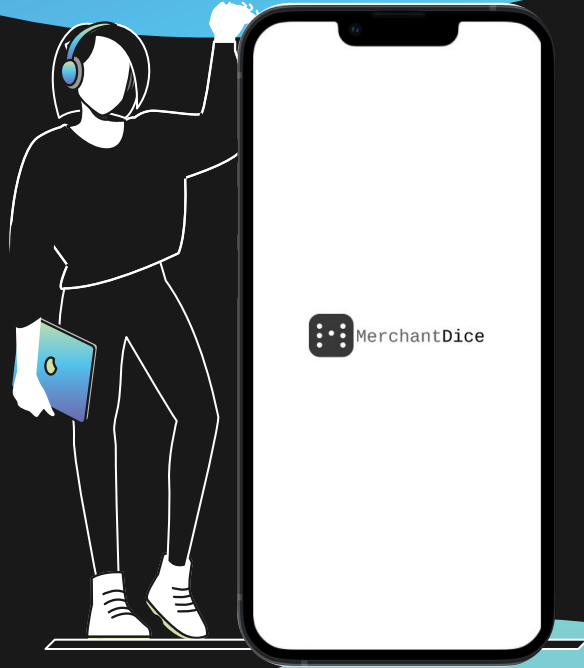


++



# MerchantDice

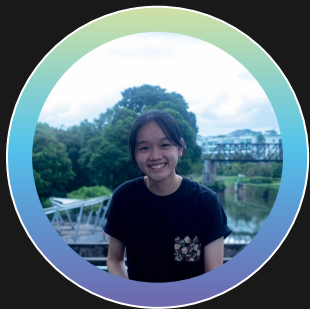
## Lab 5 Presentation

Brought to you by: APT200

||||

.....

# OUR TEAM



Ling Yin  
Project Manager



Andre Lee  
Lead Developer



Andre Lim  
Frontend Developer



Trevor  
Backend Developer

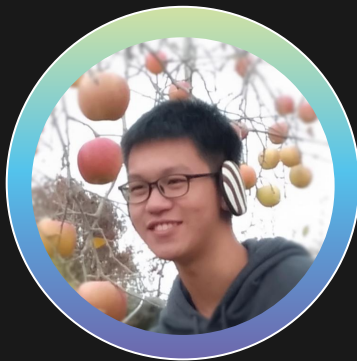


# OUR TEAM



Kaisheng

QA Manager



Andrew

QA Engineer



Yang Yang

Release Engineer/Manager



# TABLE OF CONTENTS

01

## Product Introduction

Introducing the  
key functionalities  
of our app and the  
brains behind it

02

## Project Management

Project  
management tools  
and strategies

03

## Design for Maintainability

How we designed  
for change and  
maintainability

04

## Quality Assurance

How QA was  
conducted



# TABLE OF CONTENTS

05

## Risk Management

Analyzing and  
controlling threats

06

## Use Case

Project  
Functionalities

07

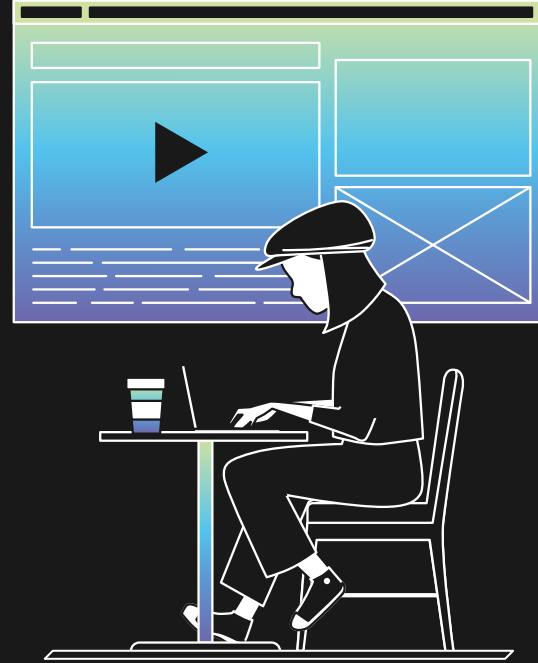
## Demo!!

Showing you the  
product



01

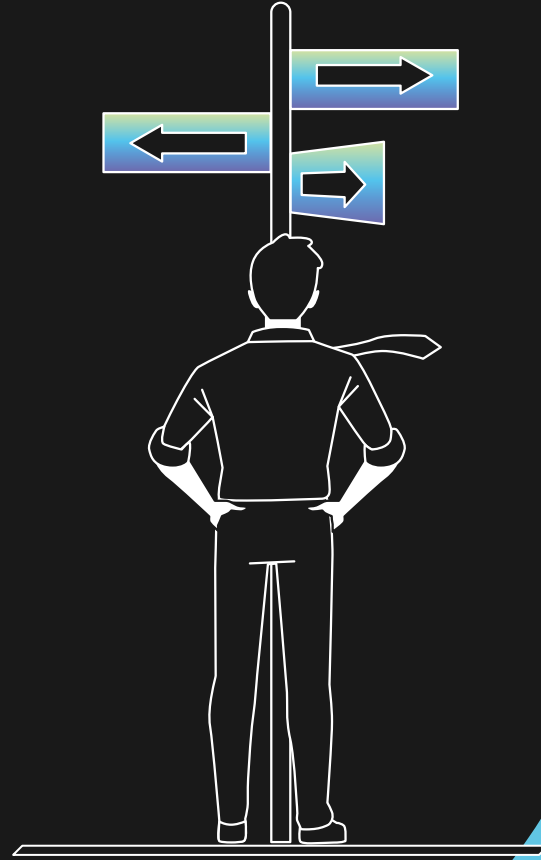
# PRODUCT INTRODUCTION





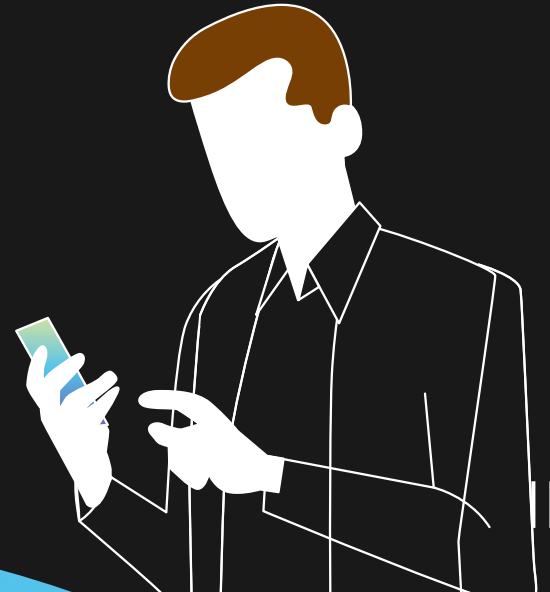
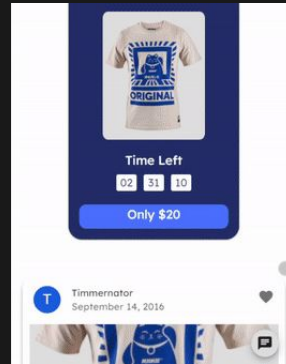
# Problem Statement

With the rise of awareness about the **environmental impact of fast fashion**, thrifting - buying and selling second hand items, has become an increasingly popular means of shopping amongst Singaporeans. However, there are **little known avenues for such exchanges to take place**



# SOLUTION

MerchantDice





# PRODUCT OVERVIEW



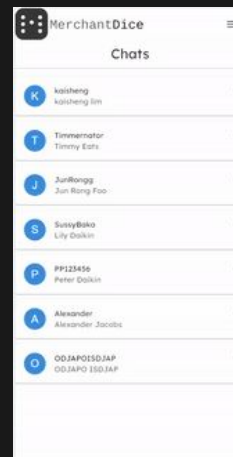
COMMUNICATE

SELL

Display. Advertise. Sell.

BUY

Search. Favourite. Buy.

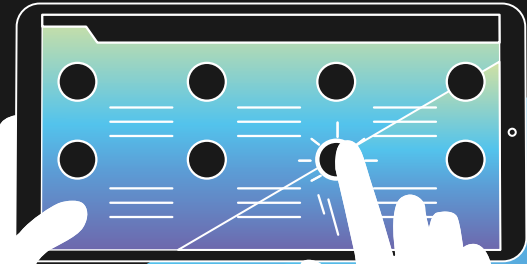


02

++

||||

# PROJECT MANAGEMENT



# PROJECT MANAGEMENT



PROJECT  
ORGANISATION



PROJECT  
MANAGEMENT  
TOOLS



PROJECT  
ESTIMATION

+

+

•

•

•

•

•

•

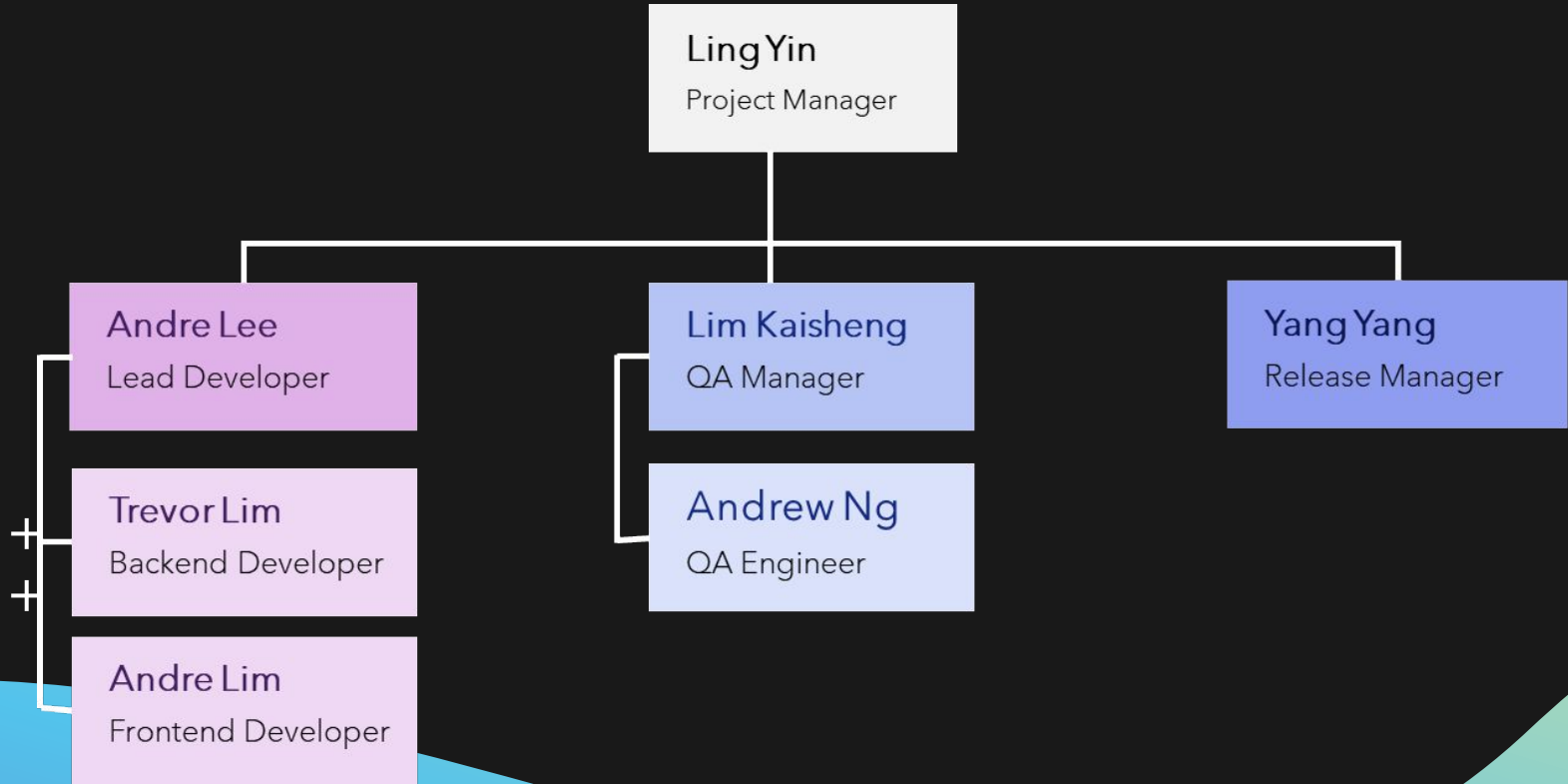
•

•

•

•

# ORGANISATION CHART



# AGILE METHODOLOGY



Jira

Project tracking and planning  
app that helps to keep track of  
the backlog and facilitate sprint  
planning

+ +

# COMMUNICATION CHANNELS



Zoom

Scrum meetings are held on Zoom.



Telegram

Telegram is used for informal discussions within the team

# VERSION CONTROL



Google Drive

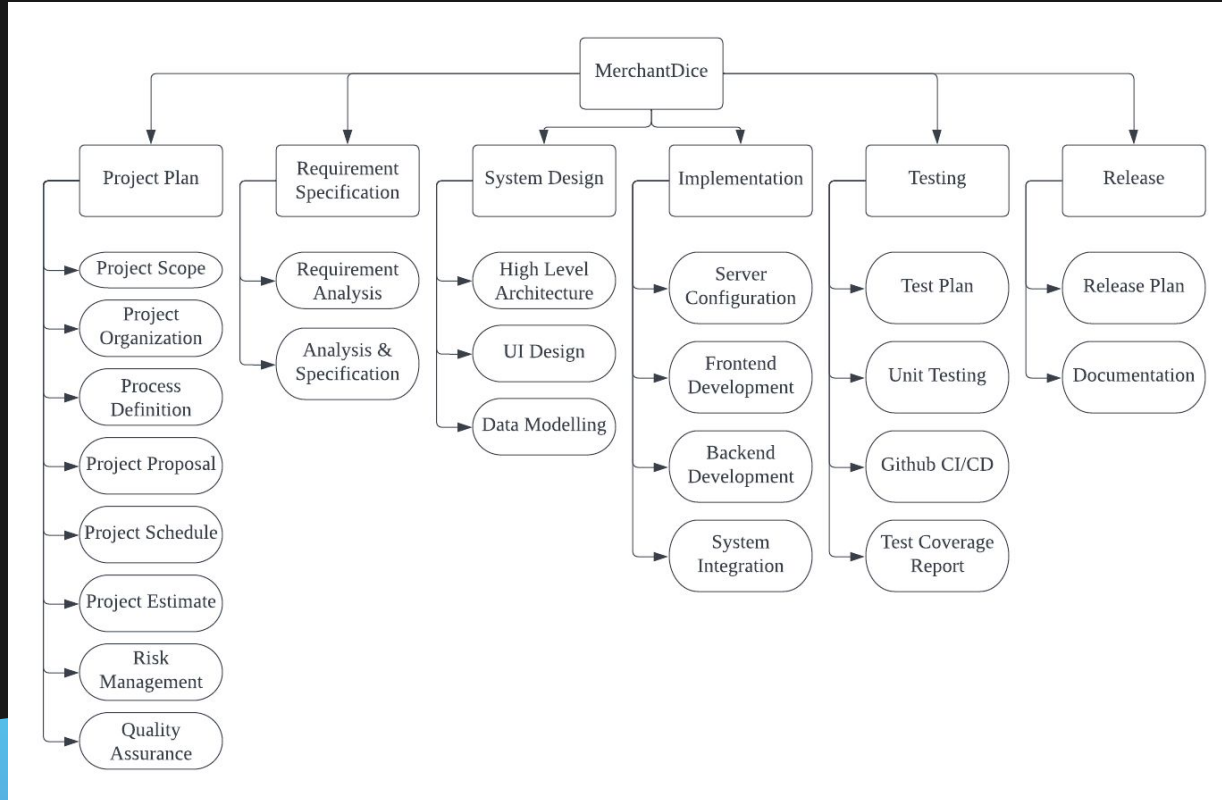
Used to do version control for documentation. Used to store meeting minutes



Github

Version control for our code base

# WORK BREAKDOWN STRUCTURE





	Personnel	Start Date	End Date	Duration (days)	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12
					M	T	W	F	M	T	W	F	M	T	W	F
1 Project Proposal and Project Requirements																
1.1	Project Proposal	Ling Yin	25/8/22	7/9/22	14											
1.2	Use Case Diagram	Andre Lee	25/8/22	7/9/22	14											
1.3	Team Information	Ling Yin	25/8/22	7/9/22	14											
1.4	UI Mockups	Andre Lim	31/8/22	7/9/22	8											
1.5	Database Schema	Trevor	31/8/22	7/9/22	8											
2 Software Quality Management																
2.1	Quality Plan	Andre Lee	8/9/22	12/9/22	5											
2.2	Software Requirement Specification	Ling Yin	8/9/22	12/9/22	5											
3 Project Planning																
3.1	Risk Management	Ling Yin	13/9/22	16/9/22	4											
4 Prototype Construction																
4.1	Frontend Development	Andre Lim	9/9/22	29/9/22	21											
4.1.1	Home Page	Andre Lim	9/9/22	15/9/22	7											
4.1.2	NaviBar	Andre Lim	16/9/22	18/9/22	3											
4.1.3	Login Page	Lim Kai Sheng	9/9/22	29/9/22	21											
4.1.4	Registration Page	Lim Kai Sheng	9/9/22	29/9/22	21											
4.1.5	Product	Andre Lim	19/9/22	28/9/22	10											
4.1.6	Manage Product Page	Andre Lim	23/9/22	26/9/22	4											
4.2	Backend Development	Trevor	9/9/22	29/9/22	21											
4.2.1	User Account DB	Trevor	9/9/22	16/9/22	8											
4.2.2	User Authentication	Trevor	19/9/22	29/9/22	11											
4.2.3	Product DB	Trevor	9/9/22	16/9/22	8											
4.2.4	API Endpoints	Andrew	19/9/22	23/9/22	5											
5 Configuration Management																
5.1	Software Maintainability	Andre Lee	6/10/22	10/10/22	5											
5.2	Configuration Management Plan	Ling Yin	6/10/22	10/10/22	5											
5.3	Change Management Plan	Ling Yin	6/10/22	10/10/22	5											
5.4	Release Plan	Yang Yang	6/10/22	10/10/22	5											
6 Software Development																
6.1	Frontend Development	Andre Lim	29/9/22	20/10/22	22											
6.1.1	Catalog	Lim Kai Sheng	29/9/22	20/10/22	22											
6.1.2	Favourite Function	Andre Lim	1/10/22	2/10/22	2											
6.1.3	Chat	Andre Lim	5/10/22	20/10/22	16											
6.1.3	History	Andre Lim	1/10/22	4/10/22	4											
6.1.4	Routing Pages	Lim Kai Sheng	29/9/22	20/10/22	22											
6.2	Backend Development	Trevor	29/9/22	20/10/22	22											
6.2.1	Chat	Andrew	5/10/22	20/10/22	16											
6.2.2	Recommendation function	Andrew	1/10/22	2/10/22	2											
6.2.3	History	Andrew	1/10/22	4/10/22	4											
6.3	Dockerize	Lim Kai Sheng	20/10/22	22/10/22	2											
6.4	Deploy	Ling Yin	20/10/22	22/10/22	2											
7 Software Testing																
7.1	Github Action Config	Andrew	20/10/22	29/10/22	9											
7.2	Test Plan	Lim Kai Sheng	20/10/22	29/10/22	9											
7.3	Test Cases and Requirements Test Coverage Report	Lim Kai Sheng	20/10/22	29/10/22	9											
7.4	Full CI/CD for github	Lim Kai Sheng	20/10/22	29/10/22	9											
8 Project Wrap Up																
8.1	Presentation Slides	Ling Yin	29/10/22	10/11/22	12											
8.4	Documentation	Andre Lee	29/10/22	10/11/22	12											

# PROJECT BUDGET

## Hardware

7 x 14-inch Macbook Pro	Total Cost: \$17,500
10-Core CPU	
16-Core GPU	
16GB Unified Memory	
1TB SSD Storage <sup>1</sup>	



# PROJECT BUDGET

## Software

Visual Studio Code	\$0.00
Amazon AWS EC2 Instance (Deployment)	\$0.00
MongoDB Database Server	\$0.00
Google Drive	\$0.00



# PROJECT BUDGET

## Other Resources

7 Employees with 772.8 working hours with \$16.00/hour	\$12364.80
--	------------





\$29,864.80

Total Proposed Budget

+

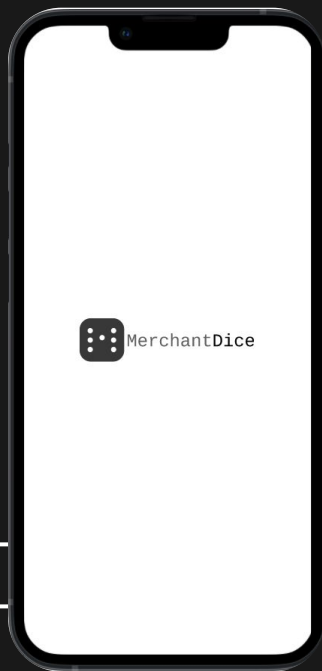
+

03

# DESIGN for MAINTAINABILITY



# EARLY PLANNING DESIGN FOCUS



## SCALABILITY

Basic vs Future Functionalities

## MODULARITY

Functional requirements  
can change

bcrypt

jwt

multer

## MAINTAINABILITY

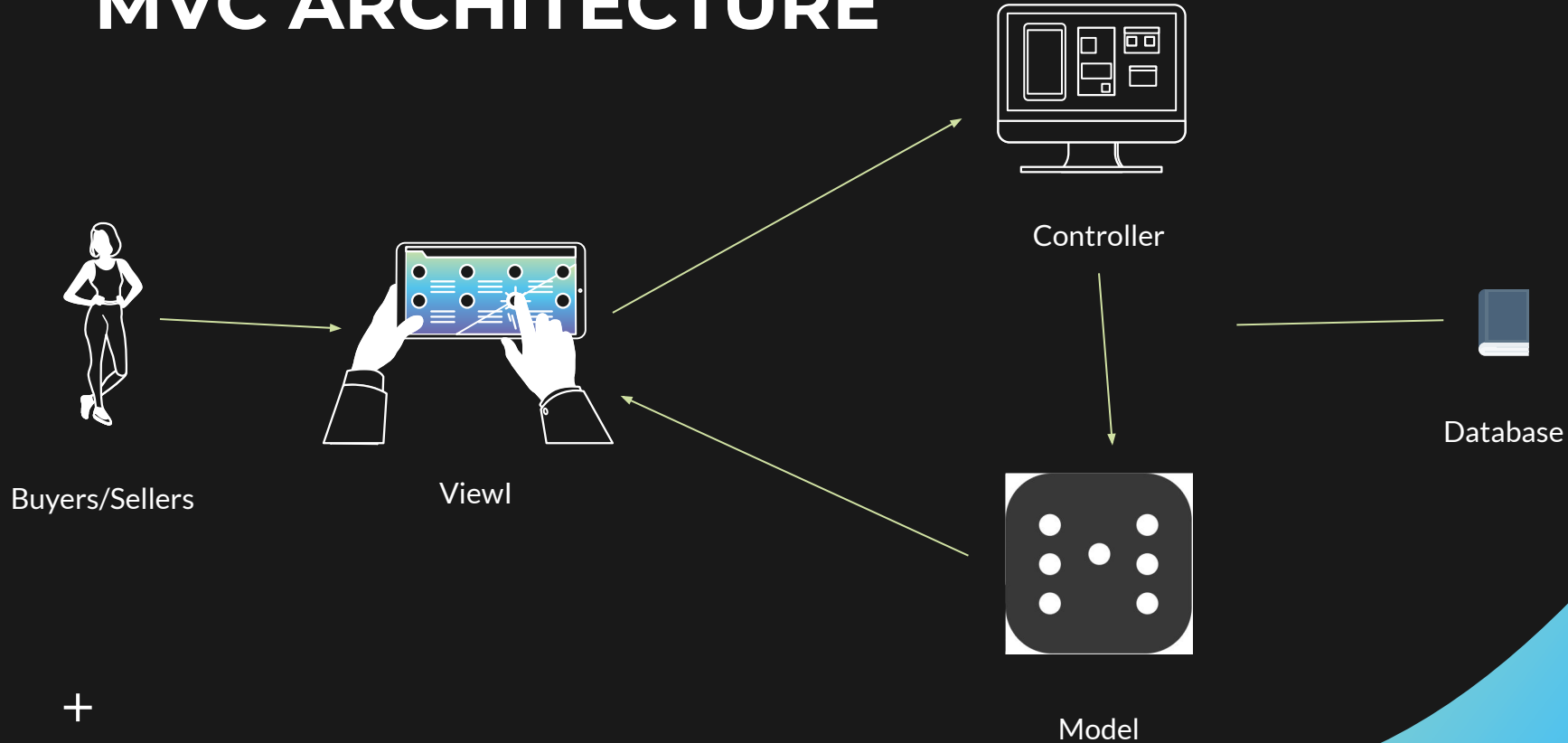
Future of the project

GitHub

ES6 Standard

SwaggerUI

# MVC ARCHITECTURE



+

+

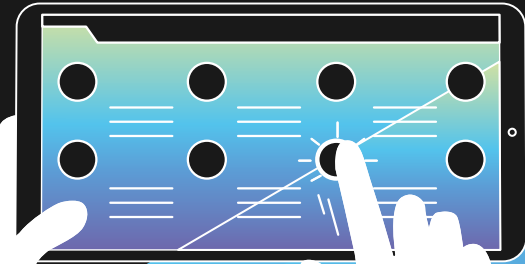


04

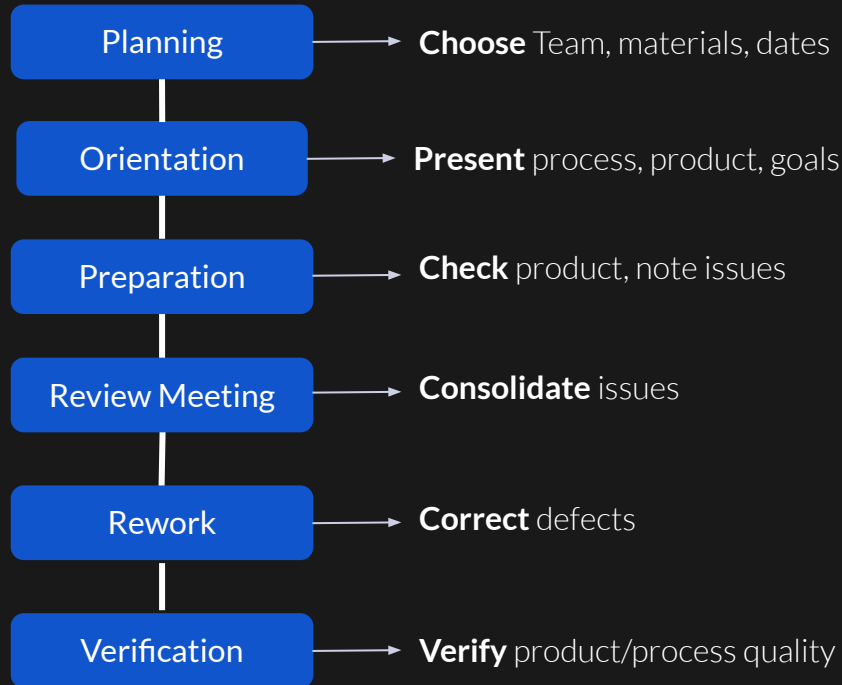
# Software Quality Assurance

++

||||



# TECHNICAL REVIEW



+

+

# TECHNICAL REVIEW



## Walkthroughs

- Informal walkthrough
- Developers not directly involved to perform review and raise issues found
- Iterative rectification of issues identified

## Inspections

- Formal inspection from external reviewers with defined roles
- Identify issues and severity, for the entire project
- To be done repeatedly until all errors resolved

+

+

# MANAGEMENT REVIEW

- By Project manager, Quality Manager and Lead Developer
- Every 2 weeks
- Include:
  - Status of Schedules
  - Confirm requirements
  - Verify effectiveness
- Help meet time and budget constraints

+ +



# AUDITS

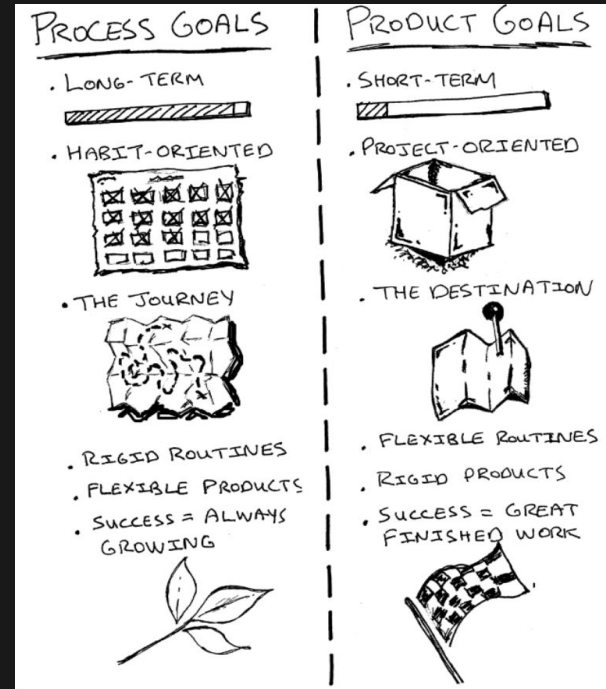


1. Consult people external to the organisation to conduct independent evaluation
2. Audit to be conducted once per year
3. For conformation to regulations, standard and plans such as the Project Plan



# ..... Process Goals

- Identify key areas of improvement
- Establish a **Quality Culture** in the team
- Traceability in product lifecycle, technical soundness, conformance to requirements etc.



# ABILITY TO PERFORM



## Resources & Training

- All required hardware and software resources
- Training as part of CZ3002 and relevant industry standard.

## Tools

- Project Management using Google Drive
- Version Control by Git
- VS code and docker as development tool

## Organizational Structure

- + - 2 subteams: Frontend & Backend.
- + - Members from 2 sub-teams with same component work together to integrate.

# ACTIVITIES PERFORMED



## Plans & Procedures

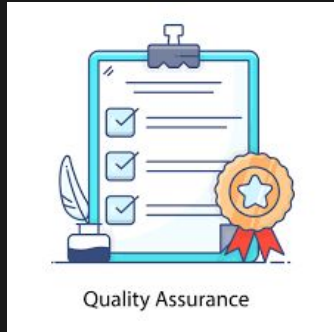
- System Requirement Specifications
- Change Management Plan
- Configuration Management Plan
- Design report on software maintainability
- Project Plan
- Quality Plan
- Risk Management Plan
- Release Plan

## Traceability

- ‡ All the documents are tracked and kept in SVN.
- ‡ Necessary corrections through a change request workflow.



# MEASUREMENT AND ANALYSIS



Quality Assurance

- Weekly assessments and discussions
- Quantifiable and verifiable performance metrics.
- Analysis and suggestions based on PM and current state.
- Changes follow Software Quality Assurance and management

+

+

# API MANUAL TESTING

REST API 1.0.0

[ Base URL: localhost:3001/ ]

Schemes

HTTP

**POST** /auth/register

Parameters Try it out

Name	Description
body object (body)	Example Value Model

```
{  "username": "my",  "password": "my",  "firstName": "my",  "lastName": "my"}
```

Parameter content type  
application/json

Responses Response content type: application/json

Code	Description
200	OK
400	Bad Request
500	Internal Server Error

**PUT** /user/{id}

Parameters Try it out

Name	Description
id string (path)	id

body  
object  
(body)

Example Value Model

```
{  "username": "my",  "password": "my",  "currentUsername": "my"}
```

Parameter content type  
application/json

Responses Response content type: application/json

Code	Description
200	OK
403	Forbidden
500	Internal Server Error

**POST** /auth/login

**GET** /user/{id}

**PUT** /user/{id}

**DELETE** /user/{id}

**GET** /user/

**PUT** /user/{id}/follow

**PUT** /user/{id}/unfollow

**POST** /posts/

**GET** /posts/{id}

**PUT** /posts/{id}

**DELETE** /posts/{id}

**PUT** /posts/{id}/like

**GET** /posts/{id}/timeLine

**POST** /upload/

**POST** /chat/

**GET** /chat/{userId}

**GET** /chat/find/{firstId}/{secondId}

**POST** /message/

**GET** /message/{chatId}

**GET** /image/

**POST** /image/

**GET** /image/{id}

**DELETE** /image/{id}

# AUTOMATED TESTING



```
PASS test/testToRunSequentially
sequentially run tests
POST /auth/register
  given a username and password
    ✓ should respond with a 200
    ✓ when the username and password are valid
  ✓ should respond with 401
POST /auth/login
  given a username and password
    ✓ should respond with a 200
    ✓ when the password is wrong
    ✓ should respond with 401
    ✓ when the username and password are invalid
POST /chat
  given valid senderID and recipientID
    ✓ should respond with a 200
  given missing senderID or recipientID
    ✓ should respond with a 400
GET /chat
  given valid userID for recipient
    ✓ should respond with a 200
  given empty userID for recipient
    ✓ should respond with a 400
  given valid first and last name
    ✓ should respond with a 200
POST /message
  given valid chatID, senderID, and message
    ✓ should respond with a 200
GET /message
  given valid chatID
    ✓ should respond with a 200
  given missing chatID
    ✓ should respond with a 400
  given valid senderID
    ✓ should respond with a 200
  get all users
    ✓ should respond with a 200
  get single user
    ✓ should respond with a 200
PUT /user
  given same userID for request and response
    ✓ should respond with a 200
  given different userID for request and response
    ✓ should respond with a 400
DELETE /user
  given userAdmin
    ✓ should respond with a 200
  given same userID for request and response
    ✓ should respond with a 200
  given different userID for request and response
    ✓ should respond with a 400
```

Test Suites: 1 passed, 1 total  
Tests: 20 passed, 20 total  
Snapshots: 0 total  
Time: 8.941 s  
Ran all test suites.

✓ chore: beautify product page	main	2 days ago	29s	...
CI Testing #16: Commit b61c8e8 pushed by AndrewNYK				
✓ CD Docker Hub		3 days ago	2m 46s	...
CD Docker Hub #8: completed by Interstellarkai				
✓ chore: clean up	main	3 days ago	25s	...
CI Testing #15: Commit aa6fa77 pushed by Interstellarkai				
✓ CD Docker Hub		3 days ago	2m 26s	...
CD Docker Hub #7: completed by Interstellarkai				
✓ feat: cloud CD	main	3 days ago	35s	...
CI Testing #14: Commit f124105 pushed by Interstellarkai				
✓ CD Docker Hub		4 days ago	3m 23s	...
CD Docker Hub #6: completed by Interstellarkai				
✓ chore: reduction in space	main	4 days ago	30s	...
CI Testing #13: Commit 1bd3dc pushed by Interstellarkai				
✗ CD Docker Hub		4 days ago	16s	...
CD Docker Hub #5: completed by Interstellarkai				
✓ fix: wait for CI to complete before CD	main	4 days ago	31s	...
CI Testing #12: Commit 2c517fc pushed by Interstellarkai				
✓ fix: wait for CI to complete before CD	main	4 days ago	32s	...
CI Testing #11: Commit a6575a4 pushed by Interstellarkai				

< Previous 1 2 3 4 5 ... 13 14 Next >

Check : View Shortlist : Filled Shortlist  
Check : Edit Shortlist : Remove Existing Shortlist  
End Program

ogle-chrome  
r/mac64/96.0.4664.45/chromedriver] found in cache  
\*\*\*\*\*

\*\*\*\*\*  
✓

\*\*\*\*\*  
✓

d  
✓

✓

✓

word)  
firm password)  
✓

\*\*\*\*\*  
✓

\*\*\*\*\*  
✓

✓

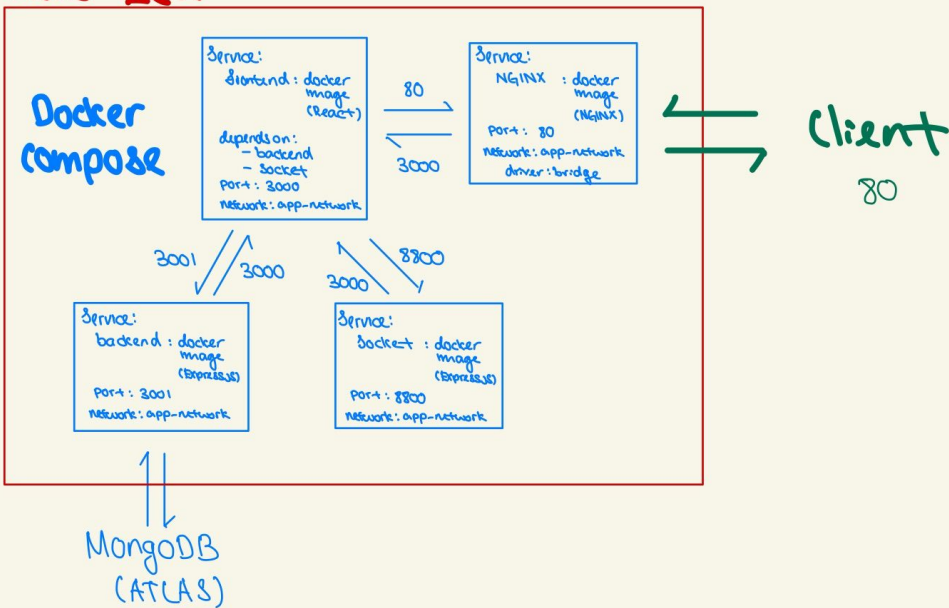
✓

d)  
✓

\*\*\*\*\*  
✓

# System Architecture

AWS EC2

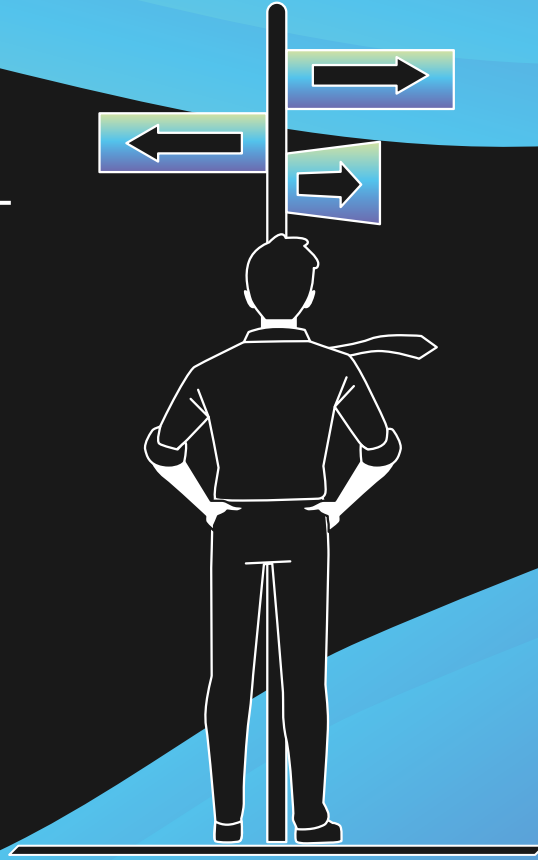


05

# Risk MANAGEMENT

++

|||



# RISK MANAGEMENT PROCEDURE

Identification



Analysis



Response Planning



Monitoring, Controlling and Reporting



# RISK IDENTIFICATION

## Who?

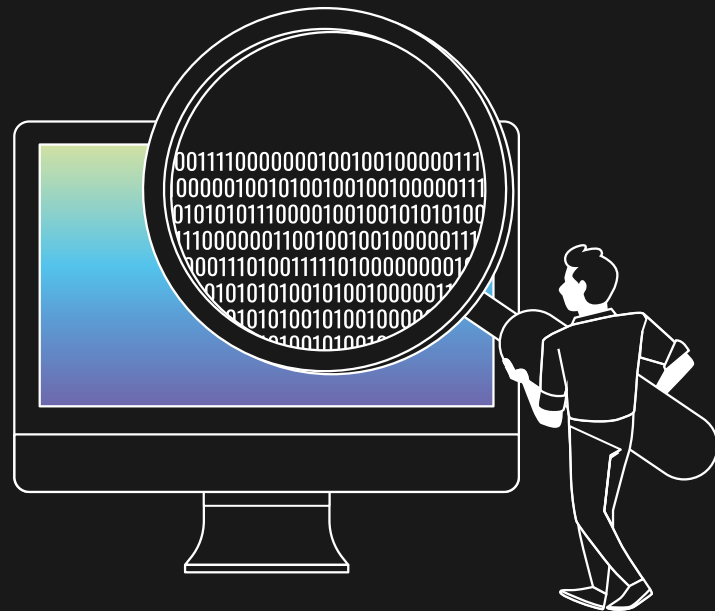
- APT200
- Potential users of MerchantDice

## What?

- Organizational culture
- Project scope
- Cost/effort estimates
- etc.

## How?

- Bi-weekly discussions
- Document risks in shared folder



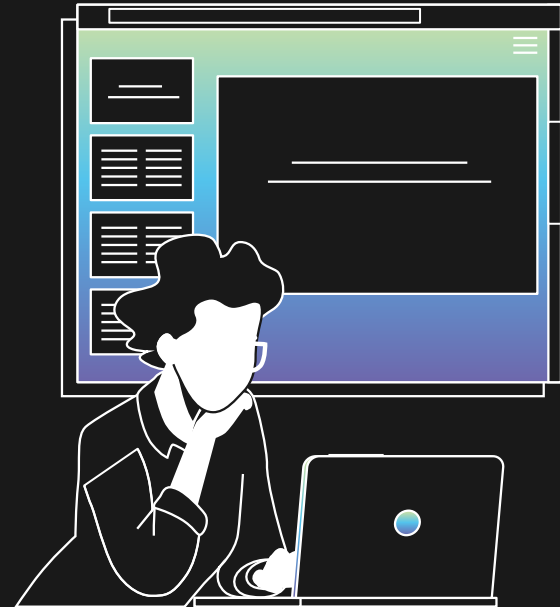
# RISK ANALYSIS

## Qualitative

- Utilize knowledge and experience
- Impact severity (**high**, **medium**, **low**)

## Quantitative

- Estimated numerical rating
- Probability of occurrence (e.g. 25%)





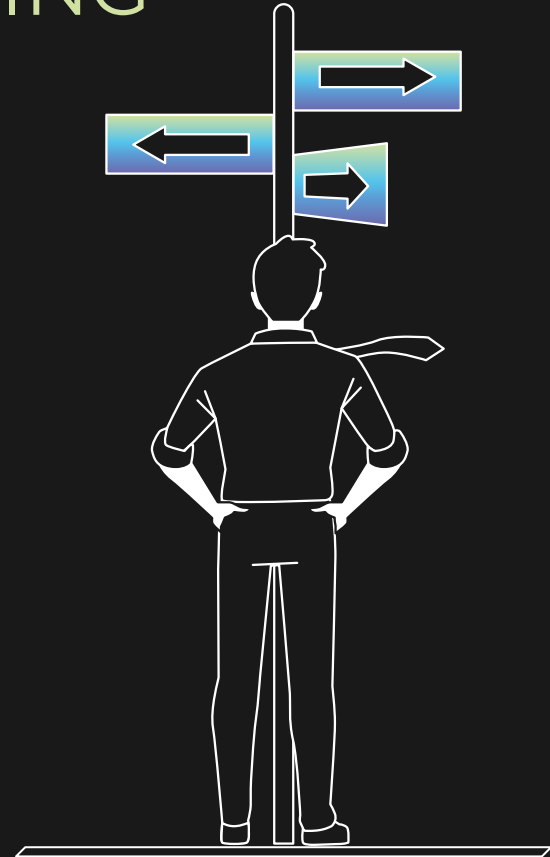
# RISK RESPONSE PLANNING

**Avoid**

**Mitigate**

**Accept**

**Transfer**



# RISK MONITORING, CONTROLLING AND REPORTING

Monitor existing risks

- Residual risks

“Top 10 Risk List”

- Maintained by APT200

Analyze project change requests

- Assess Risk impact
- Update Risk Status



# RISK LOG

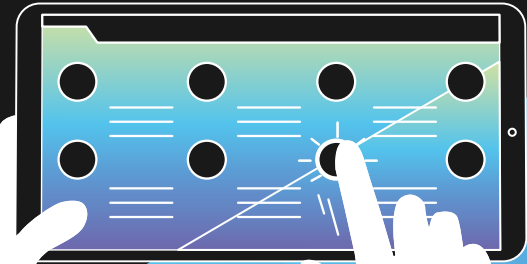
Risk Identified	Impact Severity	Probability	Impacts	Risk Reduction
More changes to requirements than anticipated	High	25%	Need for a requirements documentation update/complete redesign	Be rigorous in eliciting requirements. Make customers aware of potential repercussions of requirement changes.
Problems coordinating within group	Moderate	40%	Inability to measure progress/portions of projects not completed	Manages and motivate team members. Ensure effective communication between team members

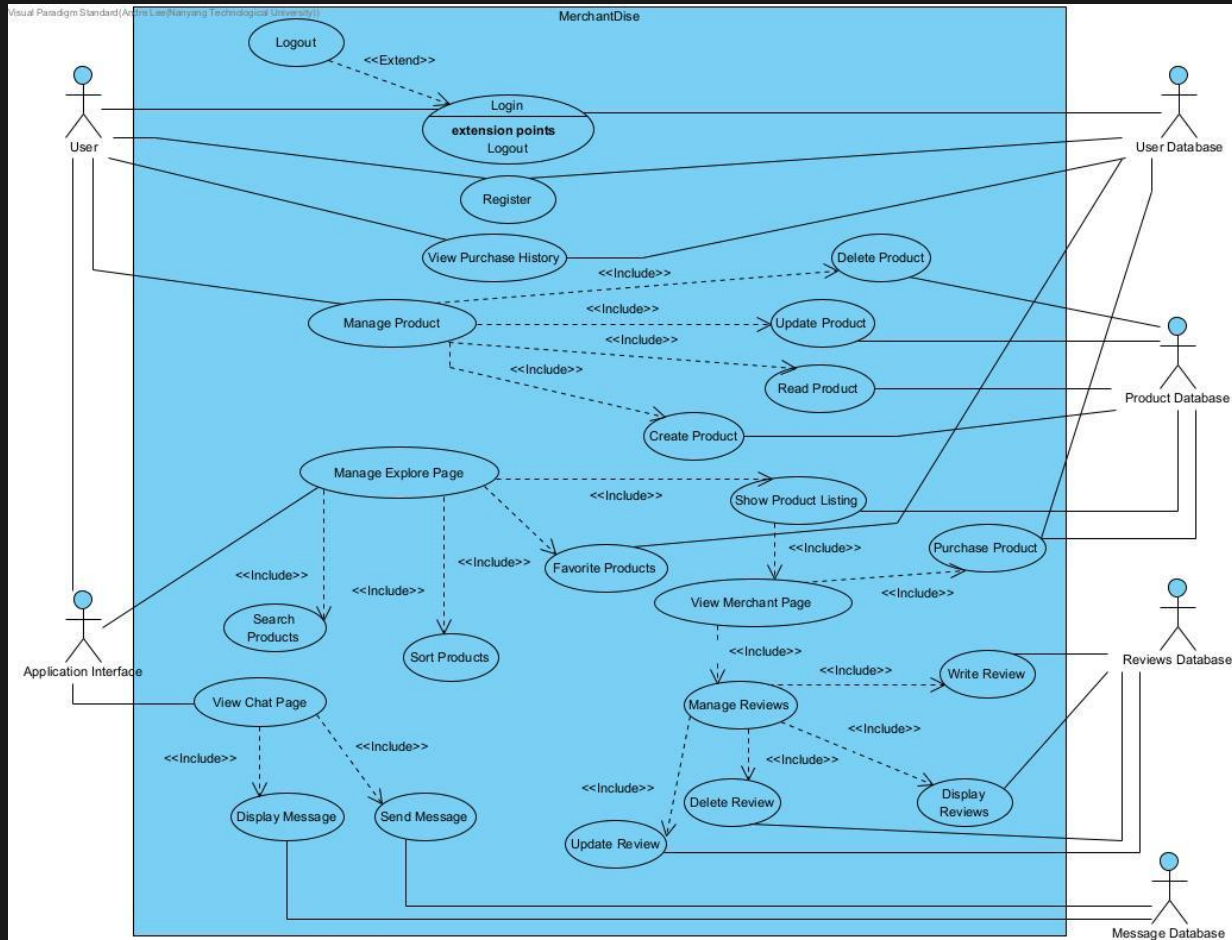
06

++

||||

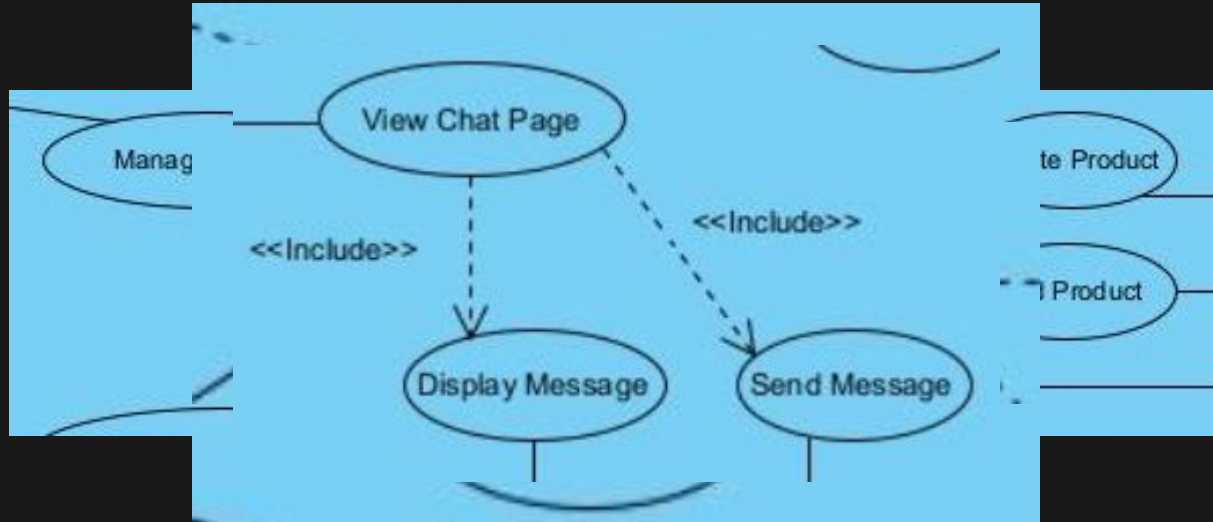
# USE CASE DIAGRAM





++

# Main Functionalities



07

++

||||

DEMO

