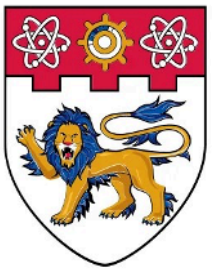


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

# SC2006 – Software Engineering

## Lab 4 Deliverables

| | |
|---|---|
| **Lab Group** | SCSC |
| **Team** | HobbyHive |
| **Members** | Teo Rong Xuan (U2421554F) |
| | Jain Divisha (U2423825L) |
| | Chua Jing Yi Jax (U2421642A) |
| | Ishita Dhananjaya (U2420909L) |
| | Palagiri Afreen Mahtaj (U2422632F) |
| | Swetha Sudhakar (U2420696G) |

# Table Of Contents

# 1. Black Box Testing

## I. AuthenticationController

Control class to test - **AuthenticationController**

### Functional Overview

The Authentication Controller class manages **user authentication** within the application, covering both **user registration (signup) and user login for the admin and general public.** Black box testing was conducted to verify that user credentials are validated correctly, appropriate messages are displayed for invalid inputs, and successful actions result in correct user creation or login responses.

When a new user registers for the application, the controller will ensure essential information is collected, validating the input dat,a ensuring correctness and completeness. Prior to storing the user's password in the database, the password will undergo a secure hashing process. The AuthenticationController will also generate a user ID and assign it to the newly registered user. This unique identifier becomes a key element in distinguishing and managing user data.

To authenticate the user, the controller retrieves the hashed password associated with the provided email from the database. The retrieved hashed password is then compared with the user's provided password. If the comparison results in a match, signifying successful authentication, the user will be directed to the home page. Information managed by the AuthenticationController in the login process includes email and password.

### Testing Technique

Testing was performed using **Equivalence Partitioning** and **Boundary Value Analysis**, targeting:

- Valid and invalid email formats
- Missing fields
- Password mismatch and length violations
- Duplicate registration attempts
- Correct and incorrect login credentials
- Existing and non-existing Users

## II. Equivalence Class and Boundary Value Testing

### Equivalence Class Testing

A basic black-box test design technique in which test cases are designed to execute representatives from equivalence partitions. Equivalence classes (partitions) are portions of an input or output domain. The behaviour of a component or system is assumed to be the same for every member of a partition class. Test cases are designed to cover each partition at least once. The operation of equivalence partitioning is performed by splitting a set (domain) into two or more disjoint sets where all the members of each subset share some trait in common i.e.

- Produces the same output
- If one catches a bug, the others probably will too
- If one value does not catch a bug, the others probably will not either

This trait is not shared with the members of other subsets.
Valid equivalence classes describe valid situations, and the system should handle them normally.
Invalid equivalence classes describe invalid situations and the system should reject them.

### Boundary Value Testing

Boundary Value Testing is an extension of equivalence partitioning and applies only when the members of an equivalence class are ordered. It checks how the system behaves at the edges or limits of acceptable input values.

1. **Login Function**

**Valid Equivalence Class:** Usernames and passwords with correct formats.
**Invalid equivalence Class:** Usernames and passwords with empty, incorrect formats or missing information.

2. **Sign-up function**

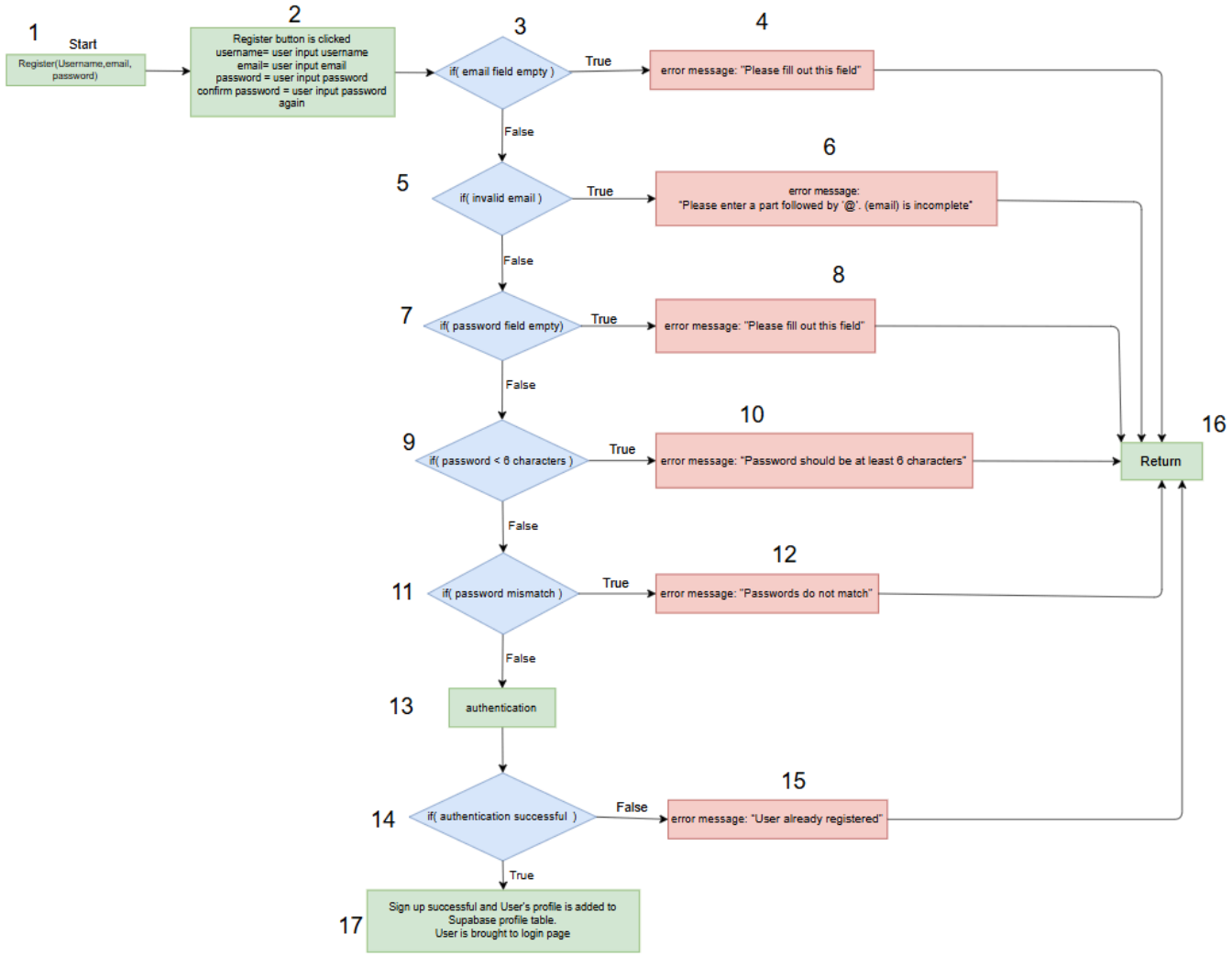
**Valid Equivalence Class:** Username, Email, Password and Confirm Password with correct formats.
**Invalid Equivalence Class:** Username, Email, Password and Confirm Password with incorrect formats or missing information.

# 2. White Box Testing

## I. Register/SignUp

### I.I Control Flow Graph

*If the image is unclear, please refer to the raw png file that is uploaded together with this document.*




### I.II Basic Path Testing

**CyclomaticComplexity**=|decisionpoints|+1=6+1=7

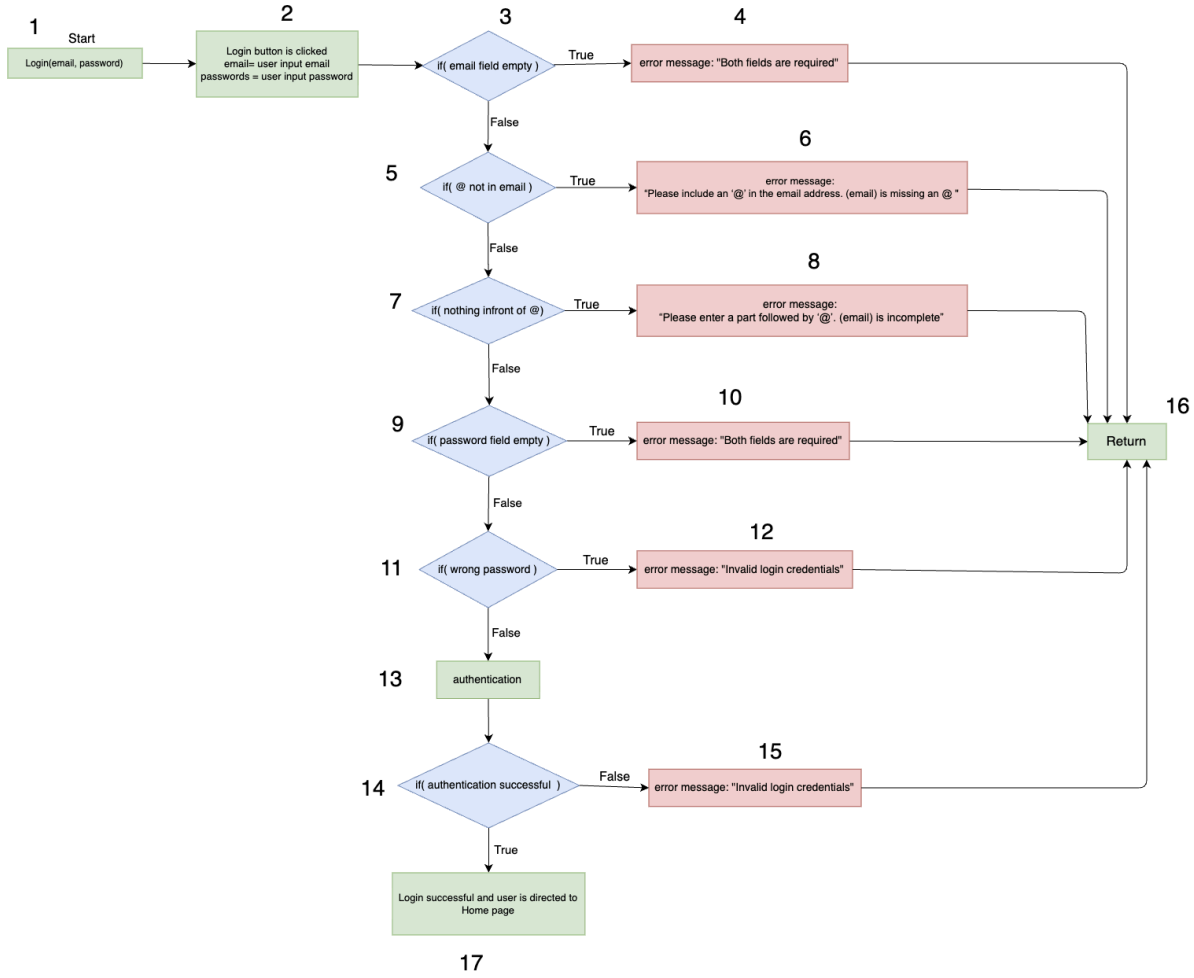
**BasisPaths**

Baselinepath: 1,2,3,5,7,9,11,13,14,17
Basispath 2: 1,2,3,4,16
Basispath 3: 1,2,3,5,6,16
Basispath 4: 1,2,3,5,7,8,16
Basispath 5: 1,2,3,5,7,9,10,16
Basispath 6: 1,2,3,5,7,9,11,12,16
Basispath 7: 1,2,3,5,7,9,11,13,14,15,16

## II. Login

### II.I Control Flow Graph

*If the image is unclear, please refer to the raw png file that is uploaded together with this document.*




### II.II Basic Path Testing

**CyclomaticComplexity**=|decisionpoints|+1=6+1=7

**BasisPaths**

Baselinepath: 1,2,3,5,7,9,11,13,14,17
Basispath 2: 1,2,3,4,16
Basispath 3: 1,2,3,5,6,16
Basispath 4: 1,2,3,5,7,8,16
Basispath 5: 1,2,3,5,7,9,10,16
Basispath 6: 1,2,3,5,7,9,11,12,16
Basispath 7: 1,2,3,5,7,9,11,13,14,15,16

## 3. Test Cases and Results

### I. Register/Sign Up

Red: Invalid input

| Test case ID (REGISTER) | Description | Input | Oracle | Log | Pass? |
|---|---|---|---|---|---|
| Test case 1 | Valid signup | Username = “johnnytest”<br>Email = “user1@gmail.com”<br>password= “johnboy”<br>Confirm password= “johnboy” | User successfully created; profile inserted in Supabase profiles table | User successfully created; profile row verified in profiles | Yes |
| Test case 2 | Invalid email | Username = “johnboy225”<br>Email = “@gmail.com”<br>password= “hello225”<br>Confirm password= “hello225” | “Please enter a part followed by ‘@’.’@gmail.com’ is incomplete”<br>(unable to signup) | “Please enter a part followed by ‘@’.’@gmail.com’ is incomplete”<br>(unable to signup) | Yes |
| Test case 3 | Missing email | Username = “johnboy225”<br>Email = “”<br>password= “hello225”<br>Confirm password= “hello225” | “Please fill out this field.”<br>(unable to signup) | “Please fill out this field.”<br>(unable to signup) | Yes |
| Test case 4 | Missing password | Username = “johnboy225”<br>Email = “johnboy@gmail.com”<br>password= “”<br>Confirm password= “” | “Please fill out this field.”<br>(unable to signup) | “Please fill out this field.”<br>(unable to signup) | Yes |
| Test case 5 | Duplicate signup | Username = “johnnytest”<br>Email = “user1@gmail.com”<br>password= “johnboy”<br>Confirm password= “johnboy” | “User already registered”<br>(unable to signup) | “User already registered”<br>(unable to signup) | Yes |
| Test case 6 | Password Mismatch | Username = “johnboy225”<br>Email = “johnboy@gmail.com”<br>password= “hello225”<br>Confirm password= “hello224” | “Passwords do not match”<br>(unable to signup) | “Passwords do not match”<br>(unable to signup) | Yes |
| Test case 7 | Invalid password | Username = “johnboy225”<br>Email = “johnboy@gmail.com”<br>password= “alice”<br>Confirm password= “alice” | “Password should be at least 6 characters”<br>(unable to signup) | “Password should be at least 6 characters”<br>(unable to signup) | Yes |

## II. Login

Red: Invalid input

| Test case ID (LOGIN) | Description | Input | Oracle | Log | Pass? |
|---|---|---|---|---|---|
| Test case 1 | Valid login | Email = “user1@gmail.com” password= “johnboy” | Successful login; Supabase returns user object | Successful login; user object returned | Yes |
| Test case 2 | Invalid email (nothing in front of @) | Email = “@gmail.com” password= “johnboy” | “Please enter a part followed by ‘@’.’@gmail.com’ is incomplete” (unable to login) | “Please enter a part followed by ‘@’.’@gmail.com’ is incomplete” (unable to login) | Yes |
| Test case 3 | Invalid email (without @) | Email = “user” password= “johnboy” | “Please include an ‘@’ in the email address. ‘user1 is missing an ‘@’” (unable to login) | “Please include an ‘@’ in the email address. ‘user1 is missing an ‘@’” (unable to login) | Yes |
| Test case 4 | Missing email | Email = “” password= “johnboy” | “Both fields are required.” (unable to login) | “Both fields are required.” (unable to login) | Yes |
| Test case 5 | Missing password | Email = “user1@gmail.com” password= “” | “Both fields are required.” (unable to login) | “Both fields are required.” (unable to login) | Yes |
| Test case 6 | Wrong password | Email = “user1@gmail.com” password= “123456” | “Invalid login credentials” (unable to login) | “Invalid login credentials” (unable to login) | Yes |
| Test case 7 | Unregistered email | Email = “user2@gmail.com” password= “hello225” | “Invalid login credentials” (unable to login) | “Invalid login credentials” (unable to login)) | Yes |

## 4. Demo Script

### Ishita

Good morning everyone, my name is Ishita, and together with my group members Jax, Divisha, Rongxuan, Afreen and Swetha, we'll be introducing to you our website, HobbyHive. (next slide) I'll start off by introducing the background and problem we're addressing, before we go through the live demo, software principles and future plans.(**next slide)**

In Singapore's fast-paced, work-focused culture, many young adults find themselves stuck in a loop. After long hours at the office or studying, it's hard to make time for hobbies, let alone find people to share them with. Furthermore, it's often difficult to know where to look. Many hobby groups are scattered across different platforms like Telegram or Reddit, making it inconvenient. As a result, they experience a poor work-life balance and a disconnection from their interests and community **(next slide)**

This is where HobbyHive comes in. HobbyHive is a community-based platform for young adults in Singapore to pursue their hobbies together. Users can join nearby hobby events or even host their own events for others in the neighbourhood to join. Our goal is to create a space where young adults can come together and strengthen the community spirit through hobbies they love.**(next slide)**

Now, let's take a look at who our expected users are. We designed HobbyHive with **three main demographics** in mind: young working professionals, university students, and young expats who are all looking for ways to unwind or connect through hobbies. As for our use-case diagram, users can adopt both roles as a host and a participant. The host can create events and manage who joins their event while the participant can search and join events on the website. And the admin can approve or flag events and users.
Next, Jax will go through the key features of our website

### Jax

Some key features will include allowing users to host events, big or small, and for each event created, there will be a built-in group chat for communication. Next, we can assist users in choosing suitable event locations. And finally, we can recommend events to users based on popularity and proximity.

**[next slide]**

Moving on to some of the APIs and datasets used. For oneMap, we used the onemap and nearest MRT API for its map rendering and address geocoding purposes. For data.gov datasets, we used CCs by PA, Libraries by NLB, and Parks by NPARKS. These datasets help facilitate location recommendations. And finally, for Supabase, we use the authentication, database, and storage APIs to manage our app.

**[next slide]**

Moving to the tech stack, for our frontend, we used Next.js for routing and API integration, Typescript for static typing and code reliability, Tailwind for styling, and DaisyUI for prebuilt components.
For our backend, we use Supabase, which is a relational cloud-based database with built-in features like rule-based access, JWT authentication, real-time websockets, and storage buckets. In addition, it is a serverless architecture allowing it to be scalable and maintainable. Finally, for hosting, we use Vercel as it is the official hosting platform.

**[next slide]**

Next, I will briefly go through one of our use cases, Create Event. Firstly, the host will need to key in the event details. Next, we pass it on to the controller to validate. Upon validation, the event is created and will be pending approval from the admin via the reviewEvent use case. Upon approval, the event will then be available for public viewing and joining. Next, I will pass my time off for the live demo.

## -LIVE DEMO-

### Divisha

Let's begin with the demo.

Before we begin, we will be using 2 personas to help us understand the working of the system better. We have Joy, 27 years old, who enjoys sports and will act as a host, and James, 25, who also enjoys sports and will act as a participant.

**Landing Page & Registration**
Starting with the landing page, this page has a bit about what we do as HobbyHive. It also has a search button for people to easily search for events they might be interested in, like baking for example.

But before we can join or learn more about any event, we will have to log in. Since Joy is a new user, she needs to register for an account, so let's set that up.

**[Click Register]**

We're going to provide details like username, email, and password.

**[Show email validation using "joy" as email]**

As you can see, if I just type "joy" without a proper email format, it shows an error, this validates that users enter proper email addresses.

**[Complete registration]**

Once registered, it takes us back to the landing page, but as you can see on the top right, we are now logged in as Joy.

**Navigation Overview**
Once you click on the hamburger menu, we can see various options like All Events, My Events, Create Event, and Group Chat.

**[Open All Events]**

These are all the various events available on the site, with details like the date and activity level visible at first glance. Since Joy hasn't joined any events, it doesn't show any recommended events just yet.

**[Open My Events and Group Chat]**

My Events and Group Chat are blank too as Joy hasn't joined any event yet.

**Creating an Event**
**[Open Create Event]**

Since Joy wants to create an event, let's open the Create Event page.

**[Explain blanks on form]**

Here we can fill in the event title, description, category, like sports, arts, or cooking, and then we have the date and time. Note that the date must be at least 2 days in advance. This gives time for admin review and allows participants to plan ahead.

We also set the maximum number of participants and the activity level, which helps people know what to expect, whether it's a casual meetup or something more intense.

**[Explain location section]**

Now for the location feature. We can search for any location in Singapore, and it'll show up on the interactive map. Let me demonstrate: if I search for "Chinese Garden," it pins the location here. If I change it to "Ang Mo Kio," it updates to that location. And if I search for "Chinese Garden" again, it goes back to the original spot.

The system also automatically calculates and displays the nearest MRT station, which makes it super convenient for participants to find their way.

**[Fill out form and create event]**

Let me complete the rest of the form with event details and an image, then create the event.

**Event Approval Workflow**
**[Go to My Events]**

Now if we go to My Events, you can see Joy's newly created event is here with a "Pending" status. This means it's waiting for admin approval.

**[Go to All Events]**

And if we check All Events, notice that Joy's event is not visible here yet. This is because all events need to be approved by an admin before they go live on the platform, this helps us maintain quality and safety.

**Admin Dashboard**
**[Logout and login as admin]**

Let me logout and login as an admin to show you the other side.

**[Navigate to Admin Control Panel]**

This is the Admin Control Panel. Here, admins can review and moderate all the events and users on the platform. You can see pending events that need approval, reported events, and user accounts that need review.

**[Open Review Events]**

Here's Joy's event in the pending queue. As an admin, I can review the details, check if it follows community guidelines, if the information is appropriate, and if it's a legitimate event.

**[Approve the event]**

Everything looks good, so I'm going to approve this event.

**Back to Joy's Account**
**[Logout and login as Joy]**

Now let's logout and login back as Joy to see what's changed.

**[Go to All Events]**

Look at that! The event is now live on All Events, and since Joy created a sports event and her profile indicates she's interested in sports, the system now shows it as a recommended event. Our recommendation algorithm learns from user preferences and activity.

**[Go to My Events]**

In My Events, you can see the status has changed from "Pending" to "Approved," and now Joy has full control over this event.

**[Go to Group Chat]**

And over here in Group Chat, a new group chat has been automatically created for this event. This allows Joy and participants to communicate before and after the event.

**[Show Edit and Cancel Events buttons in My Events]**

You'll also notice that Joy now has options to Edit Event or Cancel Event. If she needs to change the time, location, or any details, she can edit it. And if for some reason the event can't happen, she can cancel it, and all participants will be notified.

**Handoff to Participant View**
And that's the host experience! Now I'll hand it over to Rong Xuan, who will take us through the participant view and show you how James joins and interacts with events.

## Rong Xuan

**Participant Experience - James**
Now let me show you the participant experience. I'm going to login as James.

**[Login as James]**

Once logged in, we're back at the landing page. Let me click on the navbar and go to All Events.

**[Navigate to Events]**

Here we can see all the available events. Now, James is specifically interested in cycling, so let's use the search bar to find cycling events.

**[Type "cycling" in search bar]**

Great! Now we can see cycling-related events. But let's say James wants to be even more specific with his search. Let me show you the filter options.

**[Open filters]**

We can filter by category, I'll select Sports and Fitness, and just to show you how it works, I'll also add Baking.

For location, we have two options: we can either use our current location, or we can specify a location like "Ang Mo Kio." Let me set it to Ang Mo Kio.

Then we can set the range, I'll set this to greater than 10 kilometers to see events in a wider area.

And for the date, let's filter from 13th November to 30th November to see what's happening in the next couple of weeks.

**[Apply filters]**

And there we go, the events are now filtered based on James's preferences.

**[Click on an event]**

Let me click on this event to see more details.

**Event Details Page**
This is the event details page. Here you can see everything about the event, the full description, the exact location on the map with the nearest MRT station, the date and time, activity level, and how many spots are still available.

You can also see who's hosting the event, in this case, it's Joy. Let me click on her profile.

**[Click on host profile, then follow]**

This takes us to Joy's profile page where we can see her bio, the events she's hosting, and her followers. James is interested in the events she hosts, so let's follow her.

**[Click Follow button]**

Perfect! Now James is following Joy.

**[Navigate to James's own profile]**

Let me go to James's profile now. As you can see, it reflects that James is now following Joy, the count has updated.

**Edit Profile**
**[Click Edit Profile]**

James wants to update his profile, so let's click Edit Profile. Here he can change his username, bio, and add more information about his interests.

**[Upload profile photo]**

He can also upload a profile photo to personalize his account. Let me upload one now.

**[Save changes]**

And we're done! The profile is now updated with the new photo and information.

**Joining an Event**
**[Go back to All Events]**

Now let's go back to All Events and actually join an event.

**[Click on an event and join]**

James finds an event he's interested in, so let's click Join Event.

**[Confirmation appears]**

Great! James has successfully joined the event.

**Group Chat**
**[Navigate to Group Chat]**

Now if we go to Group Chat, you'll see something new has appeared. A group chat for the event James just joined is now available here.

**[Click on the group chat]**

Let's open it up. This is where all the participants of the event can communicate. James can ask questions, coordinate meetup details, or just chat with other participants.

**[Join another event]**

Let me have James join one more event to show you how it works with multiple events.

**[Go back to Events, join another event]**

Done! James is now part of two events.

**[Go back to Group Chat page]**

And if we check the Group Chat page again, both group chats are now showing up here. Each event gets its own dedicated chat room.

**Engaging in Conversation**
**[Open a group chat and type a message]**

Let me show you the conversation feature. James can type a message here, let's say "Hey everyone! Looking forward to the event!"

**[Send message]**

And there it is! The message appears in real-time. Other participants will see this and can respond, making it easy to build community before the event even happens.

**My Events**
**[Navigate to My Events]**

Now let's check My Events. Here you can see all the events James has joined. There are two tabs, one for events he's participating in, and one for events he's hosting. Since James is only a participant, all his events show up under the Participated tab.

**Leaving an Event / Being Removed**
**[Go back to Group Chat]**

Let's say something comes up and James can't make it to one of the events anymore. Or maybe the host needs to remove him for some reason. Let me demonstrate what happens.

**[Leave the event or simulate host removal]**

If James leaves the event, or if the host removes him, you'll notice the group chat for that event disappears from his Group Chat page.

**[Go back to My Events]**

And if we check My Events, that event is no longer showing up here either. Everything updates automatically to reflect his current participation status.

**Full Event - Cannot Join**
**[Go back to All Events]**

Now let me show you one more thing. James wants to join the Night Swim event, so let's try to join.

**[Click on Night Swim event]**

But look, the participants are full! The Join button is either disabled or shows "Event Full." This prevents overbooking and ensures the host can manage the event properly.

**[Attempt to join - show it's not possible]**

So James can't join this one, but he can follow the host or save it for future reference if similar events are posted.

**Handoff**
And that's the participant experience! James can discover events, filter based on his preferences, join events, chat with other participants, manage his profile, and keep track of everything through My Events and Group Chat.

Now I'll hand it back to Divisha for the next section.

## Divisha

**Reporting & Moderation Features**
**[Continue as James]**

Now let's say James joined another event, this time it's a badminton event. But while browsing, he comes across an event or a user that violates the community guidelines. Let me show you how the reporting system works.

**[Navigate to an event or user profile]**

Here, James notices something inappropriate, maybe the event description is misleading, or a user is behaving inappropriately. He can flag this.

**[Click on flag/report button]**

When James clicks the flag or report button, he can select the reason for reporting,whether it's spam, inappropriate content, harassment, or something else. This helps our moderation team understand the issue quickly.

**[Submit report for event]**

Let's submit a report for this event.

**[Navigate to a user profile and report]**

And let's also report a problematic user, let's say User1 has been sending inappropriate messages or posting concerning content.

**[Submit report for user]**

Done! Both reports have been submitted and are now in the admin queue for review.

Admin Review & Banning
**[Logout and login as admin]**

Now let me logout and login as admin to show you what happens on the moderation side.

**[Navigate to Admin Control Panel]**

In the Admin Control Panel, admins can see all reported content, both events and users. Let me open the Review Events section first.

**[Open reported events]**

Here's the event James flagged. As an admin, I can review the details and decide whether it violates our community guidelines.

**[Ban the event]**

This event is indeed inappropriate, so I'm going to ban it. This removes it from the platform completely.

**[Navigate to Review Users]**

Now let's check the reported users. Here's User1 that James reported, and I can also see User2 who has multiple violations.

**[Ban User1 and User2]**

I'm going to ban both accounts. When a user is banned, they lose access to their account and can no longer create or join events.

**[Confirm bans]**

Both users are now banned from the platform.

**Banned User Experience**
**[Logout and try to login as User1 or User2]**

Now let's see what happens when a banned user tries to log in. Let me logout and try to login as User1.

**[Enter credentials and attempt login]**

As you can see, the system shows an error message: "Your account has been banned for violating community guidelines." The user cannot access their account at all.

**[Try User2 as well]**

Same thing with User2, they're locked out and can't access the platform.

**Event Visibility After Ban**
**[Logout and login as James]**

Now let's go back to James's account and see what happened to that banned event.

**[Navigate to All Events]**

If we go to All Events and try to search for the event that was banned, it's completely gone. It doesn't show up in search results, it's not visible anywhere on the platform.

**[If James had joined that event, check My Events]**

And if James had joined that event before it was banned, it would also be removed from his My Events page automatically. The system cleans everything up to maintain platform integrity.

This moderation system helps us keep HobbyHive safe and welcoming for everyone. Users can report issues, admins can review and take action, and everything updates in real-time across the platform.

**-LIVE DEMO END-**

## Swetha

### Slide 13

For HobbyHive, we followed several software engineering best practices to ensure our system was maintainable and scalable. Firstly, we implemented Clear Separation of Concerns using an MVC-style structure, separating our models, controllers, and views. This improved maintainability, debugging, and testability since each component has a distinct responsibility. Secondly, we emphasized comprehensive and structured documentation, which helped us plan before implementation and maintain consistent development across all team members.

### Slide 14

We also made use of user stories to clearly define our main functions from a user's perspective.For example, in the Event Creation story, users can log in, fill out event details, and receive validation feedback if there are errors. Once created, events are stored and displayed for access. Similarly, for the Group Chat feature, users can create or join chats linked to events, send messages, and manage members. These stories helped us translate real-world user needs into structured system behavior.

### Slide 15

We adopted the Scrum methodology to manage development efficiently.

Our sprint cycle was set to two weeks, with Divisha as our Scrum Master and Ishita as our Product Manager. Divisha helped to highlight deliverables for each week and ensured tasks were completed on time. Ishita helped to schedule weekly meetings and communicate with TA as the TA in this case is our client. Each sprint included planning, implementation, and review phases. Weekly check-ins helped us track progress and adapt quickly. Using this iterative process ensured that we stayed aligned and continuously improved the product based on feedback.

### Slide 16

Our application uses three key design patterns.

First, we use the Facade Pattern through our controllers. These can be found in folders like app/admin/controllers and app/host/controllers. They hide complex business logic and provide a simple interface to the frontend.

Second, we implement the Publisher-Subscriber Pattern for real-time messaging. This allows users to chat in real-time during events.

Finally, we use the Singleton Pattern for authentication token management. Our oneMapTokenManager ensures only one OneMap API token exists at any time, which automatically refreshes every 72 hours.

**Slide 17**

Our application follows SOLID principles for clean architecture.

Starting with Single Responsibility: Each module does one thing well. For example, our moderate events, review events and review users which each have unique functionality

For Open-Closed Principle: We can extend functionality without changing existing code. Our categoryLocationMapping config file is perfect for this, we can add new event categories just by adding entries, no code modification needed.

Liskov Substitution is ensured through TypeScript interfaces. Whether an event is hosted or attended, they all use the same Event interface, so they're completely interchangeable in our system.

Finally, Interface Segregation: Our folder structure enforces this. The host folder only has access to hosting features, while participant folder only accesses profile features. Each module depends only on what it actually needs.

**Slide 18-19**

For testing, we performed black box testing to verify that our registration feature worked as expected under different user inputs. Each test case had a clear input, expected output, also known as the oracle, and a log to verify if the outcome matched. Let’s take Test Case 1, the Valid Signup example. Here, the user enters a valid username, email, and matching passwords. The expected behavior is that the user’s profile is successfully created in our Supabase database, and they can view their profile in the app. During testing, we confirmed that the signup process completed without any errors, and the data was correctly stored, so this test case passed. By systematically going through each case, including invalid, missing, or duplicate inputs, we ensured that the signup module was robust and reliable.

**Slide 20**

Finally, we used white box testing to analyze the internal logic of our login function.

This flowchart shows the control paths, including conditions for empty fields, invalid emails, and wrong passwords. Our baseline path represents a successful login scenario. By covering different paths, we confirmed that each logical branch was correctly implemented and error-handling worked as expected.

## Afreen

**Slide 19**

This slide shows the traceability of the Login feature across our system artifacts.
Starting from the Use Case Diagram, the *Login* use case defines how a user interacts with the system to authenticate and gain access.

In the Sequence Diagram, we see this interaction flow, the user inputs credentials, the system verifies them through FindAccount and verifyPassword, and finally creates a session before redirecting the user.

This is directly implemented in our login/page.tsx file, specifically in the handleSubmit() function, where we call Supabase's signInWithPassword() method.
There are additional logic checks for banned users, where it retrieves their role, and redirects them accordingly.

This clear traceability from diagram to implementation demonstrates how our design aligns with our final code execution.

**Slide 20**

For our future development, we plan to introduce AI-powered event summaries to automatically generate concise overviews of past group chats.
This feature will help participants quickly catch up and maintain continuity between events.

We also plan to implement cross-platform synchronization with messaging apps like Telegram and WhatsApp.

This integration allows users to communicate seamlessly across different platforms, making HobbyHive more accessible and engaging.

Thank you!