**ChuParty**

**Final Project Definition**

**By:**

Bar Kazzaz 204455737 [barkazzaz@gmail.com](mailto:barkazzaz@gmail.com)

Inbar Hachmon 313289068 [inbarcha@gmail.com](mailto:inbarcha@gmail.com)

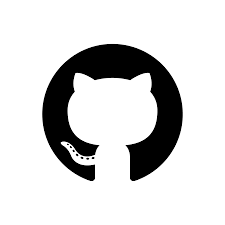
Nitzan Lahav 200953305 [nitzan.la4@gmail.com](mailto:nitzan.la4@gmail.com)

Dvir Ben Shlush 313533192 [dvirbsh95@gmail.com](mailto:dvirbsh95@gmail.com)

Shahar Levi 039541131 [shaharLe88@gmail.com](mailto:shaharLe88@gmail.com)

**Supervisor:**

Dr. Eliav Menashe

[](https://github.com/AmitSaar1994/FinalProjectCS2019.git)https://github.com/InbarCha/ChuParty.git

**Table of Contents**

[**1.**](#_gjdgxs) [**PROJECT DESCRIPTION**](#_Project_description) **3**

[**2.**](#_30j0zll) **RELATED WORK 3**

[**3.**](#_1fob9te) **FUNCTIONAL DESCRIPTION / REQUIREMENTS 4**

[3.1.](#_3znysh7) Permissions and Users 4

[3.2.](#_2grqrue) Login Page 6

[3.2.1.](#_tyjcwt) *Login Page Layout* 6

[3.3.](#_1t3h5sf) Home Page 6

[3.3.1.](#_4d34og8) *Home Page LAYOUT* 6

[3.4.](#_2s8eyo1) Subject 6

[3.5.](#_3fwokq0) Exam Page 7

3.5.1 *Exam Page layout* 7

[3.6.](#_3rdcrjn) Exams dashboard 7

[3.7.](#_44sinio) Feedback Dashboard 7

3.7.1  *Feedback dashboard layout* 7

[3.8.](#_1v1yuxt) admin dashboard 8

3.8.1 *user management*  8

3.8.2 *exam management* 8

[**4.**](#_3whwml4) **ARCHITECTURE** 9

[4.1.](#_2bn6wsx) Each Module description 10-11

[**5.**](#_qsh70q) **WORK PLAN** 12

[**6.**](#_3as4poj) **CLIENT SIDE** 13

[6.1.](#_1pxezwc) Usage Illustration 13

[6.2.](#_49x2ik5) Mockup 14

[*6.2.1.*](#_2p2csry) *Login page* 14

[6.2.2.](#_147n2zr) *Home Page* 15

[6.2.3.](#_3o7alnk) *subjects* 16-17

[6.2.4.](#_23ckvvd) *exam dashboard* 18-19

[6.2.5.](#_ihv636) *feedback dashboard* 20

[*6.2.6.*](#_32hioqz) *admin dashboard* 21

[**7.**](#_1hmsyys) **SERVER SIDE 23**

[7.1.](#_41mghml) API 23

# Project description

**The Goal: Create an interface with better training for students and an easier way to maintain exams (create/read/update/delete).**

*ChuParty* is a web application created to improve managing, creating and editing exams.

We will allow you to create a new exam with an easy to follow template which has a great user interface to enable fast and easy editing.

Our system will have full backwards compatibility with the current exam files.

This means that you will be able to upload to our system an exam that has already been written and saved as PDF, and have it fully editable via our console.

The application will be able to assist creating an exam from a pool of questions, while maintaining a fair and unified difficulty level based on user statistics.

From the student’s end – you will be able to solve exams online, and get the relevant feedback according to the answer provided.

# Related Work

Our project is all about improving the exam experience.

Today in most of the Chupars (the currently available training exams) there are many weaknesses.

Some examples are:

* Files are usually in PDF format, causing various difficulties while navigating between exams/questions within the exam/question to the relevant answer/questions with the same subject.

Basically you are required to manually go through every page and look for the relevant part in the file.

* Answers are hand-written and have poor readability.
* Exams are not editable as are the exams’ answers. Meaning an exam with a mistake is never corrected, and most of the time the mistake is not mentioned anywhere in the document.
* There is no way to navigate exam questions based on a specific subject.
* In some cases the answer is written in the same section as the question, hurting the student’s ability to try and work to an answer without being “spoiled”.
* Answers are sometimes out of order.

While there are other online exam systems, our system will be the most suitable to the college’s current system and will provide all of the above mentioned benefits, while staying intuitive and easy to use.

Allowing a smooth transition to the better exams system *ChuParty* is our priority.

# Functional Description / Requirements

Our project will include a web application that will provide users with the following:

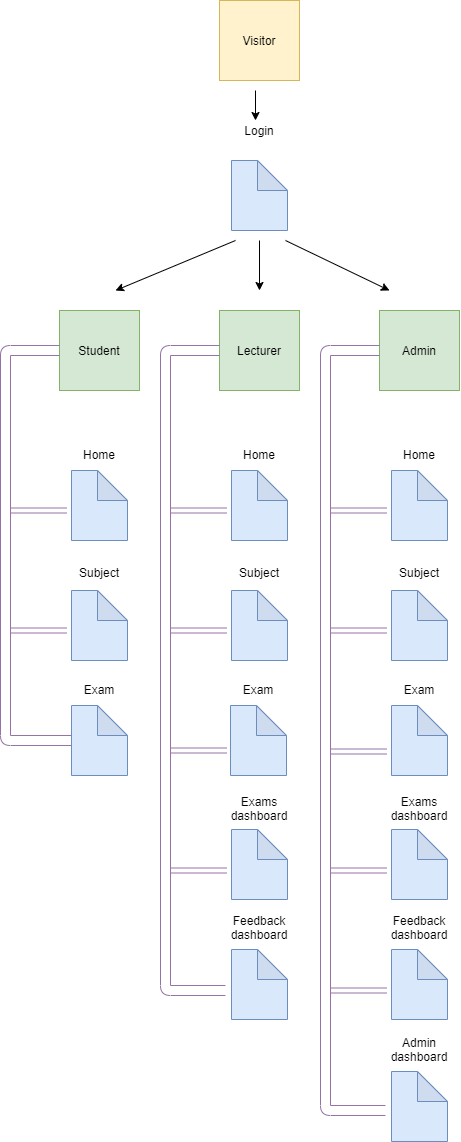
* Signup/Login – with relevant permission groups for students, lecturers (exam writers) and admins.
* For lecturers:
  + Uploading a file – system will use OCR to read and parse past exams.
  + Reading exams.
  + Editing uploaded exams.
  + Deleting irrelevant exams.
  + Reading students feedback regarding the exam.
  + Testing the exams.
* For students:
  + Taking exams.
  + Providing feedback for any mistakes within the exam.
* For admins:
  + All lecturer and student permissions.
  + Managing user groups.
  + Reading feedback provided regarding the system.
* Data Base (create/read/update/delete):
  + Users.
  + Documents (exams).

## Permissions and Users

Chuparty will not be available prior to logging in.

Users will be granted with different permission levels.

1. **Visitor (not logged in user):** Will be able to navigate only to the login page.
2. **Student:** Logged in students will have access to the home page, and each topic he is signed in for (for example OOP in Cpp). Also will be able to select an exam to take online or download.
3. **Lecturer:** Logged in lecturers will have access to the home page, the exams dashboard and the feedback dashboard.
4. **Admin:** Has access to the entire application and the admin dashboard.



## Login page

The Login page is the only screen that you may navigate to while not being logged in.

This user state is referred to as "Visitor".

The Login page includes a single view for a visitor where the visitor will be able to either login or submit a signup request (to be handled by an admin user in case the user was not signed up already).

### Login page layout

1) Buttons of "login" and "register".

2) Input fields for username and password.

## Home Page

The Home page is the first screen all users see after login.

This page contains a navigation menu for each of the next sub views the user may reach.

### Home page layout

1. Welcome message.
2. Navigation menu.
3. Logout button.
4. Search bar.
5. Top 10 (or less) available subject according to statistics and relevance (for example per semester).

## Subject

This screen presents all exams per subject in a sorted manner.

Each exam is a link to the relevant exam page.

## Exam page

This is the page where you take the exam.

Each user will be able to select his answer and get instant feedback.

### Exam page layout

1. Navigation nemu.
2. Logout button.
3. Search bar.
4. Interactive questions and answers.
5. Feedback button -> input field.

## Exams dashboard

The screen used for creating/uploading/deleting exams.

Users will be able to create a new exam from template or import a file of already written exam.

Each new exam has the following require properties:

1. Subject ID.
2. Lecturer ID.
3. Release/Update Date.

## Feedback dashboard

The application is designed in order to make the training experience more suitable for the students and the lecturers creating the training exams.

Meaning, having the best user experience available is a major part of the application design, and a way of managing users' feedback is a requirement.

### Feedback dashboard layout

1. Navigation nemu.
2. Feedback table containing all feedback related to the current user.
3. Each feedback when clicked expands and presents the body of the feedback with a reply button.

## Admin dashboard

In the admin dashboard the user may adjust settings such as:

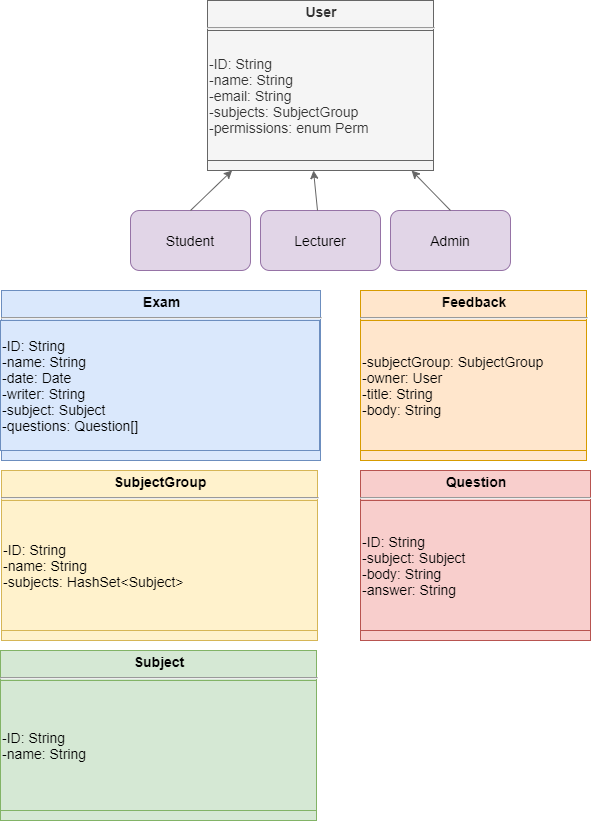
### User management

* Adjust roles and permissions.
* Reset password.

### Exams management

* Assign an exam to a different subject.

# Architecture



* 1. **Each Module description**
     1. **User**

The type of the user currently logged in.

This may vary between one of the three types:

Student, Lecturer and Admin.

Each user will have an ID, name, Email and a permission level.

The permission level will be used to make sure users are not able to operate functionality of higher level users (for example a student should not be able to edit an exam).

* + 1. **Exam**

Object of the exam.

Each exam will have an ID, name, date, writer, subject group and the exams' questions.

ID will be used to identify the exam within the system.

Name will be used for readability (may serve as a title of the exam).

Date will be the date the exam was written.

Writer will be a lecturer name.

Subject group will be the group of subjects that the exam relates to.

Questions will be the array of questions the exam presents.

* + 1. **Subject**

Subject is the object that will be used to represent which topic the exam covers.

Examples of subject could be "OOP with Cpp", "Infinite Mathematics" etc.

Each subject will have an ID and a name for readability purposes.

* + 1. **SubjectGroup**

SubjectGroup is simply a group of subjects.

Each SubjectGroup will have an ID, name and a set of subjects.

This will be used to group subjects together and find subjects easily.

An example of a subject group could be "Computer Science".

* + 1. **Feedback**

Each user may submit a feedback via the web application.

Feedback could be submitted to lecturers in order to ask for a fix in an exam such as typos or asking for an updated answer, or could be submitted to an admin in order to fix functionality, or ask to reset a password.

Each feedback will have a SubjectGroup, an owner which will handle the feedback, a title and the body of the feedback as submitted by the user.

* + 1. **Question**

The object used to fill the exams.

Each question will have an ID, subject, body of the question and the relevant answer.

# Work plan

**Technologies:**

Client : React

Server : Python (Django)

DB : MongoDB

**Infrastructure:**

1. Setup Server / **Owner:** Inbar & Nitzan
2. Onboard the application to the server / **Owner:** Inbar & Nitzan

**Server:**

1. Exam building (using a file and, if required, some OCR) / **Owner:** Inbar & Nitzan
2. Machine learning (for exam generating) / **Owner:** Shahar & Bar
3. Statistics (for lecturers to review and ML to use) / **Owner:** Bar

**DB:**

1. Setup DB (MongoDB) / **Owner:** Dvir

**Client:**

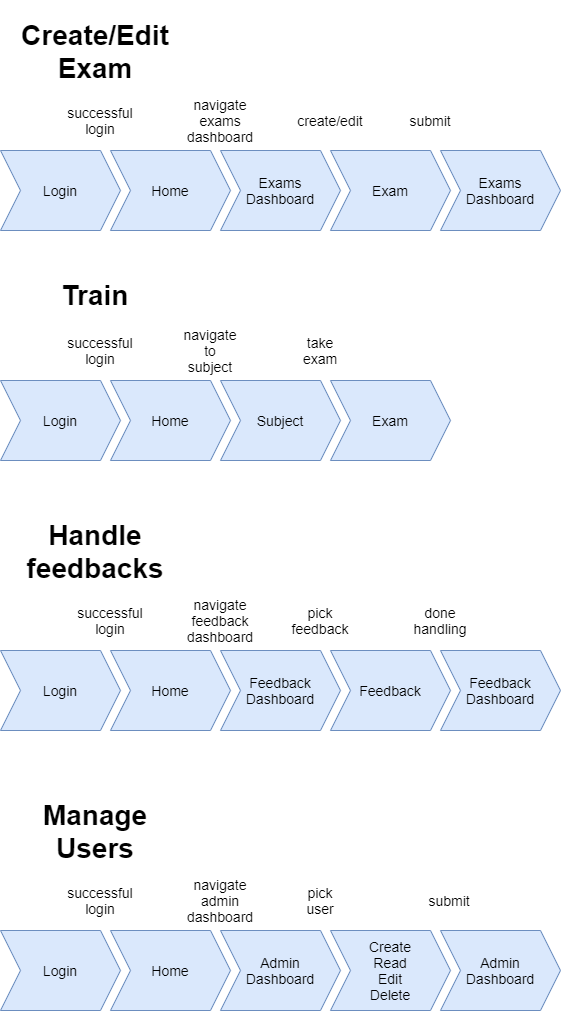
1. Login/Register view / **Owner:** Dvir
2. Home view / **Owner:** Dvir
3. Subject view / **Owner:** Inbar & Nitzan
4. Exam view / **Owner:** Bar
5. Exams dashboard view / **Owner:** Bar
6. Feedback dashboard view / **Owner:** Shahar
7. Admin dashboard view / **Owner:** Shahar

**Integration and testing:**

1. Testing / **Owner:** Everyone
2. Bugs handling / **Owner:** Everyone

# Client side

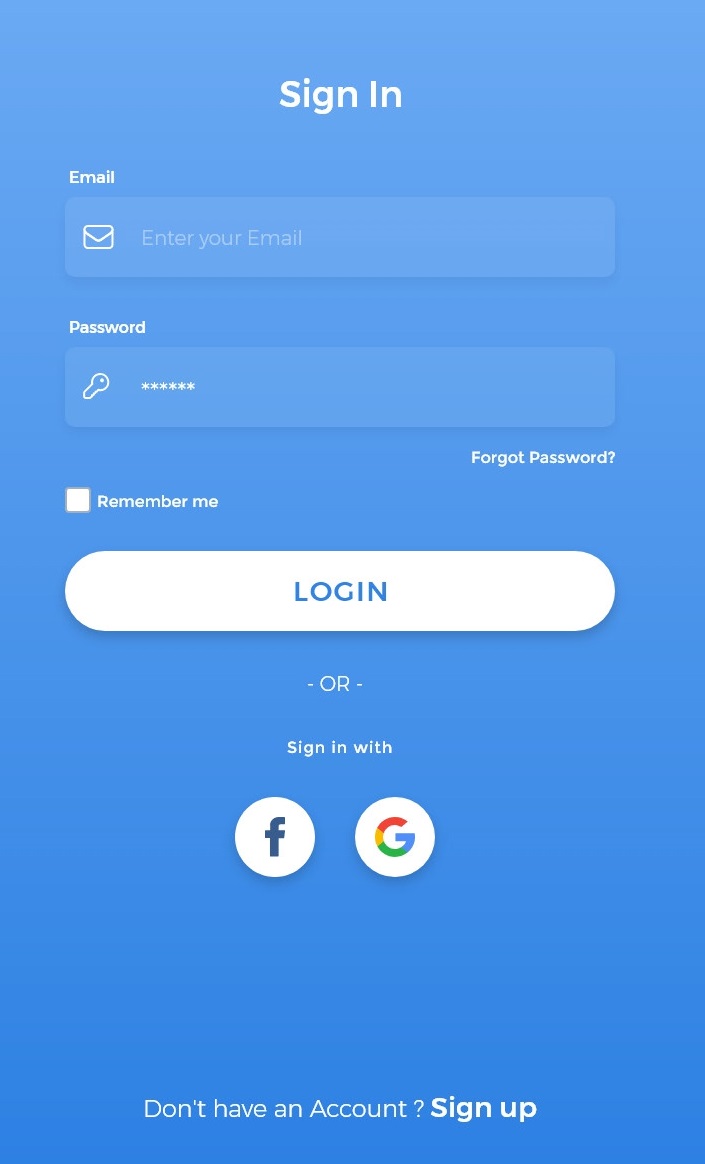
## Usage Illustration



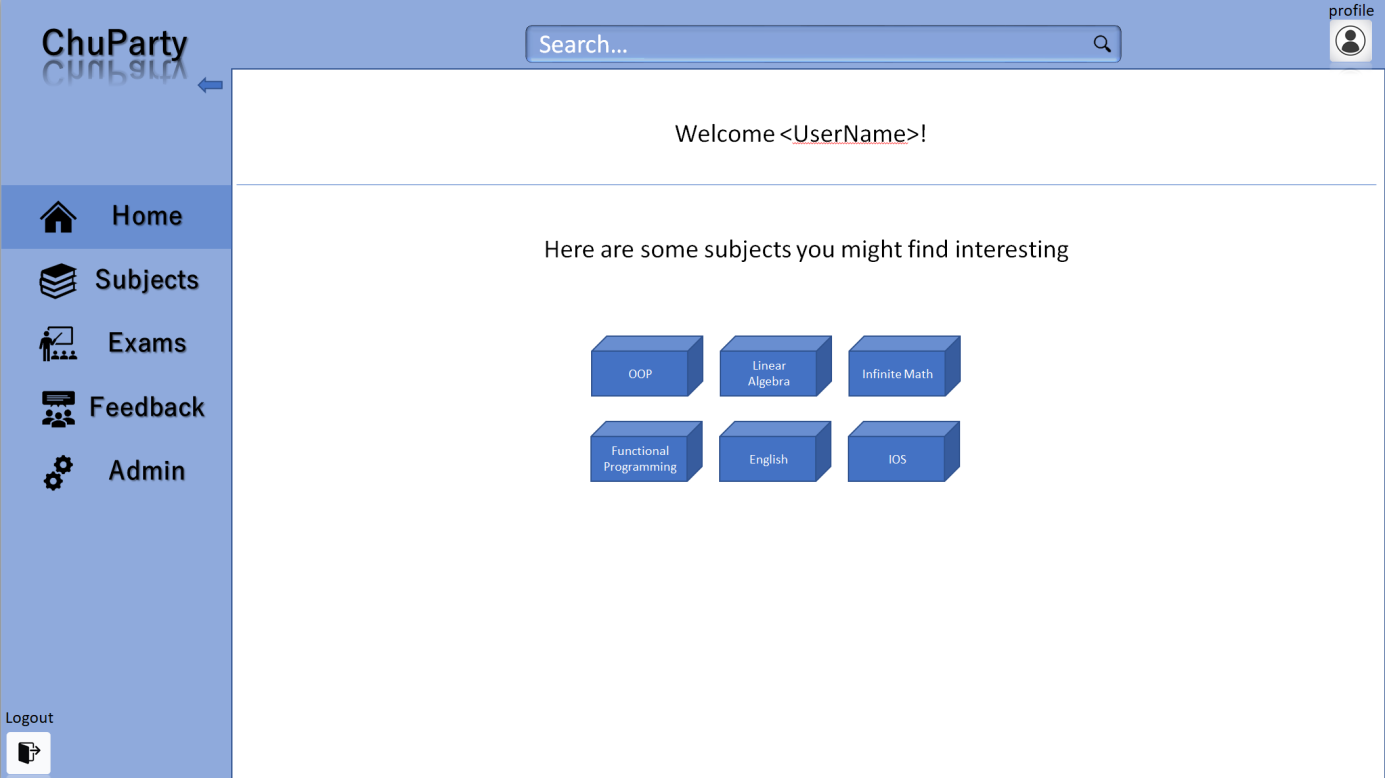
## Mockup

Graphic illustration of the different screens in your app.

### Login

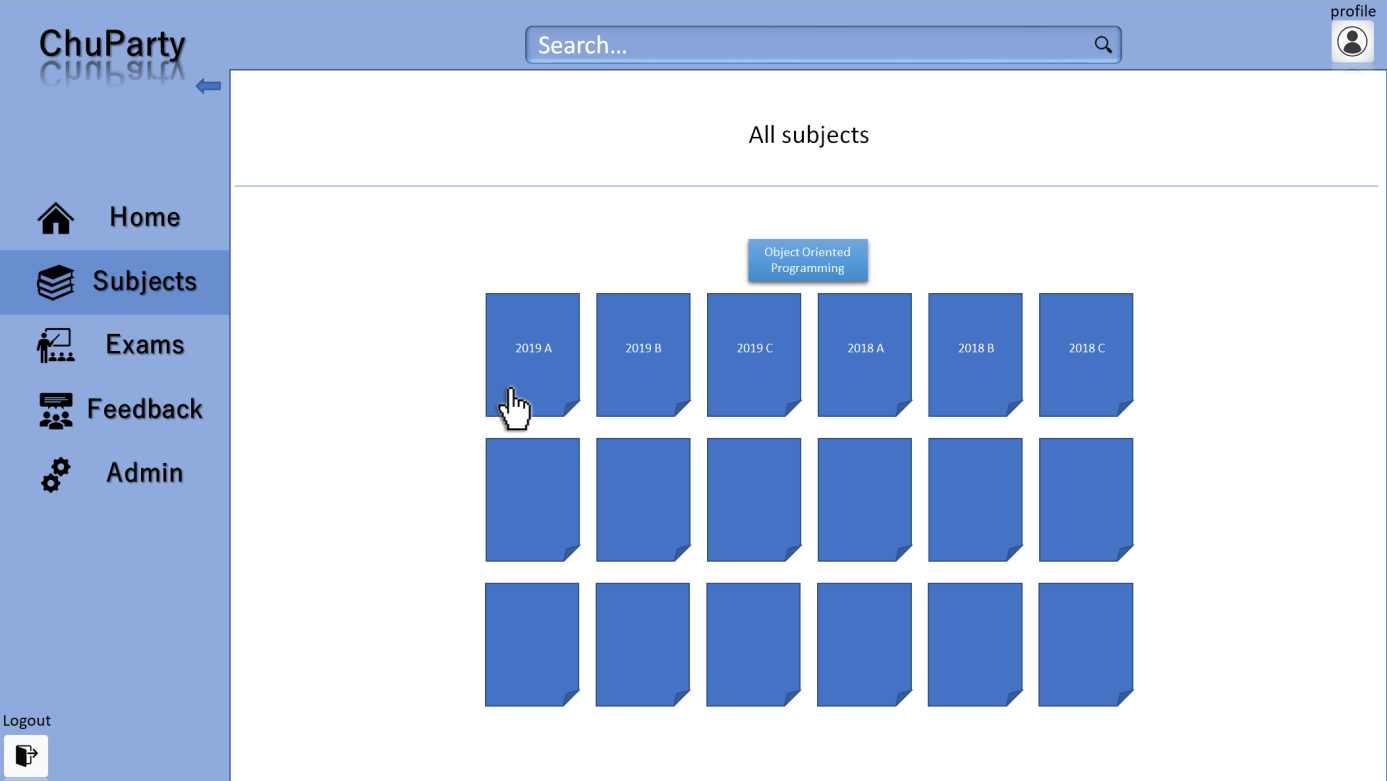
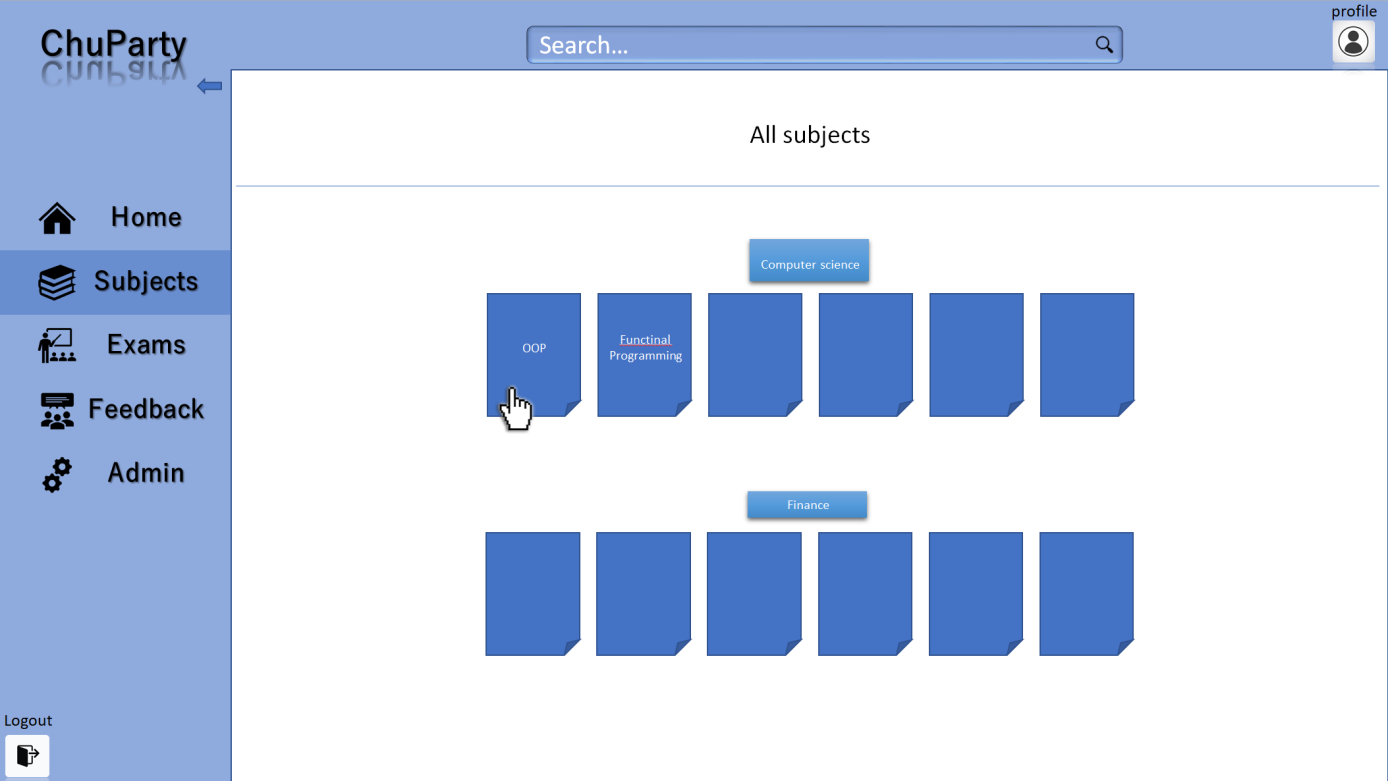
**

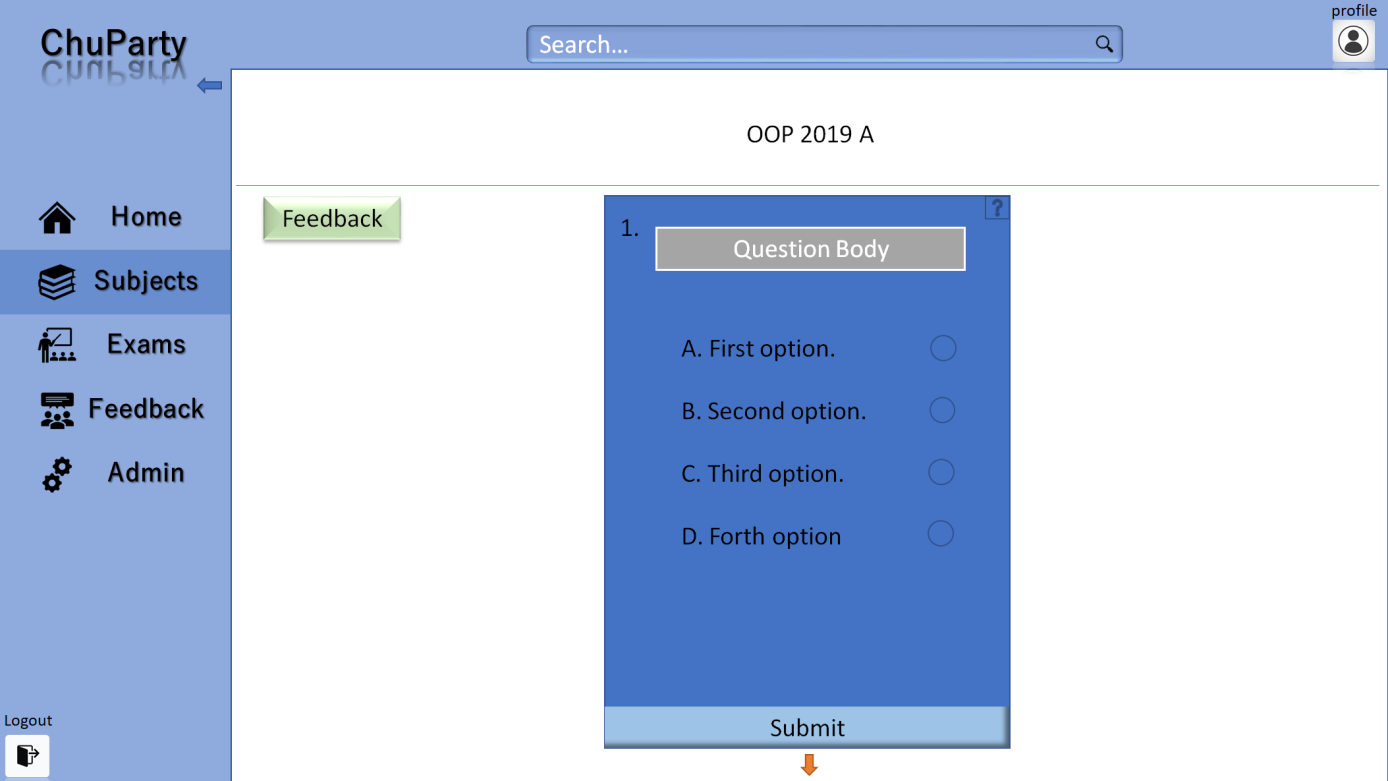
### Home Page



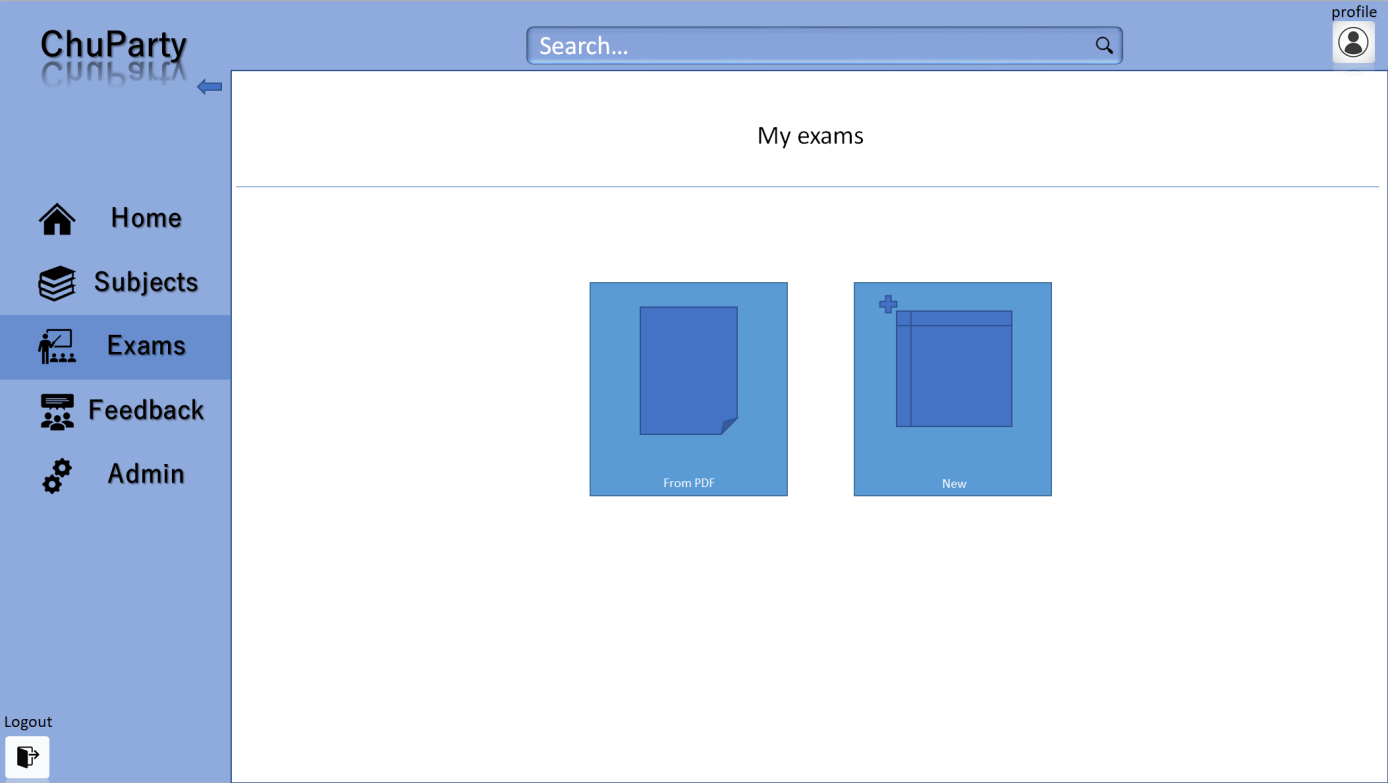
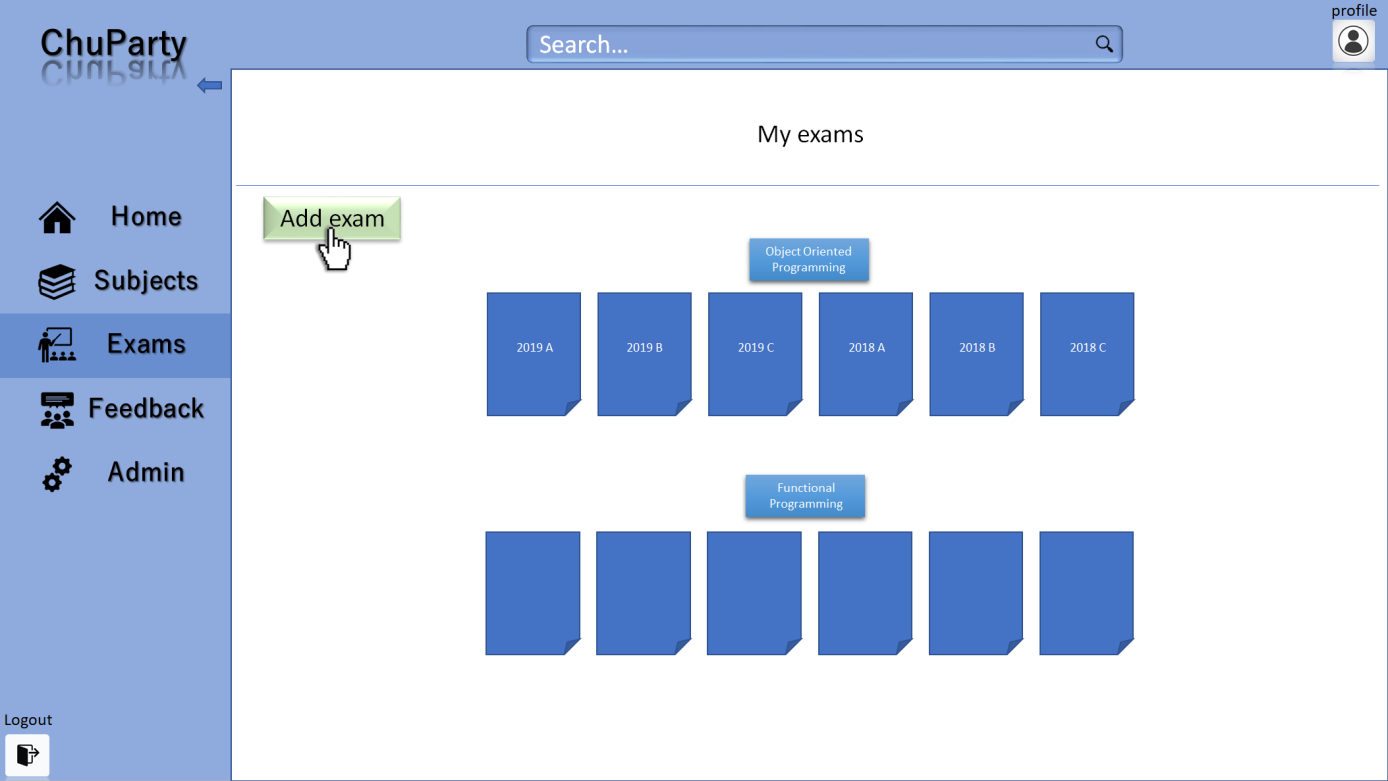
### User Hotel History

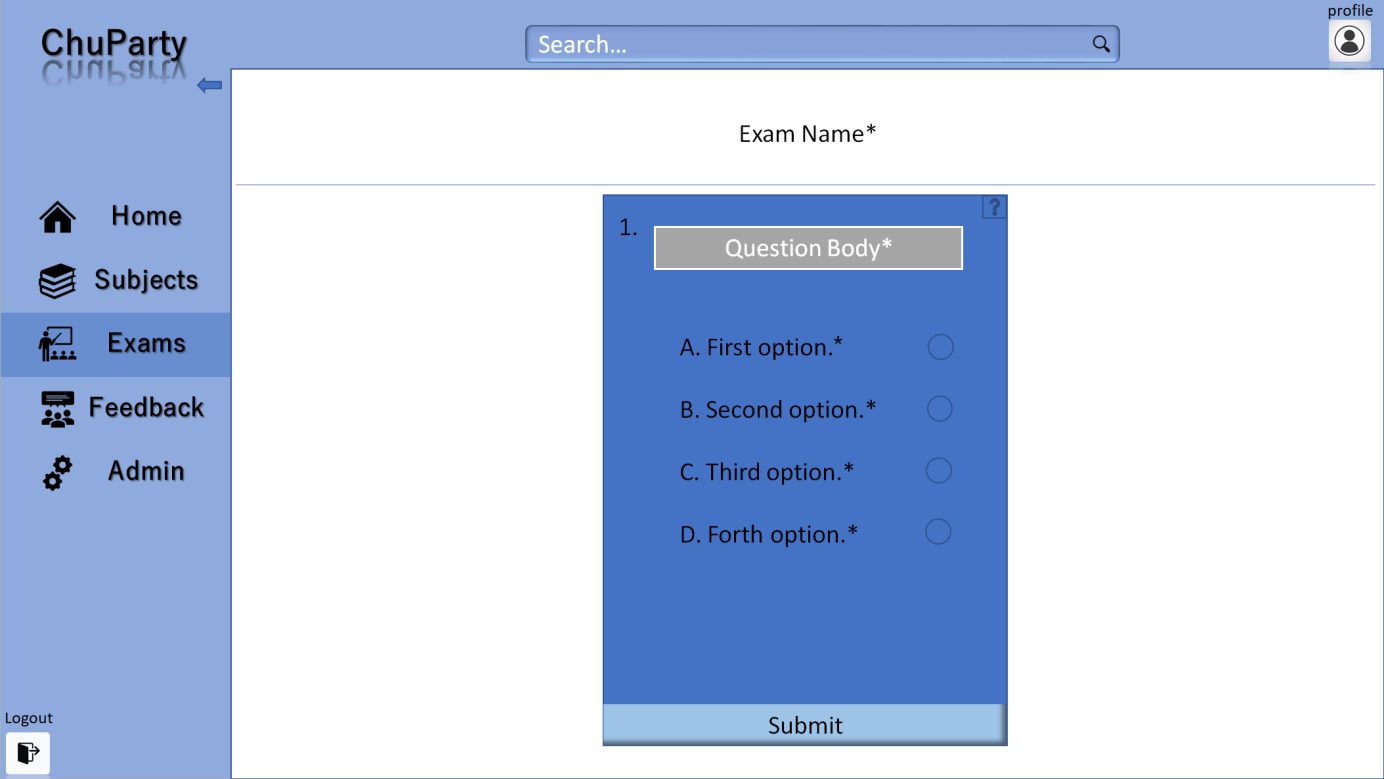
### Subjects



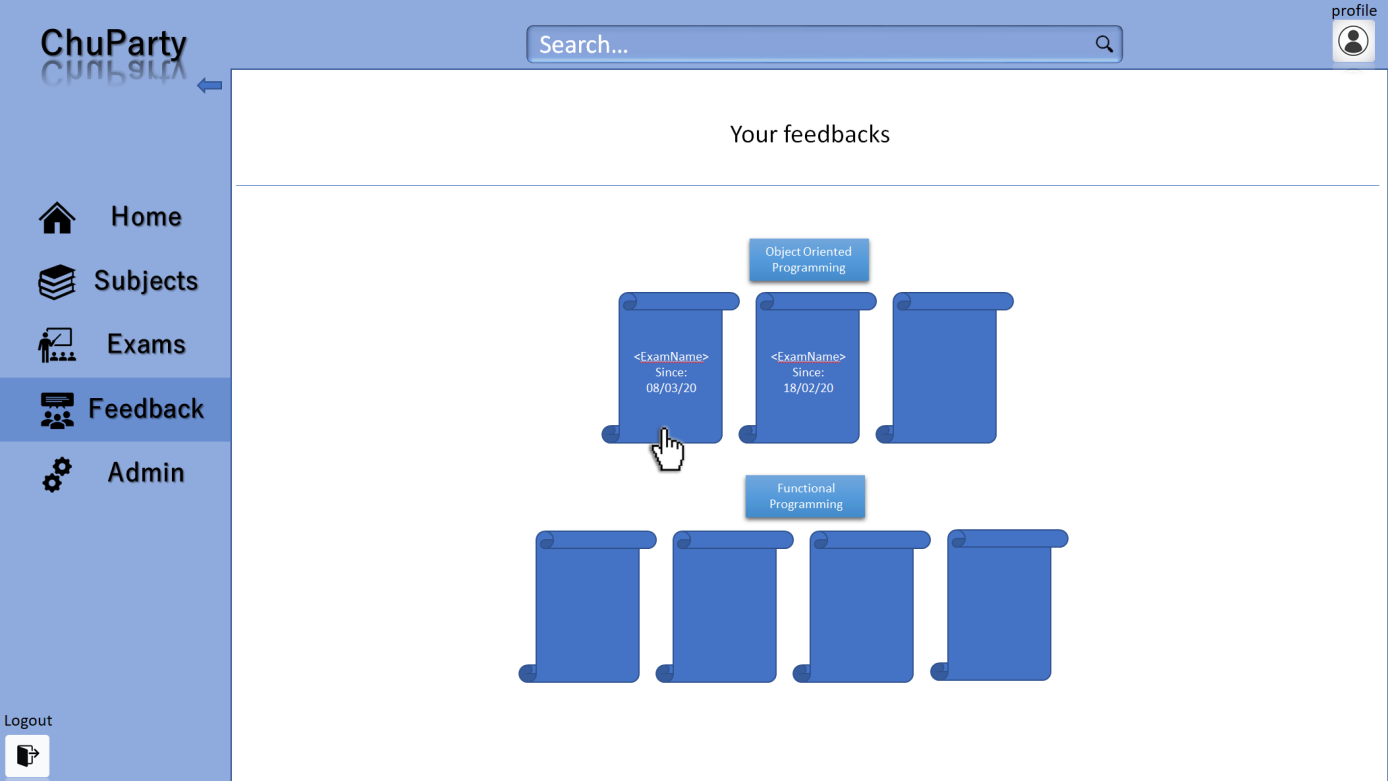
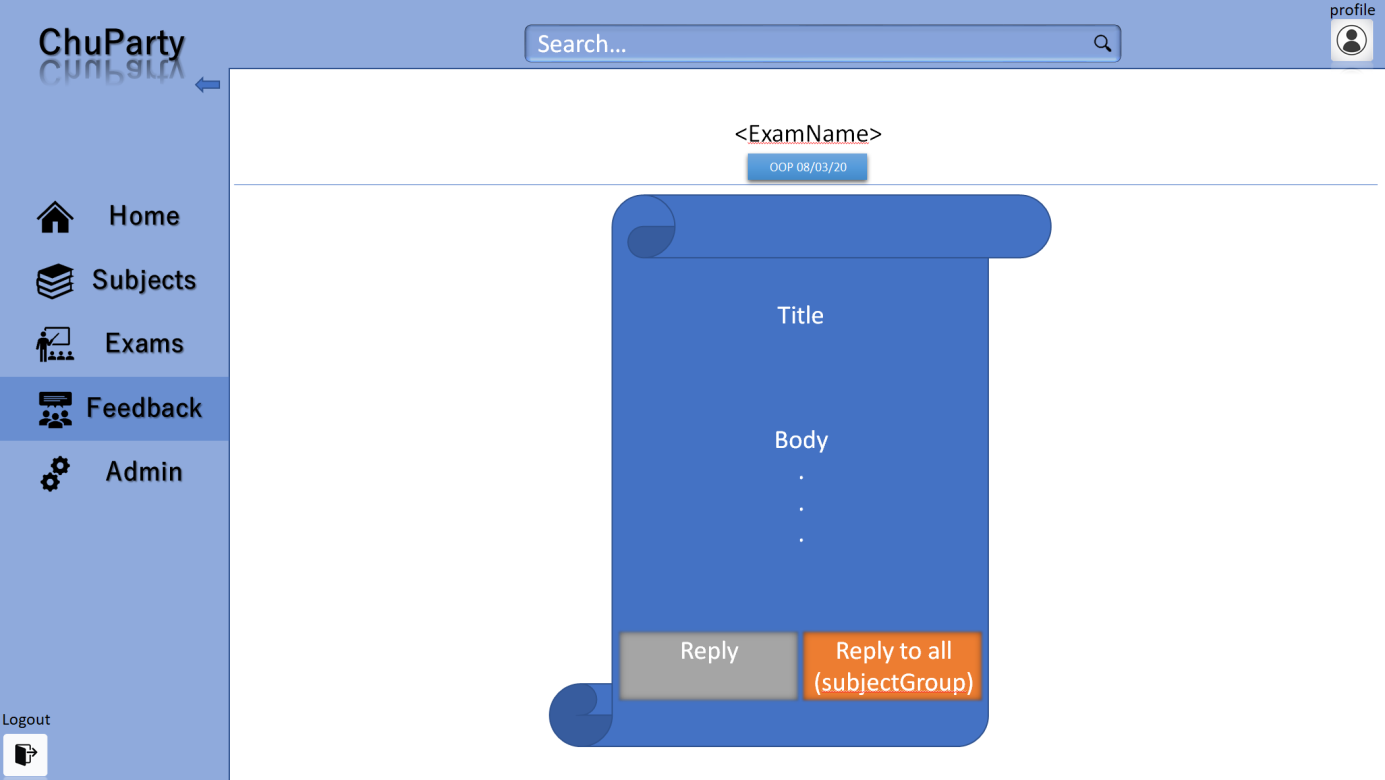
****

### Exams dashboard

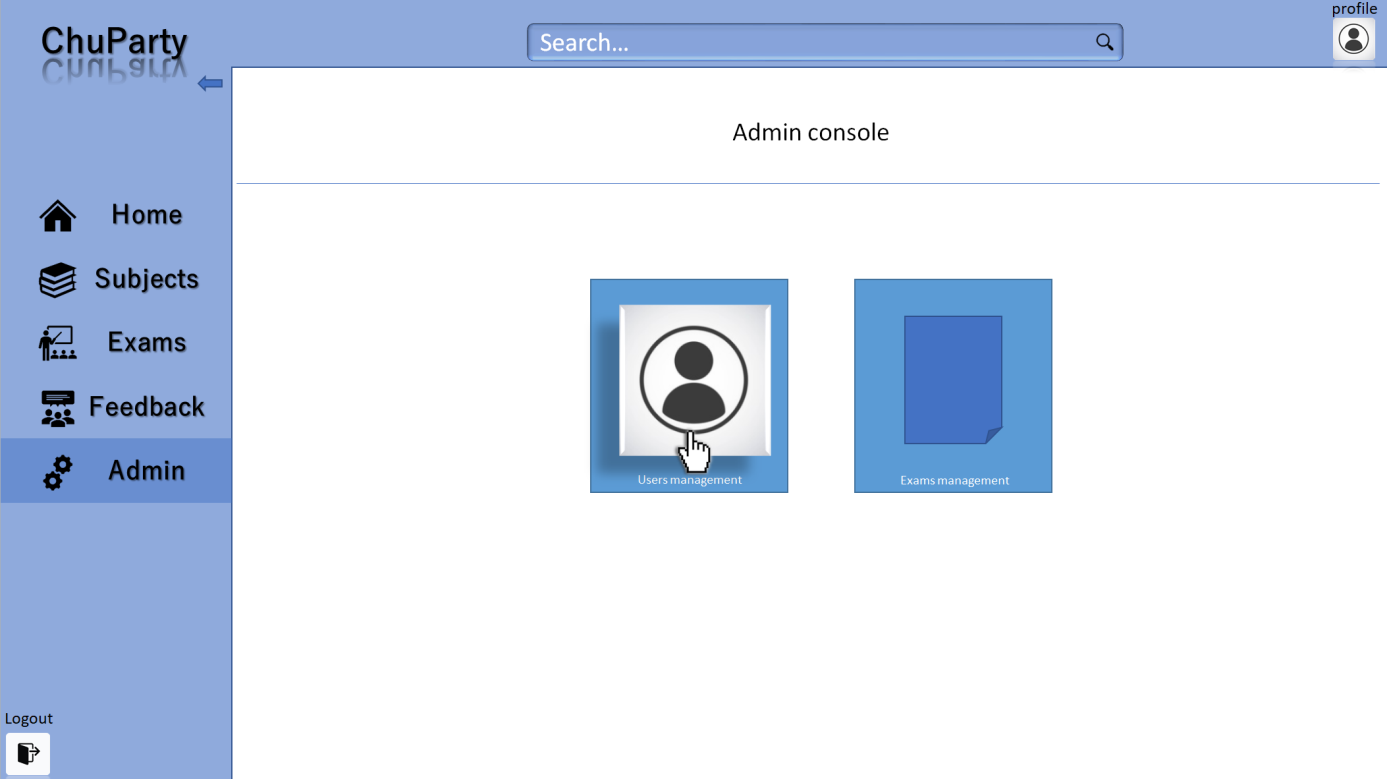


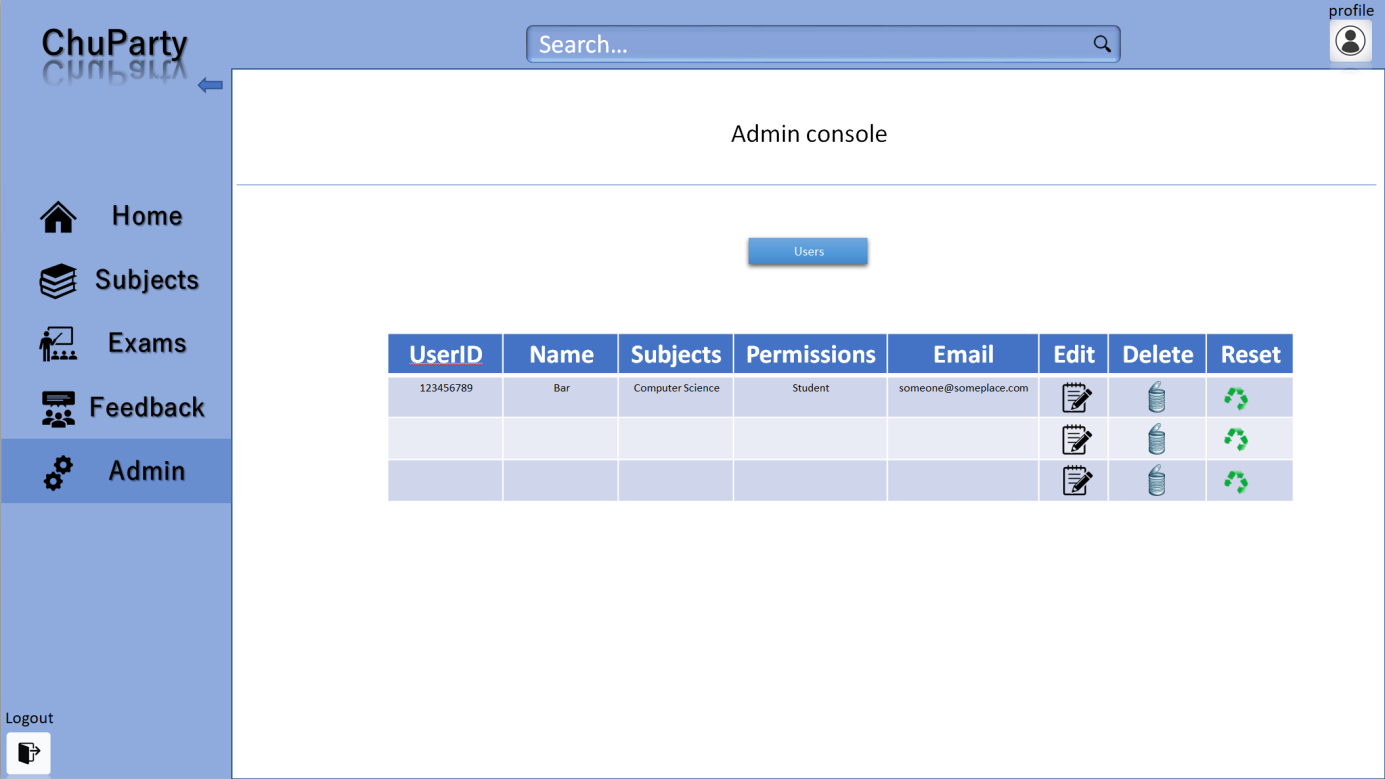
****

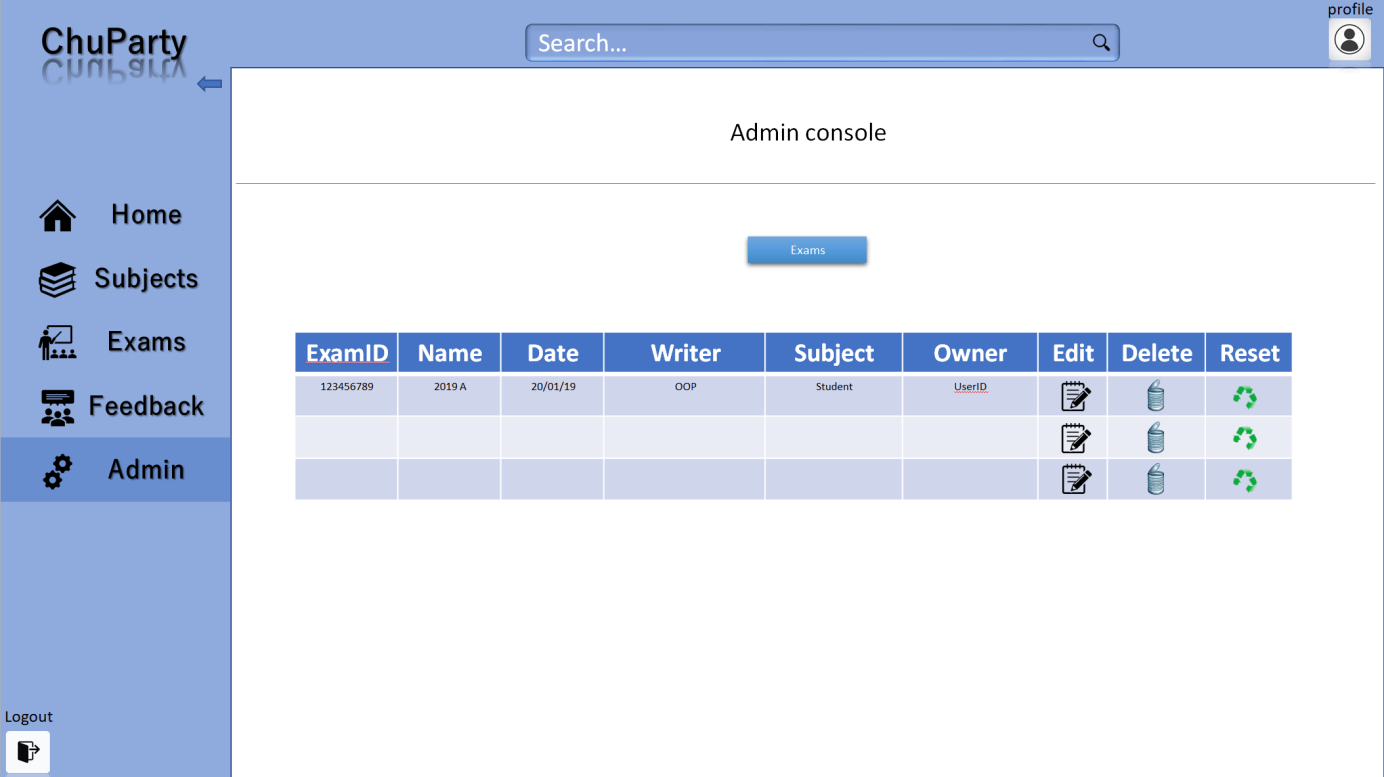
### Feedback dashboard

****

### Admin dashboard





****

# Server Side

The server will be responsible for serving each of the views, communicating with the DB, generating the exams and improve over time by analyzing user statistics.

## API

All API calls will require the user credentials (user ID).

Some may require additional parameters as described in the table below:

|  |  |
| --- | --- |
| **API** | **Description** |
| /api/login | Gets: credentials ; Returns: success/fail |
| /api/logout | Gets: credentials ; Returns: success/fail |
| /api/register | Gets: credentials ; Returns: success/fail |
| /api/getSubjects | Gets: userID ; Returns SubjectGroup[ ]: |
| /api/getExams | Gets: SubjectID ; Returns Exam[ ]: |
| /api/setExam | Gets: Exam; Returns: success/fail |
| /api/generateExam | Gets: Subject ; Returns: Exam |
| /api/pdfToExam | Gets: file ; Returns: Exam |
| /api/deleteExam | Gets: examID ; Returns: success/fail |
| /api/getAllUsers | Gets: credentials ; Returns: User[ ] |
| /api/deleteUser | Gets: UserID ; Returns: success/fail |
| /api/getQuestion | Gets: QuestionID ; Returns: Question |
| /api/deleteQuestion | Gets: QuestionID ; Returns: success/fail |
| /api/search | Gets: searchStr ; Returns:  exam[] | subject[] | subjectGroup[] |