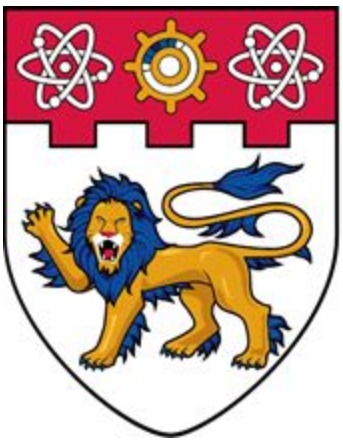


NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE

CZ3002 Advanced Software Engineering Project:

Project Plan - CashTrack

Version 1.3

Team Members:

Ravishankar Amrita (U1822377F)

Datta Anusha (U1822948G)

Kumar Mehul (U1822146E)

Alex Leong (U1921599D)

S Sri Kalki (U1921575L)

Nicklaus Tan (19403385F)

Elliott Ong (U1922981C)

Daniel Loe (U1921408A)

Lab Group: B2

School of Computer Science and Engineering (SCSE)

Revision History

Revision Number	Date	Primary Author(s)	Comments
1.0	Feb 16th, 2021	Ravishankar Amrita, S Sri Kalki, Datta Anusha, Alex Leong, Nicklaus Tan, Kumar Mehul, Elliott Ong and Daniel Loe	Initial Draft
1.1	Feb 25th, 2021	Ravishankar Amrita, S Sri Kalki, Datta Anusha, Alex Leong, Nicklaus Tan, Kumar Mehul, Elliott Ong and Daniel Loe	Revised Efforts, Duration and Team Size Estimation, Quality Assurance and Risk Management
1.2	Mar 5th, 2021	Nicklaus Tan, Kumar Mehul, Elliott Ong and Daniel Loe	Revised Work Breakdown Structure, Work Package Details, Best Practices Checklists
1.3	Mar 12th, 2021	Ravishankar Amrita, S Sri Kalki, Datta Anusha	Finalized Version

Table of Contents

1 Introduction

1.1 Project Overview

CashTrack is a web-based expense tracker that acts as a one stop destination to track personal expenses, ease the process of resolving shared bills and view comprehensive insights into users' spending patterns.

1.2 Project Description and Scope

CashTrack is intended to be a part of the large e-commerce shift towards digital payments (including contactless), instant payments, and cash displacement. Our team, Runtime Terror, has been contracted to fulfill the agent portion of the CashTrack software, which will then serve to provide an integrated solution to ease the process of splitting bills amongst friends, tracking the status of the payments owed and more. To improve application experience, Runtime Terror has procured permission to perform data design and requirements research independent of the client's eventual e-commerce ventures.

CashTrack provides an integrated platform solution that enables users to collectively record, analyse and settle their monetary transactions between each other. The following are the primary features of this application:

1. **Personal Expense Tracking**
   CashTrack allows tracking of an individual's expenditure, more specifically how much they owe and are owed by other people, as well as their own expenditure such as on food, travel, entertainment etc. Users are able to add personal expense records by navigating to the 'Personal Expenses' page. They can categorise these personal expense records by using category 'tags' such as food and travel.

2. **Shared Expense Tracking**
   As users expand their social circles, frequent get-togethers may be expected, consequently causing splitting bills to become a recurring event. Hence, CashTrack allows splitting bills, tracking the payment statuses, while actively monitoring and notifying the parties involved. Users are able to create shared expense records with friends and family, for bills where the entire payment was made by the user. Further, users adding the record can choose the category of the record, use the automatic split by ratio/percentage/shares option, and also add comments to this shared expense record.

3. **Insights into Personal and Shared Expense Spending Patterns**
   CashTrack performs a comprehensive analysis of users' expense data over time or by categories. This aims to reveal any underlying spending patterns or interesting insights into the user's payment trends. Further, it helps the user to effectively visualise their transactions histories for all both personal and shared expenses.

   Finally, the application also provides a chat feature to increase user engagement as well as to clarify doubts regarding their shared expense transactions.

2 Project Organization

2.1 Team Structure

The following is the list of executive roles of the project team:

- **Project Manager:** Nicklaus Tan
- **Lead Developer:** Kumar Mehul
- **Front-end Developer:** Ravishankar Amrita
- **Back-end Developer:** Datta Anusha
- **Release Engineer:** Alex Leong
- **Quality Assurance Manager:** S Sri Kalki
- **Quality Assurance Engineer:** Daniel Loe
- **Quality Assurance Engineer:** Elliott Ong

2.2 Roles and Responsibilities

The roles and responsibilities of each team member has been detailed as follows:

2.2.1 Project Manager: Nicklaus Tan

- Supervises project progress
- Approves and executes project plan
- Manages and motivates team members
- Allocates tasks and reports project status to team members
- Represents the team to external clients

2.2.2 Lead Developer: Kumar Mehul

- Overall technical lead for the project
- Directs the technical aspect of product release
- Ensures system architecture is well designed

2.2.3 Front-End Developer: Ravishankar Amrita

- Ensures that the system design is technically feasible
- Ensures that user input is tested before being sent to the back-end
- Collaborate effectively with back-end developer and lead developer

2.2.4 Back-End Developer: Datta Anusha

- Implement working product using the detailed design document
- Responsible for integration of server side logic with user interface
- Designs and implements data storage solutions
- Collaborate effectively with front-end developer and lead developer
- Ensures stability and response time of the system meet the requirements

2.2.5 Release Engineer: Alex Leong

- Creates baseline and conducts release reviews
- Measures and monitors project progress
- Ensures application releases are delivered on time and within budget
- Make use of Version Control Systems to coordinate release content and effort
- Builds and integrates changes for delivery

2.2.6 Quality Assurance Manager: S Sri Kalki

- Supervises overall product and process quality
- Responsible for creating quality documentation and reports
- Ensures software meets quality benchmark

2.2.7 Quality Assurance Engineer(s): Daniel Loe, Elliott Ong

- Ensures acceptable software quality
- Designs testing strategies
- Develops and manages test plan
- Verify software requirements align with user requirements
- Executes test procedure

2.3 Team Communication

CashTrack's core team communication channels and workflows include the following:

- Weekly development meetings held on Tuesdays
- Bi-monthly Risk Management Reviews scheduled
- Text based group announcements and updates sent via Whatsapp
- Zoom platform used to conduct for live online face-to-face meeting
- File sharing facilitated through Google Drive cloud storage and the team's Media Wiki page
- Source code control facilitated via GitHub & Jira
- To improve efficiency, the team is divided into two key subgroups
    - Documentation Team
    - Development Team

3 Process Definition

3.1 Lifecycle Model

The CashTrack team intends to use the Agile Methodology for the project management and development purposes. Using the Agile Framework will enable the team to deliver a superior quality product since testing is an integral part of the project execution phase in this method.

Moreover, this method will provide a constant feedback loop from the customers to all the people involved in project development, providing adaptability to changing requirements according to the market and customer needs. This also allows the team to reduce the go-to-market time by developing a first minimum viable product (MVP).

Agile ensures reduced risks and increased flexibility, with continuous improvements and improved team morale as the teams are self-organised and self-managing.

3.2 SCRUM

Specifically for this project, the team followed a SCRUM methodology for project planning and development. Some of the good SCRUM practices adopted to ensure a efficient development lifecycle are as follows:

1. **Organise Backlog**

The team consistently maintained a list of todo items using feedback from users and the development team, to help prioritize and keep the list clean and ready to be worked on at any given time. This process is also called Backlog Grooming.

2. **Sprint Planning**

During this event, the team decides the scope of what to accomplish during the next sprint. This process includes discussion and planning to move tasks and stories from the backlog to the sprint in order to meet the sprint goal, keeping in mind what's feasible to complete in two weeks.

3. **Sprint**

A sprint is the actual time period when the scrum team works together to achieve an increment. Sprints are usually around two weeks. This helps the team learn from past experiences and apply that insight to future sprints. During the sprint, the scope may be re-negotiated between the Product Owner and the Development Team if necessary.

4. Daily Scrum or Stand-up

Throughout the sprint duration, the team must conduct a daily scrum or stand-up meeting. This is a super short meeting that is usually held in the morning and lasts for about 15 minutes. The emphasis is on the quickness of the meeting, with the goal of the meeting being the team to be on the same page, aligned with the sprint goal, and to get a plan out for the next 24 hours. The stand-up is the time to voice any concerns you have with meeting the sprint goal or any blockers.

5. Sprint Reviews

At the end of the sprint, the team conducts a sprint review, which is an informal session to view a demo to inspect the increment produced by the sprint. This review meeting is for the Development Team to present the backlog items to the stakeholders and teammates for feedback. The Product Owner may assess this work and approve it for the release. The Product Owner also reworks the backlog based on the current sprint, and that serves as a starting point for the next sprint planning session.

6. Sprint Retrospectives

The last step of the sprint is the retrospective, where the team evaluates how they worked and build a plan for how to improve. This provides an empirical foundation for teams, enabling them to deliver more frequently, with higher value and better outcomes for customers.

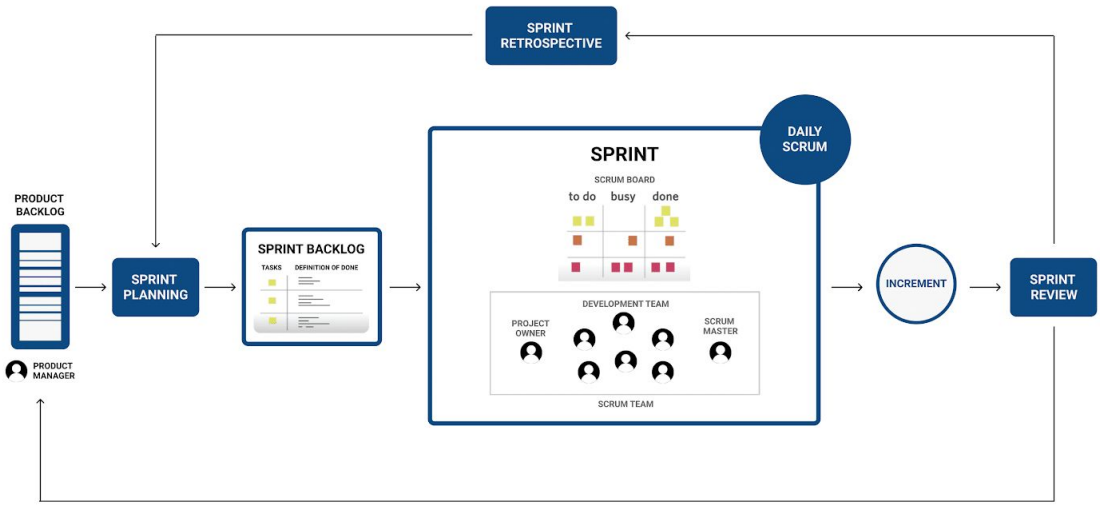
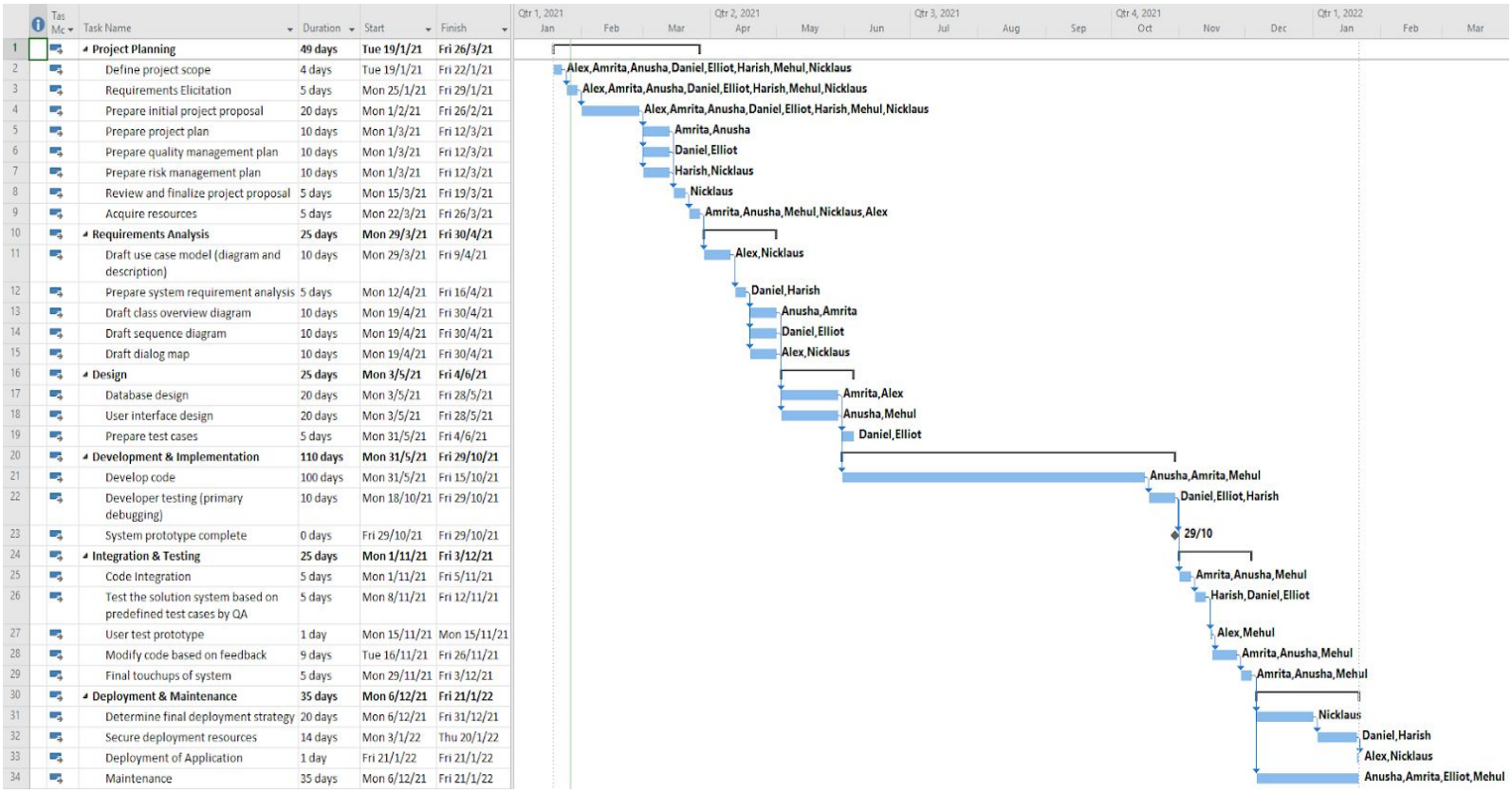


Figure 1, SCRUM lifecycle (Created by Team Runtime Terror)

4 Schedule

The detailed tasks, activity dependencies and schedule for CashTrack project may be found in the Gantt Chart below:

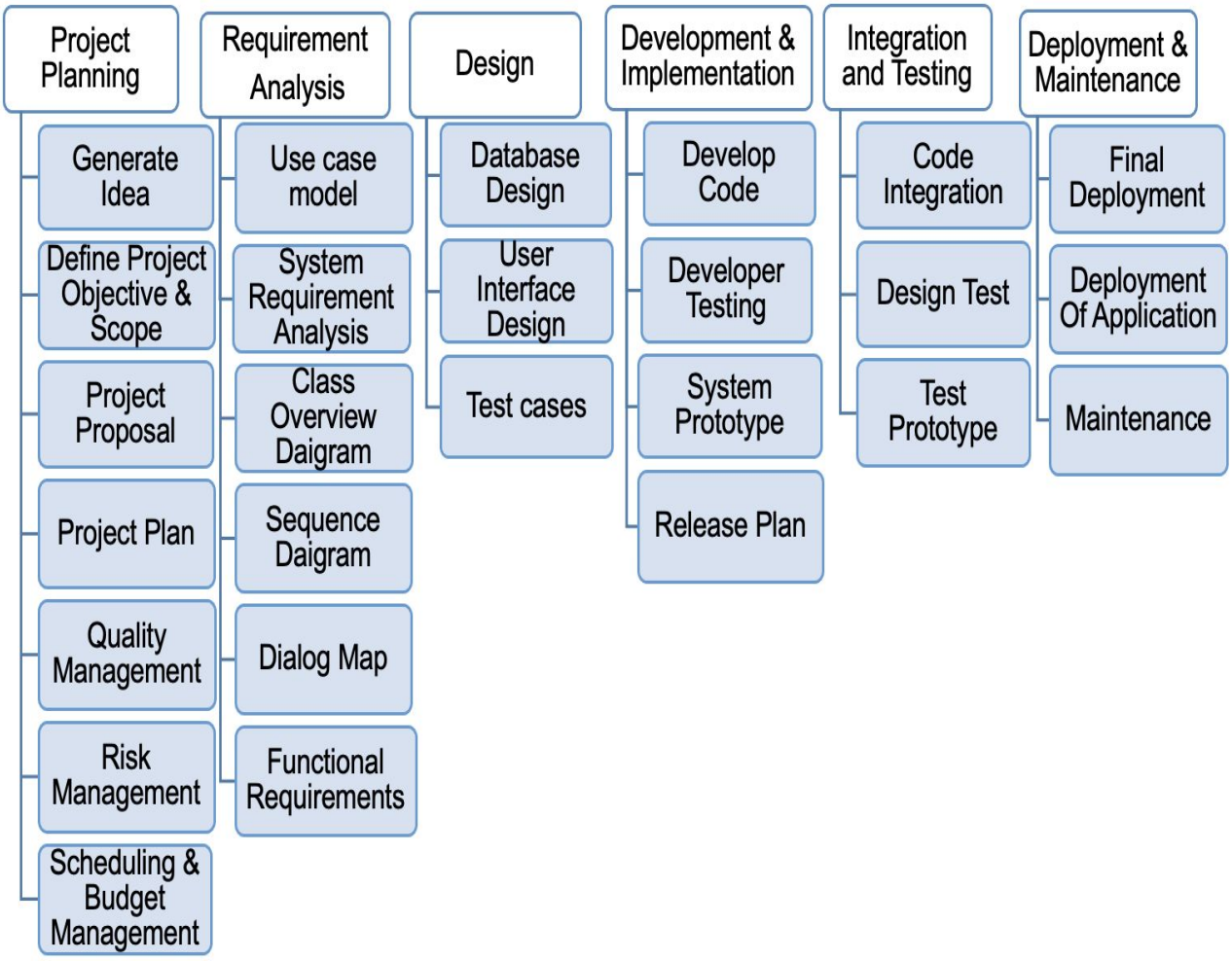
4.1 Activity Dependencies and Schedule



	Task Mc	Task Name	Duration	Start	Finish
1		**Project Planning**	**49 days**	**Tue 19/1/21**	**Fri 26/3/21**
2		Define project scope	4 days	Tue 19/1/21	Fri 22/1/21
3		Requirements Elicitation	5 days	Mon 25/1/21	Fri 29/1/21
4		Prepare initial project proposal	20 days	Mon 1/2/21	Fri 26/2/21
5		Prepare project plan	10 days	Mon 1/3/21	Fri 12/3/21
6		Prepare quality management plan	10 days	Mon 1/3/21	Fri 12/3/21
7		Prepare risk management plan	10 days	Mon 1/3/21	Fri 12/3/21
8		Review and finalize project proposal	5 days	Mon 15/3/21	Fri 19/3/21
9		Acquire resources	5 days	Mon 22/3/21	Fri 26/3/21
10		**Requirements Analysis**	**25 days**	**Mon 29/3/21**	**Fri 30/4/21**
11		Draft use case model (diagram and description)	10 days	Mon 29/3/21	Fri 9/4/21
12		Prepare system requirement analysis	5 days	Mon 12/4/21	Fri 16/4/21
13		Draft class overview diagram	10 days	Mon 19/4/21	Fri 30/4/21
14		Draft sequence diagram	10 days	Mon 19/4/21	Fri 30/4/21
15		Draft dialog map	10 days	Mon 19/4/21	Fri 30/4/21
16		**Design**	**25 days**	**Mon 3/5/21**	**Fri 4/6/21**
17		Database design	20 days	Mon 3/5/21	Fri 28/5/21
18		User interface design	20 days	Mon 3/5/21	Fri 28/5/21
19		Prepare test cases	5 days	Mon 31/5/21	Fri 4/6/21
20		**Development & Implementation**	**110 days**	**Mon 31/5/21**	**Fri 29/10/21**
21		Develop code	100 days	Mon 31/5/21	Fri 15/10/21
22		Developer testing (primary debugging)	10 days	Mon 18/10/21	Fri 29/10/21
23		System prototype complete	0 days	Fri 29/10/21	Fri 29/10/21
24		**Integration & Testing**	**25 days**	**Mon 1/11/21**	**Fri 3/12/21**
25		Code Integration	5 days	Mon 1/11/21	Fri 5/11/21
26		Test the solution system based on predefined test cases by QA	5 days	Mon 8/11/21	Fri 12/11/21
27		User test prototype	1 day	Mon 15/11/21	Mon 15/11/21
28		Modify code based on feedback	9 days	Tue 16/11/21	Fri 26/11/21
29		Final touchups of system	5 days	Mon 29/11/21	Fri 3/12/21
30		**Deployment & Maintenance**	**35 days**	**Mon 6/12/21**	**Fri 21/1/22**
31		Determine final deployment strategy	20 days	Mon 6/12/21	Fri 31/12/21
32		Secure deployment resources	14 days	Mon 3/1/22	Thu 20/1/22
33		Deployment of Application	1 day	Fri 21/1/22	Fri 21/1/22
34		Maintenance	35 days	Mon 6/12/21	Fri 21/1/22

Figure 2 : Project Schedule

4.2 Work Breakdown Structure



4.3 Work Packages

The entire project has been segmented into important phases of the software development life cycle. These include the following:

1. Project Planning
2. Requirement Analysis
3. Design
4. Development & Implementation
5. Integration & Testing
6. Deployment & Maintenance

4.4 Activity Dependencies

The following table describes the dependencies of the deliverable work packages:

Table 1 : Activity Dependencies of Work Packages

Work Package #	Work Package Description	Duration	Dependencies
X01	Project Planning	49 days	--
X02	Requirement Specification	25 days	X01
X03	Design	25 days	X02
X04	Development & Implementation	110 days	X03
X05	Integration and Testing	25 days	X04
X06	Deployment & Maintenance	35 days	X05

The following Activity Network Diagram describes the above in more graphical detail:

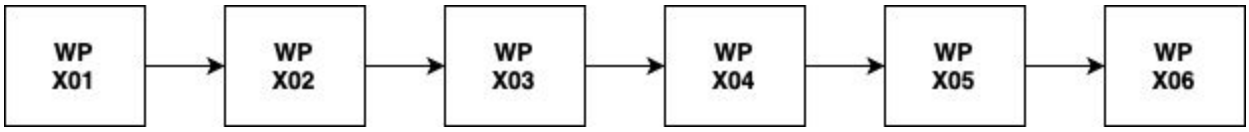


Figure 3 : Activity Network Diagram of Dependencies

4.5 Work Package Details

Further details regarding the work packages are listed below. Team members have been assigned as primary responsibility for each work package and must coordinate the completion of that package.

Table 1 : WP X01 - Project Planning

Project	CashTrack Web Application
Work Package	X01 - Project Planning (1 of 6)
Assigned To	Alex Leong, Ravishankar Amrita, Anusha, Daniel Loe, Elliott Ong, S Sri Kalki, Kumar Mehul, Nicklaus Tan
Effort	49 PD
Start Date	19 January 2021
Purpose	To formulate an introductory overview of the project, which to be further refined in the subsequent work packages.
Input	None
Activities	This work package includes developing a project concept and an overview which includes an objective, a set of proposed deliverables, schedule and budget as well as managing quality and risk throughout the software development life cycle. The respective members responsible for this work package will be involved in formulating it into a formal report.
Outputs	A written document of the Project Proposal, Project Plan, Quality Management, Risk Management and Scheduling & Budget Management.

Table 2 : WP X02 - Requirement Analysis

Project	CashTrack Web Application
Work Package	X02 - Requirement Analysis (2 of 6)
Assigned To	Alex Leong, Ravishankar Amrita, Anusha, Daniel Loe, Elliott Ong, S Sri Kalki, Nicklaus Tan
Effort	25 PD

Start Date	09 March 2021
Purpose	To establish a detailed and clear understanding of the target users' needs, which must be addressed effectively by the software project team.
Input	Target Users' Requirements
Activities	This work package includes identifying and interviewing the target audience as well as understanding customers' and system requirements. This will be formally documented to capture the user's interaction with the system, between objects instances in the system, model state transitions and system behavior under specific conditions.
Outputs	A written document of the Use Case Model, System Requirement Specifications, Class Overview Diagram, Sequence Diagram, Dialog Map and Functional Requirements.

Table 3 : WP X03 - Design

Project	CashTrack Web Application
Work Package	X03 - Design Specifications (3 of 6)
Assigned To	Alex Leong, Ravishankar Amrita, Anusha, Daniel Loe, Elliott Ong, Kumar Mehul
Effort	25 PD
Start Date	03 May 21
Purpose	To design the frontend and backend of the system as well as develop test cases for future prototypes.
Input	WP X01-02
Activities	This work package includes translating high level architecture into visual design as a low-fidelity prototype. It will be further refined into a high-fidelity prototype. Furthermore, data will be classified and organised based on its interrelationships to form the system's database design. In addition, a specific set of actions will be developed to verify the system's functionality will be formally documented.
Outputs	A written document of the Database Design, User Interface Design

	and Test cases.

Table 4 : WP X04 - Development & Implementation

Project	CashTrack Web Application
Work Package	X04 - Development & Implementation (4 of 6)
Assigned To	Ravishankar Amrita, Anusha, Daniel Loe, Elliott Ong, S Sri Kalki, Kumar Mehul
Effort	110 PD
Start Date	31 May 2021
Purpose	To develop and implement the system as preliminary developer testing.
Input	WP X01-03
Activities	This work package includes implementing and performing debugging testing to ensure the system modules are as per the requirements specification and other associated documents. The system will be broken down into releases and implemented according to the schedule.
Outputs	A written document of Release Plan as well as System Application Prototype

Table 5 : WP X05 - Integration and Testing

Project	CashTrack Web Application
Work Package	X05 - Integration and Testing (5 of 6)
Assigned To	Alex Leong, Ravishankar Amrita, Anusha, Daniel Loe, Elliott Ong, S Sri Kalki, Kumar Mehu
Effort	25 PD
Start Date	01 November 2021

Purpose	To identify and fix logical and syntactical errors produced during the implementation of the system as well as finalising the system before deployment.
Input	WP X01-04
Activities	This work package includes black box testing to verify functionality of the system without internal code structure as well as white box testing to verify internal system workings via input-output flow. In addition, user testing will be conducted to improve interface design. If any issues are identified, it will be documented and fixed in the earliest possible time.
Outputs	A written document of test cases report as well as Finalised System Application

Table 6 : WP X06 - Deployment & Maintenance

Project	CashTrack Web Application
Work Package	X06 - Deployment & Maintenance (6 of 6)
Assigned To	Alex Leong, Ravishankar Amrita, Anusha, Daniel Loe, Elliott Ong, S Sri Kalki, Kumar Mehul, Nicklaus Tan
Effort	35 PD
Start Date	06/12/2021
Purpose	To deploy and maintain the System Application.
Input	WP X04
Activities	These work packages include developing deployment strategy plans and securing development resources to execute the actual development on the planned date. In addition, continuous improvements and debugging will be done to ensure it meets application’s objectives.
Outputs	This will include SVN repository as well as successful deployment and maintenance of System Application.

5 Project Estimates

Project estimation is done to estimate the effort, resources, time and costs needed to develop the system. By creating a baseline of the project using project estimates, though it is not possible to completely eliminate uncontrollable factors affecting the execution of the project such as last minute overheads, unforeseen expenditure and personnel issues, the project team will be able to better adapt to these factors by using the estimates as a reference point then making the appropriate changes.

This section describes the methods used to produce the estimations as well as its justifications.

5.1 Code Size Estimation using Function Points

We calculated unadjusted function points based on the complexity of functions provided by this system. Code size is then estimated by adjusted function point.

5.1.1 Unadjusted Function Points

CashTrack supports the following proposed Functional Requirements:

1. General User

    1.1. System must allow User to register an account.
    1.2. System must allow User to log in.
        1.2.1. System must allow User to log out.
    1.3. System must allow User to invite people to the application.
    1.4. System must allow User to create a group involving two or more users.
        1.4.1. System must allow User to invite other users to the group.
        1.4.2 System must allow User to remove other users to the group.
    1.5. System must allow User to chat with other users.
        1.5.1. System must send new chat message notifications in real-time.
    1.6. System must allow User to set a desired personal expense limit amount for given time frame
        1.6.1. System must allow the User to set a percentage for the expense limit, which when reached must notify the User of the same.
        1.6.2. System must send a notification to User when User exceeds the set limit.
    1.7. System must allow User to input their personal expenses
        1.7.1. System must allow User to select the category/type of inputted expense.
        1.7.2. System must allow User to update/modify details of selected personal expense records.
        1.7.3. System must allow User to delete selected personal expense records.
    1.8. System must allow User to view data analytics of their expense records.
        1.8.1. System must be able to visualise spending patterns based on personal expenses by categories using pie charts.
        1.8.2. System must be able to visualise spending patterns based on shared expenses with another user over time using line charts.

1.9. System must allow a User to delete their CashTrack account.

2. Payee

2.1. System must allow User to split an expense, by creating a shared expense record, with one or more individuals or a group.
2.2. System must allow User to split the expense Equally, by Percentage, by Shares, by custom amounts, or with a currency exchange feature.
2.3. System must allow User to acknowledge payment from another User (Payer).
2.3.1 System must send a notification to User when image evidence has been added to the shared expense record they created.
2.4. System must send a notification to User when an expense they created receives a comment.
2.5. System must allow the User to make comments for any expense record they are involved in.
2.6 System must allow Users (Payee) to update/delete a shared expense record.

3. Payer

3.1. System must allow User to Settle payment with User (Payee)
3.1.1. System must allow User to upload image evidence
3.2. System must allow User to comment on an expense that they are involved in
3.3. System must send a notification to User when an expense they are involved in receives a comment.
3.4 System must send a notification to the User when an expense they are involved in is edited or deleted.
3.5. System must send a notification to the User (Payer) when another User (Payee) acknowledges their payment for a shared expense.

The measure of unadjusted function points is based on five primary component elements of these functions: Inputs, Outputs, Inquiries, Logical Files, and Interfaces. Each element ranges from Low Complexity, Medium Complexity to High Complexity.

Element	Details	Complexity
Inputs	Account Information (Username, Password, Email)	Low
	Select friend from friend's list to add to group Selection	Low
	Personal Expense Limit	Low
	Input/Update/Delete Expense Record	Low
	Upload transaction proof evidence	Medium
	Chart type selection via menu	Low

Outputs	User transaction data analytics report	Medium
	Expense Record	Low
	Notifications	Low
Inquiries	Acknowledgement	Low
	Login	Low
	Data Retrieval	Low
Logical Files	Account Database	Low
	Expense Records Database	High
Interfaces	Register Interface	Low
	Login Interface	Low
	Expense record creation Interface	Medium
	Group creation Interface	Low
	Chat Interface	Low
	Account Profile Interface	Low
	Data Analytics Interface	Medium
	Upload transaction proof Interface	Medium

Calculation of Unadjusted Function Points:

Characteristic	**Low**		**Medium**		**High**	
Inputs	5	× 3	1	× 4	0	× 6
Outputs	2	× 4	1	× 5	0	× 7
Inquiries	3	× 3	0	× 4	0	× 6
Logical Files	1	× 7	0	× 10	1	× 15
Interfaces	5	× 5	3	× 7	0	× 10
Unadjusted FP	64		30		15	
Total=L+M+H	109					

5.1.2 Adjusted Function Points

SCORING (0-5)	INFLUENCE LEVEL
0	No Influence
1	Insignificant Influence
2	Slight Influence
3	Average Influence
4	Significant Influence
5	Strong Influence

Influence Factors	Detail	Score
Data Communications	Application is more than a front-end, and supports more than one type of teleprocessing communications protocol.	4
Distributed Functions	Distributed processing and data transfer are online and in both directions.	1
Performance	Response time or throughput is critical during all business hours. No special design for CPU utilization was required. Processing deadline requirements with interfacing systems are constraining.	3
Heavily used	Several security and timing considerations were taken into consideration and evaluated	2
Transaction rate	Daily peak transaction period is anticipated.	4
On-line data entry	More than 30% of transactions are interactive data entry	5
End-user efficiency	Stated requirements for user efficiency requires the need for moderate consideration of human factors regarding ease of use of the application	3
On-line data update	Online update of major internal logical files is included.	1
Complex processing	The degree to which processing logic influenced the development of the application. No complex processing present.	0

Reusability	The application was specifically packaged and/or documented to ease re-use, and the application is customized by the user at source code level.	3
Installation Ease	No special considerations were stated by the user and no special setup is required for installation.	0
Operational Ease	Effective start-up, back-up, and recovery processes were provided, but no operator intervention is required (count as two items).	1
Multiple sites	User requirements do not require considering the needs of more than one user/installation site.	0
Facilitate change	Flexible query and report facility is provided that can handle complex requests. Design principles such as low coupling and high cohesion were considered to better facilitate enhancements/modifications of the system.	3
Total score	30	
Influence Multiplier = Total score × 0.01 + 0.65 = 30 × 0.01 + 0.65 = 0.95		
Adjusted FP = Unadjusted FP × Influence Multiplier = 109 × 0.95 = 103.55		

5.1.3 Lines of Code

Language	QSM SLOC/FP Data
JavaScript *	47

According to the Quantitative Software Management resource (https://www.qsm.com/resources/function-point-languages-table), each Function Point requires an estimate of 47 lines of code if the application is implemented using JavaScript.

Therefore, we have: **Lines of Code** = 103.55 FP × 47 LOC/FP = 4866.85 = **4867 LOC**

5.2 Efforts, Duration and Team Size Estimation

To estimate the effort and duration required for the project, we use function points as the basis to calculate Effort, Duration, Team size and finally the schedule. The estimates are expanded to account for project management and extra contingency time to obtain the total average effort estimates. From these averages, the duration of each work package in working days is estimated based on the following calculations.

Working Days	5 days in a week
Effort = Size / Production Rate	4867 LOC / (62 LOC/PD)* = 78.5 Person Day (PD) *LOC/PD statistics based on Industrial Benchmarks, 1997: Canada
Duration = 3 × (Effort) $^{1/3}$	3 × (78.5) $^{1/3}$ = 12.85 Person Day (PD)
Initial schedule = Effort / Working Days	78.5 / 5 = 15.7 Weeks
Team size = Effort / Duration	78.5 / 12.85 = 6.11 Persons
Compression Rate = Calculated Team Size / Actual Team Size	7 / 8 = 0.88
Desired Schedule = Initial Schedule × Compression Rate	15.7 Weeks x 0.88 = 13.82 Weeks
Working Hours	8 hours in a working day
Total Person-Hours (PH) = Effort × Daily Working Hours	78.5 × 8 = 628 Person-Hours (PH)

5.2.1 Distribution of Effort

1990's Industry Data	Work Package	Distribution	Estimates
Preliminary Design (18 %)	Project Plan	9%	56.52
	Requirement Specification	9%	56.52
Detailed Design	User Interface	7%	43.96

(25 %)	Technical Architecture	11%	69.08
	Data Modeling	7%	43.96
Code & Unit Testing (26 %)	Code & Unit testing	21%	131.88
	Online Documentation	5%	31.4
Integration & Test (31 %)	Integration & Quality Assurance	31%	194.68
	Extrapolated total effort		628
	2% for project management		e
	3% for contingency		18.84
	Total effort		659.4

Duration estimates within the table are based on the assumption that each and every Team member works an equal amount on any given work package during the full duration of the project.

An additional 5% of buffer time on top of the calculated total person hours estimated will be allocated for project management and contingency planning. Though having the buffer increases the work hours required in the short run, it will benefit the development of the system in the long run as by taking the time to properly plan the project management and contingency plans, classic mistakes such as process-related mistakes, product-related mistakes and technology-related mistakes can be avoided. This increases the overall efficiency in the development of the system as less backtracking and reworking of the system will be needed.

5.3 Cost Estimates

The project's expenses can be broken down into the following table:

Category		**Project Expenditures**	**Rate ($ per Unit / Day)**	**Quantity (Units / No. of days)**	**Total (SGD)**
SELECTION PROCESS		Travel & Expenses	$40.00	35	$1,400.00
	Software Costs	Private Git Repository	$205 per month	12	$2,460.00
		Database	$350 per month	12	$4,200.00

IMPLEMENTATION PROCESS	**Hardware Costs**	Servers	$340 per month	12	$4,080.00
	Manpower Costs	Project Manager	$6000 per month	12	$72,000.00
		QA Manager	$5500 per month	12	$66,000.00
		QA Engineer x 2	$4600 per month	12	$110,400.00
		Lead Developer	$4800 per month	12	$57,600.00
		Release Engineer	$4250 per month	12	$51,000.00
		Front-End Developer	$4250 per month	12	$51,000.00
		Back-End Developer	$4250 per month	12	$51,000.00
CONTINGENCY		Contingency	10% of Implementation costs	-	$46,554.00
MAINTENANCE COST		Software (Private Git Repository)	$205 per month	12 months	$2,460.00
		Hardware (Server)	$340 per month	12 months	$4,080.00
		Office Rental	$450 per month	12 months	$5,400.00
TOTAL COSTS					**$529,634.00**

6 Product Checklist

The plan is that the items listed below will be delivered on the stated deadlines.

Product	Estimated deadline	Location of delivery
Project Proposal	26/02/2021	Wiki
System Requirement Specification (SRS)	12/03/2021	Wiki
Quality Plan	12/03/2021	Wiki
Project plan	12/03/2021	Wiki
Risk Management Plan	12/03/2021	Wiki
Use Case Model	9/04/2021	Wiki
Class Diagram	30/04/2021	Wiki
Sequence Diagram	30/04/2021	Wiki
Dialog Map Diagram	30/04/2021	Wiki
Database Design	28/5/2021	Wiki
User interface Design	28/5/2021	Wiki
Test cases	4/6/2021	Wiki
System Prototype	29/10/2021	SVN
Developer test cases	29/10/2021	Wiki
Release plan	15/11/2021	Wiki
Finalized System Application	03/12/2021	SVN
Requirement Test Coverage Report	31/12/2021	Wiki
Documentation of Code	21/01/2022	SVN

7 Best Practice Checklist

Practices
Documentation must be in a standardized format: - Consistent font, font size and font colour for headers and body - Consistent line spacing and paragraphing
Requirement specification must follow the following standards: - Accurate, consistent and achievable - Unambiguous, cannot be open to multiple interpretations - Functional and Non-Function requirements must be listed - Constraints must be listed and double checked with requirements if they can still be implemented within the constraints - Requirements must meet stakeholders specifications - Avoid complex functions - Requirements must be approved and accepted by stakeholders
Modelling must follow the following standards: - Model elements must be labelled and include descriptions of each element - Model must have high cohesion and loose coupling - No redundant definition of class attributes or methods - Clear and concise descriptions
Application design must follow the following practices: - Design must be maintainable, allow future maintenance and expansion - Design must be intuitive - Design must be robust and able to handle errors without crashing - Appropriate software design pattern must be considered - Sufficient details must be provided before coding can commence - Design implementation must be realistic
Implementation of application must follow the following practices: - Suitable and efficient programming language must be chosen to implement the design - Test cases / unit testing must be documented - Source codes written must have high cohesion and loose coupling - Source code written must be understood easily by other developers
Testing of application must follow the following practices: - Test cases must cover as many scenarios as possible - Test cases must be created to test every requirement - Test cases must exist for bocan u guys boundary conditions - Test cases must exist to check for incorrect inputs

- Stress test and load test cases must be included
Quality Management must follow the following practices: - Application must adhere all Functional Requirements - Application must adhere all Non-Functional Requirements - Training for personnel must be conducted if required - Weekly code review by QA manager and engineer - There must be a proper error and bug reporting system to facilitate formulation of solutions to the problem
Application user interface design must follow Shneiderman's Eight Golden Rules: - Strive for consistency (Same layout / terminology) - Cater to Universal Usability (Appealing to by people of all race and religion, etc.) - Offer Informative Feedback (Errors, Warnings when user does an inappropriate action) - Design Dialogs to yield closure (Start, Middle, End) - Permit Easy reversal of actions (Undo / Back button) - Support Internal Locus of Control (Let users feel in charge of the interface. The interface should respond quickly to user inputs) - Reduce Short term memory load - Prevent Errors

8 Risk Management

Risk Management is covered in further detail in a separate document called “Risk Assessment”. The document is available in the Team Wiki with the Project Plan. The “Risk Assessment” document includes plans and strategies to identify, control and resolve risks that may occur during the process of development. It also includes the documentation and monitoring of these potential risks.

Risk	Probability	Effects	Mitigation Strategy
Key developers are ill and unavailable at critical times	Moderate	Serious	Reorganise teams such that there is more overlap of work such that team members will understand each other’s work.
Changes to requirements that require major design rework are proposed	Moderate	Serious	Eg. Event where MongoDB becomes deprecated due to a more advanced and efficient database system being released, or requirement change in database system framework to use other forms other than MongoDB. Ensure proper documentations of database systems components to facilitate migration of database systems to another platform by enhancing traceability of the

			existing database system. Ensure additional time buffers for development during the project planning phase. If appropriate, prepare a briefing document for senior management and stakeholders addressing the issue and seek approval to change over to the new database system.. Show them better alternatives and advantages of switching the current database systems to more advanced / efficient alternatives that have been released to the market.
Software components that should be reused contain defects that limit functionality	Moderate	Serious	Eg. Using any free open source codes that have defects that were undetected Carry out software testing on the open source code before using open source codes to ensure proper

			functionalities. Replace potentially defective components with bought-in components of known reliability.
The database used in the system cannot process as many transaction per second as expected (Hardware Resources)	Moderate	Serious	Eg. Scaling the application to handle a larger user base than initially expected. Explore alternative high-performance database equipments and the possibility of procuring it
Organisational financial problems forces reduction in project budget	Low	Catastrophic	During the planning phase, additional budgeting buffers can be added for unforeseen budget cuts or increase in development costs (Resources such as hardware price increasing). Alternative is to prepare a briefing document for senior management showing how the project will make important contributions to the goals of the

			business.
Time required to develop software is underestimated	High	Serious	Using Scrum Software Development Method, progress of each team member can be discussed during daily Scrum meetings. Conduct weekly team meetings to discuss the progress of project development, consider adding additional capable developers to the team or outsource some parts of the project to reputable software contractors.

9 Quality Assurance

Quality Assurance is covered in further detail in a separate document called “Quality Plan”. The document is available in the Team Wiki. Detailed guidelines and ideas are listed in the “Quality Plan” document to ensure quality assurance.

Multiple points are discussed in the “Quality Plan” document such as software reviews, documentation, implementation, management, etc. Details such as standards and conventions are introduced which are to be maintained and fulfilled throughout the duration of the project. Topics covered in the “Quality Plan” document will ensure good practices in specifications and implementations.

9.1 Project Management

The Project Manager will be responsible for approving:

- The system requirement specification document
- The overall time scale for the project
- The choice of system development life cycle
- The choice of software development tools and techniques utilised
- The selection of project teams
- The training of project teams

9.2 Product Assessments

The following product assessments will be conducted by SQ personnel:

- Audit
- Formal Inspection
- Review
- Analysis

9.3 Process Assessments

The following process assessments will be conducted by SQ personnel:

- Audit
- Assessment
- Analysis

9.4 Roles in Quality Management

9.4.1 Quality Assurance Manager (QAM)

Responsibilities include, but are not limited to:

- Secure and manage SQ personnel resource levels
- Ensure that SQ personnel have office space and the appropriate tools to conduct SQ activities
- Provide general guidance and direction to the SQ personnel responsible for conducting software quality activities and assessments
- Assist SQ personnel in the resolution of any issues/concerns and/or risks identified as a result of software quality activities
- Escalate any issues/concerns/risks to project management

9.4.2 Software Quality (SQ) Personnel

Responsibilities include, but are not limited to:

- Develop and maintain the project software quality assurance plan
- Generate and maintain a schedule of software quality assurance activities
- Conduct process and product assessments, as described within this plan
- Identify/report findings, observations, and risks from all software assurance related activities to the QAM

9.5 Software Quality Programme

These practices and conventions are tools used to ensure a consistent approach to software quality for all programs/projects.

The most important software qualities for CashTrack are as follows:

- Functionality

  The sum or any aspect a software application can achieve for a user.

- Usability

  The sum or any aspect to achieve the quantified objectives used by users with effectiveness, efficiency, and satisfaction in a quantified context of usability of failure-free operation of a software application for a specified period

- Reliability

  The probability in a specified environment

- Maintainability

  The degree to which a software application can be understood, repaired, or enhanced.

9.6 Software Standard Metrics

A software metric provide a quantitative measure of software quality characteristics which must be achieved in the final product.The following standard metrics are the minimum planned metrics that will be collected, reported, and maintained in the area of software quality assurance:

- **Critical Defects Percentage** - Amount of defects that are being reported as critical defects [(Critical Defects / Total Defects Reported) X 100%]
- **Lines of code** - Used to measure the size of a computer program by counting the number of lines in the text of the program's source code. Lines of code is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate programming productivity or maintainability once the software is produced.
- **Cyclomatic complexity** - A matric used to indicate the complexity of a program, having multiple nested branching statements increases the complexity of the program.
- **Rework Effort Ratio** - Amount of work done to rework problems or errors in a section instead of completing the section [(Actual rework efforts spent in that phase/ total actual efforts spent in that phase) X 100%]
- **Schedule slippage** - Schedule slippage measures the amount of time a single test or a test suite has been delayed from its original baseline schedule. [(Actual Project Duration – Planned Project Duration / Planned Project Duration) X 100%]
- **Test review efficiency** - The amount of test cases that are reviewed within a certain time frame (Number of tests reviewed /Total time)

9.7 Testing

SQ manager and engineer will be in charge of test management processes. This includes testing of the software system components described in the test plans. They will also monitor and assure that testing schedules are adhered, approve testing procedures and ensure proper documentations of test anomalies and the closure of these test anomalies.

9.8 Problem Reporting and Corrective Action

SQ personnel generate, track, and trend assessment findings and observations in a centralized Reporting and Corrective Action System. The location of the system will be on the Team’s MediaWiki, with the use of an EXCEL spreadsheet.

Development team meetings will be carried out on a weekly basis. Meeting minutes and documentations will be reviewed during the meetings and all reported problems by team members will be issued a deadline for it to be resolved.

All reported problems will be documented in the team's meeting minutes and an external EXCEL spreadsheet that will be uploaded and updated on the team's MediaWiki “Problems Encountered” page. Each problem that is reordered will consist of various information such as team members assigned to resolve the problem, date line to resolve the problem, status of problem and the date and team member that the problem was discovered.

10 Monitoring & Control

Many procedures are required in order to be able to successfully monitor the progress of a software project. Some of the most important are:

10.1 Quantitative Measurement of Resource Consumption

Usage of resources is an important factor, improper resource management may lead to potential risks and problems. Quantitative measurements will be needed for the estimation of resources required throughout the lifespan of the project.

10.2 Identification of key Project Risks

It is a good practice to identify major risks that may arise throughout the project early. Early identification of such risks allows the team to prevent potential risks and handle the problem correctly by having ample time for corrective/mitigation actions to be carried out. Examples of major risks and preventive measures are discussed in the “Risk Assessment” document on Team Wiki.

10.3 Project Progress Reviews

Cashtrack Team has SQ personnel to assess the project’s risk management process and will be participating in bi-monthly risk management meetings which can be used to identify potential software risks which will be reported to QAM and project manager so that they can decide the next course of actions to be taken. The main communication medium that will be used is Whatsapp for text based group messages and Zoom for live online face-to-face meeting, these two modes of communication will be used to discuss problems and changes during the lifecycle of the project. Github will be used as the main repository to keep track of changes and development of the product and also act as a version control system. Jira will also be used to keep track of project development using Agile Software Development Methodology. Google Drive will be used as a cloud storage to store minor team documents. Lastly, Team WIki will be used to store all documentations and reports of the whole project. Final source codes must be uploaded to the SVN repository and Github.

10.4 Task Decomposition & Timeline Planning

The deadlines for each deliverable along with project milestones will be decided during the planning of the project timeline. Using Scrum Agile Software Development Method, each task of a deliverable will be broken down into smaller manageable sub tasks to facilitate distribution of work and responsibilities for each Scrum Sprint. The timeline and scheduled tasks are present in the Gantt Chart, where the different tasks and specific milestones are set and the task status is tracked and discussed during daily Scrum

meetings. In the event that the product development fails to meet expected deadlines and milestones, the project manager will have to reevaluate the project timeline and milestones and decide if the section that is behind schedule can be left out of the development if it is not critical or important (additional bonus features), while the development team will try their best to work harder (possible overtime if section is of critical importance) and make up for lost time. A more detailed version of task decomposition is present in the Work Breakdown Structure (WBS).