Project Chrono

by Chronos

<u>Team Members:</u> Aarti Arasada, Ajaydeep Singh, Steve Arina Amatya, Lakshya Aggarwal, Ramita Dhamrongsirivadh, Thomas Ballarino, Matthew, Chuchen Li
<u>Team Manager</u>: Akash Buchi

1. Project Description

Chronos is a robust, user-friendly timekeeping software specifically designed for the modern work environment. Its purpose is to enable companies to accurately track and manage employee work hours, thus streamlining payroll processes and promoting productivity.
This application is primarily used by two types of users: the employees who clock in and out using the software, and the administrators who manage and monitor the time records.
Non-functional constraints for the application include ensuring high performance to manage large amounts of data seamlessly, and robust security measures to safeguard sensitive employee information.
The application mainly caters to two user types:

1. **Employees**: They can use the application to clock in and out, thus recording their working hours.
2. **Managers**: They can oversee employee working hours, generate reports, and manage employee data.

Non-functional requirements include high performance, ensuring quick response times even with a large number of concurrent users, and robust security measures to protect sensitive employee and company data.

2. Third Party Software (Backend)

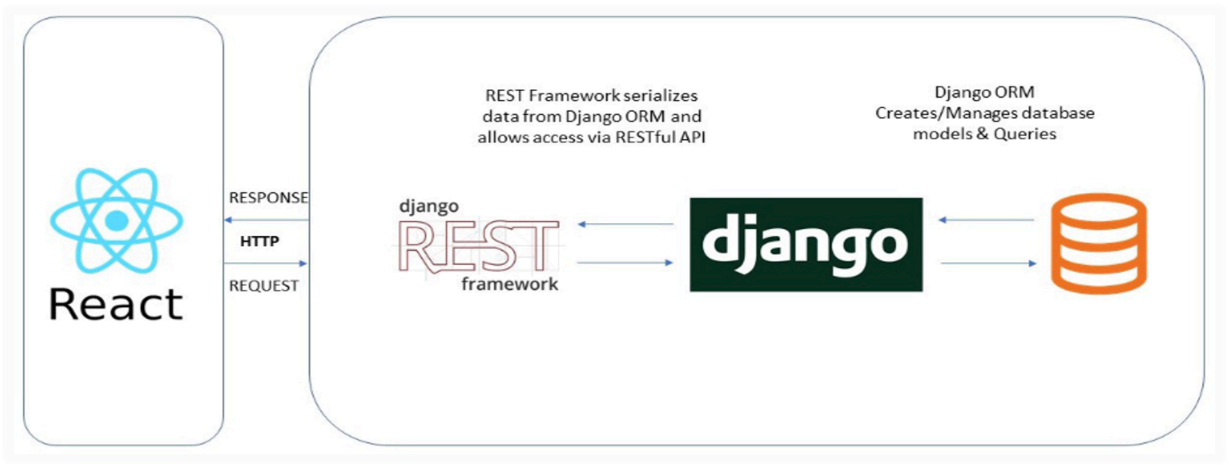
Software Title	Description	Purpose
Django	A high-level Python web framework for rapid development.	Used to create the backend and API functionality.
Swagger API	An open-source API documentation and testing tool.	Used for API documentation and testing purposes.

PostgreSQL	An open-source relational database management system.	Database used to store application data.
Git	A distributed version control system for tracking code changes.	Used for source code management.
Docker	A platform for automating deployment and management of apps.	Used for containerization and deployment.

Third Party Software (Front End)

Software Title	**Description**	**Purpose**
React	A JavaScript library for building user interfaces	Used to build the frontend components of the application
HTML/CSS	Standard web technologies for structuring and styling web content	Used in creating and designing the application's pages
Jest	JavaScript testing framework maintained by Facebook	Used for creating unit tests for the application
React Router	Standard library for routing in React	Used for managing navigation and URL paths
Axios	Promise-based HTTP client for the browser and Node.js	Used for making HTTP requests to the backend server

3. Architectural Diagram


React
RESPONSE
HTTP
REQUEST
REST Framework serializes data from Django ORM and allows access via RESTful API
django REST framework
django
Django ORM Creates/Manages database models & Queries

4. **Difficulties Encountered**
    1. Integrating third-party libraries and frameworks.
    2. Ensuring data security and privacy.
    3. Optimizing performance for a scalable solution.
    4. Ensuring consistent behavior and appearance across different browsers and platforms.
    5. Managing state for larger components proved challenging.

5. **What Worked Well**
    - Use of Django for rapid backend development.
    - Containerization with Docker for easy deployment.
    - Modular design for easy maintenance and extensibility.
    - React facilitated creating reusable components, leading to faster development and less code.
    - The use of CSS helped in creating a consistent and appealing user interface.

6. **Technical Issues**
    - Initial learning curve with Django and Swagger API.
    - Minor difficulties configuring PostgreSQL for secure access.
    - Challenges faced in load testing and optimizing the application for performance.
    - Initial difficulty in managing async operations in React. However, mastering it led to a more seamless user experience.

7. Testing

We employed a combination of unit, integration, and user testing during the development process. We used the Django Test Framework for unit and integration tests to ensure the proper functioning of individual components and their interactions. Additionally, we conducted user tests to evaluate the usability and overall user experience of the application. For Frontend testing: Unit tests were extensively implemented throughout the development process using Jest, a JavaScript testing framework. These tests ensured that individual components and functions of the application worked as expected. Additionally, integration tests were conducted to ensure the correct interaction between different components of the system.

8. Installation and Running Notes

Please refer to the README.md file in the repository for detailed instructions on installing, running, and testing the application:
https://github.com/nguyenvothuan/chrono - backend
https://github.com/ArinaAmatya/Chronos-Frontend - frontend

To run this project locally, you need to have Node.js and npm installed on your machine. Follow the steps below:

- Clone the repository:bash

Copy code {git clone https://github.com/ArinaAmatya/Chronos-Frontend.git}

- Change into the directory:bash

Copy code {cd Chronos-Frontend}

- Install the dependencies:

Copy code {npm install}

- Run the application:sql

Copy code {npm start}

To run the test suite, use the following command:

- npm test

Make sure the backend server is running and reachable from your system, as the frontend application will need to make requests to it.

Note: This is a minimum viable product (MVP) prototype and not a complete system.

9. Future Improvements

As an MVP prototype, there are several improvements that can be made to the Chrono Employee Time Keeping System in future iterations. Some of these enhancements include:

- Implementing additional features such as automated timesheet generation, employee scheduling, and time-off requests.
- Enhancing the user interface and user experience by developing a responsive web front-end or mobile application.
- Incorporating advanced data analysis and visualization tools to provide more in-depth insights for administrators.
- Expanding the system's integration capabilities to support seamless integration with other HR management and payroll systems.
- Implementing advanced features like real-time sync of clock in/out data.
- Enhancing the user interface with better visualizations of time data.
- Improving mobile responsiveness for better accessibility across various devices.

10. Support and Maintenance

To ensure the long-term success of the Chrono Employee Time Keeping System, we will provide ongoing support and maintenance services, including:

- Regular software updates to address bugs, security vulnerabilities, and performance optimizations.

- User training and documentation to help users make the most of the system's features.
- Technical support to assist with any issues encountered during installation, configuration, or usage of the software.

11. Conclusion

The Chrono Employee Time Keeping System provides an efficient and scalable solution for tracking, managing, and reporting on employee work hours. By leveraging powerful third-party software like Django and Swagger API, we have created a secure and performant backend that can be easily deployed using containerization technologies like Docker. Our development process included rigorous testing to ensure a high-quality product, and we have identified future improvements to enhance the system further. With ongoing support and maintenance, we are confident that the Chrono Employee Time Keeping System will be a valuable tool for companies looking to streamline their employee time management processes. The frontend of the Chronos timekeeping software has been built with a focus on user experience and scalability. It's designed in a way that makes future enhancements and additions to the system quite seamless. With continuous improvements and feature additions, we aim to make Chronos a go-to solution for companies looking to manage their employees' time effectively.

12. User Documentation

To ensure a smooth onboarding experience for both employees and administrators, we will provide comprehensive user documentation that covers the following topics:

- Getting started: A step-by-step guide for initial setup and configuration of the Chrono Employee Time Keeping System.
- Employee usage: Instructions for employees on how to clock in, clock out, and view their time records.
- Administrator usage: Guidance for administrators on how to manage employees, monitor work hours, and generate reports.
- Troubleshooting: A list of common issues and solutions to help users resolve any problems they may encounter while using the system.

13. Challenges and Solutions

- One significant challenge faced during the project was handling complex state
- management. The issue was resolved by implementing the Context API and hooks, which made state management more streamlined and manageable.
- The initial user interface design was not very intuitive, which was identified during user testing. The interface was then revised and enhanced based on the user feedback received.

14. License and Terms of Use

The Chrono Employee Timekeeping System is licensed under the [Insert License Name] open-source license. Users are free to modify, distribute, and use the software in accordance with the terms and conditions specified in the license agreement. By using the software, users agree to comply with all applicable laws and regulations and acknowledge that the software is provided "as is' ' without warranty of any kind.

15.Acknowledgements

We would like to express our gratitude to the following individuals and organizations for their invaluable support and contributions to the development of the Chrono Employee Time Keeping System:

- The Django Project and its contributors are responsible for creating a powerful and versatile web framework.
- The Swagger API team for providing an excellent API documentation and testing tool.
- The PostgreSQL community for developing a robust and reliable database management system.