



Imam Muhammad Ibn Saud Islamic University
Computer Science Department
College of Computer and Information Sciences



CS290 Project

Fill The Blank

Course Title: Software Engineering

Instructor: Ibrahim Abdulmonem

Semester: Second.2023

Section : 173

Abdulrahman Al-Afif	443015728
Hamad Al-Moqbel	443017887
Waleed Drweesh	442021115
Ziyad Al-Rushud	441017200
Abdurabu Saleh	443018831
Mashhour Al-Eid	444005220

Scope of the project:

The scope of the project is the design and development of "*Fill the blank*", a digital educational game aimed at teaching the English alphabet to children. The game will provide interactive exercises where children can fill in missing letters in words to improve their understanding of English letters.

Objectives:

Engage children in a fun and interactive learning experience. Help children learn and remember English letters. Improve children's spelling and word formation skills. Respond immediately and reinforce for correct answers. Observe and record children's progress and achievements.

Target Audience:

The target audience for the game is children between the ages of 4 and 8 who are learning English as a primary or secondary language. This game will cater to beginners and intermediates in English proficiency.

Client(s):

A customer for this project could be an educational institution, a language learning platform, or a developer of a children's educational app. The specific customer will depend on the context and the purpose of the game.

The vision of the "*Fill the blank*" platform is to create a fun and effective teaching tool that helps children learn the English alphabet in a fun and interactive way. The space will offer features and capabilities designed to enhance the learning experience and meet the needs of young students. Key components of the platform's vision include.

Vision:

The vision of the "*Fill the blank*" platform is to create a fun and effective teaching tool that helps children learn the English alphabet in a fun and interactive way. The space will offer features and capabilities designed to enhance the learning experience and meet the needs of young students. Key components of the platform's vision include.

Interactive games: The platform will provide a user-friendly and fun interface that engages children through interactive games. The game will display words with missing letters, and kids can use the virtual keyboard to *Fill the blanks* or drag and drop letters into the right places.

Continuous learning: The platform will offer a structured learning experience that starts with basic letters and progresses to more complex vocabulary. It will provide a range of problems to engage students at different stages of language development.

Educational Context: The platform will have libraries of vocabulary exercises, covering a wide range of vocabulary suitable for young learners. Careful vocabulary choices will coincide with early reading and phonics lessons to ensure educational relevance.

Feedback and reinforcement: The platform will provide children with immediate feedback for each word they complete, positive markers for correct answers, and instructions for incorrect ones. This form of feedback will help children understand their progress and encourage them to continue learning.

Progress Monitoring: The platform includes a progress tracking system that allows parents, teachers, guardians, or children to monitor their learning progress. And this could be included.

User requirements:

1. **Gameplay:**
 - Users should be able to play the *Fill the blank* game, making guesses to reveal a hidden word based on figure.
2. **User Interface:**
 - The game should have an intuitive and visually appealing user interface.
 - Clear display of the *Fill the blank* figures so the letters can be guessed, and the word with blanks.
3. **Word Selection:**
 - The game should randomly select words and figures related from a predefined list or category.
4. **Input Mechanism:**
 - Users should be able to input letters as their guesses.
 - The game should validate and handle user inputs, ensuring they are valid characters.
5. **Visual Feedback:**
 - Provide visual feedback for correct and incorrect guesses.
6. **Winning and Losing Conditions:**
 - Declare the player a winner if they successfully guess the word before the timer ends.
 - Declare the player a loser if they use up all allowed time for the level.
7. **Scoring and Progression:**
 - Implement a scoring system based on time taken to solve each puzzle.
 - Allow users to reset the game for a new round.
8. **Accessibility:**
 - Design the game with accessibility in mind, ensuring it is usable by individuals with various abilities.
9. **Language Support:**
 - If applicable, support multiple languages.
10. **User Engagement:**

- Implement features to keep users engaged, such as animations, sound effects, or additional game modes.
- 11. Instructions and Help:**
 - Provide clear instructions on how to play the game.
 - Include a help section or tutorial if needed.
- 12. Platform Support:**
 - Ensure that the game works seamlessly on both iOS and Android platforms.
- 13. Performance:**
 - The game should respond promptly to user inputs, maintaining a smooth experience.

Non-Functional Requirements:

1. Performance:

- Ensure responsive and smooth gameplay with minimal lag.

2. Compatibility:

- Support multiple devices, including desktops, tablets, and smartphones.

3. User Interface:

- Design an intuitive and visually appealing interface.

4. Security:

- Safeguard user information and prevent unauthorized access.

5. Language Support:

- Support multiple languages for a diverse user base.

6. Usability and Accessibility:

- Prioritize usability and include accessibility features for diverse users.

7. Scalability:

- Design the system to handle potential growth in users and features.

8. Offline Capability:

- Consider implementing offline capabilities for uninterrupted gameplay.

9. Error Handling:

- Provide clear feedback for user errors or unexpected issues.

10. Maintainability:

- Design a modular and maintainable codebase for ease of updates.

11. Scoring System:

- Optionally implement a scoring system to track player achievements.

Functional Requirements:

1. Word Selection:

- System randomly selects words from a predefined list or category.

2. User Input:

- Players input letters as guesses during the game.

3. Word Display:

- Display selected word with underscores for unknown letters.
- Update display for correct guesses.

4. *Fill the blank* figures

- Visually represent the score lowering as incorrect guesses accumulate.

5. Winning Condition:

- Declare player a winner if they guess the word before the timer is fully finished.

6. Losing Condition:

- Declare player a loser if they exceed allowed available time.

7. Game Reset:

- Provide an option for players to reset the game for a new round.

8. Timer display:

- The app may include a timer to add a time pressure element to the game.

9. User registration:

- The app should allow users to create an account or log in using their existing credentials.

System Requirements:

The expected system requirements for a game such as a *Fill the blank* game on iOS and Android can vary depending on the complexity of the game and the specific features implemented. However, here are some general guidelines for the expected system requirements:

For Android:

Operating System: Android 5.0 Lollipop or later (or the latest versions supported by your game)

Processor: Android devices with ARMv7 architecture or higher are commonly supported. Specific CPU architectures like ARM, x86, or MIPS should be considered based on your target audience.

Memory (RAM): 1 GB or higher, although 2 GB or more is recommended for smoother performance.

Storage Space: Similar to iOS, the required storage space can vary, but estimating 200 MB to 500 MB for the app installation is a good starting point.

Display and Resolution: Android devices come in various screen sizes and resolutions, so the game should be designed to adapt to different display configurations. Support for different aspect ratios and resolutions is important.

Internet Connectivity: is optional to place the player in the leaderboard.

Graphics: Support for OpenGL ES 2.0 or higher is common for Android devices to ensure smooth rendering of graphics and visual effects.

Input Methods: Touch screen input is expected, along with support for relevant gestures or interactions. Additionally, consider compatibility with gamepad controllers for a better gaming experience.

Login/Registration Page:

As required in the first phase we made a login/sign-up page screen.

Login Screen:

A mobile app mockup of a login screen. The status bar at the top shows the time 5:44 and various icons. A back arrow and the text 'Login' are at the top left. The main area contains two input fields labeled 'Username' and 'Password'. Below these is a light blue 'Login' button. At the bottom, there is a link that says 'Create a new account'. A red 'BUG' sticker is in the top right corner.

5:44

← Login

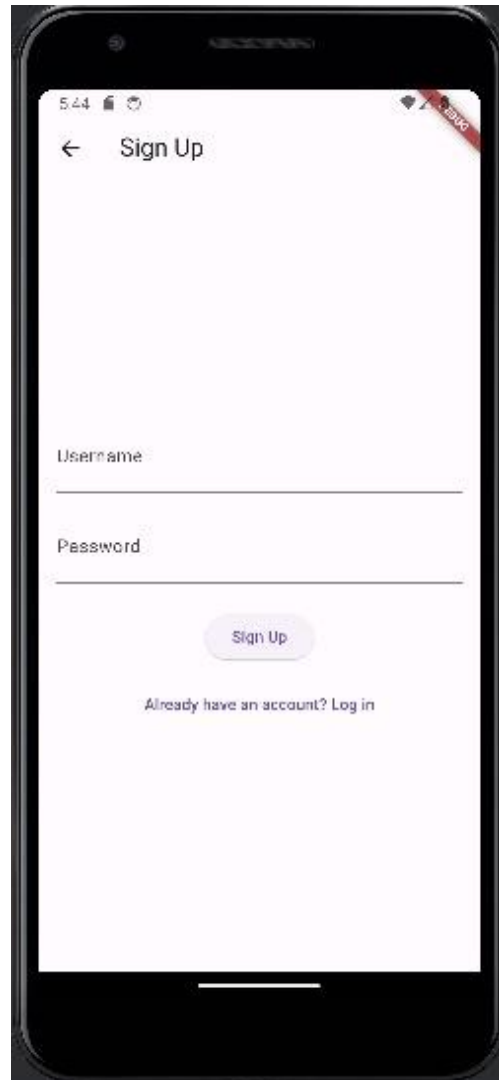
Username

Password

Login

Create a new account

Sign-up Screen:

A mobile app mockup of a sign-up screen. The status bar at the top shows the time 5:44 and various icons. A back arrow and the text 'Sign Up' are at the top left. The main area contains two input fields labeled 'Username' and 'Password'. Below these is a light blue 'Sign Up' button. At the bottom, there is a link that says 'Already have an account? Log in'. A red 'BUG' sticker is in the top right corner.

5:44

← Sign Up

Username

Password

Sign Up

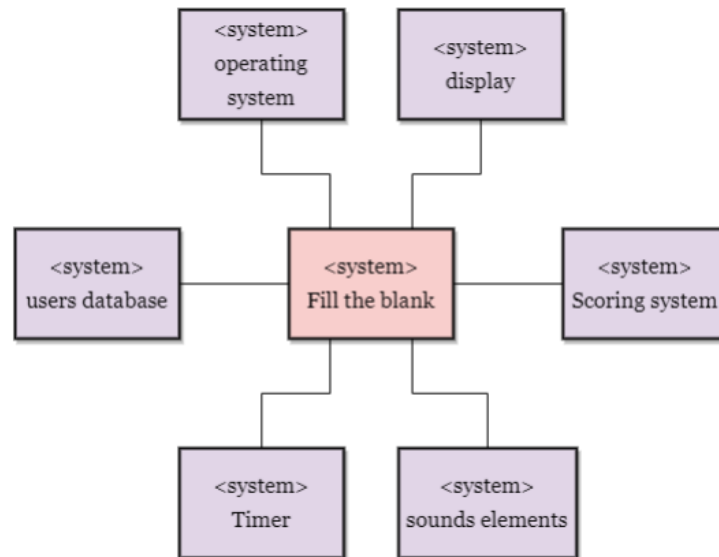
Already have an account? Log in

Source code is attached.

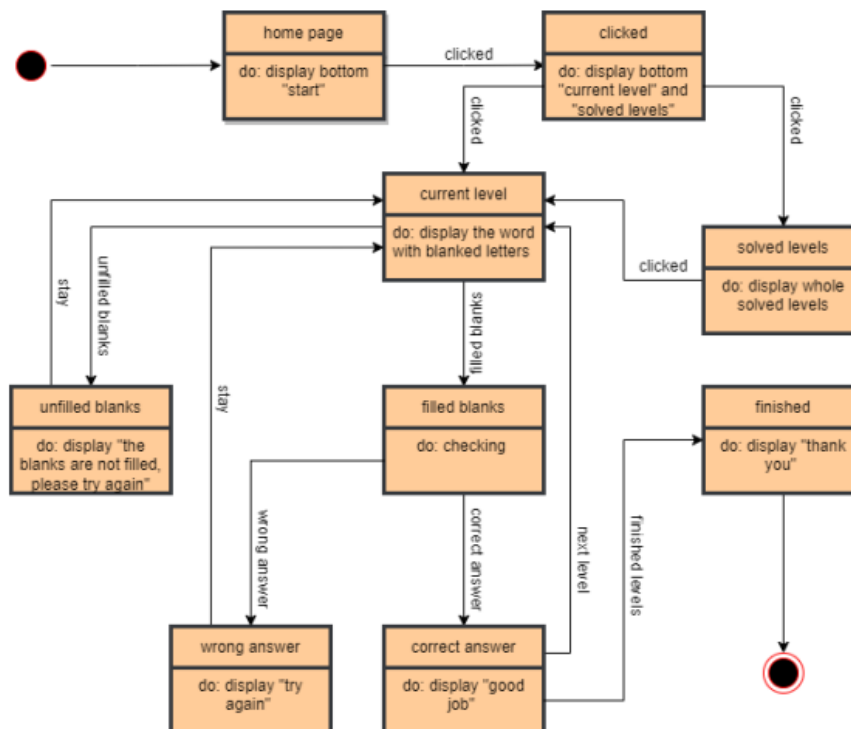
Phase 2:

System Models:

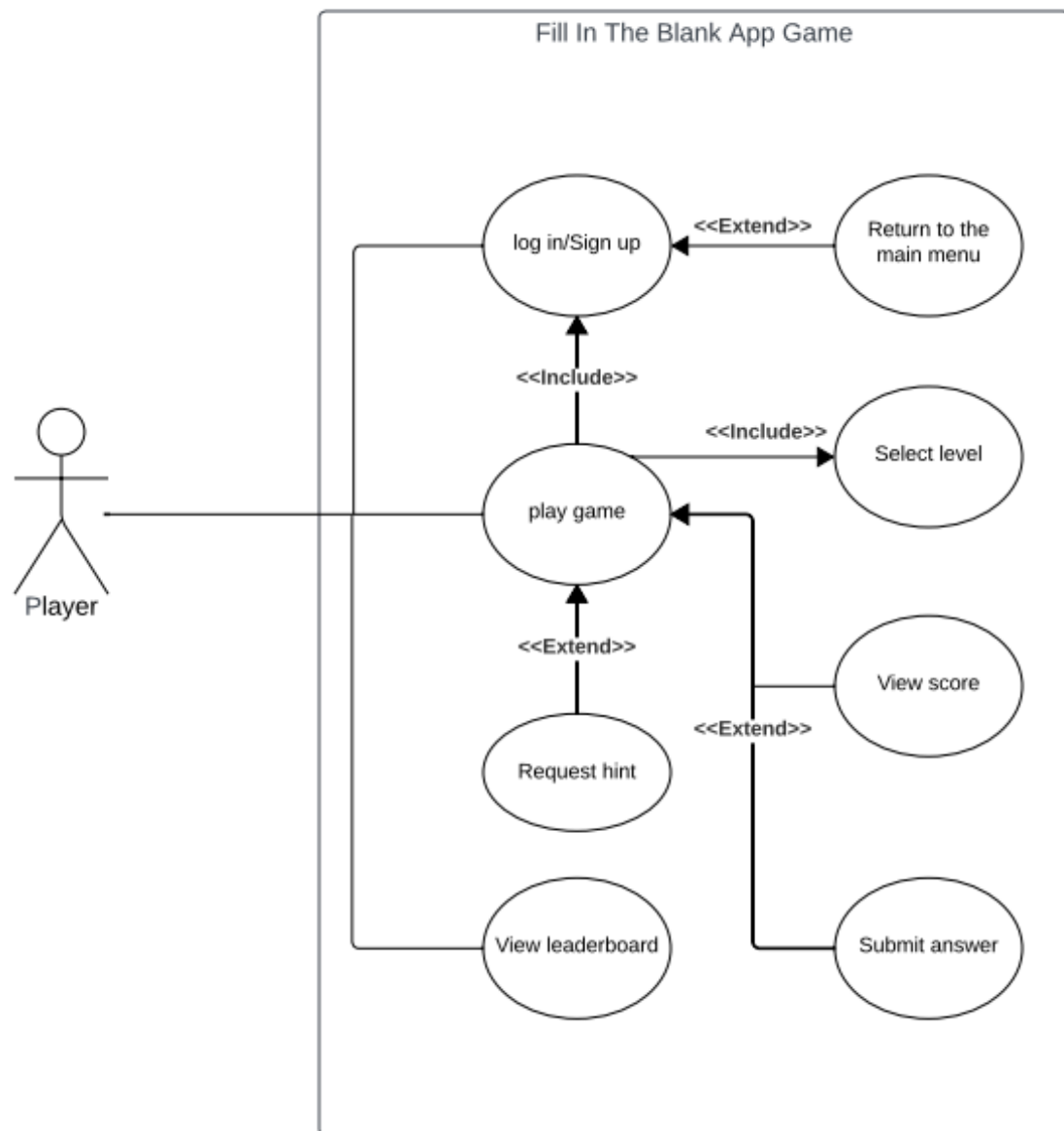
Context Diagram:



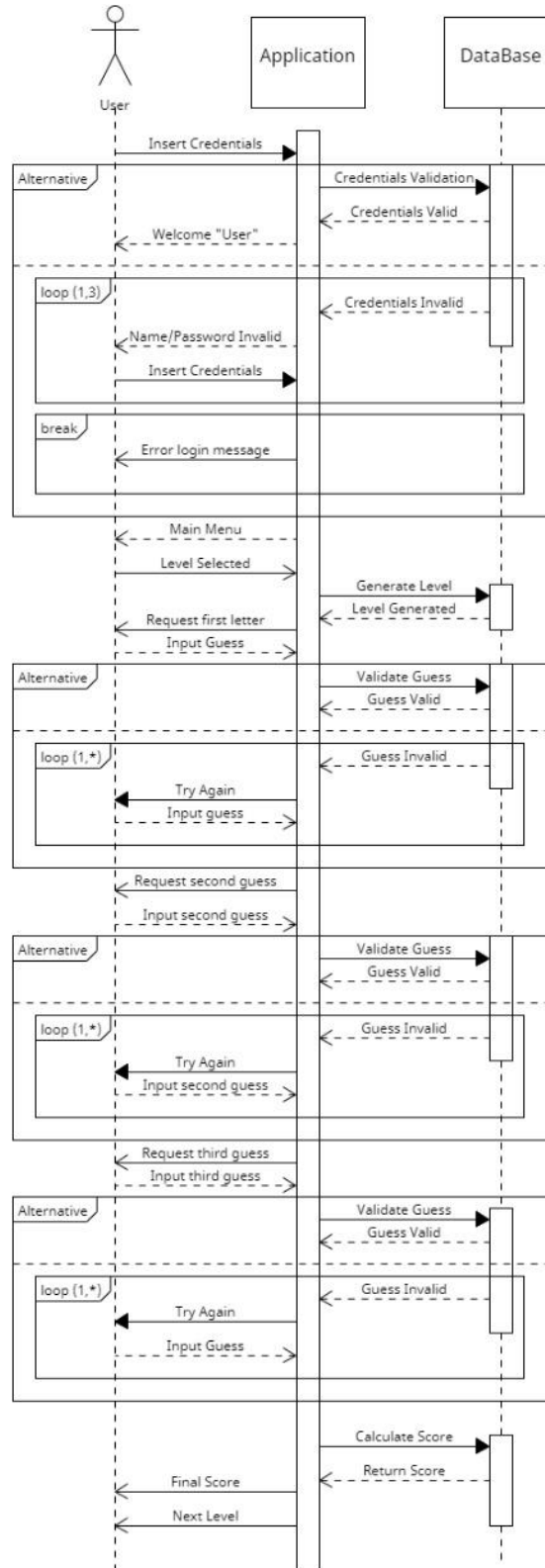
State Diagram:



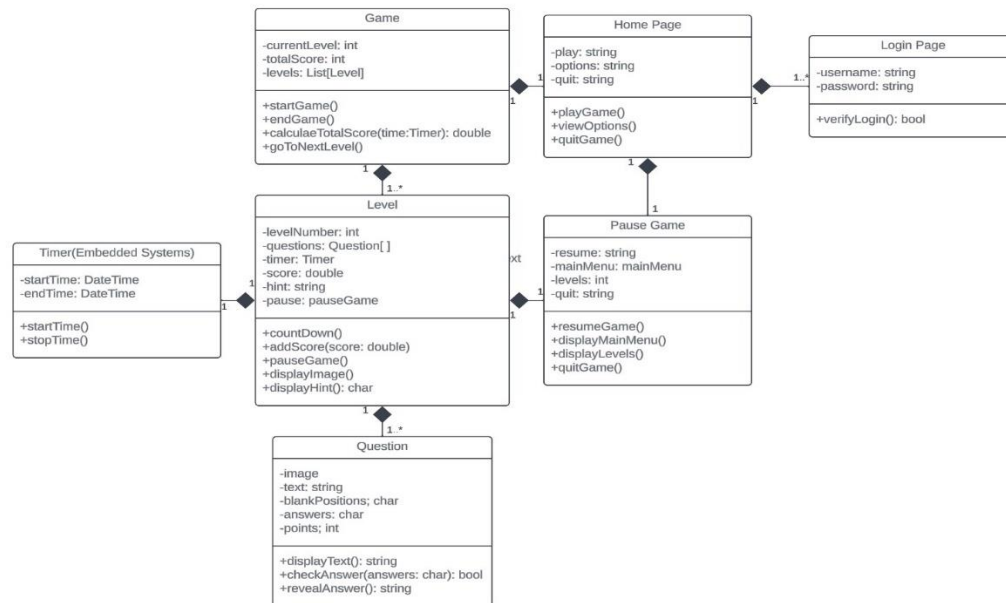
Use Case Diagram:



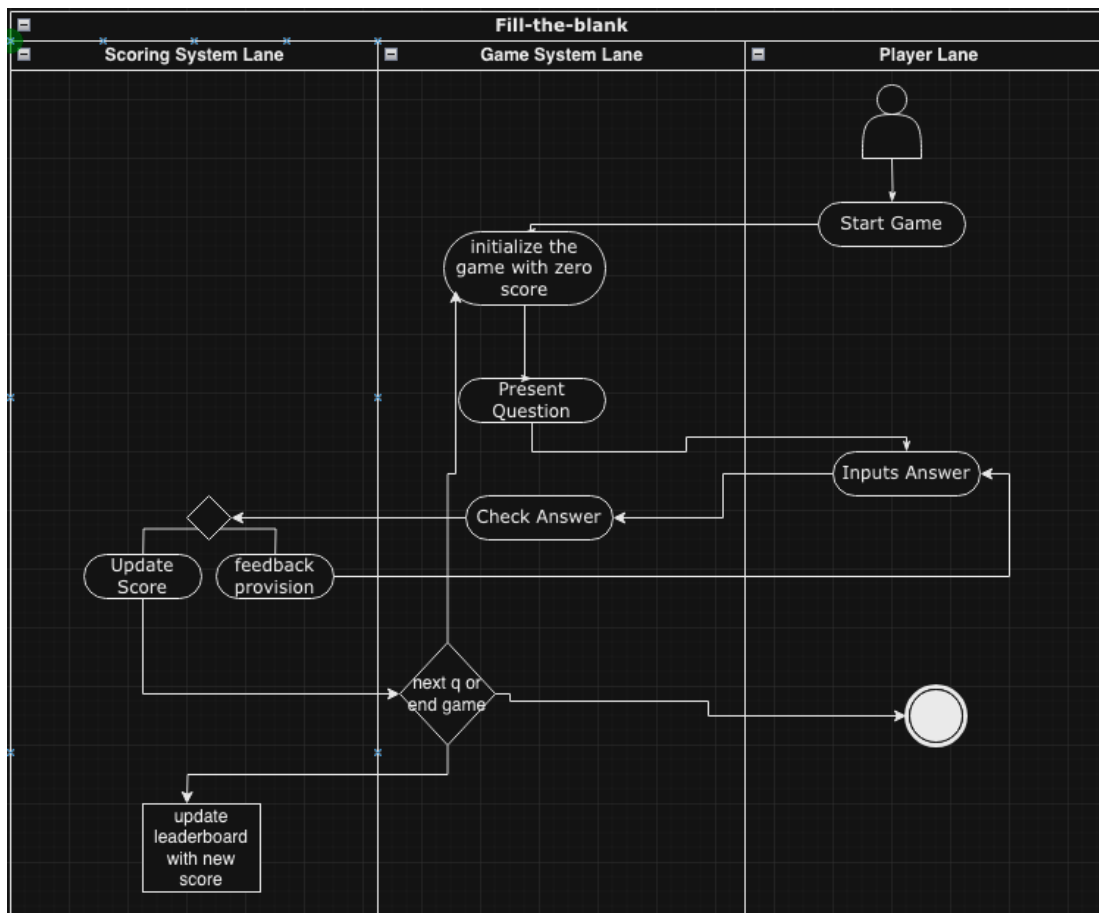
Sequence Diagram:



Class Diagram:



Activity Diagram:



Use Case Description:

Use Case: Play *Fill the Blank* Game

Goal:

The primary goal of this use case is to allow players to engage in the "*Fill the Blanks*" game, where they can enjoy guessing missing letters in words associated with images, earn scores, and progress through different levels.

Actors:

Player: The user who interacts with the game.

Preconditions:

The game application is installed and accessible on the player's device.

The player has launched the game.

Main Success Scenario:

- 1 .The player starts the game.
- 2 .The game presents a menu allowing the player to select a level.
- 3 .The player chooses a level, each associated with an image and a word with missing letters.
- 4 .The game retrieves level data from the database.
- 5 .The game displays the image with missing letters to the player.
- 6 .The player guesses the missing letters in the word.
- 7 .The game validates the player's guess.
 - If the guess is correct, the player's score increases, and the database is updated.
 - If the guess is incorrect, the player's score decreases, and the database is updated.
- 8 .The game provides feedback on the correctness of the guess.
- 9 .The player continues guessing until the word is complete or the available attempts are exhausted.

10 .The game checks if the level is complete.

- If the level is complete, the player progresses to the next level.
- If not, the player continues playing the current level.

11 .The game displays the result of the completed level, showing the player's score and any rewards earned.

12 .The player can choose to play another level or exit the game.

Postconditions:

The player's progress and score are saved in the database.

The player has the option to play another level or exit the game.

Alternative Flows:

If the player chooses to exit the game at any point, the game saves the current progress and exits.

Exceptions:

If there are technical issues or errors during database communication, the game displays an error message and may prompt the player to try again.

Special Requirements:

The game should have an intuitive user interface that allows players to easily navigate and interact with the levels.

Images and words associated with levels should be age-appropriate for the target audience (children).

Implementation report:

1. Introduction:

The purpose of this report is to outline the implementation details for a *Fill the blank* app game. This type of game involves presenting a sentence or phrase with one or more missing words or easy questions, and the player's objective is to Fill the blanks with the correct words to complete the sentence or solve the question. The report will cover key aspects such as game mechanics, user interface design, database management, and monetization strategies.

2. Game Mechanics:

- Sentence Generation: Develop an algorithm to generate a wide variety of sentences or phrases with missing words. The algorithm should ensure that the sentences are grammatically correct and engaging for the players.
- Difficulty Levels: Implement multiple difficulty levels to cater to players of different skill levels. Easy levels can have simpler sentences with fewer missing words, while harder levels can have more complex sentences or phrases.
- Hints and Clues: Provide players with hints or clues to assist them in filling in the blanks
- Time Limit: Introduce a time limit for hole levels to add a sense of urgency and challenge. the player will lose 10 seconds if they submit wrong.

3. User Interface Design:

- Clear and Intuitive Layout: Design a user-friendly interface that allows players to easily navigate through the game
- Input Mechanism: Implement a text input field where players can enter their answers. The input field should provide the letters listed in the code before.
- Visual Feedback: Provide immediate visual feedback to players upon submitting their answers. Highlight correct answers and provide hints or corrections for incorrect answers.

4. Database Management:

-To ensure a diverse and engaging gameplay experience, a robust sentence database will be created. The database will be expanded to include a wide range of categories such as general knowledge, pop culture, sports, and more. This expansion will provide players with a rich variety of sentences and phrases to complete, keeping the game fresh and appealing.

5. Monetization Strategies:

In future iterations of the *Fill the blank* app game, the following monetization strategies will be considered.

- In-App Purchases:

To enhance the gameplay experience, the option for in-app purchases will be implemented. Players will have the opportunity to buy hints, clues, or additional sentence packs to assist them in completing challenging levels. These in-app purchases will provide players with a sense of progression and customization, allowing them to tailor their gameplay experience to their preferences.

- Advertisements:

Non-intrusive advertisements will be incorporated into the app to generate revenue. Careful consideration will be given to the placement and frequency of ads to ensure they do not disrupt the gameplay experience. Advertisements can be displayed between levels or offered as an optional viewing for players who wish to earn in-game rewards. This monetization strategy will provide a source of revenue while maintaining a positive user experience.

6. Testing and Feedback:

- Internal Testing: Conduct thorough testing of the app to identify and fix any bugs, glitches, or usability issues.

- User Feedback: Solicit feedback from early users or beta testers to gather insights and make necessary improvements to the game mechanics, user interface, and overall gameplay experience.

7. Conclusion:

The implementation of a *Fill the blank* app game requires careful consideration of game mechanics, user interface design, database management, and monetization strategies. By following the outlined implementation details and conducting thorough testing, the resulting app can provide an engaging and entertaining experience for players. Regular updates and user feedback will be essential for maintaining and improving the game over time.

Phase 3:

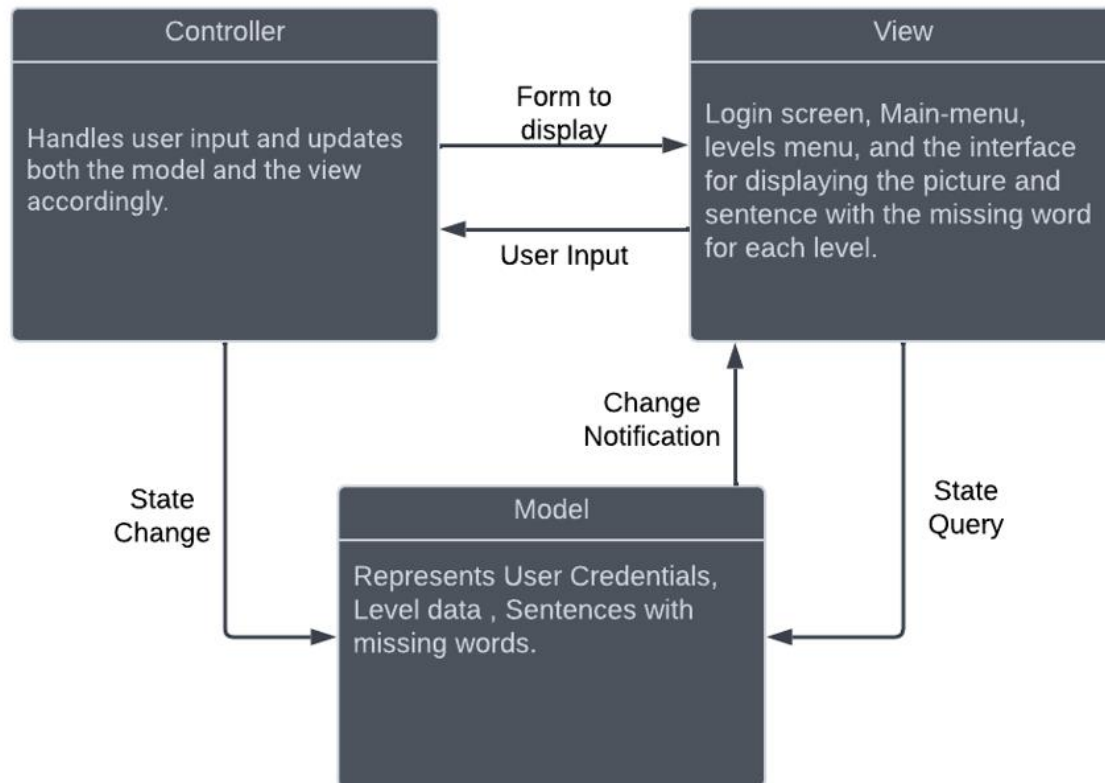
Architectural Pattern:

We Choose MVC Pattern design based on The choice of the Model-View-Controller (MVC) pattern for *Fill The Blank* was based on several factors:

- **Separation of Concerns:** The MVC pattern separates the application into three interconnected components: the Model, View, and Controller. This separation allows for a clear division of responsibilities, making the codebase easier to understand, maintain, and modify. In your game project, this means that the logic for managing game data (Model), displaying the user interface (View), and handling user input (Controller) are distinct and modular, which simplifies development and maintenance.
- **Scalability:** The MVC pattern promotes scalability by organizing the codebase in a way that new features or components can be added without extensive modifications to existing code. This is particularly beneficial for a game project like yours, where you may need to add new levels, game mechanics, or user interface elements over time.
- **Reusability:** Components in the MVC pattern are designed to be reusable, as each component has a specific responsibility and can be used across different parts of the application. In your game project, this means that views and controllers can be reused for different screens or levels, reducing redundancy and promoting code efficiency.
- **Testability:** The separation of concerns in the MVC pattern facilitates testing, as each component (Model, View, Controller) can be tested independently. This is crucial for ensuring the reliability and stability of your game project, as you can easily write unit tests for each component to verify its functionality.

Overall, the MVC pattern was chosen for *Fill The Blank* because it provides a structured and organized approach to development, promotes scalability and maintainability, encourages code reuse, and facilitates testing, all of which are essential considerations for building a successful and robust game application.

MVC Pattern Design:



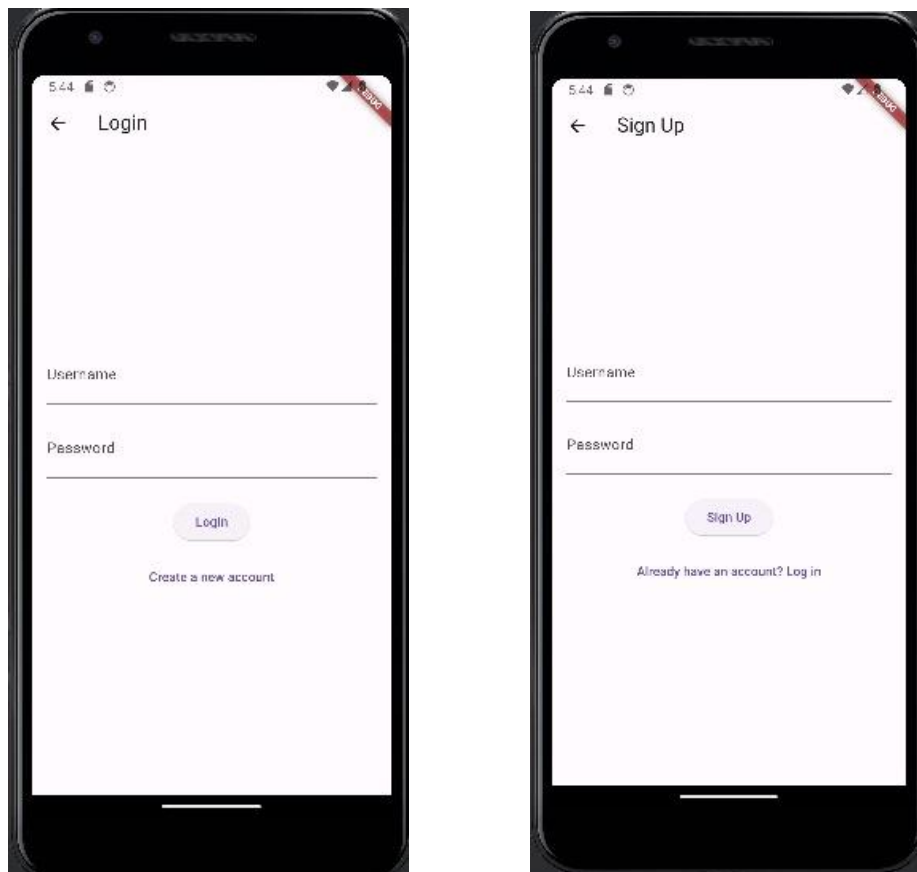
- 1) Model: Represents user credentials for login, level data, and the sentences with missing words for each level.
- 2) View: Represents the user interface components of **Fill The Blank**. This includes the login screen, main menu, levels menu, and the interface for displaying the picture and sentence with the missing word for each level.
- 3) Controller: Handles user input and updates both the model and the view accordingly. For example, the controller would manage actions such as logging in, navigating through menus, and submitting answers for each level.

Progression:

In a world where ideas and technology blend together, a group of students decided to create something special—an educational game app called ***Fill the blank***. Their adventure had three main parts, each with its own challenges and victories.

Phase 1: The Beginning – Planning

First, they had a big meeting to decide on their goal: to make a game that was both fun and educational. They formed a team and chose to make a ***Fill the blank*** game app. They spent time planning everything out, like what the game should do, who would want to play it, and what they needed to make it work. They even made a plan in Excel to keep track of how long each task would take. The first thing they made was the login screen, which was the start of bringing their game to life.



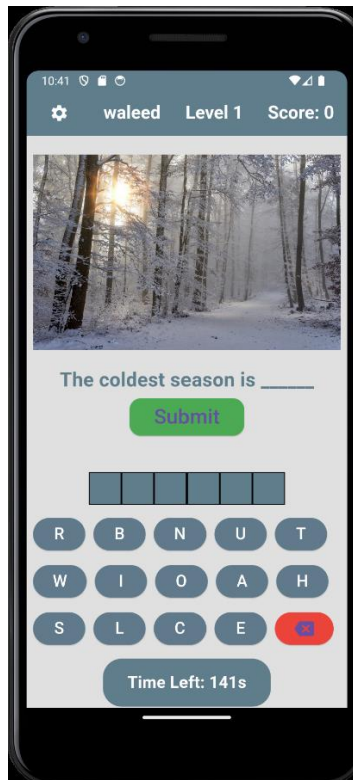
**This was the initial version of the login/registration page.*

Phase 2: Building the Game - Programming

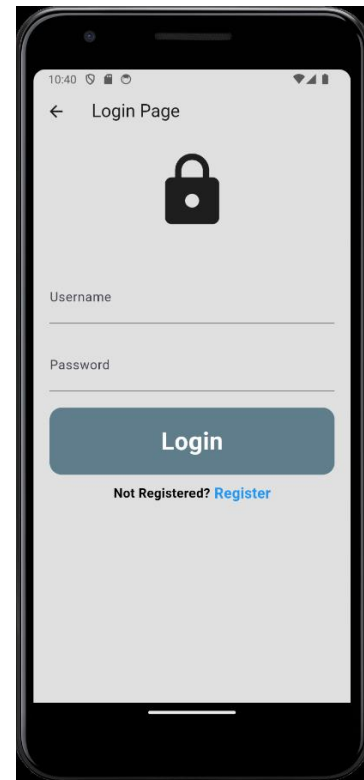
Next, they moved on to designing the game. This meant drawing out diagrams to show how the game would work, like maps of the game's actions and how users would move through the game. They divided up the work, with each person responsible for a different part of the game's design. They met online often to talk about how the game was coming along and to make sure everything was going smoothly. By the end of this phase, the game was almost finished. They tested it, fixed any problems, and made it better based on what they learned from the tests.



**The main menu design, they focused on making it pretty- and simple for the user.*



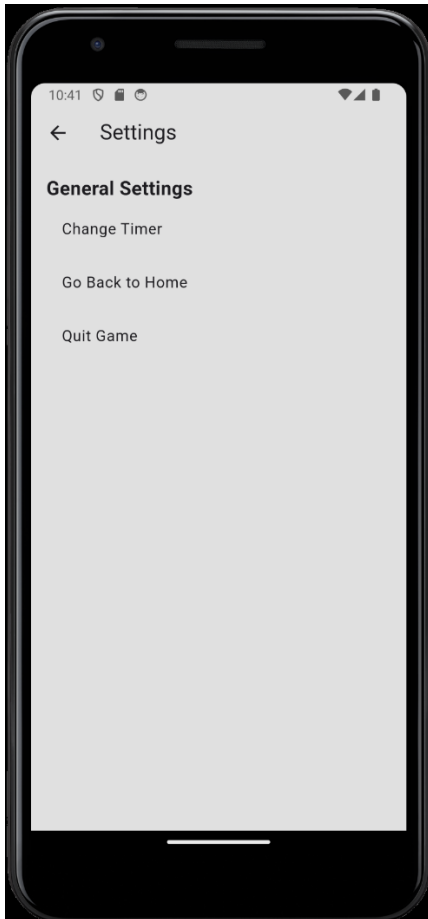
**Example of the level design.*



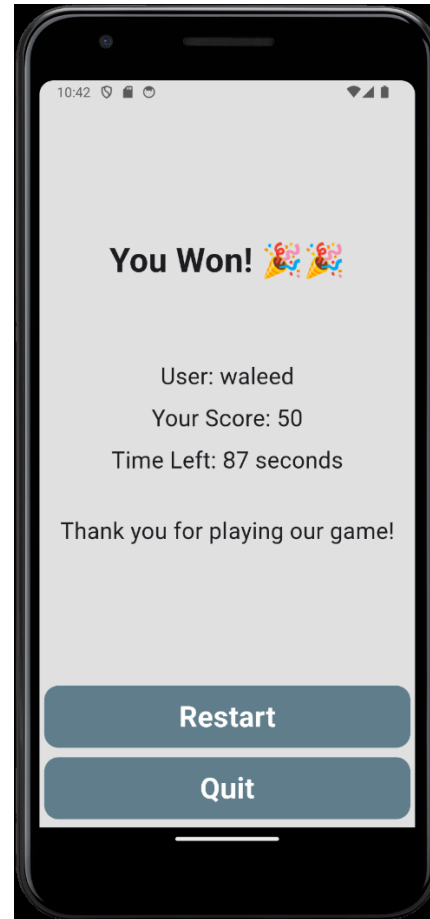
**The final version of the login/registration page.*

Phase 3: Finishing Touches - Designing

In the final phase, they focused on making the game perfect. They looked at every part of the game, made changes based on feedback and testing, and had many discussions to make sure the game was the best it could be. They worked on making the game run smoothly and made final adjustments. The last step was to get the game ready to share with the world. They also took a photo of the game to show off what they had achieved.



**Settings menu was added in the finishing-touches it was as simple as possible for younger users.*



**Celebration page for completing all the levels and finishing the game.*

And that's the story of how the **Fill The Blank** game app came to be. It was a journey full of teamwork, creativity, and hard work, and in the end, they created something they were proud to share.

Challenges That Faced Us & Lessons learned:

Challenges Faced:

- **Time Management:** Balancing the demands of the project with our academic responsibilities proved to be a significant challenge. With multiple exams and schoolwork, we had to carefully manage our time to meet the deadlines for each project phase.
- **Scope Creep and Requirement Changes:** Despite meticulous planning, we encountered instances of scope creep and requirement changes. Adapting to these changes while staying within project constraints was challenging. Open communication with stakeholders and embracing agile principles were crucial in accommodating changes effectively.
- **Documentation Overload:** We underestimated the time and effort required to maintain comprehensive documentation throughout the project. Striking a balance between documenting progress and focusing on development tasks proved challenging. Agile documentation practices became essential to avoid unnecessary verbosity.
- **Backlog Management:** While using an Excel document for backlog management seemed straightforward initially, maintaining a clear and prioritized list of tasks became increasingly challenging as the project progressed. Regular backlog grooming sessions were necessary to adapt to evolving requirements effectively.
- **Complexity of System Models:** Understanding and effectively utilizing various system models, including context and state diagrams, use case diagrams, sequence diagrams, class diagrams, and activity diagrams, presented a steep learning curve. We had to invest additional time and effort to grasp the intricacies of these models and apply them effectively in our project.
- **Flutter Doctor Issue:** Early on, we encountered challenges with Flutter Doctor, which presented errors that were initially perplexing. We had to dedicate time to understand Flutter's dependencies and ecosystem better to troubleshoot effectively.
- **Complexity of MVC Pattern:** Implementing the Model-View-Controller (MVC) pattern in our educational game required careful coordination among team members. Balancing separation of concerns while ensuring smooth communication between model, view, and controller components was challenging.

Lessons Learned:

- **Time Management:** Through the challenges of balancing project deadlines with academic responsibilities, we learned the importance of effective time management. Implementing strategies such as prioritizing tasks, setting realistic goals, and scheduling dedicated project time helped us navigate through busy schedules while ensuring progress on the software engineering project. This experience highlighted the significance of proactive time management skills in successfully completing complex projects within limited timeframes.
- **Continuous Learning:** Embrace a growth mindset and continuously seek out resources to enhance skills and overcome technical challenges.
- **Collaboration and Communication:** Prioritize effective collaboration and communication within the team to address issues promptly and share valuable insights.
- **Adaptability:** Stay flexible and adaptable in the face of unforeseen challenges and changes, embracing agile principles to accommodate evolving requirements.
- **Simplicity in Design:** Strive for simplicity in design and implementation, breaking down tasks into manageable chunks and focusing on delivering a minimal viable product (MVP).
- **Problem-Solving:** Confronting various technical challenges and unforeseen obstacles throughout the project, we learned the value of adaptability in problem-solving. Being open to exploring alternative solutions, seeking assistance from peers and mentors, and approaching problems with a flexible mindset enabled us to overcome hurdles effectively. This experience underscored the importance of adaptability in navigating the complexities of software engineering projects and finding innovative solutions to emerging challenges.

Reflecting on these challenges and lessons learned, we recognize the importance of perseverance, teamwork, and a commitment to continuous improvement in successfully navigating software engineering projects.

Conclusion:

The project's goal was to develop *"Fill the Blank"*, an interactive educational game designed to help children aged 4 to 8 enhance their English vocabulary and word recognition skills. By filling in missing words in sentences, children engage in a fun and interactive learning experience that aims to improve their understanding of English words, spelling, and sentence formation. The game is developed with the intention to make learning enjoyable, helping children to learn and remember new words while also improving their ability to form sentences.

The development of the game was structured into three phases: planning, programming, and designing. In the initial phase, the team focused on outlining the game's objectives, defining its target audience, and establishing a clear vision for how the game would function and look. This included determining the features and capabilities that would make the game both educational and entertaining for its young audience.

Throughout the project, the team encountered and overcame various challenges, such as managing their time effectively, adapting to changes in the project's scope, and resolving technical issues. These experiences taught them valuable lessons about the importance of communication, the need for continuous learning and adaptability, and the benefits of keeping designs simple and user-friendly.

The final product is a game that provides an engaging and educational experience for children. It features interactive gameplay where children can select or type in missing words to complete sentences, progressing through levels of increasing difficulty. The game offers visual feedback for both correct and incorrect answers, encouraging children to learn from their mistakes and celebrate their successes. A progress tracking system is included to monitor the children's learning journey, making it possible for parents and educators to observe and support their development.

In conclusion, the *"Fill the Blank"* game exemplifies the effectiveness of thoughtful planning, collaborative teamwork, and a commitment to creating a learning tool that is both fun and educational. It stands as a valuable resource for children beginning to explore the English language, offering a foundation for building vocabulary and sentence structure skills in an engaging and supportive environment.