HACK YOUR FUTURE Denmark

ATOMIC LEARNING SYSTEMS

Final Project Guide
Atomic Learning

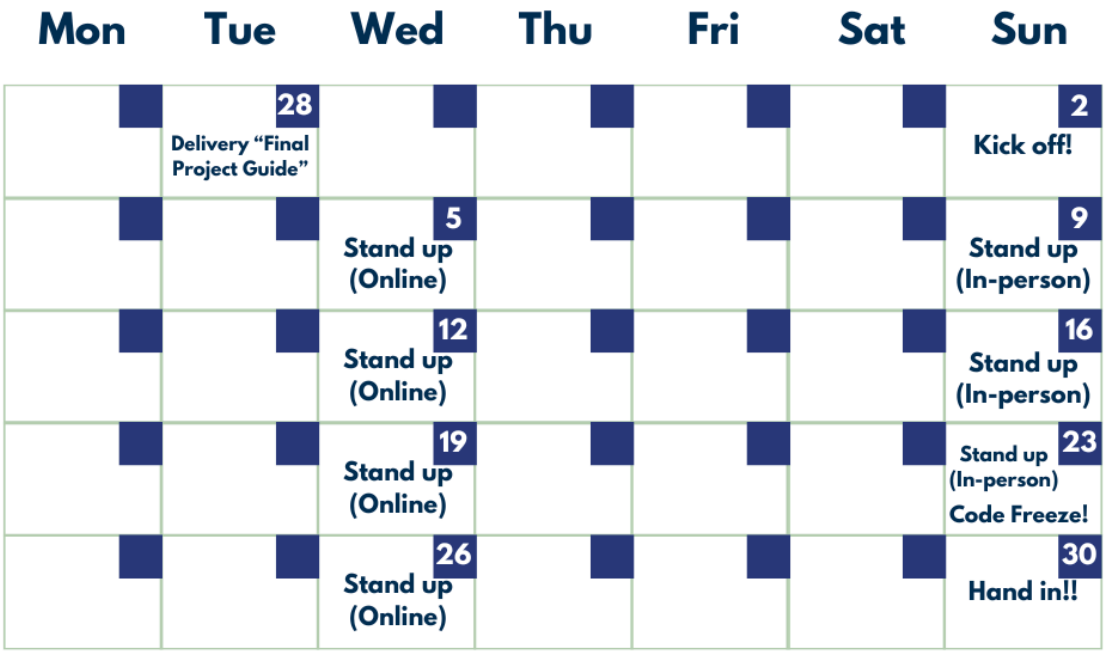
Overview

This document is for mentors and trainees to understand the final project. It explains the structure, roles, and tools. This helps everyone know their responsibilities and the project's goals.

Your team should use these guidelines to plan every step of the process. Please review the document thoroughly, and use **Kick-off Sunday** to align yourself with the team and mentors. Please be ready to share your input on that day.

Project Timeline

The total duration of the project is 5 Sunday sessions, distributed along 4 weeks, **starting on June 2nd and finishing by June 30th.**



Important things to consider for ***each date***:

•**Kick-off Date (June 2nd):** On this day, you will meet the **clients**, and you will plan how you will be working during the first week: How your team is going to be divided, and what are the tasks that you should focus on. Define what "done" means and what "done done" means for your team.

•**For each Stand-Up session (online): Everyone in the group needs to assist**. It will be a 30 minute meeting in which there will also be mentors. The focus will be on trainees, which will have to answer 3 main questions:
-What did you do from our last stand up?
-Are you facing any challenges? If yes, how do you plan to overcome them?
-What are you going to do until the next stand up?

•**For each Stand-Up session (In-person)**: Sundays will start with a discussion of 30 minutes, in which the team must answer the 3 main questions mentioned above. After that, everyone will focus on coding, and helping each other in case that someone is facing an issue. Mentors will be there to assist, if needed. Last 30 minutes of the session will be about **planning next-week work,** and doing some retrospective to see what worked, or didn't during the week.

In between those dates, team members should **always communicate with each other, since that is part of this final challenge!** To come to the stand up session on Wednesday saying "I couldn't do my task because I did not understand what I should do" is a no-go 🙂.

•**CodeFreeze (June 23rd):** This means that you will no longer make huge modifications in your code, but you will **focus on fixing bugs.**

Hand in (June 30th): You will do a presentation of the final product to the clients, that should be already deployed and properly working.

Working Structure

Your team needs to follow Agile methods, meaning that each trainee will code, and also handle their assigned role (we will explain about the different roles later in this document).

Your Amazing Team


Sanaz


Niloufar


Anastasiia


Youssef


Angel


Ihor


Anjali

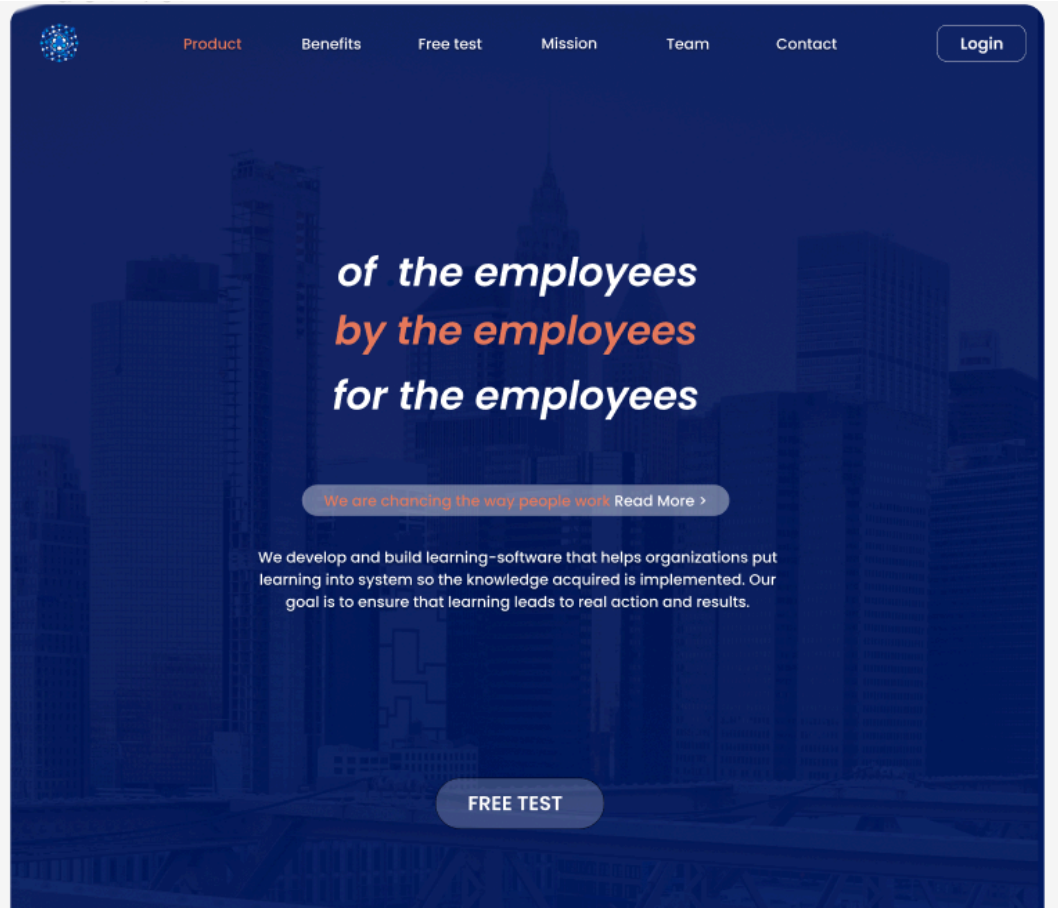
About Your Client: Atomic Learning

In their own words: "*We help organizations build an infrastructure that leads talent towards achievements. Our aim is to address the critical challenge of employee retention by offering choices and flexibility within the workplace. Our innovative platform seamlessly integrates learning with practical work, enabling individuals to not only discover their potential career paths but also to immediately apply their newly acquired knowledge.*"

You can read more about them in the company document that was sent together with this document.

What Does Your Client Need?

You will be developing their **landing page,** which should look like this:



You can check the entire design in the [figma prototype,] that you must use as a guide when building the project. **Do not modify anything in the prototype.**

Tools and Processes

slack Trello GitHub

We will use Slack for daily communication, Trello for task management and Github for version control.

Git Instructions: Managing Branches and Resolving Conflicts

Creating a Branch

1. Open your terminal.
2. Navigate to your project directory.
3. Create a new branch: git checkout -b development/name-of-person/feature-name

Switching Branches

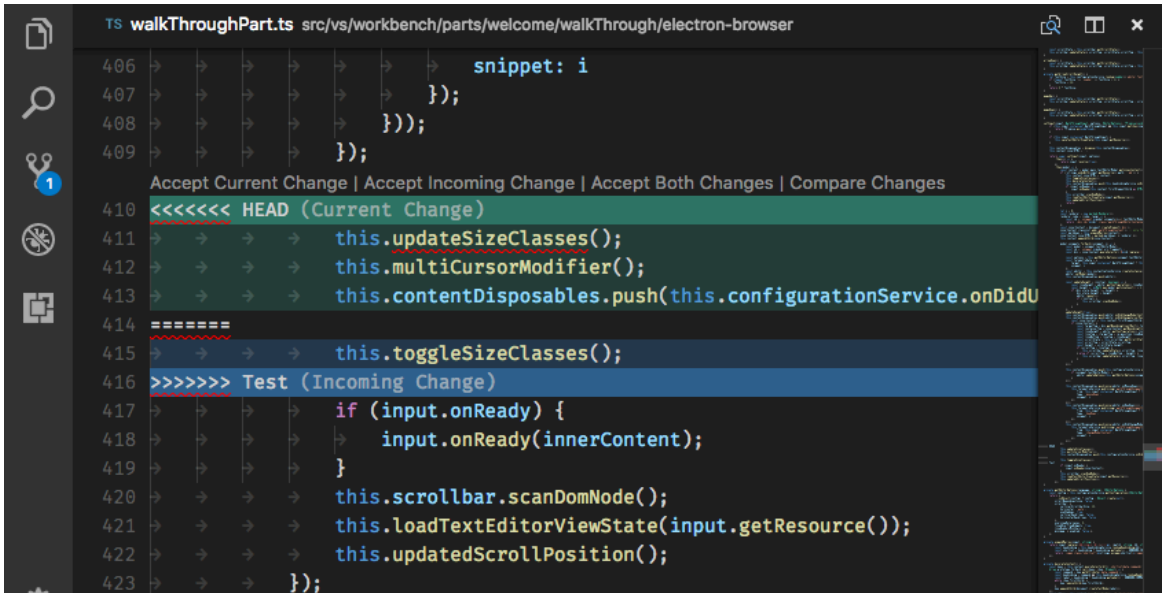
1. Make sure all your changes are committed: git commit -m "message".
2. Switch to another branch: git checkout branch-name.

Merging Branches

1. Switch to the branch you want to merge into (usually main): git checkout main.
2. Merge the other branch into main: git merge branch-name.

Resolving Conflicts

When a conflict occurs, Git will mark the conflict in the affected files.



Example of how a conflict looks like in visual studio code

1. Open the files and look for lines with conflict markers:
2. Edit the file to resolve the conflict. (You have several options. Either accept the current change, the incoming one, or accept both and then manually delete what you don't need. Keep in mind that it is always a good idea to communicate with the person who wrote the code that is generating a conflict with your code.)
3. Once resolved, add the file: git add file-name.
4. Commit the changes: git commit -m "Resolved conflict".

Project Roles

Your team needs to assign roles to ensure coverage of all project aspects. These roles add responsibilities, but do not take away from the primary role of coding.

Roles and Responsibilities

- **Deployment Specialist:** Manages the deployment process, meaning that it should check that the production is deployed and running.

- **Git Specialist:** Handles version control and branch management. This means that this person will be responsible for checking up on PRs to make sure that they're merged.

- **Design Leader:** Oversees the design aspects of the project. Ensures that the team is following the figma prototype properly.

- **Testing Lead (QA):** Ensures code quality through testing. This means that this person will be responsible for testing the main branch once in a while, to make sure that everything is working.

- **Scrum Master:** Facilitates Scrum processes. Will be responsible for the communication inside the team, making sure that everyone is aligned towards the same objective.

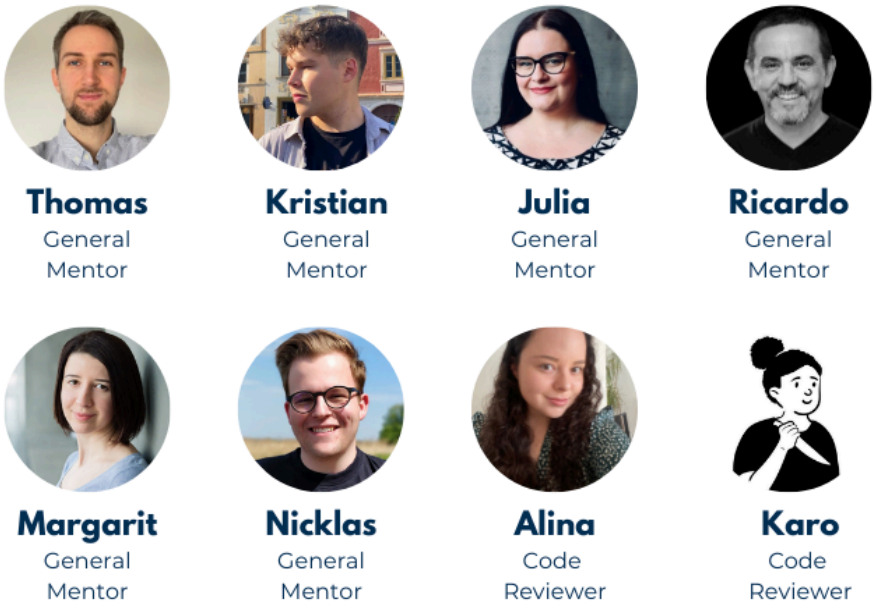
- **Status Reporter:** Updates stakeholders on project progress . This person ensures that the communication with the client is successful. For example, if the team needs to communicate something to the client in slack, this person will be responsible to send the message.

- **Dependency Manager:** Manages project dependencies. Meaning that is responsible for the node packages, giving the go ahead when other members of the team want to add a new package.

There would generally only be one role of each per team. This is something to discuss internally within your team.

Disclaimer: You have these responsibilities, but if at any point you are uncertain about what to do, or how to, just bring it to discussion with mentors on a Sunday session.

Mentors That Will Support You



Roles' meaning:

- **General Mentors:** Will join all (or some) Sunday sessions, and will be available via Slack to answer questions and offer support. They are responsible for guiding and sparring with trainees, but will not do the tasks for them. They might help with code reviews if needed.

- **Code Reviewers:** Dedicated to reviewing the submitted code weekly, and providing feedback. Also available in Slack during the week. This is a remote role.

Documentation: How should you document your code?

All tech companies document their code in various ways, and we have our own approach to ensure clarity and maintainability. Here is how we will be doing it:

In-Code Comments

When coding, clear and helpful in-code comments are vital. Make sure your comments are **easy to understand,** avoiding unnecessary technical language. Keep comments brief but informative, pointing out important details and edge cases. Remember to **update comments** as the code evolves to maintain clarity and usefulness.

Here are is a good example of how to write your comments:

```
// Convert temperature from Celsius to Fahrenheit
function celsiusToFahrenheit(celsius) {
    return (celsius * 9/5) + 32;
}
```

Example of a good code comment.

And an example of what you should NOT do.

```
/**
* For the brave souls who get this far: You are the chosen ones,
* the valiant knights of programming who toil away, without rest,
* fixing our most awful code. To you, true saviors, kings of men,
* I say this: never gonna give you up, never gonna let you down,
* never gonna run around and desert you. Never gonna make you cry,
* never gonna say goodbye. Never gonna tell a lie and hurt you.
*/
```

Example of a bad code comment.

Commit messages

Effective commit messages are essential for project clarity and collaboration. Crafting easy-to-read, meaningful messages is crucial when developing a solution.

Here are some tips on how to do that:

- **Be Descriptive**: Clearly describe what changes were made and why.
- **Keep it Concise:** Limit to 50-72 characters for the summary.
- **Separate Subject and Body**: Use a short subject, followed by a blank line, then a detailed body if needed.
- **Use Keywords for Linking:** Reference issues or tasks, e.g., "Fixes -nameofthetaskintrello".

```
Add login and signup endpoints

- Updated database to support user roles
- Added login and signup endpoints

fixes -nameofthetrellocard
```

Example of a commit message.

Your Team Tasks Before and During Sunday

0. **Name creation:** Come up with an **original name for your team**.
1. **Role Selection:** Individually think about the roles you are interested in for the project. Start discussing them with your team in Slack, and take the final decision on Sunday.
2. **Client Interaction:** Prepare questions for the clients to understand the project requirements better. The companies will be presenting themselves on Sunday.
3. **Wireframes:** Review the provided figma prototype wireframe and think about how you will contribute to its creation.
4. **Backlog Formation:** Make an online meeting with your team, and create a Trello board, with its different columns (backlog, to do, done, etc), and start thinking about the backlog or tasks needed to complete the project (like Katrine did in her example). You will be finishing this board on Sunday.