Chapter 7
Conclusion and Future Work

7.1 Conclusion

In conclusion, this software project helped me learned and understanding the significant efforts that goes into making the quality software that exists today. It adds to the curiosity of knowing how do other software handle the issues faced by me and how do they overcome them.

The importance of software engineering was quire apparent from the start of the project and taught me that coding is just a part of developing a software. Proper documentation, requirements gathering, architecture design, planning, implementation, etc are the key pillars in building quality software. It helped me grasp the importance of feasibility study, carrying out survey of technologies and why it is key even before starting the development of the project, as it encouraged me to find alternative technologies of solving and achieving the desired functionality.

A major need identified was the need for a local database alongside the remote MySQL database. Mobile applications need to work even in conditions when there is poor network connection or even the lack of it. Adhering to an offline-first approach enhances user experience and is a vital part of any mobile application. Caching is also important to improve system performance and helps to reduce the load on the database. Local database, caching and remote database all together should form the data storage mechanism of any mobile application.

Hosting the MySQL database was another major issue that I faced since the knowledge of cloud computing is required in order to achieve this. This explains the need to explore other relational database technologies such as Postgres, for which there are more hosting providers than MySQL.

In the context of writing code, recognizing the ripple effects of changes in one module on other interconnected parts of the system underscored the necessity for comprehensive testing and validation at each development stage. The need for automated testing was realized because it not possible to manually carry out all the test cases and eventually miss out on some executing some test cases. This leads to errors and issues later on and figuring this out is very time consuming which leads to a significant increase in development time of the project. Hence automated testing is very important for building large scale applications.

Effective time management and accurate project time estimation are other learnings from this project. Prioritizing what work needs to be done and how much time it will take comes with experience, which this project has given me.

The dart language comes with a bunch of features and introduces a strongly-typed nature to JavaScript. This helps to get started with it quite easy. Flutter leverages the benefits of Dart, and proves to be a powerful tool for cross-platform mobile app development giving great performance on each platform by making use of its inbuilt engine.

Navigating the complexity of state management, particularly with Provider state management, was the trickiest part of frontend development. Creating reusable components, leveraging OOPS paradigms such as Inheritance and Polymorphism were key to write code that is readable and maintainable.

Exploring Node.js and Express deepened the understanding of backend development, covering API design, Socket.IO and implementing JWT for secure communication. It helped me grasp the core concepts of API development and taught me that API development requires careful planning before implementing them.

7.2 Limitations of the system

- **Database Constraints**: The hosted is not free of limitations. The constraints "Max User limit reached" and 10MB storage considerably affect the performance of the system.
- **No Automated Testing:** No automated testing tools are used, which makes the testing process quite tedious, inefficient, and very time consuming. Also, a lot of times the test cases that have been missed during manual testing are the cause of bugs.
- **No local database:** There is no local database used to cache data locally on the device. Local database caching gives a much better user experience and enables the app to work in poor or no network conditions. Also, performing operations from the local database are much faster as compared to operations performed on remote databases and significantly increases the performance of the app.
- **Repetition of Same Code**: There are redundant pieces of code on both frontend and backend which need to be refactored in order to make the code more change-friendly.
- **No Verification of Progress**: There is no verification of the progress of the reported item when the user updates it. More stages need to be introduced along with increased involvement of the admin to avoid any scams.
- **No verification of images before uploading**: There should be a verification system in place that checks the images that are uploaded by the users.

7.3 Future Scope

- **Verification before the users can chat:** There should be security questions in place before two users can communicate with each other. This requires NLP concepts to be implemented.
- **Item Matching using Flask**: Any two similar items that are reported, can be identified and the concerned users can be notified about the same. This also requires NLP concepts This would enhance the user experience and increase the system's effectiveness.
- **Notifications**: Send notifications to the users wherever required.
- **Add More Stages to Progress**: More stages need to be introduced to accurately depict the current status of the reported item.
- **Local Database**: Implement a local database which caches the database stored remotely. In case of poor network connection, this database would be used and it guarantees that the application would work in the College campus where network connectivity is a problem.
- **Use Gemini to Detect Images Before Being Uploaded**: Recently in March of 2024, Google introduced "Gemini", their Generative AI solution. This can be integrated in Flutter using their package. Explore its computer vision feature and whether it can detect inappropriate images uploaded by the users. This proactive approach ensures that only appropriate and relevant images are processed, enhancing the overall content moderation and user experience.
- **Develop the Admin Side**: Introduce an administrative module to oversee and manage various aspects of the system. Admin functionalities should include the ability to manage categories, users, FAQs, and reported items, providing centralized control and monitoring.
- **Store Locations Around the College in the Database:** All the possible locations can be stored in the database and can be used to provide suggestions to the "Location" input field when a new item is being reported. This will help maintain consistency in the data that is entered and make it easier to find similar items using NLP at the backend.
- **Add Versioning in APIs**: Implement versioning in APIs to ensure compatibility and smooth transitions during future updates. This practice enables the introduction of new features without disrupting existing functionalities.