
# Checkpoints

## Checkpoint 1

**Presented to Paul:**

- Project plan (Emphasis on Phasing)

**Key Takeaways:**

- The current structure of the phasing section, based on design principles (like Empathize and Define), is counterproductive since these were already completed in the previous group project.

- Any path we explore—such as testing and researching React for development—should be documented, including any challenges and failures.

- The phasing should be reorganized into a time-based format, such as a weekly or daily schedule, with clear goals and deliverables.

## Checkpoint 2

**Presented to Dirk:**

- Updated Figma wireframes

- Questions about GitLab workflow (branching, etc.)

- Ideas for incorporating different programming languages into the project

**Key Takeaways:**

- The updated wireframes were well-received. The pages are now coherently linked, unlike the previous version.

- Hyperlink positions should be prioritized based on expected user behavior.

- The "Forgot password?" link should be grouped with the password input field to follow the Gestalt principle of association.

- Sending users an automatically generated password without giving them the option to change it introduces security risks. A safer reset method should be implemented. A user verification would not be possible for this project of our level yet so it will not implemented.

**-** Adopting a new programming language at this stage may lead to a lack of visual polish due to time constraints. HTML, CSS, and JavaScript remain more manageable for maintaining visual quality.

- Dirk recommended meeting with the client to clarify whether functionality or visual design is the higher priority. The MoSCoW list should be updated and deliverables clearly defined.

- Dirk visually demonstrated GitLab branching. We now understand how to use feature branches and merge into main once tested and stable. This helps avoiding loss of code

# Checkpoint 3

**Presented:**

- Trello workspace

- Updated phasing in the project plan

**Key Takeaways:**

- No major changes needed to the overall timeline.

- Focus should now shift to maintaining and updating Trello with daily, actionable tasks.

- In a professional setting, the **project owner** typically creates and assigns tasks, while **developers select** which ones to take on.

**-** Each Trello task should be **labeled with its corresponding Git branch name** and **referenced in commit messages** to maintain clear traceability between code and task.

- Establish a routine for reviewing and updating Trello tasks daily or during stand-ups.

- Consider assigning a team member to oversee task clarity and consistency.

---

### Presented to Paul

- **Trello Work Board**
- **Two BELCO Websites deployed by Vercel (Belco Education and Belco Alliance)**
- **Git Repository**

---

### 🔑 Key Takeaway

Although we worked on separate websites, we used this structure as an opportunity to **learn Next.js collaboratively**, helping each other debug and improve our code during work sessions at university. This showed that collaboration can still be meaningful even when working on parallel components, as long as **knowledge-sharing and support** remain central.

- Assign **relevant and specific titles** to each page (e.g., "BELCO Alliance Programs – International Exchange"). Helps improve **Google search visibility** and **user navigation**.
- Use **branches** effectively for collaborative development:
- Create separate branches for new features or individual contributions.

- Merge to `main` only after peer review and testing.