Meeting Notes

Project: Building a Programming IDE for a Computing Class
Date: 6th February, 2025
Attendees: Godwin Idowu, Yaw Oppong-Krampah, Wisdom Akanwe, Braeden Singleton, Calvin Deka, Connor Moss, Nick Nelson

Agenda:

1. **Overview of the IDE Requirements**
    - Ability to edit and execute code.
    - User authentication (Login/Signup).
    - Project creation, saving, and execution.
    - Error feedback and debugging messages.
2. **Collaboration Features**
    - Screen sharing and real-time editing capabilities.
3. **GitHub Integration**
    - Saving and loading projects directly from GitHub.
    - Cloning repositories and managing version control.
4. **Programming Challenges Module**
    - Assigning coding problems.
    - Testing and submission automation.
5. **Selection of IDE: Eclipse Che**
    - Web-based IDE for cloud-based development.
    - Supports multiple programming languages.
    - Containerized workspaces for consistency.
    - Collaboration and pair-programming features.
    - Built-in terminal and debugger.
    - Seamless GitHub and GitLab integration.
6. **Team Responsibilities**
    - **Common Tasks for All Teams:**
        1. IDE Setup.
        2. User Authentication (Login/Signup).
        3. Project Creation and Execution.

4. GitHub Integration for Saving and Loading Projects.
- **Team 1:**
    1. Code Editing.
    2. Collaboration Features (Screen Sharing, Real-Time Editing).
- **Team 2:**
    1. GitHub Integration (Saving & Loading Projects).
    2. Automated Testing and AI-based challenge assignment.

7. **Installation and Configuration of Eclipse Che**

    - **GitHub OAuth Setup:**

        1. Navigate to GitHub Developer Settings > OAuth Apps.
        2. Create a new OAuth application with the following details:
            - **Application Name:** Eclipse Che
            - **Homepage URL:** `https://<your-che-server-url>`
            - **Authorization Callback URL:** `https://<your-che-server-url>/api/oauth/callback`
        3. Register the application and copy the Client ID and Client Secret.
    - **Configuring GitHub in Eclipse Che:**

```
Unset
export CHE_OAUTH_GITHUB_CLIENTID=<your_client_id>

export CHE_OAUTH_GITHUB_CLIENTSECRET=<your_client_secret>
```

    - Restart Che for the changes to take effect.

    - **Creating and Running Workspaces:**

        1. Open Eclipse Che (`http://localhost:8080`).
        2. Click **Create a New Workspace**.
        3. Select a template (Node.js, Python, Java, etc.).
        4. Click **Start Workspace**.
    - **Cloning a GitHub Repository:**

        1. Navigate to Git > Clone Repository.

2. Enter the GitHub repository URL:

```
Unset
https://github.com/user/repo.git
```

3. Click **Clone** and start coding.

8. **Automating Builds with GitHub Actions**

   - Add a workflow file in `.github/workflows/ci.yml`:

```
Unset
name: Eclipse Che CI

on: [push, pull_request]

jobs:

  build:

    runs-on: ubuntu-latest

    steps:

      - name: Checkout code

        uses: actions/checkout@v2

      - name: Install Dependencies

        run: npm install

      - name: Run Tests

        run: npm test
```

   - This setup ensures automatic build and testing for every commit and pull request.

9. **Summary of Key Setup Steps**

Task	Command/Action
Install Eclipse Che (Docker)	`docker run --rm -it --name che -p 8080:8080 eclipse/che`
Install Eclipse Che (Kubernetes)	`kubectl apply -f https://raw.githubusercontent.com/eclipse/che/main/deploy/kubernetes/che.yaml`
Access Che Locally	`http://localhost:8080`
Expose Che on Kubernetes	`kubectl port-forward svc/che 8080:80 -n che`
Setup GitHub OAuth	Create OAuth App in GitHub Developer Settings
Clone GitHub Repo	Git > Clone Repository

To be worked on:

- **Team 1:** Implement collaboration and editing features.
- **Team 2:** Work on GitHub integration, auto-testing, and AI-based coding challenge module.
- **All Members:** Ensure IDE setup and GitHub authentication are functional.