**inSight 2017**

**Department : Hostel Office**

**Problem Title : Digital hostel resident’s record and complaint management**

**Problem Code : PS01**

**Problem Description :**

Every year there are large numbers of students opting to reside in the hostels. It is a very difficult task to maintain and search manual records for the details of these students. If such software solutions can be developed enabling the hostel department with the features mentioned below, it would reduce severe amount of man power and time.

Features the system should include:

* Provisions to add and modify details of the students (including personal details, academic details, room number, hostel block etc.)
* Provisions of searching of records based on any of the fields as mentioned above.
* Registering of complaint by the students. They should also be able to track their complaint status.

**Proposed solution:**

We propose to build a web application which will allow the hostels to maintain a digital record of the students who are staying in the hostel and allow them to perform various operations like accessing hostel information, digital notice board, lodging complaints etc.

How the system works:

* The Hostel authorities will first register themselves as the administrators for the application this registration will happen via command line interface for security reasons also any number of admins can register using a secrete key.
* The students need to register themselves for the application using the signup page by providing their personal information such as name, mobile number etc. along with all the other necessary details such as their academic details, room number etc.
* The details such as password will be encrypted using a hash function for security reasons and the same hash value will used to verify the user details during authentication.
* The students will have access to various information about the hostel such as capacity, mess details, warden contact information etc. in the main page.
* The personal details will be displayed in their profile page which can be viewed by the students or admin. The students will have the option to edit /modify their personal details at any point of time.
* Since these details will be stored the database the hostel authorities will be able to generate a record of any sort in a table form. The record can also be saved in a pdf format for their later use.
* The admin will be able to search for a particular student (s) using name, usn, room no and hostel block. Also the admin can assign fine/due to a particular student along with its status which will appear on the students profile page. A message will also be sent to the student as and when the due status changes using Twilio API.
* The admin will also be able to generate a report of all students who have an unpaid fine/due in a table format which can be saved as pdf for later use.
* The admin can also generate a notice using the application which will be displayed in the notice board page. This notice can be of any form such as plaintext, images, video (link via YouTube etc.). These notices will be paginated and will be displayed from new to old order.
* The students can also lodge their complaint via complaint page which will be sent for approval to the admin once the admin approves it will be displayed in the complaint page along with the status and a short description.
* A message will be sent to students registered phone number once his complaint has been approved by the admin

The web application will be built in python using the Flask framework along with various other libraries such as SQLAlchemy (an ORM), Flask-Weasyprint (to convert html to pdf), Flask-Bcrypt (A cryptographic hash function for encrypting/decrypting passwords), Tinymce (A WYSIWYG editor plugins for creating notices) etc. We will use SQLite3 database for storing user data and also TWILIO ( A cloud telephony service) for sending messages to the user.

**Technology Stack:**

**OS**: Linux

**Server:** Apache

**Database**: SQLITE3

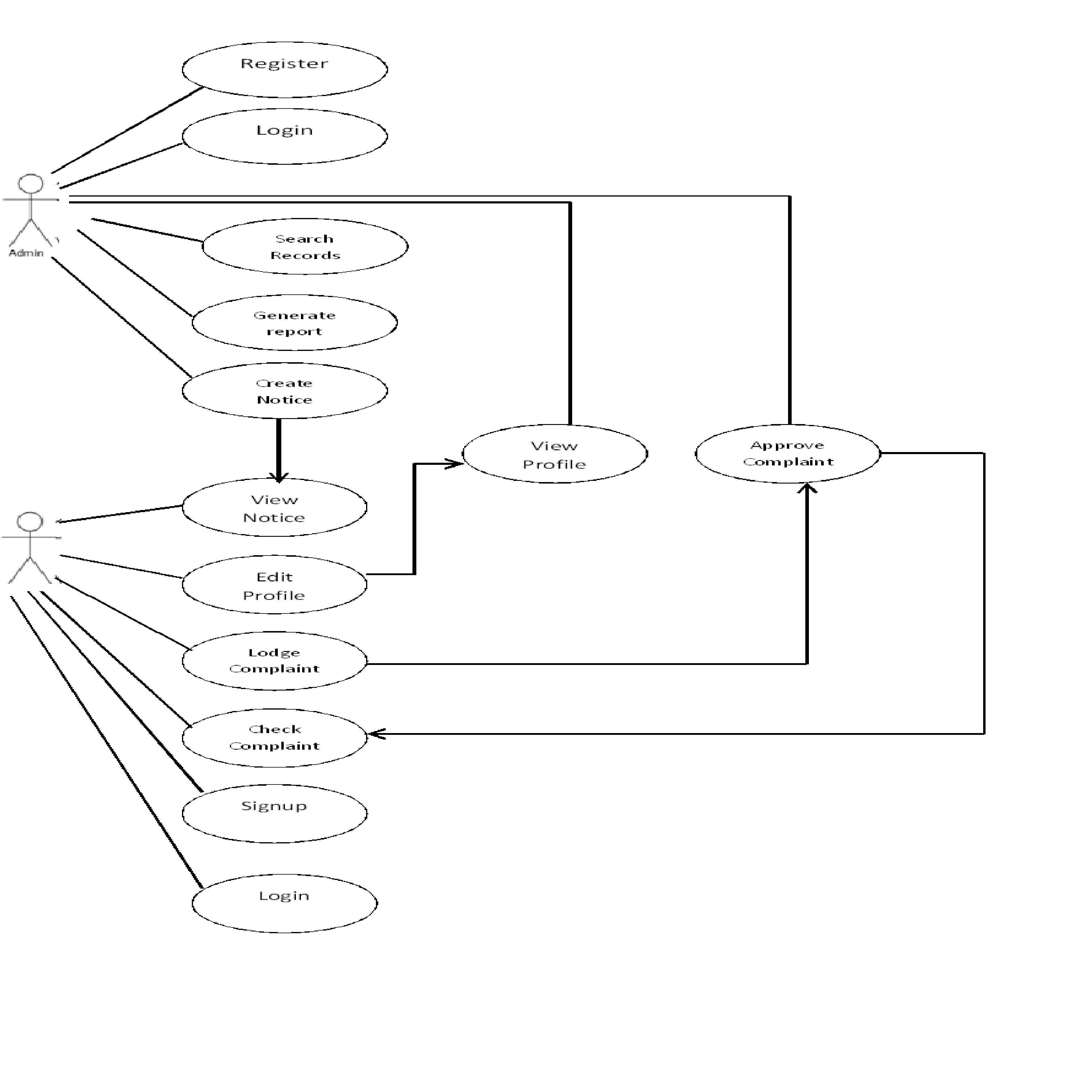
**Scripting Language**: Python

**Web-Framework**: Python FLASK

**Frontend-framework**: Twitter Bootstrap

**Object Relational Mapper (ORM)**: SQLAlchemy**Cloud Telephony Service**: Twilio API (For demonstration purpose only as the service has limited functionality in the free version)

**Use Case:**



|  |  |
| --- | --- |
| **UC 1** | **Login/Signup** |
| **Description** | A Visitor can log in with certain credentials to become an authorised user or Signup to become a authorised user |
| **Actors** | Visitor, User, Server |
| **Normal Sequence** | 1. A Visitor visits a website, and goes to the “login” page  2. The Visitor enters in a username and password, and submits the form to the Server  3. The Server checks their username and password against the list of existing user accounts, and veriﬁes that the account exists  4. If account does not exist then the user is redirected to signup page to complete signup process and again redirected to login page afterwards  5. The Server updates the Visitor’s session to indicate that they are an authorised User  6. The Visitor becomes a User, and is redirected to a Profile page |

|  |  |
| --- | --- |
| **UC 2** | **Edit Profile** |
| **Description** | A Visitor can edit his personal details ,academic details etc. |
| **Actors** | User, Server |
| **Normal Sequence** | 1. The User requests the server to provide the Edit profile form  2. The Visitor is ﬁlling out an entry ﬁeld in a form (e.g. an e-mail address in a registration form)  3. Once the ﬁeld is ﬁlled out, the Client submits the value to the Server for validation  4. If the Server ﬁnds the input is invalid (e.g. the address is malformed), it returns the result to the Client  5. The Client graphically highlights the ﬁeld, and displays a message next to it informing the Visitor the input is invalid, why it is invalid, (“this e-mail address already exists in our database;”)  6. The Visitor corrects the error (e.g. a valid email address)  7. The input is submitted again to the Server, and it is now correct; the ﬁeld highlight and error message is removed  8.The updated information is saved in the database and the user is redirected to the profile page where the updated information is displayed |

|  |  |
| --- | --- |
| **UC 3** | **View Profile** |
| **Description** | A User can view his personal information along with academic details etc. |
| **Actors** | User, Server |
| **Normal Sequence** | 1. A Visitor requests for the profile page  2. The Server constructs the page by querying the details of current user  3. The Visitor receives the HTML-only page  4. The received HTML page is displayed by the browser |

|  |  |
| --- | --- |
| **UC 4** | **Create Notice** |
| **Description** | An admin can create a notice to be published in the digital notice board |
| **Actors** | Admin, Server |
| **Normal Sequence** | 1. The Admin requests the server for the Create Notice page  2. The Admin is presented with a WYSIWYG editor  3. The Admin uses the WYSIWYG editor to create the notice containing of Plaintext, Images, videos etc.  4. The WYSIWYG then uses the input to create a HTML code for the notice  5. The HTML code is then checked for any malicious script, tag in any such script, tag in discovered then it will be bleached  6. The Code is then saved in the database as a String of characters  7. The Admin is then redirected to the Notices page where the Notice is displayed |

|  |  |
| --- | --- |
| **UC 5** | **View Notices** |
| **Description** | A Visitor can view all the notices published in the digital notice board |
| **Actors** | Admin, User, Server |
| **Normal Sequence** | 1. A Visitor requests for the Notices page  2. The Server constructs the page by querying list of all Notices present in the database  3. The Server then paginates the list of notices and returns the html only page.  4. The Visitor receives the HTML-only page and is displayed by the browser  5 . The Visitor selects a particular notice from the page and he is then redirected to that page where the full notice is displayed. |

|  |  |
| --- | --- |
| **UC 6** | **Lodge Complaint** |
| **Description** | A User can file a compliant indicating the problems faced by him/her. |
| **Actors** | User, Server |
| **Normal Sequence** | 1. The User requests the server to provide the Complaint form  2. The Visitor is ﬁlling out an entry ﬁeld in a form (e.g. an e-mail address in form)  3. Once the ﬁeld is ﬁlled out, the Client submits the value to the Server for validation  4. If the Server ﬁnds the input is invalid (e.g. the address is malformed), it returns the result to the Client  5. The Client graphically highlights the ﬁeld, and displays a message next to it informing the Visitor the input is invalid, why it is invalid, (“this e-mail address already exists in our database;”)  6. The Visitor corrects the error (e.g. a valid email address)  7. The input is submitted again to the Server, and it is now correct; the ﬁeld highlight and error message is removed  8.The Complaint is then stored in a temporary table where it is submitted for approval by admin  9. The user is then redirected to his profile page. |

|  |  |
| --- | --- |
| **UC 7** | **Approve Complaint** |
| **Description** | The Admin can approve or reject a complaint filed by the user. |
| **Actors** | Admin, Server |
| **Normal Sequence** | 1. The Admin requests for the Approve Complaints page  2. The Server constructs the page by querying list of all Complaints present in the database  3. The Server then paginates the list of Complaints and returns the html only page.  4. The Admin receives the HTML-only page and is displayed by the browser  5 . The Admin selects a particular Complaint from the page and he is then redirected to that page where the full complaint is displayed.  6. The Admin then either Approves the complaint by clicking approve button or rejects it by clicking the reject button.  7. If the complaint is approved then the complaint is moved from temporary table to a permanent table in database and the student is sent the message of approval .  8. If the complaint is rejected and the entry is removed from the table and the student is sent a rejection message |

|  |  |
| --- | --- |
| **UC 8** | **Check Complaint Status** |
| **Description** | A Visitor can view the status of the complaints filed |
| **Actors** | Admin, User, Server |
| **Normal Sequence** | 1. A Visitor requests for the Complaints page  2. The Server constructs the page by querying list of all Approved Complaints by the user present in the database(in case of admin all approved complaints are displayed)  3. The Server then paginates the list of complaints and returns the html only page.  4. The Visitor receives the HTML-only page and is displayed by the browser  5. The Visitor selects a particular Complaint from the page and he is then redirected to that page where the full description along with approval status is displayed. |

|  |  |
| --- | --- |
| **UC 5** | **Search record/Generate report** |
| **Description** | The Admin can Generate report or search for a particular entry |
| **Actors** | Admin, Server |
| **Normal Sequence** | 1. The admin enters the search criteria in the search box  2. The Server constructs the page by querying list of all matching entries present in the database  3. The Server then paginates the list of entries and returns the html only page.  4. The Admin receives the HTML-only page and is displayed by the browser  5. If the Admin wishes to generate report then he can do so by clicking generate report button.  6. The HTML content of the page is then converted to the pdf format using the WeasyPrint and is then returned to the Admin |

**Team Members details:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Role** | **Name** | **USN** | **Email id** | **Phone no.** |
| Team Leader | VIKRAM S RAO | 1SI14CS131 | vikramsrao96@gmail.com | 9591254802 |
| Team Member 1 | SHARANBASAPPA | 1SI14CS420 | poojarisharan28@gmail.com | 8050291788 |