CIS 376 Term Project

By: Nidhi Patel, Elishbah, Pamela, Zan, Ahmed Al-faris

1.0 Introduction

This section provides an overview of the SplitSmart Plan.

1.1 Project Scope

A description of the scope of software testing is developed. Functionality /features/ behavior to be tested is noted. In addition any functionality/features/behavior that is not to be tested is also noted. We will be testing the overall look and functionality of the SplitSmart. The table should be presented neatly and the inputs should line up creating a clean and organized look. All entries will be checked for validation and ensure that they are the correct data type if necessary. Each valid entry should be entered and should go to the correct spot and align with the other inputs.

1.2 Major software functions

Any business, product line or technical constraints that will impact the manner in which the software is to be tested are noted here.

- Our team has limited time to complete this project as well as additional commitments besides it. As each team member is busy, setting a schedule to work together and give updates on a time constraint is difficult.
- The majority of the team's programmers lack the necessary web development knowledge. As a result, additional work is needed to learn and create a usable product.

1.3 Performance/Behavior issues

Any special requirements for performance or behavior are noted here.

1) Slow Servers and Loading Time

   A shared account, which means that the website is sharing the server with hundreds or even thousands of other websites, may be used to host the servers if they are running very slowly. To resolve this problem, we might ask the hosting firm whether the website is housed on a dedicated server. The

program will examine the website's content and pinpoint the factors slowing it down.

1.4 Management and technical constraints

Due to the inexperienced nature of the contributors to this project, as well as the mandate to not over-specialize any member and see them lose experience, leads to the responsibility of every team member to take some steps to avoid every risk. This is reflected in our staff organization chart, wherein nearly every contributor is considered some variant of “developer” and “quality assurance,” although different contributors have different magnitudes of responsibility associated with these roles.

- Developers - Utilize version control features in Git, communicate with other contributors how code functions, talk with cilent.
- Quality Assurance - Apply industry-developed QA tactics to avoid software bugs, routinely seek to clarify the development intentions of developers, scheduling the responsibilities of themselves and others
- Client - Help the contributors to build a complete, comprehensive model of how the user should interact with the final product

2.0 Project Estimates

This section provides cost, effort and time estimates for the projects

2.1 Estimation techniques applied and results

COCOMO (Constructive Cost Model):

LOC (lines of code) : -

Calculation: 55	Data: 30
I/O: 22	Logic: 16
Setup: 15	Text: 34

55+30+16+15+34 = 150 lines

2.1.1 Estimation technique *m*

COCOMO Basic Equation: Effort = a * (Size) ^ b Where:
Effort: Estimated effort in person-months
Size: Estimated size of the software in lines of code (LOC)
a, b: Parameters specific to the project type and development mode

2.1.2 Estimate for technique *m*

Using a Semi-detached model due to the need of a database, intricate user interface and complexity of the program.

By using a semi-detached model we know a = 3.0 and b = 1.12

Therefore Effort = a * (LOC) ^ b

3 * (150) ^ 1.12

Effort = 876.37

Duration = a * (Effort) ^ b

2.5 * (876.37) ^ 0.35

Duration = 26.78

2.2 Reconciled Estimate

The final cost, effort, time (duration) estimate for the project (at this point in time) is presented here.

Cost: none as we are students therefore this will not cost our professor anything.

Effort: 876.37

Time: 27 days

2.3 Project Resources

People, hardware, software, tools, and other resources required to build the software are noted here.

- People: Five CIS students with varying experience
- Hardware: Personal PCs and access to lab equipment
- Software: Open source software for database design, development, and testing (IDE and database management system)
- Tools: Any open source software that may be needed for the project

3.0 Risk Management

This section describes the overall testing strategy and the project management issues that are required to properly execute effective tests.

3.1 Project Risks

Each project risk is described. The CTC format may be used.

Technical risks:

- **Unfamiliarity with different programming languages**: with many of the team members having varying levels of experience in programming, there is a risk that some members will not have experience with the programming language chosen for the project.

- **Lack of experience with databases**: some team members are not familiar with DB design and SQL programming.

Project risks:

- **Scheduling conflicts**: some team members work full time along with school. Scheduling conflicts are a concern.

3.2 Risk Table

The complete risk table is presented. Name of risk, probability, impact and RM3 pointer are provided.

Risk Title	Category	Likelihood	Impact	Cost	Priority	Mitigation Plan	Target Date
Unfamiliarity with different programming languages	Technical risk	10	9	8	16	Decide on common programming language	5/31/2023
Lack of DB experience	Technical risk	9	8	6	36	Identify relevant experience	5/31/2023

						and distribute workload accordingly	
Scheduling conflicts	Project risk	10	6	5	25	Try to work around scheduling conflicts and have independent work	7/31/2023

3.3 Overview of Risk Mitigation, Monitoring, Management

An overview of RM3 is provided here. The Complete RM3 is provided as a separate document or as a set of Risk Information Sheets.

Risk type: technical risk

Priority: 16 (11-10) * (11-9) * Cost

Risk factor: amount of programming experience by group members in chosen language

Probability: 10. The team has discussed our experience and identified that this is a risk

Impact: 9. This is a high impact risk. It can be worked around by distributing work to effectively utilize different team members experience but will be a risk throughout the project.

Monitoring approach: distribute workload and check in with team members to make sure no one feels overwhelmed by the work.

Contingency plan: by distributing the workload we can minimize the impact of this risk. All team members are capable so we can fall back on each other if necessary.

Estimated resources: a few extra days to become familiar with the code and work off of other members' code.

Risk type: technical risk

Priority: 36 (11-9) * (11-6) * Cost

Risk factor: how much experience team members have with database design

Probability: 9

Impact: 8

Monitoring approach: continually check in to make sure everyone is somewhat familiar with the assigned tasks

Contingency plan: fall back on other team members in case someone is not familiar with database design/SQL

Estimated resources: a few extra days to learn about databases.

Risk type: project risk

Priority: 25. (11-10) * (11-6) * Cost

Risk factor: alternative commitments to jobs and other classes.

Probability: 10. Will definitely occur because we have discussed our schedules.

Impact: 6. This issue can be worked around for now. We will revisit this risk if it proves to be a problem

Monitoring approach: open line of communication to ensure team members are comfortable expressing issues in scheduling or meeting times.

Contingency plan: change around the workload if absolutely necessary. Try to have at least a few days of padding before a milestone of the project in case we run behind schedule.

Estimated resources: using the additional days of padding we reserve before a milestone and having some team members pick up work for others if necessary.

4.0 Project Schedule

4.1 Project task set

1.0 Conception:

1.1 Construct Software Requirements

1.1a Meet with team members

1.1b Determine requirements

2.0 Planning:

2.1 Estimates

2.1a Time

2.1b Risks

2.2 Scheduling

2.2a Functional Decomposition

2.2b Task Network

2.2.c Timeline

3.0 Design:

3.1 Analysis

3.1a Function Analysis

3.2 UML Diagramming

3.2a Class diagram

4.0 Implementation:

4.1 Code

4.1a Implement design

4.2 Test

4.2a Meet requirements

4.2b Stability testing

4.2c Usability testing

5.0 Deployment:

5.1 Deliver

5.1a Deliver a working application.

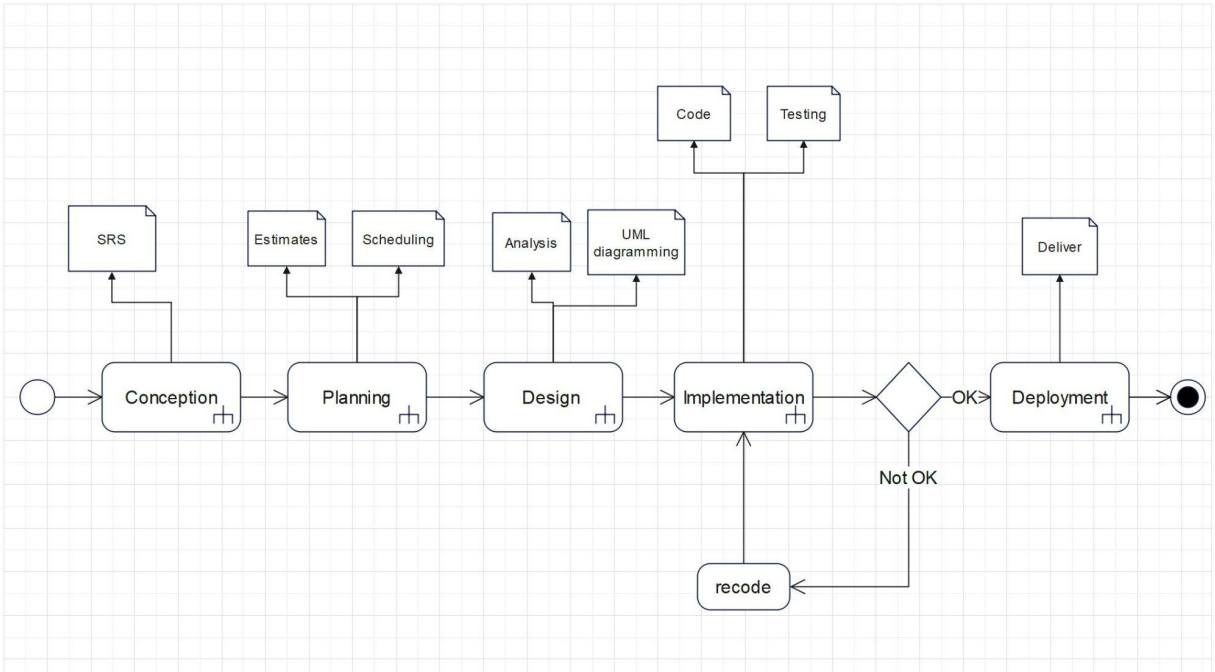
4.2 Functional decomposition

User Functions

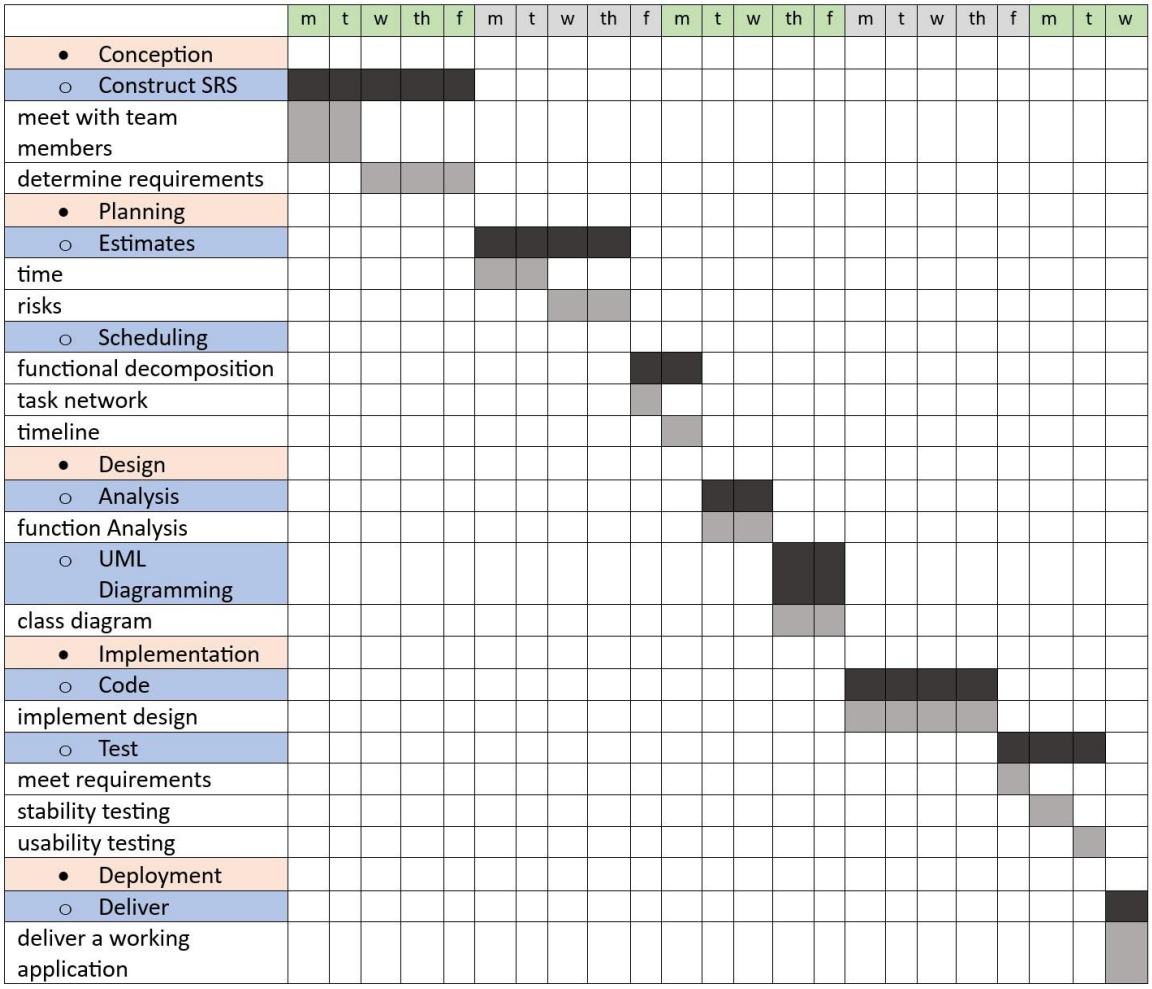
Function Name	**Description**
User Accounts	**This function will allow user to create personal account, which they can use to log in and access their own expense and balance information**
Group Management	**This function will allow users to create and manage groups of people to provide easy access for participants to track shared expenses and balances within the group.**
Expense Creation	**This function will allow users to create new expenses:** **· New expenses will contain information such as: amount, date, description, shared manner (split %) and an optional receipt image** **· Will allow users to specify which other users were involved in expense so appropriate balance adjustments can be made** **· Will have an expense approval function for review and approving expenses before they are added to the system and used to adjust balances.**

Notification System	**This function will notify users when other group member creates a new expense**
Balance Tracking	**This function will track the balance owed by each user to every other user, taking into account all expenses that have been entered**
Payment Tracking	**This function will allow users to record when payments have been made to settle balances owed, so that the balances are accurately reflected in the system**
Reporting	**This function will provide users with reports and summaries of their expenses, balances, and payments.**

4.3 Task network



4.4 Timeline chart



5.0 Staff Organization

The manner in which staff are organized and the mechanisms for reporting are noted.

5.1 Team structure

The team structure for the project is identified. Roles are defined.

- The project is very straightforward, neither large or small, because none of the team members are experts and it will probably take more time than it would for a group of experts. The tasks can be modularized in a logical manner, but communication among team members is a must to ensure the system functions properly as a whole. For these reasons, an agile team approach would be the best fit.

- The team is small, but highly motivated and self-organizing.
  We have decided that everyone will participate in each part of the plan. This way, we learn from each other, gain knowledge in every category, and benefit from new experiences. It is also a good way to organize ourselves and have experience in each respective role.
  Such as : Analysis, Design, Programming, Testing and Training

5.2 Management reporting and communication

Mechanisms for progress reporting and inter/intra team communication are identified.

1. **Regular Team Meetings**
   Regular team meetings, such as weekly status meetings, provide an opportunity for each of our team members to share progress updates, discuss challenges, and align on goals. These meetings foster communication and collaboration within the team.

2. **Progress Reports**
   Regular progress reports provide a snapshot of project status, accomplishments, and challenges. These reports can be prepared on a weekly basis and shared with everyone to keep them informed about the project's progress.

3. **Communication Tools**
   Instant messaging and communication tools, such as Microsoft Teams, E-mail, Zoom facilitate quick and efficient communication among team members. These tools help us in real-time messaging, file sharing, and the creation of dedicated channels for specific topics or project updates.

4. **Collaborative Documentation**
   Platforms such as Google Docs help our team to create and share project-related documents, specifications, meeting notes, and other relevant information. Collaborative documentation ensures that our team members have access to up-to-date information and fosters inter-team communication.

6.0 Tracking and Control Mechanisms

Techniques to be used for project tracking and control are identified.

6.1 Quality assurance and control

An overview of SQA activities is provided.

- **Setting Checkpoints:** Team sets the checkpoints at specified intervals for checking software's performance, quality, and scheduling.
- **Measure Change Impact:** For detected defects, teams should fix defects and verify whether the fix defect introduces another within the software. Teams should also check for new defects when introducing new functionalities.
- **Management Planning:** SQA strategies should be implemented for project requirements and individuals within the team. SQA should be utilized within the project in the most efficient way.
- **Reports and Records**: Teams should maintain all records and documents such as test cases, modifications, defect logs/fixes, requirements, and stakeholder meetings. All should be documented for future reference.

6.2 Change management and control

An overview of SCM activities is provided.

- Codes are kept in a common repository and changes are committed to the repository. To keep in sync with the repository, all developers need to update their working copy. Detailed reports of changes including what, why, when, and by whom changes to codes are made should be documented and accessible to all developers.
- Changes that affect user experience will be notified to client

7.0 Appendix

Requirement: Interface easy to use by new and advanced users.

Design: The interface has been designed for maximum functionality and user interface , convenience and comfortable navigation.