Indian Institute of Information Technology, Allahabad

Software Engineering

Instructors: Dr. Sonali Agarwal

****

**SOFTWARE DESIGN SPECIFICATION**

**ADOS - Automatic Door Closing System**

**GROUP MEMBERS**

Navneet Singh IIT2019056

Gautum Kumar IIT2019087

Ritik Kumar IIT2019088

Subham Kumar IIT2019093

TABLE OF CONTENT

**1. Introduction**

**1.1 Purpose of this document**

**1.2 Scope of the development project**

**1.3 Definitions, acronyms, and abbreviations**

**1.4 References**

**1.5 Overview of document**

**2. Logical Architecture**

**2.1 Logical Architecture Description**

**2.1.1. ER Diagram**

**2.1.2. Sequence Diagram**

**2.1.3. State Diagram**

**2.1.4 Class Diagram**

**2.2 Class Name:Registered**

**2.3 Class Name:Inside\_building**

**2.4 Class Name:Company**

**2.5 Class Name:fragOne**

**2.6 Class Name:fragTwo**

**2.7 Class Name:ExampleJobService**

**3 Execution Architecture**

**4 Design decisions and tradeoffs**

**5 Pseudocode for components**

**5.1 Class Name:**

**5.2 Class Name:**

**5.3 Class Name:**

**5.4 Class Name:**

**5.5 Class Name:**

**5.6 Class Name:**

**5.7 Class Name:**

**5.8 Class Name:**

**5.9 Class Name:**

**5.10 Class Name:**

**6 Appendices (if any)**

**The Software Design Specification**

**1. Introduction**

It is basically a system which is controlling the entry gate of the building. It will allow only verified users to enter the building in their given time slots and days and will notify them if their time slot expires and they are still occupying the building while keeping a flag of their presence in the building. It will also register new users after taking manager’s permission.

**1.1 Purpose of this document**

Generally, manually checking the id and identifying if it is a genuine user or not for each individual entering the gate by the security guard is inefficient and has security issues. Also many people occupy the building even after their time slot has expired. So in order to resolve these issues we have made this application.

**1.3 Definitions, acronyms, and abbreviations**

IEEE: Institute of Electrical and Electronics

Engineers SDS: Software Design Specification

ADOS: Automatic door opening system

**1.4 References**

**1.4.1** IEEE SDS template

**1.4.2** R. S. Pressman, Software Engineering: A Practitioner's Approach, 5th Ed, McGraw-Hill, 2001.

**1.5 Overview of document**

This SDS is divided into seven sections with various subsections. The sections of the Software Design Document are:

**1.5.1** **Introduction**: describes about the document, purpose, scope of

development project definitions and abbreviations used in the

document.

**1.5.2** **Conceptual Architecture/Architecture Diagram :** describes the

overview of components, modules, structure and relationships and user

interface issues.

**1.5.3** **Logical Architecture : describes Logical Architecture Description and**

**Components.**

**1.5.4** **Execution Architecture :** defines the runtime environment, processes,

deployment view.

**1.5.5** **Design Decisions and Trade-offs :** describes the decisions taken

along with the reason as to why they were chosen over other

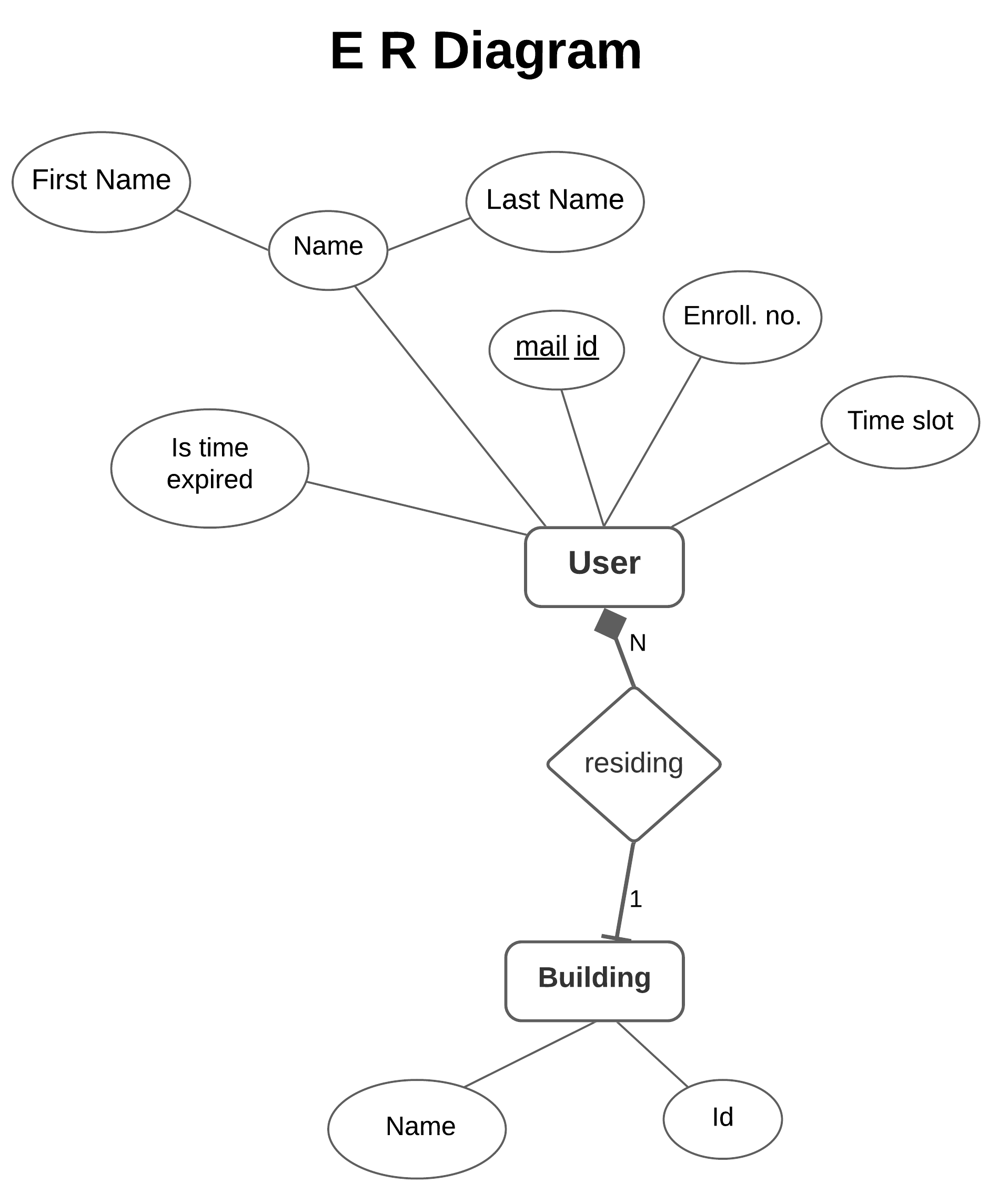
alternatives.

**1.5.6** **Pseudocode for components** : describes pseudocode, as the name indicates.

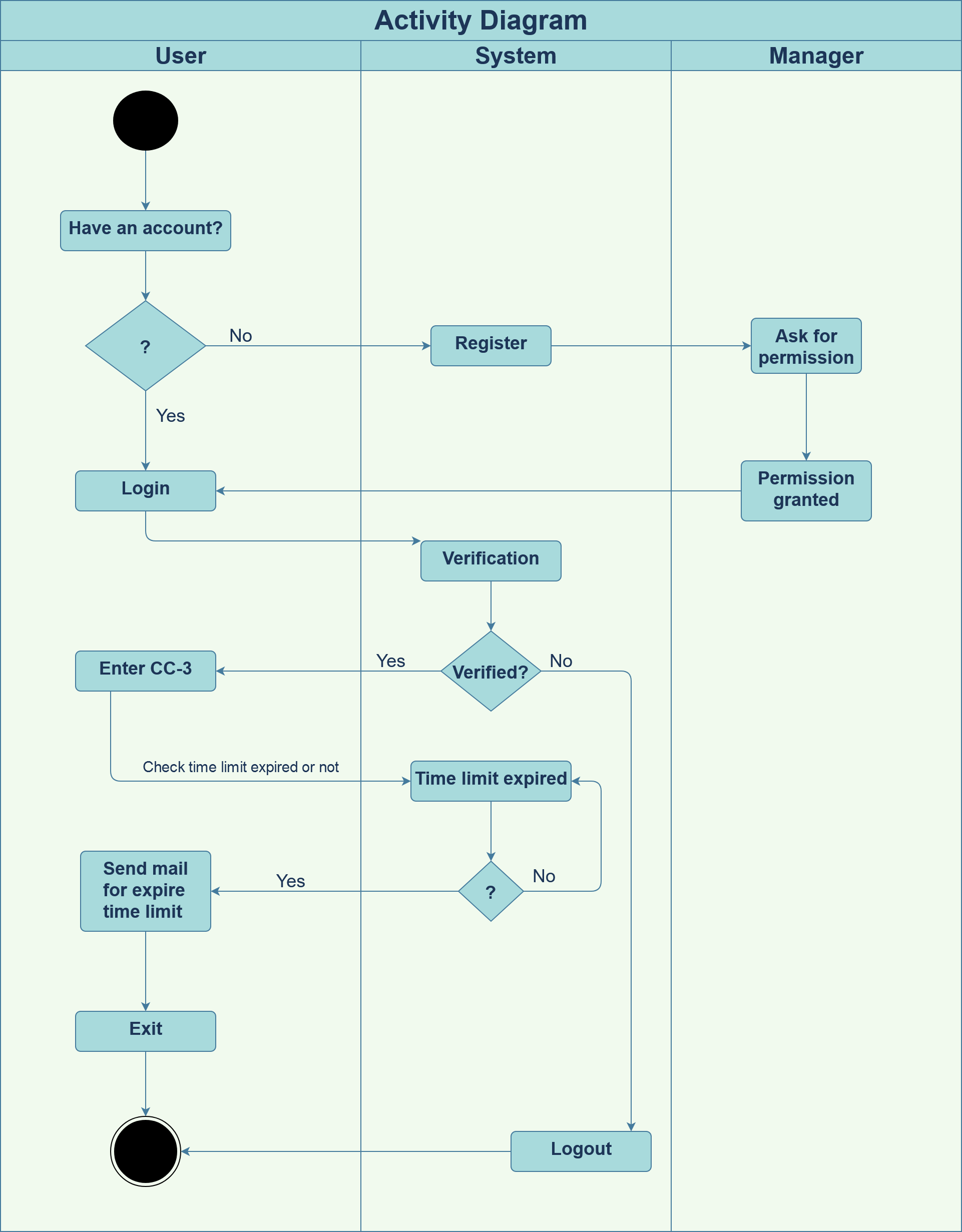
**1.5.7** **Appendices** : describes subsidiary matter if any.

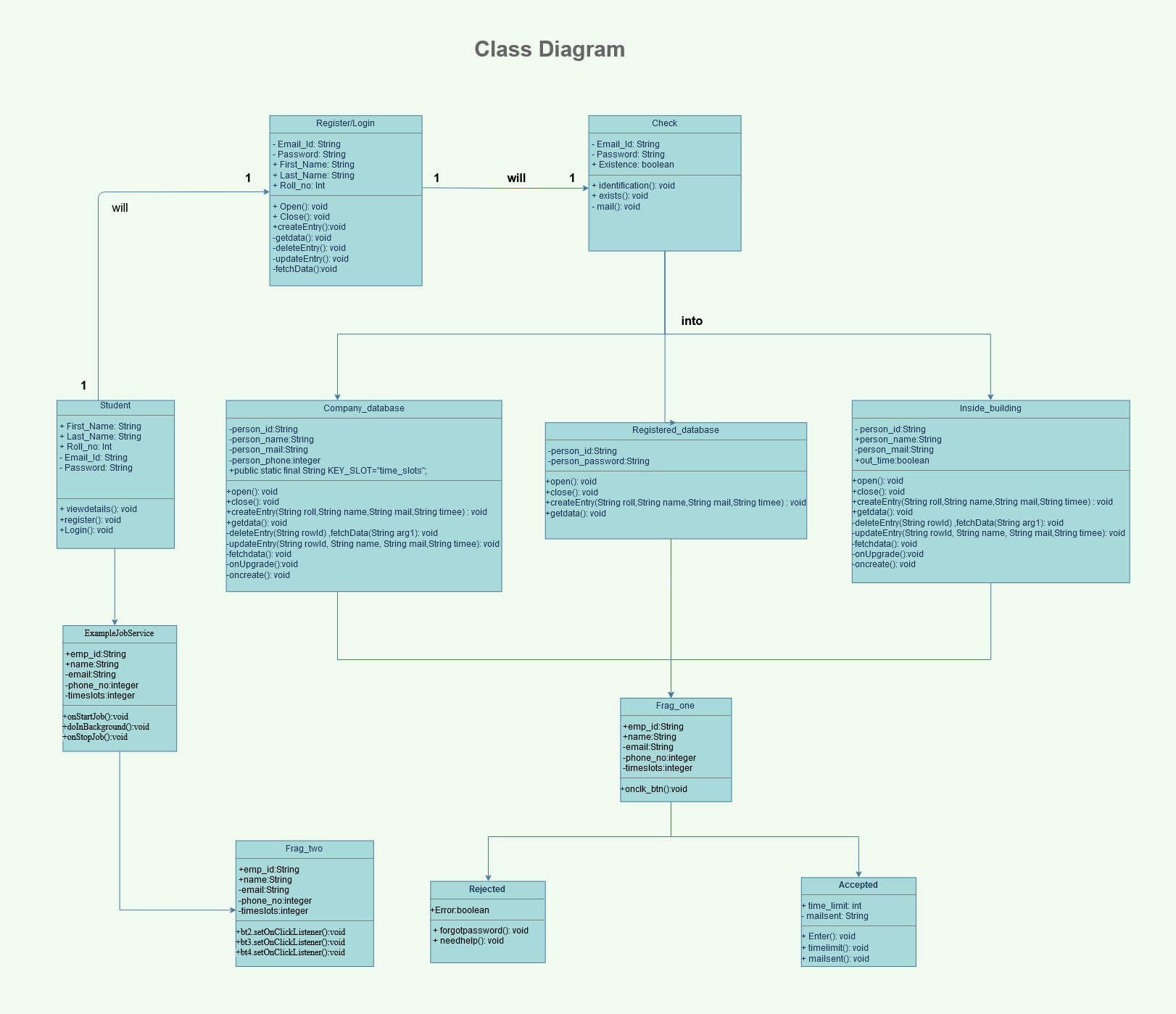
**2. Logical Architecture**

**E R Diagram:**

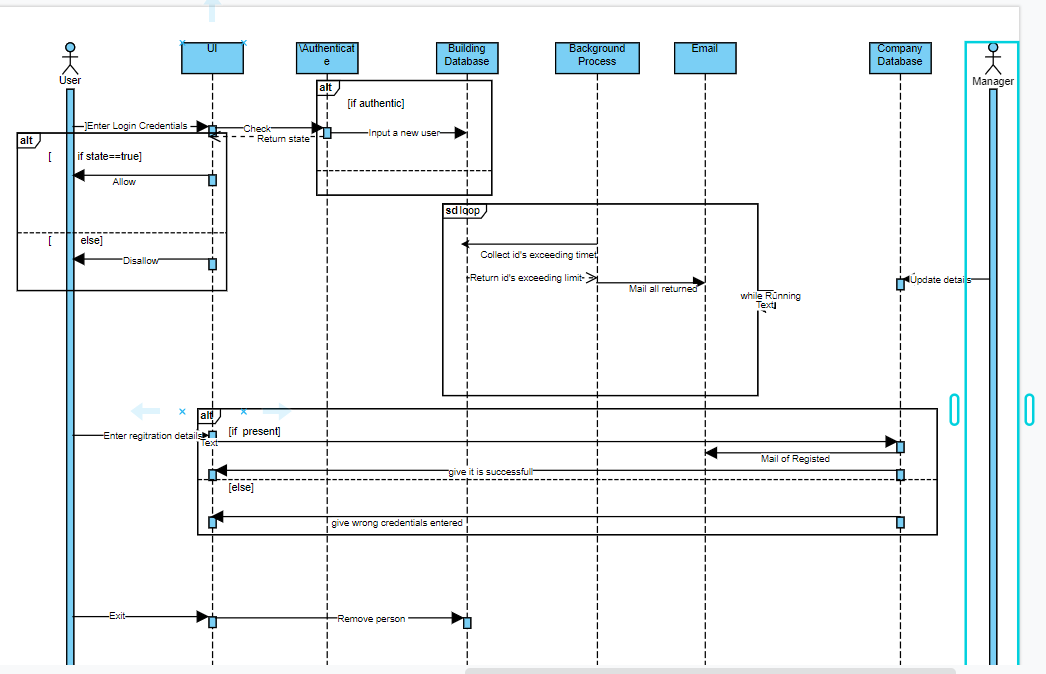
****

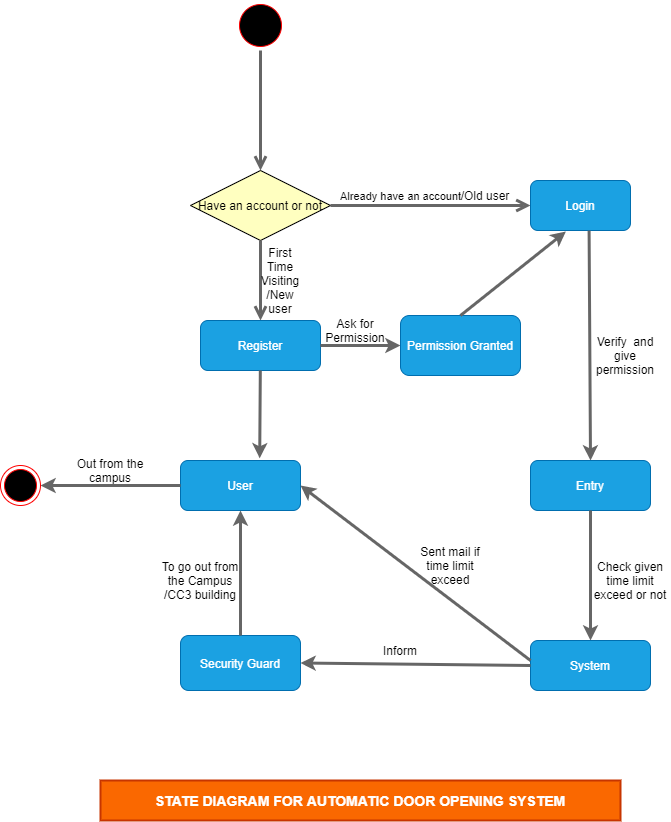
**Activity Diagram:**

****

**Class Diagram:**

**Sequence Diagram:**

****

**State Diagram:**

**2.1 Logical Architecture Description**

**2.1.1. ER Diagram Explanation:**

There is an entity named User which is at the root of the diagram and it has an attribute named mail\_id, name(First\_Name, Last\_Name), User\_Id, is\_time\_required, Time\_slot.

After selecting the User\_Id and mail\_id from the list of datasets the corresponding dataset is fetched into the software.

The entity Building(CC-3) system dataset has all the data like User\_entry\_time, Allotted\_time\_to\_user, User\_id, Building\_name etc.

**2.1.2. Sequence Diagram**

**2.1.2.1 Company Database-**

The manager/admin puts the details of employees/students who are working in the company.

**2.1.2.2 Building Database-**

It is the database which contains the list of employees who are present currently inside the building. It is updated by the system when someone logins.

**2.1.2.3 Login-**

Student/employee/user enter the login credentials in UI and after that system checks if the user is registered and if he is allowed to enter in this time slot then UI will display the message to allow the user and the building database will be updated with a new user entry.

**2.1.2.4 Registration-**

User enters the registration details and further system checks the Company database to check if the user is a member of the company or not and after that it registers the user and a mail is sent to the user’s mail.

**2.1.2.5 Logout-**

Student / employee leave the building with this feature and the building database removes the user from its record.

**2.1.2.6 Update Company Details-**

Manager/admin can update /add a new employee to this database whenever a new employee has joined the company or any employee details has changed

**2.1.3. State Diagram :**

Initial state is being shown by starting with a black dot. Final State is being shown by

the black dot surrounded by an empty circle.

**2.1.3.1 Registration :**

This process is for only new users i.e visiting for the first time which doesn't have an account for login. User will have to register first name, last name, Enrollment no, Email id .After that User will get an email from System but For to be complete registration process user have to ask permission from Manager , if manager give permission then User will registered successful in the system and will get login Currentials.

**2.1.3.2 Login:**

Users will have to login from their own login currentials . System will check and verify if given login currentials match or not from system databases. If matched then the system will give permission to enter in campus/CC3 building, if not then will not give permission to enter.

**2.1.3.3 Person present inside building:**

System will check if the given time limit to the user is expired or not, if the time limit is expired then the system will send mail to the user and inform the security guard.

**2.1.3.4 Logout:**

Users will leave campus after getting an email from the system , if the user stays in the CC3 building then the security guard will go to the user and inform that you will have to leave campus.

**2.1.4 Class Diagram**

**\*EXPLAINING THE CLASSES**

**2.2** Class Name:Registered

Description: Contains a database used to store all registered users

**2.2.1** Contains class:DBHelper

**2.2.1.1** Method1:onUpgrade()

Input:old\_version,new\_version

Output: Upgrades the database

Description: upgrades the database mentioned if old\_version does not matches new\_version

**2.2.1.2** Method2: onCreate()

Input:database db

Output:creates database

Description: creates a new database as specified

DBHelper ends

**2.2.2** Method 1:open()

Input: NONE

Output: Return class of database

Description: opens the database mentioned

**2.2.3** Method 2:close()

Input:NONE

Output:NONE

Description: closes the database mentioned

**2.2.4** Method 3:createEntry()

Input: String roll,String name

Output:NONE

Description:creates a new entry for the database

**2.2.5** Method 4:getdata()

Input: NONE

Output: String

Description: gets the data of all members in the database

**2.2.6** Method 5:deleteEntry()

Input:id

Output:NONE

Description : deletes a entry

**2.2.7** Method 6: updateEntry()

Input: String rowId, String name, String email,String phone,String slot

Output: NONE

Description: updates a entry in database

**2.2.8** Method 7:fetchdata()

Input: id

Output: cursor

Description: fetches the data for a particular id

**2.3** Class Name:Inside\_building

Description: contains a database storing info of all users currently inside building

**2.3.1** Contains class:DBHelper

**2.3.1.1** Method1:onUpgrade()

Input:old\_version,new\_version

Output: Upgrades the database

Description:upgrades the database mentioned

**2.3.1.2** Method2: onCreate()

Input:database db

Output:creates database

Description: creates a new database as specified

DBHelper ends

**2.3.2** Method 1:open()

Input: NONE

Output: Return class of database

Description: opens the database mentioned

**2.3.3** Method 2:close()

Input:NONE

Output:NONE

Description: closes the database mentioned

**2.3.4** Method 3:createEntry()

Input: String roll,String name, String email,String phone,String slot

Output:NONE

Description:creates a new entry for the database

**2.3.5** Method 4:getdata()

Input: NONE

Output: String

Description: gets the data of all members whose time limit exceeded in the database

**2.3.6** Method 5:deleteEntry()

Input:id

Output:NONE

Description : deletes a entry

**2.3.7** Method 6: updateEntry()

Input: String rowId, String name, String email,String phone,String slot

Output: NONE

Description: updates a entry in database

**2.3.8** Method 7:fetchdata()

Input:id

Output:cursor

Description: fetches the data for a particular id

**2.4** Class Name:Company

Description: Contains a database which keeps a record of all users enrolled in the company / college

**2.4.1** Contains class:DBHelper

**2.4.1.1** Method1:onUpgrade()

Input:old\_version,new\_version

Output: Upgrades the database

Description:upgrades the database mentioned

**2.4.1.2** Method2: onCreate()

Input:database db

Output:creates database

Description: creates a new database as specified

DBHelper ends

**2.4.2** Method 1:open()

Input: NONE

Output: Return class of database

Description: opens the database mentioned

**2.4.3** Method 2:close()

Input:NONE

Output:NONE

Description: closes the database mentioned

**2.4.4** Method 3:createEntry()

Input: String roll,String name, String email,String phone,String slot

Output:NONE

Description:creates a new entry for the database

**2.4.5** Method 4:getdata()

Input: NONE

Output: String

Description: gets the data of all members in the database

**2.4.6** Method 5:deleteEntry()

Input:id

Output:NONE

Description : deletes a entry

**2.4.7** Method 6: updateEntry()

Input:id

Output:NONE

Description: updates a entry in database

**2.4.8** Method 7:fetchdata()

Input:id

Output:cursor

Description: fetches the data for a particular id

**2.5** Class name:Frag\_one

Description: This class does all operation for manager/admin/controller

**2.5.1** Method1: onclick btn()

Input: emp id,name,email,phone,timeslots

Output:Updated CompanyDb

Description : updates/enters a new record

**2.6** Class name: Frag\_two

Description: This class is for all operations from user/student side

**2.6.1** Method 1: bt2.setOnClickListener()

Input:String id,String password

Output: Permission and mail

Description: performs registration and sends mail after that

**2.6.2** Method 2: bt3.setOnClickListener()

Input: String id,String password

Output:Permission and stores in BuildingDb

Description: Logins a user and states the permission

**2.6.3** Method 3: bt4.setOnClickListener()

Input: String id,String password

Output:Removes from Building Db

Description: User can log out

**2.7** Class name: ExampleJobService

**2.7.1** Method1: onStartJob()

Input:NONE

Output:NONE

Description : calls doInBackground() method to performs tasks in another thread

**2.7.2** Method2: doInBackground()

Input:BuildingDb

Output:MAIL

Description: Checks the database inside building regularly and mails all those users whose time limit is gonna expire

**2.7.3** Method 3: onStopJob()

Input:NONE

Output:NONE

Description: when a job is closed by an unexpected error. It helps to re-run the background process

**3 Execution Architecture**

Runtime environment required is any device supporting Android Operating

System with the minimum version of Ice-cream Sandwich, Android Studio as

a deployment platform.

**4 Design decisions and tradeoffs**

Two fragments in the same screen in tab layout fashion will be present , One for the manager and another for the user. We have run the checking and mailing process in another thread so that the app can respond accurately.

**5.0 Pseudocode for different components**

**5.1.1 Class: Frag\_one**

LAYOUT:CONTAINS THE USER LOGIN,EXIT,AND REGISTRATION VIEW

METHOD 1: Register()

INVOKED:When a user enter details and clicks on registration

IF(USER NOT REGISTERED && USER IS FOUND IN CompanyDb)

Register the user

MAIL for confirmation

END IF

ELSE

SHOW message “enter valid details”

END ELSE

END METHOD 1

METHOD 2:Login()

INVOKED: When the user enters the details and clicks on login

CHECK1: User has entered correct credentials by referring the RegisteredDb

CHECK2: User has come in allotted time slots by referring to CompanyDb

IF(CHECK1&&CHECK2)

UPDATE the Building database with user info

ELSE

SHOW message “enter valid details”

END METHOD 2

METHOD 3:Logout()

INVOKED: When user clicks log out after entering the credentials

REMOVE user details from building database

END METHOD 3

**5.1.2 Class: Frag\_one**

LAYOUT: Contains the manager View for the purpose of adding / updating employee/student details

METHOD 1:Updatedetail()

INVOKED:When manager adds/updates details of employee/user

CHECK1: Password entered is correct

IF(CHECK1)

UPDATE the CompanyDb with the details entered by the manager

END IF

END METHOD1

**5.1.3 Class : Example\_Job\_Service**

LAYOUT : NO LAYOUT , RUNS IN BACKGROUND

METHOD1: onStartJob()

CALL doInBackgrounf()

END METHOD1

METHOD 2: doBackgroundWork()

String s=COLLECT all the id’s who have their limit crossed present in database BuildingDb

MAIL all the id’s Collected in s

END METHOD 2

METHOD 3:onStopJob()

INVOKED: when unexpectedly backgound work has stopped

START IT AGAIN

END METHOD 3

**5.1.4 Class:BuildingDb**

String *KEY\_ROWID*="\_id";

String *KEY\_NAME*="person\_name";

String *KEY\_MAIL*="person\_mail";

String *KEY\_OUT*="out\_time";

String DATABASE\_NAME="ContactrDB";

String DATABASE\_TABLE="ContactrTable";

int DATABASE\_VERSION=1;

DBHelper ourHelper;

Context ourcontext;

SQLiteDatabase ourDatabase;

METHOD 1:BuildingDb()

ourcontext=context;

END METHOD 1

SubClass 1:DBHelper

FORM A DATABASE WITH THE ATTRIBUTES AND COLUMNS MENTIONED IN BuildingDb Class

END SubClass 1

METHOD 2: open()

OPEN the database named DATABASE\_NAME

METHOD 2 ENDS

METHOD 3: close()

OPEN the database named DATABASE\_NAME

METHOD 3 ENDS

METHOD 4: createEntry((String roll,String name, String mail,String timee)

ContentValues cv = new ContentValues();

cv.put(*KEY\_ROWID*,roll);

cv.put(*KEY\_NAME*, name);

cv.put(*KEY\_MAIL*, mail);

cv.put(*KEY\_OUT*,timee);

return ourDatabase.insert(DATABASE\_TABLE, null, cv);

END METHOD 4

METHOD 5: getdata()

RETURN all those id’s whose time allotted is less than current time

END METHOD 5

METHOD 6:deleteEntry()

DELETE the mentioned id from DATABASE\_NAME

END METHOD 6

METHOD 7:updateEntry()

DELETE the mentioned id from DATABASE\_NAME

END METHOD 7

METHOD 8:fetchData()

RETURN the details of person with given id

END METHOD 8

**5.1.5 Class: CompanyDb**

String *KEY\_ROWID*="\_id";

String *KEY\_NAME*="person\_name";

String *KEY\_MAIL*="person\_mail";

String *KEY\_OUT*="out\_time";

String DATABASE\_NAME="CompanyDB";

String DATABASE\_TABLE="CompanyTable";

int DATABASE\_VERSION=1;

DBHelper ourHelper;

Context ourcontext;

SQLiteDatabase ourDatabase;

METHOD 1:CompanyDb()

ourcontext=context;

END METHOD 1

SubClass 1:DBHelper

FORM A DATABASE WITH THE ATTRIBUTES AND COLUMNS MENTIONED IN BuildingDb Class

END SubClass 1

METHOD 2: open()

OPEN the database named DATABASE\_NAME

METHOD 2 ENDS

METHOD 3: close()

OPEN the database named DATABASE\_NAME

METHOD 3 ENDS

METHOD 4: createEntry((String roll,String name, String mail,String timee)

ContentValues cv = new ContentValues();

cv.put(*KEY\_ROWID*,roll);

cv.put(*KEY\_NAME*, name);

cv.put(*KEY\_MAIL*, mail);

cv.put(*KEY\_OUT*,timee);

return ourDatabase.insert(DATABASE\_TABLE, null, cv);

END METHOD 4

METHOD 5: getdata()

RETURN all members present in DATABASE\_TABLE

END METHOD 5

METHOD 6:deleteEntry()

DELETE the mentioned id from DATABASE\_NAME

END METHOD 6

METHOD 7:updateEntry()

DELETE the mentioned id from DATABASE\_NAME

END METHOD 7

METHOD 8:fetchData()

RETURN the details of person with given id

END METHOD 8

5.1.6 Class:Registered

String *KEY\_ROWID*="\_id";

String *KEY\_NAME*="person\_name";

String DATABASE\_NAME="RegisteredDB";

String DATABASE\_TABLE="RegisteredTable";

int DATABASE\_VERSION=1;

DBHelper ourHelper;

Context ourcontext;

SQLiteDatabase ourDatabase;

METHOD 1:

BuildingDb()

ourcontext=context;

END METHOD 1

SubClass 1:DBHelper

FORM A DATABASE WITH THE ATTRIBUTES AND COLUMNS MENTIONED IN BuildingDb Class

END SubClass 1

METHOD 2: open()

OPEN the database named DATABASE\_NAME

METHOD 2 ENDS

METHOD 3: close()

OPEN the database named DATABASE\_NAME

METHOD 3 ENDS

METHOD 4: createEntry((String roll,String name)

ContentValues cv = new ContentValues();

cv.put(*KEY\_ROWID*,roll);

cv.put(*KEY\_NAME*, name);

return ourDatabase.insert(DATABASE\_TABLE, null, cv);

END METHOD 4

METHOD 5: getdata()

RETURN all id’s in DATABASE\_TABLE

END METHOD 5

METHOD 6:deleteEntry()

DELETE the mentioned id from DATABASE\_NAME

END METHOD 6

METHOD 7:updateEntry()

DELETE the mentioned id from DATABASE\_NAME

END METHOD 7

METHOD 8:fetchData()

RETURN the details of person with given id

END METHOD 8