|  |  |
| --- | --- |
|  |  |
| **Group 9A**  **Element 2**  **System Development and Methodology** |  |
|  |  |
|  | UFCF8S-30-2  Advanced Software Development  Horizon Cinemas Booking System |
|  | Amira Soumid  Abdelmadjid Lachichi  Nathan Baker  Terence Bakhda |

**Element 2 (System Development and Methodology- 30 marks)**

|  |
| --- |
| Our team followed an Agile development methodology to design and build the Horizon Cinemas Booking System using Python and SQLite. The aim was to create a fully functional cinema management system that supports user login, film listings, show scheduling, bookings, cancellations, and reporting — all through a graphical user interface (GUI).  We split the work between four team members. We held initial planning discussions to break down the project into user stories and features, such as admin login, managing film and cinema data, booking creation, and ticket cancellation. Each of us took on roles depending on our strengths and availability.  One team member focused on the UI side, providing wireframes that gave us a good foundation for how the application might look and flow. While the final UI evolved differently as development progressed, these early mock-ups helped align our understanding and goals.  Another teammate contributed to the early stages of development, particularly in setting up the initial file structure and foundational code layout. They helped define how the system would be organised into modules, such as separating views, models, and database utilities. Their early work on connecting the database and designing some of the screen transitions laid groundwork for later development.  The bulk of the development and system integration, including the backend logic, booking system, GUI interface, and testing, was completed by another teammate. They took a lead role in ensuring the system was functional and cohesive. This included building key features such as:   * User authentication (with different roles: Admin, Manager, Booking Staff) * Film listings with dynamic date and cinema filtering * Real-time seat availability and price calculations * A flexible booking interface with validation * A cancellation module that allows searching by email or booking reference * A consistent and intuitive layout using Tkinter with dynamic content loading   Throughout, we worked iteratively, testing features and adjusting based on what worked well. We used Agile principles like incremental progress, rapid testing, and adapting to issues. Whenever something didn’t go to plan like a feature not working as intended or an image not displaying, we debugged and restructured as needed.  **Security Measures and Database Design**  We implemented password hashing using the bcrypt library to protect user credentials securely. Plain text passwords are never stored, aligning with industry best practices.  For the database, we chose SQLite due to its simplicity and integration with Python. It requires no server setup, which made it ideal for this desktop-based system. We designed the database using normalisation principles and created the following main tables:   * users: storing login info and roles * films, cinemas, screens, shows: to manage cinema content * bookings: to log every ticket sale, including seat types, pricing, and booking reference   We included additional fields in the bookings table like film\_title, screen\_number, and show\_datetime to support reporting and future auditing features.  Mock data was seeded into the database using a Python script to simulate real-world scenarios across multiple cinemas, films, and shows for up to 7 days. This was critical for testing and demonstrating different views.  **Non-Functional and UI Improvements**  In addition to functionality, we paid attention to user experience. We used dropdowns, auto-filtering, and grouped controls logically. The system was tested with edge cases (e.g., invalid email, past bookings) to make it robust. We also integrated dynamic image loading for film posters, and handled errors (e.g., missing files or wrong formats) gracefully.  We maintained a modular project structure using views, utils, and models, which made debugging and further development easier.  **Conclusion**  While this project was challenging, it was also rewarding. Taking an Agile approach meant we could stay flexible and solve problems step by step. The experience highlighted the importance of version control, testing as you build, and having a clear plan — even in a small team.  In future iterations, we would consider connecting this to an online database or adding user account creation for additional staff/users. But overall, we’re proud of what we accomplished in this phase of the project. |