

Hero TD

Design Document



Team Members:

Mingyong Cai, Mingqi Han, Xihao Li, Nicholas Meier, Rongxi Zhang, Xuan Zhao

Copyright©2018 CS307 Team 4. All rights reserved.

Index

• Purpose	3
○ Brief description	
○ For User account	
○ For Main UI	
○ For User interface	
○ For Singleplayer	
○ For multiplayer	
• Design Outline	7
○ High level overview	
○ Flow of events	
• Design Issues	10
○ Functional issues	
○ Non-function issues	
• Design details	14
○ Database scheme mockup	
○ Database scheme description	
○ Class design diagram mockup	
○ Class design diagram description	
○ State/activity diagram	
○ State/activity diagram description	
○ Login UI mockup	


Purpose

Tower Defense games are those which present users with the ability to build a collection of towers or other structures so that they might protect an objective from enemies controlled by either AI, or other users. Like most other games, a tower defense game is simply played for fun and through the use of both increasing variety and strength of enemies, as well as new towers to aid the player through more complex levels, the game retains the interest of the user as time progresses.

The basic premise of tower defense is already enticing enough for some users, but different tower defense games offer different varieties of experiences. Much like different flavors of ice cream, different tower defense games offer different experiences that appeal to different users. Existing games such as Clash of Clans and Plants vs Zombies are both tower defense games, but offer completely different experiences and appeal to very different users. We seek to add Hero TD into the mix of existing tower defense games by providing in it, our own style and twist on the genre.

For User account

As a user:

- I would like to have my own account name and password in this game.
 - I would like my account name to be unique, which means others can't have the same account name as mine.
 - I would like to be able to, optionally, change my username.
- 

- I would like to be able to reset my password.
- I would like to be able to easily recover a lost password through email.
- I would like to be able have my scores and game stats stored in my account.
- I would like to have an account page to display my information and scores.
- I would like to link my account with Facebook.
- I would like to link my account with Twitter.

For Main UI

As a user:

- I would like to be able to navigate through the application without significant difficulties.
- I would like to see different game-related background picture in the menu page.
- I would like to be able to access Play page through the menu.
- I would like to be able to access Guide page through the menu.

For User Interface

As a user:

- I would like to see my current resource, life and level.
- I would like to navigate through a large map by moving the cursor to the edge of the map
- I would like to magnify a part of the map by the corresponding finger gesture.
- I would like to check the status of monsters by clicking on them.

- I would like to see a pop-up menu of tower that has all the operations I can do to it when clicking on the tower.
- I would like to access the shop anywhere in the game.
- I would like to be able to access the shop by shortcut key.
- I would like to check the user guide by shortcut key.
- I would like to check the user guide by click the icon

For Singleplayer mode

As a user:

- I would like to play a game in infinite mode in which there are endless waves of enemies to fight against for the sake of defending my nexus.
- I would like to play a story game-type in which there is dialogue from characters which presents a story related to the normal game's experience.
- I would like to have a tutorial or user's guide to learn how to play the game and the different unquities that come with it.

For Multiplayer mode

As a user:

- I would like to play this game with other people online.
- I would like to play this game with friends I have on social media, such as Facebook or Twitter.

- I would like to have a ranking system for online matches or competitive games so that I do not play against players that are excessively above or below my skill level.
- I would like to have a chat room that would let me communicate with other players.
- I would like to check the ranking list for all the players in the game.
- I would like to have a chatting channel to chat with my allies.
- I would like to have a chatting channel to chat with all players in the current game, including allies and opponents.
- I would like to have users punished for bad behaviour, for example insulting other players in chatroom.

Design Outline

High level overview

Our project is a online Tower Defence game that allows both single-player and multiplayer modes. Our implementation will use a Client-Server model. Our server will access the data stored in our database simultaneously when users send their requests. Our server will be able to access or store the data from/into database, accept multiple client requests, and give feedback to clients.



1. Client


- provides users interface to access the game.
- displays UI features where users can access different game modes, view ranking list, check the game guide or change game settings.
- shows the current available game room for online multiplayer mode.

2. Server

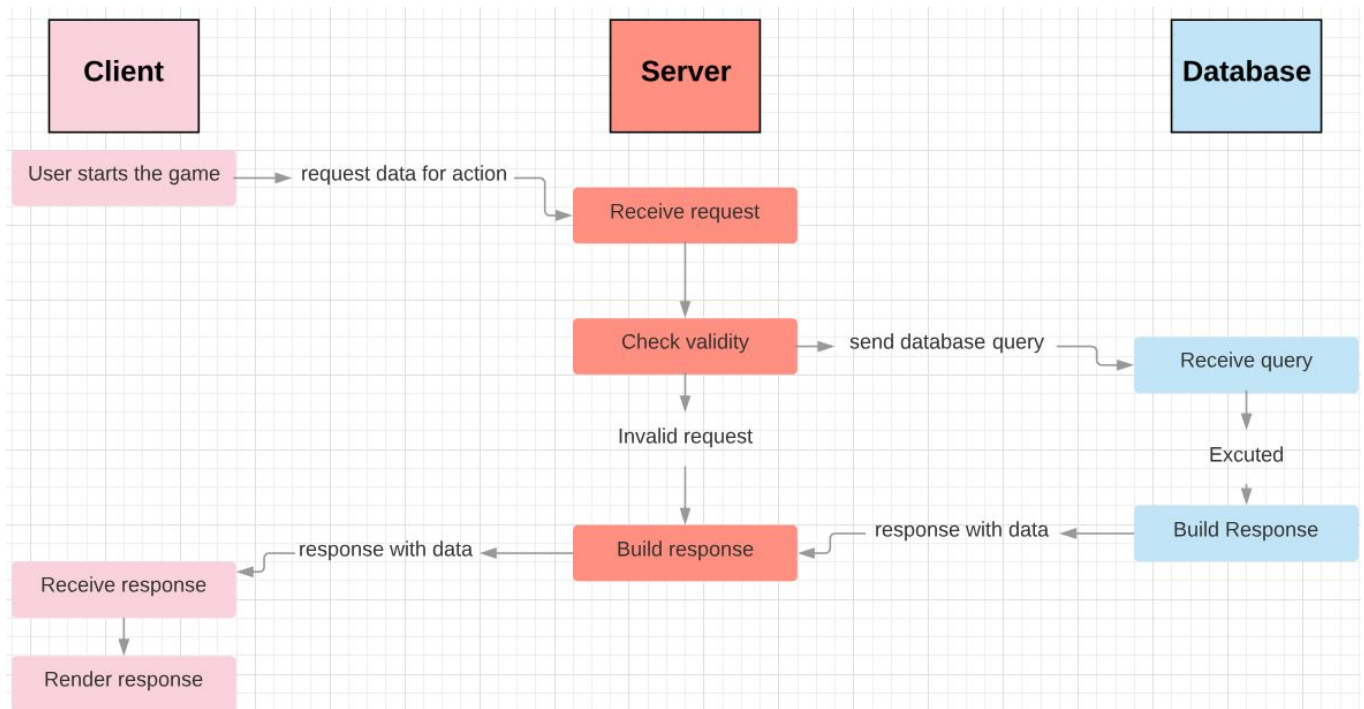
- receives and handles API requests from clients.
- checks the validity of every request and send queries to database.
- accesses and modifies the data stored in database based on the request received.
- responses the appropriate feedback to the targeted clients.

3. Database

- stores all the user information and game data.
- Typical data stored in the database will be
 - User Information

-
1. Username
 2. Password
 3. Email address
 4. Level
 5. Login_status
 6. Server_id
 7. Highest ranking
 8. Current ranking
 9. Penalty status
- ii. Game data
1. Tower HP/ATK
 2. Enemy HP/ATK
 3. Probability for lottery game
- 

Flow of events



Above is how the client, server, and database will interact with each other within one action from the user. As the user starts the game, each action done by the user will be a specific request generated by the client and this request will be sent to the server. The server will first check the validity of the request, and all the valid request will be sent to the database for access or modify the data stored in the database. Then database will build a response sent to the server, and the server will then send the response to the client.

Design Issues

Non-Functional Issues:

Design Issue #1 - What platform will we announce our TD game?

- ↳ **Option 1: Windows**

- ↳ Option 2: MacOS

- ↳ Option 3: IOS or Android

- Discussion: Only 2 of us have experience on developing with Unity and 4 out of our 6 team members are still in CS252, under which situation doesn't allow us to spend extra time on learning IOS and Android programming. Besides that, our team decided to only build a small server due to the cost. Therefore if we develop our game in IOS or Android we might need to face the problem of huge flux of users into our server. MacOS was not an option because the number of Windows users is way more than that of MacOS. Therefore Windows became our primary option for our TD game.

Design Issue #2 - How to deal with the email the user uses to login

- ↳ **Option 1: Every email address should be unique but not necessarily be valid.**

- ↳ Option 2: Every email address should be unique and should be valid.

- ↳ Option 3: Users should only login by their username not by email address

- Discussion: We will allow users to login by their email address, and each email address will be unique. But each email address doesn't need to be valid, for example 12345@111.com. Our server will check the validity for the email address, if it is valid, we will send them a confirmation email, and if it is not, it will just be a unique ID for login.

Design Issue #3 - How are we supposed to implement our underage policy when there are some violent or sexual scenes in our game

- ↳ **Option 1: Make this game a completely underages-friendly game**

↳ Option 2: Put a popup warning message when the users start their game

↳ Option 3: using a web crawler to get the user's age

➤ Discussion: Currently there are totally two types of Tower Defence game: realistic and fictional. Realistic Tower Defence games are more likely to attract players from different age levels, however since we might do a game based on a world where environment was destroyed by humanity and different monsters appeared so some of the scenes might be violent. Therefore we might just do a cartoonized game that had the same background but without those blood scenes. However option 2 is pretty much useless since users can just ignore that message, and option 3 doesn't work if the users don't log in with their Facebook accounts.

Functional Issues:

Design Issue #4 - How should Users login to the game

↳ Option 1: Users don't login when they are playing single player mode

↳ Option 2: Users login with a username that he/she created which will store in our database.

↳ Option 3: Users login with Facebook/Google account without creating his/her own username in our database

↳ **Option 4: Users login with Facebook/Google account while we create a unique username for he/she to login; If they don't choose to do so Users can create their own username and we will store them in our database**

➤ Discussion: We decide to go with Option 4 since we need to store the the points and money the players earn while they are playing even they are in the single-player mode so that those points could be used against their rivals when they are playing multi-players mode.

Design Issue #5 - How should we implement the multiplayer mode of the game

↳ Option 1: 4 users cooperate and fight against 4 streams overpowered enemies, they should cooperate with each other.

↳ **Option 2: 4 users fights for their own, they cannot attack other's tower, but they can make other's enemies overpowered.**

↳ Option3: 2 users fight against each other, they can spawn attackers at their own base and build towers near his base.

➤ Discussion: Since our game is a competitive multiplayer game, we will choose option 2. At the beginning, each stream of enemy is identical to each other, but one player can power up other player's enemies in order to distract that player. At the end of the game, the player who has the highest score or survive the longest time will win the game.

Design Issue #6 - How should we implement the map?

↳ Option 1: **Rectangular, where towers are built in assigned squares.**

↳ Option 2: Hexagon, where towers are built in assigned hexagons.

➤ Discussion: Since we choose to calculate the exact attack distance of the tower, the firing range should be a circle, and enemies inside the circle should be attacked. So choosing between using adjacent squares or hexagons as the attack range, we think a rectangular map is easier to implement.

Design Issue #7 - What time period should we implement in our game?

↳ Option 1: Ancient history or mythology based

↳ Option 2: Recent history based like WW1 or WW2

↳ **Option 3: Futuristic time period**

➤ Discussion: The time period will be around 2100 in the future. So we can add more fantastic features like weapons and enemy types.

Design Issue #8 - How should we host our backend services?

↳ Option1: we write our own server

↳ **Option2: we use AWS**

↳ Option3: we use Azure

↳ Option4: we use Google cloud.


➤ Discussion: We choose not to run our backend service on our own, since it would be too much work. And we do have access to AWS because github provide students this service, so we choose AWS

Design issue #9 - How many servers should we implement for our game

Option1: 1 server for both login/logout/checking stat functionality.

Option2: 2 servers, one dedicated to user's stats, the other dedicated to gaming related calculations.

➤ Discussion: For simplicity sake, we choose to build only one server, which both do the calculation for gaming and do the verification and status updates for users.



```

    erDiagram
        User ||--o{ Player : "has"
        Player ||--o{ Monster : "owns"
        Player ||--o{ Tower : "attacks"
        Player ||--o{ Room : "occupies"
        Monster ||--o{ Monster_type : "is of type"
        Tower ||--o{ Tower_Category : "is of category"
        Maps ||--o{ Tower : "has"
        Maps ||--o{ Room : "has"
  
```

The diagram illustrates the following entities and their attributes:

- User**: Userid, Playerid, Email, Password, Login_status(Bool), Ranking
- Player**: Playerid, Player_types, Roomid, Resources, Herotower_types
- Player_types(Enum)**: Attackers, Defenders, AI
- Monster**: Monsterid, Position, Owner, Monster_types, Activation_status(Bool)
- Monster_type**: MTid, Life, Designed_route, Skills, Speed
- Tower**: Towerid, Position, Tower_category, levels, Owner
- Tower_Category**: TCid, Life, Damage/s, Damage_Type(Bool), Build_fee
- Maps**: Mapsid, Effects, Activation_status
- Room**: Roomid, Playid, Server_Connection_Status, Room_Occupancy

Relationships are defined as follows:

- User** (1) to **Player** (many): 1:M relationship.
- Player** (1) to **Monster** (many): 1:M relationship.
- Player** (1) to **Tower** (many): 1:M relationship.
- Player** (1) to **Room** (many): 1:M relationship.
- Monster** (1) to **Monster_type** (many): 1:M relationship.
- Tower** (1) to **Tower_Category** (many): 1:M relationship.
- Maps** (1) to **Tower** (many): 1:M relationship.
- Maps** (1) to **Room** (many): 1:M relationship.

Above is our Database schema. Our database is built based on the understanding of the interactions between user and our server, and their relationship will be showed in our project. Since we are using C# in our project, the main tube connecting our frontend and backend will be the API provided by MySQL, Connector/Net. Including MySql.data.dll in Visual Studio, and connect to the database.

Tower

1. Tower is one of the basic unit that the server send back to user.
2. Includes the tower's id, position(in coordinate), activation status(if the owner has enough money to buy them), direction, owner, and tower category, levels.
3. allocated when the user send request through certain function.

Monster

1. Monster serves as player's unit.
2. Includes monster's position(in vector coordinate), activation status, moving status, destination, owner, and monster types, levels.
3. Allocated when the user send request through certain function.

Tower Category

1. Initialized at the begining of each game.
2. Includes the tower's life,,damage per second, attack range, attack preference, damage number(single or area), damage type(one time or long_lasting), model(.mb), build fee, Skills(only for hero tower).
3. Have different assigned attributes according to design.

Monster Types

1. initialized at the begining of the game.
2. Includes monster's life, position(in vector coordinates), speed, space(how much space it takes), designed routes, skills.
3. Moster attributes are designed to be unchangeble.

Server(room)

1. A room that was produced when the condition: enough number of players and enough types of players were gathered.

2. Includes room_id, player id, room_status, server_connection status.
3. rooms only open when player choose only pvp mode.

Room_status

1. an enumeration of if the room is busy, empty or full, busy means the server did not response.

User

1. User is the highest hierarchy that user can interact with.
2. Includes the user's information, like userid, playerid, password, email, level, login_status.
3. Created when a user signs up through API provided by Google or Facebook.

Player

1. Player is tightly connected to user, however, they are only created during one game and diminish after this game, while user is always exists, and keep record of all the process of it's player.
2. Includes player_types, resources, hero_tower types, rank level, room_id.

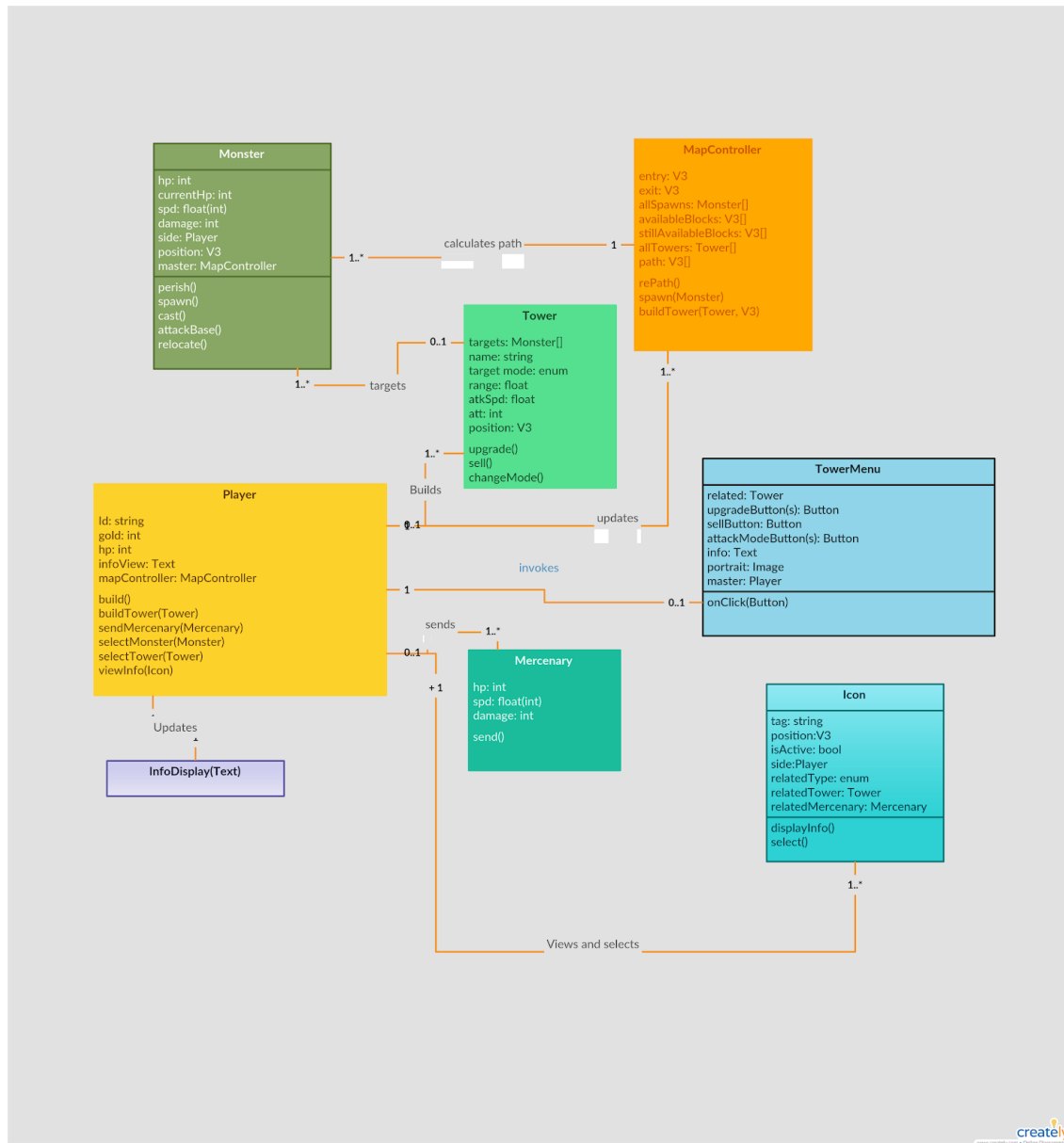
Player_types

1. an enumeration of attackers, defenders, AI.

Maps

1. it's the data that need to be downloaded, depends on different game mode, different types of maps will be used.
2. the maps includes map_types, effects, activation_status.

Class Design Mockup



Description of Class Design Diagram

Player class

The Player class holds the important stats for the current player, like current gold. It reacts to some of the player's input such as clicking on an existing tower. It also processes the requests sent from 'Icon' class and invokes the tower menu and mercenary menu.

Icon class

Icon class represents the tower icons on the side bar where the player uses to choose the tower/mercenary they want to build/send. It activates the corresponding tower/mercenary when applicable. It also send the signals to the Player class when the icons are clicked for further processing.


Monster class

The Monster class represents the enemies on the path. It holds the stats for that monster, and is in charge of updating them and react to the commands sent by the MapController. It also invokes the effect when getting hit/dies/casts spell/hits the base.

Mercenary class & Tower class

The Mercenary class and Tower class represents the game objects player can build/summon. They function pretty much similar to the Monster class, with the Tower class having extra functions like `upgrade()`, `setTargetMode()`(`changeMode()`) and such to give more control for its behavior to the Player.

Tower class also detects the Monster within its range and fires whenever it has cooled down. Mercenary however represents the Monsters the Player sends to the other Player to distract his game play. They will appear to be a Monster in the other Player's MainGame.

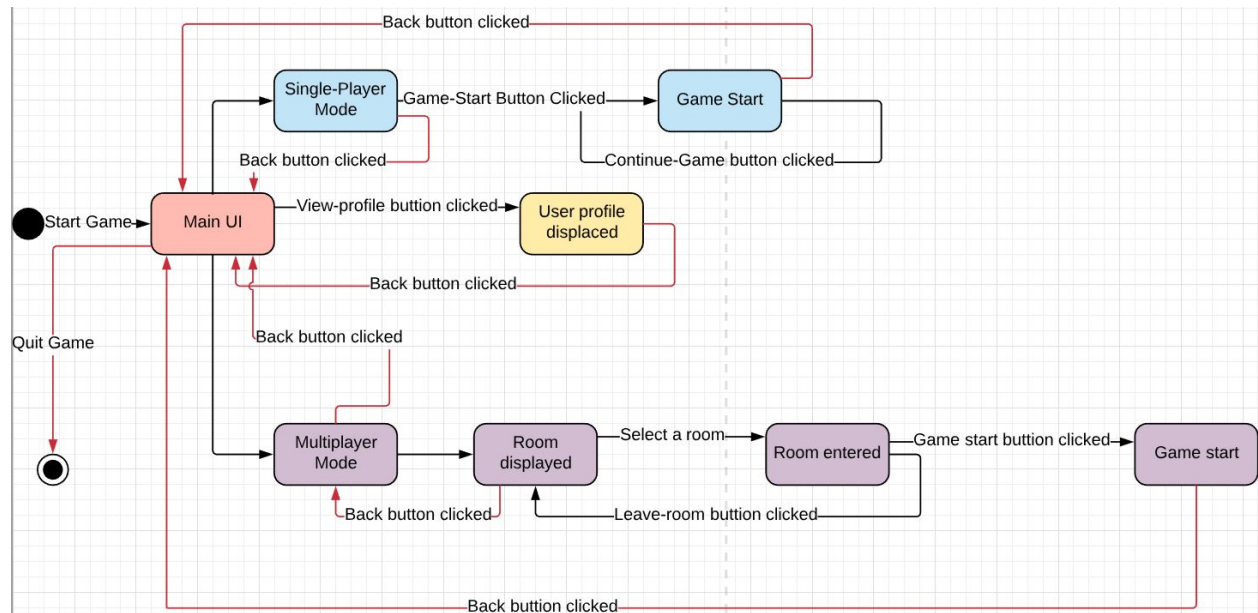


MapController class

MapController updates the map each time a change is made (like a tower is built or a new monster is summoned). It stores the spots in the map that are available to build a tower/pass a monster and directs all the monsters along the path to the Player's base.

The Infodisplay class represents an info bar that shows the player the stats of game objects on the scene. It gets updated by Player class whenever a new game object/blank spot is selected.

State/Activity Diagram



Above is the state/activity diagram for our application. When user starts the game, the main UI will be displayed, and the user can access to single-player mode, multiplayer mode, or user profile page by clicking the corresponding button. After entering one of those interface, user can go back to the main UI by clicking the back button. In the single-player mode, after clicking the game-start button, the battle will begin, and at the end of each battle, user can either continue the game or back to the main UI by clicking the corresponding button. After accessing the multiplayer mode, the game rooms will be displayed, the user can select one of the room to start their game. After entering a room, the game will start after all the players in the room clicked the game-start button. Furthermore, the user can leave the room after entering the room.

Login UI Design Mockup

Hero TD



Username:

Password:

Login

register

reset

Do not have an account? Try login with you social account.





Copyright©2018 CS307 Team 4. All rights reserved.