SC2006 Software Engineering Project

Basketball Tinder: Make Homosexual Dating Smarter!!

Why are you gay.
Who says I am gay?
You are Gay.
Join us @Basketball_Tinder to meet more gay people!

PREPARED FOR

NTU Prof

PREPARED BY

LBGT ppl

Product Description

Purpose

___app/website name____ is a website designed to create a seamless experience for basketball enthusiasts, enabling them to find and participate in games that fit their preferences while also fostering a community of like-minded sports enthusiasts. The platform is designed to be user-friendly, making it easy for users to connect, organise, and enjoy basketball games together.

Effectively, ___app/website name__ serves two main purposes. They are as follows:

1) **Connecting Basketball Players by Location and Time:**
   The primary purpose is to allow users to find or create basketball events based on their preferred location and timing. By using our website, users can quickly discover nearby basketball games that fit their schedule, whether they want to join an existing game or start one of their own.

2) **Providing Weather Forecasts for Better Planning:**
   The second purpose of _____ is to help users plan their basketball sessions more effectively by providing regional weather forecasts for

their chosen time range. This feature ensures that users can avoid disruptions due to weather and make the most of their basketball experience.

Scope

The scope of the app includes the following key functionalities:

1) Event Search:

- Users can search for basketball events in their area by selecting a location, date and a specific time range.
- The website will then display a list of basketball events created by other users which match the user's criteria, allowing them to view details such as the location of event, date, start and end times, and the number of participants required.
- Users can access further details about each event by clicking on it and it will show the picture of the location, additional details from the event organiser.
- Users can then click to join the event if it is suitable.

2) Event Creation:

- Users can create their own basketball events by specifying details such as the location, date, start time, end time, maximum number of participants needed , and any additional information.
- This feature allows users to organise basketball games that fit their schedule and location preferences, inviting others in the community to join.

3) Additional Features:

- The app provides regional weather forecasts for the selected time and location of the event, helping users plan their games effectively and avoid disruptions due to adverse weather conditions

---

Users and Stakeholders

Assumptions and Constraints

Constraints

Initial UI Mock-Up

Functional Requirements

1. Main Menu

1.1. The system must allow users to create an account.

1.2. The system must allow users to login to their account.

1.3. Each user must have a profile page

1.4. The system must allow users to use the Create Event function.

1.5. The system must allow users to use the Search Event function.

1.6. The user must be able to select and retrieve more details of a particular event including:

- 1.6.1. Location
- 1.6.2. Date
- 1.6.3. Start Time
- 1.6.4. End Time
- 1.6.5. Pax required
- 1.6.6. Picture of the location
- 1.6.7. Location on embedded map
- 1.6.8. Additional information from the event creator

2. Registration

2.2.1. The user must be able to register for a new account with information which includes:

- 2.2.1.1. Name
- 2.2.1.2. Username
- 2.2.1.3. Email Address
- 2.2.1.4. Password

2.2.2. The system must validate user inputs with the following criterias:

- 2.2.2.1. All input fields are filled up
- 2.2.2.2. Username does not exist in database

2.2.2.3. Email does not not exist in database

2.2.2.4. Password meets the security requirements in Section 3.1.1

2.2.3. If any of the above input validation fails, the system must display an error message for each validation failure.

3. Login

3.1. The user must input their username and password.

3.2. The system must validate inputs with the following criterias:

3.2.1. All input fields are filled up.

3.2.2. The username and password matches the database record.

3.3. If validation fails, the system must display an error message for each validation failure listed in Section 2.3.2.

3.4. Upon successful login, the system must redirect the user to the application's main menu.

3.5. The user must be able to request a change of password should he forget his password.

3.6. The system must send email verification code upon the user's request for change of password

4. Profile

4.1. Each user must have a profile page that displays the following:

2.4.1.1. Profile Picture

2.4.1.2. Name

2.4.1.3. Username

2.4.1.4. Short description about the user.

4.2. Each user must be able to edit the following components of their profile page:

2.4.2.1. Profile Picture

2.4.2.2. Short Description

4.3. Each user must be able to view other users' profile pages.

5. Event Creation

5.1. The system must prompt the event creator to enter the following information:

2.5.1.1. Title

2.5.1.2. Current Location or User-defined Location

2.5.1.3. Date

1. Limited to range from current date to two weeks ahead

2.5.1.4. Start Time

1. Minimally >2 hours from current time

2.5.1.5. End Time

2.5.1.6. Maximum number of participants

2.5.1.7. Additional text information and instructions (Optional)

5.2. The system must display a list or a map with pin-pointed locations of the basketball courts within a 5km radius of the chosen location

5.2.1. The user must be able to select a basketball court as the event location

6. Event View

6.1. The event view displays the following information about the event:

6.1.1. Title

6.1.2. Basketball Court Location

6.1.3. Date

6.1.4. Start Time

6.1.5. End Time

6.1.6. Current Number of Participants

6.1.7. Maximum Number of Participants

6.1.8. Additional text information and instructions (if provided by creator)
6.1.9. Link to Profile Page of Event Creator
6.1.10. Weather information from NEA Weather API:
6.1.10.1. Weather Forecast Data
6.1.10.2. Ultraviolet Index
6.1.10.3. Regional Weather (24H)
6.1.10.4. Regional Weather (2H)

6.2. The user must be able to join an event if the event is not full
6.3. The user must be able to withdraw participation in an event
6.4. The system should update the Current Number of Participants when a user joins or withdraws from an event

7. Event Search

2.6.1. The system must display events sorted by newest to oldest by default
2.6.2. The system must allow the user to enter the following filter inputs:
2.6.2.1. Current Location or User-defined Location
2.6.2.2. Date of event
2.6.2.3. Time
2.6.3. The system must filter and return a list of events created by other users based on the following criteria:
2.6.2.1. Location (Within a 5km radius of the chosen location)
2.6.2.2. Date
2.6.2.3. Time (As long as the start time that user selected is within this range)

2.6.2.4. Participation count (If event is full, then it will not appear in the list)

2.6.4. The user must be able to select an event to view the event information in the event view

8. Interface with other systems

8.1. The following APIs are used:

8.1.1. NEA Weather API from Data.gov.sg for weather forecast and ultraviolet index data

8.1.2. Google Maps API for map data

8.1.3. Google Geolocation API to retrieve user's current location

9. Format for information to be processed

9.1. Location

9.1.1. Distance should be stored with "m" as the base unit

9.1.2. Distance will be displayed in "km" and two decimal places of accuracy (e.g. 2.05km) if the distance is more than or equal to 1000m.

9.1.3. Distance will be displayed in "m" if the distance is less than 1000m

9.2. Date and Time

9.2.1. Date and Time must be jointly stored as a javascript Date object according to the ISO 8601 Datetime format

Non-Functional Requirements

1. Security

1.1. User account passwords must satisfy the following criteria:

1.1.1. Contains 12 characters or more

1.1.2. Contains at least one uppercase alphabet character, one lowercase alphabet character, one special character and one numeric character

1.2. Passwords should be hashed with the Bcrypt hash function before being stored in the database

1.3. Log-in validation should be done on both server-side and client-side

1.4. Log-in inputs should be sanitised to prevent Cross-Site Scripting Attacks

2. Maintainability

2.1. An MVC design pattern should be implemented for clear division of responsibility, keeping code organised and easier to maintain

2.2. PostgreSQL (or any other decent SQL based database) will be used to ensure easy data migration to other SQL based databases in the future if needed

3. Usability

3.1. Complies with WCAG A acessibility standard

3.2. Responsive Web Design should be implemented such that the website scales to these classes of devices:

3.2.1. Computer screens and monitors

3.2.2. Tablets

3.2.3. Mobile Phones

4. Use Case Model

3.1 Use Case Diagram

3.2 Use Case Description

Use Case ID:	1		
User Case Name:			
Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:	User		
Description:	User creates a new event with specific details, such as title, location, date, time, and maximum participants		

Pre-Conditions:	● User is loginned ● User has not created a event ● User location can be accessed or user can input manually
Post-Conditions:	● New event is created ● New event can be seen by other users on main page
Priority:	High
Frequency of Use:	Frequently
Flow of Events:	1. User logs into system 2. User chooses to create new event 3. System prompts user to enter event details a. Location (Live or user-inputted) b. Date (Restricted from current to 2 weeks ahead) c. Start and End Time (Start must be at least 2 hours ahead) d. Preferred number of participants e. Short description (optional) 4. System displays list of basketball courts within 5 km of target location 5. User reviews whether information is correct and confirms if it is, creating a new event
Alternative Flows:	● User's live location cannot be found, prompts user to manually input location ● No basketball courts found, prompts user to enter new location
Exception:	
Includes:	● Login System
Special Requirements:	● Real time location tracking ● List of basketball courts in Singapore
Assumptions:	● All basketball courts can be found on Maps API
Notes and Issues:	● Must be responsive enough for user, i.e. creating a event

	should take a minute at most. ● Can be scaled to allow users to recreate old events they made

Use Case ID:	2		
User Case Name:			
Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:	User		
Description:	Login System for User		
Pre-Conditions:	● User does not have existing account ● User is on registration page		
Post-Conditions:	● New user account is created and stored in database ● User is now able to login to system		
Priority:	High		
Frequency of Use:	Low, one time-use per user		
Flow of Events:	1. User clicks on register new account on websites registration page 2. System prompts user to input the following fields a. First Name b. Last Name c. Username d. Password (Contains 12 characters or more and contains at least one uppercase alphabet character, one lowercase alphabet character, one special character and one numeric character) 3. System validates the inputs a. Ensure username does not exist in database		

	b. Ensure email does not exist in database c. All fields are filled d. Password satisfies security criteria 4. If validations passed, new account is created and data is stored into the database
Alternative Flows:	1. Input validation failed due to not satisfying any of the criterions 2. System prompts user to fix wrong field a. If password does not meet requirements, prompt user to change b. If username exists, prompt user to change c. If email exist, prompt user to change d. If field empty, prompts user to fill up
Exception:	
Includes:	
Special Requirements:	● Database containing list of current users information exists
Assumptions:	● Users must have a valid email address
Notes and Issues:	● Sanitised inputs to prevent database injections ● System prompts to users following failed inputs are clear

Use Case ID:	3		
User Case Name:			
Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:	User		

Description:	Login to System
Pre-Conditions:	● User has an existing account ● User is currently logged out
Post-Conditions:	● User successfully login and is redirected to the main menu
Priority:	High
Frequency of Use:	Frequent

Flow of Events:	1. User navigates to login page and clicks button to login
	2. System prompts user to fill in fields
	  a. Username/email
	  b. Password
	3. Input is validated
	  a. Check if Username/Email exists in database
	  b. Check if password matches Username/Email
	  c. Check all fields filled
	4. If all validated, user is loginned and redirected to main menu

Alternative Flows:	● Input validation failed
	  a. System prompts user to fill in wrong/required fields
	  b. User fixes input and submits again to login
	● Password Forgotten
	  a. User can click on a button to reset password
	  b. Leads to profile management use case

Exception:	Profile Management
Includes:	
Special Requirements:	● Passwords in database are encrypted ● Login limit to prevent brute force
Assumptions:	● User has access to either username or email still
Notes and Issues:	● Can use cookies to keep user signed in to reduce frequency of this use case

Use Case ID:	4		
User Case Name:			
Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:	User		
Description:	Profile Management		
Pre-Conditions:	● User is loginned ● User has valid profile		
Post-Conditions:	● User profile is changed to adapt to their new inputs		
Priority:	Low		
Frequency of Use:	Low		
Flow of Events:	1. User clicks on change profile in their Profile Page 2. User has the option to change the following a. Profile picture b. Description 3. System prompts user on what to edit in profile a. If picture, prompts user to take photo or select new photo from gallery b. If description, allow user to enter a new short paragraph that describes themselves 4. User confirms changes 5. New profile reflects changes made by user		
Alternative Flows:	1. User decides to not change profile, cancelling the process and returning to profile page		
Exception:			
Includes:			

Special Requirements:	
Assumptions:	
Notes and Issues:	1. May need character limit on profile description

Use Case ID:	5		
User Case Name:			
Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:	User		
Description:	Password Management		
Pre-Conditions:	● User has an existing account		
Post-Conditions:	● User password has changed to new password as indicated		
Priority:	Medium		
Frequency of Use:	Low		
Flow of Events:	1. User can click on forgot password on login page or change password within their profile 2. Email is sent to their email as indicated in the database to validate whether they actually want to change password 3. If email is validated, user is able to change password as liked a. Password cannot be same as previous password b. Must follow same security requirements 4. Password is changed to the new password		

Alternative Flows:	1. If new password is invalid, system prompts user to enter a valid password
Exception:	
Includes:	
Special Requirements:	1. Clear old password from database after changing 2. Need send email using external API
Assumptions:	1. User still has access to their email
Notes and Issues:	

Use Case ID:	6		
User Case Name:			
Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:	User		
Description:	Allow user to view and join ongoing events		
Pre-Conditions:	● User is logged in ● Event is viewable within the system		
Post-Conditions:	● User can join ongoing events ● Amount of participants in event is updated in real time		
Priority:	High		
Frequency of Use:	High		

Flow of Events:	1. User navigates to event view page 2. Following information is displayed for each event a. Location b. Date c. Time d. Participants e. Text information (if exists) f. Link to profile of creator g. Predicted weather (from weather API) 3. If event is not full, user can click to join the event 4. Confirmation is sent to user to confirm participation 5. Event information is updated to show participation 6. If enrolled in event, user can choose to withdraw 7. Confirmation is sent to user to confirm withdrawal 8. Event information is updated to show withdrawal
Alternative Flows:	1. Weather API cannot retrieve weather 2. Error showing unable to retrieve is shown in place of weather
Exception:	Event Search
Includes:	Login System
Special Requirements:	● Exclude events with full participation on the list ● Access to weather API
Assumptions:	● Weather API is accurate and updates in real time ● System can accommodate number of users
Notes and Issues:	

Use Case ID:	7
User Case Name:	

Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:	User		
Description:	Allows users to filter for events		
Pre-Conditions:	● User is loginned ● User has access to list of events ● Users location is accessible or manual location is entered		
Post-Conditions:	● Filtered list of events is showed to the user		
Priority:	Medium		
Frequency of Use:	High		

Flow of Events:	1. System displays list of events from newest to oldest by default
	2. User can click on a button to filter for specific events
	3. System prompts for
	  a. Location
	  b. Date (Current to 2 weeks after)
	  c. Time
	4. Location filters based on inputs
	  a. Within 5km of selected location
	  b. Within selected date
	  c. Within selected timeframe
	5. List of filtered events is displayed to the user		

Alternative Flows:	● User location cannot be found automatically
	  a. Prompts user to enter manual location
	● No courts found
	  a. Informs user of lack of events and give suggestions on other events possible		

Exception:			
Includes:	Login System		

Special Requirements:	1. System needs to return filtered list quickly within 30 seconds
Assumptions:	1. Map API contains all possible basketball courts
Notes and Issues:	

Use Case ID:	8		
User Case Name:			
Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:			
Description:			
Pre-Conditions:			
Post-Conditions:			
Priority:			
Frequency of Use:			
Flow of Events:			
Alternative Flows:			
Exception:			
Includes:			
Special Requirements:			

Assumptions:	
Notes and Issues:	

Use Case ID:	9		
User Case Name:			
Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:			
Description:			
Pre-Conditions:			
Post-Conditions:			
Priority:			
Frequency of Use:			
Flow of Events:			
Alternative Flows:			
Exception:			
Includes:			
Special Requirements:			
Assumptions:			

Notes and Issues:	

Use Case ID:	10		
User Case Name:			
Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:			
Description:			
Pre-Conditions:			
Post-Conditions:			
Priority:			
Frequency of Use:			
Flow of Events:			
Alternative Flows:			
Exception:			
Includes:			
Special Requirements:			
Assumptions:			
Notes and Issues:			

Use Case ID:	11		
User Case Name:			
Created By:		**Last Updated By:**	
Date Created:		**Date Last Updated:**	
Actor:			
Description:			
Pre-Conditions:			
Post-Conditions:			
Priority:			
Frequency of Use:			
Flow of Events:			
Alternative Flows:			
Exception:			
Includes:			
Special Requirements:			
Assumptions:			
Notes and Issues:			