



BILKENT UNIVERSITY

CS353 DATABASE SYSTEMS

---

**UptownFANK Project Tracking Software**  
Final Report  
Group 28

---

*Authors:*

Fatbardh Feta

Alemdar Salmoor

Naisila Puka

Kunduz Efronova

*Student IDs:*

21600334

21500430

21600336

21600469

May 21, 2019

# Contents

<b>1</b>	<b>Brief Description</b>	<b>3</b>
<b>2</b>	<b>Final E/R</b>	<b>3</b>
<b>3</b>	<b>Final List of Tables</b>	<b>5</b>
3.1	BasicUser . . . . .	5
3.2	Phone . . . . .	5
3.3	SuperUser . . . . .	5
3.4	Team . . . . .	6
3.5	Member . . . . .	6
3.6	Board . . . . .	7
3.7	WorksOn . . . . .	7
3.8	List . . . . .	7
3.9	Card . . . . .	8
3.10	PerformsTask . . . . .	8
3.11	CheckList . . . . .	9
3.12	Item . . . . .	9
3.13	Comment . . . . .	9
3.14	Replies . . . . .	10
3.15	Upvotes . . . . .	10
3.16	Attachment . . . . .	11
3.17	Label . . . . .	11
3.18	Labelling . . . . .	11
<b>4</b>	<b>Implementation details</b>	<b>12</b>
4.1	Programming Languages . . . . .	12
4.2	Hosting the Application . . . . .	12
4.3	Database Management System . . . . .	12
4.4	Environment . . . . .	13
4.5	Frameworks Used . . . . .	13
4.6	Challenges Faced . . . . .	13

2

## 1 Brief Description

This is the final report for our project tracking software called UptownFank. UptownFank is a simple and useful tool for tracking different projects, usually shared between some team members. It provides a way for maintaining project communication and keeping track of the accomplished and to-do tasks. Each project has boards where lists can be added and seen by all the team members. Lists hold a group of cards that are related to each other by their content. Users can navigate to each of the lists of the project boards and read any of the cards in that list. Cards can also have names, descriptions, priorities and deadlines. Each project is composed of one or many members that can access all the boards. UptownFank offers a user friendly interface which is designed very straight forwardly. Our program uses some complex queries to offer more detailed info about the projects and teams that are registered in our database.

## 2 Final E/R

There was no need to change the E/R diagram that we made for the Design Report so we decided to use the same E/R diagram in the Final Report also. Our implementation is made according to this E/R:

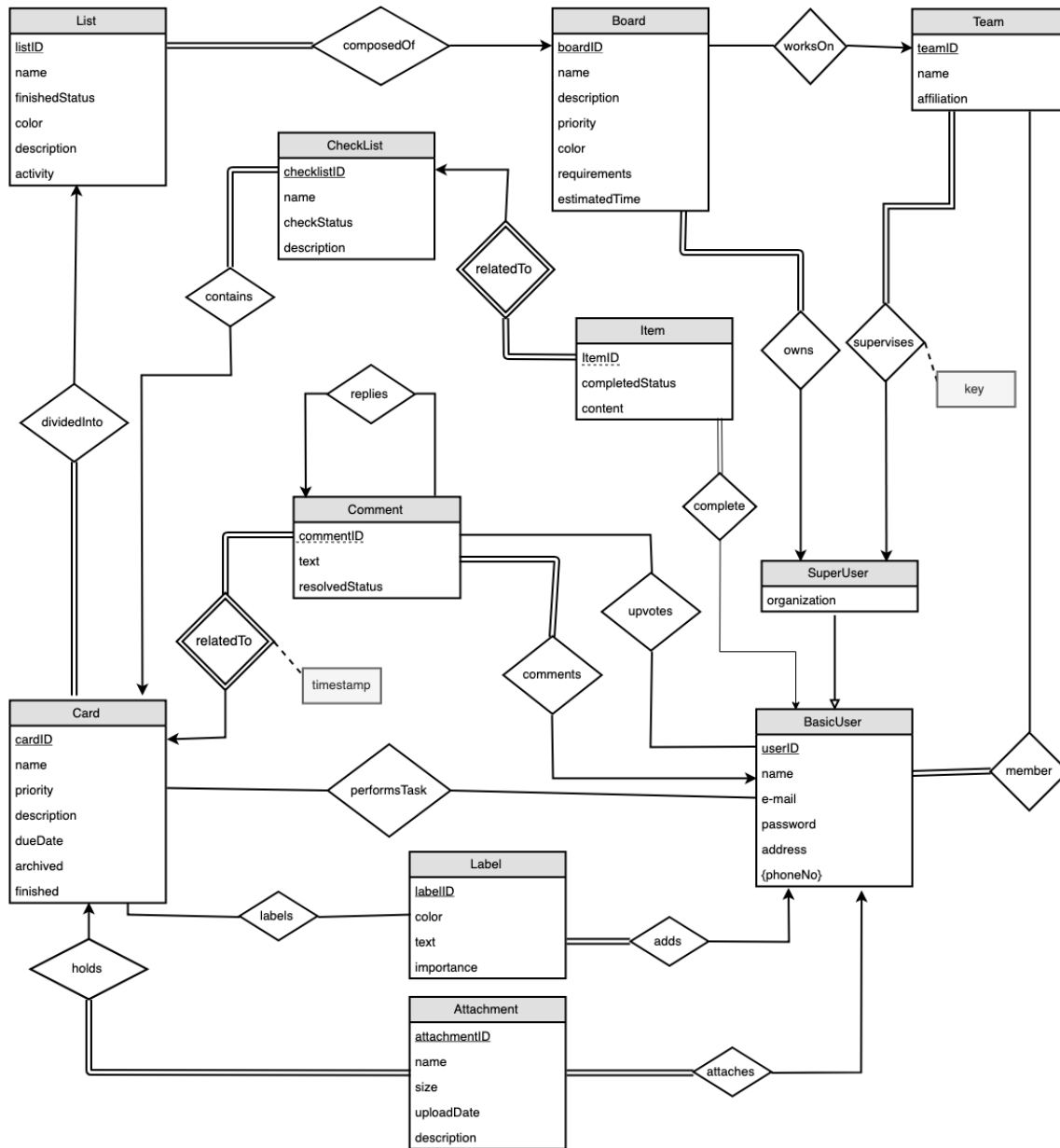


Figure 1: Entity-Relationship Diagram

### 3 Final List of Tables

In this section we show the final version of the tables that we used in our project.

#### 3.1 BasicUser

**Relational Model:** BasicUser(userID, name, email, password, address)

**Functional Dependencies:**  $\{(userID \rightarrow name, email, password, address), (email \rightarrow userID)\}$

**Primary Key:** {userID}

**Candidate Keys:** {{userID}, {email}}

**Foreign Keys:** {}

**Normal Form:** BCNF

#### 3.2 Phone

**Relational Model:** Phone(userID, phoneNo)

**Functional Dependencies:** {}

**Primary Key:** {userID, phoneNo}

**Candidate Keys:** {{userID, phoneNo}}

**Foreign Keys:**  $\{(userID \rightarrow \text{BasicUser.userID})\}$

**Normal Form:** BCNF

#### 3.3 SuperUser

**Relational Model:** SuperUser(userID, organization)

**Functional Dependencies:**  $\{(userID \rightarrow organization)\}$

**Primary Key:** {userID}

**Candidate Keys:** {{userID}}

**Foreign Keys:** {(userID → BasicUser.userID)}

**Normal Form:** BCNF

### 3.4 Team

**Relational Model:** Team(teamID, name, affiliation, supervisor, key)

**Functional Dependencies:** {(teamID → name, affiliation, supervisor, key), (key → teamID)}

**Primary Key:** {userID}

**Candidate Keys:** {{userID}, {key}}

**Foreign Keys:** {(supervisor → SuperUser.userID)}

**Normal Form:** BCNF

### 3.5 Member

**Relational Model:** Member(userID, teamID)

**Functional Dependencies:** {}

**Primary Key:** {userID, teamID}

**Candidate Keys:** {{userID, teamID}}

**Foreign Keys:** {(userID → BasicUser.userID), (teamID → Team.teamID)}

**Normal Form:** BCNF

### 3.6 Board

**Relational Model:** Board(boardID, name, description, priority, color, requirements, estimatedTime, ownerID)

**Functional Dependencies:**  $\{(\text{boardID} \rightarrow \text{name}, \text{description}, \text{priority}, \text{color}, \text{requirements}, \text{estimatedTime}, \text{ownerID})\}$

**Primary Key:** {boardID}

**Candidate Keys:** {{boardID}}

**Foreign Keys:**  $\{(\text{owner} \rightarrow \text{SuperUser.userID})\}$

**Normal Form:** BCNF

### 3.7 WorksOn

**Relational Model:** WorksOn(boardID, teamID)

**Functional Dependencies:**  $\{(\text{boardID} \rightarrow \text{teamID})\}$

**Primary Key:** {boardID}

**Candidate Keys:** {{boardID}}

**Foreign Keys:**  $\{(\text{boardID} \rightarrow \text{Board.boardID}), (\text{teamID} \rightarrow \text{Team.teamID})\}$

**Normal Form:** BCNF

### 3.8 List

**Relational Model:** List(listID, name, finishedStatus, color, description, activity, boardID)

**Functional Dependencies:**  $\{(\text{listID} \rightarrow \text{name}, \text{finishedStatus}, \text{color}, \text{description}, \text{activity}, \text{boardID})\}$

**Primary Key:** {listID}



**Candidate Keys:**  $\{\{listID\}\}$

**Foreign Keys:**  $\{(boardID \rightarrow Board.boardID)\}$

**Normal Form:** BCNF

### 3.9 Card

**Relational Model:** Card(cardID, name, priority, description, dueDate, listID, archived, finished)

**Functional Dependencies:**  $\{(cardID \rightarrow name, priority, description, dueDate, listID, archived, finished)\}$

**Primary Key:**  $\{cardID\}$

**Candidate Keys:**  $\{\{cardID\}\}$

**Foreign Keys:**  $\{(listID \rightarrow List.listID)\}$

**Normal Form:** BCNF

### 3.10 PerformsTask

**Relational Model:** PerformsTask(cardID, userID)

**Functional Dependencies:**  $\{\}$

**Primary Key:**  $\{cardID, userID\}$

**Candidate Keys:**  $\{\{cardID, userID\}\}$

**Foreign Keys:**  $\{(cardID \rightarrow Card.cardID), (userID \rightarrow User.userID)\}$

**Normal Form:** BCNF

### 3.11 CheckList

**Relational Model:** CheckList(checklistID, name, checkStatus, description, relatedCard)

**Functional Dependencies:**  $\{(\text{checklistID} \rightarrow \text{name}, \text{checkStatus}, \text{description}, \text{relatedCard})\}$

**Primary Key:** {checklistID}

**Candidate Keys:** {{checklistID}}

**Foreign Keys:**  $\{(\text{relatedCard} \rightarrow \text{Card.cardID})\}$

**Normal Form:** BCNF

### 3.12 Item

**Relational Model:** Item(itemID, relatedChecklist, completedStatus, content, completor)

**Functional Dependencies:**  $\{(\text{itemID}, \text{relatedChecklist} \rightarrow \text{completedStatus}, \text{content}, \text{completor})\}$

**Primary Key:** {itemID, relatedChecklist}

**Candidate Keys:** {{itemID, relatedChecklist}}

**Foreign Keys:**  $\{(\text{relatedChecklist} \rightarrow \text{CheckList.checklistID}), (\text{completor} \rightarrow \text{BasicUser.userID})\}$

**Normal Form:** BCNF

### 3.13 Comment

**Relational Model:** Comment(commentID, relatedCard, timestamp, resolvedStatus, commenter, text)

**Functional Dependencies:**  $\{(\text{commentID}, \text{relatedCard} \rightarrow \text{timestamp}, \text{resolved-Status}, \text{commenter}, \text{text})\}$

**Primary Key:**  $\{\text{commentID}, \text{relatedCard}\}$

**Candidate Keys:**  $\{\{\text{commentID}, \text{relatedCard}\}\}$

**Foreign Keys:**  $\{(\text{relatedCard} \rightarrow \text{Card.cardID}), (\text{commenter} \rightarrow \text{BasicUser.userID})\}$

**Normal Form:** BCNF

### 3.14 Replies

**Relational Model:** Replies(replyID, relatedCard, commentID)

**Functional Dependencies:**  $\{(\text{replyID}, \text{relatedCard} \rightarrow \text{commentID})\}$

**Primary Key:**  $\{\text{replyID}, \text{relatedCard}\}$

**Candidate Keys:**  $\{\{\text{replyID}, \text{relatedCard}\}\}$

**Foreign Keys:**  $\{(\text{replyID}, \text{relatedCard} \rightarrow \text{Comment}(\text{commentID}, \text{relatedCard})), (\text{commentID}, \text{relatedCard} \rightarrow \text{Comment}(\text{commentID}, \text{relatedCard}))\}$

**Normal Form:** BCNF

### 3.15 Upvotes

**Relational Model:** Upvotes(userID, commentID, relatedCard)

**Functional Dependencies:**  $\{\}$

**Primary Key:**  $\{\text{userID}, \text{commentID}, \text{relatedCard}\}$

**Candidate Keys:**  $\{\{\text{userID}, \text{commentID}, \text{relatedCard}\}\}$

**Foreign Keys:**  $\{(\text{commentID}, \text{relatedCard} \rightarrow \text{Comment}(\text{commentID}, \text{relatedCard})), (\text{userID} \rightarrow \text{BasicUser.userID})\}$

**Normal Form:** BCNF

### 3.16 Attachment

**Relational Model:** Attachment(attachmentID, name, size, uploadDate, description, attacher, relatedCard)

**Functional Dependencies:**  $\{(\text{attachmentID} \rightarrow \text{name}, \text{size}, \text{uploadDate}, \text{description}, \text{attacher}, \text{relatedCard})\}$

**Primary Key:** {attachmentID}

**Candidate Keys:** {{attachmentID}}

**Foreign Keys:**  $\{(\text{attacher} \rightarrow \text{BasicUser.userID}), (\text{relatedCard} \rightarrow \text{Card.cardID})\}$

**Normal Form:** BCNF

### 3.17 Label

**Relational Model:** Label(labelID, color, text, importance, adder)

**Functional Dependencies:**  $\{(\text{labelID} \rightarrow \text{color}, \text{text}, \text{importance}, \text{adder})\}$

**Primary Key:** {labelID}

**Candidate Keys:** {{labelID}}

**Foreign Keys:**  $\{(\text{adder} \rightarrow \text{BasicUser.userID})\}$

**Normal Form:** BCNF

### 3.18 Labelling

**Relational Model:** Label(cardID, labelID)

**Functional Dependencies:** {}

**Primary Key:** {cardID, labelID}

**Candidate Keys:** {{cardID, labelID}}

**Foreign Keys:**  $\{(cardID \rightarrow Card.cardID), (labelID \rightarrow Label.labelID)\}$

**Normal Form:** BCNF

## 4 Implementation details

In this section, we provide some implementation details of our web application.

### 4.1 Programming Languages

We used PHP for the back-end programs, HTML/CSS for the front-end programs, and JavaScript to implement a controller that provides connection between back-end and front-end. We chose to fully divide back-end and front-end programs in order to be able to divide the work of the team members more easily. Also, the work of the JavaScript could be done alone.

### 4.2 Hosting the Application

We used `dijkstra.ug.bcc.bilkent.edu.tr` server to host our application, since it was an easily accessible server to use for us as Bilkent students.

### 4.3 Database Management System

UptownFANK uses MySQL as its database management system. We chose MySQL because all of the team members were familiar with it, so it was easier to propagate through project steps. We wanted to use PostgreSQL since it is well-known and on its peak at these times, but only half of the team was familiar with it. Therefore, we picked MySQL, considering the amount of time in which we had to get the project delivered.

## 4.4 Environment

You may find all the source code of our project in our **open-source repository** (don't forget to star us if you enjoy UptownFANK): <https://github.com/NaisilaPuka/UptownFANK>.

## 4.5 Frameworks Used

We used Bootstrap since it is a build responsive front-end component library.

## 4.6 Challenges Faced

We learned a bit about PHP and HTML/CSS in the programming assignment we had for CS353-Database Systems course, however it was not enough. We had to learn many more new methods in order to be able to actually provide something meaningful for UptownFANK.

Our knowledge on JavaScript was limited as well, since only one member of the team fully understood it. So that member had to teach the others as well, which took some of the amount of time we had.

The other challenge was GitHub itself. Only half of the team knew GitHub before, so the other part of the team had to learn it in order to be able to work on the same remote repository on different branches, and then open pull requests, request reviews from each other etc.

# 5 Advanced DB features

Our system provides the following advanced database features.

## 5.1 Reports

In order to provide insightful analysis about our software, we have prepared the following reports:

- The team with the largest number of members, along with its supervisor and the number of members:

```
WITH membersNo AS(SELECT teamID, COUNT(userID) as memberCount
FROM Member GROUP BY teamID)
```

```
SELECT t.name, t.supervisor, m.memberCount FROM Teams t NATURAL
JOIN membersNo m
```

```
WHERE m.memberCount >= ALL (SELECT memberCount FROM mem-
bersNo);
```

- Users who are part of all projects supervised by a particular user:

```
SELECT u.name, u.email FROM BasicUser u WHERE NOT EXISTS(
(SELECT t.teamID FROM Teams t WHERE t.supervisor = @particularID)
EXCEPT
```

```
(SELECT m.teamID FROM Member m WHERE m.userID = u.userID));
```

- 10 Most popular cards in the whole database based on their comments, along with the list and board they belong to:

```
WITH commentsNo AS(SELECT relatedCard AS cardID, COUNT(commentID)
AS commentsNo FROM Comment GROUP BY relatedCard)
```

```
SELECT c.name AS CardName, c.commentsNo AS NumberOfComments, l.name
AS ListName, b.name AS BoardName
```

```
FROM commentsNo c JOIN Card cd JOIN List ls JOIN Board b
```

```
ON(c.cardID = cd.cardID AND cd.listID = ls.listID AND ls.boardID = b.boardID)
```

```
ORDER BY c.commentsNo DESC LIMIT 10;
```

For the moment, we have some insight on the first two reports of our application. These preliminary results (actual outputs) of structures used in our project come from random populations that we provided to test our application, as in Figure 2.

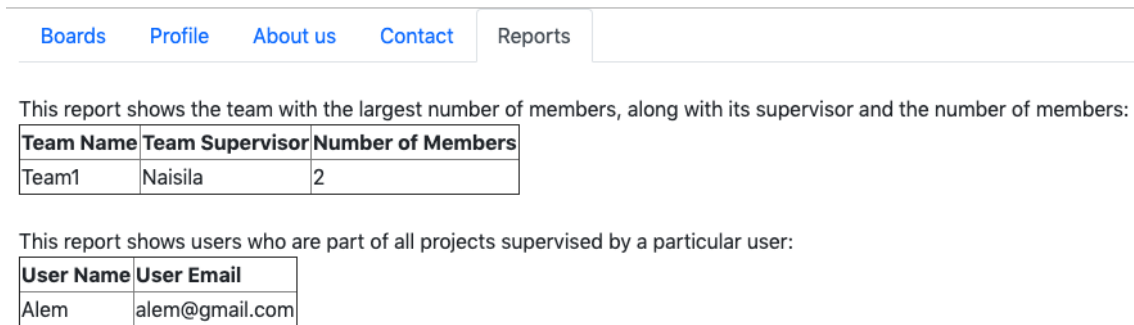


Figure 2: Simple Preliminary Report Results

## 6 User's Manual

In this section we are going to give a short explanation on how to use our program together with some of its main features and functionalities.

### 6.1 Register and Log In

In order to be able to use our program each user must firstly create an account. Our system provides support for two type of users, Super User and Basic User. Both of these users should firstly enter their Email address, password, name, physical address and phone number. When a costumer wants to register as a SuperUser they have to check the radio box corresponding to Super User and provide information regarding the Organization that they are part of.





The image shows the registration page for UptownFANK. At the top, there is a logo consisting of a blue and white geometric shape resembling a stylized 'U' or a building, with the text 'UptownFANK' and 'Your Project Tracking Software' below it. Below the logo, there are several input fields for registration: 'Email address' (containing 'john.smith@yahoo.com'), 'Password' (masked with dots), 'Name' (containing 'John Smith'), 'Address' (containing 'Florida, USA'), 'Phone Number' (containing '5438409169'), 'User Type' (with radio buttons for 'Basic User' and 'Super User'), and 'Organization' (containing 'NASA'). A blue 'Register' button is at the bottom of the form. To the right of the form, there is a Windows watermark that says 'Activate Windows Go to Settings to activate Windows.' At the very bottom, there is a link that says 'Already registered, Click here to Login!'.

Figure 3: Register-Page

After the new user has successfully finished the registration they will be logged in automatically. Then the user can access his/her account through the log in screen where they will be asked to enter their email and password. Any mistake made during the insertion of credentials will result in an error and the user won't be able to enter in their account.

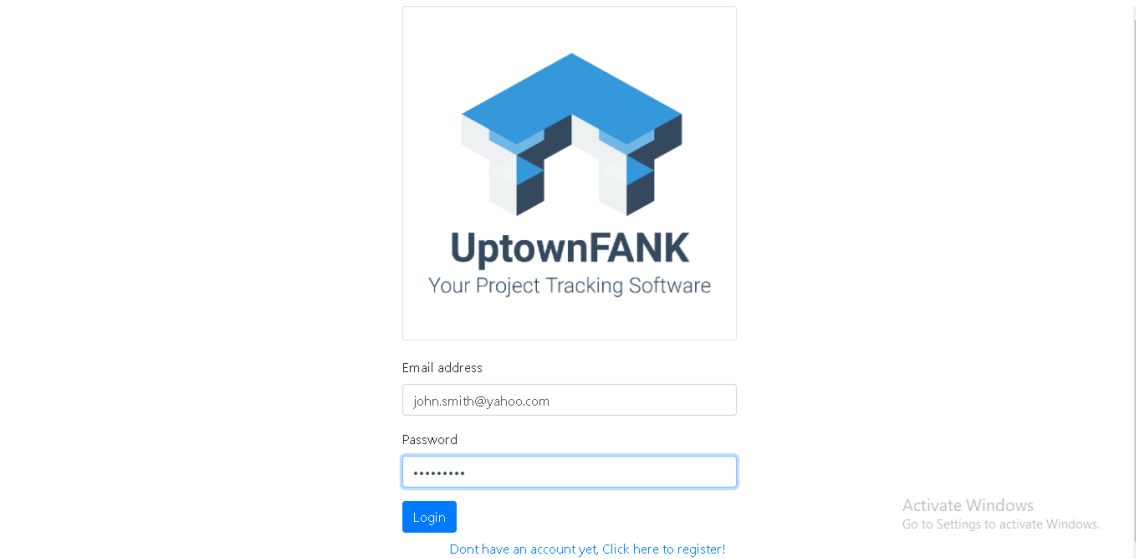


Figure 4: Log-In

## 6.2 Creating new Teams and Boards

When a super user has successfully logged in he will be able to create a team by clicking the NewTeam button. Then they should fill the pop-up window with the required info about the new team, namely the Team Name, Affiliation and Unique Key. The creator of a team can also delete the team and also assign members to the teams.

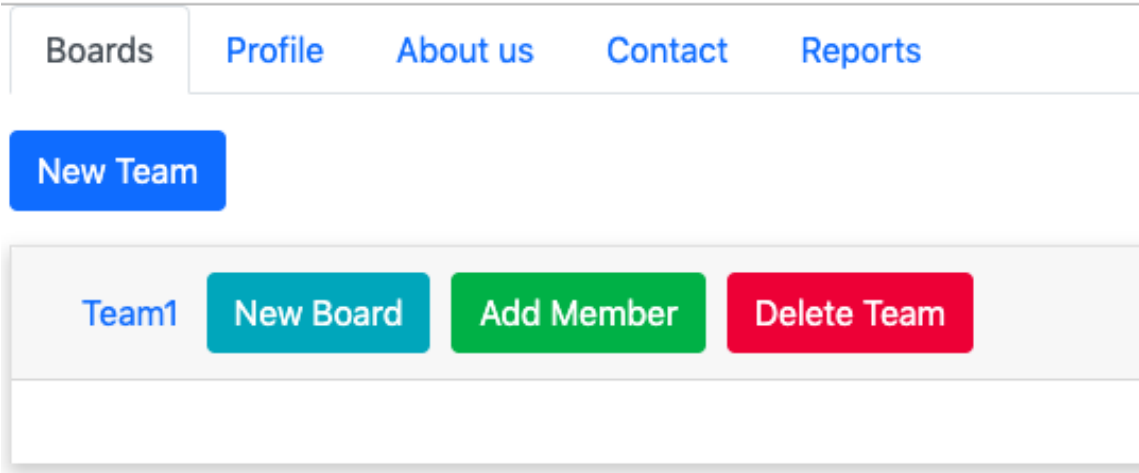
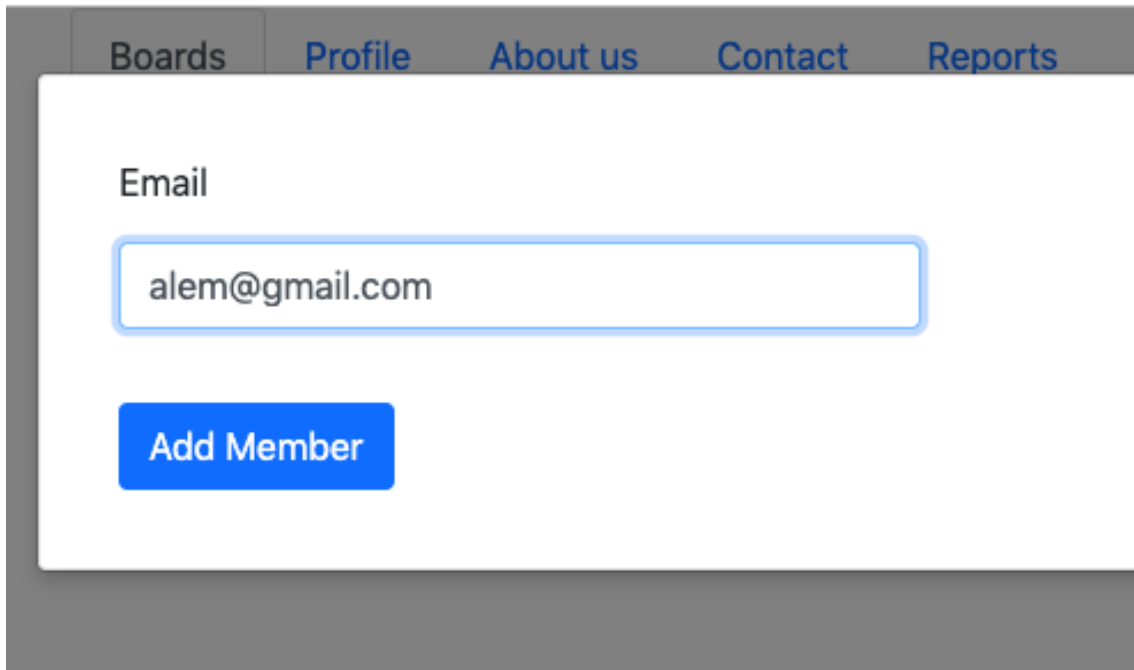


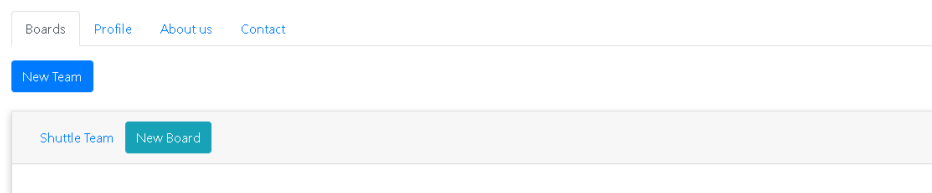
Figure 5: Add-Team



The screenshot shows a web interface with a top navigation bar containing links: Boards, Profile, About us, Contact, and Reports. Below this, there is a form titled 'Email'. Inside the form, there is a text input field containing the email address 'alem@gmail.com'. Below the input field is a blue button labeled 'Add Member'.

Figure 6: Add-Member

After a super user has created a team they can create boards inside that team by using the New Board button.



The screenshot shows a web interface with a top navigation bar containing links: Boards, Profile, About us, and Contact. Below this, there is a blue button labeled 'New Team'. Below the 'New Team' button is a light gray box containing the text 'Shuttle Team' and a green button labeled 'New Board'.

Figure 7: Added-Team

While superuser can create and delete teams normal user won't be able to make such operations. In their home screen they will only see the teams and board that

they are part of. Both normal users and super users will be able to be part of more than one team while each team can have more than one board.

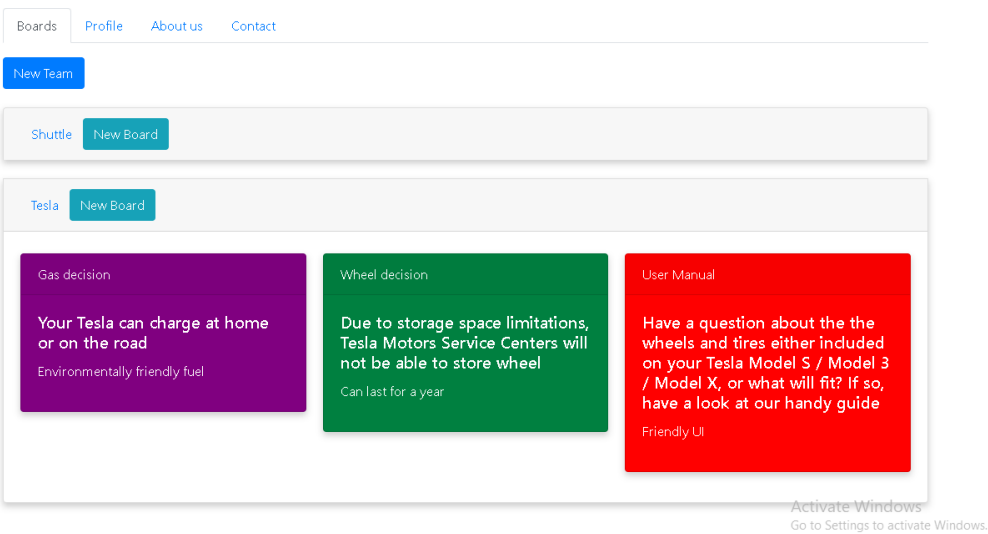


Figure 8: Create-Multiple-Teams

A super User can create a new board through the New Board button. This button displays the pop up shown below where the Super User has to fill all the necessary information regarding that board.

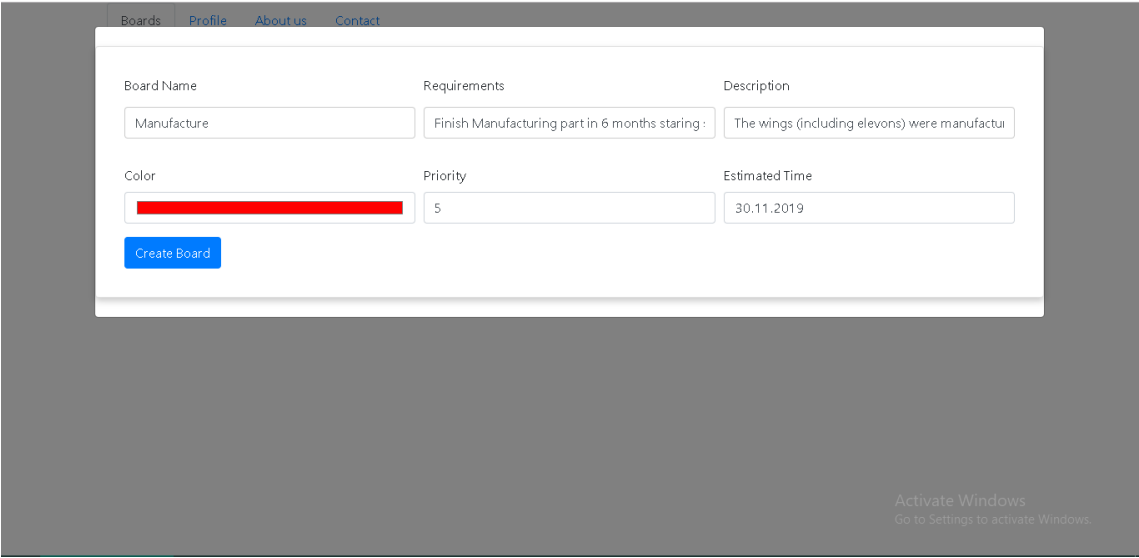


Figure 9: Create-Board

A superuser is able to create as many boards as they need.

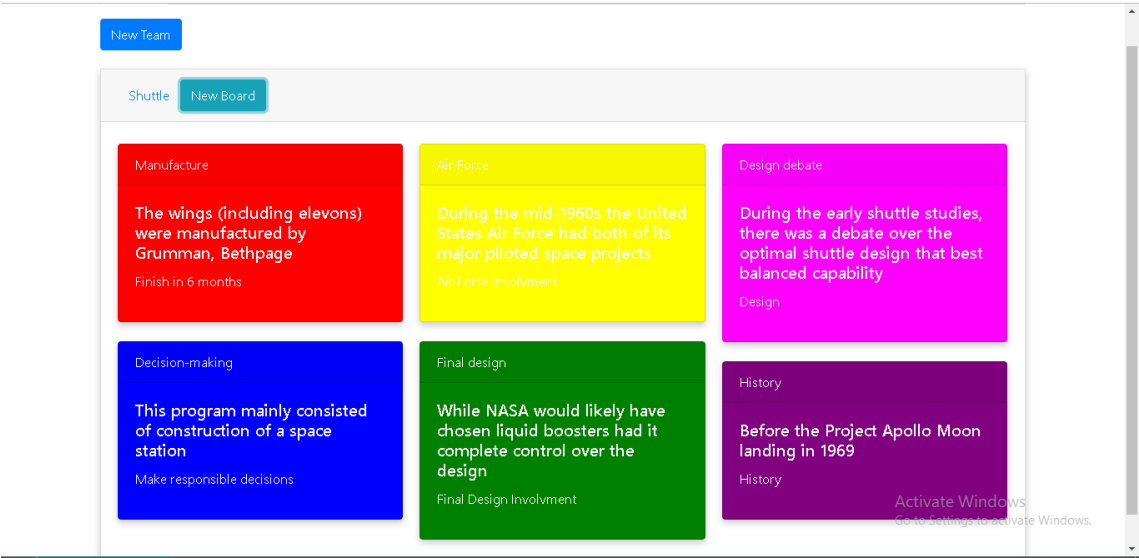


Figure 10: Multiple-Boards

### 6.3 Creating New List and Card Elements

When a Super User opens a board (by clicking on it) they will be able to create lists as part of that board. Each board can have more than one list.

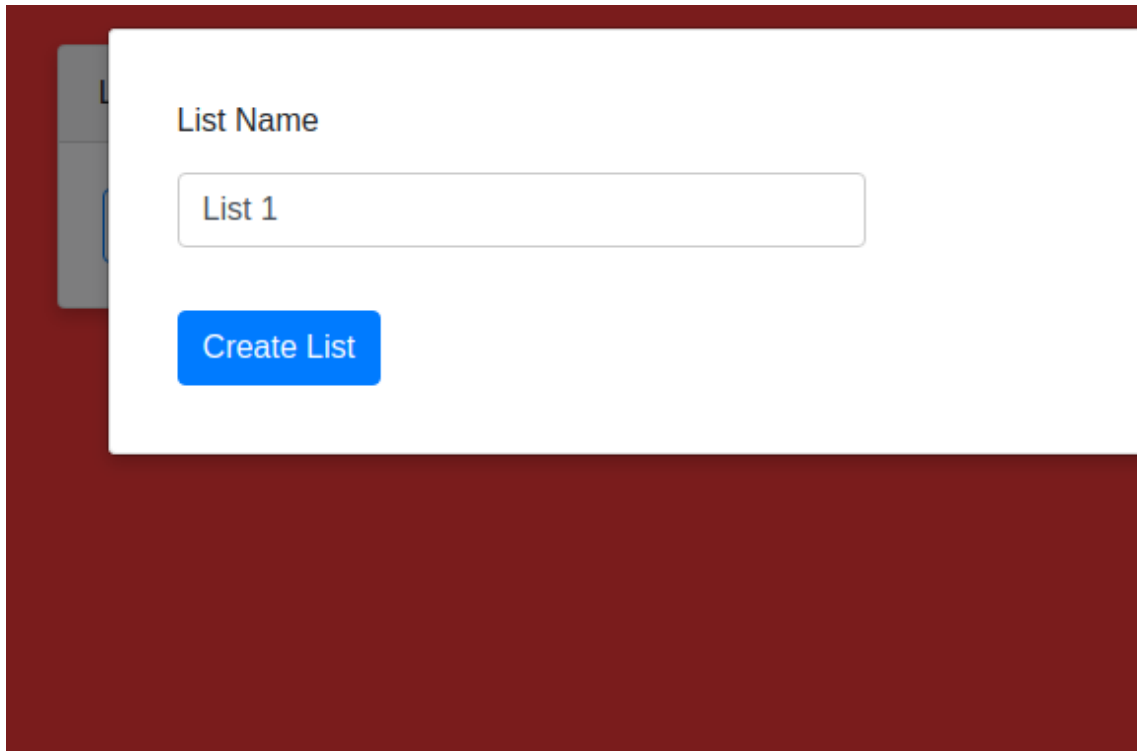


Figure 11: Create-List

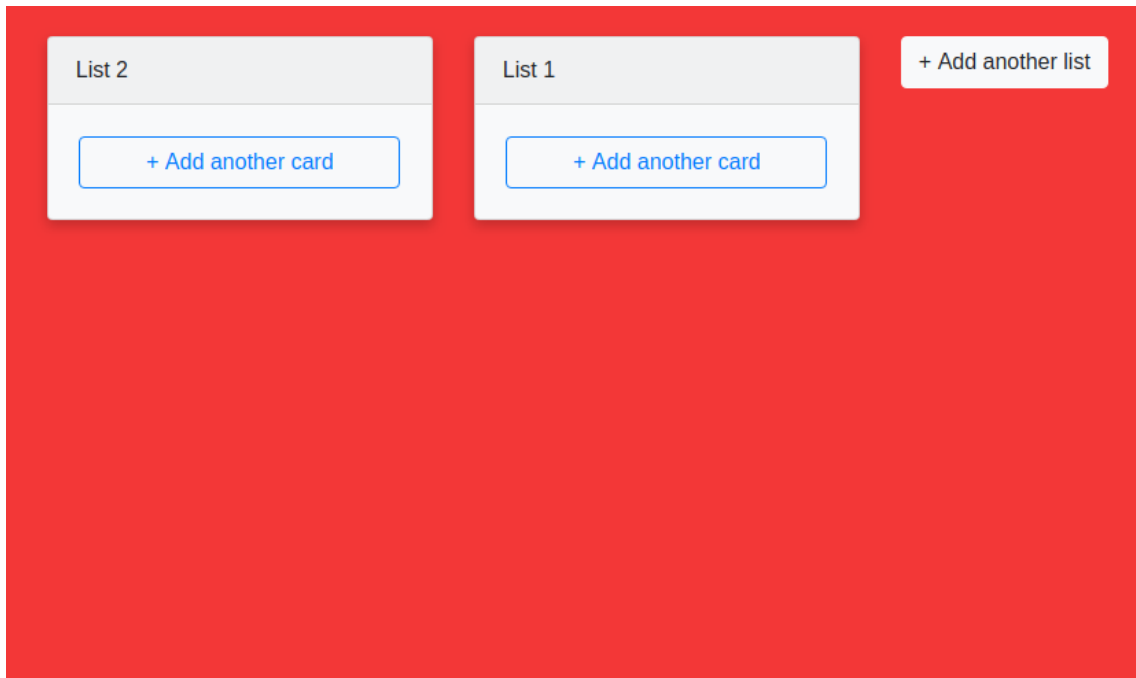


Figure 12: List-Created

After a list is created Super Users can create cards inside specific lists. They can do this by clicking the "Add another card" and filling up the pop up window generated by that button-click. After the Super User enters the necessary info they will be able to see the name of the card as part of the list. Normally each list can have more than one card. After the Super User enters the necessary info they will be able to see the name of the card as part of the list. Normally each list can have more than one card.



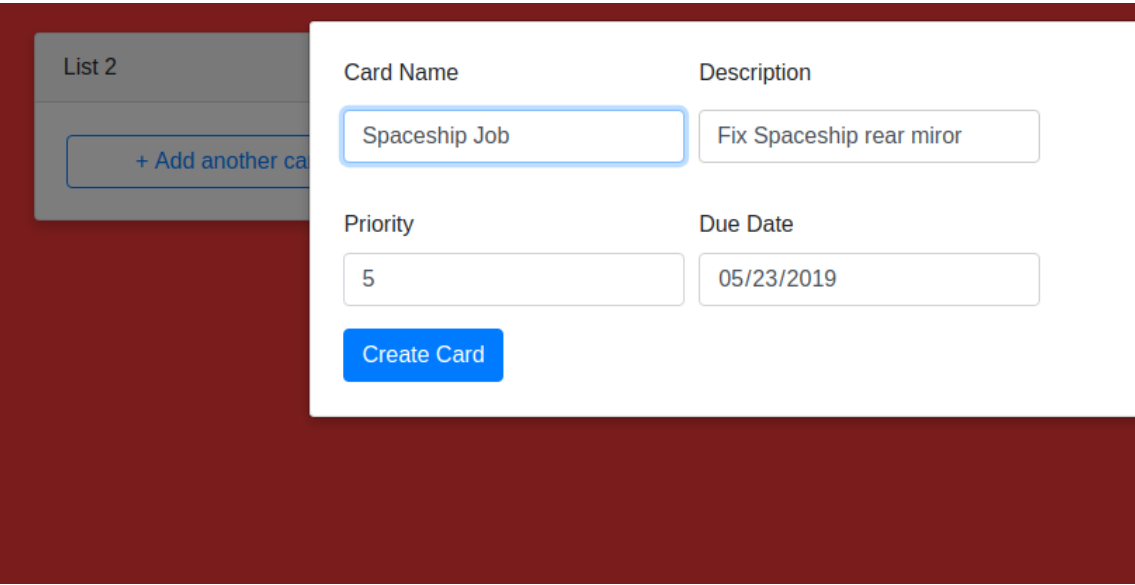


Figure 13: Create-Card

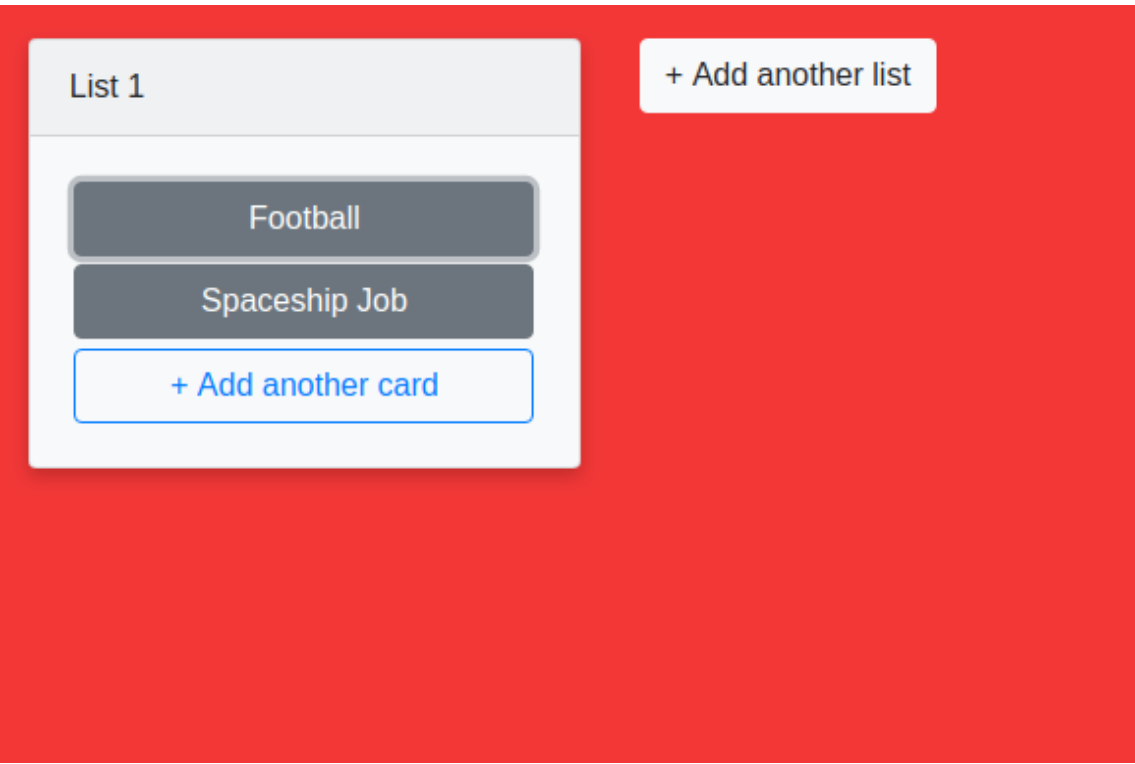


Figure 14: Cards-Created

## 6.4 Reports

In this section all users will be able to see some reports regarding the program in general. In the first table we show the team with the largest number of members, their supervisor and the count of members in that team. In the second table we show all the users that are part of all the projects owned by a specific user.

[Boards](#)

[Profile](#)

[About us](#)

[Contact](#)

Reports

This report shows the team with the largest number of members, along with its supervisor and the number of members:

Team Name	Team Supervisor	Number of Members
Team1	Naisila	2

This report shows users who are part of all projects supervised by a particular user:

User Name	User Email
Alem	alem@gmail.com

Figure 15: About-Us-Page

## 6.5 About Us page

Holds info about the developers of this project.

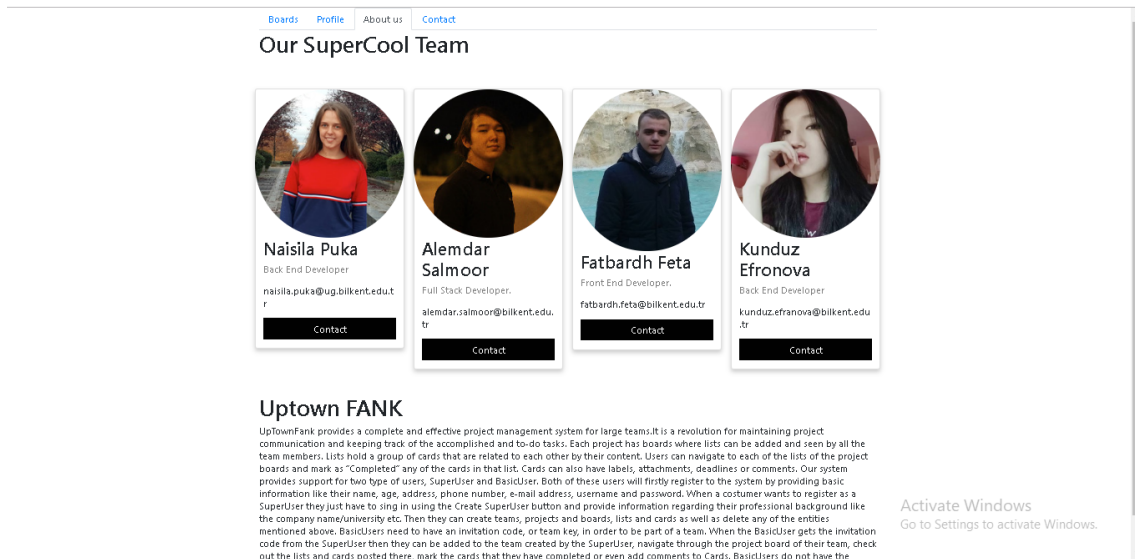


Figure 16: About-Us-Page

## 6.6 Contact page

Shows a contact form for users who want to get in touch with the developing team.

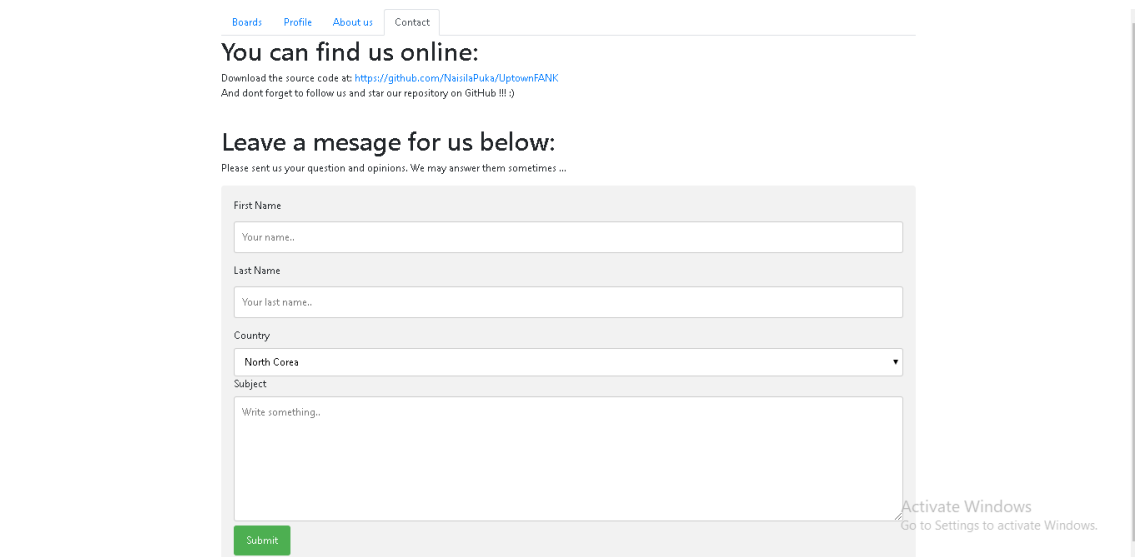


Figure 17: Contact-Us-Page