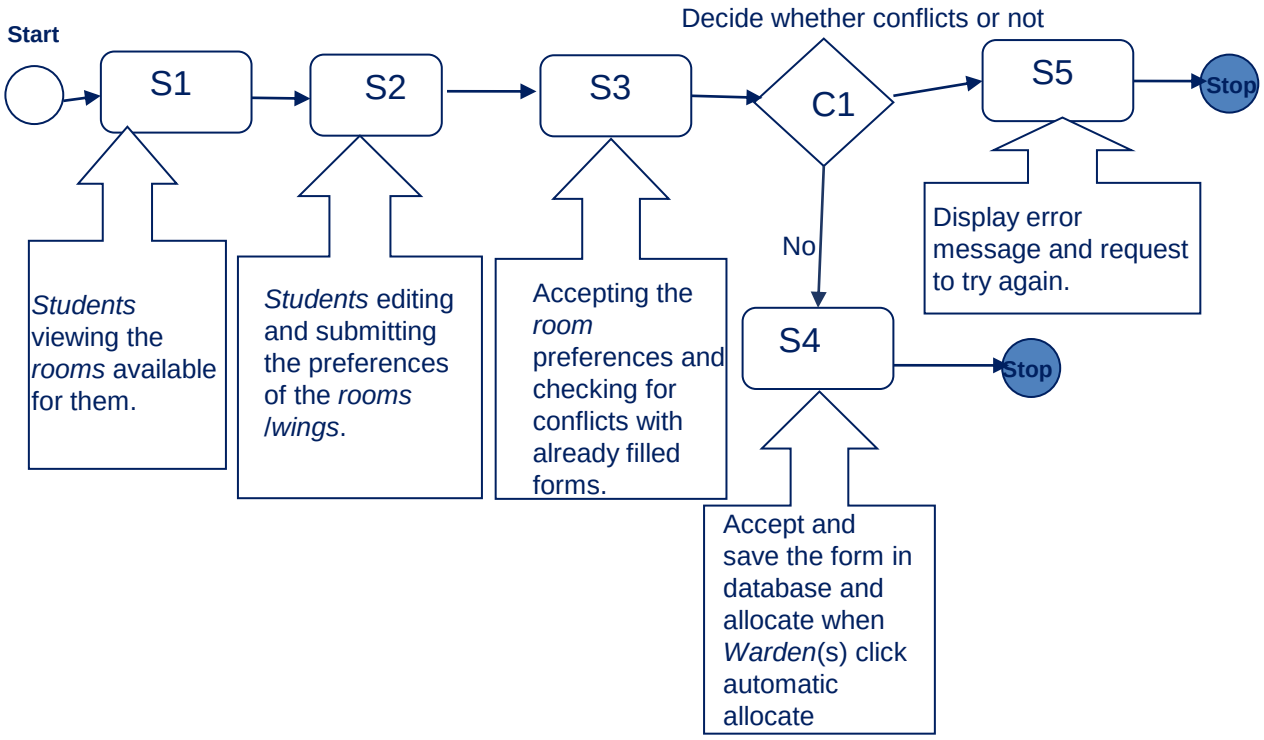
1.5 References

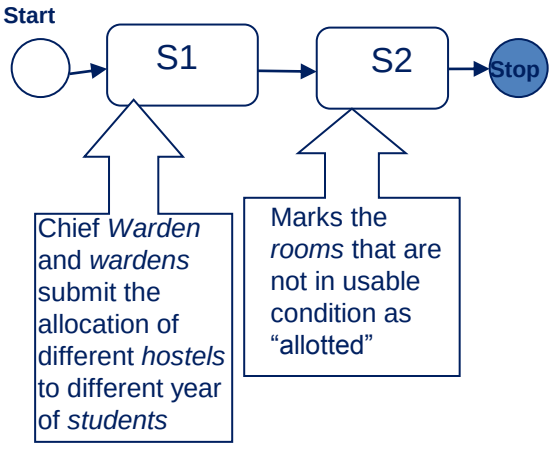
The following references are used in preparing this SRS:

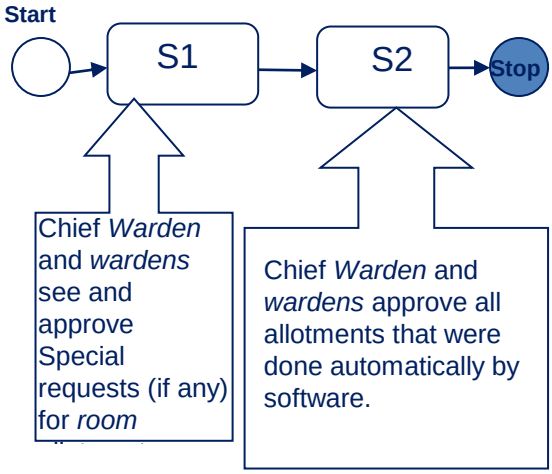
i) Minutes-of-the-Meeting between the LNMIIT SEPM Course instructor and *students* in this project, held on August 14th 2015.
ii) Minutes-of-the-Meeting between the Associate *Warden*/Chief *Warden* of the *hostels* and *students* in this project, held on August 16th 2015.

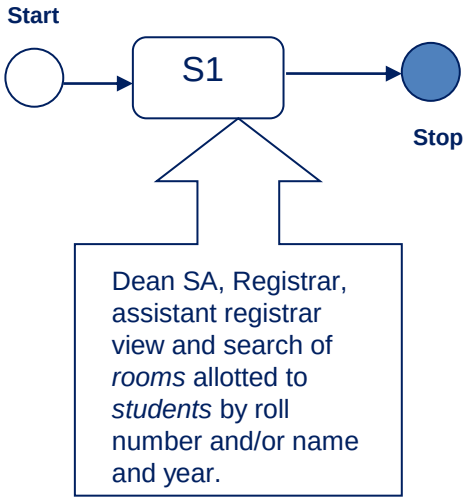
2. Overall Description

2.1 Product Perspective









2.2 Product Functions

This software package is expected to offer the following services:

a. For the *Students* of Institute:
    a. Facility to view their *hostels*, allotted *rooms*, *rooms* of other *hostels*, availability of different types of *rooms*.
    b. Facility to submit *wing* preference list and special request for *room* change.
b. For the *Wardens* of Hostels:
    a. Facility to view, modify and delete *wing* preference forms.
    b. Facility to start auto allocation of *rooms*.
    c. Facility to search in allotted *rooms* list.
    d. Facility to approve special requests of *room* change.
    e. Facility to send notifications to students.
c. For the *administration* and Medical unit
    a. Facility to search a *students*' *room* or search and know details of *residents*' of a particular *room*.

2.3 User Classes and Characteristics

This software package will be used by two categories of users:

a) ***Students* of Institute:** These users will use this software package to check their *hostel* allotments, fill *wing* preference form, view allotted *rooms* and request for change in *room* and search in the allotted *rooms* list.

b) ***Wardens*:** These users can allocate *rooms*, respond to special requests, view, search and modify the *rooms* list of each *hostel*.
c) **View only users:** *Dean SA, Registrar office, Medical Unit* comes under this group. They can only search for a *student* in the lists of *room* allocated to the *students*.

2.4 Operating Environment

The main component of the *Student* Management project is the software application, which will be limited to the Android operating system (specifically Android 4.0 and above). The application is not resource or graphics-intensive, so there are no practical hardware constraints. The app will rely on several functionalities built into Android's Application Programming Interface (API), so ensuring appropriate usage of the API will be a major concern. Beyond that, the application is a self-contained unit and will not rely on any other Android-related software components.

The application will, however, frequently interact with the database server. The database will be stored on the server using MySQL and PHP Rest API will interact with it using TP/IP (PDO). This API will be used by frontend (here Android App).

2.5 Design and Implementation Constraints

The primary design constraint is the mobile platform. Since the application is designated for mobile handsets, limited screen size and resolution will be a major design consideration. Creating a user interface which is both effective and easily navigable will pose a difficult challenge. Other constraints such as limited memory and processing power are also worth considering. This android application is meant to be quick and responsive, even when dealing with large groups and transactions, so each feature must be designed and implemented with efficiency in mind.

2.6 User Documentation

The primary goal of this application is to facilitate the process of managing *hostel* allocation. Consequently, the application will be designed to be as simple to use as possible. Nonetheless, users may still require some supplementary information about each component of this application system and this will be provided in a feature of this application: the Help menu.

2.7 Assumptions and Dependencies

The user is already having the user id and password required to access the application. In case the user does not have the user id and password he/she may contact the Institute.
All users (students) will be given double occupancy rooms. No student is a day scholar.

3. External Interface Requirements

3.1 User Interfaces

The set of User Interfaces consists of,

i) To Login and authenticate users.
ii) To Read, Edit and Submit *wing* preference forms and responses to them.
iii) To display all the required details of authentic users.
iv) To display options of search and approve to a group of users namely *Wardens*
v) To display the search results and allotment results in tabular format.
vi) To display option to send notification to users

3.2 Hardware Interfaces

A device running with **Android OS** having internet access through mobile network (2g, 3g or LTE) or wireless 802.11b/g.

3.3 Software Interfaces

This software package should have an interface with,

i) The Institute's *students* data Module
ii) The Institute's *Student* non-academic details Management Module
iii) The Institute's *wardens* and *administration* data module
iv) Software should be given full Network access permission in Android.

3.4 Communications Interfaces

This software package should be securely accessible through the internet communication channels (preferably wireless).

4. System Features

The requirements of this software package are described per each category of User:

i) All requirements of the *Wardens* and Caretakers of the *hostel*.

ii) All requirements of the *Students* residing in the *hostel*.

iii) All requirements of the *Staff* (*Dean Student affair*, *Medical unit and the Registrar*).

Business Use Case #1: All requirements of the *Wardens* and Caretakers of the *hostel*.

4.1 View, update and accepts the special response from the *students*.

4.1.1 Description and Priority

The TBD (to-be-developed) software package should facilitate the *Hostel staff* to,
i) Start the allocation process by specifying the start date.
ii) View the *room* nos. of a *student*
iii) Update the *rooms* allotted in case of any special request by the *students*.

4.1.2 Stimulus/Response Sequences

S. No.	Stimulus from the user	Response from the software
1.	Chief *warden* and *Wardens* accesses the *Hostel* Management software through the app;	Software displays the Login form for the *wardens*.
2.	Chief *Warden* and *Wardens* logs-in and fills the start date and end date of the allocation process;	Software displays the page with dates filled.
3.	*Wardens* will view the *room* changing form filled by the *students* and update the *room* nos. if they find any change and submit that.	Software will accept the changes and fix the final allotment of *rooms*.

4.1.3 Functional Requirements

As per the above table described in 4.1.2.

Business Use Case #2: All requirements of the *Students* residing in the *Hostel*.

4.2 Submit the *room* preferences

4.2.1 Description and Priority

The TBD (to-be-developed) software package should facilitate the *Students* to,
i) View the rooms and filling a wing form.
ii) Give the preferences of the *rooms*/*wings*.
iii) Submit the *room* preferences.
iv) View their *room* allotment.

The priority of this requirement is 99 (without the *students* there will be no allotment of *rooms*).

4.2.2 Stimulus/Response Sequences

S. No.	Stimulus from the user	Response from the software
1.	The *student* accesses the *Hostel* Management software through the app;	Software displays the Login form for the *students*.
2.	The *student* logs-in with their username and password assigned to them;	Software displays the screen with four options: a) Notifications b) *Wing* Form c) Search d) Special Request
3.	The *student* selects the "Wing Form" option;	Software displays the *wing* form that has to be filled by the *students* which includes the preferences section and the room nos.
4.	The *student* fills the wing form and clicks "Save";	The software displays the *wing* form with details and an option of editing the form.
3.	The *students* submits the *wing* form;	Software displays the submitted *wing f*orm. Software will validate that there are no conflicts, viz, a *students*' name is not filled twice. If a *students*' name is filled twice then there are errors, the software will re-display the screen with the roll nos. of students causing conflicts.
4.	If errors are displayed, the *students* will correct the errors and re-submit the preferences;	Software will display a message that you have successfully filled the form and your preferences are being taken.

5.	The *students* review the *rooms* allotted to them according to their preferences after all the *wing*/*room* forms are being filled;	Software will display a screen where you can find your allotted *room*.
6.	After allotment if a *student* does not like his/her *room*.	Software displays a *room* changing form after the allotment asking for the particular reason of *room* change. (Special request for *room* change.

Business Use Case #2: All requirements of the *Staff* (Dean *Student* affair, Medical unit and the Registrar)

4.3 View and search the *rooms* allotted to the *students*.

4.3.1 Description and Priority

The TBD (to-be-developed) software package should facilitate the user group having Dean SA, Registrar and assistant registrar to,
i) Search and view *rooms* allotted to different *students*.

4.3.2 Stimulus/Response Sequences

S No.	Stimulus from the user	Response from the software
1.	Dean SA, Registrars and others with view only rights accesses the *Hostel* Management software through the app;	Software displays the Login form for the *wardens*.
2.	Dean SA, Registrars and others with view only rights logs-in and view the whole list or search for a particular *students room*.	Software will give information of the *room* and *student*(s)

4.3.3 Functional Requirements

As per the above table described in 4.3.2.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Each transaction (situation-response case), on an average taken for a duration of 100 hours or till 1000 transactions are successfully completed, should have a turn-around time of less than 5 seconds under the following load on the software:

i) There are 100 users accessing this software per hour.
ii) There are 1000 users registered to use the software.

5.2 Safety Requirements

Almost all of the data related to the would be stored securely in database tables whose access would be limited to the administrators;

5.3 Security Requirements

This software should,

i) Authenticate each user, who logs in;
ii) When the user performs any actions, authorize him / her to perform the actions allowed for the user and display an error message if found to be not authorized;

5.4 Software Quality Attributes

5.4.1 Reliability

A failure should not occur while the user is submitting the wing form.

5.4.2 Extensibility/Maintainability

The software will be scalable and any further changes can be implemented depending upon the level of modification required.

5.4.3 Usability

This software is being offered to be used in a university scenario and the users of this will be hostel residents (under graduate students, institute's faculty and staff) and the system administrator.

5.4.4 Availability

The software related to this particular module comes in the Android package.

5.4.5 Security

The module must not be accessible by the outsiders (any person who is not a part of that institute) and hence, an authorization service is being implemented, so that only those students/instructors who are having a valid Username and Password will be given access to this module.

5.4.6 Visual Aesthetics

The application will be aesthetically appealing to the user in order to make the application more desirable. We will test whether or not we have successfully created an aesthetically appealing application by using team members as test subjects.

5.4.7 Connectivity

The application will use online Hosting Hostinger (free version). Therefore it is not much reliable and needs manual backups. Upgradation to a paid hosting service will make sure that there is 99.99% availability of server.

5.5 Business Rules

Wardens can access, modify and update the *room* allotment. *Students* can only submit requests so that they will be approved by *Warden*.
Students can only submit request one time to change the allotted room and hostel.
Medical unit and *administration* has only viewing rights.
There are total x *rooms.*
The *rooms* are divided into n groups called *wings*.
These *wings* have integer less than equal to x/n *rooms.*
The values of x and n are *hostel* and *floor* specific.
Single rooms are first given to *Final year students* and unused single *room(s)* are kept for special cases or given to *third year student(s).*
Students can only submit request one time to change the allotted *room* and *hostel* so the final *roommate/room no.* once allotted will not be changed back for that year/sem.

6. Other Requirements

NIL

Appendix A: Glossary

NIL

Appendix B: Analysis Models

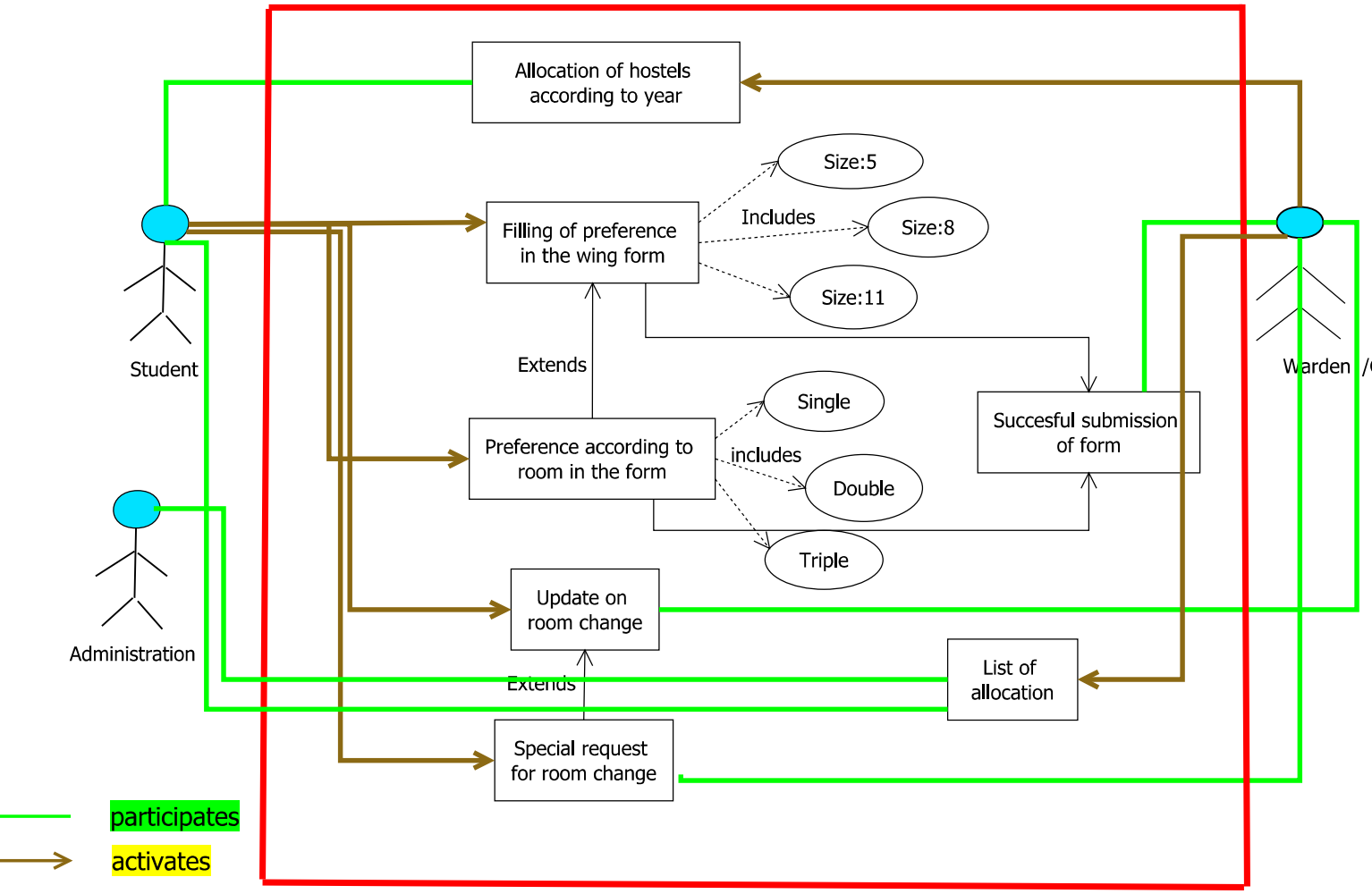
NIL

Appendix C: To Be Determined List

NIL

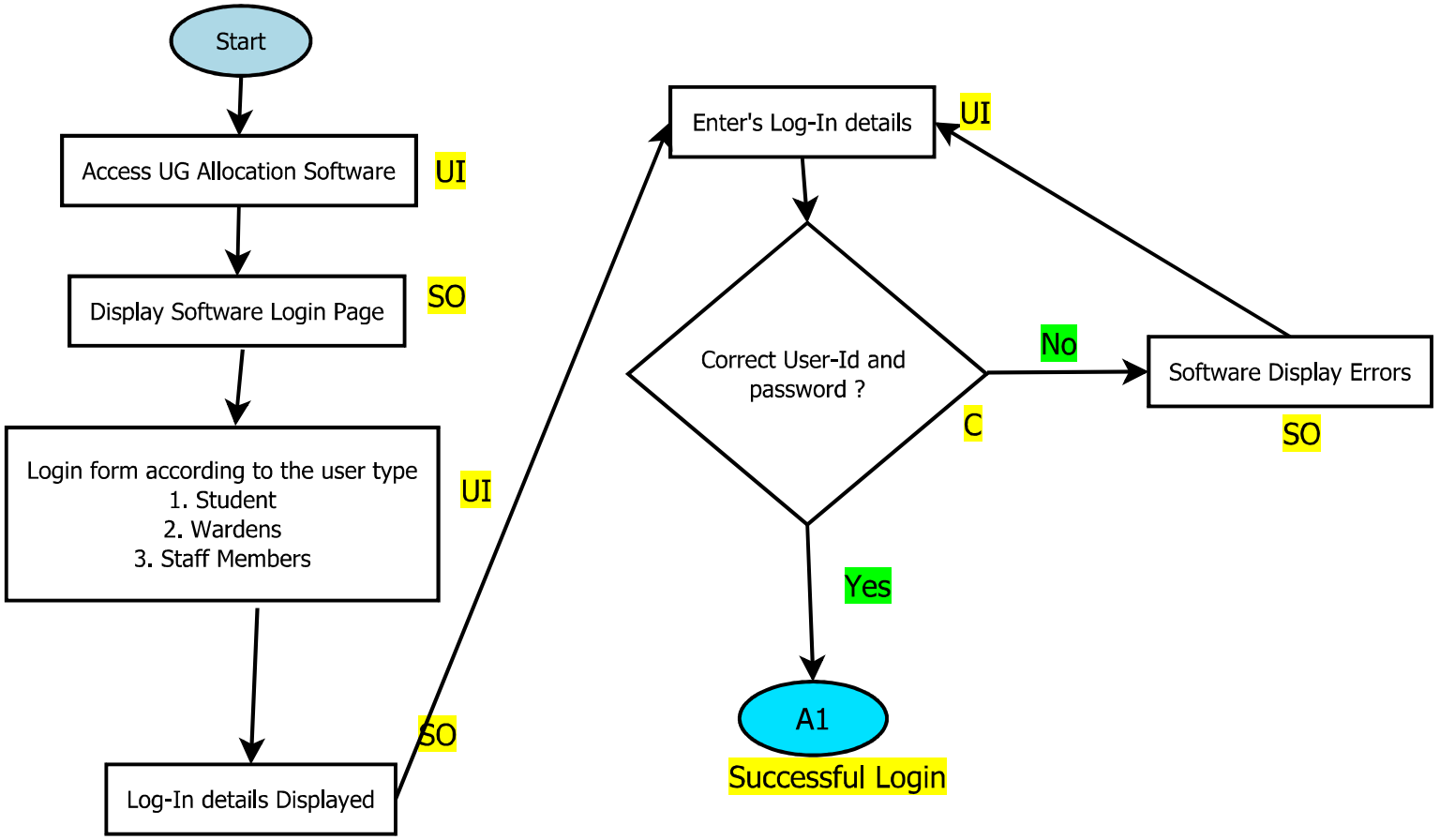
USE CASE OBJECT DIAGRAM

OBJECT: ALLOCATION

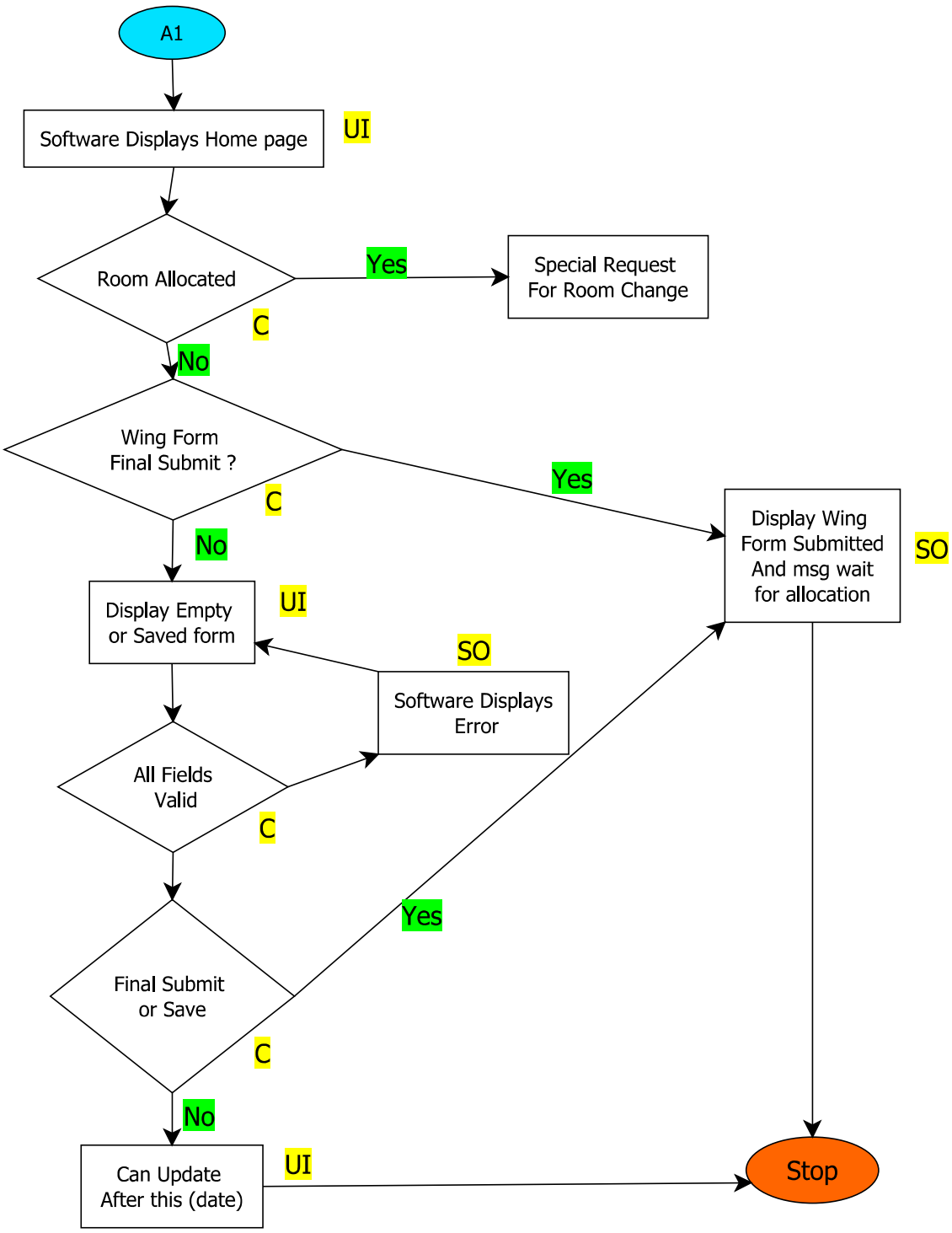


USE CASE ACTIVITY DIAGRAMS

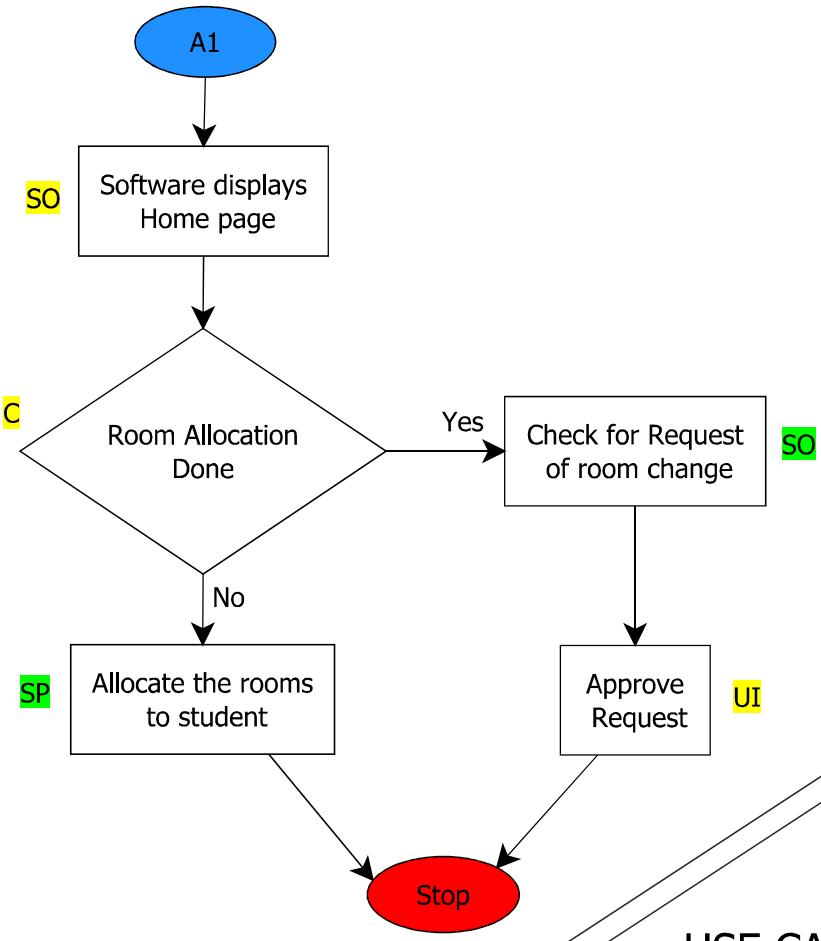
USE CASE ACTIVITY DIAGRAM FOR LOGIN



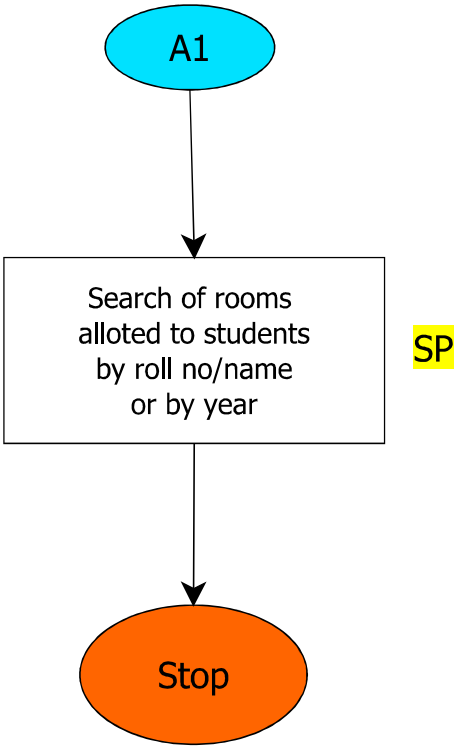
USE CASE ACTIVITY DIAGRAM FOR STUDENTS



USE CASE ACTIVIY DIAGRAM FOR WARDENS/CARETAKERS



USE CASE ACTIVIY DIAGRAM FOR ADMINISTRATION



PART 2

2. a) TEST CASES WITH RESULTS

#	TS1
Title	Verify "Wing allocation" facility for user: students
Description	To test the different scenarios that might arise while an user is trying to view and submit wing allocation form

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output	RESULTS Test A	RESULTS Test B
TC1	Verify that user is able to go to wing allocation form from dashboard		Student User is logged in and wing allocation activity is active and user has not yet saved or submitted the form	User sees wing allocation form	Click on wing allocation in dashboard	"Wing form" page for the user is displayed	P	P
TC2	Verify that user is able to save the form	TC1	Student User is logged in and wing allocation activity is active and user has not yet saved or submitted the form	User fills and saves wing allocation form	1. Fill 0 or more fields 2. Click on save form	The "Wing Form" dialog is shown with a *"Form saved successfully!"* message	P	P
TC3	Verify that user is able to view the saved the form	TC1 and TC2	Student User is logged in and wing allocation activity is active and user has only saved the form	User views and edits the saved form	1. Click on wing allocation in dashboard 2. Click on edit	"Wing form" page for the user is displayed with fields retreived from the saved form	F	P
TC4	Verify that user is able to submit the form	(TC1 & TC2) or TC3	Student User is logged in and wing allocation activity is active and user has not yet submitted the form		1. Click on wing allocation in dashboard 2. fill form and click on submit	The "Wing Allocation" dialog is shown with a *"submitted successfully!"* message	P	P
TC5	Verify that user is not able to go to wing allocation form from dashboard when allocation activity is inactive		Student User is logged in and wing allocation activity is inactive or ended		Click on wing allocation in dashboard	The "Wing Allocation" dialog is shown with a *"wing allocation process is not active!"* message	F	P

P: Passed
F: Failed

#	TS2
Title	Verify "notification" facility
Description	To test the different scenarios that might arise while an user is trying to view notifications

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output	RESULTS Test A	RESULTS Test B
TC1	Verify that user is able to go to notifications from dashboard		User is logged in and there are unread notifications	User sees notifcations	1. Click on notification in dashboard	"Notification" page for the user is displayed	P	P
TC2	Verify that user is able to go to notifications from navigation drawer		User is logged in and there are unread notifications	User sees notifcations	1. Click on notification in navigation drawer	"Notifications" page for the user is displayed	--	--
TC3	Verify that mark as read functionality in notifications is working	TC1 or TC2	User is logged in and there is atleast one notification	Number of notifications gets reduced	1. Click on notification in dashboard 2. Click on notification 3. Click on the 'Mark as read ' button	The notification which is "marked as read" gets hidden and number of unread notifications reduces by one	--	--
TC4	Verify that user sees no unread notifications message after click on notifications		User is logged in and there is no unread notification		1. Click on notification in dashboard	The "Notification" dialog is shown with a *"No unread notifications!"* message	--	--

Removed feature : --

P: Passed
F: Failed

#	TS3
Title	Verify "wing allocation process" facility for User: wardens
Description	To test the different scenarios that might arise while an user is trying to start wing allocation process

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output	RESULTS Test A	RESULTS Test B
TC1	Verify that user is able to view "wing allocation process" from dashboard		Warden User is logged in and wing allocation activity is inactive	User sees "wing allocation process" form	1. Click on "wing allocation process" in dashboard	"Wing form" page for the user is displayed	P	P
TC2	Verify that user is able to fill and submit the form	TC1	Student Warden is logged in and wing allocation activity is inactive	User submits details temporarily and views them	1. Fill start date, end date (start date>= current date && end date >start date) and select hostels for different years(overlapping allowed) 2. click on "submit"	The "Allocation process" screen with all filled details is displayed	P	P
TC3	Verify that user is able to submit "allocation process details" and send "notification"	TC1 and TC2	Student Warden is logged in and wing allocation activity is inactive	User views details filled and submits and sends notification	1. Click on "submit and send notification"	the allocation process is set to active an all users are given a notification	P	P
TC4	Verify that user is able to view, edit and approve Auto allocated list		Student Warden is logged in and wing allocation activity is ended and user has not yet approved the auto allocated list	Final list is approved and all users are notifies	1. Click on auto allocated list 2. click on edit 3. click on approve	All users are notified about the final allocated list	P	P

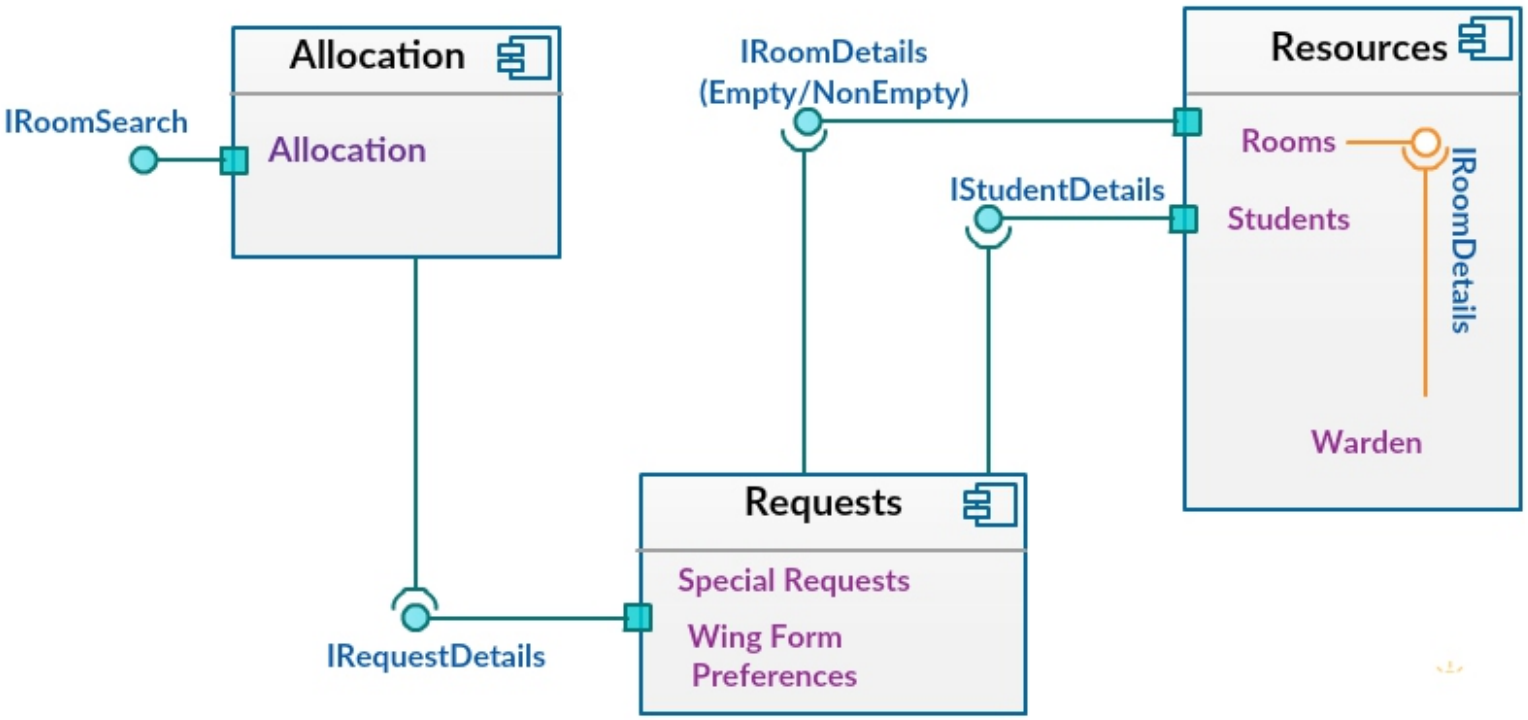
P: Passed
F: Failed

#	TS4
Title	Verify "special request" facility for User: students and wardens
Description	To test the different scenarios that might arise while an user is using special request facility

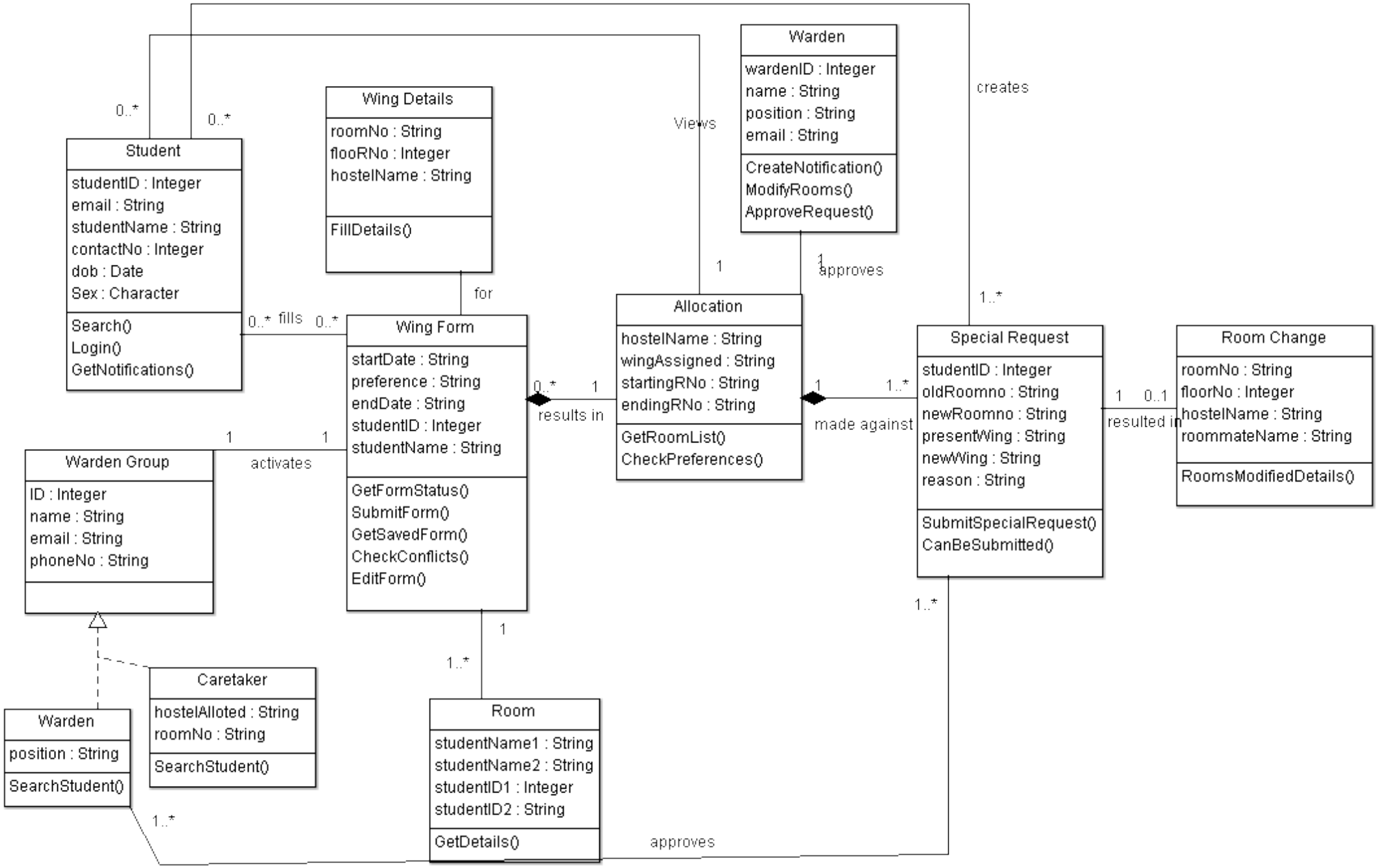
#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output	RESULTS Test A	RESULTS Test B
TC1	Verify that user is able to view and submit "special request" from dashboard		Student User is logged in and has no pending request	User sees and submits special request form, also the warden users are given notification	1. Click on "special request" in dashboard 2. Fill reason for room change request and submit	"Special request" form is displayed, edited and submitted giving message "submitted successfully!"	P	P
TC2	Verify that user is not able to submit "special request" when a request is pending	TC1	Student User is logged in and has a pending request	User sees message that his request is already submitted	Click on "special request" in dashboard	The "special request" dialogue with message "you already have a pending request! Be patient!" is displayed	F	P
TC3	Verify that user is able to see and respond to special request	TC1	User Warden is logged in and there is a pending request	User views and responds to special request which is also sent to the user student who created the request	1. Click on "special request" in dashboard 2. fill response and click respond	The special request is marked completed and user is shown message "response sent!"	P	P
TC4	Verify that user is able to see and approve special request	TC1	User Warden is logged in and there is a pending request	User views and approves special request which is also sent to the user student who created the request	1. Click on "special request" in dashboard 2. Select a special request 3. Click on approve	The special request is marked completed and user is shown message "request approved!"	P	P

P: Passed
F: Failed

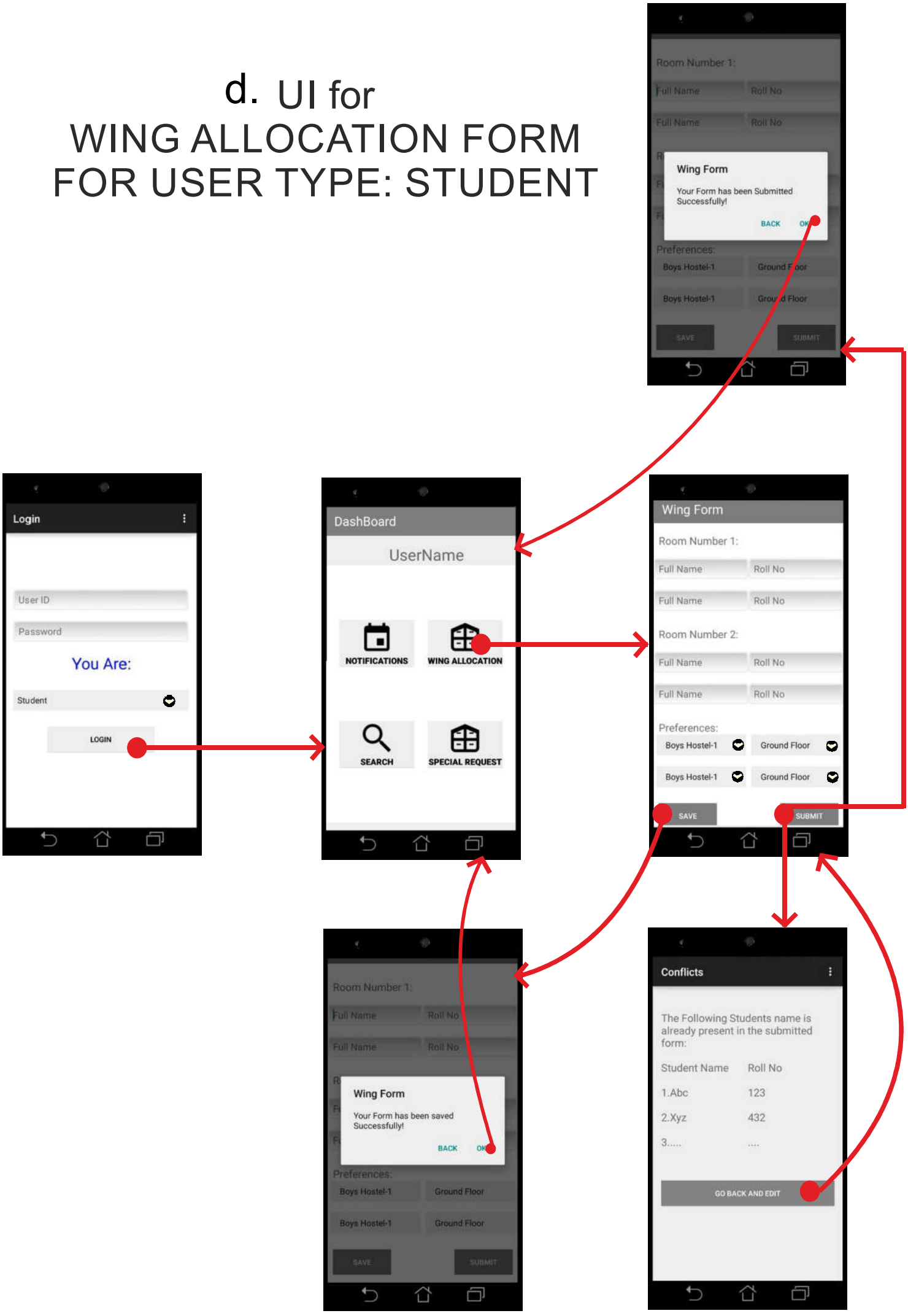
2. b) Component Diagram(revised)



2. c) Class Diagram(revised)

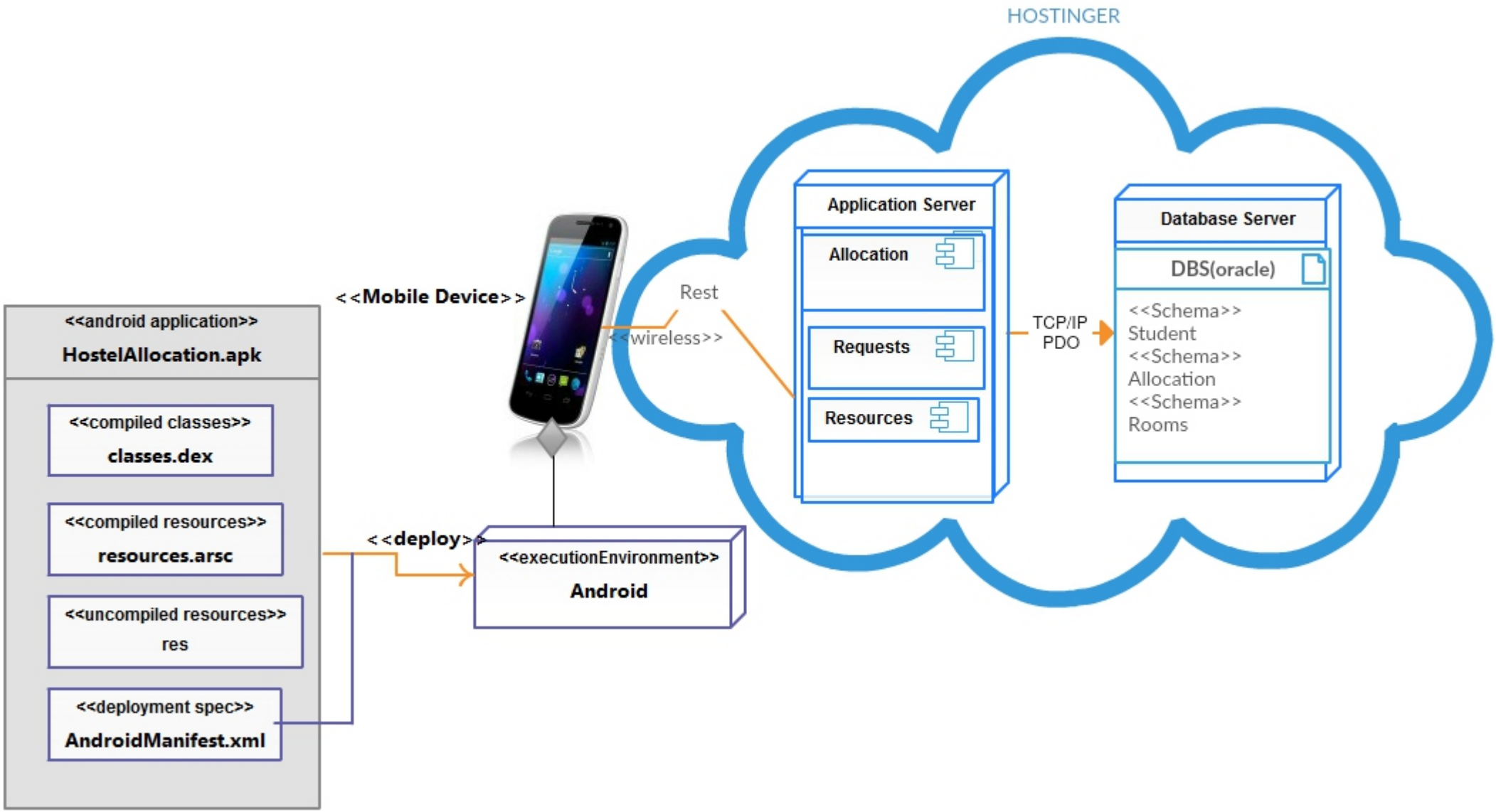


d. UI for WING ALLOCATION FORM FOR USER TYPE: STUDENT



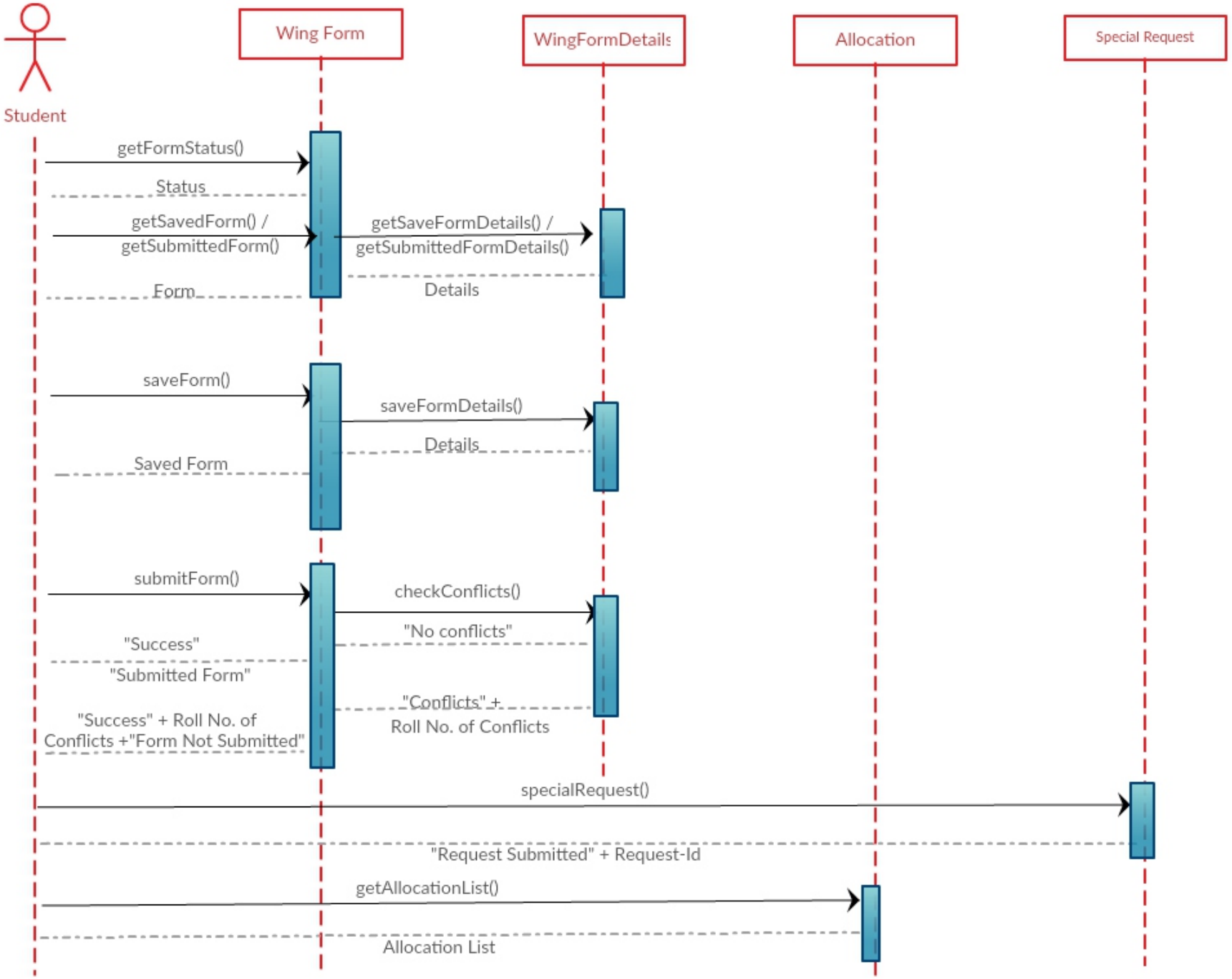
PART 3

3. a) Deployment Diagram



3 b Sequence Diagram for Student

wing form submission, searching in allocation, submitting special request



3c code snippets

PHP backend API code

```php
<?php
session_start();
$response = array();
// post checkform=anything, uid = id of user (student)
// response
// {
//   "success": 1/0
//   "message": (when success 1) "noFormPresent"/"savedFormPrese
// }
require 'connect.php'; //estabilishes connection with DB
if(isset($_POST ['checkform'])){
    $uid = !empty($_POST['uid']) ? trim($_POST['uid']) : null;//
    $sql = "SELECT COUNT(*) AS num FROM wingform where sid= :uid
    $stmt = $pdo->prepare($sql);
    $stmt->bindValue(':uid', $uid);
    $stmt->execute();

    $row = $stmt->fetch(PDO::FETCH_ASSOC);
    if($row['num']>0){
        $response["success"] = 1;
        $response["message"] = "submittedFormPresent";
        echo json_encode($response);
    }
    else{//Checking for saved form
        $sql = "SELECT COUNT(*) AS num FROM savedwingform where
        $stmt = $pdo->prepare($sql);
        $stmt->bindValue(':uid', $uid);
        $stmt->execute();

        $roww = $stmt->fetch(PDO::FETCH_ASSOC);
        if($roww['num']>0){
            $response["success"] = 1;
            $response["message"] = "savedFormPresent";
            echo json_encode($response);
        }
        else if($roww['num']==0){
            $response["success"] = 1;
            $response["message"] = "noFormPresent";
            echo json_encode($response);
        }
        else
        {
            $response["success"] = 0;
            $response["message"] = "Unknown Error";

            echo json_encode($response);
        }
    }//end of checking for saved form
}
else
{
    $response["success"] = 0;
    $response["message"] = "Unknown Error 2";

    echo json_encode($response);
}
```

XML in Android for UI elements

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    android:background="@color/background"
    tools:context="com.example.ankurshukla.hostel.Activity.Student_Dashboard">

    <!-- including the customized toolbar replacing the actionbar
    already present -->
    <include layout="@layout/actionbar"/>

    <!-- Displaying the username of login person   -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="UserName"
        android:layout_marginTop="15dp"
        android:background="@color/background"
        android:layout_gravity="center"
        android:id="@+id/display_sname"
        android:gravity="center"
        android:textSize="30dp"/>

    <!-- for displaying the no of notification
    if present so that student can check the notifications is present   -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/student_notify"
        android:textSize="22dp"
        android:paddingLeft="10dp"/>

    <!--including layout which has all the buttons
    through which student can call and do all functions present in the app
    by clicking on the various button present-->
    <include layout="@layout/fragment_student"/>

    <!-- end of layout-->
</LinearLayout>

```

Java Class in Android

```
37
38  public class Student_Dashboard extends AppCompatActivity {
39
40      Button wing,notification,search,special_req;//buttons by which student will access the different functionality
41      //wing button direct it to the wing form if allocation has started and redirect to the wing form or saved form as present
42      //notification show the no of notifications if present
43      //search allows them to search for students room no hostel after allocation
44      //special req allows them to end request for room change after allocation
45      TextView name,student_notify;
46      // name testview showing the name of login user after taking from server
47      //notify tells the no of notification if present
48      String rqid;
49      //rqid is used to redirect the submitted request if present else redirect them to fill one request
50
51      @Override
52      protected void onCreate(Bundle savedInstanceState) {
53          super.onCreate(savedInstanceState);
54          Intent intent = getIntent();//for taking value of rqid whether it is present or not
55          rqid = intent.getStringExtra("rqid");
56          setContentView(R.layout.activity_student__dashboard);
57
58          android.support.v7.app.ActionBar actionBar=getSupportActionBar();
59          actionBar.hide();
60
61          wing= (Button) findViewById(R.id.btn_student_wing);//link of the wing form button in xml to use in java
62          notification= (Button) findViewById(R.id.btn_student_notify);//link of the notification  button in xml to us in java
63          search = (Button)findViewById(R.id.btn_student_search);//link of the w search button in xml to us in java
64          name=(TextView)findViewById(R.id.display_sname);//link of textview to show name of the login user
65          student_notify = (TextView)findViewById(R.id.student_notify);//no of notifications is hshown by this
66          special_req = (Button)findViewById(R.id.btn_student_special);//link of the special request form button in xml to us in jav
67
68          String display = AppController.getString(Student_Dashboard.this, "username");//taking the name of user stored in the phone
69          name.setText(display);//displaying the name of user in dashboard
70
71          //number takees the number of notification in database and according to it is showing the no of notification if present
72          final String number = AppController.getString(Student_Dashboard.this, "noOfNotify");
73          if(number.equals("0")){
74              student_notify.setText("");
75          }else{
76              String notifym_msg = "You Have total " +number+ " number of Notifications!";
77              student_notify.setText(notifym_msg);
78          }
79
80          //checkform is called first to check whether is there any saved or submiited form present or not
81          //when clicked on wing from button and checkform receives the response from server according to which it is redirected
82          wing.setOnClickListener(new View.OnClickListener() {
83              @Override
84              public void onClick(View v) {
85
86                  String uid = AppController.getString(Student_Dashboard.this,"Student_id");
87                  CheckForm(uid);
88              }
89          });
90
91          //it shows the notification if present by calling get notification method when clicked else show msg that no notification
92          notification.setOnClickListener(new View.OnClickListener() {
93              @Override
94              public void onClick(View v) {
95                  if(number.equals("0")){
96                      AlertDialog.Builder alertdialogBuilder=new AlertDialog.Builder(Student_Dashboard.this);
```

3. d) Conclusion

The android application "*UG Hostel Allocation and Management*" is a user friendly application for managing hostel allocation process in Institutions. It has been designed to automate and look after the overall processing of records of students residing in a large hostel. The app enables the user to submit the room allocation form and also to search for student details in a particular room. The developed system provides solution to manual hostel allocation problems and also provides information such as hostel information and hostel room information. It facilitates the administration to provide quick and error less allotment of rooms for the students.

The software offers effectiveness, usability, stability and time management. It provides the most flexible and adaptable standards allocation system solutions for hostel.

Current Limitations:

1. Available for android only.
2. Warden can change room of students but not room mate.
3. There is no dynamic searching (no recording of keystrokes till search clicked).
4. Roll no should be entered very carefully.

Innovative features:

1. Wardens can send notifications (messages) for a particular hostel.
2. It enables a student to save/edit a wing form and submit it afterwards.
3. Students can send any type of request to warden(meeting, room problems, roomie problems etc). These features are part of complaint management.
4. It enables the medical unit and other administration to search for a resident based on name, contact or roll no and app will give details to contact his/her guardians or roommate and also provides interface to other systems to search in allocated list.
5. User can send feedback to app developer as the app is in beta version.
6. Backend of this app is written in such a way that making web app and iOS app for the same project is very cheap (in person hours).

Extensions:

1. By providing different types of rooms (single, double and triple).
2. Assigning hostels according to respective years.
3. Complaint management of hostel.
4. Inter hostel and intra hostel events.
5. Better searching algorithms to get faster results.