Life-Cycle Model: MTS

1. Introduction

- The life-cycle model that most accurately describes the software development process of the MTS is the Agile Software Development Life Cycle. This process focuses on iterative development, collaboration, and continuous improvement. The model allowed us to adapt to changes and to continuously improve our system through the delivery of incremental functionality over the course of development.

2. Phases of the Model.

2.1 Requirements Gathering and Analysis

- The requirements were given to us in the form of the instructions for each assignment. By carefully reading and through discussion with the group we were able to understand what we needed to do in each phase development.

2.2 Design

- After gathering and understating the requirements, we began to brainstorm the best ways to meet them. After we had come to a decision, we created our architecture, UML diagrams and our SRS that described how we would implement the requirements.

2.3 Implementation

- This is where, should we have been instructed to, we would have built the website and implemented the features in code. However in the context of the class, this is more similar to the part where we got to work writing and creating our diagrams and documents.

2.4 Testing

- This is where we would have performed tests on our system in order to ensure that it works in ways that are intended and to ensure that there are no critical bugs. In the context of the class, this is more like when we received our feedback from the TA after each assignment.

2.5 Deployment

- After testing the MTS would have been pushed into live service and the system would have been fully usable by people and the system administrators. In class this is similar to implementing feedback that the TA had provided as part of the next assignment and submitting that for further review.

2.6 Maintenance

- Bugs encountered by users and planned updates continue to take place. They help to ensure that the system runs smoothly and effectively. No real clear parallel to class, but it is again most similar to implementing feedback provided by the TA after an assignment and its submission for review in the next assignment.

3. Reflection on the Model.

3.1 Effectiveness for this Project

- This model was highly effective for this class and for the development of the MTS because it allowed for Iterative development in the form of assignments and allowed us to implement feedback effectively.
- It also allied for flexibility and helped us to implement changing and expanding requirements like adding new features and new ideas that come forward. It also allowed us to change things and ideas that might not work with the new requirements each assignment brought.
- The model also allowed for effective collaboration in each stage of the process through the use of a central document.

3.3 Agile's General Applicability.

- This model works best in projects in which the requirements are likely to change over time, where collaboration between stakeholders and developers is important, and where the incremental delivery of the project for feedback is valuable.
- It is probably less effective for projects with fixed requirements and strict timelines.

4. Potential Improvements to the Life-Cycle Model

While the Agile model worked well for this project, the following enhancements could improve future implementations:

1. **Stronger Sprint Planning**: More detailed sprint goals to improve task prioritization.
2. **Automated Testing**: Greater emphasis on automated testing to accelerate the testing phase.
3. **Scalability Considerations**: Incorporate scalability testing earlier in the development process.

In terms of Scalability, I found that sometimes we would have created our plan effectively for one assignment, but when the next assignment came, there would be issues that came up with our MTS and we would have to rewrite large parts of the MTS to compensate. I also think it works best for small groups, should they get too large, it would become difficult to synchronize across the entire team.

Summary

The Movie Ticketing system is an online platform that is designed to streamline and enhance the process of browning, selecting and booking movie tickets for users, while also giving system administrators tools for proper theatre management. It was developed with a three tier architecture that ensures the system's security, scalability and its ease of use. The FrontEnd provides users with a responsive, user friendly interface while the back end manages the business logic and API interactions as well as storing the relational database. The database organizes structured data such as user profiles, bookings and showtimes.

Of the many core features of the MTS, several of the most important include user registration, login, movie browning, seat selection, ticket booking and secure payment processing. Users on the internet can search for movies by title, genre or keywords, and filter their search results based on their wants, including by theaters, showtimes, and availability. The system provides an interactive seat selection tool and integrates multiple payment options like credit cards, PayPal, and digital wallets. After booking a ticket, users will receive digital tickets via email or SMS that include a QR code to be scanned at the gate, which they can also view or manage through their accounts on the MTS.

The platform also offers an admin dashboard to movie theatre managers. It will enable them to control their theaters. Functionality included is adding, modifying, or removing movie listing, managing showtimes and generating sales and performance reports. It is important that managers have this power in order that they are able to customize the MTS to their theater. They can also create and manage other admin accounts with varying degrees of access and ability. For example they can create employee accounts that have the ability to add or refund tickets but don't have the ability to delete tickets or users from the system. This along with other security features are part of the MTS's robust security operations. Other security features include things like multi factor authentication, encrypted data and data transmission as well as the role based access controls for users, employees and administrators as discussed above.

The system was developed using an Agile Software Development Life Cycle approach. This approach emphasises iterative development and collaboration. The modular design of the MTS will help in the development of future features and projects, like loyalty programs, AI-driven movie recommendation and multilingual support. Testing on the MTS was conducted at all levels and on all points of failure (see the TestCases spreadsheet) to ensure that the MTS's functional performance, and security features work as expected.

By addressing the needs of both the users and the system administrators, the MTS, improves the movie ticketing experience, providing convenience for customers, and robust, usable management tools to theatre staff. The system's design and functionality are aligned with modern standards for usability security and performance.

Conclusion

The biggest takeaway I have from this assignment, and the class in general, is a proper, effective way to plan and execute a piece of software. Normally, when I start writing a piece of software for a project, or as a class assignment, I just start coding. I will start with the simplest things that I can and then as I bump into problems I go back and fix them. It works, but it's often slower than it would have been if the software had been properly planned. Often, I find that it's most effective when I just need to get something out there as a starting point to build off, it’s a particularly effective way to get past an idea block or being stumped.

I found the software development process to be an effective way to both plan a project that was free of inconsistencies, and a great way to work in a group. The most important thing that I took away from the process was the lower amount of revisions that I had to make to the implementation because we had planned for future requirements or because we had properly planned our MTS to support more complex requirements. My way of working also makes working in groups rather difficult, as my code is often haphazard and lacking in documentation. The Software Development Process forced me to work with my group from the very beginning of the process, and it meant that I could depend on them to communicate what was needed and to back up my work. It was something that I hadn't really encountered so far in my degree because most of my projects have been one person at most, meaning that I didn’t have to communicate or provide documentation for everything I did, I only had to provide a working program.

I think one of the biggest challenges that me in particular, but also my group bumped into was the sheer amount of choices that we had to make as well as making sure that choices that we had already made didn’t negatively impact choices further down the road. In particular I think that choosing how and whether or not our system was going to interface with third party applications. There were so many choices and it was difficult to know if we made the right one. However I think it was an important road block that we hit. In the industry there are going to be thousands of choices just like that, that we have to make and while making the right choice might be difficult, communication with your team, and attention to the client’s requirements will help to guide you in the right direction.

I found the most helpful part of the development process to be the incremental approach to the project. Instead of expecting a huge, thirty to forty page document to be submitted at the end of the class, the incremental approach to the submissions helped to guarantee that we didn’t feel too overwhelmed by the size of the project that we were working on. Equally valuable was the feedback that we got at each step, and the ability to go back and fix, enhance and build on what we had created helped to make sure that our final document wasn't ridden with mistakes.

Overall I found the SDLC and the project very helpful in expanding my horizons and I learned a lot, especially in the areas of working in a group as part of a team and how to properly plan and execute the building of a piece of software.