Report Title: Agile Report for The Smart Living Community Project

Course Code: CSE 404
Course Title: Software Engineering and ISD Laboratory

Submitted by

SHANJIDA ALAM(ID: 353)

Submitted to

Dr. Md. MUSHFIQUE ANWAR, Professor

Dr. Md. HUMAYUN KABIR, Professor

Computer Science and Engineering

Jahangirnagar University

Dhaka, Bangladesh

January 07, 2025

Contents

Report Title : Sprint 1 Process for The Smart Living Community Project

Course Code: CSE 404

Course Title: Software Engineering and ISD Laboratory

Submitted by

SHANJIDA ALAM(ID: 353)

Submitted to

Dr. Md. MUSHFIQUE ANWAR, Professor

Dr. Md. HUMAYUN KABIR, Professor

Computer Science and Engineering

Jahangirnagar University

Dhaka, Bangladesh

January 07, 2025

Contents

Chapter 1

Introduction

This report outlines the implementation of the **Sprint 1** process within our **Agile** development framework, describing the key activities, outcomes, and recommendations for future sprints. The primary focus was on establishing the foundational Scrum processes and delivering a working iteration of the project with integrated unit testing.
The adoption of Agile Scrum methodology aims to enhance our team's ability to respond to changing project requirements while maintaining consistent delivery of working software project. This report documents our first sprint implementation and its outcomes.This report also documents my personal involvement, tasks completed, and deliverables produced during the sprint.

Chapter 2

Sprint 1 Objectives

- To implement and experience the Scrum framework within a one-week Sprint.
- To establish clear roles, responsibilities, and workflows for the team.
- To create and maintain essential Scrum artifacts, including the project backlog and Sprint backlog.
- To conduct effective Sprint ceremonies, such as planning and daily Scrum meetings.
- To incorporate unit testing as part of the development process.
- To document team activities, progress, and challenges for future reference.

Chapter 3

Sprint 1 Planning

3.1 Sprint 1 Meeting Date

Date, Duration and Location: 24-OCTOBER-2024, 10:30 AM, 1 Hour 00 Minutes, CSE ROOM 203.

3.2 Sprint 1 Attendees

There are six attendess present in the meeting:

- Solaimi Hamid (SH)
- Shanjida Alam (SA)
- Irtifa Haider (IH)
- Hasneen Tamanna (HT)
- Md. Tanvir Hossain Saon (TH)
- Jubaer Ahmad Khan (JK)

3.3 Scrum Roles

- **Scrum Master:** Jubaer Ahmad Khan (JK)
- **Product Owners:** Shanjida Alam (SA)
- **Scrum Team Member:** Solaimi Hamid (SH), Irtifa Haider (IH), Hasneen Tamanna (HT), Md. Tanvir Hossain Saon (TH)

3.4 Sprint 1 Goal

- Gain a solid understanding of Android components and their features, focusing on navigation (between activities and fragments).
- Learn how to connect XML UI components with Java classes.
- Build a simple note-taking app following the MVVM architecture pattern integrated with Room Database.

3.5 Product Backlog

The Product Backlog is created by the Product Owner. It is a prioritized list of all the key features and functionalities that the team will work on during the product's life cycle. These features are not necessarily executed within a single sprint, but rather serve as a road map for the entire product development process.In the given below I provide the product backlog:

- Registration
- Login
- Manage Profile
- Access Dashboard
- Manage Service Request
- Create Event
- Create Bill
- Submit Complaints
- Create Parking Request
- Create Security Log
- Manage Directory
- Create Community Bulletin Board

3.6 Sprint 1 Backlog

The Scrum Master selected six features that were completed during Sprint 1. They are:

- Registration done by Hasneen Tamanna (HT)

- Login done by Jubaer Ahmad Khan (JK)
- Manage Profile done by Shanjida Alam (SA)
- Access Dashboard done by Md. Tanvir Hossain Saon (TH)
- Manage Service Request done by Solaimi Hamid (SH)
- Create event done by Irtifa Haider (IH)

3.7 Tools Used

- **Trello:** Task Management.
- **Discord:** Daily scrum meeting and communication with each other during Sprint 1.
- **Toggle:** Time management.

Chapter 4

Sprint 1 Execution

4.1 Daily Scrum Meeting

- **Daily Scrum Meeting 1:**

What we did yesterday?	What problems faced?	What will do today?
Created the resident profile interface, set up a new branch and started planning upcoming feature development	Flow of navigation graph	Update SRS

Here I only mention my part of the daily scrum meeting.

- **Daily Scrum Meeting 2:**

What we did yesterday?	What problems faced?	What will do today?
Updated SRS, Modified the Resident Profile UI Page	None	Will create Secretary Profile page, create Manager profile page

Here I only mention my part of the daily scrum meeting.

- **Daily Scrum Meeting 3:**

What we did yesterday?	What problems faced?	What will do today?
Created Secretary Profile Page,Created Manager Profile Page,Connected to Firebase	None	Will attempt to fetch data from the database

Here I only mention my part of the daily scrum meeting.

- **Daily Scrum Meeting 4:**

What we did yesterday?	What problems faced?	What will do today?
Successfully fetched data from the database and displayed it in the user interface, implemented the manage profile feature within the navigation component,generated documentation for the manage profile feature.	Encountered challenges while fetching data from the database, faced difficulties in updating data within the database.	Conduct unit testing to ensure the functionality is working as expected,separate the profile interface based on user roles.

Here I only mention my part of the daily scrum meeting.

4.2 My Contribution during This Sprint 1

During Sprint 1, I worked on the Manage Profile feature. I began by creating the XML layout for the user interface. Once completed, I shared this layout in the Discord channel with my teammates to gather valuable feedback on this initial component.Here I include attachment about that,

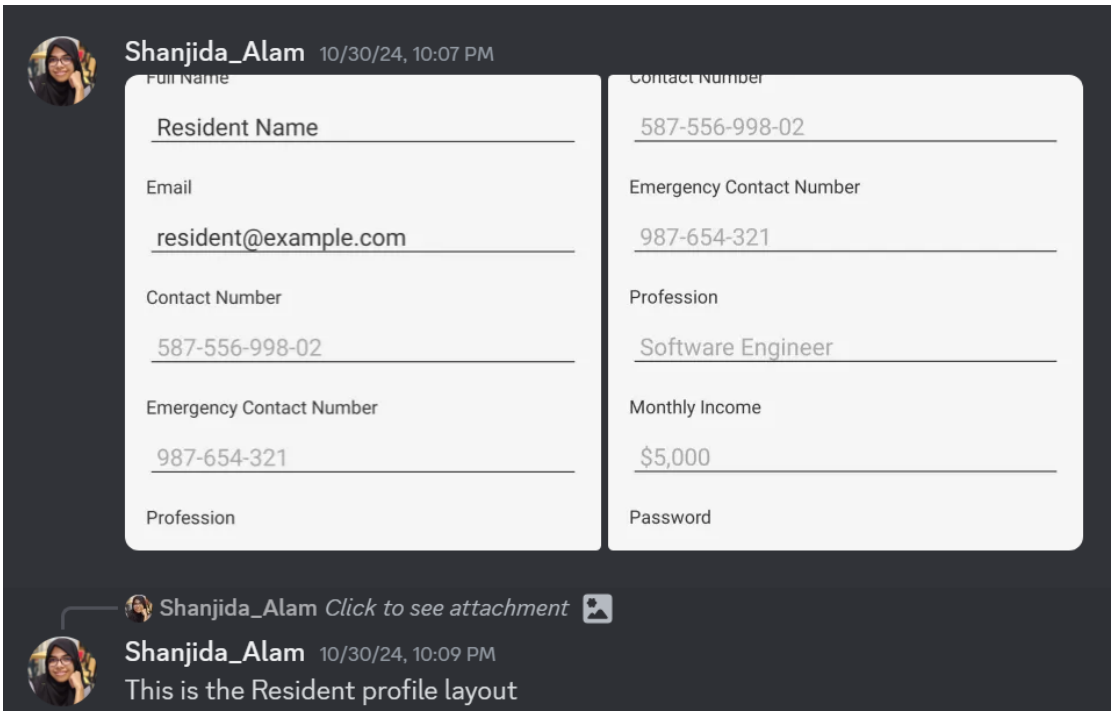


Figure 4.1: This is screenshot of my document.

After finalizing the UI design, I proceeded with the backend implementation, ensuring the codebase is well-structured and maintainable. To achieve this, I implemented the **Model-View-ViewModel (MVVM)** architectural pattern. The project's file architecture is provided for reference.

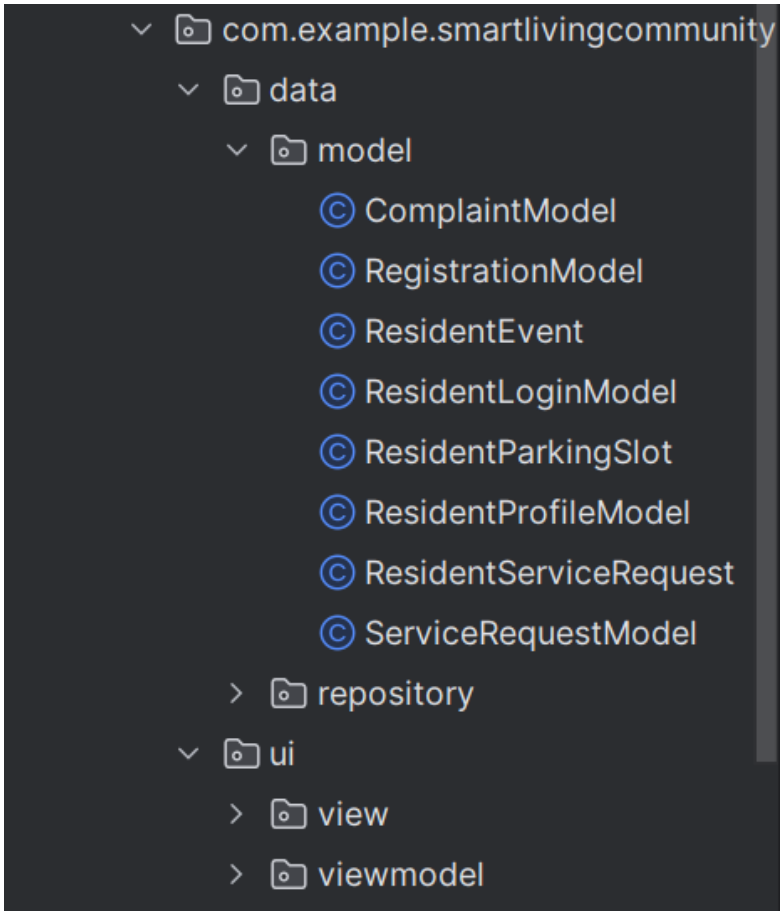
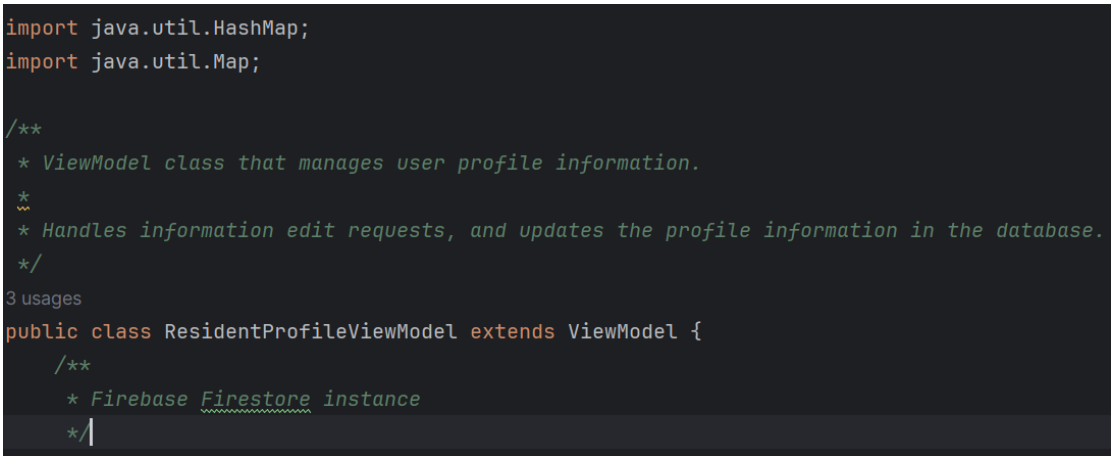


Figure 4.2: This is screenshot of MVVM architecture layout. Here, **model** contains the data and business logic, **view** displays the data and provides user interaction and binds to the **viewmodel**,**ViewModel** is an abstraction of the View, holding the logic for the View.

Next, I integrated the code with the Firebase Datastore to enable data retrieval and storage from the database. For your reference, I've attached the relevant code snippet.



```
import java.util.HashMap;
import java.util.Map;

/**
 * ViewModel class that manages user profile information.
 *
 * Handles information edit requests, and updates the profile information in the database.
 */
3 usages
public class ResidentProfileViewModel extends ViewModel {
    /**
     * Firebase Firestore instance
     */
```

Figure 4.3: This is the screenshot of the connection of the firebase datastore.

I dedicated significant effort to ensure the full implementation was completed within the deadline. To give a clear picture of my progress and time management, I've attached screenshots of my Git Bash activity along with my Toggl Track time entries. These provide an overview of the work done and the time invested in the project.

4.3 Visual Aspect of Git Bash Activity for Manage Profile

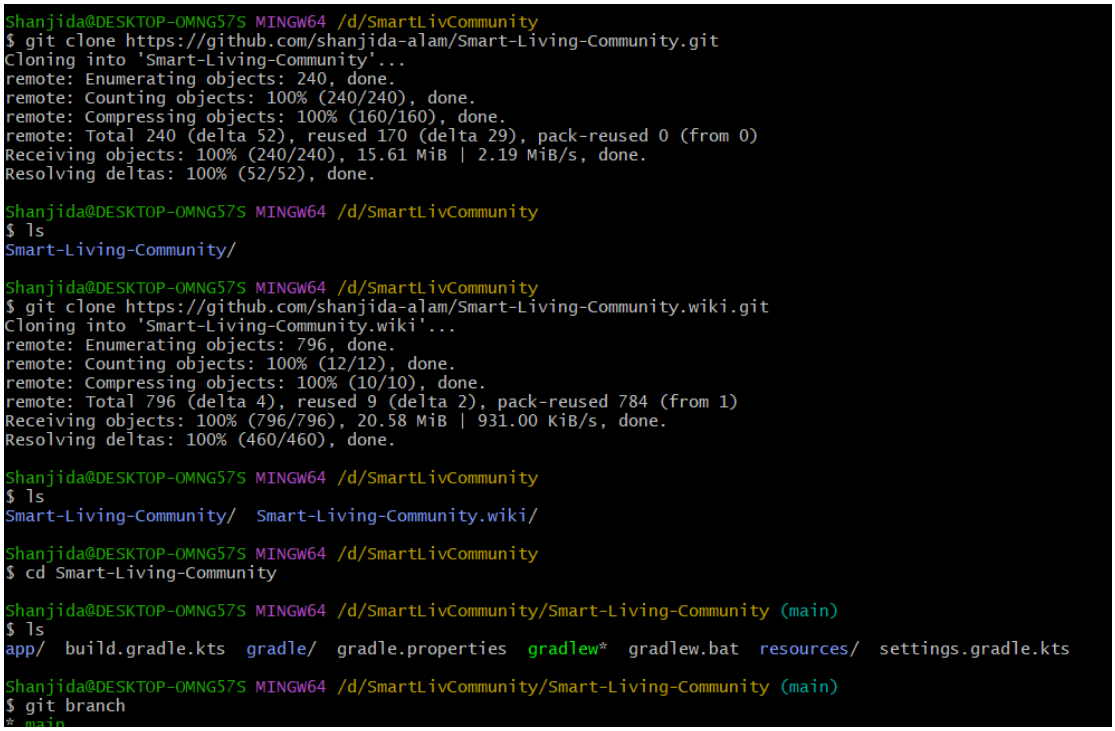


Figure 4.4: This is the screenshot of cloning the **'Smart-Living-Community'** into the local machine.

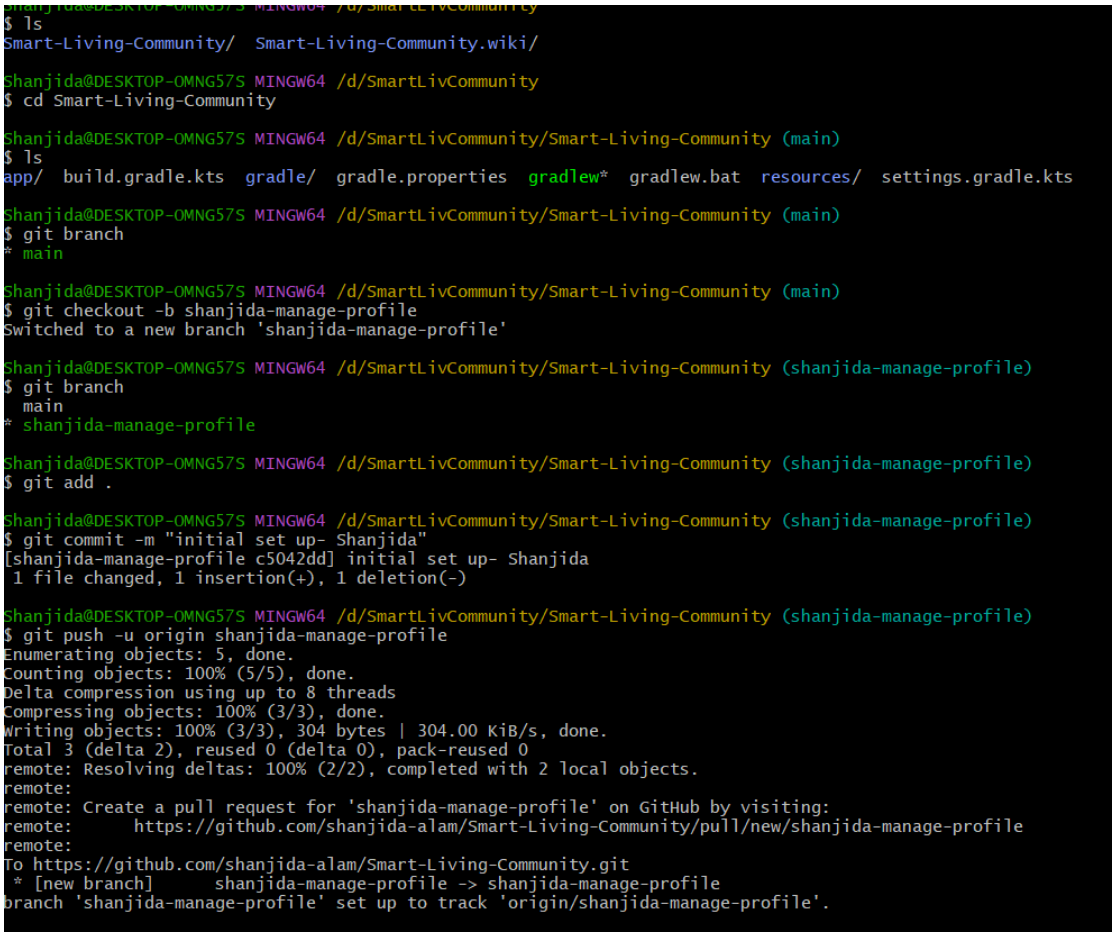


Figure 4.5: The screenshot describes the initial steps in my development workflow for the **Manage Profile** feature. It represents the creation of a new branch named **'shanjida-manage-profile'** on GitHub. Sequentially, I set up the local development environment and pushed the initial codebase to this newly created branch.

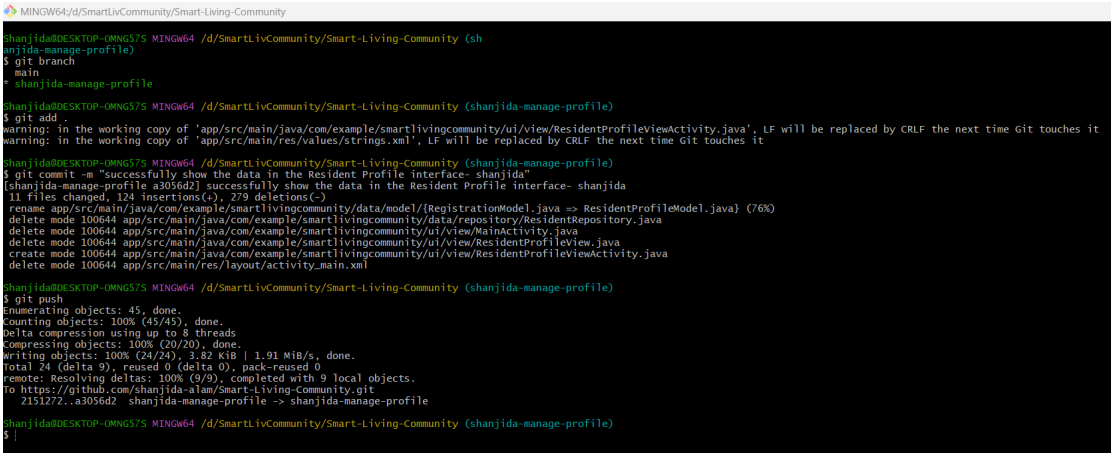


Figure 4.6: This screenshot illustrates the successful push of the code that displays resident profile data in the user interface.

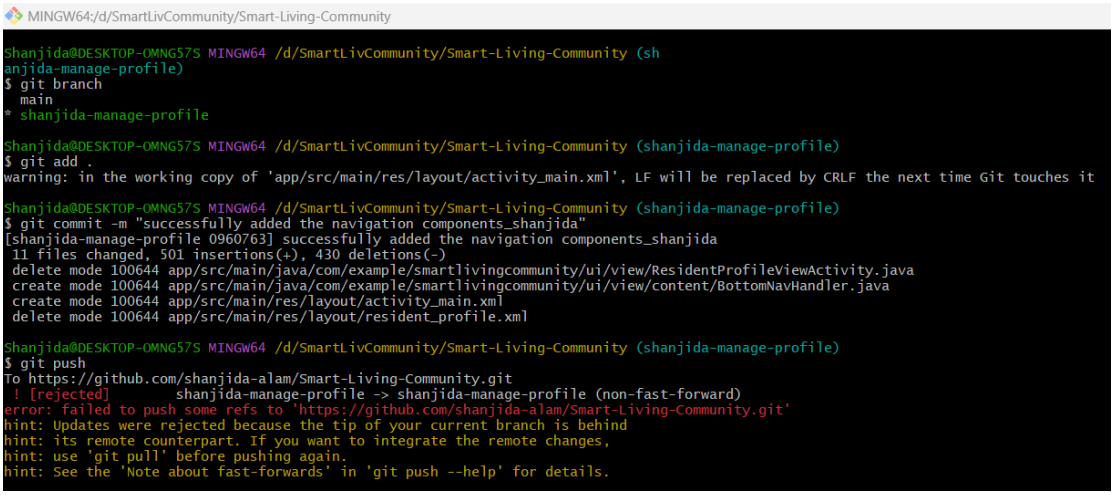


Figure 4.7: This screenshot illustrates the successful push of the code that added the navigation components. However, during the push, a merge conflict arose, which I successfully resolved.

Figure 4.8: This screenshot shows the merge conflict that occurred during the push.

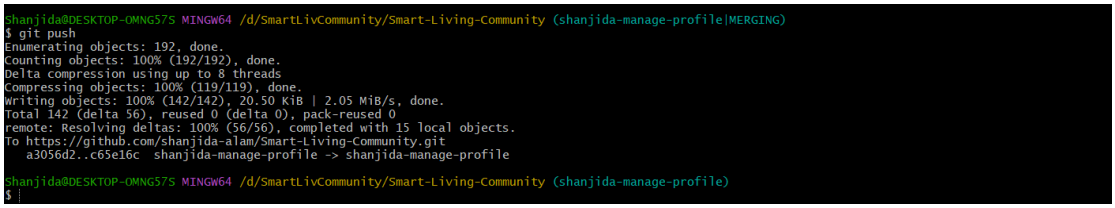


Figure 4.9: This screenshot shows that I successfully resolve the conflict.

Figure 4.10: This screenshot describes the successful push of the code that check the after logging the app is redirected the dashboard and other components.

```
Shanjida@DESKTOP-OMNG57S MINGW64 /d/SmartLivCommunity/Smart-Living-Community (shanjida-manage-profile)
$ git add .

Shanjida@DESKTOP-OMNG57S MINGW64 /d/SmartLivCommunity/Smart-Living-Community (shanjida-manage-profile)
$ git commit -m "remove the resident_profile.xml file_shanjida"
[shanjida-manage-profile 62d38d5] remove the resident_profile.xml file_shanjida
 1 file changed, 288 deletions(-)
 delete mode 100644 app/src/main/res/layout/resident_profile.xml

Shanjida@DESKTOP-OMNG57S MINGW64 /d/SmartLivCommunity/Smart-Living-Community (shanjida-manage-profile)
$ git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 590 bytes | 590.00 KiB/s, done.
Total 7 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/shanjida-alam/Smart-Living-Community.git
   0ab9859..62d38d5  shanjida-manage-profile -> shanjida-manage-profile
```

Figure 4.11: This screenshot describes the successful push of the code that remove the `resident_profile.xml` for the coding purpose.

```
MINGW64:/d/SmartLivCommunity/Smart-Living-Community

Shanjida@DESKTOP-OMNG57S MINGW64 /d/SmartLivCommunity/Smart-Living-Community (sh
anjida-manage-profile)
$ git branch
  main
* shanjida-manage-profile

Shanjida@DESKTOP-OMNG57S MINGW64 /d/SmartLivCommunity/Smart-Living-Community (shanjida-manage-profile)
$ git add .

Shanjida@DESKTOP-OMNG57S MINGW64 /d/SmartLivCommunity/Smart-Living-Community (shanjida-manage-profile)
$ git commit -m "generated documentation_shanjida"
[shanjida-manage-profile 6cf91b4] generated documentation_shanjida
 6 files changed, 234 insertions(+), 26 deletions(-)

Shanjida@DESKTOP-OMNG57S MINGW64 /d/SmartLivCommunity/Smart-Living-Community (shanjida-manage-profile)
$ git push
Enumerating objects: 111, done.
Counting objects: 100% (106/106), done.
Delta compression using up to 8 threads
Compressing objects: 100% (37/37), done.
Writing objects: 100% (43/43), 6.60 KiB | 232.00 KiB/s, done.
Total 43 (delta 20), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (20/20), completed with 14 local objects.
To https://github.com/shanjida-alam/Smart-Living-Community.git
   62d38d5..6cf91b4  shanjida-manage-profile -> shanjida-manage-profile

Shanjida@DESKTOP-OMNG57S MINGW64 /d/SmartLivCommunity/Smart-Living-Community (shanjida-manage-profile)
$
```

Figure 4.12: This screenshot shows the successful push of the code responsible for generating the project documentation.

4.4 Visual Aspect of Toggl Track for Manage Profile

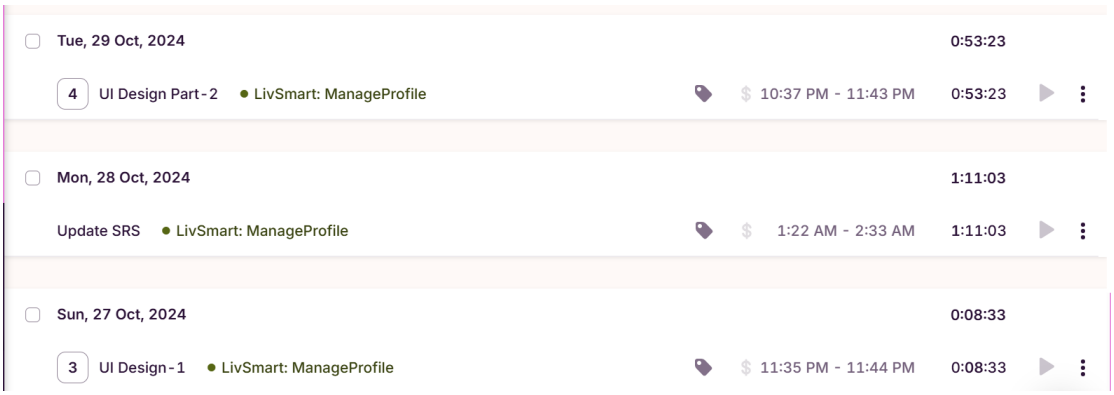


Figure 4.13: This image shows the time-tracking data for the **Manage Profile** feature. It displays three different work sessions across consecutive days in October 2024. The total duration of these three sessions is 2 hours 12 minutes and 59 seconds.

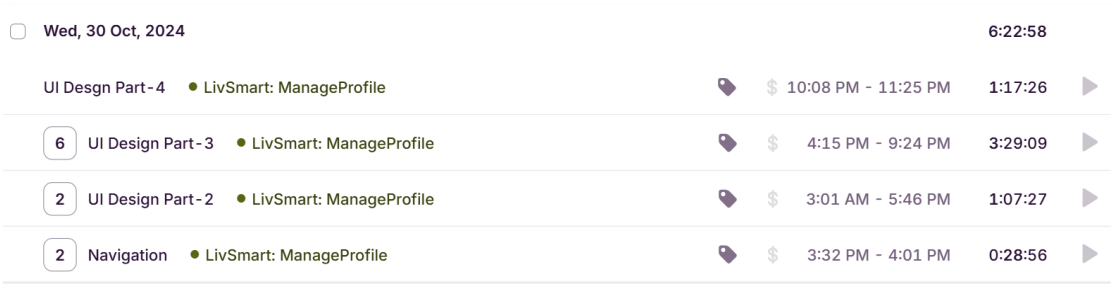


Figure 4.14: This image displays time-tracking data for the **Manage Profile** feature. It shows four different work sessions completed on Wednesday, 30th October 2024. The sessions are labeled as **UI Design Part-4, UI Design Part-3, UI Design Part-2** and **Navigation** with respective durations of 1 hour 17 minutes and 26 seconds, 3 hours 29 minutes and 9 seconds, 1 hour 7 minutes and 27 seconds and 28 minutes and 56 seconds. The total time spent on this day is 6 hours 22 minutes and 58 seconds.

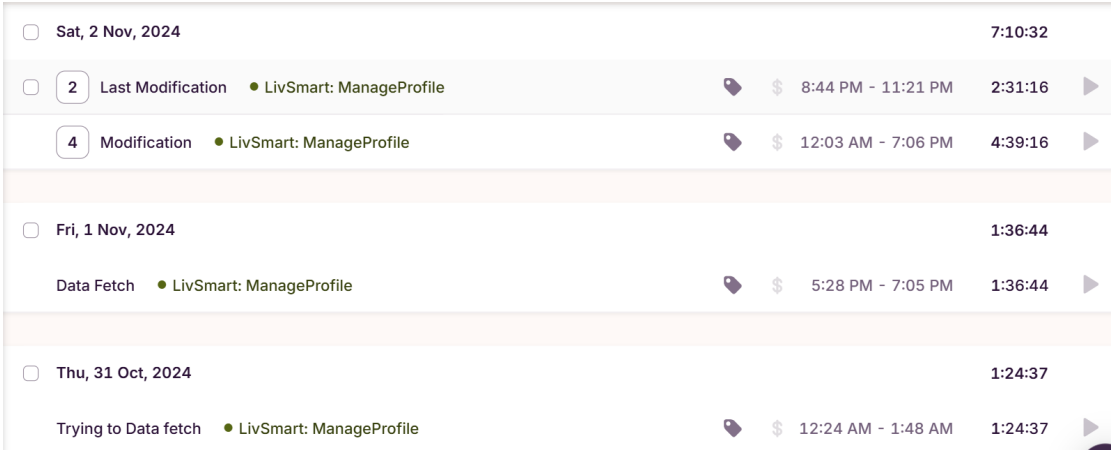


Figure 4.15: This image displays detailed time-tracking data for the **Manage Profile** feature across three consecutive days: 31st October 2024, 1st November 2024 and 2nd November 2024.The total duration of these three days is 10 hours 11 minutes and 53 seconds.

I spent a total of 18 hours 47 minutes and 50 seconds completing the **Manage Profile** feature.This valuable tool helped me track how much time I spent on the project and provided clear data about my contribution to it.

Chapter 5

Conclusion

Sprint 1 provided valuable insights into the Scrum process, emphasizing collaboration, adaptability, and delivering a working product within a constrained timeline. The lessons learned will inform improvements for future Sprints.The implementation of Sprint 1 has established a foundation for Agile development practices within the team.Upon successful execution of Sprint 1, the Manage Profile feature was delivered.

Report Title : Sprint 2 Process for The Smart Living Community Project

Course Code: CSE 404

Course Title: Software Engineering and ISD Laboratory

Submitted by

SHANJIDA ALAM(ID: 353)

Submitted to

Dr. Md. MUSHFIQUE ANWAR, Professor

Dr. Md. HUMAYUN KABIR, Professor

Computer Science and Engineering

Jahangirnagar University

Dhaka, Bangladesh

January 07, 2025

Contents

Chapter 1

Introduction

This report represents the implementation of **Sprint 2**, focusing on process improvements identified during Sprint 1's retrospective, the integration of Test-Driven Development (TDD), and continuous integration testing. Key goal is placed on improvements to our Agile methodology based on previous sprint learning.

Sprint 2 expands on the feedback and lessons learned from the previous Sprint's review and retrospective. The primary goal is to improve both the product and the development process by implementing focused enhancements. This report details my involvement in **Sprint 2**, including updates to the **product backlog**, **Sprint planning** activities, and execution steps. It also highlights the adoption of **Test-Driven Development (TDD)** and **continuous integration testing** to ensure an efficient and high-quality development workflow.

Chapter 2

Sprint 2 Objectives

- To refine the Scrum process by implementing improvements identified during Sprint 1's retrospective.
- To update and prioritize the product backlog based on feedback and changing project requirements.
- To plan and execute a new Sprint, ensuring that all tasks are manageable within the one-week time frame.
- To practice TDD and integrate continuous integration testing into the development process.
- To incorporate unit testing as part of the development process.
- To document team activities, progress, and challenges for future reference.

Chapter 3

Sprint 2 Planning

3.1 Sprint 2 Meeting Date

Date, Duration and Location: 03-November-2024, 10:30 AM, 30 Minutes, CSE ROOM 203.

3.2 Sprint 2 Attendees

There are six attendess present in the meeting:

- Solaimi Hamid (SH)
- Shanjida Alam (SA)
- Irtifa Haider (IH)
- Hasneen Tamanna (HT)
- Md. Tanvir Hossain Saon (TH)
- Jubaer Ahmad Khan (JK)

3.3 Scrum Roles

- **Scrum Master:** Jubaer Ahmad Khan (JK)
- **Product Owners:** Shanjida Alam (SA)
- **Scrum Team Member:** Solaimi Hamid (SH), Irtifa Haider (IH), Hasneen Tamanna (HT), Md. Tanvir Hossain Saon (TH)

3.4 Sprint 2 Goal

- Gain a solid understanding of Android components and their features, focusing on navigation (between activities and fragments).
- Learn how to connect XML UI components with Java classes.
- Build a simple note-taking app following the MVVM architecture pattern integrated with Room Database.

3.5 Product Backlog

The Product Backlog is created by the Product Owner. It is a prioritized list of all the key features and functionalities that the team will work on during the product's life cycle. These features are not necessarily executed within a single sprint, but rather serve as a road map for the entire product development process.In the given below I provide the product backlog:

- Registration
- Login
- Manage Profile
- Access Dashboard
- Manage Service Request
- Create Event
- Create Bill
- Submit Complaints
- Create Parking Request
- Create Security Log
- Manage Directory
- Create Community Bulletin Board

3.6 Sprint 2 Backlog

The Scrum Master selected six features that were completed during Sprint 1. They are:

- Create Bill done by Hasneen Tamanna (HT)

- Create Parking Request done by Jubaer Ahmad Khan (JK)
- Submit Complaints done by Shanjida Alam (SA)
- Create Security Log done by Md. Tanvir Hossain Saon (TH)
- Manage Directory done by Solaimi Hamid (SH)
- Create Community Bulletin Board done by Irtifa Haider (IH)

3.7 Tools Used

- **Trello:** Task Management.
- **Discord:** Daily scrum meeting and communication with each other during Sprint 2.
- **Toggle:** Time management.

Chapter 4

Sprint 2 Execution

4.1 Daily Scrum Meeting

- **Daily Scrum Meeting 1:**

What we did yesterday?	What problems faced?	What will do today?
Explored Test Driven Development	None	Create User Interface for collecting the complaints , Create Complaints Model

Here I only mention my part of the daily scrum meeting.

- **Daily Scrum Meeting 2:**

What we did yesterday?	What problems faced?	What will do today?
Created a UI for resident complaints, Created model class for Complaints, Created a new branch in GitHub	None	Generate test cases for TDD

Here I only mention my part of the daily scrum meeting.

- **Daily Scrum Meeting 3:**

What we did yesterday?	What problems faced?	What will do today?
Generated test cases, Completed the full User Interface (UI) and Attempted Test-Driven Development (TDD).	Faced difficulties in generating test cases within the TDD approach.	Work on resolving the issues faced with test case generation using TDD.

Here I only mention my part of the daily scrum meeting.

- **Daily Scrum Meeting 4:**

What we did yesterday?	What problems faced?	What will do today?
Successfully implemented TDD for the Create Complaint feature. Modified the Complaint Form UI. Successfully saved resident complaints to the database with validation.	Encountered issues generating test cases, passing failed test cases, and synchronizing data with Firebase.	Generate documentation and open a pull request for CI testing.

Here I only mention my part of the daily scrum meeting.

4.2 My Contribution during This Sprint 2

During Sprint 2, I worked on the Submit Complaints feature. I began by creating the XML layout for the user interface. Once completed, I shared this layout in the Discord channel with my teammates to gather valuable feedback on this initial component.Here I include attachment about that,

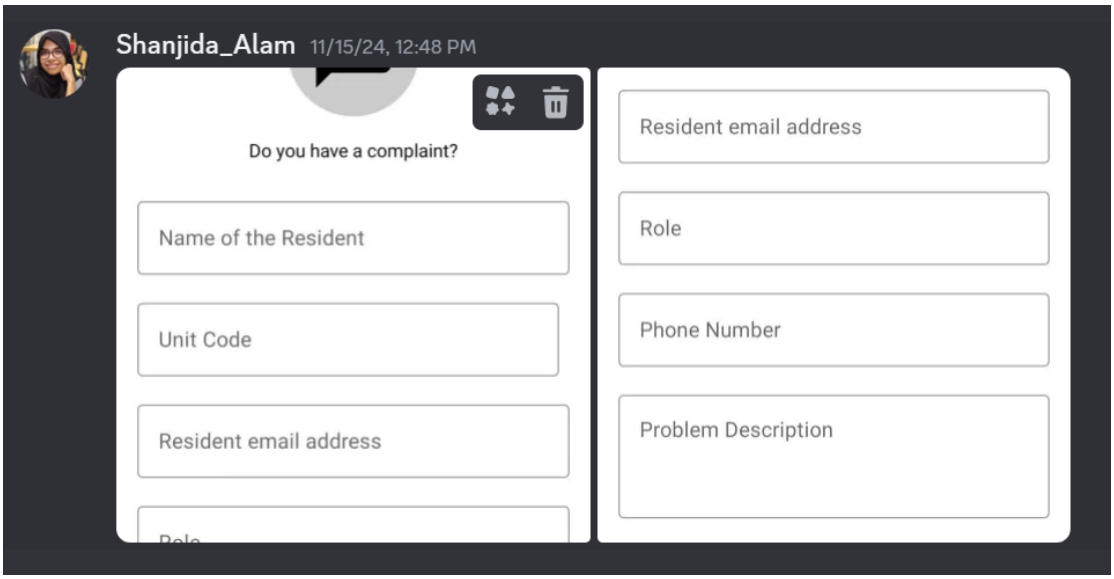


Figure 4.1: This is screenshot of my document that I shared in the Discord group with my teammates.

After finalizing the UI design, I proceeded with the backend implementation, ensuring the codebase is well-structured and maintainable. Next, I integrated the code with the Firebase Datastore to enable data retrieval and storage from the database.
I dedicated significant effort to ensure the full implementation was completed within the deadline. To give a clear picture of my progress and time management, I've attached screenshots of my Git Bash activity along with my Toggl Track time entries. These provide an overview of the work done and the time invested in the project.

4.3 Visual Aspect of Git Bash Activity for Submit Complaints

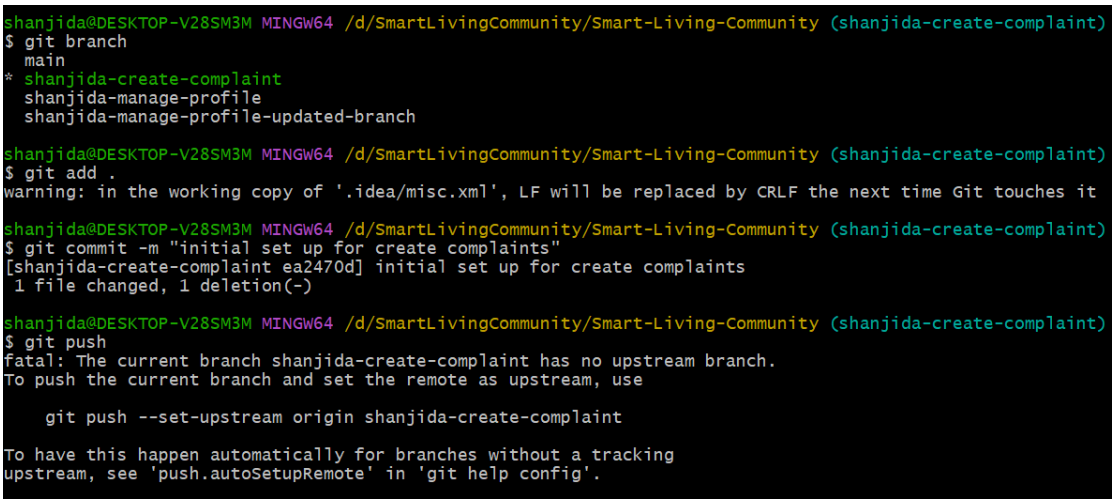


Figure 4.2: The screenshot describes the initial steps in my development workflow for the **Submit Complaint** feature. It represents the creation of a new branch named **'shanjida-create-complaint'** on GitHub. Sequentially, I set up the local development environment and pushed the initial codebase to this newly created branch.

Figure 4.3: This screenshot illustrates the successful push of the code that displays one test case passed within the TDD.

```
shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git add .

shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git commit -m "16 test cases are passed <shanjida>"
[shanjida-create-complaint 86d69a9] 16 test cases are passed <shanjida>
 6 files changed, 308 insertions(+), 151 deletions(-)
 delete mode 100644 app/src/androidTest/java/com/example/smartlivingcommunity/ComplaintTest.java
 create mode 100644 app/src/main/java/com/example/smartlivingcommunity/data/repository/ComplaintRepositoryImpl.java

shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git push
Enumerating objects: 50, done.
Counting objects: 100% (50/50), done.
Delta compression using up to 8 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (28/28), 4.03 KiB | 825.00 KiB/s, done.
Total 28 (delta 6), reused 5 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/shanjida-alam/Smart-Living-Community.git
   6fbb544..86d69a9  shanjida-create-complaint -> shanjida-create-complaint
```

Figure 4.4: This screenshot captures the successful push of the test case code, showcasing the execution and successful completion of 16 test cases within the TDD.

```
shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git add .

shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git commit -m "successfully 18 test cases are passed <shanjida>"
[shanjida-create-complaint bd8c525] successfully 18 test cases are passed <shanjida>
 4 files changed, 148 insertions(+), 17 deletions(-)

shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git push
Enumerating objects: 41, done.
Counting objects: 100% (41/41), done.
Delta compression using up to 8 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (22/22), 2.77 KiB | 944.00 KiB/s, done.
Total 22 (delta 8), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (8/8), completed with 8 local objects.
To https://github.com/shanjida-alam/Smart-Living-Community.git
   86d69a9..bd8c525  shanjida-create-complaint -> shanjida-create-complaint
```

Figure 4.5: This screenshot captures the successful push of the test case code, showcasing the execution and successful completion of 18 test cases within the TDD.

```
shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git add .
warning: in the working copy of 'app/src/main/res/layout/fragment_complaint.xml', LF will be replaced by CRLF the next time Git touches it

shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git commit -m "successfully 19 test cases are passed <shanjida>"
[shanjida-create-complaint 4b286bd] successfully 19 test cases are passed <shanjida>
 4 files changed, 99 insertions(+), 57 deletions(-)

shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git push
Enumerating objects: 43, done.
Counting objects: 100% (43/43), done.
Delta compression using up to 8 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (23/23), 2.58 KiB | 660.00 KiB/s, done.
Total 23 (delta 11), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (11/11), completed with 11 local objects.
To https://github.com/shanjida-alam/Smart-Living-Community.git
   bd8c525..4b286bd  shanjida-create-complaint -> shanjida-create-complaint
```

Figure 4.6: This screenshot captures the successful push of the test case code, showcasing the execution and successful completion of 19 test cases within the TDD.

```
shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git add .
warning: in the working copy of 'app/src/main/res/layout/fragment_complaint.xml', LF will be replaced by CRLF the next time Git touches it

shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git commit -m "uploaded the UI of LivSmart Complaint Form <shanjida>"
[shanjida-create-complaint 2d251dc] uploaded the UI of LivSmart Complaint Form <shanjida>
 1 file changed, 43 insertions(+), 26 deletions(-)

shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git push
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 881 bytes | 440.00 KiB/s, done.
Total 8 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/shanjida-alam/Smart-Living-Community.git
   4b286bd..2d251dc  shanjida-create-complaint -> shanjida-create-complaint
```

Figure 4.7: This screenshot describes the successful push of the xml code that is the UI of

```
shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git add .
warning: in the working copy of '.idea/inspectionProfiles/Project_Default.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app/src/main/java/com/example/smartlivingcommunity/ui/view/content/ComplaintFragment.java', LF will be replaced by CRLF the next time Git
ouches it
warning: in the working copy of 'app/src/main/res/layout/fragment_complaint.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app/src/main/res/values/strings.xml', LF will be replaced by CRLF the next time Git touches it

shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git commit -m "generated documentation for my codebase <shanjida>"
[shanjida-create-complaint 07a68c3] generated documentation for my codebase <shanjida>
 13 files changed, 1023 insertions(+), 211 deletions(-)
 delete mode 100644 app/src/main/java/com/example/smartlivingcommunity/data/repository/ComplaintRepositoryImpl.java
 create mode 100644 app/src/main/java/com/example/smartlivingcommunity/data/repository/ComplaintRepositoryImplementation.java

shanjida@DESKTOP-V28SM3M MINGW64 /d/SmartLivingCommunity/Smart-Living-Community (shanjida-create-complaint)
$ git push
Enumerating objects: 72, done.
Counting objects: 100% (72/72), done.
Delta compression using up to 8 threads
Compressing objects: 100% (30/30), done.
Writing objects: 100% (38/38), 12.43 KiB | 1.78 MiB/s, done.
Total 38 (delta 17), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (17/17), completed with 17 local objects.
To https://github.com/shanjida-alam/Smart-Living-Community.git
   2d251dc..07a68c3  shanjida-create-complaint -> shanjida-create-complaint
```

Figure 4.8: This screenshot shows the successful push of the code responsible for generating the project documentation.

4.4 Visual Aspect of Toggl Track for Submit Complaint

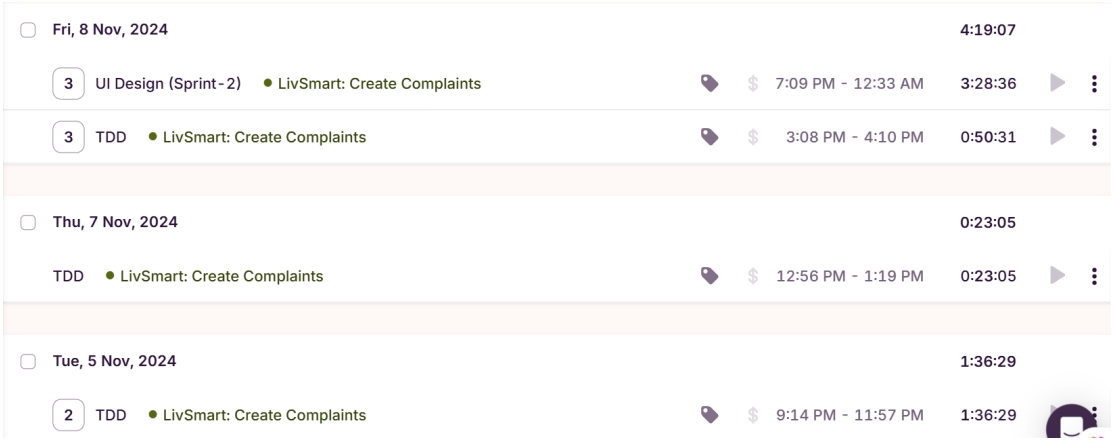


Figure 4.9: This Toggl Track log details the time spent on various tasks related to the **Submit Complaint** feature during sprint-2 in November 2024. It covers three consecutive days and includes tasks associated with both User Interface (UI) design and Test Driven Development (TDD). The total time logged for these tasks is 6 hours, 18 minutes, and 41 seconds.

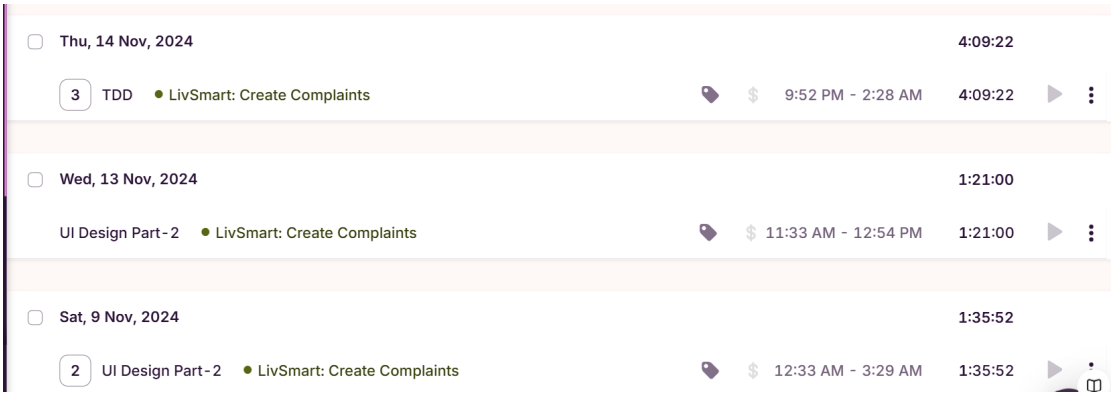


Figure 4.10: This Toggl Track log also describes the time spent on different tasks related to the **Submit Complaint** feature during sprint-2 in November 2024. It covers three days and includes tasks associated with both User Interface (UI) design and Test Driven Development (TDD). So, the total time spent over these three days was 7 hours 6 minutes and 14 seconds.

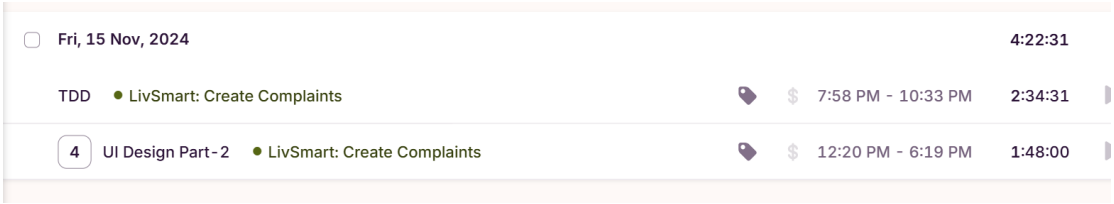


Figure 4.11: This is a Toggl time tracking log showing work done on Friday, November 15, 2024 for the **Submit Complaint** feature. The total time for the day is shown as 4 hours 22 minutes and 31 seconds.

I spent a total of 17 hours 47 minutes 26 seconds completing the **Submit Complaint** feature.Toggl is a valuable tool for tracking and measuring my contributions to the project.

Chapter 5

Conclusion

Sprint 2 demonstrated the value of iterative improvement in Scrum, showcasing enhanced planning, execution, and testing processes. The integration of TDD and continuous testing ensured better product quality and early issue detection. The insights gained will guide future Sprints for even better outcomes.

Report Title : Test Driven Development (TDD) for The Smart Living Community Project

Course Code: CSE 404

Course Title: Software Engineering and ISD Laboratory

Submitted by

SHANJIDA ALAM(ID: 353)

Submitted to

Dr. Md. MUSHFIQUE ANWAR, Professor

Dr. Md. HUMAYUN KABIR, Professor

Computer Science and Engineering

Jahangirnagar University

Dhaka, Bangladesh

January 07, 2025

Contents

Chapter 1

Introduction

Test-Driven Development (TDD) is an essential agile practice that helps development teams clarify and understand project requirements, especially in the early stages. By combining TDD with the right tools and processes, teams can create comprehensive test suites, enhance software quality, and support **Continuous Integration (CI)** workflows. This report highlights my personal contribution to implementing TDD in our project starting from Sprint 2 of the development phase. It also covers the tools we used, the challenges we encountered, and the lessons we gained along the way.

This report focuses on my individual role in incorporating TDD into our project during the development phase, starting with Sprint 2. It details my efforts in writing test cases before developing features, refining requirements through iterative feedback, and collaborating with the team to integrate TDD workflows effectively. Additionally, it highlights the tools we employed to enhance the process, the problems we encountered while implementing TDD, and the insights we gained, which have shaped our approach to development in subsequent sprints.

Chapter 2

JUnit for Test Driven Development(TDD)

2.1 What is Junit?

JUnit is a widely-used framework for unit testing Java applications. It provides annotations, assertions, and methods to test individual components of the code, ensuring they work as expected. With JUnit, developers can isolate and test each module independently, which helps catch bugs early and improve code quality.

2.2 Why Choose JUnit?

- **Reliability and Popularity:** JUnit is a stable and mature testing framework supported by a vast community. Its popularity in the Java ecosystem makes it a reliable choice with extensive documentation and resources available for troubleshooting.
- **Integration with Android Studio:** For our LivSmart project, which uses Android Studio and Java, JUnit integrates seamlessly, allowing us to write and run tests directly in the IDE.
- **Support for Test-Driven Development (TDD):** JUnit supports TDD principles, making it easier to write tests before or alongside the development of features.
- **Easy Assertion Library:** JUnit provides a simple assertion library that makes it easy to verify expected results. It simplifies test writing and helps maintain clear, readable test code.

- **Compatibility with Mockito:** JUnit pairs well with Mockito, a framework for mocking dependencies. Since we are also exploring Mockito, this makes JUnit an even better choice for unit testing.

Chapter 3

My Contribution to The TDD

3.1 Test Case Created

- Ensure `ComplaintModel` fields are validated.
- Verify that the complaint submission API returns a success response.
- Validate error handling for incomplete data.

3.2 Writing Tests Before Code

At first I started by writing failing tests for each feature. Then, incrementally wrote implementation code to make tests pass. And finally refactored the code while ensuring that tests remained green. Here, attach the test code:

```
1  @Test
2      public void submitComplaint_withValidData_shouldSucceed() {
3          /**
4           * Create a valid complaint test case
5           */
6          ComplaintModel complaint = new ComplaintModel(
7                   "A101", "John Doe",
8                  "Resident", "12345678905", "john@gmail.com", "Test
       complaint"
9          );
10
11 @Test
12     public void submitComplaint_withEmptyUnitCode_shouldFail() {
13         /**
14          * Create an invalid complaint test case
15          */
```

```
16          ComplaintModel complaint = new ComplaintModel();
17          /**
18           * Set the unit code to an empty string
19           */
20          complaint.setUnitCode("");
21          /**
22           * Set the other fields to valid values
23           */
24          complaint.setUserName("John Doe");
25          complaint.setUserRole("Resident");
26          complaint.setPhoneNumber("1234567890");
27          complaint.setEmailAddress("john@gmail.com");
28          complaint.setComplaintDescription("Test complaint");
29
30          // Execute
31          /**
32           * Call the submitComplaint method
33           */
34          LiveData<Boolean> result = viewModel.submitComplaint(
    complaint);
35
36          // Verify
37          /**
38           * Assert that the result is false
39           */
40          assertEquals(false, result.getValue());
41          verify(repository, never()).submitComplaint(any(
    ComplaintModel.class));
42      }
43
44      @Test
45      public void submitComplaint_withEmptyDescription_shouldFail() {
46          /**
47           * Create an invalid complaint test case
48           */
49          ComplaintModel complaint = new ComplaintModel(
50                  "A101", "John Doe",
51                  "Resident", "1234567890", "john@gmail.com", ""
52          );
53
54          /**
55           * Call the submitComplaint method
56           */
57          LiveData<Boolean> result = viewModel.submitComplaint(
    complaint);
58          /**
```

```
59          * Assert that the result is false
60          */
61         assertEquals(false, result.getValue());
62         /**
63          * Verify that the repository was not called
64          */
65         verify(repository, never()).submitComplaint(any(
    ComplaintModel.class));
66     }
67
68     @Test
69     public void submitComplaint_withInvalidPhoneNumber_shouldFail() {
70         /**
71          * Create an invalid complaint test case
72          */
73         ComplaintModel complaint = new ComplaintModel(
74                 "A101", "John Doe",
75                 "Resident", "123", "john@gmail.com", "Test complaint"
76         );
77
78         /**
79          * Call the submitComplaint method
80          */
81         LiveData<Boolean> result = viewModel.submitComplaint(
    complaint);
82
83         /**
84          * Assert that the result is false
85          */
86         assertEquals(false, result.getValue());
87         /**
88          * Verify that the repository was not called
89          */
90         verify(repository, never()).submitComplaint(any(
    ComplaintModel.class));
91     }
92
93     @Test
94     public void submitComplaint_withEmptyUserName_shouldFail() {
95         /**
96          * Create an invalid complaint test case
97          */
98         ComplaintModel complaint = new ComplaintModel(
99                  "A101", "",
100                 "Resident", "1234567890","john@gmail.com", "Test
    complaint"
```

```
101         );
102
103         /**
104          * Call the submitComplaint method
105          */
106         LiveData<Boolean> result = viewModel.submitComplaint(
    complaint);
107
108         /**
109          * Assert that the result is false
110          */
111         assertEquals(false, result.getValue());
112         /**
113          * Verify that the repository was not called
114          */
115         verify(repository, never()).submitComplaint(any(
    ComplaintModel.class));
116     }
117
118     @Test
119     public void submitComplaint_withEmptyUserRole_shouldFail() {
120         /**
121          * Create an invalid complaint test case
122          */
123         ComplaintModel complaint = new ComplaintModel(
124                 "A101", "John Doe",
125                 "", "1234567890", "john@gmail.com","Test complaint"
126         );
127
128         /**
129          * Call the submitComplaint method
130          */
131         LiveData<Boolean> result = viewModel.submitComplaint(
    complaint);
132
133         /**
134          * Assert that the result is false
135          */
136         assertEquals(false, result.getValue());
137         /**
138          * Verify that the repository was not called
139          */
140         verify(repository, never()).submitComplaint(any(
    ComplaintModel.class));
141     }
142
```

```
143 @Test
144     public void submitComplaint_withNullComplaint_shouldFail() {
145         /**
146          * Call the submitComplaint method
147          */
148         LiveData<Boolean> result = viewModel.submitComplaint(null);
149         /**
150          * Assert that the result is false
151          */
152         assertEquals(false, result.getValue());
153         /**
154          * Verify that the repository was not called
155          */
156         verify(repository, never()).submitComplaint(any());
157     }
158 
159      @Test
160     public void submitComplaint_withEmptyEmailAddress_shouldFail() {
161         /**
162          * Create an invalid complaint test case
163          */
164         ComplaintModel complaint = new ComplaintModel(
165                 "A101", "John Doe",
166                 "Resident", "1234567890","", "Test complaint"
167         );
168         complaint.setEmailAddress("");
169 
170         /**
171          * Call the submitComplaint method
172          */
173         LiveData<Boolean> result = viewModel.submitComplaint(
    complaint);
174         /**
175          * Assert that the result is false
176          */
177         assertEquals(false, result.getValue());
178     }
179 
180      @Test
181     public void submitComplaint_withInvalidEmailFormat_shouldFail() {
182         /**
183          * Create an invalid complaint test case
184          */
185         ComplaintModel complaint = new ComplaintModel(
186                 "A101", "John Doe",
```

```
187                 "Resident", "1234567890", "johnmail.com", "Test
    complaint"
188         );
189         complaint.setEmailAddress("invalid.email");
190
191         /**
192          * Call the submitComplaint method
193          */
194         LiveData<Boolean> result = viewModel.submitComplaint(
    complaint);
195         /**
196          * Assert that the result is false
197          */
198         as
```

Listing 3.1: ComplaintViewModelTest Class in Java

3.3 Visual Aspect of Passing Test

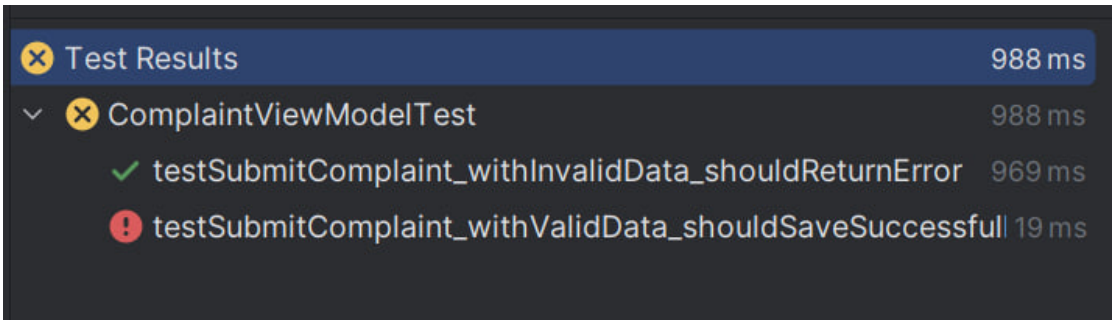


Figure 3.1: This image shows that initially InvalidData pass the test case and ValidData does not pass the test case. And write the code for passing this testcase.

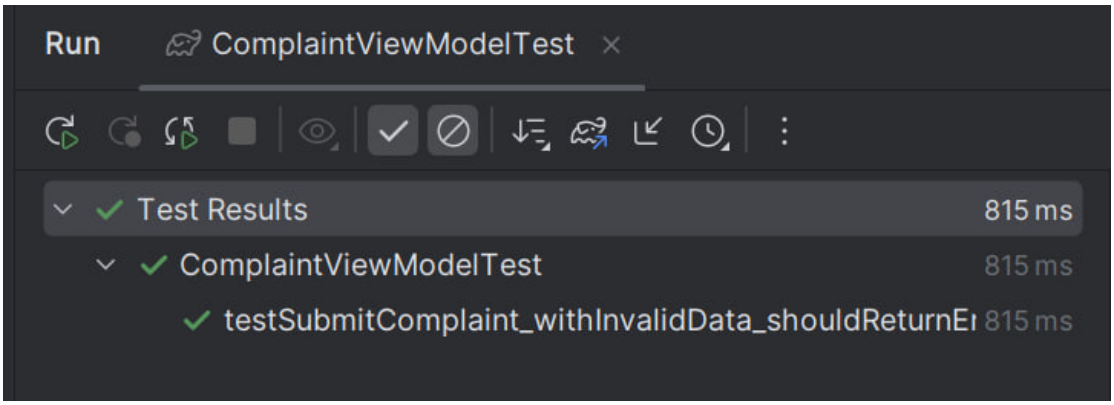


Figure 3.2: This image shows that pass the test case with the Valid Data input.

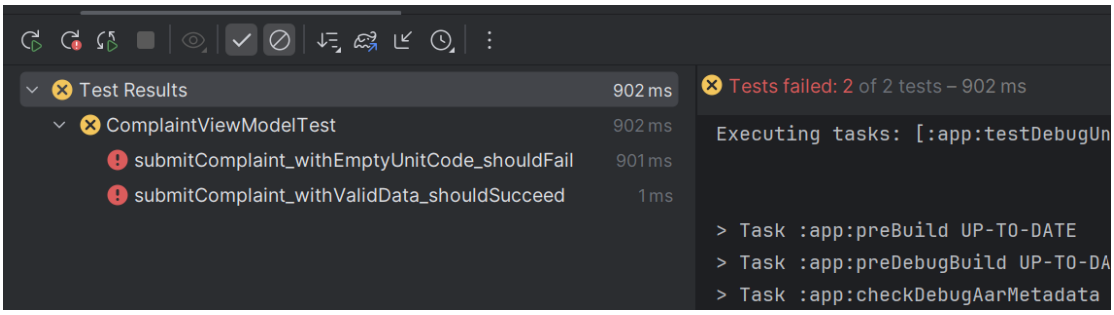


Figure 3.3: This image shows that the empty input test case does not pass in this part.

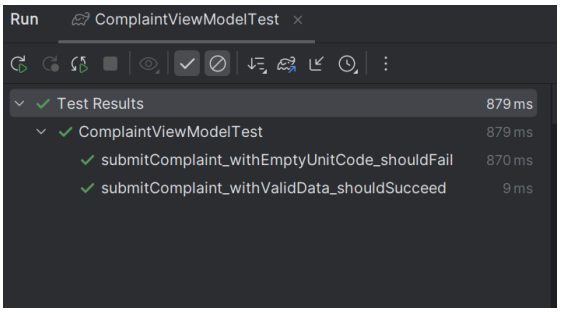


Figure 3.4: This image shows that the empty input test case pass in this part.

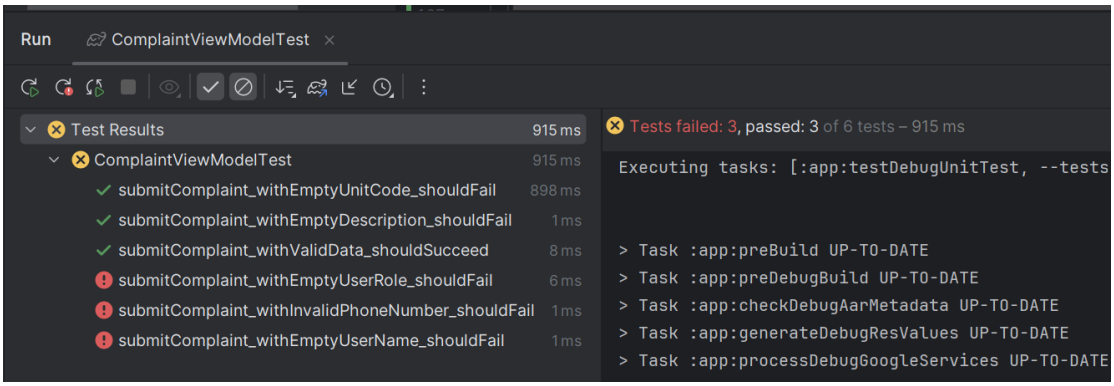


Figure 3.5: This image shows that the empty unitCode pass but emptyUserRole test case does not pass.

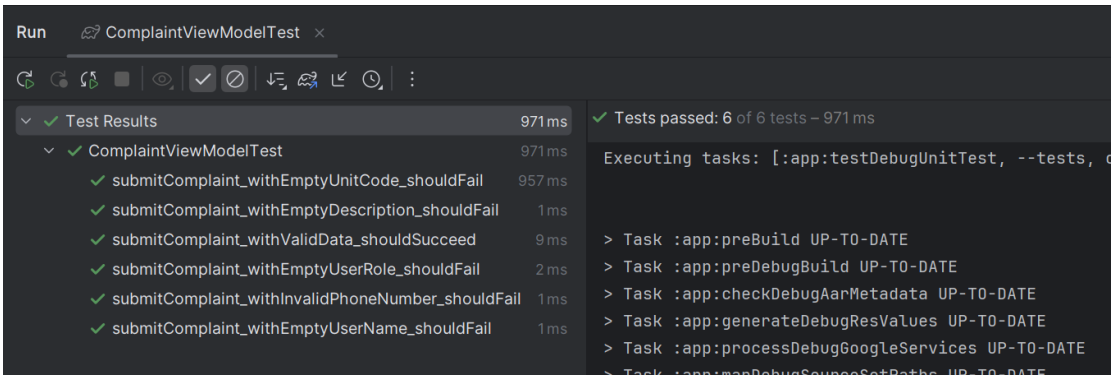


Figure 3.6: All possible test case should be passed in this part.

Chapter 4

Conclusion

Adopting TDD was a best experience for both me and the team. It not only improved our understanding of requirements but also enhanced the quality of our code and processes. By integrating TDD with CI, we ensured that our project maintained high standards of quality and reliability. I look forward to further refining this practice in future projects.