Welcome to The Dependables!

Hello [New Developer],
Welcome to the **‘AltBot’** project. We're thrilled to have you on board and look forward to collaborating with you on our exciting projects. This onboarding guide is designed to help you quickly integrate into our team and get started with your development journey.

About Us

Team Moto: Empowering Excellence through Ownership

This motto reflects our commitment to the principle of ownership, showcasing our dedication to taking responsibility for our work, code, and projects. It aims to convey a sense of reliability and dependability, aligning with our team name “The Dependables”.

Our Team:

Meet your fellow team members:

- Shreyas Labhsetwar(Team Lead): Mumbaikar at <3 who loves gymming, hiking, swimming, traveling, and studying in my free time.
- Niraj Yagnik(Team Lead): I am an ambitious and passionate CS grad student who likes working out , music, football, and bragging about my AMC stubs a-list membership.
- Onkar Litake: 2nd year MS CS, GSR at UCSD Health. Passionate ML researcher. Love playing football & CSGO.
- Jay Jhaveri: Currently Software Developer. Aspiring Software Engineer
- Andrew Ghafari: Coder fueled by coffee, foodie adventurer fueled by music, and a die hard football fanatic.
- Xuanyu Chen: 1st-year MSCS@UCSD, ZJU 2023, Experience in Computer Vision.
- Jash Makhija: I love to code!. I am incredibly passionate about building new software products, by writing clean and scalable code.
- Eddie Qiu: A regular everyday normal student.

Feel free to reach out to any of us if you have questions or need assistance. We're here to support you!

Project AltBot:

AltBot is a JavaScript-based bot designed for the Fediverse, specifically Mastodon. It automatically generates Alt Text for images, enhancing accessibility for visually impaired users

and connecting them with the vibrant, decentralized world of the Fediverse. This innovative tool exemplifies our commitment to inclusive and advanced technology solutions.

The project employs the best software engineering practices recommended by David Farley in his book Modern Software Engineering. Our code uses continuous integration to ensure consistent code quality and early detection of conflicts or issues. The code is appropriately modularized, with concerns separated and exhibiting good cohesion. The project opted not to utilize Test-Driven Development to accommodate the brief development period and the team's inexperience with TDD.

We have also created an AltBot Chrome Extension using Manifest V3 specially for the website "Mastodon.social". Our extension adds new alt text to the images directly into the HTML DOM if the original creators didn't provide one.

Getting Started

Languages/Tools: Javascript, HTML, CSS, Python, Shell, Machine Learning

Software Installation:

- Clone our GitHub repository from here.
- Install MySQL client.
    - Create a database with name: AltBotDB → Command: CREATE DATABASE AltBotDB;
    - Create a table using: CREATE TABLE Images(image_id VARCHAR(255) PRIMARY KEY, image_url VARCHAR(255), flag integer, post_id VARCHAR(255),user_id VARCHAR(255),alt_text varchar(255));
- AltBotService (Go to the designated folder)
    - Install dependencies using command: npm install
- ALTTagMLService (Go to the designated folder)
    - Install dependencies using command: pip install -r requirements.txt
- AltBot_Chrome_Extension_Mastadon (Go to the designated folder)
    - On chrome://extensions/, enable the developer mode.
    - Click Load unpacked and select the current folder.
    - Enable the extension and navigate to the Mastodon.social website. Upon doing so, you can check if the extension is installed properly by clicking on the extension's logo. A popup window will appear, indicating that our bot is currently running.
- Setting up .env file
    - We'll give you a set of credentials for Mastodon to use for the .env file in AltBot/AltBotService.
    - You'll also need to put your MySQL credentials there.
    - There is a template.env file for your understanding.
    - Make sure to add the .env file to .gitignore.

Running the code

- Go to ‘AltBot/ALTTagMLService/’ and run the ML server using: python app.py
- Go to ‘AltBot/AltBotService/’ and run the AltBot using: ./main.sh
- Incase of error related to permission, make sure to give proper permission by executing: chmod +x main.sh
- To run tests related to AltBotService:
    - Go to ‘AltBotService/
    - Run: npm ci
    - Run: npm test
- To run chrome extension, open Mastodon app and it’ll automatically generate ALT tags for the images at run time.

Version Control

We use Git as our version control system to manage and track changes to our codebase. Git allows us to collaborate effectively, track the history of our code, and manage different versions of our software. Our version control strategy revolves around feature branching. Here's an outline of the process:

Feature Branching:
- For each new feature or task, create a new branch.
- Naming convention for branches: feature/your-feature-name.

Commit Messages:
- Use meaningful and concise commit messages.
- Clearly communicate the purpose of each commit.

Pull Requests (PRs):
- After completing the feature or task, initiate a pull request.
- Naming convention for PRs: Feature: Your Feature Name.
- Include a detailed description of the changes made.

Agile Merging Practice:
- We follow an agile merging practice.
- Another team member will review your pull request.
- The reviewer will ensure code quality, adherence to coding standards, and overall functionality.
- After approval, the reviewer will merge the changes.

Code Quality Checks

JavaScript Code:

- We use unit tests to validate the functionality of our JavaScript code. This ensures that new changes do not introduce unexpected issues and helps maintain the stability of our application.
- Testing Framework: Jest
- Static Code Analysis:
    - We utilize omilint for linting JavaScript code.
    - Linting is performed automatically during code pushes to the GitHub repository.

Python Code:

- For Python code, we use flake8 for static code analysis to enforce coding standards and maintain consistency.

Collaboration Tools

GitHub Project Board: We utilize GitHub Project Boards as our primary task manager to streamline project management and keep our development process organized. The board is divided into four main sections:

- No Status: Tasks that have not been assigned a status yet are placed here.
- To-Do: Tasks that need to be addressed and are ready for assignment.
- In Progress: Tasks that team members are actively working on.
- Done: Completed tasks are moved to this section for visibility.

Team Communication - Slack: We leverage Slack as our primary team communication platform. Slack provides real-time messaging, collaboration, and file sharing, facilitating quick and effective communication among team members. Join relevant channels, stay updated, and feel free to ask questions or share updates through Slack.

Scheduling - When2Meet: When2Meet is our tool of choice for scheduling meetings and coordinating availability. It simplifies the process of finding suitable meeting times by allowing team members to indicate their availability. Use When2Meet to schedule meetings that accommodate the majority of team members.

Online Meetings - Zoom: For virtual meetings, we use Zoom, a reliable and feature-rich video conferencing platform. Zoom provides a seamless experience for team meetings, discussions, and presentations. Attendees can join meetings through the provided Zoom links.

Documentation Practices

Project Logistics Documentation:

We heavily rely on README files to communicate project logistics, updates, and meeting outcomes. README files serve as a centralized source of information, ensuring that everyone on the team is on the same page regarding project progress and recent developments.

Meeting Updates:

- README files include updates on how meetings went, decisions made, and action items assigned.
- They serve as a historical record of discussions and decisions.

Progress Tracking:

- README files may contain information about the overall project status, milestones achieved, and upcoming goals.
- Team members can refer to README files to understand the project's trajectory.

Technical Understanding Documentation:

JSDocs for JavaScript Code: For our JavaScript codebase, we use JSDocs to provide complete and extensive documentation. JSDocs enhance the understanding of the codebase by providing detailed information about functions, parameters, and return types.

Function Documentation:

- JSDocs are added above functions to describe their purpose, parameters, and expected return values.
- This allows developers to quickly understand how each function works.

Code Annotations:

- Comments and annotations within the code are used to provide additional context where necessary.
- Developers can gain insights into the logic and reasoning behind specific code segments.

Python Code Documentation: Our Python code is comprehensively documented to facilitate a clear understanding for both current and future team members.

Inline Comments:

- Comments within the code explain complex logic, algorithms, or any non-trivial operations.
- These comments aid in understanding the code's functionality.

Docstrings:

- We use docstrings to provide detailed explanations for modules, classes, functions, and methods.
- Docstrings serve as self-contained documentation that can be easily accessed.

README Files for Project Components: Each project component is accompanied by a dedicated README file that contains detailed information on how to execute, replicate, and understand the code.

- **Installation Instructions:** Step-by-step instructions on how to install dependencies and set up the development environment.
- **Database Setup:** Intricate details on how to establish the required database, including schema creation and credential setup.
- **Execution Guidelines:** Guidance on running the code, including any specific commands or configurations.
- **Troubleshooting Tips:** Information on common issues and troubleshooting steps.