CS3704

Project Milestone 3
Team: Kanban Pioneers

Team Members:
Maddie Gonzalez
Shivani Vaddepalli
Xinchen Laio
Ludwig Olaru
Jiale Song
Ziad Elgataa

Contact info:
maddiegonzalez@vt.edu
shivaniv@vt.edu
sjiale@vt.edu
xinchen@vt.edu
vocaloidfan@vt.edu
ziad21@vt.edu

April 8th, 2024

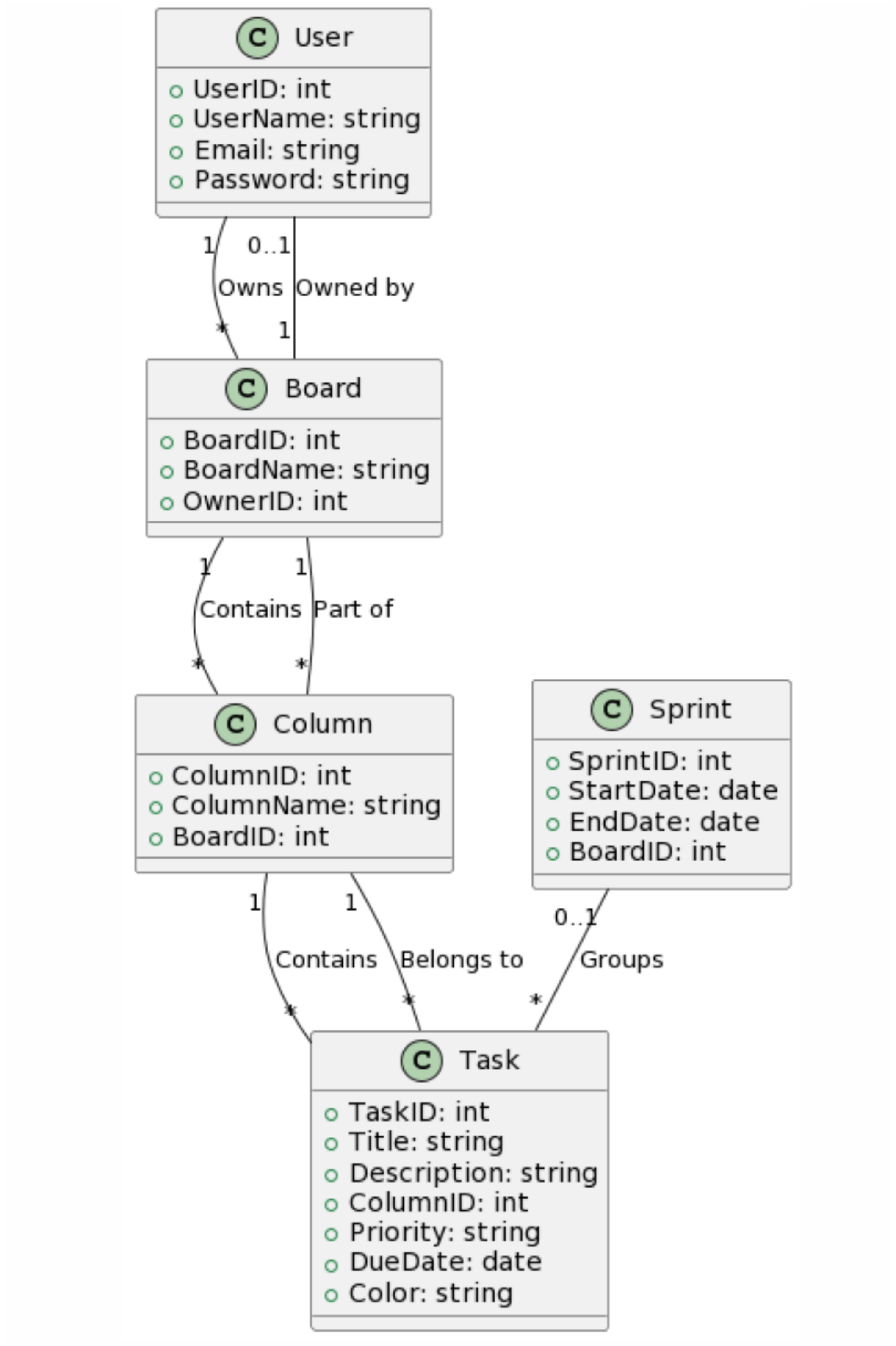
High-Level Design

The main architectural pattern that would make sense for this system is the Client-Server pattern, since we have multiple users all interacting with a common system. The client would be the user, connected to our application in a web page. Every interaction the user makes with their UI will be transmitted to the server, to store data and keep it consistent – after one user makes a change to a board and this change reaches the server, it will then come back to the rest of the clients with access to that board, so that everyone currently editing it will see the same change in real-time.

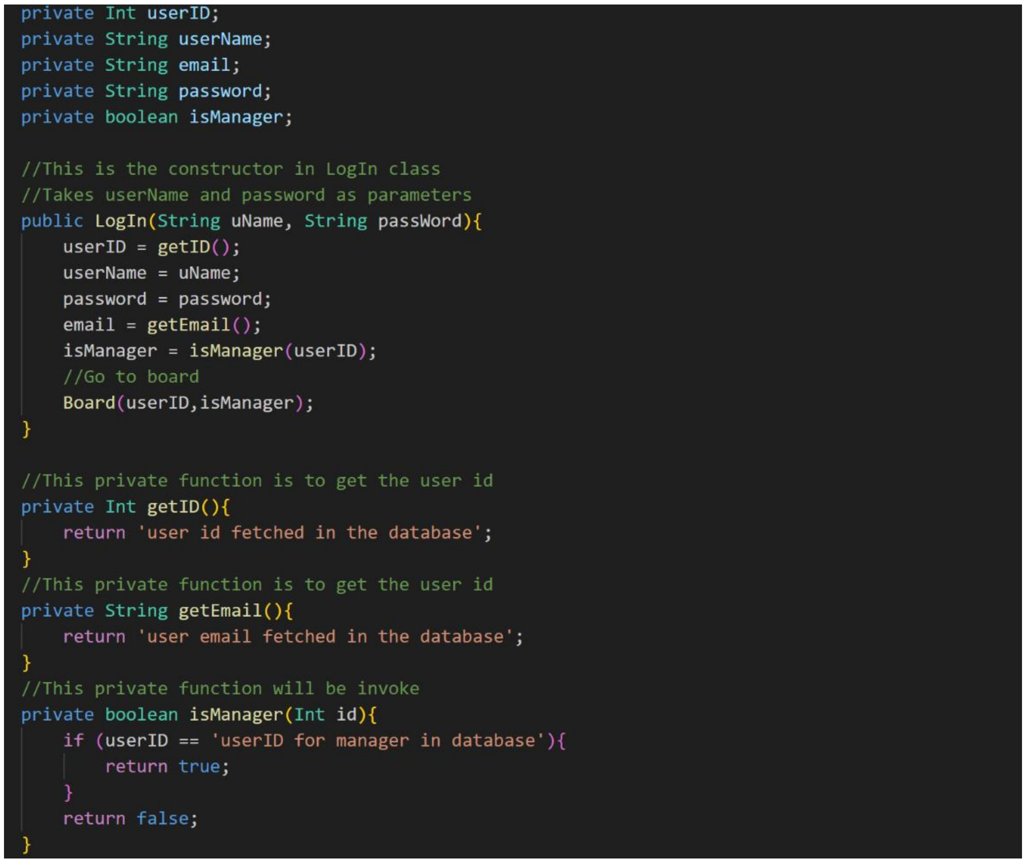
Another applicable pattern would be the Event-Driven pattern, since it would be a pretty intuitive way to design the UI, with each UI component/button/menu/etc. raising an event that would then be caught by a handler, and processed based on the event information, such as which user pressed the button. An example of this: a delete button(for a task), when pressed, would raise an event and be caught by a handler called deleteButtonHandler, which would have to send this information to the server as well as update the client's UI to remove the task.

Low-Level Design

When considering the objectives of our Kanban board project, which include improving customization and maintaining a responsive and user-friendly system, I think the best option would be the Behavioral design pattern family. Dynamic updates and seamless interactions are important to this project and This pattern group is great at creating both . The For example the Observer pattern is great when it comes to cross-user real-time updates. It is what we will need to maintain the quick 200 ms response time that will allow every activity on the board to reflect instantly for all users. The ability to perform undo actions is also a feature of the Command pattern that is essential for a seamless user experience. These patterns will help us not only meet our basic needs but also improve user experience and the maintainability of the system.

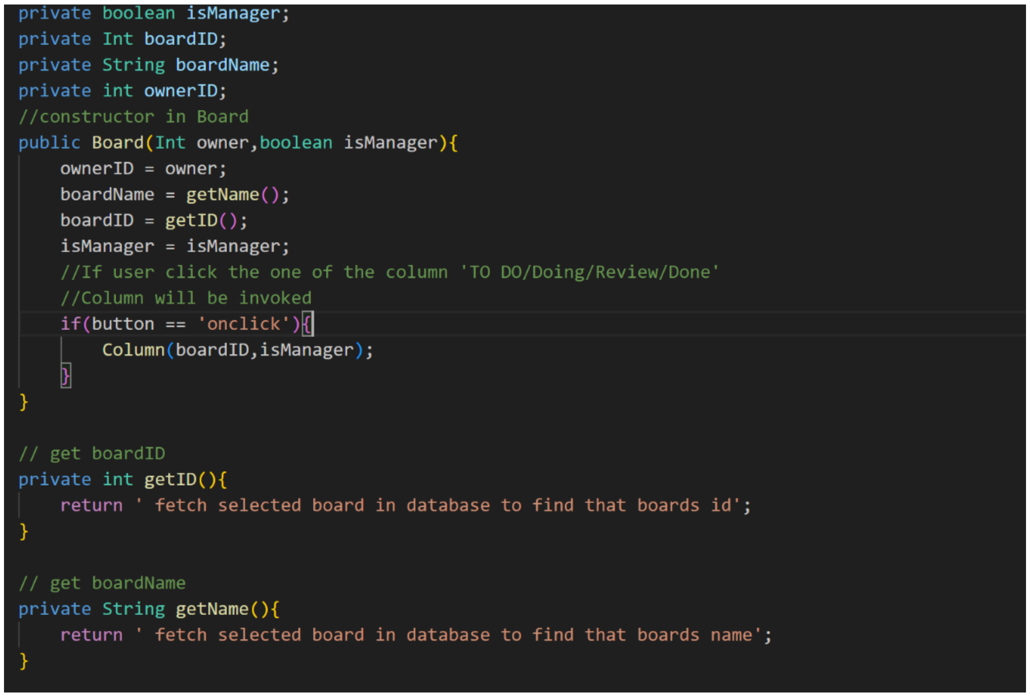


LogIn class (identify user and check if the user is manager)



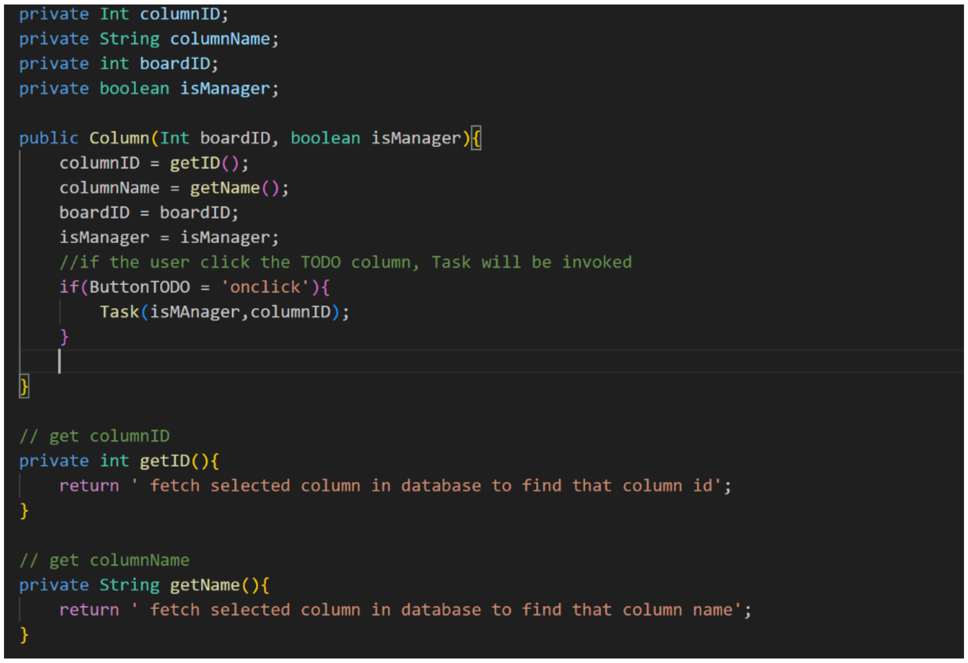
```
private Int userID;
private String userName;
private String email;
private String password;
private boolean isManager;

//This is the constructor in LogIn class
//Takes userName and password as parameters
public LogIn(String uName, String passWord){
    userID = getID();
    userName = uName;
    password = password;
    email = getEmail();
    isManager = isManager(userID);
    //Go to board
    Board(userID,isManager);
}

//This private function is to get the user id
private Int getID(){
    return 'user id fetched in the database';
}
//This private function is to get the user id
private String getEmail(){
    return 'user email fetched in the database';
}
//This private function will be invoke
private boolean isManager(Int id){
    if (userID == 'userID for manager in database'){
        return true;
    }
    return false;
}
```

Board class



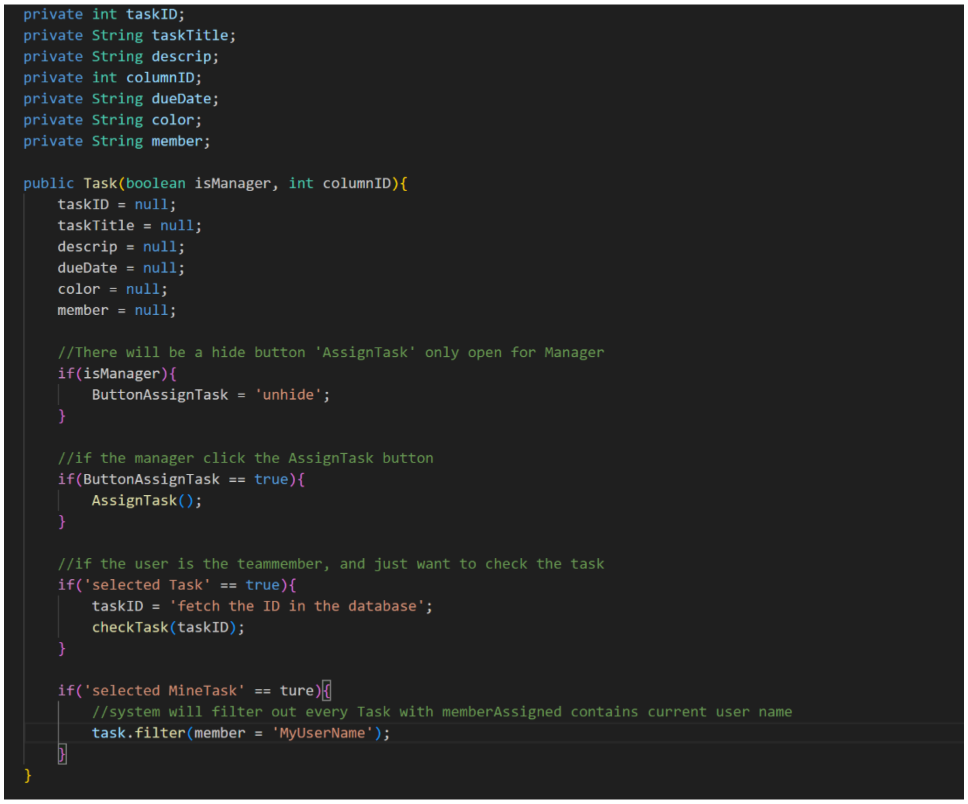
```
private boolean isManager;
private Int boardID;
private String boardName;
private int ownerID;
//constructor in Board
public Board(Int owner,boolean isManager){
    ownerID = owner;
    boardName = getName();
    boardID = getID();
    isManager = isManager;
    //If user click the one of the column 'TO DO/Doing/Review/Done'
    //Column will be invoked
    if(button == 'onclick'){
        Column(boardID,isManager);
    }
}

// get boardID
private int getID(){
    return ' fetch selected board in database to find that boards id';
}

// get boardName
private String getName(){
    return ' fetch selected board in database to find that boards name';
}
```

Column class

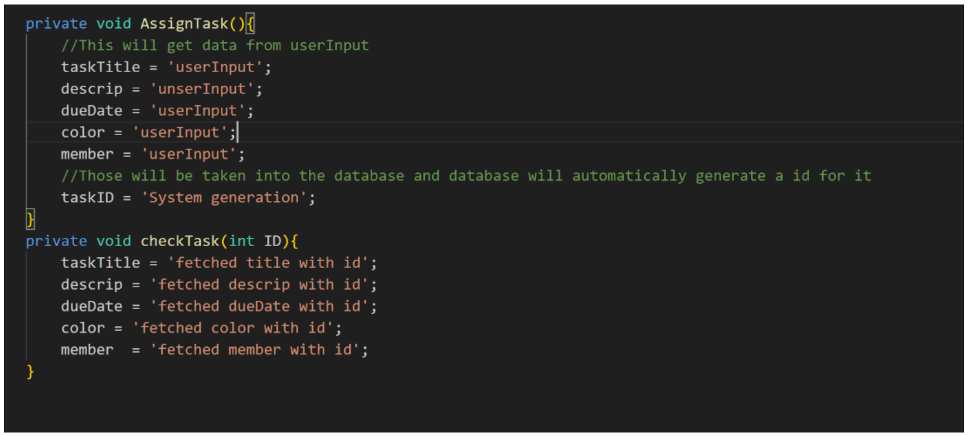


```
private Int columnID;
private String columnName;
private int boardID;
private boolean isManager;

public Column(Int boardID, boolean isManager){
    columnID = getID();
    columnName = getName();
    boardID = boardID;
    isManager = isManager;
    //if the user click the TODO column, Task will be invoked
    if(ButtonTODO = 'onclick'){
        Task(isMAnager,columnID);
    }
    
}

// get columnID
private int getID(){
    return ' fetch selected column in database to find that column id';
}

// get columnName
private String getName(){
    return ' fetch selected column in database to find that column name';
}
```

Task Class



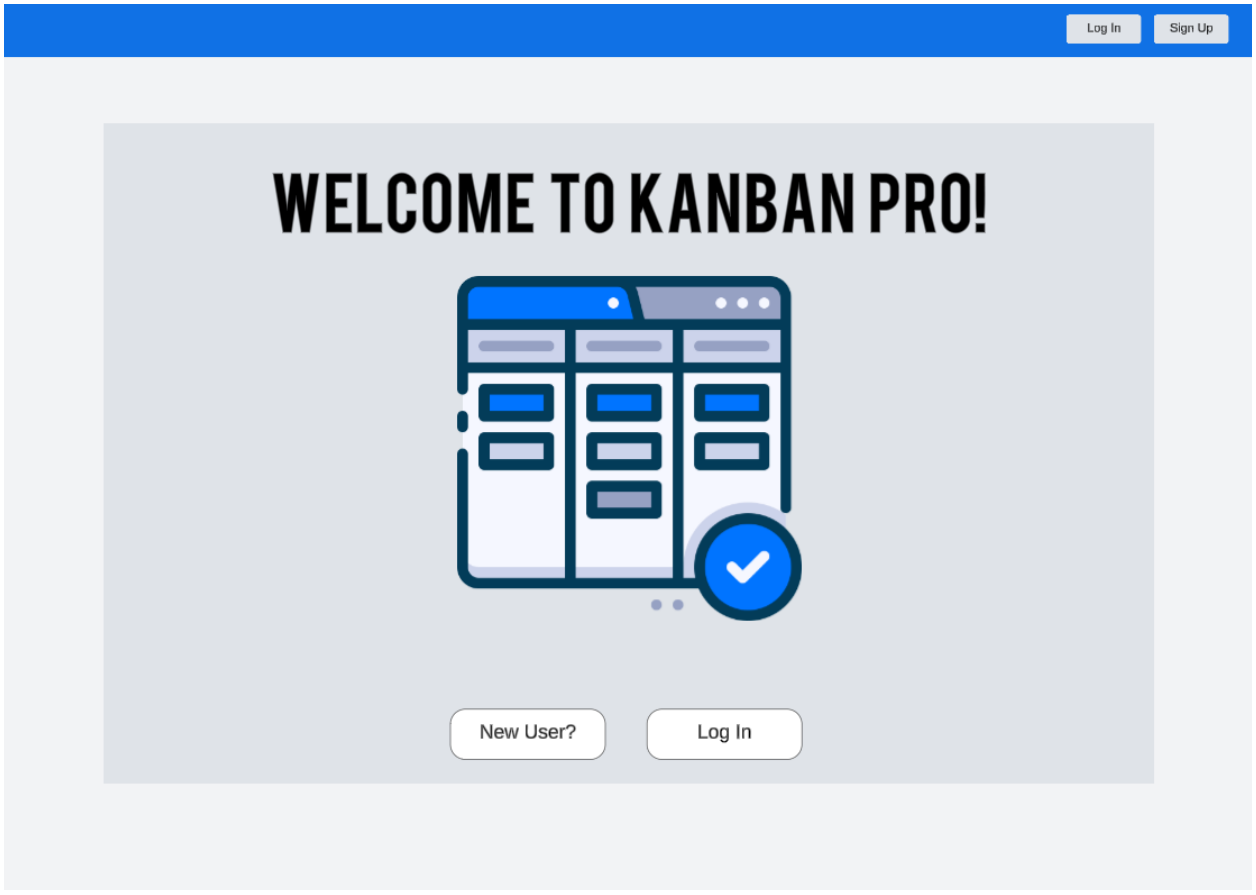
```
private int taskID;
private String taskTitle;
private String descrip;
private int columnID;
private String dueDate;
private String color;
private String member;

public Task(boolean isManager, int columnID){
    taskID = null;
    taskTitle = null;
    descrip = null;
    dueDate = null;
    color = null;
    member = null;

    //There will be a hide button 'AssignTask' only open for Manager
    if(isManager){
        ButtonAssignTask = 'unhide';
    }

    //if the manager click the AssignTask button
    if(ButtonAssignTask == true){
        AssignTask();
    }

    //if the user is the teammember, and just want to check the task
    if('selected Task' == true){
        taskID = 'fetch the ID in the database';
        checkTask(taskID);
    }

    if('selected MineTask' == ture){
        //system will filter out every Task with memberAssigned contains current user name
        task.filter(member = 'MyUserName');
    }
}
```

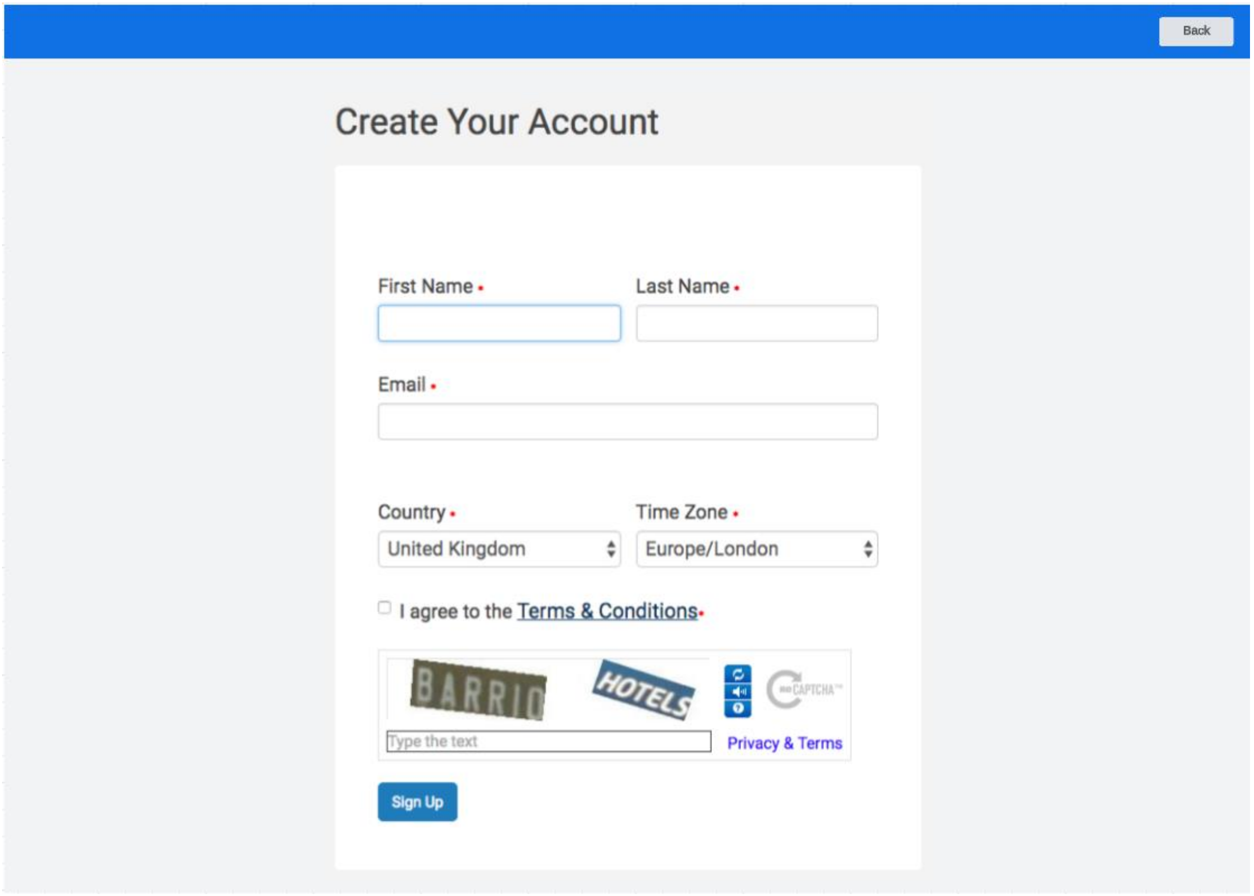


```
private void AssignTask(){
    //This will get data from userInput
    taskTitle = 'userInput';
    descrip = 'unserInput';
    dueDate = 'userInput';
    color = 'userInput';
    member = 'userInput';
    //Those will be taken into the database and database will automatically generate a id for it
    taskID = 'System generation';
}
private void checkTask(int ID){
    taskTitle = 'fetched title with id';
    descrip = 'fetched descrip with id';
    dueDate = 'fetched dueDate with id';
    color = 'fetched color with id';
    member  = 'fetched member with id';
}
```

Design Sketch

For the design sketch, we chose to create a wireframe for our web application. We chose to keep the interface very simple and user-friendly to promote accessibility and usability.




Back

Create Your Account

First Name •

Last Name •

Email •

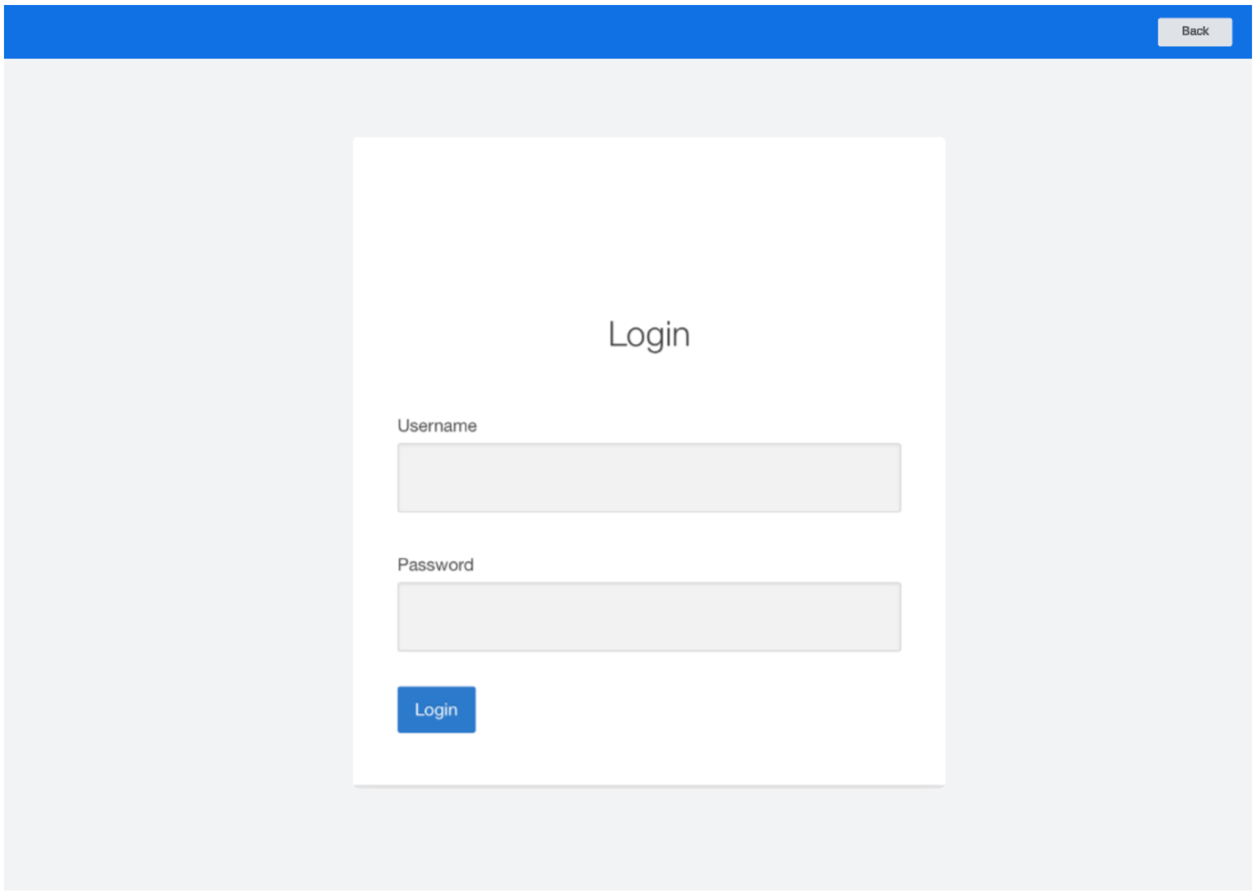
Country • United Kingdom

Time Zone • Europe/London

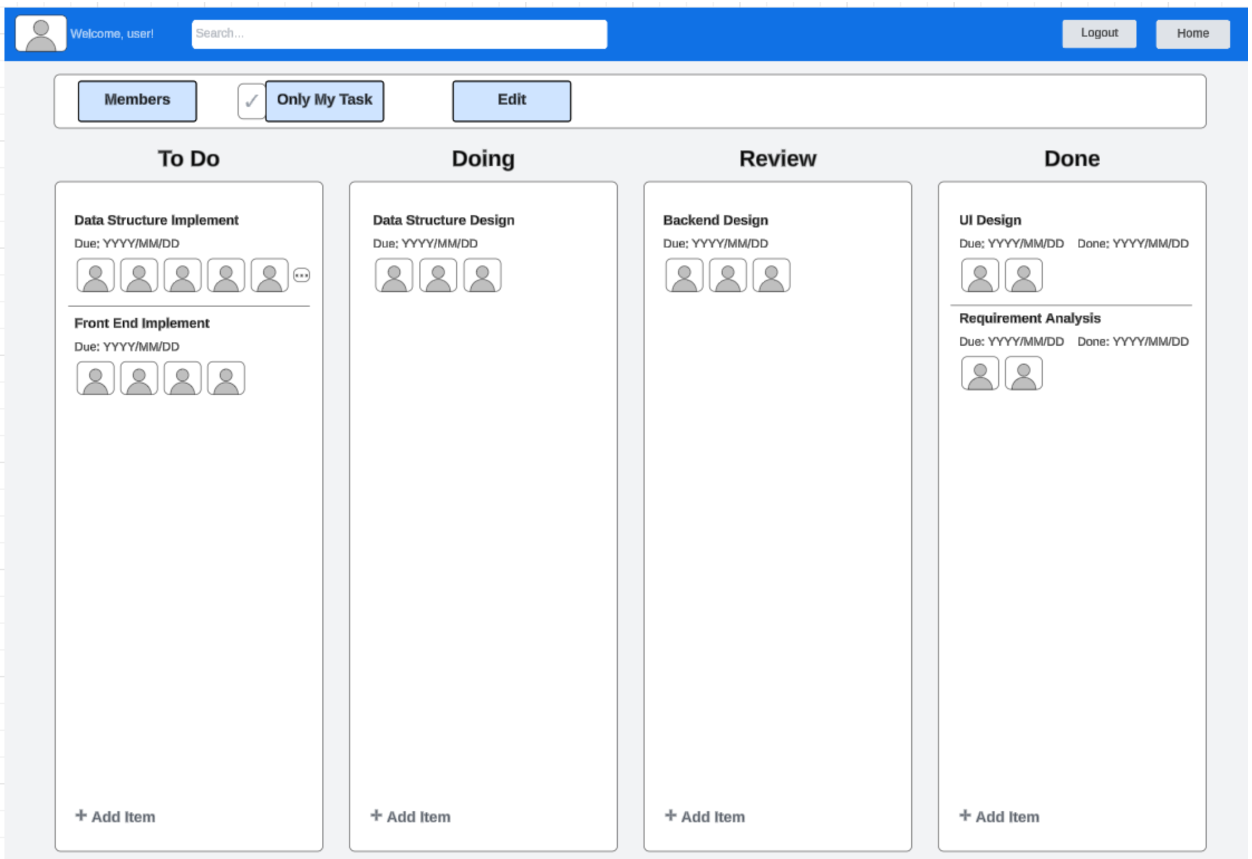
☐ I agree to the Terms & Conditions•

Sign Up

On the first page, users have the option to either create an account or log in, which opens a simple interface as seen on most common web applications. The sign up page will collect information about a user's time zone due to the nature of deadlines our tool provides, which will be discussed later.

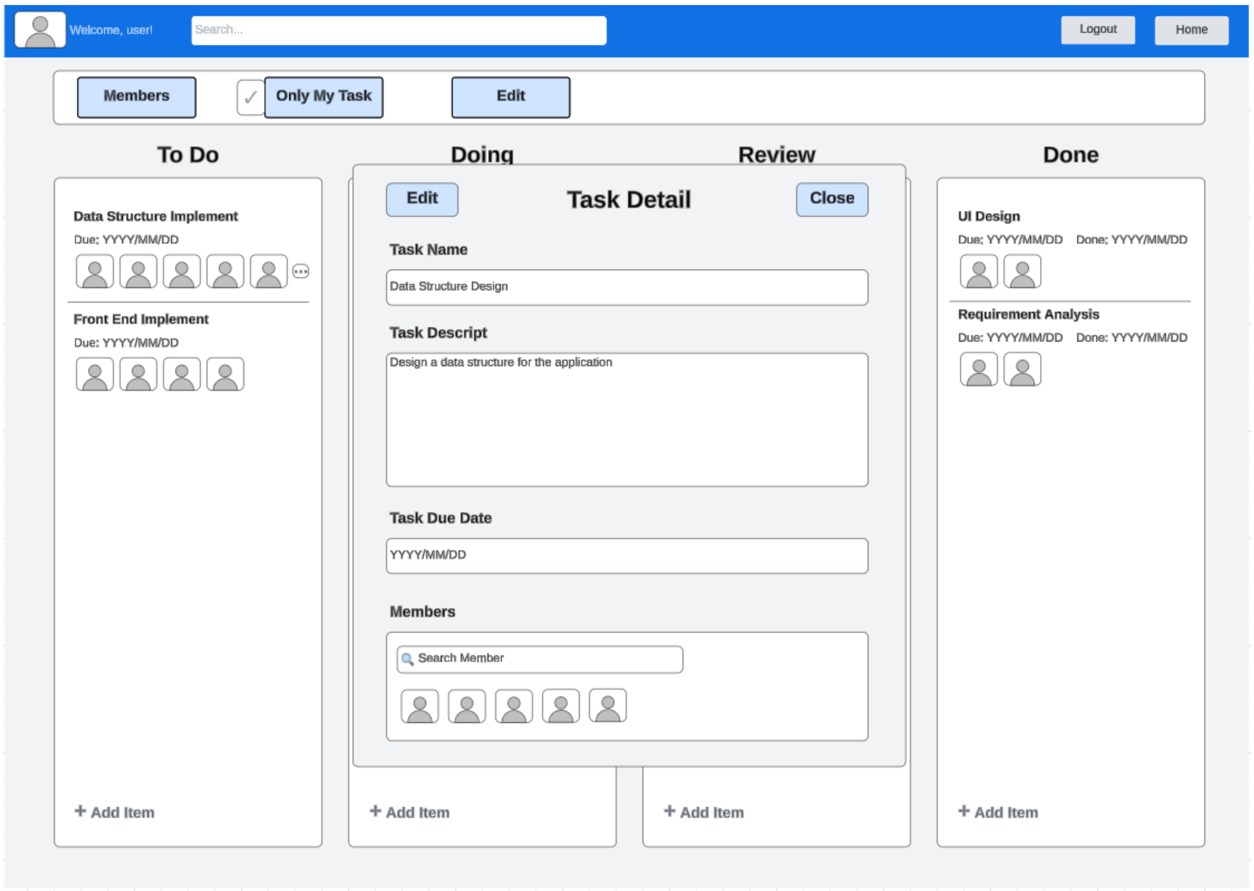


The login page will be very simple as the functionality does not require complex visuals. Once a user logs in, they will see a kanban board specific to their team and projects.



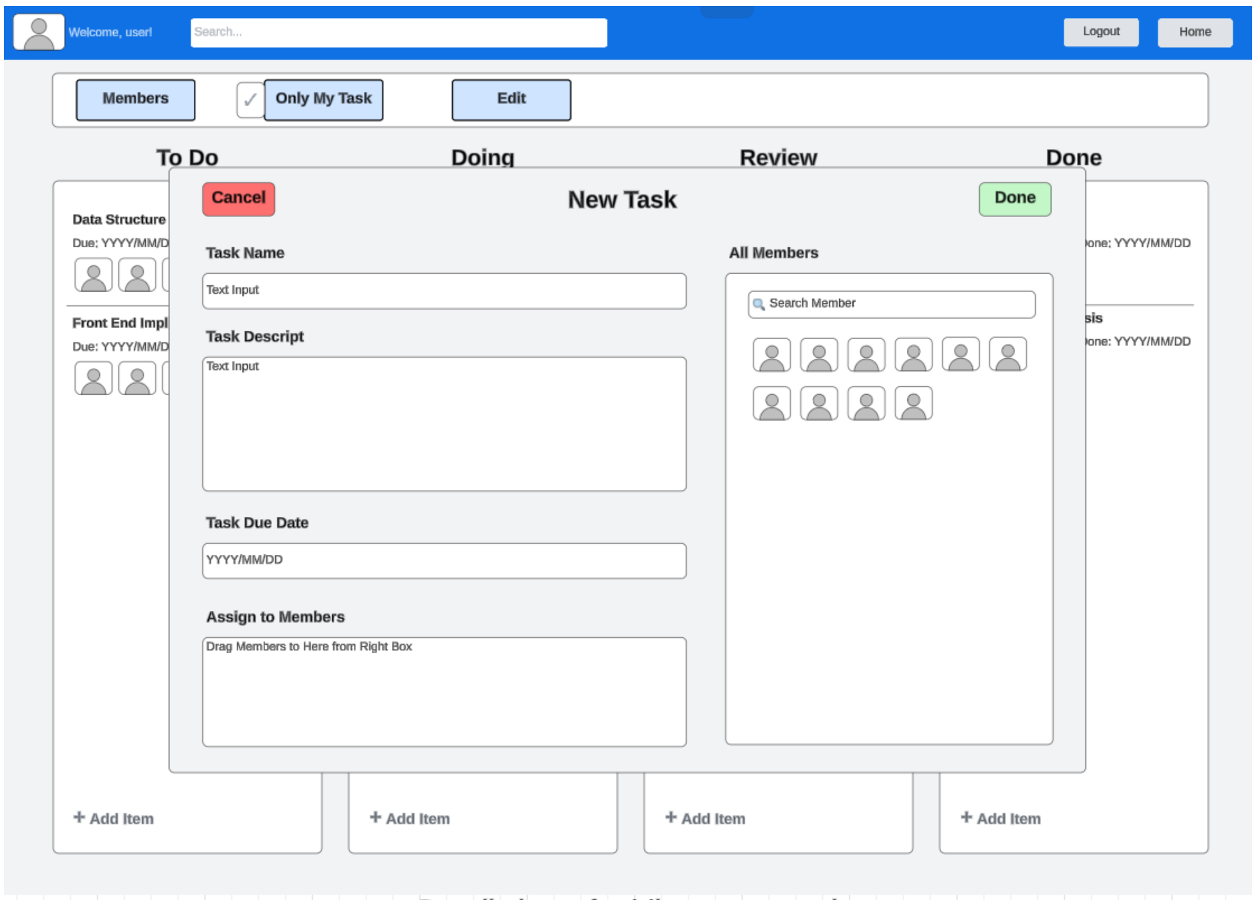
Home page. Overall View of Kanban boards, my tasks, deadlines, add new button, to-do list, etc.

The home page gives users an overall view of all the features we chose to incorporate within our app. The main view consists of the kanban board since this is the focus of our project. We chose to add filter buttons along the top where they are easily accessible to encourage users to try our new functionality.



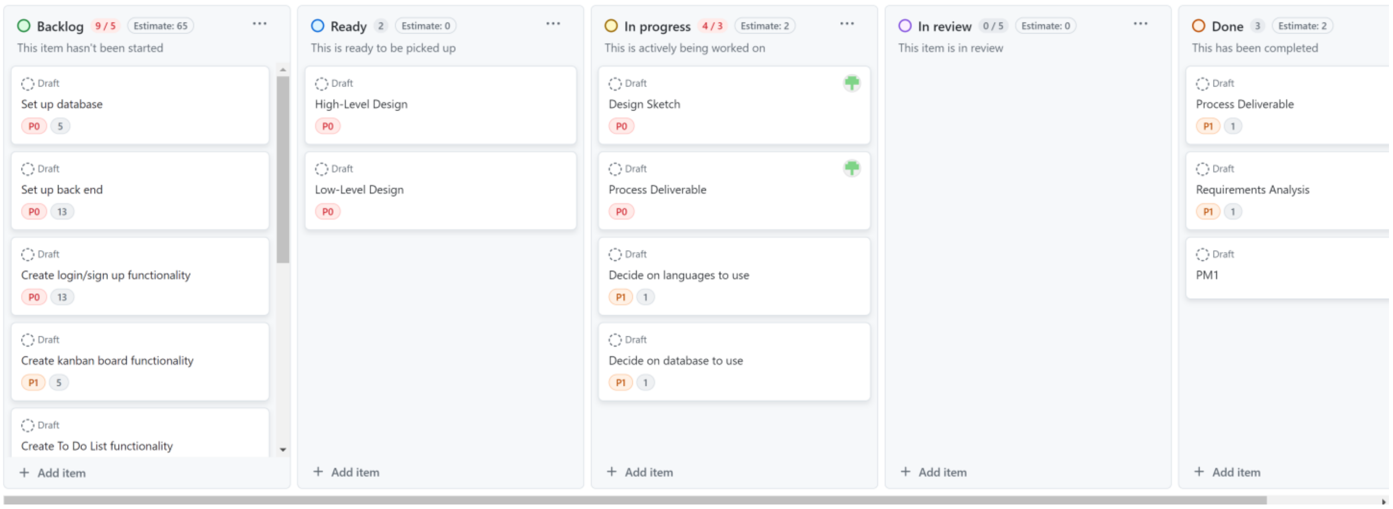
Detail view of a clicked-on task

When a user clicks on a specific task or clicks to add a new task, the detailed view or new task view (See next Figure) opens, respectively. This view grants users the ability to edit task details, assign members to tasks, and set deadlines to tasks. Again, we chose to embrace a simple interface to promote usability of this feature.



Detail view of adding a new task

Process Deliverable



Backlog 9 / 5 Estimate: 65
This item hasn't been started

- Draft — Set up database — P0 — 5
- Draft — Set up back end — P0 — 13
- Draft — Create login/sign up functionality — P0 — 13
- Draft — Create kanban board functionality — P1 — 5
- Draft — Create To Do List functionality

+ Add item

Ready 2 Estimate: 0
This is ready to be picked up

- Draft — High-Level Design — P0
- Draft — Low-Level Design — P0

+ Add item

In progress 4 / 3 Estimate: 2
This is actively being worked on

- Draft — Design Sketch — P0
- Draft — Process Deliverable — P0
- Draft — Decide on languages to use — P1 — 1
- Draft — Decide on database to use — P1 — 1

+ Add item

In review 0 / 5 Estimate: 0
This item is in review

+ Add item

Done 3 Estimate: 2
This has been completed

- Draft — Process Deliverable — P1 — 1
- Draft — Requirements Analysis — P1 — 1
- Draft — PM1

+ Add item

We still have the same tasks in our backlog as PM2, but we have moved over a few tasks to Ready and In Progress. Once we begin development soon, we will start working on most of the items in the backlog.