

Project Document

Let's Get Quizzical

Group 6

Agata Runowska-McMillan

Charlotte Shorey

Ruth Honeywood

Kamila Pruski

Kirsty Henderson

Sarah Todd

Introduction

Aims and objectives of the project

Project Vision: Our vision for the Pub Quiz App aims to extend beyond a temporary solution to current challenges and we aspire to create a lasting platform that allows individuals to continue to use together. As the world navigates through evolving circumstances, our project aims to deliver joy, entertainment, and a sense of community to users anywhere, reinforcing the power of technology to unite individuals even in times of physical isolation.

Motivation: The Pub Quiz App is intended as a response to the limitations imposed by physical distancing requirements, enabling friends, family, or colleagues to come together virtually and enjoy a pub quiz experience from the safety and comfort of their own homes. Whether separated by distance or constrained by circumstances, the app seeks to recreate the lively atmosphere of a pub quiz, where participants can test their knowledge, compete, and share moments of fun and enjoyment.

Goals: As a team our goals include aiming to acquire practical experience in the end-to-end software development workflow, including ideation, design, implementation, testing, and deployment. We hope to develop an application that provides users with a smooth and enjoyable quiz experience by creating a user-friendly interface with intuitive navigation. An important factor for our group is to explore agile development principles in order to enhance collaboration and adaptability. Ultimately our group plans to draw and expand on the knowledge base acquired through the CFG Degree programme to build an app we can be proud of.

Roadmap of the report

Introduction:

Provide a brief overview of the Pub Quiz App project.

Clearly state the aims, objectives, and vision of the project.

Highlight the motivation behind developing the app in response to global challenges.

Background:

Discuss the inspiration behind the Pub Quiz App, particularly in response to the COVID-19 pandemic.

Explain the shift from a multi-player to a single-player focus for enhanced user experience.

Emphasise the goal of creating a sense of community and entertainment through virtual engagement.



Specifications & Design:

Outline the key features and requirements of the Pub Quiz App.

Detail the diverse question categories, user-generated avatars, scoring system, and leaderboard functionalities.

Describe the user-friendly interface and the overall design and architecture of the application.

Implementation & Execution:

Provide insights into the development approach and team member roles.

Discuss the collaborative approach, the role of the Scrum Master, and the communication tools used.

Outline the tools, libraries, and technologies employed, including development tools, databases, and project management tools.

Briefly touch on the implementation process, achievements, challenges faced, and decisions made during development.

Testing & Evaluation:

Present the testing strategy, including unit testing, regression testing, and user testing.

Highlight the user testing phase, involving friends and family for diverse perspectives.

Discuss the iterative and agile development elements, such as code reviews and refactoring.

Emphasise the importance of aligning the final product with user expectations for a satisfying experience.

Conclusion:


Summarise the overall journey of developing the Pub Quiz App.

Reiterate the goals achieved, challenges overcome, and decisions made during the project.

Conclude with the significance of the app in providing joy, entertainment, and community in a virtual space.

Background

In response to the unprecedented challenges posed by global events, particularly the COVID-19 pandemic, our team has undertaken a mission to develop an innovative solution that addresses the evolving dynamics of social engagement. Recognising the need for novel methods to build connection and entertainment in times when people were unable to be in the same physical space, our team set out to develop a Python-based Pub Quiz-style application – an engaging and interactive platform that allows players to take part remotely.



Originally, the app was envisioned to enable multiple players to simultaneously engage in the quiz, fostering a sense of community even in a virtual setting. However, during the development process, a strategic decision was made to prioritise single-player functionality. This decision was driven by the goal to maximise the app's capabilities and ensure a seamless user experience. Single-player functionality allows individuals to participate at their own pace and convenience. This inclusivity is especially important for users who may prefer solo experiences or have scheduling constraints. By concentrating on single-player functionality, we can optimise the user interface and experience, ensuring a smooth and enjoyable process, plus allow for a more refined and optimised app.

The player is able to select a quiz category from a predefined list and will then be presented with 10 multiple choice questions. Each question is displayed with 4 potential answers and the player must select their chosen answer within a certain amount of time before proceeding to the next question.

Specifications and Design

App requirements

Key Features

Diverse Question Categories:

- The app will feature different question categories, ranging from general knowledge and pop culture to specific themes, ensuring a diverse and inclusive experience for all participants.

User-Generated Avatars and Nicknames:

- Participants personalise their gaming identity by entering a nickname and selecting an avatar, both of which will be displayed on the score page and added to the leaderboard.

Scoring System and Leaderboard:

- A dynamic scoring system will track participants' responses and calculate scores in real time.
- A leaderboard will showcase individual and team standings, enhancing the competitive spirit and encouraging friendly rivalry.
- The inclusion of a timer per question further enriches the gaming experience, adding an element of excitement and urgency to each participant's response.

User-Friendly Interface:

- The app will boast an intuitive and visually appealing interface, making it accessible to users of varying technical proficiency.



Design and architecture

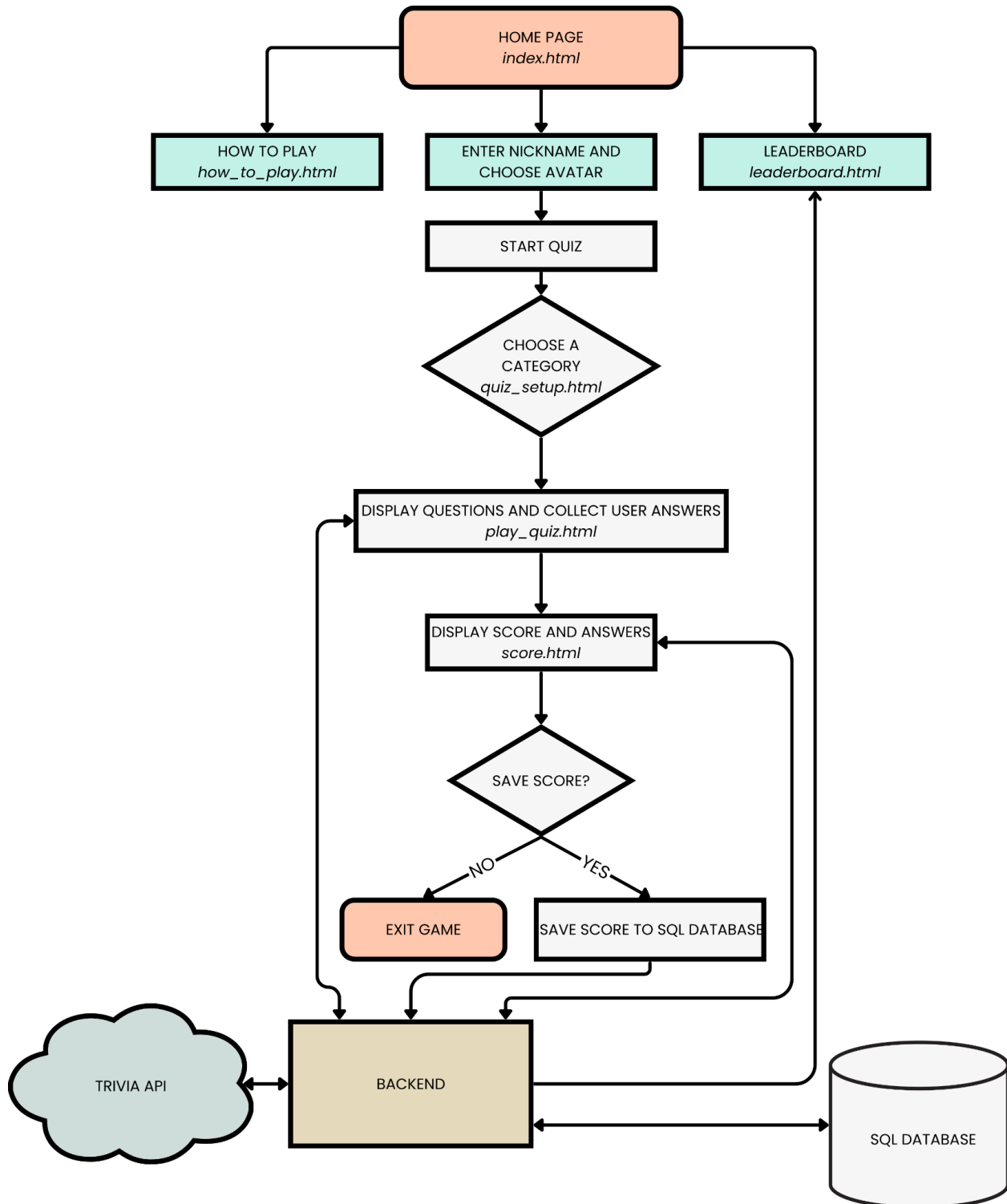
Front End

- **Start/Homepage:** Users begin at the homepage where they can enter their nickname and choose an avatar, personalising their trivia experience from the outset.
- **Category Selection:** Users select their preferred trivia category, providing a tailored quiz journey that suits their interests.
- **Question Interaction:** The quiz questions are displayed one at a time, and the user's answers are collected dynamically. Each question is timed to add an element of competition.
- **Scores and Answers:** At the end of the quiz, users receive their score and the answers to the questions, with an option to save their score to the SQL database, which then reflects on the leaderboard.
- **Leaderboard Display:** The leaderboard showcases top scores and ranks, encouraging users to improve and participate actively.

Backend

- **Backend Logic:** Python-based backend logic, facilitated by Flask, manages the game flow, user sessions, and data management.
- **Trivia API:** The backend interacts with a Trivia API to fetch questions, ensuring a varied question set for the users.

Architecture Diagram



Database

We have created a database, `user_scores`, to store user data, track scores, and manage the leaderboard. The database plays a critical role in ensuring that the application is interactive and competitive.

id	avatar	nickname	score
Unique identifier for each quiz attempt, auto-incremented to ensure uniqueness.	Stores the path to the user's chosen avatar image, allowing for a personalised gaming experience.	Records the user's chosen nickname, limited to 3-25 characters to maintain uniformity and readability.	Captures the user's score from the quiz, enforcing a score range between 0 and 10.
1	avatar-1.png	nickname	8

File Structure

- **Configurations (/config):** Stores configuration files and initialization scripts, ensuring a modular and scalable setup.
- **Database Scripts (/database):** Contains SQL scripts for database setup, schema creation, and initial data seeding.
- **Models (/models):** Defines the data structures and logic for database connections, the leaderboard system, and quiz game mechanics.
- **Routes (/routes):** Flask routes are defined here, handling web page navigation, form submissions, and game progression.
- **Static and Assets (/static):** Hosts static files including CSS for styling, custom fonts, images for avatars and UI elements, and JavaScript for frontend dynamics.
- **Templates (/templates):** HTML templates define the structure and content of each webpage
- **Tests (/tests):** Dedicated to automated tests, ensuring code quality and reliability.
- **Main Application (app.py):** The entry point of the application, tying together components and launching the Flask server.
- **Documentation and Requirements:** README.md and requirements.txt provide documentation for setup instructions and dependencies, guiding new developers and users.

Implementation and Execution

Development approach and team member roles

Our collaborative approach was the cornerstone of our product development journey. Every team member actively engaged in the decision-making process, from ideation to final specifications. Our collective brainstorming sessions laid the foundation for our project, fostering creativity and inclusivity.

Recognizing the importance of streamlined communication and organisation, we appointed Agata as the Scrum Master. In this role, she facilitated effective communication, organised team meetups, managed our JIRA board, ensuring that tasks were tracked and progress was transparent, and prepared documents for collaboration.

Here's the collaborative approach we adopted with communication and project management tools:

- JIRA: to track, manage, and allocate tasks.
- Git and GitHub: it was used by our team, leveraging GitHub Issues for tracking bugs and planning enhancements, while employing pull requests to introduce new functionalities and improvements. We adhered to a workflow where each team member created a new branch for their respective features, using a default Dev branch, which required a minimum of one reviewer for approval before merging. Additionally, branch protection rules were applied to both the Dev and Main branches, with project submission directed towards the Main branch.
- Confluence: meeting notes, storing all documents and links in one place
- Google Docs and Sheets: Activity log, working on Project Document together, logging user testing feedback
- Google Meet: weekly and impromptu meetups, and pair programming
- WhatsApp and Slack: daily communication, sharing information, making decisions, arranging meetings.

To capitalise on individual strengths and preferences, we adopted a task allocation strategy that considered each team member's unique skills. This approach not only promoted a sense of ownership but also enhanced overall productivity.

The tables below show the task allocation of the team members. Where you see repetition it was one of the tasks we did in smaller teams (e.g. unit testing, design of classes):

Agata	Charlotte	Kamila
<ul style="list-style-type: none">- Structure of the app- Leaderboard functionality- Database connection and creation- Python classes- Quiz mechanics- Unit testing- Code refactoring- Flask sessions	<ul style="list-style-type: none">- External API research, presentation to the team- API connection and questions formatting- Python classes- Exception handling- Code refactoring- Unit testing	<ul style="list-style-type: none">- User Interface- Category selection page from QuizQuestions class- Timer functionality- App design- JavaScript- CSS- HTML templates- Flask Sessions

Kirsty	Ruth	Sarah
<ul style="list-style-type: none"> - First draft of Quiz Game functions - Python classes - Unit Testing - Exception Handling - Worked on a multiplayer function until we made a decision to add it for future development. 	<ul style="list-style-type: none"> - Avatar functionality including saving to database - Start Quiz Page - Flask sessions - User Experience improvements - Project flow chart - Wireframes and app design - Presentation slides 	<ul style="list-style-type: none"> - Flask routes - Flask links and navigation - Category selection page - Quiz questions on page - How to Play Page - Navbar - Unit Testing - User Experience Improvements

Apart from above tasks all team members were responsible for:

- Reviewing and merging pull requests
- Writing this Project Document: we split sections evenly
- Choosing project name: we put ideas forward and had a vote
- Decisions about app functionality, structure, user experience
- User testing, including asking our friends and family to be our user testers and logging the feedback and correcting errors found.
- Debugging

Embracing an agile mindset, our decision-making process was not rigid but rather adaptive. Weekly meetings served as forums for major decisions, while day-to-day choices found their space in our communication channels - be it through WhatsApp, Slack, or impromptu meetups. This flexibility allowed us to promptly address blockers and pivot when necessary.

Tools and libraries

Development Tools:

- Git: Distributed Version Control System (VCS).
- PyCharm: Integrated Development Environment (IDE) for Python.
- Visual Studio Code: Integrated Development Environment (IDE) for JavaScript.

Database Development Tool:

- MySQL Workbench: For database design and development.

Development Libraries and Modules:

- Flask (v3.0.0): Web Framework for Python.
- Requests (v2.31.0): HTTP Request Library for Python.
- MySQL Connector for Python (v8.1.0): MySQL Driver for Python.

- Flask-WTF (v1.2.1): Flask Extension integrating Flask with WTForms.
- WTForms (v3.1.1): Forms Validation and Rendering Library, enhancing input handling.
- Json: JSON Encoding/Decoding Library.
- os: Module (Part of Python Standard Library).
- random: Module (Part of Python Standard Library).
- secrets: Module (Part of Python Standard Library).
- collections: Module (Part of Python Standard Library).
- unittest: Module (Part of Python Standard Library).
- Bootstrap 5.3.2: Front-end framework.

Project Management and Collaboration:

- Jira: Agile Project Management Tool.
- GitHub: Version Control Hosting Service.

We used the Trivia API, which can be found here:


<https://the-trivia-api.com/>

Implementation process (achievements, challenges, decision to change something)

We set up our files by organising them by type so they were easy to find as follows:-

- All the html files were in a templates folder.
- All route files are in a routes folder.
- All models files went in a models folder.
- All the API information went in a separate API folder.
- Our config folder contained all our JSON config information. This folder contained a config_example file to show the user what information to put in, and a config.json file containing our own information that was put into the gitignore file so that it was not shared to GitHub.
- All test files were grouped into a tests folder.
- All CSS, js and images were grouped into a static folder

Our database was set up to save the user's score and nickname after the user had finished playing the quiz and displayed it on the leaderboard. We used auto increment for the 'id' which



was also the primary key. The database also had a nickname and score columns. The nickname column was set up using a varchar type and gave it a length of between 3 and 25 characters. The score was given an INT type and was set to values between 0 and 10 which is the amount of quiz questions we have per game. The database also creates a view of the 10 top scores by arranging the inputs by descending order of score.

Whilst coding our game we made a new branch for each section we were working with a title of our name and purpose. We appointed a scrum master who oversaw the project and the pull requests as they came through. The scrum master was also in charge of ensuring everyone's sections were on track and kept the project running smoothly this kept merge conflicts to a minimum.

Our main challenge was connecting the backend of our quiz game to the front end. We had to work out how to connect the route files in flask which was new to us. We also found retrieving the user answers from the quiz quite a challenge. We solved this in the end using sessions to save the data.

We also added a welcome page in the beginning forcing the user to add their nickname and avatar which would carry through the game to be used in the scoreboard at the end.

Despite all our challenges we created a fun quiz for people to play that displays their score on a scoreboard if it is good enough.


We had initially thought we would try to create a multiplayer version of the quiz alongside a single-player version. We decided to change this and concentrate on the single-player version. This decision was driven by the goal to maximise the app's capabilities and ensure a seamless user experience. The multiplayer functionality would be something to work on for future development.

Agile development (did your team use any agile elements like iterative approach, refactoring, code reviews, etc)

Our group utilised an Agile working approach to complete the project and produce the final pub quiz app.

The development process followed an iterative approach, with the project divided into small, manageable increments or iterations. Each iteration then focused on specific features or functionalities, allowing for continuous refinement and enhancement based on feedback and evolving requirements.

As the project progressed, regular code reviews and feedback loops were established, leading to targeted refactoring efforts. This enabled better code maintainability, helped to reduce potential bugs and contributed to a more scalable and robust application.



Code reviews via pull requests in GitHub were an integral part of the development workflow, promoting collaboration and knowledge sharing within the team which was crucial to identify potential issues and maintain a high level of code integrity.

Regular communication channels were set up to encourage collaboration; a dedicated channel on slack, JIRA boards with cards denoting tasks that were in corresponding to do, in progress and completed sections. Weekly whole team meetings were scheduled with the purpose of mutual decision making about the project and also to assign tasks to each individual. As the project moved forward from the idea stage, tasks were also assigned according to priority. These happened on Thursday afternoons prior to the regular evening class, plus ad hoc meetings were arranged among particular team members collaborating on the same feature or for further investigation into problems. A dedicated section was in place within the JIRA board where tasks that group members had been assigned and were having difficulties with could be highlighted and assistance sought. A group was also created on WhatsApp for team members to discuss iteration details immediately as they were being worked on or if the app temporarily stopped working.

This approach ensured that the team remained aligned with project goals and responded effectively to challenges.

User feedback was initiated through the development process by inviting player testing and then gathering their feedback which allowed the team to address user preferences and requirements where possible. The aim here was to meet user expectations, ultimately leading to a more satisfying and user-friendly experience.


Testing and Evaluation

Testing Strategy:

To begin our testing strategy, we defined the scope and objectives of our quiz game. This was to include 10 multiple choice questions for a specific category as chosen by the user, as well as other user features like a username and a scoreboard.

Planning our game functionality reminded us of the targets of our code. This allowed us to administer thorough risk assessments by hypothesising issues which could be faced with each aspect of our plan. Some examples of issues which we considered were API and SQL connection errors. By doing this, we were better prepared for potential issues in our code as we ensured that valid and invalid cases were tested for, thus maximising our efficacy and robustness of unit testing.

As a team we were consistently creating tests for our code as our programme progressed. Additionally, we utilised regression testing prior to submitting a pull request (unless assistance



was needed to refactor the code), to ensure that existing functionalities were not inadvertently broken. After completing our functional testing to validate that all features worked as intended, we moved onto user testing.

User testing:

Once our quiz game was checked by all members of our team, we progressed to user tests which were completed by friends and family. They were instructed to do the following:

1. Enter your name, choose an avatar, start a quiz for your chosen category.
2. Save your score.
3. Come back to the home page, navigate the links there.
4. Provide feedback of any issues encountered or improvements suggested

This enabled us to identify areas of improvement as we were given an alternative perspective to running the programme, which could have fallen outside of what we considered the “correct” way. An example of this was to adapt our game to not allow the user to progress to the next question without selecting an answer.

After conducting our user tests, we returned to functional testing to ensure our changes retained the integrity of the programme.

Our holistic testing strategy ensured that not only the functionality was tested, but also that the final product aligns seamlessly with user expectations, to foster higher satisfaction and engagement levels.

System limitations


Limited Question Pool:

If the quiz relies on an external API for questions, there may be a constraint on the total number of questions available. Users who play the quiz frequently might encounter repetition of questions, leading to a less engaging experience.

API Reliability:

The app's functionality is dependent on the reliability and availability of the external trivia API. If the API experiences downtime or becomes obsolete, it could disrupt the quiz experience for users.

Single-Player Focus:



Since the decision was made to prioritise single-player functionality, users seeking a multiplayer quiz experience may find the app less suitable.

Limited Quiz Categories:

The variety of quiz categories may be limited by the available categories in the API. Users with specific interests may feel constrained if their preferred categories are not available.

Avatar and Nickname Restrictions:

The app's personalization features, such as avatars and nicknames, may have limitations in terms of options and customizations. Users might desire more diverse choices for avatars or longer/nickname options.

Dependency on Internet Connection:

The app's functionality relies on a stable internet connection, and users in areas with poor connectivity may experience disruptions.

Device Compatibility:

The app may have limitations regarding compatibility with certain devices or browsers, affecting the accessibility for some users.

Scalability Issues:

If the user base grows significantly, the current structure may face challenges in handling increased traffic and data storage requirements.

Future development possibilities

If provided with additional time, our next steps in app development would involve:

- Introducing a multiplayer feature that not only allows quiz enthusiasts to enjoy the challenge but also replicates the authentic atmosphere of a real pub quiz, fostering a sense of community engagement, which was our overarching goal.
- Integrating difficulty levels into the Pub Quiz App. The API we've utilised provides questions with different difficulty levels (easy, medium, hard). Implementing this feature would not only enhance the customization aspect for players but also add an additional layer of challenge for those seeking a more competitive quiz experience.

Conclusion

In concluding the Pub Quiz App project, our team achieved a dual objective: delivering an engaging quiz platform and gaining invaluable real-world software development experience. The decision to prioritise a seamless single-player experience underscored our commitment to user satisfaction. Throughout this project, we encountered challenges, made strategic decisions, and experienced the dynamic nature of collaborative teamwork.

By embracing agile methodologies and effective collaboration, we navigated challenges and produced a feature-rich app with interesting and satisfying features. The project illuminated the importance of adaptability and communication in a dynamic development environment.

While acknowledging limitations like potential question repetition, these insights serve as stepping stones for future improvements. The Pub Quiz App is not just a successful project but helps to showcase our collective dedication to learning about software engineering and user-centric solutions. This journey has equipped us with practical skills and a foundation for continued growth in the area of software development.