# The Red Lobsters

CrabbyTasks

# TABLE OF CONTENTS


**TABLE OF CONTENTS** ..... 2
**Project Overview** ..... 10
PROJECT NAME ..... 10
PROJECT MANAGERS ..... 10
PROJECT SPONSORS ..... 10
BUSINESS CASE ..... 10
PROBLEMS ..... 10
GOALS ..... 11
DELIVERABLES ..... 11
RISKS & OBSTACLES ..... 12
**BUSINESS PLAN** ..... 12
EXECUTIVE SUMMARY ..... 12
COMPANY DESCRIPTION ..... 12
COMPANY NAME ..... 13
BUSINESS STRUCTURE ..... 13
OWNERSHIP ..... 13
LOCATION ..... 13
COMPANY HISTORY ..... 13
MISSION STATEMENT ..... 13
PRODUCTS/SERVICES & TARGET MARKET ..... 14
BRAND VALUES AND GOALS ..... 14
VISION STATEMENT ..... 14
MARKET ANALYSIS ..... 14
TARGET MARKET ..... 14
POTENTIAL MARKET ..... 15
SWOT ANALYSIS ..... 15
COMPETITIVE ANALYSIS ..... 16
COMPETITOR DATA COLLECTION PLAN ..... 17
MARKETING EXPENSE STRATEGY CHART: ..... 17
OPERATIONAL PLAN ..... 17
MANAGEMENT AND ORGANIZATION ..... 17
ORGANIZATION CHART ..... 18
STARTUP EXPENSES & CAPITALIZATION ..... 18
REFINING THE PLAN ..... 18
ORGANIZATION & MANAGEMENT TEAM ..... 18
BOARD OF DIRECTORS ..... 18
SUPPORTING PROFESSIONALS ..... 20
SERVICE & PRODUCT LINE ..... 20
CURRENT PRODUCT ..... 20
FUTURE PRODUCT ..... 20

MARKETING & SALES..... 21
PRICE..... 21
PRODUCT..... 22
PROMOTION..... 22
PLACE..... 23
SALES FORECAST..... 23
OPERATION PLAN..... 24
SUPPLIERS..... 24
PRODUCTION..... 24
FACILITIES..... 24
LEGAL EQUIPMENT & ENVIRONMENT..... 24
SHIPPING & FULFILLMENT..... 24
INVENTORY..... 25
FINANCIAL PROJECTIONS & FUNDING REQUEST..... 25
INCOME STATEMENTS..... 25
BALANCE SHEET..... 26
CASH FLOW STATEMENTS..... 26
FUNDING REQUEST..... 27
RATIONALE..... 27
RED LOBSTERS LLC MEMBERS..... 28
CLIENT PROPOSITION..... 29
**Requirements..... 32**
1. Functional Requirements..... 32
1.1. Calendar Function..... 32
1.2. User Class..... 33
2. System Requirements..... 35
2.1. Architecture and Design Requirements..... 35
2.2. Implementation Requirements..... 36
3. Technical Requirements..... 37
3.1. Environmental Requirements..... 37
3.2. Performance Requirements..... 37
3.3. External Interface Requirements..... 38
4. Risk Management Plan..... 39
4.1. Technical Risks..... 39
4.2. Operational Risks..... 39
4.3. Financial Risks..... 39
4.4. Human Resource Risks..... 39
4.5. External Risks..... 39
4.6. Strategic Risks..... 39
4.7. Risk Mitigation Strategies..... 39
5. Risk Management Table..... 41
6. Documentation..... 43
6.1. Document Overview..... 43

6.2. Document Structure ..... 44
6.3. Intended Audience and Reading Suggestions ..... 44
6.4. Document Conventions ..... 45
6.5. References ..... 45
6.6. Contact and Feedback ..... 45
6.7. Glossary ..... 45
7. Design Requirements ..... 46
7.1. System Architecture ..... 46
7.2. User Interface Design ..... 47
8. Testing Requirements ..... 47
8.1. Testing Software Quality ..... 47
8.2. Testing Requirements ..... 47
9. Problem-Solving Strategy Requirements ..... 47
9.1. Technical Approach ..... 47
9.2. Continuous Integration/Continuous Deployment (CI/CD) ..... 49
10. Client Liaison Requirements ..... 49
10.1. Communication Plan ..... 49
10.2. Reporting Structure ..... 49
10.3. Requirement Validation ..... 49
10.4. Change Management Process ..... 49
10.5. Client Feedback Integration ..... 50
11. State/Mode Requirements ..... 50
11.1. State Transition Requirements ..... 50
11.2. Mode-Specific Requirements ..... 52
11.3. State/Mode Tests ..... 52
12. Project Planning Requirements ..... 53
12.1. Resource Allocation ..... 53
12.2. Scope Change Management ..... 53
12.3. Maximum Hours Limit ..... 53
12.4. Assigned Tasks ..... 53
12.5. Calendar ..... 53
13. Project Group Management Requirements ..... 54
13.1. Management Strategy ..... 54
13.2. Conflict Resolution Strategy ..... 54
13.3. Proactive Tasks of Management ..... 54
13.4. Progress Reports ..... 55
13.5. Accountability Model ..... 55
13.6. Acknowledgement Model ..... 55
13.7. Agenda Management ..... 55
14. Project Role Specification ..... 55
14.1. Project Manager: Austin Truong ..... 55
14.2. Implementation Manager: Julian Hernendez ..... 55
14.3. Collaboration Software Manager: Benny Chen ..... 56

14.4. Video/Media Manager: Emma Nunnink....56
14.5. Planner: Penny McMahon....56
14.6. Webmaster: Julian Hernandez....56
14.7. Documentation Manager: Graciela Jimenez....56
14.8. Requirements Manager: Penny McMahon....56
14.9. Design Manager: Qi Deng....56
14.10. Unix System Programmer: Carlos Tapia....56
14.11. Testing Manager: Nick Dilallo....57
14.12. Presentation Manager: Emma Nunnink....57
15. Stakeholders....57
15.1. The creators of this application are the Red Lobsters....57
15.2. Primary stakeholders of this application are XYZ Corporation and investors....57
15.3. Secondary stakeholders and investors of this application are Clark Elliot and DePaul University....57
16. User Profile Requirements....57
16.1. Skill Set....57
16.2. Training....57
16.3. User Types....57
17. Freeze Date....58
17.1. The official freeze date is October 26, 2023....58
17.2. Change Request Process....58
**REQUIREMENTS MATRIX....59**
Functional Requirements....59
System Requirements....70
Technical Requirements....73
Documentation....79
Design Requirements....86
Testing Requirements....89
Problem-Solving Strategy Requirements....90
Client Liaison Requirements....94
State/Mode Requirements....97
Project Planning Requirements....102
Project Group Management Requirements....104
Project Role Specifications....108
Stakeholders....111
User Profile Requirements....112
Freeze Date....113
**Design....114**
Diagrams....114
Data Dictionary....123
Table: django_migrations....123
Table: sqlite_sequence....123
Table: django_content_type....124

Table: auth_group_permissions ..... 124
Table: auth_permission ..... 124
Table: auth_group ..... 124
Table: users_new_user_model_groups ..... 125
Table: users_new_user_model_user_permissions ..... 125
Table: django_admin_log ..... 125
Table: base_task_model ..... 126
Table: django_session ..... 126
Table: token_blacklist_blacklistedtoken ..... 126
Table: token_blacklist_outstandingtoken ..... 127
Table: users_new_user_model ..... 127
Files on Disk ..... 128
Backend Directory Mapping ..... 128
Frontend Directory Mapping ..... 128
Data Structures ..... 129
Task Properties: ..... 129
User Properties ..... 130
AuthenticationToken Properties: ..... 130
Security and Privacy ..... 131
Data Encryption ..... 131
User Authentication ..... 131
Access Control ..... 132
Regular Security Audits ..... 132
Privacy Policies ..... 132
Data Retention ..... 132
Third-Party Integration ..... 132
Incident Response ..... 133
User Education ..... 133
Handling of Multiple Users, Transactions, and Requests: ..... 133
Scalable Architecture ..... 133
Database Optimization: ..... 133
Monitoring and Alerts ..... 134
Redundancy and Failover ..... 134
Data Isolation ..... 134
Quality of Service (QoS) ..... 134
Hardware ..... 134
Network Protocols ..... 135
**Plan ..... 137**
**Critical Path ..... 137**
**Dependency Graph ..... 139**
**Individual Planner ..... 140**
**System Overview ..... 142**
Architecture ..... 142

Technology....142
Backend System Configuration....142
Modularity....142
Dependencies....143
Frontend System Configuration....144
Dependencies:....144
Environment....144
System Endpoints....145
Admin Panel:....145
Base App....145
Authentication-Related Endpoints....145
Task-Related Endpoints....145
User App....146
**Backend Code....146**
Base/API....148
filters.py:....148
Serializer.py:....149
APIviews.py:....151
Base/Model:....155
Base/Admin:....157
Config/Settings:....158
Users....163
models.py:....164
views.py:....165
serializers.py:....167
**Frontend Code....168**
COMPONENTS....169
Breadcrumb.tsx....169
Card{One-Five}.tsx....171
ChartThree.tsx....172
ChartTwo.tsx....178
CheckboxFour.tsx....182
DarkModeSwitcher.tsx....184
DropdownDefault.tsx....186
DropdownNotification.tsx....189
DropdownUser.tsx....192
Header.tsx....195
ModalSettings.tsx....199
Sidebar.tsx....202
SidebarLinkGroup.tsx....209
TaskChart.tsx....210
TaskHeader.tsx....214
TaskLarge.tsx....217

TaskList.tsx ..... 223
TaskModal.tsx ..... 227
TaskSmall.tsx ..... 233
CONTEXT ..... 234
AuthContext.tsx & AuthProvider.tsx ..... 234
TaskContext.tsx & TaskProvider.tsx ..... 237
HOOKS ..... 243
fireToast.tsx ..... 243
useAxios.ts ..... 246
useColorMode.tsx ..... 248
useLocalStorage.tsx ..... 249
useTaskContext.tsx ..... 250
PAGES ..... 251
Authentication/SignIn.tsx ..... 251
Authentication/SignUp.tsx ..... 255
Dashboard/Home.tsx ..... 261
LandingPage.tsx ..... 263
Pet.tsx ..... 265
ContactPage.tsx ..... 266
Settings.tsx ..... 271
TaskManager.tsx ..... 276
**Testing Plan ..... 296**
Testing Matrix ..... 296
Functional Testing ..... 296
System Testing ..... 342
Test Code ..... 353
base/tests.py ..... 353
api/tests.py ..... 354
Test Results ..... 356
**Website Screenshots ..... 357**
**User Manual ..... 369**
Welcome to CrabbyTask's User Manual ..... 369
Getting Started ..... 369
Landing Page ..... 370
Sign In Page ..... 371
Home Page ..... 372
Task Manager Page ..... 377
**Technical Manual ..... 380**
Welcome to CrabbyTask's Technical Manual ..... 380
Development Environment Setup ..... 381
Local Deployment ..... 384
1. Clone the Repository ..... 384
2. Environment Variables ..... 384

3. Build the Frontend..... 385
4. Start the Backend..... 385
5. Database Configuration..... 385
Deployment Steps..... 386
1. Clone the Repository..... 386
2. Environment Variables..... 386
3. Build the Frontend..... 387
4. Start the Backend..... 387
5. Database Configuration..... 387
6. Proxy Setup..... 388
7. SSL Certificate..... 388
8. Monitoring and Logging..... 388
9. Backup and Recovery..... 388
10. Scaling and Load Balancing..... 389
11. Security Considerations..... 389
**Future Plans..... 390**
● Draggable Interface..... 391
● Flexible Search..... 391
● Pet System..... 391
● Exportable Data & Analytics..... 391
● Notification System & Toasts..... 392
● Mobile & Browser Compatibility..... 392
● Artificial Intelligence & Machine Learning..... 393
**TimeLogs..... 394**
a. AUSTIN TRUONG..... 394
b. BENNY CHEN..... 399
c. CARLOS TAPIA..... 401
d. EMMA NUNNINK..... 408
e. GRACIELA JIMENEZ..... 409
f. JULIAN HERNANDEZ..... 411
g. NICK DILALLO..... 415
h. PENNY MCMAHON..... 417
i. QI DENG..... 419

# Project Overview

## PROJECT NAME

CrabbyTasks

## PROJECT MANAGERS

- Austin Truong (Project Manager)
- Benny Chen (Collaboration Software Manager)
- Carlos Tapia (Unix Systems Programmer Manager)
- Emma Nunnink (Video/Media & Presentation Manager)
- Graciela Jimenez (Documentation Manager)
- Julian Hernandez (Implementation & Webmaster Manager)
- Nick Dilallo (Testing Manager)
- Penny McMahon (Planner & Requirements Manager)
- Qi Deng (Design Manager)

## PROJECT SPONSORS

Dr. Clark Elliot and XYZ Corp

## BUSINESS CASE

CrabbyTasks is a revolutionary task management software designed to address the cognitive burden and productivity challenges faced by individuals who need an efficient way to manage their tasks and responsibilities. In a fast-paced world where digital distractions abound, CrabbyTasks aims to empower users to regain control of their time and focus on what truly matters. Driven by the vision of "do more by thinking less," this project aspires to provide a comprehensive solution for personal task management.

## PROBLEMS

Modern task management tools often require users to expend significant mental effort on organizing, prioritizing, and tracking their tasks. This can lead to a feeling of overwhelm and

hinder productivity. CrabbyTasks seeks to alleviate this problem by automating routine tasks, providing intelligent recommendations, and offering task analytics. Unlike other solutions that demand users to micromanage their tasks, CrabbyTasks shifts the burden to the software, allowing users to concentrate on their work.

## GOALS

- Simplify Task Management: CrabbyTasks aims to simplify the task management process by automating repetitive tasks, categorizing tasks intelligently, and providing a clean and intuitive interface.
- Reduce Cognitive Load: By handling the minutiae of task management, CrabbyTasks frees users from the mental overhead of organizing their to-dos, allowing them to think less and do more.
- Enhance Productivity: Through features like task analytics and notifications, CrabbyTasks empowers users to make data-driven decisions and stay on top of their tasks.
- Personalization: The goal is to provide a highly personalized experience, adapting to individual work habits and preferences.
- Incorporate Machine Learning: As a long-term goal, CrabbyTasks envisions incorporating machine learning to offer predictive task management capabilities.

## DELIVERABLES

The key deliverables of the CrabbyTasks project include:

- A fully functional, user-friendly task management software application tailored for individual users.
- A web interface designed for effortless task management.
- Comprehensive documentation for users and developers, ensuring ease of use and support.
- A robust testing matrix to guarantee product quality and reliability.
- A commitment to ongoing updates and improvements based on user feedback and evolving needs.

## RISKS & OBSTACLES

While CrabbyTasks holds great promise, it's not without challenges. Risks and obstacles may include:

- Technical Challenges: Developing advanced features such as task analytics and machine learning integration requires technical expertise and innovation.
- Market Competition: The task management software market is competitive, and standing out from established solutions is a challenge.
- Marketing & User Adoption: Effectively marketing CrabbyTasks and ensuring user adoption will be crucial for success.
- Resource Management: Allocating resources efficiently and managing scope changes will require careful planning.
- Regulatory & Compliance: Staying compliant with data protection and privacy regulations is essential.

By addressing these challenges, CrabbyTasks aims to deliver a unique and valuable task management solution for individuals, helping them reclaim their time and productivity.

# BUSINESS PLAN

## EXECUTIVE SUMMARY

Red Lobsters LLC, headquartered in the vibrant city of Chicago, Illinois, is a dynamic technology company on a mission to simplify the lives of individuals through innovative software solutions. At the heart of our endeavors is CrabbyTasks, a groundbreaking task management software that redefines personal productivity. With CrabbyTasks, users can effortlessly manage their daily responsibilities, reduce cognitive load, and unleash their full potential. Our goal is to empower individuals to "do more by thinking less" and to become the go-to choice for task management, setting a new standard in the industry.

## COMPANY DESCRIPTION

Red Lobsters LLC, based in the bustling metropolis of Chicago, Illinois, is a forward-thinking technology company with a passion for making life easier for individuals. Our journey began in

2023, driven by a shared commitment to deliver innovative solutions that enhance personal productivity. Our flagship product, CrabbyTasks, embodies our dedication to user-centric design and cutting-edge technology.

## COMPANY NAME

Red Lobsters LLC

## BUSINESS STRUCTURE

Red Lobsters LLC operates as a limited liability company, combining the advantages of liability protection with operational flexibility.

## OWNERSHIP

Red Lobsters LLC is privately owned and operated by its founders, fostering a culture of agility and innovation.

## LOCATION

Our headquarters are strategically situated in Chicago, Illinois, a hub of innovation and talent, with remote team members contributing from across the globe.

## COMPANY HISTORY

Founded in 2023, Red Lobsters LLC has a rich history of developing user-friendly software solutions that empower individuals to take control of their tasks and accomplish more.

## MISSION STATEMENT

Our mission is to simplify complex tasks and empower individuals to achieve their goals with ease by providing intuitive software solutions.

### PRODUCTS/SERVICES & TARGET MARKET

CrabbyTasks, our flagship product, is designed to cater to professionals, students, freelancers, and anyone seeking a smarter way to manage personal tasks. Our target market encompasses a diverse range of individuals looking to optimize their task management.

### BRAND VALUES AND GOALS

At Red Lobsters LLC, we uphold the values of innovation, user-centricity, excellence, integrity, and adaptability. Our goals include becoming a trusted name in the task management software industry, fostering a supportive user community, and driving continuous innovation to meet evolving user needs.

### VISION STATEMENT

Our vision is to revolutionize personal productivity by providing individuals with a task management tool that adapts seamlessly to their unique needs, simplifying their lives and boosting their productivity.

## MARKET ANALYSIS

CrabbyTasks operates in a competitive market for task management software. Our analysis includes:

### TARGET MARKET

CrabbyTasks is designed for individuals who seek a more effective way to manage their personal tasks and responsibilities. Our software caters to professionals, students, freelancers, and anyone looking to optimize their task management. We recognize that the landscape of personal productivity is vast, and CrabbyTasks aims to serve a wide spectrum of users.

## POTENTIAL MARKET

The potential market for CrabbyTasks is substantial, considering the growing demand for efficient task management solutions. As more individuals face the challenges of information overload and task complexity, our software presents a valuable solution. This potential market extends to various industries, including business professionals, educators, creatives, and more.

## SWOT ANALYSIS

Our SWOT analysis assesses the internal and external factors that impact CrabbyTasks:

Strengths:

- CrabbyTasks offers a user-friendly interface and innovative features, including task analytics and automation.
- Strong development team with expertise in software design and user experience.
- Strategic location in Chicago, a tech-savvy city with a supportive ecosystem.

Weaknesses:

- As a new entrant, building brand recognition and trust will be a challenge.
- Ongoing software updates and maintenance require significant resources.
- The need to adapt to changing user preferences and technological advancements.

Opportunities:

- Growing demand for personal task management solutions.
- Expanding partnerships with educational institutions and businesses.
- Incorporating machine learning to offer predictive task management.

Threats:

- Intense competition from established task management software providers.
- Rapid changes in technology and user expectations.

- Data security and privacy concerns.

## COMPETITIVE ANALYSIS

In the task management software market, we face competition from established players such as Todoist, Asana, and Trello. Let's take a closer look at how we analyze our competition, using these companies as examples:

Strengths of Todoist:

- Established user base and brand recognition.
- Cross-platform availability and integration with third-party apps.
- Well-documented and user-friendly.

Weaknesses of Todoist:

- Learning curve for new users.
- Limited personalization and automation features in the free version.
- Pricing may not suit budget-conscious individuals.

Opportunities for CrabbyTasks:

- Offering an even more intuitive user experience.
- Providing advanced task analytics and machine learning-driven insights.
- Competitive pricing plans, including a robust free tier.

Threats from Todoist:

- Todoist may respond with product updates or pricing adjustments.
- Convincing users to switch from a well-established tool.

By conducting a detailed competitive analysis, we gain insights into the strengths, weaknesses, opportunities, and threats presented by our competitors. This knowledge informs our strategy to position CrabbyTasks as a compelling alternative in the market.

## COMPETITOR DATA COLLECTION PLAN

We have implemented a comprehensive competitor data collection plan to continuously monitor the performance and strategies of our competitors. This plan includes:

- Regularly monitoring competitor websites and product updates.
- Analyzing user reviews and feedback on competitor products.
- Tracking social media mentions and sentiment analysis.
- Conducting surveys and interviews with users who have switched from competitors to CrabbyTasks.

## MARKETING EXPENSE STRATEGY CHART:

Our marketing expense strategy includes allocating resources to various channels to reach our target audience. In the initial phase, we plan to invest approximately $100,000 in marketing efforts, distributed as follows:

- Digital Advertising: $40,000
- Social Media Marketing: $20,000
- Influencer Partnerships: $15,000
- Content Marketing: $15,000
- Email Marketing: $10,000

## OPERATIONAL PLAN

Our operational plan outlines the key activities required to launch and maintain CrabbyTasks. This includes software development, customer support, and continuous improvement efforts.

## MANAGEMENT AND ORGANIZATION

Red Lobsters LLC is led by a dedicated management team with extensive experience in software development, marketing, and business strategy.

### ORGANIZATION CHART

Our organization chart illustrates the key roles and responsibilities within Red Lobsters LLC, ensuring clear lines of communication and accountability.

### STARTUP EXPENSES & CAPITALIZATION

To kickstart CrabbyTasks, we estimate initial startup expenses of $500,000, which will cover development costs, marketing, legal, and operational expenses. Our capitalization plan includes a mix of equity investment from founders and potential seed funding.

### REFINING THE PLAN

Our business plan is a dynamic document that will be continuously refined based on market feedback, user data, and changing market conditions. We are committed to staying agile and responsive to ensure the success of CrabbyTasks.

## ORGANIZATION & MANAGEMENT TEAM

Red Lobsters LLC is established as a limited liability company (LLC) registered in the state of Illinois. Carlos Tapia is the majority owner of the company, providing the vision and direction for our endeavors.

### BOARD OF DIRECTORS

Carlos Tapia - Chief Executive Officer (CEO) and Chairman of the Board:

> Carlos Tapia, the majority owner, serves as the CEO and Chairman of the Board. He provides overall leadership and strategic direction for Red Lobsters LLC and CrabbyTasks.

Austin Truong - Chief Operations Officer (COO) and Board Member:

> Austin Truong is the COO and a board member. He oversees the day-to-day operations of the company and plays a key role in shaping CrabbyTasks' development.

Benny Chen - Chief Technology Officer (CTO) and Board Member:

Benny Chen serves as the CTO and a board member, responsible for the technical direction of CrabbyTasks and ensuring its technical excellence.

Emma Nunnink - Chief Marketing Officer (CMO) and Board Member:

Emma Nunnink is the CMO and a board member. Emma leads our marketing efforts and plays a vital role in promoting CrabbyTasks.

Julian Hernandez - Chief Implementation Officer (CIO) and Board Member:

Julian Hernandez serves as the CIO and a board member, responsible for overseeing the implementation of CrabbyTasks and maintaining its quality.

Graciela Jimenez - Chief Documentation Officer (CDO) and Board Member:

Graciela Jimenez is the CDO and a board member. She ensures that users have access to comprehensive documentation and resources for CrabbyTasks.

Nick Dilallo - Chief Testing Officer (CTO) and Board Member:

Nick Dilallo is the CTO and a board member, leading the testing efforts to maintain the high quality and reliability of CrabbyTasks.

Penny McMahon - Chief Planner Officer (CPO) and Board Member:

Penny McMahon serves as the CPO and a board member, overseeing project planning and requirements management to align CrabbyTasks with user needs.

Qi Deng - Chief Design Officer (CDO) and Board Member:

Qi Deng is the CDO and a board member, responsible for the design elements of CrabbyTasks to enhance its user experience.

### SUPPORTING PROFESSIONALS

In addition to our core management team, we collaborate with a network of supporting professionals, including legal advisors, financial consultants, and technical experts, who contribute their knowledge to ensure the success and growth of CrabbyTasks.

## SERVICE & PRODUCT LINE

Red Lobsters LLC is dedicated to simplifying personal productivity with our innovative software solutions. Our current product, CrabbyTasks, serves as the cornerstone of our service and product line. Here's a closer look at CrabbyTasks and our potential future offerings:

### CURRENT PRODUCT

CrabbyTasks: Task Management Made Effortless

CrabbyTasks is our flagship product, designed to streamline and simplify task management for individuals. It allows users to organize tasks, set priorities, and track progress with ease. Key features of CrabbyTasks include:

- Intuitive User Interface: A user-friendly interface ensures a smooth experience, making task management accessible to all.
- Task Analytics: Gain insights into task completion patterns and optimize productivity.
- Automation: Reduce manual effort with task automation and recurring task scheduling.
- Cross-Platform Compatibility: Access CrabbyTasks on web browsers and mobile devices for convenience.
- Notification System: Stay on top of tasks with timely reminders and notifications.

CrabbyTasks is ideal for professionals, students, freelancers, and anyone seeking a more efficient way to manage their tasks.

### FUTURE PRODUCT

CrabbyInsights: Unleashing the Power of Task Analytics

Building on the success of CrabbyTasks, our future product, CrabbyInsights, takes task management to the next level with advanced analytics and machine learning capabilities. CrabbyInsights aims to provide users with:

- Predictive Task Management: Leverage machine learning algorithms to forecast task completion times and recommend optimal task priorities.
- Task Efficiency Metrics: Measure your productivity and identify areas for improvement with detailed task analytics.
- Goal Tracking: Set and track personal or professional goals, with real-time progress monitoring.
- Cross-Platform Integration: Seamlessly integrate CrabbyInsights with popular productivity tools and calendars.

CrabbyInsights complements CrabbyTasks by offering advanced data-driven insights, empowering users to make informed decisions about their tasks and priorities.

At Red Lobsters LLC, we are committed to continuously enhancing our product line to meet the evolving needs of our users. CrabbyTasks and CrabbyInsights are just the beginning of our journey to revolutionize personal productivity.

## MARKETING & SALES

Red Lobsters LLC is dedicated to promoting CrabbyTasks effectively in the market. Our marketing and sales strategy encompasses a comprehensive approach to reach our target audience, generate leads, and convert them into loyal customers.

### PRICE

CrabbyTasks Pricing:

CrabbyTasks offers a flexible pricing structure to cater to the diverse needs of our users:

- Free Plan: Includes basic task management features with limited storage and support.
- Premium Monthly Plan ($9.99/month): Unlock advanced features, such as task analytics, automation, and priority support.

- Premium Annual Plan ($99.99/year): Enjoy the benefits of the Premium Plan at a discounted rate for an entire year.

## PRODUCT

CrabbyTasks Features:

- Intuitive Task Management: Easily create, organize, and prioritize tasks.
- Task Analytics: Gain insights into task completion patterns and productivity.
- Automation: Automate repetitive tasks and set recurring schedules.
- Cross-Platform Access: Use CrabbyTasks on web browsers, and a mobile app is in development for future access.
- Notification System: Stay on top of tasks with timely reminders and alerts.

## PROMOTION

Digital Advertising:

- Utilize Google Ads and Facebook Ads to target users interested in task management solutions.
- Conduct keyword research to optimize ad campaigns and drive organic traffic.

Social Media Marketing:

- Maintain active profiles on platforms such as Twitter, LinkedIn, and Instagram to engage with users and share valuable content.
- Collaborate with influencers in the productivity and tech niches to promote CrabbyTasks.

Content Marketing:

- Publish blog posts, articles, and guides on task management best practices and productivity tips.
- Develop video content and tutorials on YouTube to showcase CrabbyTasks' features.

Email Marketing:

- Build and segment email lists to deliver targeted content and product updates to subscribers.
- Send regular newsletters featuring user success stories and product announcements.

Partnerships and Affiliates:

- Partner with tech blogs and websites to feature CrabbyTasks in reviews and articles.
- Establish an affiliate program to encourage users to refer CrabbyTasks to others.

## PLACE

Distribution Channels:

- Offer direct access to CrabbyTasks through our official website.
- Launch a mobile app version in the future to provide additional access options for users.

## SALES FORECAST

- Year 1 Sales Projection: We anticipate acquiring 50,000 users within the first year, with a conversion rate of 5% from free users to premium users.
- Year 2 Sales Projection: Our user base is expected to grow to 100,000 users, driven by word-of-mouth referrals and successful marketing campaigns.
- Year 3 Sales Projection: We aim to reach 200,000 users as our products become increasingly recognized in the market.
- Year 4 and Beyond: As we continue to innovate and expand our product line, we foresee exponential growth in user acquisition and sales.

Red Lobsters LLC is committed to providing excellent value to our users while steadily growing our customer base. Our marketing and sales efforts are geared towards making CrabbyTasks accessible to individuals seeking effective task management solutions.

# OPERATION PLAN

Red Lobsters LLC's operation plan covers various aspects critical to the success of CrabbyTasks and CrabbyInsights.

## SUPPLIERS

We source our software development tools, hardware infrastructure, and other necessary resources from reputable suppliers known for quality and reliability. Collaborating with trusted suppliers ensures the smooth operation of our software products.

## PRODUCTION

The production process primarily involves software development, quality assurance, and continuous improvement. Our development team works diligently to enhance the features and performance of CrabbyTasks, with regular updates and bug fixes.

## FACILITIES

Our operations are based in Chicago, Illinois, where our team works collaboratively to develop, maintain, and support CrabbyTasks and CrabbyInsights. We maintain a productive and comfortable workspace to foster creativity and innovation.

## LEGAL EQUIPMENT & ENVIRONMENT

Red Lobsters LLC is committed to adhering to all legal requirements and environmental regulations. We ensure the ethical and responsible development of our software products, considering factors like data privacy and security.

## SHIPPING & FULFILLMENT

Since our products are software-based and delivered electronically, we do not have physical shipping or fulfillment requirements. Users can access and download CrabbyTasks directly from our website or app stores.

### INVENTORY

As a software company, we do not maintain physical inventory. Instead, we manage digital assets and software versions, ensuring seamless access to our products for users.

Our operation plan is designed to support the ongoing development, maintenance, and availability of CrabbyTasks and CrabbyInsights while adhering to legal and environmental standards.

## FINANCIAL PROJECTIONS & FUNDING REQUEST

At Red Lobsters LLC, we firmly believe that CrabbyTasks has the potential to make a significant impact on the task management software market. To realize this potential, we present our financial projections and funding request, showcasing the promising future of CrabbyTasks.

### INCOME STATEMENTS

Year 1 (2023):

- Revenue: $250,000
- Expenses: $150,000
- Net Income: $100,000

Year 2 (2024):

- Revenue: $500,000
- Expenses: $250,000
- Net Income: $250,000

Year 3 (2025):

- Revenue: $1,000,000
- Expenses: $400,000
- Net Income: $600,000

## BALANCE SHEET

Year 1 (2023):

- Total Assets: $300,000
- Total Liabilities: $50,000
- Total Equity: $250,000

Year 2 (2024):

- Total Assets: $600,000
- Total Liabilities: $100,000
- Total Equity: $500,000

Year 3 (2025):

- Total Assets: $1,200,000
- Total Liabilities: $200,000
- Total Equity: $1,000,000

## CASH FLOW STATEMENTS

Year 1 (2023):

- Operating Cash Flow: $120,000
- Investing Cash Flow: -$30,000
- Financing Cash Flow: -$10,000
- Net Cash Flow: $80,000

Year 2 (2024):

- Operating Cash Flow: $300,000
- Investing Cash Flow: -$60,000
- Financing Cash Flow: -$20,000

- Net Cash Flow: $220,000

Year 3 (2025):

- Operating Cash Flow: $600,000
- Investing Cash Flow: -$100,000
- Financing Cash Flow: -$30,000
- Net Cash Flow: $470,000

## FUNDING REQUEST

To further develop and scale CrabbyTasks, we are seeking $1,000,000 in funding. This capital infusion will be used for:

- Expanding the development team to accelerate feature enhancements and mobile app development.
- Increasing marketing and promotional efforts to reach a wider audience.
- Investing in customer support and user engagement to ensure user satisfaction and retention.
- Exploring partnerships and collaborations to extend our reach within the productivity software market.

## RATIONALE

CrabbyTasks has shown remarkable growth potential, with a substantial increase in revenue and user adoption year over year. Our commitment to enhancing the product, coupled with strategic marketing and a clear monetization strategy, positions us for continued success. The financial projections indicate a strong positive cash flow, demonstrating CrabbyTasks' sustainability and profitability.

We firmly believe that CrabbyTasks is poised to become a prominent player in the task management software industry. With the requested funding, we are confident in our ability to accelerate growth, improve user experience, and deliver exceptional value to our customers.

Red Lobsters LLC is committed to realizing the full potential of CrabbyTasks and seeks the support and recognition to continue our journey toward transforming personal productivity.

Please review the financial projections and funding request section, including the fake data and charts. If you have any further adjustments or if you'd like to move on to the next section, please let me know.

## RED LOBSTERS LLC MEMBERS

1. AUSTIN TRUONG (Project Manager): Austin Truong serves as the Project Manager, responsible for overseeing the overall development and successful launch of CrabbyTasks. His duties include project planning, task allocation, and ensuring project milestones are met. Austin has been instrumental in keeping the team organized and focused, ensuring that deadlines are consistently achieved.

2. BENNY CHEN (Collaboration Software Manager): Benny Chen takes on the role of Collaboration Software Manager, leading the development of collaborative features of the company.

3. CARLOS TAPIA (Unix Systems Programmer Manager): Carlos Tapia plays a pivotal role as the Unix Systems Programmer Manager, leveraging his deep understanding of technology and systems to enhance CrabbyTasks. He is responsible for optimizing the system's performance, ensuring seamless functionality, and bolstering system security. Carlos's profound technical expertise has been instrumental in developing the robust infrastructure of CrabbyTasks.

4. EMMA NUNNINK (Video/Media & Presentation Manager): Emma Nunnink handles the Video/Media & Presentation Manager role, creating engaging multimedia content to promote CrabbyTasks. Emma’s video tutorials and presentations have effectively conveyed the product's features and benefits to our user base.

5. GRACIELA JIMENEZ (Documentation Manager): Graciela Jimenez serves as the Documentation Manager, ensuring that comprehensive user guides and documentation are available to assist CrabbyTasks users. Her dedication to providing clear and helpful documentation has contributed to a positive user experience.

6. JULIAN HERNANDEZ (Implementation & Webmaster Manager): Julian Hernandez holds the roles of Implementation and Webmaster Manager, overseeing the product's implementation process and maintaining the official website. Julian has efficiently managed the implementation of new features and updates, ensuring seamless user experiences.

7. NICK DILALLO (Testing Manager): Nick Dilallo is responsible for testing CrabbyTasks to ensure quality and reliability. His rigorous testing procedures have helped identify and address issues, leading to a more stable product.

8. PENNY MCMAHON (Planner & Requirements Manager): Penny McMahon wears two hats as the Planner and Requirements Manager. She coordinates project planning and manages the product's requirements, ensuring that development aligns with user needs. Penny has played a crucial role in defining project goals and objectives.

9. QI DENG (Design Manager): Qi Deng leads the design efforts for CrabbyTasks, focusing on user interface and experience enhancements. Qi has introduced a visually appealing and user-friendly design, contributing to positive user feedback.

Accomplishments:

- As a team, we have successfully launched CrabbyTasks as a web application, with plans to expand to mobile platforms.
- Key features such as task management, analytics, and automation have been implemented and refined.
- Our marketing efforts have gained traction, resulting in a growing user base.
- Continuous feedback from users has been invaluable in shaping the product's roadmap.
- We have laid a strong foundation for future development, including the introduction of a premium subscription model.

## CLIENT PROPOSITION

Are you tired of juggling multiple tasks and drowning in the complexities of traditional task management tools? Look no further; CrabbyTasks is here to revolutionize the way you manage your tasks and boost your productivity.

In today's fast-paced and interconnected world, staying organized and efficient is more critical than ever. Traditional task management tools often fall short, burdening users with outdated interfaces and limited capabilities. It's time for a modern, cutting-edge solution that caters to your evolving needs.

## Why Choose CrabbyTasks?

Modern & Intuitive Design: CrabbyTasks welcomes you with a sleek and intuitive interface that makes task management a breeze. Say goodbye to clunky, outdated software, and embrace a fresh, modern approach to getting things done.

Cutting-Edge Features: CrabbyTasks doesn't just manage your tasks; it understands them. With advanced features like task analytics, intelligent automation, and real-time collaboration, you'll experience productivity like never before. Our machine learning integration, coming soon, will further elevate your task management experience.

Personalized Productivity: We recognize that everyone has unique needs and preferences. CrabbyTasks empowers you to customize your task management experience. Tailor your workflow, automate repetitive tasks, and prioritize what matters most to you.

Cross-Platform Accessibility: Your tasks are not confined to one device or location, and neither is CrabbyTasks. Access your tasks seamlessly on web browsers and, in the near future, on your mobile device with our upcoming mobile app.

Streamlined Collaboration: CrabbyTasks facilitates collaboration with your peers, whether you're working on a team project or coordinating with friends and family. Real-time updates and file sharing ensure everyone is on the same page.

Task Insights & Analytics: Gain valuable insights into your task completion patterns and productivity trends. Understand how you work and make informed decisions to optimize your workflow.

Always Evolving: CrabbyTasks is not just a product; it's a commitment to continuous improvement. We're dedicated to delivering regular updates and enhancements to meet your evolving needs.

Join the Productivity Revolution: Be part of a community that values productivity, innovation, and cutting-edge technology. CrabbyTasks is more than a task manager; it's a movement towards a more organized, efficient, and balanced life.

## Why Wait?

CrabbyTasks is the answer to your quest for a modern, feature-rich, and user-friendly task management solution. Don't settle for outdated tools that hinder your productivity. Embrace the future of task management with CrabbyTasks and experience a new level of efficiency, collaboration, and control.

Join us in redefining productivity and maximizing your potential. Try CrabbyTasks today and embark on a journey toward a brighter, more organized future.

Your Tasks, Your Way, with CrabbyTasks - The Future of Productivity.

# Requirements

## 1. Functional Requirements

### 1.1. Calendar Function

1.1.1. Display Standard Calendar Elements:

1.1.1.1. Application will have the capacity to provide 3 distinct and thematically cohesive monthly, weekly, and daily calendar views, such that the overarching theme captures cretaceous sea life through its red colors, textures, and images; While enabling simple, organized, viewable representation of at least 3 tasks relative to their respective type of view.

1.1.2. Display Saved Tasks:

1.1.2.1. The application will automatically populate monthly, weekly, daily calendar views in an organized and uniform way relative to the type of view that is currently presented to the user while also maintaining task functionality, thematic cohesiveness, and system integrity.

1.1.3. Allow Users to View Tasks in Detail:

1.1.3.1. Upon interaction with any specific task component, the application will dynamically update task view to fit a view height and width capable of holding a 1.2 rem task title and 1 rem task description with enough padding to also hold any additional feature flags and descriptors.

1.1.4. Provide Task Modification Functions:

1.1.4.1. Within the task manager page, users will find various task management options, including adding new tasks, deleting tasks, editing task details, setting task priorities, assigning task categories, and opting to receive email notifications for task updates.

1.1.4.1.1. User ability to add new tasks

1.1.4.1.2. User ability to delete tasks

1.1.4.1.3. User ability to edit task details

1.1.4.1.4. User ability to set task priorities

1.1.4.1.5. User ability to assign task categories

1.1.4.1.6. User ability to opt in or opt out of receiving email notifications for task updates

1.1.4.2. Additionally, a dedicated "Add Task" function will be available in both the weekly and daily calendar views for user convenience.

## 1.2. User Class

1.2.1. Create a New Account:

1.2.1.1. On the user login page, provide an option for users to create a new account.

1.2.1.2. Prompt users to create a preferred username and password for future access.

1.2.2. Account Login:

1.2.2.1. Allow both new and existing users to log in with their correct username and password.

1.2.2.2. Upon successful login, the application will redirect users to task manager page which contains the task dashboard and auxiliary task manipulation functions; Only then, if the user manipulates any task present on the viewfinder, the application will functionally handle all events related to acquiring, saving, and viewing of the manipulated task to lower mental user overhead.

1.2.3. Change Password:

1.2.3.1. Offer a "Change Password" option within the login section for users who wish to reset their password.

1.2.3.2. When resetting a password, require users to input their username and specify the corresponding email address in order to receive password reset instructions.

1.2.4. Notification System:

1.2.4.1. Email notification.

1.2.4.1.1. When the user is on the Task Manager page and is adding or editing a task, an option will be offered to opt-in for e-mail notifications for that specific task. The option for e-mail notifications will be defaulted to "No". If opted-in for email notifications, an additional dropdown or selection will be offered to choose how many hours/days prior to the deadline of the selected task the user wants to be notified via email.

1.2.4.2. Window notification.

1.2.4.2.1. When on the Task Manager page and the user is adding or editing a task, if the option for desktop notifications is selected, then an additional dropdown or selection will be offered to choose how many hours/days prior to the task deadline the user wants to be notified on desktop (if they are on/at the computer). The default for the desktop notification option will be "No".

1.2.4.3. Notification Preference.

1.2.4.3.1. Allow users to select an approximate date and time to receive email notifications about their tasks.

1.2.5. Personalize Website Appearance:

1.2.5.1. Allow users to select between light and dark modes for the theme of the website.

1.2.6. Task Organization:

1.2.6.1. On the task manager page and within each of the edit task sections and add task sections, allow users to modify the priority preference of any task and ensure that the changes are saved as well as represented across all views the tasks are present in.

1.2.6.1.1. See Section 1.1.4.1.4.

1.2.6.2. On the task manager page, within each of the edit task sections and add task sections, enable users to assign categories to tasks using colored flags and ensure that the changes are saved as well as represented across all views the tasks are present in.

1.2.6.2.1. See Section 1.1.4.1.5.

1.2.6.3. Filter to order tasks by priority:

1.2.6.3.1. The application will have a dedicated component in the viewfinder of the page where all complete, and incomplete tasks reside; Upon arrival of this page, tasks will be ordered by the date of which they were conceived; This component will allow the user to programmatically manipulate the order of which the tasks are painted on to the screen; Additionally, the component will not jeopardize the integrity of the system, thematic cohesiveness of the system, or responsiveness of the system;

1.2.6.4. Filter to order tasks by category:

1.2.6.4.1. On the task manager page, a function will be implemented to allow users to order tasks by category using colored flags in a drop down. By clicking on the flag/category icon next to each task, a selection of colors will be offered to set for that flag's color. The default flag color will be gray to indicate no category.

1.2.6.5. Filter to order tasks by edit or creation date:

1.2.6.5.1. In the task manager page, include a function that allows users to order tasks based on the time, respective to the user's timezone, when they were last edited or, if there are no previous edits, based on when it was created.

1.2.6.6. Filter to order tasks by a range of dates:

1.2.6.6.1. In the tasks catalog, incorporate a function that allows users to order tasks by a specified range of dates, in which they are due, entered into the filter. The due dates of the tasks should be consistent with the user's request; Ensure the calendar is updated to the correct year as well.

1.2.7. Additional Toolbar:

1.2.7.1. On the task manager page, implement a clickable button labeled "Complete Tasks" that opens a menu which provides a clear and detailed display of all of the user's previously completed tasks.

1.2.7.2. A viewable task completion scoring system will be implemented in hopes of showing the user(s) progress and encourage them to complete more tasks earlier. The scoring system will reward points upon task completion with an increasing amount depending on how much earlier the task was completed compared to its deadline. Conversely, points will be deducted if tasks are not completed on time, with the amount taken away increasing as the amount of time past the deadline increases in order to make users finish up late tasks and prevent it from happening in the future.

# 2. System Requirements

## 2.1. Architecture and Design Requirements

2.1.1. System will require Database capable of holding client demand on platform

2.1.1.1. A secured AWS EC2 instance holding Postgresql will have an exclusive channel of communication to the production server.

2.1.2. Server must concurrently handle multiple connections for several users.

2.1.2.1. A secured AWS EC2 instance has Apache2 server handling incoming requests to the server

2.1.3. System must be able to hold against security threats

2.1.3.1. Ssh Protocol will be secured using the latest verified security standards that are available to the group after comparing the pros and cons of each standard/version to make sure they work best for the project.

2.1.4. System be provide a user space separate from other users

2.1.4.1. Unauthorized users will not be able to access personal user pages. The user must have valid credentials and have a verified method of communication to reset password if necessary. This includes developers unless necessary edits or bug fixes are needed and require a look into a user's page

2.1.5. System must onboard users seamlessly

2.1.5.1. Introducing users to the system will consist of an overview of how the application functions.

2.1.6. System must onboard developer seamlessly

2.1.6.1. In the event that a new developer is introduced to the system, the developer will be introduced to the project manual and an environment image to help reduce overhead into the system.

2.1.7. System communication will be held over a trusted secured encrypted TCP connection

2.1.7.1. Inbound connections to the server will be encrypted and be up to SSL standards.

2.1.8. Production server and database will establish secure isolated communication

2.1.8.1. Database will only allow communication to the application server.

2.1.9. System will be hosted to the public at red-lobster.dev

2.1.9.1. System will have its own Domain Name and be placed on public DNS.

2.1.10. Production Server and Databases will be hosted on Virtual Amazon EC2 Instances running Ubuntu 22.04 LTS (HVM) SSD Volume Drive on 64-bit Image

## 2.2. Implementation Requirements

2.2.1. Hardware running the server will be an Amazon EC2 instance.

2.2.2. Amazon EC2 instances will have an Apache Httpd Server that handles connections to the outside world.

2.2.3. Apache 2 will handle static files and will connect with django for any dynamic requests through the WSGI protocol.

2.2.4. Django will be managing authorization, and session tokens with its internal security protocols.

2.2.5. Django will handle queries to the postgreSQL database through its internal Object Relational Mapping tools.

2.2.6. If time allows, the frontend will be routed through React.

2.2.7. The Frontend Developer Team will be creating HTML skeletons of the data to be presented.

2.2.8. The Frontend Developers will handle data visualization required to the frontend.

2.2.9. The Frontend Developer Team will be customizing the HTML skeletons with the use of popular third party Tailwind CSS Framework.

2.2.10. The Backend Developer Team will be creating Data Models to talk with the Database.

2.2.11. The Backend Developer Team will be handling the coupling and routing logic of frontend views by creating specific views of the HTML, CSS, and Javascript created by the frontend team.

2.2.12. The Backend Developer Team will handle security to prevent any malicious users from jeopardizing the system through the use of object relation

2.2.12.1. See Section 3.3.2

2.2.13. The Backend Developers Team will handle middleware and webhooks for any incoming requests to the system.

2.2.14. The Backend Developers will handle database integration and third party tool integrations to the system.

# 3. Technical Requirements

## 3.1. Environmental Requirements

### 3.1.1. Robustness and Resilience Requirements

3.1.1.1. System design will allow for scalability of the system such that it will not impede the average speed of the system by more than 3 seconds from its initial development time.

3.1.1.2. In the case of component failure, the system will prompt the user of the component failure and continue to operate without the failed component unless the component plays a pivotal role in the subsystems operations in which case the system will cease operations, and notify the user of the unexpected fault and a description of what it may entail. The developers and those overseeing the site can be notified about the error when it occurs to ensure it is handled swiftly.

### 3.1.2. Isolation Requirements

3.1.2.1. In the case where a component in the system were to fail, the overall system would continue to operate unless the component in question plays an integral operational function to that system. Users will be notified if the system fails to operate and developers and those overseeing the site are also notified to ensure that solutions are found.

### 3.1.3. Environmental Tests

3.1.3.1. During the development of the system, developers will maintain, develop, and execute system wide test cases for components of the system such that those tests will maintain the integrity of the component.

## 3.2. Performance Requirements

### 3.2.1. Speed and Responsiveness Requirements

3.2.1.1. Initial system benchmarks will be conducted through the use of profiling software that will test the speed of operations.

3.2.1.2. Initial system benchmarks will be used as a baseline comparison to functionally equivalent component substitution decisions.

3.2.1.3. Speed of the system will not surpass 4 seconds for any queries that involve retrieving information with the exception of infrastructure that is outside the governed scope of the application and artifacts outside a typical client use case.

3.2.1.4. Any component or system that does not meet expectations of any party will be audited and reviewed for potential improvements. In the case where a potential replacement to the component exists, there will be test cases created and design analysis to support the new components integration to the system(s). In the event that there does not exist a better alternative to the component in question, the party responsible for auditing the reviewing the component will discuss why this component is the current best possible solution to the problem.

3.2.2. Scalability and Load Handling Requirements

3.2.2.1. Both the developers and consumers understand and agree that the application will be capable of scaling to more than 100,000 users and will have a plan to allow improved infrastructure to hold previously stated levels of volume.

3.2.2.2. During the development of the application it is reasonable to test system load of 10 different users running different processes. Each of those 10 users will have flawless networking experiences while using the application or any component of that system with the exception of any network or environment problems outside the govern scope of the application.

3.2.2.3. During the development of the system, the developers will test the system under fictitious levels of network volume to impersonate varying levels of forecasted volume that could potentially be reached based on research about system overloading. This may include an unreasonable surge in user base.

3.2.3. Performance Tests

3.2.3.1. The developers of the application will use testing software and suites that allows for testing the application's speed, efficiency and responsiveness under test cases presented in the testing matrix maintained and filled by developers and the testing manager.

## 3.3. External Interface Requirements

3.3.1. API and Data Exchange Requirements

3.3.1.1. Under initial development of the application, Data exchange will be handled through the base backend framework Django.

3.3.1.2. If time allows, Django will be transformed into an API that follows the REST protocol. Data exchange between frontend and backend will then be left to developers discretion but must still be fully functional and maintain all the functionality that the

application originally maintained while being coupled under Django. If the client is unsatisfied with the change there is an option to go back to utilizing Django.

3.3.2. Security and Authentication Requirements

3.3.2.1. Passwords will be encrypted using MD5 Hashing to ensure no possibility of client information ever being exposed.

3.3.2.2. SQL Injections will be mitigated using Django Object Relational Mapping.

3.3.2.3. Cross Site scripting attacks will be mitigated using information sanitizers.

3.3.2.4. Cross Site Forgery attacks will be mitigated using CSRF Tokens.

3.3.2.5. Session Hijacking attacks will be mitigated using secured encrypted sessions and refreshed session cookies.

3.3.2.6. Access to resources will follow the least minimum access model.

# 4. Risk Management Plan

## 4.1. Technical Risks

4.1.1. ID T-000

4.1.2. Issues with Technologies, tools, platforms

## 4.2. Operational Risks

4.2.1. ID O-000

4.2.2. Challenges related to processes, resources, infrastructures

## 4.3. Financial Risks

4.3.1. ID F-000

4.3.2. Budget overruns, funding issues, financial losses

## 4.4. Human Resource Risks

4.4.1. ID H-000

4.4.2. loss of key personal, insufficient skill, conflicts

## 4.5. External Risks

4.5.1. ID E-000

4.5.2. Regulatory changes, market shifts, natural disasters

## 4.6. Strategic Risks

4.6.1. ID S-000

4.6.2. Incorrect assumptions, changes in strategy, misaligned with company objective

## 4.7. Risk Mitigation Strategies

### 4.7.1. Avoidance

#### 4.7.1.1. Change plans to circumvent the risk

### 4.7.2. Reduction

#### 4.7.2.1. Reduce the impact or likelihood of the risk

### 4.7.3. Transfer

#### 4.7.3.1. Shift risk to third part

### 4.7.4. Acceptance

#### 4.7.4.1. Acknowledge the risk and prepare a contingency plan

## 5. Risk Management Table

| Risk ID | Risk Description | Probability | Impact | Priority | Mitigation Strategy | Responsible Person |
|---|---|---|---|---|---|---|
| T001 | Server malfunction causing system downtime | Low | High | High | Accept and implement a backup server that will take over operations. | Unix/System Programmer |
| T002 | Hosting provider downtime | Low | High | High | Pray | Unix/System Programmer |
| H001 | Learning Django | High | Medium | Medium | Reduce development time by having a member learn necessary information and share information. | (every implementor should fall under this responsibility) |
| O001 | AWS IAM Roles | High | Medium | Low | Avoid Setting up roles by implementing a Continuous Integration and Development workflow to production. | Unix/System Programmer |

# 6. Documentation

## 6.1. Document Overview

### 6.1.1. Main Objectives:

6.1.1.1. The vision behind the creation of this product is focused on making the invaluable benefits of task-tracking software available for any workplace to implement and improve the productivity of their team. Ultimately, this is a system that fosters punctuality by rewarding the user for completing tasks and encouraging them to see out their goals through the use of in-app incentives and disincentives for completion. With many functionalities that support various styles of planning and categorizing, Crabby Tasks is built to be able to incorporate the scheduling of any kind of work or lifestyle. The application allows the user to have a piece of mind where all tasks are recorded and tracked as well as encourage punctuality and accomplishment. Teams will also benefit from Crabby Tasks as it offers a team collaboration feature.

### 6.1.2. Deliverables:

6.1.2.1. This project supplies a fully functioning task list in the form of a web application built end-to-end by our team, ready for full use by any number of teams, workers, or individuals.

6.1.2.2. The source code of this project will be supplied on GitHub for review and reference from Crabby Tasks' developers and other stakeholders.

### 6.1.3. Boundaries:

6.1.3.1. Crabby Tasks is built for task management, schedule organization and cohesion, and record keeping. The project accomplishes this goal by providing users with many different ways to view their daily, weekly, and monthly schedules along with structuring their tasks in such a way that allows them a full, unabated view.

6.1.3.1.1. This includes only the listed functional requirements listed in Section 8.

6.1.3.2. Crabby Tasks was built for usage and browsing on a desktop web browser and does not have support/functionality for any other platforms.

6.1.3.3. The task manager can manage the tasks of several users at once, but it does not support several accounts being logged in simultaneously to a single instance at a time. Any shared usage of Crabby Tasks must be on a single account in order for the tasks to be viewable at the same time. The only way to accomplish this is with separation by flags, there is no other internal user switching functionality.

6.1.3.4. Crabby Tasks is intended to be used with a working network connection. Any loss of signal will not store unsaved changes if the web page is exited and no pages or edits will load while offline.

6.1.4. Constraints:

6.1.4.1. The team behind Crabby Tasks is composed of college students with little money and time to spare which slows down the development schedule considerably.

6.1.4.2. There is an expected learning curve behind developing the project in Django, setting up a database, server, and backups, and with building an engaging and functional frontend which could affect the types of features supported by the project.

6.1.5. Assumptions:

6.1.5.1. Our team will be able to learn how to implement and thus provide a fully functional web application with all of the features and requirements laid out in this document.

6.1.5.2. Development of the project will be able to be split amongst the nine members of our team throughout its lifecycle.

6.1.6. Acceptance Criteria:

6.1.6.1. In order for the project to be considered complete and successful, it must provide each of the functional requirements listed in Section 1 and those requirements alone. Any unexpected changes to the development cycle of Crabby Tasks between the finalizing of this document and the shipping of the finished product will be considered inconsequential to the overall completion of the project.

## 6.2. Document Structure

6.2.1. This document is structured with each of the main sections designated by a single whole number starting from Section 1 and continuing until Section 16. Every section will have corresponding subsections that may have their own subsections as well, and each of these will be numbered incrementally respective to the other sections in their layer with the format X.Y.Z.

6.2.2. This document has a table of contents and sections for an introduction, overview, business objectives, a risk management table, and a conclusion. These sections will not be numbered.

## 6.3. Intended Audience and Reading Suggestions

6.3.1. This project, and thus this document, is targeted towards workplaces and individuals of any age or demographic who wish to foster healthy organizational and time management routines. Crabby Tasks is meant for those who seek support or relief from part of the mental strain that a loaded schedule can bring; Taking it off of the user's shoulders and putting it into the app instead. This methodology encourages a healthy

compartmentalization of work and gives punctuality a visually rewarding enhancement, making it useful for anyone who has access to a desktop and wishes to manage their schedule efficiently.

6.3.2. Document maintenance and Versioning

6.3.2.1. This document is managed and overseen by the members of our team. Each person contributes and each person reviews new additions to make sure that a cohesive and professional tone is maintained.

6.3.3. The versioning of this document is stored by Google Docs and details the date, time of day, editor, and specifics of individual changes. To ensure this we will avoid anonymous links and personally send this and other important documents through gmail.

## 6.4. Document Conventions

6.4.1. The documentation of the project will reflect the outline of the project booklet. As well as the descriptions and manuals of each requirement specification.

## 6.5. References

6.5.1. All sources will be referenced and properly cited by the project booklet's manual to avoid any issues that could arise down the line.

## 6.6. Contact and Feedback

6.6.1. Clientele or interested parties who have viewed this document and wish to contact our team can reach us at RedLobsterTeamBusiness@gmail.com, where at least one of our developers will look over the feedback.

## 6.7. Glossary

6.7.1. Crabby Tasks:

6.7.1.1. The phrase or title known as “Crabby Tasks” in this document refers to the web application that our team is building and intends to deliver to our client. Any mention of this phrase is referring to the whole end-to-end of the web application as its own entity.

6.7.2. Django:

6.7.2.1. Django is a high-level Python web framework that our team used to develop the website for our project.

6.7.3. REST:

6.7.3.1. The web ‘Application Programming Interface’ we will be using to better serve the frontend of our project if time allows.

6.7.4. AWS EC2:

6.7.4.1. Refers to Amazon Web Services Elastic Compute Cloud. It is one of the core services offered by Amazon Web Services (AWS), which is a cloud computing platform provided by Amazon.

# 7. Design Requirements

## 7.1. System Architecture

7.1.1. System must be maintainable

7.1.1.1. System will be able to be developed and maintained for several years, but will be adhering to all deadlines set by all stakeholders.

7.1.1.2. New developers introduced to the system will be able to develop new features as long as they have read and understood the project manual.

7.1.2. System will be scalable

7.1.2.1. Initial development will only be scalable to handle the original development team. Further and extra development would be required if there was a necessity to handle larger capacities.

7.1.2.2. There is a plan in place to allow the system to scale and handle thousands of users on our platform.

7.1.3. System will reliably be available to users.

7.1.3.1. System must be stable and running on the current release. All features promised on release will be included in the released product.

7.1.3.2. In the event that there is a system failure, there will be a backup server that can handle requests to reduce overall downtime.

7.1.4. System will respond in an efficient time.

7.1.4.1. Requests to the systems will be handled efficiently for the initial release such that they will not have a slow enough response time to warrant any complaints or issues.

7.1.4.2. There will be a plan in place for optimizing server requests through the use of code refactoring and caching systems such that requests and responses are not slow enough to be considered an issue.

7.1.5. Application will be run on the web.

7.1.5.1. The web server utilized will be Apache's HTTPD web server.

7.1.5.2. The client facing web application will utilize the Django templating engine to handle dynamic requests.

7.1.5.3. There will be a plan in place for future improvements to the client facing web-application.

### 7.2. User Interface Design

7.2.1. User Interface will be visually pleasing to end users and be finalized after coming to an agreement with client/stakeholders.

7.2.1.1. Static web design of the application will be handled by Django templating engine.

7.2.2. User Interface will be easy to use for end users

7.2.3. User Interface will be modern and hold a theme based on research regarding common interfaces used for popular platforms and websites.

## 8. Testing Requirements

### 8.1. Testing Software Quality

8.1.1. Testing is measured by how the systems, components, and processes of CrabbyTasks meet our requirements and will go underway the moment that the goals, requirements, and scope of the application are defined. With our risk management plan we should be able to detect problems fast and early to contribute to a durable and secure application.

### 8.2. Testing Requirements

8.2.1. To ensure that the requirements for which we build the base of our program are testable and meet a high threshold of quality. The Requirements testing plan is to first ensure that requirement meets our goals, apply a specific use case against the requirement, ensure that the requirement is unambiguous, check the scope of the requirement, finalize the requirement, perform and review test case, design code based off requirement and test the code off the test case.

## 9. Problem-Solving Strategy Requirements

### 9.1. Technical Approach

9.1.1. Introduction:

9.1.1.1. Our technical approach for solving the many challenges we faced, given the limited experience our team had with the system required to create the product, involved the use of free online resources and coordinated efforts to handle any issue from all sides.

9.1.2. Efficient Planning:

9.1.2.1. The structure of the development cycle works in such a way where incremental deadlines are created by clients and agreed upon by developers. Clients can expect small yet impactful iterations of our project that meet requirements set by the clients for the deliverable. Through this method, there has never been an issue that took too long to resolve and involves minimal risk of reaching a deadline without a shippable product increment.

9.1.2.2. Our planning structure allows the product to be as adaptive as it needs to be given the shifting schedules of our developers along with any unexpected roadblocks we face along the way. Should a deadline need to be pushed back, it will not risk a cascading of our entire development cycle. Even in this case, other areas of the project will likely see continuous improvement regardless of difficulties occurring in other branches.

9.1.3. Django Framework:

9.1.3.1. The collective Django experience that our team started out with was minimal, but we approached this learning curve with transparent communication, assigning responsibilities based on experience, and sharing of documentation in order to learn and grow as the project does. The use of online tutorials and resources is essential to the development of our product and assists in finding several strategies to move forward with around difficult issues.

9.1.3.2. As development continues, so does the complexity of the project, so we mitigate the difficulties team members might have understanding another's work by documenting each of the steps taken and clearly outlining the purpose of every functionality. These must be accessible to all members of the project development team and must meet group specifications in order to be reviewed and accepted.

9.1.4. End-To-End Programming:

9.1.4.1. In order to implement full stack development efficiently, especially given our limited related experience, we took the approach of outlining all of the requirements that we would need in every area of the project. That is, we considered all that we would need to assist frontend development, database implementation and management, hosting services and backup servers, system security, team communication and sharing of resources, project design and outlining, backend programming, and streamlined implementation of all of the above. In doing so, we expect to mitigate any unanticipated problems and always have a next step to take during development.

9.1.4.2. We continuously test different areas of the project during development and gather feedback from team members and potential users to identify and address issues before they come to us. Each component of the implementation has tests created for it

to ensure that every component does what it should do, and does not do what it should not do.

### 9.2. Continuous Integration/Continuous Deployment (CI/CD)

9.2.1. Github Actions will be utilized to develop the CI/CD platform.

## 10. Client Liaison Requirements

### 10.1. Communication Plan

10.1.1. There will be weekly emails and monthly video calls with stakeholders to update on the progress of the project and any oversights that may occur. All developers and other stakeholders must be present for these meetings to ensure that there are no inconsistencies or discrepancies.

10.1.2. Urgent issues related to the project or any stakeholders would consist of a phone call or direct message from the Project Manager to the concerned party.

10.1.3. All stakeholders will have access to the project website and documentation to allow them to check in personally on progress if desired.

### 10.2. Reporting Structure

10.2.1. Reports to the relevant stakeholders would consist of live demonstrations and graphics.

### 10.3. Requirement Validation

10.3.1. We will showcase a checklist of progress to be communicated following the communication plan.

### 10.4. Change Management Process

10.4.1. Any stakeholder or individual involved who wishes to suggest a change to these requirements after the freeze date must submit a formal Change Request to the original project manager or designated authority. Upon the project manager receiving the request, the project team will review the proposed changes for feasibility, impact, and alignment with project objectives and consult relevant stakeholders as necessary. After review, the change request will either be approved or rejected and all changes will be documented. Finally, the party who submitted the change request will be notified.

10.4.2. In rare circumstances, urgent changes may need to be made to the requirements specification without following the full change request process. Such changes can only be made with the approval of the project manager and must be documented.

### 10.5. Client Feedback Integration

10.5.1. Any formal feedback on development desires that would not warrant a formal change request will be submitted in writing to the project manager for review.

## 11. State/Mode Requirements

### 11.1. State Transition Requirements

11.1.1. Loading a Webpage:

11.1.1.1. When clicking on any of the navigation bar buttons, users will encounter a waiting period while the website redirects to and loads the correct page.

11.1.2. Task Manager Page:

11.1.2.1. Filtering Tasks:

11.1.2.1.1. Upon clicking the filter tasks button, a menu will display next to the button with selectable checkboxes for each of the available categories. Only one of these boxes may be checked at a time and the user can click outside of the menu, or, click the filter tasks button again to close it.

11.1.2.2. Task Updates:

11.1.2.2.1. Clicking On a Task:

11.1.2.2.1.1. When the user selects a task from the monthly, weekly, or daily calendar, a form will appear, presenting the task's details such as its deadline, priority, and description. Additionally, options to edit or delete the task will be available.

11.1.2.2.1.2. Edit Task:

11.1.2.2.1.2.1. When selecting the 'Edit Task' button, the form will update to display fields for editing the task. During editing, users can either cancel the operation to return to the details display or save their changes to refresh the page and update the task information. Clicking outside the form to close it is disabled during editing.

11.1.2.2.1.3. When the user clicks outside the "Edit Task" form, it will disappear, revealing the main task manager web page.

11.1.2.2.2. When the user adds, edits, or deletes a task, the webpage will refresh and update the displayed information to reflect their saved input.

11.1.3. Login Webpage:

11.1.3.1. When the user attempts to log in, the webpage will automatically redirect to the task manager page upon a successful attempt. If authentication fails, user will be redirected to the same page for logging in.

11.1.3.2. Create Account Button:

11.1.3.2.1. When users choose the 'Create Account' option on the login web page, a form will appear with fields for entering an email, username, and password. There will also be a 'Create Account' button for finalizing the account creation.

11.1.3.2.2. EMail Verification:

11.1.3.2.2.1. Upon successfully finalizing account creation, the 'Create Account' form will be replaced with text instructing the user to check their email and click a verification link. After verification, the previous page will confirm their email verification status.

11.1.3.3. When the user is logged in, the header “Login” button will be replaced with their username. This page will offer them the option to edit notification settings through checkboxes, to log out, and to change their password.

11.1.3.3.1. Change Password:

11.1.3.3.1.1. Selecting the change password option will display a box containing fields to enter their current password and a new one. Upon successful entry, text will replace the box informing them of a successful change.

11.1.3.4. Forgot Password:

11.1.3.4.1. Clicking the forgot password button will prompt the user with fields to enter their username and corresponding e-mail. The e-mail will redirect them to a page with a field to enter a new password. After entry, they will be redirected to the login page again.

11.1.4. Report Generator:

11.1.4.1. On-click, the report generator button will display text informing the user that they should not leave the web page and that it is generating their data, and upon generation, it will display text informing the user of a download in progress along with starting to upload the report to their computer.

11.1.4.2. If the report fails to generate, the download fails, or if the download is canceled, the web page will display text informing the user that report generation has failed.

11.1.5. Change Website Theme:

11.1.5.1. Clicking the header button labeled “Theme” will open a drop-down menu with the option to change the color theme of the website. Clicking the button again will close the menu.

## 11.2. Mode-Specific Requirements

11.2.1. Normal Mode:

11.2.1.1. In normal operation mode, the system will have all header buttons leading to the correct web pages, will handle user processing and create account functions, will display editable task information, and will perform all of its intended functionality. Failure to load webpages due to system errors or bugs will result in 404 page.

11.2.2. Debug Mode:

11.2.2.1. In Django's admin settings, it is possible to enable a debug mode where 404 pages are replaced with a page explaining the kind of error that the system caught, the paths to the files involved, and various lines explaining internal details about the error. All other 'normal' functionalities of the website will operate the same. The purpose of this mode is to test new features and see clear descriptions as to why they failed.

## 11.3. State/Mode Tests

11.3.1. For each of these tests verify that the correct boxes and forms appear where they should along with verifying that they work in both debug and normal modes.

11.3.2. Attempt to create a new account.

11.3.3. Attempt to log out.

11.3.4. Attempt to forget password function.

11.3.5. Attempt to log back in to verify the changed password.

11.3.6. Go to the newly displayed user information page:

11.3.6.1. Attempt to change password.

11.3.6.1.1. Log out and log back in to verify the changed password.

11.3.6.2. Click notification boxes to see if they correctly display checkboxes.

11.3.7. Attempt going to each of the various web pages in the header row.

11.3.8. Attempt to create several tasks and verify that they display on all three calendars (if applicable for its date).

11.3.9. Attempt to edit task information and verify that it moves, deletes, and changes correctly.

11.3.10. Attempt to generate a report:

11.3.10.1. Cancel the download and verify the web page displaying error message text.

11.3.10.2. Allow the download to complete and verify contents of the report file.

# 12. Project Planning Requirements

## 12.1. Resource Allocation

12.1.1. Set up a Discord as a means for communication, document sharing, and keeping track of the overall project.

12.1.2. Host and record meetings over Zoom to ensure that even team members not present for meetings will not be behind on the team's agenda. Ensure these recordings are shared with the other team members.

12.1.3. Host, distribute, and organize tasks over Asana. Would later be transferred to google sheets for task management and calendar organization.

12.1.4. Ensure the implementation of a Github repository for the purposes of updating and hosting files.

12.1.5. Ensure all documents necessary for the completion and development of the CrabbyTasks applications are shared with all primary stakeholders.

## 12.2. Scope Change Management

12.2.1. Schedule a meeting or discussion if the meeting is already planned to decide whether or not to proceed with the scope change. Project Manager will have final say as to how the group proceeds based on the cost and benefits the change will have for the application.

## 12.3. Maximum Hours Limit

12.3.1. Must provide developers with tasks that can be completed in less than 5 hours and the time to meaningfully complete these tasks.

## 12.4. Assigned Tasks

12.4.1. Tasks will be assigned based on project role, with each team member taking up a required responsibility befitting of the respective role. These tasks are managed and controlled through Asana by the planner and project manager.

## 12.5. Calendar

12.5.1. Developers and other stakeholders must have access to the calendar located on Asana to track progress.

12.5.2. The calendar should specify completion and due dates for all tasks.

# 13. Project Group Management Requirements

## 13.1. Management Strategy

13.1.1. The management strategy for the Red Lobsters involves communication over discord and meetings over zoom, as well as the establishment of deadlines and necessary tasks tracked and documented over Asana and a customized spreadsheet. Throughout each meeting management looks ahead at the barriers we need to climb, sets our goals for the future, and answers group questions that are essential to complete the application. On our weekly Tuesday meetings we establish the goals for the week and create any new tasks, after which we plan for times to have individual or full group meetings.

## 13.2. Conflict Resolution Strategy

13.2.1. Any conflict occurring between members of the Red Lobsters will be met with a private discussion with the project manager and all involved in the conflict itself. Ultimately these meetings are meant to resolve conflict and preserve a healthy and happy work environment.

## 13.3. Proactive Tasks of Management

13.3.1. Polling for Problems

13.3.1.1. The planner and the project manager will periodically check in with the other group members throughout the week. These check ins can occur through out of class meetings or through simple discord or text messages. They ultimately serve to benefit the group so that we may understand our current progress and tackle problems early. Group members who may have a tougher workload may reach out to other members who may not have anything scheduled in order to help. It falls on the project manager to ensure that the project is completed and that all deadlines are met.

13.3.2. Schedule Slippage

13.3.2.1. If we run into a situation where we are behind or an unforeseen issue occurs it is the responsibility of the project management and the planner to best assess the situation. The team must measure the feasibility, importance, and how many hands are necessary to complete the project. If the task is unimportant and or infeasible the project is ultimately dropped and the group moves on. Otherwise the team assesses the necessary number of people and finds places in the schedule to assign the respective tasks. Conversely when the group is ahead we may then reallocate resources in weaker areas to ensure a well rounded final product.

13.3.2.2. It is then the responsibility of the planner and management to understand how the slippage occurred and how best to prevent something like it from happening again.

### 13.4. Progress Reports

13.4.1. On Tuesdays the Red Lobsters have their mandatory group meeting. During these meetings as well as during other meetings and progress updates throughout the week team members are free to share the progress that they have made and their goals for the future of the project.

### 13.5. Accountability Model

13.5.1. The project manager or someone on their behalf must be there to monitor each group meeting and ensure that each team member is working on their necessary tasks. Team members are also to share updates on progress as well as what they might have been working on in the group Discord. Members of the Red Lobsters are also required to fill in any hours spent on the project in a time log to ensure that we know exactly who worked on what for how long.

### 13.6. Acknowledgement Model

13.6.1. During each mandatory Tuesday meeting the project manager acknowledges the hard work that has been completed during the prior week. This helps boost morale as the team is acknowledged for their hard work and dedication. For individual tasks and modules there is also feedback given both over the Asana and through the spreadsheet.

### 13.7. Agenda Management

13.7.1. The team at Red Lobster will be utilizing Asana and Google Sheets in order to keep track of our tasks and set deadlines. Discord is used for communication and keeping track of important dates and documents. Zoom is used for meetings, this is to ensure our meetings are recorded.

## 14. Project Role Specification

### 14.1. Project Manager: Austin Truong

14.1.1. Project Managers are responsible for delivering resources to team members and overall makes decisions for the group and the project at large.

14.1.2. Project Manager is the bottom line for the project as a whole, it is their responsibility to solve issues for the project should they arise.

### 14.2. Implementation Manager: Julian Hernendez

14.2.1. Implementation Managers are responsible for implementing the design for CrabbyTasks and ensure that the application runs the day it is due.

14.2.2. Implementation Managers ensure that the code is up to the specifications laid out by the developers and their clients

### 14.3. Collaboration Software Manager: Benny Chen

14.3.1. Collaboration Software Managers are responsible for getting users onboard and familiar with meeting and collaboration software.

### 14.4. Video/Media Manager: Emma Nunnink

14.4.1. Video/Media Managers are responsible for creating fully edited videos representing CrabbyTasks to investors and other stakeholders to show off overall progress.

14.4.2. Video/Media Managers are responsible for keeping backups ensuring there is a video ready to submit when necessary.

### 14.5. Planner: Penny McMahon

14.5.1. Planners are responsible for overall plan and for organizing schedules and deadlines for meetings

14.5.2. Planners should continually check in with other group members on tasks and monitor Asana to ensure tasks and goals are met.

### 14.6. Webmaster: Julian Hernandez

14.6.1. Webmasters are responsible for creating and hosting the website for all of our materials

### 14.7. Documentation Manager: Graciela Jimenez

14.7.1. Documentation Managers are responsible for documenting the project in the project book

14.7.2. Documentation Managers ensure group members have filled out time logs so that easy documentation of time spent is taken into account.

### 14.8. Requirements Manager: Penny McMahon

14.8.1. Requirements Managers are responsible for the requirements sheet and matrix

14.8.2. Requirements Managers ensure that team members understand and adheres to these requirements

### 14.9. Design Manager: Qi Deng

14.9.1. Design Managers are responsible for the overall design of CrabbyTasks from which the code will be built.

### 14.10. Unix System Programmer: Carlos Tapia

14.10.1. Unix System Programmers are responsible for handling our unix systems, as well as overall assisting with implementation and coding

14.10.2. Must have a strong technical background and have knowledge of Unix systems

### 14.11. Testing Manager: Nick Dilallo

14.11.1. Testing Managers are responsible for testing CrabbyTasks and ensuring it works without error or any bugs.

14.11.2. Testing Managers also ensure requirements are strong and testable against bad-devs or bad-clients.

### 14.12. Presentation Manager: Emma Nunnink

14.12.1. Responsible for presenting CrabbyTasks and creating the presentation showing off the culmination of our work in a limited time.

14.12.2. Ensures the presentation has everything secondary stakeholders and investors may want to see.

## 15. Stakeholders

### 15.1. The creators of this application are the Red Lobsters

15.1.1. Team consisted of Austin Truong, Benny Chen, Carlos Tapia, Emma Nunnink, Graciella Jimenez, Julian Hernendez, Nick Dilallo, Penny McMahon, and Qi Deng.

### 15.2. Primary stakeholders of this application are XYZ Corporation and investors.

### 15.3. Secondary stakeholders and investors of this application are Clark Elliot and DePaul University.

## 16. User Profile Requirements

### 16.1. Skill Set

16.1.1. Users must understand English

16.1.2. Users must have basic knowledge of web browsing, calendars, and technological comprehension.

### 16.2. Training

16.2.1. There will be a basic user tutorial available for creating accounts, creating tasks, using the calendar, and other useful features, which will be available in our application.

### 16.3. User Types

16.3.1. We expect many different kinds of users to use and interact with our program, including students, businesses, and hobbyists. Basically anyone who needs to keep track of their day to day activities. They will be required to meet the initial Skill Set Requirements.

# 17. Freeze Date

## 17.1. The official freeze date is October 26, 2023.

17.1.1. This means that no changes will be made to the document. There are situations that arise where changes may be necessary, but for those to make it onto the Requirements Sheet, they will need to undergo the process entailed below.

## 17.2. Change Request Process

17.2.1. All change request will be processed as noted in 10.4.1 and 10.4.2

# REQUIREMENTS MATRIX

## Functional Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 1.1.1.1 | 0 | Provide three distinct and aesthetically pleasing calendar views: monthly, weekly, and daily. | The client may claim that the aesthetics are not pleasing. | Ok. | No | Julian Hernandez, 09/27/2023 |
| 1.1.1.1 | 1 | Provide three distinct and visually cohesive calendar views: monthly, weekly, and daily. | The client may claim that the three calendar views are not visually cohesive. | Ok. | No | Austin Truong, 09/28/2023 |
| 1.1.1.1 | 2 | Application will have the capacity to provide 3 district and thematically cohesive monthly, weekly, and daily calendar views, such that the overarching theme captures cretaceous sea life through it's red colors, textures, and images; While enabling simple, organized, viewable representation of at least 3 tasks relative to their respective type of view. | Ok. | Ok. | Yes | Carlos Tapia, 10/03/23 |
| | | | | | | |
| 1.1.2.1 | 0 | Automatically populate all three calendars with tasks that users have added to the task manager site. | Change in wording about the calendars may cause the client to be confused about which calendars are being talked about. | Change in wording about the calendars may lead the dev to populate different calendars than the ones intended. | No | Julian Hernandez, 09/27/2023 |
| 1.1.2.1 | 1 | Automatically populate all three calendar views with tasks that users have added to the task manager site. | Error might seen when attempting to populate one of the calendars if the desired date is not in range of another calender | Error might seen when attempting to populate one of the calendars if the desired date is not in range of another calender | No | Austin Truong, 09/29/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.1.2.1 | 2 | The application will automatically populate monthly, weekly, daily calendar views in an organized and uniform way relative to the type of view that is currently presented to the user while also maintaining task functionality, thematic cohesiveness, and system integrity. | Ok | Ok | Yes | Carlos Tapia, 10/03/23 |
| | | | | | | |
| 1.1.3.1 | 0 | Enable users to access comprehensive task details by clicking on the task information section in any of the three calendars. | Change in wording about the calendars may cause the client to be confused about which calendars are being talked about. | Change in wording about the calendars may lead the dev to give the functionality to different calendars than the ones intended. | No | Julian Hernandez, 09/27/2023 |
| 1.1.3.1 | 1 | Enable users to access comprehensive task details by clicking on the task information section in any of the three calendar views. | The task details were not visible enough. | There was no specification on the font that will be used for the task details. Wording is vague and too general about "task information section" on any of the 3 views. That could mean a whole information panel of all tasks, a single individual task panel, etc. | No | Austin Truong, 09/29/2023 |
| 1.1.3.1 | 2 | Upon interaction with any specific task component, application will dynamically update task view to fit a view height and width capable of holding a 1.2 rem task title and 1 rem task description with enough padding to also hold any additional feature flags and descriptors. | Ok | Ok | Yes | Carlos Tapia 10/3/23 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.1.4.1 | 0 | Upon accessing the task information section, users will find a range of task management options, including: Adding new tasks, Deleting tasks, Editing task details, Setting task priorities, Assigning task categories, and, Opting to receive email notifications for task updates | Without specifying that the task categories are assigned based on colored flags, the client could claim that the ability to set categories was not provided as promised. | Without the task manager page being specified, the dev may place these options wherever they deem to be the 'task information section'. | No | Julian Hernandez, 09/27/2023 |
| 1.1.4.1 | 1 | Within the task manager page, users will find various task management options, including adding new tasks, deleting tasks, editing task details, setting task priorities, assigning task categories, and opting to receive email notifications for task updates. | Ok. | Ok. | Yes | Austin Truong, 10/02/2023 |
| | | | | | | |
| 1.1.4.2 | 0 | Additionally, a dedicated "Add Task" function will be available in the daily, weekly, and monthly calendar views for user convenience. | Ok. | Ok. | | Julian Hernandez, 09/27/2023 |
| | | | | | | |
| 1.2.1.1 | 0 | On the user login page, users will be provided an option to create a new account if not currently logged in. | Ok. | Ok. | | Julian Hernandez, 09/27/2023 |
| | | | | | | |
| 1.2.1.2 | 0 | Prompt users to enter their preferred username and password for future access. | Current wording could imply that the username and password to be entered will have been previously created. | Ok. | No | Julian Hernandez, 09/28/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.2.1.2 | 1 | Prompt users to create a preferred username and password for future access. | Ok. Passes bad client attack. | Ok. | Yes | Julian Hernandez, 09/28/2023 |
| | | | | | | |
| 1.2.2.1 | 0 | Allow users who have just created an account or who already have an existing account to log in. | The client may attempt to log in with incorrect information and claim that the requirement is not functional. | The dev may implement a different method other than authenticating their login information to allow users with accounts to log in, which is unintended. | No | |
| 1.2.2.1 | 1 | Allow both new and existing users to log in with their correct username and password. | Ok. | Ok. | Yes | Julian Hernandez, 09/27/2023 |
| | | | | | | |
| 1.2.2.2 | 0 | Upon successful login, redirect users to the task manager page. Ensure that task information is automatically saved after each edit for the user's convenience. | The separate sentences could imply that the task information is automatically saved whether or not the user is logged in. | The redirection to the task manager page may not be automatic. | No | Julian Hernandez, 09/27/2023 |
| 1.2.2.2 | 1 | Upon successful login, automatically redirect users to the task manager page and ensure that task information is saved after each edit for user convenience. | User was not notified when the task/information was saved | The login did not redirect the user to the task manager page. There should be an on-screen notification after the reroute that their changes/edits were auto-saved. | No | Austin Truong, 09/29/2023 |
| 1.2.2.2 | 2 | Upon successful login, Application will redirect users to task manager page which contains the task dashboard and auxiliary task manipulation functions; Only then, if the user manipulates any task present on the viewfinder, | Ok. | Ok. | Yes | Carlos Tapia, 10/3/23 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | the application will functionally handle all events related to acquiring, saving, and viewing of the manipulated task to lower mental user overhead. | | | | |
| | | | | | | |
| 1.2.3.1 | 0 | Offer a "Change Password" option within the login section for users who wish to reset their password. | Ok. | Ok. | Yes | Julian Hernandez, 09/27/2023 |
| | | | | | | |
| 1.2.3.2 | 0 | Require users to input their username and provide a preferred email address for receiving the password reset instructions. | Client may provide a separate e-mail from the one linked to their account and assume they are going to receive password reset instructions. | Dev may allow anyone with the username to send reset instructions to any e-mail which could cause security issues. | No | Julian Hernandez, 09/27/2023 |
| 1.2.3.2 | 1 | When resetting a password, require users to input their username and specify the corresponding email address in order to receive password reset instructions. | Ok. Passes bad dev attack. | Ok. Passes bad dev attack. | Yes | Julian Hernandez, 09/27/2023 |
| | | | | | | |
| 1.2.4.1.1 | 0 | When creating/add or editing a task, users will be offered an option to opt-in for e-mail notifications for that specific task. The option for e-mail notifications will be defaulted to "No". If opted-in for e-mail notifications, an email will be sent 2 days prior to the deadline of the selected task. | Client may desire different or a wider selection of times to choose from for notifications before the deadline since 2 days may be too little or too long. | Description does not specify where this function/option is to be implemented. | No | Julian Hernandez, 10/05/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.2.4.1.1 | 1 | When the user is on the Task Manager page and is adding or editing a task, an option will be offered to opt-in for e-mail notifications for that specific task. The option for e-mail notifications will be defaulted to "No". If opted-in for e-mail notifications, an additional dropdown or selection will be offered to choose how many hours/days prior to the deadline of the selected task the user wants to be notified via e-mail. | Ok. | Ok. | Yes | Austin Truong, 10/06/2023 |
| | | | | | | |
| 1.2.4.2.1 | 0 | Users will be offered an option to opt-in for desktop notifications | Client can claim that option was not on the correct page since there was no specified positioning. Client can also claim that they were not allowed to choose the amount of hours/days to be notified prior to the task deadline. | Function is not specified for a specific page and could possibly be implemented on another page without purpose. No default option is specified either, so the desired default option could be either one chosen by the developer. | No | Julian Hernandez, 10/05/2023 |
| 1.2.4.2.1 | 1 | When on the Task Manager page and the user is adding or editing a task, if the option for desktop notifications is selected, then an additional dropdown or selection will be offered to choose how many hours/days prior to the task deadline the user wants to be notified on desktop (if they are on/at the computer). The default for the desktop notification option will be "No". | Ok. | Ok. | Yes | Austin Truong, 10/06/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.2.4.3.1 | 0 | Users can select an approximate date and time to get notified through email. | Ok. | The type of notification should be specified or else the dev may implement irrelevant e-mails. | No | Julian Hernandez, 09/27/2023 |
| 1.2.4.3.1 | 1 | Allow users to select an approximate date and time to receive email notifications about their tasks. | Ok. | Ok. Passes bad dev attack. | Yes | Julian Hernandez, 09/28/2023 |
| | | | | | | |
| 1.2.5.1 | 0 | Allow users to select between light and dark modes for the theme of the website. | Ok. | Ok. | Yes | Penny McMahon, 9/26/2023 |
| | | | | | | |
| 1.2.6.1 | 0 | Edit tasks with priority preference. | It is not clear that the user has the ability to do this action. | Wording makes it sound like user's need to be able to edit all tasks with matching priority simultaneously, which is not intended. | No | Julian Hernandez, 10/05/2023 |
| 1.2.6.1 | 1 | On the task manager page and within each of the edit task sections and add task sections, allow users to modify the priority preference of any task and ensure that the changes are saved as well as represented across all views the tasks are present in. | Ok. | Ok. | Yes | Austin Truong, 10/06/2023 |
| | | | | | | |
| 1.2.6.2 | 0 | Edit tasks by catalog (colored flags). | The colored flags aspect needs to be explicitly stated or else the client may infer a different property than what we intended, which is for categories to be edited by colored flags. | Wording makes it sound like user's need to be able to edit all tasks with matching category simultaneously, which is not intended. | No | Julian Hernandez, 10/05/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.2.6.2 | 1 | On the task manager page, within each of the edit task sections and add task sections, enable users to assign categories to tasks using colored flags and ensure that the changes are saved as well as represented across all views the tasks are present in. | Ok. | Ok. | Yes | Austin Truong, 10/05/2023 |
| | | | | | | |
| 1.2.6.3.1 | 0 | Offer in the tasks catalog page with a function to order tasks by priority level. | Wording does not convey that the user is the one who selects the sorting option so the client could claim it is unpresent. | Wording does not clearly convey that there needs to be a functionality on the task manager page that allows users to sort tasks by priority level. | No | Julian Hernandez, 09/27/2023 |
| 1.2.6.3.1 | 1 | Provide a function on the tasks catalog page to allow users to order tasks by priority level. | Ok. | Tasks should have a default sort/order | No | Austin Truong, 10/02/2023 |
| 1.2.6.3.1 | 2 | The application will have a dedicated component in the viewfinder of the page where all complete, and incomplete tasks reside; Upon arrival of this page, tasks will be ordered by the date of which they were conceived; This component will allow the user to programmatically manipulate the order of which the tasks are painted on to the screen; Additionally, the component will not jeopardize the integrity of the system, thematic cohesiveness of the system, or responsiveness of the system; | Ok. | Ok | Yes | Carlos Tapia, 10/03/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.2.6.4 | 0 | Offer in the tasks catalog page with a function to order tasks by category (colored flag). | Wording does not convey that the user is the one who selects the sorting option so the client could claim it is unpresent. | Wording does not clearly convey that there needs to be a functionality on the task manager page that allows users to sort tasks by categories which are indicated by colored flags. | No | Julian Hernandez, 09/27/2023 |
| 1.2.6.4 | 1 | On the task manager page, implement a function that enables users to order tasks by category, which are indicated by colored flags. | Taskmanager page did not implement function. Client may argue that they are not allowed to customize their flag colors for any category they make. | The function did not direct the user. Every category created should have a default blank color flag and allow the user to change the flag color to whatever they want/select from a range of colors. | No | Austin Truong, 09/29/2023 |
| 1.2.6.4 | 2 | On the task manager page, a function will be implemented to allow users to order tasks by category using colored flags in a drop down. By clicking on the flag/category icon next to each task, a selection of colors will be offered to set for that flag's color. The default flag color will be grey to indicate no category. | Ok. | Ok. | Yes | Austin Truong, 10/05/2023 |
| | | | | | | |
| 1.2.6.5 | 0 | Offer in the tasks catalog with a function to order tasks by the time they are edited | Wording does not convey that the user is the one who selects the sorting option so the client could claim it is unpresent. | Wording does not clearly convey that there needs to be a functionality on the task manager page that allows users to sort tasks the last time they were edited, dev could implement sorting by random edit dates. | No | Julian Hernandez, 09/27/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.2.6.5 | 1 | In the task manager page, include a function that allows users to order tasks based on the time they were last edited. | The time was not recorded when the task was created | There was an error in the time zone | No | Julian Hernandez, 09/28/2023 |
| 1.2.6.5 | 2 | In the task manager page, include a function that allows users to order tasks based on the time, respective to the user's timezone, they were last edited or, if there are no previous edits, based on when it was created. | OK | OK | Yes. | Julian Hernandez, 10/03/2023 |
| | | | | | | |
| 1.2.6.6 | 0 | Offer in the tasks catalog with a function to order tasks by a range of dates entered into the filter. | Wording does not convey that the user is the one who selects the sorting option so the client could claim it is unpresent or that they did not know how to enter the range. | Wording does not clearly convey that there needs to be a functionality on the task manager page that allows users to sort tasks based on input of a range of time. | No | Julian Hernandez, 09/27/2023 |
| 1.2.6.6 | 1 | In the tasks catalog, incorporate a function that allows users to order tasks by a specified range of dates entered into the filter. | Dates were not consistent with the users request | The calendar was not updated to the correct year | No | Julian Hernandez, 10/03/2023 |
| 1.2.6.6 | 2 | In the tasks catalog, incorporate a function that allows users to order tasks by a specified range of dates, in which they are due, entered into the filter. The due dates of the tasks should be consistent with the user's request; Ensure the calendar is updated to the correct year as well. | Ok | Ok | Yes | Julian Hernandez, 10/03/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.2.7.1 | 0 | Complete tasks display. | Given that the requirement does not communicate the details of the display, client could claim that the display needed to be constantly viewable as opposed to being in it's own menu. | Does not concisely communicate that there needs to be an option for the user to view a display of their completed tasks. | No | Julian Hernandez, 09/28/2023 |
| 1.2.7.1 | 1 | Implement a clickable menu that displays a list of all of the user's previously completed tasks. | the clickable menu is not detailed enough to understand the function | Users were not notified when task was completed. Wording does not convey *where* or which page this functionality should/would be implemented on. | No | Austin Truong, 10/02/2023 |
| 1.2.7.1 | 2 | On the task manager page, implement a clickable button labeled "Complete Tasks" that opens a menu which provides a clear and detailed display of all of the user's previously completed tasks. | Ok. | Ok. | Yes | Austin Truong, 10/05/2023 |
| | | | | | | |
| 1.2.7.2 | 0 | Task Completion Score System. | Description does not provide anything more than the concept of the function. Client could claim that whatever system was implemented was nothing that they desired/asked for. | Does not communicate to the developer that the scoring system needs to be implemented and viewable by users. | No | Julian Hernandez, 09/28/2023 |
| 1.2.7.2 | 1 | Implement a viewable task completion scoring system. | Wording is very general, so client could claim that this system is not what they had in mind, or that the scoring system is bad. | Task completion/scoring system should have structure behind it, having a consistent amount of points for each scenario of completion: completed early, completed on time, completed late. | No | Austin Truong, 09/29/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.2.7.2 | 2 | A viewable task completion scoring system will be implemented in hopes of howing the user(s) progress and encourage them to omplete more tasks earlier. The scoring system will reward points upon task completion with an increasing amount depending on how much earlier the task was completed compared to its leadline. Conversely, points vill be deducted if tasks are ıot completed on time, with the amount taken away ncreasing as the amount of time *past* the deadline increases in order to make sers finish up late tasks and revent it from happening in the future. | Ok. | Ok. | Yes | ustin Truong, 10/05/2023 |

## System Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 2.1.1 | 0 | System will require Database capable of holding client demand on platform | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.1.1 | 0 | A secured AWS EC2 instance holding (MongoDB \|\| Postgresql \|\| DynamoDB) will have an exclusive channel of communication to the production server. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.2 | 0 | Server must concurrently handle multiple connections for several users. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| 2.1.2.1 | 0 | A secured AWS EC2 instance has Apache2 server handling incoming requests to the server | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.3 | 0 | System must be able to hold against security threats | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.3.1 | 0 | Ssh Protocol will be secured using the latest security standards | Ok | What are those standards? a bad dev could simply say they are using the best standards and instead of the actual best standards. | No | Penny McMahon, 10/20/2023 |
| 2.1.3.1 | 1 | Ssh Protocol will be secured using the latest verified security standards that are available to the group. | Ok | "Latest" does not necessarily mean the best fit for the developer's use-cases or requirements | No | Austin Truong, 10/21/2023 |
| 2.1.3.1 | 2 | Ssh Protocol will be secured using the latest verified security standards that are available to the group after comparing the pros and cons of each standard/version to make sure they work best for the project. | Ok | Ok | Yes | Austin Truong, 10/21/2023 |
| | | | | | | |
| 2.1.4 | 0 | System be provide a user space separate from other users | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.4.1 | 0 | Unauthorized users will not be able to access personal user pages. | A bad client could attempt to hijack someone's page by claiming that they simply lost their information thus being unauthorized. We need to ensure that this does not | What could be preventing devs from breaking into profiles and causing chaos? | No | Penny McMahon, 10/20/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | happen | | | |
| 2.1.4.1 | 1 | Unauthorized users will not be able to access personal user pages. The user must have valid credentials and have a verified method of communication to reset password if necessary. This includes developers unless necessary edits or bug fixes are needed and require a look into a user's page | Ok | Ok | Yes | Austin Truong, 10/21/2023 |
| | | | | | | |
| 2.1.5 | 0 | System must onboard users seamlessly | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.5.1 | 0 | Introducing users to the system will consist of an overview of how the application functions. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.6 | 0 | System must onboard developer seamlessly | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.6.1 | 0 | In the event that a new developer is introduced to the system, the developer will be introduced to the project manual and an environment image to help reduce overhead into the system. | Ok | how can we ensure that these new devs know what they are doing based on these resources. | No | Penny McMahon, 10/20/2023 |
| 2.1.6.1 | 1 | In the event that a new developer is introduced to the system, the developer will be introduced to the project manual and an environment image to help reduce overhead into the system. They will also be shadowed by or work with more experienced developers | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| 2.1.7 | 0 | System communication will be held over a trusted secured encrypted TCP connection | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.7.1 | 0 | Inbound connections to the server will be encrypted and be up to SSL standards. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.8 | 0 | Production server and database will establish secure isolated communication | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.8.1 | 0 | Database will only allow communication to the application server. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.9 | 0 | System will be hosted to the public at red-lobster.dev | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.9.1 | 0 | System will have its own Domain Name and be placed on public DNS. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 2.1.10 | 0 | Production Server and Databases will be hosted on Virtual Amazon EC2 Instances running Ubuntu 22.04 LTS (HVM) SSD Volume Drive on 64-bit Image | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |

## Technical Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| 3.1.1.1 | 0 | System design will allow for scalability of the system such that it will not impede the average speed of the system by more than 3 seconds from its initial development time. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3.1.1.2 | 0 | In the case of component failure, the system will prompt the user of the component failure and continue to operate without the failed component unless the component plays a pivotal role in the subsystems operations in which case the system will cease operations, and notify the user of the unexpected fault. | How would the client understand the error with little or no background knowledge in errors or error codes? | How can developers act quickly and be informed when this error occurs? | No | Penny McMahon, 10/20/2023 |
| 3.1.1.2 | 1 | In the case of component failure, the system will prompt the user of the component failure and continue to operate without the failed component unless the component plays a pivotal role in the subsystems operations in which case the system will cease operations, and notify the user of the unexpected fault and a description of what it may entail. The developers and those overseeing the site can be notified about the error when it occurs to ensure it is handled swiftly. | Ok | Ok | Yes | Austin Truong, 10/21/2023 |
| | | | | | | |
| 3.1.2.1 | 0 | In the case where a component in the system were to fail, the overall system would continue to operate unless the | How are clients notified during this process? | are devs notified of this failure or can they just safely ignore it? | No | Penny McMahon, 10/20/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | component in question plays an integral operational function to that system. | | | | |
| 3.1.2.1 | 1 | In the case where a component in the system were to fail, the overall system would continue to operate unless the component in question plays an integral operational function to that system. Users will be notified if the system fails to operate and developers and those overseeing the site are also notified to ensure that solutions are found. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 3.1.3.1 | 0 | During the development of the system, developers will maintain, develop, and execute system wide test cases for components of the system such that those tests will maintain the integrity of the component. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3.2.1.1 | 0 | Initial system benchmarks will be conducted through the use of profiling software that will test the speed of operations. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3..2.1.2 | 0 | Initial system benchmarks will be used as a baseline comparison to functionally equivalent component substitution decisions. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3.2.1.3 | 0 | Speed of the system will not surpass 4 seconds for any queries that involve retrieving information with the exception of | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | infrastructure that is outside the governed scope of the application and artifacts outside a typical client use case. | | | | |
| | | | | | | |
| 3.2.1.4 | 0 | Any component or system that does not meet expectations of any party will be audited and reviewed for potential improvements. In the case where a potential replacement to the component exists, there will be test cases created and design analysis to support the new components integration to the system(s). In the event that there does not exist a better alternative to the component in question, the party responsible for auditing the reviewing the component will discuss why this component is the current best possible solution to the problem. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3.2.2.1 | 0 | Both the developers and consumers understand and agree that the application will be capable of scaling to more than 100,000 users and will have a plan to allow improved infrastructure to hold previously stated levels of volume. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3.2.2.2 | 0 | During the development of the application it is reasonable to test system load of 10 different users running different processes. Each of those 10 users will have flawless networking experiences while using the | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | application or any component of that system with the exception of any network or environment problems outside the govern scope of the application. | | | | |
| | | | | | | |
| 3.2.2.3 | 0 | During the development of the system, the developers will test the system under fictitious levels of network volume to impersonate varying levels of forecasted volume that could potentially be reached. | Ok | what would comprise these fictious levels, as a bad dev could theoretically define anythingas fictious and not really try | No | Penny McMahon, 10/20/2023 |
| 3.2.2.3 | 1 | During the development of the system, the developers will test the system under fictitious levels of network volume to impersonate varying levels of forecasted volume that could potentially be reached based on research about system overloading. This may include an unreasonable surge in userbase. | Ok | Ok | Yes | Austin Truong, 10/21/2023 |
| | | | | | | |
| 3.2.3.1 | 0 | The developers of the application will use testing software and suites that allows for testing the application's speed, efficiency and responsiveness under test cases presented in the testing matrix maintained and filled by developers and the testing manager. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3.3.1.1 | 0 | Under initial development of the application, Data exchange will be handled through the base backend framework Django. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| 3.3.1.2 | 0 | If time allows, Django will be transformed into an API that follows the REST protocol. Data exchange between frontend and backend will then be left to developers discretion but must still be fully functional and maintain all the functionality that the application originally maintained while being coupled under Django. | What are the actions for if the client is unsatisified with the change | Ok | No | Penny McMahon, 10/20/2023 |
| 3.3.1.2 | 1 | If time allows, Django will be transformed into an API that follows the REST protocol. Data exchange between frontend and backend will then be left to developers discretion but must still be fully functional and maintain all the functionality that the application originally maintained while being coupled under Django. If the client is unsatisified with the change there is an option to go back to utilizing Django. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 3.3.2.1 | 0 | Passwords will be encrypted using MD5 Hashing to ensure no possibility of client information ever being exposed. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3.3.2.2 | 0 | SQL Injections will be mitigated using Django Object Relational Mapping. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3.3.2.3 | 0 | Cross Site scripting attacks will be mitigated using information sanitizers. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 3.3.2.4 | 0 | Cross Site Forgery attacks will be mitigated using CSRF Tokens. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3.3.2.5 | 0 | Session Hijacking attacks will be mitigated using secured encrypted sessions and refreshed session cookies. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |
| 3.3.2.6 | 0 | Access to resources will follow the least minimum access model. | Ok | Ok | Yes | Penny McMahon, 10/20/2023 |
| | | | | | | |

## Documentation

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 6.1.1.\\1 | 0 | The vision behind the creation of this product is focused on making the invaluable benefits of task-tracking software available for any workplace to implement and improve the productivity of their team. Ultimately, this is a system that fosters punctuality by rewarding the user for completing tasks and encouraging them to see out their goals through the use of in-app incentives and disincentives for completion. With many functionalities that support various styles of planning and categorizing, Crabby Tasks is built to be able to incorporate the scheduling of any kind of work or lifestyle. The application allows the user to have a | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | piece of mind where all tasks are recorded and tracked as well as encourage punctuality and accomplishment. Teams will also benefit from Crabby Tasks as it offers a team collaboration feature. | | | | |
| | | | | | | |
| 6.1.2.1 | 0 | This project supplies a fully functioning task list in the form of a web application built end-to-end by our team, ready for full use by any number of teams, workers, or individuals. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.1.2.2 | 0 | The source code of this project will be supplied on GitHub for review and reference. | Ok | who is the person reviewing or referencing this? a bad dev could send this to basically anyone and get confirmation | No | Penny McMahon, 10/18/2023 |
| 6.1.2.2 | 1 | The source code of this project will be supplied on GitHub for review and reference from Crabby Tasks' developers and other stakeholders. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 6.1.3.1 | 0 | Crabby Tasks is built for task management, schedule organization and cohesion, and record keeping. The project accomplishes this goal by providing users with many different ways to view their daily, weekly, and monthly schedules along with structuring their tasks in such a way that allows them a full, unabated view. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 6.1.3.1.1 | 0 | This includes only the listed functional requirements listed in Section 1. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.1.3.2 | 0 | Crabby Tasks was built for usage and browsing on a desktop web browser and does not have support/functionality for any other platforms. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.1.3.3 | 0 | The task manager can manage the tasks of several users at once, but it does not support several accounts being logged in simultaneously to a single instance at a time. Any shared usage of Crabby Tasks must be on a single account in order for the tasks to be viewable at the same time. The only way to accomplish this is with separation by flags, there is no other internal user switching functionality. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.1.3.4 | 0 | Crabby Tasks is intended to be used with a working network connection. Any loss of signal will not store unsaved changes if the web page is exited and no pages or edits will load while offline. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.1.4.1 | 0 | The team behind Crabby Tasks is composed of college students with little money and time to spare which slows down the development schedule considerably. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| 6.1.4.2 | 0 | There is an expected learning curve behind developing the project in Django, setting up a database, server, and backups, and with building an engaging and functional frontend which could affect the types of features supported by the project. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.1.5.1 | 0 | We believe that our team will be able to learn how to implement and thus provide a fully functional web application with all of the features and requirements laid out in this document. | A bad client may not believe in that resolve so we may need to use more prescise wording | A bad dev might not necessarily believe that so it is important to be precise | No | Penny McMahon, 10/18/2023 |
| 6.1.5.1 | 1 | Our team will be able to learn how to implement and thus provide a fully functional web application with all of the features and requirements laid out in this document. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 6.1.5.2 | 0 | We believe that development of the project will be able to be split amongst the nine members of our team throughout its lifecycle. | A bad client once again may not believe what we believe so it is important to be precise | A bad dev once again may not believe what we believe so it is important to be precise | No | Penny McMahon, 10/18/2023 |
| 6.1.5.2 | 1 | Development of the project will be able to be split amongst the nine members of our team throughout its lifecycle. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 6.1.6.1 | 0 | In order for the project to be considered complete and successful, it must provide each of the functional requirements listed in Section 1 and those requirements alone. Any unexpected changes to the development cycle of Crabby Tasks between the finalizing of this document and the shipping of the finished product will be considered inconsequential to the overall completion of the project. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 6.2.1 | 0 | This document is structured with each of the main sections designated by a single whole number starting from Section 1 and continuing until Section 17. Every section will have corresponding subsections that may have their own subsections as well, and each of these will be numbered incrementally respective to the other sections in their layer with the format X.Y.Z. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.2.2 | 0 | This document has a table of contents and sections for an introduction, overview, business objectives, a risk management table, and a conclusion. These sections will not be numbered. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 6.3.1 | 0 | This project, and thus this document, is targeted towards workplaces and individuals of any age or demographic who wish to foster healthy organizational and time management routines. Crabby Tasks is meant for those who seek support or relief from part of the mental strain that a loaded schedule can bring; Taking it off of the user's shoulders and putting it into the app instead. This methodology encourages a healthy compartmentalization of work and gives punctuality a visually rewarding enhancement, making it useful for anyone who has access to a desktop and wishes to manage their schedule efficiently. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.3.2 | 0 | This document is managed and overseen by the members of our team. Each person contributes and each person reviews new additions to make sure that a cohesive and professional tone is maintained. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.3.3 | 0 | The versioning of this document is stored by Google Docs and details the date, time of day, editor, and specifics of individual changes. | Ok | there can be situations where devs only view anonymous links and are therefore unaccounted for in the changelog. | No | Penny McMahon, 10/18/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 6.3.3 | 1 | The versioning of this document is stored by Google Docs and details the date, time of day, editor, and specifics of individual changes. To ensure this we will avoid anonymous links and personally send this and other important documents through gmail. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 6.4.1 | 0 | The documentation of the project will reflect the outline of the project booklet. As well as the descriptions and manuals of each requirement specification. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.5.1 | 0 | All citing sources will be referenced by the project booklets manual. | A reference may not be enough for some clients | a reference may not be enoguh for some clients and bad devs can reference something without actually properly citing it. | No | Penny McMahon, 10/18/2023 |
| 6.5.1 | 1 | All sources will be referenced and properly cited by the project booklet's manual to avoid any issues that could arise down the line. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 6.6.1 | 0 | Clientele or interested parties who have viewed this document and wish to contact our team can reach us at RedLobsterTeamBusiness@gmail.com | Ok | How can we guarantee that we have someone on the dev team that can receive these messages? How can we make sure that the developer reviewing the messages is doing a proper job? | No | Penny McMahon, 10/18/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 6.6.1 | | Clientele or interested parties who have viewed this document and wish to contact our team can reach us at RedLobsterTeamBusiness@gmail.com, where at least one of our developers will look over the feedback. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 6.7.1 | 0 | The phrase or title known as “Crabby Tasks” in this document refers to the web application that our team is building and intends to deliver to our client. Any mention of this phrase is referring to the whole end-to-end of the web application as its own entity. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.7.2 | 0 | Django is a high-level Python web framework that our team used to develop the website for our project. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.7.3 | 0 | The web ‘Application Programming Interface’ we will be using to better serve the frontend of our project if time allows. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 6.7.4 | 0 | Refers to Amazon Web Services Elastic Compute Cloud. It is one of the core services offered by Amazon Web Services (AWS), which is a cloud computing platform provided by Amazon. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |

## Design Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 7.1.1.1 | 0 | System will be able to be developed for several years. | Ok | A developer could delay developement using this requirement | No | Penny McMahon, 10/16/2023 |
| 7.1.1.1 | 1 | System will be able to be developed and maintained for several years, but will be adhering to all deadlines set by all stakeholders. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 7.1.1.2 | 0 | New developers introduced to the system will be able to develop new features as long as they have read and understood the project manual. | Ok | Ok | Yes | Penny McMahon, 10/16/2023 |
| | | | | | | |
| 7.1.2.1 | 0 | Initial development will only handle development team. | Ok | Requirement simply not clear enough | No | Penny McMahon, 10/16/2023 |
| 7.1.2.1 | 1 | Initial development will only be scalable to handle the original development team. Further and extra development would be required if there was a necessity to handle larger capacities. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 7.1.2.2 | 0 | There will be a plan in place for handling thousands of users. | Ok | Simply not concrete enough to work with | No | Penny McMahon, 10/16/2023 |
| 7.1.2.2 | 1 | There is a plan in place to allow the system to scale and handle thousands of users on our platform. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 7.1.3.1 | 0 | System must be stable on the current release. | Ok | Stability does not always guarantee running, a developer could deliver a stable but incomplete product on release. | No | Penny McMahon, 10/16/2023 |
| 7.1.3.1 | 1 | System must be stable and running on the current release. All features promised on release will be included in the released product. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 7.1.3.2 | 0 | In the event that there is a system failure, there will be a backup server that can handle requests to reduce overall downtime. | Ok | Ok | Yes | Penny McMahon, 10/16/2023 |
| | | | | | | |
| 7.1.4.1 | 0 | Requests to the systems will be handled efficiently for the initial release | Ok | A developer's definition of efficient can vary, leading to a product that does not have a set concrete benchmark for handling requests | No | Penny McMahon, 10/16/2023 |
| 7.1.4.1 | 1 | Requests to the systems will be handled efficiently for the initial release such that they will not have a slow enough response time to warrant any complaints or issues. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 7.1.4.2 | 0 | There will be a plan in place for optimizing server requests through the use of code refactoring and caching systems. | Ok | Developers might have different understandings or definitions of what "optimiziation" is and can refactor according to it, but the end result may not actually be "optimized". | No | Austin Truong, 10/26/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 7.1.4.2 | 1 | There will be a plan in place for optimizing server requests through the use of code refactoring and caching systems such that requests and responses are not slow enough to be considered an issue. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 7.1.5.1 | 0 | The web server utilized will be Apache's HTTPD web server. | Ok | Ok | Yes | Penny McMahon, 10/16/2023 |
| | | | | | | |
| 7.1.5.2 | 0 | The client facing web application will utilize the Django templating engine to handle dynamic requests. | Ok | Ok | Yes | Penny McMahon, 10/16/2023 |
| | | | | | | |
| 7.1.5.3 | 0 | There will be a plan in place for future improvements to the client facing web-application such that they reflect on and consider both end user and developer feedback. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 7.2.1 | 0 | User Interface will be visually pleasing to end users | Client may argue that the UI is *not* visually pleasing to them. | Ok | No | Austin Truong, 10/26/2023 |
| 7.2.1 | 1 | User Interface will be visually pleasing to end users and be finalized after coming to an agreement with client/stakeholders. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 7.2.1.1 | 0 | Static web design of the application will be handled by Django templating engine. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 7.2.2 | 0 | User Interface will be simple and intuitive for end users | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 7.2.3 | 0 | User Interface will be modern and hold a theme | Client may argue that UI is not modern enough or at all. | Developer may have a different meaning or idea of what "modern" is. | No | Austin Truong, 10/26/2023 |
| 7.2.3 | 1 | User Interface will be modern and hold a theme based on research regarding common interfaces used for popular platforms and websites. | Ok | Ok | Yes | |
| | | | | | | |

## Testing Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 8.1.1 | 0 | Testing is measured by how the systems, components, and processes of CrabbyTasks meet our requirements and will go underway the moment that the goals, requirements, and scope of the application are defined. With our risk management plan we will be able to detect problems fast and early to contribute to a durable and secure application that poses no functional or security issues. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8.2.1 | 0 | To ensure that the requirements for which we build the base of our program are testable and meet a high threshold of quality. The Requirements testing plan is to first ensure that requirement meets our goals, apply a specific use case against the requirement, ensure that the requirement is unambiguous, check the scope of the requirement, finalize the requirement, perform and review test case, design code based off requirement, and finally test the code off the test case. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |

## Problem-Solving Strategy Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 9.1.1.1 | 0 | Our technical approach for solving the many challenges we faced, given the limited experience our team had with the system required to create the product, involved the use of free online resources and coordinated efforts to handle any issue from all sides. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 9.1.2.1 | 0 | The structure of the development cycle works in such a way where incremental deadlines are created by which clients can expect small yet impactful iterations of our project. Through this method, there | Deadlines can be placed in a way that could make developement impossible. | Developers could submit unfinished versions of the project for the sake of meeting a deadline, requirements must be met for these | No | Penny McMahon, 10/18/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | has never been an issue that took too long to resolve and involves minimal risk of reaching a deadline without a shippable product increment. | | meetings | | |
| 9.1.2.1 | 1 | The structure of the development cycle works in such a way where incremental deadlines are created by clients and agreed upon by developers. Clients can expect small yet impactful iterations of our project that meet requirements set by the clients for the deliverable. Through this method, there has never been an issue that took too long to resolve and involves minimal risk of reaching a deadline without a shippable product increment. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 9.1.2.2 | 0 | Our planning structure allows the product to be as adaptive as it needs to be given the shifting schedules of our developers along with any unexpected roadblocks we face along the way. Should a deadline need to be pushed back, it will not risk a cascading of our entire development cycle. Even in this case, other areas of the project will likely see continuous improvement regardless of difficulties occurring in other branches. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 9.1.3.1 | 0 | The collective Django experience that our team started out with was minimal, but we approached this learning curve with transparent communication, assigning responsibilities based on experience, and sharing of documentation in order to learn and grow as the project does. The use of online tutorials and resources is essential to the development of our product and assists in finding several strategies to move forward with around difficult issues. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 9.1.3.2 | 0 | As development continues, so does the complexity of the project, so we mitigate the difficulties team members might have understanding another's work by documenting each of the steps taken and clearly outlining the purpose of every functionality. | Ok | This documentation needs to be clear and accessible or else bad developers will simply make poor notes that may be hard to access | No | Penny McMahon, 10/18/2023 |
| 9.1.3.2 | 1 | As development continues, so does the complexity of the project, so we mitigate the difficulties team members might have understanding another's work by documenting each of the steps taken and clearly outlining the purpose of every functionality. These must be accessible to all members of the project development team and must meet group specifications in order to be reviewed and accepted. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 9.1.4.1 | 0 | In order to implement full stack development efficiently, especially given our limited related experience, we took the approach of outlining all of the requirements that we would need in every area of the project. That is, we considered all that we would need to assist frontend development, database implementation and management, hosting services and backup servers, system security, team communication and sharing of resources, project design and outlining, backend programming, and streamlined implementation of all of the above. In doing so, we expect to mitigate any unanticipated problems and always have a next step to take during development. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 9.1.4.2 | 0 | We continuously test different areas of the project during development and gather feedback from team members and potential users to identify and address issues before they come to us. Each component of the implementation has tests created for it to ensure that every component does what it should do, and does not do what it should not do. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 9.2.1 | 0 | Github Actions will be utilized to develop the CI/CD platform. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |

## Client Liaison Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 10.1.1 | 0 | There will be weekly emails and monthly video calls with stakeholders to update on the progress of the project and any oversights that may incur. | There is no guarantee that the clients have to show up to these meetings, meaning they can just not show up to these meetings. | Does not clarify which developers need to be present if any meaning some bad developers could skip out on these meetings | No | Penny McMahon, 10/16/2023 |
| 10.1.1 | 1 | There will be weekly emails and monthly video calls with stakeholders to update on the progress of the project and any oversights that may occur. All developers and other stakeholders must be present for these meetings to ensure that there are no inconsistencies or discrepencies. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 10.1.2 | 0 | Urgent issues would consist of a phone call or direct message to the concerned party. | Ok | No one is listed as being responsible for this phone call or direct message, which could cause confusion when issues do arise. | No | Penny McMahon, 10/16/2023 |
| 10.1.2 | 1 | Urgent issues related to the project or any stakeholders would consist of a phone call or direct message from the Project Manager to the concerned party. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 10.1.3 | 0 | Stakeholders will also have access to the project website and documentation. | Ok | It does not specify which stakeholders have access meaning some | No | Penny McMahon, 10/16/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | stakeholders could get left behind. | | |
| 10.1.3 | 1 | All stakeholders will have access to the project website and documentation to allow them to check in personally on progress if desired. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 10.2.1 | 0 | Reports to the relevant stakeholders would consist of live demonstrations and graphics. | Ok | Ok | Yes | Penny McMahon, 10/16/2023 |
| | | | | | | |
| 10.3.1 | 0 | We will showcase a checklist of progress to be communicated following the communication plan. | Ok | Ok | Yes | Penny McMahon, 10/16/2023 |
| | | | | | | |
| 10.4.1 | 0 | Any stakeholder or individual involved who wishes to suggest a change to these requirements after the freeze date must submit a formal Change Request to the original project manager or designated authority. Upon the project manager receiving the request, the project team will review the proposed changes for feasibility, impact, and alignment with project objectives and consult relevant stakeholders as necessary. After review, the change request will either be approved or rejected and all changes will be documented. Finally, the party who submitted the change request will be notified. | Ok | Ok | Yes | Penny McMahon, 10/16/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 10.4.2 | 0 | In rare circumstances, urgent changes may need to be made to the requirements specification without following the full change request process. Such changes can only be made with the approval of the project manager and must be documented. | Ok | Ok | Yes | Penny McMahon, 10/16/2023 |
| | | | | | | |
| 10.5.1 | 0 | Any formal feedback on development desires that would not warrant a formal change request will be submitted in writing to the project manager for review. | Ok | Ok | Yes | Penny McMahon, 10/16/2023 |
| | | | | | | |

## State/Mode Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 11.1.1.1 | 0 | When clicking on one of the header buttons, users will encounter a brief waiting period while the website redirects to and loads the correct page. | The waiting period was not brief for a client with bad network connection. | Only one of the header buttons may have this functionality while the others do not. | No | Julian Hernandez, 09/26/2023 |
| 11.1.1.1 | 1 | When clicking on any of the navigation bar buttons, users will encounter a waiting period while the website redirects to and loads the correct page. | Ok. | Ok. | Yes | Austin Truong, 10/02/2023 |
| | | | | | | |
| 11.1.2.1 | 0 | Upon clicking the filter tasks button, a menu will display next to the button with checkboxes for each of the available categories. Only one of these boxes may be checked at a time and the | Clicking outside of the filter tasks button does not result in the menu closing. | The menu appears but the checkboxes are unselectable. | No | Julian Hernandez, 09/26/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | user can click outside of the menu or the filter tasks button again to close it. | | | | |
| 11.1.2.1 | 1 | Upon clicking the filter tasks button, a menu will display next to the button with selectable checkboxes for each of the available categories. Only one of these boxes may be checked at a time and the user can click outside of the menu, or, click the filter tasks button again to close it. | Ok | Ok | Yes | Austin Truong 10/26/2023 |
| | | | | | | |
| 11.1.2.2.1 | 0 | When the user selects a task from the monthly, weekly, or daily calendar, the page will display a box-form containing the details of the task such as its exact deadline, priority, and description. It will also display the option to edit or delete the task. | It is unclear what a 'box-form' is so it could easily be stated that it is unpresent. | Only one of the options to edit or delete tasks might be given due to the lack of plurality. | No | Julian Hernandez, 09/26/2023 |
| 11.1.2.2.1 | 1 | When the user selects a task from the monthly, weekly, or daily calendar, a form will appear, presenting the task's details such as its deadline, priority, and description. Additionally, options to edit or delete the task will be available. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 11.1.2.2.2.1 | 0 | When selecting the edit task button, the box will update to display fields to edit the details of the given task. While the user is editing a task, they can either cancel the operation which will return them to the details display, or save their entry | It is unclear what a 'box-form' is so it could easily be stated that it is unpresent. | The state in which the user can not click outside of the field is unclear and may not work as intended. | No | Julian Hernandez, 09/26/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | which will refresh the page and update the information of the task. They will also be blocked from clicking outside of the box-form to close the display during this state. | | | | |
| 11.1.2.2.2.1 | 1 | When selecting the 'Edit Task' button, the form will update to display fields for editing the task. During editing, users can either cancel the operation to return to the details display or save their changes to refresh the page and update the task information. Clicking outside the form to close it is disabled during editing. | Ok | Ok | Yes | Austin Truong, 10/27/2023 |
| | | | | | | |
| 11.1.2.2.3 | 0 | When the user clicks outside of the field of the box-form, the box will disappear and display the main task manager web page. | It is unclear what a 'box-form' is so it could easily be stated that it is unpresent. | No specific purpose or action resulting in the form is mentioned, so this could be implemented in one place but not another. | No | Julian Hernandez, 09/26/2023 |
| 11.1.2.2.3 | 1 | When the user clicks outside the "Edit Task" form, it will disappear, revealing the main task manager web page. | Ok | Ok | Yes | Austin Truong, 10/27/2023 |
| | | | | | | |
| 11.1.2.2.2 | 0 | When the user adds, edits, or deletes a task, the webpage will refresh and update the displayed information to match their input. | Ok. | Bad-dev may not save the information in the database to be displayed later. | No | Julian Hernandez, 09/26/2023 |
| 11.1.2.2.2 | 1 | When the user adds, edits, or deletes a task, the webpage will refresh and update the displayed information to reflect their saved input. | Ok. | Ok | Yes | Austin Truong, 10/27/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| 11.1.3.1 | 0 | When the user attempts to log in, the webpage will redirect to the task manager page upon successful attempt and will notify them of an incorrect username or password when it is incorrect. | The client would expect these two separate occurences to happen in the same state. | The redirection may not be implemented to be automatic by a bad dev. | No | Julian Hernandez, 09/26/2023 |
| 11.1.3.1 | 1 | When the user attempts to log in, the webpage will automatically redirect to the task manager page upon a successful attempt. If authentication fails, user will be redirected to the same page for logging in. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 11.1.3.2.1 | 0 | When the user selects this option on the login web page, a form will appear with fields asking them to enter an email, username, and password. There will also be a button to finalize account creation labeled “Create account”. | Without the option listed in the requirement specifically the client could claim that any button was supposed to have this functionality. | Without the option listed specifically, the option may not be reachable or visible. | No | Julian Hernandez, 09/26/2023 |
| 11.1.3.2.1 | 1 | When users choose the 'Create Account' option on the login web page, a form will appear with fields for entering an email, username, and password. There will also be a 'Create Account' button for finalizing the account creation. | Ok | Ok | Yes | Austin Truong, 10/1/2023 |
| | | | | | | |
| 11.1.3.2.2.1 | 0 | Upon finalizing account creation, the ‘create account’ form will be replaced with text prompting the user to check their e-mail and click a verification link. Upon | The client may not know which status was confirmed once they are verified. | Ok. | No | Julian Hernandez, 09/26/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | verification, the previous page which prompted them to check their e-mail will now confirm this status. | | | | |
| 11.1.3.2.2.1 | 1 | Upon successfully finalizing account creation, the 'Create Account' form will be replaced with text instructing the user to check their email and click a verification link. After verification, the previous page will confirm their email verification status. | Ok | Ok. | Yes | Austin Truong, 10/1/2023 |
| | | | | | | |
| 11.1.3.3 | 0 | Selecting the change password option will display a box containing fields to enter their current password and a new one. Upon successful entry, text will replace the box informing them of a successful change. | Ok | Ok | Yes | Austin Truong, 10/25/2023 |
| | | | | | | |
| 11.1.3.4.1 | 0 | Clicking the forgot password button will prompt the user with fields to enter their username and corresponding e-mail. The e-mail will redirect them to a page with a field to enter a new password. After entry, they will be redirected to the login page again. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 11.1.4.1 | 0 | On-click, the report generator button will display text informing the user that they should not leave the web page and that it is generating their data, and upon generation, it will display text informing the user of a download in | Ok | Ok | Yes | Austin Truong, 10/26/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | progress along with starting to upload the report to their computer. | | | | |
| | | | | | | |
| 11.1.4.2 | 0 | If the report fails to generate, the download fails, or if the download is canceled, the web page will display text informing the user that report generation has failed. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 11.1.5.1 | 0 | A button will be displayed on the header bar next to the user account to give users the option to switch the website between light and dark mode. | Ok | Ok | Yes | Austin Truong, 10/27/2023 |
| | | | | | | |
| 11.2.1.1 | 0 | In Django's admin settings, it is possible to enable a debug mode where 404 pages are replaced with a page explaining the kind of error that the system caught, the paths to the files involved, and various lines explaining internal details about the error. All other 'normal' functionalities of the website will operate the same. The purpose of this mode is to test new features and see clear descriptions as to why they failed. | Ok | Ok | Yes | Austin Truong, 10/27/2023 |
| | | | | | | |

## Project Planning Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 12.1.1 | 0 | Set up a Discord as a means for communication, | Ok | Ok | Yes | Austin Truong, |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | document sharing, and keeping track of the overall project. | | | | 10/26/2023 |
| | | | | | | |
| 12.1.2 | 0 | Host and record meetings over Zoom to ensure that even team members not present for meetings will not be behind on the team's agenda. Ensure these recordings are shared with the other team members, and remind each who did not attend to watch the recording(s). | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 12.1.3 | 0 | Host, distribute, and organize tasks over Asana to keep team on track and on top of progress. Would later be transferred to google sheets for task management and calendar organization. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 12.1.4 | 0 | Ensure the implementation of a Github repository for the purposes of updating and hosting files as well as making sure developers decouple their changes. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 12.1.5 | 0 | Ensure all documents necessary for the completion and development of the CrabbyTasks applications are shared with all primary stakeholders to prevent any discrepencies or inconsistencies. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 12.2.1 | 0 | Meetings and discussiosn will regularly be held within the team to decide whether or not to proceed with the scope change. Project Manager will have final say as to how the group proceeds based on the cost and benefits the change will have for the application. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 12.3.1 | 0 | Must provide developers with tasks that can be completed in less than 5 hours and the time to meaningfully complete these tasks. If a task is deemed to be longer than 5 hours, it will be broken down into smaller components that take less than 5 hours each. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 12.4.1 | 0 | Tasks will be assigned based on project role, with each team member taking up a required responsibility befitting of the respective role. These tasks are managed and controlled through Asana by the planner and project manager and will be reviewed by another developer once each task is completed. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 12.5.1 | 0 | Developers and other stakeholders must have access to the calendar located on Asana to track progress. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 12.5.2 | 0 | The calendar should specify completion and due dates for all tasks. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |

## Project Group Management Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 13.1.1 | 0 | The management strategy for the Red Lobsters involves communication over discord and meetings over zoom, as well as the establishment of deadlines and necessary tasks tracked and documented over Asana and a customized spreadsheet. Throughout each meeting management looks ahead at the barriers we need to climb, sets our goals for the future, and answers group questions that are essential to complete the application. On our weekly Tuesday meetings we establish the goals for the week and create any new tasks, after which we plan for times to have individual or full group meetings. | Ok | Ok | Yes | Penny McMahon, 10/23/2023 |
| | | | | | | |
| 13.2.1 | 0 | Any conflict occurring between members of the Red Lobsters will be met with a private discussion with the project manager and all involved in the conflict itself. Ultimately these meetings are meant to resolve conflict and preserve a healthy and happy work environment. | Ok | Ok | Yes | Penny McMahon, 10/23/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 13.3.1.1 | 0 | The planner and the project manager will periodically check in with the other group members throughout the week. These check ins can occur through out of class meetings or through simple discord or text messages. They ultimately serve to benefit the group so that we may understand our current progress and tackle problems early. Group members who may have a tougher workload may reach out to other members who may not have anything scheduled in order to help. It falls on the project manager to ensure that the project is completed and that all deadlines are met. | Ok | Ok | Yes | Penny McMahon, 10/23/2023 |
| | | | | | | |
| 13.3.2.1 | 0 | If we run into a situation where we are behind or an unforeseen issue occurs it is the responsibility of the project management and the planner to best assess the situation. The team must measure the feasibility, importance, and how many hands are necessary to complete the project. If the task is unimportant and or infeasible the project is ultimately dropped and the group moves on. Otherwise the team assesses the necessary number of people and finds places in the schedule to assign the respective tasks. | Ok | How does this cover when the group gets unexpectedly ahead? | No | Penny McMahon, 10/23/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 13.3.2.1 | 1 | If we run into a situation where we are behind or an unforeseen issue occurs it is the responsibility of the project management and the planner to best assess the situation. The team must measure the feasibility, importance, and how many hands are necessary to complete the project. If the task is unimportant and or infeasible the project is ultimately dropped and the group moves on. Otherwise the team assesses the necessary number of people and finds places in the schedule to assign the respective tasks. Conversely when the group is ahead we may then reallocate resources in weaker areas to ensure a well rounded final product. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 13.3.2.2 | 0 | It is then the responsibility of the planner and management to understand how the slippage occurred and how best to prevent something like it from happening again. | Ok | Ok | Yes | Penny McMahon, 10/23/2023 |
| | | | | | | |
| 13.4.1 | 0 | On Tuesdays the Red Lobsters have their mandatory group meeting. During these meetings as well as during other meetings and progress updates throughout the week team members are free to share the progress that they have made and their goals for the future of the project. | Ok | Ok | Yes | Penny McMahon, 10/23/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| 13.5.1 | 0 | The project manager or someone on their behalf must be there to monitor each group meeting and ensure that each team member is working on their necessary tasks. Team members are also to share updates on progress as well as what they might have been working on in the group Discord. Members of the Red Lobsters are also required to fill in any hours spent on the project in a time log to ensure that we know exactly who worked on what for how long. | Ok | Ok | Yes | Penny McMahon, 10/23/2023 |
| | | | | | | |
| 13.6.1 | 0 | During each mandatory Tuesday meeting the project manager acknowledges the hard work that has been completed during the prior week. This helps boost morale as the team is acknowledged for their hard work and dedication. For individual tasks and modules there is also feedback given both over the Asana and through the spreadsheet. | Ok | Ok | Yes | Penny McMahon, 10/23/2023 |
| | | | | | | |
| 13.7.1 | 0 | The team at Red Lobster will be utilizing Asana and Google Sheets in order to keep track of our tasks and set deadlines. Discord is used for communication and keeping track of important dates and documents. Zoom is used for meetings, this is to ensure our meetings are recorded. | Ok | Ok | Yes | Penny McMahon, 10/23/2023 |

## Project Role Specifications

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 14.1.1 | 0 | Project Managers are responsible for delivering resources to team members and overall makes decisions for the group and the project at large. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.1.2 | 0 | Project Manager is the bottom line for the project as a whole, it is their responsibility to solve issues for the project should they arise. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.2.1 | 0 | Implementation Managers are responsible for implementing the design for CrabbyTasks and ensure that the application runs the day it is due. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.2.2 | 0 | Implementation Managers ensure that the code is up to the specifications laid out by the developers and their clients | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.3.1 | 0 | Collaboration Software Managers are responsible for getting users onboard and familiar with meeting and collaboration software. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 14.4.1 | 0 | Video/Media Managers are responsible for creating fully edited videos representing CrabbyTasks to investors and other stakeholders to show off overall progress. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.4.2 | 0 | Video/Media Managers are responsible for keeping backups ensuring there is a video ready to submit when necessary. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.5.1 | 0 | Planners are responsible for overall plan and for organizing schedules and deadlines for meetings | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.5.2 | 0 | Planners should continually check in with other group members on tasks and monitor Asana to ensure tasks and goals are met. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.6.1 | 0 | Webmasters are responsible for creating and hosting the website for all of our materials | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.7.1 | 0 | Documentation Managers are responsible for documenting the project in the project book | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.7.2 | 0 | Documentation Managers ensure group members have filled out time logs so that easy documentation of time spent is taken into account. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 14.8.1 | 0 | Requirements Managers are responsible for the requirements sheet and matrix | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.8.2 | 0 | Requirements Managers ensure that team members understand and adheres to these requirements | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.9.1 | 0 | Design Managers are responsible for the overall design of CrabbyTasks from which the code will be built. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.10.1 | 0 | Unix System Programmers are responsible for handling our unix systems, as well as overall assisting with implementation and coding | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.10.2 | 0 | Must have a strong technical background and have knowledge of Unix systems | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.11.1 | 0 | Testing Managers are responsible for testing CrabbyTasks and ensuring it works without error or any bugs. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.11.2 | 0 | Testing Managers also ensure requirements are strong and testable against bad-devs or bad-clients. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 14.12.1 | 0 | Responsible for presenting CrabbyTasks and creating the presentation showing off the culmination of our work in a limited time. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| 14.12.2 | 0 | Ensures the presentation has everything secondary stakeholders and investors may want to see. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |

## Stakeholders

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 15.1 | 0 | The primary stakeholders and creators of this application are the Red Lobsters | Ok | Ok | Yes | Penny McMahon, 10/17/2023 |
| | | | | | | |
| 15.1.1 | 0 | Team consisted of Austin Truong, Benny Chen, Carlos Tapia, Emma Nunnink, Graciella Jimenez, Julian Hernendez, Nick Dilallo, Penny McMahon, and Qi Deng. | Ok | Ok | Yes | Penny McMahon, 10/17/2023 |
| | | | | | | |
| 15.2 | 0 | Primary stakeholders of this application are XYZ Corporation and investors. | Ok | Ok | Yes | Penny McMahon, 10/17/2023 |
| | | | | | | |
| 15.3 | 0 | Secondary stakeholders and investors of this application are Clark Elliot and DePaul University. | Ok | Ok | Yes | Penny McMahon, 10/17/2023 |
| | | | | | | |

## User Profile Requirements

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| 16.1.1 | 0 | Users must understand English | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 16.1.2 | 0 | Users must have basic knowledge of web browsing, calendars, and technological comprehension. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 16.2.1 | 0 | There will be a basic user tutorial available for creating accounts, creating tasks, using the calendar, and other useful features, which will be available in our application. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |
| 16.3.1 | 0 | We expect many different kinds of users to use and interact with our program, including students, businesses, and hobbyists. Basically anyone who needs to keep track of their day to day activities. They will be required to meet the initial Skill Set Requirements. | Ok | Ok | Yes | Penny McMahon, 10/18/2023 |
| | | | | | | |

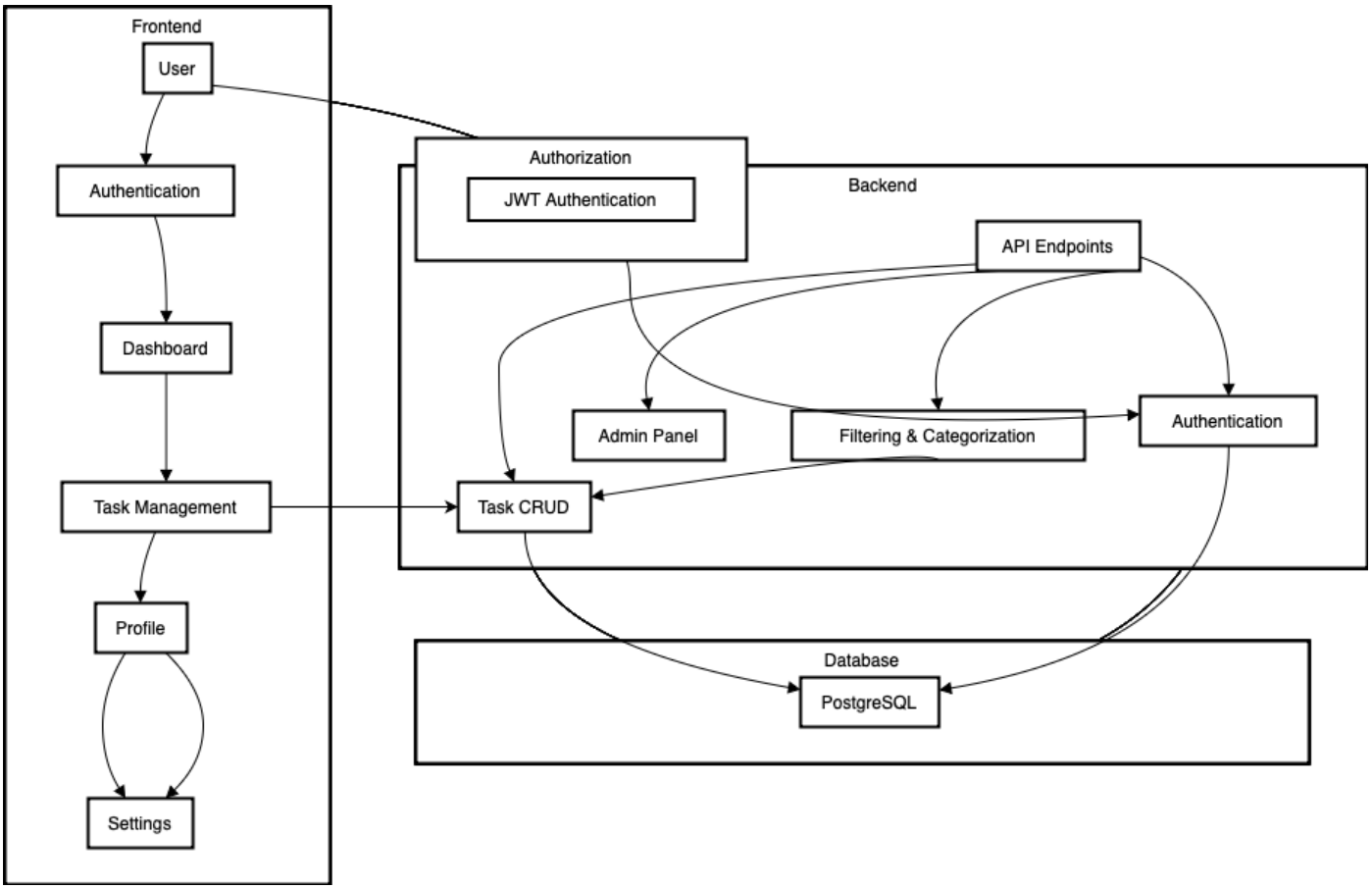
## Freeze Date

| # | Rev | Description | Bad-client attack | Bad-dev attack | Pass | Sign-off |
|---|---|---|---|---|---|---|
| 17.1 | 0 | The official freeze date is October 26th, 2023. | Ok | Ok | Yes | Penny McMahon, 10/25/2023 |
| | | | | | | |
| 17.1.1 | 0 | This means that no changes will be made to the document without following the process outlined below. | What if a client somehow forgets a crucial requirement after the freeze | What if a dev somehow forgets a crucial requirement after the freeze date? | No | Penny McMahon, 10/25/2023 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | date? | | | |
| 17.1.1 | 1 | This means that no changes will be made to the document. There are situations that arise where changes may be necessary, but for those to make it onto the Requirements Sheet, they will need to undergo the process entailed below. | Ok | Ok | Yes | Austin Truong, 10/26/2023 |
| | | | | | | |
| 17.2.1 | 0 | All change request will be processed as noted in 10.4.1 and 10.4.2 | Ok | Ok | Yes | Penny McMahon, 10/25/2023 |
| | | | | | | |

# Design

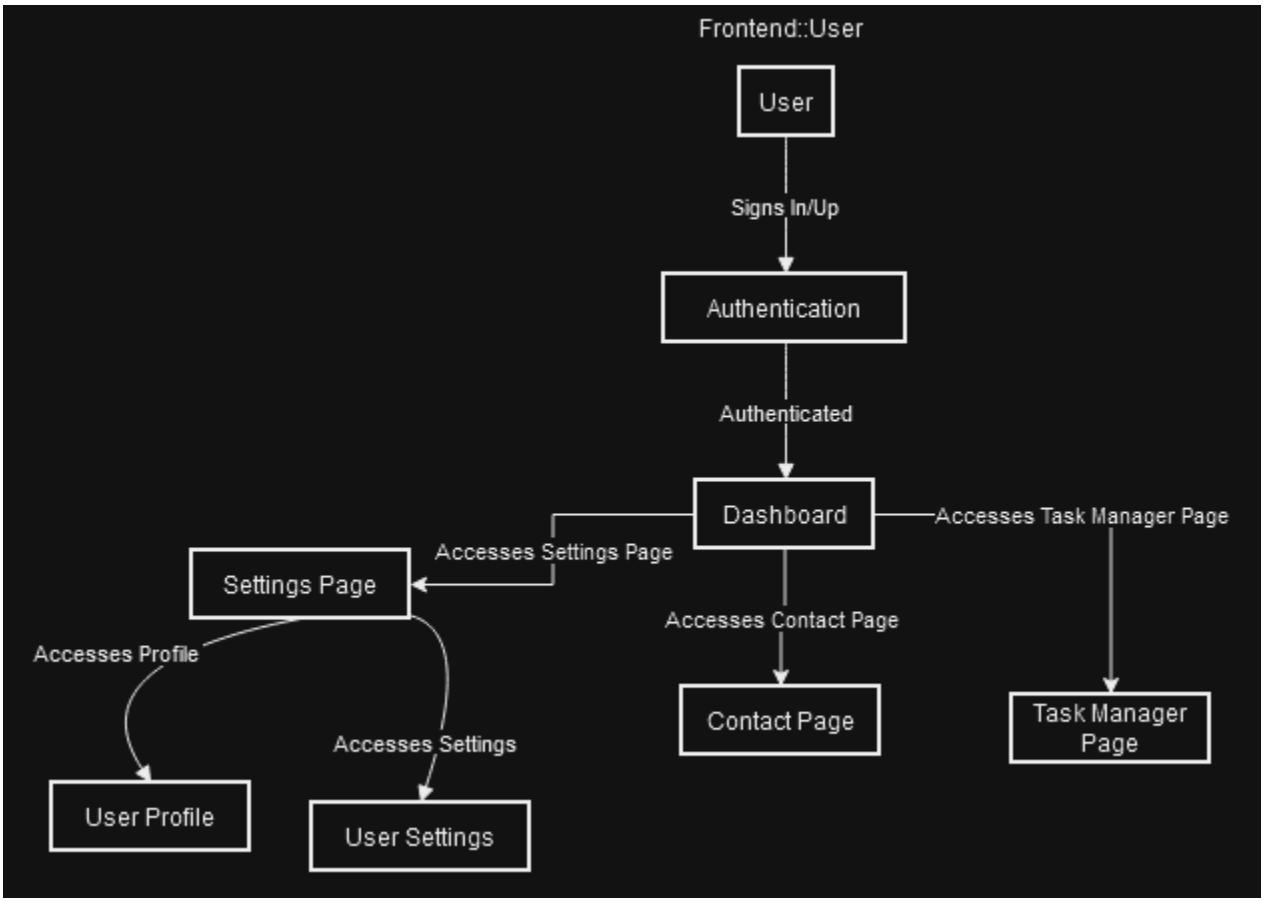
## Diagrams




This Level 0 diagram provides an overview of the CrabbyTasks system's components and their interactions:

- Frontend: The user interacts with the CrabbyTasks application through the frontend. It includes authentication, the main dashboard, task management, user profiles, and settings.
- Backend: The backend is responsible for handling API endpoints, including user authentication, CRUD operations for tasks, the admin panel, and task filtering and categorization.
- Database: PostgreSQL is used to store and manage the application's data, including user profiles, tasks, and settings.
- External Services: JWT Authentication services are utilized for secure user authentication.

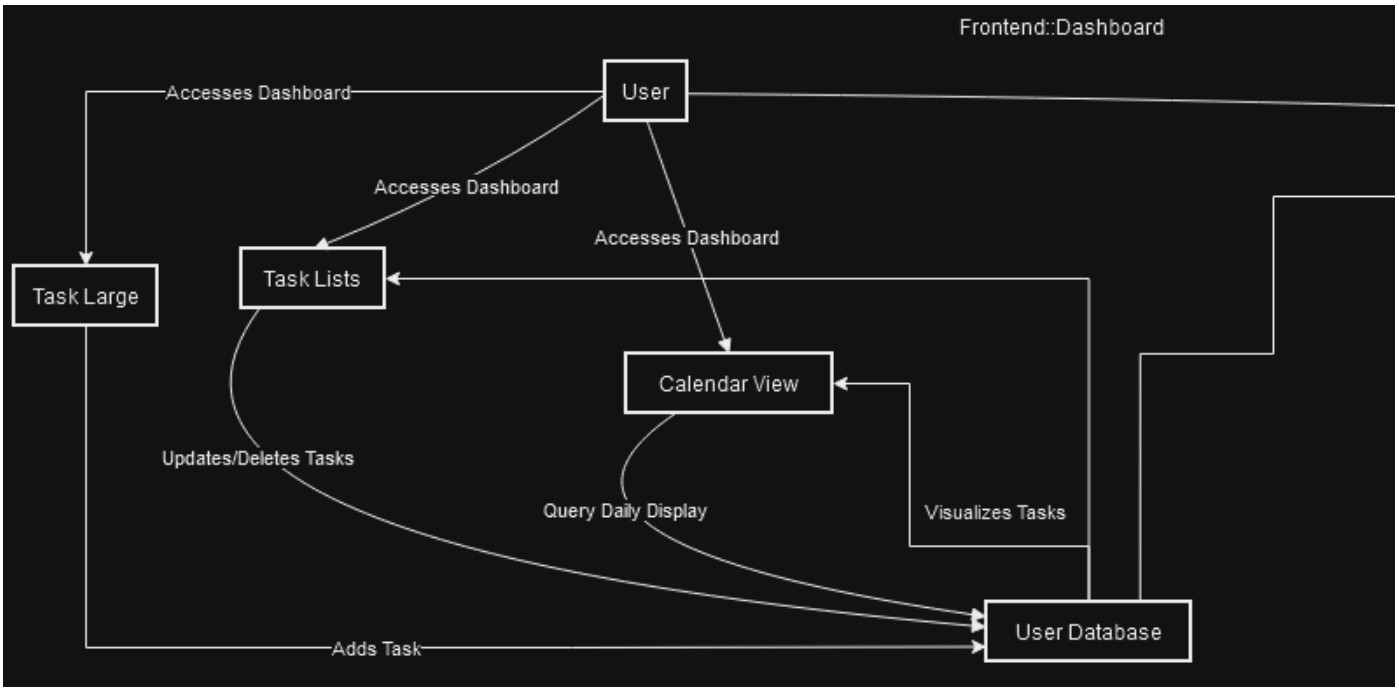
The diagram shows the flow of data and interactions between these components, highlighting how the user interacts with the frontend, which communicates with the backend to perform various operations. Data is stored and retrieved from the PostgreSQL database, and JWT Authentication services ensure secure user access.

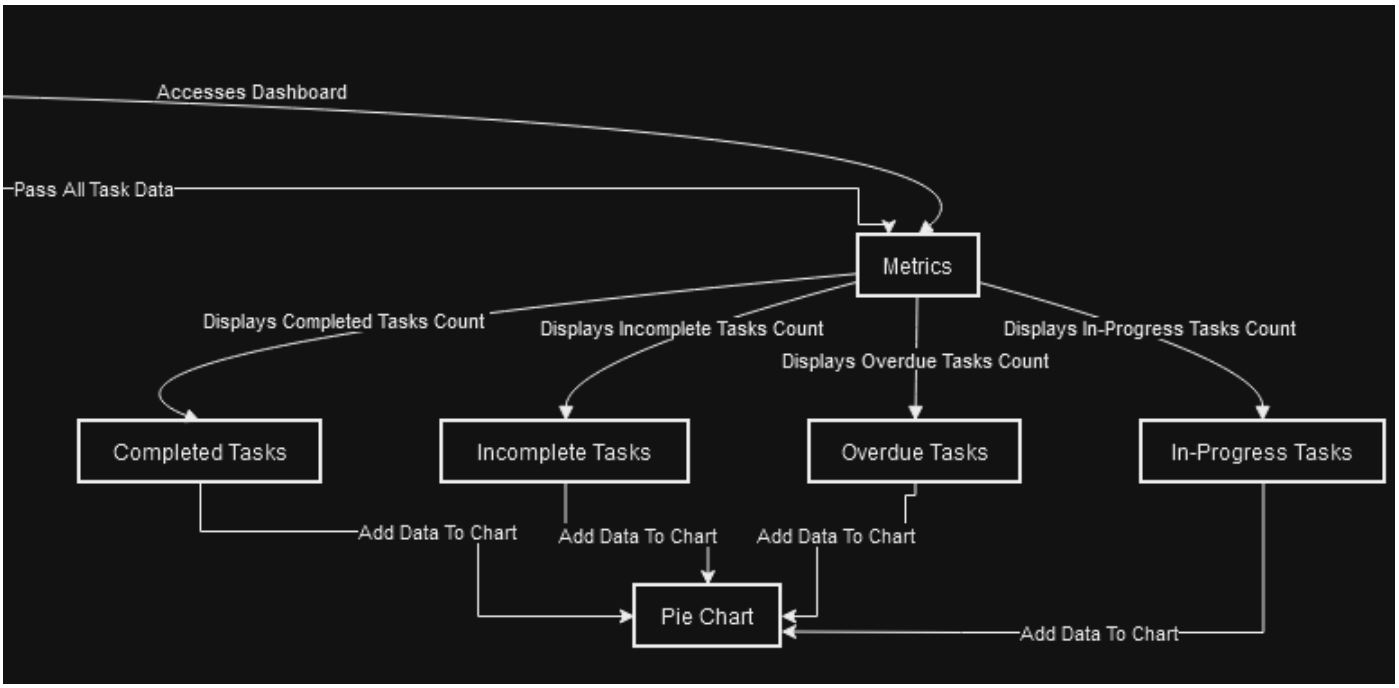



This Level 1 diagram breaks down the "Frontend::User" component into its subcomponents:

- User Profile: Users can view and manage their profiles within this subcomponent.
- User Settings: Personalized settings, such as notification preferences, are managed through this subcomponent.

The diagram shows the flow of interactions within the "Frontend::User" module, with users signing in or signing up, accessing the dashboard, and navigating to their profile or settings.

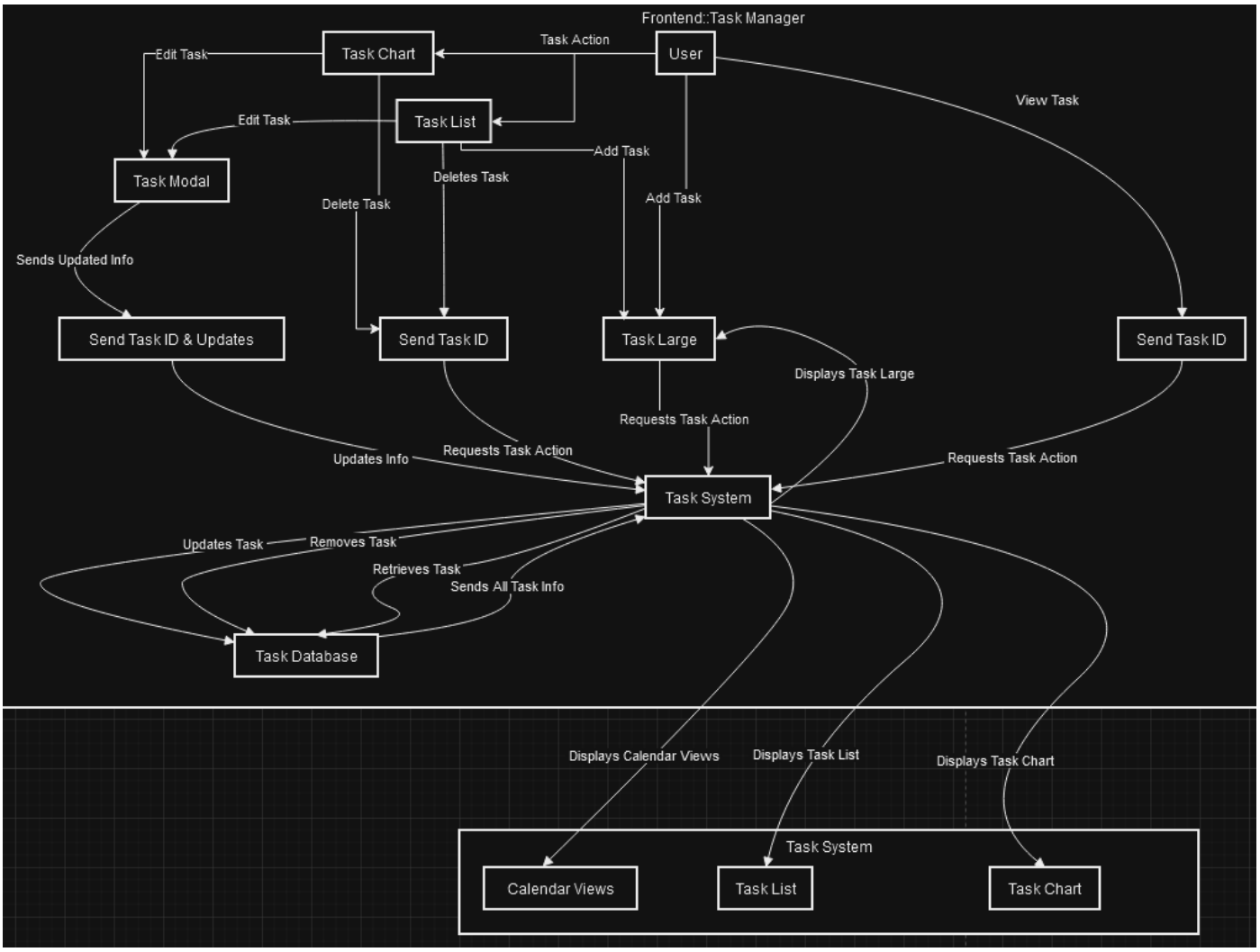



This Level 1 diagram breaks down the "Frontend::Dashboard" component into its subcomponents:

- Task Lists: Users can access and manage their tasks within this subcomponent. It includes features for task creation, updating, and deletion.
- Task Large: This subcomponent allows users to add new tasks quickly.
- Daily View: The calendar view allows users to visualize tasks and events based on due dates and schedules for the day.

The diagram illustrates the interaction between these subcomponents within the "Frontend::Dashboard" module, showing how users can switch between task lists and the calendar view.

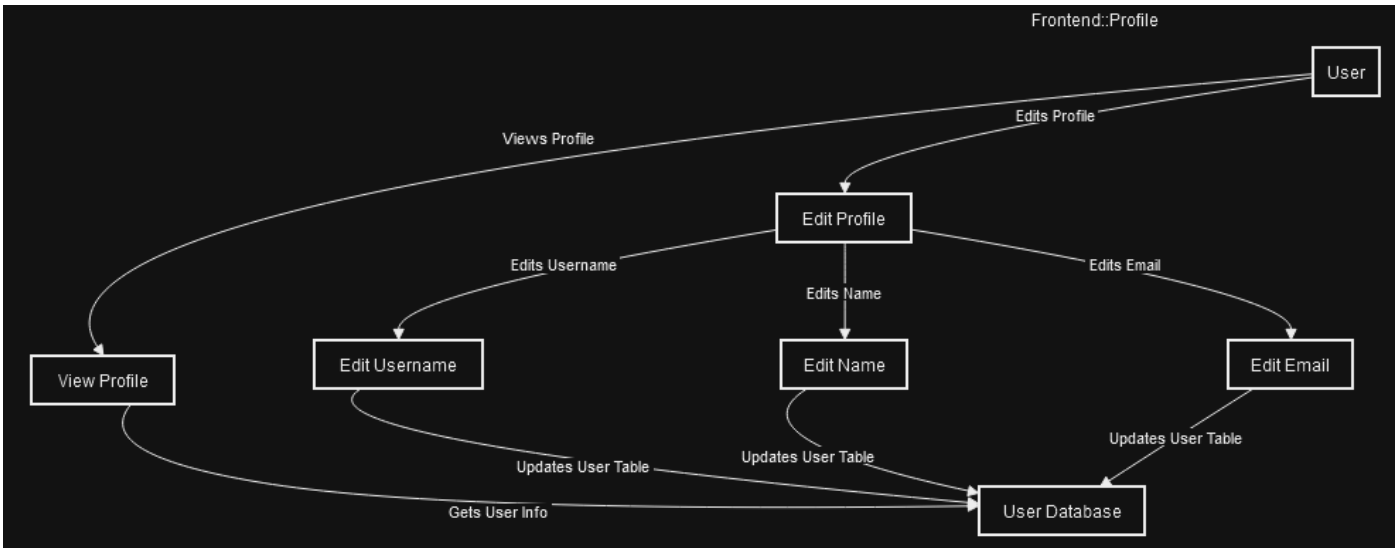



This Level 1 diagram breaks down the "Frontend::TaskManager" component into its subcomponents:

- Task Lists: Users can view and manage their tasks within this subcomponent, offering features for task creation, editing, and deletion.
- Calendar View: The calendar view allows users to visualize tasks based on due dates and schedules, offering a different perspective on task management.
- Task Modal: This subcomponent provides a modal interface for updating task details, including due dates, descriptions, and categories.

- Task Chart: Users can see an overview of all tasks and perform actions like deleting or updating tasks within this subcomponent.
- Task Large: This subcomponent allows users to add new tasks quickly.

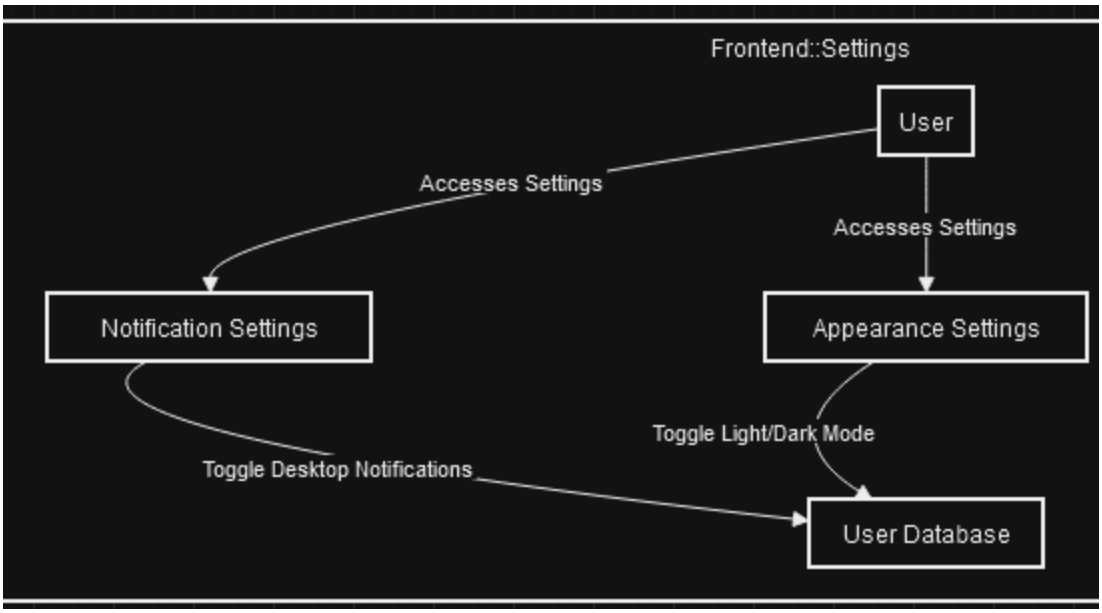
The diagram shows the interaction between these subcomponents within the "Frontend::TaskManager" module, illustrating how users can navigate between task lists, the calendar view, task modals, charts, and large task creation.




This Level 1 diagram breaks down the "Frontend::Profile" component into its subcomponents:

- View Profile: Users can view their profile information within this subcomponent, including details like their name, email, and username.
- Edit Profile: This subcomponent allows users to update their profile information, such as changing their name, email, or username.

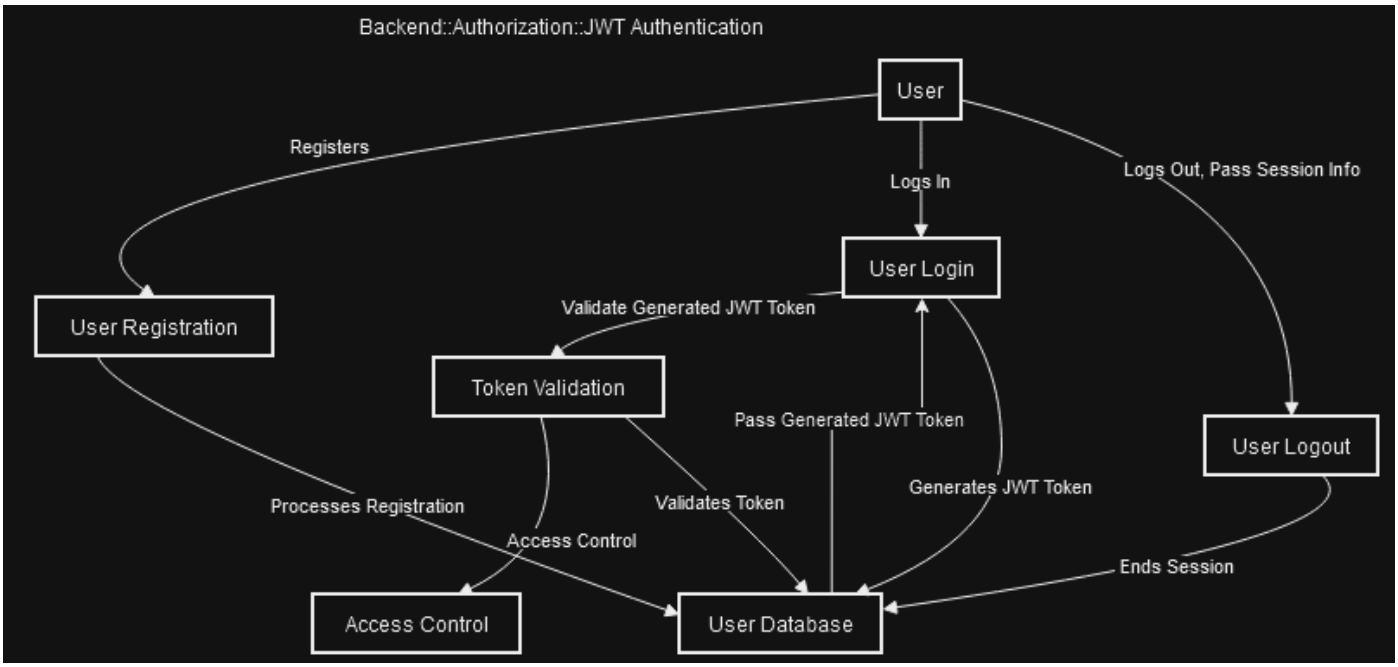
The diagram illustrates the interaction between these subcomponents within the "Frontend::Profile" module, showing how users can navigate between viewing their profile, and editing it.




This Level 1 diagram breaks down the "Frontend::Settings" component into its subcomponents:

- Notification Settings: Within this subcomponent, users can customize their notification preferences, such as desktop notifications.
- Appearance Settings: Users can adjust the visual appearance of the application, such as themes, fonts, and colors, within this subcomponent.

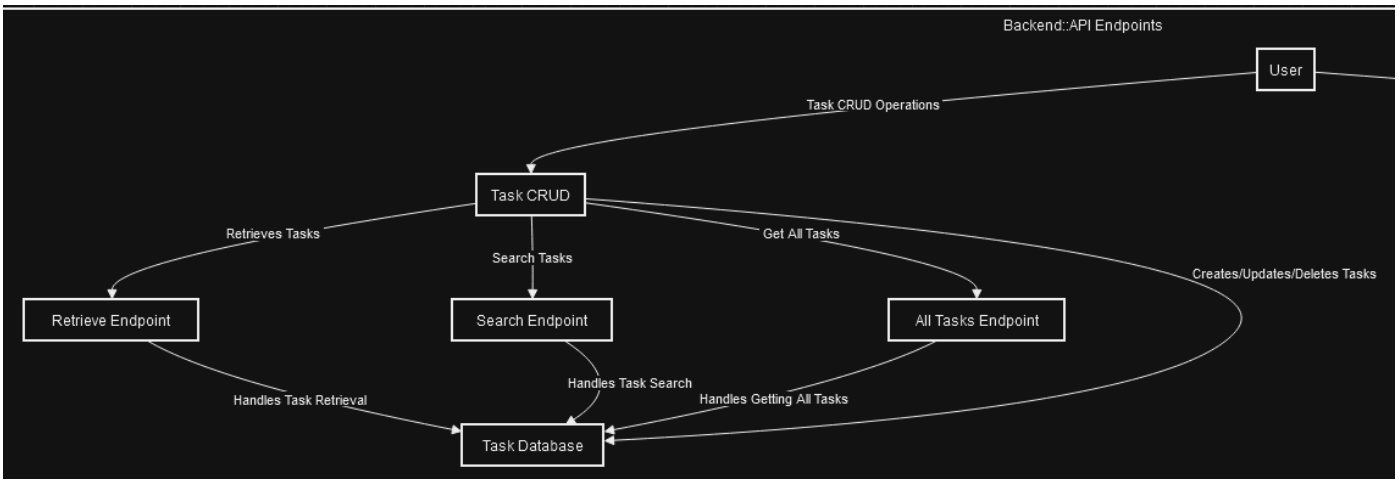
The diagram shows the interaction between these subcomponents within the "Frontend::Settings" module, illustrating how users can navigate between notification settings, appearance settings.

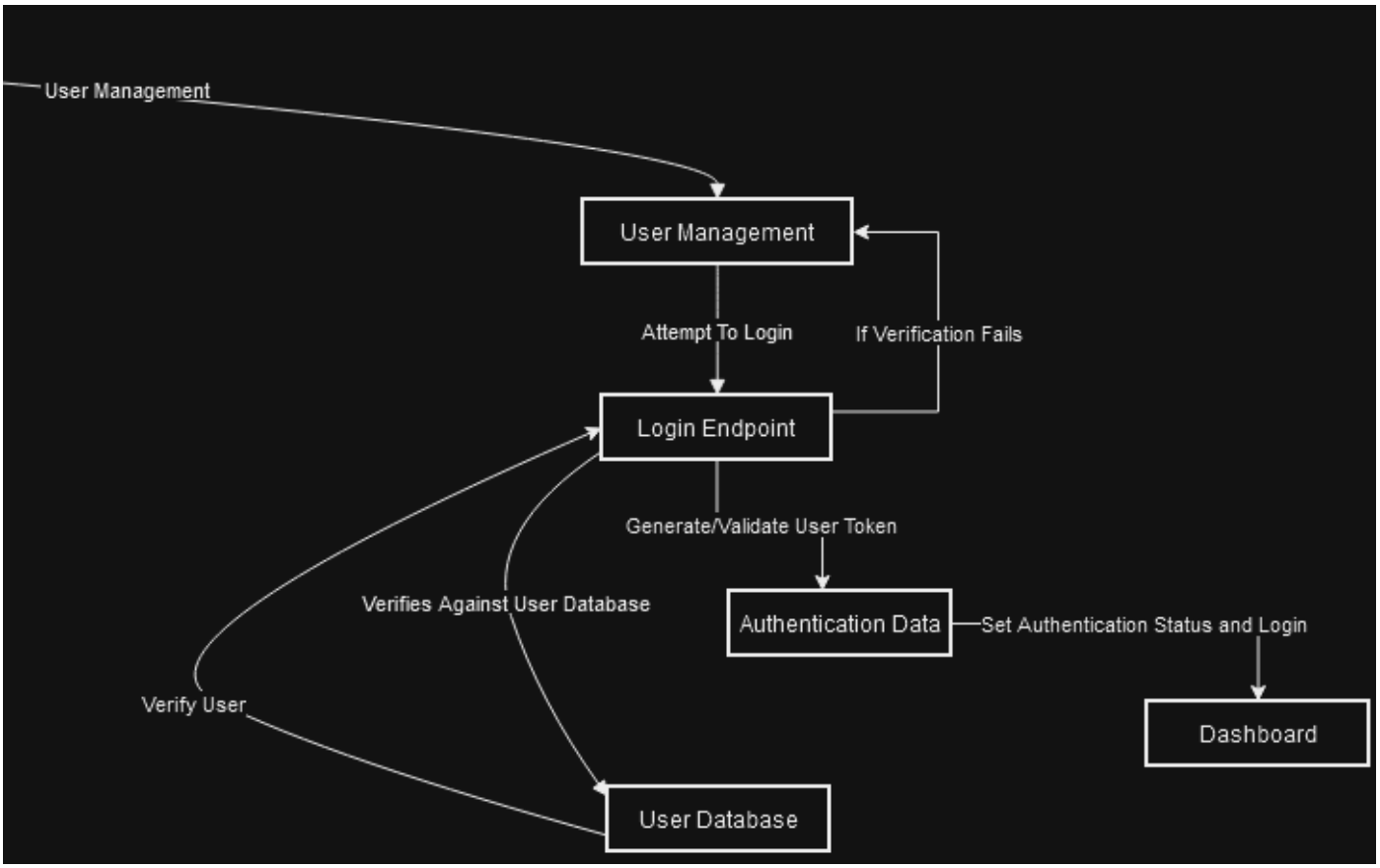



This Level 1 diagram breaks down the "Backend::Authorization::JWT Authentication" component into its subcomponents:

- User Registration: This subcomponent handles the user registration process, where new users can create accounts.
- User Login: Users can log in using their credentials through this subcomponent.
- User Logout: User session info gets passed to the database so that it can end their session and invalidate the JWT token.
- Token Validation: The backend validates JWT tokens generated during the login process within this subcomponent.
- Access Control: This subcomponent manages user access to various resources and endpoints based on their authentication status.

The diagram illustrates the flow of interactions within the "Backend::Authorization::JWT Authentication" module, from user registration to token validation and access control.

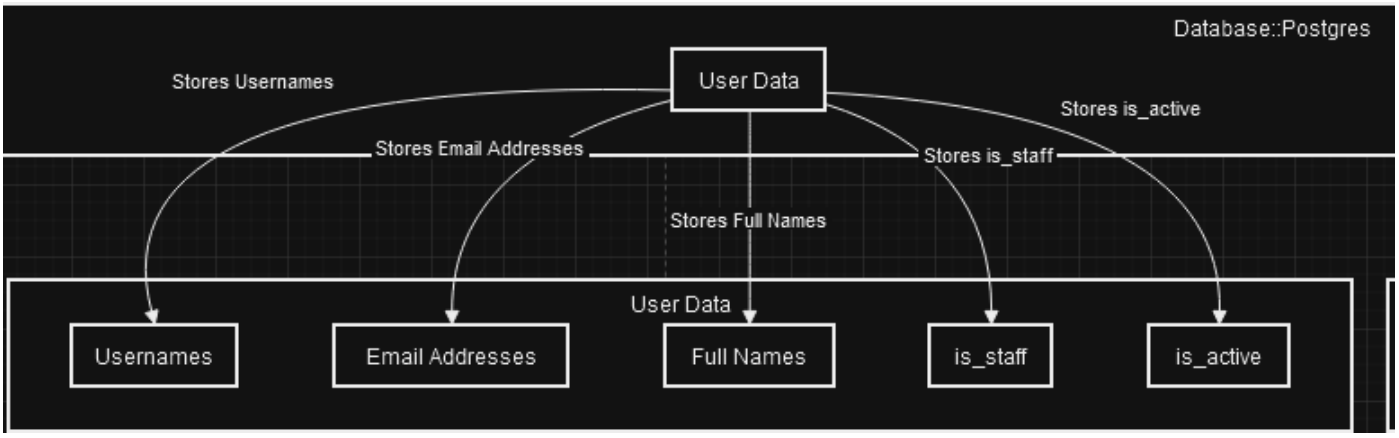


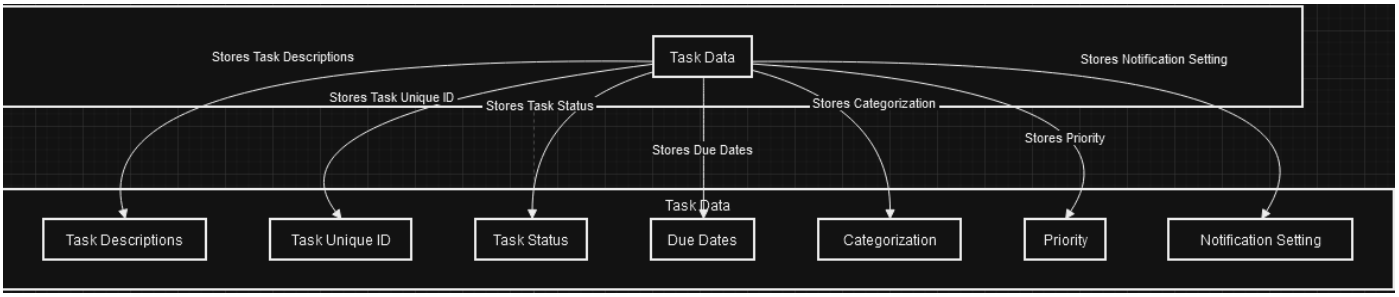



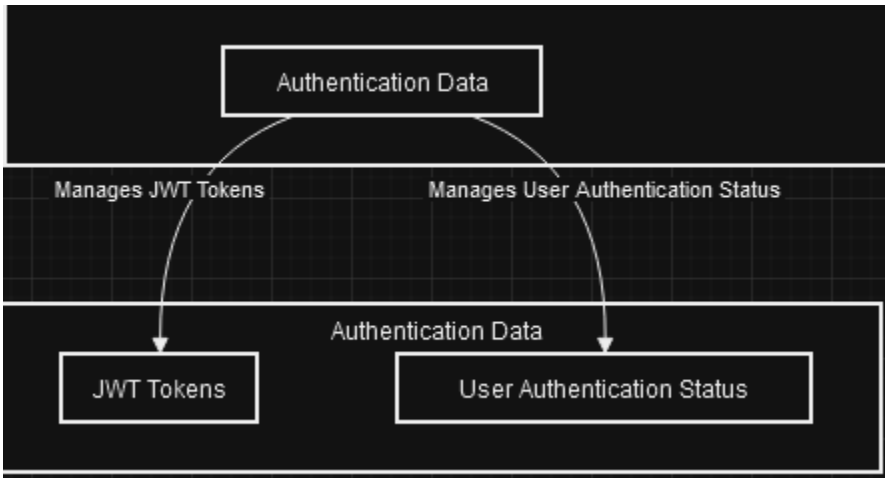

This Level 1 diagram breaks down the "Backend::API Endpoints" component into its subcomponents:

- Task CRUD: This subcomponent handles Create, Read, Update, and Delete (CRUD) operations for tasks. It includes features for task creation, editing, and deletion.
- User Management: This subcomponent manages user accounts, including user registration and login processes.

The diagram illustrates the interactions between these subcomponents within the "Backend::API Endpoints" module, showcasing how tasks are managed, filtered, and categorized, as well as how user management is integrated.




This Level 1 diagram breaks down the "Database::Postgres" component into its subcomponents:

- Task Data: This subcomponent stores task-related data, including task descriptions, due dates, and categorizations.
- User Data: Users' account information is stored in this subcomponent, including usernames, email addresses, and profile data.
- Authentication Data: This sub component manages authentication-related data, including JWT tokens and user authentication status.

The diagram illustrates the relationships and interactions between these subcomponents within the "Database::Postgres" module, showing how task data, user data, and authentication data are stored and interconnected.

## Data Dictionary

### Table: django_migrations

- id: INTEGER (Primary Key)
- app: VARCHAR(255)
- name: VARCHAR(255)
- applied: DATETIME

This model is used by Django to track database migrations. It stores information about applied migrations, including the application, migration name, and timestamp of when the migration was applied.

### Table: sqlite_sequence

- name: TEXT
- seq: INTEGER

This model is used by SQLite to keep track of the sequence of primary key values for auto-incrementing columns in tables. It stores the name of the table and the current sequence value.

### Table: django_content_type

- id: INTEGER (Primary Key)
- app_label: VARCHAR(100)
- model: VARCHAR(100)

Django uses this model to store information about the content types of objects in the database. It includes the application label and the model name associated with each content type.

### Table: auth_group_permissions

- id: INTEGER (Primary Key)
- group_id: INTEGER
- permission_id: INTEGER

This model represents the permissions associated with user groups in the authentication system. It links group IDs with permission IDs to define the permissions granted to each group.

### Table: auth_permission

- id: INTEGER (Primary Key)
- content_type_id: INTEGER
- codename: VARCHAR(100)
- name: VARCHAR(255)

This model stores information about individual permissions in the authentication system. It includes the content type ID, codename, and name of each permission.

### Table: auth_group

- id: INTEGER (Primary Key)
- name: VARCHAR(150)

The auth_group model represents user groups in the authentication system. It includes a unique name for each group.

### Table: users_new_user_model_groups

- id: INTEGER (Primary Key)
- new_user_model_id: BIGINT
- group_id: INTEGER

This model establishes a many-to-many relationship between the custom user model (users_new_user_model) and user groups. It allows users to belong to multiple groups.

### Table: users_new_user_model_user_permissions

- id: INTEGER (Primary Key)
- new_user_model_id: BIGINT
- permission_id: INTEGER

This model establishes a many-to-many relationship between the custom user model (users_new_user_model) and individual permissions. It allows users to have specific permissions.

### Table: django_admin_log

- id: INTEGER (Primary Key)
- object_id: TEXT
- object_repr: VARCHAR(200)
- action_flag: SMALLINT UNSIGNED
- change_message: TEXT
- content_type_id: INTEGER
- user_id: BIGINT
- action_time: DATETIME

Django uses this model to log administrative actions and changes made in the admin panel. It includes details about the changed object, the action taken, and the user responsible.

### Table: base_task_model

- status: VARCHAR(11)
- task: VARCHAR(30)
- description: VARCHAR(250)
- created_at: DATETIME
- updated_at: DATETIME
- deadline: DATETIME
- priority: VARCHAR(7)
- notifications: VARCHAR(12)
- user_id_id: BIGINT
- category: VARCHAR(8)
- task_id: CHAR(32) (Primary Key)

This custom model represents tasks in the application. It includes fields for task status, description, creation and update timestamps, deadlines, priorities, notifications, user IDs, category, and a unique task ID.

### Table: django_session

- session_key: VARCHAR(40) (Primary Key)
- session_data: TEXT
- expire_date: DATETIME

This model is used by Django to store session data for users. It includes a session key, session data, and an expiration date.

### Table: token_blacklist_blacklistedtoken

- blacklisted_at: DATETIME
- token_id: BIGINT (Primary Key)

This model appears to be related to token blacklisting. It stores information about blacklisted tokens, including the timestamp of blacklisting and the associated token ID.

### Table: token_blacklist_outstandingtoken

- token: TEXT
- created_at: DATETIME
- expires_at: DATETIME
- user_id: BIGINT
- jti: VARCHAR(255)
- id: INTEGER (Primary Key)

This model may be related to outstanding tokens. It includes information about tokens, creation and expiration timestamps, user IDs, and a unique JTI (JSON Web Token Identifier).

### Table: users_new_user_model

- id: INTEGER (Primary Key)
- password: VARCHAR(128)
- last_login: DATETIME
- is_superuser: BOOLEAN
- email: VARCHAR(254)
- user_name: VARCHAR(150)
- first_name: VARCHAR(150)
- start_date: DATETIME
- about: TEXT
- is_staff: BOOLEAN
- is_active: BOOLEAN

This custom user model represents user accounts in your application. It includes fields for user authentication, such as passwords, login status, email, user name, first name, start date, user information, and active status.

This data dictionary provides a structured overview of the tables in your project's database, including their columns, data types, and primary key designations. You can use this reference to understand the data structure of your application.

## Files on Disk

### Backend Directory Mapping

- base:
  - api:
    - __init__.py
    - filters.py
    - serializer.py
    - tests.py
    - urls.py
  - migrations:
  - __init__.py
  - admin.py
  - apps.py
  - models.py
  - tests.py
  - urls.py
  - views.py
- - config:
  - __init__.py
  - asgi.py
  - settings.py
  - urls.py
  - wsgi.py
- - users:
  - models.py

### Frontend Directory Mapping

- _frontend:
- .vite
- node_modules
- public

- src:
    - Tests
    - common
    - components
    - context
    - fonts
    - hooks
    - images
    - interfaces
    - js
    - layout
    - pages
    - routes
- .gitignore
- LICENSE.md
- README.md
- index.html
- package-lock.json
- package.json
- postcss.config.cjs
- tailwind.config.cjs
- tsconfig.json
- tsconfig.node.json
- vite.config.js

## Data Structures

### Task Properties:

- `id` (integer): Unique identifier for the task.
- `title` (string): Title or name of the task.
- `description` (string): Description of the task.
- `created_at` (datetime): Timestamp when the task was created.
- `updated_at` (datetime): Timestamp when the task was last updated.
- `deadline` (datetime): Deadline for completing the task.

- `priority` (string): Priority level of the task (e.g., high, medium, low).
- `status` (string): Status of the task (e.g., pending, completed, in-progress).
- `notifications` (string): Notification settings for the task.
- `category` (string): Category or type of the task.
- `user_id` (integer): User ID associated with the task.

The Task data structure represents an individual task within the CrabbyTasks application. Each task has various properties that include its title, description, creation and update timestamps, deadline, priority level, current status, notification settings, task category, and the associated user. Tasks are created by users to track and manage their to-do items, work assignments, or personal activities.

### User Properties

- `id` (integer): Unique identifier for the user.
- `username` (string): User's username.
- `email` (string): User's email address.
- `first_name` (string): User's first name.
- `last_name` (string): User's last name.
- `start_date` (datetime): Date when the user registered.
- `about` (text): User's personal information or bio.
- `is_staff` (boolean): Indicates if the user has staff privileges.
- `is_active` (boolean): Indicates if the user account is active.

The User data structure represents a registered user of the CrabbyTasks application. Users have unique identifiers, usernames, email addresses, first and last names, registration date, personal bios, and attributes indicating their staff privileges and account status. Users can create and manage tasks within the application.

### AuthenticationToken Properties:

- `token` (string): Authentication token for the user.

- `created_at` (datetime): Timestamp when the token was created.
- `expires_at` (datetime): Timestamp when the token expires.
- `user_id` (integer): User ID associated with the token.
- `jti` (string): JSON Web Token Identifier.

The AuthenticationToken data structure is used for user authentication within the CrabbyTasks application. When a user logs in, the system generates an authentication token that is associated with the user's account. This token is used to verify the user's identity for subsequent requests to protected resources. It has properties such as the token itself, creation and expiration timestamps, a reference to the associated user, and a unique JSON Web Token Identifier (jti) for security purposes.

These data structures provide the foundation for managing user accounts, tasks, and authentication within the CrabbyTasks system. They enable users to create, track, and organize their tasks while ensuring secure access to their accounts.

## Security and Privacy

CrabbyTasks takes security and privacy seriously and has implemented various measures to ensure the safety and confidentiality of user data. Our commitment to security and privacy is reflected in the following key principles:

### Data Encryption

All data transmitted between the user's device and our servers is encrypted using industry-standard protocols such as HTTPS. This encryption ensures that any information sent to or received from CrabbyTasks remains secure and protected from unauthorized access.

### User Authentication

CrabbyTasks employs a robust user authentication system, utilizing JWT (JSON Web Tokens) to verify the identity of users during login and subsequent interactions with the application. User passwords are securely hashed and salted to prevent unauthorized access to user accounts.

## Access Control

Access to user data is strictly controlled and limited to authorized personnel only. Role-based access control (RBAC) mechanisms are in place to ensure that each user can only access the data and features relevant to their role within the application.

## Regular Security Audits

CrabbyTasks conducts regular security audits and vulnerability assessments to identify and address potential security weaknesses. This proactive approach helps us stay ahead of emerging threats and maintain a high level of security.

## Privacy Policies

We have comprehensive privacy policies in place that outline how user data is collected, used, and protected. Users are provided with clear information about our data practices, and they have the ability to manage their data preferences and consent.

## Data Retention

CrabbyTasks follows strict data retention policies, ensuring that user data is retained only for as long as necessary to provide the service or as required by applicable laws and regulations. Once data is no longer needed, it is securely deleted from our systems.

## Third-Party Integration

When integrating with third-party services or APIs, CrabbyTasks conducts due diligence to ensure that these services also adhere to high-security standards and data protection practices. We prioritize partnerships with reputable and security-conscious providers.

### Incident Response

In the event of a security breach or data incident, CrabbyTasks has a well-defined incident response plan in place. We promptly notify affected users and relevant authorities, take immediate corrective actions, and continuously improve our security measures.

### User Education

We are committed to educating our users about best practices for online security and privacy. We provide resources and tips to help users protect their accounts and personal information.

CrabbyTasks recognizes that security and privacy are ongoing concerns, and we continuously monitor and enhance our security measures to adapt to evolving threats and regulatory changes. We are dedicated to maintaining the trust and confidence of our users by safeguarding their data and ensuring a secure and private user experience.

## Handling of Multiple Users, Transactions, and Requests:

CrabbyTasks is designed to efficiently handle multiple users, concurrent transactions, and a high volume of requests to provide a seamless and responsive experience for our users. Here's how we manage these aspects:

### Scalable Architecture

Our backend infrastructure is designed with scalability in mind. We use cloud-based services and auto-scaling mechanisms to ensure that the application can handle a growing user base and increased traffic without performance degradation.

### Database Optimization:

The database is optimized for read-heavy and write-heavy operations. We implement indexing and plan to implement caching strategies to minimize query response times and improve overall database performance.

### Monitoring and Alerts

We have monitoring systems in place to track system performance, resource utilization, and error rates in real-time. If any anomalies or issues are detected, alerts are triggered, allowing our team to respond promptly.

### Redundancy and Failover

Redundancy is built into our infrastructure to minimize the impact of hardware failures or unexpected outages. Failover mechanisms automatically route traffic to healthy servers or data centers to maintain service availability.

### Data Isolation

User data is isolated to ensure that one user cannot access another user's data. Robust access controls and authentication mechanisms are in place to enforce this isolation.

### Quality of Service (QoS)

CrabbyTasks prioritizes Quality of Service to deliver a responsive and reliable application. We continuously optimize our infrastructure and codebase to meet performance expectations.

By implementing these strategies and best practices, CrabbyTasks ensures that it can handle the demands of multiple users, transactions, and requests while delivering a smooth and efficient user experience. Our commitment to robust architecture and performance optimization allows us to maintain high availability and responsiveness.

## Hardware

**Amazon EC2 Instances:** These are virtual servers provided by Amazon Web Services (AWS) that host the CrabbyTasks application. EC2 instances are scalable and offer reliable compute capacity, making them suitable for hosting web applications.

**Apache Web Server:** Apache HTTP Server, often referred to as Apache, is a widely used open-source web server software. It serves as the foundation for delivering web content to users' browsers, ensuring the application is accessible over the internet.

**PostgreSQL Database Server:** PostgreSQL is a powerful open-source relational database management system. In the context of CrabbyTasks, it stores and manages the application's data, including user information and task details.

**Computers and Laptops**: These are the development workstations used by the development team. Developers use their computers or laptops to write, test, and debug code, ensuring the application's functionality and reliability.

**Smartphones:** While not explicitly mentioned in the provided information, smartphones may be used for testing the mobile responsiveness and usability of the CrabbyTasks application. This is important to ensure a seamless user experience on mobile devices.

**Networks:** The application relies on internet networks to connect users to the hosted CrabbyTasks application. This includes local area networks (LANs), wide area networks (WANs), and the broader internet infrastructure.

**Specialty Devices:** No specialty devices; The core hardware components primarily consist of standard computing infrastructure.

These hardware components collectively enable the development, hosting, and accessibility of the CrabbyTasks application to users.

## Network Protocols

CrabbyTasks, as a web application with a Django Backend and React Frontend, utilizes several network protocols to ensure seamless communication and functionality. These protocols are fundamental for data transmission and interaction between clients and the server. Here's an expanded explanation of the network protocols CrabbyTasks employs:

**HTTP (Hypertext Transfer Protocol):** CrabbyTasks relies on HTTP as the foundation for communication between the client (web browser) and the server. HTTP is responsible for

transmitting requests for web resources (like HTML pages and JSON data) from the client to the server and delivering server responses back to the client.

**HTTPS (Hypertext Transfer Protocol Secure):** To enhance security and protect user data, CrabbyTasks implements HTTPS. HTTPS is an encrypted version of HTTP, ensuring that all data transmitted between the client and server is encrypted and secure. This is crucial when handling sensitive user information and login credentials.

**WebSocket:** WebSocket is a protocol that facilitates real-time, full-duplex communication between the client and server. CrabbyTasks uses WebSocket to enable features like real-time notifications and updates, ensuring that users receive immediate feedback on task changes and interactions with the application.

**TCP/IP (Transmission Control Protocol/Internet Protocol):** TCP/IP is the underlying suite of protocols that powers internet communication. CrabbyTasks relies on TCP/IP to ensure reliable data transmission across the internet. It handles the routing, addressing, and delivery of data packets between the web server and clients.

**SMTP (Simple Mail Transfer Protocol):** CrabbyTasks may use SMTP to send email notifications to users. For example, users might receive email alerts for task reminders or updates. SMTP is responsible for sending these emails securely to designated email addresses.

**DNS (Domain Name System):** DNS is essential for translating human-readable domain names (e.g., www.red-lobsters.dev) into IP addresses that machines can understand. CrabbyTasks relies on DNS to resolve domain names and establish connections to the server.

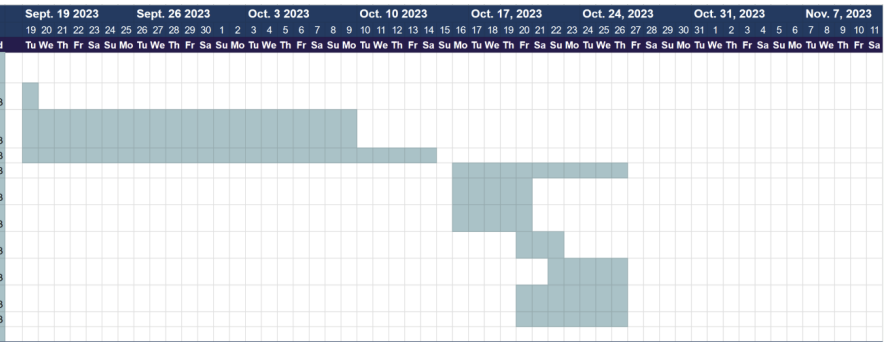
**SSH (Secure Shell):** While not a network protocol for regular application traffic, SSH may be used for secure remote server management and maintenance. CrabbyTasks administrators can use SSH to access servers securely for updates, maintenance, and troubleshooting.

These network protocols collectively ensure that CrabbyTasks operates smoothly, securely, and efficiently, providing users with a seamless task management experience while safeguarding their data and privacy.

# Plan

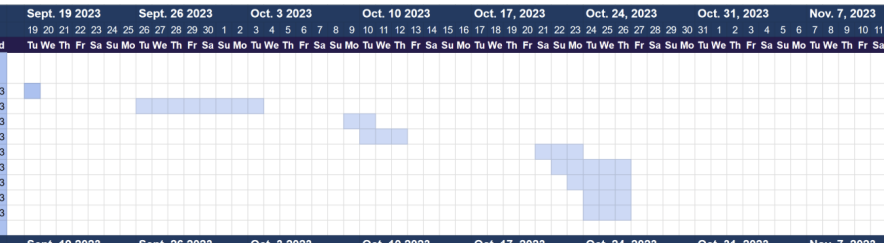
"~" Indicates Dependancy on Task Above

| Task | Assigned To | Status | Start | End |
|---|---|---|---|---|
| **Frontend Implementation** | | | | |
| Create Frontend base structure | Emma Nunnink, Austin Truong | Completed | 9/12/23023 | 9/18/2023 |
| Module 1 | Nick Dillalo, Julian Hernandez | Completed | 9/19/2023 | 10/9/2023 |
| ~Module 2 | Nick Dillalo | Completed | 9/16/2023 | 10/14/2023 |
| ~~Module 3 | Nick Dillalo | Completed | 10/16/2023 | 10/26/2023 |
| ~~Module 4 | Julian Hernandez | Completed | 10/16/2023 | 10/20/2023 |
| ~~Module 5 | Julian Hernandez | Completed | 10/16/2023 | 10/20/2023 |
| ~~Module 6 | Julian Hernandez | Completed | 9/20/2023 | 10/22/2023 |
| ~~Module 7 | Julian Hernandez | Completed | 9/22/2023 | 10/26/2023 |
| ~~Module 8 | Julian Hernandez | Completed | 9/20/2023 | 10/26/2023 |
| ~~~Module 9 | Nick Dillalo | Completed | 10/20/2023 | 10/26/2023 |



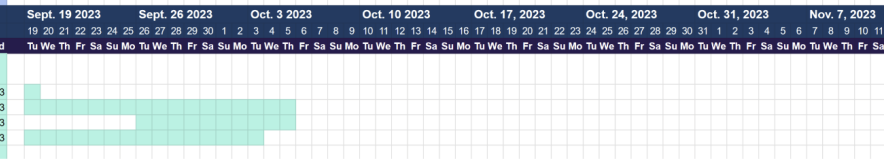

"~" Indicates Dependancy on Task Above

| Task | Assigned To | Status | Start | End |
|---|---|---|---|---|
| **Testing** | | | | |
| Test base structure | Nick Dillalo | Completed | 9/12/23023 | 9/19/2023 |
| Test Backend 2 | Nick Dillalo | Completed | 9/26/2023 | 10/3/2023 |
| Test Backend 3 | Nick Dillalo | Completed | 10/9/2023 | 10/10/2023 |
| Test Backend 4 | Nick Dillalo | Completed | 10/10/2023 | 10/12/2023 |
| Test Frontend 3 | Nick Dillalo | Completed | 10/20/2023 | 10/26/2023 |
| Test Frontend 4 | Nick Dillalo | Completed | 10/21/2023 | 10/23/2023 |
| Test Backend 5 | Nick Dillalo | Completed | 10/22/2023 | 10/26/2023 |
| Test Backend 6 | Nick Dillalo | Completed | 10/23/2023 | 10/26/2023 |
| Test Backend 7 | Nick Dillalo | Completed | 10/23/2023 | 10/26/2023 |



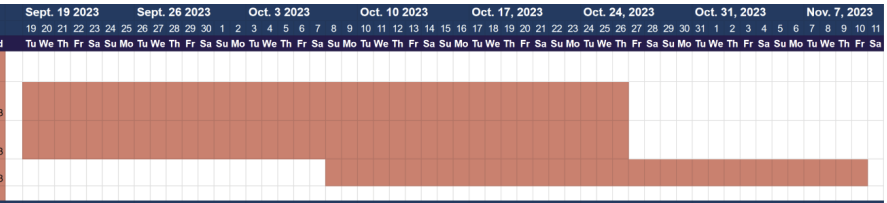

"~" Indicates Dependancy on Task Above

| Task | Assigned To | Status | Start | End |
|---|---|---|---|---|
| **Design** | | | | |
| Project Design Architecture | Qi Deng | Complete | 9/12/2023 | 9/19/2023 |
| Design Document 1 | Qi Deng | Complete | 9/19/2023 | 10/5/2023 |
| ~Design Document 2 | Qi Deng | Complete | 9/26/2023 | 10/5/2023 |
| Guidance Demo | Qi Deng | Complete | 9/19/2023 | 10/3/2023 |



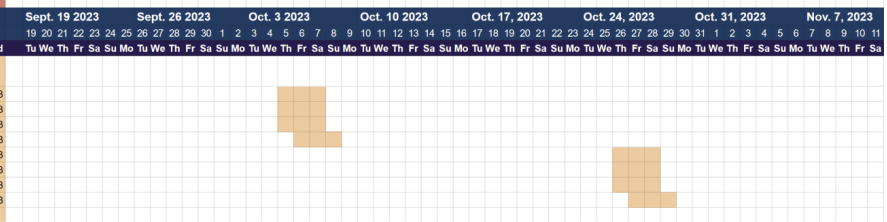

"~" Indicates Dependancy on Task Above

| Task | Assigned To | Status | Start | End |
|---|---|---|---|---|
| **Documentation** | | | | |
| Project Booklet Draft 1 | Graciella Jimenez, Benny Chen | Complete | 9/19/2023 | 10/26/2023 |
| Project Booklet Draft 2 | Graciella Jimenez, Benny Chen | Complete | 9/19/2023 | 10/26/2023 |
| ~Finalize Project Booklet | Graciella Jimenez | Complete | 10/8/2023 | 11/10/2023 |



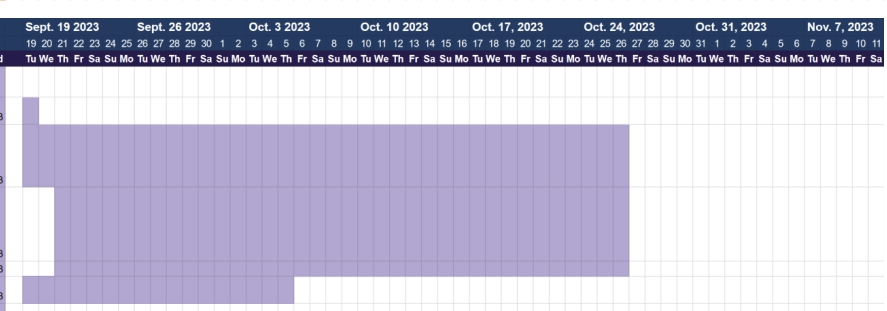

"~" Indicates Dependancy on Task Above

| Task | Assigned To | Status | Start | End |
|---|---|---|---|---|
| **Media and Presentation** | | | | |
| RDP Scripting | Emma Nunnink | Complete | 10/5/2023 | 10/7/2023 |
| ~RDP Presentation | Emma Nunnink | Complete | 10/5/2023 | 10/7/2023 |
| ~~RDP Recordings | Emma Nunnink | Complete | 10/5/2023 | 10/7/2023 |
| ~~~RDP Editing | Emma Nunnink | Complete | 10/6/2023 | 10/8/2023 |
| Preview Scripting | Emma Nunnink | Complete | 10/5/2023 | 10/72023 |
| ~Preview Presentation | Emma Nunnink | Complete | 10/5/2023 | 10/7/2023 |
| ~~Preview Recordings | Emma Nunnink | Complete | 10/5/2023 | 10/7/2023 |
| ~~~Preview Editing | Emma Nunnink | Complete | 10/6/2023 | 10/8/2023 |



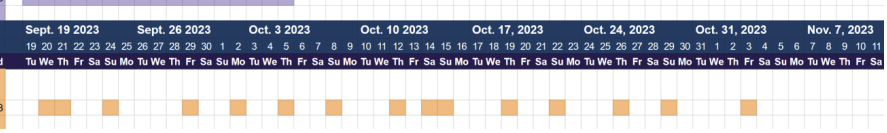

"~" Indicates Dependancy on Task Above

| Task | Assigned To | Status | Start | End |
|---|---|---|---|---|
| **Requirements, Planning and Collaboration** | | | | |
| Team Website | Julian Hernandez | Completed | 9/12/2023 | 9/18/2023 |
| Requirements Sheet | Penny McMahon, Graciela Jimenez, Qi Deng, | Completed | 9/19/2023 | 10/26/2023 |
| ~Requirements Matrix | Penny McMahon, Julian Hernandez, Graciela Jimenez | Completed | 9/21/2023 | 10/26/2023 |
| ~~Requirements Testing | Nick Dilallo | Completed | 9/21/2023 | 10/26/2023 |
| Overall Plan | Penny McMahon | Completed | 9/19/2023 | 10/5/2023 |



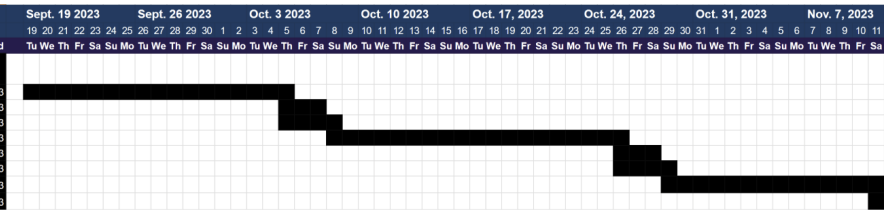

"~" Indicates Dependancy on Task Above

| Task | Assigned To | Status | Start | End |
|---|---|---|---|---|
| **Project Management** | | | | |
| Team Meetings and Check-Ins | Austin Truong | In Progress | 9/19/2023 | 11/11/2023 |

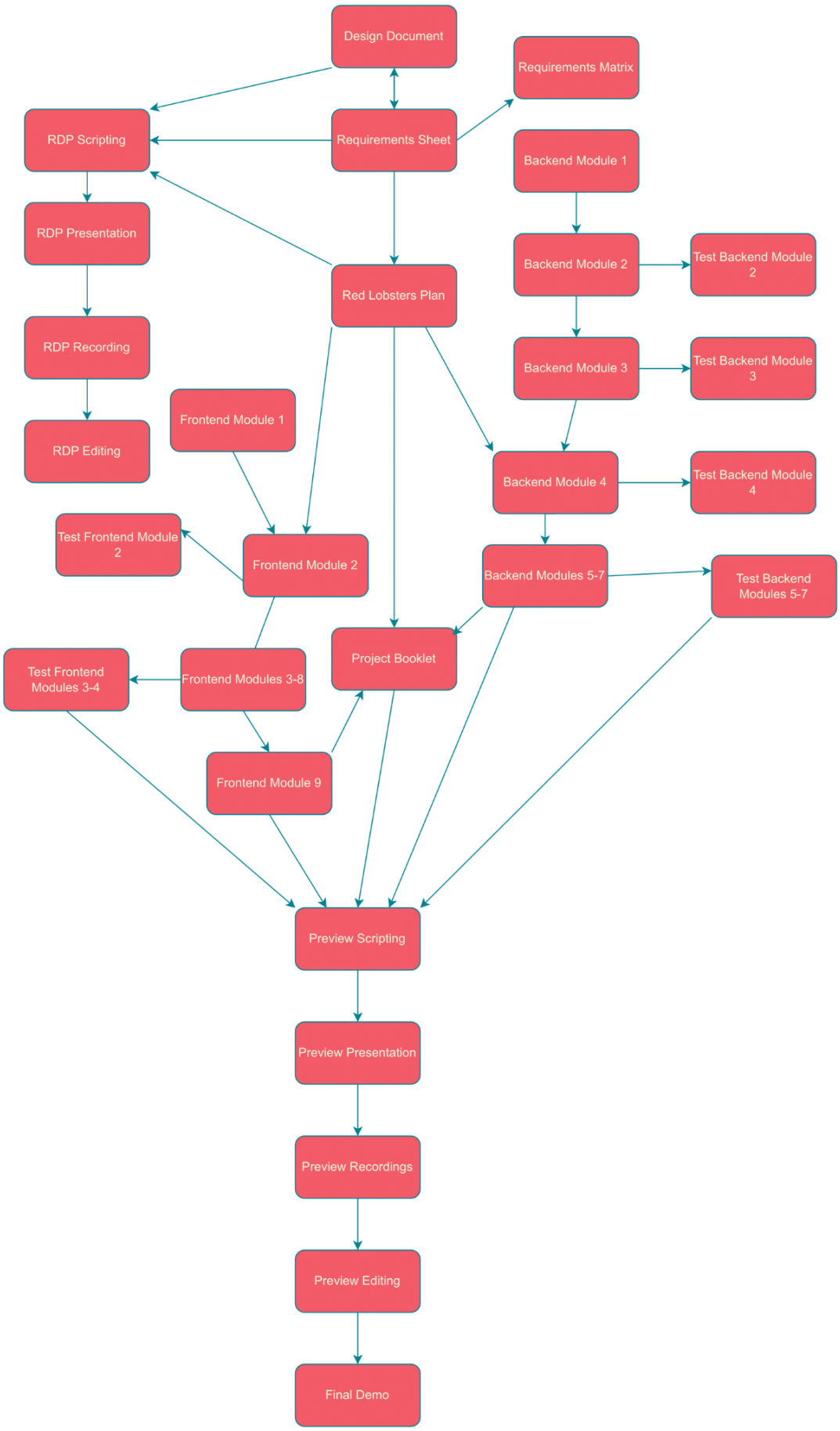



# Critical Path

| Task | Assigned To | Status | Start | End |
|---|---|---|---|---|
| **Critical Path** | | | | |
| All necessary work needed for RDP Demo | | In Progress | 9/19/2023 | 10/5/2023 |
| All recordings needed for the RDP Demo | | To-Do | 10/5/2023 | 10/7/2023 |
| All edits and presentation for RDP Demo | | To-Do | 10/5/2023 | 10/8/2023 |
| All necessary work needed for Preview Demo | | To-Do | 10/8/2023 | 10/26/2023 |
| All recordings needed for the Preview Demo | | To-Do | 10/26/2023 | 10/28/2023 |
| All edits and presentation for Preview Demo | | To-Do | 10/26/2023 | 10/29/2023 |
| All adjustments needed for the Project Manual | | To-Do | 10/29/2023 | 11/5/2023 |
| Project Manual | | To-Do | 11/5/2023 | 11/5/2023 |



# Dependency Graph



# Individual Planner

| | Task Name | Task Description | Expected Hours of Work | Feedback | Status | Starting Date | Due Date |
|---|---|---|---|---|---|---|---|
| **Carlos** | | | | | | | |
| | Conceptual Backend Framework | Building the basic html templates of what our program will be based upon. Basically the bones of our greater program | 5 | Magnificient work!!! | Completed | 9/12/2023 | 9/18/2023 |
| | Module 1 | Set up PostgresSQL database and connection to backend environment | 3 | Very nice work! | Completed | 9/20/2023 | 9/26/2023 |
| | Module 2 | Create .env file to be shared amongst developers and upload to GitHub | 0.5 | Very good! | Completed | 9/20/2023 | 9/26/2023 |
| | Module 3 | Setup CI/CD Pipeline. | 4 | Very nice | Completed | 9/26/2023 | 10/9/2023 |
| | Backend Module 4.1 | Create Task Model and functionality | 3 | Very good! | Completed | 9/26/2023 | 10/11/2023 |
| | Backend Module 5 | Configure CRUD opperations for for tasks | 5 | Wow talk about great | Completed | 10/16/2023 | 10/26/2023 |
| **Julian** | | | | | | | |
| | Team Website | Create a running and public team website which will host all things related to the Red Lobsters | 3 | Great Work! | Completed | 9/12/2023 | 9/18/2023 |
| | Requirements Matrix | Create, review and test requirements utilizng a requirements matrix. this should include bad-dev and bad-client tests to assure that each requirement is testable and full proof. | 4 | Legitimately thank yc | In Progress | 9/21/2023 | 10/26/2023 |
| | Backend Module 4.2 | Created User Model and functionality. | 2 | looking good so far | Completed | 9/26/2023 | 10/11/2023 |
| | Frontend Module 4 | Configure sign in and out | 2 | Excllent | Completed | 10/16/2023 | 10/20/2023 |
| | Frontend Module 5 | Configure the dashboard and filter out any unnecessary components in favor of items more related to Crabbytasks. | 3.5 | Wowie! it's starting tc | Completed | 10/16/2023 | 10/20/2023 |
| | Frontend Module 6 | Finalize the navbar component | 2 | yippee! looking good! | Completed | 9/20/2023 | 10/22/2023 |
| | Frontend Module 7 | Implement the Settings page | 3 | wowie zowie thanks f | Completed | 9/22/2023 | 10/26/2023 |
| | Frontend Module 8 | Implement Larry your pet lobster, including visuals for the pet and a level up feature | 4 | Larry's pretty fresh | Completed | 9/20/2023 | 10/26/2023 |
| **Nick** | | | | | | | |
| | Conceptual Framework Testing | building tests for the basic framework of crabby tasks created within the first week | 3 | Fantastic work!!! | Completed | 9/12/2023 | 9/18/2023 |
| | Frontend Module 1 | Research into HTML and CSS. Look into ways of implementing mascot character. | 3 | Nice Prep | Completed | 9/19/2023 | 10/9/2023 |
| | Frontend Module 2 | Find an overall front end template that can be used to encompass the entire websites front end. | 4 | hooray everything looks so good | Completed | 9/16/2023 | 10/14/2023 |
| | Requirements Testing | Test all requirements present within the functional and systems requirements | 5 | very nice work! really | Completed | 9/26/2023 | 10/26/2023 |
| | Frontend Module 3 | Complete the template for the task manager and calendar | 3.5 | very epic this was inc | Completed | 9/17/2023 | 10/26/2023 |
| | Frontend Module 9 | Configure final visuals of the website so that it matches our groups theming and aesthetics | 2 | great work on the fror | Completed | 10/20/2023 | 10/26/2023 |
| | Test Backend 2 | Test all backend module 2 materials and components | 0.5 | good work! | Completed | 9/26/2023 | 10/3/2023 |
| | Test Backend 3 | Test all backend module 3 materials and components | 1 | Nice | Completed | 10/9/2023 | 10/10/2023 |
| | Test Backend 4 | Test all backend module 4 materials and components | 2 | looking good | Completed | 10/10/2023 | 10/12/2023 |
| | Test Frontend 3 | Test all frontend module 3 materials and components | 1.5 | very nice | Completed | 10/20/2023 | 10/26/2023 |
| | Test Frontend 4 | Test all frontend module 4 materials and components | 1.5 | looks good | Completed | 10/21/2023 | 10/23/2023 |
| | Test Backend 5 | Test all frontend module 5 materials and components | 1.5 | nice tests | Completed | 10/22/2023 | 10/26/2023 |
| | Test Backend 6 | Test all backend module 6 materials and components | 1.5 | glad to have these | Completed | 10/23/2023 | 10/26/2023 |
| | Test Backend 7 | Test all backend module 8 materials and components | 1.5 | good work! | Completed | 10/23/2023 | 10/26/2023 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Emma** | | | | | | | |
| | Conceptual Frontend Framework | Building basic html templates for what our program will look like and will function as the base aesthetic of our program. | 5 | Fantastic job!!! | Completed | 9/12/2023 | 9/18/2023 |
| | RDP Scripting | Gather and create all necessary materials for RDP Demo presentation | 4 | Great script | Completed | 10/5/2023 | 10/7/2023 |
| | RDP Presentation | Create full presentation for the RDP Demo | 4 | fantastic presentatior | Completed | 10/5/2023 | 10/7/2023 |
| | RDP Video Recordings | All necessary recordings that make up the RDP video presentation. This includes recorded segements on the individual Design, Requirements, and planning sections | 4 | everything so far so g | Completed | 10/5/2023 | 10/7/2023 |
| | RDP Video Editing | Edits needed to create a well made demonstration of our material. | 5 | looks great! | Completed | 10/5/2023 | 10/8/2023 |
| | Preview Scripting | Gather and create all necessary materials for Preview Demo presentation | 4 | Fantastic script | Completed | 10/26/2023 | 10/28/2023 |
| | Preview Presentation | Create full presentation for the Preview Demo | 4 | brilliant work on the p | Completed | 1/26/2023 | 10/28/2023 |
| | Preview Recordings | All necessary recordings that make up the Preview Demo. This includes recorded segments on all necessary sections | 4 | recordings look brillia | Completed | 10/26/2023 | 10/28/2023 |
| | Preview Editing | Edits needed to create a well made demonstartion of our material. | 5 | The final product is m | Completed | 10/26/2023 | 10/29/2023 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Graciela** | | | | | | | |
| | Project Booklet Draft 1 | Write, gather, and create materials for the first half of the project booklet. | 4 | So far so good | Complete | 9/19/2023 | 10/26/2023 |
| | Project Booklet Draft 2 | Write, gather, and create materials for the second half of the project booklet. | 4 | Keep it up! | Complete | 9/19/2023 | 10/26/2023 |
| | Finalize Project Booklet | Make necessary edits and formatting changes to the project booklet to ensure that it is a full and concrete reflection of the Red Lobsters and Crabbytasks | 4 | It was fantastic worki | Complete | 10/8/2023 | 11/10/2023 |
| | Requirements Matrix | Create, review and test requirements utilizng a requirements matrix. this should include bad-dev and bad-client tests to assure that each requirement is testable and full proof. | 4 | Thank you very much | Complete | 9/21/2023 | 10/26/2023 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Penny** | | | | | | | |
| | Daily plan and calendar updates | update plan when necessary | N/A | | In Progress | 9/19/2023 | 11/11/2023 |
| | Red Lobsters Plan | The full plan for the Red Lobsters including Calendar and individual view for each member of the Red Lobsters | 5 | :) | Complete | 9/19/2023 | 10/5/2023 |
| | Requirements sheet | Create full sheet of general requirements for the project | 4 | :D | Complete | 9/19/2023 | 10/26/2023 |
| | Requirements Matrix | Create, review and test requirements utilizng a requirements matrix. this should include bad-dev and bad-client tests to assure that each requirement is testable and full proof. | 4 | XD | Complete | 9/21/2023 | 10/26/2023 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Qi** | | | | | | | |
| | Design Document 1 | Create all System Design Diagrams as well as any and all other design diagrams necessary for the full design document | 4 | Great Designs | Completed | 9/19/2023 | 10/5/2023 |
| | Design Document 2 | Create explanations for each diagram and design format for the purpsoes of presenting our design in the RDP Demo | 3 | Nice Work | Completed | 9/19/2023 | 10/5/2023 |
| | Guidance Demo | Create a tutorial for all group members regarding the design to ensure that the Red Lobsters as a whole is on the same page | 4 | Thank you so much | Completed | 9/19/2023 | 10/3/2023 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Austin** | | | | | | | |
| | Team Check-Ins | Daily-to-weekly check ins with group to ensure everything is happening according to plan. | 3 | Thank you for always | In Progress | 9/19/2023 | 11/11/2023 |
| | Requirements Matrix | Create, review and test requirements utilizng a requirements matrix. this should include bad-dev and bad-client tests to assure that each requirement is testable and full proof. | 4 | Honestly your help w | Completed | 9/21/2023 | 10/26/2023 |
| | Module 6 | Create Login and user functionality | 3 | looking great | Completed | 10/16/2023 | 10/22/2023 |
| | Module 7 | Create New tables and models to accomodate any changes in our system | 3 | fantastic tables | Completed | 10/18/2023 | 10/21/2023 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Benny** | | | | | | | |
| | Discord Rework | Reworking and reorganizing the discord server. | 2 | Server looks great! | Complete | 9/12/2023 | 9/18/2023 |
| | Project Booklet Draft 1 | Write, gather, and create materials for the first half of the project booklet. | 4 | So far so good | Complete | 9/19/2023 | 10/26/2023 |
| | Project Booklet Draft 2 | Write, gather, and create materials for the second half of the project booklet. | 4 | Keep it up! | Complete | 9/19/2023 | 10/26/2023 |
| | Business Plan | Write our business plan outlining the Red Lobsters business strategy | 2 | Beautiful work thank | Complete | 10/24/2023 | 10/27/2023 |

# System Overview

## Architecture

The architecture of CrabbyTasks is designed to provide a robust, scalable, and efficient platform for task management. At its core, the system follows a client-server approach, leveraging this carefully chosen technology stack to deliver a seamless user experience.

## Technology

Our application is built using well-rounded and modern technology stacks that ensure both reliability and performance. The key components of the stack include:

- Programming Languages:
  - Python
  - Typescript
- Frameworks and Libraries:
  - Django Rest Framework
  - React
- Database Management System:
  - PostgreSQL
  - SQLite3
- Web Server:
  - Apache
- Cloud Services:
  - Amazon Web Services

## Backend System Configuration

### Modularity

CrabbyTasks  uses __init__.py to structure the system into modules. Any directories containing __init__.py files are recognized as packages or modules, allowing them to be imported and used in other parts of the codebase.

## Dependencies

CrabbyTasks uses several popular dependencies in order to speed up the development process. These dependencies played an important role in meeting deadlines and completing the team's objects.

- asgiref==3.7.2
- boto3==1.28.54
- botocore==1.31.54
- certifi==2023.7.22
- charset-normalizer==3.2.0
- coverage==7.3.2
- Django==4.2.5
- django-appconf==1.0.5
- django-cors-headers==4.3.0
- django-dotenv==1.4.2
- django-filter==23.3
- django-storages==1.14
- djangorestframework==3.14.0
- djangorestframework-simplejwt==5.3.0
- gunicorn==21.2.0
- idna==3.4
- jmespath==1.0.1
- packaging==23.1
- psycopg2-binary==2.9.7
- PyJWT==2.8.0
- python-dateutil==2.8.2
- pytz==2023.3.post1
- rcssmin==1.1.1
- requests==2.31.0
- rjsmin==1.2.1
- s3transfer==0.6.2
- six==1.16.0
- sqlparse==0.4.4

- urllib3==1.26.16

## Frontend System Configuration

### Dependencies:

One of the primary goals behind CrabbyTasks was to create an application that is both contemporary and user-friendly. Throughout the development process, it was of utmost importance to guarantee the realization of this vision, and as a result, we incorporated these dependencies to help achieve this goal.

- "@headlessui/react": "^1.7.17",
- "apexcharts": "^3.41.0",
- "axios": "^1.5.1",
- "dayjs": "^1.11.10",
- "headlessui": "^0.0.0",
- "jsvectormap": "^1.5.3",
- "jwt-decode": "^3.1.2",
- "match-sorter": "^6.3.1",
- "react": "^18.2.0",
- "react-apexcharts": "^1.4.1",
- "react-dom": "^18.2.0",
- "react-hot-toast": "^2.4.1",
- "react-icons": "^4.10.1",
- "react-router-dom": "^6.14.2",
- "react-toastify": "^9.1.3",
- "sort-by": "^0.0.2"

### Environment

In accordance with our vision, we chose to include several files dedicated to crafting an aesthetically appealing website. Each of these files played a significant role in shaping the product we have developed:

- Postcss.config.cjs
- index.css
- Satoshi.css
- tailwind.config.cjs

## System Endpoints

### Admin Panel:

- Admin Panel URL: **/admin/**

### Base App

- Base App URL : **/**
- User-Related API Endpoints: **/api/user/**
- Base API Endpoints : **/api/**

### Authentication-Related Endpoints

- Token Obtain Endpoint: **/api/token/**
- Token Refresh Endpoint: **/api/token/refresh/**
- Token Blacklisting Endpoint (Logout): **/api/logout/blacklist/**

### Task-Related Endpoints

- Retrieve User Tasks: **/api/**
- Create User Task: **/api/create/**
- Update User Task: **/api/update/<str:pk>**
- Delete User Task: **/api/delete/<str:pk>**
- Retrieve User Task by Task ID: **/api/retrieve/<str:task_id>**
- Retrieve Unknown User Task: **/api/search/**
- Retrieve All Tasks: **/api/all/**

User App

- Retrieve User Profile Information: **/users/gather/**
- Create New User: **/users/create/**

# Backend Code

**_backend/base/api**: This directory contains Python files that define the API endpoints for CrabbyTasks. These endpoints handle various CRUD (Create, Read, Update, Delete) operations related to tasks, user profiles, and more. The `serializer.py` file defines how data is serialized and deserialized when interacting with the API. For example, it converts database records into JSON format for client consumption.

**_backend/base/models.py**: This file defines the database models used in CrabbyTasks. It includes the model `task_model`, and others, which represent the structure of data stored in the database. Models specify fields, data types, relationships, and validation rules for database tables.

**_backend/base/views.py**: The `views.py` file contains Python functions that handle incoming HTTP requests from the frontend. These views interact with the database models and serializers to process requests and return responses. For instance, the `TaskViewSet` class defines views for tasks, allowing tasks to be created, updated, or deleted.

**_backend/base/admin.py**: This file configures the Django admin panel. It allows authorized users, typically administrators, to manage application data through a user-friendly interface. Here, you can customize what models are accessible and how they are displayed.

**_backend/config/settings.py**: The `settings.py` file is a crucial configuration file for Django. It includes various settings related to the application, such as database settings, authentication settings, middleware, installed apps, and more. It's the central configuration hub for the entire Django project.

**_backend/users**: This directory manages user-related functionality. The `models.py` file defines the custom user model and additional user-related models. The `views.py` file contains

views for user registration, login, profile management, and authentication. Serializers in this context are used to handle user-related data.

These backend components collectively handle the core functionality of CrabbyTasks, including task management, user authentication, and data storage. They interact with the frontend to provide users with a seamless experience.

## Base/API

filters.py:

This file contains custom filters for querying tasks and other data. It's essential for allowing users to filter and search for tasks based on various criteria, such as status, priority, and deadlines. Filters play a crucial role in providing an efficient task management experience.

```
##############################################################################
# This 'filters.py' file defines filter classes using Django filters for the
# 'task_model' model in the 'base' app. These filters allow for searching and
# filtering tasks based on various fields.
##############################################################################

import django_filters
from ..models import task_model

# Filter class for tasks based on various fields.
class TaskFilter(django_filters.FilterSet):
    status = django_filters.CharFilter(lookup_expr='iexact')
    task = django_filters.CharFilter(lookup_expr='icontains')
    description= django_filters.CharFilter(lookup_expr='icontains')
    created_at= django_filters.CharFilter(lookup_expr='icontains')
    priority= django_filters.CharFilter(lookup_expr='icontains')
    notifications= django_filters.CharFilter(lookup_expr='icontains')
    deadline= django_filters.CharFilter(lookup_expr='icontains')

    class Meta:
        model = task_model
        fields = [
                'status',
                'task',
                'description',
                'created_at',
                'priority',
                'notifications',
                'deadline'
                ]
```

```
# Filter class for tasks based on status.
class StatusFilter(django_filters.FilterSet):
    status = django_filters.CharFilter(lookup_expr='iexact')  # Case-insensitive partial
match

    class Meta:
        model = task_model
        fields = ['status']

# Filter class for tasks based on priority.
class PriorityFilter(django_filters.FilterSet):
    priority = django_filters.CharFilter(lookup_expr='icontains')  # Case-insensitive
partial match

    class Meta:
        model = task_model
        fields = ['priority']
```

## Serializer.py:

This file houses the views for the CrabbyTasks API. Each view corresponds to a specific API endpoint and handles HTTP requests from the frontend. For instance, the RetrieveUserTask class defines views to perform CRUD operations on tasks. These views serve as the interface between the frontend and the database, ensuring data retrieval and manipulation.

```
################################################################################
# This 'serializers.py' file defines serializer classes using Django Rest
# Framework for the 'task_model' model in the 'base' app. These serializers
# handle the conversion of model data to JSON format and vice versa for tasks.
################################################################################

from rest_framework import serializers
from rest_framework_simplejwt.serializers import TokenObtainPairSerializer
from ..models import *
import uuid
```

```python
# JWT Authentication
class MyTokenObtainPairSerializer(TokenObtainPairSerializer):
    @classmethod
    def get_token(cls, user):
        token = super().get_token(user)
        # Sending custom info with JWT
        token['username'] = user.email
        # ...

        return token

# Serializer for retrieving task details
class TaskRetrieveSerializer(serializers.ModelSerializer):
    class Meta:
        model = task_model
        fields = [
                'task_id',
                'user_id',
                'task',
                'description',
                'status',
                'created_at',
                'priority',
                'category',
                'notifications',
                'deadline'
                ]

# Serializer for creating tasks
class TaskCreateSerializer(serializers.ModelSerializer):
    class Meta:
        model = task_model
        fields = [
                'status',
                'task',
                'description',
                'deadline',
                'priority',
                'category',
                'notifications',
                ]
```

```
    def create(self, validated_data):
        user = self.context['request'].user
        task = task_model.objects.create(user_id=user, **validated_data)
        return task

# Serializer for deleting tasks
class TaskDeleteSerializer(serializers.ModelSerializer):
    class Meta:
        model = task_model
        fields = [
                'task_id',
                ]
```

Serializers in this file are responsible for converting complex data types (like database querysets) into Python data types, such as dictionaries, that can be easily rendered into JSON. Serializers are used for data validation, transformation, and facilitating data exchange between the frontend and backend.

APIviews.py:

```
################################################################################ This
'views.py' file defines various API views for managing tasks in the 'base'
# app. It includes views for creating, retrieving, updating, and deleting tasks.
################################################################################

from ..models import *
from .serializer import *
from uuid import UUID

# Django REST API imports
from rest_framework import generics, filters
from rest_framework.response import Response

# Filtering
from django_filters import rest_framework as filters
```

```python
from .filters import *

# JWT AUTHENTICATION
from rest_framework.permissions import SAFE_METHODS, BasePermission, AllowAny,
IsAuthenticated
from rest_framework_simplejwt.views import TokenObtainPairView

# ====================================================
# TOKEN GENERATE
# Custom Token Obtain Pair View
# ====================================================
class MyTokenObtainPairView(TokenObtainPairView):
    permission_classes=[AllowAny]
    serializer_class = MyTokenObtainPairSerializer

# ====================================================
# Object level permission!!
# Custom permission to allow only authors to edit tasks
# ====================================================
class TaskUserWritePermission(BasePermission):
    message = "Only author can edit"
    def has_object_permission(self,request,view,obj):
        if request.method in SAFE_METHODS:
            return True
        return obj.user_id == request.user

#### ====================================================
#### CLASS BASED VIEW
#### ====================================================

# ====================================================
# GET ALL USER TASKS
# Retrieve tasks for the current user
# ====================================================
class RetriveUserTasks(generics.ListAPIView):
    permission_classes=[IsAuthenticated]
    serializer_class = TaskRetrieveSerializer
    def get_queryset(self):
        current_user = self.request.user
        return task_model.objects.filter(user_id=current_user)

# ====================================================
# RETRIEVE SINGLE TASK
```

```python
# Retrieve a single task for the current user
# ====================================================
class RetriveUserTask(generics.RetrieveAPIView):
    permission_classes=[IsAuthenticated]
    serializer_class = TaskRetrieveSerializer
    def get_queryset(self):
        current_user = self.request.user
        return task_model.objects.filter(user_id=current_user)

# ====================================================
# CREATE TASK
# Create a new task for the current user
# ====================================================
class CreateUserTask(generics.CreateAPIView):
    permission_classes=[IsAuthenticated]
    serializer_class = TaskCreateSerializer
    def get_queryset(self):
        current_user = self.request.user
        return task_model.objects.filter(user_id=current_user)

# ====================================================
# UPDATE TASK
# Update a task for the current user
# ====================================================
class UpdateUserTask(generics.RetrieveUpdateAPIView):
    permission_classes=[IsAuthenticated]
    serializer_class = TaskRetrieveSerializer

    def get_queryset(self):
        current_user = self.request.user
        return task_model.objects.filter(user_id=current_user)

    def get_object(self):
        pk = UUID(self.kwargs.get('pk'))
        return task_model.objects.get(task_id=pk)

    def partial_update(self, request, *args, **kwargs):
            instance = self.get_object()
            serializer = self.get_serializer(instance, data=request.data, partial=True)
            serializer.is_valid(raise_exception=True)
            serializer.save()
            return Response(serializer.data)
```

```python
# ====================================================
# DELETE TASK
# Delete a task for the current user
# ====================================================
class DeleteUserTask(generics.RetrieveDestroyAPIView):
    permission_classes=[IsAuthenticated]
    serializer_class = TaskDeleteSerializer

    def get_queryset(self):
        current_user = self.request.user
        return task_model.objects.filter(user_id=current_user)

    def get_object(self):
        pk = UUID(self.kwargs.get('pk'))
        return task_model.objects.get(task_id=pk)

# ====================================================
# SEARCH
# Retrieve tasks with filtering for the current user
# ====================================================
class RetriveUnknownUserTask(generics.ListCreateAPIView):
    permission_classes=[IsAuthenticated]
    serializer_class = TaskRetrieveSerializer
    filter_backends=[filters.DjangoFilterBackend]
    filterset_class= TaskFilter

    def get_queryset(self):
        current_user = self.request.user
        return task_model.objects.filter(user_id=current_user)

# ====================================================
# GET ALL DB TASKS
# Retrieve all tasks (allow any Dev (DEV-ENDPOINTS))
# ====================================================
class RetriveAllTasks(generics.ListCreateAPIView):
    permission_classes=[DEVELOPER]
    queryset = task_model.objects.all()
    serializer_class = TaskRetrieveSerializer

# CRUD DEV TESTS (allow any user)
class RetriveAllTask(generics.RetrieveUpdateDestroyAPIView):
    permission_classes=[DEVELOPER]
    serializer_class = TaskRetrieveSerializer
```

```
    def get_queryset(self):
        task_id = self.request.task_id
        return task_model.objects.all()
```

## Base/Model:

The admin.py file plays a role in integrating CrabbyTasks with Django's admin panel. Admins can use the panel to manage application data without directly accessing the database. The file allows customization of how models are displayed in the admin interface and specifies which models are accessible to admins.

```
##############################################################################
# This 'models.py' file defines the data models for the 'base' app of the Django
# project. It includes a model called 'task_model' that represents tasks with
# various attributes such as status, description, deadline, priority, and more.
##############################################################################

from django.db import models
from django.conf import settings
from django.utils import timezone
import uuid

# Choices for task status.
STATUSES = (
    ('ToDo', 'To Do'),
    ('In-Progress', 'In-Progress'),
    ('Complete', 'Complete'),
    ('Overdue', 'Overdue'),
)

# Choices for task notifications.
NOTIFICATIONS = (
    ('None', 'None'),
    ('Desktop Only', 'Desktop Only'),
    ('Email Only', 'Email Only'),
```

```
    ('Both', 'Both'),
)

# Choices for task priorities.
PRIORITIES = (
    ('1', 'Highest'),
    ('2', 'High'),
    ('3', 'Medium'),
    ('4', 'Low'),
    ('5', 'Lowest'),
)

# Choices for task categories.
CATEGORIES = (
    ('Work', 'Work'),
    ('School', 'School'),
    ('Chores', 'Chores'),
    ('Personal', 'Personal'),
    ('Other', 'Other'),
)


# Model for representing tasks.
class task_model(models.Model):
    user_id         = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
#hidden
    task_id         = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    status          = models.CharField(max_length=11, choices=STATUSES, default='ToDo',
editable=True)
    task            = models.CharField(max_length=30, editable=True)
    description     = models.CharField(max_length=250, editable=True, default="insert
description")
    created_at      = models.DateTimeField('Created', default=timezone.now)
    updated_at      = models.DateTimeField('Updated', auto_now=True)
    deadline        = models.DateTimeField()
    priority        = models.CharField(max_length=7, choices=PRIORITIES, default=2)
    category        = models.CharField(max_length=8, choices=CATEGORIES, default='Other')
    notifications   = models.CharField(max_length=12, choices=NOTIFICATIONS, default='None')

     # Custom manager for querying only 'ToDo' status tasks.
    class tasks_objects(models.Manager):
        def for_user(self, user):
            return super().get_queryset().filter(user_id=user)
```

```
    objects = models.Manager()
    tasks_objects = tasks_objects()

    class Meta:
        ordering = ['-status',]

    def __str__(self):
           return self.task
```

This file defines Django models for CrabbyTasks. Models represent the structure of the database tables and define fields for storing data. For example, the Task model defines fields such as status, task, description, created_at, updated_at, deadline, priority, notifications, and user_id_id. These models are the backbone of the application's data storage and retrieval.

## Base/Admin:

```
##############################################################################
# This 'admin.py' file defines the admin interface for the 'base' app of the
# Django project. It registers the 'task_model' model and customizes its
# display in the admin panel.
##############################################################################

from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from django.forms import TextInput, Textarea, CharField
from django import forms
from .models import *

# Register your models here.

# Register the 'task_model' model in the admin panel.
@admin.register(task_model)
class UserAdmin(admin.ModelAdmin):

list_display=('task','description','status','priority','deadline','notifications','user_id')
```

## Config/Settings:

This central configuration file contains settings that affect the entire Django project. It incorporates various components into the system, such as database connections, middleware, authentication, installed apps, and more. Changes to settings here can impact the behavior of the entire application.

```
################################################################################
# This 'settings.py' file contains the Django project's configuration settings.
# It includes settings related to the project's database, security, static files,
# internationalization, authentication, REST framework, CORS headers, and more.
################################################################################


from pathlib import Path
from datetime import timedelta

import os
# Define the base directory of the project.
BASE_DIR = Path(__file__).resolve().parent.parent

# Define the project's secret key.
SECRET_KEY = os.environ.get("DJANGO_SECRET_KEY")

# Set debug mode based on the "DEBUG" environment variable.
DEBUG = str(os.environ.get("DEBUG")) == "1"

# Define allowed hosts for the project.
ALLOWED_HOSTS = ["*","red-lobster.dev"]

# Define installed apps for the project.
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'base.apps.MainConfig',
```

```
    'users.apps.UsersConfig',
    'rest_framework',
    'rest_framework_simplejwt',
    'rest_framework_simplejwt.token_blacklist',
    'corsheaders', #Cross Origin Resource Sharing Headers
    'django_filters',
]

# Define middleware for the project.
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

# Define the project's root URL configuration.
ROOT_URLCONF = 'config.urls'

# Define templates for the project.
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'base/templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

# Define the WSGI application for the project.
WSGI_APPLICATION = 'config.wsgi.application'
```

```python
# Database
# Define the database settings based on environment variables.
DB_ENGINE               =os.environ.get("DB_ENGINE")

DB_PRODUCTION_NAME      =os.environ.get("DB_PRODUCTION_NAME")
DB_PRODUCTION_USERNAME  =os.environ.get("DB_PRODUCTION_USER")
DB_PRODUCTION_PASSWORD  =os.environ.get("DB_PRODUCTION_PASSWORD")
DB_PRODUCTION_HOST      =os.environ.get("DB_PRODUCTION_HOST")
DB_PRODUCTION_PORT      =os.environ.get("DB_PRODUCTION_PORT")

DB_DEVELOPER_NAME       =os.environ.get("DB_DEVELOPER_NAME")
DB_DEVELOPER_USERNAME   =os.environ.get("DB_DEVELOPER_USER")
DB_DEVELOPER_PASSWORD   =os.environ.get("DB_DEVELOPER_PASSWORD")
DB_DEVELOPER_HOST       =os.environ.get("DB_DEVELOPER_HOST")
DB_DEVELOPER_PORT       =os.environ.get("DB_DEVELOPER_PORT")

PROD_IS_AVAIL = all([
    DB_PRODUCTION_NAME,
    DB_PRODUCTION_USERNAME,
    DB_PRODUCTION_PASSWORD,
    DB_PRODUCTION_HOST,
    DB_PRODUCTION_PORT,
])

DEV_IS_AVAIL = all([
    DB_DEVELOPER_NAME,
    DB_DEVELOPER_USERNAME,
    DB_DEVELOPER_PASSWORD,
    DB_DEVELOPER_HOST,
    DB_DEVELOPER_PORT,
])

if DEBUG:
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.sqlite3',
            'NAME': BASE_DIR / 'db.sqlite3',
        }
    }
else:
    DATABASES = {
            'default': {
```

```
                'ENGINE':DB_ENGINE,
                'NAME':DB_PRODUCTION_NAME,
                'USER':DB_PRODUCTION_USERNAME,
                'PASSWORD':DB_PRODUCTION_PASSWORD,
                'HOST':DB_PRODUCTION_HOST,
                'PORT':DB_PRODUCTION_PORT,
                }
            }

# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators
# Define password validation settings.
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/
# Define internationalization settings.
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/
# Define static files settings.
if not DEBUG:
    STATIC_ROOT = '/var/www/static'
```

```
STATIC_URL = 'static/'

# Define the default primary key field type.
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'


# Define settings for REST framework, including authentication.
SIMPLE_JWT = {
    "ACCESS_TOKEN_LIFETIME": timedelta(minutes=100),
    "REFRESH_TOKEN_LIFETIME": timedelta(days=1),
    "ROTATE_REFRESH_TOKENS": False,
    "BLACKLIST_AFTER_ROTATION": True,
    "UPDATE_LAST_LOGIN": True,

    "ALGORITHM": "HS256",
    "SIGNING_KEY": SECRET_KEY,
    "VERIFYING_KEY": "",
    "AUDIENCE": None,
    "ISSUER": None,
    "JSON_ENCODER": None,
    "JWK_URL": None,
    "LEEWAY": 0,

    "AUTH_HEADER_TYPES": ("Bearer",),
    "AUTH_HEADER_NAME": "HTTP_AUTHORIZATION",
    "USER_ID_FIELD": "id",
    "USER_ID_CLAIM": "user_id",
    "USER_AUTHENTICATION_RULE":
"rest_framework_simplejwt.authentication.default_user_authentication_rule",

    "AUTH_TOKEN_CLASSES": ("rest_framework_simplejwt.tokens.AccessToken",),
    "TOKEN_TYPE_CLAIM": "token_type",
    "TOKEN_USER_CLASS": "rest_framework_simplejwt.models.TokenUser",

    "JTI_CLAIM": "jti",

    "SLIDING_TOKEN_REFRESH_EXP_CLAIM": "refresh_exp",
    "SLIDING_TOKEN_LIFETIME": timedelta(minutes=5),
    "SLIDING_TOKEN_REFRESH_LIFETIME": timedelta(days=1),

    # "TOKEN_OBTAIN_SERIALIZER":
    "TOKEN_OBTAIN_SERIALIZER": "my_app.serializers.MyTokenObtainPairSerializer",
```

```
    "TOKEN_REFRESH_SERIALIZER":
"rest_framework_simplejwt.serializers.TokenRefreshSerializer",
    "TOKEN_VERIFY_SERIALIZER": "rest_framework_simplejwt.serializers.TokenVerifySerializer",
    "TOKEN_BLACKLIST_SERIALIZER":
"rest_framework_simplejwt.serializers.TokenBlacklistSerializer",
    "SLIDING_TOKEN_OBTAIN_SERIALIZER":
"rest_framework_simplejwt.serializers.TokenObtainSlidingSerializer",
    "SLIDING_TOKEN_REFRESH_SERIALIZER":
"rest_framework_simplejwt.serializers.TokenRefreshSlidingSerializer",
}

APPEND_SLASH=False

#Rest Framework settings
AUTH_USER_MODEL = "users.new_user_model"

#https://www.django-rest-framework.org
REST_FRAMEWORK = {
        'DEFAULT_PERMISSION_CLASSES':[
            'rest_framework.permissions.IsAuthenticated',
            ],
        'DEFAULT_AUTHENTICATION_CLASSES': (
            'rest_framework_simplejwt.authentication.JWTAuthentication',
            ),
        }

# Configure CORS settings for cross-origin resource sharing.
# CORS_ALLOWED_ORIGINS = ALLOWED_ORIGINS
```

## Users

The users directory within the CrabbyTasks backend is responsible for managing all aspects related to user accounts, authentication, and user-specific data. It's a fundamental component of the system that ensures secure and personalized user experiences. Here's a breakdown of its key elements:

models.py:

This file defines custom user models and other models related to user data. Custom user models allow CrabbyTasks to tailor the user data storage according to the application's specific requirements. It includes fields like email, user_name, start_date, about, and others, which help create and maintain user profiles and authentication information.

```
################################################################################
# This 'models.py' file defines Django models for the 'users' app. It includes
# the 'custom_account_manager_model' class, which manages user creation,
# particularly the creation of superusers. It also defines the 'new_user_model'
# class, which represents the user model, including fields like email, username,
# first name, start date, about, and various permissions and flags.
################################################################################

from django.db import models
from django.utils import timezone
from django.utils.translation import gettext_lazy as _
from django.contrib.auth.models import AbstractBaseUser, PermissionsMixin, BaseUserManager

# CustomAccountManagerModel manages user creation, including superusers.
class custom_account_manager_model(BaseUserManager):

    def create_superuser(self, email, user_name, first_name, password, **other_fields):

        other_fields.setdefault('is_staff', True)
        other_fields.setdefault('is_superuser', True)
        other_fields.setdefault('is_active', True)

        if other_fields.get('is_staff') is not True:
            raise ValueError(
                'Superuser must be assigned to is_staff=True.')
        if other_fields.get('is_superuser') is not True:
            raise ValueError(
                'Superuser must be assigned to is_superuser=True.')

        return self.create_user(email, user_name, first_name, password, **other_fields)

    def create_user(self, email, user_name, first_name, password, **other_fields):
```

```
        if not email:
            raise ValueError(_('You must provide an email address'))
        email = self.normalize_email(email)
        user = self.model(email=email, user_name=user_name, first_name=first_name,
**other_fields)
        user.set_password(password)
        user.save()
        return user

#Changed the user_model that was originally in base.api
# new_user_model represents the user model with various fields and flags.
class new_user_model(AbstractBaseUser, PermissionsMixin):

    email = models.EmailField(_('email address'), unique=True)
    user_name = models.CharField(max_length=150, unique=True)
    first_name = models.CharField(max_length=150, blank=True)
    start_date = models.DateTimeField(default=timezone.now)
    about = models.TextField(_('about'), max_length=500, blank=True)
    is_staff = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)

    objects = custom_account_manager_model()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['user_name', 'first_name']

    def __str__(self):
        return self.user_name
```

views.py:

The views within this file handle various user-related operations, including user registration, authentication, profile management, and interactions with user-specific data. For example, user registration views facilitate the creation of new user accounts, while authentication views manage the login and logout processes. Profile management views allow users to update their profiles, including personal information and preferences. These views ensure that users can interact with the application securely and efficiently.

```python
################################################################################
# This 'views.py' file defines various Django Rest Framework (DRF) views for
# handling user-related operations, authentication, and profile retrieval.
# It includes views for user registration, token blacklist, and user profile
# retrieval.
################################################################################

from rest_framework_simplejwt.views import TokenObtainPairView
from rest_framework import status
from rest_framework.response import Response
from rest_framework.views import APIView
from rest_framework import generics
from . serializers import CustomUserSerializer, UserProfileSerializer
from . models import new_user_model
from rest_framework_simplejwt.tokens import RefreshToken
from rest_framework.permissions import AllowAny, IsAuthenticated


# CustomUserCreate view allows user registration with a POST request.
class CustomUserCreate(APIView):
    permission_classes = [AllowAny]

    def post(self, request, format='json'):
        serializer = CustomUserSerializer(data=request.data)
        if serializer.is_valid():
            user = serializer.save()
            if user:
                json = serializer.data
                return Response(json, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


# BlacklistTokenUpdateView allows blacklisting a JWT token.
class BlacklistTokenUpdateView(APIView):
    permission_classes = [AllowAny]
    # authentication_classes = ()

    def post(self, request):
        try:
            refresh_token = request.data["refresh_token"]
            token = RefreshToken(refresh_token)
            token.blacklist()
            return Response(status=status.HTTP_205_RESET_CONTENT)
```

```
        except Exception as e:
            return Response(status=status.HTTP_400_BAD_REQUEST)

# RetriveUserTask view retrieves user profiles for authenticated users.
class RetriveUserTask(generics.RetrieveAPIView):
    permission_classes=[IsAuthenticated]
    serializer_class = UserProfileSerializer
    def get_queryset(self):
        current_user = self.request.user
        return new_user_model.objects.filter(user_id=current_user)
```

## serializers.py:

Serializers in this file are vital for data serialization and deserialization related to user-specific data. They ensure that data exchanged between the frontend and backend is in a structured and secure format. For instance, serializers are used to validate and format user registration data, ensuring that user accounts are created accurately and securely. They play a crucial role in maintaining data integrity and consistency.

```
################################################################################
# This 'serializers.py' file contains Django Rest Framework (DRF) serializers
# for the 'users' app. It defines two serializers: 'CustomUserSerializer' and
# 'UserProfileSerializer,' primarily used for user-related data serialization.
################################################################################

from rest_framework import serializers
from .models import new_user_model

# CustomUserSerializer serializes user registration data.
class CustomUserSerializer(serializers.ModelSerializer):
    """
    Currently unused in preference of the below.
```

```python
    """
    email = serializers.EmailField(required=True)
    user_name = serializers.CharField(required=True)
    password = serializers.CharField(min_length=8, write_only=True)

    class Meta:
        model = new_user_model
        fields = ('email', 'user_name', 'password')
        extra_kwargs = {'password': {'write_only': True}}

    def create(self, validated_data):
        password = validated_data.pop('password', None)
        # as long as the fields are the same, we can just use this
        instance = self.Meta.model(**validated_data)
        if password is not None:
            instance.set_password(password)
        instance.save()
        return instance

# UserProfileSerializer serializes user profile data.
class UserProfileSerializer(serializers.ModelSerializer):
    class Meta:
        model: new_user_model
        fields = ('email', 'user_name', 'first_name')
```

# Frontend Code

## COMPONENTS

### Breadcrumb.tsx

The `Breadcrumb` component is an essential part of the system's user interface. It plays a crucial role in providing users with a clear and intuitive navigation experience. This React component generates a breadcrumb navigation element that helps users understand their current location within the application.

By taking a 'pageName' prop, it dynamically displays the current page's title, ensuring that users are aware of their context. The breadcrumb also includes a link to the dashboard, allowing users to navigate back to the main dashboard easily.

Incorporated with CSS classes for styling and utilizing the 'react-router-dom' library for routing, this component offers a straightforward and flexible solution for including breadcrumbs in web pages. Its importance lies in enhancing user-friendliness, reducing navigation confusion, and making the application more user-centric by providing clear contextual information.

```
/**********************************************************************
 * Component: Breadcrumb
 *
 * Logic: This file defines a React component that generates a breadcrumb
 * navigation element. It takes a 'pageName' prop and displays it as the
current
 * page title. The breadcrumb also contains a link to the dashboard and the
 * current page name. The component's structure is designed with CSS classes
for
 * styling and uses the 'react-router-dom' library for routing. It provides a
 * simple, flexible way to include breadcrumbs in web pages.
 *
 * Type: Component
```

```
**********************************************************************/

import { Link } from 'react-router-dom';
interface BreadcrumbProps {
  pageName: string;
}
const Breadcrumb = ({ pageName }: BreadcrumbProps) => {
  return (
    <div className="mb-6 flex flex-col gap-3 sm:flex-row sm:items-center
sm:justify-between">
      <h2 className="text-title-md2 font-semibold text-black
dark:text-white">
        {pageName}
      </h2>

      <nav>
        <ol className="flex items-center gap-2">
          <li>
            <Link to="/">Dashboard /</Link>
          </li>
          <li className="text-primary">{pageName}</li>
        </ol>
      </nav>
    </div>
  );
};

export default Breadcrumb;
```

## Card{One-Five}.tsx

These card components serve a similar purpose within the application. They are designed to display user task statistics in a visually appealing card format, providing users with quick insights into their task completion performance.

Each card imports the 'useTaskContext' hook to access user task statistics and uses the 'useEffect' hook to handle any necessary side effects. These components log userTasksStats and calculate a task completion score based on user statistics, which is then displayed on the card. The cards feature rounded borders, background colors, and shadows to enhance their visual appeal and align with the application's design.

Their importance lies in offering users an at-a-glance view of their task completion progress, fostering motivation, and helping users track their productivity. These components contribute to a user-friendly and informative user interface, making task management more engaging and efficient.

```
/**********************************************************************
 * Component: CardOne - CardFive
 *
 * Logic: These files define a React component for displaying task statistics in
 * a card format. It imports the 'useTaskContext' hook to access user task
 * statistics and 'useEffect' for handling side effects. Inside the component,
 * it logs userTasksStats and calculates a task completion score based on user
 * statistics. The card has a rounded border, background color, and shadow for
 * styling.
 *
 * Type: Component
**********************************************************************/

import { useEffect } from 'react';
import useTaskContext from '../hooks/useTaskContext';

const CardOne = (props) => {
```

```
  const { userTasksStats } = useTaskContext();

  useEffect(()=>{
    },[]);

  return (
    <div className="rounded-sm border border-stroke bg-white py-6 px-7.5 shadow-default
dark:border-strokedark dark:bg-boxdark">
      <div className="flex h-11.5 w-11.5 items-center justify-center rounded-full
bg-[#fa9dca] dark:bg-[#fa9dca]">
        <svg width="18px" height="18px" viewBox="0 -0.5 1025 1025" className="icon"
fill="white" version="1.1" xmlns="http://www.w3.org/2000/svg">
       </svg>
      </div>

      <div className="mt-4 flex items-end justify-between">
        <div>
          <h4 className="text-title-md font-bold text-black dark:text-white">
           {userTasksStats?.level + (userTasksStats?.completed * 2)}
          </h4>
          <span className="text-sm font-medium">Task Completion Score</span>
        </div>
      </div>
    </div>
  );
};

export default CardOne;
```

## ChartThree.tsx

The 'ChartThree.tsx' component plays a role in the system by displaying task analytics using a donut chart. This component imports essential dependencies like 'ApexOptions' for configuring

the chart and 'ReactApexChart' for rendering it. It also utilizes other required interfaces and hooks for efficient functionality.

The logic within this component sets up chart options and initializes chart data using the 'useTaskContext' hook, which allows it to access user task statistics and data. The component calculates and displays the percentage of tasks that are 'Completed,' 'To-Do,' 'Overdue,' and 'In-Progress' in an easily digestible format.

By presenting task analytics in a visually appealing and informative manner, 'ChartThree.tsx' aids users in understanding their task management performance. It provides a quick overview of task completion status, helping users make informed decisions and improve their productivity within the application.

```
/**********************************************************************
 * Component: ChartThree
 *
 * Logic: This file defines a React component for displaying task analytics
 * using a donut chart. It imports necessary dependencies, including
 * 'ApexOptions' for chart configuration, and 'ReactApexChart' for rendering the
 * chart. The component also imports other required interfaces and
 * hooks. It sets up chart options and initializes chart data using the
 * 'useTaskContext' hook. The component calculates and displays the percentage
 * of tasks completed, to-do, overdue, and in-progress.
 *
 * Type: Component
**********************************************************************/

import { ApexOptions } from 'apexcharts';
import React, { useState } from 'react';
import ReactApexChart from 'react-apexcharts';

import { useEffect, useContext } from 'react';
import { ComponentType } from 'react';

import { Task } from '../interfaces/Task';
import useTaskContext from '../hooks/useTaskContext';
```

```
interface ChartThreeState {
  series: number[];
}

const options: ApexOptions = {
  chart: {
    type: 'donut',
  },
  colors: ['#10B981', '#375E83', '#259AE6', '#FFA70B'],
  labels: ['Complete', 'To-Do', 'Overdue', 'In-Progress'],
  legend: {
    show: true,
    position: 'bottom',
  },

  plotOptions: {
    pie: {
      donut: {
        size: '65%',
        background: 'transparent',
      },
    },
  },
  dataLabels: {
    enabled: false,
  },
  responsive: [
    {
      breakpoint: 2600,
      options: {
        chart: {
          width: 380,
        },
      },
    },
    {
      breakpoint: 640,
      options: {
        chart: {
          width: 200,
        },
      },
```

```
    },
  ],
};

const ChartThree: React.FC = () => {
  const {userTasks, userTasksStats } = useTaskContext();

  const chartItems = [
    {
      title: "Completed",
      percent: 65,
      color:'Primary',
      },
    {
      title: "Incomplete",
      percent: 34,
      color:'Primary',
      },
    {
      title: "Past",
      percent: 12,
      color:'Primary',
      },
    {
      title: "Past",
      percent: 56,
      color:'Primary',
      },
    ];

  const [state, setState] = useState<ChartThreeState>({
    series: [userTasksStats.completed, userTasksStats.todo, userTasksStats.overdue ,
userTasksStats.inProgress],
  });

  useEffect(()=>{
    setState({
      series: [
      userTasksStats.completed,
      userTasksStats.todo,
      userTasksStats.overdue ,
      userTasksStats.inProgress
      ],
```

```
    });
    },[])

  return (
    <div className="col-span-12 rounded-sm border border-stroke bg-white px-5 pt-7.5 pb-5
shadow-default dark:border-strokedark dark:bg-boxdark sm:px-7.5 xl:col-span-4">
      <div className="mb-3 justify-between gap-4 sm:flex">
        <div>
          <h5 className="text-xl font-semibold text-black dark:text-white">
            Task Analytics
          </h5>
        </div>
      </div>

      <div className="mb-2">
        <div id="chartThree" className="mx-auto flex justify-center">
          <ReactApexChart
            options={options}
            series={state.series}
            type="donut"
          />
        </div>
      </div>

      <div className="-mx-8 flex flex-wrap items-center justify-center gap-y-3">
        <div className="w-full px-8 sm:w-1/2">
          <div className="flex w-full items-center">
            <span className="mr-2 block h-3 w-full max-w-3 rounded-full bg-[#10B981]"></span>
            <p className="flex w-full justify-between text-sm font-medium text-black
dark:text-white">
              <span> Completed </span>
              <span> {userTasks?.length > 0 ? ((state.series.at(0) / userTasks.length ) *
100).toFixed(0) : 0}% </span>
            </p>
          </div>
        </div>

        <div className="w-full px-8 sm:w-1/2">
          <div className="flex w-full items-center">
            <span className="mr-2 block h-3 w-full max-w-3 rounded-full bg-[#375E83]"></span>
            <p className="flex w-full justify-between text-sm font-medium text-black
dark:text-white">
              <span> To-Do </span>
```

```
              <span> {userTasks?.length  > 0 ? ((state.series.at(1) / userTasks.length ) *
100).toFixed(0) : 0}% </span>
            </p>
          </div>
        </div>

        <div className="w-full px-8 sm:w-1/2">
          <div className="flex w-full items-center">
            <span className="mr-2 block h-3 w-full max-w-3 rounded-full bg-[#259AE6]"></span>
            <p className="flex w-full justify-between text-sm font-medium text-black
dark:text-white">
              <span> Overdue </span>
              <span> {userTasks?.length  > 0 ? ((state.series.at(2) / userTasks.length ) *
100).toFixed(0) : 0}% </span>
            </p>
          </div>
        </div>

        <div className="w-full px-8 sm:w-1/2">
          <div className="flex w-full items-center">
            <span className="mr-2 block h-3 w-full max-w-3 rounded-full bg-[#FFA70B]"></span>
            <p className="flex w-full justify-between text-sm font-medium text-black
dark:text-white">
              <span> In-Progress </span>
              <span> {userTasks?.length  > 0 ? ((state.series.at(3) / userTasks.length ) *
100).toFixed(0) : 0}% </span>
            </p>
          </div>
        </div>

      </div>

    </div>
  );
};

export default ChartThree;
```

## ChartTwo.tsx

The 'ChartTwo.tsx' component plays a crucial role in the system by displaying a stacked bar chart representing weekly tasks' completion status. This component imports various dependencies such as 'ApexOptions', 'React', 'useState', and 'ReactApexChart' to create and render the chart effectively.

The logic within this component involves fetching data for 'Completed' and 'Incomplete' tasks and presenting them in the form of a stacked bar chart. The chart's appearance and behavior are configured using the 'options' object, which defines various settings such as colors, chart type, responsiveness, and data labels.

The component enhances the user experience by providing a visual representation of task completion progress over a week. Users can quickly assess their performance and track task completion trends, making it an essential element for effective task management within the application.

```
/**********************************************************************
 * WORK-IN-PROGRESS:
 *   Attach the task context to update automatically
 *
 * Component: ChartTwo
 *
 * Logic: This file defines a React component for displaying a stacked bar chart
 * representing weekly tasks. It imports dependencies such as 'ApexOptions',
 * 'React', 'useState', and 'ReactApexChart'. The component fetches data for
 * 'Completed' and 'Incomplete' tasks and displays them as a stacked bar chart.
 *
 * Type: Component
**********************************************************************/

import { ApexOptions } from 'apexcharts';
import React, { useState } from 'react';
import ReactApexChart from 'react-apexcharts';
```

```
const options: ApexOptions = {
  colors: ['#3C50E0', '#80CAEE'],
  chart: {
    fontFamily: 'Satoshi, sans-serif',
    type: 'bar',
    height: 335,
    stacked: true,
    toolbar: {
      show: false,
    },
    zoom: {
      enabled: false,
    },
  },

  responsive: [
    {
      breakpoint: 1536,
      options: {
        plotOptions: {
          bar: {
            borderRadius: 0,
            columnWidth: '25%',
          },
        },
      },
    },
  ],
  plotOptions: {
    bar: {
      horizontal: false,
      borderRadius: 0,
      columnWidth: '25%',
      borderRadiusApplication: 'end',
      borderRadiusWhenStacked: 'last',
    },
  },
  dataLabels: {
    enabled: false,
  },

  xaxis: {
```

```
    categories: ['M', 'T', 'W', 'T', 'F', 'S', 'S'],
  },
  legend: {
    position: 'top',
    horizontalAlign: 'left',
    fontFamily: 'Satoshi',
    fontWeight: 500,
    fontSize: '14px',

    markers: {
      radius: 99,
    },
  },
  fill: {
    opacity: 1,
  },
};

interface ChartTwoState {
  series: {
    name: string;
    data: number[];
  }[];
}

const ChartTwo: React.FC = () => {
  const [state, setState] = useState<ChartTwoState>({
    series: [
      {
        name: 'Completed',
        data: [3,4,1,3,4,2,7],
      },
      {
        name: 'Incomplete',
        data: [5,1,2,1,2,6,3],
      },
    ],
  });

  return (
    <div className="col-span-12 rounded-sm border border-stroke bg-white p-7.5 shadow-default
dark:border-strokedark dark:bg-boxdark xl:col-span-4">
      <div className="mb-4 justify-between gap-4 sm:flex">
```

```
        <div>
          <h4 className="text-xl font-semibold text-black dark:text-white">
            Weekly Tasks
          </h4>
        </div>
        <div>
          <div className="relative z-20 inline-block">
            <select
              name="#"
              id="#"
              className="relative z-20 inline-flex appearance-none bg-transparent py-1 pl-3
pr-8 text-sm font-medium outline-none"
            >
              <option value="">This Week</option>
              <option value="">Last Week</option>
            </select>
            <span className="absolute top-1/2 right-3 z-10 -translate-y-1/2">
              <svg width="10" height="6" viewBox="0 0 10 6" fill="none"
xmlns="http://www.w3.org/2000/svg" >
              </svg>
            </span>
          </div>
        </div>
      </div>

      <div>
        <div id="chartTwo" className="-ml-5 -mb-9">
          <ReactApexChart
            options={options}
            series={state.series}
            type="bar"
            height={370}
          />
        </div>
      </div>
    </div>
  );
};

export default ChartTwo;
```

## CheckboxFour.tsx

The 'CheckboxFour.tsx' component serves the purpose of rendering a pair of checkboxes with labels, mainly used in the 'Notifications Settings' page. The component uses local state to track the checked state of both checkboxes and toggles them when one is clicked.

'CheckboxFour.tsx' enhances the user experience by providing an intuitive interface for managing notification preferences, allowing users to easily customize their notification settings according to their preferences and needs.

```
/**********************************************************************
 * Component: CheckboxFour
 *
 * Logic: This file defines a React component for rendering a pair of checkboxes
 * with labels. It uses the state to track the checked state of both checkboxes
 * and toggles them when one is clicked.
 *
 * Settings page: Notifications Settings
 *
 * Type: Component
**********************************************************************/

import { useState } from 'react';

const CheckboxFour = () => {
  const [firIsChecked, firSetIsChecked] = useState<boolean>(true);
  const [secIsChecked, secSetIsChecked] = useState<boolean>(false);

  return (
    <div>
      <label
        htmlFor="checkboxLabelFour"
        className="flex cursor-pointer select-none items-center mb-4"
      >
        <div className="relative">
          <input
            type="checkbox"
```

```
          id="checkboxLabelFour"
          className="sr-only"
          onChange={() => {
            secSetIsChecked(!secIsChecked);
            firSetIsChecked(!firIsChecked);
          }}
        />
        <div
          className={`mr-4 flex h-5 w-5 items-center justify-center rounded-full border ${
            firIsChecked && 'border-primary'
          }`}
        >
          <span
            className={`h-2.5 w-2.5 rounded-full bg-transparent ${
              firIsChecked && '!bg-primary'
            }`}
          >
            {' '}
          </span>
        </div>
      </div>
      Desktop Notifications
    </label>
    <label
      htmlFor="checkboxLabelFour_sec"
      className="flex cursor-pointer select-none items-center"
    >
      <div className="relative2">
        <input
          type="checkbox"
          id="checkboxLabelFour_sec"
          className="sr-only"
          onChange={() => {
            secSetIsChecked(!secIsChecked);
            firSetIsChecked(!firIsChecked);

          }}
        />
        <div
          className={`mr-4 flex h-5 w-5 items-center justify-center rounded-full border ${
            secIsChecked && 'border-primary'
          }`}
        >
```

```
            <span
              className={`h-2.5 w-2.5 rounded-full bg-transparent ${
                secIsChecked && '!bg-primary'
              }`}
            >
              {' '}
            </span>
          </div>
        </div>
        No Notifications
      </label>
    </div>
  );
};

export default CheckboxFour;
```

## DarkModeSwitcher.tsx

The DarkModeSwitcher component is a React component that renders a dark mode switcher toggle. It allows users to switch between dark and light modes in the application.

```
/**************************************************************************
 * Component: DarkModeSwitcher
 *
 * Logic: This file defines a React component for rendering a dark mode switcher
 * toggle. It uses the "useColorMode" hook to toggle between dark and light
 * modes.
 *
 * Type: Component
**************************************************************************/


import useColorMode from '../hooks/useColorMode';

const DarkModeSwitcher = () => {
```

```
  const [colorMode, setColorMode] = useColorMode();

  return (
    <li>
      <label
        className={`relative m-0 block h-7.5 w-14 rounded-full ${
          colorMode === 'dark' ? 'bg-primary' : 'bg-stroke'
        }`}
      >
        <input
          type="checkbox"
          onChange={() => {
            if (typeof setColorMode === 'function') {
              setColorMode(colorMode === 'light' ? 'dark' : 'light');
            }
          }}
          className="dur absolute top-0 z-50 m-0 h-full w-full cursor-pointer opacity-0"
        />
        <span
          className={`absolute top-1/2 left-[3px] flex h-6 w-6 -translate-y-1/2 translate-x-0
items-center justify-center rounded-full bg-white shadow-switcher duration-75 ease-linear ${
            colorMode === 'dark' && '!right-[3px] !translate-x-full'
          }`}
        >
          <span className="dark:hidden">
            <svg
              width="16"
              height="16"
              viewBox="0 0 16 16"
              fill="none"
              xmlns="http://www.w3.org/2000/svg"
            >

            </svg>
          </span>
          <span className="hidden dark:inline-block">
            <svg
              width="16"
              height="16"
              viewBox="0 0 16 16"
              fill="none"
              xmlns="http://www.w3.org/2000/svg"
            >
```

```
            </svg>
          </span>
        </span>
      </label>
    </li>
  );
};

export default DarkModeSwitcher;
```

## DropdownDefault.tsx

The 'DropdownDefault.tsx' component is responsible for rendering a dropdown menu that can be opened and closed by clicking a button. It also automatically closes when clicking outside of the dropdown or pressing the "Esc" key.

'DropdownDefault.tsx' enhances user interactions and navigation by providing an accessible and responsive dropdown menu that simplifies user interface elements and improves the overall user experience.

```
/**********************************************************************
 * Component: DropdownDefault
 *
 * Logic: This file defines a React component for rendering a dropdown menu. The
 * dropdown menu can be opened and closed by clicking a button. It also closes
 * when clicking outside of it or pressing the "Esc" key.
 *
 * Type: Component
**********************************************************************/


import { useEffect, useRef, useState } from "react";
```

```
const DropdownDefault = () => {
  const [dropdownOpen, setDropdownOpen] = useState(false);

  const trigger = useRef<any>(null);
  const dropdown = useRef<any>(null);

  // close on click outside
  useEffect(() => {
    const clickHandler = ({ target }: MouseEvent) => {
      if (!dropdown.current) return;
      if (
        !dropdownOpen ||
        dropdown.current.contains(target) ||
        trigger.current.contains(target)
      )
        return;
      setDropdownOpen(false);
    };
    document.addEventListener("click", clickHandler);
    return () => document.removeEventListener("click", clickHandler);
  });

  // close if the esc key is pressed
  useEffect(() => {
    const keyHandler = ({ keyCode }: KeyboardEvent) => {
      if (!dropdownOpen || keyCode !== 27) return;
      setDropdownOpen(false);
    };
    document.addEventListener("keydown", keyHandler);
    return () => document.removeEventListener("keydown", keyHandler);
  });

  return (
    <div className="relative">
      <button ref={trigger} onClick={() => setDropdownOpen(!dropdownOpen)}>
        <svg
          width="18"
          height="18"
          viewBox="0 0 18 18"
          fill="none"
          xmlns="http://www.w3.org/2000/svg"
        >
                </svg>
```

```
      </button>
      <div
        ref={dropdown}
        onFocus={() => setDropdownOpen(true)}
        onBlur={() => setDropdownOpen(false)}
        className={`absolute right-0 top-full z-40 w-40 space-y-1 rounded-sm border
border-stroke bg-white p-1.5 shadow-default dark:border-strokedark dark:bg-boxdark ${
          dropdownOpen === true ? "block" : "hidden"
        }`}
      >
        <button className="flex w-full items-center gap-2 rounded-sm py-1.5 px-4 text-left
text-sm hover:bg-gray dark:hover:bg-meta-4">
          <svg
            className="fill-current"
            width="16"
            height="16"
            viewBox="0 0 16 16"
            fill="none"
            xmlns="http://www.w3.org/2000/svg"
          >
            <g clipPath="url(#clip0_62_9787)"></g>
            <defs>
              <clipPath id="clip0_62_9787">
                <rect width="16" height="16" fill="white" />
              </clipPath>
            </defs>
          </svg>
          Edit
        </button>
        <button className="flex w-full items-center gap-2 rounded-sm py-1.5 px-4 text-left
text-sm hover:bg-gray dark:hover:bg-meta-4">
          <svg
            className="fill-current"
            width="16"
            height="16"
            viewBox="0 0 16 16"
            fill="none"
            xmlns="http://www.w3.org/2000/svg"
          >
          </svg>
          Delete
        </button>
      </div>
```

```
    </div>
  );
};

export default DropdownDefault;
```

## DropdownNotification.tsx

The 'DropdownNotification.tsx' component is responsible for rendering a notification dropdown menu that can be opened and closed by clicking a link icon. It also automatically closes when clicking outside of the dropdown or pressing the "Esc" key.

'DropdownNotification.tsx' enhances the user experience by providing an accessible and responsive notification system, allowing users to stay informed about important events or updates within the application.

```
/**********************************************************************
 * WORK-IN-PROGRESS:
 *    Notification Context needs to be added
 *
 * Component: DropdownNotification
 *
 * Logic: This file defines a React component for rendering a notification
 * dropdown menu. The dropdown menu can be opened and closed by clicking a link
 * icon. It also closes when clicking outside of it or pressing the "Esc" key.
 *
 * Type: Component
**********************************************************************/


import { useEffect, useRef, useState } from 'react';
import { Link } from 'react-router-dom';
```

```
const DropdownNotification = () => {
  const [dropdownOpen, setDropdownOpen] = useState(false);

  const trigger = useRef<any>(null);
  const dropdown = useRef<any>(null);

  useEffect(() => {
    const clickHandler = ({ target }: MouseEvent) => {
      if (!dropdown.current) return;
      if (
        !dropdownOpen ||
        dropdown.current.contains(target) ||
        trigger.current.contains(target)
      )
        return;
      setDropdownOpen(false);
    };
    document.addEventListener('click', clickHandler);
    return () => document.removeEventListener('click', clickHandler);
  });

  // close if the esc key is pressed
  useEffect(() => {
    const keyHandler = ({ keyCode }: KeyboardEvent) => {
      if (!dropdownOpen || keyCode !== 27) return;
      setDropdownOpen(false);
    };
    document.addEventListener('keydown', keyHandler);
    return () => document.removeEventListener('keydown', keyHandler);
  });

  return (
    <li className="relative">
      <Link
        ref={trigger}
        onClick={() => setDropdownOpen(!dropdownOpen)}
        to="#"
        className="relative flex h-8.5 w-8.5 items-center justify-center rounded-full
border-[0.5px] border-stroke bg-gray hover:text-primary dark:border-strokedark dark:bg-meta-4
dark:text-white"
      >
        <span className="absolute -top-0.5 right-0 z-1 h-2 w-2 rounded-full bg-meta-1">
          <span className="absolute -z-1 inline-flex h-full w-full animate-ping rounded-full
```

```
bg-meta-1 opacity-75"></span>
        </span>

        <svg
          className="fill-current duration-300 ease-in-out"
          width="18"
          height="18"
          viewBox="0 0 18 18"
          fill="none"
          xmlns="http://www.w3.org/2000/svg"
        >

        </svg>
      </Link>

      <div
        ref={dropdown}
        onFocus={() => setDropdownOpen(true)}
        onBlur={() => setDropdownOpen(false)}
        className={`absolute -right-27 mt-2.5 flex h-90 w-75 flex-col rounded-sm border
border-stroke bg-white shadow-default dark:border-strokedark dark:bg-boxdark sm:right-0
sm:w-80 ${
          dropdownOpen === true ? 'block' : 'hidden'
        }`}
      >
        <div className="px-4.5 py-3">
          <h5 className="text-sm font-medium text-bodydark2">Notification</h5>
        </div>

        <ul className="flex h-auto flex-col overflow-y-auto">
          {/* <-- Notification List item -->*/}
          <li>
            <Link
              className="flex flex-col gap-2.5 border-t border-stroke px-4.5 py-3
hover:bg-gray-2 dark:border-strokedark dark:hover:bg-meta-4"
              to="#"
            >
              <p className="text-sm">
                <span className="text-black dark:text-white">
                  You have a notification
                </span>{' '}
                This is the test notification that the user will receive
              </p>
```

```
              <p className="text-xs">Sun Nov 5, 2023</p>
            </Link>
          </li>
          {/* <-- Notification List item -->*/}
        </ul>
      </div>
    </li>
  );
};

export default DropdownNotification;
```

## DropdownUser.tsx

The 'DropdownUser.tsx' component represents a user account dropdown menu, allowing users to access their account settings and log out of their account. Users can toggle the dropdown open and closed by clicking on their user profile section. Additionally, the dropdown closes when clicking outside of it or pressing the "Esc" key, enhancing user experience and usability.

```
/**********************************************************************
 * WORK-IN-PROGRESS:
 *    Dynamically add the user settings based on AuthContext
 *
 * Component: DropdownUser
 *
 * Logic: This file defines a React component that represents a user account dropdown menu.
 * It allows users to access their account settings and log out of their account.
 * The dropdown can be toggled open and closed by clicking on the user profile section.
 * It also closes when clicking outside of it or pressing the "Esc" key.
 *
 * Type: Component
**********************************************************************/

import { useEffect, useRef, useState, useContext } from 'react';
import AuthContext from '../context/AuthContext';
import { useNavigate } from 'react-router-dom';
```

```
import { Link } from 'react-router-dom';
import Larry from "../images/icon/larrylvl1.png"

const DropdownUser = () => {

  const {logoutUser} = useContext(AuthContext);
  const [dropdownOpen, setDropdownOpen] = useState(false);
  const trigger = useRef<any>(null);
  const dropdown = useRef<any>(null);

  // close on click outside
  useEffect(() => {
    const clickHandler = ({ target }: MouseEvent) => {
      if (!dropdown.current) return;
      if (
        !dropdownOpen ||
        dropdown.current.contains(target) ||
        trigger.current.contains(target)
      )
        return;
      setDropdownOpen(false);
    };
    document.addEventListener('click', clickHandler);
    return () => document.removeEventListener('click', clickHandler);
  });

  // close if the esc key is pressed
  useEffect(() => {
    const keyHandler = ({ keyCode }: KeyboardEvent) => {
      if (!dropdownOpen || keyCode !== 27) return;
      setDropdownOpen(false);
    };
    document.addEventListener('keydown', keyHandler);
    return () => document.removeEventListener('keydown', keyHandler);
  });

  return (
    <div className="relative">
      <Link
        ref={trigger}
        onClick={() => setDropdownOpen(!dropdownOpen)}
        className="flex items-center gap-4"
        to="#"
```

```
      >
        <span className="hidden text-right lg:block">
          <span className="block text-sm font-medium text-black dark:text-white">
            Carlos
          </span>
          <span className="block text-xs">tapiacarlos32@gmail.com</span>
        </span>

        <span className="h-12 w-12 flex justify-center rounded-full">
          <img src={Larry} alt="User" />
        </span>

        <svg
          className={`hidden fill-current sm:block ${
            dropdownOpen ? 'rotate-180' : ''
          }`}
          width="12"
          height="8"
          viewBox="0 0 12 8"
          fill="none"
          xmlns="http://www.w3.org/2000/svg"
        >
                </svg>
      </Link>

      {/* <!-- Dropdown Start --> */}
      <div
        ref={dropdown}
        onFocus={() => setDropdownOpen(true)}
        onBlur={() => setDropdownOpen(false)}
        className={`absolute right-0 mt-4 flex w-62.5 flex-col rounded-sm border
border-stroke bg-white shadow-default dark:border-strokedark dark:bg-boxdark ${
          dropdownOpen === true ? 'block' : 'hidden'
        }`}
      >
        <ul className="flex flex-col gap-5 border-b border-stroke px-6 py-7.5
dark:border-strokedark">
          <li>
            <Link
              to="/settings"
              className="flex items-center gap-3.5 text-sm font-medium duration-300
ease-in-out hover:text-primary lg:text-base"
            >
```

```
              <svg
                className="fill-current"
                width="22"
                height="22"
                viewBox="0 0 22 22"
                fill="none"
                xmlns="http://www.w3.org/2000/svg"
              >

              </svg>
              Account Settings
            </Link>
          </li>
        </ul>
        <button className="flex items-center gap-3.5 py-4 px-6 text-sm font-medium
duration-300 ease-in-out hover:text-primary lg:text-base" onClick={logoutUser} >

          </svg>
          Log Out
        </button>
      </div>
      {/* <!-- Dropdown End --> */}
    </div>
  );
};

export default DropdownUser;
```

## Header.tsx

The 'Header.tsx' component represents the website's header and provides a navigation menu, a dark mode switcher, and user-related dropdowns for notifications and user account settings. The header's display is conditionally controlled based on the current page.

```
/*********************************************************************
```

```
 * WORK-IN-PROGRESS:
 *    Add auth context to change the signed in user
 *
 * Component: Header
 *
 * Logic: This file defines a React component for the website's header. It
 * includes a navigation menu, a dark mode switcher, and user-related dropdowns
 * for notifications and user account settings. The header is conditionally
 * displayed based on the current page.
 *
 * Type: Component
**********************************************************************/

import { Link, useLocation } from 'react-router-dom';
import DarkModeSwitcher from './DarkModeSwitcher';
import DropdownNotification from './DropdownNotification';
import DropdownUser from './DropdownUser';

const Header = (props: {
  sidebarOpen: string | boolean | undefined;
  setSidebarOpen: (arg0: boolean) => void;
}) => {

  //Code for search bar only being on TaskManager Page:
  const location = useLocation(); //Get current page
  const headerPaths = ['/taskManager']; // Define a list of paths where you want the header
to appear
  const shouldDisplayHeader = headerPaths.includes(location.pathname); //Determine whether
current location is within list of specified paths

  return (
    <header className="sticky top-0 z-999 flex w-full bg-white drop-shadow-1 dark:bg-boxdark
dark:drop-shadow-none">
      <div className="flex flex-grow items-center justify-between py-4 px-4 shadow-2 md:px-6
2xl:px-11">
        <div className="flex items-center gap-2 sm:gap-4 lg:hidden">
          {/* <!-- Hamburger Toggle BTN --> */}
          <button
            aria-controls="sidebar"
            onClick={(e) => {
              e.stopPropagation();
              props.setSidebarOpen(!props.sidebarOpen);
            }}
```

```
            className="z-99999 block rounded-sm border border-stroke bg-white p-1.5 shadow-sm
dark:border-strokedark dark:bg-boxdark lg:hidden"
          >
            <span className="relative block h-5.5 w-5.5 cursor-pointer">
              <span className="du-block absolute right-0 h-full w-full">
                <span
                  className={`relative top-0 left-0 my-1 block h-0.5 w-0 rounded-sm bg-black
delay-[0] duration-200 ease-in-out dark:bg-white ${!props.sidebarOpen && '!w-full delay-300'
                    }`}
                ></span>
                <span
                  className={`relative top-0 left-0 my-1 block h-0.5 w-0 rounded-sm bg-black
delay-150 duration-200 ease-in-out dark:bg-white ${!props.sidebarOpen && 'delay-400 !w-full'
                    }`}
                ></span>
                <span
                  className={`relative top-0 left-0 my-1 block h-0.5 w-0 rounded-sm bg-black
delay-200 duration-200 ease-in-out dark:bg-white ${!props.sidebarOpen && '!w-full delay-500'
                    }`}
                ></span>
              </span>
              <span className="absolute right-0 h-full w-full rotate-45">
                <span
                  className={`absolute left-2.5 top-0 block h-full w-0.5 rounded-sm bg-black
delay-300 duration-200 ease-in-out dark:bg-white ${!props.sidebarOpen && '!h-0 !delay-[0]'
                    }`}
                ></span>
                <span
                  className={`delay-400 absolute left-0 top-2.5 block h-0.5 w-full rounded-sm
bg-black duration-200 ease-in-out dark:bg-white ${!props.sidebarOpen && '!h-0 !delay-200'
                    }`}
                ></span>
              </span>
            </span>
          </button>
          {/* <!-- Hamburger Toggle BTN --> */}

        </div>

        <div className="hidden sm:block">
          <form action="https://formbold.com/s/unique_form_id" method="POST">

          {shouldDisplayHeader && (
```

```
            <div className="relative">
              <button className="absolute top-1/2 left-0 -translate-y-1/2">
                <svg
                  className="fill-body hover:fill-primary dark:fill-bodydark
dark:hover:fill-primary"
                  width="20"
                  height="20"
                  viewBox="0 0 20 20"
                  fill="none"
                  xmlns="http://www.w3.org/2000/svg"
                >

                </svg>
              </button>

              <input
                type="text"
                placeholder="Type to search..."
                className="w-full bg-transparent pr-4 pl-9 focus:outline-none"
              />
            </div>
            )}

          </form>
        </div>

        <div className="flex items-center gap-3 2xsm:gap-7">
          <ul className="flex items-center gap-2 2xsm:gap-4">
            {/* <!-- Dark Mode Toggler --> */}
            <DarkModeSwitcher />
            {/* <!-- Dark Mode Toggler --> */}

            {/* <!-- Notification Menu Area --> */}
              {/*<DropdownNotification />*/}
            {/* <!-- Notification Menu Area --> */}

            {/* <!-- Chat Notification Area --> */}
            {/* <DropdownMessage /> */}
            {/* <!-- Chat Notification Area --> */}
          </ul>

          {/* <!-- User Area --> */}
          <DropdownUser />
```

```
          {/* <!-- User Area --> */}
        </div>
      </div>
    </header>
  );
};

export default Header;
```

## ModalSettings.tsx

The ModalSettings.tsx component defines a React modal component for user settings. It provides a form that allows users to input settings related to User settings. This modal is intended for use in the User Settings page.

```
/**********************************************************************
 * WORK-IN-PROGRESS:
 *    Update the user settings through this form (Settings page)
 *
 * Component: ModalSettings
 *
 * Logic: This file defines a React component for a settings modal.
 * The modal allows users to input settings related to User settings.
 *
 * Used in User Settings
 *
 * Type: Component
**********************************************************************/



import React, { useState } from "react";
import dataJSON from '../../public/data.json';

export const Modal = ({ closeModal, onSubmit, defaultValue }) => {
```

```
  const [formState, setFormState] = useState(
    defaultValue || {
      email: "",
      username: "price",
    }
  );
  const [errors, setErrors] = useState<string[]>([]);

  const handleChange = (e) => {
    setFormState({ ...formState, [e.target.name]: e.target.value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    onSubmit(formState);
    closeModal();
  };

  return (
    <div
      className="modal-container fixed z-50 flex top-25 bottom-5 "
      onClick={(e) => {
        if (e.target.className === "modal-container") closeModal();
      }}
    >

      <div className="modal rounded-sm border border-stroke bg-white shadow-default
dark:border-strokedark dark:bg-boxdark overflow-auto">
      <div className="border-b border-stroke py-4 px-7 dark:border-strokedark">
        <div className="w-full flex justify-end">
      <strong className="text-xl align-center cursor-pointer "
      onClick={closeModal}
      >&times;</strong>
      </div>
        <form>
        <div className="grid grid-cols-3 gap-5 justify-normal">
          <div className="form-group w-full col-span-3">
            <label  className="mb-3 block text-sm font-medium text-black dark:text-white"
htmlFor="id">Bond ID (Input "ALL" to track all bonds with paramaters below)</label>
            <input className="w-full rounded border border-stroke bg-gray py-3 pl-3 pr-4.5
text-black focus:border-primary focus-visible:outline-none dark:border-strokedark
dark:bg-meta-4 dark:text-white dark:focus:border-primary" name="email"
```

```
onChange={handleChange} value={formState.email} />
          </div>

          <div className="form-group ">
            <label className="mb-3 block text-sm font-medium text-black dark:text-white"
htmlFor="para">Parameter</label>
            <div className="relative z-20 w-full rounded border border-stroke p-1.5 pr-8
font-medium outline-none transition focus:border-primary active:border-primary
dark:border-form-strokedark dark:bg-form-input">
                  <div className="flex flex-wrap items-center"></div>
                  <span className="m-1.5 flex items-center justify-center rounded
border-[.5px] border-stroke bg-gray py-1.5 px-2.5 text-sm font-medium dark:border-strokedark
dark:bg-white/30">
                      {formState.username}
                  </span>
                    <span className="absolute top-1/2 right-4 z-10 -translate-y-1/2">
                      <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                        <g opacity="0.8">
                        </g>
                      </svg>
                    </span>
                  </div>
            </div>

          <br></br>
          <button className="btn flex justify-center rounded bg-primary py-2 px-6 font-medium
text-gray hover:shadow-1"
                      type="submit" onClick={handleSubmit}>
            Submit
          </button>
        </form>
      </div>
      </div>
    </div>

  );
};
```

## Sidebar.tsx

The Sidebar.tsx component defines a React sidebar component for navigation within the application. It manages the open/close state of the sidebar, responds to user interactions, and provides links to different sections of the application.

```
/**********************************************************************
 * WORK-IN-PROGRESS:
 *    Add the pet page
 *
 * Component: Sidebar.tsx
 *
 * Logic: This component creates a sidebar for navigation within the
 * application. It manages the open/close state of the sidebar, responds to user
 * interactions, and provides links to different sections of the application.
 * The behavior of the sidebar depends on the user's actions and the state of
 * the application.
 *
 * Type: Component
**********************************************************************/


import React, { useEffect, useRef, useState } from 'react';
import { NavLink, useLocation } from 'react-router-dom';
import Logo from "../images/logo/CrabbyTasksLogo.png"
import SidebarLinkGroup from './SidebarLinkGroup';

interface SidebarProps {
  sidebarOpen: boolean;
  setSidebarOpen: (arg: boolean) => void;
}

const Sidebar = ({ sidebarOpen, setSidebarOpen }: SidebarProps) => {
  const location = useLocation();
  const { pathname } = location;

  const trigger = useRef<any>(null);
  const sidebar = useRef<any>(null);
```

```
const storedSidebarExpanded = localStorage.getItem('sidebar-expanded');
const [sidebarExpanded, setSidebarExpanded] = useState(
  storedSidebarExpanded === null ? false : storedSidebarExpanded === 'true'
);

// close on click outside
useEffect(() => {
  const clickHandler = ({ target }: MouseEvent) => {
    if (!sidebar.current || !trigger.current) return;
    if (
      !sidebarOpen ||
      sidebar.current.contains(target) ||
      trigger.current.contains(target)
    )
      return;
    setSidebarOpen(false);
  };
  document.addEventListener('click', clickHandler);
  return () => document.removeEventListener('click', clickHandler);
});

// close if the esc key is pressed
useEffect(() => {
  const keyHandler = ({ keyCode }: KeyboardEvent) => {
    if (!sidebarOpen || keyCode !== 27) return;
    setSidebarOpen(false);
  };
  document.addEventListener('keydown', keyHandler);
  return () => document.removeEventListener('keydown', keyHandler);
});

useEffect(() => {
  localStorage.setItem('sidebar-expanded', sidebarExpanded.toString());
  if (sidebarExpanded) {
    document.querySelector('body')?.classList.add('sidebar-expanded');
  } else {
    document.querySelector('body')?.classList.remove('sidebar-expanded');
  }
}, [sidebarExpanded]);

return (
  <aside
    ref={sidebar}
```

```
      className={`absolute left-0 top-0 z-9999 flex h-screen w-72.5 flex-col
overflow-y-hidden bg-black duration-300 ease-linear dark:bg-boxdark lg:static
lg:translate-x-0 ${
        sidebarOpen ? 'translate-x-0' : '-translate-x-full'
      }`}
    >
      {/* <!-- SIDEBAR HEADER --> */}
      <div className="flex items-center max-w-[290px] gap-2 px-6 py-5.5 lg:py-6.5">
        <NavLink to="/">
          <img src={Logo} alt="Logo" />
        </NavLink>

        <button
          ref={trigger}
          onClick={() => setSidebarOpen(!sidebarOpen)}
          aria-controls="sidebar"
          aria-expanded={sidebarOpen}
          className="block lg:hidden"
        >
        </button>
      </div>
      {/* <!-- SIDEBAR HEADER --> */}

      <div className="no-scrollbar flex flex-col overflow-y-auto duration-300 ease-linear">
        {/* <!-- Sidebar Menu --> */}
        <nav className="mt-5 py-4 px-4 lg:mt-9 lg:px-6">
          {/* <!-- Menu Group --> */}
          <div>
            <h3 className="mb-4 ml-4 text-sm font-semibold text-bodydark2">
              MENU
            </h3>

            <ul className="mb-6 flex flex-col gap-1.5">
              {/* <!-- Menu Item Dashboard --> */}
              <SidebarLinkGroup
                activeCondition={
                  pathname === '/' || pathname.includes('dashboard')
                }
              >
                {(handleClick, open) => {
                  return (
                    <React.Fragment>
                      <NavLink
```

```
                        to="#"
                        className={`group relative flex items-center gap-2.5 rounded-sm py-2
px-4 font-medium text-bodydark1 duration-300 ease-in-out hover:bg-graydark
dark:hover:bg-meta-4 ${
                          (pathname === '/' ||
                            pathname.includes('dashboard')) &&
                          'bg-graydark dark:bg-meta-4'
                        }`}
                        onClick={(e) => {
                          e.preventDefault();
                          sidebarExpanded
                            ? handleClick()
                            : setSidebarExpanded(true);
                        }}
                      >
                        <svg
                          className="fill-current"
                          width="18"
                          height="18"
                          viewBox="0 0 18 18"
                          fill="none"
                          xmlns="http://www.w3.org/2000/svg"
                        >

                        </svg>
                        Dashboard
                        <svg
                          className={`absolute right-4 top-1/2 -translate-y-1/2 fill-current
${
                            open && 'rotate-180'
                          }`}
                          width="20"
                          height="20"
                          viewBox="0 0 20 20"
                          fill="none"
                          xmlns="http://www.w3.org/2000/svg"
                        >
                        </svg>
                      </NavLink>
                      {/* <!-- Dropdown Menu Start --> */}
                      <div
                        className={`translate transform overflow-hidden ${
                          !open && 'hidden'
```

```
                        }`}
                      >
                        <ul className="mt-4 mb-5.5 flex flex-col gap-2.5 pl-6">
                          <li>
                            <NavLink
                              to="/home"
                              className={({ isActive }) =>
                                'group relative flex items-center gap-2.5 rounded-md px-4
font-medium text-bodydark2 duration-300 ease-in-out hover:text-white ' +
                                (isActive && '!text-white')
                              }
                            >
                              Home
                            </NavLink>
                          </li>
                        </ul>
                      </div>
                      {/* <!-- Dropdown Menu End --> */}
                    </React.Fragment>
                  );
                }}
              </SidebarLinkGroup>
              {/* <!-- Menu Item Dashboard --> */}

              {/* <!-- Menu Item Task Manager --> */}
              <li>
                <NavLink
                  to="/taskManager"
                  className={`group relative flex items-center gap-2.5 rounded-sm py-2 px-4
font-medium text-bodydark1 duration-300 ease-in-out hover:bg-graydark dark:hover:bg-meta-4 ${
                    pathname.includes('calendar') &&
                    'bg-graydark dark:bg-meta-4'
                  }`}
                >
                  <svg
                    className="fill-current"
                    width="18"
                    height="18"
                    viewBox="0 0 18 18"
                    fill="none"
                    xmlns="http://www.w3.org/2000/svg"
                  >
```

```
                </svg>
                Task Manager
              </NavLink>
            </li>
            {/* <!-- Menu Item Task Manager --> */}

            {/* <!-- Menu Item Profile --> */}
            <li>
              <NavLink
                to="/profile"
                className={`group relative flex items-center gap-2.5 rounded-sm py-2 px-4
font-medium text-bodydark1 duration-300 ease-in-out hover:bg-graydark dark:hover:bg-meta-4 ${
                  pathname.includes('profile') && 'bg-graydark dark:bg-meta-4'
                }`}
              >
                <svg
                  className="fill-current"
                  width="18"
                  height="18"
                  viewBox="0 0 18 18"
                  fill="none"
                  xmlns="http://www.w3.org/2000/svg"
                >
                </svg>
                Contact Us
              </NavLink>
            </li>
            {/* <!-- Menu Item Profile --> */}

            {/*<!-- Menu Item Pet -->
            <li>
              <NavLink
                to="/pet"
                className={`group relative flex items-center gap-2.5 rounded-sm py-2 px-4
font-medium text-bodydark1 duration-300 ease-in-out hover:bg-graydark dark:hover:bg-meta-4 ${
                  pathname.includes('pet') && 'bg-graydark dark:bg-meta-4'
                }`}
              >
                <svg
                  className="fill-current"
                  width="18"
                  height="19"
                  viewBox="0 0 18 19"
```

```
                    fill="none"
                    xmlns="http://www.w3.org/2000/svg"
                  >
                    <g clipPath="url(#clip0_130_9756)">

                    </g>
                    <defs>
                      <clipPath id="clip0_130_9756">
                        <rect
                          width="18"
                          height="18"
                          fill="white"
                          transform="translate(0 0.052124)"
                        />
                      </clipPath>
                    </defs>
                  </svg>
                  Pet
                </NavLink>
              </li>
              {/*<!-- Menu Item Pet -->*/}

              {/* <!-- Menu Item Settings --> */}
              <li>
                <NavLink
                  to="/settings"
                  className={`group relative flex items-center gap-2.5 rounded-sm py-2 px-4
font-medium text-bodydark1 duration-300 ease-in-out hover:bg-graydark dark:hover:bg-meta-4 ${
                    pathname.includes('settings') &&
                    'bg-graydark dark:bg-meta-4'
                  }`}
                >
                  <svg
                    className="fill-current"
                    width="18"
                    height="19"
                    viewBox="0 0 18 19"
                    fill="none"
                    xmlns="http://www.w3.org/2000/svg"
                  >
                    <g clipPath="url(#clip0_130_9763)">
                    </g>
                    <defs>
```

```
                    <clipPath id="clip0_130_9763">
                      <rect
                        width="18"
                        height="18"
                        fill="white"
                        transform="translate(0 0.052124)"
                      />
                    </clipPath>
                  </defs>
                </svg>
                Settings
              </NavLink>
            </li>
            {/* <!-- Menu Item Settings --> */}
          </ul>
        </div>


      </nav>
      {/* <!-- Sidebar Menu --> */}
    </div>
  </aside>
 );
};

export default Sidebar;
```

## SidebarLinkGroup.tsx

The SidebarLinkGroup.tsx component defines a React component for a collapsible sidebar link group. This component is used to create groups of sidebar links that can be expanded or collapsed by the user when clicked.

```
/**********************************************************************
 * Component: SidebarLinkGroup
```

```
 *
 * Logic: This file defines a React component for a collapsible sidebar link
 * group. The component handles toggling the group's open/closed state when a
 * user clicks on it.
 *
 * Type: Component
*********************************************************************/


import { ReactNode, useState } from 'react';

interface SidebarLinkGroupProps {
  children: (handleClick: () => void, open: boolean) => ReactNode;
  activeCondition: boolean;
}

const SidebarLinkGroup = ({
  children,
  activeCondition,
}: SidebarLinkGroupProps) => {
  const [open, setOpen] = useState<boolean>(activeCondition);

  const handleClick = () => {
    setOpen(!open);
  };

  return <li>{children(handleClick, open)}</li>;
};

export default SidebarLinkGroup;
```

## TaskChart.tsx

The TaskChart.tsx component defines a React component for displaying a list of tasks in a table format. This component is used to render tasks, including their due dates, and provides options for editing and deleting tasks.

```
/*********************************************************************
 * Component: TaskChart
 *
 * Logic: This file defines a React component for displaying a list of tasks in
 * a table format. It handles tasks' due dates and provides options for editing
 * and deleting tasks.
 *
 * Used in Home page
 *
 * Type: Component
*********************************************************************/

import { useEffect, useState, useContext, useRef } from 'react';
import React, { ComponentType } from 'react';
import dayjs from 'dayjs';
import Loader from '../common/Loader';

import { Task } from '../interfaces/Task';
import AuthContext from '../context/AuthContext';
import useTaskContext from '../hooks/useTaskContext';
import TaskModal from '../components/TaskModal';

const TaskChart = (props) => {
    // loading state for taskData updated
    const [loading, setLoading] = useState(true);

    // Only get current day items
    const { date } = props;

    // Passing task to modal via state
    const [ taskData, useTaskData ] = useState(null);
    const [ taskToEdit, setTaskToEdit ] = useState(null);
    const [ isModalVisible, setIsModalVisible ] = useState(false);

    // userTasks
    const { userTasks, setUserTasks, updatedUserTasks, deleteUserTask, updateUserTask } =
useTaskContext();


    const holdDate = date;

    const handleDelete  = (task_id) =>{
        deleteUserTask(task_id);
```

```
      }

    const handleEditTask = (taskObj) =>{
        setTaskToEdit(taskObj);
        setIsModalVisible(true);
      }

    const handleModalClose = () =>{
        setTaskToEdit({});
        setIsModalVisible(false);
      }

    useEffect(()=>{
      setLoading(false);
        },[userTasks])

    return (
    <>
      <div className="rounded-sm border border-stroke bg-white px-5 pt-6 pb-2.5
shadow-default dark:border-strokedark dark:bg-boxdark sm:px-7.5 xl:pb-1">
        <div className="max-w-full overflow-x-auto">
          <table className="w-full table-auto">
            <thead>
              <tr className="bg-gray-2 text-left dark:bg-meta-4">
                <th className="min-w-[370px] py-4 px-4 font-medium text-black dark:text-white
xl:pl-11">
                  Task
                </th>
                <th className="min-w-[120px] py-4 px-4 font-medium text-black
dark:text-white">
                  Due Date
                </th>
                <th className="py-4 px-4 font-medium text-black dark:text-white">
                  Actions
                </th>
              </tr>
            </thead>
            <tbody>
            {loading ? ( <tr><td col-span="3" >Loading... </td></tr> ) : (  userTasks?.length
> 0 ? ( userTasks.map((task,index)=>(
                <tr key={task.task_id}>
                  <td className="border-b border-[#eee] py-5 px-4 pl-9 dark:border-strokedark
xl:pl-11">
```

```
                    <h5 className="font-medium text-black dark:text-white"> {task.task} </h5>
                    <p className="text-sm">{task.description}</p>
                  </td>
                  <td className="border-b border-[#eee] py-5 px-4 dark:border-strokedark">
                    <p className="text-black
dark:text-white">{dayjs(task.deadline.toString().substring(0,10)).format('ddd MMM DD,
YYYY').toString()}</p>
                  </td>
                  <td className="border-b border-[#eee] py-5 px-4 dark:border-strokedark">
                    <div className="flex items-center space-x-3.5">
                      <button className="hover:text-primary"
onClick={()=>{handleEditTask(task)}}>
                       <svg width="18px" height="18px" viewBox="0 0 1024 1024"
className="icon" version="1.1" xmlns="http://www.w3.org/2000/svg">
                         </svg>
                       </button>
                       <button className="hover:text-primary"
onClick={()=>handleDelete(task.task_id)} >
                         <svg width="18px" height="18px" viewBox="0 0 1024 1024"
className="icon"  version="1.1" xmlns="http://www.w3.org/2000/svg">
                        </svg>
                      </button>
                    </div>
                  </td>
                </tr>
            )):(
              <tr>
                <td className="border-b border-[#eee] py-5 px-4 pl-9 dark:border-strokedark
xl:pl-11">
                  <h5 className="font-medium text-black dark:text-white"> No Tasks Set </h5>
                  <p className="text-sm"> Add one! </p>
                </td>
                <td className="border-b border-[#eee] py-5 px-4 dark:border-strokedark">
                  <p className="text-black dark:text-white"> None </p>
                </td>
                <td className="border-b border-[#eee] py-5 px-4 dark:border-strokedark">
                  <div className="flex items-center space-x-3.5">
                    <button className="hover:text-primary">
                      <svg className="fill-current" width="18" height="18" viewBox="0 0 18
18" fill="none" xmlns="http://www.w3.org/2000/svg" >
                       </svg>
                    </button>
                  </div>
```

```
                </td>
              </tr>
            ))}
            </tbody>
          </table>
        </div>
      </div>
      {isModalVisible && <TaskModal taskData={taskToEdit} onClose={handleModalClose}/>}
    </>
    );
  };

  export default TaskChart;
```

## TaskHeader.tsx

The TaskHeader.tsx component defines a React component for displaying task-related headers and actions. This component includes options for adding tasks and interacting with user avatars.

```
/**************************************************************************
 * WORK-IN-PROGRESS:
 *    Center the Popup, and fix components
 *
 * Component: TaskHeader
 *
 * Logic: This file defines a React component for displaying task-related
 * headers and actions. It includes options for adding tasks and interacting
 * with user avatars.
 *
 * Type: Component
**************************************************************************/


import { useEffect, useRef, useState } from 'react';
```

```
const TaskHeader = () => {
  const [popupOpen, setPopupOpen] = useState(false);

  const trigger = useRef<any>(null);
  const popup = useRef<any>(null);

  // close on click outside
  useEffect(() => {
    const clickHandler = ({ target }: MouseEvent) => {
      if (!popup.current) return;
      if (
        !popupOpen ||
        popup.current.contains(target) ||
        trigger.current.contains(target)
      )
        return;
      setPopupOpen(false);
    };
    document.addEventListener('click', clickHandler);
    return () => document.removeEventListener('click', clickHandler);
  });

  // close if the esc key is pressed
  useEffect(() => {
    const keyHandler = ({ keyCode }: KeyboardEvent) => {
      if (!popupOpen || keyCode !== 27) return;
      setPopupOpen(false);
    };
    document.addEventListener('keydown', keyHandler);
    return () => document.removeEventListener('keydown', keyHandler);
  });

  return (
    <div className="flex flex-col gap-y-4 rounded-sm border border-stroke bg-white p-3
shadow-default dark:border-strokedark dark:bg-boxdark sm:flex-row sm:items-center
sm:justify-between">
      <div>
        <h3 className="pl-2 text-title-lg font-semibold text-black dark:text-white">
          Tasks
        </h3>
      </div>
      <div className="flex flex-col gap-4 2xsm:flex-row 2xsm:items-center">
```

```
        <div className="flex -space-x-2">
          <button className="h-9 w-9 rounded-full border-2 border-white dark:border-boxdark">
            <img src={userSeven} alt="User" />
          </button>
          <button className="h-9 w-9 rounded-full border-2 border-white dark:border-boxdark">
            <img src={userEight} alt="User" />
          </button>
          <button className="h-9 w-9 rounded-full border-2 border-white dark:border-boxdark">
            <img src={userNine} alt="User" />
          </button>
          <button className="h-9 w-9 rounded-full border-2 border-white dark:border-boxdark">
            <img src={userTen} alt="User" />
          </button>
          <button className="flex h-9 w-9 items-center justify-center rounded-full border
border-stroke bg-white text-primary dark:border-strokedark dark:bg-[#4f5e77]
dark:text-white">
            <svg
              className="fill-current"
              width="16"
              height="16"
              viewBox="0 0 16 16"
              fill="none"
              xmlns="http://www.w3.org/2000/svg"
            >

            </svg>
          </button>
        </div>

        <div>
          <button
            ref={trigger}
            onClick={() => setPopupOpen(!popupOpen)}
            className="flex items-center gap-2 rounded bg-primary py-2 px-4.5 font-medium
text-white hover:bg-opacity-80"
          >
            <svg
              className="fill-current"
              width="16"
              height="16"
              viewBox="0 0 16 16"
              fill="none"
              xmlns="http://www.w3.org/2000/svg"
```

```
            >

            </svg>
            Add task
          </button>
        </div>
      </div>
    </div>
  );
};

export default TaskHeader;
```

## TaskLarge.tsx

The TaskLarge.tsx component defines a React component for adding new tasks. It provides input fields for specifying task details such as task name, description, status, priority, notifications, deadline, and category.

```
/**********************************************************************
 * Component: TaskLarge
 *
 * Logic: This file defines a React component for adding new tasks. It provides
 * input fields for specifying task details such as task name, description,
 * status, priority, notifications, deadline, and category.
 *
 * Used in Home
 *
 * Type: Component
**********************************************************************/

import useTaskContext from '../hooks/useTaskContext';
import dayjs from 'dayjs';
import { useContext, useState } from 'react';
import fireToast from '../hooks/fireToast';
```

```
const TaskLarge = () =>{

  const { createUserTask } = useTaskContext();

  const [formData, setFormData] = useState({
        status:'ToDo',
        task:'',
        description:'',
        deadline:dayjs(),
        priority:'5',
        category:'Other',
        notification:'None',
    };

  const handleInputChange = (e) => {
    const {name, value} = e.target;
    setFormData({
      ...formData,
      [name]:value,
      });
    };

    // Event handler to handle form submission
    // TODO add fire Toaster on Add
  const handleSubmit = (e) => {
    e.preventDefault();
    createUserTask(formData);
    setFormData({
        status:'ToDo',
        task:'',
        description:'',
        deadline:dayjs(),
        priority:'5',
        category:'Other',
        notification:'None',
    })
  };

  return (
    <>
      <div className="rounded-sm border border-stroke bg-white shadow-default
dark:border-strokedark dark:bg-boxdark">
```

```
        <form onSubmit={handleSubmit}>
          <div className="p-6.5">
              <div className="mb-3">
                <label className="mb-3 block text-black dark:text-white">
                  Add Task
                </label>
                <input value={formData.task} onChange={handleInputChange} id="task"
name="task" type="text" placeholder="Task" className="w-full rounded-lg border-[1.5px]
border-stroke bg-transparent py-3 px-5 font-medium outline-none transition
focus:border-primary active:border-primary disabled:cursor-default disabled:bg-whiter
dark:border-form-strokedark dark:bg-form-input dark:focus:border-primary" />
              </div>
              <div className="mb-3">
                <label className="mb-3 block text-black dark:text-white">
                  Description
                </label>
                <div>
                  <textarea  value={formData.description} onChange={handleInputChange}
id="description" name="description" rows={2} placeholder="Description" className="w-full
rounded-lg border-[1.5px] border-stroke bg-transparent py-3 px-5 font-medium outline-none
transition focus:border-primary active:border-primary disabled:cursor-default
disabled:bg-whiter dark:border-form-strokedark dark:bg-form-input dark:focus:border-primary"
>
                  </textarea>
                </div>
              </div>
              <label className="mb-3 block text-black dark:text-white">
                Status
              </label>
              <div className="relative z-20 bg-white dark:bg-form-input">
                <span className="absolute top-1/2 left-4 z-30 -translate-y-1/2">
                <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                  <g opacity="0.8">
                  </g>
                </svg>
                </span>
                <select  value={formData.status} id="status" name="status"
onChange={handleInputChange} className="relative z-20 w-full appearance-none rounded border
border-stroke bg-transparent py-3 px-12 outline-none transition focus:border-primary
active:border-primary dark:border-form-strokedark dark:bg-form-input">
                  <option value="ToDo">To Do</option>
                  <option value="In-Progress">In-Progress</option>
```

```
                  <option value="Complete">Complete</option>
                  <option value="Overdue">Overdue</option>
                </select>
                <span className="absolute top-1/2 right-4 z-10 -translate-y-1/2">
                  <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                    <g opacity="0.8">
                      </path>
                    </g>
                  </svg>
                </span>
              </div>
              <label className="mb-3 block text-black dark:text-white">
                Priority
              </label>
              <div className="relative z-20 bg-white dark:bg-form-input">
                <span className="absolute top-1/2 left-4 z-30 -translate-y-1/2">
                <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                  <g opacity="0.8">
                  </g>
                </svg>
                </span>
                <select  value={formData.priority} id="priority" name="priority"
onChange={handleInputChange} className="relative z-20 w-full appearance-none rounded border
border-stroke bg-transparent py-3 px-12 outline-none transition focus:border-primary
active:border-primary dark:border-form-strokedark dark:bg-form-input">
                  <option value="5"> Highest</option>
                  <option value="4">High</option>
                  <option value="3">Medium</option>
                  <option value="2">Low</option>
                  <option value="1">Lowest</option>
                </select>
                <span className="absolute top-1/2 right-4 z-10 -translate-y-1/2">
                  <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                    <g opacity="0.8">
                    </g>
                  </svg>
                </span>
              </div>
              <label className="mb-3 block text-black dark:text-white">
                Notifications
```

```
              </label>
              <div className="relative z-20 bg-white dark:bg-form-input">
                <span className="absolute top-1/2 left-4 z-30 -translate-y-1/2">
                <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                  <g opacity="0.8">
                  </g>
                </svg>
                </span>
                <select  value={formData.notification} id="notification" name="notification"
onChange={handleInputChange} className="relative z-20 w-full appearance-none rounded border
border-stroke bg-transparent py-3 px-12 outline-none transition focus:border-primary
active:border-primary dark:border-form-strokedark dark:bg-form-input">
                  <option value="None">None </option>
                  <option value="Desktop">Desktop </option>
                  <option value="Email">Email </option>
                  <option value="Both">Both </option>
                </select>
                <span className="absolute top-1/2 right-4 z-10 -translate-y-1/2">
                  <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                    <g opacity="0.8">
                    </g>
                  </svg>
                </span>
              </div>
              <div className="mb-3 block">
                <label className="mb-3 block text-black dark:text-white">
                  Deadline
                </label>
                <div className="relative">
                  <input value={formData.deadline} onChange={handleInputChange} id="deadline"
name="deadline" type="date" className="custom-input-date custom-input-date-1 w-full rounded
border-[1.5px] border-stroke bg-transparent py-3 px-5 font-medium outline-none transition
focus:border-primary active:border-primary dark:border-form-strokedark dark:bg-form-input
dark:focus:border-primary" />
                </div>
              </div>
              <label className="mb-3 block text-black dark:text-white">
                Category
              </label>
              <div className="relative z-20 bg-white dark:bg-form-input">
                <span className="absolute top-1/2 left-4 z-30 -translate-y-1/2">
```

```
                <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                  <g opacity="0.8">
                  </g>
                </svg>
                </span>
                <select value={formData.category} id="category" name="category"
onChange={handleInputChange} className="relative z-20 w-full appearance-none rounded border
border-stroke bg-transparent py-3 px-12 outline-none transition focus:border-primary
active:border-primary dark:border-form-strokedark dark:bg-form-input">
                  <option value="Work">Work </option>
                  <option value="School">School </option>
                  <option value="Chores">Chores </option>
                  <option value="Personal">Personal </option>
                  <option value="Other">Other </option>
                </select>
                <span className="absolute top-1/2 right-4 z-10 -translate-y-1/2">
                  <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                    <g opacity="0.8">
                    </g>
                  </svg>
                </span>
              </div>
              <button type="submit" className="flex w-full justify-center rounded bg-primary
p-3 font-medium text-gray">
                Add Task
              </button>
            </div>
          </form>
      </div>
    </>
  )
}

export default TaskLarge;
```

## TaskList.tsx

The TaskList.tsx component defines a React component for displaying a list of tasks. It is used in the "TaskManager Weekly" and provides options to delete or edit tasks.

```
/*********************************************************************
 * WORK-IN-PROGRESS:
 *   Make the TaskList Center on screen
 *
 * Component: TaskList
 *
 * Logic: This file defines a React component for displaying a list of tasks. It
 * uses the `useTaskContext` hook to access user tasks and provides options to
 * delete or edit tasks.
 *
 * Used in TaskManager Weekly
 *
 * Type: Component
*********************************************************************/

import React, { ComponentType } from 'react';
import Loader from '../common/Loader';
import { Task } from '../interfaces/Task';
import TaskModal from '../components/TaskModal';
import { useEffect, useState, useContext } from 'react';
import dayjs from 'dayjs';
import useTaskContext from '../hooks/useTaskContext';

const TaskListRows = (props) =>{

  const { userTasks, deleteUserTask, updateUserTask } = useTaskContext();
  const [ taskToEdit, setTaskToEdit ] = useState({});
  const [ isModalVisible, setIsModalVisible ] = useState(false);

  const  handleDelete  = (task_id) =>{
    deleteUserTask(task_id);
  };

  const handleEditTask = (taskObj) =>{
    setTaskToEdit(taskObj);
```

```
    setIsModalVisible(true);
    };
    const handleModalClose = () =>{
      setTaskToEdit({});
      setIsModalVisible(false);
      };

    useEffect(()=>{
      },[userTasks])

      {isModalVisible && <TaskModal taskData={taskToEdit} onClose={handleModalClose}/>};


  if (!userTasks || userTasks.length == 0) {
    return (
      <tr>
        <td className="border-b border-[#eee] py-5 px-4 pl-9 dark:border-strokedark
xl:pl-11">
          <h5 className="font-medium text-black dark:text-white">
            No Tasks Set
          </h5>
          <p className="text-sm">Add one!</p>
        </td>
        <td className="border-b border-[#eee] py-5 px-4 dark:border-strokedark">
          <p className="text-black dark:text-white">None</p>
        </td>
        <td className="border-b border-[#eee] py-5 px-4 dark:border-strokedark">
          <div className="flex items-center space-x-3.5">
            <button className="hover:text-primary">
             {/* TODO Change this SVG*/}
              <svg className="fill-current" width="18" height="18" viewBox="0 0 18 18"
fill="none" xmlns="http://www.w3.org/2000/svg" >
              </svg>
            </button>
          </div>
        </td>
      </tr>
    )}
  // For every task, create a pretty row
  return (
  <>
  {userTasks.filter((x : Task) => (x.deadline.toString().substring(8,10) == props.dateHolder)
&& (x.status == "ToDo" ||
```

```
   x.status == "In-Progress" || x.status == "Overdue")).map((task,index)=>{
      return (
        <>
        <tr key={task.task_id}>
          <td className="border-b border-[#eee] py-5 px-4 pl-9 dark:border-strokedark
xl:pl-11">
            <h5 className="font-medium text-black dark:text-white">
              {task.task}
            </h5>
            <p className="text-sm">{task.description}</p>
          </td>
          <td className="border-b border-[#eee] py-5 px-4 dark:border-strokedark">
            <p className="text-black dark:text-white">{
dayjs(task.deadline.toString().substring(0,10)).format('ddd MMM DD, YYYY').toString() }</p>
          </td>
          <td className="border-b border-[#eee] py-5 px-4 dark:border-strokedark">
            <div className="flex items-center space-x-3.5">

            <button className="hover:text-primary" name="task_id"
onClick={()=>{handleEditTask(task)}}>
                <svg width="18px" height="18px" viewBox="0 0 1024 1024" className="icon"
version="1.1" xmlns="http://www.w3.org/2000/svg">
               </svg>
              </button>

              <button className="hover:text-primary" name="task_id"
onClick={()=>{handleDelete(task.task_id)}}>
                <svg width="18px" height="18px" viewBox="0 0 1024 1024" className="icon"
version="1.1" xmlns="http://www.w3.org/2000/svg">
                </svg>
              </button>


            </div>
          </td>
        </tr>
        {isModalVisible && <TaskModal taskData={taskToEdit} onClose={handleModalClose}/>}
      </>
    );
    })
  }
  </>);
  };
```

```
const TaskList = (props) => {
    return (
        <div className="fixed z-10 inset-0 overflow-y-auto">
        <div className="flex items-end justify-center min-h-screen pt-4 px-4 pb-20
text-center sm:block sm:p-0">
          <div className="fixed inset-0 transition-opacity">
            <div className="absolute inset-0 bg-gray-500 opacity-75" onClick={() =>
props.setTaskView(false)}></div>
          </div>
          <span className="hidden sm:inline-block sm:align-middle sm:h-screen"></span>&#8203;
          <div
            className="inline-block align-bottom bg-white rounded-lg text-left
overflow-hidden shadow-xl transform transition-all sm:my-8 sm:align-middle sm:max-w-5xl
sm:w-full"
            role="dialog"
            aria-modal="true"
            aria-labelledby="modal-headline"
          >
            <div className="rounded-sm border border-stroke bg-white px-5 pt-6 pb-2.5
shadow-default dark:border-strokedark dark:bg-boxdark sm:px-7.5 xl:pb-1">
              <div className="max-w-full overflow-x-auto">
                <table className="w-full table-auto">
                    <thead>
                      <tr className="bg-gray-2 text-left dark:bg-meta-4">
                        <th className="min-w-[370px] py-4 px-4 font-medium text-black
dark:text-white xl:pl-11">
                          Task
                        </th>
                        <th className="min-w-[120px] py-4 px-4 font-medium text-black
dark:text-white">
                          Due Date
                        </th>
                        <th className="py-4 px-4 font-medium text-black dark:text-white">
                          Actions
                        </th>
                      </tr>
                    </thead>
                    <tbody>
                      <TaskListRows dateHolder={props.holdDate}/>
                  </tbody>
                </table>
```

```
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};

export default TaskList;
```

## TaskModal.tsx

The TaskModal.tsx component defines a React component for displaying a modal dialog for editing tasks. Users can input or update task details such as task name, description, status, priority, notifications, deadline, and category within this modal.

```
/**********************************************************************
 * WORK-IN-PROGRESS:
 *    Change the Dark mode toggle to function with Modal
 *
 * Component: TaskModal
 *
 * Logic: This file defines a React component for displaying a modal dialog for
 * editing tasks. Users can input task details like task name,
 * description, status, priority, notifications, deadline, and category.
 *
 * Type: Component
**********************************************************************/


import useTaskContext from '../hooks/useTaskContext';
import dayjs from 'dayjs';
import { useContext, useState, useEffect } from 'react';
```

```
const TaskModal = ( props ) =>{
  const { taskData, onClose } = props;
  const { updateUserTask } = useTaskContext();
  const [formData, setFormData] = useState(taskData);

  const handleInputChange = (e) => {
    const {name, value} = e.target;
    setFormData({
      ...formData,
      [name]:value,
      });
    };
    // Event handler to handle form submission
  const handleExit = (e) => {
    e.preventDefault();
    setFormData({
        status:'ToDo',
        task:'',
        description:'',
        deadline:dayjs(),
        priority:'5',
        category:'Other',
        notification:'None',
    });
    onClose();
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    updateUserTask(formData);
    setFormData({
        status:'ToDo',
        task:'',
        description:'',
        deadline:dayjs(),
        priority:'5',
        category:'Other',
        notification:'None',
    });
    onClose();
  };
```

```
  return (
    <>
    <div className="fixed inset-0 bg-black opacity-40"></div>
      <div className="relative z-50 w-full max-w-md p-6 mx-auto bg-white rounded-lg
shadow-lg">
        <button onClick={handleExit} className="absolute top-0 right-0 p-4 hover:text-primary
text-gray-500">
          <svg className="fill-current text-red-500" width="20" height="20"
xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 20 20" >
            <g opacity="0.8">
              <path d="M6 18L18 6M6 6l12 12" />
            </g>
          </svg>
        </button>
        <form onSubmit={handleSubmit}>
          <div className="p-6.5">
              <div className="mb-3">
                <label className="mb-3 block text-black dark:text-white">
                  Add Task
                </label>
                <input value={formData.task} onChange={handleInputChange} id="task"
name="task" type="text" placeholder="Task" className="w-full rounded-lg border-[1.5px]
border-stroke bg-transparent py-3 px-5 font-medium outline-none transition
focus:border-primary active:border-primary disabled:cursor-default disabled:bg-whiter
dark:border-form-strokedark dark:bg-form-input dark:focus:border-primary" />
              </div>
              <div className="mb-3">
                <label className="mb-3 block text-black dark:text-white">
                  Description
                </label>
                <div>
                  <textarea  value={formData.description} onChange={handleInputChange}
id="description" name="description" rows={2} placeholder="Description" className="w-full
rounded-lg border-[1.5px] border-stroke bg-transparent py-3 px-5 font-medium outline-none
transition focus:border-primary active:border-primary disabled:cursor-default
disabled:bg-whiter dark:border-form-strokedark dark:bg-form-input dark:focus:border-primary"
>
                  </textarea>
                </div>
              </div>
              <label className="mb-3 block text-black dark:text-white">
                Status
```

```
              </label>
              <div className="relative z-20 bg-white dark:bg-form-input">
                <span className="absolute top-1/2 left-4 z-30 -translate-y-1/2">
                <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                  <g opacity="0.8">
                  </g>
                </svg>
                </span>
                <select  value={formData.status} id="status" name="status"
onChange={handleInputChange} className="relative z-20 w-full appearance-none rounded border
border-stroke bg-transparent py-3 px-12 outline-none transition focus:border-primary
active:border-primary dark:border-form-strokedark dark:bg-form-input">
                  <option value="ToDo">To Do</option>
                  <option value="In-Progress">In-Progress</option>
                  <option value="Complete">Complete</option>
                  <option value="Overdue">Overdue</option>
                </select>
                <span className="absolute top-1/2 right-4 z-10 -translate-y-1/2">
                  <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                    <g opacity="0.8">
                      </path>
                    </g>
                  </svg>
                </span>
              </div>
              <label className="mb-3 block text-black dark:text-white">
                Priority
              </label>
              <div className="relative z-20 bg-white dark:bg-form-input">
                <span className="absolute top-1/2 left-4 z-30 -translate-y-1/2">
                <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                  <g opacity="0.8">

                  </g>
                </svg>
                </span>
                <select id="priority" name="priority" onChange={handleInputChange}
className="relative z-20 w-full appearance-none rounded border border-stroke bg-transparent
py-3 px-12 outline-none transition focus:border-primary active:border-primary
dark:border-form-strokedark dark:bg-form-input" defaultValue={formData.priority}>
```

```
                  <option value="5"> Highest</option>
                  <option value="4">High</option>
                  <option value="3">Medium</option>
                  <option value="2">Low</option>
                  <option value="1">Lowest</option>
                </select>
                <span className="absolute top-1/2 right-4 z-10 -translate-y-1/2">
                  <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                    <g opacity="0.8">

                    </g>
                  </svg>
                </span>
              </div>
              <label className="mb-3 block text-black dark:text-white">
                Notifications
              </label>
              <div className="relative z-20 bg-white dark:bg-form-input">
                <span className="absolute top-1/2 left-4 z-30 -translate-y-1/2">
                <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                  <g opacity="0.8">

                  </g>
                </svg>
                </span>
                <select  value={formData.notification} id="notification" name="notification"
onChange={handleInputChange} className="relative z-20 w-full appearance-none rounded border
border-stroke bg-transparent py-3 px-12 outline-none transition focus:border-primary
active:border-primary dark:border-form-strokedark dark:bg-form-input">
                  <option value="None">None </option>
                  <option value="Desktop">Desktop </option>
                  <option value="Email">Email </option>
                  <option value="Both">Both </option>
                </select>
                <span className="absolute top-1/2 right-4 z-10 -translate-y-1/2">
                  <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                    <g opacity="0.8">
                      </path>
                    </g>
                  </svg>
```

```
                </span>
              </div>
              <div className="mb-3 block">
                <label className="mb-3 block text-black dark:text-white">
                  Deadline
                </label>
                <div className="relative">
                  <input onChange={handleInputChange} id="deadline" name="deadline"
type="date" className="custom-input-date custom-input-date-1 w-full rounded border-[1.5px]
border-stroke bg-transparent py-3 px-5 font-medium outline-none transition
focus:border-primary active:border-primary dark:border-form-strokedark dark:bg-form-input
dark:focus:border-primary" value={formData.deadline.toString().substring(0,10)}/>
                </div>
              </div>
              <label className="mb-3 block text-black dark:text-white">
                Category
              </label>
              <div className="relative z-20 bg-white dark:bg-form-input">
                <span className="absolute top-1/2 left-4 z-30 -translate-y-1/2">
                <svg width="20" height="20" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                  <g opacity="0.8">
                  </g>
                </svg>
                </span>
                <select value={formData.category} id="category" name="category"
onChange={handleInputChange} className="relative z-20 w-full appearance-none rounded border
border-stroke bg-transparent py-3 px-12 outline-none transition focus:border-primary
active:border-primary dark:border-form-strokedark dark:bg-form-input">
                  <option value="Work">Work </option>
                  <option value="School">School </option>
                  <option value="Chores">Chores </option>
                  <option value="Personal">Personal </option>
                  <option value="Other">Other </option>
                </select>
                <span className="absolute top-1/2 right-4 z-10 -translate-y-1/2">
                  <svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg" >
                    <g opacity="0.8">
                    </g>
                  </svg>
                </span>
              </div>
```

```
              <button type="submit" className="flex w-full justify-center rounded bg-primary
p-3 font-medium text-gray">
                Update Task
              </button>
            </div>
        </form>
      </div>
    </>
  )
}

export default TaskModal;
```

## TaskSmall.tsx

The TaskSmall component defines a React component for displaying a small task or event item. This component is designed to appear as a clickable item with optional additional details that become visible when hovering over the item.

```
/**********************************************************************
 * WORK-IN-PROGRESS:
 *    Make the small tasks fill with their designated monthly calendar box
 *    with their information
 *
 * Component: TaskSmall
 *
 * Logic: This file defines a React component for displaying a small task or
 * event item. It appears as a clickable item with optional additional details
 * that become visible on hover.
 *
 * Type: Component
**********************************************************************/
```

```
import { useEffect, useRef, useState } from 'react';


const TaskSmall = () => {

  return (
  <div className="group h-16 w-full flex-grow cursor-pointer py-1 md:h-30">
    <span className="group-hover:text-primary md:hidden">
      More
    </span>
    <div className="event invisible absolute left-2 z-99 mb-1 flex w-[200%] flex-col
rounded-sm border-l-[3px] border-primary bg-gray px-3 py-1 text-left opacity-0
group-hover:visible group-hover:opacity-100 dark:bg-meta-4 md:visible md:w-[190%]
md:opacity-100">
      <span className="event-name text-sm font-semibold text-black dark:text-white">
        Redesign Website
      </span>
      <span className="time text-sm font-medium text-black dark:text-white">
        1 Dec - 2 Dec
      </span>
    </div>
  </div>
  );
};

export default TaskSmall;
```

## CONTEXT

### AuthContext.tsx & AuthProvider.tsx

The AuthContext.tsx file defines an authentication context using React's createContext API. This context is used to manage authentication state and functionality. It doesn't contain any specific logic but serves as the foundation for authentication-related functionality.

The AuthProvider.tsx file is a React component that provides authentication-related functionality and state to other components.

```
/**********************************************************************
 * WORK-IN-PROGRESS:
 *    Add User information in settings, and store in state
 *
 * File: AuthProvider.tsx
 *
 * Logic: Provides authentication-related functionality and state to other components.
 *
 * Type: React Component
**********************************************************************/

import { useState, useEffect, useContext } from 'react'
import { useNavigate } from 'react-router-dom';
import jwt_decode from "jwt-decode";
import useAxios from "../hooks/useAxios";
import axiosInstance from "../axios";

import Loader from "../common/Loader/index";
import AuthContext from './AuthContext';

const AuthProvider = ({ children }) => {
    let [authTokens, setAuthTokens] = useState(()=>localStorage.getItem('authTokens') ?
JSON.parse(localStorage.getItem('authTokens')) : null);

    let [loading, setLoading] = useState(true);

    const navigate = useNavigate();

    const getUserProfile = async (e) => {
      const authenticated_axios = useAxios();
      const res = await authenticated_axios.get('user/gather/');
      const userProfileObj = res.data;
      console.log(userProfileObj);
      }


/* ********************************************************
 *  AUTH A USER
```

```
 * ************************************************************/
    const loginUser = async (e) => {
      try{
        e.preventDefault();
        const res = await axiosInstance.post(`token/`,
        {
          email:e.target.email.value,
          password:e.target.password.value
          }
        );
        if (res.status === 200){
          let data = await res.data;
          setAuthTokens(()=>data);
          localStorage.setItem('authTokens', JSON.stringify(data));
          getUserProfile();
          navigate('/');
        } else {
          console.error("It shouldn't get here");
        }
      } catch (e){
        console.error("Error with login: ", e)
      }
    };

/* ************************************************************
 *  CREATE A USER
 * ************************************************************/
    const signupUser = async (formData) => {
      try{
        const res = await axiosInstance.post(`user/create/`, {
          email:formData.email,
          user_name:formData.username,
          password:formData.password
          }
        );
        if (res.status === 200){
          navigate('/auth/signin/');
        } else {
          console.error("It shouldn't get here");
        }
      } catch (e){
        console.error("Error with login: ", e)
      }
```

```
    };

/* ************************************************************
 *  LOGOUT THE USER
 * ************************************************************/
    let logoutUser = () => {
        setAuthTokens(null);
        localStorage.removeItem('authTokens');
        navigate('/pages/landingpage');
    }

    let contextData = {
        // userProfile:userProfile, // TODO add user profile
        authTokens:authTokens,
        setAuthTokens:setAuthTokens,
        loginUser:loginUser,
        logoutUser:logoutUser,
        signupUser:signupUser,
    }

    useEffect(() => {
        // getUserProfile(); // TODO USER PROFILE
        setLoading(false);
    }, [authTokens, loading]);


    return (
      <AuthContext.Provider value={contextData} >
        { loading ? <Loader/> : children }
      </AuthContext.Provider>
    );
};

export default AuthProvider;
```

## TaskContext.tsx & TaskProvider.tsx

The TaskContext.tsx file defines a React context for managing task-related data and functions. It's a basic context definition without specific logic, serving as a foundation for managing task-related data and functions within the application.

The TaskProvider.tsx file is a React context provider responsible for managing task-related data and functions.

```
/*********************************************************************
 * File: TaskContext.tsx
 *
 * Logic: Defines a React context for managing task-related data and functions.
 *
 * Type: React Context
*********************************************************************/


import { createContext } from 'react'

export const TaskContext = createContext(null);

export default TaskContext;
```

```
/*********************************************************************
 * WORK-IN-PROGRESS:
 *    Change way tasks are updated
 *
 * File: TaskProvider.tsx
 *
 * Logic: Defines a React context provider for managing task-related data and functions.
 *
```

```
 * Type: React Context Provider
**********************************************************************/


import { useState, useEffect, useContext, useRef } from 'react'
import useAxios from "../hooks/useAxios";

import TaskContext from './TaskContext';

const TaskProvider = ({ children }) => {

    let [userTasks, setUserTasks] = useState({});

    let [userTasksStats, setUserTasksStats] = useState({
        level:0,
        todo:0,
        inProgress:0,
        completed:0,
        overdue:0,
      });

    const axiosInstance = useAxios();

    /* ****************************************
     *  GET USER TASKS (QUERY)
     * ****************************************/
    const getUserTasks = async () => {
      try{
        const response = await axiosInstance.get();
        setUserTasks(response.data);
        setUserTasksStats({
            level:0,
            todo:0,
            inProgress:0,
            completed:0,
            overdue:0,
          });
      } catch (e) {
        console.error("Error Fetching: ",e);
      }
    }

    /* ****************************************
```

```
    *  GET TASK STATS (LOCAL)
    * ****************************************/
const getTasksStats = () => {
  if (Object.keys(userTasks).length > 0){
    const updatedStats = {
        level:0,
        todo:0,
        inProgress:0,
        completed:0,
        overdue:0,
      }
    userTasks.forEach((task)=>{
        if (task.status == "ToDo") {
          updatedStats.todo++;
        } else if (task.status == "In-Progress"){
          updatedStats.inProgress++
        } else if (task.status == "Complete") {
          updatedStats.completed++;
        } else if (task.status == "Overdue") {
          updatedStats.overdue++;
        } else {
        console.error("UserTasksStats Error")
        }
        setUserTasksStats(updatedStats);
        }
      )
    }
  }

  /* ****************************************
   *  CREATE TASK (QUERY)
   * ****************************************/
  const createUserTask = async (formData) => {
    try{
      const res = await axiosInstance.post("create/",{
        status      : formData.status,
        task        : formData.task,
        description : formData.description,
        priority    : formData.priority,
        notification: formData.notification,
        deadline    : formData.deadline,
        category    : formData.category,
        })
```

```
        getUserTasks();
        // setUserTasks([...userTasks, res.data]);
        // setUpdatedUserTasks(!updateUserTask);
      } catch (e) {
        console.error("Creating Error: ",e);
      }
    }

    /* ****************************************
     *  UPDATE TASK (QUERY)
     * ****************************************/
    const updateUserTask = async (formData) => {
      try {
        const res = await axiosInstance.patch(`update/${formData.task_id}`,{
            status      : formData.status,
            task        : formData.task,
            description : formData.description,
            priority    : formData.priority,
            notification: formData.notification,
            deadline    : formData.deadline,
            category    : formData.category,
          });
        getUserTasks();
        // const updated_task_id = res.data.task_id;
        // const updatedTasks = (prevTasks) => {prevTasks.map((task)=>task.task_id ==
updated_task_id ? res.data : task)}
        // setUserTasks(updatedTasks);
        // setUpdatedUserTasks(!updateUserTask);
      } catch (e) {
        console.error("Error Updating: ", e);
      }
    }

    /* ****************************************
     *  DELETE TASK (QUERY)
     * ****************************************/
    const deleteUserTask = async (task_id) => {
      try {
        const res = await axiosInstance.delete(`delete/${task_id}`);
        // const updatedTasks = userTasks.filter((x)=>x.task_id != task_id);

        getUserTasks();
        // const updated_task_id = res.data.task_id;
```

```
        // const updatedTasks = (prevTasks) => {prevTasks.filter((task)=>task.task_id ==
updated_task_id)}
        // setUserTasks(updatedTasks);
        // setUpdatedUserTasks(!updateUserTask);
      } catch (e) {
        console.error("Error Deleting: ", e);
      }
    }

    let contextData = {
        userTasks:userTasks,
        setUserTasks:setUserTasks,
        // updatedUserTasks:updatedUserTasks,

        getUserTasks:getUserTasks,
        createUserTask:createUserTask,
        deleteUserTask:deleteUserTask,
        updateUserTask:updateUserTask,

        // updatedStats:updatedStats,
        userTasksStats:userTasksStats,
    }

    useEffect(() => {
      getUserTasks();
      // setUpdatedStats(!updatedStats);
    },[]); // Runs once on mount

    useEffect(() => {
      getTasksStats();
    }, [userTasks]); // Runs on mount and UpdatedUserTasks

    return (
      <TaskContext.Provider value={contextData} >
        { children }
      </TaskContext.Provider>
    );
};

export default TaskProvider;
```

## HOOKS

### fireToast.tsx

The fireToast.tsx hook defines functionality for creating and displaying toasts (notifications) based on specific data conditions.

```
/**********************************************************************
 * WORK-IN-PROGRESS:
 *    Attach the toast to fire on updated state
 *
 * File: fireToast.tsx
 *
 * Logic: Defines a hook for creating and displaying toasts (notifications) based on data
conditions.
 *
 * Type: React Hook
**********************************************************************/



import toast from 'react-hot-toast';
import dataJSON from '../../public/data.json';


const createToast=(title: string, msg: string, type: number)=>{toast.custom((t) => (

    <div
      className={`${
        t.visible ? 'animate-enter' : 'animate-leave'
      }
      max-w-md w-full ${type=='0'?"bg-[#04b20c]":type=='1'?"bg-[#eab90f]":"bg-[#e13f32]"}
shadow-lg rounded-lg pointer-events-auto flex ring-1 ring-black ring-opacity-5`}
    >
      <div className="flex-1 w-0 p-4 ">
        <div className="flex items-start">
```

```
          <div className="ml-3 flex-1">
            <p className="text-sm font-medium text-white">
              {title}
            </p>
            <p className="mt-1 text-sm text-white">
              {msg}
            </p>
          </div>
        </div>
      </div>
      <div className="flex">

          <button
          onClick={() => toast.dismiss(t.id)}
          type="button"
          className="mr-2 box-content rounded-none border-none opacity-100 hover:no-underline
hover:opacity-50 focus:opacity-50 focus:shadow-none focus:outline-none text-white"
          data-te-toast-dismiss
          aria-label="Close">
          <span
            className="w-[1em] focus:opacity-100 disabled:pointer-events-none
disabled:select-none disabled:opacity-25 [&.disabled]:pointer-events-none
[&.disabled]:select-none [&.disabled]:opacity-25">
            <svg
              xmlns="http://www.w3.org/2000/svg"
              fill="none"
              viewBox="0 0 24 24"
              stroke-width="1.5"
              stroke="currentColor"
              className="h-6 w-6">
              <path
                stroke-linecap="round"
                stroke-linejoin="round"
                d="M6 18L18 6M6 6l12 12" />
            </svg>
          </span>
        </button>

      </div>
    </div>
  ))};
```

```
const fireToast = () => {
  const alertSettings=localStorage.getItem("alertSettings");
  if (alertSettings){
    for (const alertSetting of JSON.parse(alertSettings)) {
      console.log(alertSetting);

      const
value=isNaN(parseFloat(alertSetting.value))?alertSetting.value:parseFloat(alertSetting.value)
;
      const para=alertSetting.criterion<2?"delta_"+alertSetting.para:alertSetting.para;
      if (alertSetting.id=="ALL"){
        Object.keys(dataJSON).map((id:string)=>
        {
          const condition=alertSetting.criterion=='0'?value<=-1*dataJSON[id][para]:
          alertSetting.criterion=='1'||alertSetting.criterion=='3'?value>=dataJSON[id][para]:
          alertSetting.criterion=='2'?value<=dataJSON[id][para]:
          value==dataJSON[id][para];
          const
realValue=alertSetting.criterion=='0'?dataJSON[id][para]*-1:dataJSON[id][para];
          if (condition){
            const msg=`${alertSetting.para} of ${id} ${alertSetting.criterion==0?"goes down
by":alertSetting.criterion==1?"goes up by":alertSetting.criterion==2?"is smaller
than":alertSetting.criterion==3?"is greater than":"is equal to"} ${realValue}`;
            createToast(id,msg,alertSetting.type)
          }


        }

        );
      }
      else{
        const id=alertSetting.id;

        const condition =
          alertSetting.criterion=='0' ?  value >= -1 * dataJSON[id][para]:
          alertSetting.criterion=='1' ||
          alertSetting.criterion=='3' ?  value >= dataJSON[id][para]:
          alertSetting.criterion=='2' ?  value <= dataJSON[id][para]:
value==dataJSON[id][para];

          const
realValue=alertSetting.criterion=='0'?dataJSON[id][para]*-1:dataJSON[id][para];
```

```
          if (condition){
            const msg=`${alertSetting.para} of ${id} ${alertSetting.criterion==0?"goes down
by":alertSetting.criterion==1?"goes up by":alertSetting.criterion==2?"is smaller
than":alertSetting.criterion==3?"is greater than":"is equal to"} ${realValue}`;
            createToast(id,msg,alertSetting.type)
          }
        }
    };
  }
}

export default fireToast;
```

## useAxios.ts

The useAxios.tsx hook defines functionality for creating an Axios instance with authentication headers and handling token refresh. This hook is designed to simplify the process of making authenticated API requests using Axios, automatically refreshing the access token when necessary. It helps maintain a seamless user experience when working with authentication in the application.

```
/* ************************************************************
* File: useAxios.tsx
*
* Logic: Defines a hook for creating an Axios instance with authentication
* headers and handling token refresh.
*
* Type: React Hook
************************************************************/


import { useContext } from 'react'
import AuthContext from '../context/AuthContext'
import axios from 'axios'
```

```
import jwt_decode from "jwt-decode";
import dayjs from 'dayjs'


const baseURL = 'http://127.0.0.1:8000/api/'

const useAxios = () => {
    const {authTokens, setAuthTokens} = useContext(AuthContext)

    const axiosInstance: AxiosInstance = axios.create({
      baseURL,
      timeout: 5000,
      headers: {
        'Authorization': `Bearer ${authTokens.access}`,
        'Content-Type': 'application/json',
        accept: 'application/json',
      },
    });

    axiosInstance.interceptors.response.use(async res => {
        return res;
    });

    axiosInstance.interceptors.request.use(async req => {
        const user = jwt_decode(authTokens.access)
        const isExpired = dayjs.unix(user.exp).diff(dayjs()) < 1;

        if(!isExpired) return req

        const response = await axios.post(`${baseURL}token/refresh/`, {
            refresh: authTokens.refresh
          });

        localStorage.setItem('authTokens', JSON.stringify(response.data))
        setAuthTokens(response.data)

        req.headers.Authorization = `Bearer ${response.data.access}`
        return req
    })

    return axiosInstance;
}
```

```
export default useAxios;
```

## useColorMode.tsx

The useColorMode.tsx hook is responsible for managing and applying color themes (light and dark) to the application.

```
/*************************************************************************
 * File: useColorMode.tsx
 *
 * Logic: Manages and applies color themes (light and dark) to the application.
 *
 * Type: React Hook
*************************************************************************/


import { useEffect } from 'react';
import useLocalStorage from './useLocalStorage';

const useColorMode = () => {
  const [colorMode, setColorMode] = useLocalStorage('color-theme', 'light');

  useEffect(() => {
    const className = 'dark';
    const bodyClass = window.document.body.classList;

    colorMode === 'dark'
      ? bodyClass.add(className)
      : bodyClass.remove(className);
  }, [colorMode]);

  return [colorMode, setColorMode];
};

export default useColorMode;
```

## useLocalStorage.tsx

The useLocalStorage.tsx hook is responsible for managing and persisting data in the browser's local storage. This hook provides a convenient way to manage and persist data in the browser's local storage. It abstracts away the complexities of dealing with local storage and allows components to interact with stored data in a straightforward manner.

```
/**********************************************************************
 * File: useLocalStorage.tsx
 *
 * Logic: Manages and persists data in the browser's local storage.
 *
 * Type: React Hook
**********************************************************************/


import { useEffect, useState } from 'react';

type SetValue<T> = T | ((val: T) => T);

function useLocalStorage<T>(
  key: string,
  initialValue: T
): [T, (value: SetValue<T>) => void] {
  // State to store our value
  // Pass  initial state function to useState so logic is only executed once
  const [storedValue, setStoredValue] = useState(() => {
    try {
      // Get from local storage by key
      const item = window.localStorage.getItem(key);
      // Parse stored json or if none return initialValue
      return item ? JSON.parse(item) : initialValue;
    } catch (error) {
      // If error also return initialValue
      console.log(error);
      return initialValue;
```

```
    }
  });

  // useEffect to update local storage when the state changes
  useEffect(() => {
    try {
      // Allow value to be a function so we have same API as useState
      const valueToStore =
        typeof storedValue === 'function'
          ? storedValue(storedValue)
          : storedValue;
      // Save state
      window.localStorage.setItem(key, JSON.stringify(valueToStore));
    } catch (error) {
      // A more advanced implementation would handle the error case
      console.log(error);
    }
  }, [key, storedValue]);

  return [storedValue, setStoredValue];
}

export default useLocalStorage;
```

## useTaskContext.tsx

The useTaskContext.tsx hook is a custom React hook that provides access to the TaskContext using the useContext hook. This custom hook simplifies the process of accessing the values and functions provided by the TaskContext within components. Components that import and use this hook can easily access and interact with task-related data and functionality from the context.

```
/**********************************************************************
 * File: useTaskContext.tsx
 *
 * Logic: Provides access to the TaskContext using useContext.
```

```
 *
 * Type: Custom React Hook
*********************************************************************/

import TaskContext from '../context/TaskContext';
import { useContext } from 'react';

export default function useTaskContext() {
  return useContext(TaskContext);
}
```

## PAGES

### Authentication/SignIn.tsx

The SignIn.tsx component plays a crucial role in the overall functionality of the application. It serves as the entry point for users to access their accounts and is seamlessly integrated with the rest of the application. This component is a pivotal part of the application, facilitating user access while maintaining design consistency and security. Its seamless integration with the AuthContext and user registration features ensures that users can smoothly transition from signing in to utilizing the full functionality of the application.

```
/*********************************************************************
 * File: SignIn.tsx
 *
 * Logic: Defines a sign-in page component for the application. It includes a
 * form for users to enter their email and password to log in.
 *
 * Type: React Component
*********************************************************************/
```

```
import { useContext } from 'react'
import AuthContext from '../../context/AuthContext';
import { Link } from 'react-router-dom';
import LogoDark from '../../images/logo/logo-dark.svg';
import Logo from '../../images/logo/CrabbyTasksLogo.png';

const SignIn = () => {
  let { loginUser } = useContext(AuthContext);
  return (
    <>
      <div className="rounded-sm border border-stroke bg-white shadow-default
dark:border-strokedark dark:bg-boxdark">
        <div className="flex flex-wrap items-center">
          <div className="hidden w-full xl:block xl:w-1/2">
            <div className="py-17.5 px-26 text-center">
              <div className = "flex justify-center">
              <Link className="mb-5.5 inline-block" to="/">
                <img src={Logo} alt="Logo" className="dark:hidden max-w-[250px]"/>
              </Link>
              </div>

              <p className="2xl:px-20 font italic">
                While the crab's sideways movement may appear aimless,
                it symbolizes life's unpredictability, reminding us that
                the path to truth is often found in unexpected directions.
              </p>

              <span className="mt-15 inline-block">
                <svg
                  width="350"
                  height="350"
                  viewBox="0 0 350 350"
                  fill="none"
                  xmlns="http://www.w3.org/2000/svg"
                >
                </svg>
              </span>
            </div>
          </div>

          <div className="w-full border-stroke dark:border-strokedark xl:w-1/2
xl:border-l-2">
            <div className="w-full p-4 sm:p-12.5 xl:p-17.5">
```

```
              <span className="mb-1.5 block font-medium">Start for free</span>
              <h2 className="mb-9 text-2xl font-bold text-black dark:text-white
sm:text-title-xl2">
                Sign In to CrabbyTasks!
              </h2>

              <form onSubmit={loginUser}>
                <div className="mb-4">
                  <label className="mb-2.5 block font-medium text-black dark:text-white">
                    Email
                  </label>
                  <div className="relative">
                    <input
                      id="email"
                      name="email"
                      type="email"
                      placeholder="Enter your email"
                      className="w-full rounded-lg border border-stroke bg-transparent py-4
pl-6 pr-10 outline-none focus:border-primary focus-visible:shadow-none
dark:border-form-strokedark dark:bg-form-input dark:focus:border-primary"
                    />

                    <span className="absolute right-4 top-4">
                      <svg
                        className="fill-current"
                        width="22"
                        height="22"
                        viewBox="0 0 22 22"
                        fill="none"
                        xmlns="http://www.w3.org/2000/svg"
                      >
                        <g opacity="0.5">
                                                  </g>
                      </svg>
                    </span>
                  </div>
                </div>

                <div className="mb-6">
                  <label className="mb-2.5 block font-medium text-black dark:text-white">
                    Password
                  </label>
                  <div className="relative">
```

```
                    <input
                      id="password"
                      name="password"
                      type="password"
                      placeholder="6+ Characters, 1 Capital letter"
                      className="w-full rounded-lg border border-stroke bg-transparent py-4
pl-6 pr-10 outline-none focus:border-primary focus-visible:shadow-none
dark:border-form-strokedark dark:bg-form-input dark:focus:border-primary"
                    />

                    <span className="absolute right-4 top-4">
                      <svg
                        className="fill-current"
                        width="22"
                        height="22"
                        viewBox="0 0 22 22"
                        fill="none"
                        xmlns="http://www.w3.org/2000/svg"
                      >
                        <g opacity="0.5">
                        </g>
                      </svg>
                    </span>
                  </div>
                </div>

                <div className="mb-5">
                  <input
                    type="submit"
                    value="Sign In"
                    className="w-full cursor-pointer rounded-lg border border-primary
bg-primary p-4 text-white transition hover:bg-opacity-90"
                  />
                </div>



                <div className="mt-6 text-center">
                  <p>
                    Don't have any account?{' '}
                    <Link to="/auth/signup" className="text-primary">
                      Sign Up
                    </Link>
```

```
                  </p>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </>
  );
};

export default SignIn;
```

## Authentication/SignUp.tsx

The SignUp.tsx component is a crucial part of the application as it enables users to create accounts and gain access to the full functionality of CrabbyTasks. This is an essential component that contributes to the overall user journey within the CrabbyTasks application. It seamlessly integrates with user authentication, maintains design consistency, and enhances security, offering users a complete and secure sign-up experience.

```
/**********************************************************************
 * WORK-IN-PROGRESS:
 *    Finish the Auth Sign up logic and attach
 *
 * File: SignUp.tsx
 *
 * Logic: Defines a sign-up page component for the application. It includes a
 * form for users to sign up by providing their name, email, and password.
 *
 * Type: React Component
**********************************************************************/

import {useState, useContext} from 'react';
```

```
import  axiosInstance  from '../../axios';
import AuthContext from '../../context/AuthContext';
import { Link, useNavigate } from 'react-router-dom';
import Logo from '../../images/logo/CrabbyTasksLogo.png';

const SignUp = () => {
  let { signupUser } = useContext(AuthContext);
  const navigate = useNavigate();
  const initialFormData = Object.freeze({
    email:'',
    username:'',
    password:'',
  });

  const [formData, updateFormData] = useState(initialFormData);
  const handleChange = (e) => {
    updateFormData({
      ...formData,
      [e.target.name]: e.target.value.trim(),
      });
    }

  const handleSubmit = (e) => {
    e.preventDefault();
    signupUser(formData);
    }
  return (
    <>
      <div className="rounded-sm border border-stroke bg-white shadow-default
dark:border-strokedark dark:bg-boxdark">
        <div className="flex flex-wrap items-center">
          <div className="hidden w-full xl:block xl:w-1/2">
            <div className="py-17.5 px-26 text-center">
            <div className = "flex justify-center">
            <Link className="mb-5.5 inline-block" to="/">
                <img src={Logo} alt="Logo" className="dark:hidden max-w-[250px]"/>
              </Link>
              </div>

              <p className="2xl:px-20">
                Much like the crab, the human journey involves shedding
                old shells and embracing change, for growth requires the
                courage to leave behind the safety of the familiar.
```

```
              </p>

              <span className="mt-15 inline-block">
                <svg width="350" height="350" viewBox="0 0 350 350" fill="none"
xmlns="http://www.w3.org/2000/svg" >
              </span>
            </div>
          </div>

          <div className="w-full border-stroke dark:border-strokedark xl:w-1/2
xl:border-l-2">
            <div className="w-full p-4 sm:p-12.5 xl:p-17.5">
              <span className="mb-1.5 block font-medium">Start for free</span>
              <h2 className="mb-9 text-2xl font-bold text-black dark:text-white
sm:text-title-xl2">
                Sign Up to CrabbyTasks
              </h2>

              <form onSubmit={handleSubmit}>
                <div className="mb-4">
                  <label className="mb-2.5 block font-medium text-black dark:text-white">
                    Name
                  </label>
                  <div className="relative">
                    <input
                      id="username"
                      name="username"
                      type="text"
                      placeholder="Enter your full name"
                      className="w-full rounded-lg border border-stroke bg-transparent py-4
pl-6 pr-10 outline-none focus:border-primary focus-visible:shadow-none
dark:border-form-strokedark dark:bg-form-input dark:focus:border-primary"
                      onChange={handleChange}
                    />

                    <span className="absolute right-4 top-4">
                      <svg
                        className="fill-current"
                        width="22"
                        height="22"
                        viewBox="0 0 22 22"
                        fill="none"
                        xmlns="http://www.w3.org/2000/svg"
```

```
                    >
                      <g opacity="0.5">
                      </g>
                    </svg>
                  </span>
                </div>
              </div>

              <div className="mb-4">
                <label className="mb-2.5 block font-medium text-black dark:text-white">
                  Email
                </label>
                <div className="relative">
                  <input
                    id="email"
                    name="email"
                    type="email"
                    placeholder="Enter your email"
                    className="w-full rounded-lg border border-stroke bg-transparent py-4
pl-6 pr-10 outline-none focus:border-primary focus-visible:shadow-none
dark:border-form-strokedark dark:bg-form-input dark:focus:border-primary"
                    onChange={handleChange}
                  />

                  <span className="absolute right-4 top-4">
                    <svg
                      className="fill-current"
                      width="22"
                      height="22"
                      viewBox="0 0 22 22"
                      fill="none"
                      xmlns="http://www.w3.org/2000/svg"
                    >
                      <g opacity="0.5">
                      </g>
                    </svg>
                  </span>
                </div>
              </div>

              <div className="mb-4">
                <label className="mb-2.5 block font-medium text-black dark:text-white">
                  Password
```

```
                  </label>
                  <div className="relative">
                    <input
                      id="password"
                      name="password"
                      type="password"
                      placeholder="Enter your password"
                      className="w-full rounded-lg border border-stroke bg-transparent py-4
pl-6 pr-10 outline-none focus:border-primary focus-visible:shadow-none
dark:border-form-strokedark dark:bg-form-input dark:focus:border-primary"
                      onChange={handleChange}
                    />

                    <span className="absolute right-4 top-4">
                      <svg
                        className="fill-current"
                        width="22"
                        height="22"
                        viewBox="0 0 22 22"
                        fill="none"
                        xmlns="http://www.w3.org/2000/svg"
                      >
                        <g opacity="0.5">
                        </g>
                      </svg>
                    </span>
                  </div>
                </div>

                <div className="mb-6">
                  <label className="mb-2.5 block font-medium text-black dark:text-white">
                    Re-type Password
                  </label>
                  <div className="relative">
                    <input
                      type="password"
                      placeholder="Re-enter your password"
                      className="w-full rounded-lg border border-stroke bg-transparent py-4
pl-6 pr-10 outline-none focus:border-primary focus-visible:shadow-none
dark:border-form-strokedark dark:bg-form-input dark:focus:border-primary"
                      onChange={handleChange}
                    />
```

```
                    <span className="absolute right-4 top-4">
                      <svg
                        className="fill-current"
                        width="22"
                        height="22"
                        viewBox="0 0 22 22"
                        fill="none"
                        xmlns="http://www.w3.org/2000/svg"
                      >
                        <g opacity="0.5">
                        </g>
                      </svg>
                    </span>
                  </div>
                </div>

                <div className="mb-5">
                  <input
                    type="submit"
                    value="Create account"
                    className="w-full cursor-pointer rounded-lg border border-primary
bg-primary p-4 text-white transition hover:bg-opacity-90"
                  />
                </div>

                <div className="mt-6 text-center">
                  <p>
                    Already have an account?{' '}
                    <Link to="/auth/signin" className="text-primary">
                      Sign in
                    </Link>
                  </p>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </>
  );
};

export default SignUp;
```

## Dashboard/Home.tsx

The Home.tsx component is the central hub of the CrabbyTasks application, providing users with access to their task-related data, statistics, and charts. It seamlessly integrates with the application's context, offers a responsive design, and ensures users have a comprehensive view of their tasks and performance.

```
/*********************************************************************
 * WORK-IN-PROGRESS:
 *    Attach the chart logic
 *
 * File: Home.tsx
 *
 * Logic: Defines the home page component for the application. It displays
 * various cards with user statistics, task charts, and other Task components.
 *
 * Type: React Component
*********************************************************************/


import CardFour from '../../components/CardFour.tsx';
import CardFive from '../../components/CardFive.tsx';
import CardOne from '../../components/CardOne.tsx';
import CardThree from '../../components/CardThree.tsx';
import CardTwo from '../../components/CardTwo.tsx';
import ChartThree from '../../components/ChartThree.tsx';
import ChartTwo from '../../components/ChartTwo.tsx';
import TaskLarge from '../../components/TaskLarge.tsx';
import TaskChart from '../../components/TaskChart.tsx';

import useTaskContext from '../../hooks/useTaskContext';
import {useEffect} from 'react';
import dayjs from 'dayjs';

const Home = () => {

  const today = dayjs();
  const {userTasksStats, updatedStats} = useTaskContext();
```

```
  useEffect(()=>{
    },[updatedStats]);

  return (
    <>
    { userTasksStats ?
      (
       <div className="grid grid-cols-1 gap-4 md:grid-cols-2 md:gap-6 xl:grid-cols-5
2xl:gap-7.5">
       {/*Total Views, Pofit, Product, and Users 'Cards' */}
        <CardOne   userTasksStats={userTasksStats} />
        <CardFive  userTasksStats={userTasksStats} />
        <CardTwo   userTasksStats={userTasksStats} />
        <CardThree userTasksStats={userTasksStats} />
        <CardFour  userTasksStats={userTasksStats} />
      </div>
      ) : (
        <div>Loading Cards...</div>
      )
    }

      <div className="mt-4 grid grid-cols-12 gap-4 md:mt-6 md:gap-6 2xl:mt-7.5 2xl:gap-7.5">
        <div className="col-span-4 xl:col-span-8">
        <TaskChart date={today.toString()}/>
        </div>
        <div className="col-span-4 xl:col-span-4">
          <TaskLarge />
        </div>
        <div className="col-span-4 xl:col-span-8">
        <ChartTwo/>
        </div>
        <ChartThree />

      </div>
    </>
  );
};

export default Home;
```

## LandingPage.tsx

LandingPage.tsx serves as the first point of contact for users visiting CrabbyTasks. It effectively communicates the application's purpose, encourages user engagement through the sign-in call to action, and provides an aesthetically pleasing and responsive design to create a positive initial user experience.

```
/**********************************************************************
 * File: LandingPage.tsx
 *
 * Logic: Defines the landing page component for the application. It displays a
 * welcoming message and encourages users to sign in.
 *
 * Type: React Component
**********************************************************************/


import { Link } from 'react-router-dom';
import background from "../images/landing-bg.jpg";

const LandingPage = () => {
    const backgroundImageStyle = {
        backgroundImage: `url(${background})`,
        backgroundSize: "cover"
    }

    return (
        <div className="LandingPage">
        <div style= {backgroundImageStyle}>
        <div className="bg-gradient-to-r from-boxdark">
            <div className= "flex h-screen ">
                <div className="m-auto">
                    <div className= "mb-8">
                    <div className='flex justify-center'>
                        <h1 className="text-5xl text-whiteGray font-bold">
                            Welcome to CrabbyTasks, your new best friend!
                        </h1>
                    </div>
                    </div>
```

```
                    <div className = "mb-8">
                    <div className='flex justify-center'>
                        <h3 className="max-w-2xl text-2xl text-center text-whiteGray
font-regular">
                            CrabbyTasks is an easy-to-use and engaging scheduling manager, a
To-Do list that thrives as you do when you complete your daily tasks.
                        </h3>
                    </div>
                    </div>
                    <div className= "mb-8">
                    <div className='flex justify-center'>
                        <p className="text-2xl text-secondary font italic">
                           Are you ready to fulfill your dreams... and get things done?
                        </p>
                    </div>
                    </div>
                    <div className= "mb-16">
                    <div className='flex justify-center'>
                        <Link
                        to="/auth/signin"
                        className="inline-flex items-center justify-center rounded-full
bg-gradient-to-r from-primary to-secondary py-4 px-10 text-center font-bold text-white
hover:bg-opacity-500 lg:px-8 xl:px-10"
                        >
                            Sign In Now
                        </Link>
                    </div>
                    </div>

                </div>
            </div>
        </div>
        </div>
        </div>

    )
}

export default LandingPage;
```

## Pet.tsx

Pet.tsx is an early-stage component that is intended to display information about a user's pet Lobster and metrics. However, it is currently a work in progress, and the actual implementation of this functionality is pending. Once completed, this page will provide users with insights and details about their virtual pet within the CrabbyTasks application.

```
/*********************************************************************
 * WORK-IN-PROGRESS:
 *    Implement the pet page
 *
 * File: Pet.tsx
 *
 * Logic: Defines the Pet page component for the application. It displays your
 * pet Lobster, level, and other user metrics.
 *
 * Type: React Component
*********************************************************************/

import Breadcrumb from '../components/Breadcrumb';

const Pet = () => {
    return (
        <>
            <Breadcrumb pageName="Pet" /> {/*Title that goes above content */}
        </>
    );
};

export default Pet;
```

## ContactPage.tsx

ContactPage.tsx is an early-stage component intended to display information about the team or organization behind CrabbyTasks. It provides a brief overview of the team, offers placeholders for social media links, and features visual elements like cover images and a profile picture. The component is currently a work in progress, with the potential for future enhancements related to social media integration.

```
/*********************************************************************
 * WORK-IN-PROGRESS:
 *    Add social button logic
 *
 * File: ContactPage.tsx
 *
 * Logic: Renders a profile or contact page for a team or organization. It
 * displays information about the team's name, description, and social media
 * links.
 *
 * Type: React Component
*********************************************************************/


import Breadcrumb from '../components/Breadcrumb';
import CoverOne from '../images/cover/contactbg.jpg';
import PFP from "../images/cover/contactPFP_cropped.png"

const ContactPage = () => {
  return (
    <>
      <Breadcrumb pageName="Team Behind CrabbyTasks" />

      <div className="overflow-hidden rounded-sm border border-stroke bg-white shadow-default
dark:border-strokedark dark:bg-boxdark">
        <div className="relative z-20 h-35 md:h-65">
          <img
            src={CoverOne}
            alt="ContactPage cover"
            className="h-full w-full rounded-tl-sm rounded-tr-sm object-cover object-center"
```

```
          />
        </div>
        <div className="px-4 pb-6 text-center lg:pb-8 xl:pb-11.5">
          <div className="relative z-30 mx-auto -mt-22 h-30 w-full max-w-30 rounded-full
bg-white/20 p-1 backdrop-blur sm:h-44 sm:max-w-44 sm:p-3">
            <div className="relative drop-shadow-2 scale-125">
              <img src={PFP} alt="ContactPage" />
            </div>
          </div>
          <div className="mt-4">
            <h3 className="mb-1.5 text-2xl font-semibold text-black dark:text-white">
              Red Lobsters LLC
            </h3>
            <p className="font-medium italic">Made in TypeScript JSX</p>


            <div className="mt-4.5 mx-auto max-w-180">
              <h4 className="font-semibold text-black dark:text-white">
                About Us
              </h4>
              <p className="mt-4.5">
                We at Red Lobsters LLC have a succinct focus on making features
                from the user's perspective. From easy to understand UI to
                quality of life implementation; client satisfaction is our
                passion. We are a team of programmers willing to be contracted
                for any fullstack web development project.
              </p>
            </div>

            <div className="mt-6.5">
              <h4 className="mb-3.5 font-medium text-black dark:text-white">
                Like the website? Follow us on:
              </h4>
              <div className="flex items-center justify-center gap-3.5">
                <a
                  href="#"
                  className="hover:text-primary"
                  aria-label="social-icon"
                >
                  <svg
                    className="fill-current"
                    width="22"
                    height="22"
```

```
        viewBox="0 0 22 22"
        fill="none"
        xmlns="http://www.w3.org/2000/svg"
      >
        <g clipPath="url(#clip0_30_966)">
        </g>
        <defs>
          <clipPath id="clip0_30_966">
            <rect width="22" height="22" fill="white" />
          </clipPath>
        </defs>
      </svg>
    </a>
    <a
      href="#"
      className="hover:text-primary"
      aria-label="social-icon"
    >
      <svg
        className="fill-current"
        width="23"
        height="22"
        viewBox="0 0 23 22"
        fill="none"
        xmlns="http://www.w3.org/2000/svg"
      >
        <g clipPath="url(#clip0_30_970)">
        </g>
        <defs>
          <clipPath id="clip0_30_970">
            <rect
              width="22"
              height="22"
              fill="white"
              transform="translate(0.666138)"
            />
          </clipPath>
        </defs>
      </svg>
    </a>
    <a
      href="#"
      className="hover:text-primary"
```

```
            aria-label="social-icon"
          >
            <svg
              className="fill-current"
              width="23"
              height="22"
              viewBox="0 0 23 22"
              fill="none"
              xmlns="http://www.w3.org/2000/svg"
            >
              <g clipPath="url(#clip0_30_974)">
              </g>
              <defs>
                <clipPath id="clip0_30_974">
                  <rect
                    width="22"
                    height="22"
                    fill="white"
                    transform="translate(0.333862)"
                  />
                </clipPath>
              </defs>
            </svg>
          </a>
          <a
            href="#"
            className="hover:text-primary"
            aria-label="social-icon"
          >
            <svg
              className="fill-current"
              width="22"
              height="22"
              viewBox="0 0 22 22"
              fill="none"
              xmlns="http://www.w3.org/2000/svg"
            >
              <g clipPath="url(#clip0_30_978)">
              </g>
              <defs>
                <clipPath id="clip0_30_978">
                  <rect width="22" height="22" fill="white" />
                </clipPath>
```

```
                    </defs>
                  </svg>
                </a>
                <a
                  href="#"
                  className="hover:text-primary"
                  aria-label="social-icon"
                >
                  <svg
                    className="fill-current"
                    width="23"
                    height="22"
                    viewBox="0 0 23 22"
                    fill="none"
                    xmlns="http://www.w3.org/2000/svg"
                  >
                    <g clipPath="url(#clip0_30_982)">
                    </g>
                    <defs>
                      <clipPath id="clip0_30_982">
                        <rect
                          width="22"
                          height="22"
                          fill="white"
                          transform="translate(0.666138)"
                        />
                      </clipPath>
                    </defs>
                  </svg>
                </a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </>
  );
};

export default ContactPage;
```

## Settings.tsx

Settings.tsx is a component that provides users with the ability to update their personal information and notification settings. It uses state to manage the data for the notification settings table and allows users to edit and save changes. It is a work in progress, and specific notification options may be added in the future.

```
/*********************************************************************
 * WORK-IN-PROGRESS:
 *    add the form data for the modal settings
 *
 * File: Settings.tsx
 *
 * Logic: This component renders a settings page where users can update their
 * personal information and notification settings. It includes forms for editing
 * personal details and checkboxes for notification preferences.
 *
 * Type: React Component
*********************************************************************/



import Breadcrumb from '../components/Breadcrumb';
import fireToast from '../hooks/fireToast';
import { Table } from "../components/TableSettings";
import { Modal } from "../components/ModalSettings";
import { useState,useEffect } from "react";
import CheckboxFour from '../components/CheckboxFour';

const Settings = () => {
  const [modalOpen, setModalOpen] = useState(false);
  const [rows, setRows] =
useState(localStorage.getItem("alertSettings")?JSON.parse(localStorage.getItem("alertSettings
")):[]);

  useEffect(() => {
    // storing input name
    localStorage.setItem("alertSettings", JSON.stringify(rows));
  }, [rows]);
```

```
  const [rowToEdit, setRowToEdit] = useState(null);

  const handleDeleteRow = (targetIndex) => {
    setRows(rows.filter((_, idx) => idx !== targetIndex));
  };

  const handleEditRow = (idx) => {
    setRowToEdit(idx);

    setModalOpen(true);
  };

  const handleSubmit = (newRow) => {
    rowToEdit === null
      ? setRows([...rows, newRow])
      : setRows(
          rows.map((currRow, idx) => {
            if (idx !== rowToEdit) return currRow;
            return newRow;
          })
        );
  };

  return (
    <>
      <div className="mx-auto max-w-270">

        <Breadcrumb pageName="Settings" />

        <div className="grid grid-cols-5 gap-8">
          <div className="col-span-5 xl:col-span-3">
            <div className="rounded-sm border border-stroke bg-white shadow-default
dark:border-strokedark dark:bg-boxdark">
              <div className="border-b border-stroke py-4 px-7 dark:border-strokedark">
                <h3 className="font-medium text-black dark:text-white">
                  Personal Information
                </h3>
              </div>
              <div className="p-7">
                <form action="#">
                  <div className="mb-5.5 flex flex-col gap-5.5 sm:flex-row">
                    <div className="w-full sm:w-1/2">
                      <label
```

```
                        className="mb-3 block text-sm font-medium text-black dark:text-white"
                        htmlFor="fullName"
                      >
                        Full Name
                      </label>
                      <div className="relative">
                        <span className="absolute left-4.5 top-4">
                          <svg
                            className="fill-current"
                            width="20"
                            height="20"
                            viewBox="0 0 20 20"
                            fill="none"
                            xmlns="http://www.w3.org/2000/svg"
                          >
                            <g opacity="0.8">
                            </g>
                          </svg>
                        </span>
                        <input
                          className="w-full rounded border border-stroke bg-gray py-3 pl-11.5
pr-4.5 text-black focus:border-primary focus-visible:outline-none dark:border-strokedark
dark:bg-meta-4 dark:text-white dark:focus:border-primary"
                          type="text"
                          name="fullName"
                          id="fullName"
                          placeholder="Carlos Tapia"
                          defaultValue="Carlos Tapia"
                        />
                      </div>
                    </div>

                    <div className="w-full sm:w-1/2">
                      <label
                        className="mb-3 block text-sm font-medium text-black dark:text-white"
                        htmlFor="phoneNumber"
                      >
                        Username
                      </label>
                      <input
                        className="w-full rounded border border-stroke bg-gray py-3 px-4.5
text-black focus:border-primary focus-visible:outline-none dark:border-strokedark
dark:bg-meta-4 dark:text-white dark:focus:border-primary"
```

```
                    type="text"
                    name="username"
                    id="username"
                    placeholder="careless"
                    defaultValue="careless"
                  />
                </div>
              </div>

              <div className="mb-5.5">
                <label
                  className="mb-3 block text-sm font-medium text-black dark:text-white"
                  htmlFor="emailAddress"
                >
                  Email Address
                </label>
                <div className="relative">
                  <span className="absolute left-4.5 top-4">
                    <svg
                      className="fill-current"
                      width="20"
                      height="20"
                      viewBox="0 0 20 20"
                      fill="none"
                      xmlns="http://www.w3.org/2000/svg"
                    >
                      <g opacity="0.8">
                      </g>
                    </svg>
                  </span>
                  <input
                    className="w-full rounded border border-stroke bg-gray py-3 pl-11.5
pr-4.5 text-black focus:border-primary focus-visible:outline-none dark:border-strokedark
dark:bg-meta-4 dark:text-white dark:focus:border-primary"
                    type="email"
                    name="emailAddress"
                    id="emailAddress"
                    placeholder="tapiacarlos32@gmail.com"
                    defaultValue="tapiacarlos32@gmail.com"
                  />
                </div>
              </div>
```

```
                  <div className="flex justify-end gap-4.5">
                    <button
                      className="flex justify-center rounded border border-stroke py-2 px-6
font-medium text-black hover:shadow-1 dark:border-strokedark dark:text-white"
                      type="submit"
                    >
                      Cancel
                    </button>
                    <button
                      className="flex justify-center rounded bg-primary py-2 px-6 font-medium
text-gray hover:shadow-1"
                      type="submit"
                      onClick={fireToast}
                    >
                      Save
                    </button>
                  </div>
                </form>
              </div>
            </div>
          </div>
          <div className="col-span-5 xl:col-span-2">
             {/* <!-- Checkbox and radio button--> */}
          <div className="rounded-sm border border-stroke bg-white shadow-default
dark:border-strokedark dark:bg-boxdark">
            <div className="border-b border-stroke py-4 px-6.5 dark:border-strokedark">
              <h3 className="font-medium text-black dark:text-white">
                Notification Settings
              </h3>
            </div>
            <div className="flex flex-col gap-5.5 p-6.5">
              <CheckboxFour />
            </div>
          </div>

          </div>
        </div>
      </div>
    </>
  );
};
```

```
export default Settings;
```

## TaskManager.tsx

TaskManager.tsx is a dynamic component that allows users to switch between different calendar views (Day, Week, Month), select dates, and view tasks for selected dates. It also handles date checking for tasks and renders components accordingly.

```
/**********************************************************************
 * File: TaskManager.tsx
 *
 * Logic: This component displays a task manager page with options to view tasks
 * in day, week, or month view. It also provides functionality to open and view
 * tasks for a specific date.
 *
 * Type: React Component
**********************************************************************/


import { SetStateAction, useEffect, useState } from 'react';
import Breadcrumb from '../components/Breadcrumb';
import TaskList from '../components/TaskList';
import TaskChart from '../components/TaskChart';
import useAxios from '../hooks/useAxios';
import dayjs from 'dayjs';

// Common Interface imports
import { Task } from '../interfaces/Task';

import { useContext } from 'react';
import React, { ComponentType } from 'react';
import Loader from '../common/Loader';
import useTaskContext from '../hooks/useTaskContext';

const TaskManager = () => {
```

```
  const [calendarView, setCalendarView] = useState("Month"); //Default calendar view is
monthly
  const [taskView, setTaskView] = useState(false); //Default is that task menu is not open
  const [passDate, setDatePass] = useState("00"); //Default is that task menu is not open
  const [canCheckDates, setDateBool] = useState(false);

  const handleCalendarChange = (view: SetStateAction<string>) => {
    setCalendarView(view);
  }

  const openTasks = (date: string) => {
    setDatePass(date);
    setTaskView(true);
    setDateBool(true);
  }

  const {userTasks} = useTaskContext();

  const handleDateCheckHelper = (dateDay : String) => {
    if (canCheckDates && userTasks.length != 0)
    {
      const holdCount = userTasks.filter((x : Task) => (x.deadline.toString().substring(8,10)
== dateDay) && (x.status == "ToDo" ||
      x.status == "In-Progress" || x.status == "Overdue")).length
      if (holdCount == 0) {
        return "";
      }
      else {
        return holdCount;
      }
    }
    else
    {
      return "";
    }
  };

  const handleDateCheck = (dateDay : String) => {
    var holdCount = handleDateCheckHelper(dateDay);
      if (holdCount == 0) {
        return "";
      }
      else {
```

```
        return holdCount;
      }
  };

  useEffect(()=>{
    });

  const CalendarCell = (props) => {
    const {dateTimeValue, onClickHandler} = props;
    return (
        <td className="ease relative h-20 cursor-pointer border border-stroke p-2 transition
duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25 md:p-6
xl:h-31">
          <span className="font-medium text-black dark:text-white" onClick={() =>
onClickHandler(dateTimeValue)}>
            <div>
              {Number(dateTimeValue)}
            </div>
            <div className="ml-40 mt-8 text-danger">
              {handleDateCheck(dateTimeValue)}
            </div>
          </span>
        </td>
      )
    }


  const MonthCalendar = () => {
    return (
      <>
        <div className="w-full max-w-full rounded-sm border border-stroke bg-white
shadow-default dark:border-strokedark dark:bg-boxdark">
          <table className="w-full">
            <thead>
              <tr className="grid grid-cols-7 rounded-t-sm bg-primary text-white">
                <th className="flex h-15 items-center justify-center rounded-tl-sm p-1
text-xs font-semibold sm:text-base xl:p-5">
                  <span className="hidden lg:block"> Sunday </span>
                  <span className="block lg:hidden"> Sun </span>
                </th>
                <th className="flex h-15 items-center justify-center p-1 text-xs
font-semibold sm:text-base xl:p-5">
                  <span className="hidden lg:block"> Monday </span>
```

```
                  <span className="block lg:hidden"> Mon </span>
                </th>
                <th className="flex h-15 items-center justify-center p-1 text-xs
font-semibold sm:text-base xl:p-5">
                  <span className="hidden lg:block"> Tuesday </span>
                  <span className="block lg:hidden"> Tue </span>
                </th>
                <th className="flex h-15 items-center justify-center p-1 text-xs
font-semibold sm:text-base xl:p-5">
                  <span className="hidden lg:block"> Wednesday </span>
                  <span className="block lg:hidden"> Wed </span>
                </th>
                <th className="flex h-15 items-center justify-center p-1 text-xs
font-semibold sm:text-base xl:p-5">
                  <span className="hidden lg:block"> Thursday </span>
                  <span className="block lg:hidden"> Thur </span>
                </th>
                <th className="flex h-15 items-center justify-center p-1 text-xs
font-semibold sm:text-base xl:p-5">
                  <span className="hidden lg:block"> Friday </span>
                  <span className="block lg:hidden"> Fri </span>
                </th>
                <th className="flex h-15 items-center justify-center rounded-tr-sm p-1
text-xs font-semibold sm:text-base xl:p-5">
                  <span className="hidden lg:block"> Saturday </span>
                  <span className="block lg:hidden"> Sat </span>
                </th>
              </tr>
            </thead>
            <tbody>
              {/* <!-- Line 1 --> */}
              <tr className="grid grid-cols-7">
              <CalendarCell dateTimeValue={"01"} onClickHandler={openTasks}/>

                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("02")}>
                  <div>
                      2
                    </div>
                    <div className="ml-40 mt-8 text-danger">
```

```
                    {handleDateCheck("02")}
                  </div>
                </span>
              </td>
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("03")}>
                <div>
                    3
                  </div>
                  <div className="ml-40 mt-8 text-danger">

                    {handleDateCheck("03")}
                  </div>
                </span>
              </td>
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("04")}>
                <div>
                    4
                  </div>
                  <div className="ml-40 mt-8 text-danger">

                    {handleDateCheck("04")}
                  </div>
                </span>
              </td>
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("05")}>
                <div>
                    5
                  </div>
                  <div className="ml-40 mt-8 text-danger">
```

```
                    {handleDateCheck("05")}
                  </div>
                </span>
              </td>
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("06")}>
                <div>
                    6
                  </div>
                  <div className="ml-40 mt-8 text-danger">

                    {handleDateCheck("06")}
                  </div>
                </span>
              </td>
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("07")}>
                <div>
                    7
                  </div>
                  <div className="ml-40 mt-8 text-danger">

                    {handleDateCheck("07")}
                  </div>
                </span>
              </td>
            </tr>
            {/* <!-- Line 1 --> */}

            {/* <!-- Line 2 --> */}
            <tr className="grid grid-cols-7">
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("08")}>
                <div>
```

```
                    8
                  </div>
                  <div className="ml-40 mt-8 text-danger">

                    {handleDateCheck("08")}
                  </div>
                </span>
              </td>
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("09")}>
                <div>
                    9
                  </div>
                  <div className="ml-40 mt-8 text-danger">

                    {handleDateCheck("09")}
                  </div>
                </span>
              </td>
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("10")}>
                <div>
                    10
                  </div>
                  <div className="ml-40 mt-8 text-danger">

                    {handleDateCheck("10")}
                  </div>
                </span>
              </td>
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("11")}>
                <div>
                    11
```

```
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("11")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("12")}>
                  <div>
                      12
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("12")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("13")}>
                  <div>
                      13
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("13")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("14")}>
                  <div>
                      14
                    </div>
```

```
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("14")}
                    </div>
                  </span>
                </td>
              </tr>
              {/* <!-- Line 2 --> */}
              {/* <!-- Line 3 --> */}
              <tr className="grid grid-cols-7">
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("15")}>
                  <div>
                      15
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("15")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("16")}>
                  <div>
                      16
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("16")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("17")}>
```

```
                  <div>
                      17
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("17")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white"onClick={() =>
openTasks("18")}>
                  <div>
                      18
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("18")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("19")}>
                  <div>
                      19
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("19")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("20")}>
                  <div>
```

```
                    20
                  </div>
                  <div className="ml-40 mt-8 text-danger">

                    {handleDateCheck("20")}
                  </div>
                </span>
              </td>
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("21")}>
                <div>
                    21
                  </div>
                  <div className="ml-40 mt-8 text-danger">

                    {handleDateCheck("21")}
                  </div>
                </span>
              </td>
            </tr>
            {/* <!-- Line 3 --> */}
            {/* <!-- Line 4 --> */}
            <tr className="grid grid-cols-7">
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("22")}>
                <div>
                    22
                  </div>
                  <div className="ml-40 mt-8 text-danger">

                    {handleDateCheck("22")}
                  </div>
                </span>
              </td>
              <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
```

```
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("23")}>
                  <div>
                      23
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("23")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("24")}>
                  <div>
                      24
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("24")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("25")}>
                  <div>
                      25
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("25")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
```

```
openTasks("26")}>
                  <div>
                      26
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("26")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("27")}>
                  <div>
                      27
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("27")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("28")}>
                  <div>
                      28
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("28")}
                    </div>
                  </span>
                </td>
              </tr>
              {/* <!-- Line 4 --> */}
              {/* <!-- Line 5 --> */}
              <tr className="grid grid-cols-7">
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
```

```
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("29")}>
                  <div>
                      29
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("29")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("30")}>
                  <div>
                      30
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("30")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white" onClick={() =>
openTasks("31") }>
                  <div>
                      31
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("31")}
                    </div>
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 bg-gray dark:bg-strokedark dark:border-strokedark
```

```
dark:hover:bg-meta-4 md:h-25 md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white">
                    1
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 bg-gray dark:bg-strokedark dark:border-strokedark
dark:hover:bg-meta-4 md:h-25 md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white">
                    2
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 bg-gray dark:bg-strokedark dark:border-strokedark
dark:hover:bg-meta-4 md:h-25 md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white">
                    3
                  </span>
                </td>
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 bg-gray dark:bg-strokedark dark:border-strokedark
dark:hover:bg-meta-4 md:h-25 md:p-6 xl:h-31">
                  <span className="font-medium text-black dark:text-white">
                    4
                  </span>
                </td>
              </tr>
              {/* <!-- Line 5 --> */}
            </tbody>
          </table>
        </div>
        {/* <!-- ====== Calendar Section End ====== --> */}
      </>
    );
  };

  const WeekCalendar = () => {
    return (
      <>
        <div className="w-full max-w-full rounded-sm border border-stroke bg-white
shadow-default dark:border-strokedark dark:bg-boxdark">
          <table className="w-full">
            <thead>
```

```
              <tr className="grid grid-cols-7 rounded-t-sm bg-primary text-white">
                <th className="flex h-15 items-center justify-center rounded-tl-sm p-1
text-xs font-semibold sm:text-base xl:p-5">
                  Sun
                </th>
                <th className="flex h-15 items-center justify-center p-1 text-xs
font-semibold sm:text-base xl:p-5">
                  Mon
                </th>
                <th className="flex h-15 items-center justify-center p-1 text-xs
font-semibold sm:text-base xl:p-5">
                  Tue
                </th>
                <th className="flex h-15 items-center justify-center p-1 text-xs
font-semibold sm:text-base xl:p-5">
                  Wed
                </th>
                <th className="flex h-15 items-center justify-center p-1 text-xs
font-semibold sm:text-base xl:p-5">
                  Thur
                </th>
                <th className="flex h-15 items-center justify-center p-1 text-xs
font-semibold sm:text-base xl:p-5">
                  Fri
                </th>
                <th className="flex h-15 items-center justify-center rounded-tr-sm p-1
text-xs font-semibold sm:text-base xl:p-5">
                  Sat
                </th>
              </tr>
            </thead>
            <tbody>
              {/* Weekly calendar rows */}
              <tr className="grid grid-cols-7">
                {/* Sunday */}
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31"
                  onClick={() => openTasks("22") }>
                  <span className="font-medium text-black dark:text-white">
                  <div>
                      29
                    </div>
```

```
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("29")}
                    </div>

                  </span>
                </td>
                {/* Monday */}
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31"
                  onClick={() => openTasks("30") }>
                  <span className="font-medium text-black dark:text-white">
                  <div>
                      30
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("30")}
                    </div>

                    </span>
                  {/* Add your content for each cell here */}
                </td>
                {/* Tuesday */}
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31"
                onClick={() => openTasks("31") }>
                  <span className="font-medium text-black dark:text-white">

                  <div>
                      31
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                      {handleDateCheck("31")}
                    </div>

                    </span>
                  {/* Add your content for each cell here */}
                </td>
                {/* Wednesday */}
```

```
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31"
                >
                  <span className="font-medium text-black dark:text-white">
                  <div>
                      1
                    </div>
                    <div className="ml-40 mt-8 text-danger">


                    </div>

                    </span>
                  {/* Add your content for each cell here */}
                </td>
                {/* Thursday */}
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31"
              >
                  <span className="font-medium text-black dark:text-white">
                  <div>
                      2
                    </div>
                    <div className="ml-40 mt-8 text-danger">


                    </div>
                    </span>
                  {/* Add your content for each cell here */}
                </td>
                {/* Friday */}
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31"
                >
                  <span className="font-medium text-black dark:text-white">
                  <div>
                      3
                    </div>
                    <div className="ml-40 mt-8 text-danger">
```

```
                    </div>
                  </span>
                  {/* Add your content for each cell here */}
                </td>
                {/* Saturday */}
                <td className="ease relative h-20 cursor-pointer border border-stroke p-2
transition duration-500 hover:bg-gray dark:border-strokedark dark:hover:bg-meta-4 md:h-25
md:p-6 xl:h-31"
                >
                  <span className="font-medium text-black dark:text-white">
                  <div>
                      4
                    </div>
                    <div className="ml-40 mt-8 text-danger">

                    </div>
                    </span>
                  {/* Add your content for each cell here */}
                </td>
              </tr>
            </tbody>
          </table>
        </div>
      </>
    );
  };

  const DayCalendar = () => {
    return (
      <>
        <div className="w-full max-w-full rounded-sm border border-stroke bg-white
shadow-default dark:border-strokedark dark:bg-boxdark">
          <TaskChart date={dayjs().date().toString()}/>
        </div>
      </>
    );
  };

  return (
    <>
      <Breadcrumb pageName="Task Manager" /> {/*Title that goes above calendar/page content
*/}
```

```
      {/* Day Week Month Buttons Start */}
      <div className="flex justify-end">
        <div className="inline-flex items-center rounded-md bg-whiter p-1.5 dark:bg-meta-4">
          <button className={`rounded py-1 px-3 text-xs font-medium text-black hover:bg-white
hover:shadow-card dark:text-white dark:hover:bg-boxdark ${calendarView === 'Day' ? 'bg-white
shadow-card dark:bg-boxdark' : ''}`}
            onClick={() => handleCalendarChange("Day")}>
            Day
          </button>
          <button className={`rounded py-1 px-3 text-xs font-medium text-black hover:bg-white
hover:shadow-card dark:text-white dark:hover:bg-boxdark ${calendarView === 'Week' ? 'bg-white
shadow-card dark:bg-boxdark' : ''}`}
            onClick={() => handleCalendarChange("Week")}>
            Week
          </button>
          <button className={`rounded py-1 px-3 text-xs font-medium text-black hover:bg-white
hover:shadow-card dark:text-white dark:hover:bg-boxdark ${calendarView === 'Month' ?
'bg-white shadow-card dark:bg-boxdark' : ''}`}
            onClick={() => handleCalendarChange("Month")}>
            Month
          </button>
        </div>
      </div>
      {/* Day Week Month Buttons End */}

      {/* <!-- ====== Calendar Section Start ====== --> */}
      {calendarView === "Month" && (<MonthCalendar />)}
      {calendarView === "Week" && (<WeekCalendar />)}
      {calendarView === "Day" && (<DayCalendar />)}
      {/* {taskView && ( openTaskList())} */}

      {taskView && (<TaskList setTaskView={setTaskView} holdDate = {passDate}/>)}
      {/* <!-- ====== Calendar Section End ====== --> */}
    </>
  );
};

export default TaskManager;
```

# Testing Plan

## Testing Matrix

### Functional Testing

| Item Number | Revision Number | Web Page | Acceptance Criteria | Test Description | Expected Result | Actual Result | Pass? | Date of Test | Comments | Sign-Off Name |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.1.1.1** | **0** | **Task Manager** | Application will have the capacity to provide 3 district and thematically cohesive monthly, weekly, and daily calendar views,<br><br>such that the overarching theme captures cretaceous | 1. Refreshing or loading the Task Manager page | 1.Upon loading the Task Manager Page, the calendar with all tasks are loaded to their default display. | 1. Only the weekly calendar is present. Also, testing requirements did not consider the full scope of functional requirements in this section. | No | 10/02/2023 | Need to fix implementation Daily and Monthly calendar views in both front-end and back-end. | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | sea life through it's red colors, textures, and images; While enabling<br><br>simple, organized, viewable representation of at least 3 tasks relative to their respective type of view. | | | | | | | |
| | | | | 2. Selecting an option for display on the Task Manager Calendar and validating the updated display. | 2. Selection of a calendar display option causes the calendar to refresh and update to the new layout/display with all of the appropriate tasks | 2. The weekly calendar was not interactable. | No | 10/02/2023 | Need to adjust the HTML of the Task Manager page to make the calendar's | Reviewed NO by Nick Dilallo, Testing Manager, 10/02/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | interactable and clickable. | |
| **1.1.1.1** | **1** | **Task Manager** | Application will have the capacity to provide 3 district and thematically cohesive monthly, weekly, and daily calendar views,<br><br>such that the overarching theme captures cretaceous sea life through it's red colors, textures, and images; While enabling<br><br>simple, organized, viewable | 1. Refreshing or loading the Task Manager page | 1.Upon loading the Task Manager Page, all calendars with all tasks are loaded to their default display of being filled with tasks with corresponding dates. | All 3 calendar types are present. However, the proper site colors are set. Additionally, the tasks have a signifier from the calendar view. | Yes | 10/27/2023 | Red colors swapped for a blue theme. Instead of previewing task descriptions from the calendar, the number of total tasks for that day is shown. | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | representation of at least 3 tasks relative to their respective type of view. | | | | | | | |
| | | | | 2. Selecting an option for display on the Task Manager Calendar and validating the updated display. | 2. Selection of a calendar display option causes the calendar to refresh and update to the new layout/display with all of the appropriate tasks | Selection of a date on the calendar causes the page to update so that it includes a table view of the user's tasks for the appropriate day. | Yes | 10/27/2023 | N/A | Reviewed YES by Nick Dilallo, Testing Manager, 10/27/2023 |
| ---------- | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.1.2.1** | **0** | **Task Manager** | The application will automatically populate monthly, weekly, daily calendar views in an organized and uniform way<br><br>relative to the type of view that is currently presented to the user while also maintaining task functionality, thematic cohesiveness,<br><br>and system integrity. | *1. Duplicated Test - See Item 1.1.1.1 - Test #1* | *1. Duplicate - See Item 1.1.1.1 - Expected Result #1* | 1. Only the weekly calendar is present. | No | 10/02/2023 | Need to fix implementation Daily and Monthly calendar views in both front-end and back-end. | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2. After adding a task, load each of the calendar views | 2. The information that was input, as well as any prior inputted data, is properly displayed in each calendar view. | 2. Add task functionality works correctly, but only in the weekly calendar view. | No - due to lack of testability of Daily and Monthly calendar views. | 10/02/2023 | Need to fix implementation Daily and Monthly calendar views in both front-end and back-end. Also, need to add clarity to testing description #2. | Reviewed NO by Nick Dilallo, Testing Manager, 10/02/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.1.2.1** | **1** | **Task Manager** | The application will automatically populate monthly, weekly, daily calendar views in an organized and uniform way<br><br>relative to the type of view that is currently presented to the user while also maintaining task functionality, thematic cohesiveness,<br><br>and system integrity. | *1. Duplicated Test - See Item 1.1.1.1, Revision 1 - Test #1* | *1. Duplicate - See Item 1.1.1.1, Revision 1 - Expected Result #1* | All 3 calendar types are present. The tasks are previewed from the calendar view. | Yes | 10/27/2023 | Instead of showing description previews, the total number of tasks for a day is shown on the calendar. | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2. After adding a task, the webpage must reload and each of the correct calendar views should display the new task. | 2. The information that was input, as well as any prior inputted data, is properly displayed in each calendar view. | 2. A preview for the days tasks is shown to the user. | Yes | 10/27/2023 | Preview of tasks should be added | Reviewed YES by Nick Dilallo, Testing Manager, 10/27/2023 |
| ---------- | | | | | | | | | | |
| **1.1.3.1** | **0** | **Task Manager** | Upon interaction with any specific task component, application will dynamically update task view to fit a view height and width capable of | 1. After adding one or more tasks with their required details, click on any listed task component. | 1. A view of the task and all of its necessary details are displayed with the correct | 1. Tasks are not interactable and therefore no screen can be displayed | No | 10/03/2023 | Need to adjust HTMl of the calendar to make tasks interactable. Front-end development | Reviewed NO by Nick Dilallo, Testing Manager, 10/03/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | holding a 1.2 rem task title and 1 rem task description with enough padding to also hold any additional feature flags and descriptors. | | measurements/sizings | | | | needed on task details page that will be displayed on click. | |
| **1.1.3.1** | **1** | **Task Manager** | Upon interaction with any specific task component, application will dynamically update task view to fit a view height and width capable of holding a 1.2 rem task title and 1 rem task description with enough padding to | 1. After adding one or more tasks with their required details, click on any listed task component. | 1. A view of the task and all of its necessary details are displayed with the correct measurements/sizings | 1. Tasks are interactable and the table with data is displayed. | Yes | 10/28/2023 | | Reviewed Yes by Nick Dilallo, Testing Manager, 10/28/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | also hold any additional feature flags and descriptors. | | | | | | | |
| | | | | | | | | | | |
| **1.1.4.1** | **0** | **Task Manager** | Within the task manager page, users will find various task management options, including adding new tasks, deleting tasks, editing task details, setting task priorities, assigning task categories, and opting to receive | 1. Load the task manager page and attempt to add, delete, update, set priority, and set email preferences | 1. The task should be created, updated, and deleted accordingly. Email notifications for the tasks should be a present option. | 1. The tasks can perform all basic CRUD operations, however, there is no option for categorizing or setting email settings. | No | 10/24/2023 | Email Notifications and categorization options need to be added | Reviewed NO by Nick Dilallo, Testing Manager, 10/24/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | email notifications for task updates. | | | | | | | |
| **1.1.4.1** | **1** | **Task Manager** | Within the task manager page, users will find various task management options, including adding new tasks, deleting tasks, editing task details, setting task priorities, assigning task categories, and opting to receive email notifications for task updates. | 1. Load the task manager page and attempt to add, delete, update, set priority, and set email preferences | 1. The task should be created, updated, and deleted accordingly. Email notifications for the tasks should be a present option. | 1. The tasks can perform all basic CRUD operations and options for notifications and categorization are present when creating a task. | Yes | 10/28/2023 | | Reviewed Yes by Nick Dilallo, Testing Manager, 10/28/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |
| **1.1.4.2** | **0** | **Task Manager** | Additionally, a dedicated "Add Task" function will be available in both the weekly and daily calendar views for user convenience. | 1. Load the task manager page and select the option for adding a new task. | 1. The modal for creating a new task should be opened. | 1. Selecting a date on the calendar causes the modal is opened and a new task can be opened. | Yes | 10/28/2023 | N/A | Reviewed YES by Nick Dilallo, Testing Manager, 10/28/2023 |
| | | | | | | | | | | |
| **1.2.1.1** | **0** | **Login Page** | On the user login page, provide an option for users to | 1. Load the Login Page | 1. The Default view/layout should be present | 1. Upon load the default sign in screen with | Yes | 10/25/2023 | N/A | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | create a new account. | | | textboxes for input is displayed | | | | |
| | | | | 2. Look for button to sign in | 2. The button for signing up is present and clickable by the user | 1. The button is not present | No | 10/25/2023 | Button for redirect is not present | |
| | | | | 3. Selection of 'Sign Up' button redirects the user to a page with the necessary options for creating a new account | 3. The Sign Up Page is present and has all necessary features | 3. The 'Sign Up' Page does not exist | No | 10/25/2023 | Only Login is present | Reviewed NO by Nick Dilallo, Testing Manager, 10/25/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.1.1** | **1** | **Login Page** | On the user login page, provide an option for users to create a new account. | 1. Load the Login Page | 1. The Default view/layout should be present | 1. Upon load the default sign in screen with textboxes for input is displayed | Yes | 10/25/2023 | N/A | |
| | | | | 2. Look for button to sign in | 2. The button for signing up is present and clickable by the user | 1. The button is present at the bottom of the Login area | Yes | 10/28/2023 | N/A | |
| | | | | 3. Selection of 'Sign Up' button redirects the user to a page with the necessary options for | 3. The Sign Up Page is present and has all necessary features | 3. The 'Sign Up' Page exists and has all of the required fields for making an account such as | Yes | 10/28/2023 | Page functions as required | Reviewed YES by Nick Dilallo, Testing |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | creating a new account | | Name, Email, and Password | | | | Manager, 10/28/2023 |
| | | | | | | | | | | |
| **1.2.2.1** | **0** | **Login Page** | Allow both new and existing users to log in with their correct username and password. | *1. Duplicated Test - See Item 1.2.1.1 - Test #1* | *1. Duplicated Test - See Item 1.2.1.1 - Test #1* | 1. Upon load the default sign in screen with textboxes for input is displayed | Yes | 10/25/2023 | N/A | |
| | | | | 2. Create a new account with custom username/email and password. Then attempt to login to the newly | 2. The website main website should be loaded and the user should be signed into the account they | 2. The user is not able to sign into a custom account | No | 10/26/2023 | Only one username/password combination works - The | Reviewed NO by Nick Dilallo, Testing Manager, 10/26/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | created account after reloading the website | created/logged into. | | | | admin account | |
| **1.2.2.1** | **1** | **Login Page** | Allow both new and existing users to log in with their correct username and password. | *1. Duplicated Test - See Item 1.2.1.1 - Test #1* | *1. Duplicated Test - See Item 1.2.1.1 - Test #1* | 1. Upon load the default sign in screen with textboxes for input is displayed | Yes | 10/28/2023 | N/A | |
| | | | | 2. Create a new account with custom username/email and password. Then attempt to login to the newly created account | 2. The website main website should be loaded and the user should be signed into the account they created/logged into. | 2. The user can create a new account and then sign into it. | Yes | 10/28/2023 | N/A | Reviewed YES by Nick Dilallo, Testing Manager, 10/28/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | after reloading the website | | | | | | |
| | | | | | | | | | | |
| **1.2.2.2** | **0** | **Login Page** | Upon successful login, Application will redirect users to task manager page which contains the task dashboard and auxiliary task manipulation functions; Only then, if the user manipulates any task present on the viewfinder, the | 1. Login to an existing CrabbyTasks account | 1. The site should successfully login and redirect to the Dashboard page of the website. | 1. After entering a valid email/password combination, the site logs in successfully and loads the initial Dashboard page | Yes | 10/28/2023 | N/A | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | application will functionally<br><br>handle all events related to acquiring, saving, and viewing the manipulated task to lower mental user overhead. | | | | | | | |
| | | | | 2. Attempt to manipulate existing tasks and create new ones for the account from the dashboard. | 2. The user can perform all CRUD operations on pre-existing and newly created tasks on the signed-in account. The site should automatically handle all events | 2. The tasks are able to be manipulated and new ones can be created with the same functionality. Performing any of these operations causes the site | Yes | 10/28/2023 | N/A | Reviewed YES by Nick Dilallo, Testing Manager, 10/28/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | related to the user's actions. | to handle the event. | | | | |
| | | | | | | | | | | |
| **1.2.3.1** | **0** | **Login Page** | Offer a "Change Password" option within the login section for users who wish to reset their password. | *1. Duplicated Test - See Item 1.2.1.1 - Test #1* | *1. Duplicated Test - See Item 1.2.1.1 - Test #1* | 1. Upon load the default sign in screen with textboxes for input is displayed | Yes | 10/26/2023 | N/A | |
| | | | | 2. Find and select the button for resetting an account's password | 2. The user is redirected to a page where they can reset their password through | 2. There is no option or page for the user to reset their password | No | 10/26/2023 | Page and button for redirection need to be made. Connection | Reviewed NO by Nick Dilallo, Testing |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | their account's email. | | | | with the backend is also required. | Manager, 10/26/2023 |
| | | | | | | | | | | |
| **1.2.3.2** | **0** | **Login Page** | When resetting a password, require users to input their username and specify the corresponding email address in order to receive password reset instructions. | 1. Load the Password Reset Page | 1. The Default view/layout should be present with all necessary fields/textboxes. | 1. Upon load the default sign in screen with textboxes for username and email are present. | No | 10/26/2023 | Page does not exist - *See Test 1.2.3.1 v0* | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2. Enter a valid username/email combination and press the reset button | 2. Instructions are given to the user on how they can proceed with resetting the password for their account. | 2. Page does not exist | No | 10/26/2023 | Page does not exist - *See Test 1.2.3.1 v0* | Reviewed NO by Nick Dilallo, Testing Manager, 10/26/2023 |
| | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.4.1.1** | **0** | **Task Manager** | When the user is on the Task Manager page and is adding or editing a task, an option will be offered to opt-in for e-mail notifications for that specific task. The option for e-mail notifications will be defaulted to "No". If opted-in for email notifications, an additional dropdown or selection will be offered to choose how many hours/days prior to the deadline of the selected task the | 1. With the task add or edit menu is open find and select the email notifications category. | 1. Before any option is chosen by the user, the option "No" should be selected. Clicking on the dropdown reveals multiple options for the notifications | 1. No option exists for notifications | No | 10/24/2023 | Task edit/Add menus do not have notification functionality | Reviewed NO by Nick Dilallo, Testing Manager, 10/24/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | user wants to be notified via email. | | | | | | | |
| **1.2.4.1.1** | **1** | **Task Manager** | When the user is on the Task Manager page and is adding or editing a task, an option will be offered to opt-in for e-mail notifications for that specific task. The option for e-mail notifications will be defaulted to "No". If opted-in for email notifications, an additional dropdown or selection will be offered to choose how many | 1. With the task add or edit menu is open find and select the email notifications category. | 1. Before any option is chosen by the user, the option "No" should be selected. Clicking on the dropdown reveals multiple options for the notifications | 1. The option "No is selected by default. Opening the dropdown reveals various options for the user to select from for notifications relating to the specific task. | Yes | 10/28/2023 | N/A | Reviewed YES by Nick Dilallo, Testing Manager, 10/28/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | hours/days prior to the deadline of the selected task the user wants to be notified via email. | | | | | | | |
| | | | | | | | | | | |
| **1.2.4.2.1** | **0** | **N/A** | When on the Task Manager page and the user is adding or editing a task, if the option for desktop notifications is selected, then an additional dropdown or selection will be offered to choose how many hours/days prior to | 1. When creating a new Task, select the proper notification setting as the option for 'Notifications'. | 1. The 'Notifications' option should be set/chosen. | 1. The selection for notifications is successfully saved and chosen for the user. | Yes | 11/01/2023 | All dropdown options work as intended | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | the task deadline the user wants to be notified on desktop (if they are on/at the computer). The default for the desktop notification option will be "No". | | | | | | | |
| | | | | 2. Check for the new dropdown menu for selecting a time to be reminded with notifications | 2. A new dropdown exists asking the user to input how many hours/days before the due date they would like to be reminded. | 2. A new dropdown menu for inputting a time to be reminded does not exist/appear. | No | 11/01/2023 | Dropdown needs to be created and linked to Notifications dropdown. | Reviewed NO by Nick Dilallo, Testing Manager, 11/01/2023 |
| | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.4.3.1** | **0** | **Task Manager** | Allow users to select an approximate date and time to receive email notifications about their tasks. | *1. Duplicated Test - See Item 1.2.4.2.1 - Test #1* | *1. Duplicated Test - See Item 1.2.4.2.1 - Test #1* | *1. Duplicated Test - See Item 1.2.4.2.1 - Test #1* | Yes | 11/01/2023 | | |
| | | | | *2. Duplicated Test - See Item 1.2.4.2.1 - Test #2* | *2. Duplicated Test - See Item 1.2.4.2.1 - Test #2* | *2. Duplicated Test - See Item 1.2.4.2.1 - Test #2* | No | 11/01/2023 | Dropdown for reminder time not implemented | Reviewed NO by Nick Dilallo, Testing Manager, 11/01/2023 |
| | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.5.1** | **0** | **Navbar/All Pages** | Allow users to select between light and dark modes for the theme of the website. | 1. Identify Slider in the Navbar for changing themes. | 1. Toggle button should be present in the navbar of each page. | 1. Button is not present in the navbar or website. | No | 10/20/2023 | Toggle button needs to be implemented visually. | |
| | | | | 2. Click on the theme button slider and identify change in website colors. | 2. Site theme should change from light to dark or dark to light depending on the current theme. | 2. Theme is not changeable. | No | 10/20/2023 | Toggle button does not exist. Needs to be connected to the backend in order to save for the specific user. | Reviewed NO by Nick Dilallo, Testing Manager, 10/20/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.5.1** | **1** | **Navbar/All Pages** | Allow users to select between light and dark modes for the theme of the website. | 1. Identify Slider in the Navbar for changing themes. | 1. Toggle button should be present in the navbar of each page. | 1. Button is present in the navbar of the website. | Yes | 10/25/2023 | | |
| | | | | 2. Click on the theme button slider and identify change in website colors. | 2. Site theme should change from light to dark or dark to light depending on the current theme. | 2. Site theme is changed to the opposite theme upon clicking the button. | Yes | 10/25/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 10/25/2023 |
| | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.6.1** | **0** | **Task Manager** | On the task manager page and within each of the edit task sections and add task sections, allow users to modify the priority preference of any task and ensure that the changes are saved as well as represented across all views the tasks are present in. | 1. When creating a new task or editing a task, open the option for priority and select a new setting. | 1. The option to change or set the priority preference of the task should be present and interactable. | 1. When updating or creating a task there is a dropdown present labeled priority which allows for the user to select from a list of options for the priority of the task. | Yes | 11/01/2023 | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2. Return to the Task Manager page and confirm that the update has been saved and is updated in any location that it is visible. | 2. The modified or created task should have its new setting saved and updated visually across the website. | 2. The task properly saved to the backend and is dynamically updated wherever it appears to reflect the new priority. | Yes | 11/01/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 11/01/2023 |
| | | | | | | | | | | |
| **1.2.6.2** | **0** | **Task Manager** | On the task manager page, within each of the edit task sections and add task sections, enable users to assign categories to tasks | 1. When creating a new task or editing a task, open the option for category and | 1. The option to change or set the category of the task should be present and interactable. | 1. When updating or creating a task there is a dropdown present labeled Category which allows for the | Yes | 11/01/2023 | Colored Flags swapped for Category titles such as work, school, ... | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | using colored flags and ensure that the changes are saved as well as represented across all views the tasks are present in. | select a new setting. | | user to select from a list of options for the category of the task. | | | | |
| | | | | *2. Duplicated Test - See Item 1.2.6.1 - Test #2* | *2. Duplicated Test - See Item 1.2.6.1 - Test #2* | *2. Duplicated Test - See Item 1.2.6.1 - Test #2* | No | 11/01/2023 | Tasks with same category are not grouped. Category is not visual unless the task is clicked into. | Reviewed NO by Nick Dilallo, Testing Manager, 11/01/2023 |
| | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.6.3.1** | **0** | **Dashboard** | The application will have a dedicated component in the viewfinder of the page where all complete, and incomplete tasks reside; Upon arrival of this page, tasks will be ordered by the date of which they were conceived; This component will allow the user to programmatically manipulate the order of which the tasks are painted on to the screen; Additionally, the component will not | 1. As a user with pre-existing tasks, load the Dashboard and identify the viewfinder. | 1. These viewfinder holding the user's tasks should be present and have all of the users tasks. | 1. The viewfinder is present and contains all of the user's created tasks - Whether it be complete or not. | Yes | 10/23/2023 | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | jeopardize the integrity of the system, thematic cohesiveness of the system, or responsiveness of the system; | | | | | | | |
| | | | | 2. Identify whether the tasks are present in the order they were conceived. | 2. The user's tasks present in the viewfinder should be ordered by date of conception. | 2. The tasks are not ordered properly. | No | 10/23/2023 | Implement a proper ordering for the tasks when they are placed in the viewfinder. | Reviewed NO by Nick Dilallo, Testing Manager, 10/23/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.6.3.1** | **1** | **Dashboard** | The application will have a dedicated component in the viewfinder of the page where all complete, and incomplete tasks reside; Upon arrival of this page, tasks will be ordered by the date of which they were conceived; This component will allow the user to programmatically manipulate the order of which the tasks are painted on to the screen; Additionally, the component will not | 1. As a user with pre-existing tasks, load the Dashboard and identify the viewfinder. | 1. The viewfinder holding the user's tasks should be present and have all of the users tasks. | 1. The viewfinder is present and contains all of the user's created tasks - Whether it be complete or not. | Yes | 10/28/2023 | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | jeopardize the integrity of the system, thematic cohesiveness of the system, or responsiveness of the system; | | | | | | | |
| | | | | 2. Identify whether the tasks are present in the order they were conceived. | 2. The user's tasks present in the viewfinder should be ordered by date of conception. | 2. The tasks are ordered properly - Date of conception. | Yes | 10/28/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 10/28/2023 |
| | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.6.4** | **0** | **Task Manager** | On the task manager page, a function will be implemented to allow users to order tasks by category using colored flags in a drop down. By clicking on the flag/category icon next to each task, a selection of colors will be offered to set for that flag's color. The default flag color will be grey to indicate no category. | 1. As a user with pre-existing tasks, set the category for each. Confirm whether the grouping is updated and correct. | 1. The user's tasks are ordered/grouped based on the category attached to the task. | 1. The category for each task does not change their position/ordering. | No | 11/01/2023 | Grouping by category is not an implemented feature. Note: Colored flags swapped for categories such as work, school, ... | Reviewed NO by Nick Dilallo, Testing Manager, 11/01/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |
| | | | | | | | | | | |
| **1.2.6.5** | **0** | **Task Manager** | In the task manager page, include a function that allows users to order tasks based on the time, respective to the user's timezone, they were last edited or, if there are no previous edits, based on when it was created. | 1. As a user with pre-existing tasks, check for a function/option to change how the tasks are ordered. | 1. The user's tasks should have the option to be ordered by either date edited or last created. | 1. There is no option for modifying the order of the user's tasks. | No | 10/28/2023 | Tasks are ordered by date conceived with no option to change this. | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2. Select a new ordering from the provided options for the tasks. | 2. Selection of the ordering option causes the tasks to be ordered by the newly selected option. This is updated in real time for the user. | 2. There is no option for modifying the order of the user's tasks. | No | 10/28/2023 | Tasks are ordered by date conceived with no option to change this. | Reviewed NO by Nick Dilallo, Testing Manager, 10/28/2023 |
| | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.6.6** | **0** | **Dashboard/Task Manager** | In the tasks catalog, incorporate a function that allows users to order tasks by a specified range of dates, in which they are due, entered into the filter. The due dates of the tasks should be consistent with the user's request; Ensure the calendar is updated to the correct year as well. | 1. As a user with pre-existing tasks, check for a function/option to filter the tasks by due date. | 1. The user's tasks should have the option to be filtered by due date. | 1. There is no option for filtering the user's tasks. | No | 10/28/2023 | Tasks cannot be filtered. | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2. Select a custom date range to filter tasks by. | 2. Input of a date range causes the tasks to be filtered by the newly selected option. This is updated in real time for the user. | 2. There is no option for filtering the user's tasks. | No | 10/28/2023 | Tasks cannot be filtered. | Reviewed NO by Nick Dilallo, Testing Manager, 10/28/2023 |
| | | | | | | | | | | |
| **1.2.7.1** | **0** | **Task Manager** | On the task manager page, implement a clickable button labeled "Complete Tasks" that opens a menu which | 1. Acting as a user with task history, open the Task Manager page. | 1. The "Complete Tasks" button will be displayed and interactable by the user. | 1. There is no "Complete Tasks" button. | No | 10/26/2023 | Complete Tasks Menu does not exist, however a running count of the | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | provides a clear and detailed<br><br>display of all of the user's previously completed tasks. | | | | | | completed tasks does. | |
| | | | | 2. Click on the "Complete Tasks" button. | 2. A menu/display window is displayed on the screen. | *Above tasks failed* | No | 10/26/2023 | | |
| | | | | 3. Visually confirm that all of the necessary and correct information is being displayed | 3. The menu that is displayed on the screen is the "Complete Tasks" menu as intended and its contents are filled with the | *Above tasks failed* | No | 10/26/2023 | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | for the task that was selected. | corresponding testing data. | | | | | |
| | | | | 4. Close the "Complete Tasks" menu and add a mark another task that was previously not completed as complete. Re-open the "Complete Tasks" menu by clicking on the "Complete Tasks" button once again. | 4. When a new task is completed using the complete task functionality, it is dynamically displayed when the "Complete Tasks" menu is next opened and every time thereafter. | *Above tasks failed* | No | 10/26/2023 | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 5. Click outside of the "Complete Tasks" menu or click the "Complete Tasks" button again. | 5. These attempts to click outside of the "Complete Tasks" menu or clicking the "Complete Tasks" button while the menu is open result in the menu closing as intended. | *Above tasks failed* | No | 10/26/2023 | | Reviewed NO by Nick Dilallo, Testing Manager, 10/26/2023 |
| | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1.2.7.2** | **0** | **Task Manager** | A viewable task completion scoring system will be implemented in hopes of showing the user(s) progress and encourage them to complete more tasks earlier. The scoring system will reward points upon task completion with an increasing amount depending on how much earlier the task was completed compared to its deadline. Conversely, points will be deducted if tasks are not | 1. Check whether a page exists for tracking the user's score. | 1. A page for tracking a user's score exists and is present on the sidebar. | 1. A "Pets" page exists where the user's pet is shown and tracks the user's score. | Yes | 11/02/2023 | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | completed on time, with the amount taken away increasing as the amount of time past the deadline increases in order to make users finish up late tasks and prevent it from happening in the future. | | | | | | | |
| | | | | 2. Marking a task as complete needs to dynamically increase the Task Completion Score. | 2. Marking a task as complete will increase the amount displayed as the Task Completion Score. | 2. Completing a task does not affect the user's current score. | No | 11/02/2023 | Score system has not been implemented on backend or frontend, only the page to hold it. | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 2.1. Marking a task as complete needs to dynamically increase the Task Completion Score by a variable amount based upon the time remaining until its due date. | 2.1. Marking a task as complete with approximately a day's time remaining until its due date, 24 hours, will increase their Task Completion Score by 24. | *Above tasks failed* | No | 11/02/2023 | | |
| | | | | 3. Marking a task as complete past its due date needs to dynamically decrease the | 3. Marking a task as complete approximately a day's time after its due date, 24 hours, will decrease their | *Above tasks failed* | No | 11/02/2023 | | Reviewed NO by Nick Dilallo, Testing Manager, 11/02/2023 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Task Completion Score. | Task Completion Score by 24. | | | | | |

## System Testing

| Item Number | Revision Number | Acceptance Criteria | Test Description | Expected Result | Actual Result | Pass? | Date of Test | Comments | Sign-Off Name |
|---|---|---|---|---|---|---|---|---|---|
| **2.1.1** | **0** | System will require Database capable of holding client demand on platform | 1. Act as multiple clients by inputting custom data and ensure it is properly stored/saved to the database. | 1. Each account's information is transferred to the backend database and stored. It can be retrieved at any time. | 1. Client demand such as inputting or requesting stored information works properly. Exchange between user and backend works as expected. | Yes | 10/25/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 10/25/2023 |
| | | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **2.1.1.1** | **0** | A secured AWS EC2 instance holding (MongoDB \|\| Postgresql \|\| DynamoDB) will have an exclusive channel of communication to the production server. | 1. Access the server online. Check AWS to ensure the connection is made. | 1. The production server should be accessible online as it is hosted by the EC2 Instance and connected with any changes that are made. | 1. The production server is accessible and is linked on AWS to the EC2 instance that was created. | Yes | 10/18/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 10/18/2023 |
| | | | | | | | | | |
| **2.1.2** | **0** | Server must concurrently handle multiple connections for several users. | 1. Working from two different windows, sign into multiple accounts at the same time on the website. | 1. Both accounts will sign in properly and without issue. Both users can access all features of the website without any issue. | 1. Both accounts were able to sign in and use all features such as creating, updating, or deleting tasks at the same time as another user was signed in and doing the same thing. | Yes | 10/25/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 10/25/2023 |
| | | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **2.1.2.1** | **0** | A secured AWS EC2 instance has Apache2 server handling incoming requests to the server | 1. Ensure the Apache2 server is running and has a link to the proper EC2 instance. | 1. The Apache2 server is running and has no errors. The EC2 instance is linked to the same Apache2 server. | 1. The server is running without issue and linked to the instance. It can handle all incoming request allowing for the website to work properly. | Yes | 10/18/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 10/18/2023 |
| | | | | | | | | | |
| **2.1.3** | **0** | System must be able to hold against security threats | 1. As arbitary user, attempt to sign in from foreign laptop to default port 22 | 1. Connection refused due to invalid keys and invalid port number | 1. Server refused connection because I attempted to connect to default port 22 without valid key | Yes | 11/11/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |
| | | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **2.1.3.1** | **0** | Ssh Protocol will be secured using the latest verified security standards that are available to the group after comparing the pros and cons of each standard/version to make sure they work best for the project. | 1. Run security audit for standard security practices:Re-generate the RSA and ED25519 keys, Remove small Diffie-Hellman moduli,Enable the RSA and ED25519 keys,Restrict supported key exchange, cipher, and MAC algorithms, | 1. Score A on Standard Security Audits using www.sshaudit.com | 1. Score 93/100 on sshaudit.com | | 11/11/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |
| | | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **2.1.4** | **0** | System must provide a user space separate from other users | 1. As user one, add a task and other information to the backend. Then as user two, confirm that the information from user one did not translate over and vice-versa. | 1. Any data stored by one user should be separate from all other users who have their own data. | 1. When one user adds a task it affects all other users. There is only one account in the background holding everything. | No | 10/18/2023 | | Reviewed NO by Nick Dilallo, Testing Manager, 10/18/2023 |
| **2.1.4** | **1** | System must provide a user space separate from other users | 1. As user one, add a task and other information to the backend. Then as user two, confirm that the information from user one did not translate over and vice-versa. | 1. Any data stored by one user should be separate from all other users who have their own data. | 1. When one user adds a task its effect is not seen by any other users. All account data is stored in separate areas that are inaccessible by other users. | Yes | 10/18/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 10/24/2023 |
| | | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **2.1.4.1** | **0** | Unauthorized users will not be able to access personal user pages. The user must have valid credentials and have a verified method of communication to reset password if necessary. This includes developers unless necessary edits or bug fixes are needed and require a look into a user's page | 1. Attempt to log into a user's account without the valid credentials. | 1. If either the username, password, or both are incorrect, the user will not be granted access into an account, even if one of the criteria is correct. | 1. Attempting to sign into an account with having both the correct username and the corresponding password failed and did not sign the user in. | Yes | 10/25/2023 | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | 2. Click on the reset password option and ensure that it is sent to the email linked to the user's account if it exists. | 2. If the user has an email linked to their account, choosing the reset password option causes an email with instruction on how to reset their password is sent. | 2. The user is not able to request a reset for their account, even when it has a proper email attached. | No | 10/25/2023 | Need to implement 'Forgot Username/Password' options | Reviewed NO by Nick Dilallo, Testing Manager, 10/25/2023 |
| | | | | | | | | | |
| **2.1.5** | **0** | System must onboard users seamlessly | 1. Create an account using valid parameters. Check in the backend to make sure the user has been created with the correct username/password combination. | 1. The backend framework should have created and be holding the user's new information. | 2. The backend is properly storing the user's login info during onboarding. | Yes | 10/25/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 10/25/2023 |
| | | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **2.1.5.1** | **0** | Introducing users to the system will consist of an overview of how the application functions. | 1. Create a new account and the website for the first time. | 1. When first entering the website as a new user, the website should provide an overview/tutorial that explains the functionalities and where to find them. | 1. Landing on the main website as a user does not result in any tutorial or functionality being explained to the user. | No | 11/01/2023 | Landing page gives brief intro to the site but does not explain how to use it. | Reviewed NO by Nick Dilallo, Testing Manager, 11/01/2023 |
| | | | | | | | | | |
| **2.1.6** | **0** | System must onboard developer seamlessly | 1. Follow technical guide literally | 1. Have developer running local development servers | 1. Following guide literally caused tester to create environment | Yes | 11/11/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |
| | | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **2.1.6.1** | **0** | In the event that a new developer is introduced to the system, the developer will be introduced to the project manual and an environment image to help reduce overhead into the system. They will also be shadowed by or work with more experienced developers | 1. Have user scan project manual literally and quiz user on 1 generic aspect of the section. | 1. Answer all questions with an accuracy of 80 percent | 1. All questions answers can be found inside the project booklet | Yes | 11/11/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |
| | | | | | | | | | |
| **2.1.7** | **0** | System communication will be held over a trusted secured encrypted TCP connection | 1. initiate communicatio n with server, and then disconnect from network. | 1. Connection terminated gracefully with error message stating connection lost | 1. Connection terminated and user was redirected to error page | Yes | 11/11/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |
| | | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **2.1.7.1** | **0** | Inbound connections to the server will be encrypted and be up to SSL standards. | 1. Intercept packets between communicatio n of client and server | 1. encrypted packets | 1. Packets are not legible to a human and cannot be encrypted by common attacks | Yes | 11/11/2023 | Wireshark as interceptor | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |
| | | | | | | | | | |
| **2.1.8** | **0** | Production server and database will establish secure isolated communication | 1.Securely connect to production server; thereafter, connect to database and initiate unsecured communicatio n commands to database | 1. Database refuses response | 1. The system hangs until timeout response occurs | Yes | 11/11/2023 | | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |
| | | | | | | | | | |
| **2.1.8.1** | **0** | Database will only allow communication to the application server. | 1. Port scanning network for open ports through foreign device, and send request to | 1. Connection refused | 1. Connection refused | Yes | 11/11/2023 | toolset Linux CLI toolset: netstat, nmap, etc | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | open ports | | | | | | |
| | | | | | | | | | |
| **2.1.9** | **0** | System will be hosted to the public at red-lobster.dev | 1. Open new browser, and connect to domain | 1. CrabbyTasks Website displays Landing page | 1. Arrived to Landing page | Yes | 11/11/2023 | Tested on Safari | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |
| | | | | | | | | | |
| **2.1.9.1** | **0** | System will have its own Domain Name and be placed on public DNS. | 1. DNS Lookup red-lobster.dev from foreign device | 1. respond with elastic AWS IP | 1. IP address matches AWS information | Yes | 11/11/2023 | nslookup | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |
| | | | | | | | | | |
| **2.1.10** | **0** | Production Server and Databases will be hosted on Virtual Amazon EC2 Instances running Ubuntu 22.04 LTS (HVM) SSD Volume Drive on 64-bit Image | 1. Connect to AWS console. Lookup IP in Amazon Route 53 DNS | 1. response with the shared Hosted Virtual private cloud shared between them and hardware specifications. | 1. Server information displayed to user. | Yes | 11/11/2023 | Toolset: AWS console | Reviewed YES by Nick Dilallo, Testing Manager, 11/11/2023 |

## Test Code

base/tests.py

The 'tests.py' file contains Django test cases for the 'base' app, focusing on the 'task_model' model. The 'Test_Create_Post' class sets up test data, including a user and a task with specific attributes. The 'test_task_content' method then verifies that the task object's attributes match the expected values. This ensures the correct functioning of the 'task_model' model by testing attribute assignments during object creation.

```
################################################################################
# This 'tests.py' file contains test cases for the 'base' app of the Django project.
# It includes a test class 'Test_Create_Post' that defines setup data and tests
# for the 'task_model' model. The tests check if the task object's attributes
# match the expected values after creation.
################################################################################

from django.test import TestCase
from django.contrib.auth.models import User
from .models import task_model

# Test class for creating and testing task objects.
class Test_Create_Post(TestCase):

     # Set up test data before running the tests.
    @classmethod
    def test_setUpTestData(cls):
        test_user1= User.objects.create_user(username="test_user1",password="123456789")
        test_task=task_model.objects.create(user_id=1,
                                            status='To Do',
                                            task="test",
```

```
                                                description="TEST",
                                                deadline="1/1/2024",
                                                priority=2)
    # Test if the task object's attributes match expected values.
    def test_task_content(self):
        task = task_model.todo_tasks_objects.get(task="test");
        user        =f'{task.user_id}'
        task_title  =f'{task.task}'
        status      =f'{task.status}'
        description =f'{task.description}'
        deadline    =f'{task.deadline}'
        priority    =f'{task.priority}'

        # Assert that the attributes match the expected values.
        self.assertEqual(user,'test_user1')
        self.assertEqual(task_title,'test')
        self.assertEqual(status,'To Do')
        self.assertEqual(description,'TEST')
        self.assertEqual(deadline,'1/1/2024')
        self.assertEqual(priority,'High')
        self.assertEqual(str(task),'test')
```

## api/tests.py

The 'tests.py' file contains API view test cases for the 'base' app's tasks using Django's test framework. The 'TaskTests' class includes a method, 'test_view_posts,' which defines a URL for the API endpoint and sends a GET request. The test checks if the response status code is HTTP 200 OK, ensuring the correct functionality of the task-related API views.

```
################################################################################
# This 'tests.py' file defines test cases for the API views related to tasks in
```

```python
# the 'base' app. It uses Django's test framework to test the API endpoints.
##############################################################################

from django.urls import reverse
from rest_framework import status
from rest_framework.test import APITestCase
from ..models import task_model
from django.contrib.auth.models import User


# Test case class for API views related to tasks.
# My first test :,) Careless 231019 23:02
class TaskTests(APITestCase):
    def test_view_posts(self):
        # Define the URL for the API endpoint.
        url=reverse('api:listcreate')

        # Send a GET request to the API endpoint.
        response = self.client.get(url,format='json')

        # Check if the response status code is HTTP 200 OK.
        self.assertEqual(response.status_code, status.HTTP_200_OK)
```

# Test Results

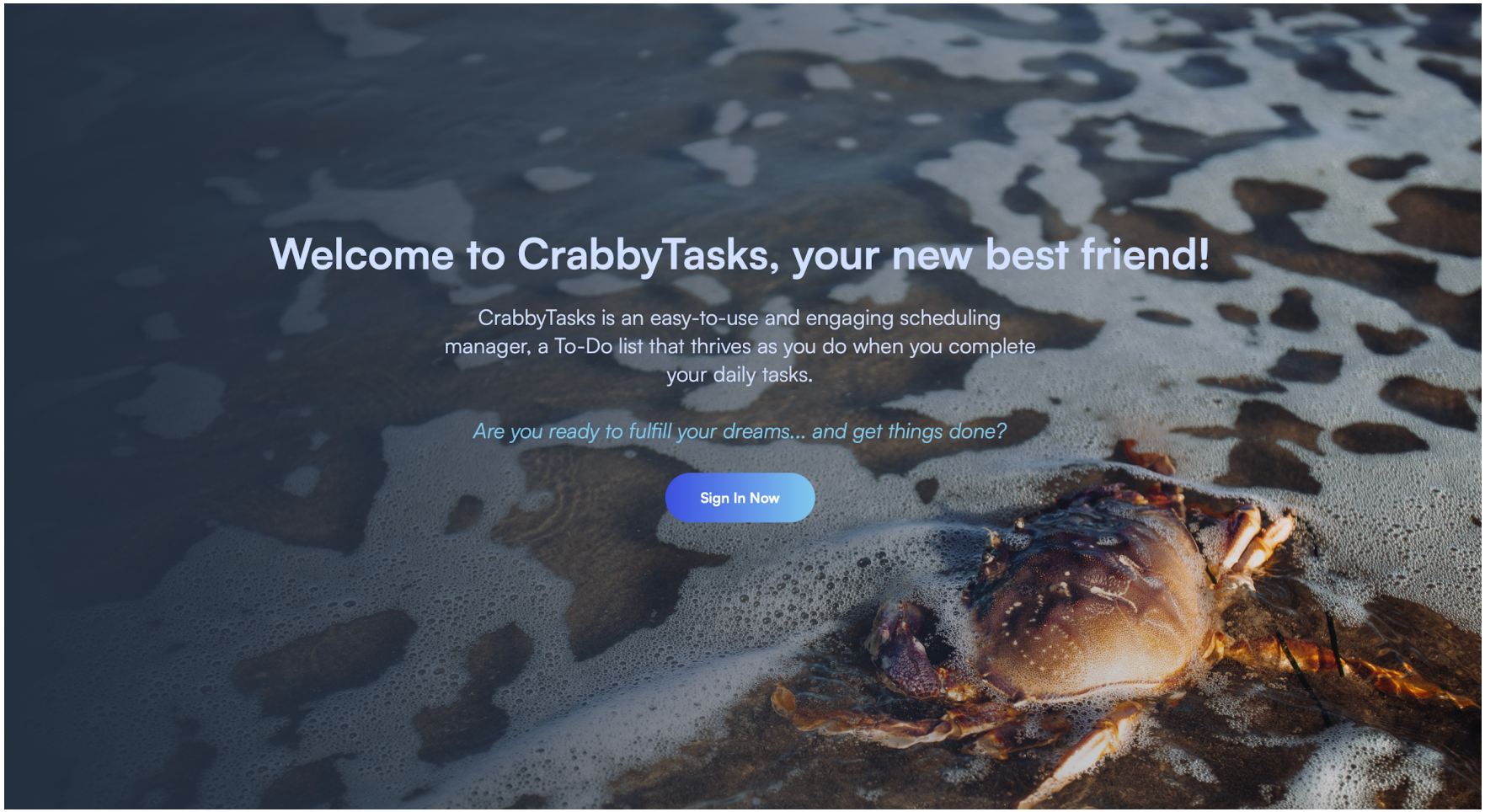
## Coverage report: 30%

*coverage.py v7.3.2, created at 2023-11-11 14:50 -0600*

| *Module* | *statements* | *missing* | *excluded* | *coverage ↑* |
|---|---|---|---|---|
| base/apps.py | 4 | 4 | 0 | 0% |
| base/tests.py | 23 | 22 | 0 | 4% |
| base/api/views.py | 78 | 69 | 0 | 12% |
| manage.py | 15 | 12 | 0 | 20% |
| base/api/serializer.py | 26 | 20 | 0 | 23% |
| base/admin.py | 8 | 6 | 0 | 25% |
| base/api/urls.py | 5 | 3 | 0 | 40% |
| config/settings.py | 42 | 22 | 0 | 48% |
| config/urls.py | 3 | 1 | 0 | 67% |
| base/models.py | 29 | 8 | 0 | 72% |
| base/urls.py | 4 | 0 | 0 | 100% |
| base/__init__.py | 0 | 0 | 0 | 100% |
| base/api/__init__.py | 0 | 0 | 0 | 100% |
| base/migrations/__init__.py | 0 | 0 | 0 | 100% |
| base/views.py | 0 | 0 | 0 | 100% |
| config/__init__.py | 0 | 0 | 0 | 100% |
| **Total** | **237** | **167** | **0** | **30%** |

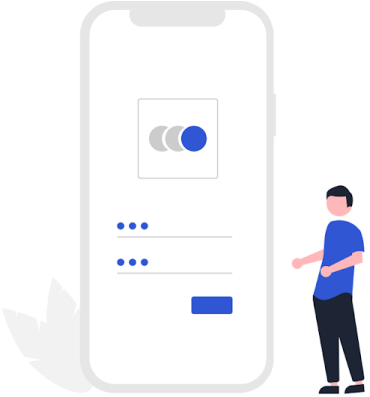
*coverage.py v7.3.2, created at 2023-11-11 14:50 -0600*

## Website Screenshots


Welcome to CrabbyTasks, your new best friend!
CrabbyTasks is an easy-to-use and engaging scheduling manager, a To-Do list that thrives as you do when you complete your daily tasks.
Are you ready to fulfill your dreams... and get things done?
Sign In Now

# CrabbyTasks

Much like the crab, the human journey involves shedding old shells and embracing change, for growth requires the courage to leave behind the safety of the familiar.



Start for free

## Sign Up to CrabbyTasks

Name

Enter your full name

Email

Enter your email

Password

Enter your password

Re-type Password

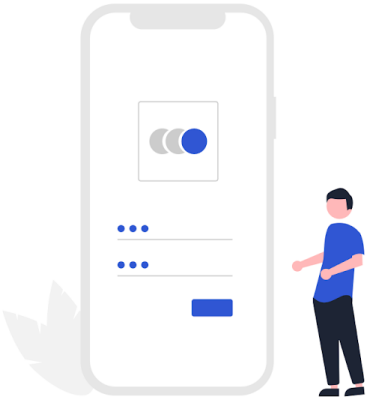
Re-enter your password

Create account

Already have an account? Sign in

# CrabbyTasks

*While the crab's sideways movement may appear aimless, it symbolizes life's unpredictability, reminding us that the path to truth is often found in unexpected directions.*



Start for free

## Sign In to CrabbyTasks!

Email

tapiacarlos32@gmail.com

Password

••••••••

Sign In

Don't have any account? Sign Up

CrabbyTasks

MENU

- Dashboard
  - Home
- Task Manager
- Contact Us
- Settings

Carlos
tapiacarlos32@gmail.com

| | |
|---|---|
| 0 | Task Completion Score |
| 0 | Tasks In-Progress |
| 2 | Tasks To-Do |
| 0 | Completed Tasks |
| 0 | Overdue Tasks |

| Task | Due Date | Actions |
|---|---|---|
| add a task<br>Oh YES | Sun Nov 05, 2023 | |
| another one<br>another task! | Sun Nov 05, 2023 | |

Add Task

Task

Description

Description

Status

To Do

Priority

Highest

Notifications

None

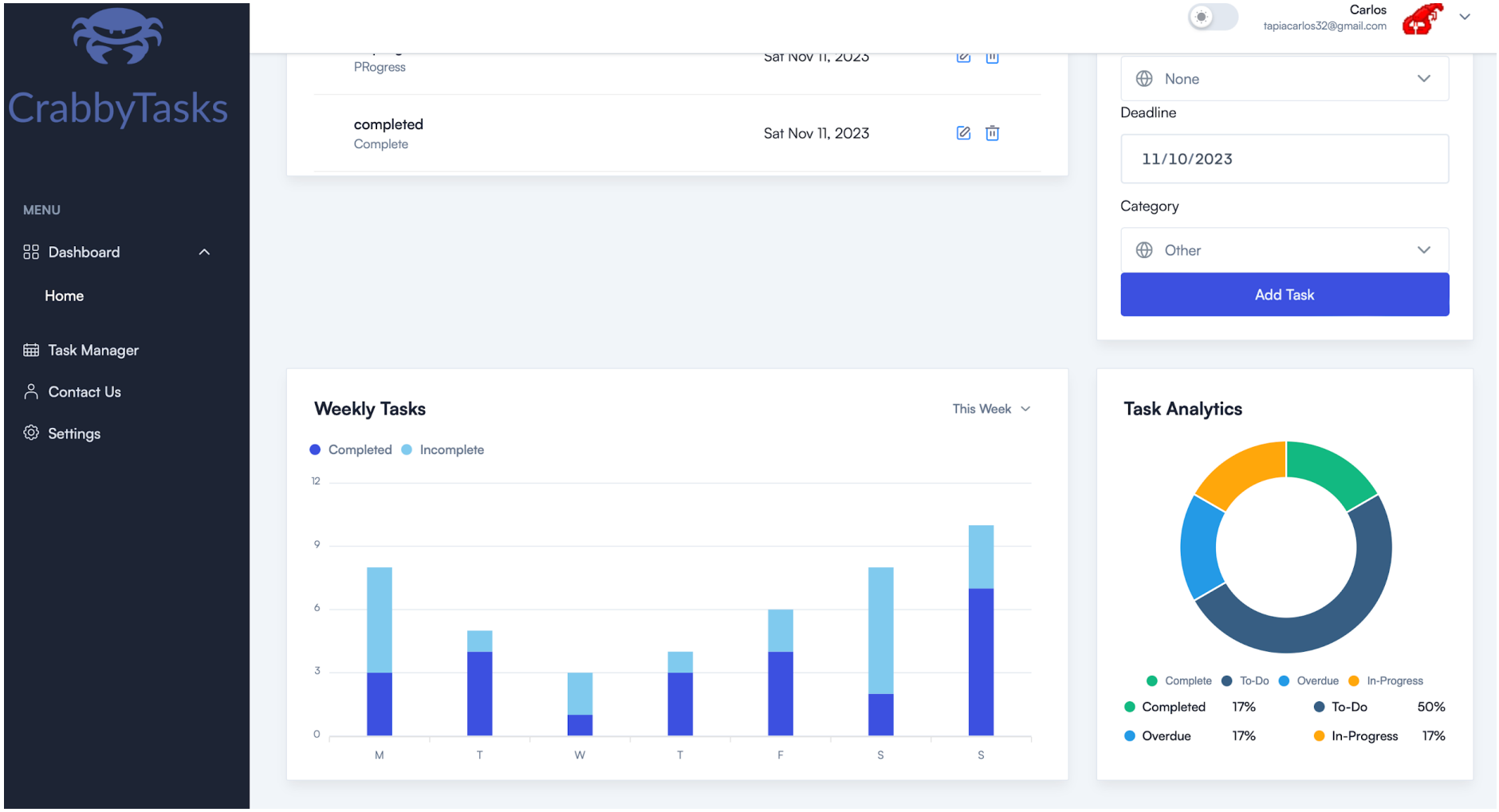
Deadline

11/10/2023

Category

Other

Add Task


CrabbyTasks
MENU
Dashboard
Home
Task Manager
Contact Us
Settings
Carlos
tapiacarlos32@gmail.com
PRogress
Sat Nov 11, 2023
completed
Complete
Sat Nov 11, 2023
None
Deadline
11/10/2023
Category
Other
Add Task
Weekly Tasks
This Week
Completed
Incomplete
12
9
6
3
0
M
T
W
T
F
S
S
Task Analytics
Complete
To-Do
Overdue
In-Progress
Completed 17%
To-Do 50%
Overdue 17%
In-Progress 17%

CrabbyTasks

MENU

- Dashboard
  - Home
- Task Manager
- Contact Us
- Settings

Type to search...

Carlos
tapiacarlos32@gmail.com

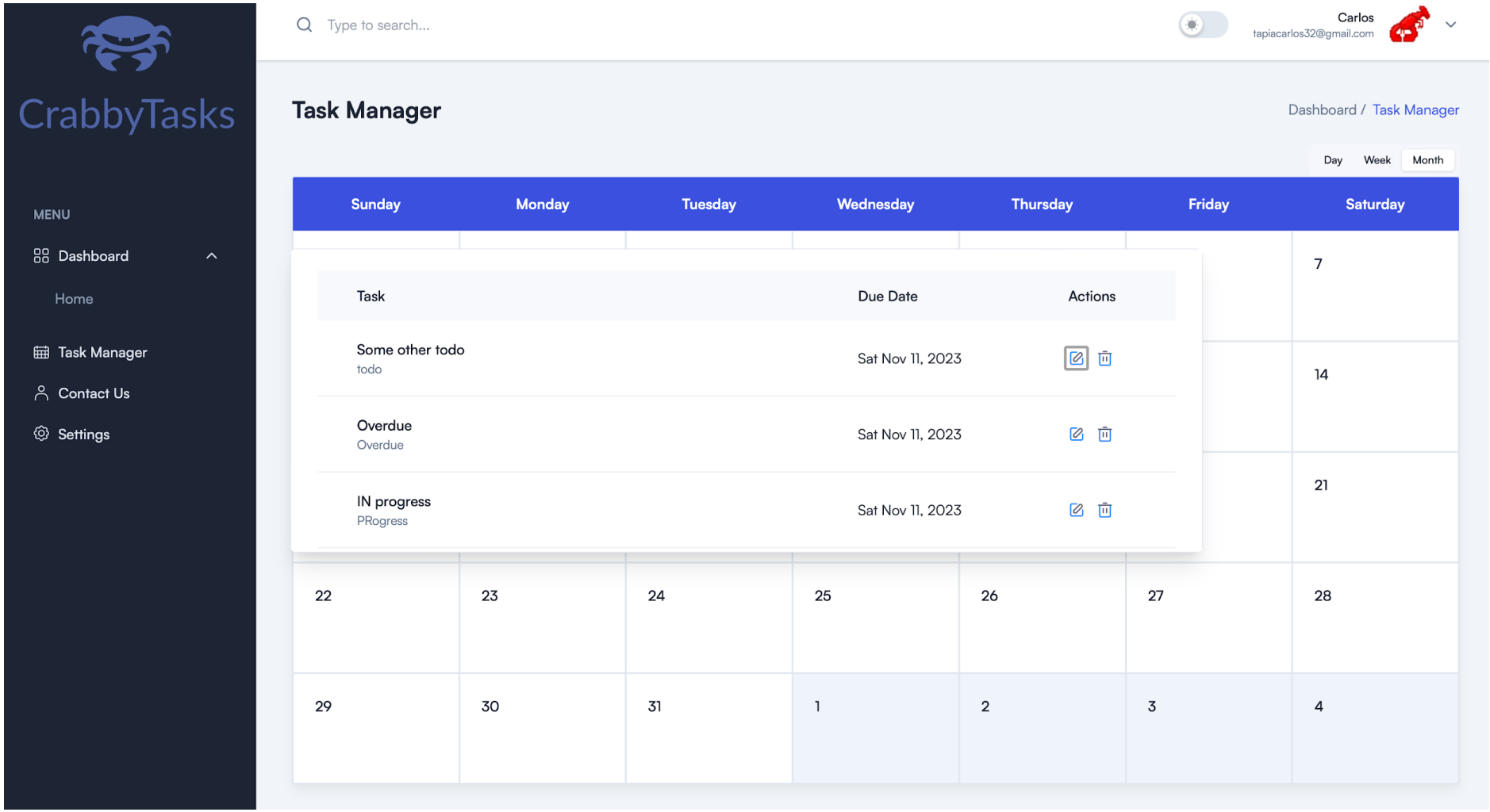
# Task Manager

Dashboard / Task Manager

Day Week Month

| Sunday | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | 31 | 1 | 2 | 3 | 4 |

Open "localhost:5173" in a new tab


CrabbyTasks
MENU
Dashboard
Home
Task Manager
Contact Us
Settings
Type to search...
Carlos
tapiacarlos32@gmail.com
Task Manager
Dashboard / Task Manager
Day
Week
Month
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Task
Due Date
Actions
Some other todo
todo
Sat Nov 11, 2023
Overdue
Overdue
Sat Nov 11, 2023
IN progress
PRogress
Sat Nov 11, 2023
7
14
21
22
23
24
25
26
27
28
29
30
31
1
2
3
4

CrabbyTasks

MENU

- Dashboard
  - Home
- Task Manager
- Contact Us
- Settings

Type to search...

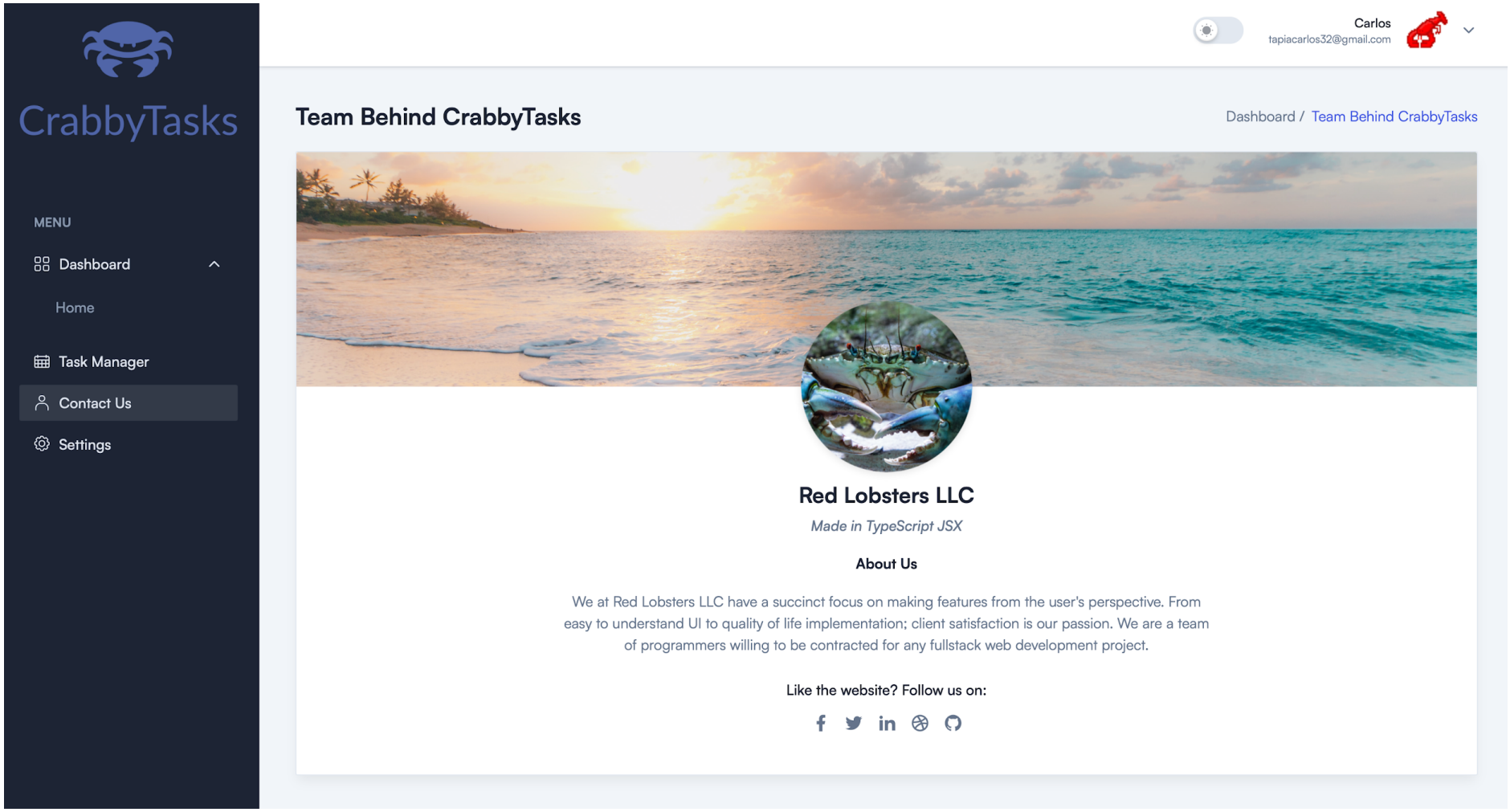
Carlos
tapiacarlos32@gmail.com

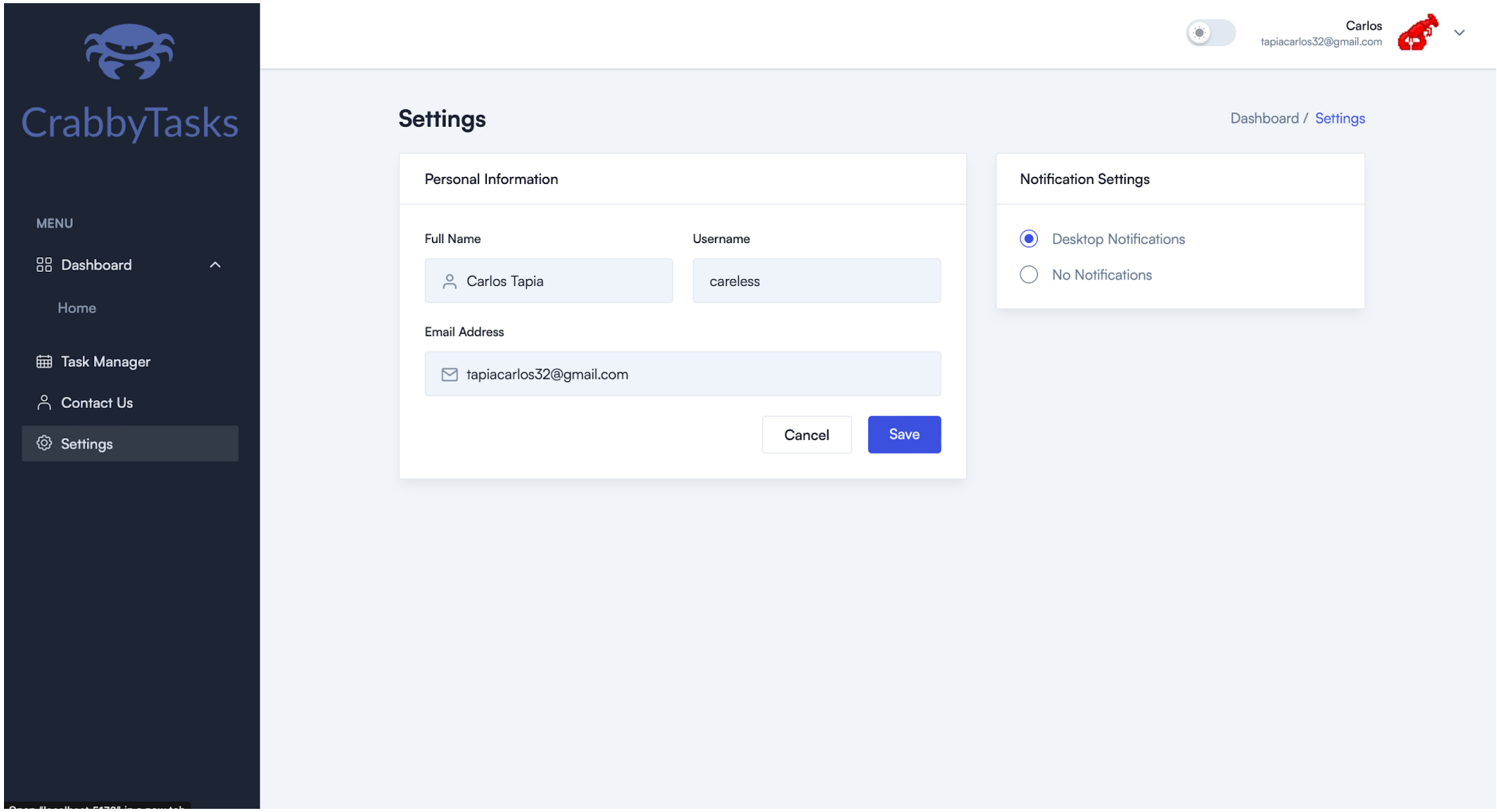
# Task Manager

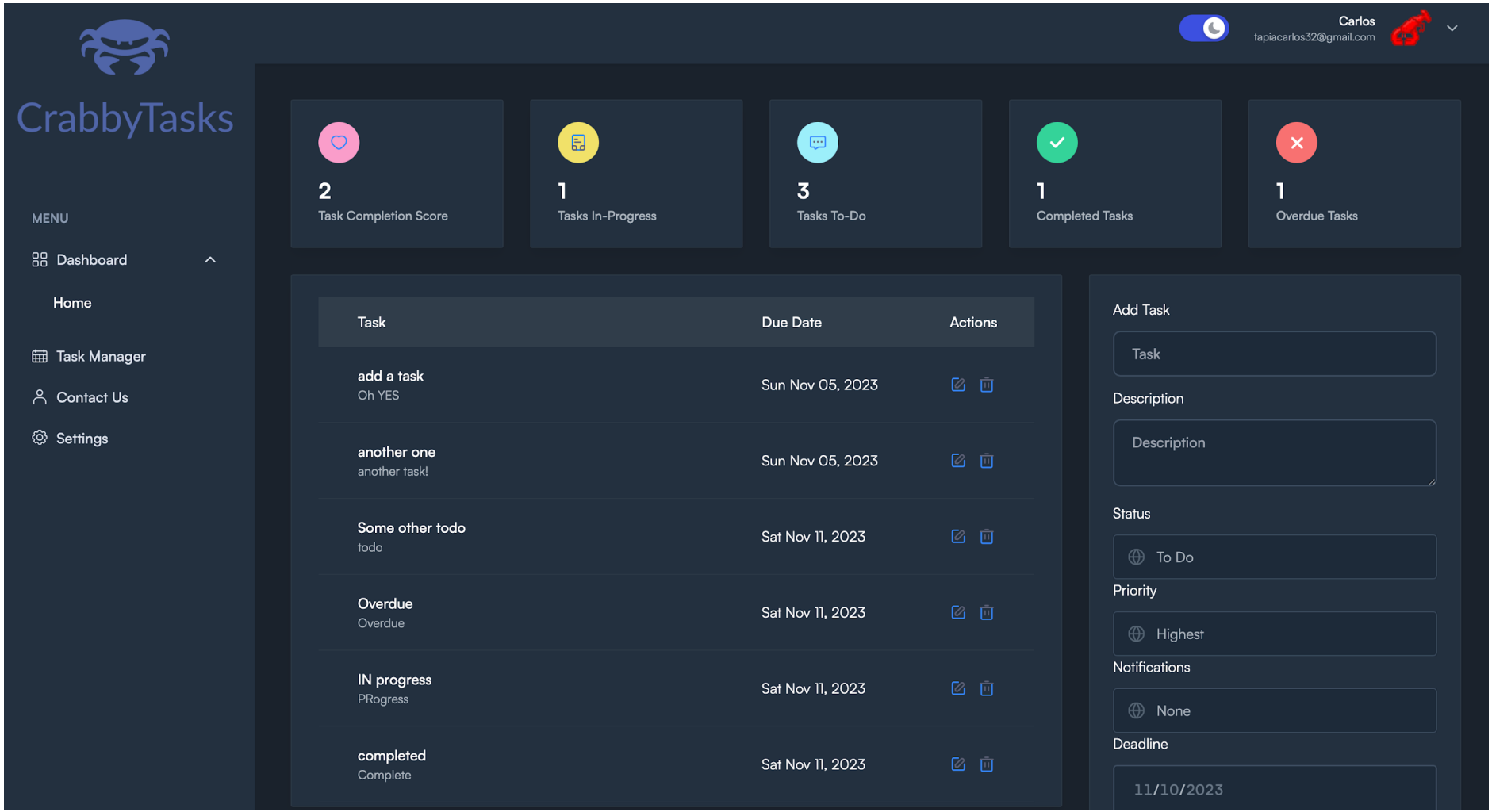
Dashboard / Task Manager

Day Week Month

| Sun | Mon | Tue | Wed | Thur | Fri | Sat |
|---|---|---|---|---|---|---|
| 29 | 30 | 31 | 1 | 2 | 3 | 4 |


CrabbyTasks
MENU
Dashboard
Home
Task Manager
Contact Us
Settings
Carlos
tapiacarlos32@gmail.com
Team Behind CrabbyTasks
Dashboard / Team Behind CrabbyTasks
Red Lobsters LLC
Made in TypeScript JSX
About Us
We at Red Lobsters LLC have a succinct focus on making features from the user's perspective. From easy to understand UI to quality of life implementation; client satisfaction is our passion. We are a team of programmers willing to be contracted for any fullstack web development project.
Like the website? Follow us on:


CrabbyTasks
MENU
Dashboard
Home
Task Manager
Contact Us
Settings
Carlos
tapiacarlos32@gmail.com
Settings
Dashboard / Settings
Personal Information
Full Name
Carlos Tapia
Username
careless
Email Address
tapiacarlos32@gmail.com
Cancel
Save
Notification Settings
Desktop Notifications
No Notifications


CrabbyTasks
Carlos
tapiacarlos32@gmail.com
MENU
Dashboard
Home
Task Manager
Contact Us
Settings
2
Task Completion Score
1
Tasks In-Progress
3
Tasks To-Do
1
Completed Tasks
1
Overdue Tasks
Task
Due Date
Actions
add a task
Oh YES
Sun Nov 05, 2023
another one
another task!
Sun Nov 05, 2023
Some other todo
todo
Sat Nov 11, 2023
Overdue
Overdue
Sat Nov 11, 2023
IN progress
PRogress
Sat Nov 11, 2023
completed
Complete
Sat Nov 11, 2023
Add Task
Task
Description
Description
Status
To Do
Priority
Highest
Notifications
None
Deadline
11/10/2023

# User Manual

## Welcome to CrabbyTask’s User Manual

Welcome to CrabbyTasks, your go-to task management application! This user manual is your guide to making the most of CrabbyTasks and ensuring a smooth and productive experience. Whether you're new to task management apps or a seasoned user, this manual will walk you through the essential features and functionalities.

## Getting Started

Before we dive into the details, let's get acquainted with the key components of CrabbyTasks:

**The Landing Page:** Your journey begins here, where you can easily access the Sign In button to get started.

**The Sign In Page:** Once you've clicked the Sign In button, you'll arrive at the Sign In page. This page allows you to log in and access your tasks.

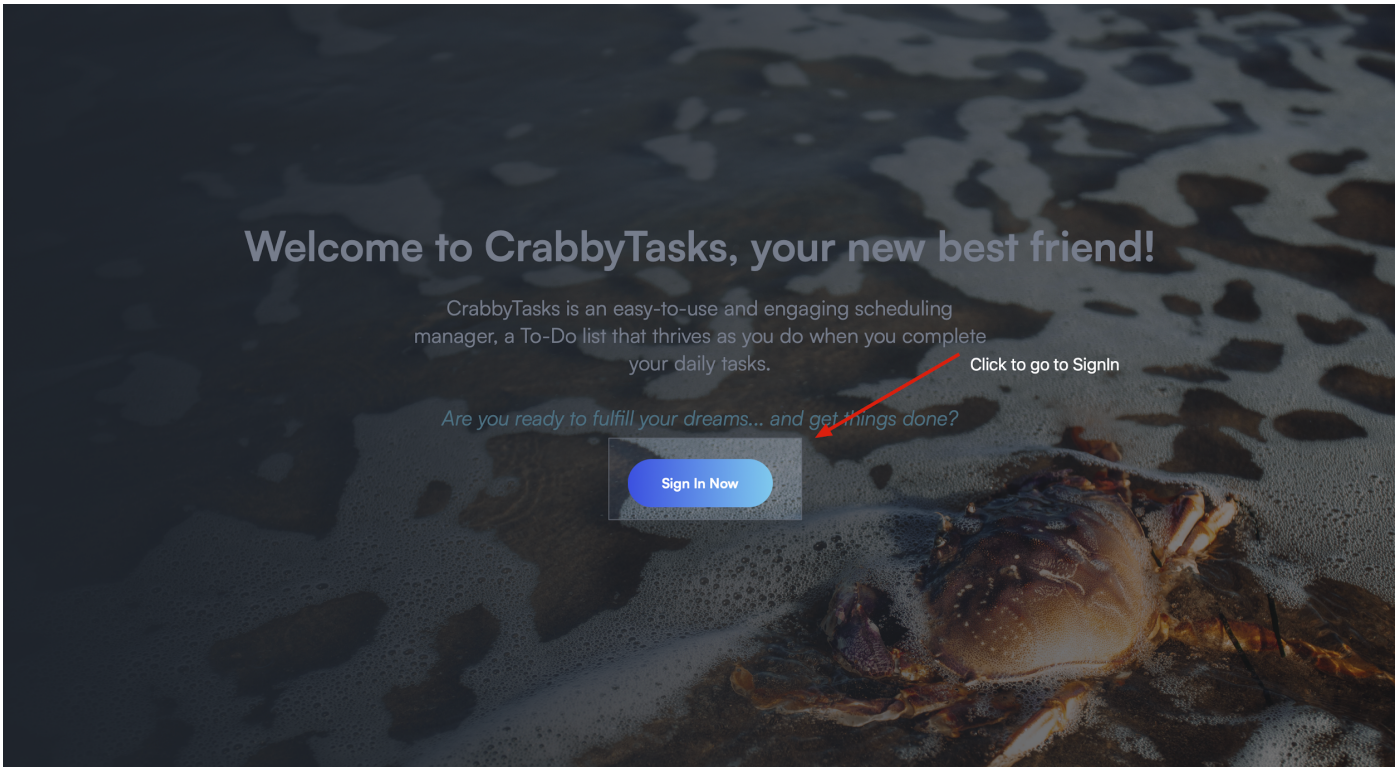
**The Home Page:** The heart of CrabbyTasks, where you'll manage your tasks and stay organized.

**The Task Manager:** This feature lets you view tasks in different timeframes, from daily to monthly, and manage them efficiently.

**The Settings:** Customize your CrabbyTasks experience by adjusting your personal settings, toggling between light and dark modes, and more.

Now, let's explore each of these components in more detail with step-by-step instructions and visual aids.

## Landing Page

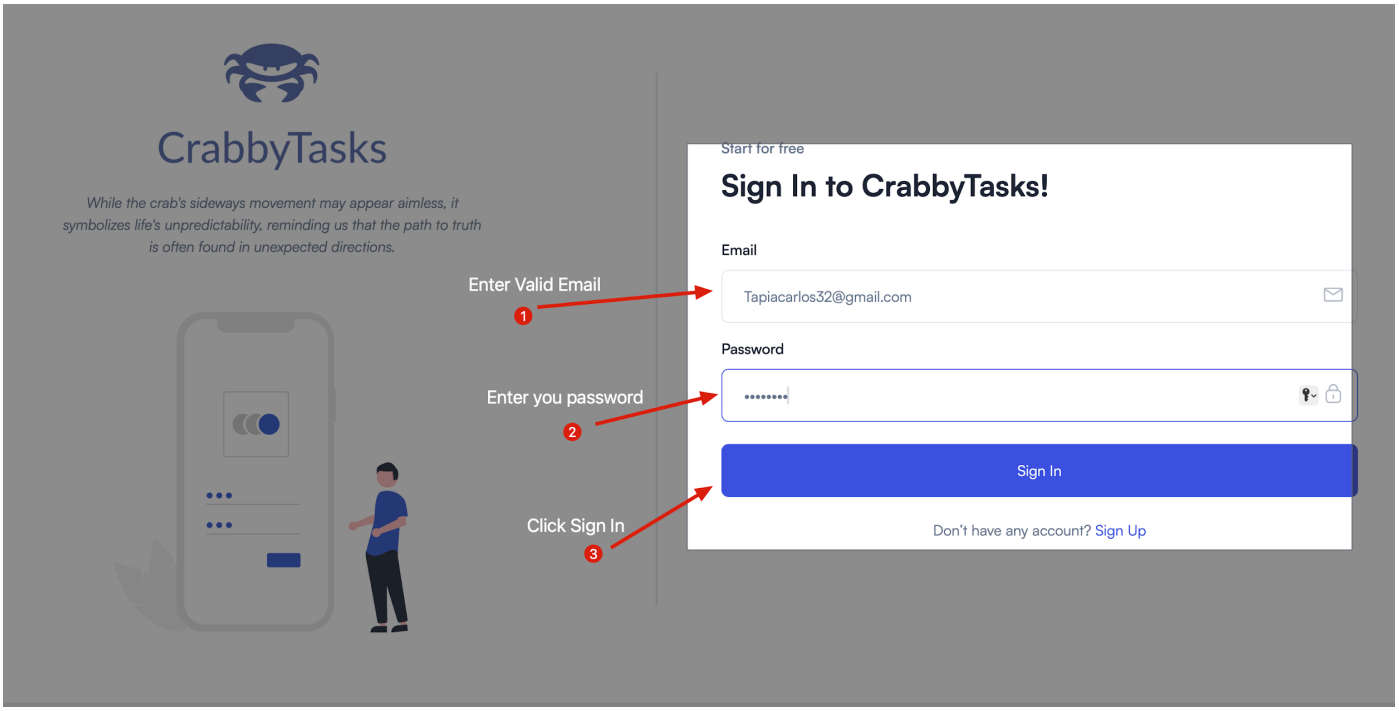



The Landing Page is your entry point to CrabbyTasks. It's designed to provide you with a straightforward and user-friendly experience. When you arrive at the Landing Page, you'll notice a prominent "Sign In" button positioned prominently in the middle of the screen.

To get started with CrabbyTasks, simply click on the "Sign In" button.

This button is your gateway to accessing your task management dashboard, where you can view, create, and manage your tasks efficiently. Whether you're a new user or returning visitor, the Landing Page ensures a hassle-free start to your task management journey.

## Sign In Page




The Sign In Page is where you can access your CrabbyTasks account by providing your credentials. This page is designed with simplicity and security in mind.

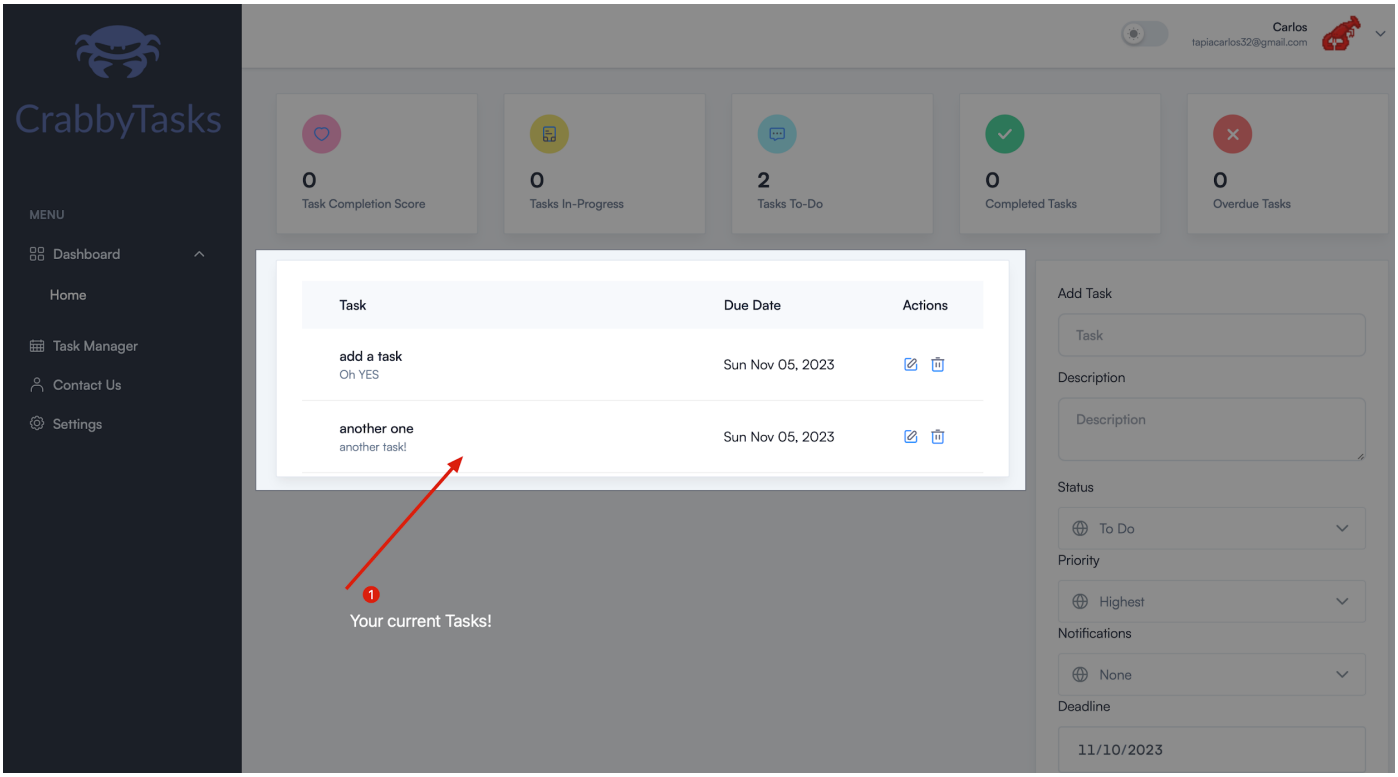
On the Sign In Page, you'll see a sign-in form with two fields: one for your email and another for your password.

- The arrow pointing to the email field indicates where to enter your valid email address.
- The arrow pointing to the password field indicates where to enter your password.

After filling in your credentials, click on the "Sign In" button to access your account.

The Sign In Page ensures that your account remains protected while providing easy access to your task management tools.

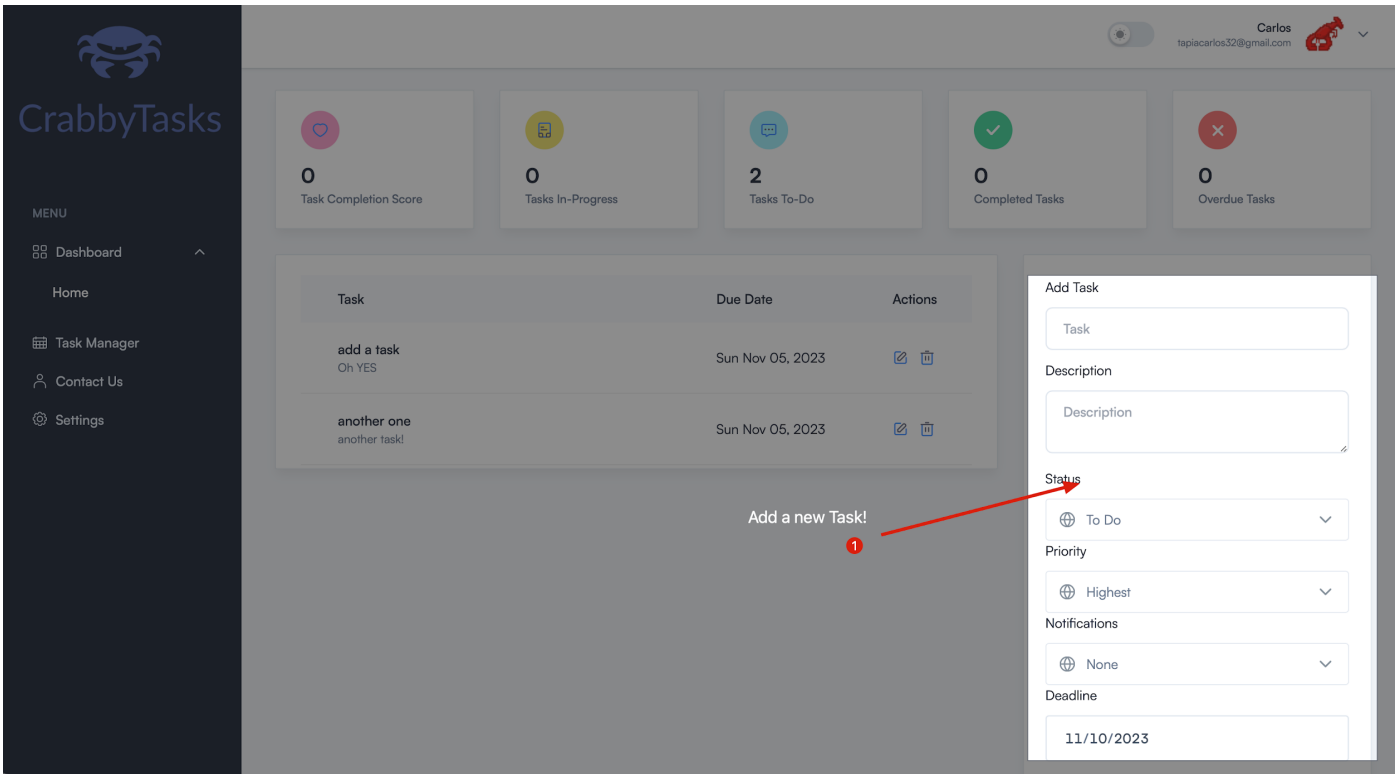
## Home Page




The Home Page is your central hub for managing your tasks efficiently. One of its key features is the Tasklist, which displays your current tasks.

- On the Home Page, you'll find the Tasklist prominently displayed, showing all your current tasks.
- The Tasklist allows you to view your tasks at a glance and helps you stay organized.

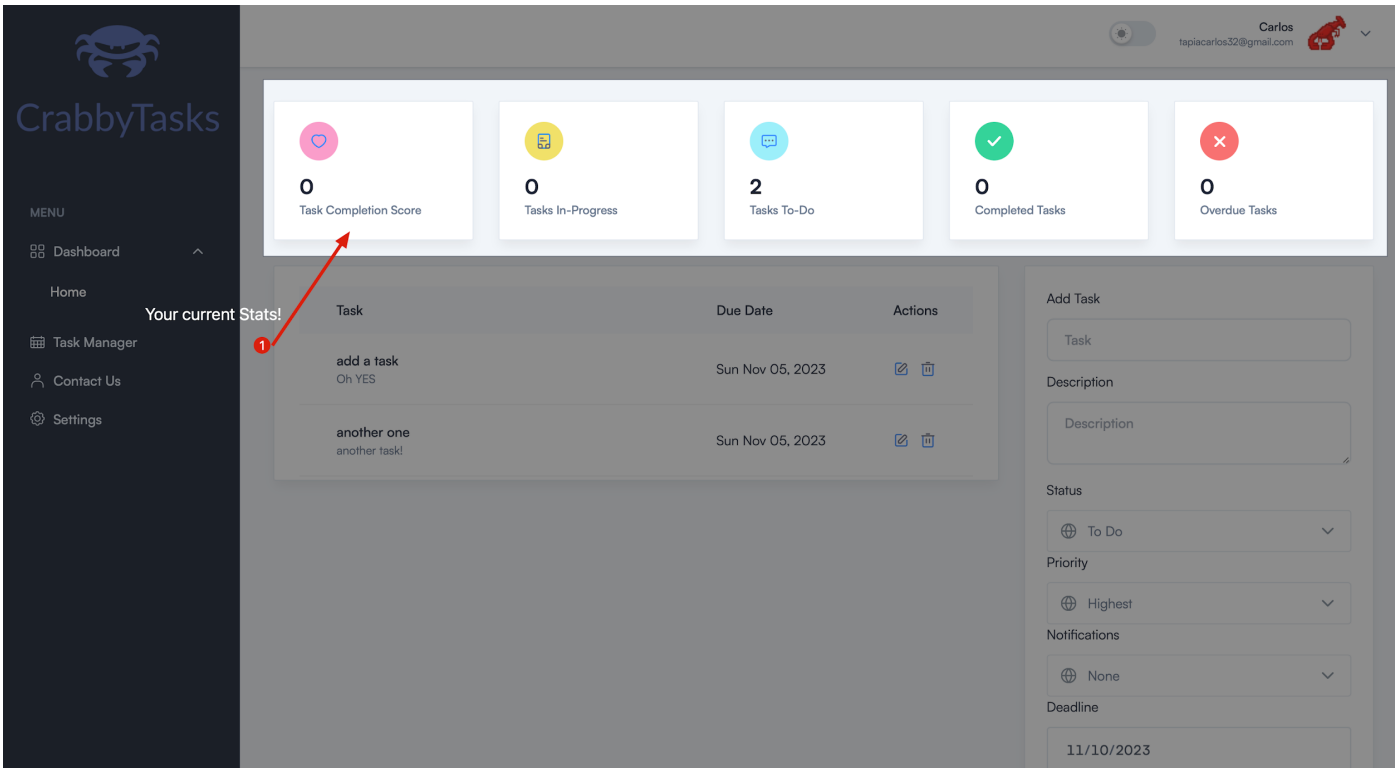
The Tasklist simplifies task management by providing a clear overview of your responsibilities.




The Home Page provides you with a detailed view of your tasks through the Add-A-Task. This feature allows you to create tasks with ease.

- Locate the TaskLarge component on the Home Page, which is designed for creating new tasks.
- Click the "Add a new task" button within the TaskLarge component to create a new task.
- You can specify task details such as the title, description, due date, and priority.

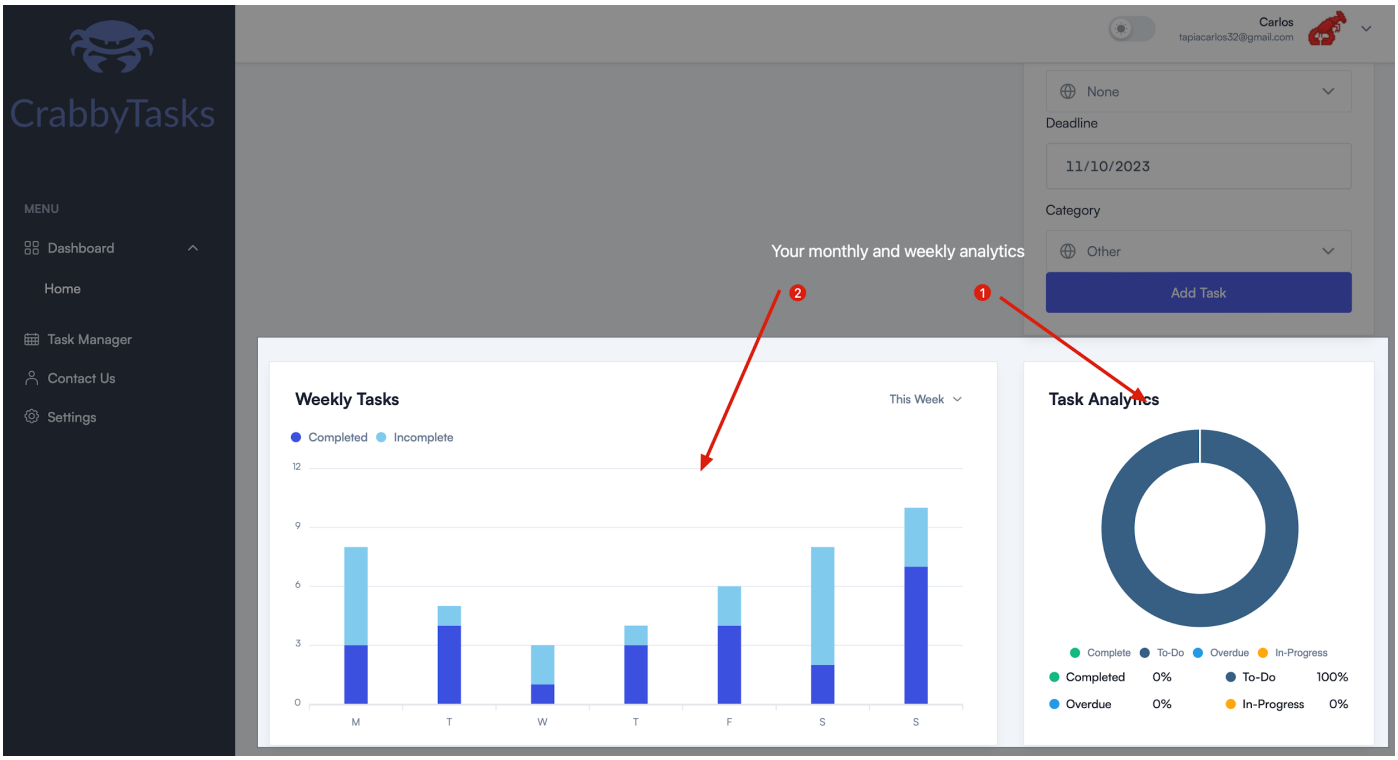
The TaskLarge component simplifies task creation and ensures you never miss an important task again.




The Home Page offers a quick overview of your task statistics through Task Statistic cards. These cards provide valuable insights into your task management progress.

- Locate the Task Statistic cards on the Home Page, which display your current task statistics.
- The cards include information such as the total number of tasks, completed tasks, and overdue tasks.
- Use these statistics to gauge your task management performance and track your progress.

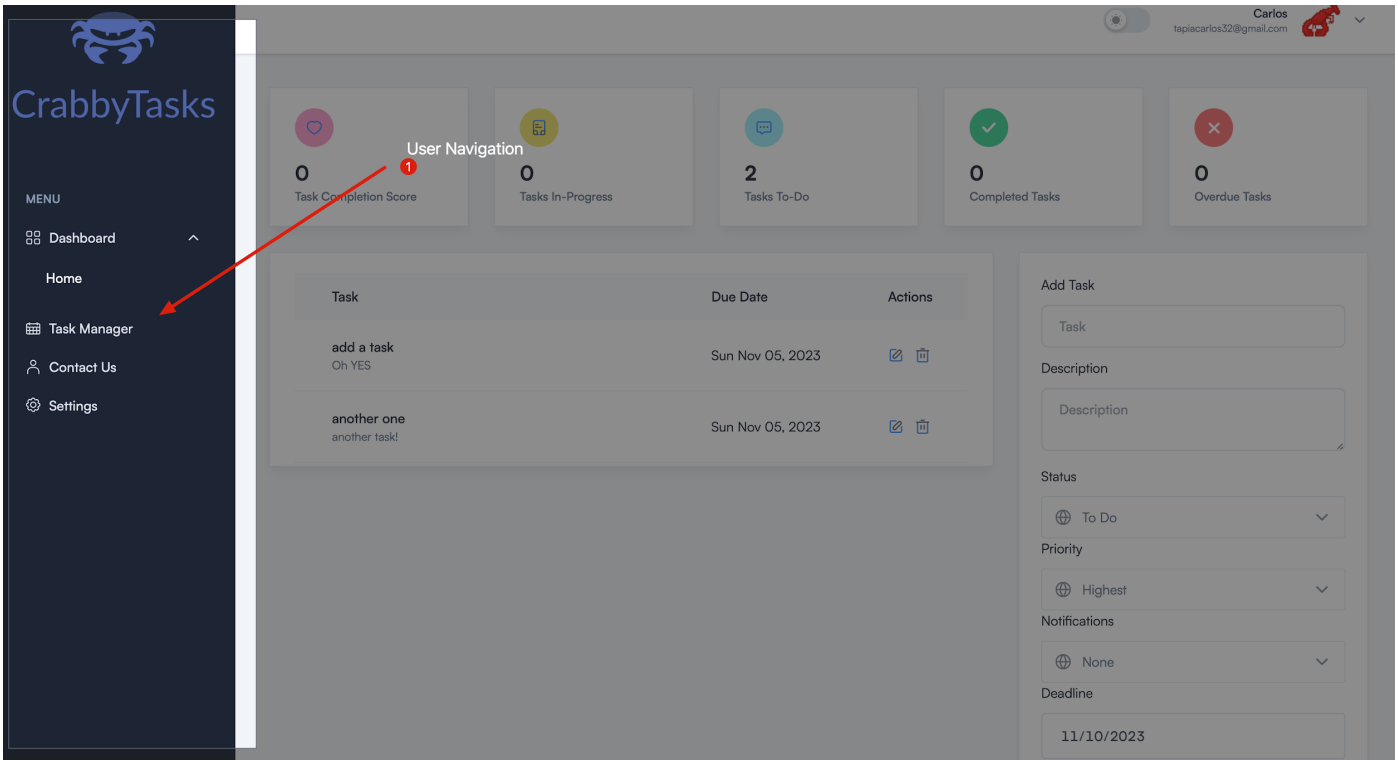
The Task Statistic cards are a handy tool for staying organized and informed about your tasks.




The Home Page features insightful Task Analytics Charts, including weekly and monthly analytics, to help you visualize your monthly and weekly task performance.

- Locate the Task Analytics Charts on the Home Page, which provide graphical representations of your task data.
- TaskAnalytics displays your monthly task statistics, giving you a broader overview of your task progress over time.
- Weekly Chart presents your weekly task analytics, allowing you to dive into your weekly task achievements.

These charts are interactive and user-friendly, enabling you to make data-driven decisions and optimize your task management strategy.

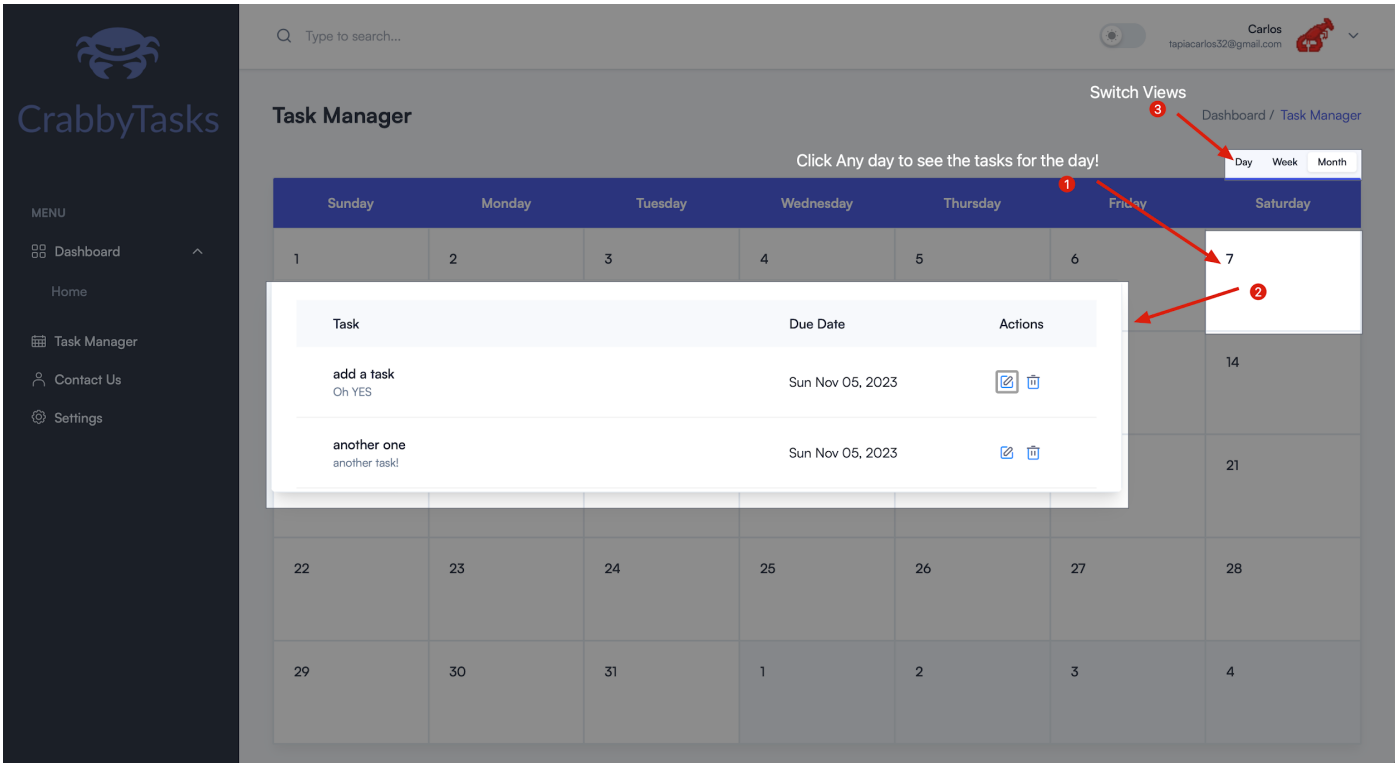



The Home Page features a convenient User Navigation Sidebar that provides easy access to essential functions.

- Locate the User Navigation Sidebar on the left side of the Home Page.
- This sidebar contains quick links to various sections of the application, making it effortless to navigate.
- Use the sidebar to access your tasks, settings, and other essential features with a single click.

The User Navigation Sidebar streamlines your experience, ensuring you can efficiently go to your tasks and preferences.

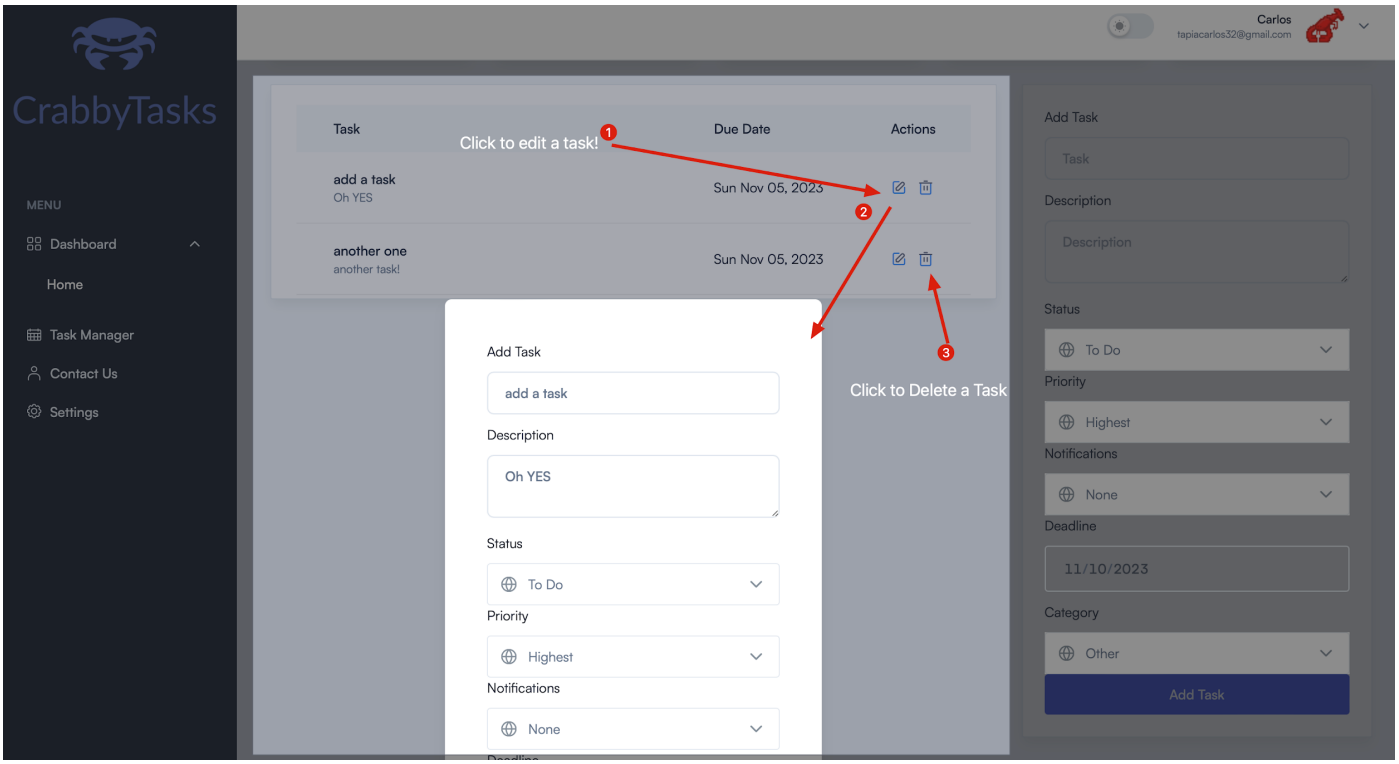
## Task Manager Page




The Task Manager Page empowers you to efficiently manage your tasks and stay organized.

- Navigate to the Task Manager Page using the sidebar or other navigation options.
- Click on a calendar cell to access tasks for a particular day.
- Use the task list popup to manage and update your tasks.
- Experiment with different calendar views using the View Switcher to find the one that fits your workflow.

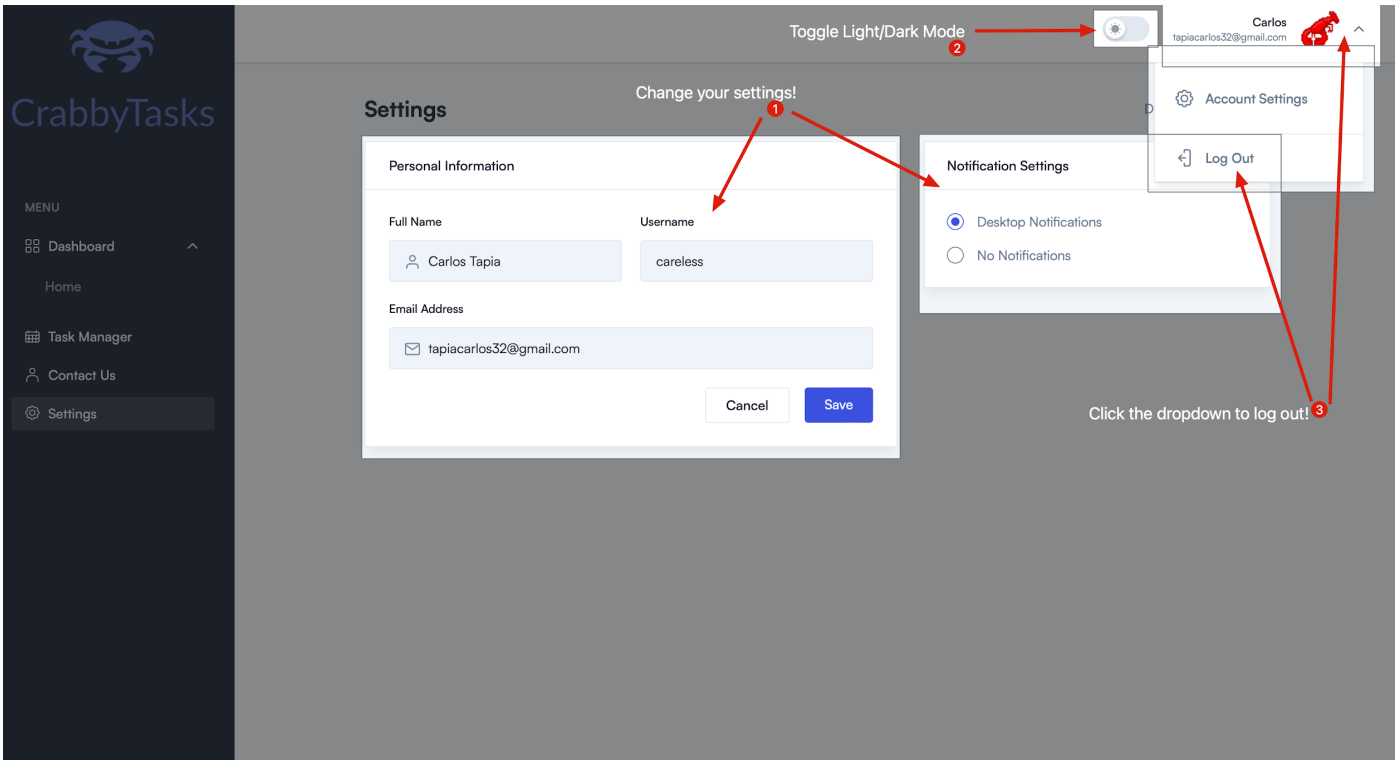
With the Task Manager Page, you have the tools to effectively organize and track your tasks, ensuring you stay on top of your schedule.




After clicking the "Edit" button on a task within the TaskChart component, a task modal pops up. This modal allows you to make edits to task details, such as the title, due date, and status. Additionally, you can delete tasks directly from this modal.

- Locate a task on the Task that you want to edit.
- Click the "Edit" button associated with the task.
- The Task Modal will appear, allowing you to update task information or delete the task if needed.

With the Task Modal, you have a quick and convenient way to manage your tasks, ensuring they are up to date and aligned with your goals.




The Settings Page is where you can personalize your CrabbyTasks experience and manage your user profile. This screenshot highlights several key elements:

- To enable Dark Mode, toggle the switch to the right. To switch back to Light Mode, toggle it to the left.
- Click on the fields under "User Settings" to update your personal information as needed.
- Adjust your notification preferences by selecting your desired options.
- Click the user dropdown menu in the upper-right corner and select "Log Out" to securely log out of your account.

Customize your CrabbyTasks experience, stay informed with notifications, and easily access your user profile settings on the Settings Page.

# Technical Manual

## Welcome to CrabbyTask’s Technical Manual

The Technical Manual for CrabbyTasks serves as a comprehensive guide for developers and administrators who are involved in the setup, maintenance, and security of the CrabbyTasks application. This manual provides insights into the technical aspects of the project, helping you understand the development environment, security measures, deployment procedures, troubleshooting, and maintenance.

Why Understanding the Technical Aspects Is Important

Understanding the technical aspects of CrabbyTasks is crucial for several reasons:

- Efficient Development: Developers can work more efficiently when they have a clear understanding of the project's tools.
- Security: Knowledge of security measures is essential to protect user data and prevent potential vulnerabilities.
- Deployment: Knowing how to deploy the application to production ensures a smooth user experience.
- Maintenance: Proper maintenance keeps the application up-to-date and minimizes downtime.

In this section, we provide an overview of the Technical Manual's purpose and highlight the importance of diving into the technical details of CrabbyTasks.

## Development Environment Setup

Deploying CrabbyTasks to a production environment is a crucial step in making the application accessible to users. This section outlines the deployment process, including the required infrastructure and configurations.

Before deployment,  ensure you have the following environment prepared:

Server: You'll need a server to host the application. Popular choices include cloud providers like AWS, Azure, or a VPS (Virtual Private Server).

Web Server: You'll need a web server to handle https requests to the application. Popular choices are Apache2, and Nginx. Due to the nature of Django, Apache is preferred. Ensure to have the following configurations set to allow for a correct production Django application.

```
sudo apt-get update
sudo apt-get install python3-pip apache2 libapache2-mod-wsgi-py3
```

Once installed, configure the following site through apache:

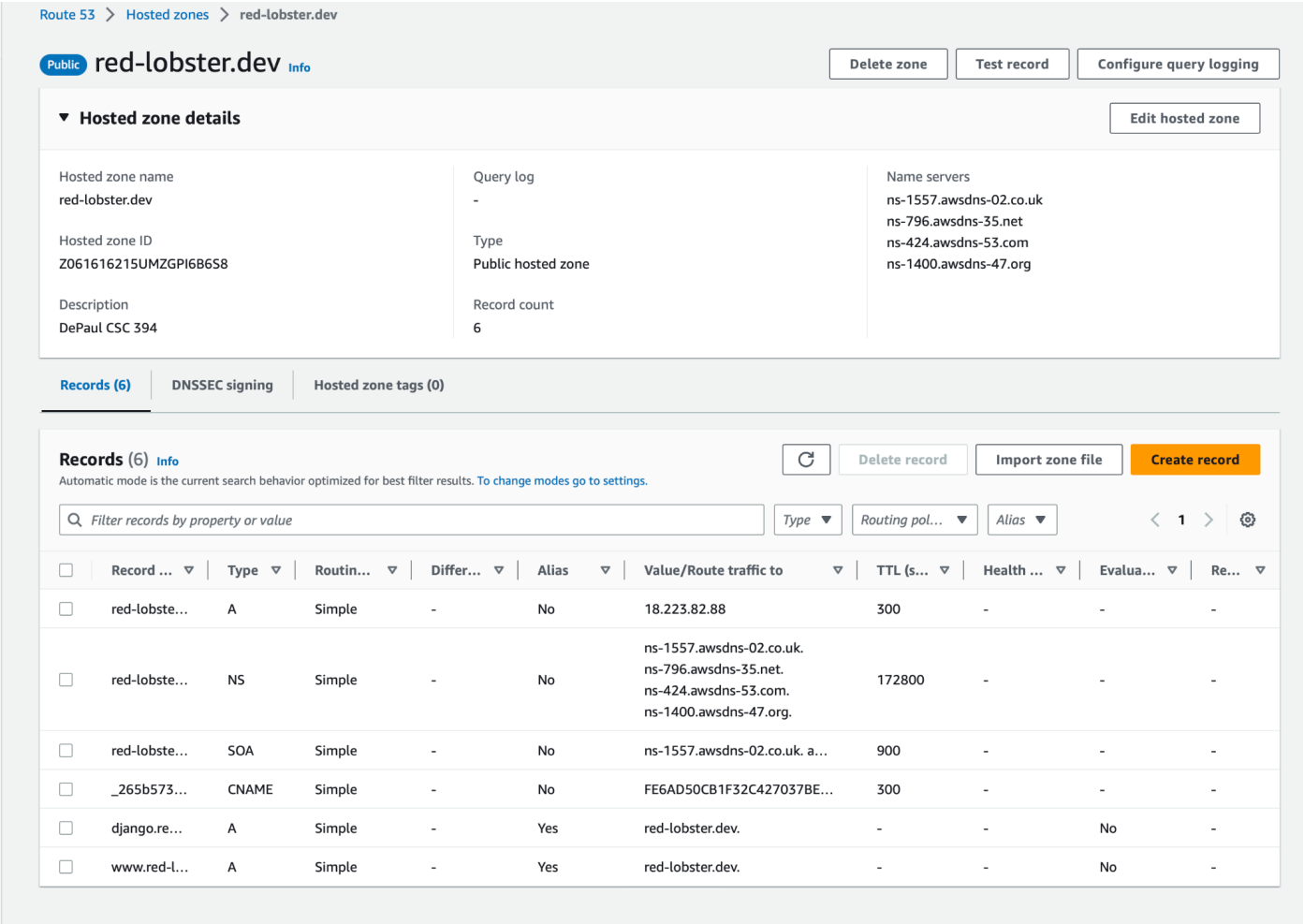
```
vim /etc/apache2/sites-available/000-default.conf
```

```
ServerName red-lobster.dev

<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
                ServerAdmin webmaster@localhost
```

```
                DocumentRoot /var/www/html

                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined

                SSLEngine on

                AddType text/css .css
                AddType application/javascript .js
                <FilesMatch "\.(cgi|shtml|phtml|php)$">
                                SSLOptions +StdEnvVars
                </FilesMatch>

                <Directory /usr/lib/cgi-bin>
                                SSLOptions +StdEnvVars
                </Directory>

                Alias /static/ /var/www/html/static/
                Alias /media/ /var/www/html/media/

                <Directory /home/ubuntu/static>
                        SSLOptions +StdEnvVars
                        Require all granted
                </Directory>

                <Directory /home/ubuntu/media>
                        Require all granted
                </Directory>

                WSGIDaemonProcess CrabbyTasks
python-path=/django/CrabbyTasks/_backend python-home=/django/CrabbyTasks/env/
                WSGIProcessGroup CrabbyTasks
                WSGIScriptAlias /api
/django/CrabbyTasks/_backend/config/wsgi.py
                <Directory /django/CrabbyTasks/>
                        <Files wsgi.py>
```

```
                    Require all granted
</Files>
```

Domain: If you plan to use a custom domain, make sure it's configured and points to the server's IP address.



## Local Deployment

1. Clone the Repository

Start by cloning the CrabbyTasks repository to your server.

```
git clone https://github.com/CarelessOutcast/CrabbyTasks.git
```

2. Environment Variables

Set up environment variables for your production environment. These variables should include database credentials, API keys, and any other sensitive information.

```
Filename: CrabbyTasks/_backend/.env
ENV_TYPE=development # or production
DJANGO_SECRET_KEY=your-secret-key
DEBUG=1
DB_ENGINE=your-django-db-backend-engine


DB_DEVELOPER_NAME=you-db-name
DB_DEVELOPER_USER=your-db-name
DB_DEVELOPER_PASSWORD=your-db-password
DB_DEVELOPER_HOST=your-host-route
DB_DEVELOPER_PORT=your-port-db

Filename: CrabbyTasks/_frontend/src/.env
NODE_ENV=dev # or prod
```

### 3. Build the Frontend

Navigate to the project's frontend directory and run the dev server code.

```
cd _frontend
npm install
npm run dev
```

### 4. Start the Backend

Start the backend server using a process manager.

```
Python -m venv CrabbyTasks/env
Source CrabbyTasks/env/bin/activate
Pip install -r CrabbyTasks/_backend/requirements.txt
#IMPORTANT: Set up a database before running if not sqlite3!
Python manage.py runserver
```

### 5. Database Configuration

Ensure the database is configured correctly, and all tables and indexes are created.

```
Python manage.py makemigrations
Python manage.py migrate
```

## Deployment Steps

Follow these steps to deploy CrabbyTasks:

1. Clone the Repository

Start by cloning the CrabbyTasks repository to your server.

```
git clone https://github.com/CarelessOutcast/CrabbyTasks.git
```

2. Environment Variables

Set up environment variables for your production environment. These variables should include database credentials, API keys, and any other sensitive information.

```
Filename: CrabbyTasks/_backend/.env
ENV_TYPE=development # or production
DJANGO_SECRET_KEY=your-secret-key
DEBUG=1
DB_ENGINE=your-django-db-backend-engine


DB_DEVELOPER_NAME=you-db-name
DB_DEVELOPER_USER=your-db-name
DB_DEVELOPER_PASSWORD=your-db-password
DB_DEVELOPER_HOST=your-host-route
DB_DEVELOPER_PORT=your-port-db

Filename: CrabbyTasks/_frontend/src/.env
NODE_ENV=dev # or prod
```

### 3. Build the Frontend

Navigate to the project's frontend directory and build the production-ready frontend code.

```
 cd _frontend
npm install
Npm build
```

### 4. Start the Backend

Start the backend server using a process manager.

```
Python -m venv CrabbyTasks/env
Source CrabbyTasks/env/bin/activate
Pip install -r CrabbyTasks/_backend/requirements.txt
#IMPORTANT: Set up a database before running if not sqlite3!
Python manage.py runserver
```

### 5. Database Configuration

Ensure the database is configured correctly, and all tables and indexes are created.

```
Python manage.py collectstatic
Python manage.py makemigrations
Python manage.py migrate
```

## 6. Proxy Setup

If you're using a reverse proxy like Nginx or Apache, configure it to route incoming requests to the Django backend.

## 7. SSL Certificate

For HTTPS, obtain and configure an SSL certificate for your domain. Tools like Let's Encrypt can help with this. Add the

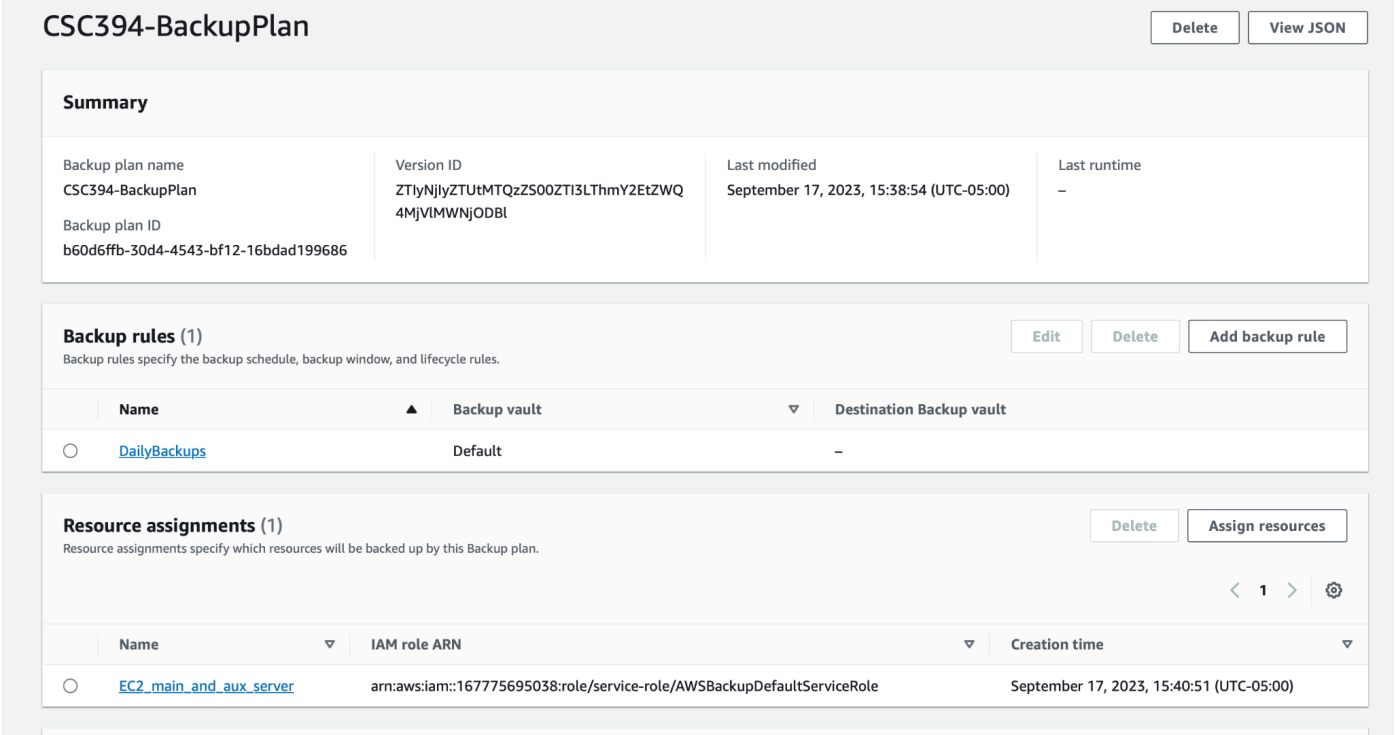
```
ServerName red-lobster.dev

<IfModule mod_ssl.c>
  <VirtualHost _default_:443>

     ... other configuration

    SSLCertificateFile /etc/ssl/red-lobster_dev/red-lobster_dev.crt
    SSLCertificateKeyFile /etc/ssl/red-lobster_dev/red-lobster.key
    SSLCertificateChainFile/etc/ssl/red-lobster_dev/red-lobster_dev.ca-bundle

  </VirtualHost>
</IfModule>
```

## 8. Monitoring and Logging

Implement monitoring and logging solutions to track the application's performance and troubleshoot issues. AWS console can monitor your solutions.

## 9. Backup and Recovery

Establish a regular backup and recovery plan to prevent data loss.




## 10. Scaling and Load Balancing

As the user base grows, we're considering implementing load balancing and horizontal scaling to distribute incoming traffic across multiple server instances using nginx or AWS Load Balancing. This ensures high availability and improved performance.

## 11. Security Considerations

Pay special attention to security during deployment:

- Keep all software components, including the server's operating system, up to date with security patches.
- Configure firewalls to restrict access to only necessary ports and services.

- Implement strong access controls and authentication mechanisms.

Successful deployment is a critical milestone in bringing CrabbyTasks to users. Follow these steps and best practices to ensure a secure and reliable deployment.

## Future Plans

CrabbyTasks will continue tailoring to individual uses and functions for the sake of effortless task management that will ensure user support. There is a commitment to produce a high-quality and reliable service that is constantly improving through user feedback. Adding new features while enhancing already existing ones will increase the chances of having a competitive advantage in the market for professional fields such as curricular activities and business areas; moreover, expanding to different industries can help promote CrabbyTasks' growth and popularity.

A strong focus on marketing and promotional efforts will help it reach a larger audience and become global. As the service gains traction, advertising and blogging on different social media platforms will become more frequent in order for CrabbyTasks to maintain exposure to the public. Customer support will also be a priority so as to improve service consistency alongside establishing customer satisfaction and retention. Red Lobsters will continue to work with their current partners for the pursuit of stronger relationships and trust, which guarantees that any future plans stay on track.

The primary purpose of CrabbyTasks is to increase the productivity of users and streamline workflow through the use of proper planning, scheduling, and organization. While CrabbyTasks currently has the fundamental parts of a task management system, there are several functionalities that have been envisioned as part of the project's evolution:

- Draggable Interface
    - A dynamic and draggable interface reinforces the aspect of customizability and focuses more on the user experience than actual functionality, which is important in its own right as end users commonly desire several quality of life features that can make the product or service more engaging and accessible. Being able to move pop-up windows and certain sections of the website can help improve visibility by reducing visual clutter, increase interface responsiveness by having less components be refreshed, etc. The public reception of CrabbyTasks lies not only in its task management capabilities but also its ability to appeal to the users as they are the ones who will determine the service's success and popularity.
- Flexible Search
    - The addition of a flexible search engine will allow users to query for tasks and their data based on multiple parameters, such as task titles, descriptions, deadlines, categories, status, and so forth. Uniquely customizable searches would have the option to be stored in their own database table for future reuse, saving time and reducing hassle for users who have queries they frequently use.
- Pet System
    - The pet system revolves around the concept of encouraging user efficiency by implementing a mascot that will dynamically change its status and appearance according to how diligent users have been. To determine performance, a point system of sorts will accompany the pet and tasks, adding or subtracting points depending on whether or not the user has completed their tasks on time, how early they finished those tasks, or whether or not they have any overdue tasks and how past the deadline they are. Ideally, users will be motivated to finish their work in a timely fashion so as to level up their pet and keep them in a happy state.
- Exportable Data & Analytics
    - Being able to freely filter and export desired data lets users analyze their progress over time, find patterns in their schedules or lives, and make better use of their

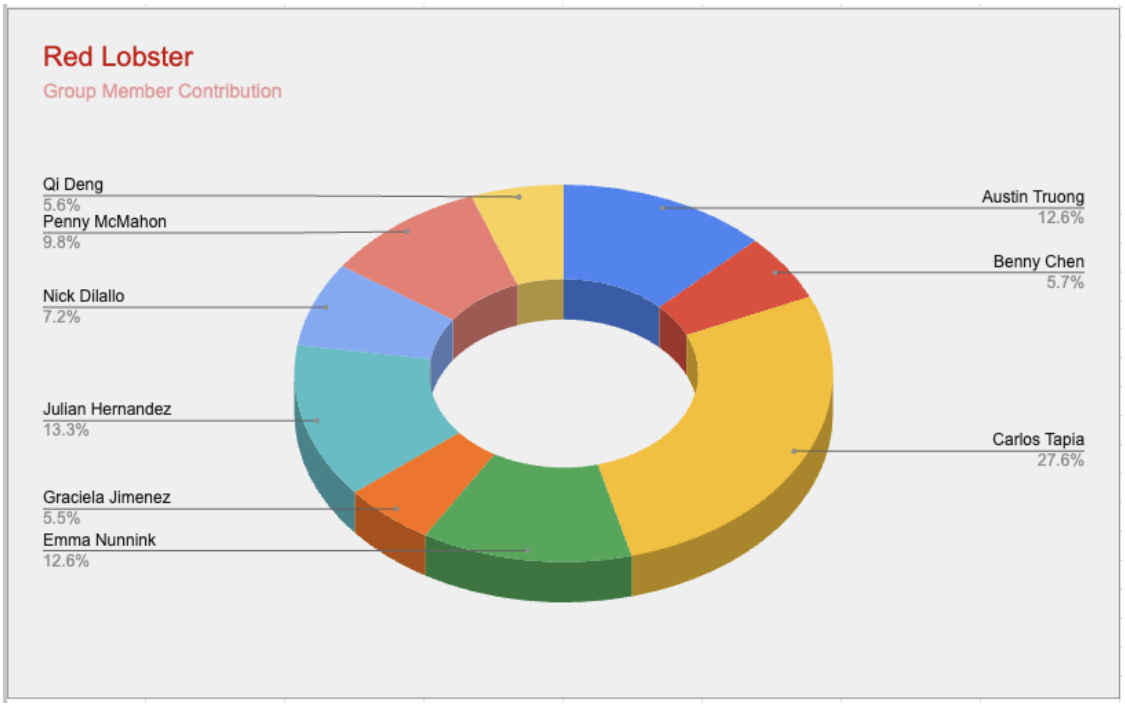
resources so that they can improve task and time management skills. In conjunction with flexible searches, users would be allowed to take those filtered tasks and export them into a format that is more readable and can be shared with others. Exportation of data has many uses, including personal improvement and tracking, cross-schedule organization with others, and so forth.

- Notification System & Toasts
    - Notifications are a solid method of reducing the chances a task deadline is missed. While CrabbyTasks is made for managing tasks and developing management skills, it is completely possible that users will forget deadlines every now and then; however, reminder notifications can be an easy solution that helps keep users on top of their schedule without being a constant annoyance. Implementing such a system would give the users the ability to set up email notifications for each individual task if they wanted as well as the frequency of those reminders in order to properly reflect their priorities. Toast notifications, which are akin to pop-up messages that briefly provide an alert or information to users, can be allowed as well for when users are actively on their device and CrabbyTasks.
- Mobile & Browser Compatibility
    - There is no guarantee that users will always be at their computer or on the correct browser when they want to access a website/application, and attempting to do so on a different device or browser can result in a negative user experience without proper compatibility. Often, there are users who spend little to no time at a computer, which means that the only apparatus they can use is their phone. There will be errors and irregularities with trying to transfer from one form to the other if a product or service that is supposed to be used in a certain way is used differently, leading to time consumption and frustration. Having an appropriate mobile application or website will make sure that users are able to comfortably access and utilize CrabbyTasks regardless of their medium or method.

- Artificial Intelligence & Machine Learning
    - In line with our vision to reduce mental overhead, we plan on incorporating machine learning models that will further enhance user experience. The primary use of machine learning will be to automate and suggest tasks in order to alleviate the mental toll placed on the user. The model will specifically be capable of learning the user's daily habits and provide suggestions for tasks at certain intervals of the day. In addition to suggestions, it will also be capable of tagging, categorizing, and organizing tasks automatically, thereby improving user efficiency. The suggestion model will be built using techniques like singular value decomposition or clustering. Similarly, there will be large language models to keep tasks categorized and organized.

# TimeLogs

| Name | Total Hours |
|---|---|
| **Red Lobsters** | **779.65** |
| Austin Truong | 98.2 |
| Benny Chen | 44.8 |
| Carlos Tapia | 215.2 |
| Emma Nunnink | 98.5 |
| Graciela Jimenez | 43.25 |
| Julian Hernandez | 103.45 |
| Nick Dilallo | 56.25 |
| Penny McMahon | 76.5 |
| Qi Deng | 43.5 |




a. AUSTIN TRUONG

| Name | Date | Hours | Full Description |
|---|---|---|---|
| Austin Truong | 9/12/2023 | 2 | Met with group; Role Distribution; Project topic development; Establish communication; Create contact info document; Submit Attendance and role worksheets |
| Austin Truong | 9/13/2023 | 2.75 | Met with group; Fleshed out group project idea; Round table to share experience/skill set; Create Task/Jira Board and Assigned Tasks; Go over role responsibilities in-depth; Setup rough deadlines; Create meeting overview/takeaways; Determine platform/server/materials used; Reformat Timelog document with Documentation Manager |
| Austin Truong | 9/14/2023 | 1.5 | Met with group; Reassigned Requirements Manager Role; Reassess individual/group responsibilities; Solidify plan/process for project; Determined languages, platform, etc. |
| Austin Truong | 9/14/2023 | 2 | Create recordings folder + shared with team; create meeting overview/takeaways; adjusted tasks on Asana Board; research Django; reviewed partially-completed work/tasks |
| Austin Truong | 9/15/2023 | 5 | Research Django, work on creating function views, database models/tables, and url paths |
| Austin Truong | 9/15/2023 | 1 | Review work done by team members |
| Austin Truong | 9/16/2023 | 7 | Refine database models, views; discuss with Unix Programmer/Manager and Webmaster about implementation; discuss project priorities and plan moving forward after concept demo |
| Austin Truong | 9/16/2023 | 2 | Met with group; gauge project progress/completion with team; determine concept demo priorities; establish homogenous ideas and concepts going forward, discuss remaining work for next couple days; decide on game plan for next week after concept demo |
| Austin truong | 9/16/2023 | 1 | Created meeting overview, researched more on Django properties and syntax for code |
| Austin Truong | 9/16/2023 | 1 | Worked on resolving merge conflicts with Unix Programmer/Manager; fixed errors in codebase |
| Austin Truong | 9/17/2023 | 1.5 | Met with group; make sure materials are ready for presention/video demo; refined project booklet; created meeting overview; pushed minor changes to main branch |
| Austin Truong | 9/17/2023 | 0.7 | Reviewed and finalized testing matrix and frontend changes; |
| Austin Truong | 9/19/2023 | 0.5 | Look over video demos; brainstorm ideas (alone + discussed with some group members) for project and future plans |
| Austin | 9/19/2023 | 1 | Met with group; discussed RDP demo; retrospective on concept demo and |

| | | | |
|---|---|---|---|
| Truong | | | improvements; finalized language and framework choice (Django); went over plans/meetings for remaining week |
| Austin Truong | 9/19/2023 | 0.5 | Met with Unix/Systems Programmer to discuss implementation plans, preparation, and ideas for upcoming meeting (tomorrow 9/20); also briefly talked about AWS IAM roles possibility and alternatives down the line |
| Austin Truong | 9/19/2023 | 0.3 | Review meeting contents and create overview/takeaway |
| Austin Truong | 9/20/2023 | 2 | Met with group; go over more about RDP and each section; create base draft of client proposition; list potential design diagrams in document; discuss planning, time log, and organization structure moving forward, cleaned up + created new Asana boards for implementation teams |
| Austin Truong | 9/20/2023 | 1 | Rewatch recording to make meeting overview/takeaways; research on business edge cases to find references/links for team board; |
| Austin Truong | 9/21/2023 | 1.5 | Met with group; discussed progress completed from yesterday to today; touch base with requirements and design managers as a team; determine priorities of tasks and what to keep in mind; |
| Austin Truong | 9/21/2023 | 0.6 | Make meeting overview; research on AWS Fargate/Docker; |
| Austin Truong | 9/23/2023 | 1 | Reviewed progress/completed work on RDP Requirements Document |
| Austin Truong | 9/23/2023 | 4 | Work on AWS Assignment + AWS Quiz |
| Austin Truong | 9/24/2023 | 2 | Reviewed more progress on Requirements Document; Review/Work on Client Proposal Document; Minor changes/additions to Requirements Document |
| Austin Truong | 9/26/2023 | 1.5 | Meet with team; refine Asana board; create new tasks; refine/review project booklet, client proposal, requirements document, and requirements matrix |
| Austin Truong | 9/28/2023 | 2 | Work on requirements document; work on reviewing requirements matrix bad client/dev functional sections; test docker setup |
| Austin Truong | 9/29/2023 | 1 | Meet with team; discuss progress; refine Asana board and tasks; decide on branching/pipeline structure with implementers; touch base with design manager |
| Austin Truong | 9/30/2023 | 2 | Look over project booklet examples; refine requirements document; review requirements matrix; look into database table structure needed later |
| Austin Truong | 10/2/2023 | 1 | Review requirements document progress; work on bad client/dev attacks document |
| Austin Truong | 10/3/2023 | 3 | Meet with team; discuss progress; review and refine documents with team; determine upcoming priorities and deadlines; touch up requirements matrix |

| | | | |
|---|---|---|---|
| Austin Truong | 10/4/2023 | 2 | Review requirements document; look over planning document; briefly review Asana board and tasks; |
| Austin Truong | 10/5/2023 | 1 | Revise requirements matrix + bad dev/client sections; sign off on revisions |
| Austin Truong | 10/5/2023 | 1.5 | Meet with team; discuss progress and deliverables for demo; determine tasks and game plan for next 2-3 days; identify dependencies |
| Austin Truong | 10/5/2023 | 0.5 | Discuss briefly with Systems Programmer about designs/diagrams, alternatives/fallbacks |
| Austin Truong | 10/6/2023 | 1 | Finish requirements matrix attacks and sign off on functional requirements sections; removed unnecessary minor section from requirements document |
| Austin Truong | 10/6/2023 | 2 | Meeting with Systems Programmer, Video/Presentation Manager, and Design Manager to go over diagrams and presentation/video structure(s). |
| Austin Truong | 10/8/2023 | 0.5 | Check-in progress with Systems Programmer & Video/Presentation Manager |
| Austin Truong | 10/8/2023 | 1 | Meet with team; discuss progress for demos and what is left to be finished; admin quizzes |
| Austin Truong | 10/8/2023 | 2 | Work with Implementation Manager and Planning/Documentation Manager on Planning section of presentation/video |
| Austin Truong | 10/10/2023 | 1.5 | Meet with team; refine Asana board; create new tasks; determine future priorities for project; assign/designate group members to frontend/backend teams; discuss pipelining and plans for next few days |
| Austin Truong | 10/12/2023 | 1.5 | Meet with team; flesh out planning and tasks for preview demo; go in-depth with specific project functionalities to clear out any misunderstandings; make unanimous decisions regarding backend and frontend implementation; |
| Austin Truong | 10/14/2023 | 0.5 | Meet with Backend Implementation team; set up virtual environement; create tasks on Asana Board for backend team; discuss scheduling and tasks for next few days; |
| Austin Truong | 10/15/2023 | 0.5 | Meet with team; progress checkup between frontend and backend teams; decide on React templates and dependencies; |
| Austin Truong | 10/15/2023 | 0.75 | Set up virtual environment and server; test React template page working on server; look at table attributes with respect to functionality planning to be implemented |
| Austin Truong | 10/16/2023 | 1 | Alter table models to clean up unnecessary attributes and add new ones; create dropdown/categories and models for certain attributes; create forms.py file to work with the dropdown forms |
| Austin Truong | 10/17/2023 | 2 | Meeting with team; discuss frontend/backend team progress and changes; decide on frontend elements and functionality; determine priorities for overall team and individual implementation teams |

| | | | |
|---|---|---|---|
| Austin Truong | 10/17/2023 | 1 | Revert forms.py file after talking with frontend and backend teams to decide dropdowns are easier done with frontend elements; review changed and new files added |
| Austin Truong | 10/19/2023 | 1 | Team meeting to discuss planning and module changes/progress; decide on creating landing page for website; create new tasks for members not present |
| Austin Truong | 10/20/2023 | 1 | Review changes on backend team's branch for JWT authorization, access tokens, and endpoints; research event notification |
| Austin Truong | 10/21/2023 | 0.4 | Review PR; discuss with Unix Programmer implementation strategies |
| Austin Truong | 10/21/2023 | 1 | Clear up questions about sending requests/models to frontend; suggest query parameters and using less endpoints that are more flexible; think about how many endpoints are needed overall; make executive decision on what team needs to immediately prioritize |
| Austin Truong | 10/22/2023 | 1.5 | Meeting with team; discuss progress made for frontend/backend; determine what/what not to include for frontend UI; test our query params and flexible queries; research django rest framework serializers and filters; go over preview demo requirements and components with team; |
| Austin Truong | 10/22/2023 | 1.5 | 1-on-1 with Unix Programmer to research/brainstorm how to use Django rest framework serializers, optional parameters/fields, and api requests to reduce amount of endpoints; conceptualize/pseudo-code point completion system with Unix Programmer |
| Austin Truong | 10/22/2023 | 0.3 | Determine priorities and what's left to be done/completed before the preview demo for both frontend/backend teams |
| Austin Truong | 10/23/2023 | 0.2 | Look over changes in developer branches; cut down unnecessary endpoint(s)/code. |
| Austin Truong | 10/24/2023 | 1.5 | Team meeting to discuss team progress on requirements/planning, frontend/backend, and plans for project booklet; decide on scheduling/plans for preview demo components and preparation; |
| Austin Truong | 10/24/2023 | 2 | Review requirements document; continue working on requirementst spreadsheet; look over PR/changes in developer branches |
| Austin Truong | 10/25/2023 | 0.4 | Review developer PR for updating frontend logos, UI, etc. |
| Austin Truong | 10/26/2023 | 2 | Continue working on/revising requirements spreadsheets (almost complete); work with backend team to give proper code snippets to Video/Presentation manager for demo |
| Austin Truong | 10/26/2023 | 1 | Meeting with team to discuss progress on website, go over state/mode requirements section to make sure there are no discrepencies/inconsistencies with actual website; go over components for preview demo; Presentation/Video Manager informs team about progress |

| | | | |
|---|---|---|---|
| | | | and what they need to start on the demo |
| Austin Truong | 10/27/2023 | 0.3 | Review PR for calendar changes and merge branches |
| Austin Truong | 10/27/2023 | 1 | Review/comment on preliminary preview demo script |
| Austin Truong | 10/28/2023 | 1 | Finish approving/revising requirements spreadsheet; discuss with team over discord regarding implementation progress and preview demo progress |
| Austin Truong | 10/28/2023 | 0.5 | Research error in frontend page file due to React, typescript, context, and null objects |
| Austin Truong | 10/29/2023 | 0.5 | Meeting with team to finalize progress, requirements, and components for preview demo |
| Austin Truong | 10/31/2023 | 1.5 | Meeting with team to discuss unanimous decisions regarding project booklet, upcoming meetings, quizzes/submssions, etc. |

b. BENNY CHEN

| Name | Date | Hours | Full Description |
|---|---|---|---|
| Benny Chen | 9/12/2023 | 1.5 | Met with group; Role Distribution; Project topic development; Establish communication |
| Benny Chen | 9/17/2023 | 3 | Created Conflunece Page, Refine Discord for Efficiency, Began process of consolidation of materials to a singular communication software |
| Benny Chen | 9/19/2023 | 1.5 | Group meeting, Solidified Roles, Discussion plans for upcoming weeks |
| Benny Chen | 9/26/2023 | 1.5 | Group meeting, Overview of Requirements & Design on what needs to be done, Creating Carrywater Tasks, Reminder of Future Tasks & Deadlines |
| Benny Chen | 10/2/2023 | 1 | Review requirements document progress; client/dev attacks doc progress; |
| Benny Chen | 10/6/2023 | 1.5 | Assisting Graciela's Documentation by Refining / Brainstorming / Documentation of multiple documents (Time Log, Red Lobster Planning, Project Booklet) (WIP Cannot connect communication) |
| Benny Chen | 10/10/2023 | 1.5 | Group meeting, Set reminder to finalize all documentation so that it will be ready for transfer to Project Booklet (Planning Intialed), Establish Frontend and Backend team (Full Implementation work from now on) |
| Benny Chen | 10/16/2023 | 1.6 | Preparing for meeting with Graciela by creating Asana Board with Documentation Agenda & Deadlines, Project Booklet Outline, and Time Log Template Idea, 1 on 1 Meeting with Graciela (Any questions/comments regarding Project Booklet, New Time Log organization, confirmation of |

| | | | |
|---|---|---|---|
| | | | certain documents to be finalize.) |
| Benny Chen | 10/17/2023 | 1.5 | Group meeting, Reviewed of Frontend & Backend plan outline, Consolidated jobs needed to be done in the Frontend & Backend |
| Benny Chen | 10/24/2023 | 1.5 | Group meeting, Finalized Backend & Frontend plans to prepare for Preview Demo, Update status on each member and what needs to be done, Reminder for future tasks to be started on ASAP |
| Benny Chen | 10/27/2023 | 1 | Created Business Plan for Preview Demo Presentation |
| Benny Chen | 10/31/2023 | 1.5 | Group meeting, Discussed what final tasks are needed to be done and preparation for future documentations submissions |
| Benny Chen | 11/1/2023 | 2.3 | Reformatted Project Manual, Created Project Manual Cover Page Design and Header Design, Reformatted and fix Time Log issues |
| Benny Chen | 11/2/2023 | 3.5 | Created a table of content that is ready to be finalize so that informations can be build upon it (revision will be made to the table of content if there are missing informatons), 80% completion on Project Overview seciton in project manual, Reformatted whole document to default (Once informations are complete then formatting will reoccur, this is to prevent unncessary time waste in between), Complete Project Overview section and continue working on Business Plan 25% completion. |
| Benny Chen | 11/3/2023 | 3.5 | Continued working on Business Plan (This section is carefully done to ensure all informations/details are correct and simple to catch investors right at the beginning. Began structuring PROJECT JOBS & RESPONSIBILITIES SPECIFICATIONS sections. Small meeting regarding questions about project manual. Continued working on second portion of Business Plan (Marketing & Sales, Product & Service Line), Created management team structure inforgraphic |
| Benny Chen | 11/4/2023 | 1 | Started working Communcation sections, Completed marketing & sales section. |
| Benny Chen | 11/5/2023 | 2.8 | Nearly Completed Business Plan, Completed Communications section, Nearly completed Project Jobs & Responsiblities Specifications sections (waiting for approval for bio cards) |
| Benny Chen | 11/6/2023 | 3 | Started creations of Finance section of Buiness Plan, Inputed Meeting Notes |
| Benny Chen | 11/7/2023 | 3.75 | Group meeting, Refined project manual outline, Reviewed Github permissions, Testing with code block extensions google doc, Share a copy of project manual for group to input information/data, Planning for this week and upcoming week, Completed Meeting Notes, Bio cards approved and completed PROJECT JOBS & RESPONSIBILITIES SPECIFICATIONS sections, (Need to review certain sections and confirmation on certain informations) |

| | | | |
|---|---|---|---|
| Benny Chen | 11/8/2023 | 2.85 | Created new sections, Completed Planner Sections, Completed Time Logs Section (added more details), Completed Buisness Plan (ready to be reviewd), Started Client Proposition (Need confirmation from Graciela for which google doc is official) |
| Benny Chen | 11/9/2023 | 1.5 | Completed Github Insights |

### c. CARLOS TAPIA

| Name | Date | Hours | Full Description |
|---|---|---|---|
| Carlos Tapia | 9/9/2023 | 3 | 7:07-10:03pm; Aws assignment; AWS learning; Researching Project Ideas; Researching CRUD |
| Carlos Tapia | 9/12/2023 | 1.5 | Met with group; Role Distribution; Project topic development; Establish communication |
| Carlos Tapia | 9/13/2023 | 4 | Met with group; Discussed features, deadlines, responsibilities; Set up Asana; Wordpress user Setup; Server Setting up; Implementation / Design Discussion |
| Carlos Tapia | 9/14/2023 | 1.25 | 8:14 - 9:29am; Production Server Research; Full Stack Deployment research; Architecture exploration; AWS console exploration |
| Carlos Tapia | 9/14/2023 | 3.25 | 10:01 - 1:16pm; SSL certification Research; Apache server configuration; AWS Network routing; HTTPS Troubleshooting; DNS Resolution; Domain Purchasing; Domain Instalation; SSL Apache Configuation; AWS Routing; |
| Carlos Tapia | 9/14/2023 | 1.5 | 1:28 - 2:59pm; Continued SSL Apache configuration; SSL Purchase; Continued AWS route 53 CName & A routes (SSL verificatoin); DNS waiting & internet propagatation; |
| Carlos Tapia | 9/14/2023 | 1 | 3:27 - 4:22pm; Red-lobster officially live; (Odd) Fix issue with Apache Virtual host files with SSL (HTTP/HTTPS); |
| Carlos Tapia | 9/14/2023 | 1 | 4:51 - 5:51pm; Wordpress running on Red-Lobster; Discord Discussion w/Team; Wordpress User creation; Group files hosted on my instance; |
| Carlos Tapia | 9/14/2023 | 0.5 | 5:54 - 6:19pm; Project look up on Clark Elliot's Site & D2l Discussion board; |
| Carlos Tapia | 9/14/2023 | 2.25 | 6:22 - 8:38pm; Video conferencing; IAM AWS roles research; Introduction to Django research; |
| Carlos Tapia | 9/15/2023 | 0.25 | 9:16 - 9:38am; Planning work day and objectives for to accomplish |
| Carlos Tapia | 9/15/2023 | 4.75 | 9:54 - 2:38pm; Django research; Django production server readiness; Django Apache communication; Discord answering; Django troubleshooting |

| | | | |
|---|---|---|---|
| Carlos Tapia | 9/15/2023 | 4.25 | 2:42 - 7:02pm; Django Development; Discord communication; Github attaching; Django is Live; |
| Carlos Tapia | 9/15/2023 | 4.5 | 7:09 - 11:35pm; Django Development; |
| Carlos Tapia | 9/16/2023 | 0.75 | 8:46 - 9:28am; planning word day and objectives to accomplish; Django Development & Research; |
| Carlos Tapia | 9/16/2023 | 2.5 | 9:47 - 12:25pm; Django Development; Github on EC2; |
| Carlos Tapia | 9/16/2023 | 0.5 | 12:30 - 12:53pm; Discord w/ Austin & Julian |
| Carlos Tapia | 9/16/2023 | 0.5 | 1:09 - 1:38pm; Discord Discussion |
| Carlos Tapia | 9/16/2023 | 1 | 1:43 - 2:39pm; Django Development |
| Carlos Tapia | 9/16/2023 | 4 | 2:45 - 6:38pm; Django Semi Functional; |
| Carlos Tapia | 9/16/2023 | 1.5 | 6:38 - 8:14pm; Video Meeting; Overview of task to be completed; emphasized on roles; |
| Carlos Tapia | 9/16/2023 | 3.75 | 8:14 - 11:57pm; Finished Django Development |
| Carlos Tapia | 9/17/2023 | 4.5 | 2:08 - 6:37pm; Research Backups on AWS; Create backup servers; Create Backup Schedules for Instances; Researching Redundancy for Databases; |
| Carlos Tapia | 9/17/2023 | 1 | 6:29 - 7:32pm; Meeting with Team; |
| Carlos Tapia | 9/18/2023 | 2 | 1:20- 3:33pm; Red Lobster discussion about demo with team on discord; Fixing concept demo frontend for emma; Giving emma the pictures and my contributions as description she needs for the demo; |
| Carlos Tapia | 9/18/2023 | 0.5 | 10:07 - 10:30am; Researching IAM roles; Troubleshooting IAM roles for team access to EC2 instance; |
| Carlos Tapia | 9/20/2023 | 2 | 6:29 - 8:30pm; Meeting with Team; Discuss following week requirements for requirements, design, planning demos; Changed the Asana boards; Created Carry Water tasks; Talked with Project manager about system infrastructure. |
| Carlos Tapia | 9/20/2023 | 0.7 | 3:07 - 3:47pm; Researching Production level django setups; Looking into how to create a database as a standalone unit; Looking into different databases that easily incorporate into django; |

| | | | |
|---|---|---|---|
| Carlos Tapia | 9/22/2023 | 3.75 | 1:32 - 5:16pm; Reviewed D2l Lectures for requirements, design, and planning sections; Figuring out different database options to host production level db; AWS Quiz; Figuring out how to make django production level ready; Looking into what AWS has to offer our team; Understanding and Tinkering with apache config file and other modules that it has; Understanding what a load balancer is and how apache can work as one; How to secure a production level database via the command line; Looking at billing and usage rates on AWS through cloudwatch; Contacting Project manager for his help on connecting to my personal Ec2 instance & further trying to troubleshoot IAM roles on AWS; Giving PM key files to my EC2 instance and actually setting up ssh keys; |
| Carlos Tapia | 9/22/2023 | 3 | 5:39 - 8:39pm; Figuring out what is required for the requirements document by looking for examples online; Looking at Blue Rhino examples; |
| Carlos Tapia | 9/22/2023 | 2 | 9:07 - 10:58pm; Filled out the requirement sections for system requirements (9.1); Made a template the document section; Made the Risk mitigation table and started to fill out; Wrote the freeze date; Formatted the document (look); Started to fill out client LIason section (7.1 - 7.5); Filled out design requirements section (5.1 -5.2); Filled template for project planning (1.1) |
| Carlos Tapia | 9/22/2023 | 1.75 | 11:41 - 1:24am;Figuring out what design documents we need; Finding software that will help me create the documents (Axure & Drawio);Understanding what software architecture models we can use; Looking for more examples; |
| Carlos Tapia | 9/23/2023 | 5 | 12:43 - 5:41pm; Researching what state, class, database, dataflow, logic, object diagrams; Created concept diagrams (state, class, database, dataflow, logic, and object) for our project using mermaid markdown syntax and drawio; Shared the diagrams with the team and asked for feedback; Discord conversation with the team about what should be getting done; |
| Carlos Tapia | 9/23/2023 | 2 | 7:39 - 9:32pm; Installing Axure for prototyping the frontend; Running through the Axure tuturial; Reading Mongodb documentation; Creating a new instance to host MongoDB on AWS; Looking at billing and cloudwatch for rates; |
| Carlos Tapia | 9/24/2023 | 2 | 12:44 - 3:03pm; Securing mongodb instance by revoking security ports; Securing mongodb with ufw; Troubleshooting connection to mongodb of local django project; Setting Ulimit resources on AWS Instance for optimal performance; Solving connection issue caused by priviledges when originally installing mongodb; Creating new mongodb users with least priviledges to database; |
| Carlos Tapia | 9/24/2023 | 2.5 | 4:00 - 6:38pm; Video conference with team to see progress on project; |
| Carlos Tapia | 9/24/2023 | 0.75 | 7:12 - 8:00pm; Making django migrations to the mongodb; Connection successful; Creating mongodb tables; |

| | | | |
|---|---|---|---|
| Carlos Tapia | 9/24/2023 | 3 | 8:04 - 11:09pm: Found out that django does NOT work natively with Non-Sql databases; Deciding if it is worth continuing with django without some native features; Deciding that team would rather have django features to reduce development friction; |
| Carlos Tapia | 9/25/2023 | 1.75 | 12:04 - 1:54pm; Communicating to team on the set back; Agreeing that we are pivoting to PostgresSql; Starting to read postgres documentation; Downloading PGAdmin; Deleting Mongodb instance; |
| Carlos Tapia | 9/25/2023 | 2 | 3:04 - 5:04pm; Understanding IP CIDR Blocks/ subnetting; Creating a Virtual Private Cloud to connect EC2 web server and EC2 Database on AWS; Installing Postgress, and configuring postgres, adding postgres users (developer & production); |
| Carlos Tapia | 9/25/2023 | 1.75 | 5:50 - 7:35pm; Installing Docker; Learning the basics of Docker; Learning the basics of Docker compose; Creating a basic Django application on docker images; |
| Carlos Tapia | 9/25/2023 | 1.5 | 8:49 - 10:22pm; Configuring Postgres to allow connection to local development environment; Starting to establish different environment variables for different branches; |
| Carlos Tapia | 9/26/2023 | 1.25 | 8:17 - 9:28am; Trying to understand how AWS allows programmatic access to AWS resources; Configuring Postgres to allow connection to Docker version of local development environment; |
| Carlos Tapia | 9/26/2023 | 0.75 | 9:53 - 10:35am; Connecting pgadmin to postgres databse; Finally connecting Docker Django env to postgres database on local docker file; |
| Carlos Tapia | 9/26/2023 | 1 | 10:56 - 11:50am; Installing Docker on web server; Learning how I can programatically change the docker image to take production env variables rather than development variables; looking to introducte CI CD pipeline for team using Github actions; |
| Carlos Tapia | 9/26/2023 | 1.5 | 3:32 - 4:37pm; Continuing to look into github actions for Ci / CD pipeline; Continuing to learn how to automate docker images to run my application for two different environments (dev and production); |
| Carlos Tapia | 9/26/2023 | 1.5 | 6:59 - 8:38pm; Class and Meeting with team to establish goals for the week; |
| Carlos Tapia | 9/26/2023 | 0.25 | 8:40 - 9:02pm; Logging all my hours for the week; |
| Carlos Tapia | 10/2/2023 | 1 | 4:11 - 5:11pm; Review requirements document progress; client/dev attacks doc progress; |
| Carlos Tapia | 10/3/2023 | 2 | 7:00 - 8:58pm; Meet with team; discuss progress; review and refine documents with team; determine upcoming priorities and deadlines; touch up requirements matrix; |

| | | | |
|---|---|---|---|
| Carlos Tapia | 10/5/2023 | 1 | 7:11 - 8:15pm; Team Discussion on tasks that need to be completed to be able to submit RDP demo; Discussed upcoming tasks that will need to be completed; |
| Carlos Tapia | 10/7/2023 | 4.5 | 1:33 - 5:59pm; Helping Presentation Manager create Design demo script; |
| Carlos Tapia | 10/7/2023 | 2 | 6:09 - 8:00pm; Iterating over the design demo requirements with Presentation Manager; Working to understand exactly what was needed in the demo; |
| Carlos Tapia | 10/7/2023 | 2.5 | 11:06 - 1:51am; Trying to record professional video takes for the presentation manager; Failed to acquire the professional takes because of fatigue; |
| Carlos Tapia | 10/8/2023 | 1 | 11:45 - 12:47pm; Recorded all the sub videos (20 design requirements, check Design Outline) for the Design Section of RDP Demos; |
| Carlos Tapia | 10/8/2023 | 2.25 | Video Conference call with Presentation Manager to help guide insertion of video assets into a cohesive story. |
| Carlos Tapia | 10/10/2023 | 2 | 6:51 - 9:04pm; Met with group; Discussed seperation of frontend and backend; Outlining planning for the week, and how we can all contrib; |
| Carlos Tapia | 10/10/2023 | 1.75 | 9:04 - 10:42pm; Looking for optimal way to seperate frontend; Understanding what frontend actually means and planning on how to it plays into production env. |
| Carlos Tapia | 10/13/2023 | 2.5 | 7:20 - 9:50pm; Actually seperating out frontend; Cleaning up project directory; Integrating frontend designs, and general project structure; |
| Carlos Tapia | 10/14/2023 | 2 | 2:15 - 4:04pm; Researching Auth with jwt, OAuth2.0; Understanding basic structure of seperating/securing api/routes; |
| Carlos Tapia | 10/15/2023 | 0.5 | 6:00 - 6:20pm; Quick meeting to check on progress of project |
| Carlos Tapia | 10/15/2023 | 1.5 | 6:30 - 8:02pm; Erasing local frontend; Exchanging swapping project frontend with open source dashboard, and cleaning up repo; |
| Carlos Tapia | 10/15/2023 | 2 | 8:32 - 11:24pm; Meeting with frontend lead (Nick) to discuss open source dashboard modifications, and tasks; Documenting and understanding the dashboard; |
| Carlos Tapia | 10/15/2023 | 0.75 | 11:33 - 12:19am; Understanding REST api protocol; Installing tools (postman); Adding Rest Api Views and endpoints; |
| Carlos Tapia | 10/17/2023 | 2.25 | 6:51 - 9:08pm; Weekly Meeting discussed frontend tasks with everyone; Discussed high level overview of how I imagine backend and frontend to function with team; Review progress; |
| Carlos Tapia | 10/18/2023 | 1.5 | 8:16 - 9:53pm; Quiz Runthrough |

| | | | |
|---|---|---|---|
| Carlos Tapia | 10/19/2023 | 1 | 8:18 - 9:26am; Creating JWT endpoints on Rest framework; |
| Carlos Tapia | 10/19/2023 | 1.5 | 9:52 - 11:18am; Looking at incorporating Oauth2 for Dashboard signin |
| Carlos Tapia | 10/19/2023 | 6 | 8:35 - 2:39am; Add to task model; fixed task defaults; Added serializer to tasks; Added tasks endpoints; Added Api testcases; Added test database; |
| Carlos Tapia | 10/20/2023 | 5 | 2:45 - 7:41am; Refactor User model and User auth to work with JWT; FLUSHED DEV Database; Fixed STUPID MIGRATIONS ERRORS; |
| Carlos Tapia | 10/20/2023 | 2 | 2:14 - 4:14pm; Added React axios calls to endpoints to store access_token on local storage; Added user creation to React Signup; |
| Carlos Tapia | 10/20/2023 | 0.75 | 4:14 - 4:45pm; Logging hours since 10/8; Letting everyone know Basic Login stuff is done; Looking to add Oauth2 |
| Carlos Tapia | 10/21/2023 | 1.5 | 3:56 - 5:25pm; Looking into creating the CRUD endpoints on the django framework; Understanding all the different ways you can implement an endpoint with Pros/Cons |
| Carlos Tapia | 10/21/2023 | 1.75 | 6:10 - 8:00pm; Creating the first endpoint for the CRUD (Retrive all tasks); Learning how to create the filter for database queries; Creating an endpoint to get a specific query using the filters with Project manager; |
| Carlos Tapia | 10/21/2023 | 1.5 | 8:05 - 9:38pm; Understanding how to filter database objects based on the request (user); Django Database managers; |
| Carlos Tapia | 10/21/2023 | 2.5 | 9:43 - 12:21am; Understanding basic permission, system permission, object permissions; Implementing the foundational permissions for objects; |
| Carlos Tapia | 10/22/2023 | 3 | 12:45 - 3:46am; Fixing an authentication error that wasn't letting requests pass through to the api; Revamped the JWT frontend requests; |
| Carlos Tapia | 10/22/2023 | 2.75 | 2:13 - 5:00pm; Creating the First Create endpoint in django; Starting to develop the Frontend input box using Tailwind CSS and React; |
| Carlos Tapia | 10/22/2023 | 1 | 5:07 - 6:07pm; attempting to fix axios header issue that wasn't sending permission token to api for requests |
| Carlos Tapia | 10/22/2023 | 1 | 6:59 - 8:01pm; Talking with PM about how to implement more endpoints for the system |
| Carlos Tapia | 10/22/2023 | 1 | 9:16 - 10:24pm; Still attempting to fix axios header issue |
| Carlos Tapia | 10/22/2023 | 0.5 | 11:14 - 11:54pm; fixed the axios header issue; Create endpoint was finalized; |
| Carlos Tapia | 10/23/2023 | 1.75 | 12:03 - 1:57am; Creating the remaining functions for the frontend task form; |

| | | | |
|---|---|---|---|
| Carlos Tapia | 10/23/2023 | 1.5 | 2:08 - 3:47am; Creating the delete function to the frontend; |
| Carlos Tapia | 10/23/2023 | 3.5 | 3:04 - 6:41pm; Refactoring the backend endpoint for UpdateDelete to fit axios request. Officially finished Create and Retrieve endpoint for both frontend and backend. |
| Carlos Tapia | 10/24/2023 | 2.5 | 7:37 - 10:13am; Finishing up the Delete endpoint for the form submission; Starting the update frontend update task modal; |
| Carlos Tapia | 10/24/2023 | 1.25 | 10:31 - 11:49am; Creating the dashboard task list chart; Creating a global task context for all the sub components; |
| Carlos Tapia | 10/24/2023 | 0.75 | 12:12 - 1:03pm; Continuing the task modal after creating the task chart; |
| Carlos Tapia | 10/24/2023 | 1.75 | 4:28 - 6:24pm; Finishing the update task modal; Trying to resolve issue with tasks breaking the site because of axios asynchronous process of retriving tasks is undefined prior to retrieval. |
| Carlos Tapia | 10/24/2023 | 3 | 6:55 - 10:01pm Weekly Meeting with the team; We NEED to finish basic crud. |
| Carlos Tapia | 10/24/2023 | 2.25 | 10:51 - 1:09am; Continuing to try to fix axios task context breaking the site because of race conditions; Fixed by adding loading state; |
| Carlos Tapia | 10/25/2023 | 5.5 | 1:14 - 6:43pm; Finally finishing up the update task, and delete to not break the site on load; Officially finished update and delete for both frontend and backend. |
| Carlos Tapia | 10/25/2023 | 2.75 | 7:00 - 9:50pm; Looking to fix issue with global task stats breaking the site because another race condition. |
| Carlos Tapia | 10/27/2023 | 1.5 | 12:42 - 2:18pm; Refactoring Task Context object to hold the task stats on retrieval after not being able to fix race condition per component basis. |
| Carlos Tapia | 10/27/2023 | 3 | 3:49 - 6:56pm; Finally Fixing the Task Context to work with stats. Adding private routes for the authenticated users on the frontend; |
| Carlos Tapia | 10/27/2023 | 1 | 7:04 - 8:00pm; Fixing calendar issue with off by one error; |
| Carlos Tapia | 10/27/2023 | 2.5 | 8:03 - 10:28pm; Fixing issue with charting; Refactoring sign up page for the application; |
| Carlos Tapia | 10/27/2023 | 1 | 11:47 - 12:50am; Finalizing the last of the changes with the Implementation manager; |
| Carlos Tapia | 10/28/2023 | 1.5 | 1:22 - 12:50am; Fixing aesthetic issues with site and making sure everything is functioning correctly through personal testing; |
| Carlos Tapia | 10/28/2023 | 3 | 9:37 - 12:43am; Looking for production issues for the application; Deployment and building issues; |

| | | | |
|---|---|---|---|
| Carlos Tapia | 10/29/2023 | 2.25 | 3:00 - 5:21pm; Recording the Preview Demo with Presentation manager; |
| Carlos Tapia | 10/29/2023 | 2 | 5:48 - 7:54pm; Recoding security and backup section for Presentation manager; |
| Carlos Tapia | 10/29/2023 | 2 | 8:48 - 10:48pm; Watching all the demos; reading the discussion posts; Logging my hours for the week; |
| Carlos Tapia | 10/31/2023 | 1.5 | 7:10 - 8:40pm; Meeting with the team; |
| Carlos Tapia | 11/5/2023 | 4.5 | 3.27 - 8:00pm; Commenting the codebase; cleaning up the entire codebase components, pages, etc; |

### d. EMMA NUNNINK

| Name | Date | Hours | Full Description |
|---|---|---|---|
| Emma Nunnink | 9/12/2023 | 1.5 | Met with group; Discussed features, deadlines, responsibilities, set up |
| Emma Nunnink | 9/13/2023 | 1.5 | Met with group; Discussed deadlines; Set up Asana; Set up group youtube channel |
| Emma Nunnink | 9/14/2023 | 3 | Met with group; Started creating front end with html/css |
| Emma Nunnink | 9/15/2023 | 2.5 | Worked on HTML/CSS for frontend |
| Emma Nunnink | 9/16/2023 | 2 | Met with group; Solidified plan for video demo |
| Emma Nunnink | 9/17/2023 | 9 | Filmed and edited entirety of video Demo |
| Emma Nunnink | 9/18/2023 | 8 | Finished and posted Demo video to Panopto and Youtube |
| Emma Nunnink | 9/19/2023 | 2 | Met with group; Went over plans for future presentations |
| Emma Nunnink | 9/29/2023 | 1 | Met with group; Discussed upcoming deadlines and progress in requirements, plan, and design documents |
| Emma Nunnink | 10/2/2023 | 1 | Met with group; Finalized deadlines for all material needed in project demo and set times for when video progress should begin |
| Emma Nunnink | 10/6/2023 | 8 | Began video editing process on Requirements portion of RDP demo; Worked with team members to create recordings and figure out video structure |

| | | | |
|---|---|---|---|
| Emma Nunnink | 10/7/2023 | 10 | Worked on script for Design portion of RDP demo; Finalized edits for Requirements and collected footage for other parts of the video |
| Emma Nunnink | 10/8/2023 | 12 | Finalized edits for Design portion; Recorded and edited Planning portion; Uploaded RDP demo to Panopto and Youtube |
| Emma Nunnink | 10/10/2023 | 2 | Met with group; Discussed future plans for frontend development |
| Emma Nunnink | 10/17/23 | 2 | Weekly meeting; Reviewed frontend progress and implementations that need to be made for Preview demo |
| Emma Nunnink | 10/23/2023 | 1 | Watched tutorials for DaVinci Resolve |
| Emma Nunnink | 10/24/2023 | 2 | Weekly meeting; Discussed what needs to be completed for preview demo |
| Emma Nunnink | 10/25/2023 | 1.5 | Wrote script for planning section of Preview Demo |
| Emma Nunnink | 10/26/2023 | 4.5 | Met with group; Wrote scripts for intro, requirements and design portions of Preview Demo; Edited planning section script |
| Emma Nunnink | 10/27/2023 | 8 | Edited intro, requirements, design, and planning sections of video |
| Emma Nunnink | 10/28/2023 | 6 | Edited preexisting sections to release quality |
| Emma Nunnink | 10/29/2023 | 6 | Finished editing preview demo and posted to D2L |
| Emma Nunnink | 11/7/2023 | 2 | Met with group to go over what needs to be finished for project booklet; updated CrabbyTasks readme |

### e. GRACIELA JIMENEZ

| Name | Date | Hours | Full Description |
|---|---|---|---|
| Graciela Jimenez | 9/12/2023 | 1.5 | Met with group; Role Distribution; Project topic development; Establish communication |
| Graciela Jimenez | 9/13/2023 | 1.75 | Met with group; Discussed features, deadlines, responsibilities; Set up Asana |
| Graciela Jimenez | 9/14/2023 | 1 | AWS Assigment |
| Graciela Jimenez | 9/15/2023 | 1.5 | Group meeting, overview of task to be completed, emphasized on roles. |
| Graciela | 9/16/2023 | 2 | Edit project booklet |

| | | | |
|---|---|---|---|
| Jimenez | | | |
| Graciela Jimenez | 9/17/2023 | 1 | Team meeting |
| Graciela Jimenez | 9/18/2023 | 1 | Discussed team roles as well as what time works best to meet. We also discussed the baseline plan in regard to the front and back end of the concept. |
| Graciela Jimenez | 9/19/2023 | 2 | Team meeting discussion on requirements, design, and planning. As well as the client's specifications on how the project will be implemented. |
| Graciela Jimenez | 9/20/2023 | 1 | Online research on the best project manual templates with a focus to be able to collaboration with the team. |
| Graciela Jimenez | 9/21/2023 | 2 | worked on client proposition to incorporate the team's project components. |
| Graciela Jimenez | 9/22/2023 | 0.75 | The team meeting discussed the requirement document and subtasks for the responsibilities and upcoming do to task in ASANA |
| Graciela Jimenez | 9/23/2023 | 1 | The team meeting discussed RDP, tasks that need to be filled, discussed group responsibilities to make sure work is completed when assigned |
| Graciela Jimenez | 9/24/2023 | 1.5 | Team meeting after class to discuss further tasks that need to be completed for the requirements, such as bad client/nad developer and any other updates for the planing and design |
| Graciela Jimenez | 9/25/2023 | 2 | Team discussion on tasks that need to be completed to be able to submit the project. Disscused upcoming tasks that are in progress. |
| Graciela Jimenez | 9/26/2023 | 0.5 | Group meeting, an overview of the task to be completed for backend and front end requirements |
| Graciela Jimenez | 9/27/2023 | 1 | Met with Benny to discuss the project booklet as well as the ASANA tasks. we also discussed new project template for time logs |
| Graciela Jimenez | 10/17/2023 | 2.25 | weekly meeting for class lectures, we later regroup for team meeting and discussed further tasks that need to be completed |
| Graciela Jimenez | 10/23/2023 | 0.5 | worked on time log made changes and messaged benny about new format for time logs |
| Graciela Jimenez | 10/24/2023 | 2 | The team meeting discussed tasks that need to be completed for the following team meeting as well as individual meetings. |
| Graciela Jimenez | 10/26/2023 | 1 | Meet with the group to discuss future tasks that needed to be completed to submit the demo as we go over current tasks that are still in progress |
| Graciela Jimenez | 10/29/2023 | 1 | Group meeting to finalise all tasks that needed to be completed for the demo and video requirements to be submitted |
| Graciela Jimenez | 10/29/2023 | 2 | worked on the project booklet added sections to the overview as well as completed task |

| | | | |
|---|---|---|---|
| Graciela Jimenez | 10/30/2023 | 2 | Worked on project booklet added more information regarding the structure, goals, views, and design |
| Graciela Jimenez | 10/31/2023 | 1.5 | Team meeting we discussed the project booklet and the information that needs to be prioritized in it, we also discussed upcoming assignments that need to be completed. |
| Graciela Jimenez | 11/3/2023 | 0.5 | Met with Penny and Benny to discuss the project booklet in regard to the requirements section as well as the rest of the information that is required. |
| Graciela Jimenez | 11/5/2023 | 2 | Worked on the project booklet and added more information in regard to the design and meeting notes, worked on the format. |
| Graciela Jimenez | 11/6/2023 | 2 | Worked on the project booklet and researched the financial section that needs to be included. added the financial section but still needs revision not fully completed. |
| Graciela Jimenez | 11/7/2023 | 1 | Reviewed group meeting notes on the red lobster meeting recordings but still in progress to watch for the the rest of the recordings |
| Graciela Jimenez | 11/8/2023 | 3 | Reviewed weekly group meeting recordings for note verification and any information that was discussed that needs to be included in the project booklet for the meeting section |

### f. JULIAN HERNANDEZ

| Name | Date | Hours | Full Description |
|---|---|---|---|
| Julian Hernandez | 9/12/2023 | 1.5 | Met with group; Role Distribution; Project topic development; Establish communication |
| Julian Hernandez | 9/13/2023 | 2 | Met with the group; Talked about project details, contents of the demo and individual tasks, and deadlines; Created tasks with the group for Asana board; Implementation / Design Discussion for demo and overall project |
| Julian Hernandez | 9/13/2023 | 3.7 | Worked on AWS Assignment |
| Julian Hernandez | 9/13/2023 | 1.5 | Worked on Group Admin Website. Has links to time logs, carry water, youtube, and task board. Also has project overview and planned features. |
| Julian Hernandez | 9/14/2023 | 1 | Worked on Group Admin Website. Created a new page for group resources specifically, added project worksite link, and added link to Capstone Course Website! |
| Julian Hernandez | 9/15/2023 | 1.5 | Watched recording of group meeting. Worked on Group Admin Website. Create a new page for the Demo Project Resources and WIP specifically, added an inline view of the Project Design Booklet, added |

| | | | |
|---|---|---|---|
| | | | link to the Requirements Document, and added link to Google Drive with recorded group meetings. |
| Julian Hernandez | 9/15/2023 | 1.75 | Created a github repo and made a how-to file for the group. |
| Julian Hernandez | 9/15/2023 | 3 | Created skeleton for backend software and uploaded it to github. |
| Julian Hernandez | 9/15/2023 | 1.5 | Made some changes to requirements document and helped with github issues. |
| Julian Hernandez | 9/16/2023 | 1 | Getting started quiz and configuring django server on local machine. |
| Julian Hernandez | 9/16/2023 | 3.5 | Discussed future plans after the project demo with some teammates, researched different frameworks than Django that we could use, and helped make webpages for the demo. |
| Julian Hernandez | 9/16/2023 | 2 | Met with the group and talked about remaining considerations for project demo, reviewed and revised requirements document, and discussed our plan for after project demo. |
| Julian Hernandez | 9/16/2023 | 0.5 | 12:30 - 12:53pm Discord w/ Austin & Carlos |
| Julian Hernandez | 9/17/2023 | 0.5 | Attempts to migrate wordpress site to AWS host. |
| Julian Hernandez | 9/17/2023 | 1 | Changes to project site task page. |
| Julian Hernandez | 9/17/2023 | 1 | Team meeting going over our concept demo and adding finishing touches to certain documents. |
| Julian Hernandez | 9/19/2023 | 2 | Class Meeting - Demo Feedback; Met with group; Finalized Roles; Discussed schedule for meeting days/times; Chose Django as backend framework |
| Julian Hernandez | 9/20/2023 | 1.5 | Worked on crabby tasks and configured the input task fields to be more clear along with structuring the HTML page to better suit further functional additions. |
| Julian Hernandez | 9/20/2023 | 2 | Meeting with Team; Discuss following week requirements for requirements, design, planning demos; Changed the Asana boards; Created Carry Water tasks. |
| Julian Hernandez | 9/21/2023 | 0.5 | Added new links, client proposition and grading checklists, to resource website for RDP project. |
| Julian Hernandez | 9/21/2023 | 0.5 | Made some additions to the proposition section of the RDP proposal document and researched some design diagrams. |
| Julian | 9/21/2023 | 1 | Created framework and doc for RPD - Requirements doc and imported |

| | | | |
|---|---|---|---|
| Hernandez | | | some on the list items we had in the demo req. doc |
| Julian Hernandez | 9/22/2023 | 2 | Worked on the backend of webapp and created functional update task and delete task buttons. |
| Julian Hernandez | 9/22/2023 | 0.75 | Watched recording of team meeting where they discussed the requirement document and subtasks for the responsibilities and upcoming do to task in ASANA |
| Julian Hernandez | 9/22/2023 | 1.5 | Added state documentation (transitions, modes, and testing information) to the RDP - requirements document. |
| Julian Hernandez | 9/23/2023 | 1.5 | Added technical approach/problem solving strategies to our RDP - Requirements document. |
| Julian Hernandez | 9/24/2023 | 2 | Met with the group and went over progress on the RDP - Requirements document and design sheets. Also set priorities for the next few days and created many tasks for the group to do. |
| Julian Hernandez | 9/24/2024 | 0.75 | Added to the Documentation section of the RDP - Requirements document. |
| Julian Hernandez | 9/25/2023 | 1 | Finished AWS assignment and completed quiz. |
| Julian Hernandez | 9/25/2023 | 1 | Finished Documentation section of RDP - Requirements document and researched other frontend API that we could add to our project. |
| Julian Hernandez | 9/26/2023 | 1.5 | Filled out some of the RDP - Requirements matrix with bad client and bad dev attacks. |
| Julian Hernandez | 9/26/2023 | 1.5 | Met with group; Assigned tasks with deadlines; Reviewed Requirements doc, planning, and design and what still needs to be completed for each; Took on task to review requirements matrix as bad client and bad dev. |
| Julian Hernandez | 9/27/2023 | 0.5 | Watched planning video and took the planning quiz for class. |
| Julian Hernandez | 9/27/2023 | 1.5 | Added bad client and bad dev attacks and signed for the functional requirements in the requirements matrix and signed off on them. |
| Julian Hernandez | 9/28/2023 | 1 | Added bad client and bad dev attacks and signed for the state/mode requirements in the requirements matrix and signed off on them. |
| Julian Hernandez | 9/29/2023 | 0.5 | Met with group and touched base about using the planning spreadsheet along with talking about the progress made on design documents. |
| Julian Hernandez | 9/30/2023 | 0.5 | Added in system requirements to the requirements matrix sheet. Also, added planning spreadsheet to group admin site. |
| Julian Hernandez | 10/3/2023 | 1.5 | Met with group after class and discussed progress as well what modules needed to be added to the plan document, what items need to be completed on the requirements document and matrix, and what our deadlines were for having a stable version ready to present in the RDP |

| | | | |
|---|---|---|---|
| | | | demo. |
| Julian Hernandez | 10/3/2023 | 0.5 | Added bad client/bad developer notes on the requirements matrix and updated the matrix with some revisions to requirements. |
| Julian Hernandez | 10/4/2023 | 2 | Created implementation plan for front-end and back-end with almost all of the macro-details each side will need to complete. |
| Julian Hernandez | 10/5/2023 | 0.5 | Added in some more bad client/bad dev notes to the requirements matrix. Also fixed some wording in the requirements document. |
| Julian Hernandez | 10/06/2023 | 1 | Added many more items to the testing matrix and requirements matrix. |
| Julian Hernandez | 10/06/2023 | 7 | Recorded voicing for the Requirements section of the RDP - Demo. |
| Julian Hernandez | 10/08/2023 | 2 | Helped write a script for the planning section of the demo. |
| Julian Hernandez | 10/08/2023 | 1 | Met with the group and talked about the RDP - demo progress along with taking the admin quiz |
| Julian Hernandez | 10/8/2023 | 1 | Reviewed RDP demo before upload with group. |
| Julian Hernandez | 10/10/2023 | 2 | Met with group; Discussed future plans for frontend development |
| Julian Hernandez | 10/14/2023 | 1.5 | Discussed backend implementation tasks with Austin and Carlos, and frontend implementation tasks with Qi. |
| Julian Hernandez | 10/16/2023 | 0.5 | Pulled new front-end skeleton for our project, got it to run on local machine, and listened to recording of group meeting. |
| Julian Hernandez | 10/17/2023 | 2 | Met with class and talked with group about the quiz, our upcoming goals and tasks for the preview, and what everyone would be doing. |
| Julian Hernandez | 10/19/2023 | 2 | Watched lecture materials and took the quiz for class. |
| Julian Hernandez | 10/20/2023 | 3 | Watched hour long video on React basics and created landing page for the website. |
| Julian Hernandez | 10/22/2023 | 1.5 | Created many new tasks on Asana and adjusted the dashboard/home page of the website to reflect the data that we will have on our website. |
| Julian Hernandez | 10/22/2023 | 1.5 | Met with group to discuss progress on website, to discuss presentation requirements, and to decide what needed to come next. |
| Julian Hernandez | 10/24/2023 | 1 | Met with Carlos to fix a bug in my .env that was preventing me from logging into our website for development |
| Julian Hernandez | 10/24/2023 | 3 | Worked on making website dashboard pull values from the database task information in order to reflect the user's data. |

| | | | |
|---|---|---|---|
| Julian Hernandez | 10/26/2023 | 8 | Worked on making task manager have specific tasks for every individual date, on adding indicators for days with tasks, on making the tasks that appear on the dates be editable/deletable, and on making task requests based on date seamless between any of the views. |
| Julian Hernandez | 10/26/2023 | 2 | Met with group to talk about what needed to be done for preview recording and about what needed to be done for documents, plan, design, and final site changes. |
| Julian Hernandez | 10/27/2023 | 4 | Worked on making indicators only show for days that have tasks, on making the given tasks values appear on the edit form, and on making sure the site has no major implementation flaws. |
| Julian Hernandez | 10/28/2023 | 1 | Worked on making final changes and merges for the production push for the website. All processes are now async and nothing crashes. |
| Julian Hernandez | 10/31/2023 | 2 | Met with group for class and tuesday meeting, discussed project booklet requirements and what resources the group needed to complete it. |
| Julian Hernandez | 11/5/2023 | 2 | Met with group after class and discussed what was left to do for the project booklet. We also discussed the remaining other assignments in the class. |
| Julian Hernandez | 11/10/2023 | 4.5 | Created design documents for task manager, dashboard, profile, user, and settings. |

### g. NICK DILALLO

| Name | Date | Hours | Full Description |
|---|---|---|---|
| Nick Dilallo | 9/12/2023 | 1.5 | Met with group; Role Distribution; Project topic development; Establish communication |
| Nick Dilallo | 9/13/2023 | 1.75 | Met with group; Discussed features, deadlines, responsibilities; Set up Asana |
| Nick Dilallo | 9/13/2023 | 1 | AWS Assignment |
| Nick Dilallo | 9/14/2023 | 1 | Met with group; Reassessed individual responsibilities/roles; Solidify understanding of concept demo requirements |
| Nick Dilallo | 9/14/2023 | 3 | AWS Assignment |
| Nick Dilallo | 9/15/2023 | 1.5 | Created Test Matrix Spreadsheet for Demo testing criteria |
| Nick Dilallo | 9/15/2023 | 2 | Worked on front end for webpage; Progress on Home Page and attempt to create main calendar |

| | | | |
|---|---|---|---|
| Nick Dilallo | 9/16/2023 | 2 | Worked on front end for webpage; Made contact page; Attempt to make TM |
| Nick Dilallo | 9/16/2023 | 2 | Met with group; Discussed work progress and deadlines; Planning for presentation videos; Assign jobs for what needs to be completed; Discussed future plans for real project |
| Nick Dilallo | 9/17/2023 | 1.5 | Created new Test Matrix Spreadsheet for the Demo specifically; Added visual tests for each page; Added functional tests for each page; |
| Nick Dilallo | 9/19/2023 | 2 | Class Meeting - Demo Feedback; Met with group; Finalized Roles; Discussed schedule for meeting days/times; Chose Django as backend framework; Split frontend/backend teams |
| Nick Dilallo | 9/21/2023 | 1.5 | Met with group; discussed progress completed from yesterday to today; touch base with requirements and design managers as a team; determine priorities of tasks and what to keep in mind; |
| Nick Dilallo | 9/22/2023 | 1.5 | Completed AWS Assignment |
| Nick Dilallo | 9/26/2023 | 1.5 | Met with group; Assigned tasks with deadlines; Reviewed Requirements doc, planning, and design and what still needs to be completed for each; Took on task for testing matrix creation |
| Nick Dilallo | 10/2/2023 | 1 | Review requirements document progress; client/dev attacks doc progress; |
| Nick Dilallo | 10/3/2023 | 1.5 | Meet with team; discuss progress; review and refine documents with team; determine upcoming priorities and deadlines |
| Nick Dilallo | 10/3/2023 | 1.5 | Created Testing Matrix; Determined Outline/Plan for testing; Inserted all item numbers for required functionality for testing |
| Nick Dilallo | 10/5/2023 | 1.5 | Met with team; Determine final steps for RDP presentation; Discuss/review requirements for video; Updated requirements document |
| Nick Dilallo | 10/5/2023 | 1 | Worked on Testing Matrix; Began writing test descriptions/expected results; |
| Nick Dilallo | 10/6/2023 | 2 | Testing Matrix Work; Tested multiple requirements; Developed way of handling duplicate tests; Added Revisions |
| Nick Dilallo | 10/8/2023 | 1 | Met with group; Discussed all final matters for RDP demo; Planning section scripting; |
| Nick Dilallo | 10/10/2023 | 1.5 | Met with group; Discussed generic SQL implementation and connections with frontend; Ideas for frontend - Use template?; Switch to React for frontend; Pages to be made/priorities |
| Nick Dilallo | 10/14/2023 | 1.5 | Selecting Frontend template; Attempting to open on local machine; running into errors; |
| Nick | 10/15/202 | 0.5 | Meet with group; Discuss progress; Decided on ideal frontend template if |

| | | | |
|---|---|---|---|
| Dilallo | 3 | | possible |
| Nick Dilallo | 10/15/2023 | 1.5 | Meet with Carlos about frontend; Discuss what to keep from template/remove; Functionality/features on each page; Made notes on possible tasks to be divided |
| Nick Dilallo | 10/16/2023 | 2 | Deepen understanding of the template's layout/format; Begin updating to fit CrabbyTasks; Change search bar in header; Add/update additional pages/routes |
| Nick Dilallo | 10/17/2023 | 3 | Requirements reviewing; bad dev/client; weekly meeting |
| Nick Dilallo | 10/20/2023 | 2 | Frontend development; Learn layout of React template; Created and renamed new pages; Remove some unnecessary features |
| Nick Dilallo | 10/21/2023 | 3.5 | Work on Task Manager Page; Add weekly and single day view; Selection of a date causes modal to appear; |
| Nick Dilallo | 10/22/2023 | 1.5 | Attempt to add functional calendar; Found import from online; Attempt to blend the two css styles |
| Nick Dilallo | 10/24/2023 | 2 | Weekly meeting; Planning for demo; Progress review; Discuss what is complete and what needs to be done for demo |
| Nick Dilallo | 10/29/2023 | 1 | Discuss Preview Demo completion; PM Quiz for demo; Discuss what we're missing |
| Nick Dilallo | 10/31/2023 | 1.5 | Meeting with the team; Discuss Booklet; Final Steps |
| Nick Dilallo | 11/7/2023 | 1.5 | Meeting with team; Discuss missing components of project booklet; Github cleanup |

### h. PENNY MCMAHON

| Name | Date | Hours | Full Description |
|---|---|---|---|
| Penny McMahon | 9/12/2023 | 1.5 | Met with group; Role Distribution; Project topic development; Establish communication |
| Penny McMahon | 9/13/2023 | 1.75 | Met with group; Discussed deadlines; Set up Asana; Distributed tasks; Planned upcoming meetings; |
| Penny McMahon | 9/14/2023 | 3 | Met with group; Requirement Document; Reviewed Requirements information individually |
| Penny McMahon | 9/15/2023 | 0.5 | Minor Requirements document edits; |
| Penny McMahon | 9/16/2023 | 1.5 | Requirements document table of contents and cover; Minor edits; Studied Requirements; |

| | | | |
|---|---|---|---|
| Penny McMahon | 9/17/2023 | 1.5 | Met with group; finalized requirements; reviewed planning; |
| Penny McMahon | 9/19/2023 | 2 | Met with group; group roles; going forward decisions; |
| Penny McMahon | 9/20/2023 | 2 | Met with group; redid Asana; distributed tasks; discussed RDP presentation; Planning around calendar; |
| Penny McMahon | 9/21/2023 | 2 | researched timeline software; created requirements matrix; research planning and requirements from videos; |
| Penny McMahon | 9/21/2023 | 0.75 | met with group; acquisitioned for tasks; organized tasks for requirements document; |
| Penny McMahon | 9/22/2023 | 2 | Researched requirement references; added components to the requirements sheet like project responsibilities and stakeholders; filled in project planning requirements; started introduction; |
| Penny McMahon | 9/24/2023 | 2 | Met with group; reviewed requirements and designs; |
| Penny McMahon | 9/24/2023 | 1.5 | Looked into requirements layouts; Filled in Requirements matrix; |
| Penny McMahon | 9/26/2023 | 1.5 | Project meeting; task set up; |
| Penny McMahon | 9/27/2023 | 2 | Took the planning and requirements quizzes; Added tasks to the Asana board; Edited deadlines; |
| Penny McMahon | 9/28/2023 | 2 | Started converting the plan to an organized spreadsheet to help with organization and overall presentation; |
| Penny McMahon | 9/29/2023 | 3 | Continued adding to the spreadsheet plan to ensure that it is robust; Put forth module system of organization; Created calendar in the spreadsheet; made plans for the critical path |
| Penny McMahon | 9/30/2023 | 1 | watched meeting recording; (Soft tine) thought up the critical path and overall layout for aspects of the plan including the calendar; |
| Penny McMahon | 10/1/2023 | 2 | Added the critical path and various deadlines to the plan;filled in the calendar with present tasks; |
| Penny McMahon | 10/2/2023 | 2 | Cleaned up and restructured requirements; met with group to discuss module system; Added new tasks to plan |
| Penny McMahon | 10/3/2023 | 3 | Added project management, user, and testing requirements; met with group to discuss the next week; Editted requirement table of contenets; |
| Penny McMahon | 10/5/2023 | 4 | Continued to prune and work on the plan; configured individual view; met with group to discuss presentation |
| Penny | 10/6/2023 | 3 | Finishing touches on the dependency diagram and planning sheet; |

| | | | |
|---|---|---|---|
| McMahon | | | individual view of calendar; recording for presentation; hypothetical; |
| Penny McMahon | 10/8/2023 | 2 | Planning Script; met with group to discuss admin quizzes; |
| Penny McMahon | 10/10/2023 | 1.5 | weekly team meeting; planning for the next few weeks; |
| Penny McMahon | 10/12/2023 | 2 | Met with groups reformatted and expanded requirements matrix |
| Penny McMahon | 10/13/2023 | 1 | continued expanding requirements |
| Penny McMahon | 10/15/2023 | 2 | Met with group; expanded requirements; review certain requirements; |
| Penny McMahon | 10/16/2023 | 2 | Requirements testing and expanding the matrix with bad-client/dev reviews; |
| Penny McMahon | 10/17/2023 | 3.5 | Requirements reviewing; bad dev/client; writing new requirements; weekly meeting |
| Penny McMahon | 10/18/2023 | 1.5 | Requirements bad client/dev testing; |
| Penny McMahon | 10/19/2023 | 2 | Reformatted frontend planning; continued requirements tesing |
| Penny McMahon | 10/20/2023 | 1.5 | requirements testing for software and technical requirements; time logs; |
| Penny McMahon | 10/23/2023 | 0.5 | requirements reviewing and revising for the project planning requirements; Overall cleaning up of requirements; |
| Penny McMahon | 10/24/2023 | 2 | Team Meeting; Planning for the next week; revieing progress; |
| Penny McMahon | 10/25/2023 | 3 | Requirements reviewing; finalizing the plan; finalizing dependency graph |
| Penny McMahon | 10/26/2023 | 2 | Copying confirmed, reviewed, and test requirements into the final requirements sheet; go over script for planning; |
| Penny McMahon | 10/27/2023 | 2 | Add any final adjustments to plan and requirements sheets; record video for the planning portion of the presentation; |
| Penny McMahon | 10/29/2023 | 1 | met with group to discuss progress on the presentation; |
| Penny McMahon | 10/31/2023 | 1.5 | met with group to discuss how to proceed with booklet |

### i. QI DENG

| Name | Date | Hours | Full Description |
|---|---|---|---|
| Qi Deng | 9/12/2023 | 1.5 | Met with group; Role Distribution; Project topic development; Establish communication |
| Qi Deng | 9/13/2023 | 2 | Met with group; Discuss deadlines; Sharing project ideas; Set up Asana; Planning future meetings; Assign jobs |
| Qi Deng | 9/13/2023 | 2.5 | AWS Assignment |
| Qi Deng | 9/14/2023 | 1.5 | Red Lobster Website UI Design |
| Qi Deng | 9/16/2023 | 2.5 | UML Diagram and Functional Designs |
| Qi Deng | 9/16/2023 | 3.5 | Django Study |
| Qi Deng | 9/16/2023 | 2 | Team meeting; Discussed work progress and deadlines; Planning for presentation materials; Assign jobs |
| QI Deng | 9/19/2023 | 1.5 | Met with group members, assigned jobs for the upcoming week, and decide work priority. |
| Qi Deng | 9/19/2023 | 4 | Making the production guidance for better work organization for later production stages. |
| Qi Deng | 9/20/2023 | 3 | Make diagrams for the design demo including ( |
| Qi Deng | 9/22/2023 | 2.5 | Making progress on the production guidance demo, and start working on design requirements on the requirement document. |
| Qi Deng | 9/26/2023 | 1.5 | Group meeting, review the RDP demo document progress, discuss materials needed. |
| Qi Deng | 9/29/2023 | 2.5 | Finalize the diagram (Level 0, level 1, module function relationship, etc.) Edit the design demo booklet. |
| Qi Deng | 10/3/2023 | 1.5 | Meet with team; discuss progress; review and refine documents with team; determine upcoming priorities and deadlines |
| Qi Deng | 10/7/2023 | 2 | Making the design presentation website. |
| Qi Deng | 10/8/2023 | 2.5 | Make two level 2 diagrams with the code that performs those specific tasks. |
| Qi Deng | 10/10/2023 | 1.5 | Class meeting and team meeting. Broke up into two different groups: frontend and backend. Talked about tasks that need to be completed in the upcoming week. |
| Qi Deng | 10/14/2023 | 0.5 | Attend frontend team meeting, discussed fontend tasks going ahead. |
| Qi Deng | 10/16/2023 | 2 | Starting to write html code for the website pages. |

| | | | |
|---|---|---|---|
| Qi Deng | 10/24/2023 | 2 | Team meeting; Discussed work progress and deadlines. |
| Qi Deng | 10/26/2023 | 1 | Team meeting, review some completed work, discuss areas that need improvement. |