Quark – A gamified time tracking application and website

Assignment Report

CI536 – Integrated Group Project

27th May 2022

Lab Tutor: Karl Cox

Group Members:

Kim Lam (19823013)

Karan Vani (19800011)

Fin Watling (20838503)

Ben Edmondson (20812940)

Danil Bronnikov (20833663)

Jazer Barclay (20837308)

I confirm that I have a Learning Support Plan for which includes adjustment deadlines as recommended by the Disability and Dyslexia Team, and agreed by the School. I understand the deadline for my assessment has been adjusted (as per the required School protocol) and that this should be taken into consideration when my assessment is marked/ graded.

Table of Contents

1. Introduction

This report aims to cover how our group functioned as an agile team to analyse, design, develop and release a prototype time tracking application, designed to aid in knowledge absorption and retention with analytics to improve future study and work sessions.

Our goal was to find ways we could improve the efficiency of study for students and other knowledge workers. We also wanted to address the issue of extended work sessions by introducing healthy habits of regular breaks to stretch, re-hydrate or simply rest.

If we could improve work efficiency, we could get more done within the same amount of time. By reducing the negative impacts of working longer and including regular rest, we can also improve general health in the future.

This goal of improved health and greater work efficiency will serve as the business opportunity our team will pursue.

To solve this problem we will need to find a method of increasing performance during work sessions whilst also introducing regular breaks. Our research on deep work and methods of study uncovered the Pomodoro technique; a work-rest timer that aids in learning and knowledge retention over extended periods of time.

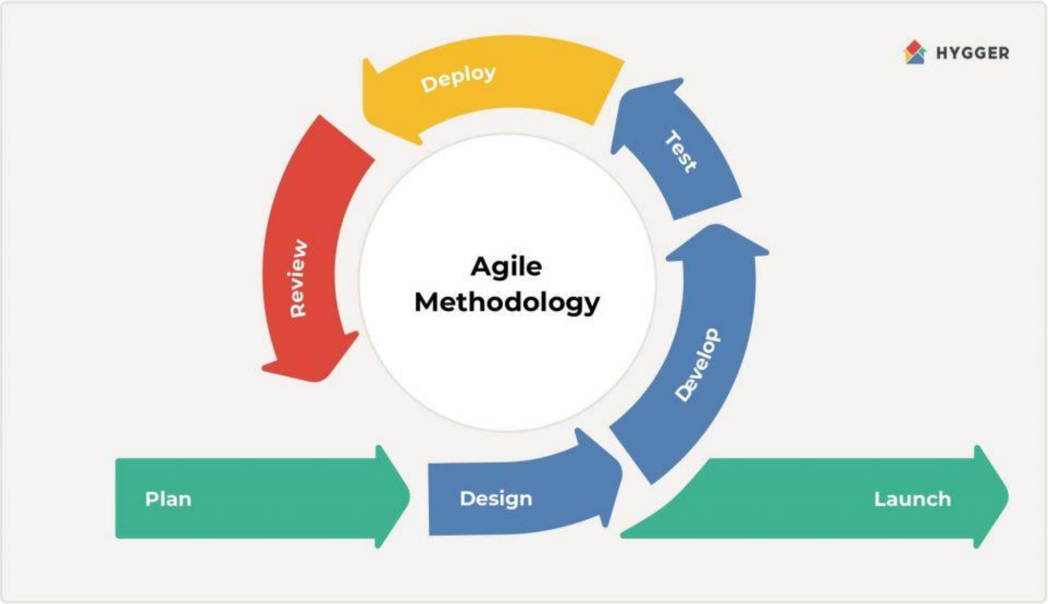
Our plan is to implement this technique in an application the user can run to keep track of their work sessions. For helpful insight, a website where recorded sessions can be displayed could potentially improve future study sessions or encourage greater use of the app.

2. Methodology

To execute on developing a functional and useful application as a team, we first needed a plan on how we would work together in order to achieve the desired outcome.

2.1. Work Process – Agile

In our first week's lab, we came together and set out to use the agile work process. We felt it best, as a team, to have a project leader that would help guide the work process rather than use the scrum approach. Having worked as a developer in a small company where the agile work process was the standard and with the most experience with full-stack development, Jazer took the role of project leader.

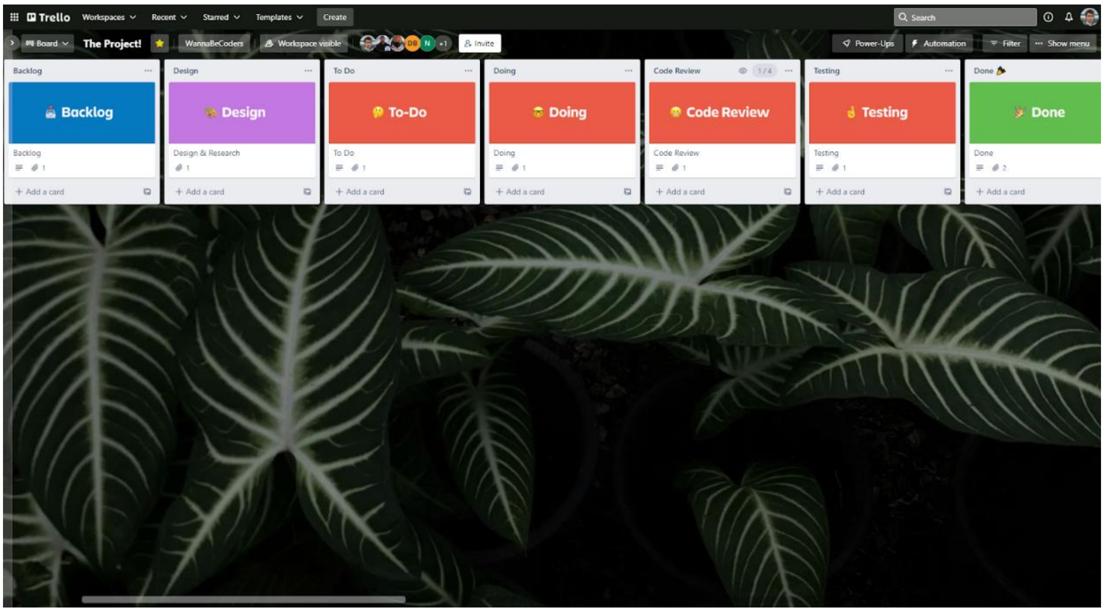


Software lifecycle from an agile approach

Our decision for agile over scrum was also influenced by the requirements for regular meetings. Many of us work non-fixed hours and cannot always attend a meeting every day, even remotely.

As an alternative, we chose to hold regular meetings every Tuesday and Friday. Tuesdays we were all on campus for lectures and Fridays we had our lab check-in with our tutor (Karl) where we had 2 hours to go over anything we wanted to cover or discuss.

To keep track of our progress with the project we used the Kanban-style tool Trello. We used it to track requirements and links to resources which we could reference as we planned and developed the project. Over time as items were completed, we moved them left-to-right from the to-do and doing sections to the complete section at the end. After one of our labs, we used the recommendation from our tutor to use individual weeks to show what was completed each week rather than a long single "completed" list.



Kanban-style board in Trello

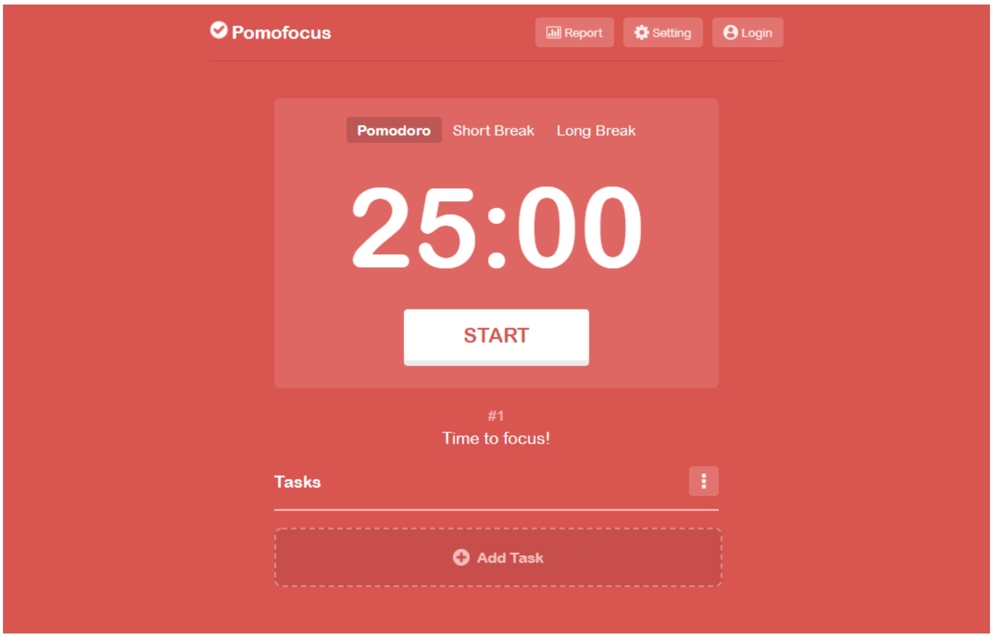
Combined with this Kanban board we implemented a buddy system to ensure no task was left or forgotten. This provided a learning opportunity for someone less experienced to learn from the other in implementing a solution. This proved very effective since we did encounter multiple members of our team at different stages of development needing to take time away for personal reasons.

Work was assigned by the project lead to each pair and extra tasks were added to the backlog in Trello for groups that finished their assigned task before the next meeting.

2.2. Research – Existing applications and websites

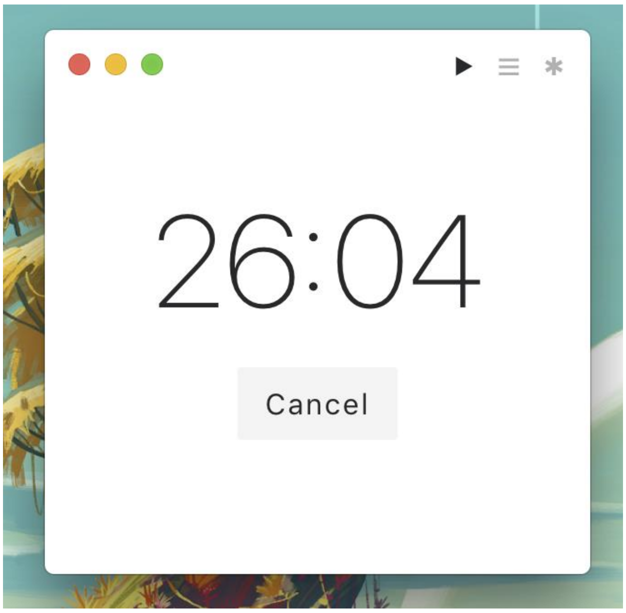
Before making design plans or writing any code, we first searched for existing applications that implement a similar solution to the one we are looking to solve.

We found many web-based applications that run within the browser. These are useful for day-to-day use however do not keep track of your working sessions over any period of time. These seem to rely on you managing this which we found adds unnecessary friction to our workflow.



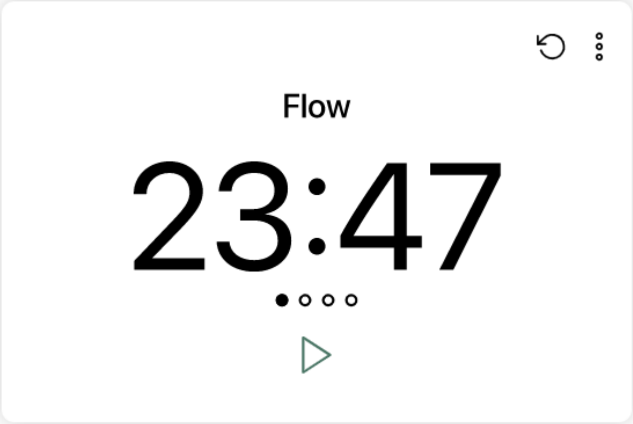
Browser-based pomodoro timer – Pomofocus.io

We also found desktop applications that do a similar thing where an individual session may be tracked while the application is running, however, it is lost upon restart.



Another Mac Based Pomodoro application -Thomas

The minimalist design of the desktop application “Flow” we found the best in class however the application is restricted to MacOS and unavailable on other platforms. We feel that any system should be capable of running this style of application and we endeavour to enable this on Windows, macOS and Linux distributions.



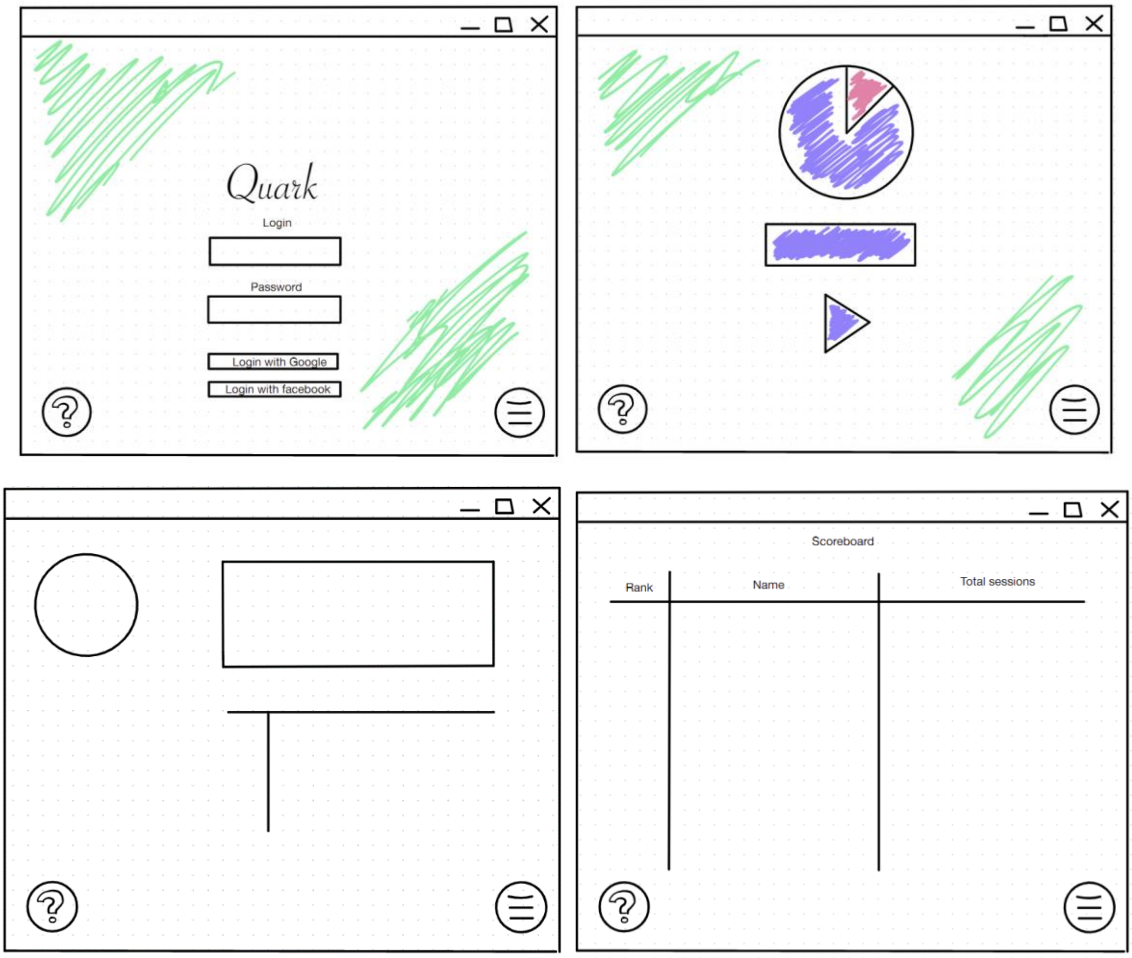
MacOS only pomodoro timer – Flow.io

2.3. Design – From sketch to wireframe

Using the information gathered from our research, we set to design the application including the desktop application and the website interface for viewing user stats.

We started with a blank canvas and documented the key features we wished to see in our application. This we condensed to 3 key points: intuitive, minimalist and simple to use.

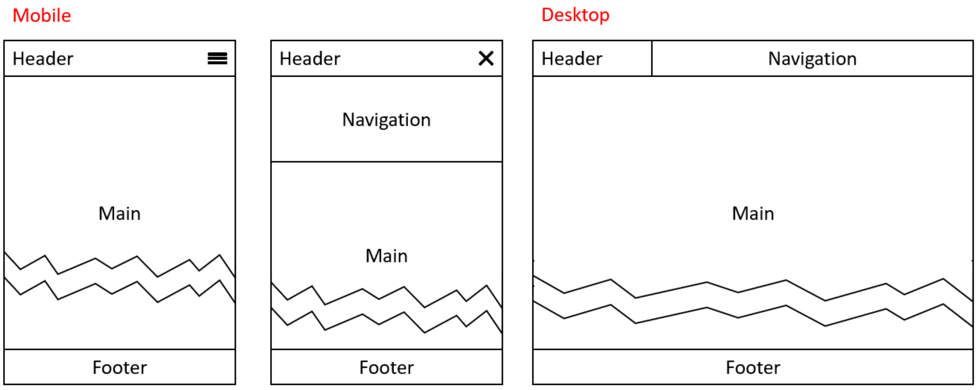
From this we created our first crude sketches of the core interfaces such as the desktop timer where sessions are viewed, started and recorded. We also created a very bare design for the user interface for the website. This consisted of the login page, user stats and leader board.



First sketch designs made on the iPad of login screen, timer, profile and leader board from left to right, top to bottom

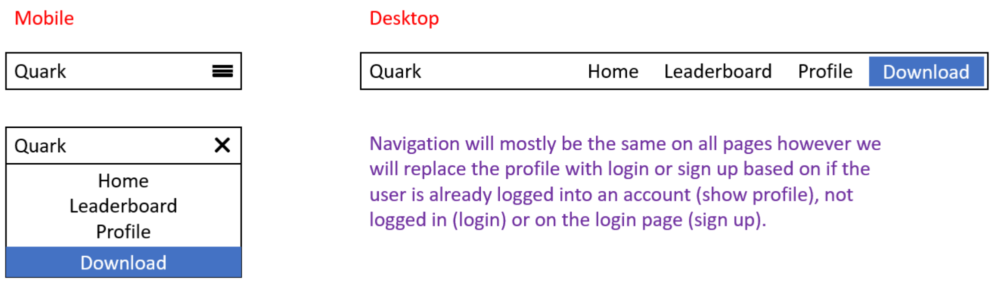
As we completed more of our project plan mentioned in the next section, we started expanding our design to include more concrete layouts matching the required data and functionality.

Page Layout



Wireframe design of website page layout

Navigation Design



Navigation will mostly be the same on all pages however we will replace the profile with login or sign up based on if the user is already logged into an account (show profile), not logged in (login) or on the login page (sign up).

Wireframe of navigation for mobile and desktop view of website

Part of our assignment is thinking more in-depth about the user archetypes who will be using the application and catering to their use cases. For our application, we wanted it to be very minimal for the general user but detailed enough on the website to glean useful insights at all analysis levels.

We also took developers into consideration and exposed the API to the internet. This provides other developers with the ability to use the fundamentals of the project within either their own applications extending functionality further.

Regardless of framework or stack, we needed a consensus on what features each section would require. From our research and wireframes, we created a list of actions that the end-user would need to perform to use the application and website. These came together on the server-side needing to store and manage the data. This influenced the creation of the Software Requirements Specification document. This SRS document outlines how the project would function. This can be found in Appendix C under the document “Quark Software Requirements Specification”. An table showing a subset of the requirements can be found below.

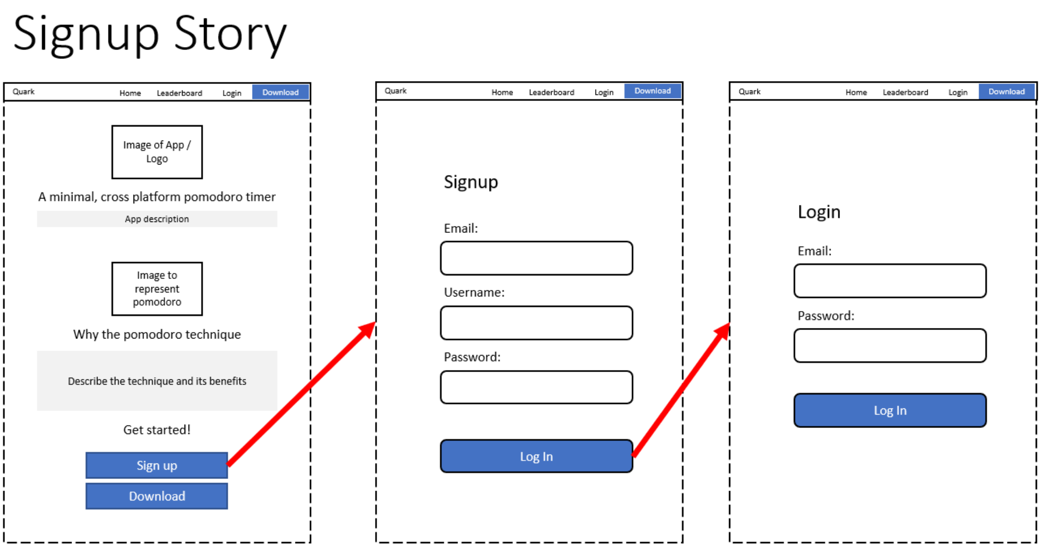
ID	Requirement	Type	Priority
FR1	(Web) Users can successfully sign up on the website with their email, username and password.	Functional	Must have
FR2	(Web) Users can log into their accounts after the server authenticates the data.	Functional	Must have
FR3	(Web) Leaderboard will display the top 100 users after the leaderboard tab is pressed.	Functional	Must have

FR4	(Web) Users can select to view the leaderboard by day, week and month.	Functional	Should have
FR5	(Web) Users can view a graph showing how much units have been gained by day, week and month.	Functional	Should have
FR6	(App) After clicking start, the timer should begin counting down from 25 minutes.	Functional	Must have
FR7	(App) After clicking pause, the timer should temporarily stop at the same time.	Functional	Should have
FR8	(App) After clicking stop, the timer restarts and starts from the beginning.	Functional	Should have
FR9	(App) After 25 minutes have passed, the timer should begin the 5 minute break.	Functional	Must have
FR10	(App) After the 5 minute break timer ends, begin the 25 minute work break again.	Functional	Must have
FR11	(App) After a 5 minute break timer, a pomodoro unit is sent to the database to be added to your account.	Functional	Must have
FR12	(Web) User is able to download the jar file after clicking the download button.	Functional	Must have

Table of requirements from the SRS documentation

We did have to backtrack and reiterate our designs to match the new requirements documented in the SRS. This was a rather simple process with the new functional requirements clear and interface requirements easier to design for.

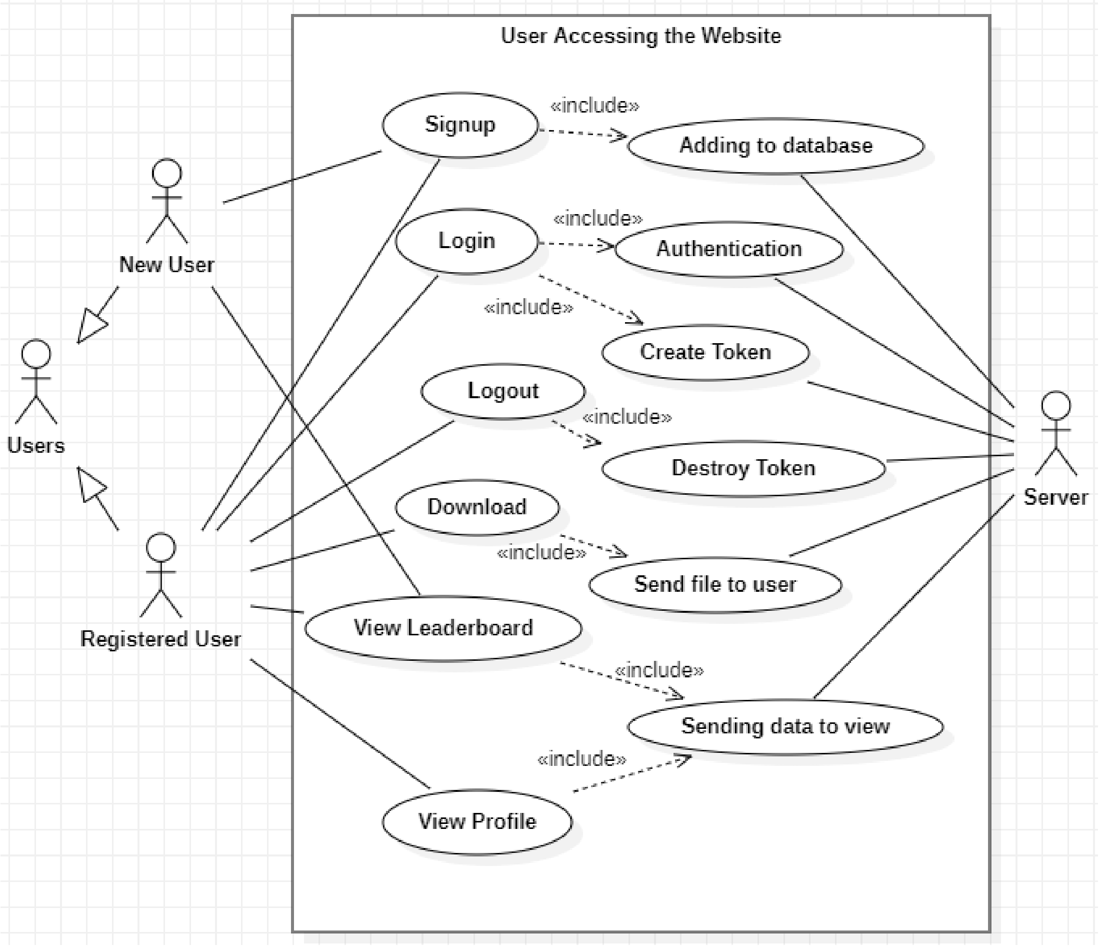
Going a step deeper, we looked at how each of the pages of the website and the screens within the application will link together. For a clearer view, we designed user stories that storyboard the user’s journey through each of the key steps they will undertake using the application from signup and login to logging sessions and viewing statistics.



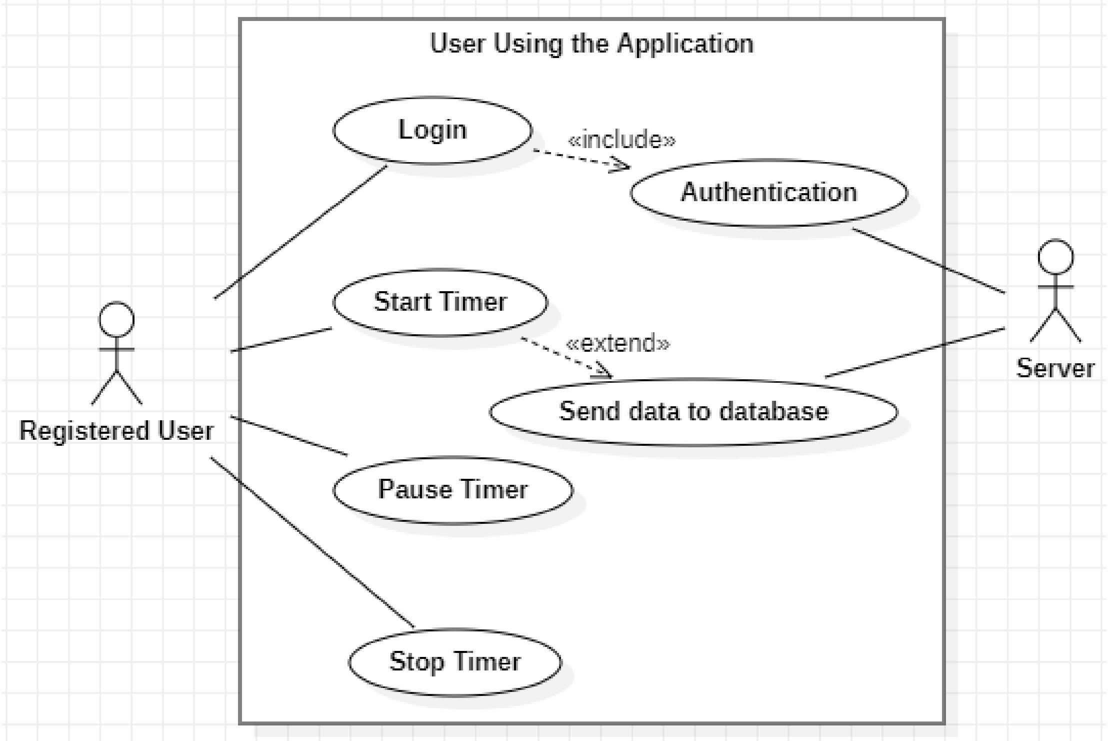
User story for signup process

Part of our considerations was for accessibility. This included making the website mobile responsive using a mobile-first approach which has now become industry standard. Also for the desktop application, making forms navigational via the tab and space keys makes interaction via other input devices for handicapped users far easier.

Some UML use case diagrams were drawn to identify the main functionality users will be doing when accessing both the website and the desktop application. Both figure 1a and 1b will show what users can do on both the website and on the application as well as what happens in the backend of what the server does when the user interacts with the applications.

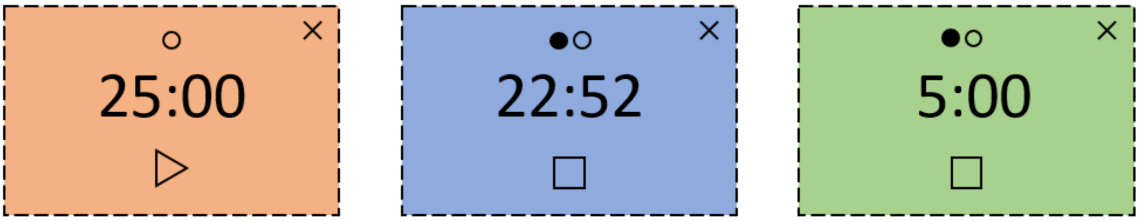


UML use case diagram of user accessing the website



UML use case diagram of user accessing the application

From the SRS, revised designs and user stories, we reached a final layout and map of the screens we desired to create. This will serve as the basis we will develop for and hopefully expand upon once we reach the minimum viable product.



High fidelity representations of the timer's multiple states

2.4. Project Planning

With designs in hand, we next looked at how we would implement this with a top-down approach.

Through our research and design, we had a good idea as a group of what our application would need in terms of functionality. To gain the flexibility of using preferred technology stacks such as using vanilla JavaScript or using a framework such as react, we found a layout that is modular. This meant any individual section could be built independently and would integrate seamlessly with other sections. As such the database, API, website and application could all be built in any language and would come together via HTTP requests and API calls.

Using the SRS document, tasks were added to the Trello board which were assigned to pairs with future tasks added to the backlog. In hindsight, all requirements could have been added from the beginning allowing for a greater overview of progression through the project.

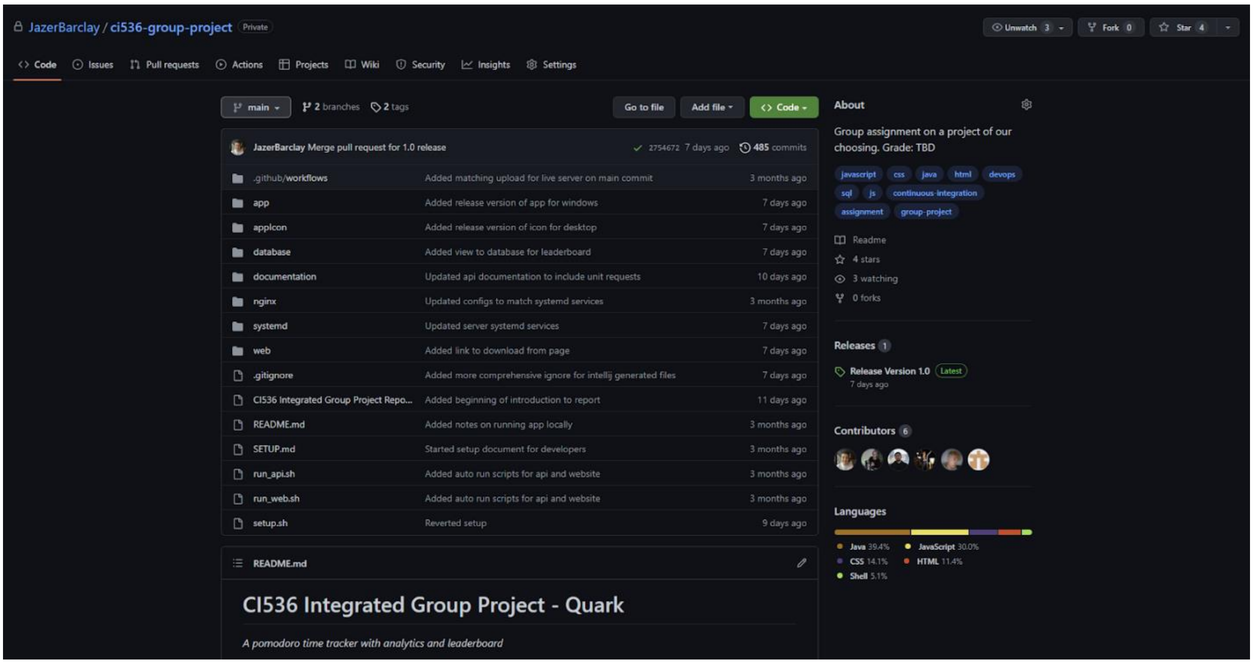
We wanted to take a test-driven development approach to the process. This way we could make more incremental changes to the application and ensure code functioned as required rather than debugging near the end of development.

To keep the code clean for all developers to work with, we employed a standard code layout convention. This followed the google style guide used primarily in our Java code and extended partially to our code in the website and API.

Throughout the development process, comments should be added to blocks of code where their function is not implicit. This reduces the mess that can accumulate due to unnecessary commenting, however, maintains clarity for more complex segments of code or structures.

2.5. Version Control – Git and GitHub

For tracking the changes made to our codebase, we use Git and GitHub. We chose this compared to other version control systems such as SVN due to its decentralised design and offline functionality. We all have learned to use the Git tool for the past 2 years and were proficient in its use either via the command line or from the GitHub desktop application.



JazerBarclay Merge pull request for 1.0 release ✓ 2754672 7 days ago 485 commits

.github/workflows	Added matching upload for live server on main commit	3 months ago
app	Added release version of app for windows	7 days ago
appIcon	Added release version of icon for desktop	7 days ago
database	Added view to database for leaderboard	7 days ago
documentation	Updated api documentation to include unit requests	10 days ago
nginx	Updated configs to match systemd services	3 months ago
systemd	Updated server systemd services	7 days ago
web	Added link to download from page	7 days ago
.gitignore	Added more comprehensive ignore for intellij generated files	7 days ago
CI536 Integrated Group Project Repo...	Added beginning of introduction to report	11 days ago
README.md	Added notes on running app locally	3 months ago
SETUP.md	Started setup document for developers	3 months ago
run_api.sh	Added auto run scripts for api and website	3 months ago
run_web.sh	Added auto run scripts for api and website	3 months ago
setup.sh	Reverted setup	9 days ago

README.md

CI536 Integrated Group Project - Quark

A pomodoro time tracker with analytics and leaderboard

About

Group assignment on a project of our choosing. Grade: TBD

javascript css java html devops sql js continuous-integration assignment group-project

Readme
4 stars
3 watching
0 forks

Releases 1

Release Version 1.0 Latest
7 days ago

Contributors 6

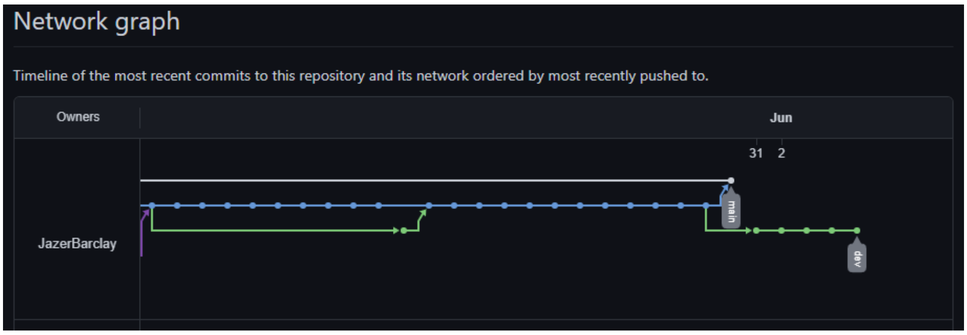
Languages

● Java 39.4% ● JavaScript 30.0%
● CSS 14.1% ● HTML 11.4%
● Shell 5.1%

GitHub project overview showing contributors, branches, files and folders

All members of the team had access to the repository and could make changes to the codebase. Each change could be committed to their local copy of the repository and pushed to GitHub for other developers to work with.

We had a mainline stable branch (“main”) and a development branch (“dev”). We would all work together making alterations to the dev branch until we felt that we had reached a point where the product was stable enough to be merged into “main”. This way we can create checkpoints in our code that we could revert to or simply view as we made future changes.


Network graph
Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.
Owners
Jun
31
2
JazerBarclay
main
dev

Mainline and development branches merging and separating

Our project leader was in charge of merging changes between the team's commits and would help solve any merge conflicts as they arose. This included changes made to the development branch and merges between the development and mainline branches.

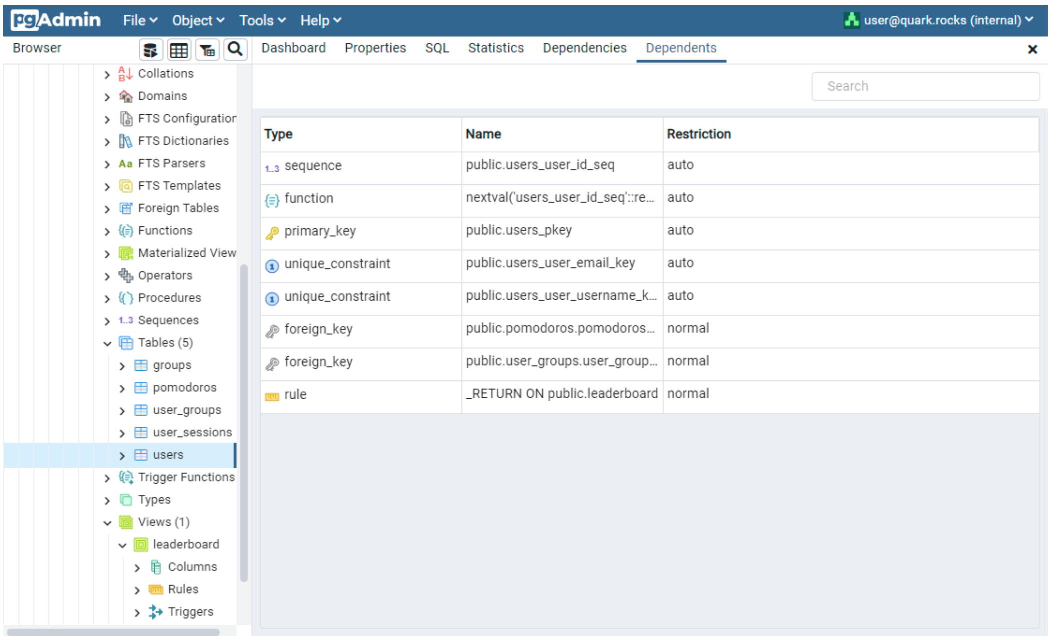
We used webhooks in GitHub and Discord to track and log all code changes to the "git-commits" text section of our discord group. This allowed for real-time monitoring of the development process and was especially helpful when in the working meetings voice channel as we could talk about our changes as they occur.

This approach worked really well and we were all able to work on different parts of the system at the same time whilst tracking our code changes with each push.

2.6. Tech Stack – PostgreSQL, Node.JS, Java, HTML, CSS, JS

Our choice of technology stack was twofold: suitability to the requirements and ease of development for our team. This works to our advantage since our project is quite wide in scope and a complete prototype.

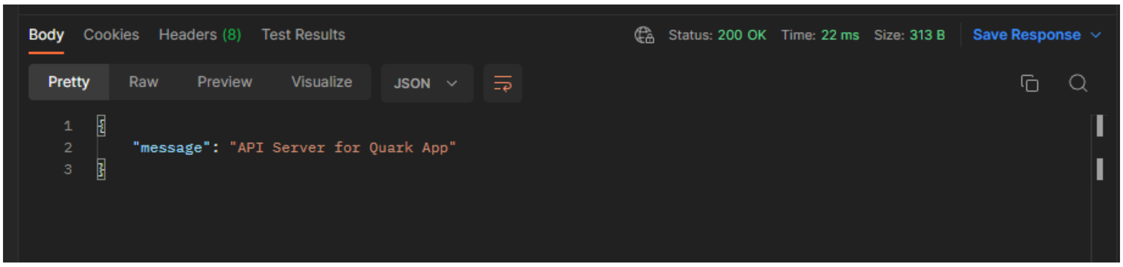
For the database we used PostgreSQL. We used this due to its built-in function triggers and easy deployment on docker, which we utilised. It uses the same syntax as regular SQL and has a UI counterpart to view the data stored.



Type	Name	Restriction
sequence	public.users_user_id_seq	auto
function	nextval('users_user_id_seq'::re...	auto
primary_key	public.users_pkey	auto
unique_constraint	public.users_user_email_key	auto
unique_constraint	public.users_user_username_k...	auto
foreign_key	public.pomodoros.pomodoros...	normal
foreign_key	public.user_groups.user_group...	normal
rule	_RETURN ON public.leaderboard	normal

View of the PostgreSQL database via the associated web view PGAdmin4

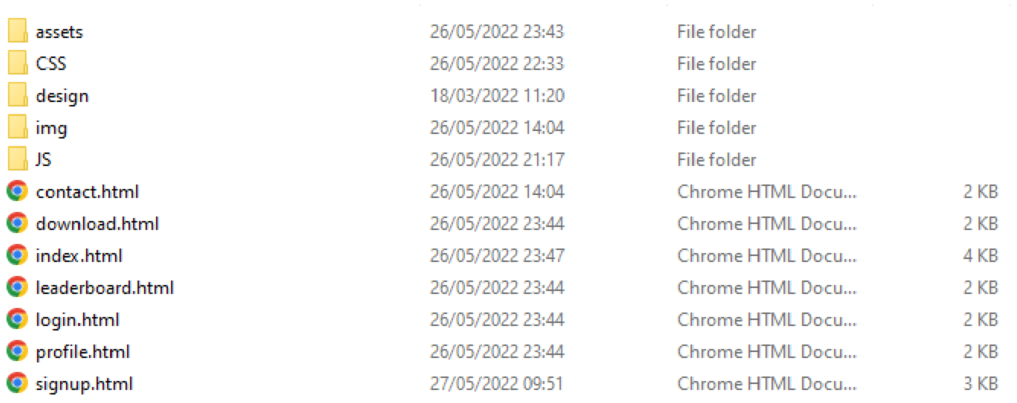
For the API we used Node.js with express to arbitrate the connection to the database. This combination provided us with an API that was both secure and extendable which could also scale to keep up with a significant amount of requests if the application became popular.



```
{
    "message": "API Server for Quark App"
}
```

View of response object from API testing

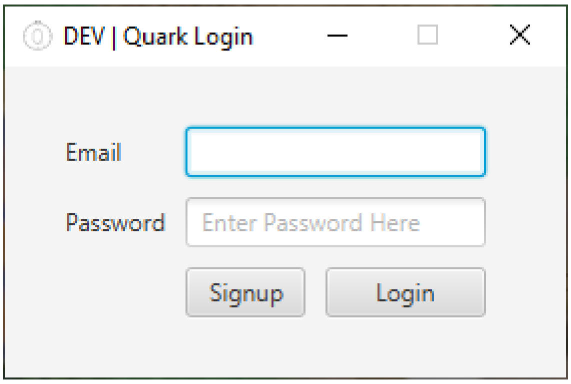
For the web application, we used vanilla HTML, CSS and JavaScript. We also used the Fetch API to make back-end API requests. This setup is easy to develop and is highly flexible as many libraries support this stack. For example, we used an open-source graph library called Chart.js to populate and plot the graph to show user units.



assets	26/05/2022 23:43	File folder	
CSS	26/05/2022 22:33	File folder	
design	18/03/2022 11:20	File folder	
img	26/05/2022 14:04	File folder	
JS	26/05/2022 21:17	File folder	
contact.html	26/05/2022 14:04	Chrome HTML Docu...	2 KB
download.html	26/05/2022 23:44	Chrome HTML Docu...	2 KB
index.html	26/05/2022 23:47	Chrome HTML Docu...	4 KB
leaderboard.html	26/05/2022 23:44	Chrome HTML Docu...	2 KB
login.html	26/05/2022 23:44	Chrome HTML Docu...	2 KB
profile.html	26/05/2022 23:44	Chrome HTML Docu...	2 KB
signup.html	27/05/2022 09:51	Chrome HTML Docu...	3 KB

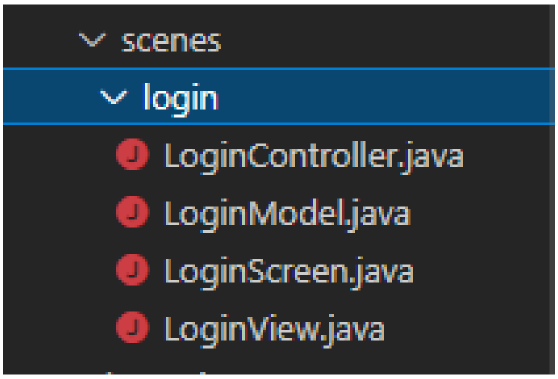
View of the files showing only html documents and accompanying JS and CSS documents in their folders

For the desktop application, we used Java with JavaFX. We used the Model, View, Controller design pattern (MVC) to streamline the development process of the application. This involved separating the state of the application which is stored in the model, the interface design stored in the view and the application logic stored in the controller. Troubleshooting and debugging if anything went wrong was far easier using this method streamlining the development process.



A view of the login view class without interaction connected (just a visual window)

Our UX and UI considerations required a modern system and development approach. We created a template MVC model using the JavaFX library making interface design and implementation a standard method.



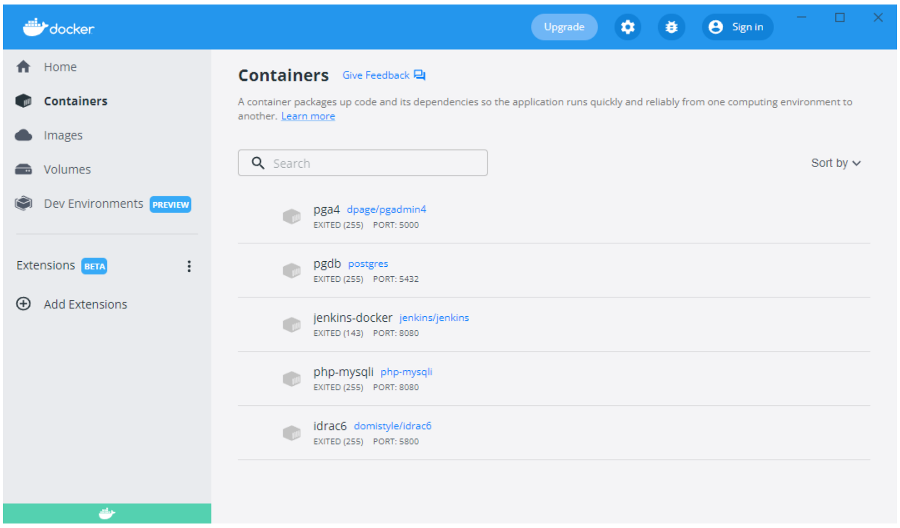
```
scenes
  login
    LoginController.java
    LoginModel.java
    LoginScreen.java
    LoginView.java
```

A view of the login model, view and controller java classes and their screen that they converge within to create the interface

2.7. Testing – JUnit, Postman, Deployment and Manual Testing

For testing, we split the project into their respective parts and did localised testing eventually coming together to connect them forming the final product.

We first started on our local system developing on our own hardware. This would become a tedious problem-solving task had we not used package management tools such as npm and maven. This was also kept at bay using Docker which creates standard containers for each segment of our project resolving the issue of “it works on my machine” issues.



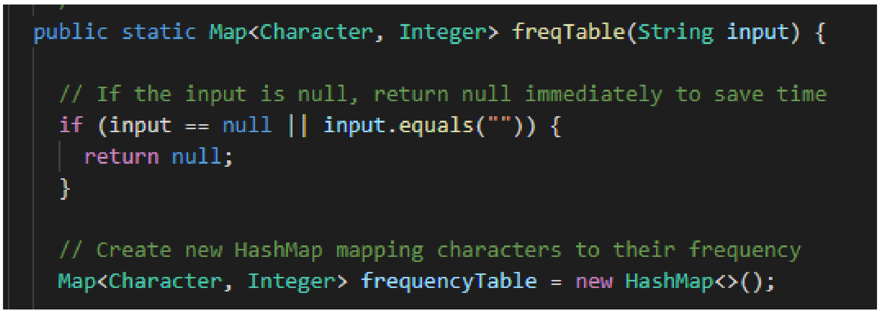
Containers

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

pga4	dpage/pgadmin4	EXITED (255) PORT: 5000
pgdb	postgres	EXITED (255) PORT: 5432
jenkins-docker	jenkins/jenkins	EXITED (143) PORT: 8080
php-mysqli	php-mysqli	EXITED (255) PORT: 8080
idrac6	domistyle/idrac6	EXITED (255) PORT: 5800

User interface for the Docker container system

A standard practice in all parts of our project is to create functions and methods that check fail conditions first to fail fast. This approach is used to reduce computation that is not required if an error state will be reached resulting in wasted work. This is also a great method of reducing nested statements which can make code unreadable.



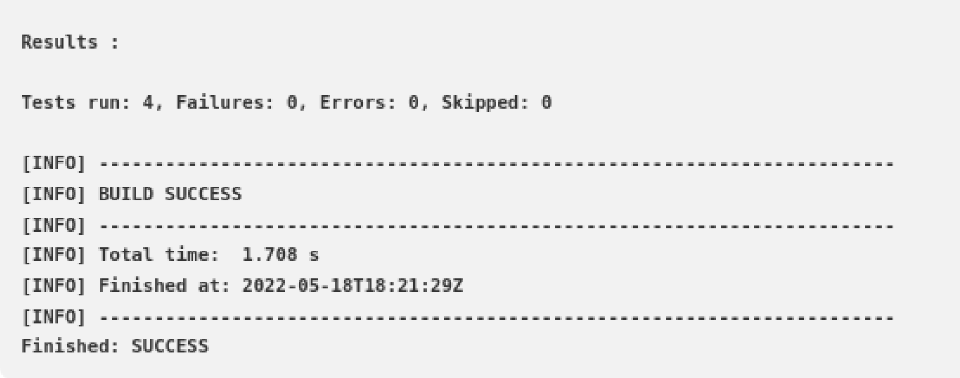
```java
public static Map<Character, Integer> freqTable(String input) {

  // If the input is null, return null immediately to save time
  if (input == null || input.equals("")) {
    return null;
  }

  // Create new HashMap mapping characters to their frequency
  Map<Character, Integer> frequencyTable = new HashMap<>();
```

An example showing the fail early approach where we do a fail state check before creating new variables and populating them

To test the desktop application, we used IntelliJ and Eclipse integrated development environments combined with the JUnit library. This allowed for unit tests of specific blocks or whole classes to be tested for their functionality. Since we are also using Maven for package management, we can create automation scripts on developer systems which run the tests before launching the complete application. This way we can ensure the system is running optimally before reaching the UI testing which we did manually.



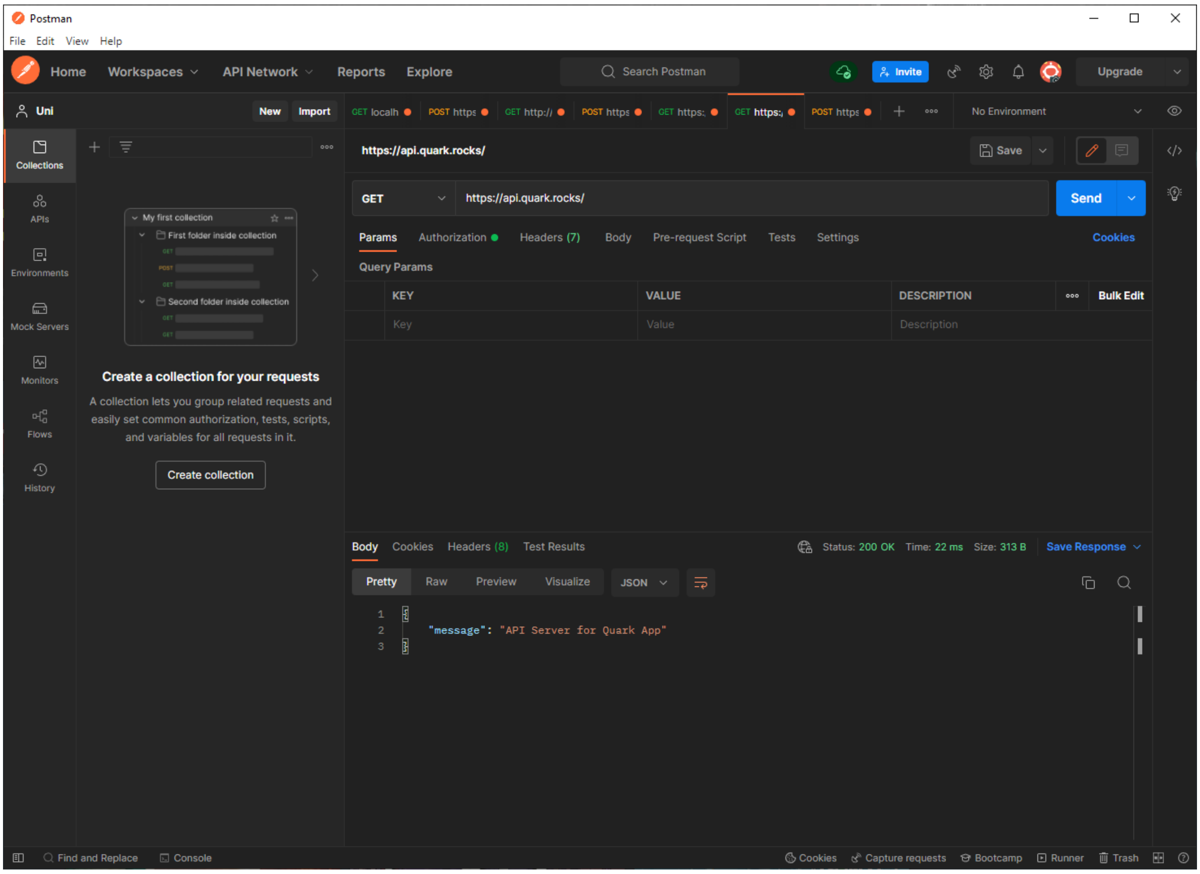
```
Results :

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.708 s
[INFO] Finished at: 2022-05-18T18:21:29Z
[INFO] ------------------------------------------------------------------------
Finished: SUCCESS
```

Test results from running JUnit test against the application

To test the API, we used an application called Postman. This tool allows a developer to craft custom requests to an endpoint and log the results in a clearly viewable window. We used this to create a set of custom requests conforming to the API documentation and run it against the local API deployment.



https://api.quark.rocks/

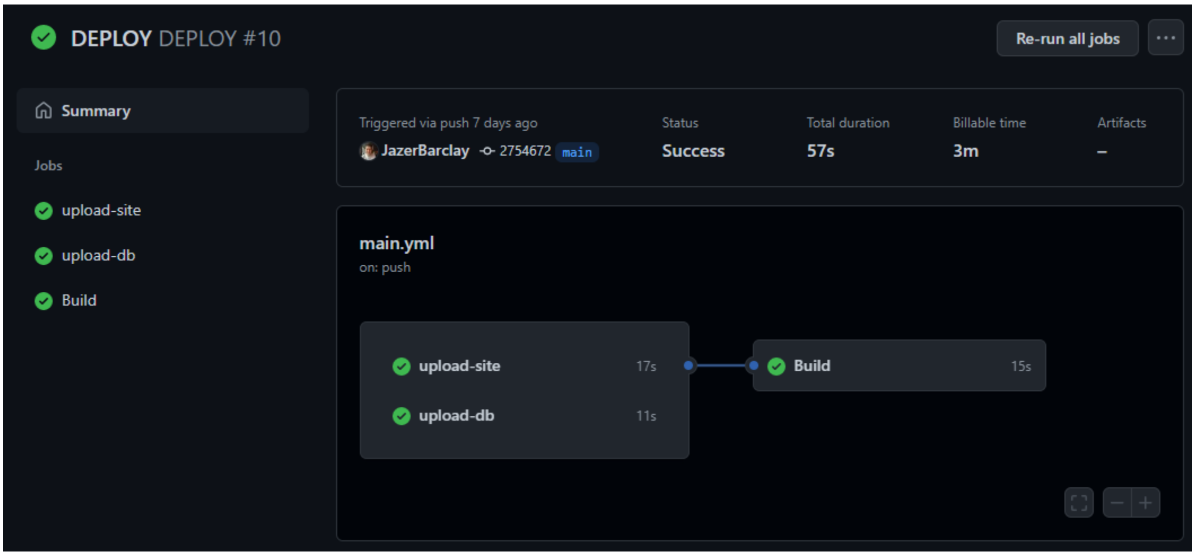
GET https://api.quark.rocks/

Status: 200 OK Time: 22 ms Size: 313 B

```
{
    "message": "API Server for Quark App"
}
```

Postman interface showing how we interacted with the API from a testing point of view

On the topic of deployment, we created 2 automations on GitHub called “GitHub Actions”. These are scripts we can use to perform a set of actions which run internally on GitHub’s virtual private servers. We created a deploy script for the mainline and development branches to deploy them for remote testing and live releases.



DEPLOY DEPLOY #10

Triggered via push 7 days ago: JazerBarclay -o- 2754672 main

Status: Success

Total duration: 57s

Billable time: 3m

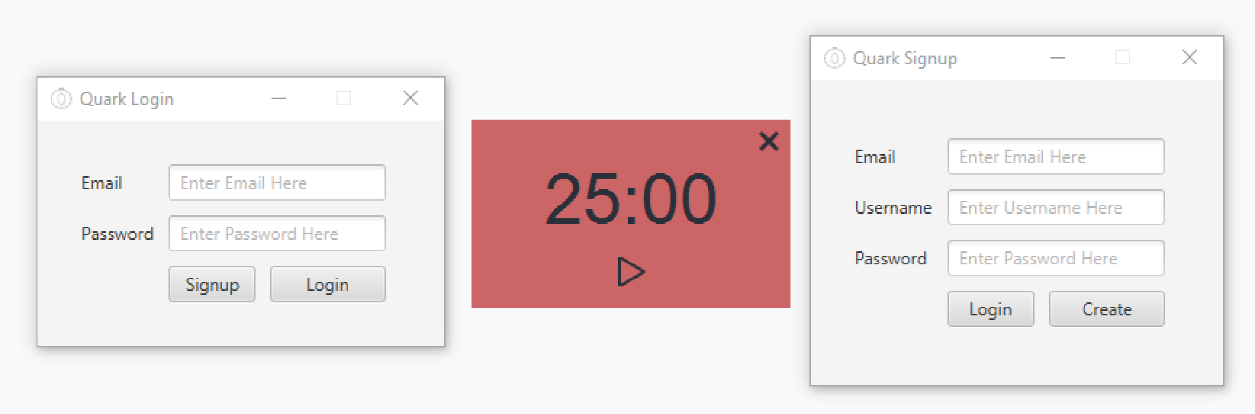
Artifacts: –

main.yml
on: push

upload-site 17s
upload-db 11s
Build 15s

GitHub Action’s build pipeline we created to build and deploy the website

For all the user interface testing, we primarily applied manual testing while working through a checklist of key features we wanted. This included referencing our SRS and the user stories design documents.



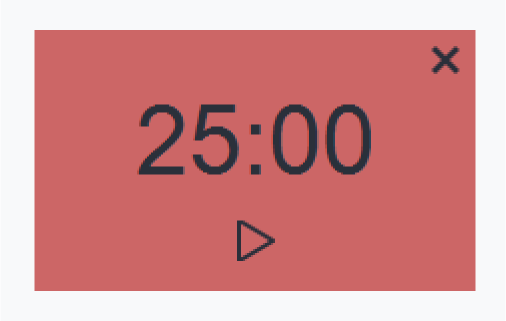
Manually testing the user interfaces from setup, login, and timer

3. Product Description

Quark is a simple, minimalistic yet powerful time management application that utilises the Pomodoro Technique to track productivity. Users can sign-up via the website or desktop application after which they can log in to their account. By visiting the website, a user can view their past performance to help guide future sessions using the app.

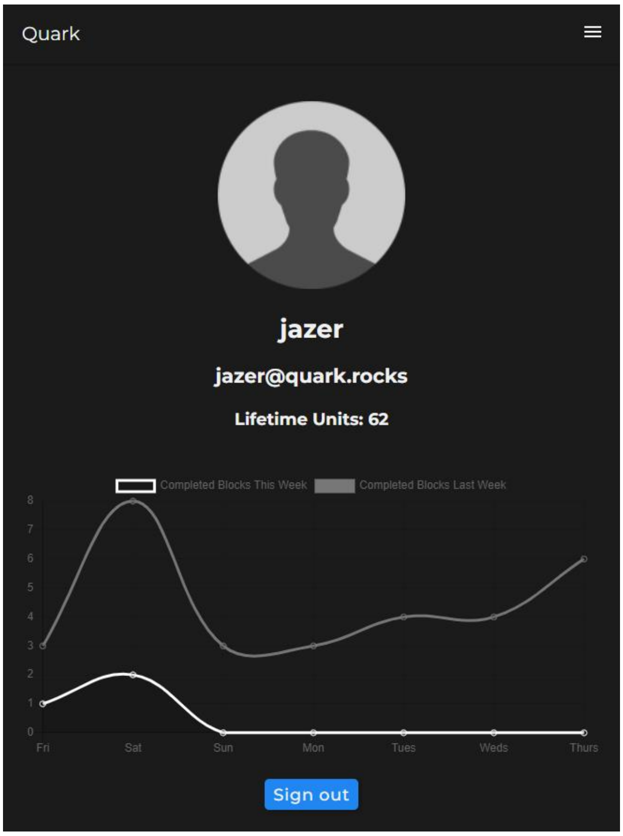
Using the SRS documentation, and API document combined with the design sketches we created the desktop application, website and server-side interaction via the API and database.

When logged in, the desktop application is a minimalistic, Pomodoro based timer, which logs and sends completed units (25 working minutes and 5 break minutes), to the database under the logged-in account. Once one complete unit is logged, the application then waits for the user to start the timer again manually. This is to confirm the user is ready to commit another 25 minutes to their work and is also used as protection against cheating to log extra units in without working.



Final design of the timer interface from first launch

The website acts like a hub where you can view your productivity analytics, and how many units you’ve completed over the past 2 weeks. This helps track and improve your working hours. This information is all compiled for ease of access in the user profile. By using Quark, you are automatically ranked via the leader board against all other users of the system. This provides an incentive to try and “beat the high score'', gamifying the system to a certain degree. Each user’s completed units are displayed on the leader board, and currently only the top 100 players are displayed.



Quark

jazer

jazer@quark.rocks

Lifetime Units: 62

Final design of user profile on iPad size portable device showing user info and session stats.

4. Legal, Ethical and Security Issues

The Data Protection Act 2018 (DPA2018) is the UK's implementation of the General Data Protection Regulation (GDPR). All data collected, both general and sensitive, must adhere to the guidelines provided by the government and follow their data protection principles (Gov UK, 2018).

To protect users' data, we have decided to make it so that the users will log in to their accounts through their email and have their user stats displayed with a username. This not only helps keep the user's email safe from viewing by other members, but also allows people to identify this member through a displayed username, meaning that users can identify friends and new members without needing more sensitive information such as an email address or full name.

No sensitive data is collected or needed for this application to fully function. This can protect users in cases where malicious players intend to identify and track users when using this app. Not only will they not be able to do so, but they also won't be able to see the email of the users, so they are unable to track them at all. In a situation where our system is compromised, hackers will not be able to obtain any valuable data from us as none is stored in our system.

When the password is created, it gets saved in our system after getting encrypted via a salt hash function. This makes it very difficult for malicious hackers to reverse engineer the password and relate it to specific accounts. It also protects from their accounts on other services being compromised if they should have the same password on our system and another.

The main benefit of using a token is that it would allow users to continuously use features of a site or application without the need to repeatedly enter their password and other account details. Adjustments have also been made to the token so that it can be destroyed at the end of its life cycle either via expiration date or manually logging out. This can prove to be more effective when accessing accounts in public areas so that it can kick you out of your account in case you forget to log out yourself.

Within our code, data has been modified and sanitised to prevent hackers from carrying out SQL injection attacks when data is being sent through the desktop application. Since all data sent to and received from the database is managed through the API, we can check the data before being stored and validate the data sent.

The design of the website has been made in a way that makes buttons big and colourful to help clearly indicate what is happening on the screen. No small prints are used on the website, so nothing is made to be difficult to be seen. Everything is written in the same size and is clearly shown on the page, so the users know exactly what to expect from our services.

5. Evaluation of Fitness for Purpose

One of our primary objectives was to create a simple and intuitive experience for the user when interacting with our application. The design of the desktop application is simple and effective, with colour coding to display information to the user at a glance.

Due to the desktop application's minimalistic design, it works well when positioned in the bottom corner of the screen. This allows the user to keep working at a steady and natural pace without being distracted by our application, enhancing concentration-time when compared to other applications of a similar nature.

The leader board and the profile page graph auto-update as soon as a block is completed. This gives an instant and measurable reward to the user for completing work and should be satisfying for the user to see their progress is tracked over the two-week period.

The implementation of the application as a whole is extremely secure, all user information is stored on a secure database accessible only through the API.

The API was built to be both as modular and secure as possible, using features such as the login token and salted and hashed passwords to keep outside sources from accessing any sensitive information.

We believe our finished product satisfies the requirements we set for ourselves when in the planning and development stage of our project. Our application is secure, widely compatible, intuitive and provides sufficient analytics of our work time as required. Although not part of the requirements initially, the leader board page adds a certain edge to the product that we hadn't envisioned in the initial development stages, showing that further expansion of the product is both possible, and viable.

6. Critical Review

6.1. What went well?

Throughout the project, the group came together as a team to work on individual tasks to benefit the group goal. Nobody complained about the workload and everyone in the group shared valuable input and constructive feedback on the direction of the project. The final product is definitely something to be proud of. Under the close guidance of our project leader, we were all able to contribute something vital to the final product. Another aspect to be proud of was the initial planning stages of our project. Our brainstorming sessions as a group gave us the chance to fine-tune the project's direction. These sessions helped shape the final implementation in a way that wouldn't be possible without them.

6.2. What was learned?

The main thing we learned during our project is that working with and coordinating a large group can be extremely difficult. During the development process, many of us had issues external to the project that none of us could foresee. This meant that at any given time we could be 1-2 group members down, sometimes for as long as 2 weeks at a time. We now know for next time that when in the planning stage of a project, although unlikely to happen, we should have protocols in place if members of the group cannot work because of unforeseen circumstances.

6.3. What could be improved?

Our project would have benefited greatly from a checkpoint-style workflow. Often when working on the project we were waiting around for other parts of the project to be completed before we could start on the tasks we each set out. For example, before creating the login page, we needed the login API to be created. This caused a slight disconnect in the progression of the group and wasted some time. A checkpoint-style workflow would largely prevent this from happening as it means that we develop the project in a linear fashion. We could create a node map of tasks and work through them as a group one by one.

6.4. What will be carried through to the next project?

The database style and API we created are key parts of any multi-platform project. They were built in a way that we can easily refactor them to be used in another project due to their robustness. We will also be utilising git and GitHub in the same way as we did in the project as using it to track changes throughout this project has proved invaluable, especially if mistakes are made and we need to revert to a previous working version of the repository.

7. References

Gov uk. (2018) Data protection. Available at: https://www.gov.uk/data-protection (Accessed: 02 June 2022)

PomoFocus. (N.D) PomoFocus Application. Available at: https://pomofocus.io/ (Accessed: 02 June 2022)

Yugen GmbH. (2022) Flow Application. Available at: https://flowapp.info/ (Accessed: 02 June 2022)

Seekrtech. (2022) Forest Application. Available at: https://www.forestapp.cc/ (Accessed: 02 June 2022)

8. Appendix

8.1. Appendix 1 – Weekly Minutes Overview

Week 1:

This week we mostly covered what the plans are for this semester and what we will be doing each week and the overall goal of the module. In the lab, we had to plan to make a group and begin our research and planning as well.

Week 2:

We grouped up together and began planning what we will be doing for the week. We mainly set up a development environment for all members of the team to make it easier to build our project. We set up GitHub for all members as well as downloaded visual studio code to begin coding the project. We also started to use Trello to help track what tasks to be done and are doing for each week.

As part of our mid-week meeting, we started our first iterations of the design for the desktop application, website, and API. This will help us get started next week with development supporting our outline.

Week 3:

We started practising and testing the tools like GitHub and began creating the desktop application following an MVC (Model, View, Controller) model. We also chose an application to create which was to implement a Pomodoro timer app following the Pomodoro technique which will also connect to a website using APIs.

With major progress being made on the application and API, we would need better designs to work with. For this, we created a more detailed drawing design of all screens for the website and desktop app.

Week 4:

The team started to plan what we are going to implement into the system to give it complexity such as what kind of data we want to have and what we need to produce a good fully functional product. We also created a base site and had it successfully hosted so that we can see progress.

This was then linked to GitHub via GitHub Actions. This way our changes made to the repository can be deployed to the server without individual developers needing to upload the latest version. This works out more secure and less hassle.

Week 5:

Our group leader began the creation of the backend API to allow the website and application to connect together. This was documented as sections completed as this was new territory for all of us.

Week 6:

Due to the lecturer falling ill, we hosted our group meeting online for this week. We added website security checks for the hosted site and created more in-depth designs for the website.

Week 7:

Our lecturer was still ill, so we hosted our group meeting online again. We started getting work done on the desktop application by adding a countdown timer and a tracker. The database has also been set up to work with the API so that data could start to be passed around.

Week 8:

We started the draft write up of our report and started adding requirement tables and entity-relationship diagrams. A design of the desktop application was made and the login support on the API and website was successfully integrated.

Week 9:

We created an entity-relationship diagram for the database and moved the signup route to point to the user post to fit the MVC design model of the desktop application.

Week 10:

This week, we updated the API responses to return in all lower case and added an introduction page to the actual website. We also made the profile page display 2 weeks unit histories and some basic stats so that users can compare how their previous 2 weeks went.

Week 11:

Big progress has been made this week. We finished off the API and finished the API document to easily show what responses can be expected for different routes. Login authentication was also made for both successful and failed login attempts and a fixed-up profile page. We added some easy keyboard shortcuts to make it easier for typists such as pressing enter to login and signup.

Week 12:

Our final week of work brought the product to the live production system. The first full version of the application was compiled for release and added to GitHub. This was also added to the live website so our end users can get a copy of it. The live deployment now uses a fixed database that is not reset upon change like our development environment. User stats are now persistent making long term use viable completing our system requirements.

8.2. Appendix 2

GitHub: https://github.com/JazerBarclay/ci536-group-project

Trello: https://trello.com/invite/b/Om4BROvV/6e87b9362f6c55d9a45ec60924b332a9/the-project

8.3. Appendix 3 – Documents concatenated below

In appendix 3 you will find attached the following:

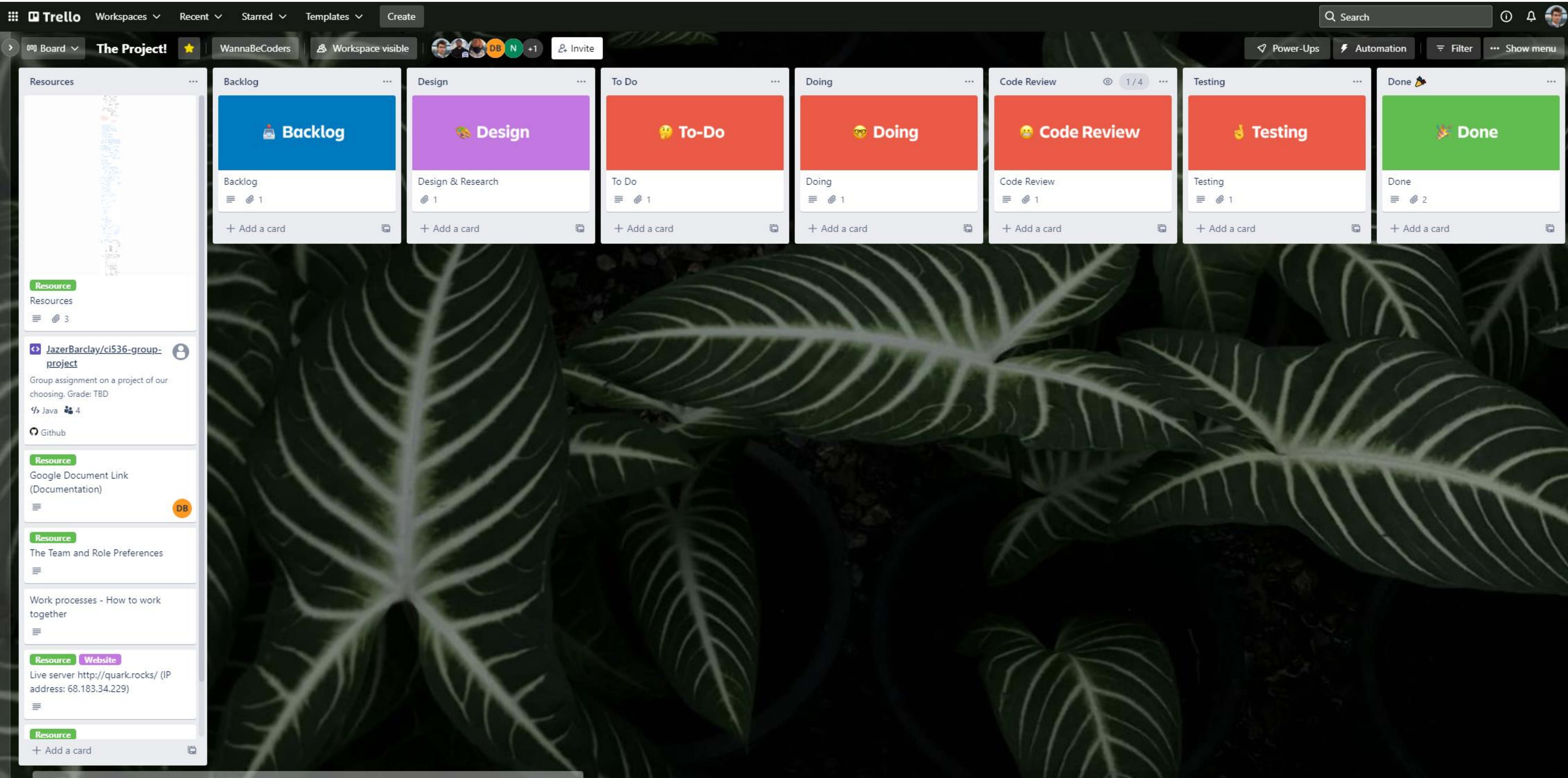
1. Individual contributions document of team member's contribution
2. Trello meeting screenshots
3. Entity relationship diagram for database
4. Version 2 hand drawn designs
5. Final wireframes and user stories
6. Software requirements specification

CI536 – Quark Group Submission Statement of Contribution

Student name	Percentage contribution
Jazer Barclay	100%
Kim Lam	100%
Karan Vani	100%
Fin Watling	100%
Ben Edmonson	100%
Danil Bronnikov	100%

Trello Setup and Completed List

Setup



Resources

Resource
Resources
3

JazerBarclay/ci536-group-project
Group assignment on a project of our choosing. Grade: TBD
Java 4
Github

Resource
Google Document Link (Documentation)
DB

Resource
The Team and Role Preferences

Work processes - How to work together

Resource Website
Live server http://quark.rocks/ (IP address: 68.183.34.229)

Resource

Backlog
Backlog
Backlog
1

Design
Design
Design & Research
1

To Do
To-Do
To Do
1

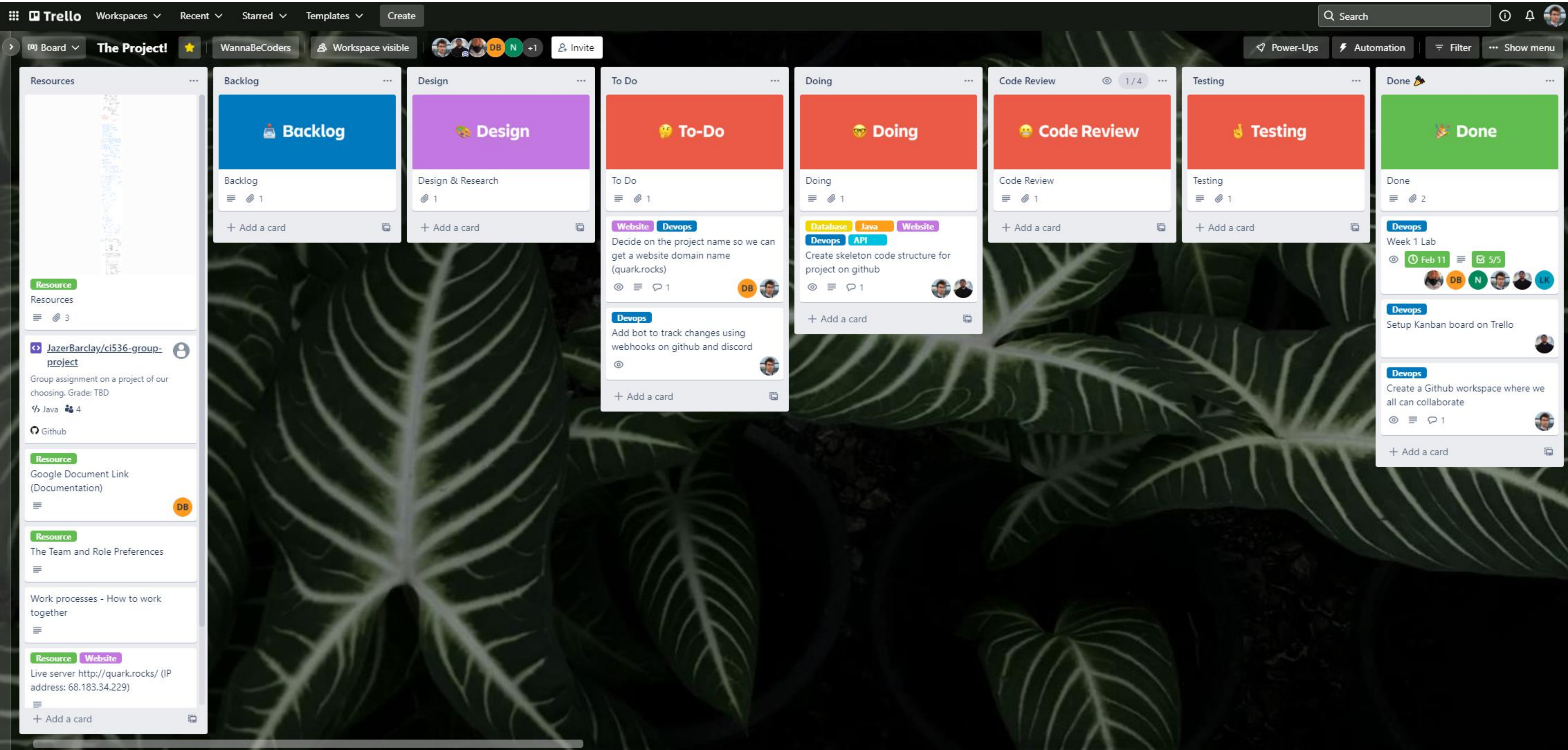
Doing
Doing
Doing
1

Code Review 1 / 4
Code Review
Code Review
1

Testing
Testing
Testing
1

Done
Done
Done
2

Week 1



Resources

- Resource — Resources — 3
- JazerBarclay/ci536-group-project — Group assignment on a project of our choosing. Grade: TBD — Java — 4 — Github
- Resource — Google Document Link (Documentation) — DB
- Resource — The Team and Role Preferences
- Work processes - How to work together
- Resource, Website — Live server http://quark.rocks/ (IP address: 68.183.34.229)
- + Add a card

Backlog

- Backlog — Backlog — 1
- + Add a card

Design

- Design — Design & Research — 1
- + Add a card

To Do

- To-Do — To Do — 1
- Website, Devops — Decide on the project name so we can get a website domain name (quark.rocks) — 1 — DB
- Devops — Add bot to track changes using webhooks on github and discord
- + Add a card

Doing

- Doing — Doing — 1
- Database, Java, Website, Devops, API — Create skeleton code structure for project on github — 1
- + Add a card

Code Review 1 / 4

- Code Review — Code Review — 1
- + Add a card

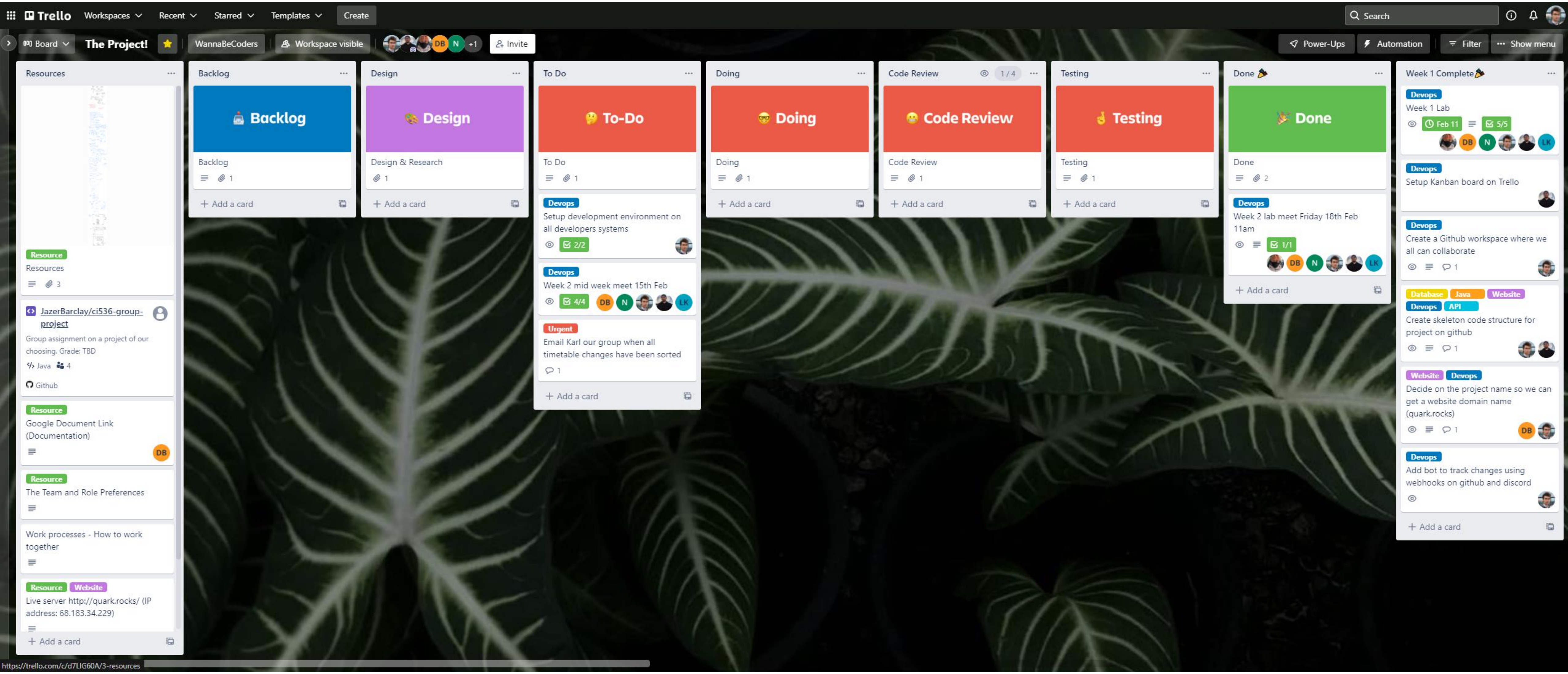
Testing

- Testing — Testing — 1
- + Add a card

Done

- Done — Done — 2
- Devops — Week 1 Lab — Feb 11 — 5/5 — DB, N, LK
- Devops — Setup Kanban board on Trello
- Devops — Create a Github workspace where we all can collaborate — 1
- + Add a card

Week 2



Resources

- Resource — Resources — 3
- JazerBarclay/ci536-group-project — Group assignment on a project of our choosing. Grade: TBD — Java — 4 — Github
- Resource — Google Document Link (Documentation) — DB
- Resource — The Team and Role Preferences
- Work processes - How to work together
- Resource, Website — Live server http://quark.rocks/ (IP address: 68.183.34.229)
- + Add a card

Backlog

- Backlog — Backlog — 1
- + Add a card

Design

- Design — Design & Research — 1
- + Add a card

To Do

- To-Do — To Do — 1
- Devops — Setup development environment on all developers systems — 2/2
- Devops — Week 2 mid week meet 15th Feb — 4/4 — DB N LK
- Urgent — Email Karl our group when all timetable changes have been sorted — 1
- + Add a card

Doing

- Doing — Doing — 1
- + Add a card

Code Review — 1 / 4

- Code Review — Code Review — 1
- + Add a card

Testing

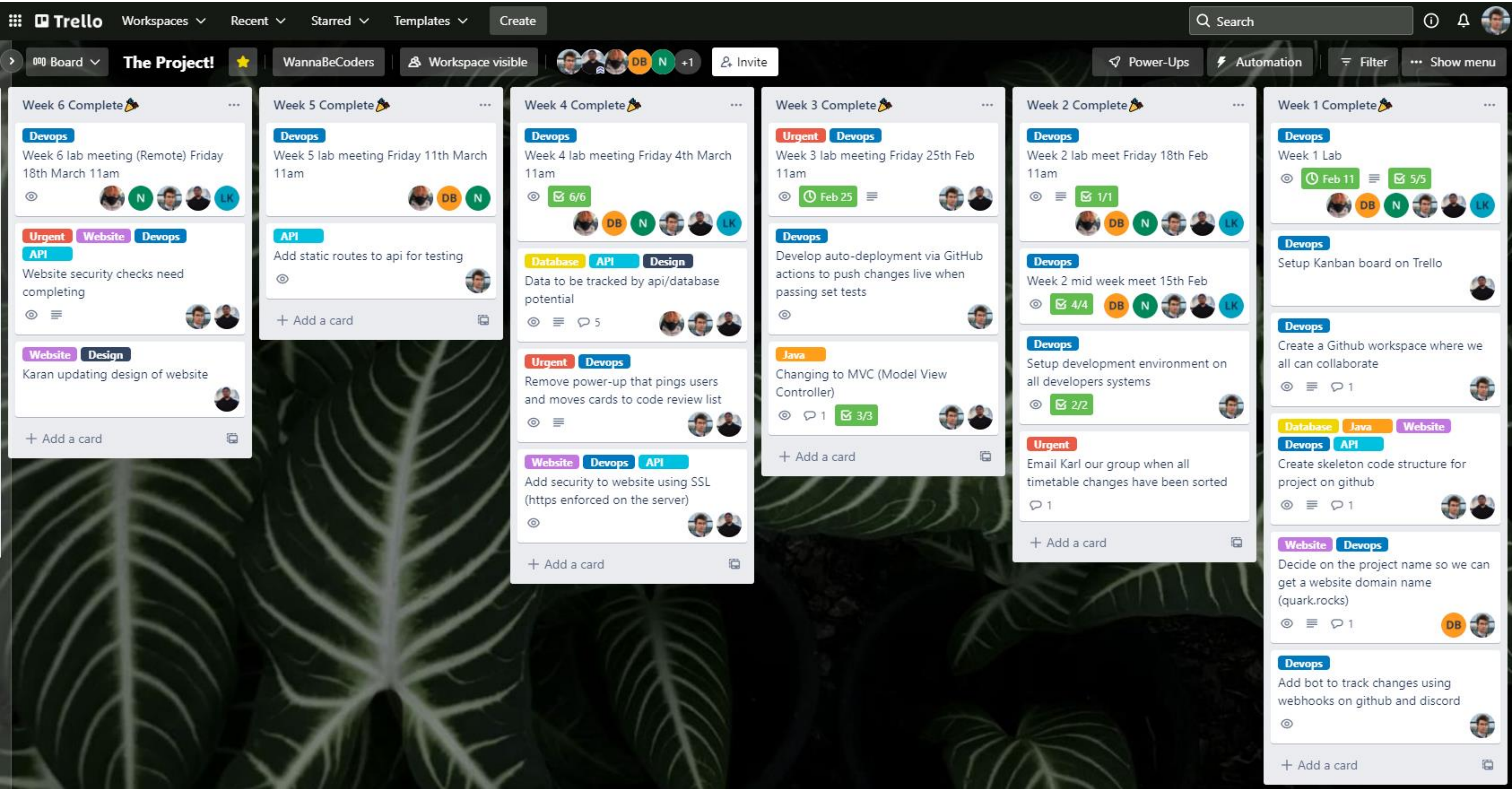
- Testing — Testing — 1
- + Add a card

Done

- Done — Done — 2
- Devops — Week 2 lab meet Friday 18th Feb 11am — 1/1 — DB N LK
- + Add a card

Week 1 Complete

- Devops — Week 1 Lab — Feb 11 — 5/5 — DB N LK
- Devops — Setup Kanban board on Trello
- Devops — Create a Github workspace where we all can collaborate — 1
- Database, Java, Website, Devops, API — Create skeleton code structure for project on github — 1
- Website, Devops — Decide on the project name so we can get a website domain name (quark.rocks) — 1 — DB
- Devops — Add bot to track changes using webhooks on github and discord
- + Add a card



Week 6 Complete

- Devops
  Week 6 lab meeting (Remote) Friday 18th March 11am
- Urgent Website Devops API
  Website security checks need completing
- Website Design
  Karan updating design of website

+ Add a card

Week 5 Complete

- Devops
  Week 5 lab meeting Friday 11th March 11am
- API
  Add static routes to api for testing

+ Add a card

Week 4 Complete

- Devops
  Week 4 lab meeting Friday 4th March 11am
  6/6
- Database API Design
  Data to be tracked by api/database potential
  5
- Urgent Devops
  Remove power-up that pings users and moves cards to code review list
- Website Devops API
  Add security to website using SSL (https enforced on the server)

+ Add a card

Week 3 Complete

- Urgent Devops
  Week 3 lab meeting Friday 25th Feb 11am
  Feb 25
- Devops
  Develop auto-deployment via GitHub actions to push changes live when passing set tests
- Java
  Changing to MVC (Model View Controller)
  1 3/3

+ Add a card

Week 2 Complete

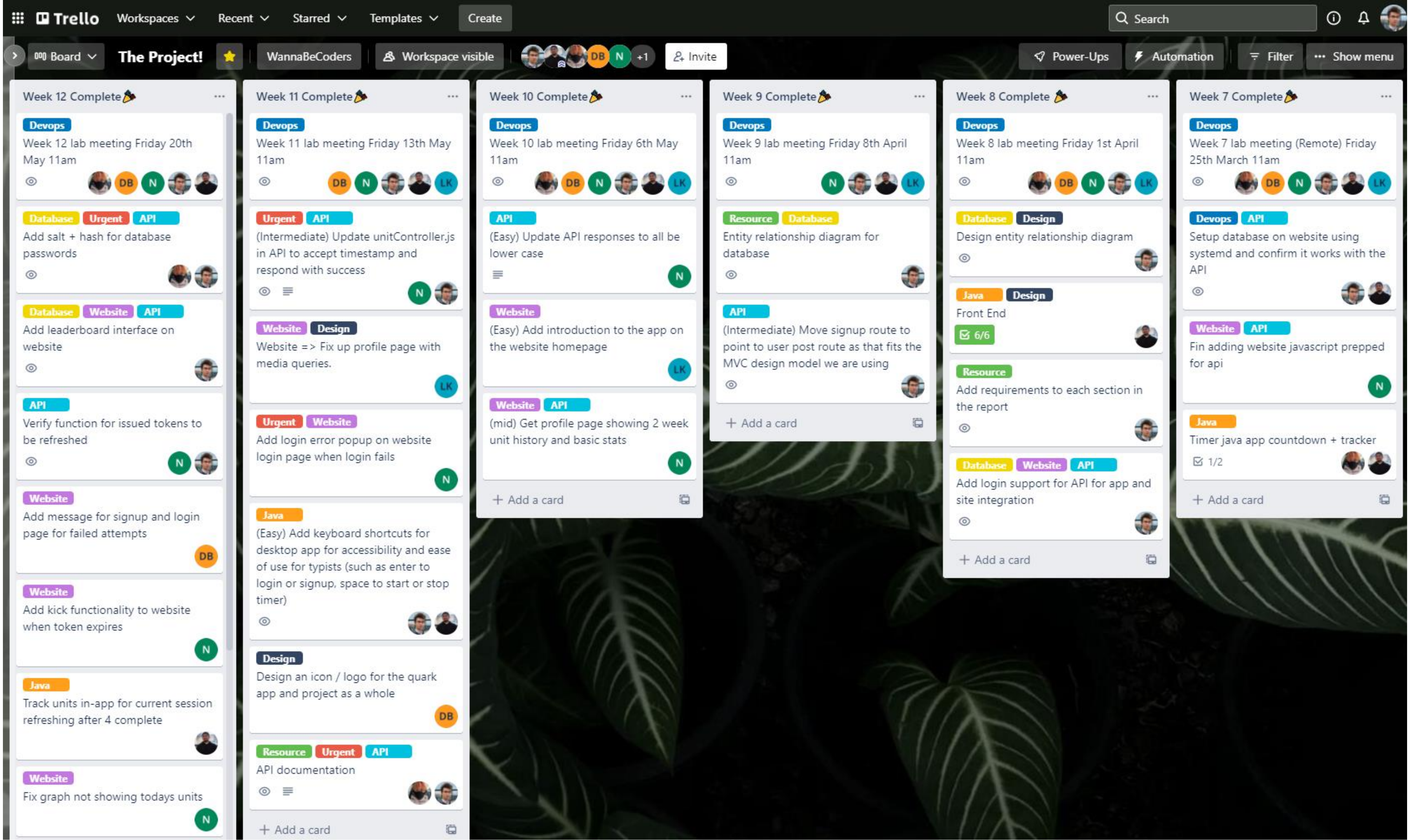
- Devops
  Week 2 lab meet Friday 18th Feb 11am
  1/1
- Devops
  Week 2 mid week meet 15th Feb
  4/4
- Devops
  Setup development environment on all developers systems
  2/2
- Urgent
  Email Karl our group when all timetable changes have been sorted
  1

+ Add a card

Week 1 Complete

- Devops
  Week 1 Lab
  Feb 11 5/5
- Devops
  Setup Kanban board on Trello
- Devops
  Create a Github workspace where we all can collaborate
  1
- Database Java Website Devops API
  Create skeleton code structure for project on github
  1
- Website Devops
  Decide on the project name so we can get a website domain name (quark.rocks)
  1
- Devops
  Add bot to track changes using webhooks on github and discord

+ Add a card



Week 12 Complete

Devops
Week 12 lab meeting Friday 20th May 11am
DB N

Database Urgent API
Add salt + hash for database passwords

Database Website API
Add leaderboard interface on website

API
Verify function for issued tokens to be refreshed
N

Website
Add message for signup and login page for failed attempts
DB

Website
Add kick functionality to website when token expires
N

Java
Track units in-app for current session refreshing after 4 complete

Website
Fix graph not showing todays units
N

Week 11 Complete

Devops
Week 11 lab meeting Friday 13th May 11am
DB N LK

Urgent API
(Intermediate) Update unitController.js in API to accept timestamp and respond with success
N

Website Design
Website => Fix up profile page with media queries.
LK

Urgent Website
Add login error popup on website login page when login fails
N

Java
(Easy) Add keyboard shortcuts for desktop app for accessibility and ease of use for typists (such as enter to login or signup, space to start or stop timer)

Design
Design an icon / logo for the quark app and project as a whole
DB

Resource Urgent API
API documentation

+ Add a card

Week 10 Complete

Devops
Week 10 lab meeting Friday 6th May 11am
DB N LK

API
(Easy) Update API responses to all be lower case
N

Website
(Easy) Add introduction to the app on the website homepage
LK

Website API
(mid) Get profile page showing 2 week unit history and basic stats
N

+ Add a card

Week 9 Complete

Devops
Week 9 lab meeting Friday 8th April 11am
N LK

Resource Database
Entity relationship diagram for database

API
(Intermediate) Move signup route to point to user post route as that fits the MVC design model we are using

+ Add a card

Week 8 Complete

Devops
Week 8 lab meeting Friday 1st April 11am
DB N LK

Database Design
Design entity relationship diagram

Java Design
Front End
6/6

Resource
Add requirements to each section in the report

Database Website API
Add login support for API for app and site integration

+ Add a card

Week 7 Complete

Devops
Week 7 lab meeting (Remote) Friday 25th March 11am
DB N LK

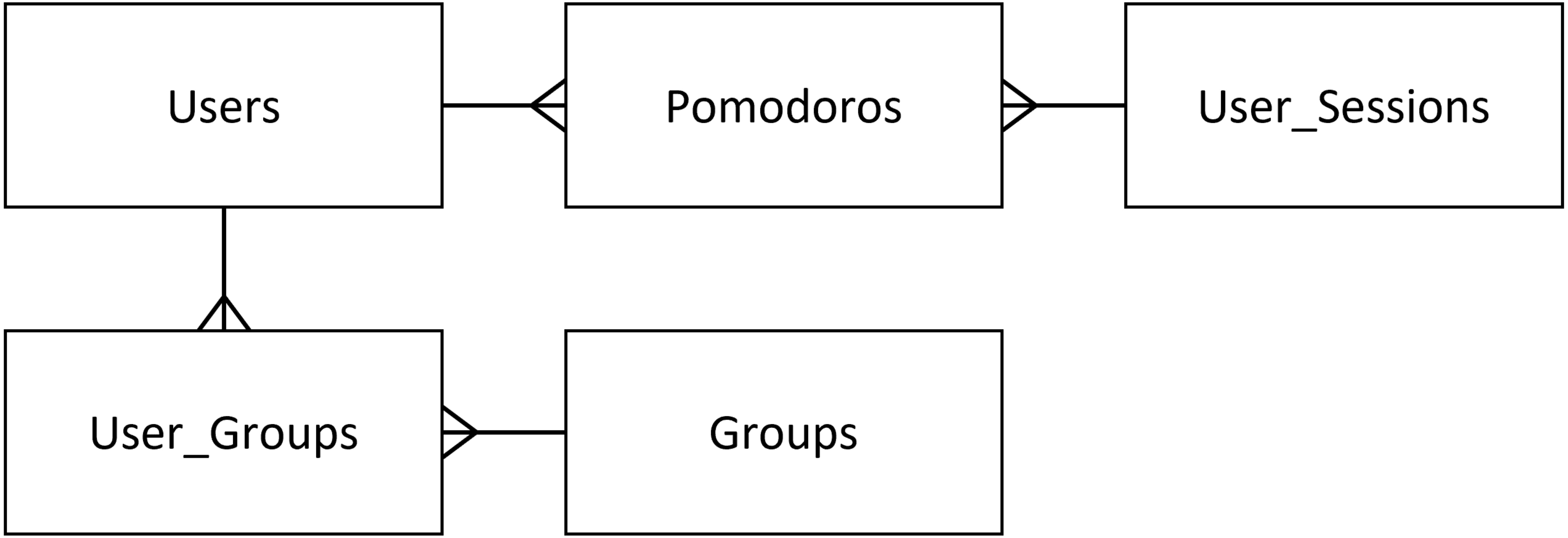
Devops API
Setup database on website using systemd and confirm it works with the API

Website API
Fin adding website javascript prepped for api
N

Java
Timer java app countdown + tracker
1/2

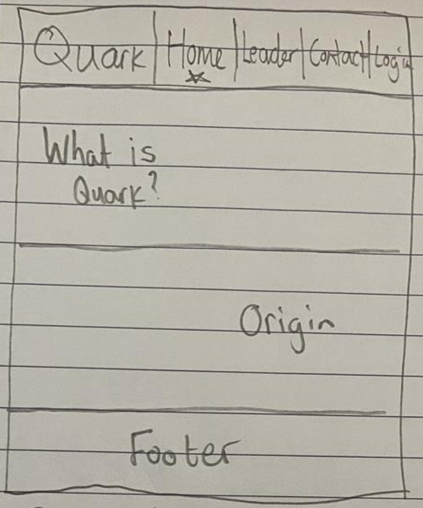
+ Add a card

Quark Database – Entity Relationship Diagram

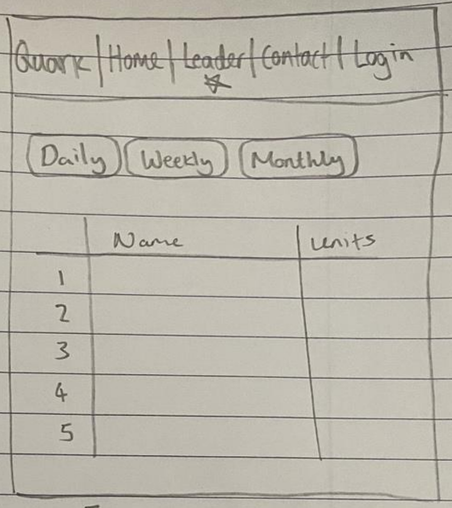


Quark Version 2 Designs

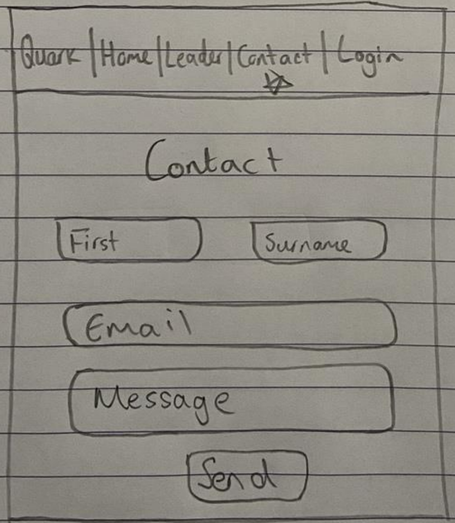
Webpage Design



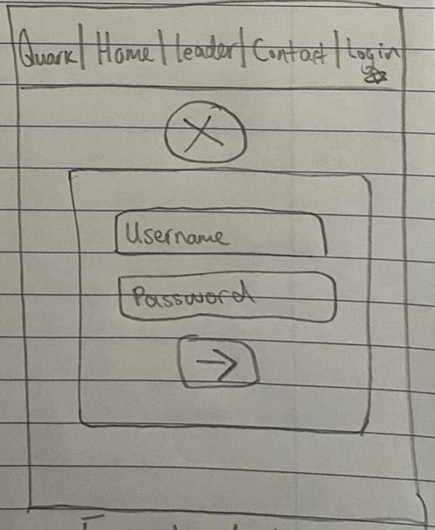
Example home page



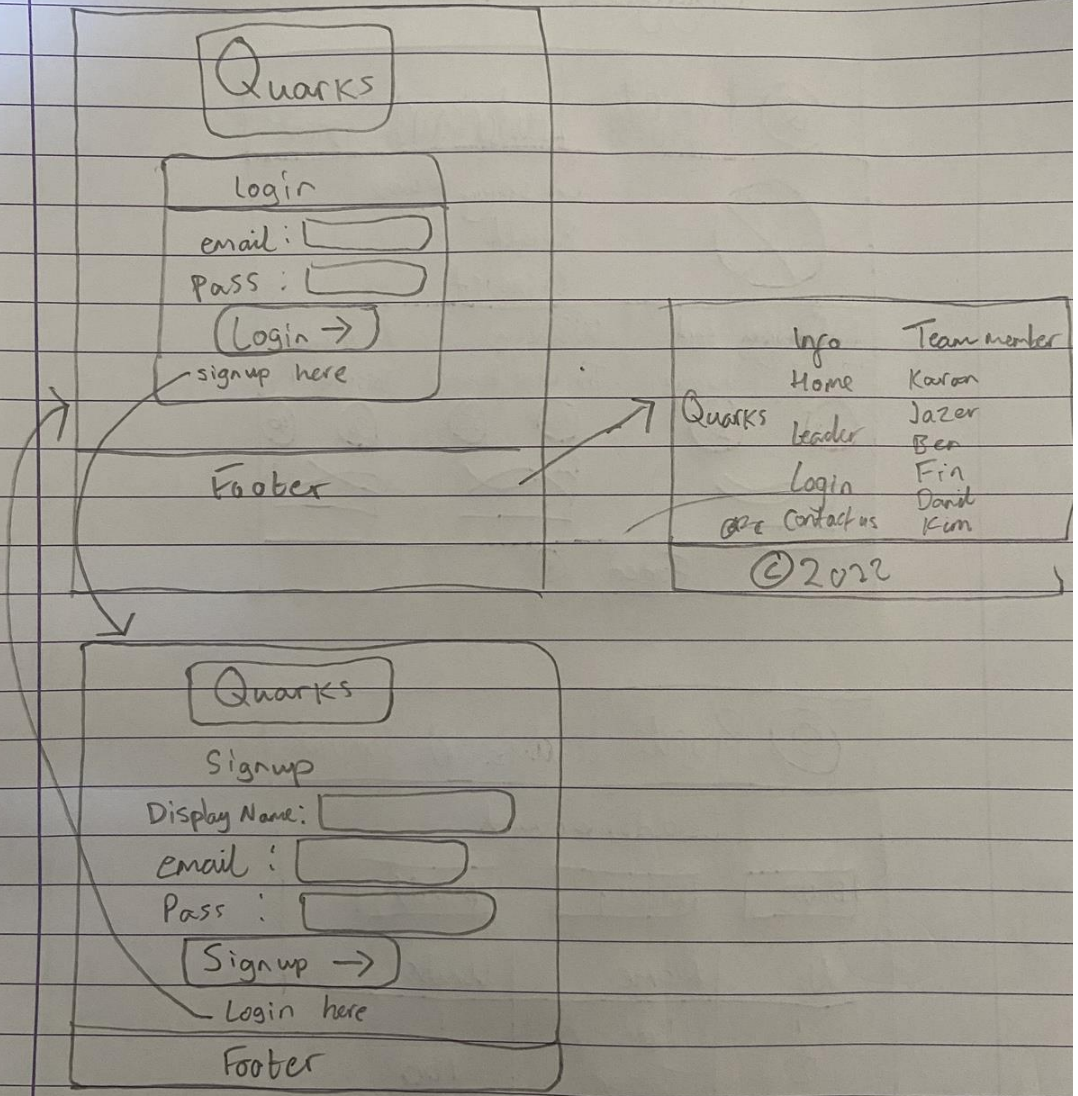
Example leaderboard



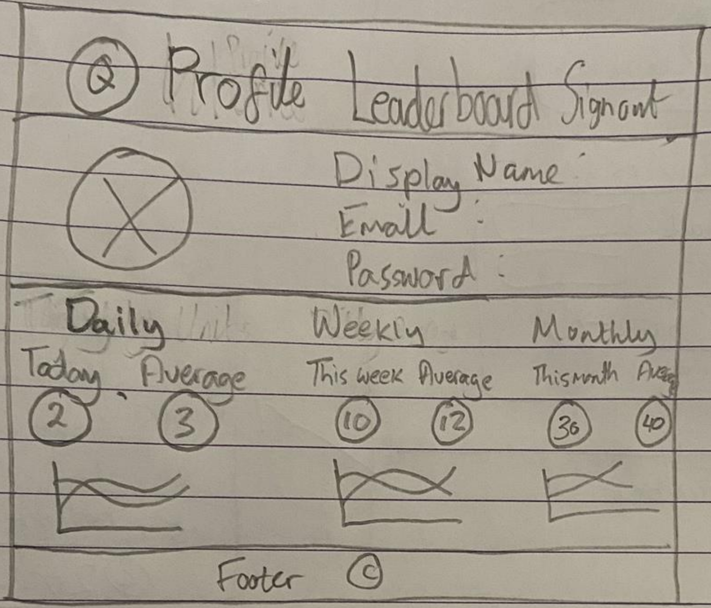
Example Contact

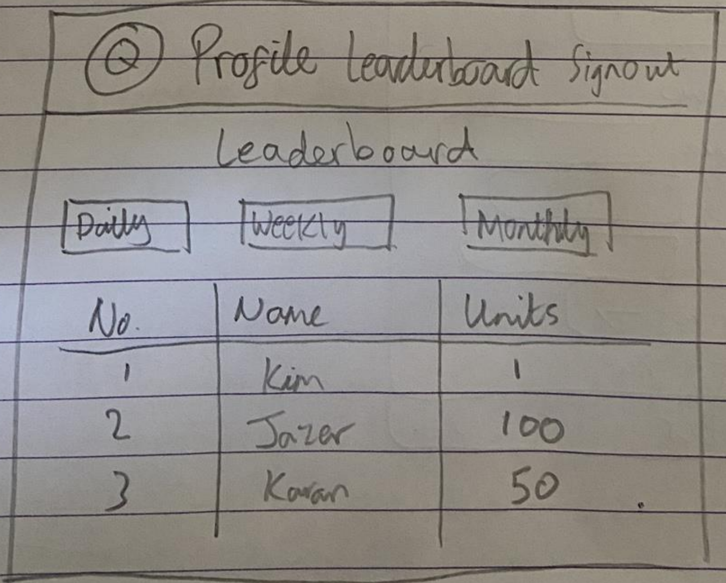


Example Login



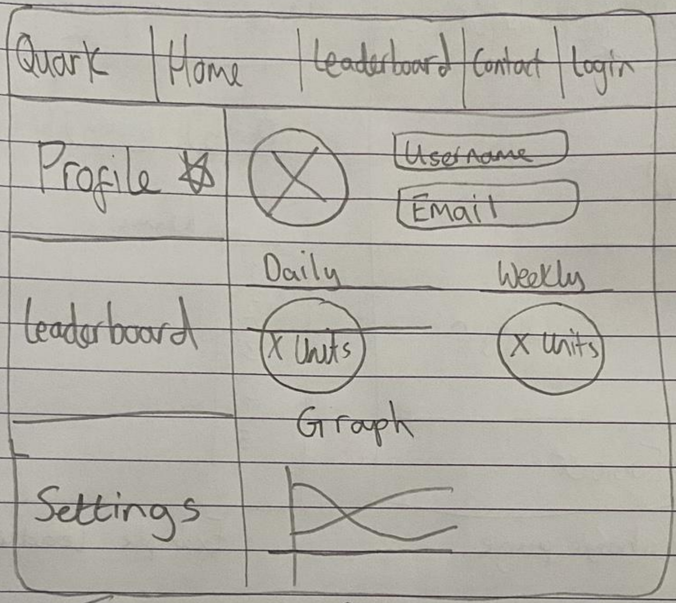
Logged In



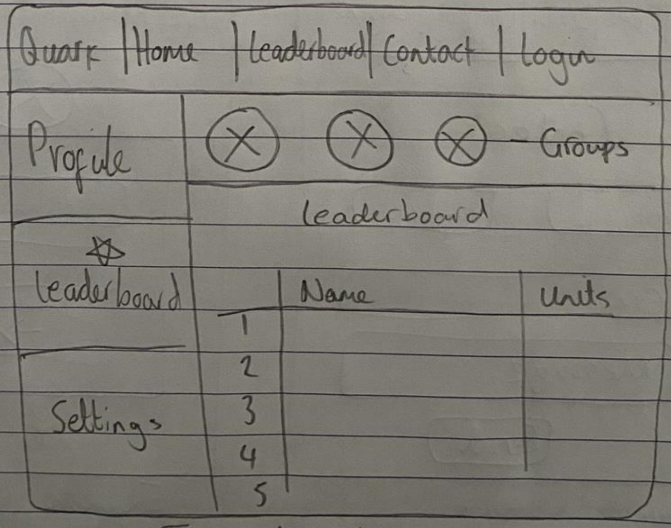


No.	Name	Units
1	Kim	1
2	Jazer	100
3	Karan	50

After Login Webpage



Example Main Profile



	Name	Units
1		
2		
3		
4		
5		

Example Leaderboard

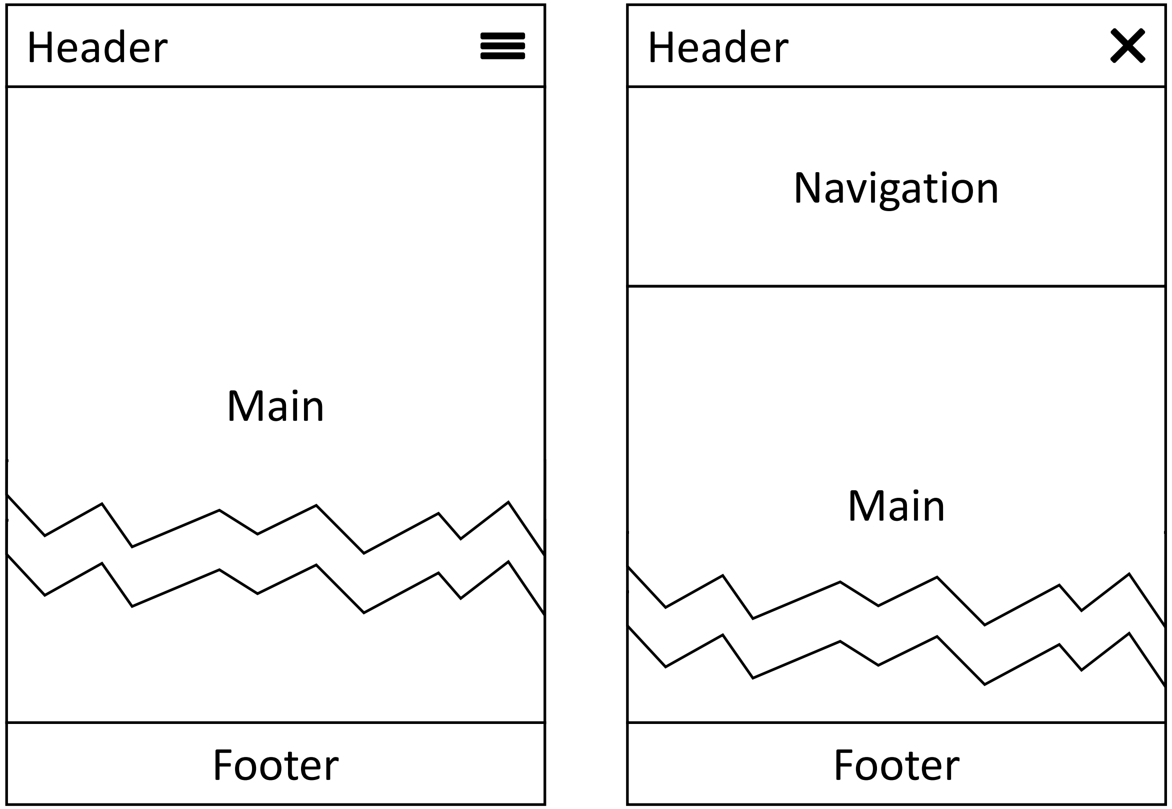
Quark Wireframes and user stories

A collection of design and layout considerations for the quark desktop application and website

Website

Page Layout

Mobile



Header ≡

Main

Footer

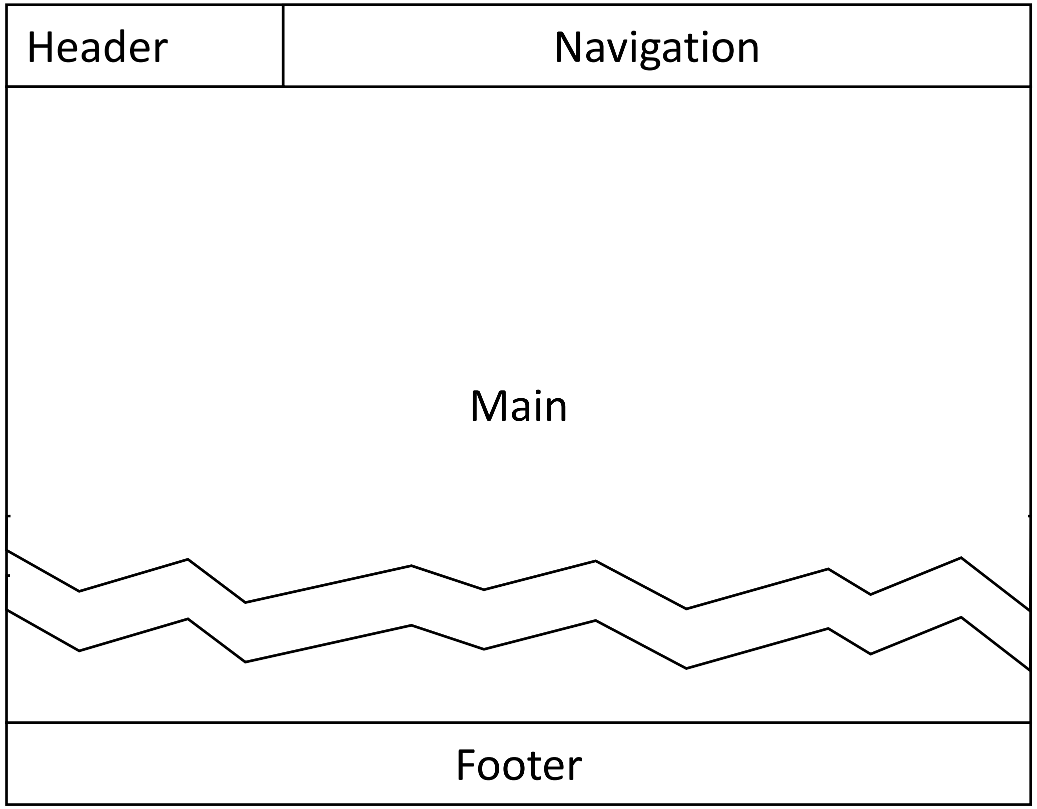
Header ✕

Navigation

Main

Footer

Desktop



Header

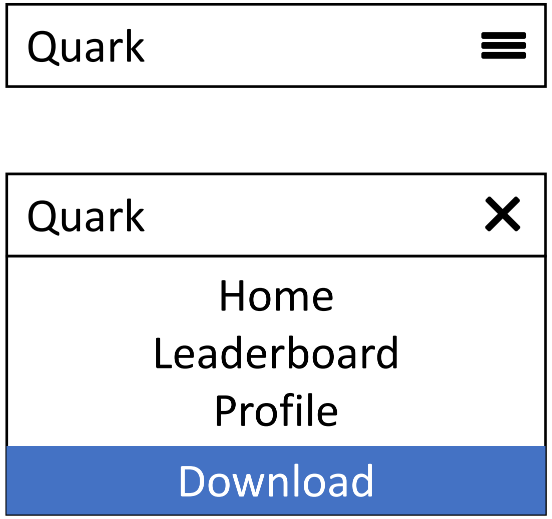
Navigation

Main

Footer

Navigation Design

Mobile



Quark ≡

Quark ✕

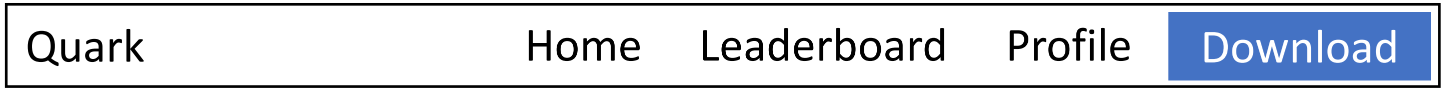
Home

Leaderboard

Profile

Download

Desktop



Quark Home Leaderboard Profile Download

Navigation will mostly be the same on all pages however we will replace the profile with login or sign up based on if the user is already logged into an account (show profile), not logged in (login) or on the login page (sign up).

Homepage

We want the homepage to welcome new users to the application.

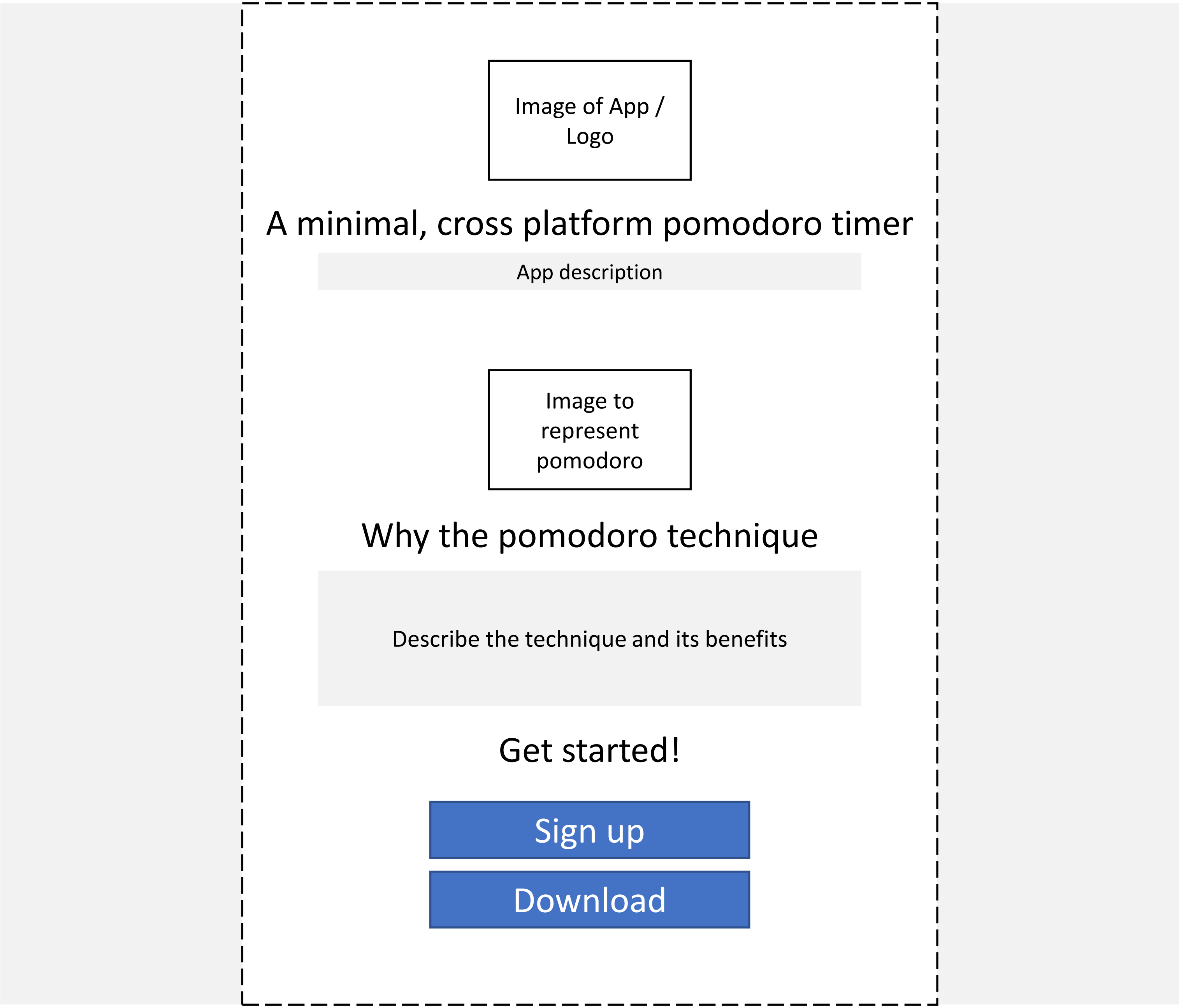


Image of App / Logo

A minimal, cross platform pomodoro timer

App description

Image to represent pomodoro

Why the pomodoro technique

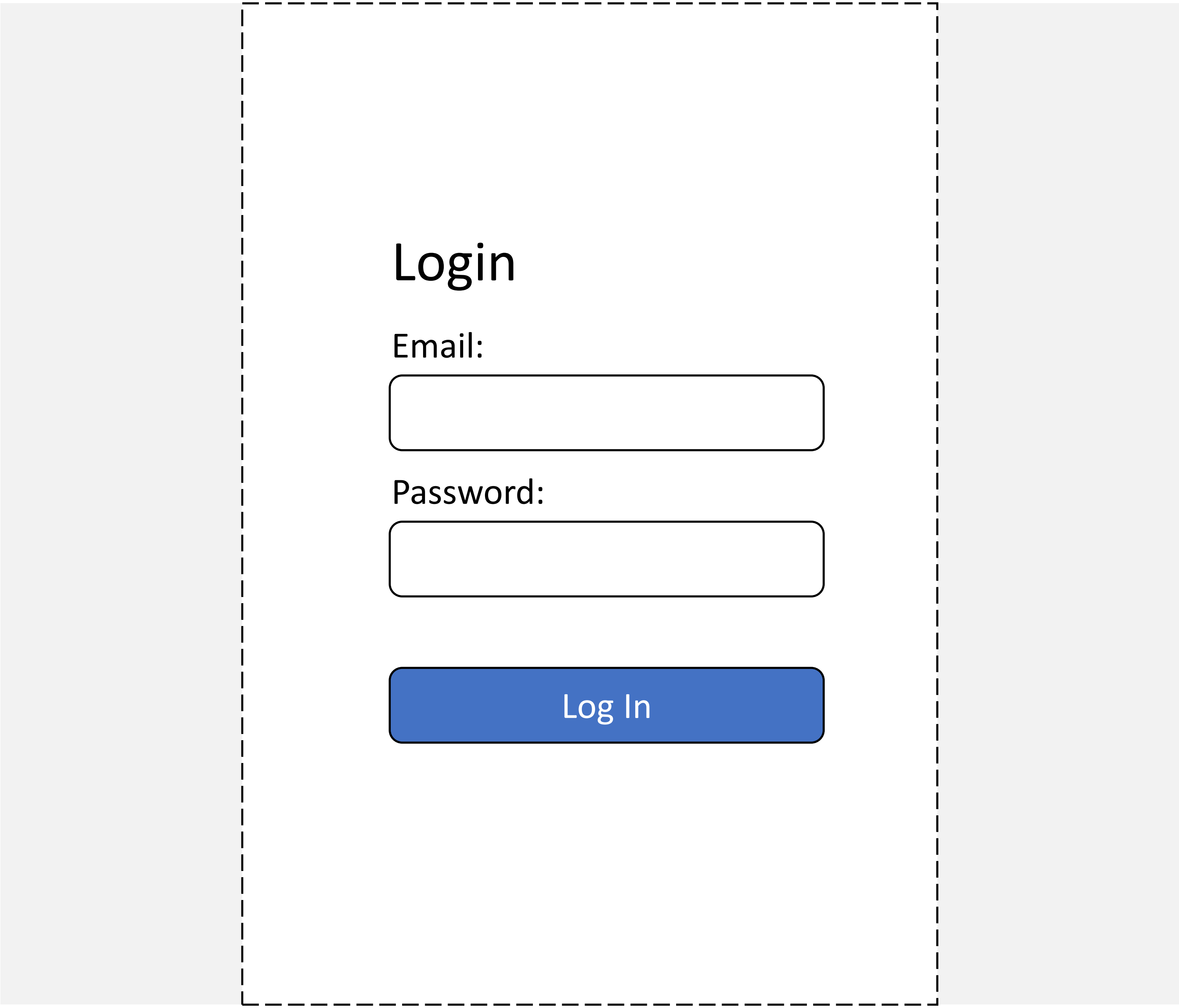
Describe the technique and its benefits

Get started!

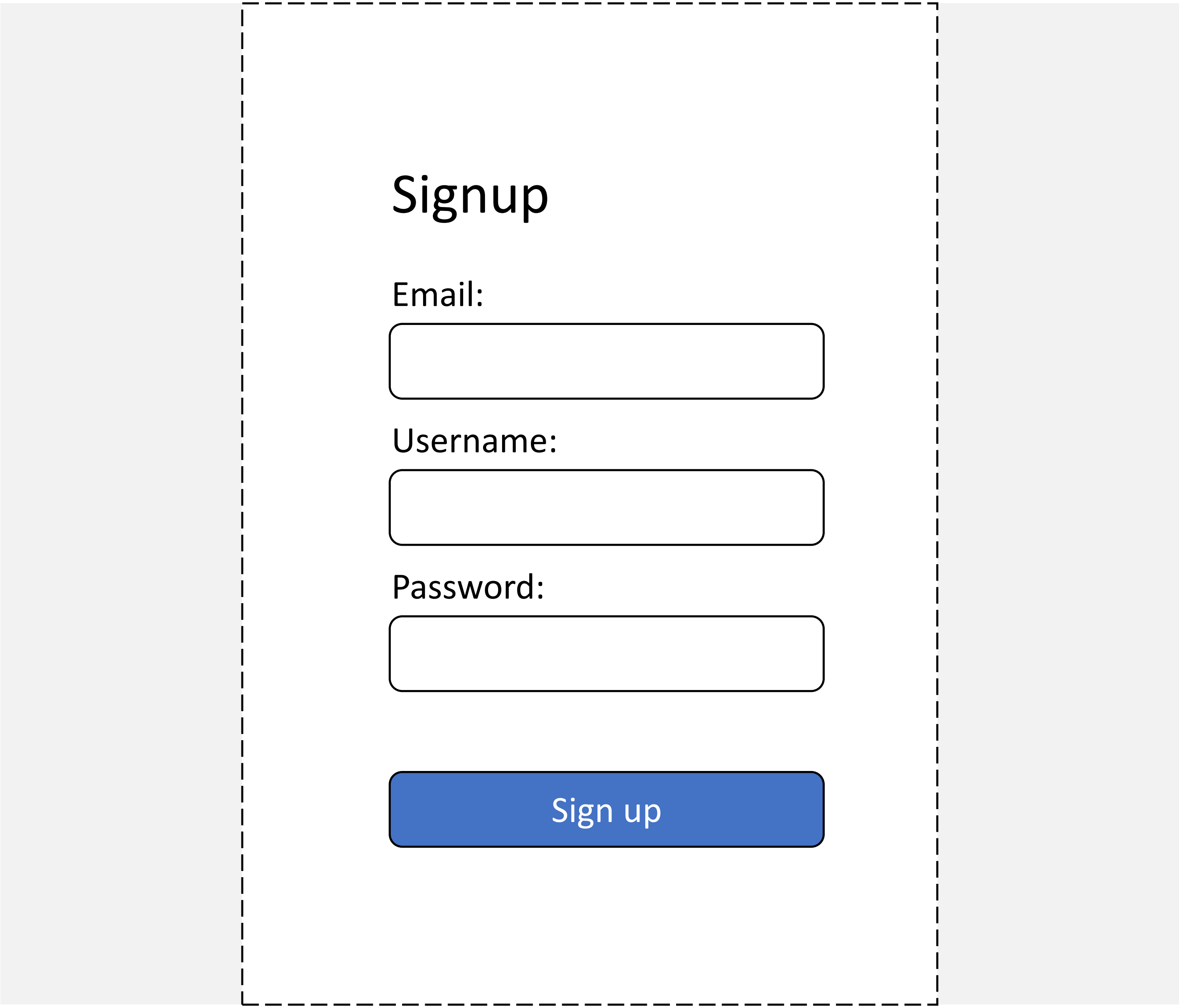
Sign up

Download

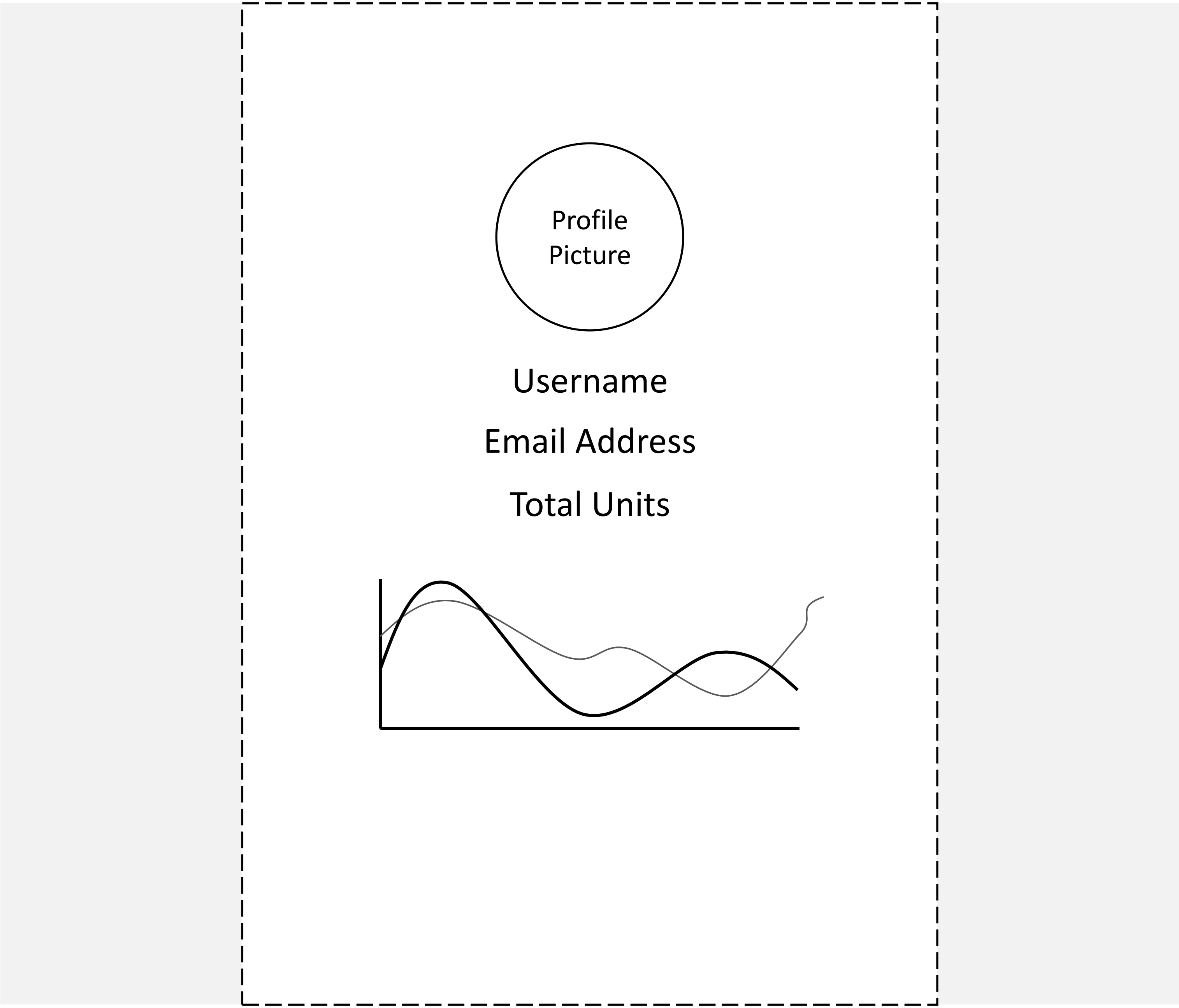
Login Page



Sign Up Page



Profile Page



Profile Picture

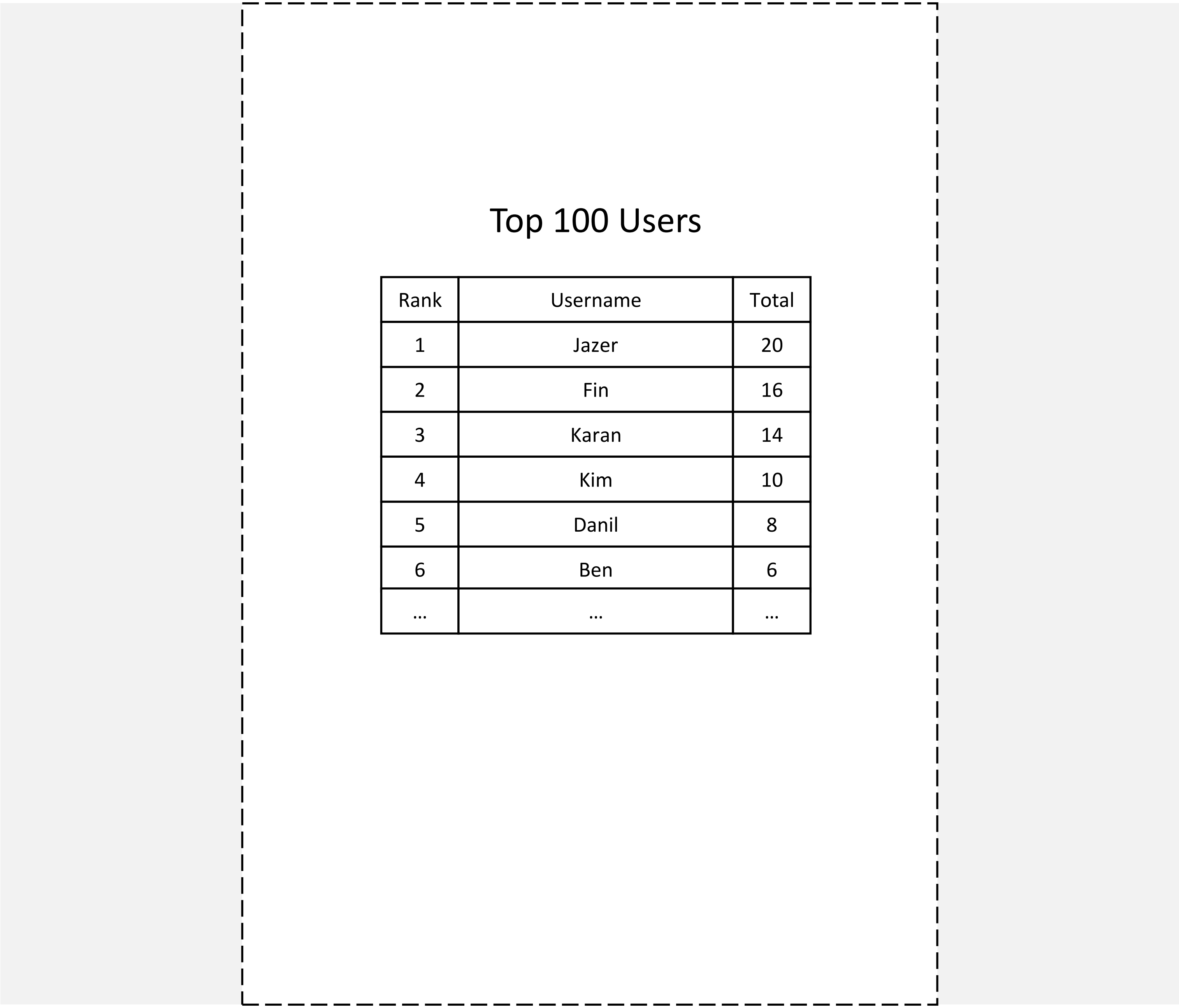
Username

Email Address

Total Units

Leaderboard

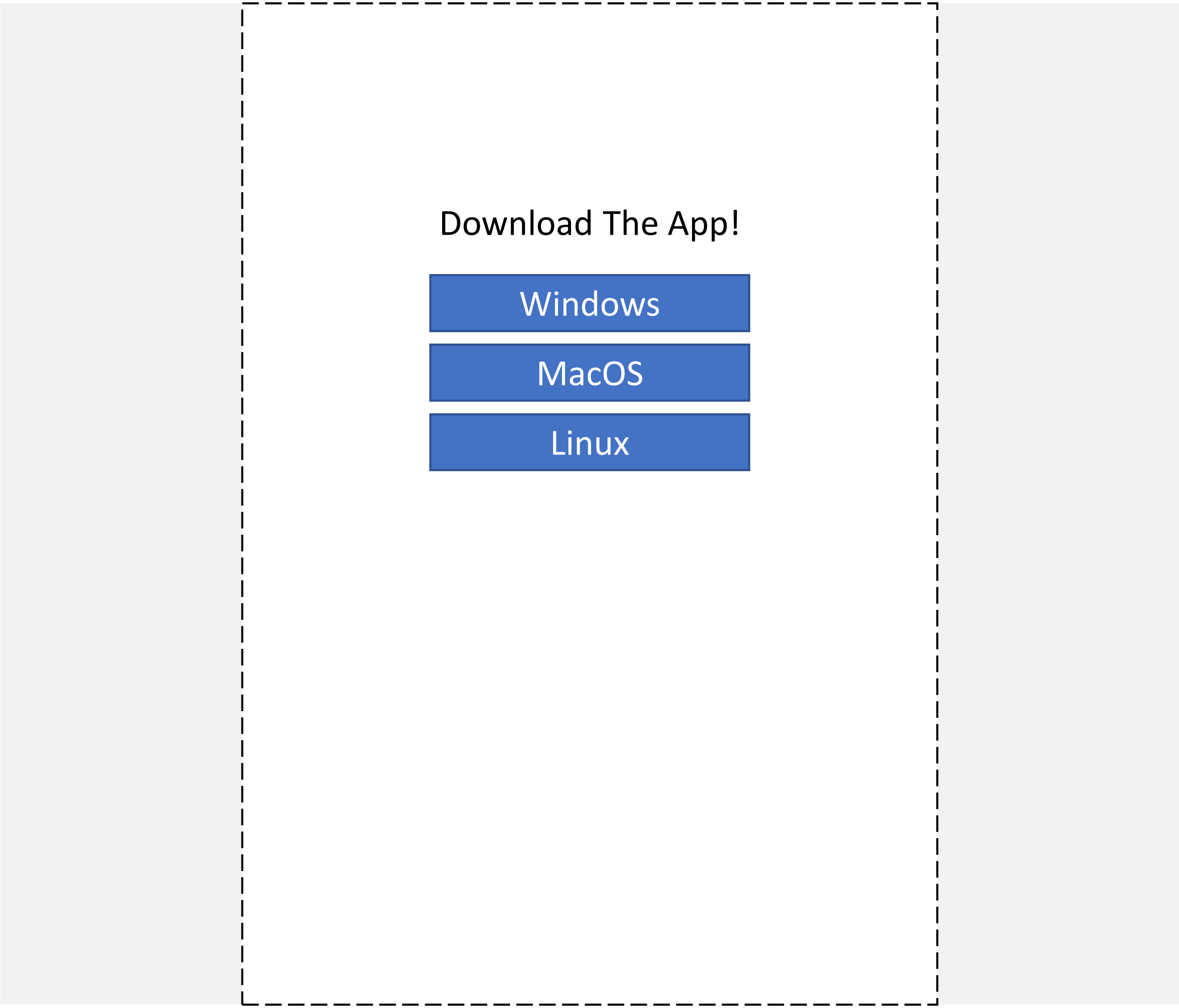
Only visible to logged in users



Top 100 Users

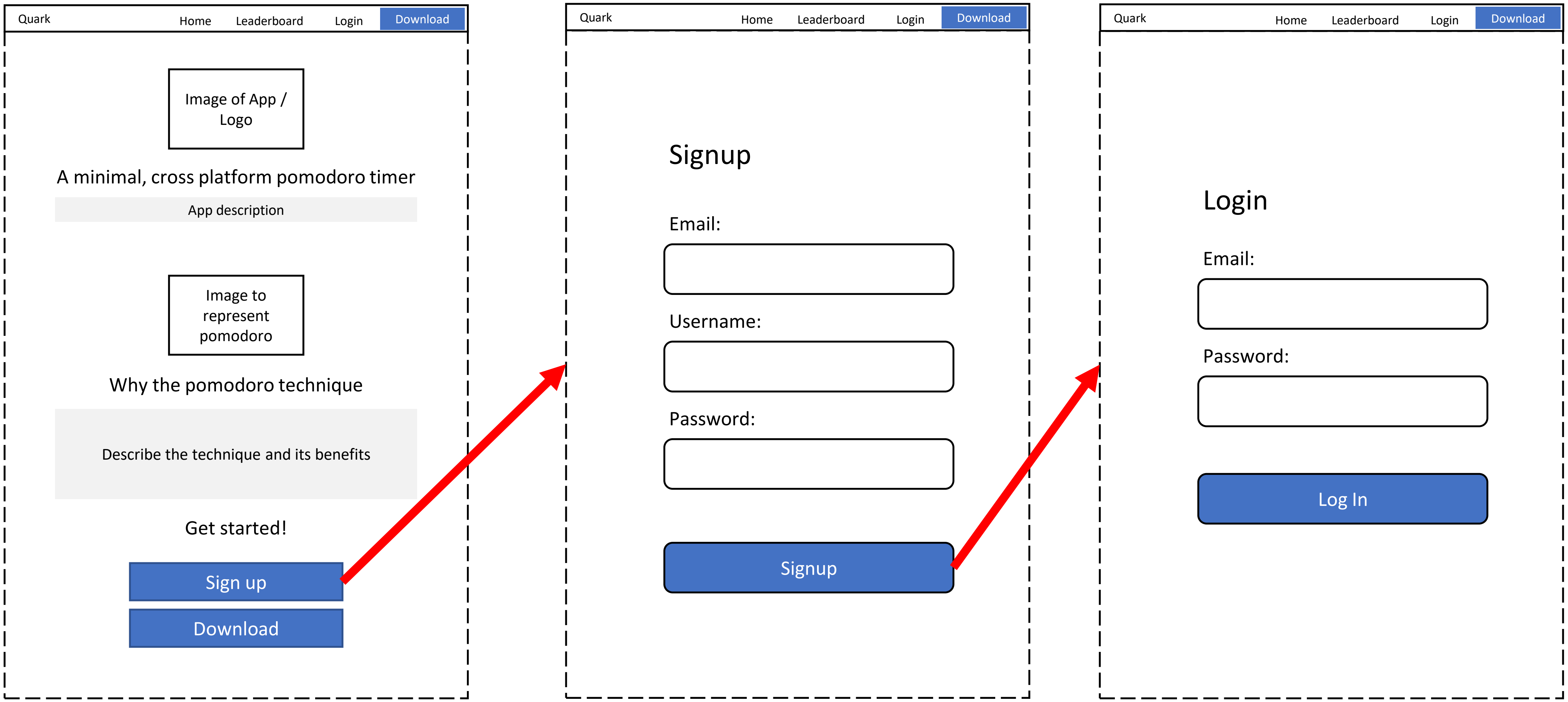
Rank	Username	Total
1	Jazer	20
2	Fin	16
3	Karan	14
4	Kim	10
5	Danil	8
6	Ben	6
…	…	…

Download Page



Download The App!

Signup Story



Quark Home Leaderboard Login Download

Image of App / Logo

A minimal, cross platform pomodoro timer

App description

Image to represent pomodoro

Why the pomodoro technique

Describe the technique and its benefits

Get started!

Sign up

Download

Quark Home Leaderboard Login Download

Signup

Email:

Username:

Password:

Signup

Quark Home Leaderboard Login Download

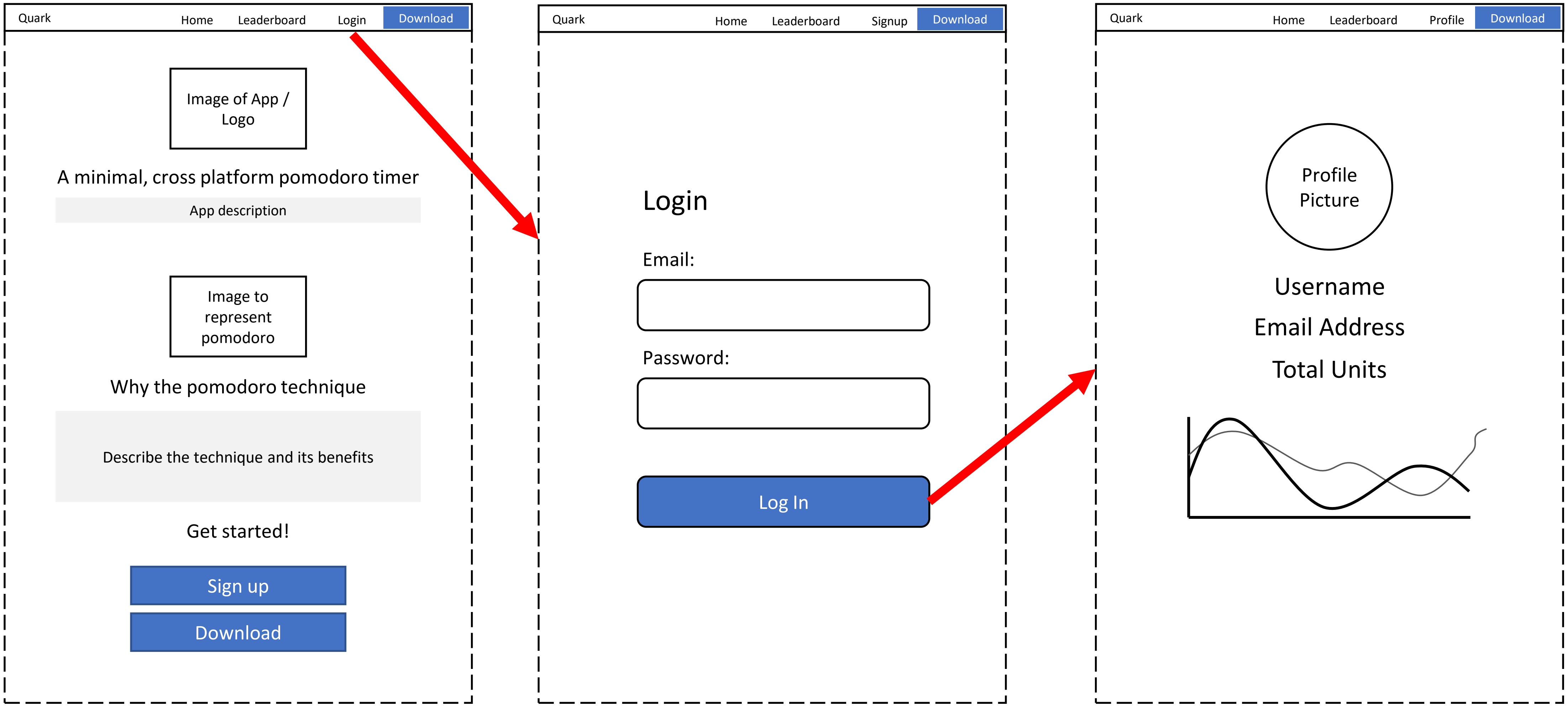
Login

Email:

Password:

Log In

Login / View Profile Story



Quark Home Leaderboard Login Download

Image of App / Logo

A minimal, cross platform pomodoro timer

App description

Image to represent pomodoro

Why the pomodoro technique

Describe the technique and its benefits

Get started!

Sign up

Download

Quark Home Leaderboard Signup Download

Login

Email:

Password:

Log In

Quark Home Leaderboard Profile Download

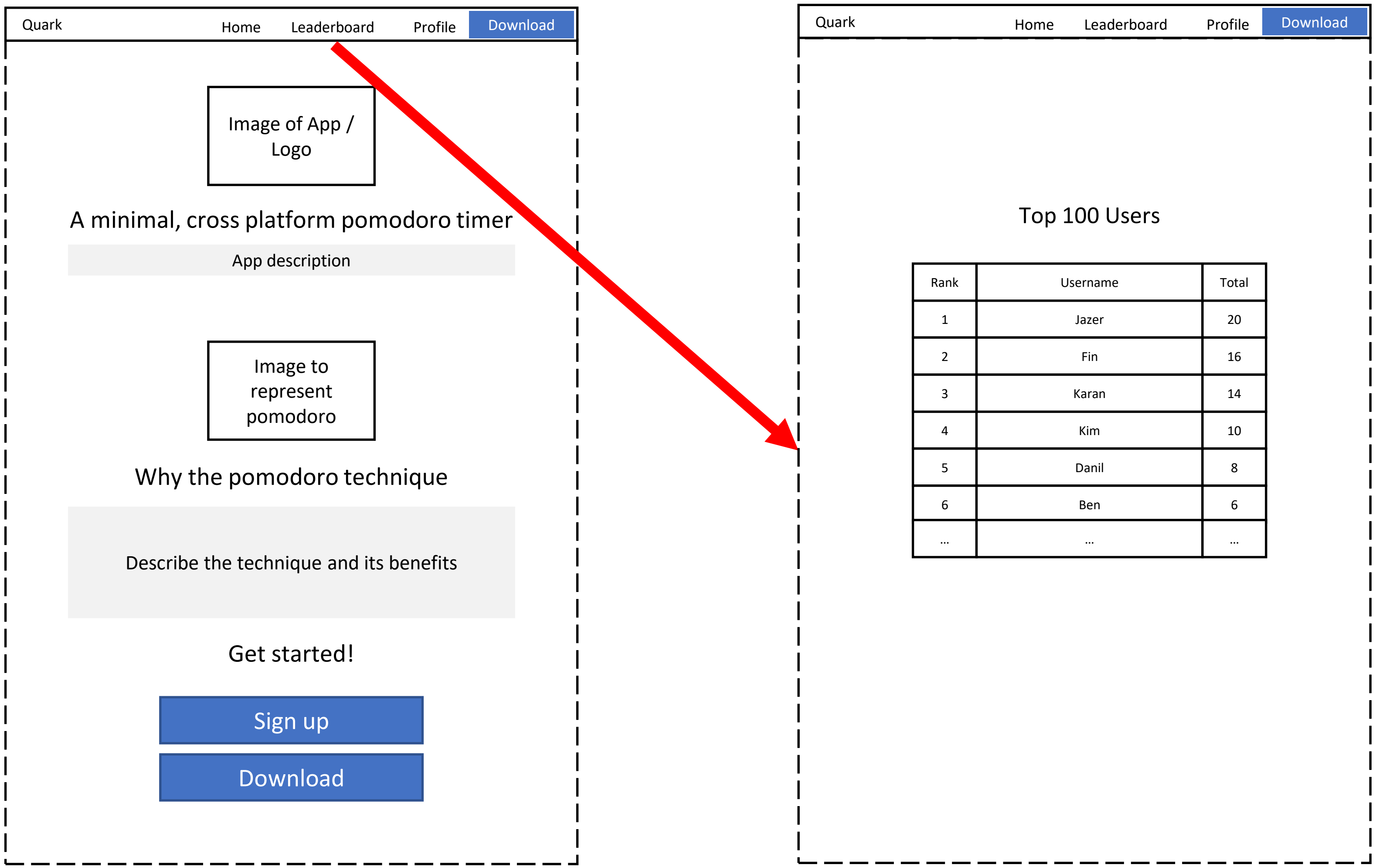
Profile Picture

Username

Email Address

Total Units

View Leaderboard Story



Quark Home Leaderboard Profile Download

Image of App / Logo

A minimal, cross platform pomodoro timer

App description

Image to represent pomodoro

Why the pomodoro technique

Describe the technique and its benefits

Get started!

Sign up

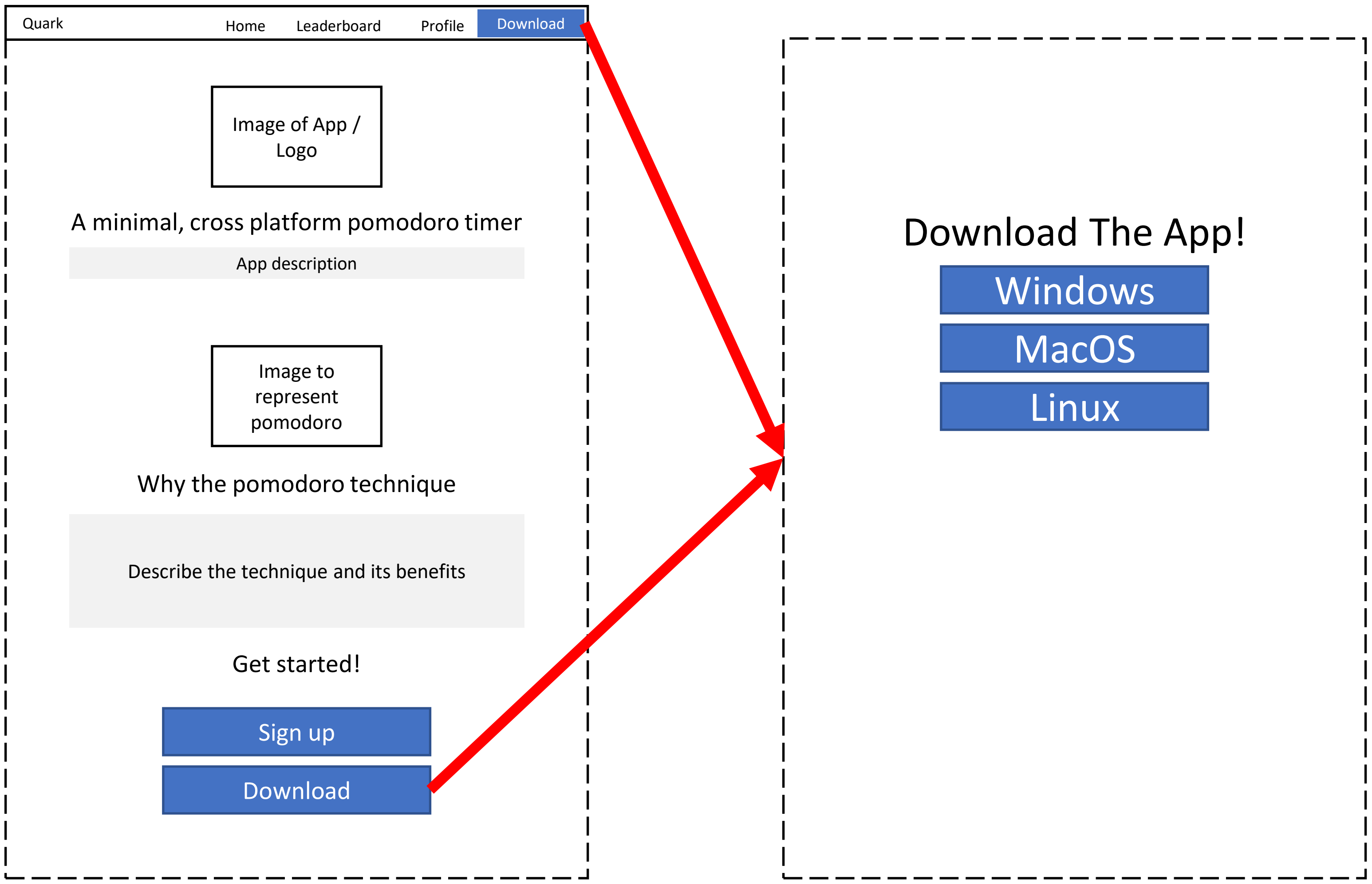
Download

Quark Home Leaderboard Profile Download

Top 100 Users

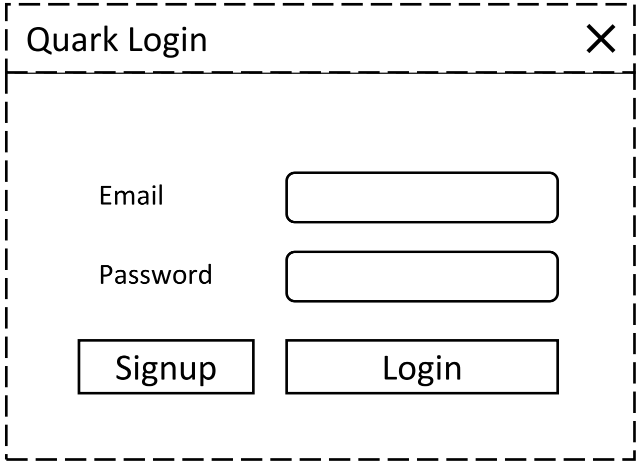
Rank	Username	Total
1	Jazer	20
2	Fin	16
3	Karan	14
4	Kim	10
5	Danil	8
6	Ben	6
…	…	…

Download Application Story

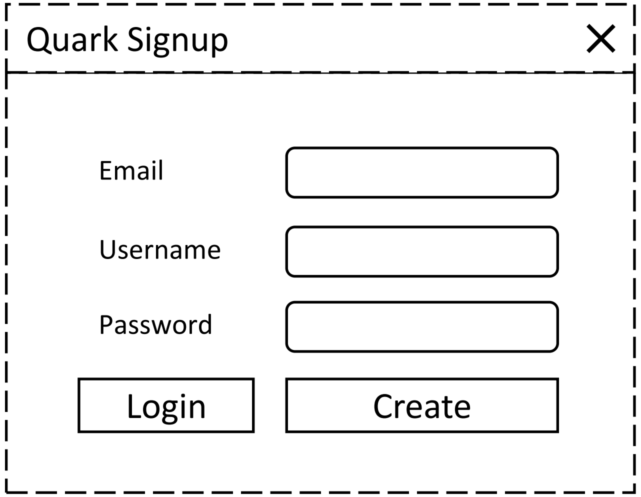

Quark
Home
Leaderboard
Profile
Download
Image of App / Logo
A minimal, cross platform pomodoro timer
App description
Image to represent pomodoro
Why the pomodoro technique
Describe the technique and its benefits
Get started!
Sign up
Download
Download The App!
Windows
MacOS
Linux

Desktop Application

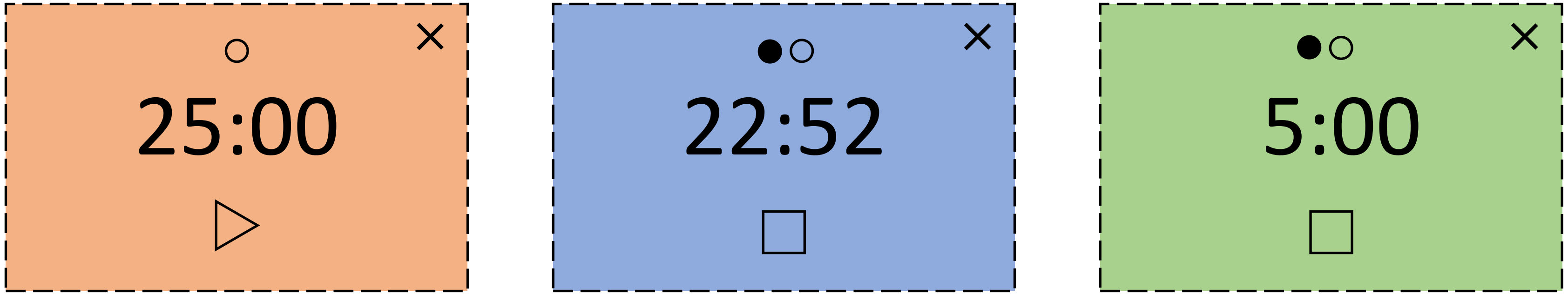
Login Screen



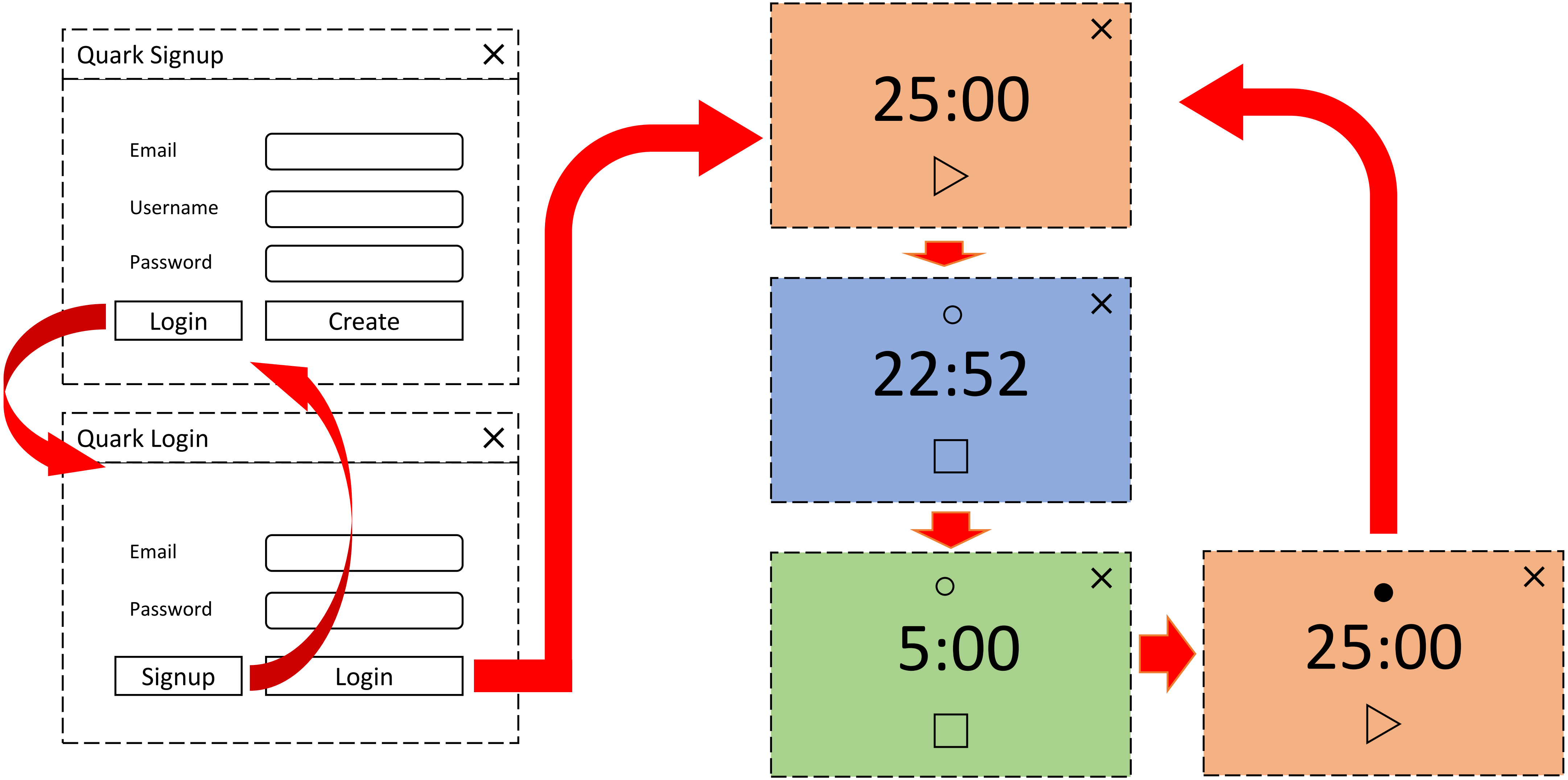
Signup Screen



Timer Screen



User Story (Signup, Login, Timer states)



Quark App

System Requirements Specification

Version 1.0 approved

Prepared by Jazer Barclay

Edited by Fin Watling and Kim Lam

University of Brighton

3rd June 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Jazer Barclay	22/03/22	Initial Version	0.1
Fin and Kim	02/06/22	First Version Complete	1.0

1. Introduction

1.1 Purpose

The purpose for this document is to explain the process by which the preliminary requirements from our project planning were analysed and refined.

1.2 Intended Audience and Reading Suggestions

This product is primarily targeted at knowledge workers in the academic field such as students in both foundational and further education and researchers.

This app, however, can be used for simple time tracking over extended periods of time allowing a more general usage.

1.3 Product Scope

This product is a software application which is accessed from a desktop application and is used to track work units in increments of 30 minutes with a split of 25 minutes of work and 5 minutes of rest.

The application extends to a website that can be accessed for historic tracking data for an individual to view and to compare against other users of the application in a ranked leader board.

1.4 References

- IEEE 830
- Business Analysis, Requirements, and Project Management (Chapter 4) [K. Cox (2022)]
- CI536 Requirements Documentation (Week 3) by Karl Cox (2022)

1.5 Overview

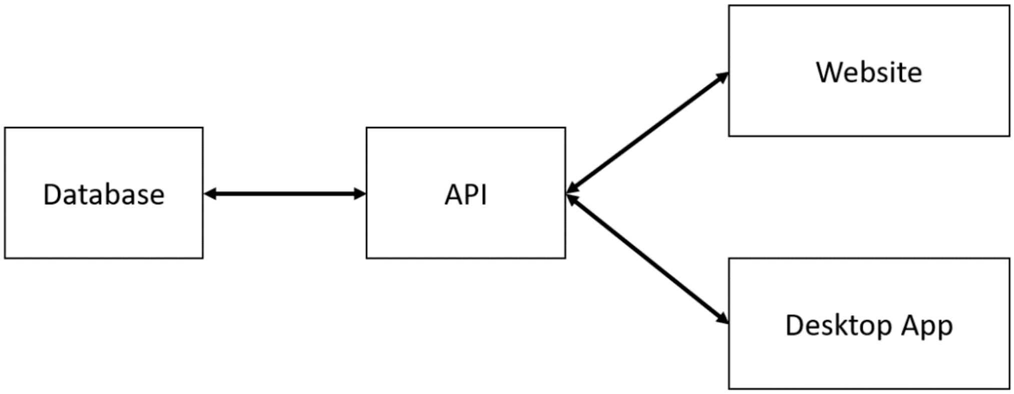
This document is laid out as per the IEEE-830 standard.

2. Overall Description

This system is designed to be a facility for tracking a user's time during extended work sessions and has many applications such as recording study sessions, research on topics that require great depth and complex writing or development tasks.

2.1 Product Perspective

The time tracking application is broken down into a back-end, website and desktop application. The back-end serves as the data storage system where custom user implementations can access the infrastructure for tracked session storage. The website is where the user may view their tracked information in a standard format and view the stats of other users of the system. The desktop application serves as a simple, minimal method to track their work sessions and upload the data to the back-end system.



2.2 Product Functions

The functionality is split between the website where insight is generated and the desktop app which serves as a method of time tracking via a fixed timer. The functions are labelled for where they will need implementing to meet the user requirements.

- Signup (Website & Application)
- Login (Website & Application)
- Start timer (Application)
- Stop timer (Application)
- Log 30 minute session (Application)
- View personal historic stats (Website)
- View other user's historic stats (Website)
- View leader board of all users (Website)

2.3 User Classes and Characteristics

There are three types of users in this system. The first two are authorised users and the third is non-authorised users. The first is the administrator. They have direct access to the server and thus the data, infrastructure and source code. They will be responsible for hosting and managing the services exposed to the second type of user.

The second type of user is the end user. They are the authorised users, customers, who have signed up for the service and operate the desktop application. Their main interaction is via the use of the timer and accessing their profile via the website or API.

The third type of user is the unauthorised user. They have not signed up for the service and as a result have no profile data of their own on the system. They may wish to access other end user's data on the system which they are not allowed access to until they have signed up for the service, becoming an end user.

All these users must have basic computer skills which include working with a web browser such as Google Chrome or Mozilla Firefox. Since all interactions with the UI of the desktop and website, the system cannot be used without access and knowledge of how desktop applications and specifically web browser functionality as part of their operating system.

2.4 Operating Environment

The desktop application will be supported on any desktop or laptop device running an up-to-date version of the Windows, MacOS or Linux operating systems.

The website will be accessible by any modern web browser from any device with a minimum screen resolution of 280x360 and JavaScript support.

2.5 Design and Implementation Constraints

- Data must be stored in a relational database for fast queries and optimal storage
- Passwords must be stored encrypted via a salt and hash function
- Unauthorised users must not be allowed to interact with the data via the website or associated API
- Secure communication between server and client via HTTPS connections
- Robust enough to handle server error responses and notify the end user

2.6 User Documentation

Along side the source code for the application and website, a PDF manual for the server-side API will be available for the developer audience.

3. External Interface Requirements

3.1 User Interfaces

On the website, there will be a navigation panel that allows easy access to traverse between any pages. This will have a wildcard link that, depending on login status and active screen, will show different links.

The pages must conform to a standard layout and scheme being mobile responsive and mutable for desktop use.

The desktop application must remain small in size and be unobtrusive to the user’s view. To ensure its focus is available, the window will remain top most compared to other apps on the screen.

3.2 Hardware Interfaces

The websites will be accessible via mobile and desktop devices with varying screen sizes. This must be taken into consideration for layout and design factors.

3.3 Software Interfaces

The website and application will both share a connection to a server-side database via an API. This API will be public facing allowing direct interaction with it. All data stored in the database will only be visible via the API reducing the attack surface.

These communications must be over an encrypted connection via HTTPS.

3.4 Communications Interfaces

Communications via the API and website must be over an encrypted connection via HTTPS.

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 Website Signup

4.1.1 Description and Priority

A new or existing user can create a new account using their given email address, username and password. This is a high priority feature that is required for any user to become an active member.

4.1.2 Stimulus/Response Sequences

The user will first access the website via search engine or url. They will click on the signup button on the home page and fill in the required form. Upon completion and successful account creation, the user will be forwarded to the login screen. Here they can use their new account details to perform a login. Should the validation of user details fail upon signup, the form will display an appropriate error.

4.1.3 Functional Requirements

Link from homepage to the signup page

Form needed on signup page

Submit form interacts with API

API checks if user exists already, if not, saves data to database and redirects user to login

4.2 Website Login

4.1.1 Description and Priority

An existing user can login to the web application and visit the profile and leader board pages, given that the correct user details are supplied in the login form.

This is a high priority feature that is required for any user to log into their account.

4.1.2 Stimulus/Response Sequences

The user will first access the website via search engine or url. They will click on the login button on the home page, from here they will supply their login details and if correct they will be redirected to their profile page.

4.1.3 Functional Requirements

Link from homepage to the login page

Form needed on login page

Submit form interacts with API

API checks if user exists and if details are correct, if so the user is redirected to profile page.

4.3 Application Timer Start

4.1.1 Description and Priority

An existing user can login to the desktop application and click the start button to begin tracking a unit with Quark.

This is a high priority feature that is required for any user to log units with Quark.

4.1.2 Stimulus/Response Sequences

The user will open the java file downloaded from our webpage, log in or sign up and log in with their details through the application and click the start button to begin tracking a unit.

4.1.3 Functional Requirements

Open application jar file

Log in or sign up and log in with their details

Click the start button to begin tracking units

4.4 Website Leaderboard Display

4.1.1 Description and Priority

An existing user can login to the web application and visit the leaderboard page which displays the top 100 users.

This is a medium priority feature that we would like to implement to add extra functionality to Quark.

4.1.2 Stimulus/Response Sequences

The user will first access the website via search engine or URL. They will click on the login button on the home page, from here they will supply their login details and if correct they will be redirected to their profile page. From here they can click the leaderboard link, and this will take them to the leaderboard which shows the top 100 ranking users.

4.1.3 Functional Requirements

Link from homepage to the login page

Form needed on login page

Submit form interacts with API

API checks if user exists and if details are correct, if so, the user is redirected to profile page.

Link from profile page to leaderboard page

Leaderboard page displays top 100 users

4.5 Website Leaderboard Filters

4.1.1 Description and Priority

An existing user can sort the leaderboard by top users this day, week, month, and year.

This feature is of low priority. We would like to add it, however it is not required.

4.1.2 Stimulus/Response Sequences

Once the user is logged in, they navigate to the leaderboard page. When viewing the leaderboard, the users will be able to press tabs at the top of the page to filter by day, week, month, and year.

4.1.3 Functional Requirements

Link from homepage to the login page

Form needed on login page

Submit form interacts with API

API checks if user exists and if details are correct, if so, the user is redirected to profile page.

From here the user navigates to the leaderboard page

Once on the leaderboard page the user can filter the view of the leaderboard

4.6 Website View Profile Graph

4.1.1 Description and Priority

An existing user can login to the web application and visit the profile page, on this page the user can see the graph of their commits over the last two weeks.

This feature is required for our profile page to be functional.

4.1.2 Stimulus/Response Sequences

The user will first access the website via search engine or url. They will click on the login button on the home page, from here they will supply their login details and if correct they will be redirected to their profile page. On this page they can scroll down and will be presented with their units from the last 2 weeks plotted on the graph.

4.1.3 Functional Requirements

Link from homepage to the login page

Form needed on login page

Submit form interacts with API

API checks if user exists and if details are correct, if so, the user is redirected to profile page.

On profile page loading, the JS interacts with the API to collect the user data for the last two weeks, this then gets plotted on the graph for the user to see.

4.7 Application Pausing

4.1.1 Description and Priority

In the case of some unintended event occurring, the user might want to temporarily pause the event so that they could continue to study later in time. Although helpful, it doesn’t necessarily prevent the main purpose of what the pomodoro technique does.

This can be an extra feature and isn’t required for the application

to fully function.

4.1.2 Stimulus/Response Sequences

The user will open the jar file and log in. They will click start to begin the timer; the user can click pause to temporarily stop the timer where it is at.

4.1.3 Functional Requirements

Open the jar file.

Login with details.

Click start timer to begin timer.

Click on pause.

4.8 Application Stop and Restart

4.1.1 Description and Priority

Should the user encounter any errors, the user can stop the timer to restart the timer. This shouldn't prevent what the main timer does so it is not as important as the timer counting down for the technique to be effective.

This isn't required but can be a useful extra feature.

4.1.2 Stimulus/Response Sequences

The user will click on the jar file then log into the system. Click on the start timer to begin the timer, then click on stop to stop the timer and a 25-minute timer will be shown waiting to begin counting down.

4.1.3 Functional Requirements

Open Jar file.

Login with details.

Click start button to begin timer.

Click on stop.

4.9 Application Break Timer

4.1.1 Description and Priority

When the 25 minutes study time has finished, the app should then start a 5-minute timer to let the user know that they should take a quick break. This helps maintain the users mental state by allowing them to stretch and feel refreshed.

This is a required to allow users to properly rest.

4.1.2 Stimulus/Response Sequences

The user will log in to the app. User will click on start to begin the timer. The timer will then start a 25-minute timer then a 5-minute timer.

4.1.3 Functional Requirements

Open application jar file.

Login with details.

Click start button to begin timer.

25 Minute timer ends.

5 Minute break time starts.

4.10 Application Restarting Timer

4.1.1 Description and Priority

When the 5 minutes break time ends, it should restart the 25-minute timer again to restart the pomodoro unit of time.

This is a must have feature so that we can quickly restart the

timer.

4.1.2 Stimulus/Response Sequences

The user will log in to the app. User will click on start to begin the timer. The timer will then start a 25-minute timer then a 5-minute timer. Afterwards, it will send a unit and a 25-minute timer begins again.

4.1.3 Functional Requirements

Open application jar file.

Login with details.

Click start button to begin timer.

25 Minute timer ends.

5 Minute break timer ends.

25 Minute timer starts again.

4.11 Application Sending Units to Database

4.1.1 Description and Priority

After each complete pomodoro time of 25 minutes study and 5 minutes break, the system needs to be able to send the unit to the API so that it can be added to the database connected to the users account to keep track of their total units. This is very important as it is the main function of our app.

This feature is required so that data can be tracked

4.1.2 Stimulus/Response Sequences

When beginning the countdown, it will countdown from 25 minutes, then 5 minutes. After the 5 minute break is over, a unit will be sent to an API which will add a unit to an existing database.

4.1.3 Functional Requirements

Open application jar file.

Login with details.

Click start button to begin timer.

25 Minute timer ends.

5 Minute break timer ends.

Unit gets sent to the API.

API will add unit to database.

4.12 Application Access

4.1.1 Description and Priority

For the user to use the app, the app needs to be downloaded so that the user can join other members to share stats and for users to manage their time effectively. The download button needs to be able to download a jar file when clicked.

This feature is required so that users can use the app.

4.1.2 Stimulus/Response Sequences

The user will first access the website via search engine or URL. They will click on the download button on the top right. Since an account is required, it will direct you to the login page where you can signup if you don't have an account made. Once you log in, clicking on the download button will direct you to a page showing downloads for 3 different versions. Users can then click on the windows button to download the jar file.

4.1.3 Functional Requirements

Link from homepage to downloads.

Redirects to login page if not logged in.

If logged in, will go to downloads page.

3 Clickable links are shown.