CSE 360 Project Report Number 5

Team <W21>

Team Member Names:

1. Aditya Maheshbhai Gabani
2. Viet Le
3. Vatsal Maniar
4. Smrita Pandey
5. Sumit Patel
6. Shivansh Shrivastava

# Table of Contents

[Table of Contents i](#_heading=h.gjdgxs)

[1.](#_heading=h.30j0zll) Executive Summary 1

[2.](#_heading=h.1fob9te) Concept of Operations 2

[3.](#_heading=h.3znysh7) Requirements Definition 3

[3.1.](#_heading=h.2et92p0) Overview 3

[3.2.](#_heading=h.tyjcwt) Context 3

[3.2.1.](#_heading=h.3dy6vkm) Sequence Diagrams 3

[3.3.](#_heading=h.1t3h5sf) Use Cases 3

[3.3.1](#_heading=h.4d34og8) Overview 3

[3.3.2](#_heading=h.2s8eyo1) Use Case Details 3

[3.4.](#_heading=h.17dp8vu) Software Requirements Documentation 3

[4.](#_heading=h.3rdcrjn) Architecture 4

[4.1.](#_heading=h.26in1rg) Architecturally Significant Tradeoffs 4

[4.2.](#_heading=h.lnxbz9) Architectural Analysis 4

[4.3.](#_heading=h.35nkun2) Logical and Structural Decomposition 4

[5.](#_heading=h.1ksv4uv) Design 5

[5.1.](#_heading=h.44sinio) Design Goals 5

[5.2.](#_heading=h.2jxsxqh) Design Elements 5

[5.3.](#_heading=h.z337ya) Upside and Downside Risks 5

[5.3.1.](#_heading=h.3j2qqm3) Risk Reduction Prototype 1 5

[5.3.2.](#_heading=h.1y810tw) Risk Reduction Prototype 2 … n 5

[5.4.](#_heading=h.4i7ojhp) Design Reuse 5

[5.5.](#_heading=h.2xcytpi) Design Reuse Prototyping 6

[5.5.1.](#_heading=h.1ci93xb) Design Reuse Prototype 1 6

[5.5.2.](#_heading=h.3whwml4) Design Reuse Prototype 2 … n 6

[6.](#_heading=h.2bn6wsx) Final Integrated Prototype 7

[6.1.](#_heading=h.qsh70q) Effort Logger Prototype Goals 7

[6.2.](#_heading=h.3as4poj) Implemented Key Features 7

[6.2.1.](#_heading=h.1pxezwc) Key Feature Demonstration 1 7

[6.2.2.](#_heading=h.49x2ik5) Key Feature Demonstration … n 7

[6.3.](#_heading=h.2p2csry) Unimplemented Key Features 7

[6.3.1.](#_heading=h.147n2zr) Key Feature Not Implemented 1 7

[6.3.2.](#_heading=h.3o7alnk) Key Feature Not Implemented 2 … n 7

[7.](#_heading=h.23ckvvd) Conclusion 8

[8.](#_heading=h.ihv636) Appendix A: Credit Sheet 9

# Executive Summary

The EffortLogger application is a project aimed at helping individuals and organizations track their work and monitor their productivity. The project is currently in its fifth phase, where the team is working on refining the documentation deliverables and creating a final risk reduction prototype.

The application's core functionality includes the ability to log time, create custom tasks, generate reports, and analyze productivity trends. The team has made significant progress in implementing these features and has incorporated user feedback throughout the development process.

In phase 4, the team developed a detailed requirements definition, which included use cases and sequence diagrams. These have been expanded into comprehensive text descriptions, which have helped to guide the development of the prototype.

One of the key challenges in this phase has been managing upside and downside risks associated with the project. To mitigate these risks, the team has produced short outlines of potential risks and will produce detailed descriptions and screencasts for each significant external component used in the application.

Design reuse prototyping is a new section that has been added in this phase. This section is used when the team decides to use significant external components produced by others, such as database or encryption libraries. The team will produce short outlines that will be expanded into text descriptions and screencasts for the final project submission.

The final integrated prototype will align with the architecture and design and will be well documented. The team has made internal comments that specify the authorship, explain how the implementation aligns with the design, and how the results demonstrate that the design goal for this prototype has been satisfied.

Overall, the EffortLogger application is a promising project that addresses a real-world need for individuals and organizations. The team is excited to showcase their progress in this final risk reduction prototype and is committed to delivering a high-quality final product in the next and final phase of the project.

# Concept of Operations

Concept of Operations (ConOps) is an overview of a system's function, capabilities, and objectives. It is a high-level documentation that outlines the overall goals, purpose, and tasks. A system's intended use and how it will function in its environment are both described in this thorough paper. It is also an important document for stakeholders to comprehend the goals of the system and how it will achieve those goals, the ConOps is a crucial tool. This documentation allows people with no technical background to understand the concepts and plans that are being made including stakeholders, users, regulators, and supporting organizations.

**2.1. Solution Concept of Operation**

The ConOps outlines the system's technical specifications and performance characteristics. It describes the functions that the system will perform and provides information on its technical specifications, such as its processing speed, storage capacity, and input/output capabilities. The ConOps also provides performance characteristics, such as response time, reliability, and accuracy, which help to ensure that the system meets the requirements of its users. This lets the users know what to expect from the system while also giving the developers an idea of what they should be accomplishing. Creating a diagram that shows the flow of the major processes and functions allows the project team to keep track of the progress. While allowing the stakeholders, users, regulators, and supporting organizations to follow the progress. The external, operational, and support environments describe what the solution will interact with and the location where the operation of solution will take place. It outlines the physical requirements of the operational environment like the type of facilities, equipment, hardware and software, staff, and any necessary materials required for the solution.

This table describes the high-level processes:

| Attributes | The effort logger tracks the users’ interaction time with a software and records the date and time. |
| --- | --- |
| Process Name | The user interacts with a front end program, a Java/JavaFX application that allows them to record their time and date they worked on their project. The data is then recorded into a backend application storing the information which then allows the employers to see. |
| Purpose | This allows the employers to keep track of their employees’ time and effort, so they can determine the value of their employee to the company. This helps the company save money from hiring too many or too few employees in order to get the work they need to get done. |
| Description | Describe the process at a high-level. The more granular the level detail should appear within the operational scenarios |
| Priority | Rank the priority |
| Frequency | Identify metrics and/or expectations for the high-level process |

*This table was taken from the CONOPS Template Example*

**2.2. User Classes and Operational Scenarios**

The duties of the system and its supporting entities are outlined in the ConOps. It gives details on the human roles and duties as well as the support organizations in charge of running and maintaining the system. This data is essential for making sure the system is run and maintained correctly, and that users are supported as required. Have in depth detailed explanations of each major process in the system. Give examples of how the solution works and whether it satisfies the users' needs. For the scenarios run, it should show the inputs that will trigger the system, how the user will interact with it, and what output should be generated for the system. Implement multiple test cases with success and failure scenarios, including any setbacks during the test run.

**2.3. Assumptions and Constraints**

Provide details of any assumptions and constraints that may affect the solutions’ process. Provide information on the new system’s development, implementation, requirements, maintenance, and retirement of the solution. Outlines the steps that will be taken to develop and implement the system, as well as the strategies that will be used to maintain and sustain it over its lifecycle. This ensures that the system is developed, implemented, and maintained effectively.

**2.4. Conclusion**

In conclusion, the Concept of Operations (ConOps) is a thorough document that describes the function, potential, and objectives of a system. It offers a precise and succinct knowledge of the intended use of the system and how it will interact with its surroundings. Users, stakeholders, regulators, and support organizations may all benefit from using the ConOps to better understand the goals of the system and how it will achieve them. The ConOps contributes to ensuring that the system is used effectively and efficiently and that it produces the required results by delivering this information. The ConOps is very important and efficient in the communications between the development team and the users, stakeholders, regulators, and support organizations. It also serves as a guideline for both sides.

Accurate time tracking is a vital aspect of project management, especially for large-scale projects such as those managed by NASA. To achieve this, EffortLogger, a time-tracking application, is integrated into existing tools used by stakeholders. The EffortLogger integration requirements are outlined in the NASA Concept of Operations Annotated Outline.

Firstly, EffortLogger should be integrated with the existing tools used by the stakeholders. The integration should be seamless, requiring minimal effort from the user. This ensures that all work hours spent on NASA projects are accurately captured without adding any extra burden on the user.

Secondly, the integration should allow for real-time data transfer. This enables stakeholders to make informed decisions based on up-to-date information. The real-time data transfer also ensures that any changes made by stakeholders are immediately reflected in the data captured by EffortLogger.

Thirdly, the integration should allow for the easy retrieval of data to enable reporting and analysis. Non-technical stakeholders should be able to retrieve data easily to enable reporting and analysis, ensuring that the project is on track and that decisions can be made based on reliable information.

Fourthly, the integration should be secure and comply with NASA security protocols. This is to ensure that the data captured by EffortLogger is secure and cannot be accessed by unauthorized personnel. Compliance with NASA security protocols also ensures that the integration is in line with the organization's standards, reducing the risk of security breaches.

In conclusion, EffortLogger integration is an essential aspect of accurate time tracking for NASA projects. The integration requirements outlined in the NASA Concept of Operations Annotated Outline ensure that stakeholders are aware of the need for EffortLogger integration and the expectations set for all stakeholders, both technical and non-technical. By working together and complying with NASA security protocols, the integration can be successfully implemented, and accurate time tracking can be achieved.

# Requirements Definition

## Overview

EffortLogger is a web-based application that provides users with an efficient and user-friendly way to track their tasks' time and effort accurately. This system is intended to help users manage their work better, plan their time more effectively, and make informed decisions based on their productivity and efficiency. The following subsections provide a high-level overview of the functional and non-functional requirements of the EffortLogger system.

**Functional Requirements:**

The functional requirements are the features and capabilities that the system must provide to its users. The following is a list of the functional requirements of the EffortLogger system:

1. User Management: The system provides user registration, authentication, and authorization functionalities to ensure that only authorized users can access the system and its features.
2. Task Management: The system enables users to create, edit, and delete their tasks. It should also allow them to track the time spent on each task and manage their workloads efficiently.
3. Reporting: The system provides users with the ability to generate reports based on their task data. These reports should include details such as the time spent on each task, the amount of effort expended, and the overall productivity of the user.
4. Integration: The system should be integrated with other productivity tools such as Google Calendar, Trello, and Asana, to enhance its usability and improve users' efficiency.
5. Security: The system must ensure that user data is secure and protected from unauthorized access. It must also provide backup and recovery functionalities to ensure that user data is not lost due to system failures or other unforeseen circumstances.

**Non-Functional Requirements:**

The non-functional requirements are the quality attributes that the system must possess. These include performance, usability, reliability, security, and maintainability. The following is a list of the non-functional requirements of the EffortLogger system:

1. Performance: The system must be responsive and provide users with a seamless and efficient experience. The system should have low response times, and all features must function without significant delays.
2. Usability: The system must be intuitive and user-friendly, with clear and concise instructions and documentation. It should also have an appealing interface that is easy to navigate.
3. Reliability: The system must be reliable and available to users at all times, with minimal downtime or maintenance periods. It should also be resilient to system failures and data losses.
4. Security: The system must have robust security measures to ensure that user data is secure and protected from unauthorized access. It should also comply with industry standards for data protection and privacy.
5. Maintainability: The system must be easily maintainable, with simple and efficient mechanisms for updating and upgrading the software. It should also have clear and concise documentation for maintenance and troubleshooting purposes.

## Context

The system in context is a web-based time tracking application called EffortLogger. The application will be accessed through a web browser and will allow users to track their time spent on various projects and tasks. The system will be designed to support multiple users, with each user having their own account and associated data.

### Sequence Diagrams

Sequence diagrams are a type of interaction diagram that describes the interactions between objects in a system in a chronological order. They are used to model the dynamic behavior of a system, and they can be used to represent complex scenarios that involve multiple objects and actors.

The most significant high-level sequence diagrams for this system will include:

1. User Authentication: This sequence diagram will depict the flow of events when a user tries to authenticate into the system. It will show the interactions between the user, the authentication server, and the database.
2. Add Task: This sequence diagram will show the interactions between the user and the system when a new task is added to the system. It will depict the flow of events from when the user requests to add a task to when the task is stored in the database.
3. Update Task: This sequence diagram will show the interactions between the user and the system when a task is updated. It will depict the flow of events from when the user requests to update a task to when the updated task is stored in the database.
4. View Task: This sequence diagram will show the interactions between the user and the system when a user wants to view a task. It will depict the flow of events from when the user requests to view a task to when the task details are displayed on the user interface.
5. Delete Task: This sequence diagram will show the interactions between the user and the system when a user wants to delete a task. It will depict the flow of events from when the user requests to delete a task to when the task is removed from the database.
6. Search Task: This sequence diagram will show the interactions between the user and the system when a user wants to search for a task. It will depict the flow of events from when the user enters the search criteria to when the matching tasks are displayed on the user interface.

These sequence diagrams will provide a better understanding of how the system operates and how different objects and actors interact with each other. They will also help in identifying potential issues and bottlenecks in the system design.

## Use Cases

### Overview

The use cases represent the functionalities that the system will provide to its users. It is essential to identify all the possible use cases to understand the system's behavior and requirements fully. In this section, we provide an overview of the use cases, along with a supporting high-level Use Case Diagram.

The EffortLogger system's primary purpose is to enable users to track their tasks' time and effort accurately. To achieve this goal, the system will provide the following use cases:

1. Login: This use case enables users to log in to the system by providing valid credentials.
2. Logout: This use case enables users to log out of the system securely.
3. Create Task: This use case enables users to create new tasks by providing the task details, such as name, description, start and end time, and other relevant information.
4. Update Task: This use case enables users to update the existing task details, such as its name, description, start and end time, and other relevant information.
5. Delete Task: This use case enables users to delete the existing task from the system.
6. Start Task Timer: This use case enables users to start the timer for a task once they start working on it.
7. Stop Task Timer: This use case enables users to stop the task timer once they have completed their work.
8. View Tasks: This use case enables users to view the list of all tasks created by them.
9. Generate Report: This use case enables users to generate a report for a specified time range to get a summary of their tasks' time and effort.
10. Set Preferences: This use case enables users to set their preferences for the system, such as their preferred time zone, date format, and other relevant settings.

### Use Case Details

1. Login:

Description: This use case allows the user to log in to the system by providing valid credentials. The system verifies the user's credentials and grants access to the system if they are correct.

1. Logout:

Description: This use case allows the user to log out of the system securely. The system clears the user's session and returns them to the login screen.

1. Create Task:

Description: This use case allows the user to create a new task by providing the task details, such as name, description, start and end time, and other relevant information. The system validates the input and adds the task to the database.

1. Update Task:

Description: This use case allows the user to update the existing task details, such as its name, description, start and end time, and other relevant information. The system validates the input and updates the task in the database.

1. Delete Task:

Description: This use case allows the user to delete an existing task from the system. The system removes the task from the database.

1. Start Task Timer:

Description: This use case allows the user to start the timer for a task once they start working on it. The system starts the timer for the task and records the start time.

1. Stop Task Timer:

Description: This use case allows the user to stop the task timer once they have completed their work. The system stops the timer for the task and records the end time.

1. View Tasks:

Description: This use case allows the user to view the list of all tasks created by them. The system retrieves the tasks from the database and displays them to the user.

1. Generate Report:

Description: This use case allows the user to generate a report for a specified time range to get a summary of their tasks' time and effort. The system retrieves the tasks for the specified time range from the database and generates a report summarizing the time and effort spent on each task.

1. Set Preferences:

Description: This use case allows the user to set their preferences for the system, such as their preferred time zone, date format, and other relevant settings. The system saves the user's preferences in the database.

## Software Requirements Documentation

In this section, we will provide software requirements documentation that captures the shared vision and agreements between the key stakeholders to ensure that each can perform their required tasks as specified in Figure 4.16 of the textbook. This document is not a standalone formal requirements document as described in Figure 4.17. Instead, it will be used in conjunction with the other elements of this deliverable to provide the technical details required for detailed design, design validation, implementation, testing, and operations with continuous integration and continuous deployment.

The software requirements documentation for the EffortLogger system will include the following components:

1. Functional Requirements: These requirements specify the system's functionality, including the features and capabilities that the system must have to fulfill the users' needs. This section will include use cases, user stories, and functional requirements specification documents.
2. Non-Functional Requirements: These requirements specify the system's non-functional characteristics, such as performance, reliability, security, usability, and scalability. This section will include the non-functional requirements specification document.
3. Data Requirements: These requirements specify the data that the system must store, process, and retrieve. This section will include data models, database schema, and data dictionary.
4. System Architecture: This section will describe the system's high-level architecture, including the system components, their interactions, and the system's deployment and scaling strategy.
5. Interface Requirements: These requirements specify the system's external interfaces, including the user interface, API, and integration with other systems.
6. Operational Requirements: These requirements specify the system's operational characteristics, such as availability, maintainability, and disaster recovery.
7. Performance Requirements: These requirements specify the system's performance characteristics, such as response time, throughput, and resource utilization.
8. Security Requirements: These requirements specify the system's security characteristics, such as authentication, authorization, and data protection.
9. Legal and Regulatory Requirements: These requirements specify the legal and regulatory requirements that the system must comply with, such as data privacy, intellectual property, and accessibility.

Together, these components of the software requirements documentation will provide the technical detail required for the EffortLogger system's design, development, testing, deployment, and operation.

# Architecture

In this section, we will update our previous work on architecture and expand it to serve the needs of high-level technical stakeholders, such as requirements engineers, architects, and designers.

## Architecturally Significant Tradeoffs

To ensure that the high-priority quality requirements address the customer expectations, we must make some architecturally significant design tradeoffs. The most critical tradeoffs are as follows:

1. Performance vs. Scalability: The system must handle a large number of concurrent users and tasks. To achieve this, we must decide between optimizing the system's performance or making it more scalable to handle more users and tasks. We chose to focus on scalability as it aligns with the system's primary objective of handling a large number of tasks.
2. Flexibility vs. Security: The system must be flexible enough to allow users to customize it to their preferences, while also being secure enough to protect sensitive information. We decided to prioritize security over flexibility to ensure that users' data is protected from unauthorized access.
3. Modularity vs. Complexity: The system's architecture must strike a balance between modularity and complexity to allow for easy maintenance and updates. We chose to prioritize modularity to ensure that the system can be easily extended and maintained over time.

## Architectural Analysis

To ensure that the system meets its requirements, we conducted an architectural analysis to identify the system's critical components and their interactions. The analysis involved evaluating the system's functional and non-functional requirements, its existing architecture, and its integration with other systems.

The analysis led us to conclude that the system should be designed as a distributed system, with the following key components:

1. User Interface: This component is responsible for interacting with the user and presenting the system's functionality.
2. Task Management: This component is responsible for managing the creation, modification, and deletion of tasks.
3. Timer Management: This component is responsible for managing the timers associated with each task.
4. Reporting: This component is responsible for generating reports based on user input.
5. Data Storage: This component is responsible for storing user and task data.

## Logical and Structural Decomposition

**Architecturally Significant Tradeoffs -**

Architecturally critical design tradeoffs are essential decisions that must be made to ensure that the high-priority quality requirements address the customer expectations. The following are some critical design tradeoffs that need to be considered:

* **Security vs. Usability:** The system must be secure enough to protect sensitive data, but at the same time, it must be easy to use for the end-users. This tradeoff requires a careful balance between security measures and user-friendliness. For instance, the authentication process must be robust enough to ensure the system's security, but it should not be too complicated that it becomes challenging for users to log in.
* **Performance vs. Scalability:** The system must be able to handle increasing amounts of data and users without any significant decrease in performance. However, ensuring high performance often comes at the cost of scalability. For instance, using a specific database may provide faster queries, but it may not be as scalable as a distributed database.
* **Maintainability vs. Functionality:** The system must be easy to maintain, which means it should have a well-structured codebase, good documentation, and clear design patterns. However, adding more functionality to the system may make it more complex, which can reduce maintainability. Thus, it is essential to strike a balance between maintainability and functionality.
* **Cost vs. Quality:** The system must be developed within a specific budget, but it must also meet high-quality standards. This tradeoff requires making decisions about how much to invest in different components of the system. For instance, investing in high-quality hardware may increase costs, but it may also improve system performance and reliability.
* **Flexibility vs. Standardization:** The system must be flexible enough to adapt to changing business requirements, but it must also adhere to certain standards. A highly customized system may be more flexible, but it may be more challenging to maintain and may not integrate well with other systems. Thus, it is important to strike a balance between flexibility and standardization.

In summary, addressing the high-priority quality requirements that meet customer expectations requires carefully considering tradeoffs between security and usability, performance and scalability, maintainability and functionality, cost and quality, and flexibility and standardization. The critical design tradeoffs will ultimately determine the success of the system, and thus, they must be carefully considered and evaluated during the design process.

## Architectural Analysis-

The analysis focused on identifying the high-level requirements of the effort logging and task management system and then breaking them down into more specific components and subcomponents. The goal was to create a system architecture that met the needs of the customer while also ensuring that the system was scalable, maintainable, and easily extensible.

The analysis revealed that the system required several key components, including a user interface, an authentication and security module, a task management module, an effort tracking module, a reporting module, a data management module, and an integration module. Each of these components had specific functions and requirements, which were carefully considered during the decomposition process.

During the decomposition process, the components were further divided into smaller sub-components, each with a specific role and set of requirements. This allowed for a more granular analysis of the system architecture and helped ensure that each component could be easily maintained and extended in the future.

The allocation of requirements to each component was based on several factors, including the component's purpose, its technical capabilities, and its scalability. For example, the user interface was designed to be highly user-friendly and intuitive, with a focus on ease of use and accessibility. The authentication and security module was designed to be highly secure and robust, with advanced encryption algorithms and multi-factor authentication capabilities. The data management module was designed to be highly scalable, with the ability to handle large volumes of data and quickly retrieve information when required. The task management module was designed to be highly flexible, allowing team members to easily manage and track their tasks within the product team-of-teams structure.

Overall, the logical and structural decomposition of the system architecture was essential for ensuring that the effort logging and task management system met the high-priority quality requirements and addressed the customer expectations. By carefully analyzing the system's requirements and decomposing them into specific components and sub-components, the system will be able to meet the customer's needs while also being scalable, maintainable, and easily extensible.

**Logical and Structural Decomposition -**

Based on the description of the Effort Logging and Task Management system, the following pre-existing and to-be-created components and libraries can be identified:

1. Pre-existing components:
   1. JavaFX and Scene Builder for the User Interface
   2. Authentication and Security Module for secure access to the system
   3. Database management system for storing data
2. To-be-created components:
   1. Task management module for creating, updating and deleting tasks
   2. Time tracking module for logging time spent on tasks
   3. Reporting module for generating reports on task progress and time spent.

**User Interface:**

* Requirement: Provide a user-friendly interface for accessing the system.
* Rationale: The User Interface is the front-end of the system and provides the primary means for users to interact with the system. It needs to be user-friendly and intuitive to ensure ease of use.

**Authentication and Security Module:**

* Requirement: Ensure secure access to the system and protect confidential information
* Rationale: The Authentication and Security Module is responsible for verifying user credentials and protecting sensitive information from unauthorized access. It needs to be secure to ensure the confidentiality and integrity of the data.

**Task Management Module:**

* Requirement: Manage and track tasks within the firm's product team-of-teams structure
* Rationale: The Task Management Module is responsible for managing and tracking tasks within the organization. It needs to be structured in a way that supports the product team-of-teams structure to ensure effective collaboration and communication among team members.

**Effort Tracking Module:**

* Requirement: Log and track user efforts for each task
* Rationale: The Effort Tracking Module is responsible for tracking the user efforts for each task. It needs to be able to log and track the user efforts accurately to ensure that the data collected is reliable and useful for reporting and analysis.

**Reporting Module:**

* Requirement: Generate real-time reports on task progress and user efforts
* Rationale: The Reporting Module is responsible for generating real-time reports on task progress and user efforts. It needs to be able to generate reports that are accurate and up-to-date to provide the necessary information for decision-making.

**Integration Module:**

* Requirement: Integrate with other systems such as calendars, email systems, and project management tools
* Rationale: The Integration Module is responsible for integrating with other systems to provide seamless interaction and data exchange. It needs to be able to integrate with other systems to ensure that the Effort Logging and Task Management system can be used in conjunction with other tools and systems.

# Design

## Design Goals

The design goals for the EffortLogger application are closely aligned with the goals outlined in the previous deliverables, such as the Concept of Operations and the Requirements Definition. The primary goal of the design is to provide a robust and secure system that enables users to log their effort on various tasks seamlessly. The design must support implementers, testers, integrators, operators, and other stakeholders to ensure that the application meets critical quality requirements, such as security and privacy.

To achieve these design goals, the following design principles have been adopted:

1. Modular design: The application will be designed using a modular approach to enable easy integration with other systems and to promote ease of maintenance.
2. Scalability: The design must be scalable to support the growth of users and tasks over time.
3. Security: The design must ensure the confidentiality, integrity, and availability of user data. Authentication and authorization mechanisms will be implemented to protect user data from unauthorized access.
4. User-friendly interface: The application must have an intuitive and user-friendly interface that enables users to log their effort quickly and easily.
5. Performance: The application must perform optimally, even with a large number of users and tasks.

The design will support implementers, testers, integrators, and operators by providing detailed documentation on the architecture and implementation of the application. The design will also include security and privacy measures, such as encryption and access controls, to ensure that users' data is secure. The modular design will enable easy integration with other systems, reducing the complexity of the application and promoting ease of maintenance.

Overall, the design goals of the EffortLogger application are to provide a secure, scalable, and user-friendly platform for logging effort on various tasks. The design will support all stakeholders in their roles, ensuring that they can focus on their work without needing to be overly worried about critical quality requirements, such as security and privacy.

## Design Elements

The design elements of the EffortLogger application are the building blocks that make up the overall architecture of the system. Each element plays a critical role in ensuring that the functional and non-functional requirements of the system are met.

The key design elements of the EffortLogger application include:

1. User Interface (UI) Design: The UI design is a critical element of the system as it provides the user with a means to interact with the system. The UI design must be intuitive, easy to use, and provide the necessary functionality to support the system's requirements.
2. Data Management: Data management is an essential element of the system, as it involves the storage, retrieval, and manipulation of data. The design of the data management component must ensure that data is stored securely, efficiently, and is easily accessible when needed.
3. Application Programming Interface (API) Design: The API design is an essential element of the system, as it enables the integration of the EffortLogger application with other applications and systems. The API design must be well-defined, secure, and provide the necessary functionality to support the system's requirements.
4. Security Design: The security design is a crucial element of the system as it ensures that the system is secure from unauthorized access, data breaches, and other security threats. The design of the security component must ensure that the system is secure at all times and that all sensitive data is protected.
5. Performance Design: The performance design is an important element of the system as it ensures that the system meets its performance requirements. The design of the performance component must ensure that the system is responsive, scalable, and can handle the expected workload.

Each of these design elements must be carefully designed and implemented to ensure that they meet the system's requirements. To determine the extent to which the implementation of each element satisfies the requirements, testing and evaluation must be performed at each stage of development. Testing and evaluation will identify any issues with the implementation of the design elements and ensure that they are corrected before the system is deployed.

## Upside and Downside Risks

For each design element identified in section 5.2, the team conducted a risk prototyping exercise to identify any potential upside or downside risks associated with that element. The following is a summary of the risks identified for each design element:

1. User Interface: The upside risk identified for the user interface design element is that a well-designed and intuitive interface can lead to increased user adoption and satisfaction, resulting in greater product success. The downside risk identified is that a poorly designed interface can lead to user frustration and abandonment of the product. To mitigate these risks, the team conducted user testing and feedback sessions throughout the design process to ensure a user-friendly and effective interface.
2. Data Storage and Management: The upside risk identified for the data storage and management design element is that a well-designed and efficient system can result in faster and more accurate data retrieval and analysis. The downside risk identified is that a poorly designed system can lead to data corruption, loss, and security breaches. To mitigate these risks, the team implemented strong encryption and security measures and conducted thorough testing of the data management system.
3. Reporting and Analytics: The upside risk identified for the reporting and analytics design element is that a well-designed system can provide valuable insights and actionable information to users, leading to improved decision-making and outcomes. The downside risk identified is that a poorly designed system can result in inaccurate or incomplete reporting, leading to misinformed decisions. To mitigate these risks, the team implemented rigorous testing and validation of the reporting and analytics system.
4. Integration with Other Systems: The upside risk identified for the integration design element is that a well-designed integration can result in increased functionality and interoperability with other systems. The downside risk identified is that a poorly designed integration can lead to compatibility issues and system failures. To mitigate these risks, the team conducted extensive testing and validation of the integration with other systems.
5. Security and Privacy: The upside risk identified for the security and privacy design element is that a well-designed and robust system can protect user data and prevent security breaches. The downside risk identified is that a poorly designed system can lead to data leaks and breaches, resulting in serious consequences for users and the company. To mitigate these risks, the team implemented strong encryption and security measures, conducted extensive penetration testing, and followed industry-standard best practices for security and privacy.

### Network Security Risk Reduction Prototype

This prototype focuses on reducing the downside risk related to network security vulnerabilities. We will be implementing several security measures, such as encryption of sensitive data, setting up a firewall, and implementing authentication mechanisms to prevent unauthorized access. A screencast showcasing these measures will be provided in Project Report 6.

### User Interface Risk Reduction Prototype

This prototype focuses on improving the user interface to reduce the downside risk related to usability issues. We will be conducting several rounds of user testing and implementing changes based on feedback received, such as improving the layout, simplifying certain actions, and adding tooltips to clarify functionality. A screencast showcasing the updated user interface will be provided in Project Report 6.

## Design Reuse

In this section, we will document the rationale for the reuse of any existing code, libraries, or system components that will be used in the EffortLogger application. The use of external components can provide many benefits, such as reducing development time and effort, improving the quality of the system, and enhancing the overall performance.

One significant external component that we will use is the Firebase Realtime Database. The Firebase Realtime Database is a cloud-hosted database that provides real-time synchronization and data persistence, which is crucial for the EffortLogger application to function efficiently. Using this database will allow users to log their efforts and update their progress in real-time, without any data loss or inconsistency.

Another significant external component that we will use is the Google Sign-In API. This API allows users to sign in to the EffortLogger application using their Google account credentials. This API provides secure authentication and authorization services, ensuring that only authorized users can access the application and its features.

We have chosen to use these external components because they are proven, reliable, and widely used in the industry. Additionally, using these components will save us a significant amount of time and effort, allowing us to focus on implementing the unique features and functionalities of the EffortLogger application.

## Design Reuse Prototyping

We also prototyped the use of some external solutions to help us achieve some unique aspects of the app's design. For example:

### Design Reuse Prototype 1

Title: Image Capture and Processing

To enable users to add images of their efforts, we are thinking about integrating an image capture and processing feature using the Google Vision API. The API will allow us to detect objects and text within the captured images, enabling us to enhance the user experience and provide more information to users about their efforts. We would be testing the integration of the Google Vision API with the app's design using a prototyping tool.

# Final Integrated Prototype

[prototype code](https://drive.google.com/drive/folders/1hcOzXKnSq132mGC0rV3UC5lcw6K47OvL?usp=sharing)

The final integrated Effort Logger prototype is the fully functional version of the application that incorporates all the individual prototypes and features developed during the project's lifecycle. This prototype represents the end-product that meets all the requirements, goals, and specifications of the Effort Logger application.

## Effort Logger Prototype Goals

The primary goal of the final integrated Effort Logger prototype is to provide a complete and fully functional application that allows users to log and track their efforts efficiently. The prototype should meet all the requirements and specifications outlined in the project scope and deliver a seamless and intuitive user experience.

Additionally, the prototype should showcase the team's ability to develop and integrate various software components, APIs, and external solutions to create a robust and reliable application. The prototype should demonstrate the team's competence in software design, development, testing, and deployment while following industry best practices and coding standards.

Finally, the prototype should meet the user's needs and expectations by delivering a high-quality, user-centric experience. The prototype should provide a responsive and reliable application that users can access from multiple devices and platforms. Overall, the final integrated Effort Logger prototype should represent a successful and valuable solution to the problem of tracking personal efforts.

## Implemented Key Features

### **Login and Logout Functionality in EffortLogger App**

**Key Feature:** The EffortLogger app has a well-designed login page prototype that allows legitimate users to securely access the app's features while preventing security breaches. The app displays an error message when an unauthorized user tries to login with incorrect credentials. The dashboard streamlines workflows and enhances the user experience while maintaining data security.

**Description:** The login and logout functionality is a crucial aspect of any app that deals with sensitive user information. The EffortLogger app recognizes this importance and has implemented a robust login and logout system to ensure that the user's data is secure and the app is easy to use.

The login page prototype of the EffortLogger app sets the tone for the user experience of the app. It is the first interaction that users have with the app and needs to be well-designed to ensure a positive first impression. The login page prototype of the EffortLogger app is a preliminary design of the page where users will enter their login information to access the app's features. The login page prototype is important because it plays a crucial role in setting the tone for the user experience of the app. The EffortLogger app displays an error message when an unauthorized user tries to login with incorrect credentials. This approach maintains the app's security and allows legitimate users to identify and address login issues. This feature is essential for preventing security breaches and maintaining the privacy of user information.

Once the user has logged in, they can securely access the dashboard to manage their tasks, projects, and time logs. The dashboard streamlines workflows and enhances the user experience while maintaining data security. The EffortLogger app has designed the dashboard to be user-friendly, allowing users to easily view and manage their tasks, projects, and time logs. The dashboard is also customizable, allowing users to tailor it to their preferences.

The EffortLogger app allows users to log out with a push of a button once they are done using the app. This feature is essential for maintaining the security of the user's account and preventing unauthorized access to the app. By providing a quick and easy way to log out, the app encourages users to take an active role in protecting their information, enhancing the overall user experience. When the user logs out, they are redirected back to the secure login page, ensuring that their sensitive data and credentials are protected.

### Timer Functionality in EffortLogger

**Key Feature:** EffortLogger is a time-tracking tool that allows users to track their time spent on various tasks and projects. One of its key features is the Timer, which provides users with a convenient and accurate way to track the time spent on a specific task.

The Timer in EffortLogger is a simple but powerful tool that allows users to start and stop a timer to track the duration of their work on a specific task. Here's a detailed description of how it works:

**Description:**The timer in EffortLogger can be used to measure the amount of time spent on tasks, projects, or any other activities. It can be started, stopped, and paused as needed, and it automatically records the duration of each task.

When a user starts the timer, EffortLogger creates a new entry in the task list, including the name of the task and the start time. As the timer continues to run, the duration of the task is updated in real-time, and the user can see exactly how much time has been spent on the task so far.

The timer can also be paused and resumed as needed. When the user pauses the timer, the task entry is temporarily suspended, and the duration is no longer updated. This allows users to take breaks or switch to other tasks without interrupting their time tracking.

EffortLogger also provides a detailed log of all task entries, including the start time, duration, and any notes or comments added by the user. This log can be used to analyze productivity trends and identify areas where time is being wasted or where more time is needed.

Overall, the timer feature in EffortLogger provides users with a simple and effective way to track their time and stay on top of their workloads. By accurately measuring the amount of time spent on each task, users can gain valuable insights into their productivity and make informed decisions about how to manage their time more effectively.

### Create Tasks/Teams

In Phase 5 of the project, we developed a prototype focusing on creating tasks and teams. The main goal of this prototype was to provide a user-friendly interface for team creation and task management. We implemented the following key features:

1. Team creation: The prototype allows users to create teams and invite team members by email. The team creator can assign team roles and set access permissions.
2. Task creation: Users can create tasks and set deadlines. The tasks can be assigned to team members or individual users.
3. Task management: The prototype includes a task dashboard where users can view and manage all their tasks. Users can view task details, mark tasks as complete, and add comments or attachments to tasks.

The task and team management features were developed using Java/JavaFX. The prototype includes a simple and intuitive user interface, making it easy for users to create teams and manage their tasks. We plan to integrate this prototype with the other prototypes developed by my teammates in Phase 6 of the project.

Other features haven’t been changed since the last phase and we plan on developing them and integrating them into the final prototype soon.

## Unimplemented Key Features

Provide an overview of the key features that were not implemented (a short outline that may change and will be expanded for the final report). The details about each are provided directly below.

### Storing User Credentials in a Database for Improved Security

**Key Feature Not Implemented:** The EffortLogger app does not currently store user credentials in a database. This feature will be implemented in the final prototype to enhance the security of the app.

**Description**: User credentials, such as usernames and passwords, are crucial for ensuring the security of an app. Storing these credentials in a database is a standard practice for securing user data. By storing user credentials in a database, the app can ensure that the sensitive information is encrypted and secure. Unfortunately, due to time constraints, the current version of the EffortLogger app does not have this feature. However, in the final prototype, storing user credentials in a database will be implemented to improve the security of the app.

The EffortLogger app is designed to help users manage their tasks, projects, and time logs. As users input this sensitive information into the app, it is essential that their data is protected. With a database storing user credentials, the app can ensure that the sensitive information is encrypted and protected from unauthorized access. In addition to protecting user data, storing credentials in a database can also make it easier to manage user accounts. The database can keep track of user accounts and authentication tokens, making it easier to manage user accounts and credentials in a secure way.

The EffortLogger app has already implemented robust login and logout functionality, displaying error messages when an unauthorized user tries to log in and redirecting users to the secure login page after logging out. By implementing the feature of storing user credentials in a database, the app can further enhance its security measures and provide a more reliable and secure tool for users to manage their tasks, projects, and time logs.

### Storing all the Tasks/Teams and reminders regarding deadlines of each tasks

**Key Feature Not Implemented:** Task Prioritization: The current version of the EffortLogger app does not have the feature of task prioritization. This feature will be implemented in the final prototype to enable users to prioritize their tasks according to their importance and deadline.

1. Task Reminder: The EffortLogger app does not currently have a reminder feature for tasks. This feature will be implemented in the final prototype to remind users of their upcoming deadlines.
2. User Profile: The current version of the EffortLogger app does not have a user profile feature. This feature will be implemented in the final prototype to allow users to customize their profiles and manage their information.
3. Notification System: The EffortLogger app does not currently have a notification system to notify users of any changes in their tasks or teams. This feature will be implemented in the final prototype to keep users informed of any updates to their tasks or teams.
4. File Upload: The EffortLogger app does not currently have the feature of file upload. This feature will be implemented in the final prototype to allow users to upload files related to their tasks.

These key features are crucial for enhancing the user experience of the EffortLogger app and providing users with a comprehensive tool to manage their tasks, projects, and time logs. In the final prototype, these features will be implemented to enable users to prioritize their tasks, receive reminders of upcoming deadlines, customize their profiles, stay informed of updates to their tasks and teams, and upload files related to their tasks.

Task prioritization will enable users to rank their tasks according to their importance and deadline, allowing them to focus on the most critical tasks first. Task reminders will ensure that users never miss an important deadline and can complete their tasks on time. The user profile feature will allow users to customize their profiles, view their activity history, and manage their information, providing a more personalized experience. The notification system will keep users informed of any updates to their tasks or teams, ensuring that they are up-to-date with their progress. Finally, the file upload feature will allow users to upload files related to their tasks, such as documents, images, or videos, to keep everything in one place.

Overall, these features will enhance the functionality of the EffortLogger app and provide users with a comprehensive tool to manage their tasks, projects, and time logs. These key features will be implemented in the final prototype, providing users with a more reliable and secure tool to manage their workload.

### 

### Key Feature Not Implemented 2 … n

Do the same for all the remaining demonstrations.

# Conclusion

The EffortLogger project has made significant progress since Phase 4, with the team focusing on developing and prototyping key features that will form the foundation of the final product. In Phase 5, the team continued this work, refining and integrating these features into a final prototype.

One of the key features developed in Phase 4 was the user profile system, which was integrated into the final prototype in Phase 5. This system allows users to create and manage their profile information, including their name, email address, and password. The user profile system also provides a way for users to manage their privacy settings and control who can view their information.

Another important feature developed in Phase 4 was the effort logging system. This system allows users to log their efforts, including the amount of time spent on a particular task, the task description, and the date and time of the effort. The effort logging system was integrated into the final prototype in Phase 5, along with additional features such as real-time synchronization and data persistence using the Firebase Realtime Database.

In Phase 5, the team also focused on developing and prototyping several other key features that will be essential to the success of the final product. One of these features was the task creation and management system. This system allows users to create new tasks, set deadlines, and assign tasks to specific teams or team members. The task creation and management system also provides a way for users to view the status of their tasks and monitor the progress of their team members.

Another feature that was developed and prototyped in Phase 5 was the team management system. This system allows users to create and manage teams, invite team members, and assign tasks to specific teams or team members. The team management system also provides a way for users to view the progress of their teams and monitor the performance of individual team members.

Throughout the development and prototyping process, the team focused on using proven external solutions and reuse whenever possible. For example, the team used the Firebase Realtime Database to provide real-time synchronization and data persistence, and the Google Sign-In API to provide secure authentication and authorization services.

Overall, the work done in Phase 5 has been crucial to the development of the final Effort Logger prototype. By refining and integrating key features, and by using proven external solutions and reuse wherever possible, the team has laid a strong foundation for the final product. Moving forward, the team will need to focus on finalizing the design and implementing additional features and functionalities that will make the Effort Logger a useful and user-friendly tool for tracking and managing team efforts.

# Appendix A: Credit Sheet

1 page, single space 10 pt. font: Times or Calibri

This appendix lists the members of the team and includes a description of the contribution that team member has made to this submission. Each team member must provide text for their contribution at a team meeting just prior to the submission of this deliverable and the entire team must agree. If a team member fails to provide this information and/or does not participate in the agreement process to fill out this table, that member will receive no credit for the submission of this deliverable.

| Team Member Name | Contributions |
| --- | --- |
| Aditya Maheshbhai Gabani | Wrote the Introduction and Conclusion sections |
| Viet Le | Refined the documentation for Concept of Operations |
| Vatsal Maniar | Upgraded prototype for Viewing Tasks and integrated with Sumit Patel’s prototype of Creating Tasks. Also, I wrote part of section 5. Design |
| Smrita Pandey | Refined the requirements definition. Defined a timer key feature for section 6.2. |
| Sumit Patel | Upgraded prototype for Creating Tasks and integrated with Vatsal Maniarl’s prototype of Viewing Tasks. Also, I wrote part of section 5. Design |
| Shivansh Shrivastava | Refined the Architecture documentation, and about login features for section 6.2, 6.3 |