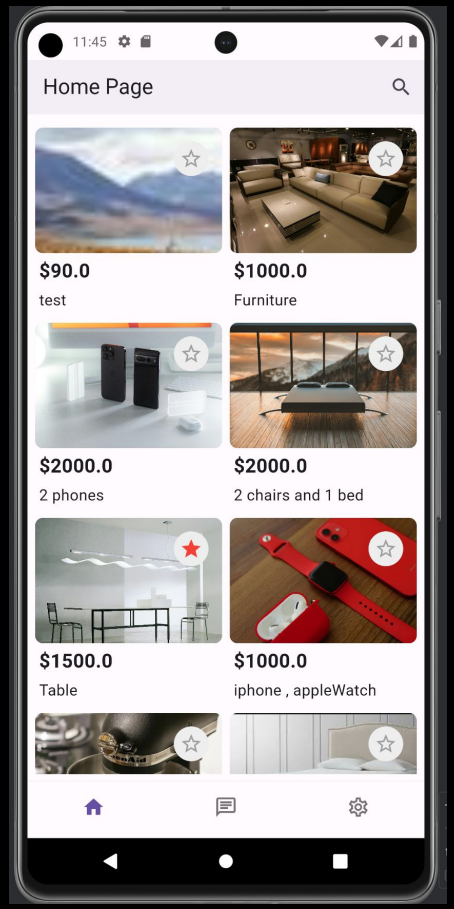


Uwaterloo Exchange by Group 15

The student hub for buying, selling, and Swapping Essentials

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

Why we wanted to make this app?

The Uwaterloo Exchange app was created to address the need for students to easily buy, sell, and exchange items within the Uwaterloo community. With features tailored for students, it provides a dedicated platform to conveniently list and find textbooks, electronics, clothing, and other items.

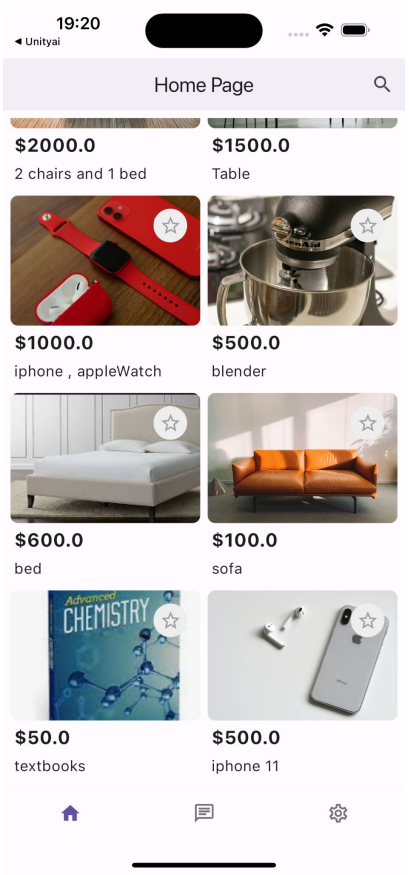
- Verified users
- Nearby exchange
- Dedicated platform
- Scam less likely

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

Key Features

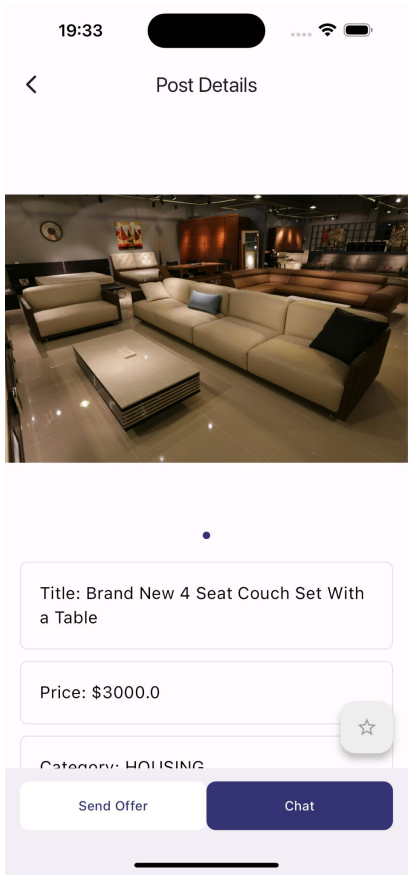
- Post management
  - Create, browse, update, and delete listings.
  - Bookmark / remove a listing for a customer
  - Own listing page
  - Mark Order as completed for completed transaction
  - Track Completed Order with a separate listing page
- Customer profile management
  - Create, read, update, and delete customer profiles.
- Authentication
  - Incorporate authentication into our software.
- Chat
  - Enable chat functionality between customers.
  - Directly Send Offer to other user

User Interface



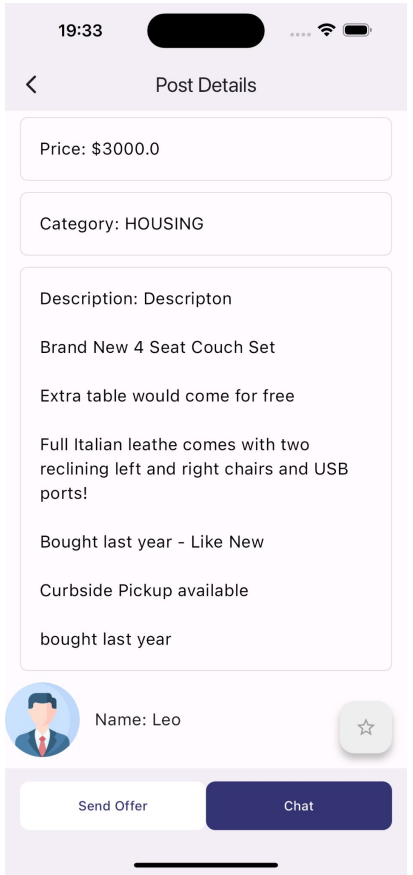
Main Page

People can check for available listings



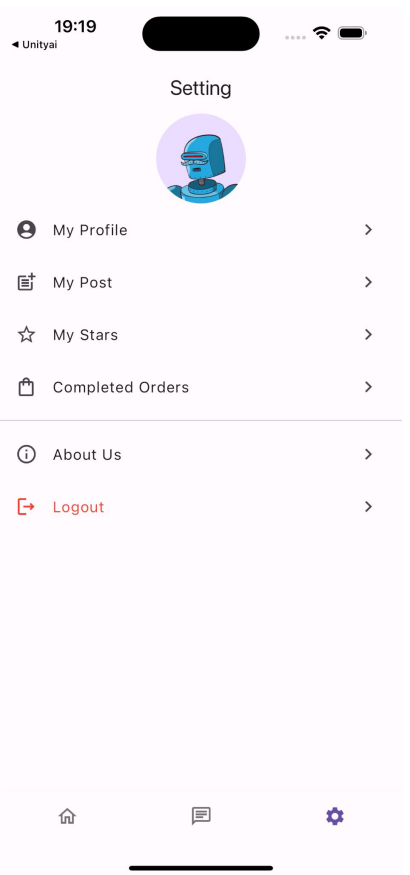
Listing Details

People may expand a listing



Listing Details

People may check for additional details



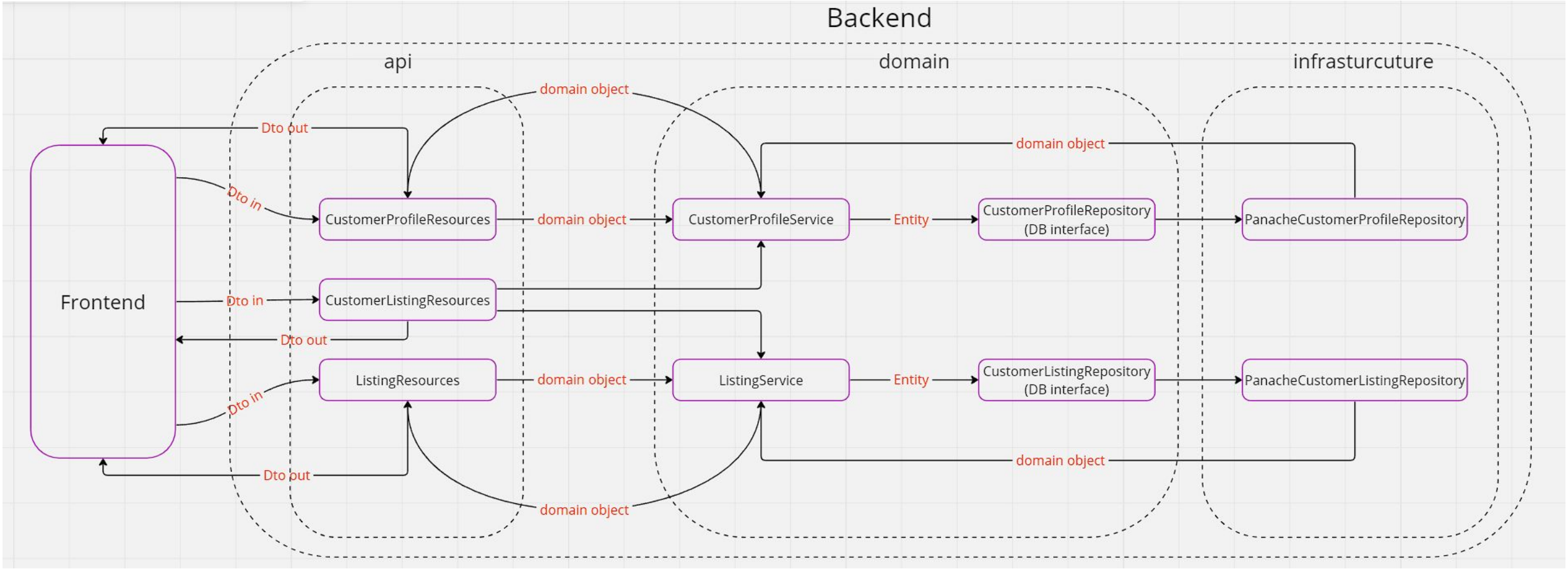
Customer Profile

People update their profile and manage their own listings

Real Time Chat Functionality

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

Project Architecture

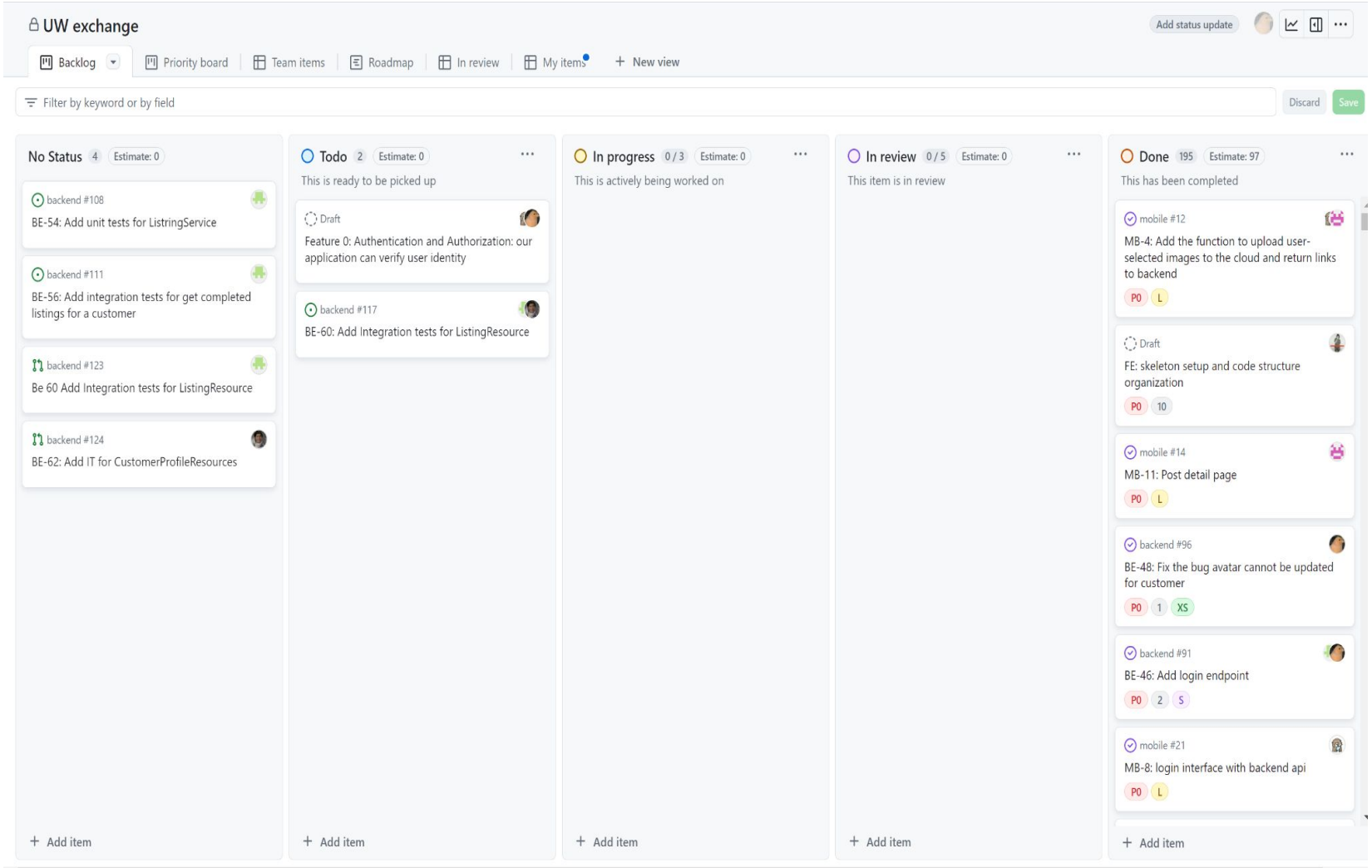


Practices Followed – Agile

user stories and features are broken into tasks.

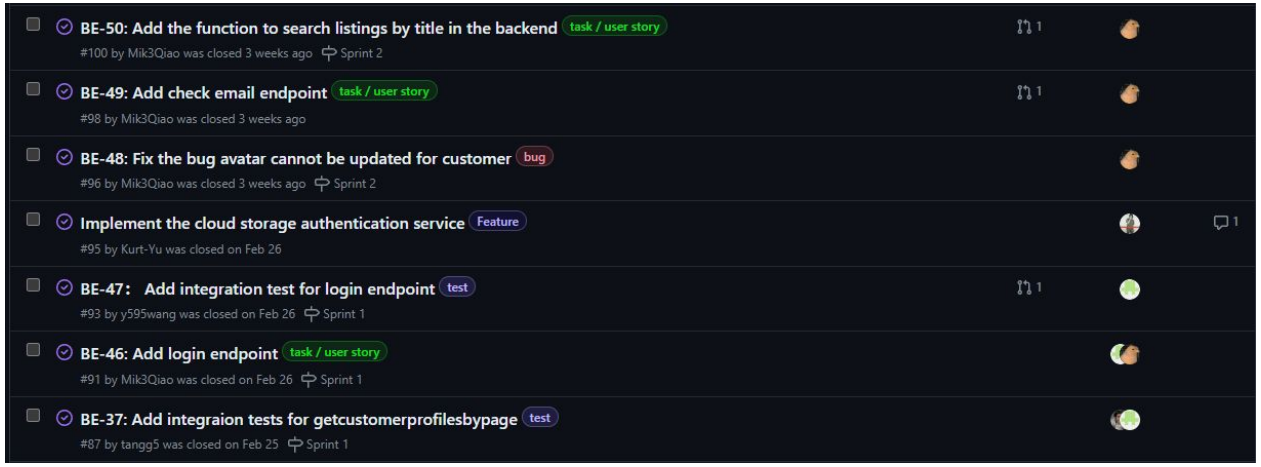
We used **kanban** board to track frontend and backend issues/tasks:

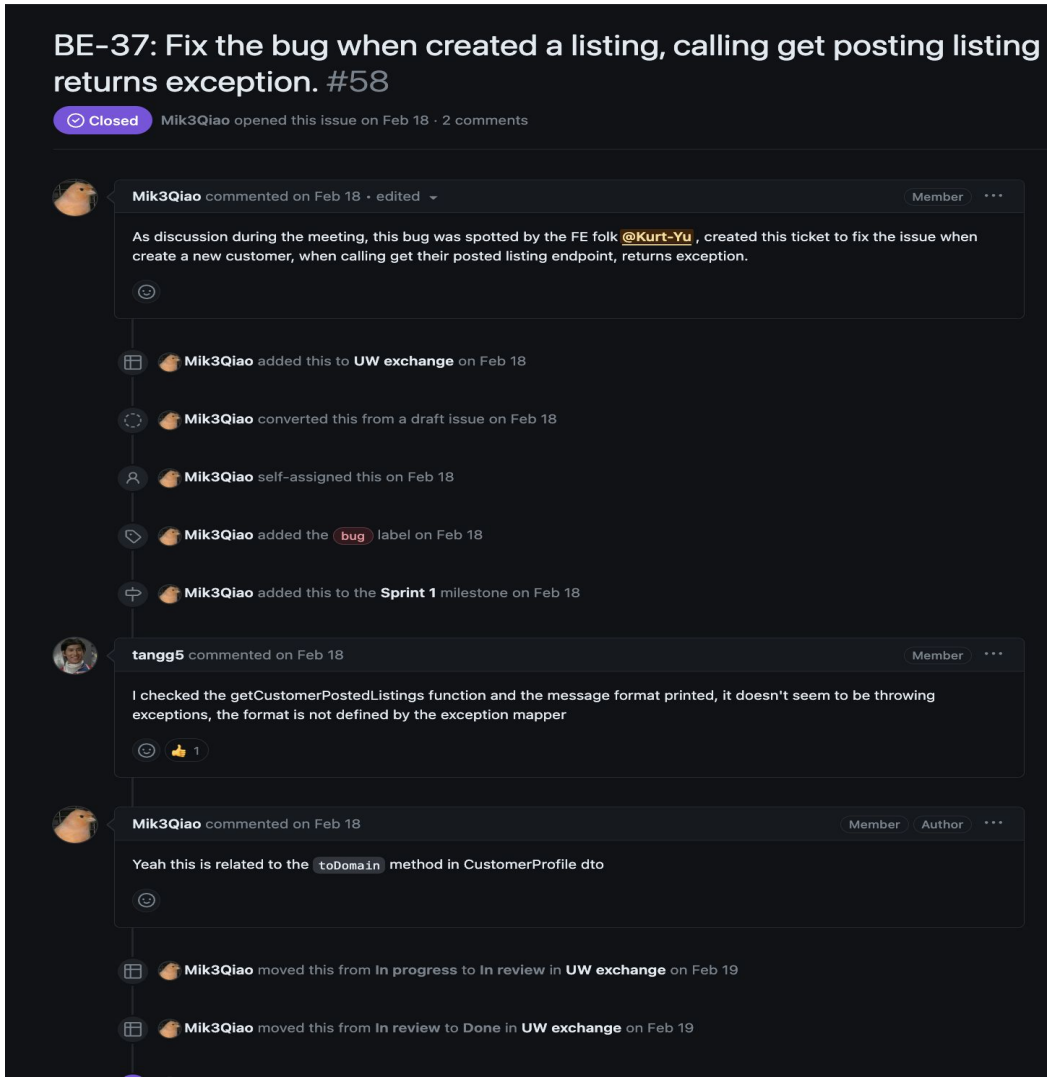
- TODO
  - Something need to be considered for next sprint.
- In progress:
  - Should have tasks for the current sprint
- In Review with a maximum of 5 tasks:
  - Promotes code review within the team
- Done



Practices Followed – Issues

- Each Feature is divided into user stories, and each user story is then used to create an issue
- We have a numbering system for the issues, for example this one is BE-37, which helps us to track commits related to specific issue
- Issues are then tagged with different categories: **task/user story, test, documentation, bug etc.**

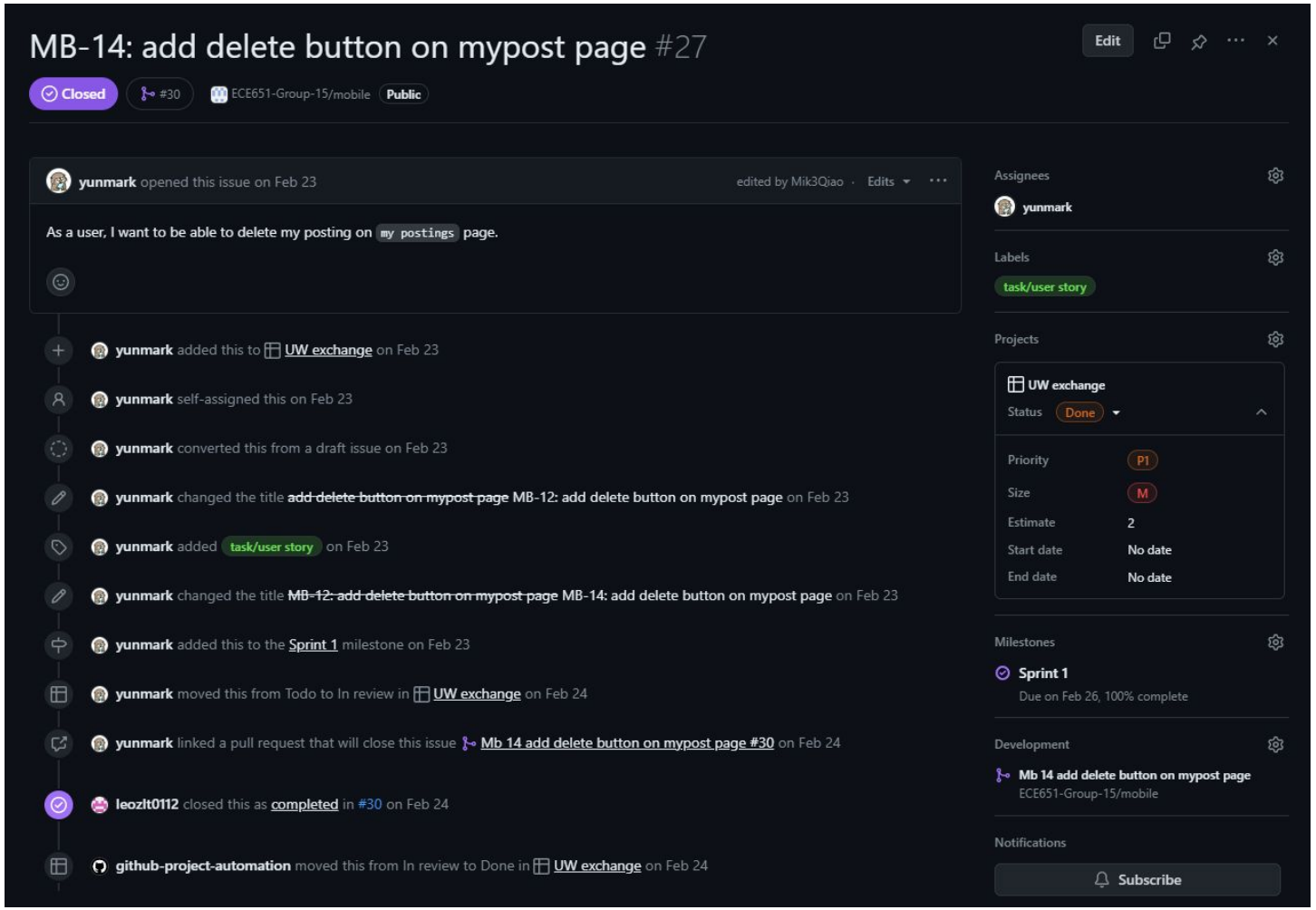




Practices Followed – Sprint Planning: Planning Poker

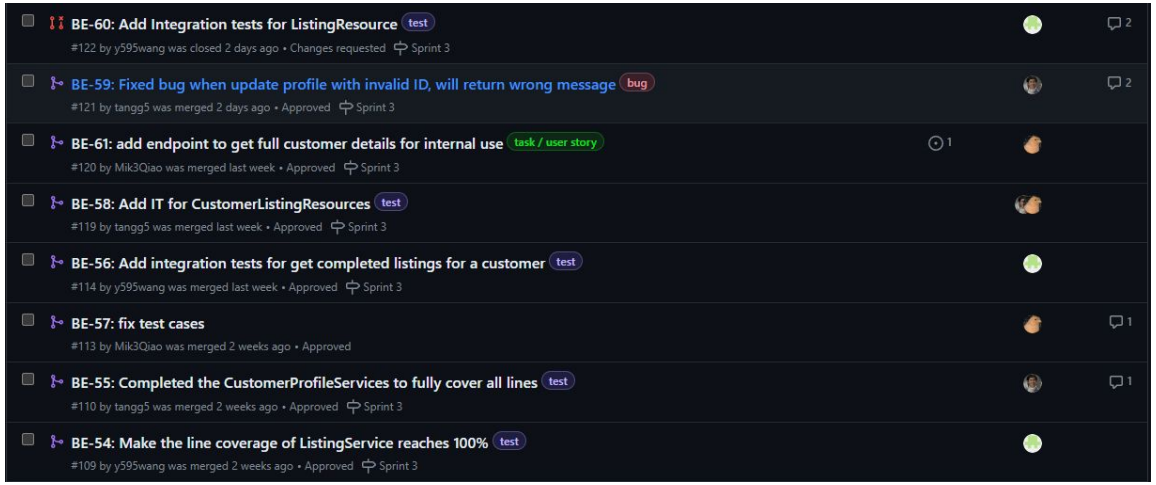
For example this specific MB ticket: Mobile

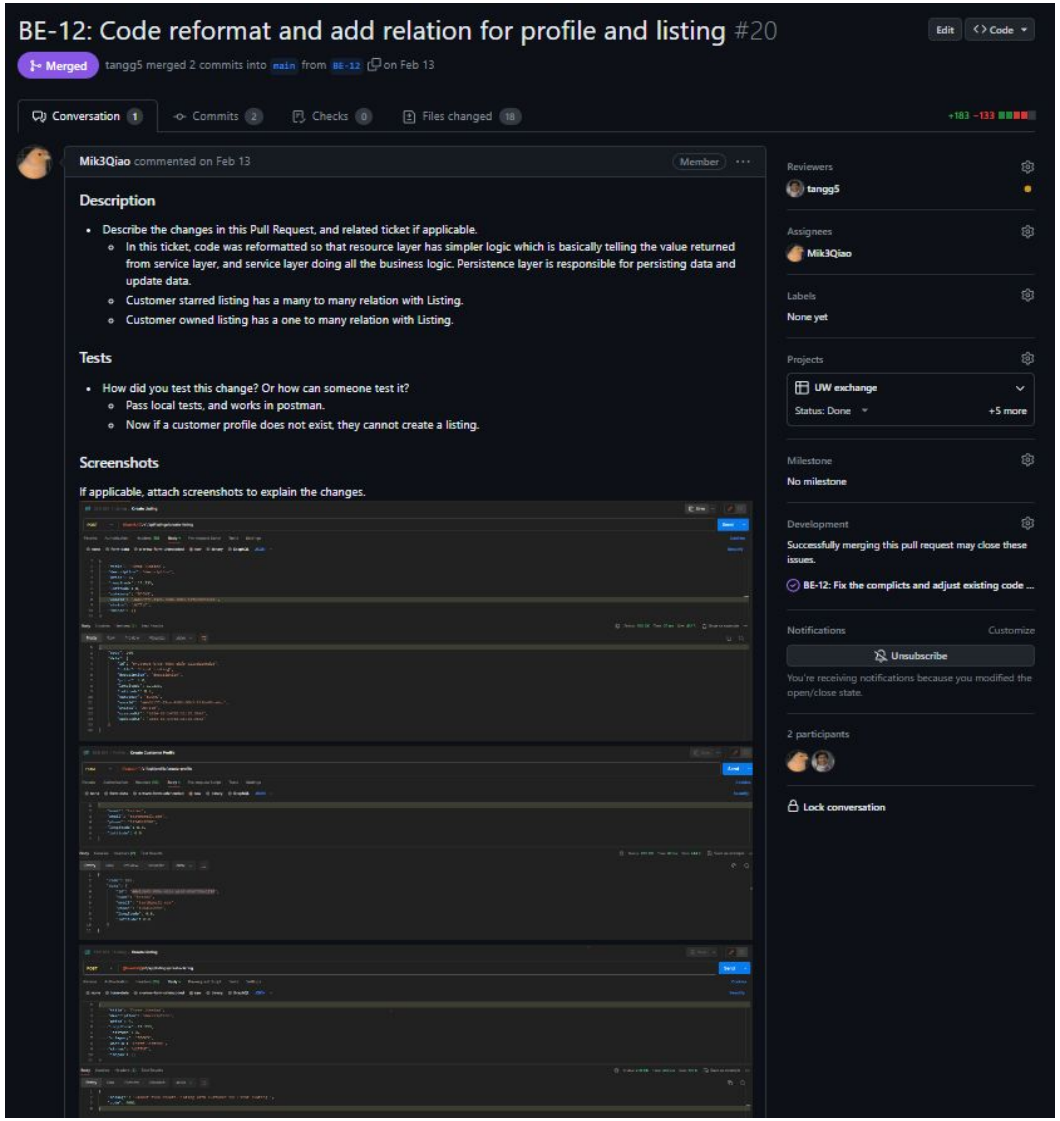
- We have a numbering system for the issues: **MB-14** is included in commit message, easy to tell which commit is related to which task
- During the team meeting, we use planning poker to point the tickets:
  - For example this one is **P1** means it's less important than **P0**, **P0** might be something like getting my postings
  - 2 points means the effort level for this function is **medium-low**
- This task is then moved to **Sprint 1**



Practices Followed – PRs

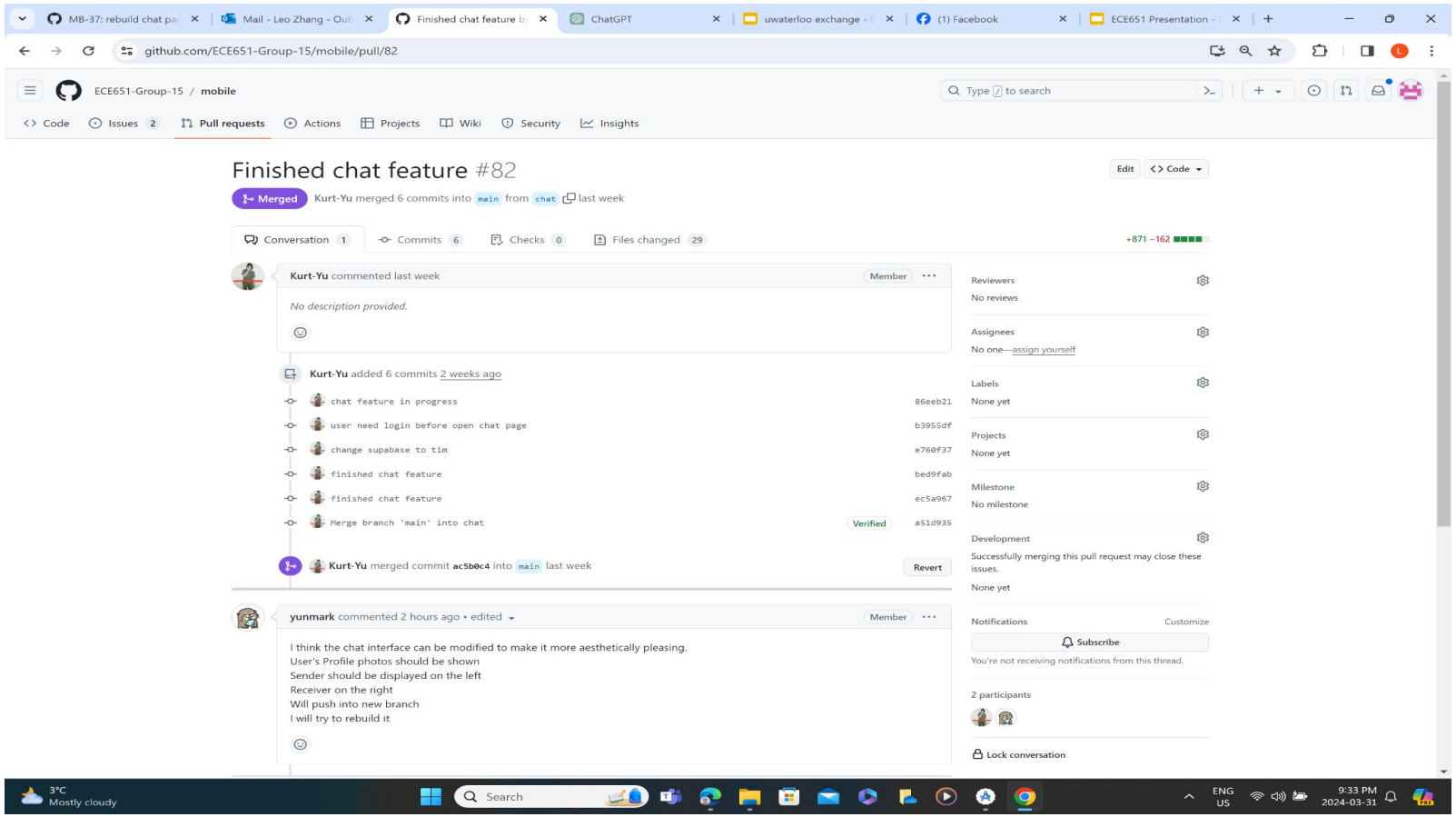
- We have a PR template which includes necessary information needed.
- Author will need to provide a brief description of the changes made
- Main branch is protected.
  - At least one Team Member from the group will need to review the changes in order to be merged to main





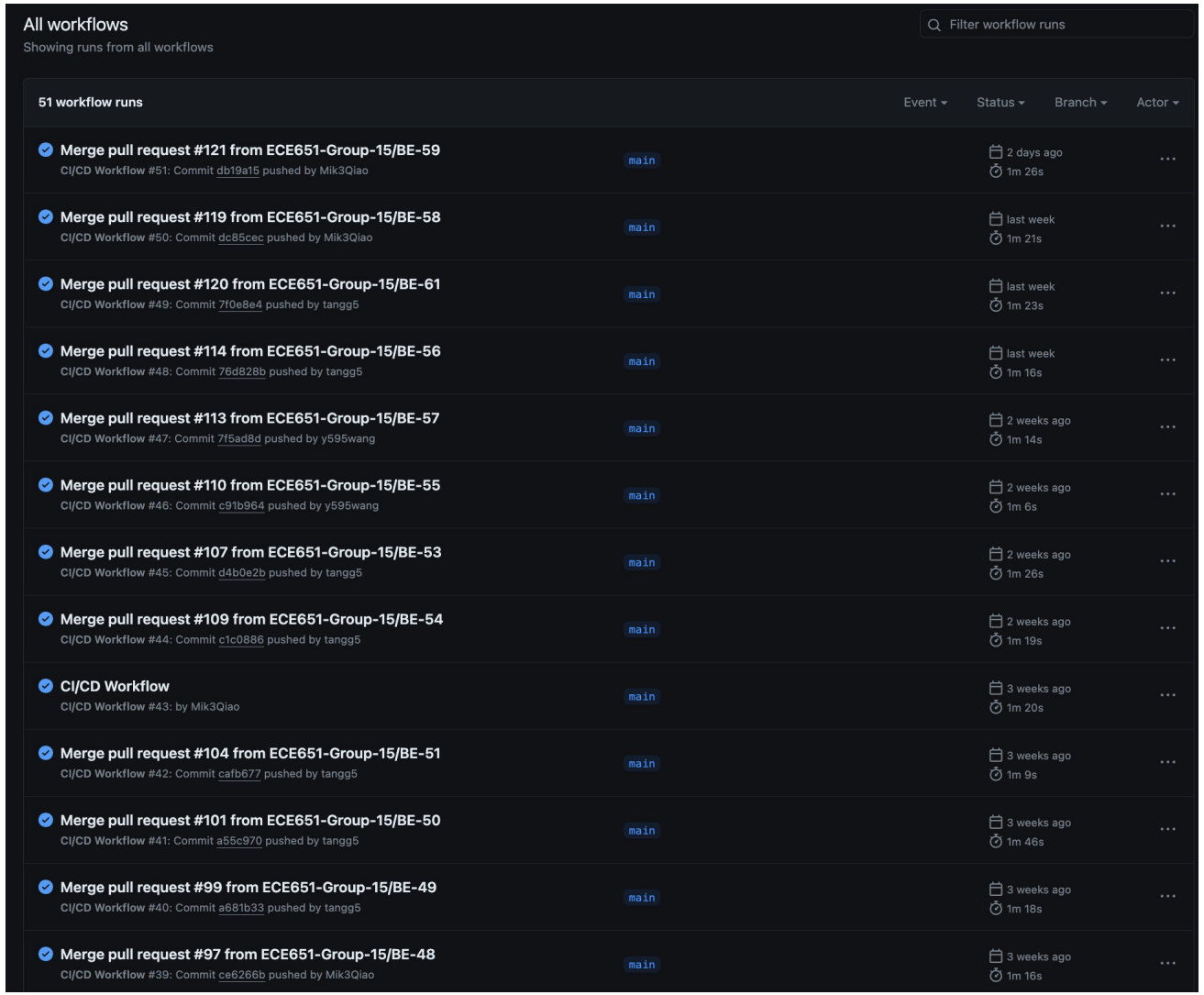
PRs Continued

Code Review Example



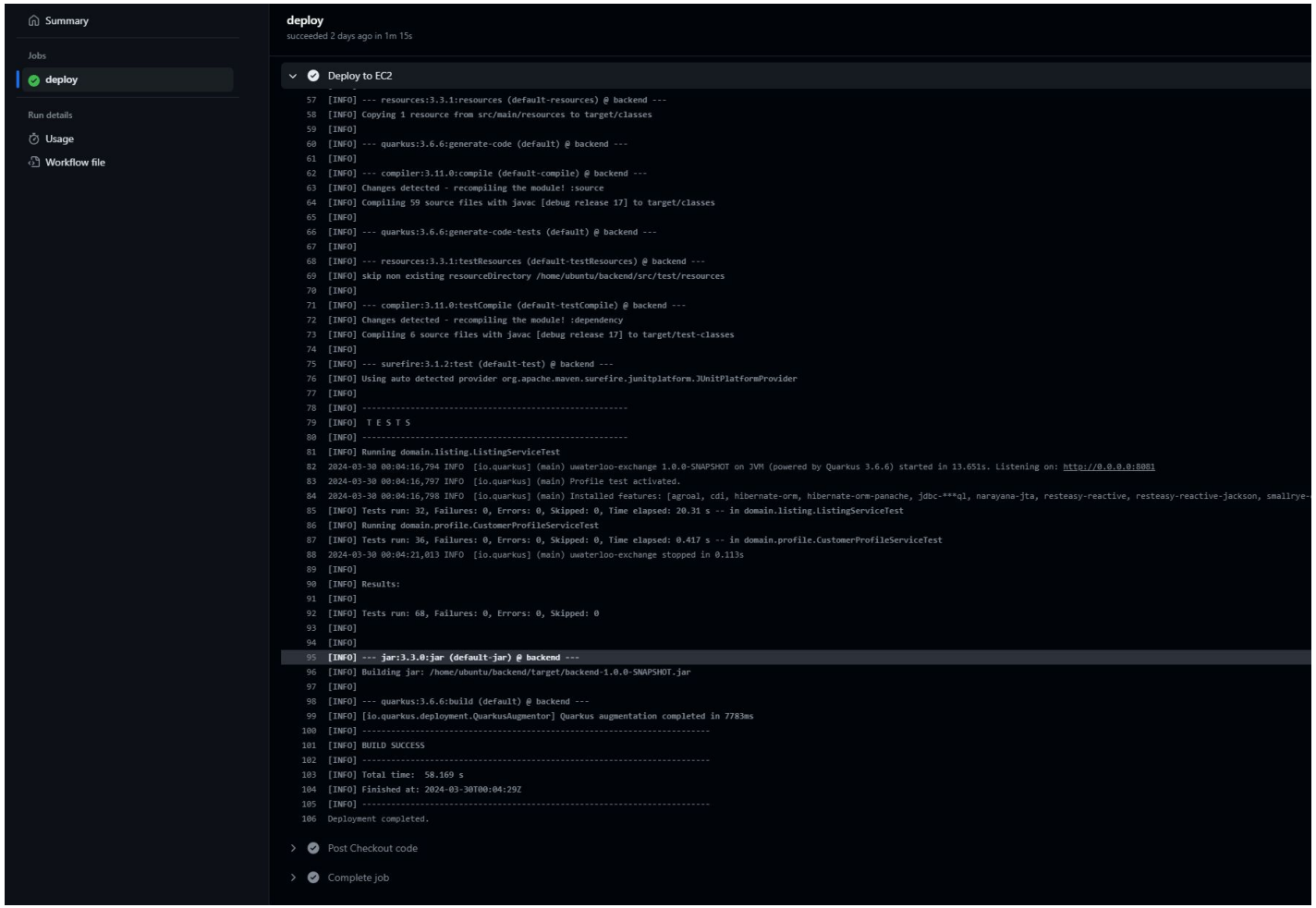
Practices Followed – CI/CD

- Realized via:
  - Github action
  - Github Secrets which stores the **Sensitive information** that was intentionally hidden from the user
  - AWS EC2 instance which runs our application



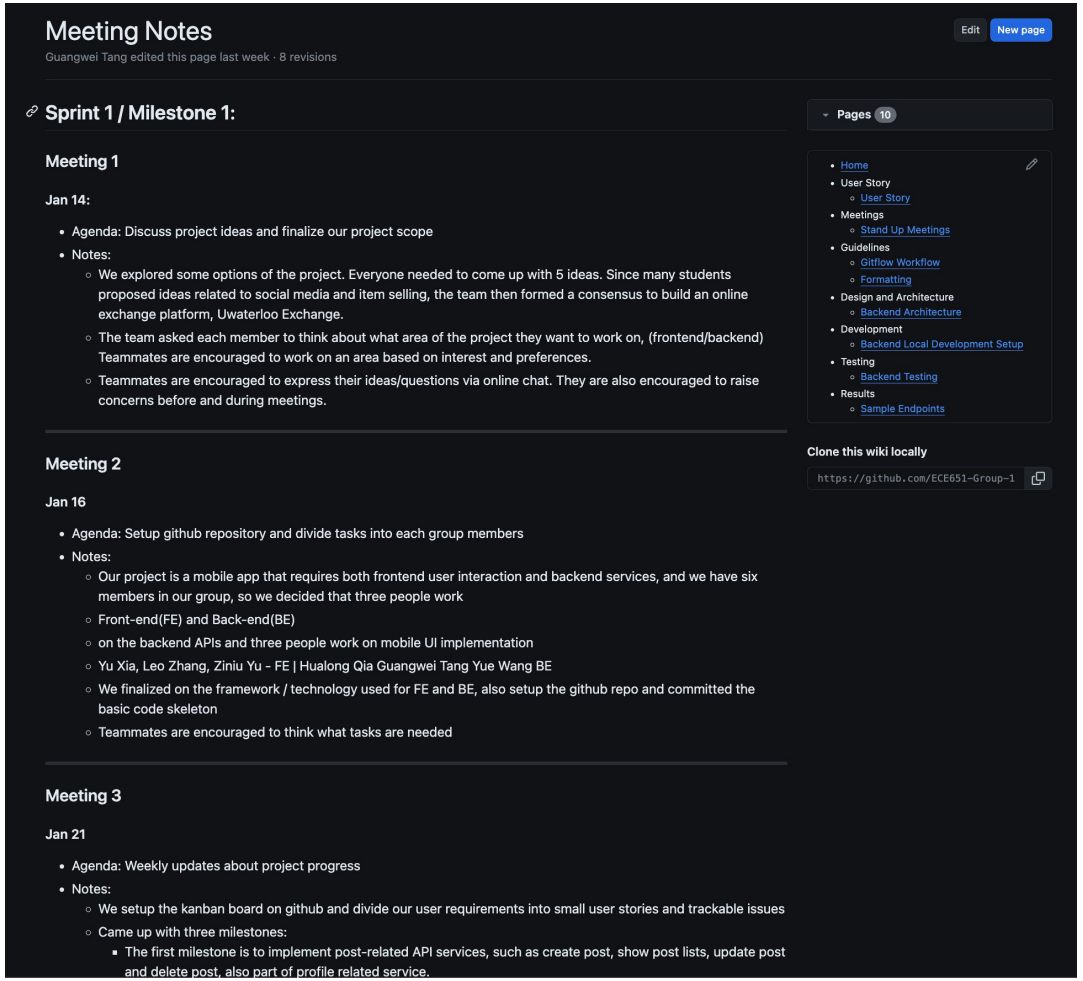
Practices Followed – CI/CD

- Github actions
  - It is triggered every time a PR got merged to **main**
- When a PR get merged to main:
  - Github action initialize the connection to our AWS EC2 instance
  - Pull the changes and build **while running all existing tests**
  - Replace existing running instance



Practices Followed – Team Meetings

- Online Meeting – MS Teams – Every Tuesday and Sunday
- Ad-hoc Meetings scheduled throughout the term
  - Online & in-person
  - Depends on the nature of the issue
- Topics and contents involved will be recorded in the meeting notes
- All members are actively engaged



Meeting Notes

Edit | New page

Guangwei Tang edited this page last week · 8 revisions

Sprint 1 / Milestone 1:

Meeting 1

Jan 14:

- Agenda: Discuss project ideas and finalize our project scope
- Notes:
  - We explored some options of the project. Everyone needed to come up with 5 ideas. Since many students proposed ideas related to social media and item selling, the team then formed a consensus to build an online exchange platform, Uwaterloo Exchange.
  - The team asked each member to think about what area of the project they want to work on, (frontend/backend) Teammates are encouraged to work on an area based on interest and preferences.
  - Teammates are encouraged to express their ideas/questions via online chat. They are also encouraged to raise concerns before and during meetings.

Meeting 2

Jan 16

- Agenda: Setup github repository and divide tasks into each group members
- Notes:
  - Our project is a mobile app that requires both frontend user interaction and backend services, and we have six members in our group, so we decided that three people work
  - Front-end(FE) and Back-end(BE)
  - on the backend APIs and three people work on mobile UI implementation
  - Yu Xia, Leo Zhang, Ziniu Yu - FE | Hualong Qia Guangwei Tang Yue Wang BE
  - We finalized on the framework / technology used for FE and BE, also setup the github repo and committed the basic code skeleton
  - Teammates are encouraged to think what tasks are needed

Meeting 3

Jan 21

- Agenda: Weekly updates about project progress
- Notes:
  - We setup the kanban board on github and divide our user requirements into small user stories and trackable issues
  - Came up with three milestones:
    - The first milestone is to implement post-related API services, such as create post, show post lists, update post and delete post, also part of profile related service.

Pages 10

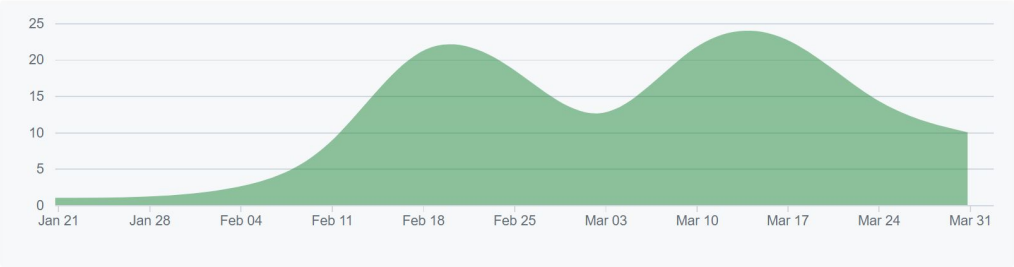
- Home
- User Story
  - User Story
- Meetings
  - Stand Up Meetings
- Guidelines
  - Gitflow Workflow
  - Formatting
- Design and Architecture
  - Backend Architecture
- Development
  - Backend Local Development Setup
- Testing
  - Backend Testing
- Results
  - Sample Endpoints

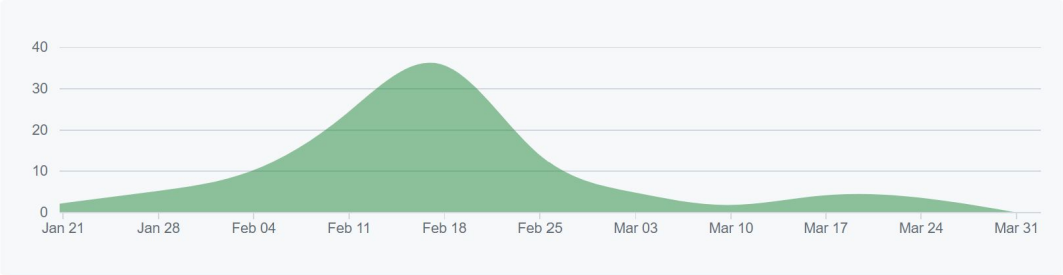
Clone this wiki locally

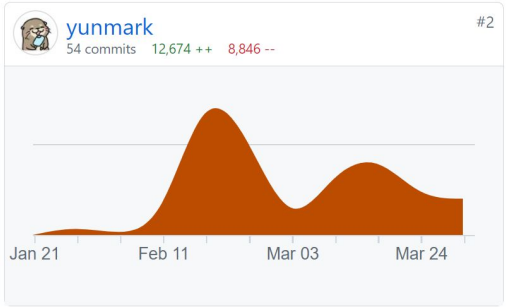
https://github.com/ECE651-Group-1

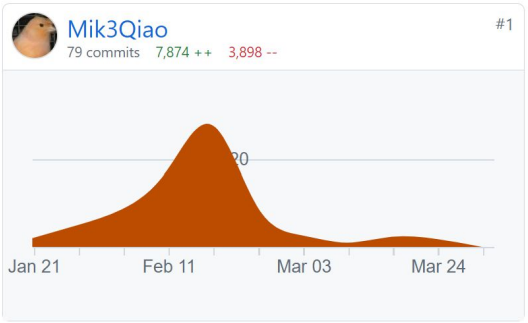
Practices Followed – Contributions

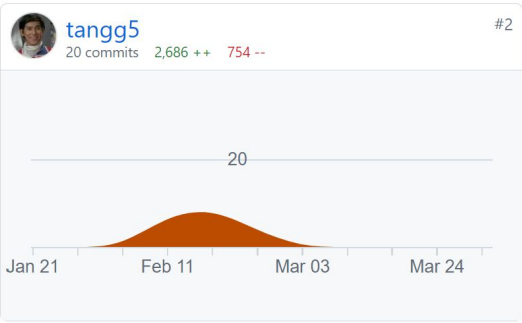
All group members were actively engaged and fairly contributed

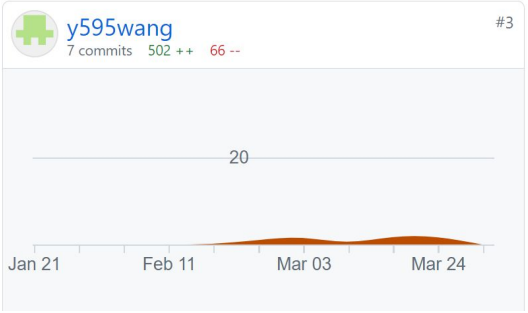












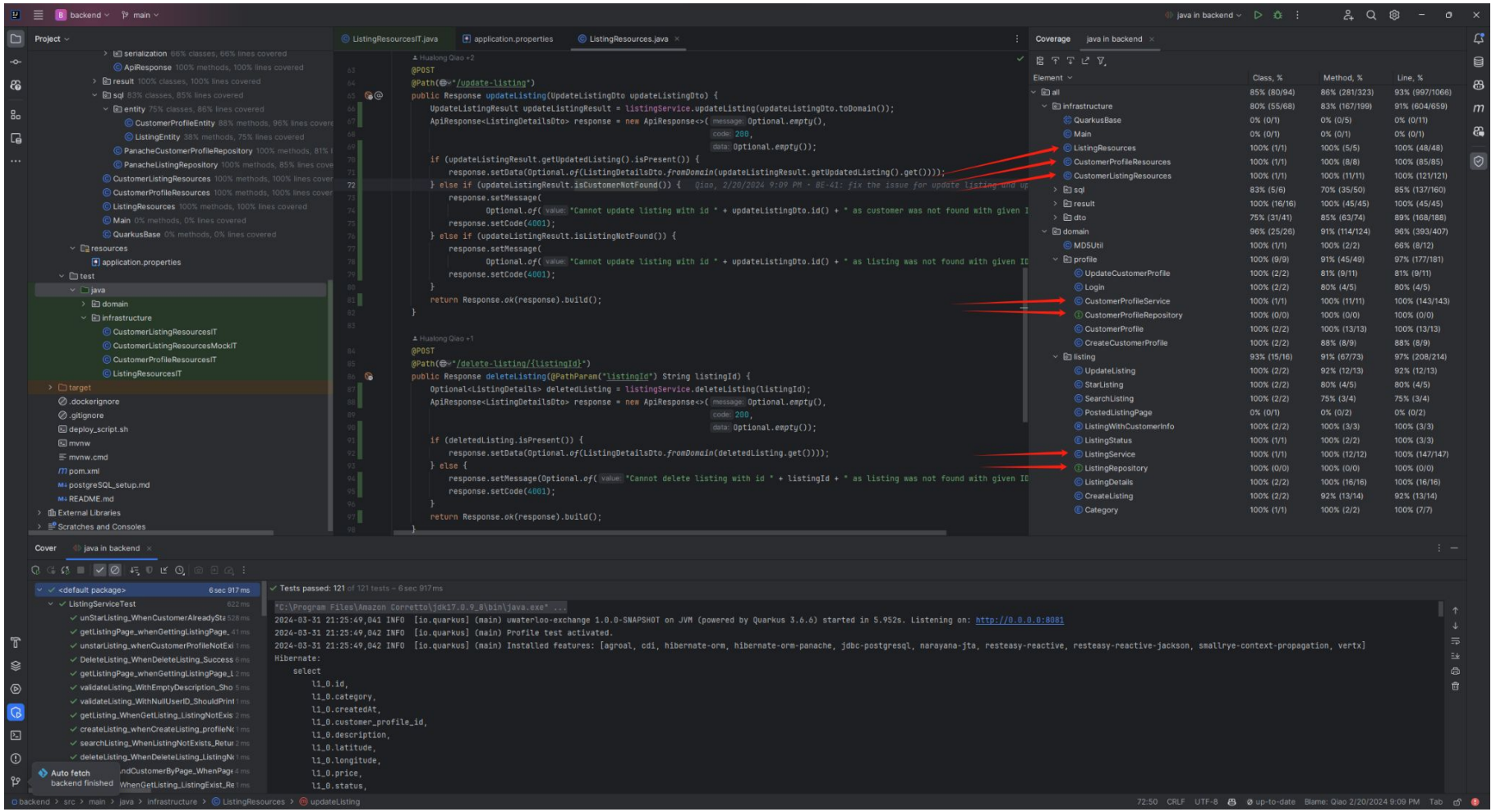
Practices Followed – Testing

Backend using onion architecture:

- Resources (api)
- Services
- Domain

Line coverage for all **Apis, services, and repositories (db operations)** are 100%

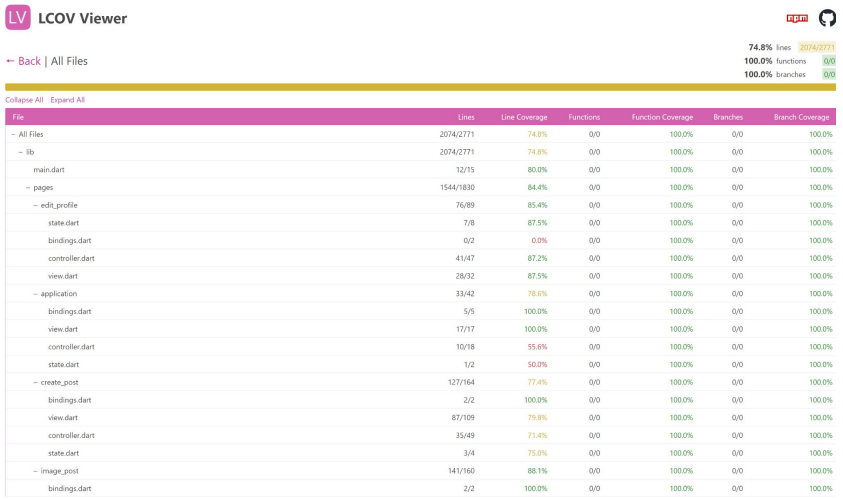
Junit, rest assured for IT

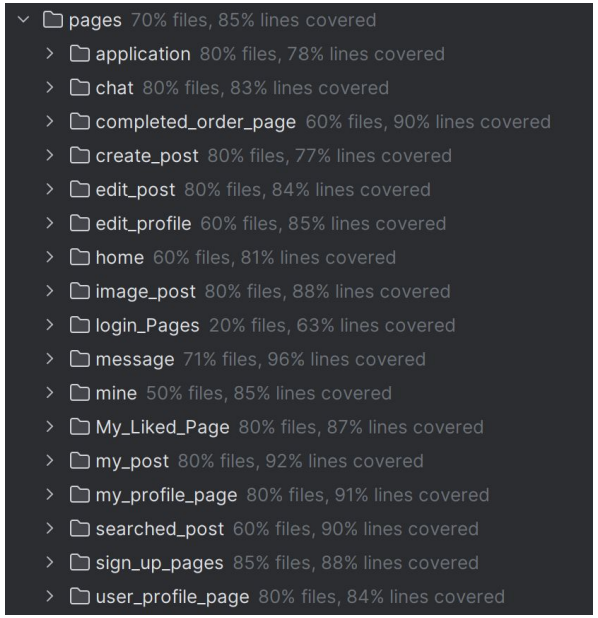


Practices Followed – Testing Continued

Frontend using flutter integration test architecture

- Simulated automated UI testing





Practices Followed – Frontend Standard

MVC principle

- increase readability
- provides modularity

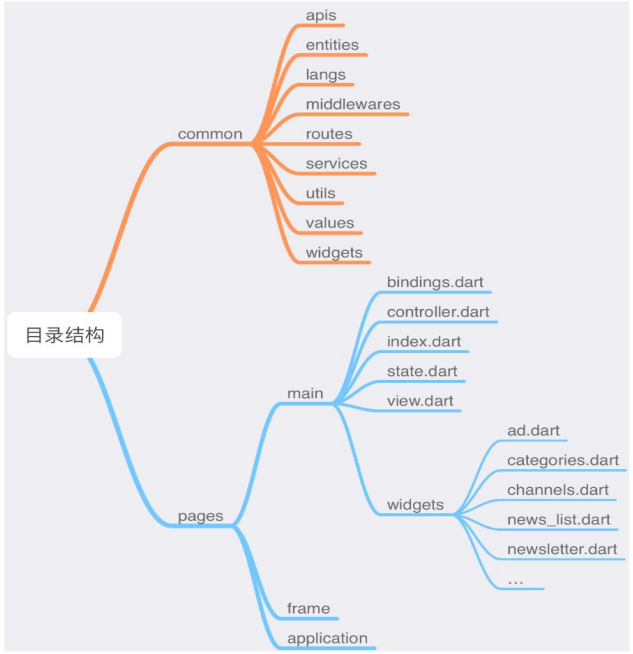
Mobile Development Wiki

Kurt edited this page on Feb 26 · 1 revision

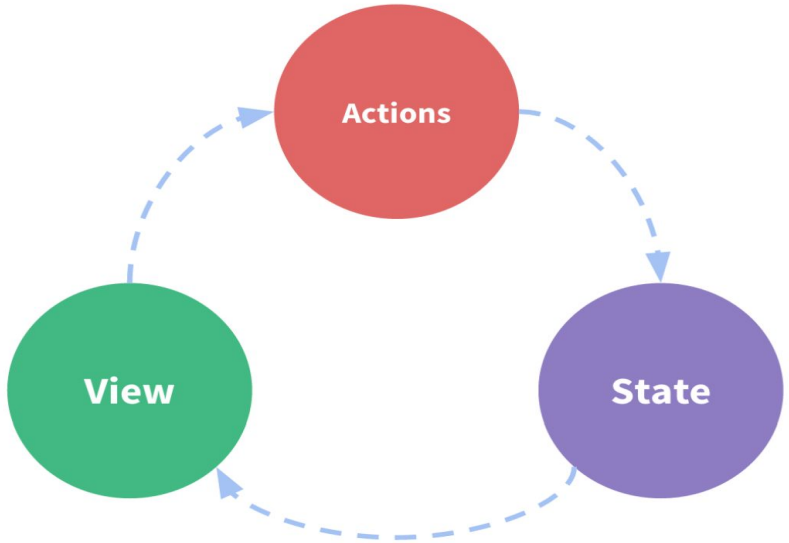
Code Structure

common folder

Name	Description
apis	http API definitions
entities	Data models
langs	multi-language support
middlewares	middleware components
routes	page route definitions
services	getx global services
utils	tool box
values	constant variables
widgets	common and reusable widgets



pages folder



Each page has the following files, based on the philosophy of redux, we divide the code into view/state/action and make our structure more readable and improves the modularity.

Name	Description
bindings.dart	data bindings
controller.dart	controller
index.dart	entrance of each page
state.dart	state variables
view.dart	view
widgets	widget components

Tech Stacks

Cross Platform mobile App

The frontend is built with Dart which provides high performance mobile UI. Runs on Android, iOS, Web, Windows, MacOS & Linux

Java Backend

Backend built with Java, Quarkus, Panache & Postgres for scalability & performance, lombok to simplify codebase. Also use onion architecture for our project

PostgreSQL database with Panache ORM

The app uses PostgreSQL for data storage and Panache for simplified database interactions.

CI/CD pipelines with GitHub Actions

The app uses GitHub Actions workflows for continuous integration and deployment to ensure quick and reliable updates.

Hosted on AWS EC2

The app backend and database runs on Amazon Web Services for scalability and reliability.

Unit & Integration Testing

JUnit 5, REST Assured to enable test driven development

The app uses a modern tech stack optimized for performance, scalability and efficient development.

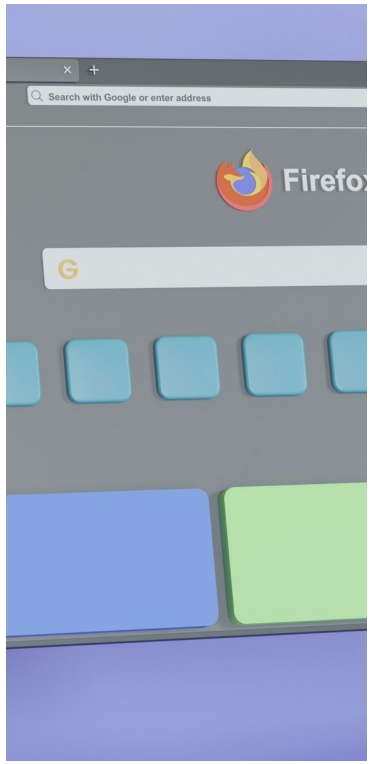
Cross-platform

Android

The app can run on Android devices.

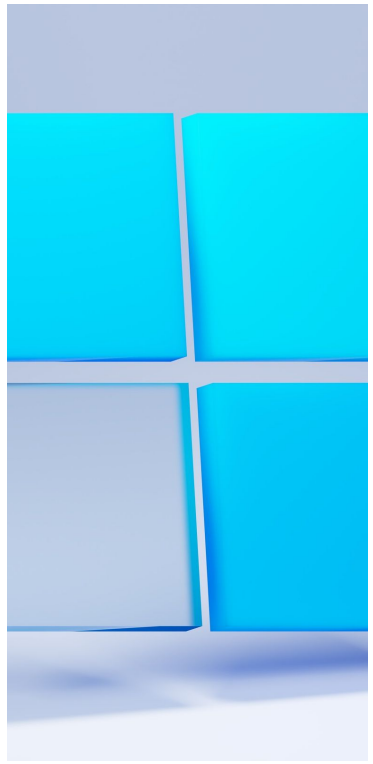
Apple

The app can run on iOS devices like iPhone and iPad.



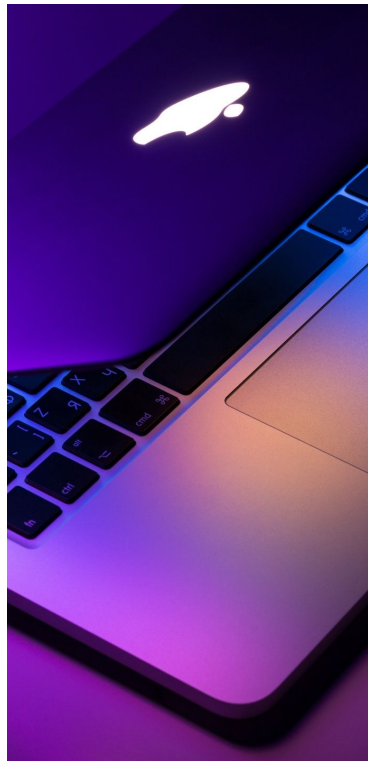
Web browser

The app can run on web browsers like Chrome, Firefox and Safari.



Windows

The app can run on Windows PCs and tablets.



MacOS

The app can run on Mac computers.

Linux

The app can run on Linux operating systems.

Project Milestones

Milestone 1 (Done)

Implement the post-related feature, e.g. user should be able to create/update/delete/read posts

Milestone 2 (Done)

Implement user authentication and authorization

Milestone 3(Done)

Implement the real-time messaging functionality, e.g. user should be able to send message to other users for real-time communication

Project Management

- **Project feature and user story breakdown:**

  https://github.com/orgs/ECE651-Group-15/projects/2

- **Backend issue tracking:**

  https://github.com/ECE651-Group-15/backend/issues

- **Frontend issue tracking:**

  https://github.com/ECE651-Group-15/mobile/issues

- **Kanban Board:**

  https://github.com/orgs/ECE651-Group-15/projects/2/views/1

- **Weekly meeting records:**

  https://github.com/ECE651-Group-15/backend/wiki/Meeting-Notes