Project Postmortem Report

Course Name/Number: Computer Science 3307B - Object-Oriented Design and Analysis

Date: April 8th, 2024

Team Number: 11

Parth Bhandari (pbhanda2)

Sara Mehravar (smehrava)

Ashna Mittal (amitta6)

Manpreet Saini (msaini53)

Siddhant Saraf (ssaraf)

Table of Contents:

Project Summary

In the quest to revolutionize the learning experience through technology, our team chose to develop 'Intellecto', an innovative revision app designed to personalize and enhance the study process. This project, conceived from the ground up to incorporate advanced spaced repetition algorithms alongside a user-centric flashcard system, was selected with the dual goals of addressing a real-world educational need and leveraging cutting-edge object-oriented design and analysis principles. The inception of Intellecto was driven by a recognition of the diverse learning styles and the unique challenges faced by learners in retaining information over time. The rationale behind this selection was not only to fulfill academic requirements but also potentially address a universal need—enhancing learning efficiency through technology.

Intellecto's development journey was structured around the implementation of both fundamental and sophisticated features, directly informed by user stories that articulated the needs and aspirations of our target audience. The core functionalities—such as the spaced repetition algorithm, flashcard creation, deck organization, review sessions, and data storage—were identified as essential for providing a solid and reliable user experience and regarded as foundational in creating a versatile tool that could serve a broad spectrum of users—from students grappling with dense academic subjects to professionals and lifelong learners seeking to expand their knowledge horizons. Each feature was meticulously chosen and designed to foster an environment where users could effortlessly create, organize, and review study materials, thus optimizing their retention and recall of information over time. For instance, the spaced repetition algorithm, a cornerstone of our app, was implemented to dynamically adjust review schedules based on individual performance, thereby ensuring that learning efficiency was maximized.

Additionally, the project embraced optional and wishlist features, including gamification elements, collaborative functionalities, advanced statistics and reporting, and import/export capabilities. These were envisioned to not only enrich the user experience but also to introduce elements of engagement, collaboration, and detailed progress tracking, further setting Intellecto apart in the digital learning space.

The project's development was steeped in software management best practices, with particular emphasis on modularity, encapsulation, and scalability. These principles guided our design decisions and development methodologies, ensuring that Intellecto not only met its current specifications but was also poised for future expansion and innovation. The use of Unified Modeling Language (UML) diagrams facilitated a clear visualization of the system's architecture, promoting a structured and efficient development process.

We adopted agile methodologies as the backbone of our project management approach. This decision was pivotal, not only in ensuring the efficient progression of the project but also in fostering an environment of continuous collaboration, adaptation, and learning. From the outset, we recognized the inherent complexities and uncertainties of developing a feature-rich educational tool like Intellecto. The adoption of agile facilitated a high degree of collaboration within our group. Weekly stand-up meetings became a cornerstone of our routine, providing a platform for each member to share progress, identify obstacles, and seek support. This ongoing dialogue ensured that no team member was isolated in their tasks, fostering a strong sense of unity and shared purpose.

Our journey with Intellecto has been a testament to the power of combining technical rigour with a deep understanding of user needs. It underscored the importance of critical analysis, collaborative problem-solving, and creative innovation in software development. As we reflect on our journey, we are not only proud of the product we have created but also deeply enriched by the process, ready to carry these invaluable insights and experiences into our future endeavours in the field of computer science and beyond.

Key Accomplishments

What went right?

- **Implementation of Algorithm:** Our project's success can largely be attributed to our effective implementation of the required features, especially the spaced repetition algorithm and flashcard system. These components worked exceptionally well, significantly enhancing the study experience by adapting to individual learning styles and needs.
- **Decentralized Storage Model:** By creating a separate JSON file for deck names (deck_names.json) and individual JSON files for each deck's questions, the project benefited from a decentralized storage model. This design decision facilitated easier management of decks and their corresponding questions. It allowed for more granular control over the data, such as adding, removing, or updating decks and questions independently without affecting other parts of the dataset. Separating the decks into different JSON files also allowed the application to load up smaller files each time a deck was accessed, decreasing the overall lookup time and hence improving the time complexity of the algorithm.
- **Dynamic Difficulty Adjustment:** Implementing a system where each question's difficultyRating could be dynamically adjusted and saved during a test session was challenging but went remarkably well. It allowed us to tailor the difficulty of questions based on user feedback, leading to a more personalized testing experience. This feature's successful implementation demonstrated our software's adaptability and responsiveness to user input.

What worked well?

- **Flexibility and Scalability:** The use of JSON for data storage proved to be exceptionally flexible and scalable. JSON files are easy to work with, and their human-readable format makes debugging and manual editing straightforward when necessary. This approach also seamlessly supports the addition of new decks or questions, as it only requires the creation or modification of lightweight JSON files.
- **Modularity and Clean Codebase:** The division between deckmainwindow.cpp for handling deck management and mainwindow.cpp for question management within decks led to a more modular and maintainable codebase. This separation of concerns ensured that modifications in one area of the application (e.g., deck management) could be made with minimal impact on other areas (e.g., question handling).
- **Robust Data Handling:** Despite initial challenges, our JSON structure and handling mechanisms proved robust enough to support the complex requirements of updating difficultyRating dynamically. The seamless serialization and deserialization of updated question objects into and from JSON format worked exceptionally well, ensuring data integrity and consistency across sessions.
- **Efficient Data Model Updates:** The ability to update difficultyRating in real-time during a test and reflect these changes immediately in the corresponding JSON files was a testament to the efficient design of our data models and update mechanisms. This efficiency minimized latency and ensured a smooth user experience during tests.
- **User Interface (UI):** The success of user interface(UI) can be attributed to its strategic utilization of multiple windows to enhance feature accessibility and user engagement. This design approach ensured that upon selecting a specific functionality, users were presented with a dedicated window for that feature rather than overcrowding a single window with multiple functionalities. The design played a pivotal role in elevating the user experience by promoting a clean, organized and intuitive interaction environment. This methodology effectively mitigates cognitive overload by compartmentalizing features and content, allowing users to focus on the task at hand without unnecessary distraction.
- **Teamwork/ Communication:** The project's success was significantly bolstered by exemplary teamwork and communication within the team. This synergy facilitated a collaborative environment where ideas were freely exchanged, changes were collectively addressed, and solutions were effectively implemented. The foundation of our success lies in the deliberate strategies adopted to foster clean, open and effective communication channels among team members, ensuring that everyone was aligned with the project's goals, responsibilities and timelines.

What was found to be particularly useful?

- **Rapid Development and Iterative Testing:** The decision to use JSON files for storing decks and questions enabled rapid development and testing cycles. Changes to the data structure could be quickly implemented and tested without the need for complex database migrations or schema updates (as required in SQL). This agility was particularly useful during the iterative development and testing phases of the project.
- **Ease of Data Serialization and Deserialization:** The project's reliance on JSON for data storage capitalized on the ease of serialization and deserialization provided by libraries such as Cereal. This choice streamlined the process of saving and loading data, reducing the amount of boilerplate code and lowering the potential for errors.
- **User Engagement and Retention:** By allowing users to choose the difficulty rating of questions, we not only improved the accuracy of our content's difficulty levels but also increased user engagement and retention by showing the question repeatedly (as per the Space Based Algorithm).

What design decisions contributed to the success of the project?

- **Development of Required Features:** One of our key design decisions was prioritizing the development of the core functionalities over the optional and wishlist features. This focus ensured that we developed a solid foundation for Intellecto, which was crucial for its reliability and performance. Additionally, the decision to use C++ for the backend, complemented by JavaScript frameworks for dynamic UI elements, contributed to the robustness and scalability of our application.
- **Using Multiple Windows for User Interface:** The implementation of multiple windows in the user interface(UI) significantly contributes to the success of a project by enhancing user experience through improved navigation and organization.
- **Integration of Polymorphic Serialization:** The design decision to integrate polymorphic serialization, as seen with the registration of TextQuestion with Cereal, allowed for a flexible and extendable system capable of handling various question types. This approach ensures the application can be easily extended to support new types of questions or content without significant refactoring.
- **Flexible JSON:** Our decision to adopt JSON for our data storage was crucial since it could accommodate additional properties like difficultyRating without requiring extensive refactoring. This foresight ensured that we could enhance our application's features without compromising existing functionalities.
- **UI-Driven Design:** The careful design of the UI and its interaction with the underlying data models (e.g., the clear separation between deck and question management) contributed significantly to the project's usability and success. The intuitive interface design allowed users to easily navigate through decks, add or edit questions, and take tests, ensuring a positive user experience.
- **Intuitive UI for Difficulty Rating:** The design decision to include intuitive UI elements for setting and updating question difficulty ratings contributed significantly to this feature's success. By making it easy for users to adjust difficulty levels, we ensured the accuracy of the times the questions had to be reviewed after the current review (according to our algorithm).

In conclusion, the project's success can largely be attributed to the thoughtful integration of flexible, scalable data management strategies, the use of JSON for data storage, and a modular, UI-driven design approach. These decisions facilitated a smooth development process and resulted in a robust, user-friendly application capable of meeting the project's goals and accommodating future expansion.

Key Problem Areas

What went wrong?

Design decisions in a project can significantly impact the success and smooth execution of the project. In our case, the initial decision to utilize a single JSON file for managing all decks and their flashcards within our application was one such critical design choice that presented substantial challenges. Additionally, we had underestimated the technical complexity of integrating JSON with QT and C++, as well as the implementation of the spaced repetition algorithm.

What project processes didn't work well?

- **Inflexible Design Approaches:** Initially committing to a single JSON file for all decks and flashcards highlighted an inflexibility in the project's design approach. This decision led to a series of complications that were not only technical but also procedural, as it forced us into a corner with limited scalability and increased risk of data corruption.
- **Lack of Early Testing Across Different Devices:** The project faced challenges in ensuring consistent performance across different operating systems (OS) due to the lack of early and regular cross-platform testing not being part of the initial project methodology. This oversight required us to allocate additional resources to address compatibility issues, impacting the project timeline and workload distribution.

Technical Challenges Encountered

- **Cross-Platform Compatibility:** Ensuring that the application performed seamlessly across Mac, Linux, and Windows platforms emerged as a significant challenge. This required a broad range of testing and optimization efforts to address compatibility issues, which not only consumed considerable time but also introduced delays in the development process.
- **Complexity in Data Storage and Manipulation:** The decision to use a single JSON file for data storage presented multiple challenges, including difficulties in managing nested and relational data, increased risk of data corruption, and limitations in scalability. The lack of querying capabilities compounded these challenges, making it difficult to efficiently manage and manipulate the data.
- **Integration of Technologies:** The integration of JSON with QT and C++ presented its own set of challenges, requiring us to invest substantial effort into understanding and implementing this aspect of the project. Additionally, seamlessly incorporating a spaced repetition algorithm into the user interface proved to be more complex than anticipated, indicating a possible underestimation of the technical challenges involved.

Design Decision Challenges

- **Complexity in Data Management:** Managing all decks and their flashcards within a single JSON file made it exceedingly difficult to extract unique questions specific to each deck. This complexity arose from the inherent limitations of JSON as a data storage format when handling nested and relational data, especially without the support of querying capabilities present in traditional databases.
- **Increased Risk of Data Corruption:** With all data consolidated in a single file, any error in data manipulation—whether adding, editing, or deleting information—carried the risk of corrupting the entire dataset. This risk was particularly pronounced given the manual handling required for editing JSON files without the safeguards provided by database management systems.
- **Impediments to Scalability:** As the project grew, the single-file approach likely became unsustainable. Scalability concerns would emerge, not just in terms of file size but also in the complexity of managing and navigating the data structure efficiently. This would directly impact the application's performance and user experience.

Effects/Impact

- **Delayed Progress:** The difficulties in extracting unique questions and managing deck-specific data likely led to delays. These delays would stem from the additional development time required to implement workarounds for data extraction and management. It potentially threatened our deadline to deliver the project on time.
- **Increased Error Rate:** The heightened risk of data corruption also lead to an increased error rate in data manipulation tasks. This scenario not only impacted the integrity of the application's data but also its reliability from the user's perspective.
- **Development Frustrations:** Encountering such fundamental issues was demoralizing for the team, potentially affecting morale and productivity. The technical debt accumulated from patching the initial design choice also lead to frustrations.

Corrective Actions

- **Decentralization of Data Storage:** The decision to create new JSON files for each new deck and save its unique questions within its own file was a significant pivot. This approach simplified data management by isolating deck data, thereby reducing the complexity of data operations and minimizing the risk of widespread data corruption.
- **Enhanced Data Manipulation:** By decentralizing data storage, our team was able to implement more straightforward mechanisms for adding, editing, and deleting decks and their contents. This change likely improved development efficiency and reduced the likelihood of errors.
- **Improved Scalability and Performance:** Separating the decks into individual files improved the scalability of our application. It became easier to manage a growing number of decks and flashcards without negatively impacting the application's performance.
- **Implementing Cross-Platform Testing Early:** The experience underscored the importance of incorporating cross-platform compatibility tests early in the development cycle, ensuring that the application performs consistently across all targeted platforms.
- **Technical Skill Enhancement:** Our team invested time in enhancing their technical understanding and skills, particularly in integrating JSON with QT and C++, which was crucial for overcoming the technical challenges encountered.
- **Adapting Project Methodologies:** Recognizing the limitations of the initial project processes, our team became more flexible in their design approach, incorporating regular reviews and adjustments to the project plan to better accommodate technical complexities and ensure scalability.

This strategic shift in the data management approach and understanding of data storage illustrates the importance of adaptability in project management and software development. By recognizing the limitations of our initial design decision and taking decisive action to address these challenges, our team was able to mitigate potential negative impacts on our project and steer it back towards a path of successful implementation.

Lessons Learned

Throughout the journey of developing Intellecto, our team navigated a landscape rich with challenges and opportunities for growth. This experience has been a crucible of learning, offering profound insights into both the nuances of software development and the dynamics of effective team collaboration. Reflecting on this journey, several key lessons have emerged, shaping our perspectives as budding software engineers and team members.

Key Takeaways

1. **Continuous Testing:** A key takeaway from this project would be the importance of early and continuous testing, especially when integrating complex algorithms with user interfaces. This lesson was vividly illustrated when we initially encountered unexpected behaviours in our spaced repetition algorithm during user trials. There was confusion over the scheduling of review sessions, a problem that could have been identified and addressed much earlier had we implemented a more rigorous testing regimen from the start. This experience highlighted the critical need to intertwine development with continuous testing to ensure both the algorithm's efficacy and its seamless integration into the user interface, ultimately enhancing user satisfaction and trust in the application.
2. **Effective Communication:** We also learned the significance of clear communication and equitable workload distribution within the team to prevent bottlenecks and ensure smooth progress. Regular check-ins facilitated a transparent work environment where challenges were promptly addressed, and everyone was aware of their responsibilities and the timeline for execution. This practice of open communication ensured that no team member felt isolated, fostering a cohesive and supportive team dynamic. In future endeavours, prioritizing communication and maintaining a transparent workflow will remain a top priority.
3. **User-Centered Design:** Another key takeaway was the significance of maintaining a user-centered approach throughout the development process. Engaging with potential users early and often provided invaluable insights that shaped the direction and features of Intellecto. This experience taught us the importance of building with the user in mind, a principle that will guide our approach to software development moving forward.
4. **Technical Proficiency and Problem-Solving:** On a technical level, the project challenged us to apply our programming skills and object-oriented design principles in a real-world context. We learned the importance of writing clean, maintainable code and designing scalable systems—skills that are fundamental to our growth as software developers. Facing and overcoming technical challenges also enhanced our problem-solving abilities, teaching us to approach complex issues with perseverance and creativity.
5. **Iterative Development and Flexibility:** One of the most impactful lessons learned was the value of iterative development and flexibility in our work. Adopting agile methodologies underscored the importance of being adaptable, allowing us to pivot or adjust features based on user feedback and testing outcomes. This approach not only enhanced the quality of Intellecto but also ingrained in us a mindset of openness to change—a mindset that we would carry forward into future projects.

What could be done differently

1. **Data Management Approach**: Opting for a more scalable and flexible data management solution from the start would mitigate many of the issues faced. Shifting from a single JSON to seperate JSONs for each deck took up a lot of time and resources. Exploring database options or more sophisticated JSON structures could provide better support for the application's needs.
2. **Incorporate Testing Early**: Implementing cross-platform and device compatibility testing as an integral part of the development cycle would help identify and resolve issues much earlier, saving time and resources.
3. **Iterative Design and Feedback Loops**: Engaging users in the design process through regular feedback and testing sessions could guide the development more effectively, ensuring that the application meets user needs and expectations.

What would be done the same way

1. **Decentralization of Data Storage**: This approach proved to be a significant improvement in managing data complexity and enhancing the application's scalability and performance.
2. **Modular Codebase**: The division of responsibilities within the codebase facilitated easier management and adaptability of the project, a practice that should be continued in future projects.
3. **Team Collaboration and Communication**: Effective teamwork and communication were pivotal to overcoming challenges, making it essential to maintain these dynamics in any project setting.
4. **Dynamic Difficulty Rating System**: The implementation of a system that allows users to dynamically adjust the difficulty rating of flashcards during review sessions proved to be a significant success. This feature not only personalized the learning experience but also enhanced the effectiveness of the study sessions by adapting the content to the user's proficiency level in real-time. Keeping such a user-responsive mechanism in future projects would continue to add substantial value by tailoring the learning process to individual needs, thus enhancing user engagement and retention.
5. **Spaced Repetition Algorithm**: The core functionality of the app, the spaced repetition algorithm, stands out as a pivotal feature for its success. This algorithm's ability to schedule reviews based on the user's memory of each flashcard item, ensuring that information is reviewed at optimal intervals for memory retention, is fundamental. The effectiveness of this approach in improving long-term retention of information makes it an indispensable aspect of the application. Continuing to refine and incorporate this algorithm in similar projects would ensure that the educational tools developed are not only innovative but also grounded in proven cognitive science principles, thereby maximizing learning outcomes.

Technical Challenges

Despite our achievements, we encountered several technical challenges. The most significant was ensuring the application's performance across different devices since the team had different devices like Mac, Linux and Windows. This required extensive testing, optimization and frequent changes. Additionally, implementing and logic and integrating the spaced repetition algorithm seamlessly with the user interface proved to be more complex than anticipated. The implementation of the Data storage functionality which we completed using JSON took a lot of time and threatened our project deadline at one point. Moreover, understanding JSON integration with QT and C++ was another area which required us to put in a lot of effort.

New Processes or Best Practices Developed

1. **Decentralized Data Management**: This method emerged as a best practice for handling complex data structures in applications requiring high levels of data manipulation and storage.
2. **Regular Code Reviews and Pair Programming**: Implementing regular code reviews and encouraging pair programming sessions fostered a culture of collaboration and continuous improvement. These practices not only improved code quality but also facilitated knowledge sharing among team members, contributing to a more cohesive and skilled team.
3. **Doxygen Commenting and Documentation**: The importance of thorough documentation was highlighted, particularly for complex projects. We learnt tools like Doxygen for documenting the codebase can which aided in maintenance and future development. Adopting a consistent commenting style and maintaining up-to-date documentation are practices that greatly benefits project sustainability and team collaboration.

Areas for Improvement

1. **Data Structure Complexity**: Further refining the data management strategy to address the inherent limitations of the chosen storage format could enhance efficiency and reduce risks of corruption.
2. **Advanced User Interface Accessibility**: Building upon the initial success, there's room to further improve the app's accessibility features. Making the app more accessible to people with various disabilities, such as implementing voice commands or enhancing visual contrast, would not only broaden the user base but also align with inclusive design principles.
3. **Comprehensive Cross-Device Synchronization**: Ensuring that users can seamlessly switch between devices without losing progress or facing inconsistencies in their learning material is crucial. Improving cloud synchronization and account management features would enhance the user experience, making the platform more versatile and convenient.

Future Project Considerations

Looking forward, we would focus on enhancing the app's accessibility features and exploring the integration of advanced technologies such as speech-text recognition. These improvements would not only make Intellecto more inclusive but also extend its usability and appeal.

This project has indeed sparked a deeper interest in the field of educational technology. The challenges we faced and overcame have equipped us with a broader perspective on software development, particularly in creating applications that have a real-world impact. Given the chance, we would eagerly embark on similar projects in the future, armed with the knowledge and experience gained from this endeavour, especially in data management, cross-platform development, and user interface design, which when combined provide a strong foundation for tackling future projects with similar scopes or technical requirements more effectively.

User Story Analysis

Introduction

In the development of any software project, particularly those focused on educational technology, assessing the outcome of implemented features against predefined user stories and acceptance criteria is crucial. This process not only validates the functionality and usability of the product but also highlights areas of success and opportunities for improvement. The distinction between what was achieved and what was deferred due to time constraints provides valuable insights into the project's current state and potential directions for future development or adjustment. This analysis revisits the pass/fail outcomes of the project's features, highlighting the successes achieved under time constraints and outlining how the optional features, though not yet completely implemented due to these constraints, represent opportunities for future enhancements to enrich the user experience.

Required Features:

Spaced Repetition Algorithm		
Priority: Highest	**Story points:** 15	**Assignees:** 3 People * 5 days
User Story: As a user, I can have my review sessions scheduled based on difficulty of each flashcard so that I can improve retention over time.		
Acceptance criteria: • The algorithm schedules review sessions based on user performance, with easily answered flashcards appearing less frequently than difficult ones. • Users receive feedback on when to expect flashcards for review again. • The scheduling mechanism is adaptable based on ongoing performance metrics.		
Acceptance Testing: Users rate their memory of a flashcard, and the system schedules the next review appropriately. The scheduling logic is verified through repeated reviews. • A flashcard reviewed as easy is scheduled further out. (Pass) • A difficult flashcard appears more frequently. (Pass)		

Flashcards Creation		
Priority: Highest	**Story points:** 15	**Assignee:** 3 people * 5 days
User Story: As a user, I can create flashcards with a front for questions and a back for answers so that I can customize my learning materials.		
Acceptance criteria: • Users can create flashcards with distinct fronts (questions) and backs (answers). • Flashcards support text and, optionally, images or links. • Created flashcards are editable and deletable.		
Acceptance Testing: User creates a flashcard, inputs content for both sides, and saves it successfully to a deck. • Flashcards can be created with customizable questions (fronts) and answers (backs). (Pass) • Flashcards can be updated. (Pass) • Flashcards can be deleted. (Pass) • Flashcards are saved and retrievable. (Pass)		

Deck Organization		
Priority: High	**Story points:** 6	**Assignee:** 2 people * 3 days
User Story: As a user, I can organize my flashcards into decks or categories so that I can study efficiently by topic.		
Acceptance criteria: • Users can create name and organize flashcards into decks. • Decks can be edited, including adding, removing, or reordering flashcards. • Users can navigate between different decks easily.		

Acceptance Testing: User creates a new deck, assigns flashcards to it, and retrieves these organized sets for review.

- New decks can be created. (Pass)
- Flashcards can be grouped into user-defined decks. (Pass)
- Existing decks can be renamed. (Pass)
- Decks can be edited. (Pass)
- Decks can be deleted. (Pass)

Review Sessions		
Priority: High	**Story points:** 6	**Assignee:** 3 people * 2 days
User Story: As a user, I can review flashcards, rate their difficulty, and have the app schedule the next review so that I can study more effectively.		
Acceptance criteria: - During review sessions, users can rate the difficulty of flashcards. - The system tracks user progress and adjusts future review sessions accordingly. - Users can pause and resume review sessions without losing progress.		
Acceptance Testing: During a review session, the user rates flashcards, and the app adjusts subsequent review timings accordingly. - User can rate flashcard difficulty during review. (Pass) - Next review session is scheduled based on difficulty rating. (Pass)		

Data Storage		
Priority: High	**Story points:** 9	**Assignee:** 3 people * 3 days
User Story: As a user, I want my flashcards, review history, and progress to be stored so that I can track my learning over time.		
Acceptance criteria: - User data, including flashcards, review history, and performance metrics, is securely stored in the database. - Data integrity is maintained, with no loss on app updates or migrations.		
Acceptance Testing: User data persists across sessions, with no loss or corruption of data. - Data is persistently stored and secure. (Pass) - Unique JSON files are created for individual decks. (Pass) - A separate JSON file is created to store the name of the decks created. (Pass) - JSON files can be renamed, edited, and deleted. (Pass)		

Graphical User Interface (GUI)		
Priority: Highest	**Story points:** 9	**Assignee:** 3 people * 3 days
User Story: As a user, I can interact with a clean, intuitive UI so that creating, reviewing, and organizing flashcards is easy and straightforward.		
Acceptance criteria: - The GUI is intuitive, with clear navigation paths for creating, reviewing, and organizing flashcards. - Key functionalities are easily accessible from the main interface. - The design is responsive, ensuring usability across devices.		
Acceptance Testing: Users can navigate the app easily, perform essential functions without confusion, and express satisfaction with the interface usability. - UI is user-friendly and facilitates easy navigation. (Pass) - Key functionalities (create, review, organize) are easily accessible. (Pass)		

Optional Features:

Advanced Statistics and Reporting		
Priority: Medium	**Story points:** 4	**Assignee:** 2 people * 2 days
User story: As a user, I can view detailed analytics of my performance so that I can track my improvement over time.		
Acceptance criteria: • Detailed analytics on user performance are available, including progress over time and areas of strength or weakness. • Reports are generated after each review session. • Users can set goals and track their progress towards them.		
Acceptance Testing: After a review session, the user receives a report detailing their performance, which accurately reflects their learning progress. • Analytics show user performance and progress. (Pass) • Reports are available after each study session. (Pass)		

Gamification Elements		
Priority: Low	**Story points:** 6	**Assignee:** 3 people * 2 days
User story: As a competitive learner, I can participate in quiz challenges and earn rewards so that I am motivated to keep learning.		
Acceptance criteria: • Quizzes and challenges are available for users to test their knowledge. • Users earn points or badges for accomplishments. • Leaderboards display user rankings, encouraging competition.		
Acceptance Testing: Users engage in quiz challenges, earn badges, and appear on scoreboards, enhancing their learning motivation. • Users can compete in quizzes. (Fail) • Achievements and scores are tracked and rewarded. (Fail)		

Collaborative Feature		
Priority: Low	**Story points:** 10	**Assignee:** 5 people * 2 days
User story: As a user, I can share decks and collaborate on flashcards with friends so that we can enhance our learning experience together.		
Acceptance criteria: • Users can share decks with others and collaborate in real-time. • Changes made by collaborators are tracked and can be reviewed. • Permissions can be set for who can edit or view shared decks.		
Acceptance Testing: Users successfully share a deck with another user and collaborate on creating a new flashcard. • Decks can be shared and edited by multiple users. (Fail) • Collaborative sessions can be initiated. (Fail)		

Import/Export Functionality		
Priority: Medium	**Story points:** 6	**Assignee:** 2 people * 3 days
User story: As a user, I can import and export decks so that I can share materials and have backups.		

Acceptance criteria:

- Users can import decks from external sources or export their decks in a standardized format.
- The import/export process is clear and supports common file formats.
- Integrity of flashcards is maintained during the import/export process.

Acceptance Testing: A user exports a deck, then imports it into another account, with all data intact and correctly formatted.

- Users can import/export decks in a common format. (Fail)
- The process is straightforward and user-friendly. (Fail)

Customization Options		
Priority: Low	**Story points:** 3	**Assignee:** 3 people * 1 day
User story: As a user, I can customize flashcards with different templates, fonts, and colors so that I can personalize my learning experience.		

Acceptance criteria:

- Users can customize the appearance of flashcards and the app interface, including themes, fonts, and colors.
- Customization settings are saved and applied consistently across sessions.
- Users can preview changes before applying them.

Acceptance Testing: Users customize a flashcard's appearance and save their preferences for future cards.

- Flashcards are highly customizable. (Fail)
- Changes are reviewable before saving. (Fail)

Conclusion

Our major focus on successfully delivering our project's required functionalities within the given timeframe has resulted in a strong foundation for our Intellecto application, capable of supporting effective and personalized learning. The deferred implementation of some of the optional features, while a reflection of time constraints, presents a roadmap for the application's future development. These features, once incorporated, can significantly enhance user engagement, customization, and collaborative learning opportunities. Moving forward, we have a clear direction for incremental enhancements that can transform the application into a more comprehensive and interactive learning tool. Our strategic decision to prioritize core functionalities ensures that users have a reliable and effective tool from the outset, with the promise of more engaging features to come as the application evolves.