Project Title

by

Author(s) Name

Stevens.edu

September 11, 2024

© Author(s) Name
Stevens.edu
ALL RIGHTS RESERVED

Project Title

Author(s) Name
Stevens.edu

This document provides the requirements and design details of the PROJECT. The following table (Table 1) should be updated by authors whenever major changes are made to the architecture design or new components are added. Add updates to the top of the table. Most recent changes to the document should be seen first and the oldest last.

Table 1: Document Update History

Date	**Updates**
08/22/2023	DDM: • Updated dsnManual.tex with *newcommand(s)*{} for easier references of requirements, figures, and other labels.
10/25/2023	DDM: • Added chapters on use cases (Chapter 5) and user stories (Chapter 4).
10/11/2023	DDM: • Added chapters on requirements (Chapter 3) and glossary.
09/18/2023	DDM: • Added chapter on development plan (Chapter 2).

Table of Contents

1 Introduction
– Author Name **1**
1.1 Compiling EPS files with PSFrag on Overleaf 2

2 Development Plan
– Author Name **3**
2.1 Introduction (required) 3
2.2 Roles and Responsibilities (required) 3
2.3 Method (required) 4
2.3.1 Software 4
2.3.2 Hardware 4
2.3.3 Backup plan (individual and project) 4
2.3.4 Review Process 4
2.3.5 Build Plan 5
2.3.6 Modification Request Process 5
2.4 Virtual and Real Workspace 5
2.5 COMMUNICATION PLAN (required) 5
2.5.1 Heartbeat Meetings 5
2.5.2 Status Meetings 6
2.5.3 Issues Meetings 6
2.6 Timeline and Milestones (required) 6
2.7 Testing Policy/Plan (required) 6
2.8 Risks (required) 7
2.9 Assumptions (required) 7
2.10 DISTRIBUTION LIST 7
2.11 IRB Protocol (required) 7
2.12 Required Resource and Budget (required) 7
2.13 Worry Beads (optional) 7
2.14 Documentation Plan (optional) 7
2.15 Build Plan (optional) 8
2.16 Others (optional) 8

3 Requirements
– Author Name **9**
3.1 Stakeholders . . . 9
3.1.1 Customers . . . 9
3.1.2 Sponsors . . . 9
3.1.3 Engineering and Technical Persons . . . 9
3.1.4 Regulators . . . 9
3.1.5 Third Parties . . . 9
3.1.6 Competitors . . . 9
3.2 Key Concepts . . . 9
3.3 User Requirements . . . 10
3.4 System (Constraints) Requirements . . . 10
3.5 Non-functional (Quality) Requirements . . . 11
3.6 Domain (Business) Requirements . . . 11
3.7 Other Requirement Types . . . 11

4 User Stories
– Author Name **12**

5 Use Cases
– Author Name **13**
5.1 Table of Use Cases . . . 13
5.2 Use Case Diagrams . . . 13
5.3 Use Case . . . 13

6 User Interface Design
– Author Name **16**
6.1 User Persona . . . 16
6.2 User Interface Design . . . 16

7 Logical View
– Author Name **17**

8 Process View
– Author Name **18**

9 Development View
– Author Name **19**

10 Physical View
– Author Name **20**

11 Team
– Author Name **21**

12 Git Homework **22**
– Author Name

Bibliography **24**

List of Tables

1 Document Update History iii

3.1 User Requirements Table 10

5.1 Use Cases Table 13
5.2 . Use Case First 14

List of Figures

5.1 Sample use case diagram, with reference to use case UC_1. 15

Chapter 1

Introduction

– Author Name

All projects should have a small introduction. Here we provide some example LaTeX commands. The first example, which is commented out in LaTeX, is how to introduce an EPS file as an image into the document.

Here is a sample citation [1].

```
\begin{figure}
\psfrag{a }{\Large stvDataObject}
\psfrag{b }{\large Object 1}
\psfrag{c }{\large Object 2}
\psfrag{d }{\large Object N}
\psfrag{e }{\large $Count:N$}
\psfrag{f }
{\hspace{-0.2in}\large \begin{tabular}{c} Sample \\ Table \end{tabular}}
\centering
\scalebox{1}{\includegraphics{eps/dsnHelloWorld.eps}}
\caption{\label{Figure::dsnHelloWolrd} Sample EPS image.}
\end{figure}
```

If using Overleaf, then you include a PNG or a JPG file as shown next:

```
\begin{figure}
\centering
\scalebox{0.8}{\includegraphics{png/stvAgileProcess.png}}
\caption{\label{Figure::stvAgile} PNG Image included.}
\end{figure}
```

If compiling on Windows, as LaTeX-to-PDF (and not LaTeX-to-PS-to-PDF), you need to comment out the pdfmark package from dsnManual.tex:

```
%\usepackage[pdfmark,
%breaklinks=true,
```

```
%colorlinks=true,
%citecolor=blue,
%linkcolor=blue,
%menucolor=black,
%pagecolor=black,
%urlcolor=blue
%]{hyperref} % links in pdf
```

1.1 Compiling EPS files with PSFrag on Overleaf

In the header file un-comment the following lines. (Viewable only in the .tex code, not in PDF):

Note that with psfrag used this way, we cannot include references in the psfrag command. We can only replace text and use mathematical formulas.

Chapter 2

Development Plan

– Author Name

2.1 Introduction (required)

This is a brief description of the project, at most 2 paragraphs. You should point to key documents, e.g., requirements documents, architecture review documents, state the problem you are trying to solve and state success criteria. Briefly state what is the problem you are trying to solve and perhaps a user story of how it will be used.

2.2 Roles and Responsibilities (required)

These vary for each type of product. For small projects, folks may serve multiple roles. This is a list of common roles we have used for software development:

1. Development Lead (only 1 name)
2. Buildmeister (only 1 name)
3. Architect (only 1 name)
4. Developers (multiple)
5. Test Lead (only 1 name)
6. Testers (multiple)
7. Documentation (multiple, usually ALL)
8. Documentation Editor (only 1 name)
9. Designer (only 1 name)
10. User advocate (only 1 name)
11. Risk Management (only 1 name)

12. System Administrator (only 1 name)
13. Modification Request Board (1 leader, multiple representatives)
14. Requirements Resource (usually 1 name)
15. Customer Representative (multiple)
16. Customer responsible for acceptance testing

2.3 Method (required)

These are unique to software development, although there may be some overlap.

2.3.1 Software

1. Language(s) with version number including the compiler if appropriate
2. Operating System(s) with release number
3. Software packages/libraries used with release/version number
4. Code conventions – this should preferably be a pointer to a document agreed to and followed by everyone

2.3.2 Hardware

1. Development Hardware
2. Test Hardware
3. Target/Deployment Hardware

2.3.3 Backup plan (individual and project)

Risk management.

2.3.4 Review Process

1. Will you do architecture, usability, design, security, privacy or code reviews?
2. What approach will you use for the reviews (formal, informal, corporate standard)?
3. Who is responsible for the reviews and resolving any issues uncovered by the reviews?
4. Code readings?

2.3.5 Build Plan

1. Revision control system and repository used
2. Continuous integration
3. Regularity of the builds – daily
4. Deadlines for the builds – deadline for source updates
5. Multiplicity of builds
6. Regression test process – see test plan

2.3.6 Modification Request Process

1. MR tool
2. Decision process (board – if more than paragraph should point to alternate description)
3. State whether there will be two process streams one during development and one after development

2.4 Virtual and Real Workspace

It is great to have a project room, it is also great to use a wiki or some document repository system so long as it is private. Google docs is not private but may be sufficient for a class or university project.

2.5 COMMUNICATION PLAN (required)

2.5.1 Heartbeat Meetings

This section describes the operation of the "heartbeat" meetings, meetings that take the pulse of the project. Usually, these meetings are weekly, and I prefer to have them early in the day before folks get into their regular routine, but this is not necessary. The meeting should include only necessary individuals – no upper-level management or lurkers. It should have a set agenda, with the last part of the meeting reviewing open issues and risks. It should be SHORT, thirty minutes or less is ideal. Notes should be provided after the meeting and issues should be tracked and reviewed each meeting, usually at the end.

2.5.2 Status Meetings

Status meetings have management as their target and should be held less frequently than heartbeat meetings, preferably biweekly, monthly or quarterly depending on the duration of the project. It is solely to provide status for the project. If issues arise, they should be addressed at a separate meeting (see next item). These should be short. This section should describe the format and periodicity.

2.5.3 Issues Meetings

If a problem does arise, never surprise your manager. Schedule a meeting at his or her earliest convenience. This section describes how alerts will arise and the governance of when to trigger an alert – usually after a discussion at the heartbeat meeting.

2.6 Timeline and Milestones (required)

This section should be crisp containing 4-10 milestones for the duration of the project, each of which would trigger a re-issuing of this document to report on progress. Each milestone should define a 100list begin time and end time. Each time you re-issue this document you should highlight changes with italics or bold with the track change features – colors will not show up on a photocopy.

2.7 Testing Policy/Plan (required)

This section should describe your testing methodology, e.g., you may include your plans for:

1. Test driven development
2. Unit testing
3. Integration testing
4. System testing
5. Acceptance testing
6. Regression testing
7. Testing for critical quality attributes
8. Any frameworks being used for your testing

At least indicate if certain type of testing will be practiced, if so, when to start, and what is the stopping criteria. For example, unit testing is recognized as important, and recommended. But sometimes start-up companies intentionally skip unit testing for speed to market.

2.8 Risks (required)

Ideally it occurs because of an established risk management program. If that does not exist, do the best you can to enumerate the risks and explain how they will be track, monitored, and mitigated.

2.9 Assumptions (required)

It may be clear to the project insiders what assumptions are being made about staffing, hardware, vacations, rewards, ... but make it clear to everyone else and to the other half of the project that cannot read your thoughts.

2.10 DISTRIBUTION LIST

Who receives this document?

2.11 IRB Protocol (required)

Does your project need IRB application. Refer to
https://sit.instructure.com/courses/67496/pages/irb-information?module_item_id=1960974

2.12 Required Resource and Budget (required)

Describe what resources, hardware, software, and other resources, you need for your project. Include a tentative budget for your resource. The more accurate the better.

2.13 Worry Beads (optional)

This section describes the things as manager I am most worried about at the time of latest document issue. This section is useful because it helps you to focus on the parts most likely to fail. Sometimes, I segment the worries by time scale: day, week, month, quarter ... lifetime.

2.14 Documentation Plan (optional)

Many years ago we had much too much documentation, now we have precious little – this must change. Write documentation as if you'll need to personally support the project forever – you just might need to and you'll be glad you took the time to document the obvious, the not so obvious and the obscure. As an example, it's useful to document alternate architectures and designs you did not pursue along with the rationale. What were the "gotchas" you were trying to avoid?

2.15 Build Plan (optional)

When builds and testing become complex, this might be a separate section or point to a separate document.

2.16 Others (optional)

Other aspects that are relevant to your project Any other aspects that are important to your project but not listed above, you can add them to new sections.

Chapter 3

Requirements

– Author Name

3.1 Stakeholders

People or roles who are affected, in some way, by a system and so who can contribute requirements or knowledge to help you understand the requirements:

3.1.1 Customers

Clients and users. Who are they, why do they use your system

3.1.2 Sponsors

3.1.3 Engineering and Technical Persons

3.1.4 Regulators

3.1.5 Third Parties

3.1.6 Competitors

Systems which provide similar functions.

3.2 Key Concepts

List the key concepts and their definitions that relate to your project. These concepts and terms should be used consistently throughout this document and in your project. This can point out to the Glossary chapter.

3.3 User Requirements

Abstract statements written in natural language with accompanying informal diagrams. You may use user stories or simple use case diagram(s). The user requirements should be numbered and linked to use-cases and user-stories. Here is an example.

Table 3.1: User Requirements Table

Requirement	Priority	Use Case(s)
Quality Requirement 1 (reqqFirstQualityRequirement) *The system shall perform the task in less than one second.*	MustHave	UC_1
Functional Requirement 1 (reqfFirstFunctionalRequirement) *The system shall perform the task of adding 1+1.*	Should-Have	UC_1
Constraint Requirement 1 (reqcFirstConstraintRequirement) *The system shall not interfere with local Wi-Fi communication channels.*	Could-Have	UC_1
Interface Requirement 1 (reqiFirstInterfaceRequirement) *The system shall have a C++ API.*	Would-Have	UC_1
Business Requirement 1 (reqbFirstBusinessRequirement) *The system shall be FDA approved.*	Would-Have	UC_1

At some point in your document, you will need to define your use case. For this to all work, you also need to define `newtheorem` for the requirements and use cases, in your root manual.tex.

3.4 System (Constraints) Requirements

More detailed descriptions of the services and constraints from the perspective of the system to meet the user requirements. Should be structured and precise. More detailed that describe the user requirements and focus on the basic flow, alternative flow, exceptions, pre-conditions, and post-conditions, and special requirements for each abstract user requirements. May use use- case template for this. System requirements should be numbered and traced back to the user requirements.

3.5 Non-functional (Quality) Requirements

Any important non-functional requirements for your project. These would include any performance and quality requirements (i.e. numbers, such as bandwidth, time, speed, rates, etc.). Can add here any type of a multitude of quality requirements that define, in large part, the system architecture.

3.6 Domain (Business) Requirements

Business rules and regulations that impact what the system does.

3.7 Other Requirement Types

If there is a reason to define a new, non-overlapping requirement class, you can create more requirement classes as needed, but be careful to not create too many.

Chapter 4

User Stories

– Author Name

Chapter 5

Use Cases

– Author Name

This chapter presents the use case diagrams that satisfy the requirements. Detailed description of each use case should be given in separate chapters for each use case. In this sample script, they are all left in this chapter. Move them as needed.

5.1 Table of Use Cases

To prevent gold-plating all use cases must be mapped to at least one requirement. If there are any use cases that don't have a requirement, then the use case should be removed. A table to keep track of this mapping is shown next.

Table 5.1: Use Cases Table

Use Case	Requirements	Name and Description
UC_1	$reqkQuality_1$, $reqkFunctional_1$, ...	ucFirstUseCase is a use case describing one of the usages of the system. One use case may map to multiple requirements. It must map to at least one.

You should distill the use cases/user stories that defined in your project specification. Here you should focus on the "must-have" use cases/stories, and elaborate each scenario to consider the pre-conditions, post-conditions, normal flow, and exceptional flows of each use case, if this is not considered in your project specifications.

5.2 Use Case Diagrams

5.3 Use Case

Table 5.2: . Use Case First

Use Case 1 (ucFirstUseCase) *Write Hello World! First use case.*
Requirements: $reqkFunctional_1$
Diagrams: Figure dsnHelloWorld (*Figure* 5.1)
Brief description: The system writes Hello World.
Primary actors: User
Secondary actors: None.
Preconditions: 1. Item one. 2. Item two.
Main flow: 1. Step one. 2. Step two. 3. If this then 3.1. Step one. 3.2. Step two.
Postconditions: None.
Alternative flows: None.

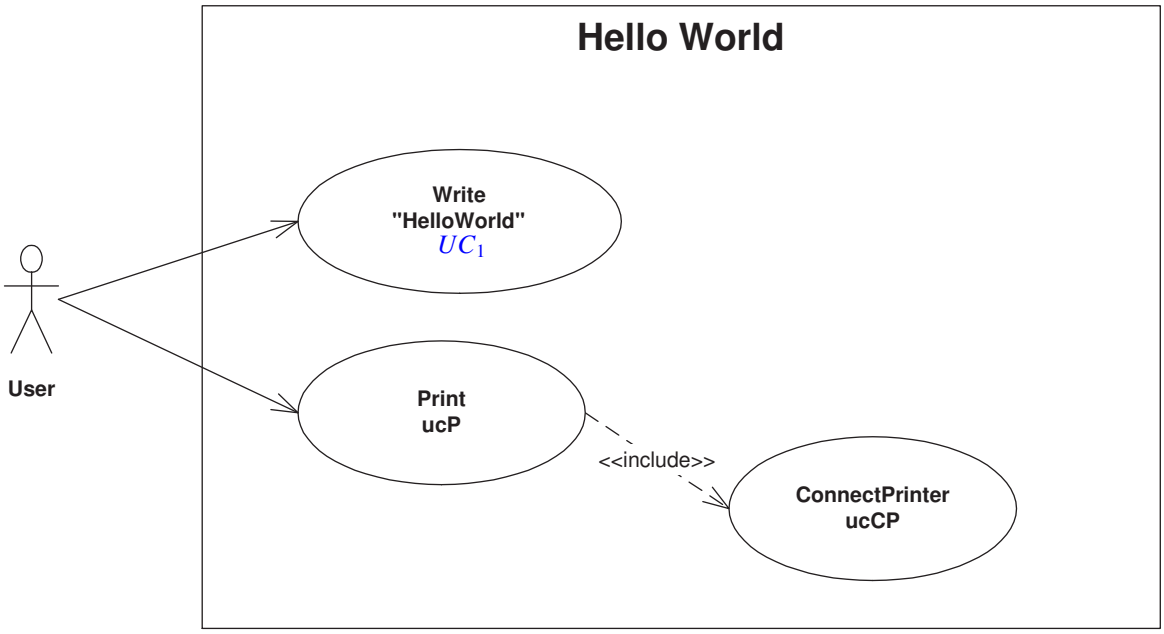


Figure 5.1: Sample use case diagram, with reference to use case UC_1.

Chapter 6

User Interface Design

– Author Name

6.1 User Persona

Define the persona of your users. You may have more than one user person for capturing different types of user roles for your system.

6.2 User Interface Design

Please provide preliminary paper prototype of your user interface. This should capture the key use cases/ user stories you defined in your project specification. Note that you do not need to cover every corner case. Make sure that your UI design covers the major use cases/stories, and the goals, tasks, and scenarios.

Chapter 7

Logical View

– Author Name

Choose a proper UML diagram, such as state diagram or class diagram to show the structure of the system for the functionality for users.

Chapter 8

Process View

– Author Name

Choose a proper UML diagram, such as activity diagram, sequence diagram, etc. to show the dynamic aspects of your system.

Chapter 9

Development View

– Author Name

Show the package diagram or component diagram of your system. Describe how the team members are going to work on this view to collaborate with each other.

Chapter 10

Physical View

– Author Name

Show the package diagram or component diagram of your system. Describe how the team members are going to work on this view to collaborate with each other.

Chapter 11

Team

– Author Name

My name is Joseph Stefanoni. I am a 4/4 Computer Engineering major with a minor in Software Engineering. I go to school at Stevens Institute of Technology. I am currently the Treasurer for the SAVE club at Stevens. I like to play video games, hike, and travel. I have interests in robotics, AI, and software development. I have coding knowledge with C++, Python, Java, and CUDA. C++ is my favorite to use.

Image1:

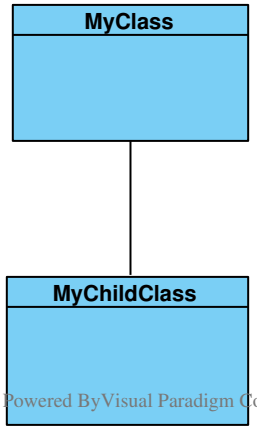
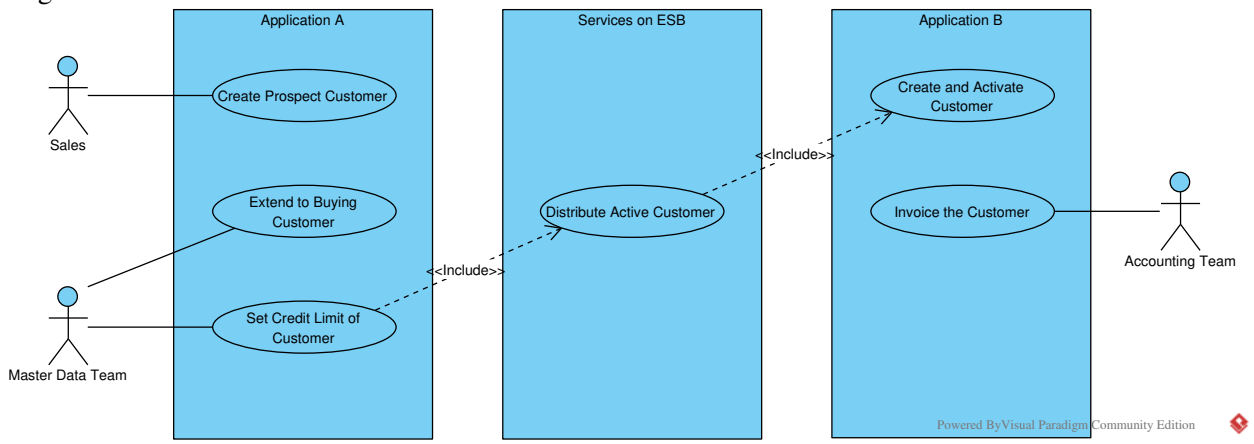


Image2:

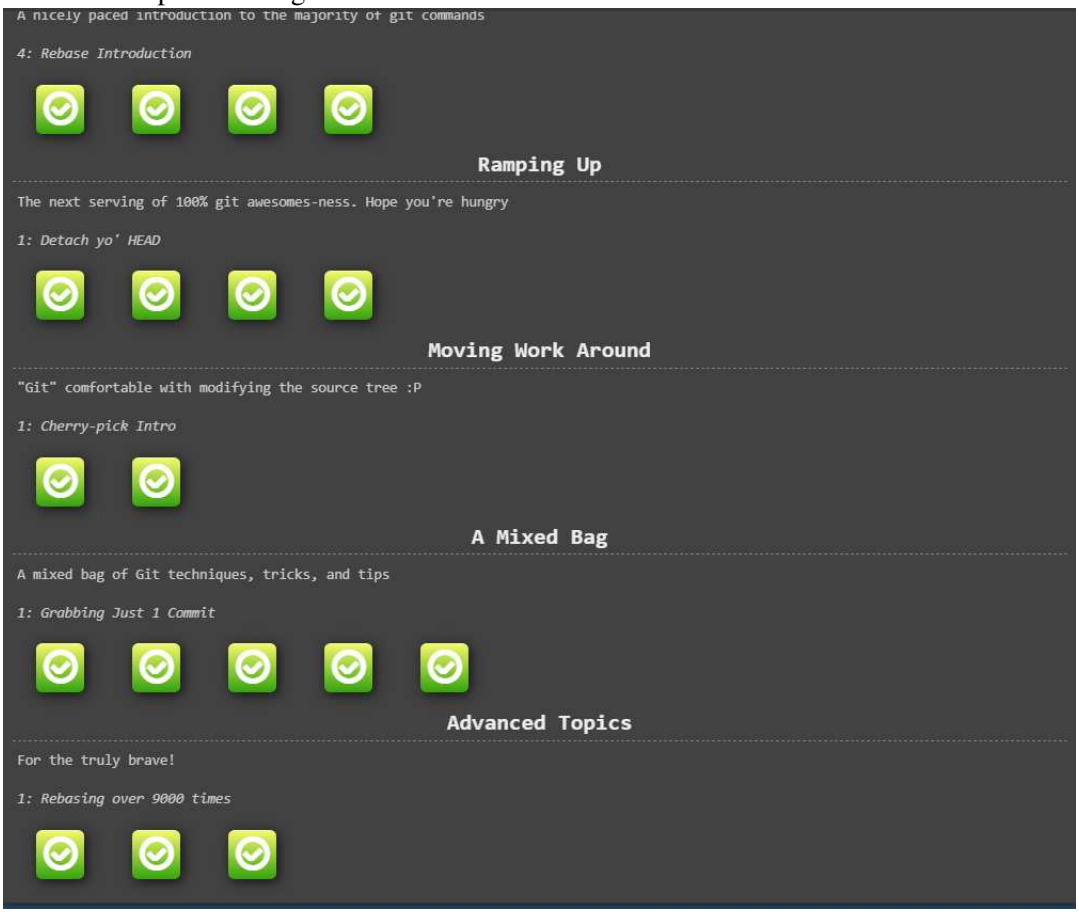


Chapter 12

Git Homework

– Author Name

Git Tutorial Completion Image:



GitHub link:

Glossary

CouldHave This defines the third highest priority requirement.The system could implement all of the tasks, requirements, or anything that is marked this way, but if resources are limited, it can be left out of the current and next version. Build in two versions from now. 10

MustHave This defines the first highest priority requirement. All of the tasks, requirements, or anything that is marked this way are build in the current version. 10

ShouldHave This defines the second highest priority requirement. The system should implement all of the tasks, requirements, or anything that is marked this way, but if resources are limited, it can be left out of the current version. Build in next version. 10

WouldHave This defines the lowest priority requirement. The system would like to implement all of the tasks, requirements, or anything that is marked this way, but only if resources are available. It can be left out of all future versions. 10

Bibliography

[1] E. J. Giorgianni and T. E. Madden, *Digital Color Management*. Addison Wesley, 1998.

Index

Chapter
    Development Plan, 3
    DevelopmentView, 19
    Git Homework, 22
    Introduction, 1
    Logical View, 17
    PhysicalView, 20
    ProcessView, 18
    Requirements, 9
    Team, 21
    Use Cases, 13
    User Stories, 12
    UserInterfaceDesign, 16

development plan, 3
Development View, 19
dsnHelloWorld, 14

git homework, 22
glossary, 12

introduction, 1, 9

Logical View, 17

physical view, 20
Process View, 18

reqbFirstBusinessRequirement, 10
reqcFirstConstraintRequirement, 10
reqfFirstFunctionalRequirement, 10
reqiFirstInterfaceRequirement, 10
reqqFirstQualityRequirement, 10

team, 21

ucFirstUseCase, 13, 14
use cases, 13
user interface design, 16